# Introduction to Inferring Evolutionary Relationships



Much of bioinformatics is essentially comparative biology. Inferences based on comparisons between entities such as motifs, sequences, genomes, and molecular structures are made. Given that living organisms and their constituent components have an evolutionary history, phylogeny properly lies at the heart of comparative biology (Harvey and Pagel, 1991). Increasingly, researchers in bioinformatics are realizing that phylogeny-based comparisons can yield important insights that can be missed using other techniques (Eisen, 1998; Rehmsmeier and Vingron, 2001). For example, a knowledge of phylogeny is vital in determining whether a set of sequences are orthologous or paralogous (Page and Charleston, 1997; Yuan et al., 1998; Storm and Sonnhammer, 2001; Zmasek and Eddy, 2001), which in turn has implications for predicting the function of a novel sequence (Eisen, 1998). Indeed, it is increasingly common for protein family databases to include gene phylogenies in addition to alignments. Examples include HOVERGEN (*http://pbil.univ-lyon1.fr/databases/hovergen.html*; Duret et al., 1994), SYSTERS (*http://systers.molgen.mpg.de*; Krause et al., 2000), and COPSE (*http://copse.molgen.mpg.de*). Phylogenetic analysis at the level of whole genomes poses new analytical challenges (Kim and Salisbury, 2001; Korbel et al., 2002; Wang et al., 2002), but promises new insights into genomic evolution.

In some cases, the role of phylogeny might not be either obvious or explicit, so that many people may well have built phylogenetic trees without necessarily realizing it. The popular multiple sequence alignment program Clustal (*UNIT 2.3*) builds a phylogeny (the "guide tree") every time it aligns sequences. Whereas Clustal constructs an initial tree to prioritize the order in which sequences are aligned, other methods infer the alignment and phylogeny simultaneously (Hein, 1990; Phillips et al., 2000).

Phylogenies also have a role to play in structural biology, especially in methods for predicting the secondary structure of RNA (Gulko and Haussler, 1996; Knudsen and Hein, 1999; Akmaev et al., 2000) and protein sequences (Li et al., 1998). Tree comparison techniques similar to those used to study host-parasite associations (Page and Charleston, 1998) have been used to tackle the problem of mapping cell-bound receptors onto the ligands to which they bind (Bafna et al., 2000).

## INTRODUCTION TO TREES

This section introduces some basic phylogenetic terms and concepts. For a fuller discussion see recent textbooks such as Page and Holmes (1998) and Hall (2001).

### Tree Terminology

A tree is a mathematical structure which is used to represent the evolutionary history of a group of sequences or organisms. This actual pattern of historical relationships is the phylogeny, or evolutionary tree, which we try to estimate. A tree consists of nodes connected by branches (also called edges). Terminal nodes (also called leaves, or terminal taxa) represent sequences or organisms for which we have data. Internal nodes represent hypothetical ancestors; the ancestor of all the sequences that comprise the tree is the root of the tree (Fig. 6.1.1). The nodes and branches of a tree may have various kinds of information associated with them. Some methods of phylogeny reconstruction (e.g., parsimony) endeavor to reconstruct the characters of each hypothetical ancestor; most methods also estimate the amount of evolution that takes place between each node on the

**Contributed by Roderic D.M. Page**
*Current Protocols in Bioinformatics* (2003) 6.1.1-6.1.13

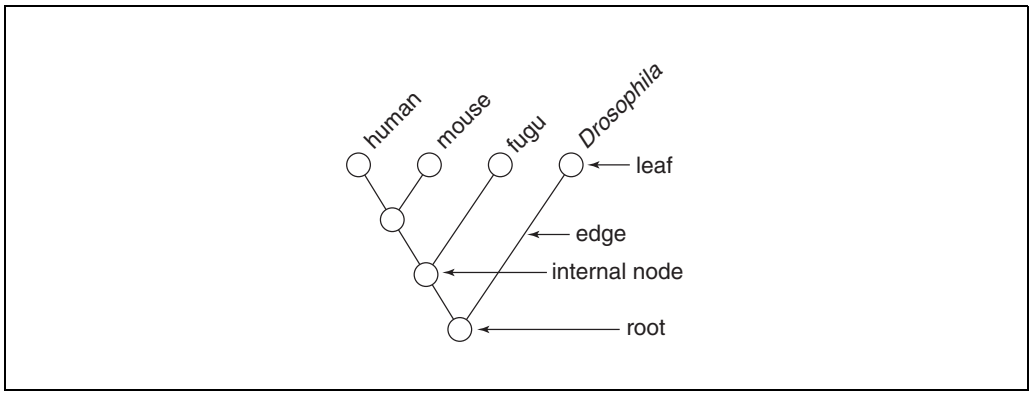


**Figure 6.1.1** A phylogeny showing some basic tree terms.

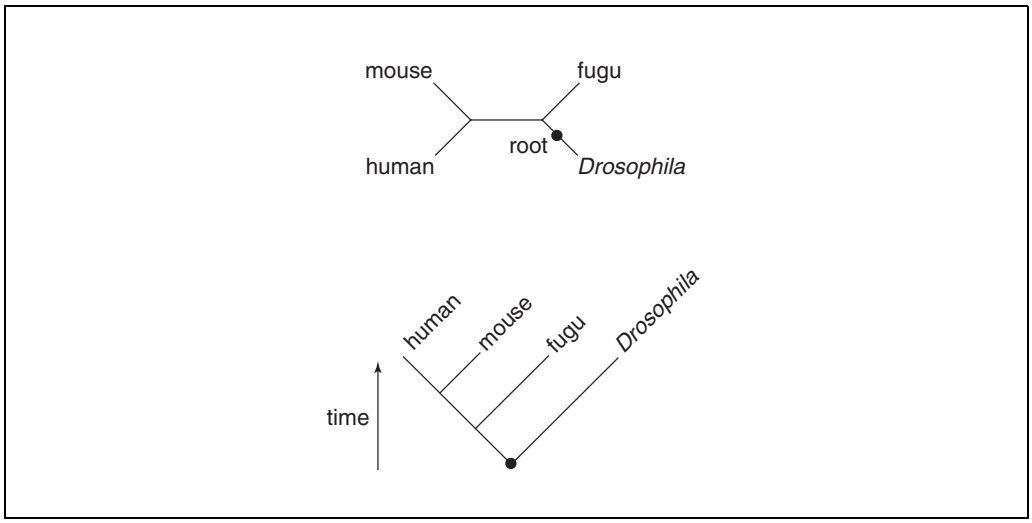


**Figure 6.1.2** Unrooted and rooted trees for human, mouse, fugu and *Drosophila*. The rooted tree (bottom) is obtained by rooting the unrooted tree along the edge leading to *Drosophila*.

tree, which can be represented as edge lengths (also called branch lengths). Nodes may also be labeled with various measures of support or confidence, such as bootstrap values or posterior probabilities (see below).

Trees can be depicted as a diagram such as Figure 6.1.1 or in a shorthand form consisting of nested parentheses, such as (((human, mouse), fugu), *Drosophila*). *UNIT 6.2* provides examples of different styles of tree drawing, and a detailed description of the nested parentheses notation.

### Rooted and Unrooted Trees

Trees can either be rooted or unrooted. A rooted tree has a node identified as the root from which all other nodes ultimately descend; hence a rooted tree has direction. This direction corresponds to evolutionary time; the closer a node is to the root of the tree the older it is in time. Rooted trees allow us to define ancestor-descendant relationships between nodes—given a pair of nodes connected by a branch, the node closest to the root is the ancestor of the node further away from the root (the descendant). Unrooted trees lack a root, and hence do not specify evolutionary relationships in quite the same way, and they do not allow us to talk of ancestors and descendants. Furthermore, sequences that may be adjacent on an unrooted tree need not be evolutionarily closely related. For example, given the unrooted tree in Figure 6.1.2, the fugu and *Drosophila* sequences are neighbors on the tree, yet the fugu is a fish and is is more closely related to humans and mice than to the

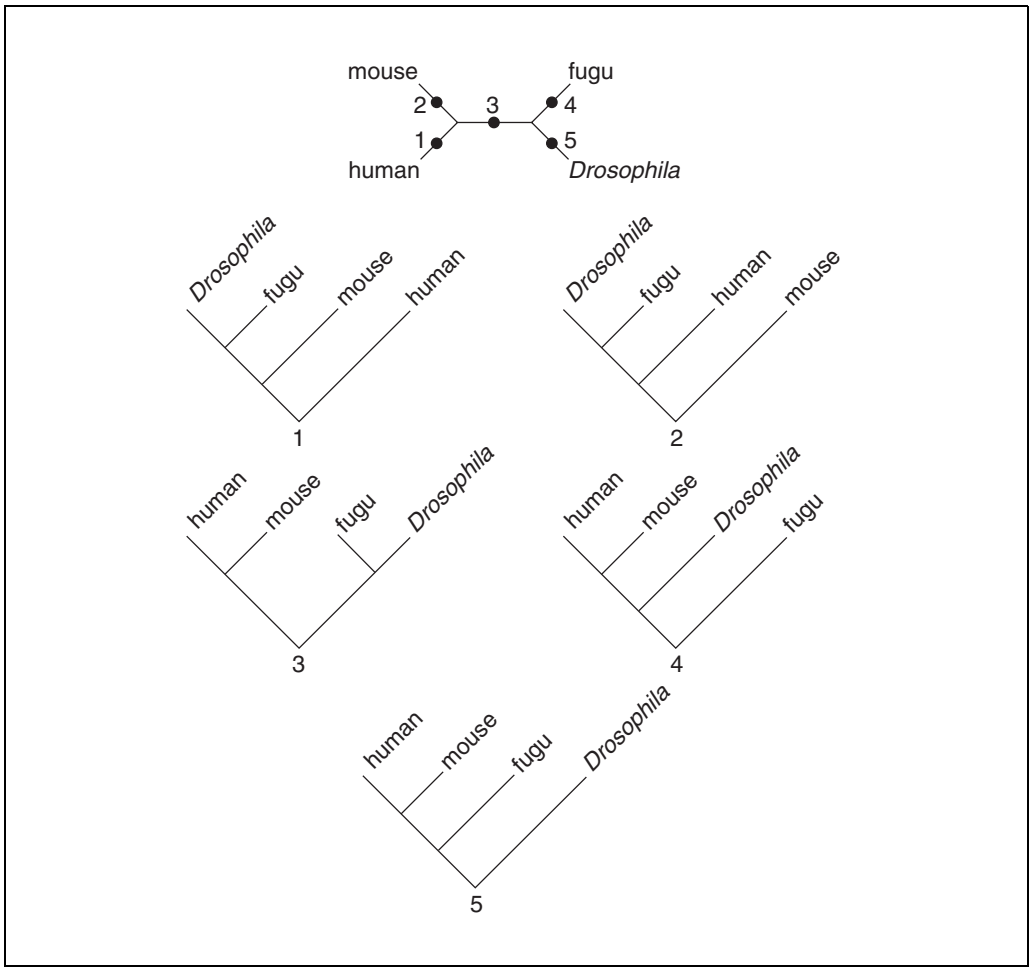


**Figure 6.1.3** The five rooted trees that can be derived from an unrooted tree for four sequences. Each of the rooted trees 1 to 5 corresponds to placing the root on the corresponding numbered edge of the unrooted tree.

fruit fly. This is because the root of the tree lies on the branch leading to *Drosophila*. Had we placed the root elsewhere, say on the branch leading to the mouse, then the fugu and *Drosophila* sequences would indeed be closely related. In fact, we could place the root on any one of the five edges in the unrooted tree; hence, this unrooted tree corresponds to a set of five rooted trees (Fig. 6.1.3).

The distinction between rooted and unrooted trees is important because most methods for reconstructing phylogenies reconstruct unrooted trees, and hence cannot distinguish among the five trees shown in Figure 6.1.3 on the basis of the data alone. In order to root an unrooted tree (i.e., decide which of the five trees is the actual evolutionary tree), we need some other source of information.

### Rooting Trees

Unrooted trees can be converted into rooted trees using a number of methods (Fig. 6.1.4). The most common is the use of an "outgroup." An outgroup is one or more sequences thought to lie outside the sequences of interest (the "ingroup"). For example, given the three vertebrate sequences from human, mouse, and fugu, we could use the invertebrate *Drosophila* to root the tree (Fig. 6.1.4A). If we are unsure of the identity of the outgroup, then other methods must be used. Midpoint rooting (Fig. 6.1.4B) places the root halfway along the longest path in the tree. This makes an implicit assumption that the rate of

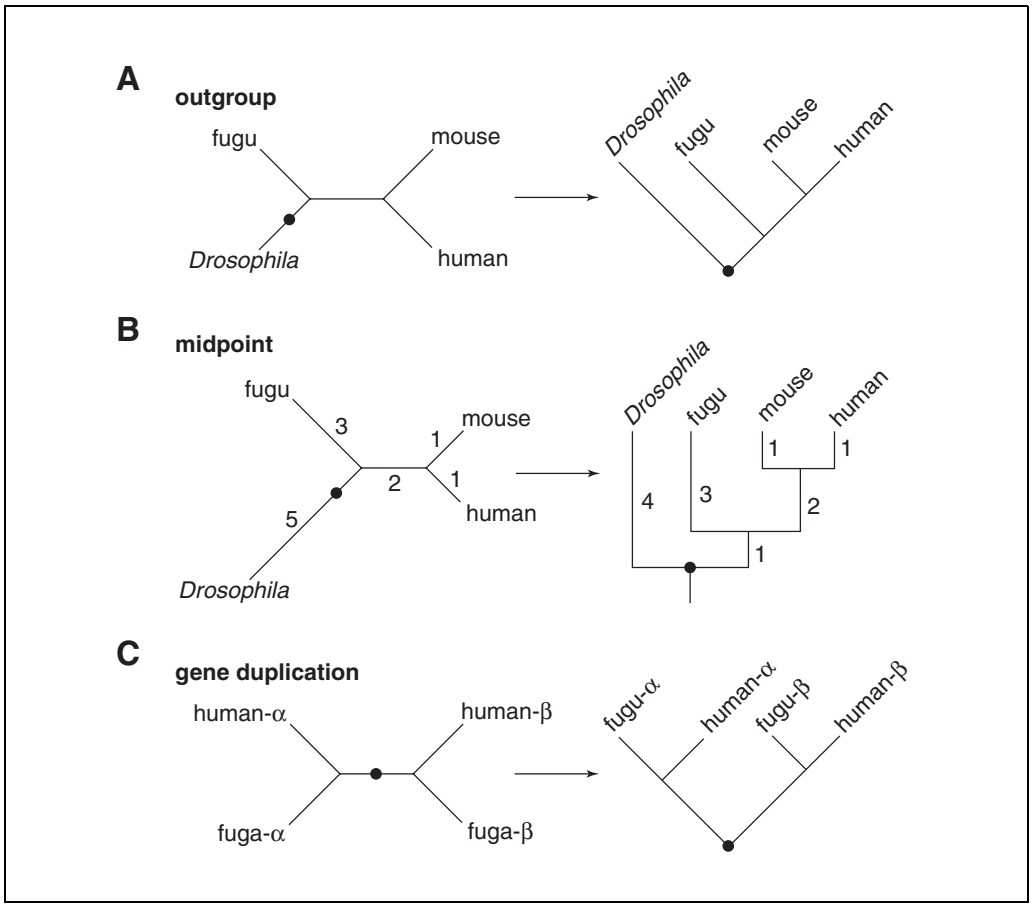


**Figure 6.1.4** Three different methods of rooting an unrooted tree. Outgroup rooting places the root between the outgroup (in this example *Drosophila*) and the ingroup (fugu, mouse, human). Midpoint rooting places the root at the midpoint of the longest path in the tree. Gene duplication rooting places the root between paralogous gene copies.

evolution among the sequences has been broadly constant over time (a "molecular clock"). If the sequences come from a gene family that has undergone a gene duplication resulting in paralogous copies (see below), then the root can be placed between those copies. In Figure 6.1.4C, the root is placed between the α and β copies in humans and fugu. In large gene families where multiple gene duplications have taken place, the gene tree can be rooted at the position requiring the least number of gene duplications (Iwabe et al., 1989).

### Gene Trees and Species Trees

There are a number of processes by which gene phylogeny can depart from species phylogeny. One of the most important is gene duplication, which can result in a species containing a number of distinct but related sequences. In the example shown in Figure 6.1.5, fugu, mice, and humans each have two copies of the same gene, α and β. A phylogeny for all six genes allows us to correctly recover the organismal phylogeny ((human, mouse), fugu) from either the α or β genes. However, if we are unfortunate enough to sequence only genes 1, 3, and 5, and are unaware that they are part of a larger gene tree, we would infer that the organismal tree was ((human, fugu), mouse). This is because gene 3 from humans is more closely related to gene 1 from fugu than is gene 5 from mouse, even though humans and mice are more closely related to each other than either is to the fugu.

Two homologous genes are orthologous if their most recent common ancestor did not undergo a gene duplication; otherwise they are termed paralogous (Fitch, 1970). In Figure

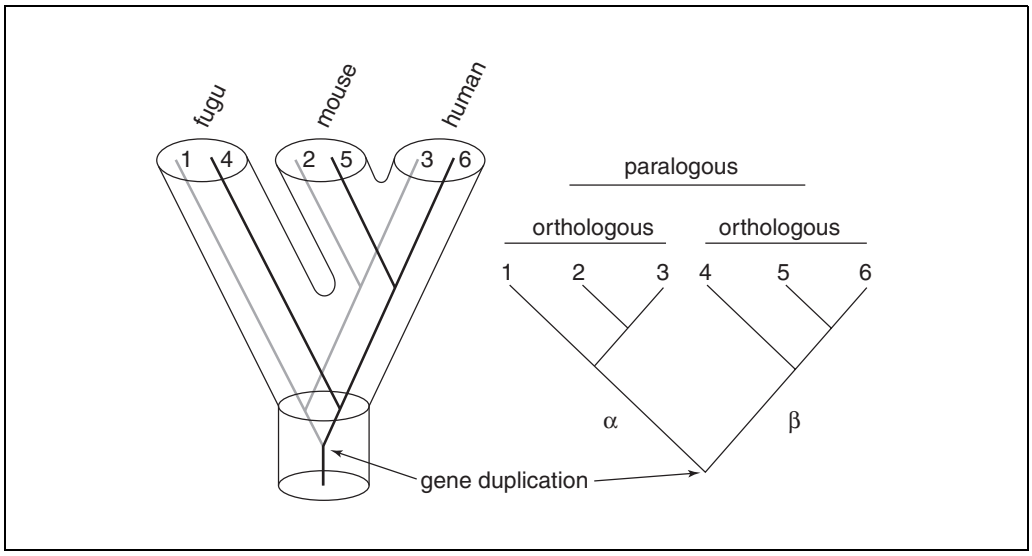


**Figure 6.1.5** Phylogeny for the fugu, mouse, and humans, and six genes (1 to 6) that stem from a gene duplication resulting in two paralogous sets of genes, α and β. The α genes 1 to 3 are orthologous with each other, as are the β genes 4 to 6. However, each α gene is paralogous with each β gene, as they are separated by a gene duplication event, not a speciation event.

**Table 6.1.1** Number of Unrooted and Rooted Trees for 2 to 10 Sequences

| No. of sequences | No. of unrooted trees | No. of rooted trees |
|---|---|---|
| 2 | 1 | 1 |
| 3 | 1 | 3 |
| 4 | 3 | 15 |
| 5 | 15 | 105 |
| 6 | 105 | 945 |
| 7 | 945 | 10,395 |
| 8 | 10,395 | 135,135 |
| 9 | 135,135 | 2,027,025 |
| 10 | 2,027,025 | 34,459,425 |

6.1.5, genes 1 to 3 are orthologous, as are genes 4 to 6, but any pair of α and β genes are paralogous. For a discussion of the implications of gene duplication for phylogenetic inference, see Page and Charleston (1997). A number of research groups have developed automated methods to distinguish between orthologous and paralogous sequences, and hence improve predictions of function (Yuan et al., 1998; Storm and Sonnhammer, 2001; Zmasek and Eddy, 2002).

## Numbers of Trees

Table 6.1.1 lists the numbers of rooted and unrooted fully resolved trees for 2 to 10 sequences. Note that the number of unrooted trees for $n$ sequences is equal to the number of rooted trees for $(n - 1)$ sequences. Note also that the number of trees rapidly reaches very large numbers: for 10 sequences there are over 34 million possible rooted trees. For a relatively modest 20 sequences, there are 8,200,794,532,637,891,559,000 possible trees! This explosion in number of trees is a fundamental problem for phylogeny

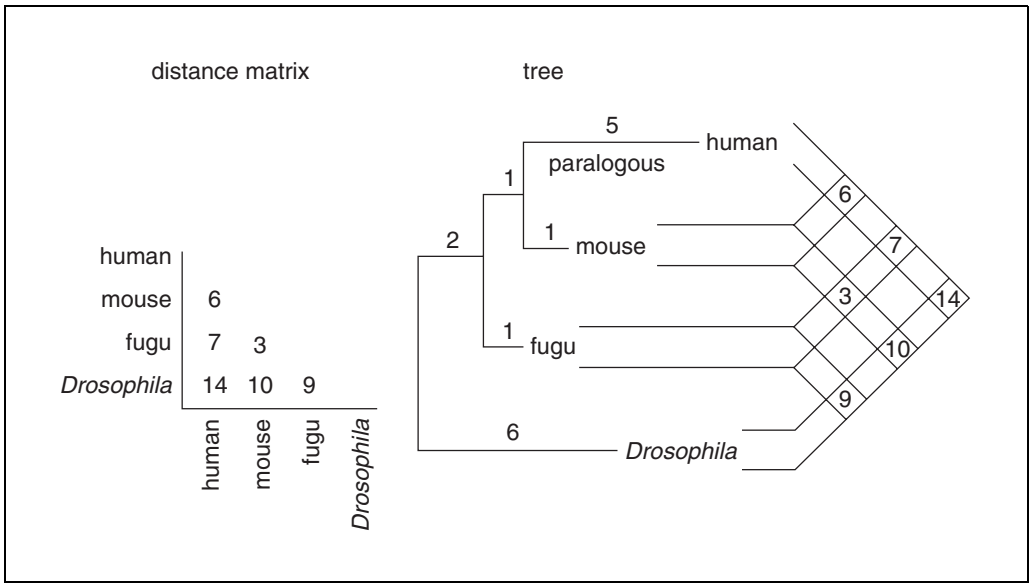


**Figure 6.1.6** An additive distance matrix between four sequences and the corresponding additive tree. For any two sequences, the value in the distance matrix corresponds to the sum of the edge lengths along the path between the two sequences on the tree (after figure 2.18 in Page and Holmes, 1998).

reconstruction methods that seek to find the optimal tree. The sheer number of trees means that for all but relatively small numbers of sequences, we cannot guarantee that the best tree will be found.

### Trees and Distances

Measures of sequence dissimilarity may be used to estimate the number of evolutionary changes that occurred in two sequences since they last shared a common ancestor. These measures quantify the evolutionary distance between the two sequences. Trees themselves can also be represented by distances, and this link has motivated a range of tree-building methods that seek to convert pairwise distances between sequences into evolutionary trees, the most popular being neighbor joining (*UNIT 6.3*).

Figure 6.1.6 shows a distance matrix and the corresponding tree. This tree represents the distance matrix exactly, in that the path between any two sequences on the tree is exactly the corresponding value in the distance matrix. The distances obtained from the tree are called tree distances (also called patristic distances), to distinguish them from observed distances, which are obtained directly from the sequences themselves. In the example shown in Figure 6.1.6, the observed and tree distances match exactly. For real data, this is rarely the case, indicating that the observed distances cannot be completely accurately represented by a tree. The discrepancy between observed and tree distances can be used to measure how good the fit is between the observed distances and the best tree representation of those distances (see *UNIT 6.3*).

## METHODS FOR INFERRING PHYLOGENIES

There are many ways to build trees, and each method has supporters and detractors (Swofford et al., 1996). For people new to the field, this plethora of methods can be confusing, even intimidating. There are various ways we can try and bring order to this situation. We can, for example, divide methods into clustering methods and optimality methods. Clustering methods follow a set of steps (an algorithm) and arrive at a tree. They

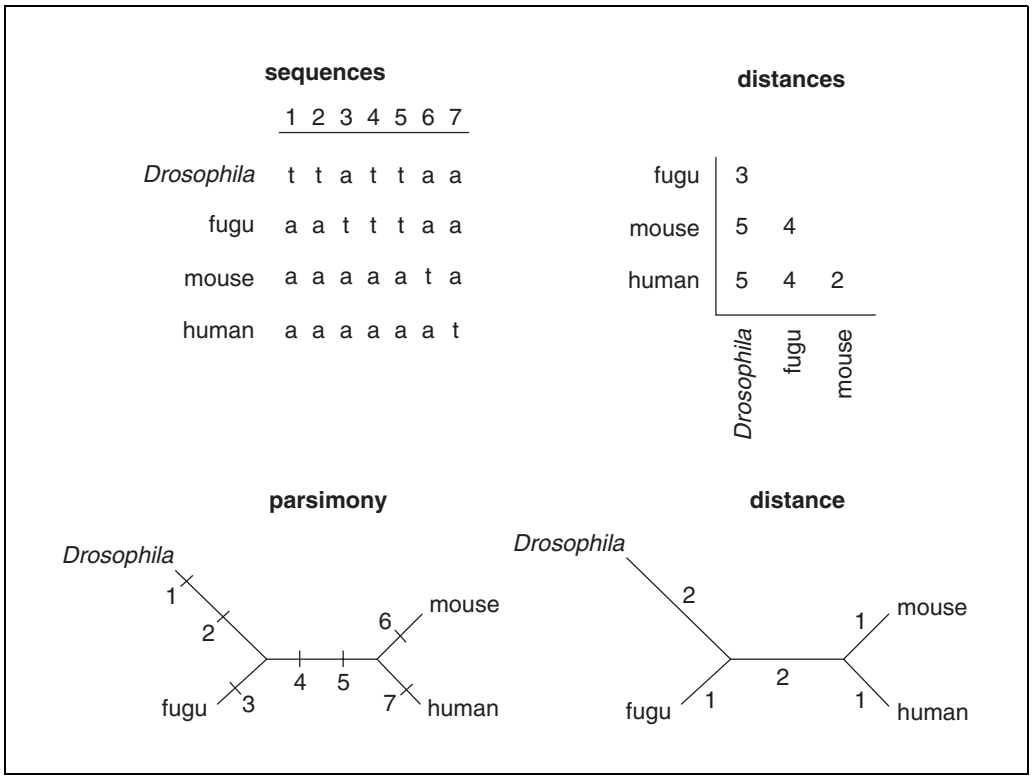


**Figure 6.1.7** A set of four DNA sequences and the corresponding distance matrix. A parsimony tree and a distance tree for the same sequence data are shown. Note that both trees have the same topology and branch lengths, but that the parsimony tree identifies which site (numbered 1 to 7) contributes to the length of each branch (after figure 6.1 in Page and Holmes, 1998).

tend to be quick (see benchmarks in *UNIT 6.3*) but may be sensitive to parameters such as the order in which the sequences are added to the growing tree. In addition, the user is often presented with a tree without a measure of how good that tree is, which makes it difficult to compare competing trees.

In contrast, optimality methods work by assigning a score to a tree (e.g., the minimum number of nucleotide substitutions required to evolve the observed sequences on that tree), and then search for the tree with the best score. Optimality methods have the advantage that we can explicitly compare the relative merits of two trees by comparing their scores. The major drawback of optimality methods is the computational burden of finding the optimal tree. As we have seen above, the number of possible trees grows alarmingly quickly as the number of sequences increases. The problem of finding the optimal tree is NP-complete, which means that there is no efficient (i.e., polynomial time) algorithm known for this problem, nor is it likely that there will ever be one. Consequently, it can only be guaranteed that the optimal tree will be found for small numbers of sequences (typically <15). For larger data sets we have no guarantee that the solution found is, in fact, optimal.

Alternatively, we can classify methods by the kind of data they require. This division is based on how the data is treated; distance methods first convert aligned sequences into a pairwise distance matrix, then input that matrix into a tree-building method, whereas discrete methods consider each nucleotide site (or some function of each site) directly. As an example, consider the following sequences and corresponding (uncorrected) distance matrix shown in Figure 6.1.7. The trees obtained by parsimony (a discrete method) and minimum evolution (a distance method) are identical in topology and branch

lengths. The parsimony analysis identifies seven substitutions and places them on the five branches of the tree. The distance tree apportions the observed distances between the sequences over the branches of the tree, and you can see that both methods arrive at the same estimates of the lengths of each branch. Under parsimony each of the seven sites requires one change, for a total of seven changes; if we sum the branch lengths on the distance tree we obtain the same value: 2 + 1 + 2 + 1 + 1 = 7. Note, however, that the parsimony tree gives us the additional information of which site contributes to the length of each branch. Once we convert sequences into distances we lose this information. Furthermore, discrete methods allow us to infer the attributes of extinct ancestors, in this case extinct nucleotide sequences.

### Distance Methods

Distance methods require pairwise distances to be computed between all the sequences prior to tree construction. Once a distance matrix is obtained, a tree can be computed using a variety of methods. *UNIT 6.3* provides a comprehensive review of these methods, including neighbor joining and its relatives BIONJ and WEIGHBOR.

### Discrete Methods

The major discrete methods are maximum parsimony (MP), maximum likelihood (ML), and Bayesian methods. Maximum parsimony chooses the tree or trees that require the fewest evolutionary changes. Maximum likelihood chooses the tree (or trees) that of all possible trees is the most likely to have produced the observed data. Bayesian methods choose the tree with the highest probability under some model of sequence evolution and tree distribution.

### Parsimony

Parsimony analysis uses the principle that we choose the tree that requires the fewest number of evolutionary events to explain the observed data. For example, given the data in Figure 6.1.8, we choose tree 3, as it requires only 10 events compared to the 12 required by trees 1 and 2.

Among parsimony's advantages are that it is relatively straightforward to understand, it apparently makes few assumptions about the evolutionary process, it has been extensively studied mathematically, and some very powerful software implementations are available (e.g., PAUP*, described in *UNIT 6.4*). However, the justification for choosing the most parsimonious tree as the best estimate of phylogeny is the subject of some controversy. Essentially, two main arguments have been presented. The first is that parsimony is a methodological convention that compels us to maximize the amount of evolutionary similarity that we can explain as homologous similarity, that is, we want to maximize the similarity that we can attribute to common ancestry. Any character that does not fit a given tree requires us to postulate that the similarity between two sequences shown by that character arose independently in the two sequences—i.e., that the similarity is due to homoplasy not homology. Hypotheses of homoplasy (such as convergence or parallel evolution) may be judged "ad hoc" in that they are attempts to explain why data do not fit a particular hypothesis. The most parsimonious tree minimizes the number of ad hoc hypotheses required, and for that reason is preferred.

The second argument is that parsimony is based on an implicit assumption about evolution, namely that evolutionary change is rare. Rarity of change implies that the tree that minimizes change is likely to be the best estimate of the actual phylogeny. Under this view, parsimony may be viewed as an approximation to maximum-likelihood methods. Of the two positions, the latter has the advantage that it is possible to explore the

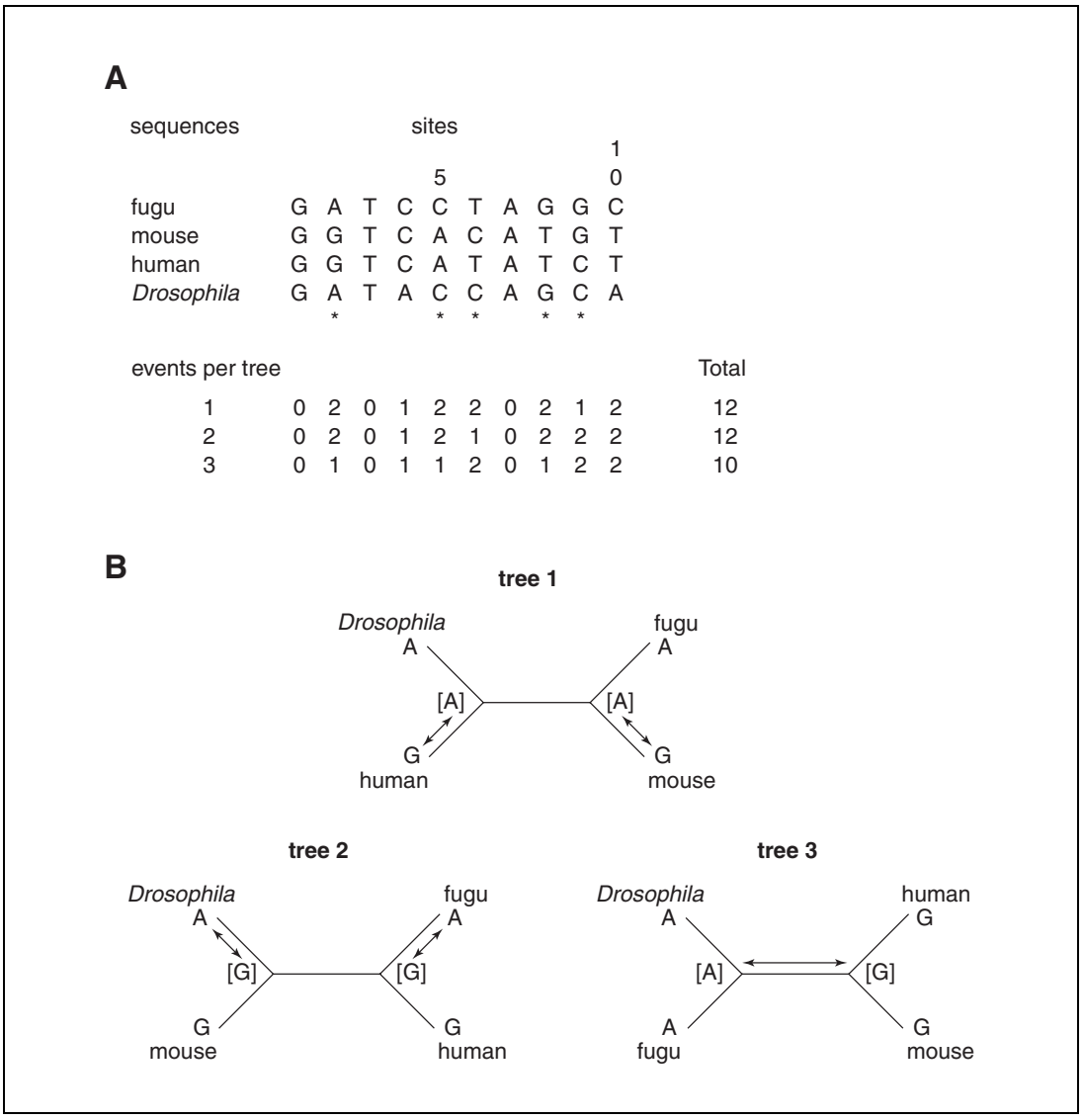


**Figure 6.1.8** The three possible unrooted trees for four sequences. For each tree, the number of events that each of the ten nucleotide sites requires to evolve on that tree is scored. The evolution of site 2 is reconstructed on each tree. Trees 1 and 2 require a minimum of two nucleotide substitutions, but tree 3 can explain by a single substitution the sharing of A by *Drosophila* and fugu and G by mouse and human. Those sites marked with an asterisk (*) are parsimony-informative, i.e., they discriminate between the three trees. The remaining sites have the same score for each tree and hence are uninformative (modified from Stewart, 1993).

circumstances under which parsimony will fail to reconstruct the correct phylogeny, and to develop a framework in which parsimony can be compared to other methods. For a detailed discussion of this issue see Steel and Penny (2000).

### Likelihood Methods

Given competing explanations for a particular outcome, which explanation should we choose? The principle of likelihood suggests that the explanation that makes the observed outcome the most likely (i.e., the most probable) occurrence is the one to be preferred. Put more formally, if given some data and a tree, the likelihood of that data is represented by L = Pr[Data|Tree], where the vertical bar should be read as "given." Maximum likelihood (ML) methods seek the tree that confers the highest likelihood on the data under

a specified model of evolution. Initially, the application of ML methods in phylogenetics was hampered by severe computational limitations, but a combination of improved algorithms for computing likelihoods and faster hardware has made their use practical for small to moderate (say, less than 500 sequences) sized data sets. Increasingly sophisticated models of sequence evolution have been developed that take into account parameters such as variation in base frequency, transition/transversion ratios, and variation in substitution rates across sites (see *UNIT 6.5*).

A powerful aspect of the ML framework is the use of likelihood ratio tests to compare the relative fit of models to data (Huelsenbeck and Rannala, 1997). For example, we can evaluate which of two competing models of nucleotide substitution best explains our data, and make a judgment about whether adding more parameters significantly improves this fit. Likelihood ratio tests underlie the MODELTEST program (*UNIT 6.5*), which provides a tool to assist users in their selection of the most appropriate model for their data.

Another motivation behind ML methods is the early theoretical realization that some evolutionary processes could mislead parsimony. For example, two rapidly evolving but unrelated lineages could be incorrectly grouped together in a parsimony tree. This phenomenon, termed "long branch attraction," was first recognized by Felsenstein (1978), and the parameter space where it becomes a problem has been dubbed the "Felsenstein zone." In the Felsenstein zone, no matter how much data we have, parsimony will converge on the wrong tree. This property is termed "inconsistency." Steel and Penny (2000) provide a useful discussion of consistency with respect to parsimony and maximum likelihood.

### Bayesian Methods

Bayesian methods differ from likelihood methods in that the former are based upon the posterior probability of a tree:

$$\Pr[\text{Tree}|\text{Data}] = \frac{\Pr[\text{Tree}|\text{Data}] \times \Pr[\text{Tree}]}{\Pr[\text{Data}]}$$

The value Pr[Tree|Data] can be interpreted as the probability that the tree is the true tree, under the given model of evolution. The application of Bayesian methods to phylogenetics is a recent development (Huelsenbeck et al., 2001), and has been made possible largely by the use of the Markov chain Monte Carlo (MCMC) numerical technique for integrating over probability distributions. A future unit will discuss Bayesian methods as implemented in the software application MrBayes (Huelsenbeck and Ronquist, 2001).

## CONFIDENCE

Most methods of tree building do not automatically provide the user with any measure of confidence in the tree. An exception is Bayesian inference (Huelsenbeck et al., 2001), in which case a "posterior probability" is associated with each node. This value represents the probability—under the given model of evolution—that the node in question occurs in the true tree. Other methods simply give the user a tree. Given that we may base important, and indeed controversial, biologically inferences upon trees, it is highly desirable to have some indication of how much faith we can place in that tree. "Faith" in this context can mean two different things: is the tree correct, and how robust is the tree? Accuracy is difficult to judge, as we almost never know the true phylogeny. There is a considerable body of simulation work (e.g., Huelsenbeck and Hillis, 1993; see also *UNIT 6.3*) that evaluates the performance of different methods under a variety of evolutionary models,

but it is not easy to apply this work to a specific data set. There are some things the user can check, such as whether the assumptions of any models used have been violated, and whether different methods give similar results see (Kim, 1993), but trees should always be treated with caution.

A popular method of evaluating the robustness of a tree is to use resampling methods such as the bootstrap and the jackknife (Felsenstein, 1985; Farris et al., 1996). Of these two methods, bootstrapping is the most widely used in phylogenetics. Bootstrapping can be applied to phylogenies by generating pseudoreplicates from the sequence data. For example, given an alignment of 1000 base pairs, we could generate a single pseudoreplicate by sampling at random, and with replacement from the original set of sites, until we had a new data set comprising 1000 nucleotide sites. Because we are sampling with replacement (i.e., any site sampled is "returned" to the data set before the next sample is taken), some sites may occur more than once in the pseudoreplicate, while others may not be represented at all. Hence, the pseudoreplicate will resemble the original data set in that it contains only sites found in that data set, but it will differ in the frequencies of different sites. From this pseudoreplicate, we would then build a tree using any of the methods described above. We then repeat this two-step process a large number of times (typically 1000), resulting in a set of bootstrap trees. The frequency with which a group of sequences appears in these bootstrap trees is the "bootstrap value" for that group, and is typically reported as a percentage. The higher the bootstrap value, the greater the support for that grouping. Deciding what level of bootstrap support one should take as indicating strong (or weak) support is something of a judgment call, and the interpretation of these values is a little controversial (Felsenstein and Kishino, 1993; Hillis and Bull, 1993). However, as a rule of thumb, many users get nervous when bootstrap values fall much below 80%.

## WHAT METHOD IS BEST?

The answer to this question depends in part on why you want the tree. If you want a quick estimate of a tree, perhaps as part of an automated analysis of large numbers of sequences, and/or many gene families, then a fast clustering method such as neighbor joining may be appropriate. This method is often employed when constructing secondary databases of gene families (e.g., HOVERGEN; Duret et al., 1994). If, however, you care only about a particular sequence or gene family, and different topologies may have very different biological interpretations, then a thorough investigation of competing models and tree topologies is warranted.

## LITERATURE CITED


Akmaev, V.R., Kelley, S.T., and Stormo, G.D. 2000. Phylogenetically enhanced statistical tools for RNA structure prediction. *Bioinformatics* 16:501-512.

Bafna, V., Hannenhalli, S., Rice, K., and Vawter, L. 2000. Ligand-receptor pairing via tree comparison. *J. Comput. Biol.* 7:59-70.

Duret, L., Mouchiroud, D., and Gouy, M. 1994. HOVERGEN: A database of homologous vertebrate genes. *Nucleic Acids Res.* 22:2360-2365.

Eisen, J. 1998. Phylogenomics: Improving functional predictions for uncharacterized genes. *Genome Res.* 8:163-167.

Farris, J.S., Albert, V.A., Källersjö, M., Lipscomb, D., and Kluge, A.G. 1996. Parsimony jackknifing outperforms neighbor-joining. *Cladistics* 12:99-124.

Felsenstein, J. 1978. Cases in which parsimony and compatibility methods will be positively misleading. *Syst. Zool.* 27:401-410.

Felsenstein, J. 1985. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution* 39:783-791.

Felsenstein, J. and Kishino, H. 1993. Is there something wrong with the bootstrap on phylogenies? A reply to Hillis and Bull. *Syst. Biol.* 42:193-200.

Fitch, W.M. 1970. Distinguishing homologous from analogous proteins. *Syst. Zool.* 19:99-113.

Gulko, B. and Haussler, D. 1996 Using multiple alignments and phylogenetic trees to detect RNA secondary structure. *In* Biocomputing: Proceedings of the 1996 Pacific Symposium (L. Hunter and T. Klein, eds.) World Scientific Press. Singapore.

Hall, B.G. 2001. Phylogenetic Trees Made Easy: A How-to Manual for Molecular Biology. Sinauer Associates, Sunderland, Mass.

Harvey, P.H. and Pagel, M.D. 1991. The Comparative Method in Evolutionary Biology. Oxford University Press. Oxford.

Hein, J. 1990. Unified approach to alignment and phylogenies. *Methods Enzymol.* 183:626-645.

Hillis, D.M. and Bull, J.J. 1993. An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Syst. Biol.* 42:182-92.

Huelsenbeck, J.P. and Hillis, D.M. 1993. Success of phylogenetic methods in the four-taxon case. *Syst. Biol.* 42:247-264.

Huelsenbeck, J.P. and Rannala, B. 1997. Phylogenetic methods come of age: Testing hypotheses in an evolutionary context. *Science* 276:227-232.

Huelsenbeck, J.P. and Ronquist, F. 2001. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 17:754-755.

Huelsenbeck, J.P., Ronquist, F., Nielsen, R., and Bollback, J.P. 2001. Bayesian inference of phylogeny and its impact on evolutionary biology. *Science* 294:2310-2314.

Iwabe, N., Kuma, K.I., Hasegawa, M., Osawa, S., and Miyata, T. 1989. Evolutionary relationship of archaebacteria, eubacteria, and eukaryotes inferred from phylogenetic trees of duplicated genes. *Proc. Natl. Acad. Sci. U.S.A.* 86:9355-9359.

Kim, J. 1993. Improving the accuracy of phylogenetic estimation by combining different methods. *Syst. Biol.* 42:331-340.

Kim, J. and Salisbury, B.A. 2001. A tree obscured by vines: Horizontal gene transfer and the median tree method of estimating species phylogeny. *Pac. Symp. Biocomput.* 6:571-582.

Knudsen, B. and Hein, J. 1999. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics* 15:446-454.

Korbel, J.O., Snel, B., Huynen, M.A., and Bork, P. 2002. SHOT: A Web server for the construction of genome phylogenies. *Trends Genet.* 18:158-162.

Krause, A., Stoye, J., and Vingron, M. 2000. The SYSTERS protein sequence cluster set. *Nucleic Acids Res.* 28:270-272.

Li, P., Goldman, N., Thorne, J.L., and Jones, D.T. 1998. PASSML: Combining evolutionary inference and protein secondary structure prediction. *Bioinformatics* 14:726-733.

Page, R.D.M. and Charleston, M.A. 1997. From gene to organismal phylogeny: Reconciled trees and the gene tree/species tree problem. *Mol. Phylog. Evol.* 7:231-240.

Page, R.D.M. and Charleston, M.A. 1998. Trees within trees: Phylogeny and historical associations. *Trends Ecol. Evol.* 13:356-359.

Page, R.D.M. and Holmes, E.C. 1998. Molecular Evolution: A Phylogenetic Approach. Blackwell Science, Oxford, U.K.

Phillips, A., Janies, D., and Wheeler, W. 2000. Multiple sequence alignment in phylogenetics. *Mol. Phylog. Evol.* 16:317-330.

Rehmsmeier, M. and Vingron, M. 2001. Phylogenetic information improves homology detection. *Proteins* 45:360-371.

Steel, M. and Penny, D. 2000. Parsimony, likelihood, and the role of models in molecular phylogenetics. *Mol. Biol. Evol.* 17:839-850.

Stewart, C.-B. 1993. The powers and pitfalls of parsimony. *Nature* 361:603-607.

Storm, C.E.V. and Sonnhammer, E.L.L. 2001. Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics* 18:92-99.

Swofford, D.L., Olsen, G.J., Waddell, P.J., and Hillis, D.M. 1996. Phylogenetic inference. *In* Molecular Systematics (D.M. Hillis, C. Moritz, and B.K. Mable, eds.) pp. 407-514. Sinauer Associates, Sunderland, Mass.

Wang, L.-S., Jansen, R.K., Moret, B.M.E., Raubeson, L.A., and Warnow, T. 2002 Fast phylogenetic methods for the analysis of genome rearrangement data: An empirical study. *In* Pacific Symposium on Biocomputing 2002 (R.B. Altman, A.K. Dunker, L. Hunter, K. Lauderdale and T.E. Klein, eds.), pp. 524-535. World Scientific Publishing. Singapore.

Yuan, Y.P., Eulenstein, O., Vingron, M., and Bork, P. 1998. Towards detection of orthologs in sequence databases. *Bioinformatics* 14:285-289.

Zmasek, C.M. and Eddy, S.R. 2001. A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics* 17:821-828.

Zmasek, C.M. and Eddy, S.R. 2002. RIO: Analyzing proteomes by automated phylogenomics using resampled inference of orthologs. *BMC Bioinformatics* 3:14.

## KEY REFERENCES

Page and Holmes, 1998. See above.

*An introduction to molecular evolution and phylogenetic analysis.*

Sanderson, M.J. and Shaffer, H.B. 2002. Troubleshooting molecular phylogenetic analyses. *Annu. R. Ecol. Syst.* 33:49-72.

*Excellent overview of the problems encountered when building phylogenies, with helpful suggestions for what (if anything) can be done.*

Hall, 2001. See above.

*A nicely written how-to manual describing how to build trees using Clustal, PAUP*, and MrBayes, among other programs,*

Swofford et al., 1996. See above.

*Detailed review of phylogenetic methods.*

## INTERNET RESOURCES

http://evolution.genetics.washington.edu/phylip/software.html

*Joe Felsenstein's list of phylogeny programs.*

http://www.bioinf.org/molsys/

*Online molecular systematics and evolution course run by The Natural History Museum, London, and the National University of Ireland, Maynooth.*

---

Contributed by Roderic D.M. Page
University of Glasgow
Glasgow, Scotland

# Visualizing Phylogenetic Trees Using TreeView

TreeView (Page, 1996) provides a simple way to view the phylogenetic trees produced by a range of programs, such as PAUP* (*UNIT 6.4*) and PHYLIP (see Internet Resources; *UNIT 6.3*), TREE-PUZZLE (Strimmer and von Haeseler, 1996), and ClustalX (Thompson et al., 1997; *UNIT 2.3*). While some phylogenetic programs (such as the Macintosh version of PAUP*) have excellent tree-printing facilities, many programs do not have the ability to generate publication-quality trees. TreeView addresses this need. The program can read and write a range of tree file formats, display trees in a variety of styles, print trees, and save the tree as a graphic file.

Protocols are presented for displaying (see Basic Protocol 1) and printing a tree (see Basic Protocol 2). The Support Protocols describe how to download and install TreeView, as well as how to display bootstrap values in trees generated by ClustalX and PAUP*.

## DISPLAYING A PHYLOGENETIC TREE

***BASIC PROTOCOL 1***

This protocol presents the basic steps for viewing a Newick tree file in TreeView. The tree file may have been generated in ClustalX (*UNIT 2.3*) or PAUP* (*UNIT 6.4*), or by some other program. The procedure below discusses the various display options, such as viewing internal node labels, choosing the style in which the tree is drawn, and designating an outgroup. The final step allows the user to define default display preferences.

### *Necessary Resources*

*Hardware*

TreeView can be run on Windows or Macintosh computers (TreeView X is a Unix port with a subset of TreeView's features)

*Software*

TreeView (see Support Protocols 1 and 2 for installation instructions)

*Files*

A tree file in Newick or NEXUS format

*An example Newick tree file is shown in Figure 6.2.1. The Newick format is described below (see Background Information).*

1. Obtain, install, and run TreeView (see Support Protocols 1 and 2).

### *Viewing a tree*

2. Select Open from the File menu. Choose the tree file from the files listed in the Open File dialog box. TreeView will read the file and display it in a Tree window (Fig. 6.2.2).

```
(Lemur_catta:106,Tarsius_syrichta:134,(Saimiri_sciureus:109,
((Hylobates:71,(Pongo:65,(Gorilla:35,(Homo_sapiens:39,Pan:41):19
):51):37):48,(M._sylvanus:42,(M._fascicularis:36,(Macaca_fuscata:12,
M._mulatta:20):27):32):79):60):90);
```

**Figure 6.2.1** A Newick tree description. This tree is displayed graphically in Figure 6.2.2.

**Contributed by Roderic D.M. Page**
*Current Protocols in Bioinformatics* (2003) 6.2.1-6.2.15

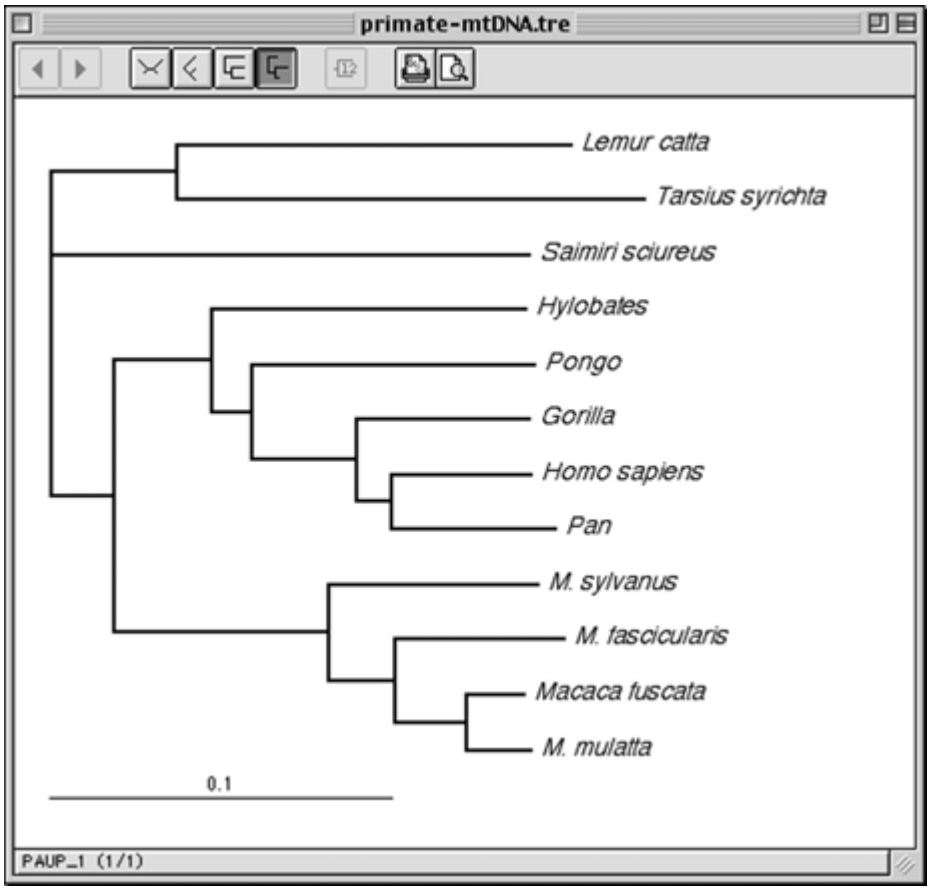


**Figure 6.2.2** A tree window in TreeView displaying a phylogenetic tree as a phylogram. The scale bar represents 0.1 substitutions per nucleotide site. The toolbar buttons enable the user to change the style in which the tree is drawn, toggle on and off the display of internal node labels, and print the tree. The screenshot shows TreeView running under MacOS; the Windows version is almost identical in appearance.

*Tree files that are associated with TreeView (for example, have the extension* `.TRE`*,* `.PHB`*,* `.PH`*,* `.DND`*, or* `.TREES` *under the Windows operating system) can also be opened by double clicking on their icon. Files can also be dragged onto the TreeView application icon, or the application itself (Windows only).*

### *Altering the display*

3. If the tree file contains more than one tree, the user can step through the trees by clicking on the Forward and Back buttons on the toolbar. The user can also choose a tree to display from a list of all the trees in the file by selecting the Choose Tree command from the Trees menu.

4. Some programs store information about the internal nodes as labels for those nodes. These labels may, for example, represent higher taxon names and bootstrap values (see Support Protocols 3 and 4). If present, internal node labels can be displayed by selecting the Show Internal Edge Labels command from the Trees menu.

   *Internal labels look best when the tree is drawn as a rectangular cladogram (see Guidelines for Understanding Results).*

5. The style in which the tree is drawn can be changed to unrooted, slanted cladogram, rectangular cladogram, or phylogram (if the tree has branch lengths; see Guidelines

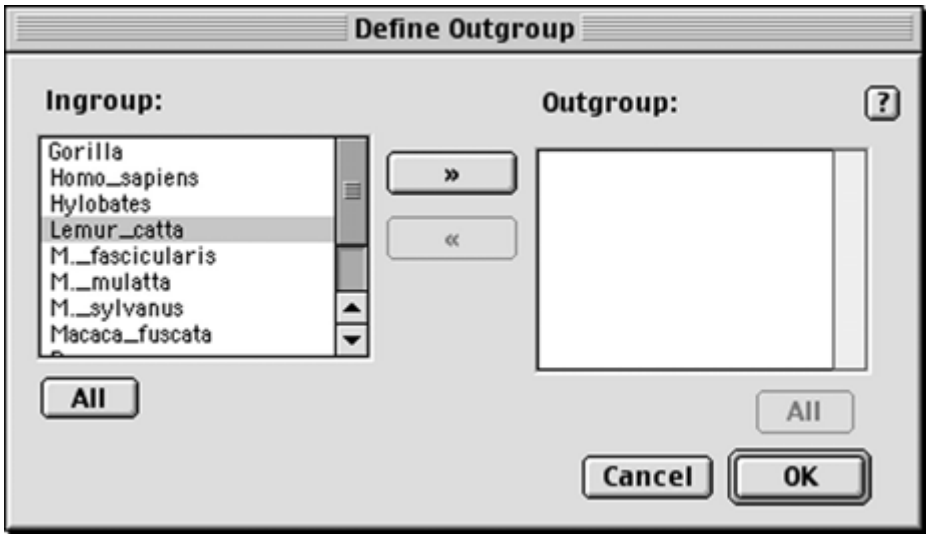


**Figure 6.2.3** Using the Define Outgroup dialog box to specify an outgroup. In this example the species *Lemur catta* is selected in the ingroup list. Double clicking the selected item, or clicking on the ">>" button will move *Lemur catta* to the outgroup list.

for Understanding Results) by selecting the corresponding command from the Trees menu, or using the buttons on the tool bar (Fig. 6.2.2).

*A rooted tree has a node identified as the root from which all other nodes ultimately descend; hence, a rooted tree has a direction. The direction corresponds to evolutionary time; the closer a node is to the root of the tree, the older it is in time. Rooted trees allow us to define ancestor-dependent relationships between nodes—given a pair of nodes connected by a branch, the node closest to the root is the ancestor of the node further away from the root (the descendant). An unrooted tree lacks a root and hence does not specify ancestor-dependent relationships. Most tree-building methods construct unrooted trees, which are then rooted using additional information, such as an outgroup (see below).*

*A cladogram is a rooted tree that simply shows relative recency of common ancestry, but with no edge lengths. A phylogram is a rooted tree where the edges have "lengths,"usually proportional to the amount of evolutionary change inferred to have occurred along that branch (see Guidelines for Understanding Results).*

6. The font used to draw the leaf labels can be set using the Style menu; the font for the internal labels is set using the Internal Label Font command from the Trees menu.

7. The tree can be drawn such that "heavier" nodes (i.e., those with more descendants) are drawn either toward the top or towards the bottom, or the original order can be restored by using the Order command from the Trees menu.

***Define an outgroup***

8. To reroot the tree, define an outgroup using the Define Outgroup command on the Trees menu. This command displays a dialog box (Fig. 6.2.3) listing those leaves that are in the ingroup and those that are in the outgroup. Leaves can be moved from one list to the other by double clicking on their name in the appropriate list box, or by selecting their names and clicking on the "<<" and ">>" buttons.

*An unrooted tree can be rooted by dividing the set of sequences or species contained in the tree into two sets—the "ingroup"and the "outgroup"—and placing the root along the edge connecting these two groups. Outgroups are assumed to be related to, but not members of,*

*the ingroup. For example, we could root a tree for human, chimp, and gorilla using a monkey, or we could root a tree for a gene family using a member of another, related gene family.*

*The tree will not be rooted with the outgroup until the user selects the Root With Outgroup command.*

***Setting default viewing preferences***

9. The user can set many of the default options that govern how a tree is drawn when the tree file is first open. Selecting the Preferences command from the Edit menu results in TreeView displaying the Preferences dialog box. The user can specify the default tree style, fonts, and order that will be used when a tree is first displayed.

10. Save and print the tree (see Basic Protocol 2).

***BASIC PROTOCOL 2***

## PRINTING ONE OR MORE PHYLOGENETIC TREES

TreeView can print a single tree on one page, one tree over several pages, or several trees on the same page.

***Necessary Resources***

*Hardware*

TreeView can be run on Windows or Macintosh computers (TreeView X is a Unix port with a subset of TreeView's features)

*Software*

TreeView (see Support Protocols 1 and 2 for installation instructions)

*Files*

A tree file in Newick or NEXUS format

*An example Newick tree file is shown in Figure 6.2.1. The Newick format is described below (see Background Information).*

1. Open the tree file in TreeView (see Basic Protocol 1).

***Printing a tree***

2a. Select the Print command from the File menu. The program will display the standard system-specific Print dialog box, from which the user can print the tree.

*In addition to printing the trees, the current tree can be saved to a graphics file by selecting the Save as Graphic command on the File menu, or can be copied to the clipboard by selecting Copy on the Edit menu. On a Macintosh the graphic is stored as a PICT file, under Windows it is stored as either the Windows Metafile (WMF) or Enhanced Windows Metafile (EMF) formats.*

***Printing a tree over more than one page***

2b. Select Print Preview from the File menu. TreeView will display the tree as it will appear on the printed page (Fig. 6.2.4).

3b. Click on the Pages popup menu to specify how many pages the tree should span. The preview is automatically updated to reflect the changed setting.

4b. Click on the Print button.

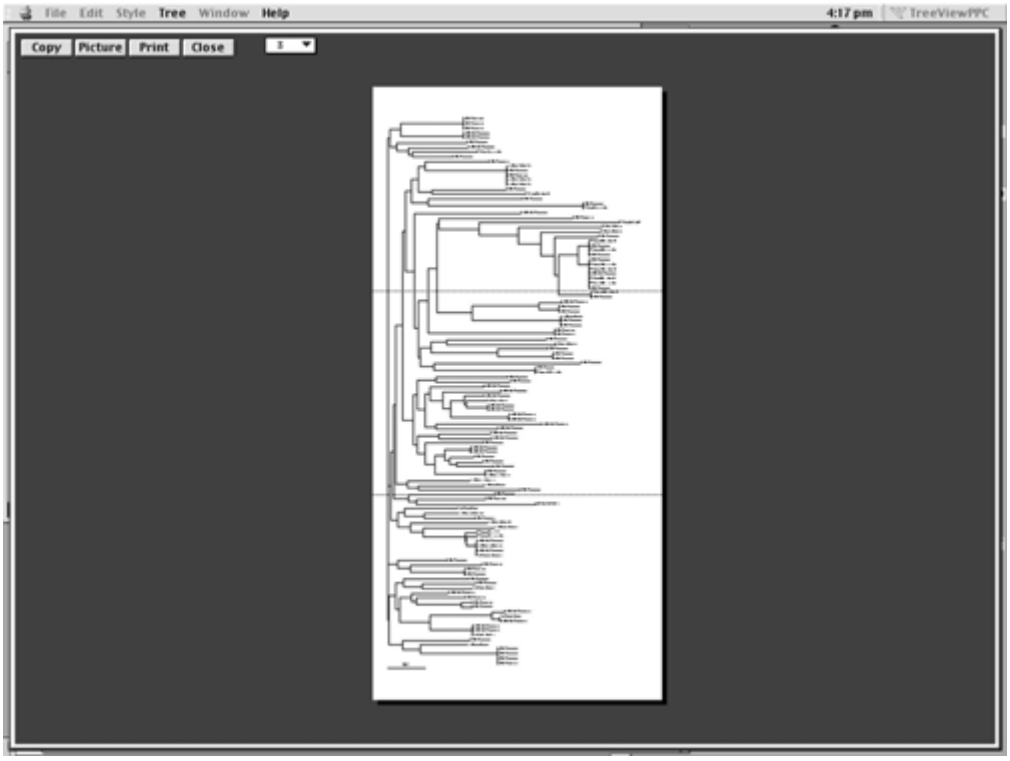


**Figure 6.2.4** The Print preview window in TreeView. In this example a large tree is being printed over three pages.

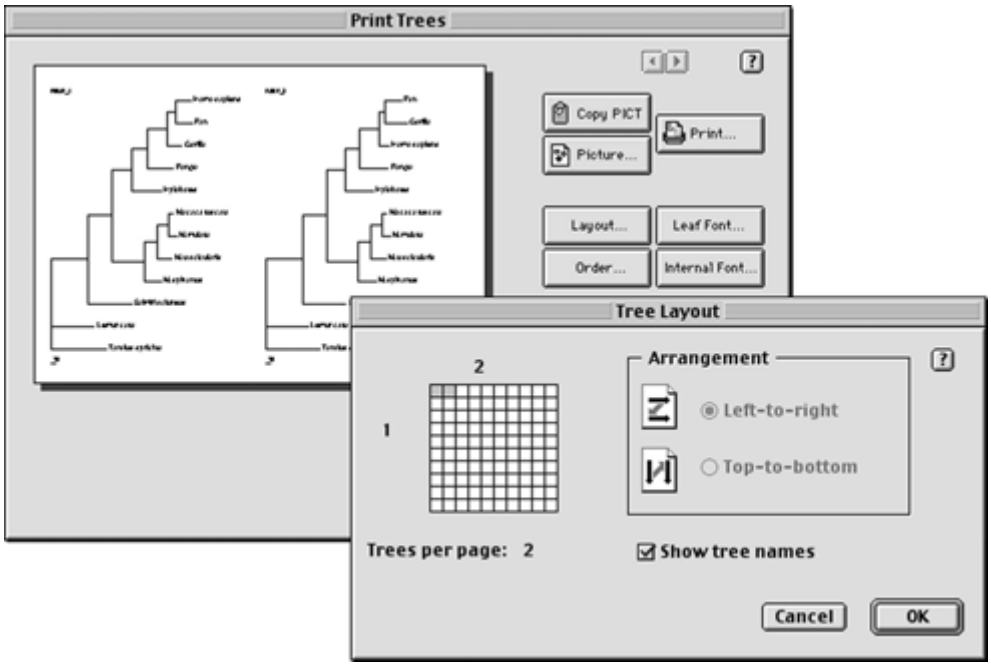


**Figure 6.2.5** The Print Trees dialog in TreeView. The Layout dialog box specifies how many trees are drawn per page, and the order in which they are drawn. Other options that can be set are the fonts used for the leaf and internal labels, tree style, branch width, and tree order.

***Printing more than one tree per page***

2c. Select Print Trees from the Trees menu. TreeView displays a dialog box showing a preview of the printed page (Fig. 6.2.5).

3c. To specify the number of trees to be displayed on a single page, click on the layout button. The program displays a dialog box containing a $10 \times 10$ grid representing the layout of trees on the page. The user can specify how many trees are to be shown (up to a maximum of 100), whether to display the names of the trees, and whether the trees are drawn left to right or top to bottom page (Fig. 6.2.5).

4c. To print the trees, click the Print button.

*In addition to printing the trees, the current page being displayed can be saved to a graphics file by clicking on the Picture button, or copied to the clipboard by clicking on the Copy button.*

***SUPPORT PROTOCOL 1***

## OBTAINING THE TREEVIEW PROGRAM

TreeView can be obtained from *http://taxonomy.zoology.gla.ac.uk/rod/treeview.html*. Executables are available for MacOS 7.5 and later, and Windows 95/NT and later.

Obtain TreeView by downloading the appropriate file from *http://taxonomy.zoology.gla.ac.uk/rod/treeview.html*. The MacOS version is packaged as a self-extracting archive. The Windows version is packaged as a zip file which must be unpacked using a utility such as WinZip (*http://www.winzip.com/*); run the program `setup.exe` to install TreeView.

Documentation for TreeView is available online at *http://taxonomy.zoology.gla.ac.uk/rod/treeview/treeview_manual.html*.

TreeView can be configured as a helper application for your Web browser by associating the MIME type `application/x-treeview` with the program TreeView. For the Navigator Web browser the MIME type can be set in the Preferences dialog box found under the Edit menu.

***SUPPORT PROTOCOL 2***

## OBTAINING THE TREEVIEW X PROGRAM

TreeView X is available as C++ source code from *http://darwin.zoology.gla.ac.uk/~rpage/treeviewx/*. TreeView X is a port of TreeView to the Unix platform. The program has a subset of the functionality of TreeView (e.g., it only displays rooted trees, printing is limited to a single page, and there is no ability to reroot trees). To display a tree using TreeView X, see Basic Protocol 1, steps 1 to 6; to print a tree, see Basic Protocol 2, steps 1 to 2a.

***Necessary Resources***

*Hardware*

A Unix system with a TCP/IP Internet connection and a Web browser

*Software*

Version 2.*x* of the wxWindows C++ class library is required (available from *http://www.wxwindows.org*); wxWindows supports either the Motif or GTK widget sets

1. To obtain TreeView X download the source (`tv-xx.tar.gz` where "`xx`" is the current version) from *http://darwin.zoology.gla.ac.uk/~rpage/treeviewx/.*

2. Unpack the distribution tarball:

```
gunzip tv-xx.tar.gz
tar-xvf tv-xx.tar
```

   This will create a directory `tv-xx` containing the source code for TreeView X.

3. Change to the TreeView X directory and issue the following commands:

```
./configure
make
```

   This will build the executable "`tv`".

4. To run TreeView X type the command

```
./tv &
```

   There is no specific install procedure once you have built the executable. You may want to copy the executable to somewhere on your path (such as `$HOME/bin`).

## DISPLAYING BOOTSTRAP VALUES IN CLUSTALX TREES

***SUPPORT PROTOCOL 3***

Bootstrap values are a measure of support for a node in a tree. These are usually given as the percentage of bootstrap trees in which that node appeared. Bootstrap trees are obtained by generating a large number (typically 1000 or more) of new data sets, each obtained by randomly resampling with replacement from the original alignment, and generating a tree from each data set.

Whereas earlier versions of Clustal stored bootstrap values as internal node labels, in ClustalX (Thompson et al., 1997) bootstrap values are stored as "branch labels." These labels are placed inside square brackets, "[ ]", after the branch lengths. Because TreeView follows the Newick standard and ignores text inside "[ ]" (any such text is treated as a comment), TreeView will not display bootstrap values saved using ClustalX's default method. This protocol ensures the bootstrap values are saved in a format TreeView can read.

### *Necessary Resources*

*Hardware*

ClustalX can be run on Macintosh, Windows, and Unix systems

*Software*

ClustalX (see *UNIT 2.3*)

*Files*

Aligned sequence file of interest

1. Run ClustalX (see *UNIT 2.3*).

2. Before computing a bootstrap tree, select the Output Format Options command on the Trees menu. ClustalX will display the Output Tree Format Options dialog box. From the Bootstrap labels on popup menu, select Node.

*In ClustalW the equivalent command line option is* `-BOOTLABELS=node`. *This is the default setting, but in the author's experience some Web CGI interfaces to ClustalW do not set this option correctly.*

3. Calculate and save the tree in ClustalX (*UNIT 2.3*).
4. Open the tree file in TreeView and select Show Internal Edge Labels from the Trees menu (see Basic Protocol 1).

***SUPPORT PROTOCOL 4***

## DISPLAYING BOOTSTRAP VALUES IN PAUP* TREES

Bootstrap values are a measure of support for a node in a tree. These are usually given as the percentage of bootstrap trees in which that node appeared. Bootstrap trees are obtained by generating a large number (typically 1000 or more) of new data sets, each obtained by randomly resampling with replacement from the original alignment, and generating a tree from each data set.

In order to correctly display bootstrap values in a bootstrap tree computed using PAUP*, the user needs to tell PAUP* (*UNIT 6.4*) how to save bootstrap values in a format that TreeView will recognize.

### *Necessary Resources*

*Hardware*

PAUP* (*UNIT 6.4*) can be run on Macintosh, Windows, and Unix systems

*Software*

PAUP* (see Internet Resources; *UNIT 6.4*)

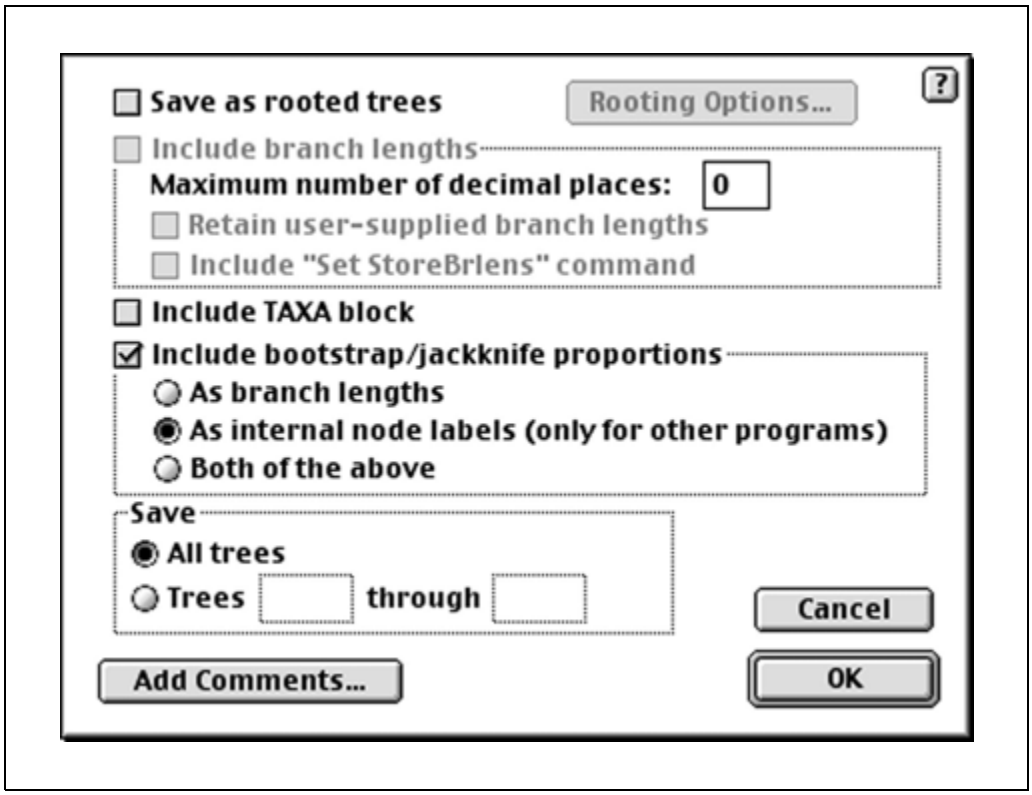


**Figure 6.2.6** PAUP* dialog box displaying options that affect how trees are saved. To save bootstrap values in a format that TreeView can display, the value of Maximum number of decimal is set to 0, the Include Bootstrap/Jackknife Proportions checkbox is checked, and the As Internal Node Labels option is selected.

*Files*

Aligned sequence file of interest

1. After the bootstrap analysis has been completed, save the trees to a file. For the Macintosh version of PAUP* select the Save Trees command from the Trees menu. The Save Trees dialog box will appear. Click on the Options button to display a dialog box listing options the user can change (Fig. 6.2.6). Set the Maximum number of decimal places to 0, and check the Include Bootstrap/Jacknife Proportions checkbox. Choose the option As Internal Node Labels (only for other programs). Click on the OK button to close the dialog box, then save the trees.

   *The equivalent command for the command line version of PAUP* (Windows and Unix) is:*

   ```
   savetree SaveBootP=NodeLabels MaxDecimals=0;
   ```

2. Open the tree file in TreeView and select Show Internal Edge Labels from the Trees menu (see Basic Protocol 1).

## GUIDELINES FOR UNDERSTANDING RESULTS

### Tree Drawing Styles

There are many different ways of drawing trees, so it is important to know whether these different ways actually reflect differences in the kind of tree, or whether they are simply stylistic conventions. For instance, the order in which the labels on a tree are drawn on a piece of paper (or computer screen) can differ without changing the meaning of the tree. The same tree can be drawn in different styles to highlight different aspects, such as branch lengths and evolutionary relationship. Figure 6.2.7 shows the same tree drawn in the four styles available in TreeView: unrooted, slanted cladogram, rectangular cladogram, and phylogram. The unrooted tree has the same topology as the three rooted styles, but lacks any indication of evolutionary direction. The two cladograms differ solely in the way the branches are drawn. Note that the length of the branches in the cladogram has no meaning, whereas in the phylogram the branch lengths are proportional to the evolutionary change inferred to have occurred along each branch.

### Scale Bar

If the tree being viewed has branch lengths, and you are viewing the tree as either unrooted or as a phylogram, then TreeView displays a scale bar in the bottom left corner of the tree window (e.g., Fig. 6.2.2). The units for this scale depend on the data and method used to construct the original tree; this information is not available to TreeView itself. Most tree-building programs save the tree with branch lengths as number of substitutions (nucleotides) or replacements (amino acids) per site. For such trees a value of "0.1"above the scale bar means 0.1 nucleotide substitutions per site. Other trees, such as those computed using parsimony, may have integer branch lengths (i.e., 1, 5, 10), and hence the scale bar will be in units of "steps."

One note of caution concerns "branch lengths" on trees obtained using PHYLIP's CONSENSE program. These branch lengths are actually the frequency of the corresponding clade in the set of input trees (if the trees were bootstrap trees then these would correspond to bootstrap values). Because PHYLIP puts these numbers where TreeView expects the branch lengths, TreeView will display a scale bar. This scale bar should be ignored; it is best to display PHYLIP consensus trees as rectangular cladograms.

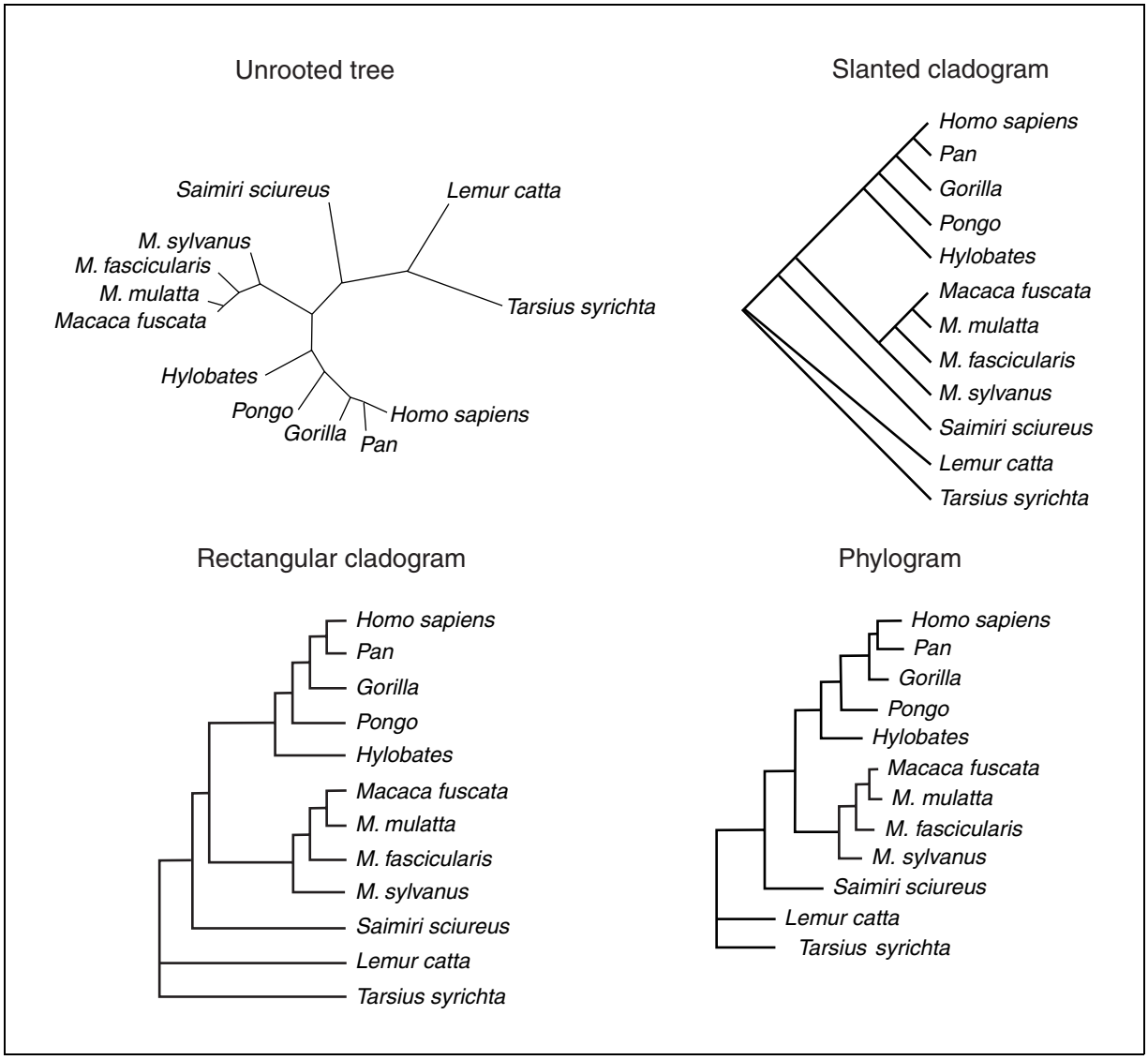


**Figure 6.2.7** The same tree drawn in the four different styles available in TreeView.

## COMMENTARY

### Background Information

#### *Newick tree format*

The "Newick" tree format (sometimes also referred to as the "New Hampshire" format) is a widely used standard for describing phylogenetic trees. It was developed by James Archie, William H.E. Day, Joe Felsenstein, Wayne Maddison, Christopher Meacham, F. James Rohlf, and David Swofford in 1986 (the name comes from the lobster restaurant at which the format was agreed upon). Typically a phylogenetic program will store one or more Newick tree descriptions in a file. More elaborate file formats, such as NEXUS (Maddison et al., 1997), embed the Newick tree description in a set of commands that provide additional information about the tree(s).

A tree description consists of a series of nested parentheses, each pair of "()" enclosing a subtree. Commas separate subtrees, and a semicolon terminates the tree description. The leaves of the tree are labeled by their names (e.g., a sequence, an organism). A label can be any printable character, except a blank (space), colon, semicolon, parentheses, or square brackets. The underscore character, "_", can be used to represent a blank; most programs will replace underscores with blanks once the tree has been read in. Labels can be enclosed in single quotes, e.g., 'Homo sapiens', in which case any character is allowed between the single quotes. A single

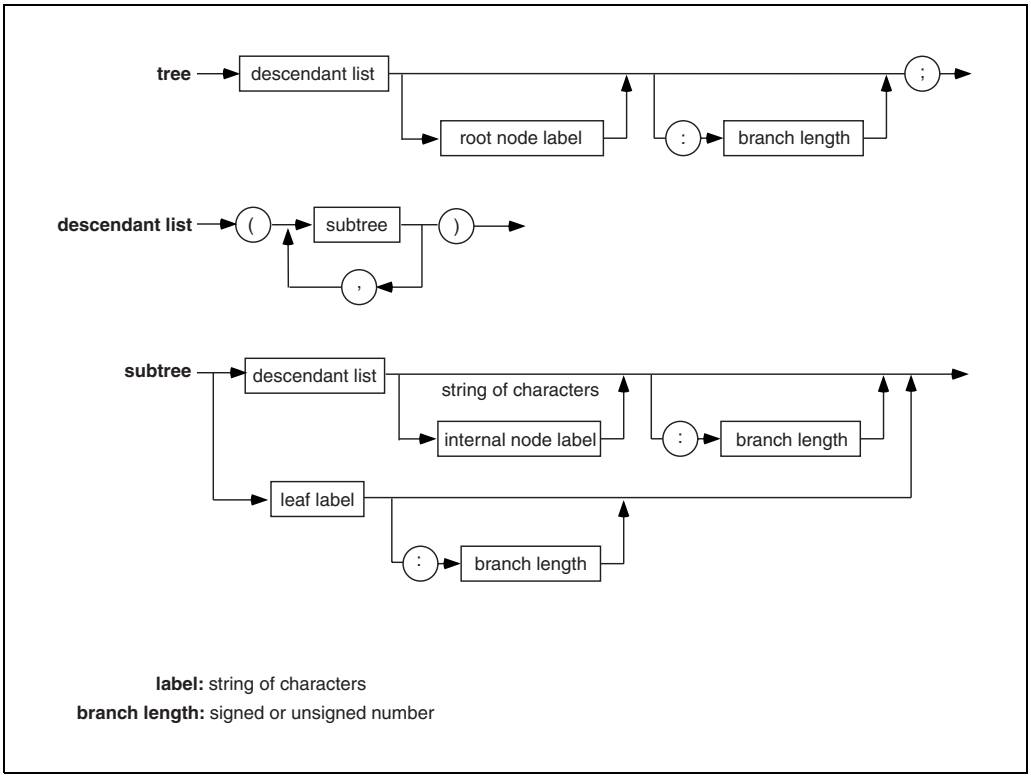


**Figure 6.2.8** Syntax diagram for the Newick Standard for describing phylogenetic trees. The names in rectangular boxes stand for elements of the tree description, such as the name of a leaf in the tree; punctuation in circular boxes are required parts of the format. Any path through the diagram that begins at the left and ends with an arrow on the right is a valid path. Alternative paths are often possible; for example, a descendant list may or may not have an associated internal node label and a branch length.

quote within a label enclosed in single quotes is represented by two single quotes, e.g., 'Huntington''s Disease'. Branch lengths are represented by integer or real numbers. Figure 6.2.8 shows a syntax diagram for the Newick format, based on Gary Olsen's description of the format (*http://rdp.cme.msu.edu/docs/treeview_newick.html*). Comments are enclosed in square brackets, "[ ]", and can appear anywhere in the tree description. Most programs will ignore these comments.

Some example Newick tree descriptions and the corresponding trees shown in Figure 6.2.9 are fully resolved. In a fully resolved tree, every internal node gives rise to two descendants; such a tree is also called a binary tree. The tree can be nonbinary. For example, the tree:

(One,Two,Three,Four,Five);

is completely unresolved.

The Newick representation of a tree is not unique. The left-right order of the nodes in the tree is biologically irrelevant, hence:

(((One,Two),(Three,Four)),Five);

and:

(Five,((Four,Three),(Two,One)));

are the same tree. Furthermore, the Newick format describes a rooted tree. Given that a binary unrooted tree for $n$ leaves corresponds to $2n - 5$ rooted trees, there is more than one way to represent the same unrooted tree using the Newick format. Figure 6.2.10, panel A, shows an unrooted tree and its associated Newick description. This same description specifies the rooted tree shown in panel B, which has a basal trichotomy (i.e., a three-way split at the root). Some programs, such as those in the PHYLIP package, use the presence of a basal trichotomy to mark the tree as unrooted. Other programs, such as those using the NEXUS standard (Maddison et al., 1997) employ additional commands in the data file to specify whether the tree is rooted or unrooted, irrespective of the degree of resolution of the basal node. For these programs, the rooted tree shown in Figure 6.2.10A is interpreted to be the same tree as the unrooted tree in Figure 6.2.10B.

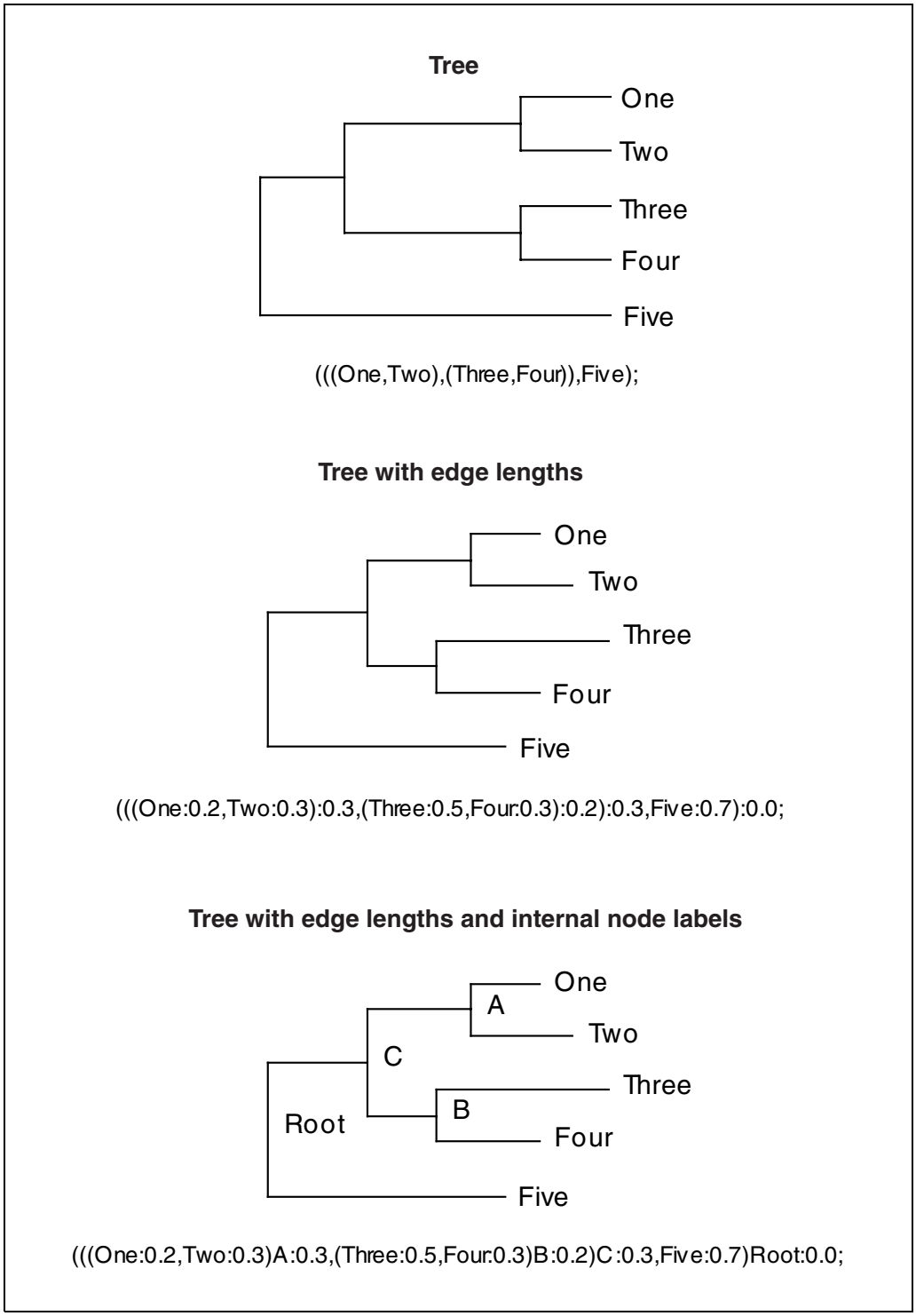


**Figure 6.2.9** Three trees and their corresponding Newick tree descriptions. The top tree description specifies the topology of the tree, the middle specifies the same tree but with the addition of branch lengths, and the bottom description adds internal node labels.

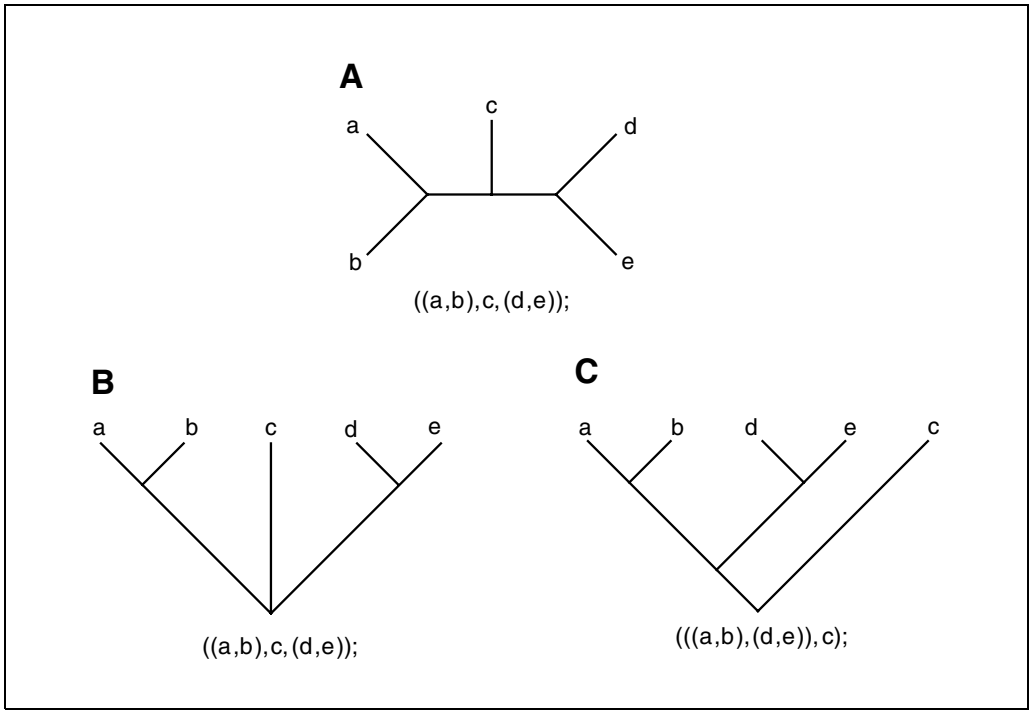


**Figure 6.2.10** An unrooted tree (**A**) and two rooted trees (**B**,**C**) that are consistent with that unrooted tree.

***Writing your own Newick format trees***

On occasion, the user might need to write a tree description by hand, rather than use one created by a program. For example, the user might need to input a tree topology into a program, rather than have that program search for a tree. To write a tree description, simply visit all nodes in the tree, starting at the root, and follow these rules:

If the node is a leaf, write the node's label, then return to the node's immediate ancestor.

If the node is an internal node:

1. If you are visiting the node for the first time, write a left parenthesis, "(", then visit the node's leftmost child.
2. If you have already visited the node, but have not yet visited all of that node's descendants, write a comma, ",", then visit the next descendant of the node (going from left to right).
3. If you have already visited the node, and you have visited all the node's descendants, write a right parenthesis, ")", and visit the node's immediate ancestor (if any). If the current node is the root, then stop.

For the tree shown in Figure 6.2.11, the tree description grows as shown in Table 6.2.1.

***Extensions to the Newick standard***

A limitation of the Newick standard is that it only allows two kinds of information to be attached to a node in a tree: a label and a branch length. In practice, we might want to associate other information with nodes and edges (e.g., bootstrap values, likelihood scores, species names for sequences, *x* and *y* graphical coordinates). Because the Newick standard treats any items enclosed in square brackets, "[ ]", as comments, this provides a mechanism where additional information can be embedded in the tree description. Programs that do not understand the embedded information can simply skip over it. This approach is used by the authors of ClustalX (Thompson et al., 1997), TreeTool, the NEXUS standard (Maddison et al., 1997), and the New Hampshire X (NHX) format developed by Christian M. Zmasek (*http://www.genetics.wustl.edu/eddy/forester/NHX.html*). In some cases, different programmers have added the same information in different, mutually incompatible ways (see the discussion of bootstrap values below). It is likely that an additional standard for describing phylogenetic trees will need to be developed to accommodate the obvious need to append more information to trees. Otherwise, the simplicity of the Newick standard may become lost as it is burdened with tasks beyond describing the topology of a tree.

***Other tree viewing programs***

There are a range of tree-viewing programs available, besides TreeView. These include NJplot (Perrière and Gouy, 1996), ATV (Zmasek and Eddy, 2001), available as either a Java applet or stand-alone application, TreeTool (*http://rdp.cme.msu.edu/download/programs/TreeTool/*), and TreeEdit (*http://evolve.zoo.ox.ac.uk/software/TreeEdit/main.html*). A comprehensive list can be found at *http://evolution.genetics.washington.edu/phylip/software.html#Plotting*. Most use either the Newick or NEXUS formats.

## Critical Parameters and Troubleshooting

Although the Newick tree description is a well established standard, there are a few areas

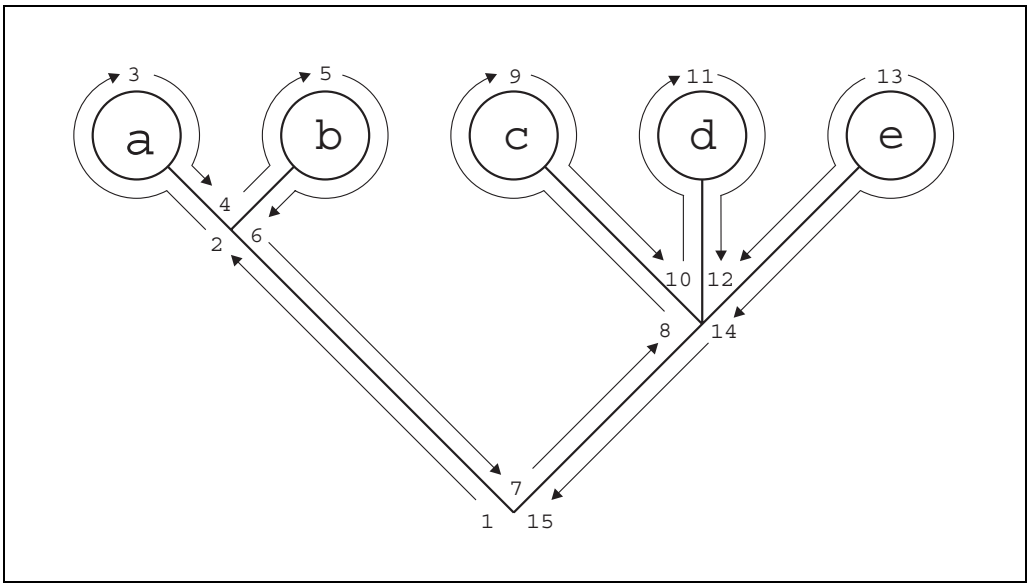


**Figure 6.2.11** An example of the order in which nodes in a tree are visited when writing a Newick tree description.

**Table 6.2.1** Tree Description Corresponding to Figure 6.2.11

| Step | Tree description |
|---|---|
| 1 | ( |
| 2 | (( |
| 3 | ((a |
| 4 | ((a, |
| 5 | ((a,b |
| 6 | ((a,b) |
| 7 | ((a,b), |
| 8 | ((a,b),( |
| 9 | ((a,b),(c |
| 10 | ((a,b),(c, |
| 11 | ((a,b),(c,d |
| 12 | ((a,b),(c,d, |
| 13 | ((a,b),(c,d,e |
| 14 | ((a,b),(c,d,e) |
| 15 | ((a,b),(c,d,e)) |
| 16 | ((a,b),(c,d,e)); |

where different tree-building programs interpret the standard slightly differently. This is especially true for bootstrap values, which can be stored as branch lengths, internal node labels, or comments, depending on the program used.

### *Displaying bootstrap values*

There is little consensus about where in a tree description one should place bootstrap values, and hence different programs support different solutions, such as replacing branch lengths with bootstrap values (PHYLIP, optional in PAUP*), using node labels (ClustalW, optional in PAUP* and ClustalX), or placing them inside brackets "[ ]" after the branch lengths (ClustalX default). To display bootstrap labels in trees obtained using ClustalX and PAUP*, see the instructions given in Support Protocols 3 and 4, respectively.

### *Copy and pasting from the clipboard*

TreeView supports copy and paste operations using the Clipboard. When the user selects Copy on the Edit menu, TreeView copies the current tree to the Clipboard as *both* a picture and as a Newick format tree description in the form of text. The picture is scaled to fit onto the current page settings for the system printer (these can be changed using the Print setup command on the File menu), and can be pasted into another application, such as a graphics program or word processor. Because both graphic and text representations of the tree are placed on the Clipboard, in programs that support both text and pictures (e.g., word processors) you may need to use that program's Paste special command (or its equivalent) to select either the picture of the tree or its text description.

## Suggestions for Further Analysis

### *Other features of TreeView*

Other features of TreeView are described in the documentation available from the program's Web site (see Internet Resources). These include the ability to paste Newick format tree descriptions from the Clipboard directly into TreeView, edit the tree's topology, and export the tree in a variety of formats.

## Literature Cited


Maddison, D.R., Swofford, D.L., and Maddison, W.P. 1997. NEXUS: An extensible file format for systematic information. *Syst. Biol.* 46:590-621.

Page, R.D.M. 1996. TreeView: An application to display phylogenetic trees on personal computers. *CABIOS* 12:357-8.

Perrière, G. and Gouy, M. 1996. WWW-Query: An on-line retrieval system for biological sequence banks. *Biochimie* 78:364-369.

Strimmer, K. and von Haeseler, A. 1996. Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.* 13:964-969.

Thompson, J.D., Gibson, T.J., Plewniak, F., Jeanmougin, F., and Higgins, D.G. 1997. The ClustalX windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res.* 25:4876-4882.

Zmasek, C.M. and Eddy, S.R. 2001. ATV: Display and manipulation of annotated phylogenetic trees. *Bioinformatics* 17:383-384.


## Key References


Page, 1996. See above.

*A short description of the program, together with example screenshots. Since this article was written, TreeView has been ported to the Unix operating system (http://darwin.zoology.gla.ac.uk/~rpage/treeviewx/).*


## Internet Resources

http://taxonomy.zoology.gla.ac.uk/rod/treeview.html

*TreeView Web site.*

http://darwin.zoology.gla.ac.uk/~rpage/treeviewx/

*TreeView X Web site. This program is a Unix port of TreeView.*

http://evolution.genetics.washington.edu/phylip/software.html#Plotting

*Joe Felsenstein's list of tree-drawing programs.*

http://rdp.cme.msu.edu/docs/treeview_newick.html

*Gary Olsen's description of the Newick tree format.*

http://evolution.genetics.washington.edu/phylip/newicktree.html

*Joe Felsenstein's description of the Newick tree format.*

http://evolution.genetics.washington.edu/phylip.html

*PHYLIP (Phylogeny Inference Package), by J. Felsenstein.*

http://paup.csit.fsu.edu/

*PAUP*, Phylogenetic Analysis Using Parsimony (*and Other Methods), by D.L. Swofford. Version 4. Sinauer Associates. Sunderland, Mass.*

---


Contributed by Roderic D.M. Page
University of Glasgow
Glasgow, Scotland

# Getting a Tree Fast: Neighbor Joining, FastME, and Distance-Based Methods

Distance methods, and especially Neighbor Joining (NJ; Saitou and Nei, 1987), are popular methods for reconstructing phylogenies from alignments of DNA or protein sequences (*UNIT 2.3*). They are fast, allowing hundreds and even thousands of taxa to be dealt with by ordinary computers. The speed of these methods greatly simplifies the use of the bootstrap procedure (Page and Holmes, 1998; Graur and Li, 2000), which assesses the confidence level of inferred clades. They provide a simple way to incorporate knowledge of the evolution of the sequences being studied, depending on how the distance matrix is estimated. Numerous simulation studies have demonstrated their topological accuracy, and, unlike parsimony methods, they are not hampered by inconsistency (or "Felsenstein zone"; Swofford et al., 1996). The popularity of NJ, among the numerous existing distance-based methods, is explained by its speed and by the fact that its topological accuracy remains relatively close to that of recent approaches—i.e., FITCH (Felsenstein, 1997), BIONJ (Gascuel, 1997a), WEIGHBOR (Bruno et al., 2000), and FastME (Desper and Gascuel, 2002, 2004). However, several simulation studies (e.g., Vinh and Von Haeseler, 2005) showed that, with a high number of taxa, NJ is outperformed by FastME, both in terms of computing time and topological accuracy. Therefore, this latter program should be considered preferable for large-scale studies.

NJ and other current distance methods do not assume a molecular clock (Page and Holmes, 1998), as opposed to the Unweighted Pair Group Method Using Arithmetic averages (UPGMA; Sokal and Michener, 1958), which is precluded for most phylogenetic studies. The basic assumption is that sequences have been evolving along a tree and independently among the lineages. This tree can differ from the species tree in cases of horizontal transfer or sequence duplication (*UNIT 6.1*). Other assumptions are related to the sequence evolution model used to estimate distances. Models applicable to distance methods are homogeneous (i.e., constant over time) and assume that each site in the sequence evolves independently. However, some model parameters can differ from site to site. For example, mutation rates can vary across sites to represent structural/functional constraints on the residues, or the fast rate of the third codon position.

Distance methods are thus "model based," just like maximum-likelihood methods (see Swofford et al., 1996, for discussion of these methods and comparison between them). However, the way the computations are performed is simpler and more approximate. Consequently, distance methods are faster than maximum-likelihood methods, but do not achieve the same topological accuracy. The comparison with parsimony is more complicated, since parsimony is sometimes inconsistent, but accurate when no long (e.g., outgroup) branch tends to attract other branches and perturb the resulting tree. A good practical approach is then to avoid parsimony when long branch attraction is suspected and otherwise to run both parsimony and distance approaches and compare the results.

Application of any distance-based method usually requires the following steps (see Fig. 6.3.1).

a. Choose a sequence evolution model and use it to estimate the distance matrix (Support Protocols 1 and 2).
b. Run the tree-building algorithm (Basic Protocol or Alternate Protocols 1, 2, or 3) and eventually return to step (a), for example to check that the resulting tree is

**Contributed by Richard Desper and Olivier Gascuel**
*Current Protocols in Bioinformatics* (2006) 6.3.1-6.3.28

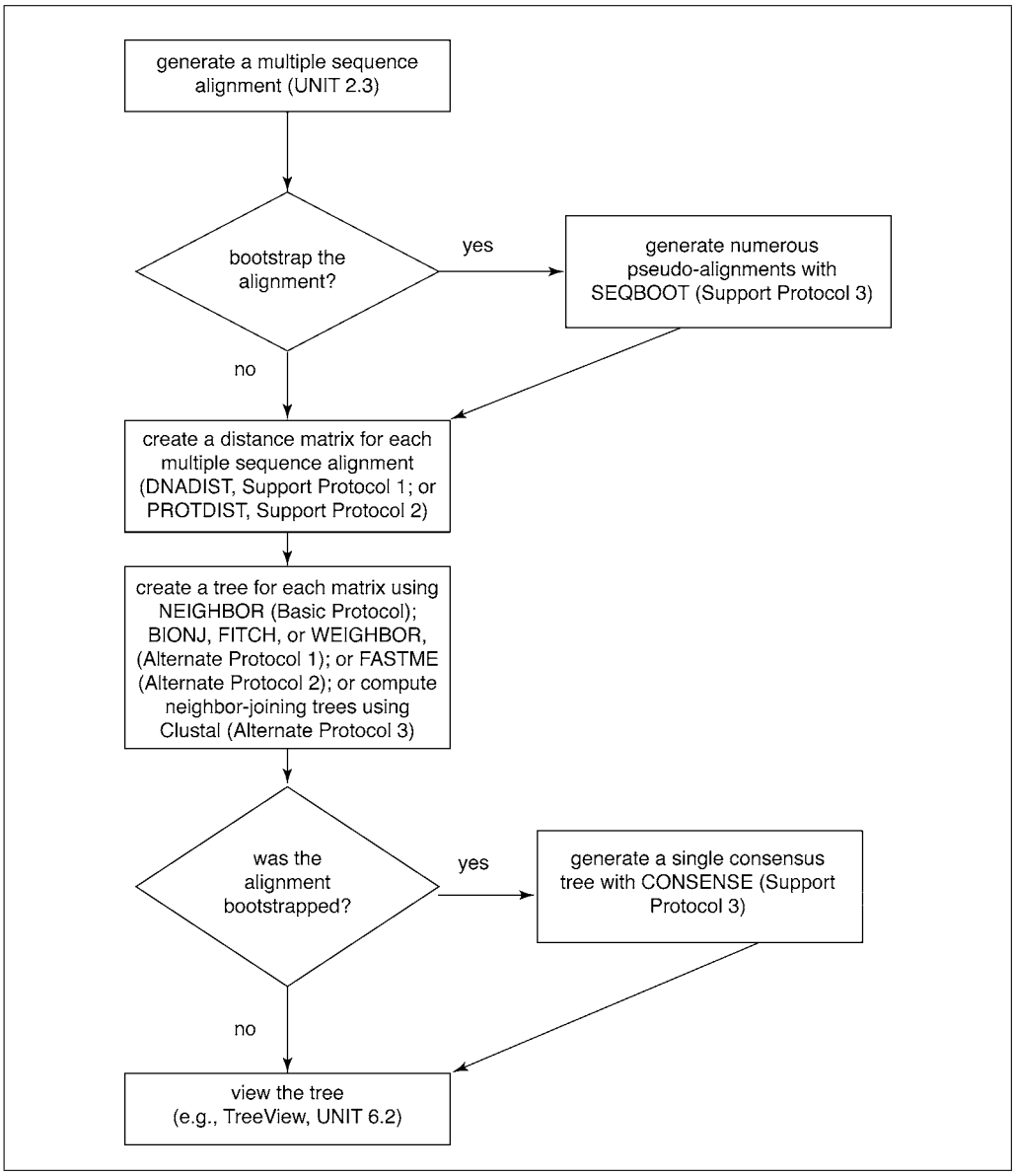


**Figure 6.3.1** Flowchart illustrating the relationship between the multiple protocols presented in this unit.

not too sensitive to the model parameter values. The influence of taxon sampling, notably the presence/absence of the outgroup taxa, also has to be checked.

c. Perform the bootstrap procedure to assess the significance level of the inferred clades (Support Protocol 3).

***BASIC PROTOCOL***

## USING THE NEIGHBOR PROGRAM FROM THE PHYLIP PACKAGE TO CONSTRUCT A PHYLOGENETIC TREE

This protocol describes the use of NEIGHBOR (see Fig. 6.3.1), included in the PHYLIP 3.6 package (latest 3.65 version is identical; Felsenstein, 1989), which is distributed by Joe Felsenstein (University of Washington) and is one of the most widely used software packages in phylogeny studies. NEIGHBOR is the PHYLIP implementation of Neighbor Joining (Saitou and Nei, 1987). Distance estimation is performed using DNADIST or PROTDIST (Support Protocols 1 and 2). To accomplish the bootstrap procedure, first resample the sites using SEQBOOT (Support Protocol 3), then apply DNADIST or PROTDIST, run NEIGHBOR, and extract the bootstrap tree using CONSENSE (Support

Protocol 3). Finally, the resulting tree can be drawn using a program such as TreeView (*UNIT 6.2*) or NJplot (Perrière and Gouy, 1996).

***Necessary Resources***

*Hardware*

PHYLIP executables are available for Windows, Mac OS 9 and OS X, and Linux. The PHYLIP C source code is also available for Unix, Linux, or OpenVMS systems.

*Software*

PHYLIP is available for free from *http://evolution.genetics.washington.edu/phylip.html*. The package contains C source codes, documentation files, and a number of different types of executables. Its Web page contains information on PHYLIP and ways to transfer the executables, source code, and documentation. The documentation is remarkably clear and complete, and provides a number of useful references.

*Files*

NEIGHBOR requires a distance matrix (or a set of distance matrices when the bootstrap procedure is used), which is estimated by DNADIST (Support Protocol 1) or PROTDIST (Support Protocol 2) from a multiple sequence alignment (e.g., *UNIT 2.3*). The file contains a number of taxa on its first line. Each taxon starts a new line with the taxon name, followed by the distance to the other taxa, and there is a new line after every nine distances. Taxon names have ten characters and must be blank-filled to be of that length. The default matrix format is square (Fig. 6.3.2) with zero distances on the diagonal. In the case of multiple matrices, as obtained with the bootstrap, matrices are given in the same format one after the other, without omitting the number of taxa at the beginning of each new matrix.

1. Download and install PHYLIP according to the program documentation (see Necessary Resources, above).
2. Generate a distance matrix for the multiple sequence alignment of interest by running either DNADIST (for DNA sequence alignments; see Support Protocol 1) or PROTDIST (for protein sequence alignments; see Support Protocol 2).
3. Begin a NEIGHBOR session in PHYLIP by double clicking on its icon.
4. At the prompt, enter the distance matrix file name and the name for the outfile, which will contain a simple representation of the output tree. The default files are `infile` and `outfile`, respectively, but the authors strongly recommend redefining these files to avoid possible confusions or deleting previously computed files.

```
    8
Candida_al  0.0000  0.0939  0.0224  0.1737  0.1632  0.2507  0.2757  0.3050
Saccharomy  0.0939  0.0000  0.0966  0.1434  0.1582  0.2381  0.2064  0.2614
Candida_tr  0.0224  0.0966  0.0000  0.1791  0.1632  0.2591  0.2855  0.3160
Protomyces  0.1737  0.1434  0.1791  0.0000  0.0259  0.2235  0.2232  0.2820
Taphrina_d  0.1632  0.1582  0.1632  0.0259  0.0000  0.2585  0.2581  0.3318
Filobasidi  0.2507  0.2381  0.2591  0.2235  0.2585  0.0000  0.1386  0.1370
Spongipell  0.2757  0.2064  0.2855  0.2232  0.2581  0.1386  0.0000  0.0791
Athelia_bo  0.3050  0.2614  0.3160  0.2820  0.3318  0.1370  0.0791  0.0000
```

**Figure 6.3.2** Distance matrix in square format.

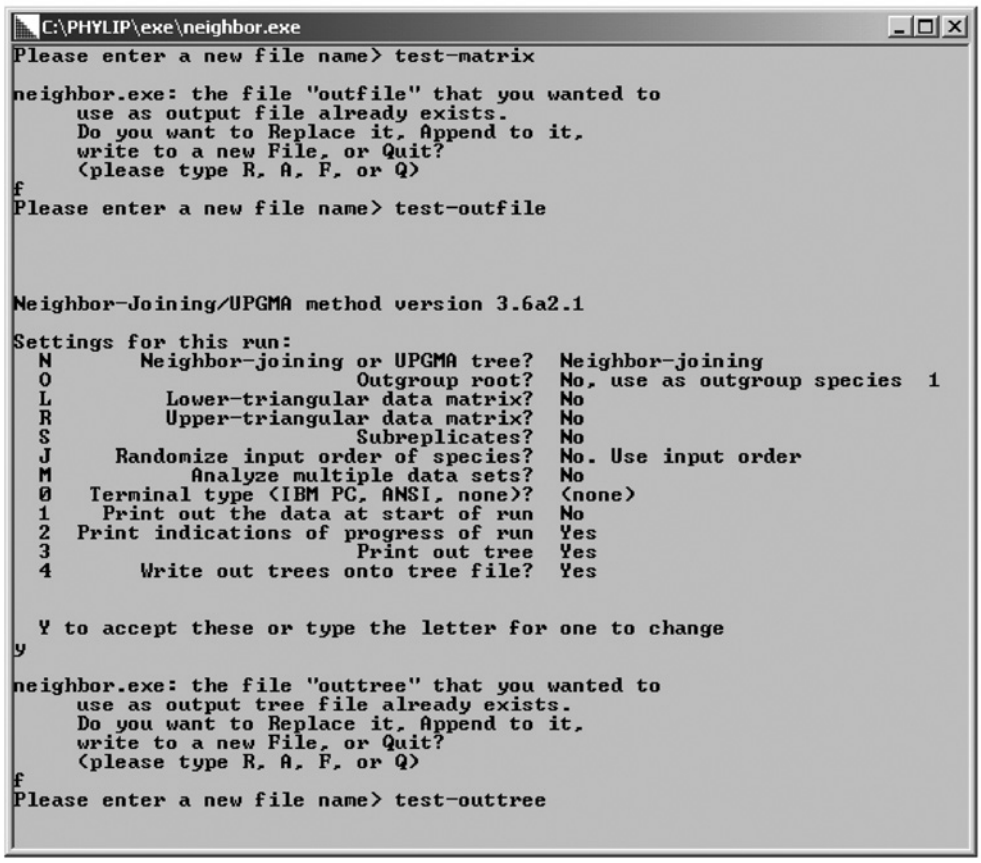


**Figure 6.3.3** The NEIGHBOR screen showing options for renaming files as well as options for settings and their defaults.

*When a file called* `infile` *already exists in the PHYLIP directory, NEIGHBOR does not ask for the input file and reads the existing* `infile`*. Similarly, the option of renaming the output is only given if a file called* `outfile` *already exists. If no such file exists, NEIGHBOR automatically writes the output to a file called* `outfile`*.*

5. After entering the file information, select among several options (see Fig. 6.3.3), which, a priori, have to be used with their default values, except **M** in the case of the bootstrap procedure. When options have been determined, type `Y` to run NEIGHBOR.

*These options are as follows. **N** defines the method to be used; NJ (default option) is preferred over UPGMA, which assumes a molecular clock. **O** makes it possible to specify which species is to be used to root the tree; when **O** is on, the user is asked for the rank of the outgroup species in the input (matrix) file, otherwise the default outgroup species is the first; this outgroup (rooting) species is used in the tree printed in the outfile. **L** and **R** have to be switched on when the matrix is not square but lower-triangular and upper-triangular, respectively. **S** has to be on when the data contain subreplicates; it allows NEIGHBOR to read the input data, but the number of replicates is ignored. **J** enables one to choose a random order of species; the user is then asked for a "seed"; however, NEIGHBOR is almost insensitive to species ordering. **M** has to be used in the case of the bootstrap procedure (Support Protocol 3) to provide the number of pseudo-matrices. **0** defines the terminal type; this may affect the ability of the programs to display their menus and results, but the* `none` *option is usually sufficient. The **1** and **2** options are used to check the data and the progress of run; the authors suggest switching them off, notably for large trees and bootstrap studies. When **3** is Yes (default value), the tree or trees are printed in the* `outfile`*; this is useful to quickly visualize trees with moderate*

```
(((Candida_tr:0.0137,Candida_al:0.0086):0.0526,Saccharomy:0.0316):0.0351,
 (Taphrina_d:0.0160,Protomyces:0.0098):0.0665,
 ((Athelia_bo:0.0600,Spongipell:0.0190):0.0480,Filobasidi:0.0612):0.0964);

(Candida_tr:0.01367,(Saccharomy:0.03307,((Protomyces:0.00957,
Taphrina_d:0.01633):0.06809,(Filobasidi:0.05464,(Spongipell:0.01908,
Athelia_bo:0.06002):0.04361):0.10745):0.03164):0.05098,Candida_al:0.00873);
```

**Figure 6.3.4** Two trees in Newick format, which were obtained from the distance matrix in Figure 6.3.2 by BIONJ and NEIGHBOR, respectively. Both trees have identical topologies, but slightly different branch lengths.

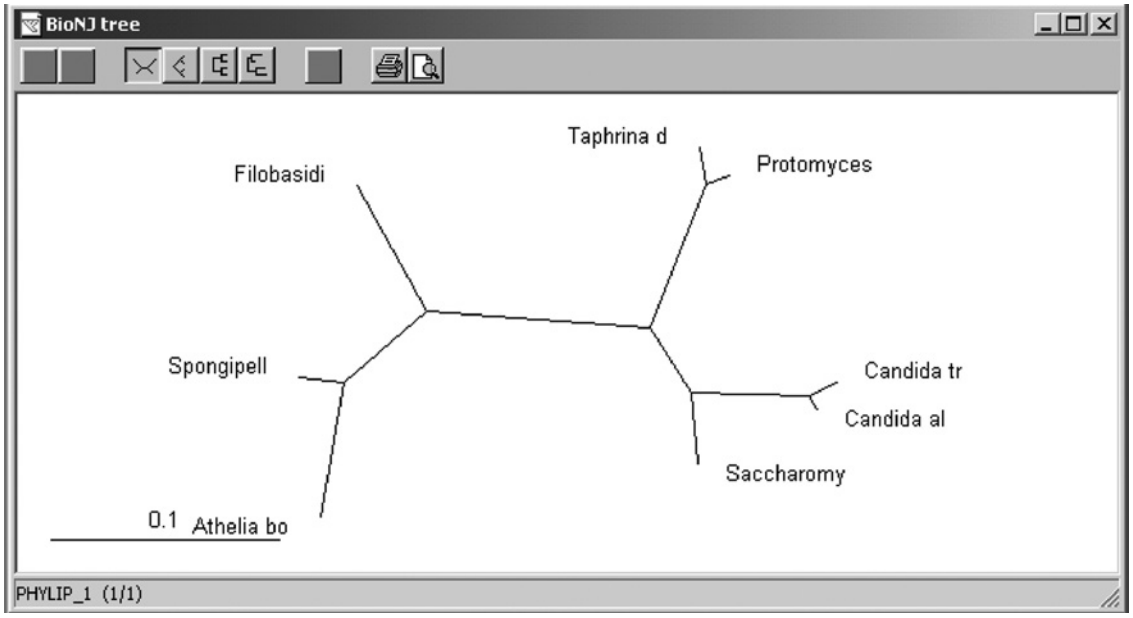


**Figure 6.3.5** TreeView representation of the BIONJ tree of Figure 6.3.4.

*numbers of taxa. When **4** is Yes (default value), the trees are written in Newick format in the* `outtree` *file, and can then be drawn using TreeView (UNIT 6.2) or, in case of multiple data sets, combined by CONSENSE to obtain the bootstrap tree (Support Protocol 3). To change the default values, simply type the option character. For example, typing* `2` *changes the progress of run status from* `Yes` *to* `No`*, and typing* `2` *again returns one to* `Yes`*.*

6. Finally, NEIGHBOR asks for the `outtree` file, which will contain the tree in Newick format (*UNIT 6.2*). The resulting tree can be visualized in the `outfile`, but a better view is obtained by applying TreeView (*UNIT 6.2*) to the `outtree` file.

   *The option of renaming the* `outtree` *file is only given if a file called* `outtree` *already exists. If no such file exists, NEIGHBOR automatically writes the output to a file called* `outtree`*, which may be a source of confusion. Inferred trees are unrooted and written in Newick format (UNIT 6.2). For example, the BIONJ tree in Figure 6.3.4 is made of three subtrees, containing (*`Candida_tr`*,* `Candida_al`*, and* `Saccharomy`*), (*`Taphrina_d` *and* `Protomyces`*) and (*`Athelia_bo`*,* `Spongipell`*, and* `Filobasidi`*), respectively, as can be shown from its TreeView representation (Fig. 6.3.5; see UNIT 6.2 for discussion of TreeView and Newick). Each subtree is made up of two subtrees or taxa; the numbers in Figure 6.3.4 indicate the branch lengths. Both trees in Figure 6.3.4 have identical topologies (even when the way they are encoded in Newick format looks quite different) but (slightly) different branch lengths.*

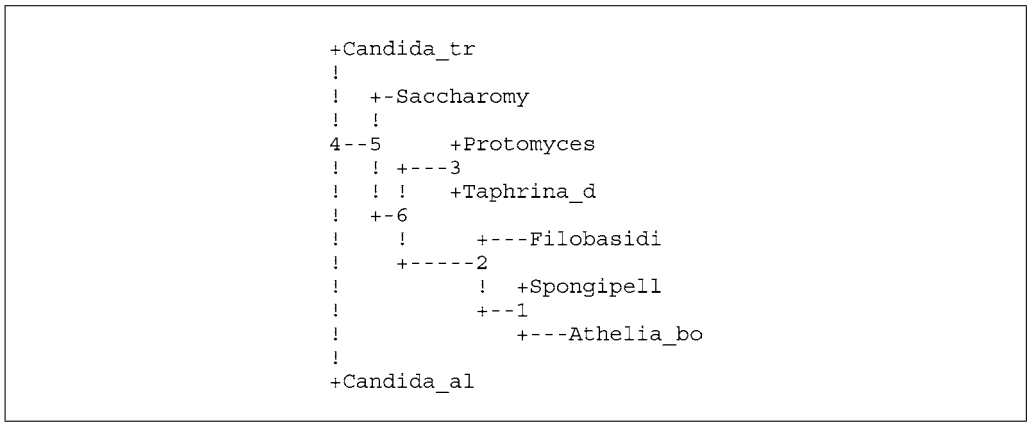


**Figure 6.3.6** NEIGHBOR tree, as represented in the `outfile`.

*Applying NEIGHBOR to the matrix of Figure 6.3.2, one obtains in the* `outfile` *the tree shown in Figure 6.3.6, while in the* `outtree` *file we have the second tree from Figure 6.3.4, in Newick format. This tree is equivalent to that of Figure 6.3.5.*

7. To assess the tree quality, bootstrap the tree according to Support Protocol 3.

***SUPPORT PROTOCOL 1***

## DISTANCE MATRIX ESTIMATION FROM DNA (OR RNA) SEQUENCES USING DNADIST

Distance estimation is the first step in reconstructing a phylogenetic tree using a distance-based method. DNADIST, from the PHYLIP package, estimates the pairwise evolutionary distances between nucleotide sequences under various models of nucleotide substitutions. These models account for hidden substitutions and incorporate knowledge about the mutation process. Distance estimation is based on the maximum-likelihood principle (Swofford et al., 1996). The model choice is sensitive and influences the distance values, and then the tree to be constructed. DNADIST reads a multiple sequence alignment and outputs a distance matrix. When the bootstrap procedure is used, the input file contains the pseudo-alignments one after the other, and the output file contains the corresponding pseudo-matrices in the same order.

### *Necessary Resources*

*Hardware*

PHYLIP executables are available for Windows, Mac OS 9 and OS X, and Linux. The PHYLIP C source code is also available for Unix, Linux, or OpenVMS systems.

*Software*

DNADIST is part of the PHYLIP package. PHYLIP is available for free from *http://evolution.genetics.washington.edu/phylip.html*. The package contains C source codes, documentation files, and a number of different types of executables. Its Web page contains information on PHYLIP and ways to transfer the executables, source code, and documentation. The documentation is remarkably clear and complete, and provides a number of useful references.

*Files*

DNADIST requires DNA multiple sequence alignments in PHYLIP format, as obtained from alignment programs such as ClustalX (*UNIT 2.3*). The first line contains the number of taxa and sites; next come the taxon data with a new line per taxon. Taxon names have ten characters and must be blank-filled to be of that

```
8 95
Candida_al    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGTTGTTGTTCTTTTATT-GACGCAAT
Saccharomy    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGGTGGTGTTTTTTTAAT-GACCCACT
Candida_tr    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGTTGTTGTTCTTTTATT-GACGCAAT
Protomyces    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGGCGATGTTCTTTTCTT-GACTCGCC
Taphrina_d    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGGCGATGTTCTTTTCTT-GACTCGCC
Filobasidi    AGTCTTAACAGTAAACGATGCCGACTAGGGATCGGCCCACGTCAATCTCT--GACTGGGT
Spongipell    AGTCTTAACAGTAAACTATGCCGACTAGGGATCGGGCGATCTCAAACTT-ATGTGTCGCT
Athelia_bo    AGTCTTAACAGTAAACTATGCCGACTAGGGATCGGACAACCTCAATTTTGATGTGTTGTT

CGGCACCTTACGAGAAATCA-AAGTCTTTGGGCCC
CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
CGGCACCTTACGAGAAATCA-AAGTCTTTGGGGG?
CGGCACCTTATGAGAAAGGA-AAGTTTTTGGGTTC
CGGCACCTTATGAGAAAAA???????????????
CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
```

**Figure 6.3.7** Alignment in interleaved PHYLIP format.

```
8 95
Candida_al    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGTTGTTGTTCTTTTATT-GACGCAAT
              CGGCACCTTACGAGAAATCA-AAGTCTTTGGGCCC
Saccharomy    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGGTGGTGTTTTTTTAAT-GACCCACT
              CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
Candida_tr    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGTTGTTGTTCTTTTATT-GACGCAAT
              CGGCACCTTACGAGAAATCA-AAGTCTTTGGGGG?
Protomyces    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGGCGATGTTCTTTTCTT-GACTCGCC
              CGGCACCTTATGAGAAAGGA-AAGTTTTTGGGTTC
Taphrina_d    AGTCTTAACCATAAACTATGCCGACTAGGGATCGGGCGATGTTCTTTTCTT-GACTCGCC
              CGGCACCTTATGAGAAAAA???????????????
Filobasidi    AGTCTTAACAGTAAACGATGCCGACTAGGGATCGGCCCACGTCAATCTCT--GACTGGGT
              CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
Spongipell    AGTCTTAACAGTAAACTATGCCGACTAGGGATCGGGCGATCTCAAACTT-ATGTGTCGCT
              CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
Athelia_bo    AGTCTTAACAGTAAACTATGCCGACTAGGGATCGGACAACCTCAATTTTGATGTGTTGTT
              CGGCACCTTACGAGAAATCA-AAGTCTTTGGGTTC
```

**Figure 6.3.8** Alignment in sequential PHYLIP format.

length. The taxon names are followed by the sequences, which must either be "interleaved" or "sequential" (Figs. 6.3.7 and 6.3.8). The sequences can have internal blanks in the sequence but there must be no extra blanks at the end of the terminated line. The three symbols `N`, `X` and `?` indicate an unknown nucleotide while a dash (–) indicates a deletion. In the case of multiple data sets, as provided by SEQBOOT, pseudo-alignments are given in the same format one after the other, without omitting the number of taxa and the number of sites at the beginning of each new set.

1. Download and install the PHYLIP package, and initialize a DNADIST session by double clicking on its icon.

2. At the prompt, enter the sequence alignment file name and the name for the output, which will contain the distance matrix. The default files are `infile` and `outfile`, respectively, but the authors strongly recommend redefining these files to avoid possible confusion, or deletion of previously computed files.

   *If a file called* `infile` *already exists in the PHYLIP directory, DNADIST does not ask for the input file, but reads the existing* `infile`*. Similarly, the option of renaming the output is only given if a file called* `outfile` *already exists. If no such file exists, DNADIST automatically writes the output to a file called* `outfile`*.*

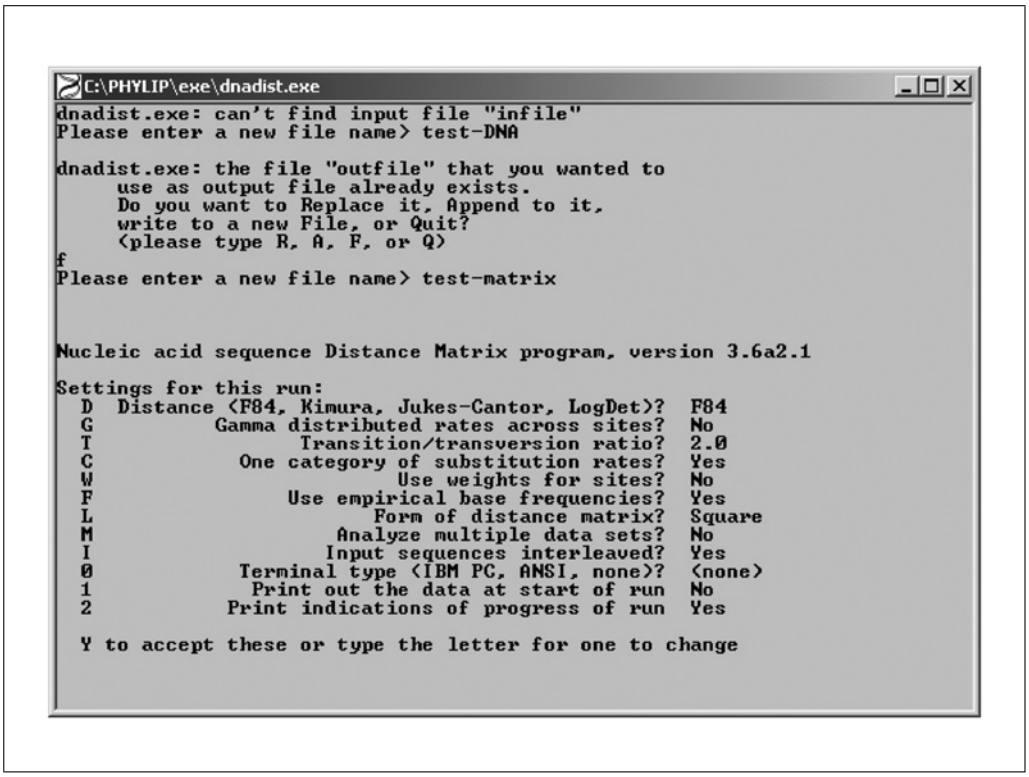


**Figure 6.3.9** The DNADIST screen with options for renaming files and setting parameters. The default parameters are shown.

3. Then the menu of Figure 6.3.9 appears, which asks for important and sensitive choices.

   *The remaining steps of this protocol primarily describe options requiring in-depth explanations or where the default values often need to be changed. More details are given in the DNADIST documentation. To change the default values, simply type the option character. For example, typing* `I` *changes the sequence format from interleaved to sequential, and typing* `I` *again returns to the interleaved format.*

***Set the parameters***

4. **D** defines the substitution model. All models assume that sites evolve independently. The four available models are nested, i.e., Jukes-Cantor is a special case of Kimura, which is a special case of F84, which is a special case of LogDet. Jukes-Cantor (Jukes and Cantor, 1969) assumes only one substitution rate, Kimura (Kimura, 1980) allows for a difference between transition and transversion rates, while F84 (Kishino and Hasegawa, 1989; Felsenstein and Churchill, 1996) is similar to Kimura but allows for different frequencies of the four nucleotides, and LogDet does not impose any restriction on the 16 rates (except those induced by the Markovian nature of the process). So LogDet (Steel, 1994) is the most flexible model, but is often overparametrized, unless the sequences are very long (say $>3000$). F84 (the default option) is a good compromise, notably when the base frequencies are not equal. When they are almost equal, Kimura is a good choice, while Jukes-Cantor is overly simple in most cases.

   *Note that all sites (informative or not) must be given to DNADIST for these models to be used in the correct way.*

5. **G** asks whether or not the substitution rates vary across sites. Biologically speaking, the answer is clearly yes. It has been demonstrated that the Gamma distribution (Swofford et al., 1996), which is defined by a parameter usually denoted as $\alpha$, is a

good model to account for this variability. $\alpha$ was estimated between 0.05 and 1.0 for numerous data sets (Yang, 1966), which indicates that rates strongly vary across sites (variability increases as $\alpha$ decreases). However, the default option of DNADIST is to not correct for this variability (i.e., $\alpha = \infty$), which is a common practice.

*Jin and Nei (1990) recommend using $\alpha$ = 1.0 or 2.0. The authors have recently demonstrated (Guindon and Gascuel, 2002) that uncorrected distances are often better suited, especially when the molecular clock is more or less satisfied. Therefore, a pragmatic approach is to use the default option, and to check whether or not using a reasonable value (e.g., 1.0 or 2.0) for $\alpha$ changes the result.*

*However, DNADIST does not use the standard $\alpha$ parameter, but rather the "coefficient of variation" (CV), which is equal to $1/\alpha^{1/2}$. One obtains CV = 2.0, 1.0, and 0.5 when $\alpha$ = 0.25, 1.0, and 4.0, respectively. Moreover, the LogDet model cannot be combined with the gamma correction.*

6. **T** asks for the transition/transversion ratio. The default value is 2.0, and there is no way to estimate this value within PHYLIP.

   *Hopefully, the results are not very sensitive to the value of this parameter (unless it is extreme). It is possible to estimate it using simple formulas from Kimura (1980).*

7. **C** allows user-defined categories, for example to specify that third-position bases have a different rate than first and second positions. This option allows the user to make up to 9 categories of sites, but, as with the LogDet model, using too many categories can make the model overparametrized. The user is asked for the relative rates within each category. The assignment of rates to sites is then made by reading a file whose default name is `categories`.

   *An example and more details are given in the DNADIST documentation. There is no program from PHYLIP for estimating the different rates, but just as with the above ratio, these parameters are not very sensitive (unless extreme).*

8. **W** allows the user to select subsets of sites. Basically it has to remain "No" (the default value), unless the user wants to check the influence of various categories of sites.

   *See DNADIST documentation for more details.*

9. **F** must remain as Yes in any practical situation.

10. **L** defines the matrix format, square (default value) or lower-triangular.

11. **M** has to be used in the bootstrap procedure (see Support Protocol 3). The user is then asked for the number of pseudo-alignments in the input file. Otherwise the default value (No) is required.

12. **I** defines the multiple sequence alignment format, which is interleaved or sequential (Fig. 6.3.7 and 6.3.8, respectively).

13. Once all options have been determined, type `Y` to compute the distance matrix.

    *With the working example of Figure 6.3.7 and all default values, DNADIST returns the matrix of Figure 6.3.2.*

## DISTANCE MATRIX ESTIMATION FROM PROTEINS USING PROTDIST

***SUPPORT PROTOCOL 2***

PROTDIST is analogous to DNADIST (Support Protocol 1). It is first necessary to provide the file names, which the program initially assumes to be `infile` and `outfile` (see Fig. 6.3.10). The `infile` contains the protein multiple sequence alignment (*UNIT 2.3*). The format is analogous to that used with nucleotide sequences (Support Protocol 1), except that, with proteins, the three symbols `X`, `-`, and `?` indicate an unknown amino acid,

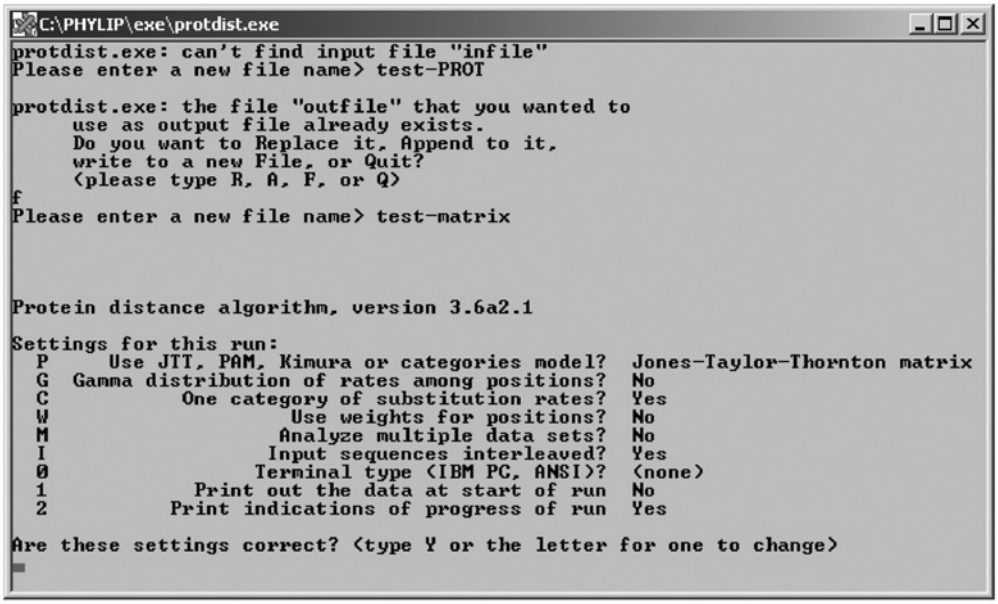


**Figure 6.3.10** The PROTDIST screen showing options for renaming files and setting parameters. The default parameter settings are shown.

a deletion, and an unknown including deletion, respectively (see PHYLIP documentation `sequence.html` for more details). The final distance matrix is written to `outfile`, unless the user selects a different name. After providing the file names, the user then deals with the options (see Fig. 6.3.10). The main option is **P**, which selects among five substitution models differing depending on the matrix of substitution rates. The substitution models are as follows:

*Dayhoff PAM 001 matrix*. This matrix (Dayhoff et al., 1979) is an empirical one that scales probabilities of change from one amino acid to another, assuming that the total change between the two amino acid sequences is 1% (*UNIT 3.5*). It allows the evolutionary distance to be computed in terms of expected fraction of amino acids changed.

*PMB (Probability Matrix from Blocks)*. This model is derived (Veerassamy et al., 2003) using the Blocks database of conserved protein motifs and is a continuation of BLOSUM scoring matrices, which are widely used for protein sequence alignments (*UNIT 3.5*). Note that this model is only available in the latest PHYLIP version (3.65).

*Jones-Taylor-Thornton model*. This model (Jones et al., 1992) is analogous to PAM, but the estimation of the probabilities of change is based on a much larger set of proteins. Thus it is to be preferred over the original PAM.

*Kimura's distance*. This model (Kimura, 1983) assumes only one substitution rate, and does not take into account which amino acids differ.

*The Categories distance*. This model, devised by Joe Felsenstein, is conceptually close to Kimura's two-parameter model for DNA sequences (Kimura, 1980). The amino acids are grouped into a series of categories, and a distinction is made between transitions (change within a category) and transversions (change from one category to another). When this option is selected, the user is asked for a number of other options (e.g., the amino acid categorization), but the authors suggest using default values that approximate the PAM model (*UNIT 3.5*).

As already stated, the Jones-Taylor-Thornton model is preferred over PAM in any situation, while PMB seems to be an interesting (but new) option. These three models, however, require heavy computation, and the same holds for the Categories model. The Kimura model is therefore a good option for large data sets or atypical (e.g., membrane) proteins. For the other options, see comments on DNADIST in Support Protocol 1.

## BOOTSTRAPPING USING SEQBOOT AND CONSENSE

***SUPPORT PROTOCOL 3***

A tree such as that shown in Figure 6.3.5 does not indicate the reliability of the inferred clades. The bootstrap procedure is a sound and accurate way to obtain this information, and its use is greatly facilitated by the speed of distance methods. Within PHYLIP, the bootstrap procedure is achieved as shown in the flowchart of Figure 6.3.1. It successively uses: (1) SEQBOOT, (2) DNADIST or PROTDIST, (3) NEIGHBOR (or any other distance method, see Alternate Protocols 1 and 2), and (4) CONSENSE.

### *Necessary Resources*

*Hardware*

PHYLIP executables are available for pre-386 DOS, 386/486/Pentium DOS, Windows 3.1, Windows 95/98/NT, 68k Macintosh, or PowerMac. The PHYLIP C source code is also available for Unix, Linux, or VMS systems.

*Software*

SEQBOOT and CONSENSE are part of the PHYLIP Package. PHYLIP is available for free from *http://evolution.genetics.washington.edu/phylip.html*. The package contains C source codes, documentation files, and a number of different types of executables. Its Web page contains information on PHYLIP and ways to transfer the executables, source code, and documentation. The documentation is remarkably clear and complete, and provides a number of useful references.

*Files*

SEQBOOT requires a multiple sequence alignment in the PHYLIP format, as obtained from alignment programs, such as ClustalX (*UNIT 2.3*); it computes pseudo-alignments by sampling at random with replacement the sites in the original (input) alignment, and outputs these pseudo-alignments in the PHYLIP format. Pseudo-alignments are processed by DNADIST or PROTDIST (Support Protocols 1 and 2) and transformed into pseudo-matrices, which are written in the PHYLIP format. The pseudo-matrix file is then used by NEIGHBOR to build pseudo-trees, written in Newick format (Basic Protocol and *UNIT 6.2*). Finally, the pseudo-tree file is used in CONSENSE to obtain the bootstrap tree, also written in Newick format.

1. After downloading and installing PHYLIP, start a SEQBOOT session by doing double clicking on its icon.

2. Create pseudo-alignments from the aligned sequences using SEQBOOT.

   *The SEQBOOT screen is illustrated along with its options in Figure 6.3.11. To obtain more reliable results, the **R** option, which corresponds to the number of replicates, has to be changed from 100 (the default value) to 1000 (or more in large studies). SEQBOOT allows for site categories and weights (options **W** and **C**, see Support Protocol 1). **F** can be used for large studies to save space on one's system (see SEQBOOT documentation file). **I**, **0**, **1** and **2** have the same meaning as in other PHYLIP programs (see Support Protocol 1); the authors suggest switching **2** to avoid displaying the (extensive and useless)* `progress of run` *on the terminal. The default values have to be conserved for the other options, which correspond to non-sequence data (**D**) or other resampling procedures (**J** and **B**).*

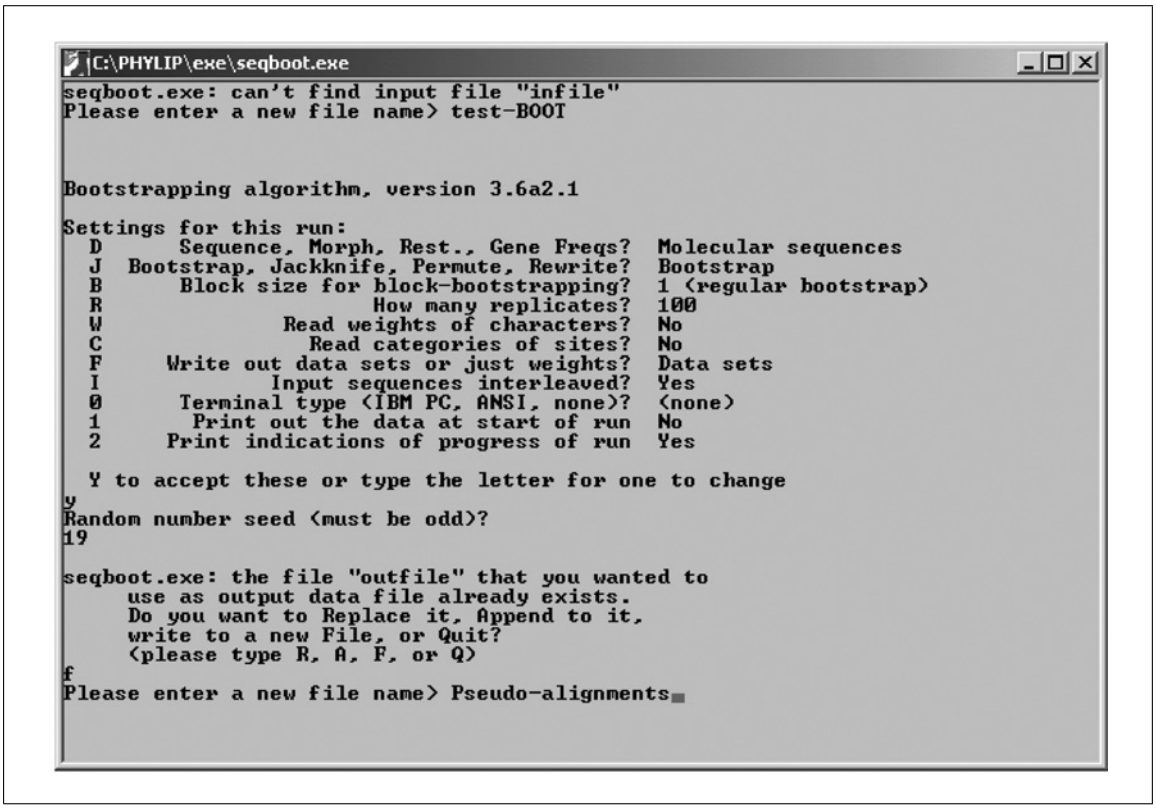


**Figure 6.3.11** The SEQBOOT screen showing options for renaming files and setting parameters. The default parameters are shown.

3. Apply DNADIST (Support Protocol 1) to the pseudo-alignment file to obtain the pseudo-matrices.

   *DNADIST is used as described in Support Protocol 1, except that the number of data sets (replicates) must be given using the **M** option. Switching the **2** option is also relevant.*

4. Apply NEIGHBOR (Basic Protocol) or any other distance method (Alternate Protocols 1 and 2) to the pseudo-matrix file, indicating the number of matrices with the **M** option, and switching the **2** option.

5. Obtain the bootstrap tree by double-clicking CONSENSE and applying it to the pseudo-tree file.

   *The CONSENSE screen is illustrated along with its options in Figure 6.3.12. The default input file name is* `intree`*, while, as for NEIGHBOR (Basic Protocol), the* `outfile` *will contain a simple representation of the bootstrap tree. The **C** option defines the type of consensus method;* `MR` *or* `Mre` *should be selected. The former will provide only clades occurring in more than 50% of the pseudo-trees, while the latter will complete these well supported clades by clades below 50%; only bootstrap supports above 50% have a clear mathematical meaning (Berry and Gascuel, 1996), but lower supports can be informative in some cases. The threshold of clade selection can also be user-defined by selecting* `M1`*. **O** has the same meaning as for NEIGHBOR (Basic Protocol) and can be used to define the outgroup species. **R** has to remain No when using NEIGHBOR and related methods that infer unrooted trees. **T** defines the terminal type, just like **0** in other PHYLIP programs (Basic Protocol). When **1** is on, CONSENSE outputs in* `outfile` *the species list and all clades that belong to at least one of the pseudo-trees. When option **3** is turned off, the* `outfile` *is not created and this cancels (among other things) the previous option. When option **4** is on, the bootstrap tree in Newick format in written in the* `outtree` *file. Finally, switching on the **2** (progress of run) option is relevant.*

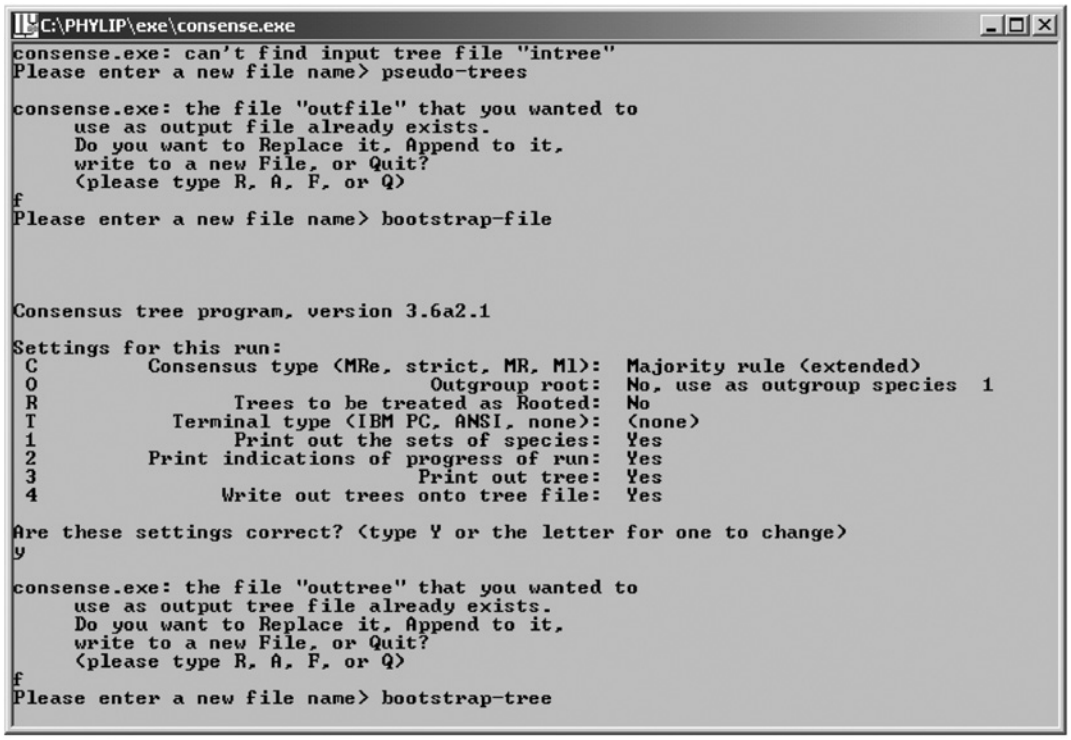


**Figure 6.3.12** The CONSENSE screen showing options for renaming files and setting parameters. The default parameters are shown.

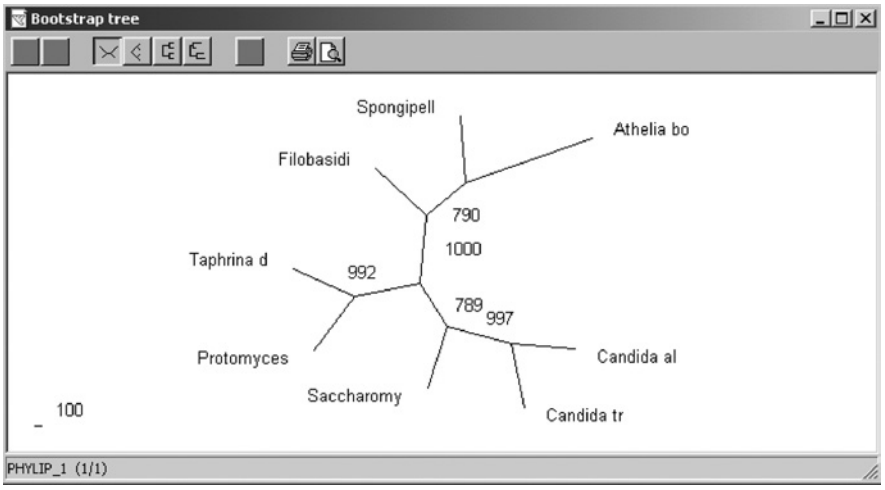


**Figure 6.3.13** TreeView representation of the bootstrap tree that is obtained with NEIGHBOR with 1000 replicates. Bootstrap supports are associated with internal branches (or clades). For example, `Spongipell, Athelia_bo` is supported by 790 pseudo-trees out of 1000.

6. Finally, CONSENSE requests a new name for the `outtree` file. Although it is possible to view the resulting tree in the `outfile`, a better view is obtained by applying TreeView (*UNIT 6.2*) to the `outtree` file.

*When applying these steps (with 1000 replicates) to the original alignment of Figure 6.3.7, the bootstrap tree of Figure 6.3.13 is obtained. The branch lengths correspond to the bootstrap supports, which are explicitly shown in the case of internal branches. Note that due to the random nature of the process, bootstrap supports can differ slightly from one run to another.*

*ALTERNATE PROTOCOL 1*

## USING BIONJ, WEIGHBOR, OR FITCH TO CONSTRUCT A TREE

This protocol provides descriptions of BIONJ and WEIGHBOR, which are PHYLIP-compatible, and FITCH, which is available in PHYLIP. These three programs have a higher topological accuracy than NEIGHBOR. The resulting trees, however, are often similar or identical to NEIGHBOR trees, at least with a low number of taxa (e.g., <20). When this number increases, the various methods tend to return different trees, and their advantage over NEIGHBOR increases. BIONJ is about the same speed as NEIGHBOR; WEIGHBOR is about 500 times slower than NEIGHBOR; and FITCH is even slower than WEIGHBOR (see below for more details). The matrix distance computation (Support Protocols 1 and 2) and the bootstrap procedure (Support Protocol 3), are not described here, since they are performed exactly as with NEIGHBOR (Basic Protocol).

### Using BIONJ

BIONJ is available free from *http://www.lirmm.fr/~w3ifa/MAAS/BIONJ/BIONJ.html*. This Web page contains documentation and articles, test sets, and executables for Windows PC and PowerMac, as well as the C source code. Once downloaded (and compiled on Unix and related systems), BIONJ must be placed in the PHYLIP directory.

BIONJ asks for the distance matrix input file and the name of the tree output file. The distance matrix must be square and written in PHYLIP format. The file can contain one or several matrices, as obtained when using SEQBOOT plus DNADIST or PROTDIST, but the user is not asked for the number of matrices. BIONJ then returns as many trees as there are matrices. These trees are written in Newick format (*UNIT 6.2*). In case of a single matrix, the resulting tree can be viewed using TreeView (*UNIT 6.2*). With multiple matrices and trees, CONSENSE (Support Protocol 3) must be used, just as with NEIGHBOR (Basic Protocol).

Applying BIONJ to the matrix of Figure 6.3.2, the tree of Figure 6.3.4 is obtained, with TreeView representation as shown in Figure 6.3.5. This tree differs from the NEIGHBOR tree (Basic Protocol) by the branch lengths but not by the topology, which is not surprising in view of the low number of taxa.

### Using WEIGHBOR

WEIGHBOR is available in C source code free from *http://www.t10.lanl.gov/billb/weighbor/*. This Web page contains documentation, the seminal article, the C source code, and selected executables for the Windows, Linux, and Solaris operating systems. Then, WEIGHBOR must be placed into the PHYLIP directory.

Just like BIONJ, WEIGHBOR asks for the input and output files, and the input file can contain one or several matrices. WEIGHBOR then asks for the sequence length and the number of symbols, i.e., 4 for DNA or RNA sequences and 20 for proteins. When the input file contains several matrices, WEIGHBOR returns the same number of trees, which must be dealt with by CONSENSE (Support Protocol 3), just as with NEIGHBOR (Basic Protocol).

### Using FITCH

FITCH is available in PHYLIP and runs on numerous systems (see Basic Protocol, Necessary Resources).

FITCH is able to deal with multiple data sets, just as NEIGHBOR, BIONJ, or WEIGHBOR, and its menu is analogous to that of NEIGHBOR (see Basic Protocol). All options must, a priori, conserve their default values, except **G** and **J** which can be used to search the tree space more extensively (at the expense of longer run times). **G** can be switched to `Yes` to search for global rearrangements that improve the least-squares fit of the tree.

**J** takes advantage of the fact that FITCH does not systematically find the same tree, depending on the taxon ordering. When **J** is switched to Yes, FITCH asks for a seed to initiate the random ordering procedure, and then for the number of times the randomization procedure has to be used. The resulting tree is the best tree that is obtained from all random orderings. The higher their number, the better the solution, but the longer the computing time. A value of 10 seems to be a reasonable compromise, but is too high for large data sets, for which the **J** option has to be switched off.

## USING FastME TO CONSTRUCT A TREE

***ALTERNATE PROTOCOL 2***

This protocol provides a description of the phylogeny program FastME, which is PHYLIP-compatible. FastME builds trees with high accuracy, at least as accurately as WEIGHBOR and FITCH in simulations, but it builds the trees much more quickly, even more quickly than NEIGHBOR and BIONJ. This protocol describes FastME version 1.0, which has been fully studied and published (Desper and Gascuel, 2002, 2004), but a new version will be made available soon that handles all of the tasks of distance methods, including calculating distance matrices from sequence data and bootstrapping. FastME 1.0 only builds trees, just like NEIGHBOR. The matrix distance computation (Support Protocols 1 and 2) and the bootstrap procedure (Support Protocol 3) are not described here, since they are performed exactly as with NEIGHBOR (Basic Protocol).

FastME complies with the minimum evolution principle (ME), which involves minimizing the tree length (sum of branch lengths). The general approach is to do a topology search, calculating the tree length for each topology and searching for the topology with shortest tree length. FastME can do the search efficiently by using small topological moves, leading to changes in tree length that can be calculated quickly, without recalculating each individual branch length. It then starts from an initial tree and refines this tree until no more moves improve the current tree.

### *Necessary Resources*

*Hardware*

FastME is written in C, and can be used on any platform that supports a C compiler. Executables are available for Windows, Linux, and Macintosh operating systems. These are command-line executables, but a PHYLIP-type interface is expected to be released soon. Software is available at the FastME home page (*http://atgc.lirmm.fr/fastme/*)

*Software*

The FastME program (*http://www.lirmm.fr/fastme*)

*Files*

See *UNIT 2.3* for the input formats. The input file can contain multiple data sets, if the user chooses, but each matrix should be square (as opposed to upper- or lower-diagonal).

1. FastME is called from the command prompt on all machines. To use FastME, open a command-line shell window (e.g., on a Windows machine by choosing Command Prompt under Accessories in the Start menu, or by choosing Run from the Start menu and typing `cmd` in the text box). Switch to the directory containing the data file(s) and call FastME by typing `fastme` at the command prompt, with the required arguments described below.

*All the options described in the following steps are displayed as part of the online help, which is obtained by the command* `fastme -h`*; see Figure 6.3.14. The bottom line in this figure runs FastME with default options, an input file called* `distfile` *that contains* `1000` *distance matrices, and an output file called* `treefile`*.*

2. Use `-i [name of input file]` to specify the name of the input file, replacing the brackets and text within them with the name of the input file.

3. Use `-b BME`, `-b GME` (or `-b NS`) to specify the method for building the initial tree. BME and GME are the greedy insertion algorithms introduced in (Desper and Gascuel, 2002); BME (`-b BME`) greedily uses balanced least-squares minimum evolution, while GME (`-b GME`) uses ordinary least-squares minimum evolution. NS (`-b NS`) uses the neighbor-joining algorithm described in the Basic Protocol.

   *With n taxa, the fastest option is to use GME, which requires computational time proportional only to $n^2$. BME requires time proportional to $n^2$ diam(T), where diam(T) is the diameter of the tree, i.e., the length (number of branches) of the longest path in T. BME is the default option, being the most accurate, but GME is slightly faster and gives topological accuracy similar to that of BME after post-processing. GME and BME are "greedy," i.e., they iteratively build on partial solutions by optimally inserting the new taxon, keeping the partial tree fixed. NS requires time proportional to $n^3$ and performs the same optimization as BME, over a slightly larger search space.*

4. Alternatively, use `-t [filename of starting tree topology]` to start the topology search from a user-provided starting tree topology, instead of the BME or GME tree.

   *For example, it can be relevant to start with NJ (Basic Protocol) or BIONJ (Alternate Protocol 1) trees, which tend to be more accurate than GME and BME while being relatively fast (but, again, the difference tends to disappear after post-processing).*

5. Use `-n [number of trees/matrices input]` to input the number of data sets to be considered. The default value for this parameter is `1`.

6. Use `-o [filename for tree output]` to specify the name of the output file for the trees.

7. Use `-s` to specify type of nearest-neighbor interchanges (NNIs), or tree swapping, to be used as a post-processing step. FastME can do NNIs to search the tree topology space, while seeking to optimize either the balanced least-squares (`-s BME`) or the ordinary least-squares (`-s OLS`) minimum evolution criterion. It is also possible to forego this step using (`-s none`).

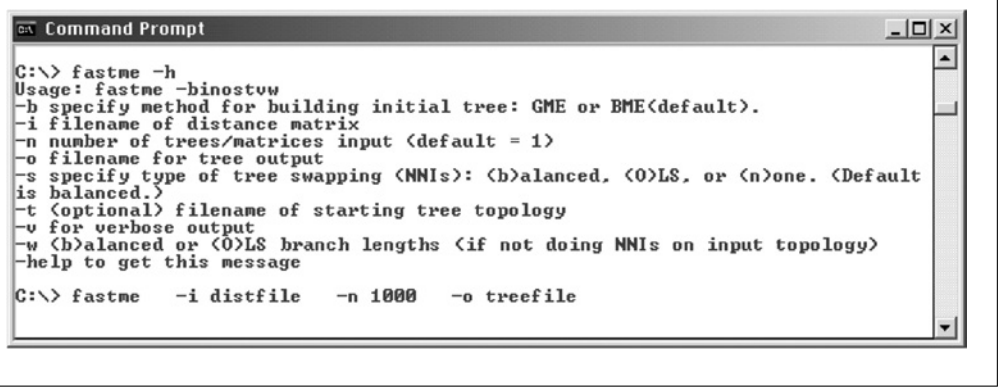


**Figure 6.3.14** FastME screen (in command-prompt window) showing options for renaming files and setting parameters.

*The default is to use balanced least-squares (`-s BME`), and it is recommended to use this setting. The post-processing step is where FastME gets its power. Each NNI requires only time proportional to n, if OLS minimum evolution is used, or proportional to n diam(T) if balanced least-squares minimum is used. The latter option is recommended, as the BLS method has been shown to be considerably more accurate than OLS minimum evolution when biological data sets are considered.*

8. Use `-w b` (for balanced) or `-w O` (for OLS) to select branch lengths to assign to a topology.

   *This option is only needed if `-t` is selected with an input topology, and `-s none` is selected, implying no topology searching is done. In this case FastME just estimates the branch lengths of this topology, according to one of the two approaches. Again, BME (`-w b` on the command line) should be preferred with biological data sets.*

## COMPUTING NJ TREES USING CLUSTAL

***ALTERNATE PROTOCOL 3***

This protocol describes the use of Clustal (see *UNIT 2.3*) to build neighbor-joining (NJ) trees. Although Clustal is not intended primarily as a tree-building program, it is a useful tool for quickly getting a tree for a set of sequences. On the other hand, it does not provide the user with all of the possibilities of PHYLIP, notably concerning distance estimation. The program is available in two versions: ClustalX (Thompson et al., 1997), which has a graphical interface, and ClustalW, which has a text-based interface. ClustalW can be used interactively through a simple menu system or from the command line, which makes it a useful tool for batch processing alignments or generating phylogenies as part of a CGI script. This protocol will provide instructions for both the graphical interface of ClustalX, and the ClustalW command line.

Clustal can output trees in a variety of formats. The default is the Newick format used by many phylogenetic programs (see *UNIT 6.2* and Basic Protocol). Clustal can also write trees in its own format, and can save the pairwise distances in PHYLIP format.

### *Necessary Resources*

*Hardware*

Clustal can be run on Macintosh, Windows, and Unix systems. For full details see *UNIT 2.3*.

*Software*

ClustalX or ClustalW

*Files*

See *UNIT 2.3* for the input formats.

### *Building an NJ tree*

Before building a tree there are various options the user can set that control how the pairwise distances between sequences are computed, and the output format for the tree. In ClustalX these options are set using the commands on the Trees menu (Fig. 6.3.15); in ClustalW they are set on the command line.

1. Install ClustalX or ClustalW and create a multiple sequence alignment (*UNIT 2.3*).
2. Selecting the Exclude positions with gaps option (command-line equivalent in ClustalW, `/TOSSGAPS`) forces Clustal to ignore any site where a gap occurs in any of the sequences when computing pairwise distances (Fig. 6.3.15).

   *A priori, this command should be chosen, because distance estimation from sequences with gaps does not have sound mathematical foundations. However, removing all sites*

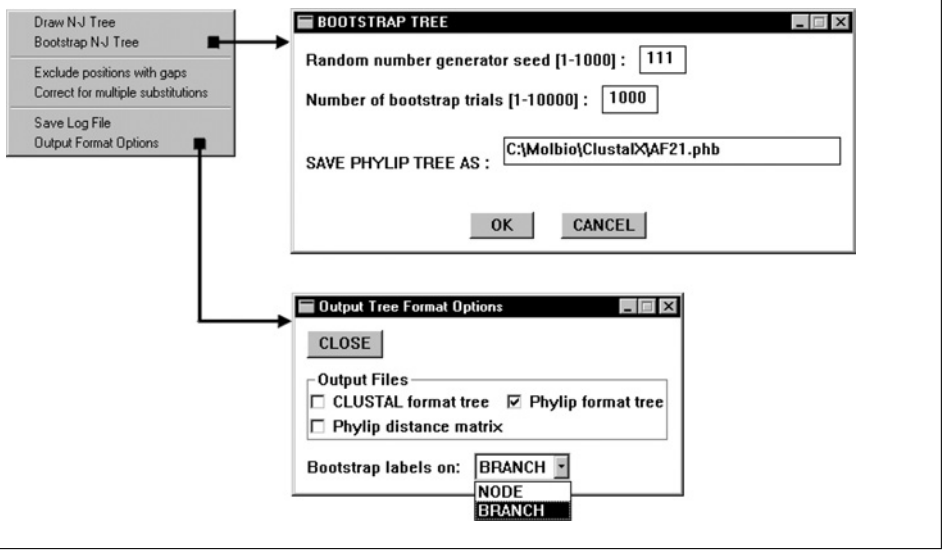


**Figure 6.3.15** The Trees menu in the program ClustalX showing the menu commands and dialog boxes used to control how the program constructs neighbor-joining trees. Note that Exclude Positions with Gaps and Correct for Multiple Substitutions are not selected. If they were selected, a check mark would appear next to each option.

*with a gap sometimes makes the phylogenetic signal so low that the resulting tree is no longer supported in the bootstrap procedure. So both approaches should be tested.*

3. If the user selects the Correct for multiple substitutions option (indicated by a tick beside the menu command; command-line equivalent in ClustalW, `/KIMURA`), then ClustalX will use either the Kimura 2-parameter model (Kimura, 1980) or Kimura's (1983) correction for nucleotides and proteins, respectively, to compute pairwise distances between sequences.

   *This should be the default option. If this option is not chosen, then there is no correction for multiple substitutions.*

4. Select the output format.

   *The menu of options for ClustalX is shown in Figure 6.3.15. The default output tree format is the Newick (or "PHYLIP") format. The command-line equivalent (ClustalW) for this format is:* `/OUTPUTTREE=phylip`. *ClustalX can also write trees in its own format by checking the CLUSTAL format tree box (*`/OUTPUTTREE=nj`*), and can save the pairwise distances in PHYLIP format by checking the PHYLIP distance matrix box (*`/OUTPUTTREE=dist`*).*

5. Having set the options for the analysis and output (or having simply taken the defaults), the command Draw N-J Tree will construct the tree.

   *In ClustalX the user is presented with a dialog box asking for confirmation of the output tree file name. Typically, the tree file is given the name of the user's sequence file plus the extension* `.phb`. *Click on OK to construct the tree. However, the Draw N-J Tree command is somewhat oddly named, as it does not "draw" the tree. To see the tree, you will need to use a tree drawing program such as TreeView (UNIT 6.2). The command-line equivalent for an NJ tree using the default settings is:*

   ```
   clustalw/INFILE=your-aligned-sequence-file/TREE
   ```

   *To use the Kimura correction, and ignore all sites with gaps, the command-line equivalent is:*

   ```
   clustalw/INFILE=your-aligned-sequence-file/TREE/
   KIMURA/TOSSGAPS.
   ```

***The bootstrap procedure***

In addition to the options that affect tree construction above, there are additional options relevant to bootstrapping.

6. Clustal stores the bootstrap values in the tree description inside square brackets, either as branch labels or as node labels.

   *The alternative placements are controlled by the Output Tree Format Options menu (Fig. 6.3.15). As discussed in UNIT 6.2, there is little consensus on how to store bootstrap values in tree descriptions. Widely used programs such as TreeView (UNIT 6.2) do not recognize bootstrap values stored as branch labels, and so in order to display these values in TreeView the bootstrap values must be placed on the nodes (command-line equivalent,* `/BOOTLABELS=node`*).*

7. Having set the options for the bootstrap analysis and output, the command Bootstrap N-J Tree will perform the bootstrapping.

   *In ClustalX the user is presented with a dialog box asking for a "seed" for the random number generator used to create the bootstrap pseudoreplicates, the number of pseudoreplicates ("trials") to generate (the default is 1000), and the name of the file to which the bootstrap tree will be written (typically the tree file is given the name of your sequence file plus the extension* `.phb`*; see Fig. 6.3.15). Click on the OK button to perform the bootstrapping. The command-line equivalent is:* `clustalw/INFILE=your-aligned-sequence-file/BOOTSTRAP`.

## GUIDELINES FOR UNDERSTANDING RESULTS

Phylogenetic trees reconstructed by distance methods do not fundamentally differ from trees reconstructed by any other approach (see *UNIT 6.1*). The main specificity is related to branch lengths. NJ and BIONJ can provide negative branch-length estimates, which have to be seen as null. Such negative values do not indicate any sort of "reverse evolution." Null (or close to zero) branches indicate an irresolution of the tree, which may correspond to a multifurcation, but more likely reflects the weakness of the phylogenetic signal. WEIGHBOR consistently set the negative branches to zero, while FITCH and FastME never provide negative branches as a result of the principles upon which they are based.

The strength of the inferred branches is measured by the bootstrap procedure. Short branches are generally poorly supported, but with distance-based approaches it may happen that long branches also have a low support. The bootstrap procedure must therefore be used, which is done at low computation cost due to the speed of these approaches. The interpretation of bootstrap supports is a difficult question, but any branch with a support lower than 50% should be considered an irresolution (Berry and Gascuel, 1996).

However, in some cases wrong inferences can have high bootstrap support. For example, when very long sequences are used (as is the case when several genes are combined within the same study), bootstrapping the data does not change the resulting tree, which may be partly erroneous. The stability of the tree then has to be tested by other approaches. Notably, the tree must be robust with respect to the presence/absence of the outgroup, which possibly attracts some ingroup taxa, to model parameter variations, and to gene sampling when several genes are combined.

## COMMENTARY

### Background Information

***The rationale of distance-based approaches***

Let $S$ be the set of sequences being studied and $T$ the true evolutionary tree of these sequences. Assume that the sequences have been correctly aligned, so that the sites correspond to homologous positions (see *UNITS 2.1 & 2.4*). Now consider the true number of substitutions that is attached to every branch of $T$, i.e., the number of substitutions that occurred in the past from the sequence situated at one branch extremity to the sequence at the other extremity. These substitution numbers are unknown but well defined. They induce the evolutionary distance between any pair of taxa, as the sum of the substitution numbers attached to the path separating both taxa in $T$. In other words, the evolutionary distance between any pair of taxa is equal to the number of substitutions from one sequence to the other. For mathematical reasons first discovered by Zaretskii (1965), there is an equivalence between the above-defined distance and $T$. Knowing $T$ and the substitution numbers per branch allows the computation of the pairwise distances between taxa. More importantly, the true tree $T$ and the substitution numbers per branch can be reconstructed from the matrix $D$ of pairwise evolutionary distances.

Obviously, and unfortunately, the true number of substitutions that separates any pair of taxa is unknown. Due to hidden (parallel or convergent) mutation events, the true number of substitutions is always greater than or equal to the number of observed differences between both sequences. When the number of differences is small, both quantities are close. However, the gap increases when the evolutionary distance increases. The distance-based approach therefore involves estimating the evolutionary distance from the observed differences, assuming a stochastic model of sequence evolution. The simplest model, that of Jukes and Cantor (1969), supposes that all sites evolve independently and identically according to a Markovian process that is defined by a unique parameter representing the instantaneous probability of change from one nucleotide to another. This model establishes a mathematical relationship between the evolutionary distance (now defined as the ratio between the true number of substitutions and the sequence length) and the proportion of observed differences. More realistic models have been proposed, such as those described above (Support Protocols 1 and 2), but the basic principle remains identical. An estimate $\hat{D}$ of $D$ is first computed, and then an estimate $\hat{T}$ of $T$ is computed using $\hat{D}$. The accuracy of $\hat{T}$ increases with the reliability of $\hat{D}$.

The estimated evolutionary distance matrix $\hat{D}$ no longer exactly fits a tree, but is usually very close to a tree. For example, the working data set of Figure 6.3.7 has been extracted from TreeBASE (*http://www.treebase.org/treebase/index.html*) and corresponds to 67 fungal sequences (accession no. M520). DNADIST and NEIGHBOR with default options construct a tree that explains more than 98% of the variance in the distance matrix (this value was computed by a simple program devised by one of the authors, which is not available from PHYLIP). The resulting tree and the distance matrix are thus extremely close, so the mere principle of the distance approach appears to be well founded in this case (and in most cases).

Even though the estimated distance matrix is usually very close to a tree, tree reconstruction from such an approximate matrix is much less obvious than in the ideal case where the matrix perfectly fits a tree. Various methods have been proposed, which differ according to the criterion they optimize and according to their tree-building strategy. For all known criteria, the optimization task is NP-hard (i.e., can require exponential computing time) so all practical methods are heuristic and do not guarantee that the best tree will be found. However, due to the closeness between the distance matrix and a tree, all (reasonable) methods usually find similar trees that are fairly accurate estimates of the true tree.

***Neighbor-Joining algorithm***

Neighbor Joining (NJ) is derived from ADDTREE (Sattath and Tversky, 1977). It was proposed by Saitou and Nei (1987) and studied in depth by several authors (Studier and Keppler, 1988; Rzhetsky and Nei, 1993; Atteson, 1997; Gascuel, 1997b; Desper and Gascuel, 2005).

NJ is an agglomerative algorithm. At each step, it uses the distance matrix $\hat{D} = \delta_{ij}$ where $i$ and $j$ are either taxa or clusters of taxa agglomerated during previous steps. Based on these distances, two taxa are selected to be merged. Denoting $r$ as the number of "taxa" in $\hat{D}$, and $Q_{ij}$ as the criterion value for the agglomeration

of $i$ and $j$, the pair agglomerated is the one minimizing:

$$Q_{ij} = (r-2)\delta_{ij} - \Delta_i - \Delta_j, \text{ where } \Delta_x = \sum_{y=1}^{r} \delta_{xy}$$

**Equation 6.3.1**

Once the pair $i, j$ to agglomerate is selected, NJ creates a new node $u$ which represents the root of the new cluster. NJ then estimates the branch lengths $\delta_{iu}$ and $\delta_{ju}$ and reduces the distance matrix by replacing the distances relative to taxa $i$ and $j$ by those between the new node $u$ and any other node $x$ using:

$$\delta_{ux} = \frac{1}{2}\left(\delta_{ix} - \delta_{iu}\right) + \frac{1}{2}\left(\delta_{jx} - \delta_{ju}\right)$$

**Equation 6.3.2**

The process stops when $r = 2$, with the last branch length being equal to the last value in the distance matrix. The successive mergings achieved by NEIGHBOR are available in its outfile.

The $Q$ criterion enables numerous interpretations, the most popular being that it corresponds to the least-squares length estimate of the tree under construction. However, this result was fully proven only recently by the authors of this unit in Desper and Gascuel (2005), where they showed that NJ actually minimizes the balanced least-squares tree length estimate, first proposed by Pauplin (2000); see below for further discussion. Accordingly, NJ tends to produce a tree with minimal length. More importantly, when applied to any tree distance $D$ that perfectly fits a tree $T$, $Q$ designates with certainty a pair of neighbors of $T$. This induces the statistical consistency of NJ, which is an essential property of phylogeny reconstruction methods, i.e., NJ recovers the true tree $T$ with certainty, as soon as $\hat{D}$ is sufficiently close to the true evolutionary distance matrix $D$.

### *The BIONJ algorithm*

The BIONJ algorithm (Gascuel, 1997a) is a variant of NJ. It is based on the fact that NJ remains consistent when its reduction formula (see Equation 6.3.2) is replaced by:

$$\delta_{ux} = \lambda_{ij}\left(\delta_{ix} - \delta_{iu}\right) + \left(1 - \lambda_{ij}\right)\left(\delta_{jx} - \delta_{ju}\right)$$

**Equation 6.3.3**

where $\lambda_{ij}$ is any number in [0,1] that varies depending on the merged pair $i, j$ but not on $x$. So once the pair $i, j$ has been selected, BIONJ computes the value $\lambda_{ij}^*$ that minimizes the sum of the variances of the $\delta_{ux}$ estimates. In this way, more reliable estimates will be available to select the pairs of taxa to be agglomerated during the next steps. Moreover, since the process is repeated at each step, these estimates will become better and better in comparison with NJ estimates as the algorithm proceeds.

To achieve this, BIONJ uses a simple first-order model of variances and covariances of evolutionary distance estimates obtained from sequences. This model indicates that the variance of any distance estimate $\delta_{xy}$ is approximately proportional to $\delta_{xy}$, while the covariance of $\delta_{xy}$ and $\delta_{zt}$ is roughly proportional to the length of the intersection of paths $(x,y)$ and $(z,t)$ in the true tree $T$ (Nei and Jin, 1989; Bulmer, 1991). This yields the formula:

$$\lambda_{ij}^* = \frac{1}{2} + \varphi$$

**Equation 6.3.4**

where $\varphi$ is a correction term that depends on $\delta_{iu}$ and $\delta_{ju}$ (at least when $i$ and $j$ are original taxa). When $\delta_{iu}$ and $\delta_{ju}$ are equal, then $\varphi = 0$, $\lambda_{ij}^* = 1/2$, and BIONJ is equivalent to NJ. When both differ, i.e., when the substitution rates vary among lineages, $\varphi$ becomes not null and places more confidence on the shorter and hence more reliable distance. So BIONJ has a clear advantage over NJ when the molecular clock is markedly violated, whereas both methods are close in the opposite case.

### *WEIGHBOR*

WEIGHBOR follows the same agglomerative scheme as NJ. It modifies the reduction step, in a way analogous to BIONJ, but also modifies the selection step to take into account the high variance of long-distance estimates. Instead of using NJ's selection criterion (see Equation 6.3.1), WEIGHBOR combines two criteria. When $i$ and $j$ are neighbors in $T$ and when $\hat{D}$ perfectly fits $T$, then one has the two following properties:

Additivity:

$\delta_{ik} - \delta_{jk}$ is independent of $k\ (\neq i, j)$

Positivity:

$\delta_{ik} + \delta_{jl} - \delta_{ij} - \delta_{kl} \geq 0$ for any $k, l\ (\neq i, j)$

**Equation 6.3.5**

Since $\hat{D}$ is imperfect, these properties are only approximately satisfied, and one has to find the pair $i$ and $j$ that fits them best. To achieve this, WEIGHBOR assumes that distance estimates are mutually independent

and have Gaussian distribution with variance as induced by the Jukes and Cantor (1969) model. Within this model, the variance of the distance estimate is proportional to the distance around 0 (as in the BIONJ model), but increases exponentially when the distance becomes larger. This model allows one to compute the likelihood that $i$ and $j$ are neighbors. Considering the above defined additivity, one has the following criterion (to be minimized):

$$\text{Additivity}(i,j) = \sum_{k\neq i,j} \frac{\left(\delta_{ik} - \delta_{jk} - \overline{\left(\delta_{ik} - \delta_{jk}\right)}\right)^2}{\text{VAR}(\delta_{ik}) + \text{VAR}(\delta_{jk})}$$

**Equation 6.3.6**

where the bar denotes the average over $k$ ($\neq i, j$). A similar criterion corresponds to the positivity property. *Additivity* is used to indicate the best pairs, which are finally selected using *Positivity*. This approach, which fully takes into account the high variance of long evolutionary distances, makes WEIGHBOR more resistant than NJ and BIONJ to the influence (attraction or distraction) of long branches.

### *FITCH*

FITCH is the implementation (Felsenstein, 1997) of the basic principles described in the seminal paper of Fitch and Margoliash (1967). Its algorithmic strategy is not agglomerative but additive. FITCH constructs a tree by iteratively adding taxa to a growing tree. At each step, it performs tree swapping to improve the goodness-of-fit, using nearest-neighbor interchange (i.e., exchange of subtrees separated by 3 branches). Finally, once a first tree has been constructed, it optionally (see discussion of FITCH in Alternate Protocol 1) performs a more extensive search in the tree space by considering global rearrangements—every subtree is removed from the tree and put back on in all possible ways so as to have a better chance of finding a better tree. The resulting tree may be sensitive to the initial taxon ordering, even when the swapping procedures tend to lower its influence. So the jumbling procedure (Alternate Protocol 1) must be used, unless there are computational time constraints.

FITCH optimizes the weighted least-squares criterion. Let $(\delta_{ij})$ be the matrix of distance estimates and $(\hat{t}_{ij})$ the distance matrix induced by the inferred tree $\hat{T}$ and its branch lengths. The weighted least-squares fitting of $\hat{T}$ is defined by:

$$\text{WLS}(\hat{T}) = \sum_{i\neq j} \frac{1}{\text{VAR}[\delta_{ij}]}\left(\hat{t}_{ij} - \delta_{ij}\right)^2$$

**Equation 6.3.7**

where $VAR[\delta_{ij}]$ is the variance of the $\delta_{ij}$ estimate. This criterion has to be minimized, and has value 0 when $\hat{T}$ perfectly represents $(\delta_{ij})$. Various solutions are possible for the variance of $\delta_{ij}$, which may be written as $VAR[\delta_{ij}] = \delta_{ij}^p$. When the power $p$ is null, all variances are equal to 1.0, and the higher variance of long distances is not taken into account. When $p = 1$, the variance of $\delta_{ij}$ is equal to $\delta_{ij}$, and the model is equivalent to that of BIONJ without the covariance terms. The best results, however, are obtained with $p = 2$, which corresponds to the solution of Fitch and Margoliash (1967) and is quite close to the WEIGHBOR model. This is the default option of FITCH.

The criterion in the above sum of squares equation not only concerns the topology of $\hat{T}$, but also its branch lengths. Minimizing this criterion induces branch length estimates which have to be positive for the approach to be consistent. This is one other default option (to be conserved) of FITCH.

### *FastME*

FastME builds trees using the following principle. For each tree $T$, least-squares length estimates are assigned to each branch. Next, the sum of the branch lengths is calculated and set to the value $l(T)$. The tree minimizing $l(T)$ is chosen, which is in spirit analogous to parsimony, but this choice requires (relatively) complex mathematical explanations to be fully understood (Desper and Gascuel, 2005). FastME allows the user to search topologies when branch lengths are estimated either by ordinary least-squares (OLS) or balanced least-squares (BLS). Ordinary least-squares branch lengths are assigned according to Equation 6.3.7, with $p = 0$ and variances constant. Balanced least-squares (Pauplin, 2000; Desper and Gascuel, 2004) represents a weighted scheme per Equation 6.3.7, with:

$$\text{VAR}\left[\delta_{ij}\right] = 2^{p_{ij}}$$

**Equation 6.3.8**

where $p_{ij}$ is the path length (number of branches) from taxon $i$ to taxon $j$ in $T$. In other words, variances in the balanced least-squares scheme increase exponentially as a function of the evolutionary distance, in a way similar to WEIGHBOR's.

FastME builds an initial tree additively, as does FITCH. Each taxon $i$ is inserted optimally into the tree built on the first $(i-1)$ taxa. Alternatively, the user can provide the initial topology, e.g., using NJ or BIONJ.

From the initial topology, FastME searches through the space of tree topologies by testing each possible nearest neighbor interchange (NNI; Fig. 6.3.16). This search can be performed quickly, as each possible NNI can be tested in constant time. While either OLS or BLS length estimates can be used, this discussion will focus on the latter, which have been demonstrated to have better statistical properties for biological data sets (Desper and Gascuel, 2004). The value of the topology change can be expressed as a linear sum of "balanced" average distances.

Balanced average distances (Pauplin, 2000) are defined with respect to a tree $T$. If $a$ and $b$ are leaves of $T$, $\delta^T_{\{a\}|\{b\}} = \delta^T_{ab}$, the distance from $a$ to $b$ in $T$, is defined. More generally, if $A$ and $B$ are the leaf sets of two disjoint subtrees of $T$, $\delta^T_{A|B}$ is defined recursively. Presuming that it is not the case that both $A$ and $B$ are singleton sets, without loss of generality there are two subtrees $B_1$ and $B_2$ that meet at an internal node to form $B$. Then the equation:

$$\delta^T_{A|B} = \frac{1}{2}\left(\delta^T_{A|B_1} + \delta^T_{A|B_2}\right)$$

**Equation 6.3.9**

is defined. Suppose $T \rightarrow T'$ is the topology transformation resulting from the NNI in Figure 6.3.16. Then:

$$l(T) - l(T') = \frac{1}{4}\left(\delta^T_{A|C} + \delta^T_{B|D} - \delta^T_{A|B} - \delta^T_{C|D}\right)$$

**Equation 6.3.10**

FastME first calculates the value of $\delta^T_{U|V}$ for each pair of disjoint subtrees $U,V$ in $T$. Using the structure of $T$, this can be done in time proportional to $n^2$, where $n$ is the number of taxa. After this is done, each possible value of an NNI can be calculated using Eq. 6.3.10 in constant time. Once the optimal NNI is found, the topology is changed and the matrix of average distances is updated in time proportional to $n\,\mathrm{diam}(T)$, where $\mathrm{diam}(T)$ is the length (number of branches) of the longest path between any taxon pair.

Desper and Gascuel (2002) and Vinh and Von Haeseler (2005) showed via simulations that BLS FastME post-processing improves the quality of the output tree when the input tree is produced by any of the major distance algorithms. Furthermore, the authors of this unit have demonstrated (Desper and Gascuel 2005) that the NJ algorithm is another type of greedy algorithm optimizing the BLS criterion, albeit with a restricted search space; Equation 6.3.1 represents the difference in BLS tree length that is obtained by agglomerating the pair $i,j$, and NJ greedily agglomerates taxon pairs until a fully resolved trees is obtained.

### *Method comparison*

Numerous computer simulations have been performed to compare the topological accuracy of phylogeny reconstruction methods. The principle is: (a) consider a "true tree," (b) evolve an initial random sequence along this tree to obtain "contemporary sequences," (c) reconstruct a tree from these sequences, (d) finally, compare the inferred tree to the true tree. Drawing definitive conclusions from such a study is difficult because the results depend on the true tree, on the evolutionary conditions, and on numerous parameters. Moreover, numerous available studies have considered a

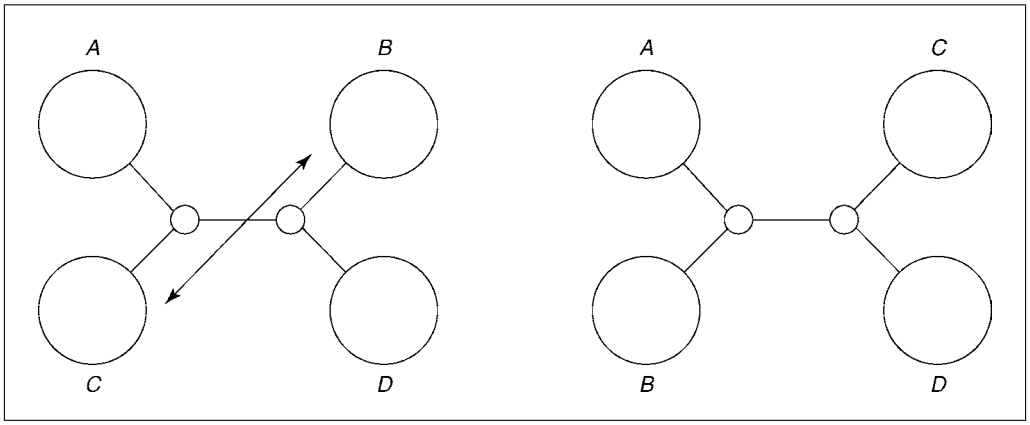


**Figure 6.3.16** Nearest Neighbor Interchange (NNI) swapping subtrees *B* and *C*.

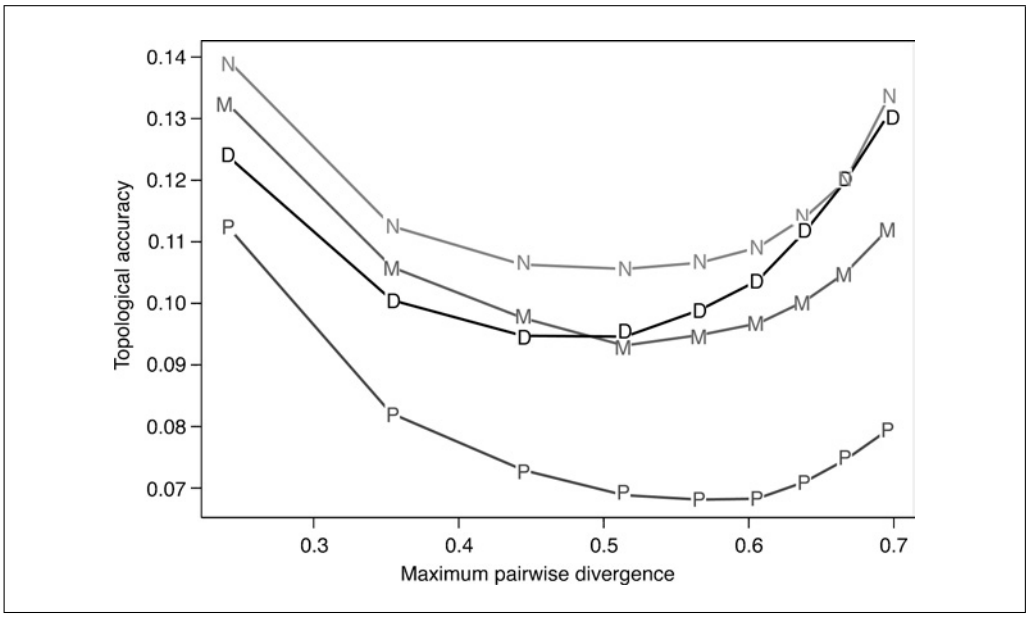


**Figure 6.3.17** Topological accuracy of NEIGHBOR (N), FastME (M), DNAPARS (D) and PHYML (P) with 5000 randomly generated 40-taxon Trees. Maximum pairwise divergence is given in number of substitutions per site. Topological accuracy is measured by the number of clades in the inferred tree that are not in the correct tree, divided by the total number of clades; 0.0 corresponds to identical trees, while 1.0 means that both trees have no clade in common.

low number of true trees and few taxa (usually 12 or less).

The authors of this unit have recently tried to overcome these limits by randomly generating thousands of trees, with realistic numbers of taxa (from 24 to 100), under a broad variety of evolutionary conditions. Figure 6.3.17 displays the topological accuracy of four programs: NJ, FastME, DNAPARS (parsimony-based algorithm used with default options, from the PHYLIP package), and PHYML (Guindon and Gascuel, 2003), which is a fast maximum-likelihood algorithm. The results were obtained with 40-taxon trees generated using the standard Yule-Harding speciation process, maximum pairwise divergence uniformly drawn from [0.1,1.1], and the molecular clock varying from full satisfaction to strong violation. Figure 6.3.17 clearly shows the advantage of using FastME rather than NJ, as FastME is more accurate than NJ whatever the maximum pairwise divergence among taxa. With this data set, WEIGHBOR and FITCH are very close to FastME, while BIONJ accuracy is midway between NJ's and FastME's (results not shown). Moreover, Figure 6.3.17 gives a clear view of the relative topological accuracy of the three main approaches: distance, parsimony, and maximum-likelihood. The latter is the best method in all conditions, and the gap with the other approaches is rather impressive. DNAPARS is globally equivalent to FastME, but its relative accuracy depends on the maximum-pairwise divergence; it performs well with low evolutionary rates, but its performance is no better than NJ's under the opposite condition.

More taxa are required to distinguish among the best distance-based methods. In Desper and Gascuel (2004), the authors of this unit generated 100-taxon trees using the Aldous distribution on trees, which generalizes the simple Yule-Harding distribution. 600-bp DNA sequences were evolved through each tree topology, with rates of evolution varying from site to site according to a covarion model (Galtier, 2001). From the resulting DNA sequences, distance matrices were calculated using the Nei and Jin (1989) estimate for gamma-distributed rates. Algorithms tested included FastME, NJ, WEIGHBOR, and the weighted least squares heuristic search of PAUP*, which is similar to FITCH but much faster. Results indicated that FastME has best topological accuracy, whatever the value of various simulation parameters related to tree shape, molecular clock violation, maximum pairwise divergence, and covarion model of sequence evolution.

Finally, very large-scale simulations with thousands of taxa were published recently by Vinh and von Haeseler (2005), to compare fast distance methods, i.e., NJ, BIONJ, BME (FastME's greedy insertion algorithm

based on balanced minimum evolution), and STC (Vinh and von Haeseler's own method based on the use of short triplets). They found that STC produced the best initial topologies for post-processing, followed by BME. More importantly, they also found that FastME post-processing (with balanced minimum evolution) improved the topological accuracy, without regard to which algorithm was used to select an initial topology. Furthermore, the accuracy was fairly uniform after post-processing without regard to the selection of the initial topology. Thus, it appears that FastME post-processing is the key step here, and that it should be used whenever large data sets are analyzed. Moreover, as the initial tree-building algorithm has low influence, using the BME algorithm for this purpose (FastME default option) is a simple, fast and accurate solution.

It can be seen that contrast between methods in Figure 6.3.17 is not very high, even when significant. The contrast between run times in Table 6.3.1 is much more impressive. NJ, BIONJ, and FastME are one order of magnitude faster than any of the other methods. They require about one-half second to deal with 250 taxa, and Vinh and Von Haeseler (2005) showed that these three methods need only a few minutes to build a tree with 5000 taxa on a standard computer (see Howe et al., 2002, for the analysis of even larger data sets). In fact, they require much less time than DNADIST requires to compute the matrix of pairwise distances. But this latter program is rather slow, due to the use of optimization-based distance estimators, and faster analytical solutions do exist to estimate evolutionary distances. WEIGHBOR is about 500 times slower than NJ and is only applicable to data sets with up to a few hundred sequences, while FITCH is very slow, as it requires about 12 hr to analyze 250 taxa. DNAPARS is also quite slow, as it requires ~8 hr to deal with 250 taxa, but TNT (see Internet Resources) is much faster and finds better trees (24 parsimony steps, with 250 taxa) than DNAPARS. DNAML is also slow, even when using the speedier option (as performed by the authors of this unit), and requires about 100 min to deal with 250 taxa; moreover, its results in terms of likelihood are rather poor as compared to PHYML (DNAML tree is about 500 log likelihood points below PHYML's, with 250 taxa). Finally, PHYML is

**Table 6.3.1** Run Times for Various Tree-Building Methods

| | Run time (sec) | | |
|---|---|---|---|
| | Number of taxa | | |
| Method[a,b,c] | 40 | 100 | 250 |
| DNADIST | 0.09 | 0.65 | 25 |
| FastME | 0.008 | 0.055 | 0.34 |
| NJ | 0.0045 | 0.035 | 0.25 |
| BIONJ | 0.0052 | 0.055 | 0.60 |
| WEIGHBOR | 1.1 | 18 | 255 |
| FITCH | 6 | 335 | 43,200 |
| DNAPARS | 6 | 230 | 30,000 |
| TNT | 5 | 13 | 330 |
| DNAML | 26 | 186 | 6,000 |
| PHYML | 7.5 | 20 | 390 |

[a]Distance estimation:, DNADIST; distance-based tree building methods, FastME, NJ, BIONJ, WEIGHBOR, and FITCH; parsimony methods, DNAPARS and TNT; maximum-likelihood methods, DNAML and PHYML.

[b]TNT was run with 500 (40 and 100 taxa) or 50 (250 taxa) starting points and TBR option. All other programs were run with default options and K2P model of sequence evolution. 40-taxon and 100-taxon (simulated) data sets were taken from (Guindon and Gascuel, 2003); results were averaged over 5000 and 30 data sets, respectively. The 250-taxon (biological) data set was used in (Stamatakis et al., 2004) to compare maximum-likelihood programs.

[c]Run times are in seconds with a Windows PC 1.8 MHz and 1.0 Gb RAM.

rather efficient as it requires about 6 min with 250 taxa; it can be used up to 500 taxa and has very good topological accuracy (Figure 6.3.17; Guindon and Gascuel, 2003).

These results show that, with very large data sets, e.g., >1000 taxa, the distance approach is the only one to be applicable, as soon as fast algorithms are used, i.e., NJ, BIONJ, or FastME. FastME should be preferred, as it clearly has the best topological accuracy among the distance methods, while NJ is worse and BIONJ is in between (Vinh and Von Haeseler, 2005). With large data sets involving a few hundred taxa, fast parsimony (e.g., TNT) and maximum-likelihood (e.g., PHYML) programs become applicable, and the latter should be preferred over any other approach, as it has the best topological accuracy. However, distance methods are still of interest for fast exploratory study, and to perform bootstrap analysis, which is very demanding in terms of computing time. Finally, with moderate data sets, e.g., <100 taxa, all methods are applicable; the use of distance methods should then be limited to fast analysis, and maximum-likelihood should be employed in other cases.

## Critical Parameters and Troubleshooting

Distance-based approaches are sensitive to the way evolutionary distances are estimated. When the sequences exhibit few differences, all sequence evolution models become equivalent, and the model choice is not crucial. For example, when two sequences have 0.1 sites that differ with 0.07 transitions and 0.03 transversions, the Jukes and Cantor distance estimate is equal to 0.1073, the Kimura two-parameter estimate is 0.1086, and the Jin and Nei estimate (with $\alpha = 1.0$) is 0.1183. Distance estimation, however, becomes very sensitive to the model choice when the maximum pairwise divergence among the sequences increases. For example, consider two sequences with half of the sites being different, with 0.35 transitions and 0.15 transversions, the Jukes and Cantor estimate is equal to 0.824, the Kimura two-parameter estimate is 1.037, and the Jin and Nei estimate ($\alpha = 1.0$) is 2.940. Therefore, data sets where the sequence divergence is too high (say >1.0) must be considered suspicious and should be discarded. Note that the presence of such high divergence makes the alignment itself very difficult and prone to errors. With more reasonable maximum divergence, the stability of the results despite model variations is a positive point. Moreover, the presence of distant outgroup taxa is a perturbation factor in all reconstruction steps (alignment, distance estimation, and tree building) and should be avoided, at least in a first analysis.

## Suggestions for Further Analysis

Distance methods are available in numerous phylogeny software packages. Notably, PAUP* (release 4.0b10; *UNIT 6.4*) provides very fast versions of NJ, FITCH, and BIONJ, as well as a larger variety of evolutionary distance estimates than that provided by DNADIST and PROTDIST. DAMBE (Xia et al., 2001) also contains an implementation of FastME and a complete environment to perform bootstrap, drawing the trees and computing the distances.

Parsimony approaches do not outperform distance methods (see Fig. 6.3.17), but their principle is so different that finding the same tree using both is generally considered to be a strong support for that tree. PAUP* (*UNIT 6.4*) and TNT (see Internet Resources) provide fast parsimony implementations.

PHYML (Guindon and Gascuel, 2003) is a fast and accurate maximum likelihood software (see Fig. 6.3.17 and Table 6.3.1), which should be preferred over other approaches with data sets that are not too large (<500 taxa). PHYML is freely downloadable at *http://atgc.lirmm.fr/phyml/*, where a Web server is also available.

## Literature Cited


Atteson, K. 1997. The performance of the NJ method of phylogeny reconstruction. *In* Mathematical Hierarchies and Biology (B. Mirkin, F.R. McMorris, F.S. Roberts, and A. Rzhetsky, eds.) pp.133-148. American Mathematical Society, Providence, R.I.

Berry, V. and Gascuel, O. 1996. Interpretation of bootstrap trees: Threshold of clade selection and induced gain. *Mol. Biol. Evol.* 13:999-1011.

Bruno, W.J., Socci, N.D., and Halpern, A.L. 2000. Weighted neighbor joining: A likelihood-based approach to distance-based phylogeny reconstruction. *Mol. Biol. Evol.* 17:189-197.

Bulmer, M. 1991. Use of the method of generalized least squares in reconstructing phylogenies from sequence data. *Mol. Biol. Evol.* 8:868-883.

Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C. 1979. A model for evolutionary change in proteins. *In* Atlas of Protein Sequence and Structure (M.O. Dayhoff, ed.), vol. 5, pp. 345-352.

Desper, R. and Gascuel, O. 2002. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *J. Comp. Biol.* 9:687-705.

Desper, R. and Gascuel, O. 2004. Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. *Mol. Biol. Evol.* 21:587-598.

Desper, R. and Gascuel, O. 2005. The minimum evolution distance-based approach to phylogenetic inference. *In* Mathematics of Evolution and Phylogeny (O. Gascuel, ed.) pp. 1-32. Oxford University Press, Oxford.

Felsenstein, J. 1989. PHYLIP–Phylogeny inference package (version 3.2). *Cladistics* 5:164-166.

Felsenstein, J. 1997. An alternating least-squares approach to inferring phylogenies from pairwise distances. *Syst. Biol.* 46:101-111.

Felsenstein, J. and Churchill, G.A. 1996. A hidden Markov model approach to variation among sites in rate of evolution *Mol. Biol. Evol.* 13: 93-104

Fitch, W.M. and Margoliash, E. 1967. Construction of phylogenetic trees. *Science* 155:279-284.

Galtier, N. 2001. Maximum-likelihood phylogenetic analysis under a covarion-like model. *Mol. Biol. Evol.* 18:866-873.

Gascuel, O. 1997a. BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.* 14:685-695.

Gascuel, O. 1997b. Concerning the NJ algorithm and its unweighted version, UNJ. Mathematical Hierarchies and Biology (B. Mirkin, F.R. McMorris, F.S. Roberts, and A. Rzhetsky, eds.) pp. 149-170. American Mathematical Society, Providence, R.I.

Graur, D. and Li, W.-H. 2000. Fundamentals of Molecular Evolution. Sinauer Associates, Sunderland, Mass.

Guindon, S. and Gascuel, O. 2002. Efficient biased estimation of evolutionary distances when substitution rates vary across sites. *Mol. Biol. Evol.* 19:534-543.

Guindon, S. and Gascuel, O. 2003. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.* 52:696-704.

Howe, K., Bateman, A., and Durbin, R. 2002. QuickTree: Building huge Neighbour-Joining trees of protein sequences. *Bioinformatics* 18:1546-1547.

Jin, L. and Nei, M. 1990. Limitations of the evolutionary parsimony method of phylogenetic analysis. *Mol. Biol. Evol.* 7:82-102.

Jones, D.T., Taylor, W.R., and Thornton, J.M. 1992. The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.* 8:275-82.

Jukes, T.H. and Cantor, C.R. 1969. Evolution of protein molecules. Mammalian Protein Metabolism (H. N. Munro, ed.) pp.21-132. 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.

Kimura, M. 1983. The Neutral Theory of Molecular Evolution. Cambridge University Press, Cambridge, U.K.

Kishino, H. and Hasegawa, M. 1989. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in *Hominoidea J. Mol. Evol.* 29:170-179.

Nei, M. and Jin, L. 1989. Variances of the average numbers of nucleotide substitutions within and between populations. *Mol. Biol. Evol.* 6:290-300.

Page, R.D.M. and Holmes, E.C. 1998. Molecular Evolution: A Phylogenetic Approach. Blackwell Scientific, Oxford, U.K.

Pauplin, Y. 2000. Direct calculation of a tree length using a distance matrix. *J. Mol. Evol.* 51:41-47.

Perrière, G. and Gouy, M. 1996. WWW-Query: An on-line retrieval system for biological sequence banks. *Biochimie* 78:364-369.

Rzhetsky, A. and Nei, M. 1993. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Mol. Biol. Evol.* 10:1073-1095.

Saitou, N. and Nei, M. 1987. The neighbor-joining method: A new method for reconstruction of phylogenetic trees. *Mol. Biol. Evol.* 4:406-425.

Sattath, S. and Tversky, A. 1977. Additive similarity trees. *Psychometrika* 42:319-345.

Sokal, R.R. and Michener, C.D. 1958. A statistical method for evaluating systematic relationships. *Univ. Kans. Sci. Bull.* 38:1409-1438.

Stamatakis, A., Ludwig, T., and Meier, H. 2004. RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics* 21:456-463.

Steel, M.A. 1994. Recovering a tree from the Markov leaf colourations it generates under a Markov model. *Appl. Math. Lett.* 7:19-23.

Studier, J.A. and Keppler, K.J. 1988. A note on the neighbor-joining algorithm of Saitou and Nei. *Mol. Biol. Evol.* 5:729-31.

Swofford, D.L., Olsen, G.L., Waddell, P.J., and Hillis, D.M. 1996. Phylogenetic inference. *In* Molecular Systematics (D.M. Hillis, C. Moritz, and B.K. Mable, eds.) pp. 407-514. Sinauer Associates, Sunderland, Mass.

Thompson, J.D., Gibson, T.J., Plewniak, F., Jeanmougin, F., and Higgins, D.G. 1997. The ClustalX windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res.* 25:4876-4882.

Veerassamy, S., Smith, A., and Tillier, E.R. 2003. A transition probability model for amino acid substitutions from blocks. *J. Comput Biol.* 10:997-1010.

Vinh, L.S. and von Haeseler, A. 2005. Shortest triplet clustering: Reconstructing large phylogenies using representative sets. *BMC Bionfirmatics* 6:92.

Xia, X. and Xie, Z. 2001. DAMBE: Data analysis in molecular biology and evolution. *J. Hered.* 92:371-373.

Yang, Z. 1996. Among-site rate variation and its impact on phylogenetic analyses. *TREE* 11:367-372.

Zaretskii, K. 1965. Reconstructing a tree from the distances between its leaves. *Uspehi Mathematicheskikh Nauk* 20:90-92 (in Russian).

## Internet Resources

http://atgc.lirmm.fr/fastme

*This web page from the authors provides FastME C source code and binaries for Windows, MAC OS and LINUX, as well as several papers to understand in depth the minimum evolution principle, its algorithms and its properties.*

http://atgc.lirmm.fr/phyml

*This web page provides PHYML binaries for Windows, MAC OS and LINUX, and a web server to run PHYML online.*

http://www.cladistics.com/webtnt.html

*Goloboff, P., Farris, S., and Nixon, K. 2000. TNT: Tree analysis using new technology. Beta version, published by the authors, Tucumán, Argentina.*

http://evolution.genetics.washington.edu/phylip/software.html

*Joe Felsenstein's Web page, containing an extensive list of phylogeny software programs, including numerous distance-based methods.*

---

Contributed by Richard Desper
Department of Biology
University College
London, United Kingdom

Olivier Gascuel
Equipe "Méthodes et Algorithmes pour la Bioinformatique"
LRMM-CNRS
Montpellier, France

# Inferring Evolutionary Trees with PAUP*



PAUP* version 4 (Swofford, 2002) is a widely used software package for inferring evolutionary trees. PAUP* implements criterion-based methods such as parsimony (see Background Information), maximum likelihood (see Background Information), and least-squares, and provides a variety of search strategies for estimating phylogenies. PAUP* can also reconstruct trees using algorithms such as UPGMA and neighbor joining (*UNIT 6.3*). In addition to reconstructing phylogenies, other features of the program include diagnosing characters and inferring ancestral states on a phylogenetic tree and testing the robustness of phylogenetic trees using several descriptive and statistical techniques.

One of PAUP*'s strengths is the rich assortment of phylogenetic methods that can be implemented within the program. However, for people not familiar with the field of phylogenetic inference, the number of available methods and options in PAUP* may seem overwhelming. This unit is designed to provide an entry-level overview of how to implement some of the most basic phylogenetic methods available in PAUP*. The Basic Protocol uses the parsimony method to infer evolutionary trees and can be used for DNA, RNA, and protein sequences. The Alternate Protocol uses a model-based, statistical approach—maximum likelihood—to infer evolutionary trees for DNA and RNA sequences only. For obvious reasons, the authors have chosen not to describe in this unit all the possible analysis options available in PAUP*. In addition, the working details of the methods implemented in some of the examples will not be described; instead, citations to relevant and accessible references will simply be provided. After reading this unit, one should have enough experience with the program to use other available methods that are described more completely in the current program documentation.

## USING PAUP* TO INFER PARSIMONY TREES FROM DNA SEQUENCES

***BASIC PROTOCOL***

PAUP* can be used to infer evolutionary trees and perform associated analyses in both interactive and batch mode. As the name implies, the interactive mode requires a user's input at various stages of the analysis, whereas batch mode allows the user to set up analyses in advance and run them without being in attendance. While both methods are useful, this protocol will focus mostly on how to operate PAUP* in the interactive mode. See Suggestions for Further Analysis in the Commentary for guidelines for running PAUP* in batch mode.

### *Necessary Resources*

*Hardware*

PAUP* is compatible with most modern hardware configurations including Apple Macintosh (Power PC- and 68K-based processors), Intel, AMD, and a number of Unix and Linux workstations (e.g., Sun, Alpha, IBM, SGI, Power PC-based, i386-based, and others)

*Software*

PAUP* is distributed by Sinauer Associates at *http://www.sinauer.com*. Three versions of the program are available: Macintosh, Windows, and Portable (see Table 6.4.1). These versions provide identical analytical capabilities, but differ in the way these analyses are controlled. The Macintosh version supports a full graphical user interface, which allows the user to execute commands via menus and the command line, while the Windows and Portable versions are almost entirely command-line driven. Some menu functions are available in the

**Contributed by James C. Wilgenbusch and David Swofford**
*Current Protocols in Bioinformatics* (2003) 6.4.1-6.4.28

**Table 6.4.1** Platforms that are Currently Supported by PAUP* Version 4.0

| Interface type | Processor | Operating System |
|---|---|---|
| Macintosh | G3/G4 | Mac OS X (classic) |
| | PowerPC and 68k-based | Mac 7.0 or later |
| Windows | i386-based | Windows 95/98/ME |
| | | Windows NT/2000 |
| Portable | i386-based | Linux 2.0 or later/FreeBSD |
| | PPC | Linux 2.0 or later |
| | Sparc and UltraSparc | Solaris |
| | Alpha | Digital Unix/True64 |
| | MIPS II and III | Irix |
| | RS6000 | AIX |
| | Possibly others | Possibly others |

Windows version; however, the functions are mostly restricted to file "open" and "edit" operations. Because the command-line interface is available in all three versions, it is used for the following examples. The location of the corresponding menu item is given in the PAUP* command reference documentation (available on the PAUP* 4.0 distribution disk and on the PAUP* Web site at *http://paup.csit.fsu.edu/commandref.pdf*). In addition, several sources already offer an overview of PAUP* using the Macintosh menu interface (Hall, 2001; also see PAUP* quick-start tutorial, *http://paup.csit.fsu.edu/quickstart.pdf*). Installation instructions are included with the software.

*Files*

Data files for PAUP* are standard text files, which adhere to the NEXUS file specifications. The NEXUS format was designed by Maddison et al. (1997) to facilitate the interchange of input files between programs used in phylogeny and classification. The text in a NEXUS file is arranged into blocks, which are delimited by the words `begin` and `end`. The text immediately following the word `begin` defines the block type. For example, the `TAXA` and `CHARACTERS` blocks shown in Figure 6.4.1 define a simple data set composed of four taxa (sequences) and 60 nucleotide characters (sites). PAUP* supports several predefined data types: DNA, RNA, nucleotide (DNA or RNA), protein, and standard. The set of predefined character-state symbols are `ACGT` for the data type DNA, `ACGU` for RNA, the standard one-letter amino acid codes for protein, and `01` for standard. In addition, standard ambiguity codes for the molecular data types are implemented by predefined "equate" macros. Additional character states other than those represented by the predefined data types can be specified using the `SYMBOLS` subcommand (see the PAUP* command reference documentation at the above URL). PAUP* data files must start with the character string `#NEXUS`. Comments can be included in a data file by enclosing them in square brackets, as shown in Figure 6.4.1. PAUP* 4.0 is also capable of translating other file formats to the NEXUS format. Supported formats include PHYLIP (*UNIT 6.3*), Hennig86, GCG/Pileup (*UNIT 3.6*), MEGA, NBRF-PIR, FreqPars, and text (space and tab-delimited). The Support Protocol below details using PAUP* to import non-NEXUS data files. The NEXUS file used for this protocol can be found in the application subdirectory labeled `Sample-NEXUS-data` and also on the *Current Protocols in Bioinformatics* Web site at *http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm*.


**Figure 6.4.1** A sample NEXUS data set composed of 4 sequences and 60 nucleotide characters.

*NOTE:* Due to formatting constraints, some commands given in the following examples span multiple lines. However, when entering commands at the `paup>` prompt or into the command-line interface, all of the text preceding the semicolon should be entered on the same line.

### *Input the sequences*

1. Start PAUP*.

   *The way in which PAUP* is started will depend on the platform the user is using. To start PAUP* on a Windows or Macintosh machine, locate the application icon and double-click it. PAUP* will automatically launch the Open File dialog box when it is first started (Fig. 6.4.2). For the Portable interface, the command* `paup` *should be in one's path and the command should be linked to the PAUP* executable (see the README file in the program installation archive and* APPENDIX 1C*). In this case, start the program by changing directories to the subdirectory called* `Sample-NEXUS-files` *included in the PAUP* archive and then type the command* `paup` *at the shell prompt.*

2. Execute the data set.

   *Macintosh and Windows users should select the file named* `primate-mtDNA.nex` *from the Open File dialog box and click the Open/Execute button. Users of the Portable interface should type the command* `execute primate-mtDNA.nex;` *at the* `paup>` *prompt. After executing the sample file, PAUP* will display comments and some general information about the data (Fig. 6.4.3). For this example, the source of the data set is given, followed by comments included in the NEXUS file, a section reporting the dimensions of the data matrix, the type of data, and several other characteristics of the data set. At this point, no analyses will have been conducted; PAUP* has simply processed the data and is now waiting to be told what to do next.*

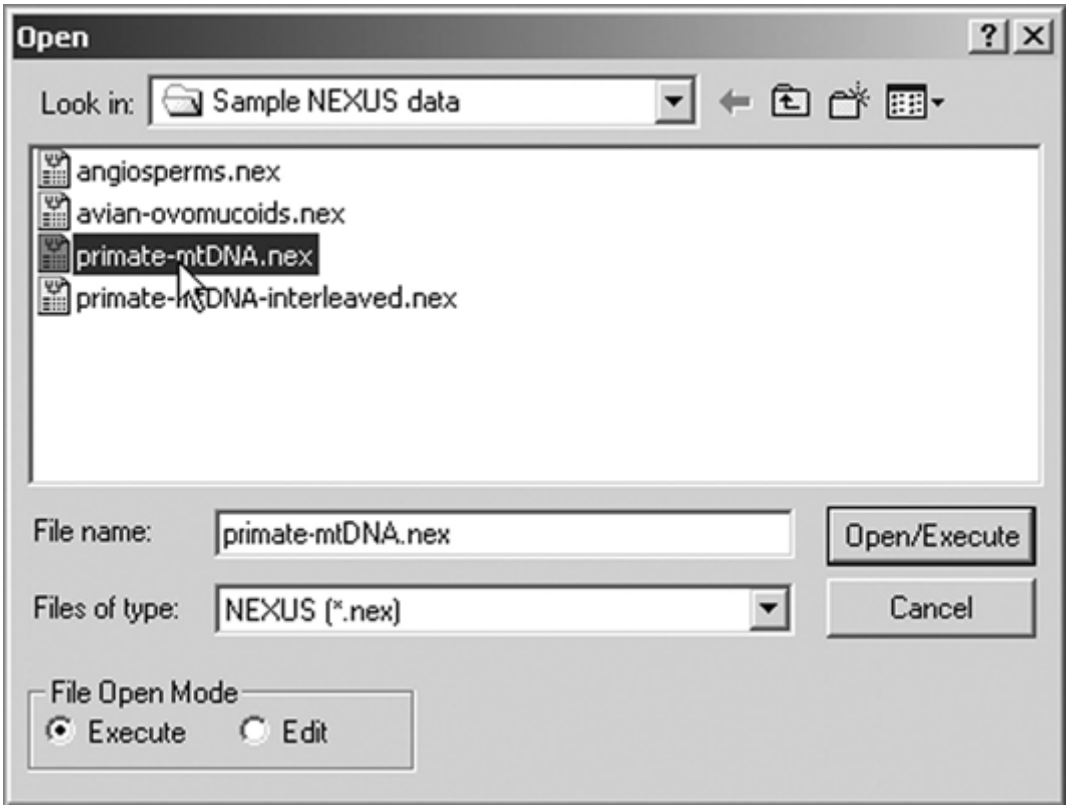


**Figure 6.4.2** The Open File dialog box automatically launched when the Windows and Macintosh versions of PAUP* are first started.

***Manage the input data***

3. Specify the sites to be included in the analysis. For example, exclude noncoding regions.

```
include coding/only;
```

*Windows and Macintosh users will enter the* `include` *command in the command-line interface located at the bottom of the main display window (Fig. 6.4.4). Portable-version users will enter commands at the* `paup>` *prompt.*

*PAUP* provides several ways to restrict an analysis to a subset of the characters included in a data matrix without permanently removing the characters from the NEXUS file. The sample data set includes protein coding and noncoding regions of primate mitochondrial DNA. Suppose one wishes to analyze only the coding regions of the data. The* `/only` *option is included when only the characters listed after the* `include` *command are to be included in the analysis. The characters belonging to these regions have already been identified in the sample file using the* `charset` *command (Fig. 6.4.5). Character sets simplify certain procedures by allowing users to refer to a group of characters with a single name (see PAUP* command reference documentation; http://paup.csit.fsu.edu/commandref.pdf). PAUP* also includes several predefined character sets that may be used with specific types of data (Table 6.4.2).*

4. Specify sequences (taxa) to be used. For example, delete all but five hominoid and three non-hominoid sequences.

```
delete M._mulatta M._fascicularis M._sylvanus
Tarsius_syrichta;
```

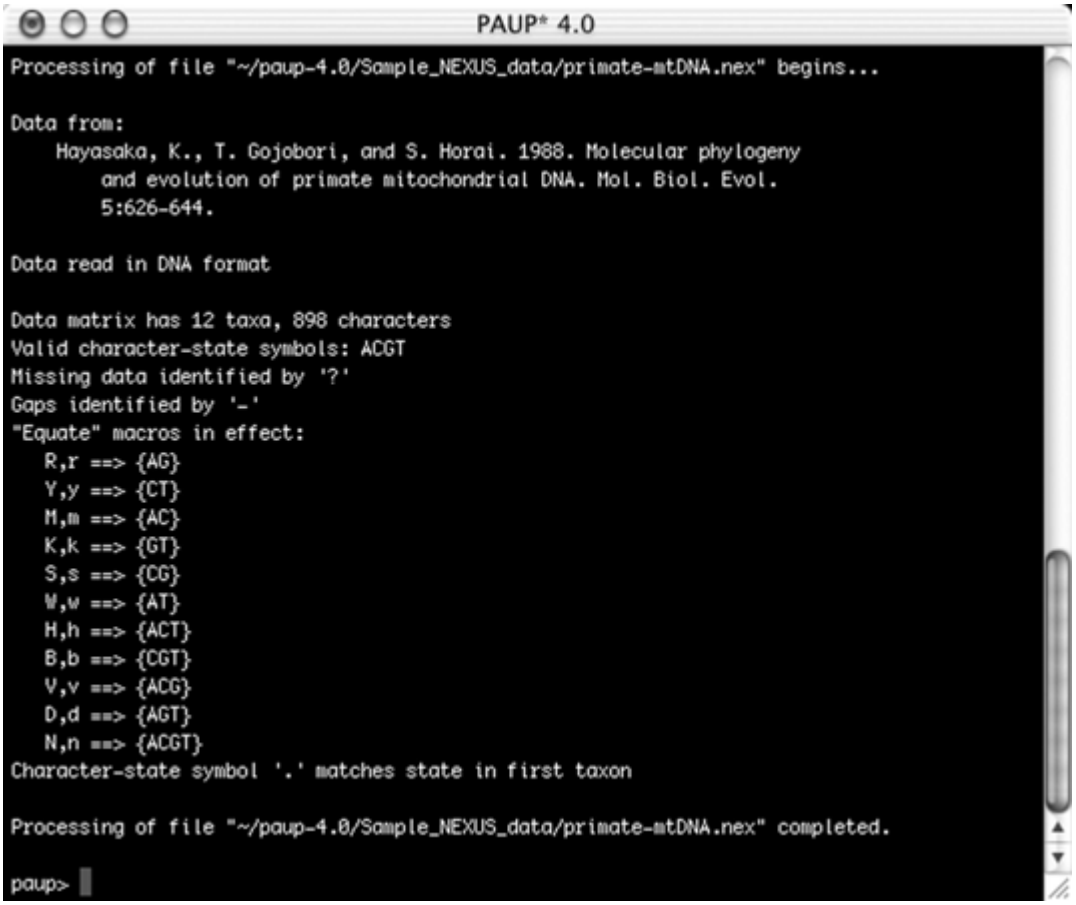


**Figure 6.4.3** The PAUP* main display after executing the sample data set.

Equivalently:

```
undelete hominoids lemur_catta macaca_fuscata
saimiri_sciureus/only;
```

*The five hominoids included in the example data set (Homo sapiens, Pan, Gorilla, Pongo, and Hylobates) have already been identified using the* `taxset` *command (Fig. 6.4.5). In the same way that a* `charset` *can be used to identify a group of characters by a single name,* `taxset` *can be used to identify a group of taxa. The* `delete` *or* `undelete` *commands can be used to perform this operation. Notice that the* `/only` *option is used with the* `undelete` *command so that other taxa already in the data set are not also included in subsequent analyses. There are no predefined taxon sets; the user must define them manually as shown in Figure 6.4.5 (see PAUP* command reference documentation; http://paup.csit.fsu.edu/commandref.pdf). Spaces in taxon names are identified by using an underscore character or by enclosing the name in single quotes. PAUP* does not pay attention to the character case in taxon labels.*

### *Select an optimality criterion and define assumptions*

5. Set the optimality criterion to parsimony.

```
set criterion=parsimony;
```

*In this case,* `parsimony` *will be used as the optimality criterion for selecting a tree (see Background Information). Because parsimony is the default criterion used by PAUP* when the program is first started, the* `set` *command given above is only necessary if the default*


**Figure 6.4.4** The command-line interface located at the bottom of the Macintosh and Windows main display window.

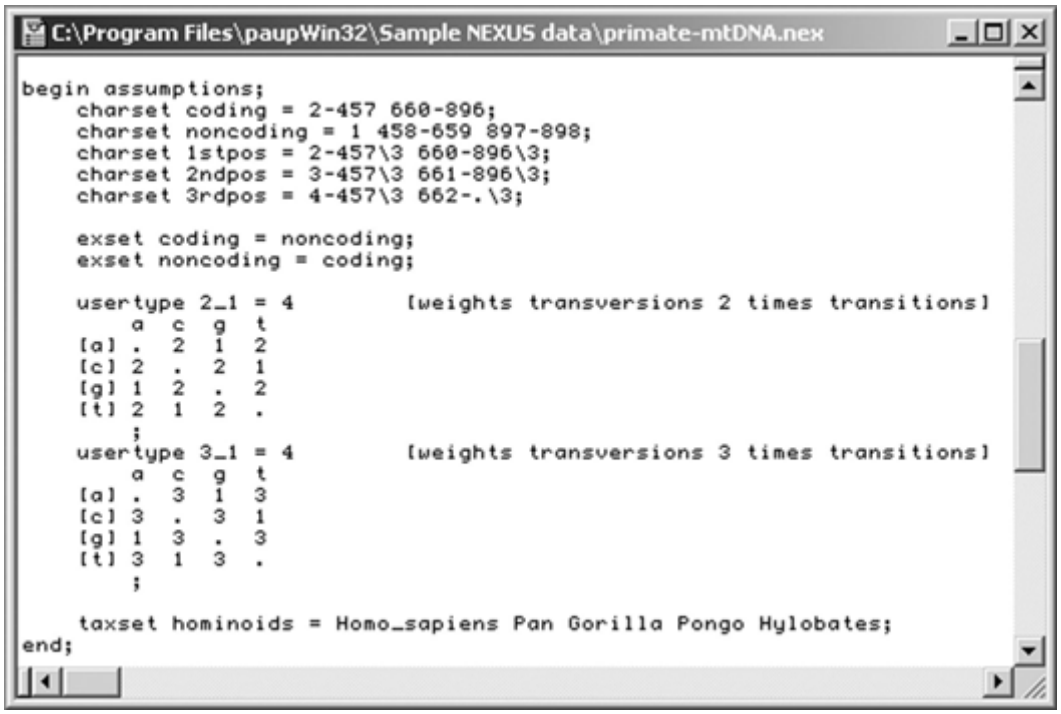


**Figure 6.4.5** The `assumptions` block included at the end of the sample data set. The block contains the character and taxa sets and the user-defined character types that are used in the example analysis.

*settings have been altered. If unsure, it is a good idea to go ahead and issue this command. The Alternate Protocol describes how to set up a search using the maximum-likelihood criterion.*

6. Consider setting character weighting. For example:

```
weights 2:2ndpos;
```

weights all transformations at nucleotides in the `2ndpos` charset (defined in Fig. 6.4.5) by a factor of two. All other transformations have the default weight of 1.

*Under the default parsimony criterion used in PAUP*, transformations from one character state to another are given the same score no matter which site in the data matrix is being examined. Under some circumstances, it may be perfectly appropriate to leave the default weighting settings unchanged. For example, it may be unnecessary to weight sites differentially when a parsimony search is used only as a way to get a tree quickly for use in the parameterization of a subsequent maximum-likelihood search (see Alternate Protocol, step 2). Also, unlike maximum likelihood, the parsimony criterion lacks an objective function that may be used to select from among the many possible weighting schemes (see*

**Table 6.4.2** Predefined Characters Sets and Their Corresponding Data Types

| Data type | Character set name | Definition |
|---|---|---|
| All data types | `Constant` | Invariant characters |
| | `Gapped` | Characters with a gap for at least one taxon |
| | `Missambig` | Characters with a gap or ambiguous character for at least one taxon |
| | `Allmissing` | Characters with a gap for all the taxa |
| | `Remainder` | Characters not previously referenced in the command |
| | `Uninf` | Characters that are constant as well as unique to a single sequence (autapomorphic) |
| DNA, RNA, and Nucleotide[a] | `Pos1` | Characters defined by current `CodonPosSet` as first positions |
| | `Pos2` | Characters defined by current `CodonPosSet` as second positions |
| | `Pos3` | Characters defined by current `CodonPosSet` as third positions |
| | `Noncoding` | Characters defined by current `CodonPosSet` as non-protein-coding sites |

[a]Only if a `Codons` block containing a `CodonPosSet` is supplied (see PAUP* command reference documentation; *http://paup.csit.fsu.edu/commandref.pdf*).

**Table 6.4.3** List of Parsimony Variants Available in PAUP*

| Parsimony variants implemented in PAUP* | Descriptive name | Command-line syntax |
|---|---|---|
| Fitch (Fitch, 1971) | Unordered | `unord` |
| Wagner (Kluge and Farris, 1969; Farris, 1970) | Ordered | `ord` |
| Camin-Sokal (Camin and Sokal, 1965) | Irreversible (normal or reversed) | `irrev.up` or `irrev.dn` |
| Dollo (Farris, 1977) | Dollo (normal or reversed) | `dollo.up` or `dollo.dn` |
| Generalized (Sankoff, 1975) | User-defined | `usertype` |

*Background Information). Therefore, without a priori information regarding the underlying properties of the sequences, it may be best to use equal weights for all sites.*

*In other situations, one may prefer to give greater weight to transformations at specific sites. For example, the data to be analyzed in this example are composed of protein-coding sequences. Because transformations at second-codon positions are more highly conserved, transformations at these positions will be given greater weight relative to transformations at first- and third-codon positions. In the example here, weighting second-position codons is relatively simple since codon positions have already been identified in the sample file using the* `charset` *command (Fig. 6.4.5). Another way to assign weights to a set of sites is to explicitly list the site number following each weight. For example the command:*

```
weights 3: 2 5 8 11, 2:1 4 7;
```

*will weight transformations at the nucleotide positions 2, 5, 8, and 11 by a factor of three and positions 1, 4, and 7 by a factor of two.*

7. Consider setting character types.

```
ctype 2_1:all;
```

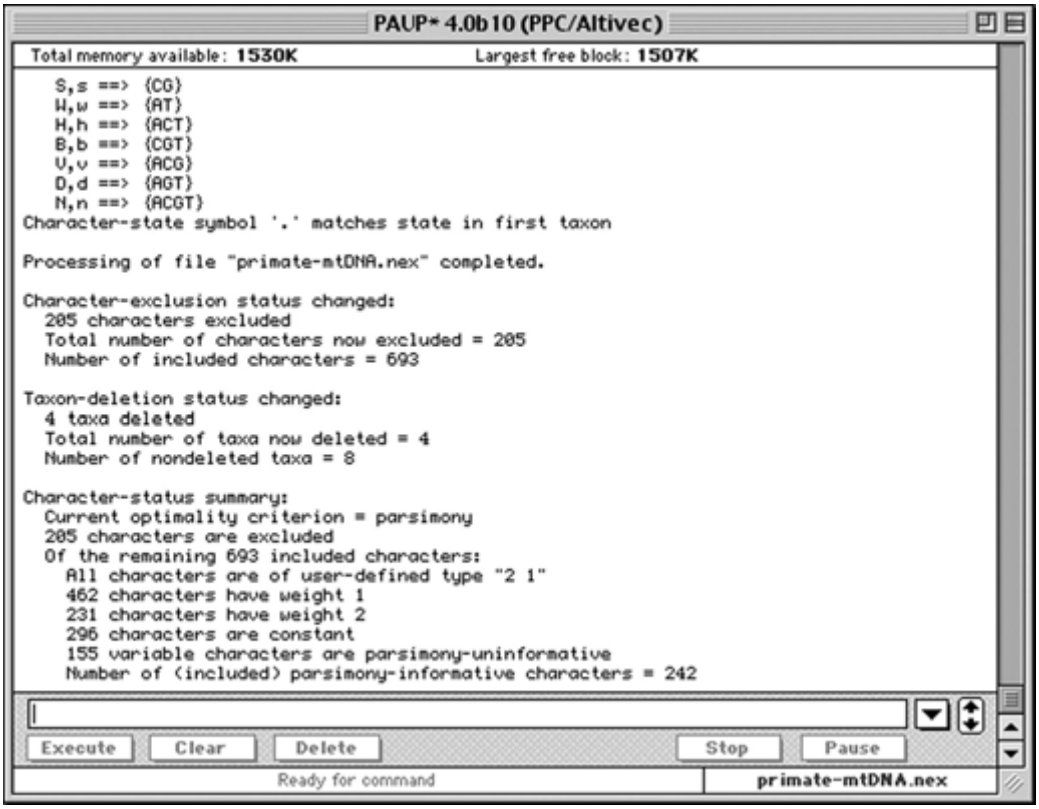


**Figure 6.4.6** Character-status summary after several data management operations.

*PAUP* allows the user to explore different parsimony variants by assigning user-defined character types (Table 6.4.3) for each character included in the data matrix. By default, PAUP* assumes that all character states are unordered and that the transformation cost from one state (nucleotide) to any other state is equal to one. For the same reasons described in step 6, the default settings may be justified, in which case the user can simply skip this step altogether. For this example, a user-defined character type will instead be declared that assigns a higher cost to transversions (changes between a purine and a pyrimidine) than to transitions (changes between two purines or changes between two pyrimidines). More specifically, the user-defined character type is set up so that the cost of transversions is twice that of transitions. A step matrix that defines this relationship in a way that PAUP* can understand is already given in the data file (Fig. 6.4.5). The command given in this step forces PAUP* to apply the transformation cost to all of the characters currently included in the analysis.*

8. Check the current character settings.

```
cstatus;
```

*Before going on to the next set of steps ("Search for a tree") it is generally good practice to check the status of the characters to be included in the search. Searching for an optimal tree can be a time-consuming operation, so a little extra caution at this stage of the analysis can save time in the long run. At the bottom of the display window shown in Figure 6.4.6, a summary of the current character status is given. The output shows that the optimality criterion is set to parsimony (step 5), 205 characters included in the input matrix will not be used in the analysis (the noncoding regions excluded in step 3), all of the remaining characters are assigned the user-defined type* `2 1` *(step 7), and transformations at 231 character sites (i.e., second codon positions) will be weighted twice that of the others (step 6). The remaining information summarizes several qualities of the data set after the*

*assumptions have been set. A more detailed summary for each character in the data matrix can be obtained by using the* `cstatus` *command and turning the* `full` *option on. For example:*

```
cstatus full;
```

***Search for a tree***

9. Select a search strategy.

   *PAUP* provides two general classes of methods for searching for optimal trees; exact and heuristic. Exact methods guarantee that the optimal tree(s) will be found, but may require prohibitive amounts of computer time for data sets composed of more than 11 to 20 sequences. On the other hand, while not guaranteeing that the optimal tree will be found, heuristic methods require far less computer time. Because many data sets compiled today are composed of more than 20 sequences, this example will focus on the options needed to conduct a heuristic search. Users interested in conducting an exact search should review the* `alltrees` *and* `bandb` *commands described in the current PAUP* command reference documentation.*

   *In addition to criterion-based methods for reconstructing evolutionary trees, PAUP* can implement neighbor joining and UPGMA clustering algorithms. These methods have the advantage of being extremely fast even for very large data sets (e.g., 100 or more sequences). However, a drawback of these methods is that they do not explicitly attempt to optimize any particular objective function and therefore do not allow comparisons of preselected trees or searches for sets of trees that include both optimal and near-optimal trees. Since neighbor joining and UPGMA are discussed in UNIT 6.3, these methods will not be discussed here, except to say that more information regarding these methods can be found in the current PAUP* command reference documentation (see commands* `upgma` *and* `nj`*).*

10. Select the heuristic search options.

```
hsearch ?;
```

   *Typing a command name followed by a question mark will generate a complete list of the options available under that command. While the list of available options under the "*`hsearch`*" command is extensive (Fig. 6.4.7), depending on the data set being analyzed, it might only be necessary to alter the default settings for a small set of the available options. This example will focus entirely on three of the options listed in Figure 6.4.7—*`start`*, *`addseq`*, and* `swap`*. Information regarding* `hsearch` *options not discussed in this protocol can be found in the PAUP* command reference documentation.*

   *In general terms, a heuristic search in PAUP* can be divided into two stages. The first stage involves getting a starting tree. The main options that deal with this stage of the search are* `start` *and* `addseq`*. By default, PAUP* uses the stepwise-addition algorithm (e.g., Farris, 1970) to get a starting tree. Other ways to get a starting tree include the neighbor-joining algorithm (*`start=nj`*; UNIT 6.3) and simply selecting one or more of the trees currently in memory (*`start=current` *or* `start=`*a range of tree numbers). For this example, the stepwise-addition option (the default) will be used to get a starting tree. The stepwise-addition option requires that the order in which sequences are joined together be specified, to construct a starting tree. PAUP* implements five addition sequence algorithms. Unfortunately, no one algorithm works best for all data sets (see Swofford et al., 1996). In this example, the random addition sequence algorithm will be used (see step 11) to construct multiple starting trees (the default is ten), each of which is then passed on to the second stage of the heuristic search, branch swapping.*

   *In simple terms, the second stage attempts to find better trees by evaluating rearrangements of the starting tree. PAUP* implements three branch-swapping algorithms listed here in ascending order of effectiveness: nearest-neighbor interchanges (option* `nni`*), subtree pruning-regrafting (option* `spr`*), and tree bisection and reconnection (option* `tbr`*). As might be expected, effectiveness comes at the cost of increased search effort. For this example, branches will be swapped using the subtree pruning-regrafting algorithm. For a*

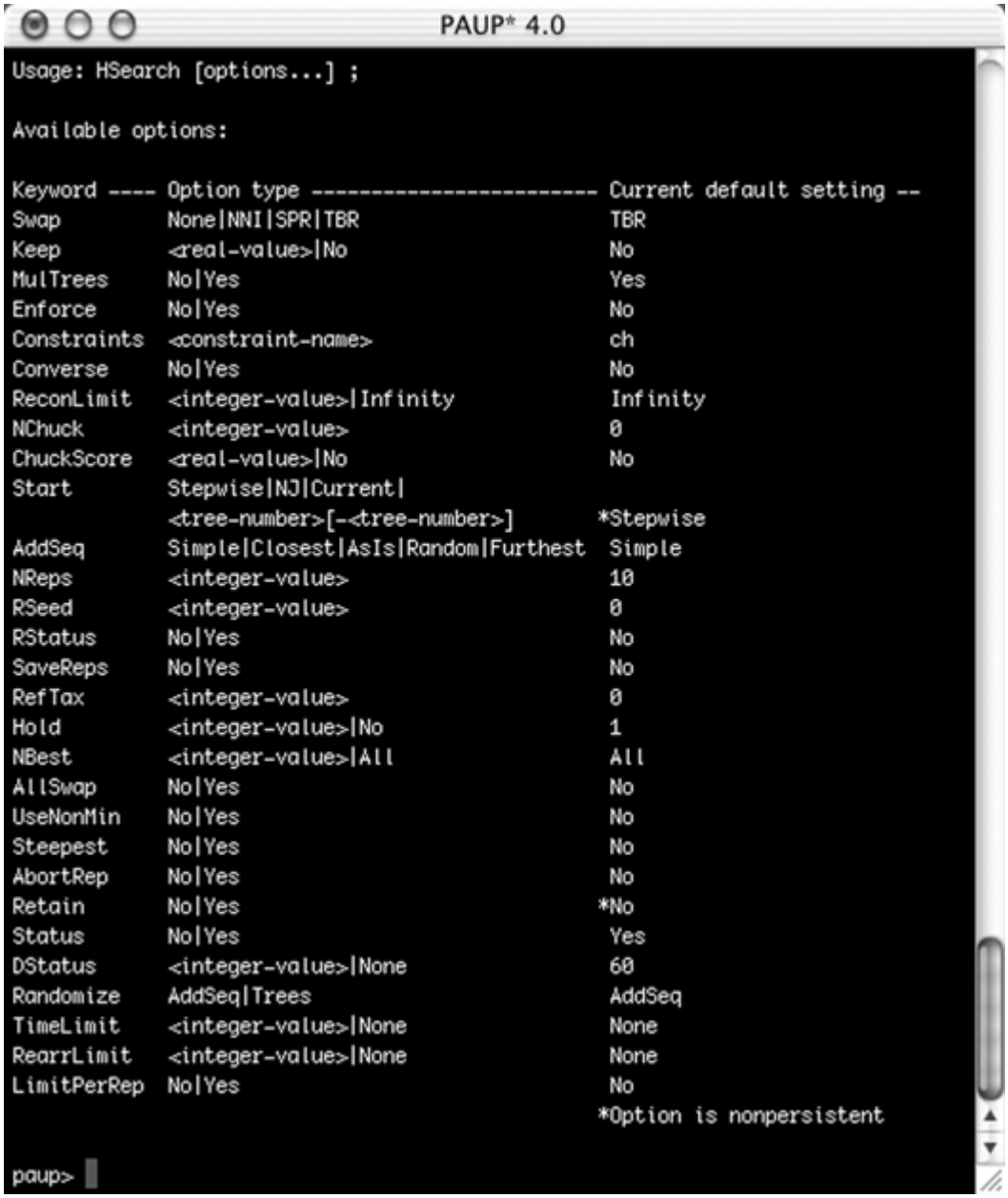


**Figure 6.4.7** A summary of the available options under the heuristic search command and the current default setting of each option.

*complete discussion of the branch-swapping algorithms used by PAUP*, see Swofford et al. (1996).*

11. Start the heuristic search.

```
hsearch start=stepwise addseq=random swap=spr;
```

Once the search is started, PAUP* will again display general information about the status of the characters and the options used for the heuristic search. When the search completes, PAUP* will display general information about the results of the search (Fig. 6.4.8). If logging was started before executing the search (e.g., `log start file=mylogfile.log;`), a record of the information output to the display

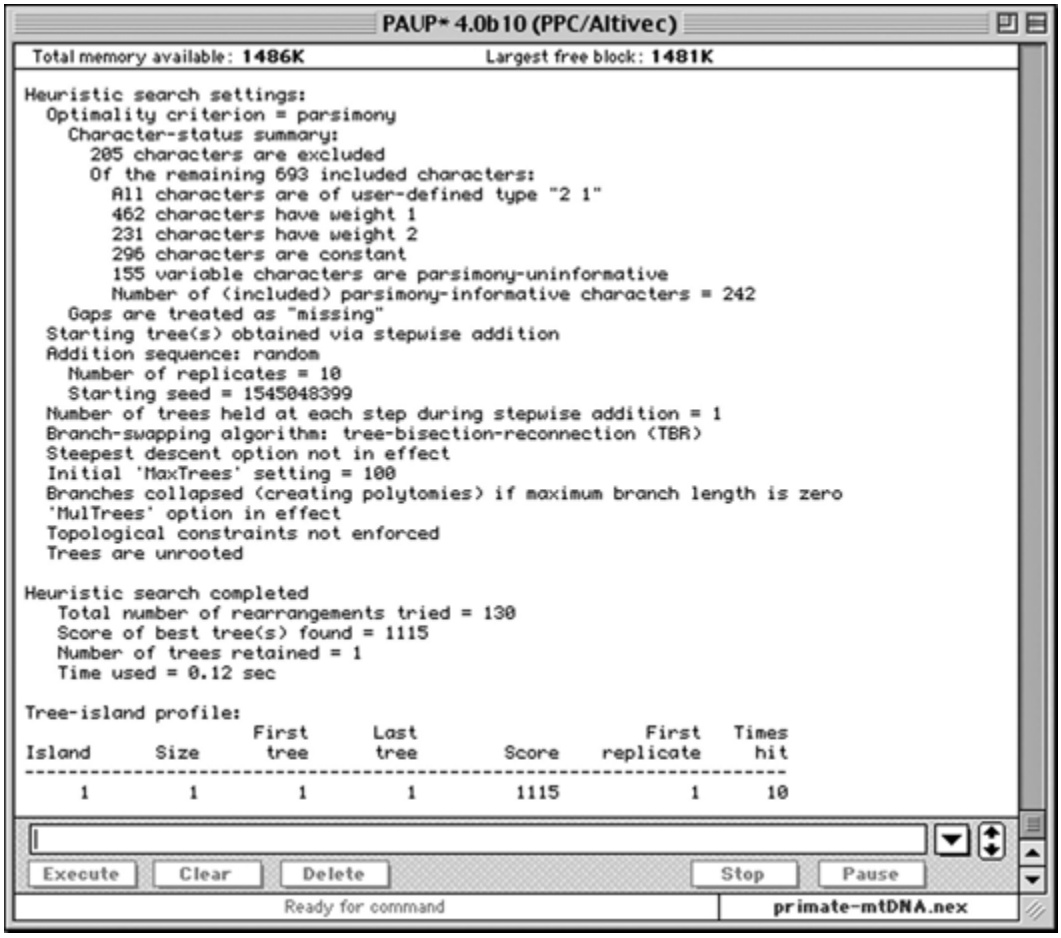


**Figure 6.4.8** The summary display of the random addition heuristic search.

window would be saved to a file. Likewise, logging can be turned off at any point in the analysis (e.g., `log stop;`).

*In some cases, multiple trees of the same length may be recovered by a search. In this case, however, a single tree from a single island with a length of 1115 is found. Islands are meant to represent sets of optimal trees separated by "seas" of suboptimal trees. To get from one island to the next, one must pass through at least one suboptimal tree (see Swofford et al., 1996; Page and Holmes, 1998).*

*Up to this point no pictures of trees have been printed to the display. In the next set of steps, procedures to display and print the tree found by the heuristic search will be covered.*

### *Print and save trees*

12. Define the outgroup sequences.

```
outgroup lemur_catta macaca_fuscata saimiri_sciureus;
```

*Ordinarily, PAUP* stores the trees found by a search, such as the one just conducted, as unrooted trees. These trees are displayed as rooted trees in the output listing using the default method of outgroup rooting. If no sequences have been assigned to the outgroup, by default, PAUP* will use the first sequence in the matrix as the outgroup. In this example, all three non-hominoids will be defined to the outgroup list. In this case, the* `outgroup` *command is used followed by the three outgroup taxa labels. For a discussion of how to select an outgroup see UNIT 6.1.*

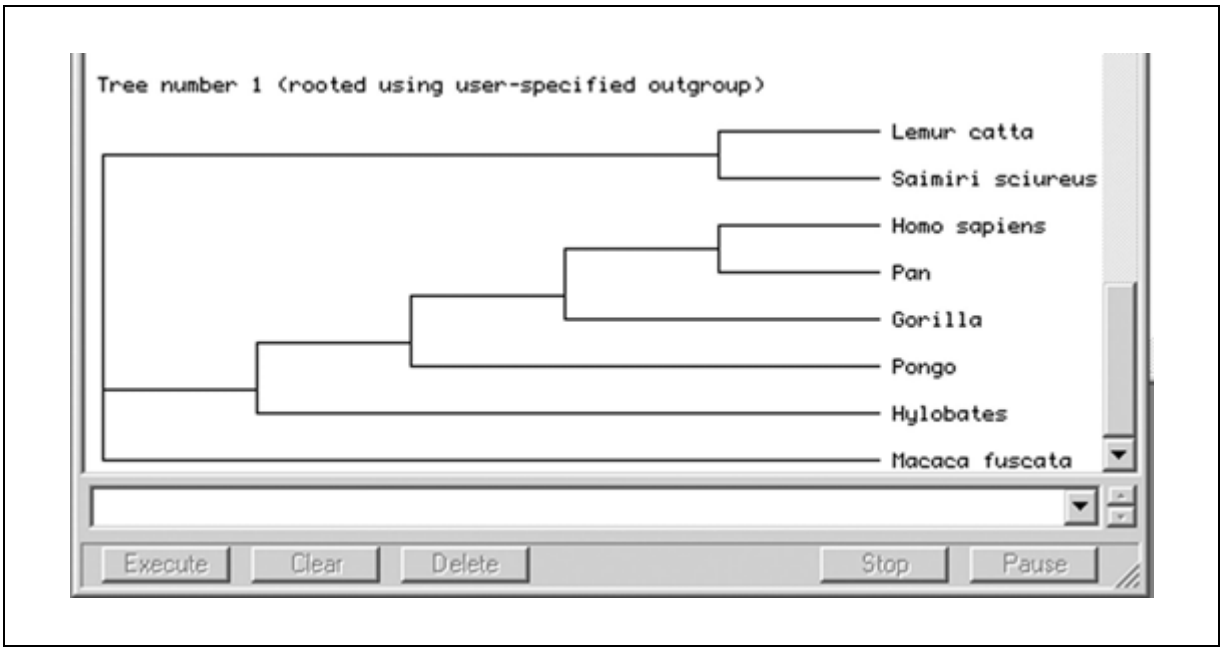


**Figure 6.4.9** A simple diagram of the tree found by the random addition sequence heuristic search.

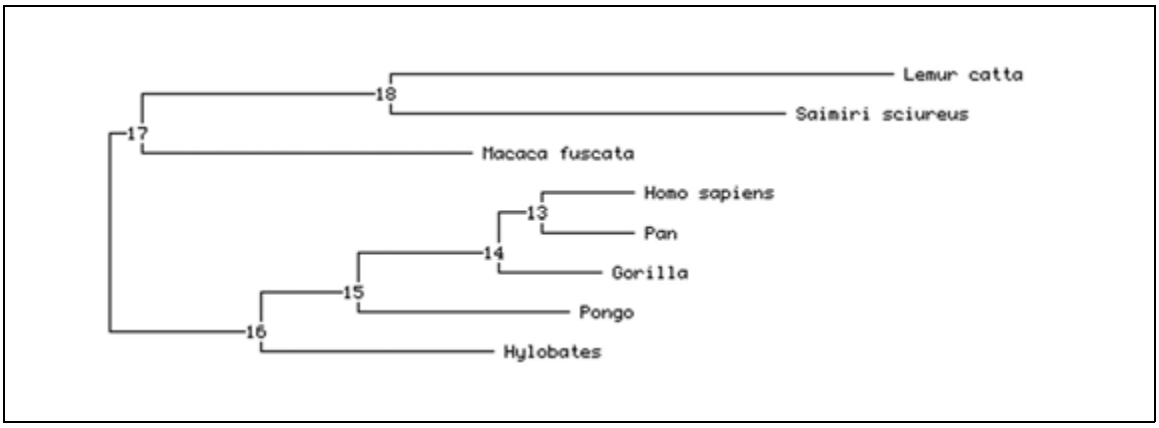


**Figure 6.4.10** Phylogram of the single most parsimonious tree.

13. Display the tree found by the search.

```
showtrees all;
```

*The simplest way to display a picture of the tree recovered by the preceding heuristic search is to use the* `showtrees` *command. This command will display a basic diagram depicting the branching order of the sequences (Fig. 6.4.9). To get more information about the tree recovered by the search (e.g., branch lengths, estimates of ancestral character states, etc.) use the* `describetrees` *command (see step 14).*

14. Display a phylogram with a list of branch lengths.

```
describetrees /plot=phylogram brlens=yes
rootmethod=outgroup outroot=monophyl;
```

*The syntax given above tells PAUP* to draw the maximum-parsimony tree again, but this time as a phylogram. In this case, the tree branch lengths are proportional to the amount of implied evolutionary change and the outgroup taxa are shown as a monophyletic sister group with respect to the ingroup sequences (Fig. 6.4.10). The command also asks PAUP* to output a complete list of branch lengths including the minimum and maximum possible*

Branch lengths and linkages for tree #1 (unrooted)

| Node | Connected to node | Assigned branch length | Minimum possible length | Maximum possible length |
|---|---|---|---|---|
| 18 | 17 | 103 | 58 | 116 |
| Lemur catta (1)* | 18 | 211 | 166 | 247 |
| Saimiri sciureus (11)* | 18 | 165 | 134 | 210 |
| Macaca fuscata (7)* | 17 | 136 | 92 | 175 |
| 17 | 17 | 79 | 46 | 113 |
| 15 | 16 | 44 | 27 | 64 |
| 14 | 15 | 56 | 30 | 70 |
| 13 | 14 | 18 | 12 | 24 |
| Homo sapiens (2) | 13 | 38 | 28 | 44 |
| Pan (3) | 13 | 40 | 34 | 50 |
| Gorilla (4) | 14 | 41 | 35 | 63 |
| Pongo (5) | 15 | 87 | 73 | 107 |
| Hylobates (6) | 16 | 97 | 74 | 130 |
| Sum | | 1115 | | |

**Figure 6.4.11** The table of branches and linkages output by the `describetrees` command.

*lengths of each branch over all of the most parsimonious reconstructions (see Swofford and Maddison, 1987; also see Fig. 6.4.11). Many more options are available under the* `describetrees` *command; rather than discuss these options here, users interested in this topic should consult the PAUP* command reference documentation.*

15. Save trees and print a high-resolution tree.

```
savetrees file=parsTree.tre brlens=yes;
```

*The Macintosh version is currently the only version of PAUP* that supports printing of high-resolution trees (using the Print Trees menu command). High-resolution pictures of the tree found by the PAUP* search can still be obtained for the Windows or the Portable versions of PAUP*; however, a couple of extra steps are required. In short, the tree must be saved to a NEXUS-formatted tree file (the default) and then opened using a third-party tree-printing software package such as the program TreeView by Rod Page. Also see UNIT 6.2, where the software package TreeView is discussed in detail.*

16. Exit the program.

```
quit;
```

## USING PAUP* TO INFER A MAXIMUM-LIKELIHOOD TREE FROM DNA SEQUENCES

***ALTERNATE PROTOCOL***

In addition to the parsimony criterion described above, PAUP* can be used to infer evolutionary trees using several maximum-likelihood models of DNA substitution (see Background Information). PAUP*, however, does not provide maximum-likelihood models for amino acid substitution. *UNIT 6.1* gives a general discussion of the maximum-likelihood method and a more detailed discussion of the subject can be found in Swofford et al. (1996). In short, the maximum-likelihood method selects the hypothesis that maximizes the probability of obtaining the observed data. In the context of phylogenetic inference, the hypothesis is the tree (i.e., the branching order of the sequences as well as the branch lengths) and the observed data are the DNA sequences. The hypotheses (i.e., trees) are evaluated under a proposed model of DNA substitution. Of course, as the number of sequences in a data matrix increases, so too does the number of possible

hypotheses that need to be evaluated. This problem is shared by all criterion-based methods; however, with maximum likelihood, scoring a single tree can be very time-consuming. In the past, this practical constraint discouraged some people from using maximum likelihood. Fortunately, advances made in computational processing power, coupled with algorithmic improvements, have made it much more practical to used this method. For example, it is now not uncommon to see 50 or more sequences being analyzed in maximum-likelihood searches.

This protocol describes one way to implement a maximum-likelihood tree search using PAUP* in practical terms. The process can be divided into three parts: (1) getting a tree; (2) selecting a model of DNA substitution; and (3) searching for the best (optimal) tree under the selected model. Essentially, steps 1 and 3 were covered in the Basic Protocol. As such, the focus will be mostly on describing how to select a model of DNA substitution that can be used in a subsequent search.

The authors also wish to make it clear that there are many ways to select an appropriate model of DNA substitution for a set of aligned sequences. Most of these methods attempt to select a model that optimizes a tradeoff between fit to the data and model complexity. In practice, methods used to select a model of sequence evolution range from fully automated—e.g., ModelTest (Posada and Crandall, 1998; *UNIT 6.5*)—to fully interactive. For this example, the model selection part is demonstrated using the fully interactive approach. This approach is suggested for two reasons. First, by walking step-by-step through the model-selection process it is hoped that the reader will become acquainted with some of the options available in PAUP* for implementing likelihood-based analyses. Second, by selecting a model interactively it is often possible to learn about specific aspects of sequence evolution that might otherwise go unnoticed.

### *Necessary Resources*

*Hardware*

PAUP* is compatible with most modern hardware configurations including Apple Macintosh (PPC and 68k-based processors), Intel, AMD, and a number of Unix and Linux workstations (e.g., Sun, Alpha, IBM, SGI, PPC-based, i386-based, and others). Note that maximum-likelihood analyses are generally more computationally demanding than parsimony or distance analyses. As such, maximum-likelihood analyses run on older hardware equipped with slower CPUs will in some cases run prohibitively slowly. In general, however, most Pentiums, PowerPCs, and Unix workstations will be able to complete a maximum-likelihood search in a "reasonable" amount of time.

*Software*

PAUP* is distributed by Sinauer Associates at *http://www.sinauer.com*. Three versions of the program are available: Macintosh, Windows, and Portable (see Table 6.4.1). These versions provide identical analytical capabilities, but differ in the way these analyses are controlled. The Macintosh version supports a full graphical user interface, which allows the user to execute commands via menus and the command line, while the Windows and Portable versions are almost entirely command-line driven. Some menu functions are available in the Windows version; however, the functions are mostly restricted to file "open" and "edit" operations. Because the command-line interface is available among all three versions, it is used for the following examples. The location of the corresponding menu item is given in the PAUP* command reference documentation (available on the PAUP* 4.0 distribution disk and on the PAUP* Web site at *http://paup.csit.fsu.edu/commandref.pdf*). In addition, several sources already offer an overview of PAUP* using the Macintosh menu

interface (Hall, 2001; PAUP* quick-start tutorial, *http://paup.csit.fsu.edu/quickstart.pdf*). Installation instructions are included with the software.

*Files*

Data files for PAUP* are standard text files, which adhere to the NEXUS file specifications. The NEXUS format was designed by Maddison et al. (1997) to facilitate the interchange of input files between programs used in phylogeny and classification. The text in a NEXUS file is arranged into blocks, which are delimited by the words `begin` and `end`. The text immediately following the word `begin` defines the block type. For example, the `TAXA` and `CHARACTERS` blocks shown in Figure 6.4.1 define a simple data set composed of four taxa (sequences) and 60 nucleotide characters (sites). PAUP* supports several predefined data types: DNA, RNA, nucleotide (DNA or RNA), protein, and standard. The set of predefined character-state symbols are `ACGT` for the data type DNA, `ACGU` for RNA, the standard one-letter amino acid codes for protein, and `01` for standard. In addition, standard ambiguity codes for the molecular data types are implemented by predefined "equate" macros. Additional character states other than those represented by the predefined data types can be specified using the `SYMBOLS` subcommand (see the PAUP* documentation). PAUP* data files must start with the character string `#NEXUS`. Comments can be included in a data file by enclosing them in square brackets, as shown in Figure 6.4.1. PAUP* 4.0 is also capable of translating other file formats to the NEXUS format. Supported formats include PHYLIP (see *UNIT 6.3*), Hennig86, GCG/Pileup (see *UNIT 3.6*), MEGA, NBRF-PIR, FreqPars, and text (space and tab-delimited). The Support Protocol below details using PAUP* to import non-NEXUS data files. The NEXUS file used for this protocol can be found in the application subdirectory labeled `Sample-NEXUS-data` and also on the *Current Protocols in Bioinformatics* Web site at *http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm*.

*NOTE:* Due to formatting constraints, some commands given in the following examples span multiple lines. However, when entering commands at the `paup>` prompt or into the command-line interface, all of the text preceding the semicolon should be entered on the same line.

***Execute the data file and get a tree quickly***

1. Start PAUP* and execute the data file.

```
execute primate-mtDNA.nex;
```

*See Basic Protocol, step 1, for a description of how to start the PAUP* application. Users who have already worked through the procedures described in the Basic Protocol should quit PAUP* before starting this example (e.g., type* `quit`*). Restarting the program will ensure that the option settings are set to the factory default and that all of the characters and taxa included in the sample file will also be included in the subsequent example analysis. An alternative to quitting the program is to manually reset the PAUP* options to their factory default values and restore character weights, the deleted taxa, and excluded characters (sites). For example:*

```
include all;
undelete all;
wts 1:all;
factory;
```

2. Get a tree quickly.

```
nj;
```

*As discussed in UNIT 6.3, the neighbor-joining method is generally a good way to get a "rough" estimate of the phylogeny very quickly. Submitting the command* `nj` *tells PAUP* to construct a neighbor-joining tree using the set of pairwise "p-distances" (the default distance for nucleotide characters). A p-distance, also known as a dissimilarity distance, is simply the total number of observed differences between a pair of sequences divided by the total number of sites compared. The p-distance is one of many distance measures implemented in PAUP*. Use the* `dset` *command to switch to another distance (e.g., perhaps one that accounts for superimposed substitutions). For example, the following command changes the pairwise distances calculated by PAUP* to the log-determinant distance (Lockhart et al., 1994; Steel, 1994).*

```
dset distance = logdet;
```

*In this example, the tree topologies obtained by using the uncorrected and corrected distances are the same. Another way to get a tree quickly is to read it from a NEXUS-formatted tree file. For example, the following syntax will replace all trees currently in PAUP* memory with the tree or trees contained in the file.*

```
gettrees file=myfavorite.tre mode=3;
```

*To add a tree or trees from a file to the tree or trees currently in memory, change the* `mode` *option to* `7` *(see the PAUP* documentation for a complete description of the* `gettrees` *options).*

***Select an appropriate model of DNA substitution***

3. Select a starting model of DNA substitution.

```
lscores 1/nst=6 rmatrix=estimate basefreq=estimate
rates=gamma shape=estimate pinvar=estimate;
```

*Start by evaluating the likelihood of the neighbor-joining tree currently in memory using the most parameter-rich DNA substitution model available in PAUP*—i.e., the General Time Reversible model (see e.g., Yang, 1994a) with six substitution types, base composition estimated from the data, and among-site rate heterogeneity estimated using the invariable sites plus gamma model (GTR + I + Γ model; Gu et al., 1995; Waddell and Penny, 1996). The* `nst=6` *and* `rmatrix=estimate` *options tell PAUP* to estimate the rate of nucleotide substitution separately for six substitution types (A ↔ C, A ↔ G, A ↔ T, C ↔ G, C ↔ T, G ↔ T). By default PAUP* does not assume that the nucleotide composition is equal; rather, base frequencies are set to the value observed in the alignment. For this example, however, the authors have chosen to estimate base composition using maximum likelihood (e.g.,* `basefreq=estimate`*). Finally, the remaining three options (i.e.,* `rates=gamma shape=estimate pinvar=estimate`*) relax the assumption that the rate of substitution among sites is equal.*

*After the* `lscores` *command is executed, it may take several minutes, depending on the speed of the computer, for PAUP* to complete the parameter optimizations. When the calculation is complete, look at the output in the display window (Fig. 6.4.12). One obvious parameter that can be eliminated from the estimation procedure is* `pinvar` *(i.e., proportion of invariable sites). The value of* `pinvar` *is equal to zero, suggesting that the gamma-distributed-rates model alone may accommodate the among-site rate heterogeneity. Furthermore, eliminating the* `pinvar` *parameter from the initial model will reduce the number of free parameters without changing the likelihood of the model. In the next iteration of the* `lscores` *command, an attempt will be made to simplify the DNA substitution model even further by assuming that all sites are changing at the same rate.*

4. Reduce the number of parameters that PAUP* needs to estimate (i.e., restrict the model).

```
lscores 1/nst=6 rmatrix=estimate basefreq=estimate
rates=equal pinvar=0;
```


**Figure 6.4.12** Model parameters estimated on the neighbor-joining tree under the General Time Reversible model (see Yang, 1994a) with among site rate heterogeneity estimated using the invariable sites plus gamma model (Gu et al., 1995; Waddell and Penny, 1996).

*The main display window (Fig. 6.4.13) shows the new parameter estimates and the log-likelihood score for the restricted substitution model. The restricted model score is roughly 225 log-likelihood units worse then the more general model—i.e., the model that allows rates to vary among nucleotide positions (see step 3). A likelihood-ratio test (Cox and Hinkley, 1974) could be used to determine whether the restricted model significantly decreases the model fit. However, in this case the magnitude of the difference is so large that among-site rate heterogeneity obviously improves the overall model fit. This result should not be terribly surprising considering that the data set is composed of nucleotide sites from coding and noncoding regions, which are clearly under different structural and functional constraints.*

*Returning to the GTR + I + Γ model output (Fig. 6.4.12) it appears that it might be possible to capture the major properties of the substitution process without having to estimate six separate rate categories. That is, in the next iteration of the* `lscores` *command substitution rates will only be estimated for transitions (A ↔ G and C ↔ T) and transversion (A ↔ C, A ↔ T, C ↔ G, G ↔ T) and the parameters needed to estimate among site rate heterogeneity using the gamma-distributed rates model will be restored.*

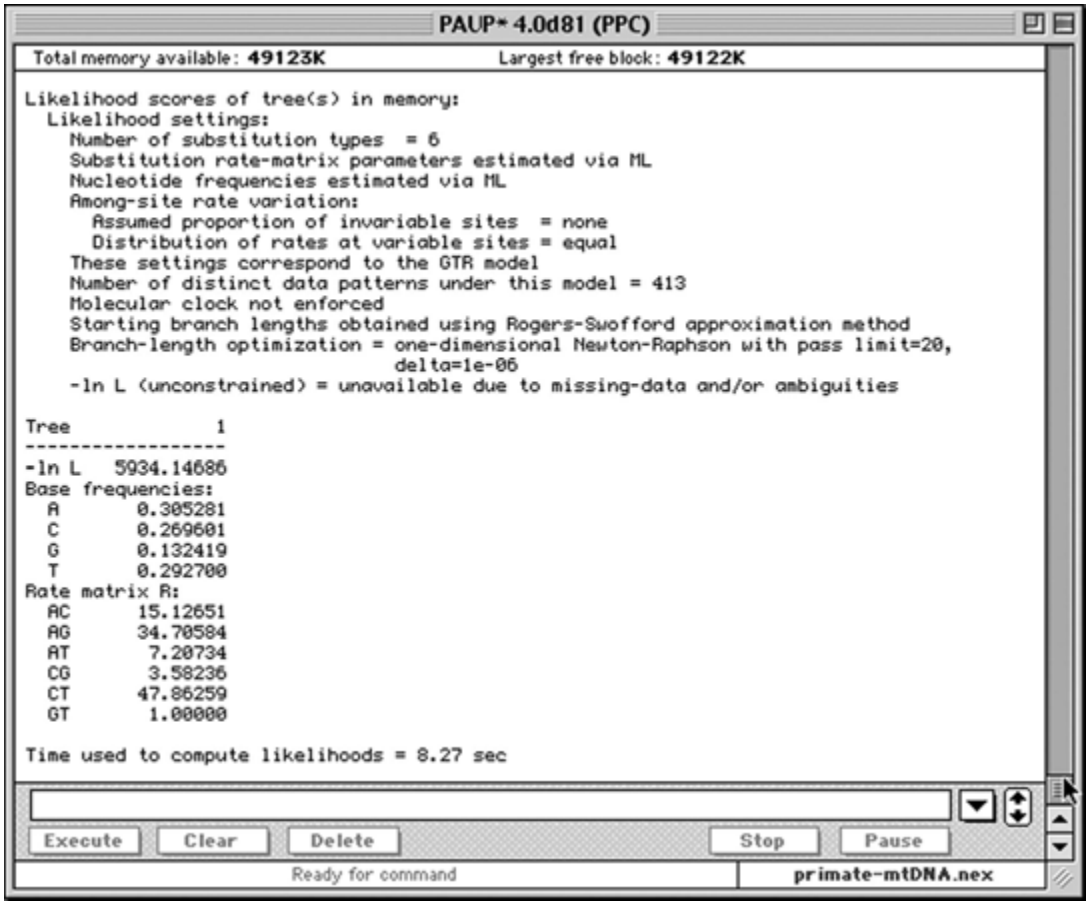


**Figure 6.4.13** Model parameters estimated on the neighbor-joining tree under the General Time Reversible model minus among site rate heterogeneity.

5. Further refine the model.

```
lscores 1/nst=2 tratio=estimate basefreq=estimate
rates=gamma shape=estimate;
```

*The model specified in this iteration of the* `lscores` *command is generally referred to as the HKY + Γ model (Hasegawa et al., 1985; Yang, 1994b). In this case, the restricted model (HKY + Γ) only differs from the more general model (GTR + Γ) by 3 log likelihood units (Fig. 6.4.14). The removal of four substitution classes seems to have a relatively small impact on the overall model fit, especially when compared to the difference achieved when the single among-site rate heterogeneity parameter was removed. Furthermore, according to a likelihood ratio test, the more parameter-rich GTR model does not significantly improve the model fit (P = 0.19776). The test statistic for the likelihood-ratio test is equal to two times the difference between the log-likelihood scores [e.g., 2(5711.94 – 5708.92)] and it is chi-square distributed with degrees of freedom equal to the difference between the number of free parameters used by each substitution model [e.g., 10(GTR + Γ) – 6(HKY + Γ)]. Therefore, with a critical value of 0.05 we reject the more complicated GTR + Γ model and will use the model and the model parameters estimated in this step in the forthcoming heuristic search.*

***Search for an optimal tree using the selected model***

6. Set the optimality criterion to Likelihood.

```
set criterion=likelihood;
```

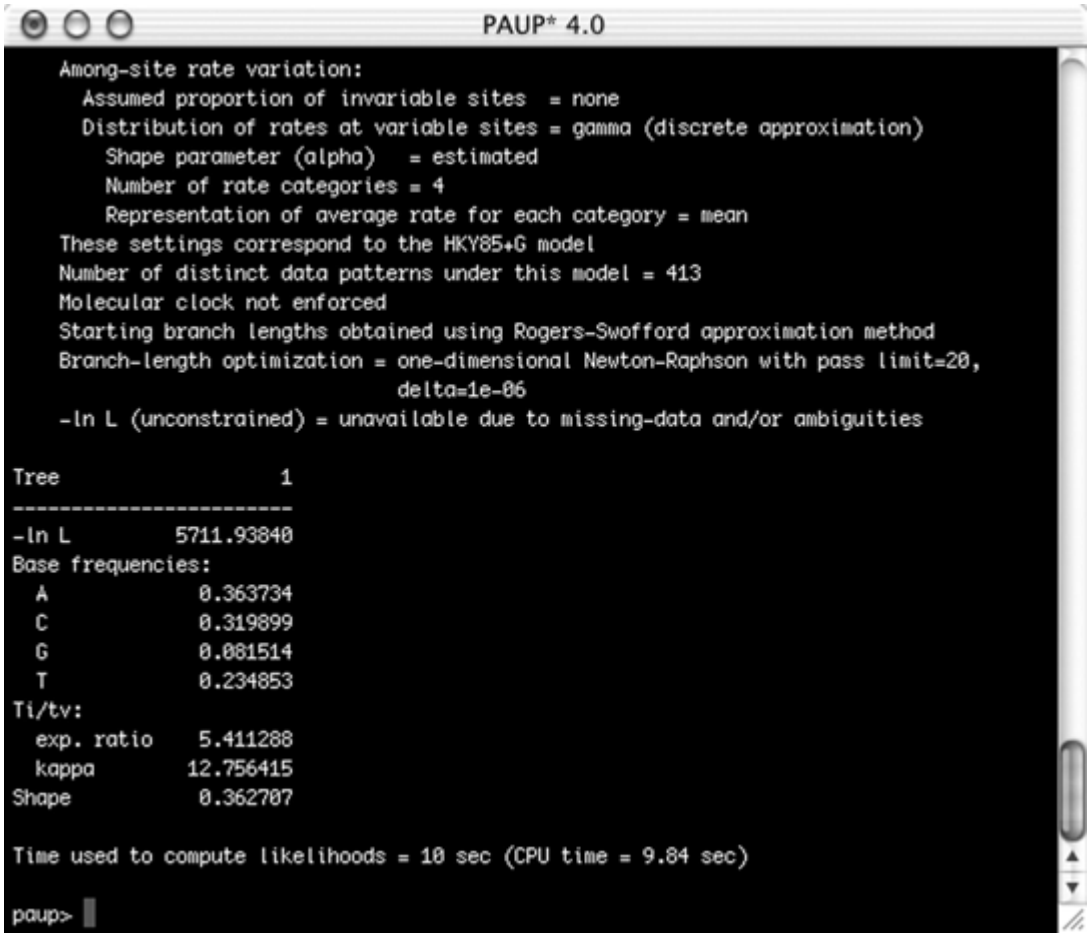


**Figure 6.4.14** Model parameters estimated on the neighbor-joining tree under the HKY + Γ model plus among the gamma distribute rates.

*By default, the optimality criterion is set to maximum parsimony. The* `set` *command is used to change the criterion so that all subsequence searches will use the selected maximum-likelihood model.*

7. Set the maximum-likelihood model parameters.

```
lset nst=2 rmatrix=previous basefreq=previous
rates=gamma shape=previous;
```

*This command is a shortcut that takes the parameters estimated by the* `lscores` *command in step 5 and fixes them for subsequent likelihood operations. Alternatively, the likelihood parameters can be set manually by copying the parameter values into an* `lset` *command. For example:*

```
lset nst=2 tratio=5.411288 basefreq=(0.36734 0.319899
0.081514) shape=0.362707;
```

*The parameter values listed above were taken from the likelihood scores output for the HKY85 + Γ model (Fig. 6.4.14).*

8. Run a heuristic search using the maximum-likelihood model.

```
hsearch;
```

*The same set of heuristic search options (see Fig. 6.4.7) is available for each of the three optimality criteria implemented by PAUP* (i.e., parsimony, maximum likelihood, and least-squares distances). The reader should refer to steps 9 through 10 in the Basic Protocol for more information regarding this step. After the search finishes, the maximum-likelihood tree can be displayed by following steps 12 through 15 of the Basic Protocol.*

9. If the topology found by the preceding heuristic search differs from the starting tree, it is possible to further refine the maximum-likelihood parameter estimates (and associated optimal tree) by repeating steps 3 to 7 above until the topology does not change (see Swofford et al., 1996). In this example, however, the topologies did not differ so the protocol will stop here.

   *The optimal tree is a function of both data and parameters, so if one changes parameters one may also change the tree. Given this relationship, ideally, one would estimate model parameters for all of the trees found during a heuristic search.For most data sets, however, such a search would be prohibitively time-consuming. Instead, the successive-approximations approach (Swofford et al., 1996) described above is an effective way to implement a model-based tree search and also arrive at a stable set of model parameters.*

***SUPPORT PROTOCOL***

## USING PAUP* TO IMPORT NON-NEXUS DATA FILES

Although a number of computer packages save multiple sequence alignments to the native PAUP* file format, situations may arise where it is necessary to analyze a data set that is not in the NEXUS format. To save the user from the trouble of having to reformat the non-NEXUS file manually, PAUP* provides a command called `tonexus` that can be used to translate several popular data file formats. The file formats that can be translated to the NEXUS format are: GCG/MSF, NBRF-PIR, PHYLIP, MEGA, FreqPars, and Text (tab or space delimited). Examples of all these data formats can be found on the PAUP* Web site at *http://www.paup.csit.fsu.edu/nfiles.html*. Getting a non-NEXUS file to the stage were it is ready to be analyzed requires two general steps: (1) converting and saving the non-NEXUS file to a NEXUS file and (2) executing the newly converted NEXUS file. An example of how to do this is given below.

### *Necessary Resources*

*Hardware*

PAUP* is compatible with most modern hardware configurations including Apple Macintosh (PPC and 68k-based processors), Intel, AMD, and a number of Unix and Linux workstations (e.g., Sun, Alpha, IBM, SGI, PPC-based, i386-based, and others).

*Software*

PAUP* is distributed by Sinauer Associates at *http://www.sinauer.com*. Three versions of the program are available: Macintosh, Windows, and Portable (see Table 6.4.1). These versions provide identical analytical capabilities, but differ in the way these analyses are controlled. The Macintosh version supports a full graphical user interface, which allows the user to execute commands via menus and the command line, while the Windows and Portable versions are almost entirely command-line driven. Some menu functions are available in the Windows version; however, the functions are mostly restricted to file "open" and "edit" operations. Because the command-line interface is available among all three versions, it is used for the following examples. The location of the corresponding menu item is given in the PAUP* command reference documentation (available on the PAUP* 4.0 distribution disk and on the PAUP* Web site at *http://paup.csit.fsu.edu/commandref.pdf*). In addition, several sources already offer an overview of PAUP* using the Macintosh menu interface (Hall, 2001; PAUP* quick-start tutorial,

*http://paup.csit.fsu.edu/downl.html*). Installation instructions are included with the software.

*Files*

A sample PHYLIP file will be used in the following example. The file can be found in the PAUP* application subdirectory labeled `Sample import data` and also on the *Current Protocols in Bioinformatics* Web site at *http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm*.

1. Start PAUP*.

   *Refer to the PAUP* application documentation for a description of how to launch the program. When the Macintosh or Windows version of PAUP* is first started, the program will automatically launch an Open File dialog box. Files opened via this dialog box must already be in the NEXUS format. Close the dialog box by clicking the Cancel button.*

2. Convert a PHYLIP-formatted file to NEXUS.

```
[Windows] cd '..\Sample import data';
[Portable] cd ../Sample_import_data;
tonexus format=phylip interleaved=yes
datatype=nucleotide fromfile=phylip_3x.dat
tofile=test.nex;
```

   *If the non-NEXUS file is not in the active PAUP* directory, then one must first use the* `cd` *command to move to the correct directory or supply the complete filename with the* `fromfile` *and* `tofile` *options. Because some directory names in this example include spaces, the full path must be enclosed by single quotes. Users of the Portable version of PAUP* can avoid some typing by using the tilde (~) symbol in place of the complete path to their home directory (e.g.,* `tofile=~/mydatafiles/paupfiles.nex`*). The options given under the* `tonexus` *command are mostly self-explanatory. The* `interleaved` *option tells PAUP* to expect the data in the PHYLIP file to be arranged into blocks. By default, PAUP* assumes that the data are nucleotide characters, so the* `datatype` *option need not be explicitly stated.*

3. Execute the converted file.

```
execute test.nex;
```

   *It is now possible to use the tree-building and character-diagnosing features described in the Basic and Alternate Protocols.*

## GUIDELINES FOR UNDERSTANDING RESULTS

The heuristic search procedure discussed above attempts to find optimal trees according to the currently selected criterion. Additionally, it will usually be of interest to determine the strength of support for the implied groupings. A common way to assess the support for the nodes of a tree topology under a specified reconstruction method is to used a nonparametric bootstrap (Felsenstein, 1985) or jackknife analysis (Penny and Hendy, 1985; Farris et al., 1996). These methods are discussed in *UNIT 6.1* so it will only be mentioned that both of these analyses are implemented in PAUP* and can be executed using the `bootstrap` or `jackknife` commands, respectively. As with the other PAUP* commands discussed in this unit, a brief description of the options available under each of these commands can be found in the current PAUP* command reference documentation available on the PAUP* Web site.

In some cases, one may be interested in knowing how the complete optimal tree compares with other candidate trees. The relative differences among the scores of candidate trees under the various optimality criteria provide a basis for this type of comparison. However,

it is natural to want to know whether the magnitude of the difference between tree scores is great enough that a specific tree can be excluded with a certain degree of confidence. Methods designed to do this—i.e., Templeton (1983), Kishino and Hasegawa (1989), and Shimodaira and Hasegawa (1999; see also Goldman et al., 2000) are also available in PAUP* and can be implemented using the options under the `pscores` and `lscores` commands. Users who are interested in these methods should consult the PAUP* documentation. A number of other tree diagnostics are available in PAUP*; however, there is not enough space to discuss them all in this unit. Again, for those interested in this topic, the authors recommend looking at the available options under commands such as `contree`, `describetrees`, `reconstruct`, and `treedist`.

Finally, the methods just mentioned attempt to assess the reliability of trees given a specific method of analysis. As discussed below (see Commentary), however, the performance of a method is not guaranteed under all circumstances. For example, the relevance of a confidence test is severely compromised if the phylogenetic reconstruction method chosen is biased in some fundamental way. For example, the problem of long-branch attraction (see Commentary) can lead to high confidence (e.g., as measured by bootstrap support) in incorrect phylogenetic groupings. One simple way to explore the possibility that results are being influenced by a particular methodological bias is to apply several different methods of analysis, each subject to its own potential biases and limitations, to the data matrix. The observation that all of the methods give similar results provides some encouragement that the results were not adversely affected by problems specific to particular methods. If, on the other hand, different methods give different results, it is important to investigate what properties of the data set might be causing some or all of the methods to be misled.

## COMMENTARY

### Background Information

#### *Model-free and model-based approaches to phylogeny*

In a perfect world, reconstruction of phylogenies would be simple. For example, sequences would have a level of variation appropriate to the problem being addressed and the potentially misleading effects of convergence, parallelism, and reversal could safely be ignored. Unfortunately, this ideal situation is rarely realized. To cope with these realities, developers of phylogenetic methodology have taken two basic and often conflicting strategies. The first approach attempts to reconstruct the phylogeny without reference to an explicit model of evolutionary change. The second class of methods adopts a statistical approach to the problem of phylogenetic inference, relying on the use of stochastic models of the evolutionary process to assess the degree of fit between candidate trees and the observed data. The parsimony and maximum-likelihood methods discussed above are the most well known examples of "model-free" and "model-based" methods, respectively. It is important, however, to realize that neither model-free nor model-based approaches are free from specific shortcomings and limitations. Furthermore, the distinction between them is not always clear; e.g., it is possible to define parsimony as a maximum-likelihood method under an extremely general (and probably too general) substitution model (Tuffley and Steel, 1997). Other methods do not sort cleanly into one of these two categories. Distance methods can range from "model-based" to "model-free" depending on whether stochastic models are used to transform raw sequence differences into expected amounts of change according to the chosen model. In the following section, some advantages and disadvantages to of the two approaches will be highlighted and users will be provided with relevant and accessible references for further study.

#### *Parsimony*

The parsimony method selects the tree that minimizes the number of "steps" (character changes or substitutions) required to explain the observed data. The parsimony criterion is often defended on the basis of simplicity: without evidence to the contrary, sequence identities are assumed to be due to common ancestry.

When conflicts are unavoidable, the explanation that minimizes the number of ad hoc assumptions of homoplasy (similarity for reasons other than common ancestry, including convergence, parallelism, and reversal) is preferred. A number of parsimony variants can be implemented in PAUP* according to the way in which character states are allowed to change. No matter what constraints are imposed on character changes, however, the goal of all parsimony-based methods remains the same—i.e., minimization of the amount of implied evolutionary change. Unordered-character or Fitch parsimony (Fitch, 1971) assigns an equal cost to transformations from any character state to any other character state. In ordered-character or Wagner parsimony (Kluge and Farris, 1969; Farris, 1970), character states are represented by a linearly ordered transformation series and changes are scored according to the number of intermediate states linking any pair of states. A number of other parsimony variants are reviewed in Swofford et al. (1996). Of these, a very general method described originally by Sankoff (1975) is the one most often applied to molecular sequence data. This "generalized parsimony" method requires the user to specify a matrix giving the cost of a change from any character state to any other. Cost matrices ("step matrices") are typically used to incorporate additional assumptions about the evolutionary process into the analysis. For example, transition bias can be accommodated by assigning higher costs to transversions than to transitions, as demonstrated in the Basic Protocol. One special case of generalized parsimony is "transversion" parsimony, in which transition substitutions are ignored by assigning them a cost of zero (or equivalently, by recoding the data matrix as two-state characters representing purine versus pyrimidine).

The combination of intuitive appeal, ease of implementation, and speed of calculation causes parsimony to remain one of the most highly used methods in contemporary phylogenetics. There are, however, several well defined problems associated with parsimony methods. The most commonly cited criticism of parsimony methods involves their potential for statistical inconsistency in many situations (Felsenstein, 1978; Hendy and Penny, 1989; Kim, 1996; see also Swofford et al., 1996). Particular inequalities in the rates of substitution along paths of an evolutionary tree can cause the probability that character states are identical due to convergent, parallel, or back substitutions to exceed the probability that identical states are shared purely due to common ancestry. As a result, the estimate of the tree topology made by parsimony can converge to an incorrect result with increasing certainty as more data are accumulated (Felsenstein, 1978). This phenomenon, typically referred to as "long-branch attraction," can lead to either artifactual groupings or inappropriately high confidence in correct groupings (see Swofford et al., 2001, for a recent discussion). A few real-data examples of apparent long-branch attraction have been identified in parsimony analyses, as well as in likelihood analyses in which overly simplistic models were used (e.g., Sullivan and Swofford, 1997). However, considerable debate continues as to whether long-branch attraction is a real phenomenon or merely an irrelevant and purely hypothetical concern, largely owing to the fact that the true tree is never known with certainty.

For the analysis of sequence data, the sensitivity of parsimony to long-branch-attraction artifacts can be reduced by giving less weight to sites that are more prone to accept changes (e.g., third codon positions in protein-coding sequences) or to substitutions that appear to occur more frequently (e.g., transitions; Hillis et al., 1994a). Unfortunately, an infinite number of possible weighting schemes exist, and it is difficult to choose among them because of the lack of a "common currency" for evaluating them: i.e., there is no basis for comparing tree lengths obtained under one weighting scheme with those from a different weighting scheme. A possible solution is to perform a "sensitivity analysis" in which a variety of weighting strategies are tested. If all weighting schemes lead to selection of the same tree(s), then this issue is less problematic. However, when different weighting schemes produce different tree topologies, some justification must be provided for the selection of a particular parsimony variant or conclusions should be limited to those that are more stable across weighting schemes.

### *Maximum likelihood*

Maximum-likelihood phylogenetic inference seeks the tree that maximizes the probability of obtaining the observed data given some model of evolutionary change. While maximum-likelihood methods have long been a part of statistical interference (see Efron, 1998), likelihood-based analyses have only relatively recently gained a foothold in phylogenetic inference—e.g., for gene frequencies see Cavalli-Sforza and Edwards (1967); for nucleotide sequences see Felsenstein (1981);

and for amino acid sequences see Kishino et al. (1990). Despite their relatively recent introduction, however, these methods have caught on rapidly and now rival parsimony methods in popularity, especially for the analysis of nucleotide sequences. The increasing acceptance of likelihood stems from several important advantages of model-based methods. For example, the maximum-likelihood model used to infer a phylogeny can be designed to explicitly estimate several key elements of the nucleotide substitution process from the observed sequence data. Also, unlike the situation involving alternative parsimony variants, maximum likelihood provides an objective criterion to measure the goodness-of-fit between specific models and the observed data. In this way, models can be designed that are complex enough to capture the most important aspects of the substitution process, but simpler models are chosen over more complex ones when the additional parameters do not significantly improve the overall fit. Another important feature that distinguishes likelihood from parsimony methods is that likelihood methods incorporate branch-length information into the phylogenetic inference procedure. That is, the probability that substitutions will occur along long branches is greater than that for shorter branches. It is this property that makes likelihood methods less sensitive to long-branch attraction—i.e., the kinds of misleading character patterns that cause parsimony to be misled are expected to occur at high frequency on the true tree under the model; thus, likelihood is not confused by them. Additionally, all characters in the data matrix contribute information to the analyses. Constant (invariant) sites and sites that require the same number of steps on any possible tree are uninformative in parsimony analysis, whereas they provide information on rates of change that is critical for likelihood inference.

One of the greatest limitations of the likelihood method is that it requires significantly more computational time for tree searches than parsimony or distance methods (Sanderson and Kim, 2000). For example, when 50 or more sequences are included in an analysis, the computational demands of the likelihood method may become so great that less thorough methods for exploration of tree space become necessary. While there are some algorithmic shortcuts that can be employed to extend the use of likelihood-based methods for large-scale phylogeny inference, the computational complexity of this method still represents an important obstacle to its widespread use and acceptance with large data sets. The authors would simply urge users to keep the analysis time in perspective; if months or even years are taken to assemble a set of sequences for phylogenetic analysis, then spending a few days or weeks for the computer runs should not be considered unreasonable.

### *Evaluating results*

Important generalizations regarding the performance of phylogenetic methods have been provided by analytical results (Felsenstein, 1978; Chang, 1996; Rogers, 1997; Tuffley and Steel, 1997), simulation studies (Huelsenbeck and Hillis, 1993; Gaut and Lewis, 1995; Huelsenbeck, 1995; Bruno and Halpern, 1999; Swofford et al., 2001), and experimental phylogenetics (Hillis et al., 1992; Hillis et al., 1994b; Cunningham et al., 1998). While these studies have certainly helped practitioners choose among the numerous available phylogenetic reconstruction methods, they do not lead to unambiguous recommendations. A common tendency is to overgeneralize the results of these studies. A method should not be rejected simply because it performs poorly for a single set of simulation parameters, nor should a method be uncritically accepted because it performs well under a restricted range of conditions (Hillis et al., 1996). Careful attention should be paid to the appropriateness of the assumptions of a method for the data being analyzed. This suggestion does not, however, imply that the assumptions of a method must be fully satisfied in order for that method to be useful, as it is often far better to use model-based methods that use imperfect models than to abandon the use of models entirely (Sullivan and Swofford, 2001).

## Critical Parameters and Troubleshooting

As emphasized above, it is unrealistic to think that a user could select an appropriate method for reconstructing a phylogeny without first examining the data set. In the same vein, it is impossible to specify a set of analysis defaults in PAUP* that will be optimal for all data sets. In general, PAUP* default settings are chosen because they are compatible with all of the data types that can be read by the program and not because they represent the "best" general analysis settings. Accordingly, it is important to consider the state of key default parame-

```
#NEXUS
[Practice batch file]
[Numbers correspond to steps in Basic Protocol.]
Begin paup;
     set autoclose=yes warntree=no warnreset=no;
     log start file=practice.log replace;
 [2] execute primate-mtDNA.nex;
 [3] include coding/only;
 [4] delete M._mulatta M._fascicularis M._sylvanus Tarsius_syrichta;
 [5] set criterion=parsimony;
 [6] weights 2:2ndpos;
 [7] ctype 2_1:all;
 [8] cstatus;
 [11]hsearch start=stepwise addseq=random swap=spr;
 [12]outgroup lemur_catta macaca_fuscata saimiri_sciureus;
 [13]showtrees all;
 [14]describetrees 1/plot=phylogram brlens=yes rootmethod=outgroup
outroot=monophyl;
 [15]savetrees file=parsTree.tre brlens=yes replace;
end;

[Numbers correspond to steps in Alternative Protocol.]
Begin paup;
 [1] include all;
 [1] undelete all /cleartrees=yes;
 [1] wts 1:all;
 [1] factory;
 [2] nj;
 [3] lscores 1/nst=6 rmatrix=estimate basefreq=estimate rates=gamma
      shape=estimate pinvar=estimate;
 [4] lscores 1/nst=6 rmatrix=estimate basefreq=estimate rates=equal
      pinvar=0;
 [5] lscores 1/nst=2 tratio=estimate basefreq=estimate rates=gamma
      shape=estimate;
 [6] set criterion=likelihood;
 [7] lscores 1/nst=2 tratio=previous basefreq=previous rates=gamma
      shape=estimate;
 [8] hsearch;
     log stop;
end;
```

**Figure 6.4.15** Commands needed to run Basic Protocol and Alternate Protocol in batch mode. Note that each command and its associated options must end with a semicolon. In this way, two commands can occupy a single line, provided that a semicolon separates them. Likewise, one command can span multiple lines provided the last line ends in a semicolon.

ters before starting an analysis. A complete enumeration of all the options available in PAUP* would be impractical in the context of this publication (such a list does exist in the current command reference document; *http://paup.csit.fsu.edu/commandref.pdf*). Instead, a few of the commands and options that are most generally used will be mentioned and discussion will be included on how to see their current default settings.

By default, the optimality criterion is set to `parsimony`. Changing the criterion requires the use of the `set` command. For example:

```
set criterion=likelihood;.
```

The `set` command controls many of the general options whose scope extends beyond a single command. Typing `set ?` will produce a list of the all the `set` command options and their current state. The specific settings for each of the three optimality criteria, `parsimony`, `likelihood`, and `distance`, are controlled by the `pset`, `lset`, and `dset` commands, respectively. Finally, search options can be found under the `alltrees`, `bandb`, or `hsearch` commands, depending on whether an exact or heuristic search is requested. Like-

wise, typing any command followed by a question mark will cause PAUP* to display a complete list of the available options and their current status.

## Suggestions for Further Analysis

All of the analyses described up to this point have been run in the interactive mode. This mode is especially useful in the exploratory stage of an analysis, when an appropriate model and corresponding set of parameters have not yet been selected. After deciding on the general requirements of an analysis, however, it is usually most efficient to include the necessary commands and option settings in "paup blocks" and run the analysis in batch mode. In this way, the analysis can be run without requiring a user to be present at the various steps of the analysis to enter commands. In addition, repeating an analysis is just a matter of re-executing the file containing the paup blocks. Some users add another level of automation to noninteractive analyses by controlling PAUP* using Shell, Perl, or Python scripts. The added layer of scripting allows users to automatically generate NEXUS analysis files, process PAUP* results, and even create and execute additional paup command files based on results from a previous analysis.

In the example shown in Figure 6.4.15, all the commands required to complete the example analyses described in the Basic and Alternate Protocols are contained in two paup blocks. A `set` command was added at the beginning of the paup block to suppress the dialog box indicating that the heuristic search has completed and several other warnings (Macintosh and Windows only). In addition, the log command was included so that the results printed to the main display are also saved to a file. To run this analysis in batch mode, copy the text given in Figure 6.4.15 to a file and save the file in the same directory as the `primate-mtDNA.nex` file. Next, start PAUP* and execute the newly saved file (see Basic Protocol, step 1).

## Literature Cited


Bruno, W.J. and Halpern, A.L. 1999. Topological bias and inconsistency of maximum likelihood using wrong models. *Mol. Biol. Evol.* 16:564-566.

Camin, J.H. and Sokal, R.R. 1965. A method for deducing branching sequences in phylogeny. *Evolution* 19:311-326.

Cavalli-Sforza, L.L. and Edwards, A.W.F. 1967. Phylogenetic analysis: Models and estimation procedures. *Am. J. Hum. Genet.* 19:233-257.

Chang, J.T. 1996. Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. *Math. Biosci.* 137:51-73l.

Cox, D.R. and Hinkley, D.V. 1974. Theoretical statistics. Chapman and Hall, London.

Cunningham, C.W., Zhu, H., and Hillis, D.M. 1998. Best-fit maximum-likelihood models for phylogenetic inference: Empirical tests with known phylogenies. *Evolution* 52:978-987.

Efron, B. 1998. R.A. Fisher in the 21st century. *Stat. Sci.* 13:95-122.

Farris, J.S. 1970. Methods for computing Wagner trees. *Syst. Zool.* 19:83-92.

Farris, J.S. 1977. Phylogenetic analysis under Dollo's Law. *Syst. Zool.* 26:77-88.

Farris, J.S., Albert, V.A., Kallersjö, M., Lipscomb, D., and Kluge, A.G. 1996. Parsimony jackknifing outperforms neighbor-joining. *Cladistics* 12: 99-124.

Felsenstein, J. 1978. Cases in which parsimony and compatibility methods will be positively misleading. *Syst. Zool.* 27:401-410.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.* 17:368-376.

Felsenstein, J. 1985. Confidence limits on phylogeny: An approach using the bootstrap. *Evolution* 39:783-789.

Fitch, W.M. 1971. Toward defining the course of evolution: Minimal change for a specific tree topology. *Syst. Zool.* 20:406-416.

Gaut, B.S. and Lewis, P.O. 1995. Success of maximum likelihood phylogeny inference in the four-taxon case. *Mol. Biol. Evol.* 12:152-162.

Goldman, N., Anderson, J.P., and Rodrigo, A.G. 2000. Likelihood-based tests of topologies in phylogenetics. *Syst. Biol.* 49:652-670.

Gu, X., Fu, Y.-X. and Li, W.-H. 1995. Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Mol. Biol. Evol.* 12:546-557.

Hall, B. 2001. Phylogenetic Trees Made Easy: A How-to Manual for Molecular Biologists. Sinauer Associates, Sunderland, Mass.

Hasegawa, M., Kishino, H., and Yano, T. 1985. Dating the human-ape split by a molecular clock of mitochondrial DNA. *J. Mol. Evol.* 22:160-174.

Hendy, M.D. and Penny, D.A. 1989. A framework for the quantitative study of evolutionary trees. *Syst. Zool.* 38:297-309.

Hillis, D.M., Bull, J.J., White, M.E, Badgett, M.R., and Molineux, I.J. 1992. Experimental phylogenetics: Generation of a known phylogeny. *Science* 255:589-592.

Hillis, D.M., Huelsenbeck, J.P., and Swofford, D.L. 1994a. Hobgoblin of phylogenetics? *Nature* 369:363-364.

Hillis, D.M., Huelsenbeck, J.P., and Cunningham, C.W. 1994b. Application and accuracy of molecular phylogenies. *Science* 264:671-677.

Hillis, D.M., Mable, B.K., and Moritz, C. 1996. Applications of molecular systematics: The state of the field and a look to the future. *In* Molecular Systematics, 2nd ed. (D.M. Hillis, C. Moritz, and B.K. Mable, eds.), pp. 515-543. Sinauer Associates, Sunderland, Mass.

Huelsenbeck, J.P. and Hillis, D.M. 1993. Success of phylogenetic methods in the four-taxon case. *Syst. Biol.* 42:247-265.

Huelsenbeck, J.P. 1995. Performance of phylogenetic methods in simulation. *Syst. Biol.* 44:17-48.

Kim, J. 1996. General inconsistency conditions for maximum parsimony: Effects of branch lengths and increasing numbers of taxa. *Syst. Biol.* 45:363-374.

Kishino, H. and Hasegawa, M. 1989. Evolution of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in *Hominoidea*. *J. Mol. Evol.* 29:170-179.

Kishino, H., Miyata, T., and Hasegawa, M. 1990. Maximum likelihood inference of protein phylogeny and the origin of chloroplasts. *J. Mol. Evol.* 30:151-160.

Kluge, A.G. and Farris, J.S. 1969. Quantitative phyletics and the evolution of anurans. *Syst. Zool.* 18:1-32.

Lockhart, P.J., Steel, M.A., Hendy, M.D., and Penny, P. 1994. Recovering evolutionary trees under a more realistic model of sequence evolution. *Mol. Biol. Evol.* 11:605-612.

Maddison, D.R., Swofford, D.L., and Maddison, W.P. 1997. NEXUS: An extensible file format for systematic information. *Syst. Biol.* 46:590-621.

Page, R.D. and Holmes, E.C. 1998. Molecular Evolution: A Phylogenetic Approach. Blackwell Science, Oxford, U.K..

Penny, D. and Hendy, M.D. 1985. Testing methods of evolutionary tree construction. *Cladistics* 1:266-272.

Posada, D. and Crandall, K.A. 1998. MODELTEST: Testing the model of DNA substitution. *Bioinformatics* 14:817-818.

Rogers, J.S. 1997. On the consistency of maximum likelihood estimation of phylogenetic trees from nucleotide sequences. *Syst. Biol.* 46:354-357.

Sanderson, M.J. and Kim, J. 2000. Parametric phylogenetics? *Syst. Biol.* 49:817-829.

Sankoff, D. 1975. Minimal mutation trees of sequences. *SIAM J. Appl. Math.* 28:35-42.

Shimodaira, H. and Hasegawa, M. 1999. Multiple Comparisons of Log-Likelihoods with Applications to Phylogenetic Inference. *Mol. Biol. Evol.* 16:1114-1116.

Steel, M. 1994. Recovering a tree from the Markov leaf colourations it generates under a Markov model. *Appl. Math. Lett.* 7:19-23.

Sullivan, J. and Swofford, D.L. 1997. Are guinea pigs rodents? The utility of models in molecular phylogenetics. *J. Mamm. Evol.* 4:77-86.

Sullivan, J. and Swofford, D.L. 2001. Should we use model-based methods for phylogenetic inference when we know assumptions about among-site rate variation and nucleotide substitution pattern are violated? *Syst. Biol.* 50:723-729.

Swofford, D.L. 2002. PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4. Sinauer Associates, Sunderland, Mass.

Swofford, D.L. and Maddison, W.P. 1987. Reconstructing ancestral character states under Wagner parsimony. *Math. Biosci.* 87:199-229.

Swofford, D.L., Olsen, G.J., Waddell, P.J., and Hillis, D.M. 1996. Phylogenetic inference. *In* Molecular systematics, 2nd ed. (D.M. Hillis, C. Moritz, and B.K. Mable, eds.). pp. 407-514. Sinauer, Sunderland, Mass.

Swofford, D.L., Waddell, P.J., Huelsenbeck, J.P., Foster, P.J., Lewis, P.O., and Rogers, J.S. 2001. Bias in phylogenetic estimation and its relevance to the choice between parsimony and likelihood methods. *Syst. Biol.* 50:525-539.

Templeton, A.R. 1983. Convergent evolution and non-parametric inferences from restriction fragment and DNA sequence data. *In* Statistical Analysis of DNA Sequence Data. (B. Weir, ed.) pp. 151-179. Marcel Dekker, New York.

Tuffley, C., and Steel, M. 1997. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. *Bull. Math. Biol.* 59:581-607.

Waddell, P.J. and Penny, D. 1996. Evolutionary trees of apes and humans from DNA sequences. *In* Handbook of symbolic evolution (A.J. Lock and C.R. Peters, eds.) pp. 53-73. Clarendon Press, Oxford, U.K.

Yang, Z. 1994a. Estimating the pattern of nucleotide substitution. *J. Mol. Evol.* 39:105-111.

Yang, Z. 1994b. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *J. Mol. Evol.* 39:306-314.

## Internet Resources

http://paup.csit.fsu.edu/

*PAUP* Web site.*

http://pauptech.csit.fsu.edu/~paupforum/

*PAUP* technical forum.*

http://mailer.csit.fsu.edu/mailman/listinfo/paupinfo/

*PAUP* information mailing list.*

http://www.sinauer.com/

*PAUP* publisher, Sinauer Associates, Inc. Web site.*

## Key References

Hillis et al., 1996. See above.

*A general discussion of issues and controversies pertaining to phylogenetic analyses.*

Page and Holmes, 1998. See above.

*An accessible introduction to phylogenetic theory, terminology, and practice.*

Swofford et al., 1996. See above.

*A detailed description of most methods commonly used in phylogenetic inference.*

Contributed by James C. Wilgenbusch and David Swofford
Florida State University
Tallahassee, Florida

# Using MODELTEST and PAUP* to Select a Model of Nucleotide Substitution



***BASIC PROTOCOL***

Models of nucleotide substitution are commonly used in the analysis of DNA sequences, especially in the estimation of evolutionary parameters and in the construction of phylogenetic trees. A common problem that a researcher has to face is the objective selection of such a model. The program MODELTEST (Posada and Crandall, 1998) offers two sound statistical procedures to carry out this choice.

The protocol given below describes the joint use of the programs MODELTEST and PAUP* (Swofford, 2000; *UNIT 6.4*) to select the best-fit model of nucleotide substitution for a given set of aligned DNA sequences. Users who are unfamiliar with PAUP* or maximum-likelihood methods are encouraged to read *UNIT 6.4*. The three basic steps within this protocol consist of the estimation of a phylogenetic tree, the estimation of likelihood scores for the candidate models, and the selection of a particular model given these likelihood scores. The first two procedures are carried out in the program PAUP* with a batch file included in the MODELTEST package. The discussion in this unit summarizes the theory behind the procedure and provides several caveats and guides for interpretation of the results. Particular comments about implementation on different platforms is also given.

Users can run PAUP* and MODELTEST either through a graphical user interface (GUI) or from a command line, depending on the platform. PAUP* is commonly used in Macintosh OS9, where a complete GUI application is available. In Windows, PAUP* has a much simpler GUI, but enough for the purposes here. MODELTEST, however, only offers the GUI for Macintosh; Windows and Unix-like environment users of MODELTEST must run the program from the command line. The protocol described below refers primarily to the GUI versions of PAUP* and MODELTEST. Step annotations describe how each step can be accomplished from the command line.

Both MODELTEST and PAUP* are available from their respective Web sites (see Internet Resources), and users are encouraged to download the most current versions. MODELTEST is freely available, while PAUP* is commercial software licensed by Sinauer.

### *Necessary Resources*

*Hardware*

Standard workstation (e.g., Macintosh, Windows, Linux, or Unix system)

*Software*

MODELTEST

*This is a stand-alone application which has been compiled for Linux, Unix, Windows, and Macintosh platforms. The C code is freely available, and therefore the program can be compiled in any platform and run on any computer.*

*MODELTEST was developed to work with the output provided by the program PAUP*, although alternative programs might be used for the calculation of likelihood scores. The interaction between PAUP* and MODELTEST is described in Figure 6.5.1. MODELTEST can be obtained from the Web (see Internet Resources).*



**Contributed by David Posada**

*Current Protocols in Bioinformatics* (2003) 6.5.1-6.5.14

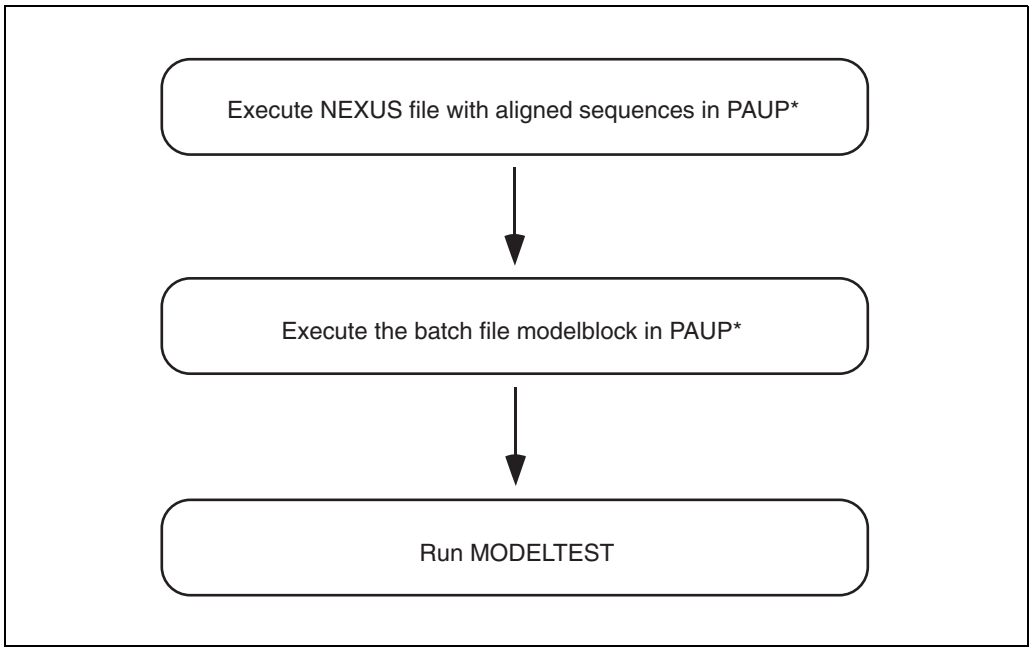


**Figure 6.5.1** Flow chart of the model selection procedure using PAUP* and MODELTEST.

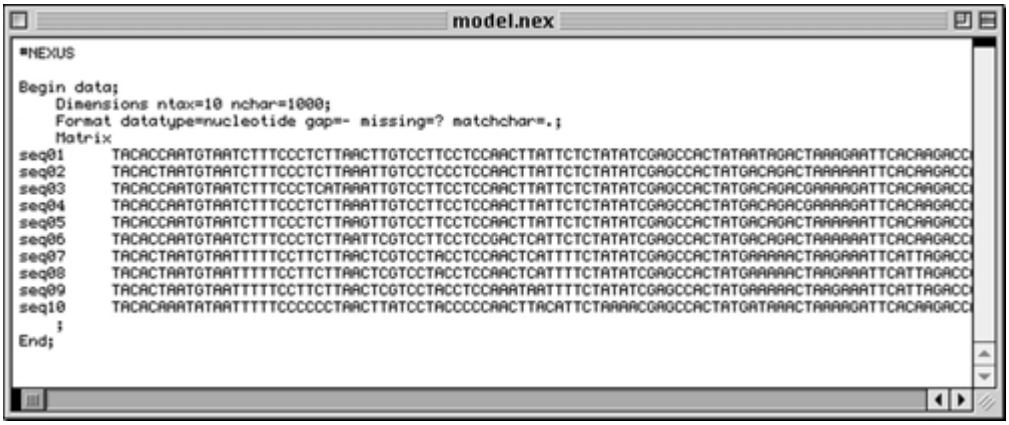


**Figure 6.5.2** Aligned sequences in sequential NEXUS format in the file `sample.nex`. This file can be read by PAUP*.

PAUP* (Sinauer; *UNIT 6.4*)

*This program is also available for a variety of platforms and available on the Web (see Internet Resources).*

*Files*

Initial data file with aligned DNA sequences in any format recognized by PAUP* (e.g., NEXUS, PHYLIP, PIR; *UNIT 6.4*)

*Figure 6.5.2 shows an example file in sequential NEXUS format. This sample file is included in the MODELTEST package.*

PAUP* batch file

*This file is included in the MODELTEST package, and is required in order to obtain an output file from PAUP* that becomes the input for MODELTEST. For a discussion of PAUP* batch files see UNIT 6.4.*

**Table 6.5.1** Files Included in the MODELTEST Package[a]

| File | Description |
|---|---|
| `README.html` | Quick instructions and comments for the users |
| `/batch/modelblock` | The batch file with PAUP* commands to obtain likelihood scores for the competing models in the proper format for MODELTEST |
| `/bin/Modeltest3.1.mac` | Macintosh (OS 9) executable |
| `/bin/Modeltest3.1.macX` | Macintosh (OS X) executable |
| `/bin/Modeltest3.1.win.exe` | Windows executable |
| `/doc/Modeltest3.1.pdf` | Documentation in PDF format |
| `/license/gpl.html` | GNU general public license |
| `/sample/sample.nex` | Example data file in NEXUS format |
| `/sample/sample.scores` | File with likelihood scores produced by PAUP* after loading `sample.nex` and executing the modelblock batch file |
| `/sample/sample.log` | A log file describing the calculations performed by PAUP* to obtain `sample.scores` |
| `/sample/sample.out` | The output file of MODELTEST resulting from the analysis of sample.scores |
| `/source/modeltest3.1.c` | ANSI C source code |
| `/source/Makefile` | Makefile for compilation of MODELTEST in Unix-like environments |

[a]Note that files are found in several different subdirectories.

MODELTEST package files (Table 6.5.1)
Example file: `sample.nex` (optional)

*This example file, included in MODELTEST and used also as an example here, is a simulated data set with ten aligned DNA sequences 1000 bp long. This alignment was simulated on a predefined tree under the HKY+Γ model with the following parameter values: mutation rate per nucleotide per site = $5 \times 10^{-5}$; base frequencies (A, C, G, T) = 0.4, 0.2, 0.1, 0.3; transition/transversion rate = 4; and α parameter of the gamma distribution = 0.4.*

*NOTE:* Users who are unfamiliar with PAUP* or maximum-likelihood methods are encouraged to read *UNIT 6.4*.

### *Load the data file in PAUP**

1. Start PAUP* by double-clicking on the application file.

   *In Unix, type* `paupx.x` *in the command line where* `x.x` *is the version of PAUP*.*

2. Select Open from the File Menu in the PAUP* interface. In the browser window that appears, check the Execute radio button, and then select the data file (in this example use `sample.nex`), which should be in NEXUS format. PAUP* will display information indicating whether the file has been correctly processed.

   *In Unix type:*

   ```
   execute sample.nex
   ```

   *If the file is not in NEXUS format, the GUI version of PAUP* can still import it if it is in PHYLIP, MEGA, PIR, or MSF format (other less common formats are also admitted). Go to the File Menu and select Import Data. Set the options to the proper Format and Data*

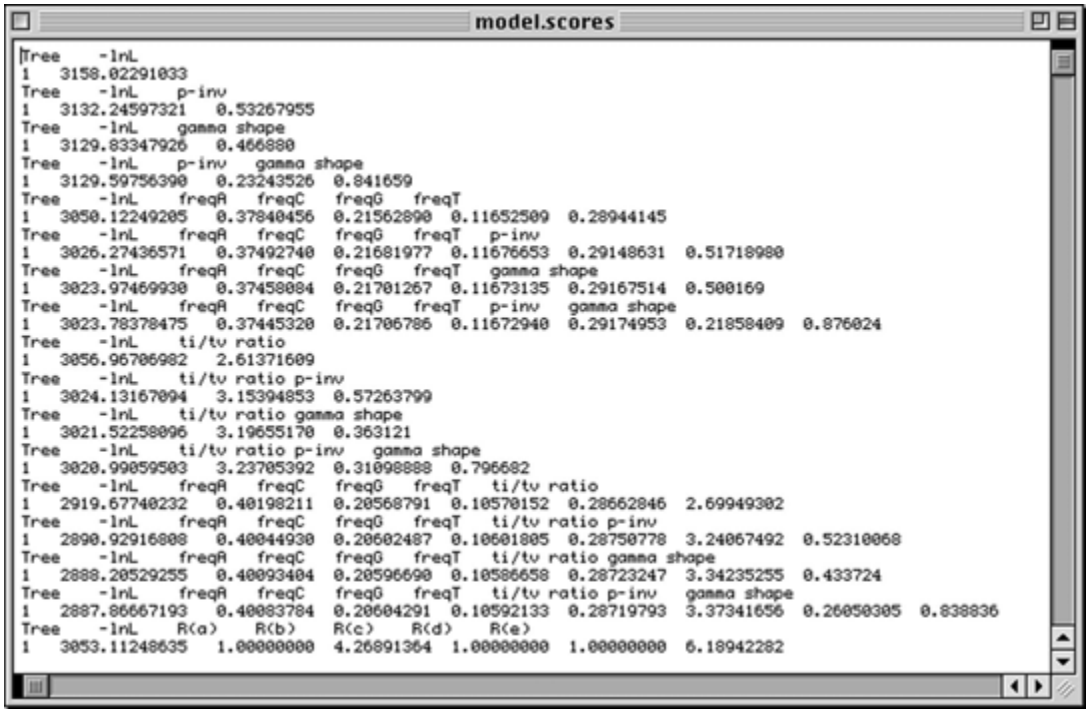


**Figure 6.5.3** Likelihood scores for different models of substitution in the file `sample.scores`. This is the format required by MODELTEST.

*Type, and check the Interleaved button if the sequences are in interleaved format. The file will be then opened automatically in the PAUP* editor in NEXUS format. The file can now be saved or executed by going to the file menu and selecting Save or Execute, respectively.*

### *Execute PAUP* modelblock batch file to generate likelihood scores for competing models*

3. Select Open from the File menu in the PAUP* interface. In the browser window that appears check the Execute radio button, browse, and then select the batch file `modelblock`. PAUP* will start processing the commands in `modelblock` to first estimate a neighbor-joining tree (*UNIT 6.3*), and then to calculate likelihood scores for 56 substitution models. The likelihood scores and parameter estimates are written to a file (e.g., `sample.scores`; Fig. 6.5.3). The likelihood calculations can take from a few minutes (e.g., `sample.nex`) to even days, depending on the complexity of the data.

   *In Unix type:*

   ```
   execute modelblock.
   ```

   *Remember that at this point only PAUP* is being run (i.e., MODELTEST has not been run yet). Two files will be created during this run in the* `PAUP*` *directory. The file* `sample.scores` *will be the input file for MODELTEST, and it is often convenient to rename it to* `yourdata.scores`*. The other file produced by PAUP*,* `sample.log`*, is just intended to check that everything has proceeded normally.*

### *Run MODELTEST*

4. Double click the MODELTEST application file. This will bring up a GUI (Fig. 6.5.4) where the input and output files can be selected, and where arguments can be specified.

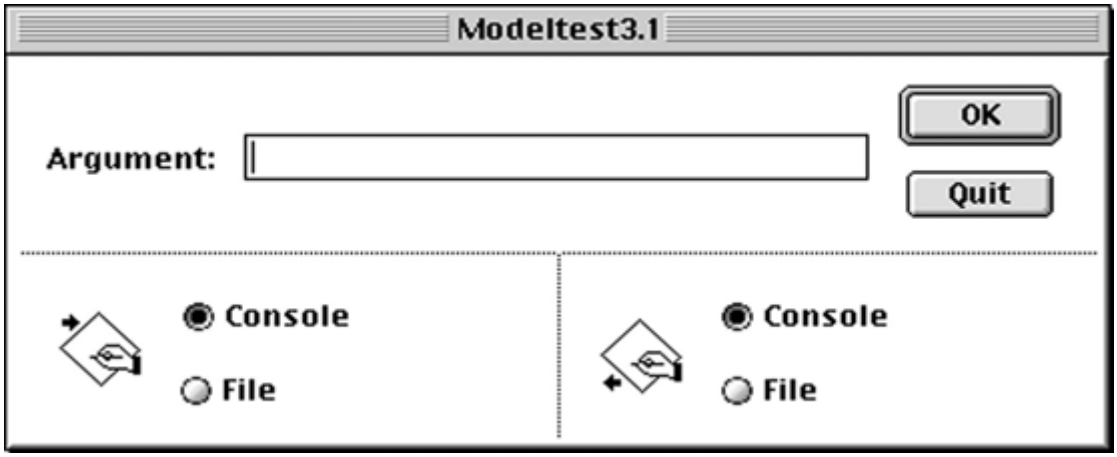


**Figure 6.5.4** MODELTEST 3.1 GUI in Macintosh OS 9. The input file can be chosen by selecting the left File button, while MODELTEST output can be redirected to a file by selecting the right File button. MODELTEST arguments can be entered in the Argument field.

*In Unix and Windows, MODELTEST has to be executed from the command line. In the case of Windows this command line is accessed through a DOS window (MS-prompt, console). Consult specific Windows documentation for information on command-line features. To automatically carry out steps 4 to 7 of this protocol with default parameters in Windows or Unix type the following at the command line:*

```
modeltest3.1 < sample.scores > modeltest.out
```

*The symbols "<" and ">" are the input and output redirection symbols (see Working with Text Files in APPENDIX 1C). In this example, "<" means "get input from the file* `sample.scores` *" and ">" means "send output to the file* `modeltest.out`*."*

*The following is an example of options that can be entered at the command line:*

```
modeltest3.1 -a0.01 < sample.scores > modeltest.out
```

*See step 7 below for further discussion of options.*

5. Click the left (input) File button. This will bring up a browser window to select the file with the likelihood scores (i.e., `sample.scores`).
6. Click the right (output) File button. This will bring up a browser window to select the file where MODELTEST will print out the results (e.g., `yourdata.modeltest.out`). Alternatively, the output can be saved after execution.
7. Type in the Argument line any of the arguments to change the default options. Possible arguments (explained in the program documentation) are:

   `-a`: α level for the LRT tests (`-a0.01` is the default value)
   `-c`: Likelihood ratio calculator mode
   `-i`: AIC calculator mode
   `-f`: Input from a file for AIC calculation
   `-?`: Help.

8. Click the Run button. The analysis will be finished in a few seconds. If an output file has not been previously selected, go to the File menu and select Save. The MODELTEST analysis is done.

## GUIDELINES FOR UNDERSTANDING RESULTS

### MODELTEST Output

The first section of the MODELTEST output consists of the log likelihood scores from the file `sample.scores`. This feature provides an easy check of the procedure and a simultaneous display of the likelihood for all the 56 substitution models compared in MODELTEST 3.1 (Fig. 6.5.5). The other sections of the output relate to two main statistical procedures to select a best-fit model, the hierarchical likelihood ratio tests (hLRT) and the Akaike information criterion (AIC).

In the case of the hLRT procedure (Fig. 6.5.6), MODELTEST first prints information about the likelihood ratio tests performed, including the corresponding *p*-value. Next, MODELTEST describes the model selected, indicating the values of the parameter of this sample. It is important to note that these estimates were obtained by PAUP*, not by MODELTEST. MODELTEST does not estimate any parameter. Finally, MODELTEST prints out a block of commands for PAUP*. These commands can be very useful if the user wants to use the best-fit model for further analyses, for example to estimate a phylogeny in PAUP*. In order to use it, this command block should be copied and pasted after the data block in the NEXUS file (i.e., `sample.nex`; *UNIT 6.4*). When such a file is executed in PAUP*, these commands will automatically set up the selected sample.

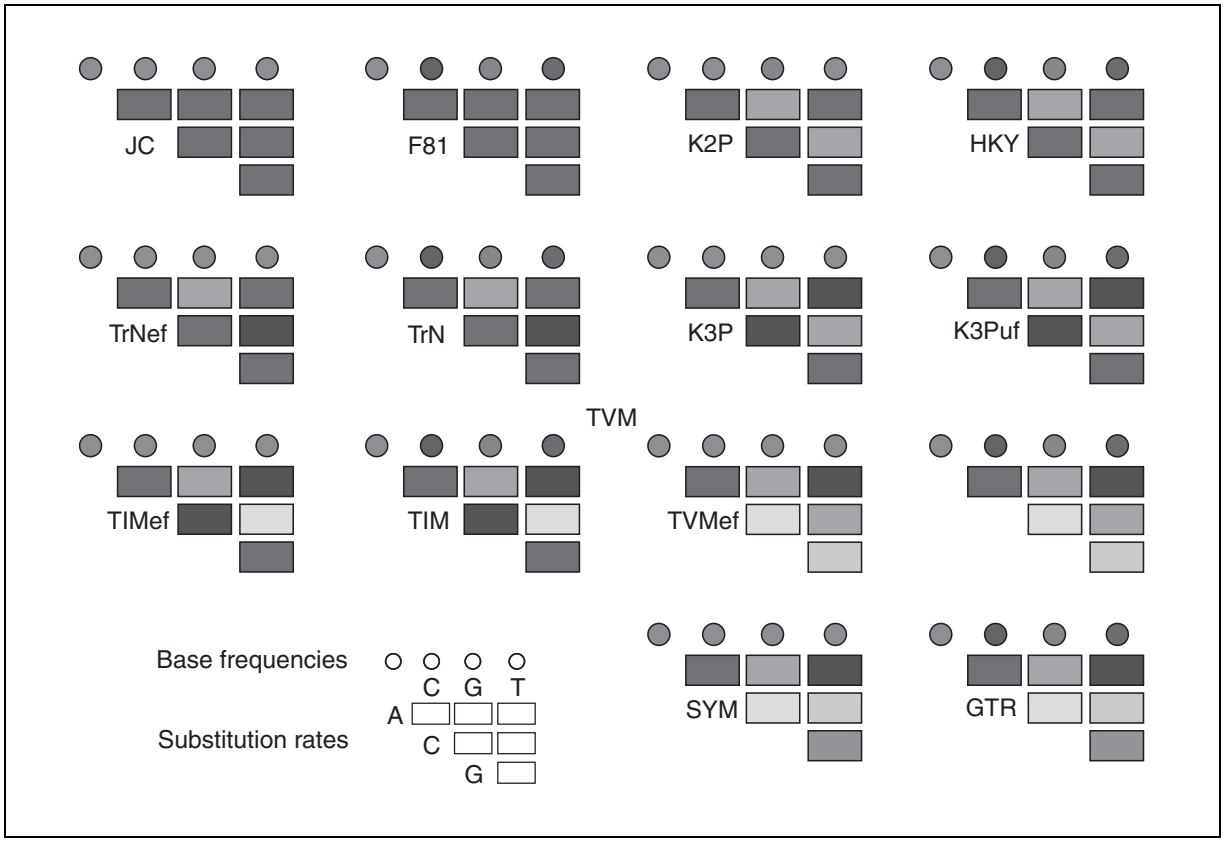


**Figure 6.5.5** Basic 14 models of substitution included in MODELTEST. Circles represent base frequencies, while rectangles represent symmetric substitution rates among nucleotides. Frequencies or rates with the same color are constrained within the same parameter. The number of free parameters for each model is the number of different colors minus two, because base frequencies have to add to one and the substitution rate G↔T is arbitrarily set to 1. Each one of these models can also include rate heterogeneity by specifying a proportion of invariable sites (+I), gamma distributed rates (+Γ), or both (+I +Γ). This makes 14 × 4 = 56 models in MODELTEST. For example, there will be JC (Jukes and Cantor, 1969), JC+I, JC+Γ, and JC+I +Γ models. *This black and white facsimile of the figure is intended only as a placeholder; for full-color version of figure, go to http://www.currentprotocols.com/colorfigures.*

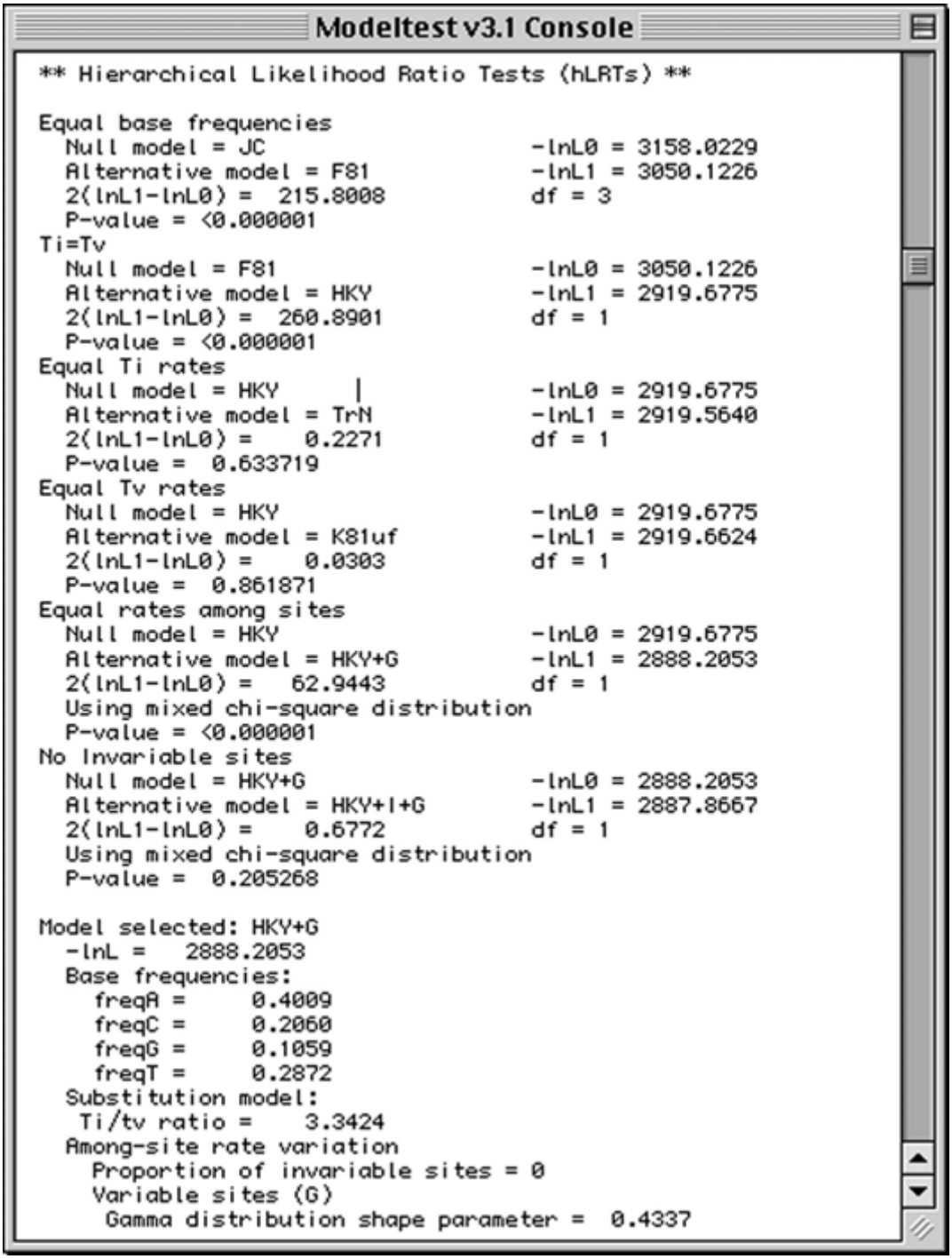


**Figure 6.5.6** Section of the MODELTEST output corresponding to the hierarchical likelihood ratio tests (hLRT).

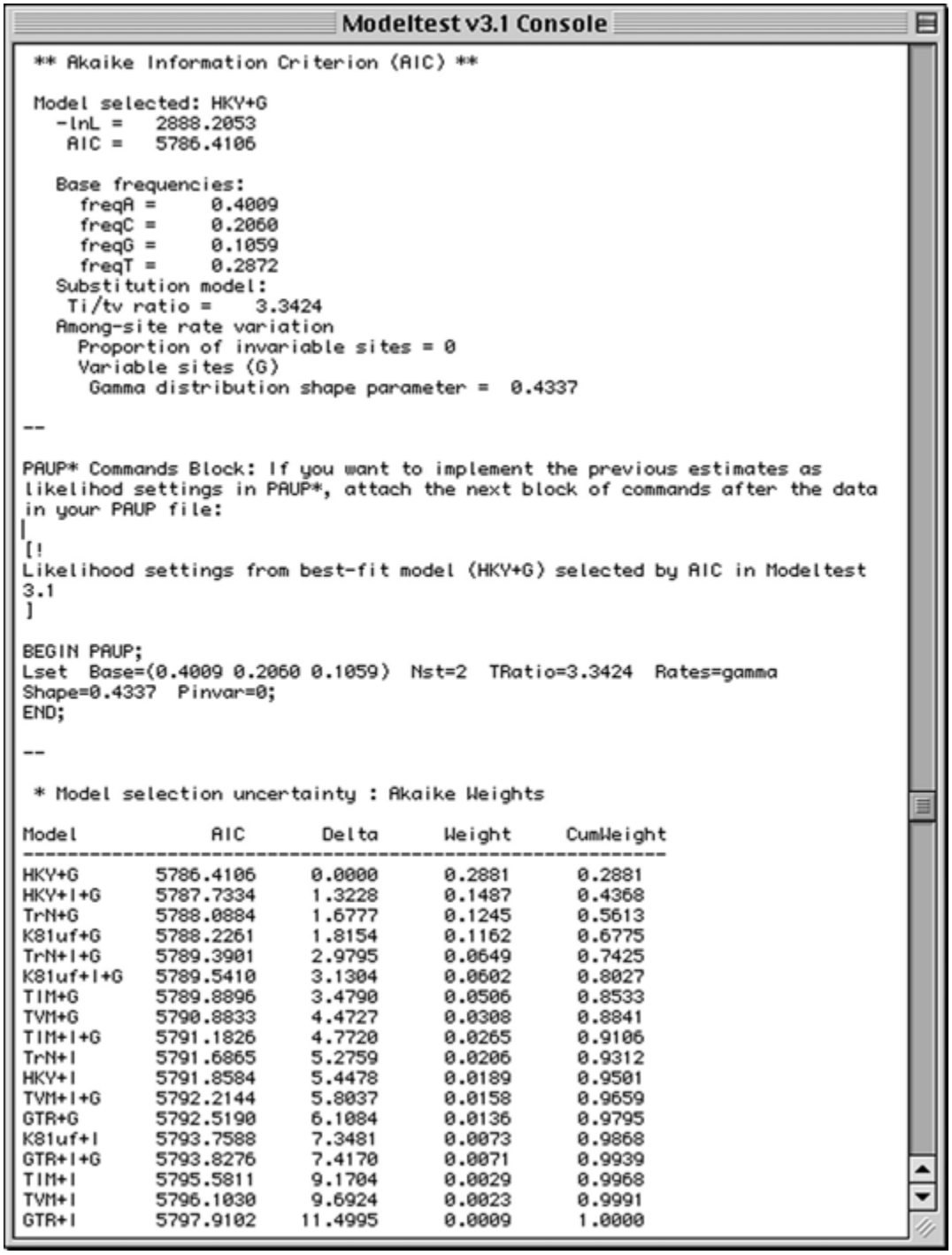


**Figure 6.5.7** Section of the MODELTEST output corresponding to the Akaike information criterion (AIC).

In the AIC section (Fig. 6.5.7), MODELTEST again prints a description of the selected model and a block of PAUP* commands. The last part of the AIC output is the AIC differences—called deltas (Δ)—and the Akaike weights for every model, which can be used to get an idea of model selection uncertainty. The models are ordered according to their Akaike weights, and the first model is always the model selected by the AIC.

### Interpreting MODELTEST Results

There are two different statistical approaches to model selection implemented in MODELTEST (LRT and AIC), and in some cases they will indeed select different models. Belief in the results of one technique over that of the other is dependent upon the philosophy of the particular user, and such a choice should be made upon a good understanding of both approaches. An introduction to these model selection strategies and some useful references can be found at the end of this unit (see Commentary).

### Results From the Example Data Set (`sample.nex`)

The hierarchical likelihood ratio tests selected the HKY+Γ (`HKY+G`) model as the best-fit model for the `sample.nex` data set (Fig. 6.5.6). There were a total of six LRT performed, two of them rejected, and four accepted. The hypothesis tested by each LRT is indicated, as well as the log likelihood corresponding to the models contrasted, the number of degrees of freedom, the value of the LRT and the associated *p*-value. Parameter estimates corresponding to the best-fit model (these estimates were calculated by PAUP*) are also indicated. If the goal is to estimate a phylogenetic tree, these parameters can be fixed and a heuristic search performed under the maximum likelihood criterion (*UNIT 6.4*). The commands required to load this model in PAUP* are below in the MODELTEST window (e.g., Fig. 6.5.7).

The AIC criterion also selected the HKY+Γ (`HKY+G`) as the best-fit model (Fig. 6.5.7). The corresponding log likelihood, AIC and parameter estimates are also indicated. In Figure 6.5.7 the set of PAUP* commands that the user can attach to its NEXUS file to specify this model in PAUP* for further analysis can also be seen. The Akaike weights are also indicated. In this case it can be seen that there is not that much confidence in the best-fit model, but it seems that there is a set of perhaps four plausible models—i.e., models having deltas ($\Delta_i$) within 1 to 2 of the best model—which have substantial support (see Background Information)—that could be considered.

In this case both strategies have identified the *true model* as the best-fit (remember that the `sample.nex` data set was simulated under the HKY+Γ model). In general, this seems the case with model selection strategies (Posada, 2001; Posada and Crandall, 2001a), which suggests that model selection strategies can successfully identify features from the data. However, a very important consideration is that in real life, the true model will not be one of the candidate models.

## COMMENTARY

### Background Information

#### *Selecting models of evolution*

Phylogenetic estimation is a problem of statistical inference. When using DNA sequences to estimate phylogenetic relationships, it is necessary to specify some probability model that describes the different probabilities of change from one nucleotide to another. However, this model is not always made explicit, like in the case of maximum parsimony. Indeed, models are not just interesting from a phylogenetic reconstruction point of view, but also because models of substitutions are themselves descriptions of the evolutionary process at the molecular level. Importantly, models do not try to mimic the exact process of molecular evolution, which indeed is unknown, but rather try to approximate it.

Common models of nucleotide substitution include parameters that describe base frequencies, substitution rates among the four nucleotides, or distribution of the rate of evolution among sites (rate variation) and among lineages (the molecular clock). Some models assume that the base frequencies are equal, while other models allow them to vary freely, with the only constraint that they have to add to one. When reversibility of change is assumed—i.e., the probability of changing from nucleotide *i* to nucleotide *j* is the same as the probability of changing from nucleotide *j* to nucleotide *i*—there are six possible substitution rates among the four nucleotides: $r_{AC}$, $r_{AG}$, $r_{AT}$, $r_{CG}$, $r_{CT}$, and $r_{GT}$. Complex models allow these six rates to vary freely, while less complex models place some constraints on their variation—e.g., all transitions have the same rate and all transversions have the same rate, but transitions and transversions do not have the same rate (i.e., $r_{AG} = r_{CT} \neq r_{AC} = r_{AT} = r_{CG} = r_{CT}$). Rate variation among sites can be included in the model by simply assuming that there is a proportion of invariable sites ($p_{inv}$) while the rest of the sites evolve at the same rate. Alternatively, each site can be assigned a certain probability of belonging to a specific rate category. This set of probabilities is conveniently described by a discrete gamma distribution ($\Gamma$) with four categories (Yang, 1994, 1996). When the shape parameter of the gamma distribution ($\alpha$) is small, most of the sites evolve very slowly, but a few sites have moderate-to-fast rates. When $\alpha$ increases, most of the sites evolve at medium rates, and a few at slow and fast rates. When $\alpha$ is infinity, all the sites evolve at the same rate. Also, it should be considered that rates of evolution may be different in different parts of the tree. Figure 6.5.5 describes the basic substitution models included in MODELTEST. For those interested in details of the models, Swofford et al. (1996) provide a comprehensive review of common models. More complex models indeed exist (e.g., Goldman and Yang, 1994; Muse and Gaut, 1994; Schöniger and von Haeseler, 1994; Thorne et al., 1998; Tuffley and Steel, 1998; Huelsenbeck and Nielsen, 1999; Huelsenbeck, 2002), but their application is still uncommon, mainly because these models have not yet been implemented in software packages.

### *Goodness of fit*

It is important to understand how well the models being used to make inferences fit the data. A way of assessing the fit of a single model to the data is to calculate the maximum value of the likelihood function under the multinomial distribution as an upper boundary to which the likelihood of any model can be compared (Goldman, 1993). The likelihood function under the multinomial distribution refers to an unconstrained model of evolution, and for *n* aligned DNA sequences of length *N* sites (excluding gapped sites) it has the form:

$$L = \prod_{b \in \Re} (p_b)^{n_b}$$

where $\Re$ is a set of $4^n$ possible nucleotide patterns that may be observed at each site, $p_b$ is the probability that any site exhibits pattern b in $\Re$ given the tree and a substitution model, and $n_b$ is the number of times the pattern b is observed out of the *N* sites. It should be realized, however, that this test is very stringent, and most models offer a significantly worse fit than the multinomial. This does not imply that the models being used today are inadequate to provide reasonable estimates, but rather that current models do not provide a perfect description of the underlying evolutionary process. Since it is never expected for a model of evolution to be correct in every detail, this test is perhaps best used to estimate how far the assumed model deviates from the underlying process that generated the data (Swofford et al., 1996).

### *Likelihood ratio test*

The likelihood ratio test (LRT) statistic is one of the most widely used tools for comparing the fit of two competing models, and is given by the equation: $\text{LRT} = 2(\ln L_1 - \ln L_0)$.

Here $L_1$ is the likelihood maximized under the more complex model (which is the alternative hypothesis) and $L_0$ is the likelihood maximized under a simpler model (null hypothesis). The value of the LRT is always equal to or greater than zero, even if the simpler model is the true one, simply because the superfluous parameters in the complex model will always provide a better explanation of the stochastic variation in the data than the simpler sample. When the models compared are nested (the simple model is a special case of the complex model) twice this statistic is asymptotically distributed as $\chi^2$ with the number of degrees of freedom equal to the difference in number of free parameters between the two models. Therefore the corresponding *p*-value can be looked up in a $\chi^2$ table or calculated using the $\chi^2$ distribution. When the *p*-value associated to the LRT is significant—i.e., smaller than some predeter-

mined value (e.g., 0.05)—it can be concluded that the additional parameters in the complex model significantly increase the fit to the data. On the other hand, an LRT close to zero suggests that the complex model does not fit the data significantly better than the simple sample.

The $\chi^2$ distribution approximation for the LRT statistic is not appropriate when the null model is equivalent to fixing some parameter at the boundary of its parameter space in the alternative model (Whelan and Goldman, 1999). In this case, the use of a mixed $\chi^2$ distribution (50% and 50%) is appropriate. An example of this situation is an LRT between two models that differ only in that the complex model includes a parameter for the proportion of invariable sites, which ranges from 0 to 1. The simple model (null hypothesis) is thus a special case of the complex model where the proportion of invariable sites is fixed to 0, which is at the boundary of the range of the parameter in the complex model (alternate hypothesis). The use of LRTs in phylogenetics is reviewed by Huelsenbeck and Crandall (1997) and Huelsenbeck and Rannala (1997).

### *Hierarchical likelihood ratio tests*

Comparing two different nested models through an LRT actually tests hypotheses about the data represented by the difference in the assumptions between the models compared. Therefore, testing several hypotheses in a hierarchical manner one "arrives" at the best-fit model for the data set at hand (Frati et al., 1997; Huelsenbeck and Crandall, 1997; Posada and Crandall, 1998). For example, to test the equal base-frequencies hypothesis, it is possible to do an LRT comparing JC versus F81 (Jukes and Cantor, 1969; Felsenstein, 1981), as these models only differ in the fact that F81 allows for unequal base frequencies (alternative hypothesis), while JC assumes equal base frequencies (null hypothesis). This is one of the strategies implemented in MODELTEST (Fig. 6.5.8).

### *Akaike information criterion*

The Akaike information criterion (AIC; Akaike, 1974) is an asymptotically unbiased estimator of the Kullback-Leibler information quantity (Kullback and Leibler, 1951), which is a measure of the information that is lost when a model is used to approximate full reality. The smaller the AIC, the better the fit of the model to the data. This is approximately equivalent to minimizing the expected Kullback-Leibler distance between the true model and the estimated sample. The AIC penalizes for the increasing number of parameters in the model, so it is taking into account not only the goodness of fit but also the variance of the parameter estimates. It is computed as: $AIC_i = -2 \ln L_i + 2 N_i$, where $N_i$ is the number of free parameters in the $i$-th model and $L_i$ is the maximum-likelihood value of the data under the $i$-th sample.

The AIC offers several interesting advantages. First, the AIC compares several candidate models simultaneously (while the LRT is a pairwise comparison). Second, it can be used to compare both nested and nonnested models. Third, model-selection uncertainty can be easily quantified using the AIC differences and Akaike weights. AIC differences ($\Delta_i$) are rescaled AICs, where the model with the minimum AIC (minAIC) has a value of 0: $\Delta_i = AIC_i - \text{minAIC}$.

The AIC differences are easy to interpret and allow a quick comparison and ranking of candidate models. As a rough rule of thumb, models having $\Delta_i$ within 1 to 2 of the best model have substantial support and should receive consideration. Models having $\Delta_i$ within 3 to 7 of the best model have considerably less support, while models with $\Delta_i > 10$ have essentially no support (Burnham and Anderson, 1998). Akaike weights ($w_i$) are the normalized relative AIC for each candidate model, and can be interpreted as the probability that a model is the best approximation to the truth given the data:

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

for $R$ candidate models. A very interesting application of the Akaike weights is that inference can be averaged from those models where the Akaike weights are nontrivial. For example, a model-averaged estimate of $\alpha$ (the shape of the gamma distribution for rate variation among sites) would be:

$$\hat{\bar{\alpha}} = \sum_{i=1}^{R} w_i \hat{\alpha}_i$$

Indeed, it is also possible to think of estimating phylogenies under the best models and combining these trees according to their Akaike weights. Burnham and Anderson (1998) provide an excellent introduction to the AIC and model selection.

### *Importance of models selection*

The relevance of models of nucleotide substitution in evolutionary studies has been extensively discussed. It is clear that the use of one

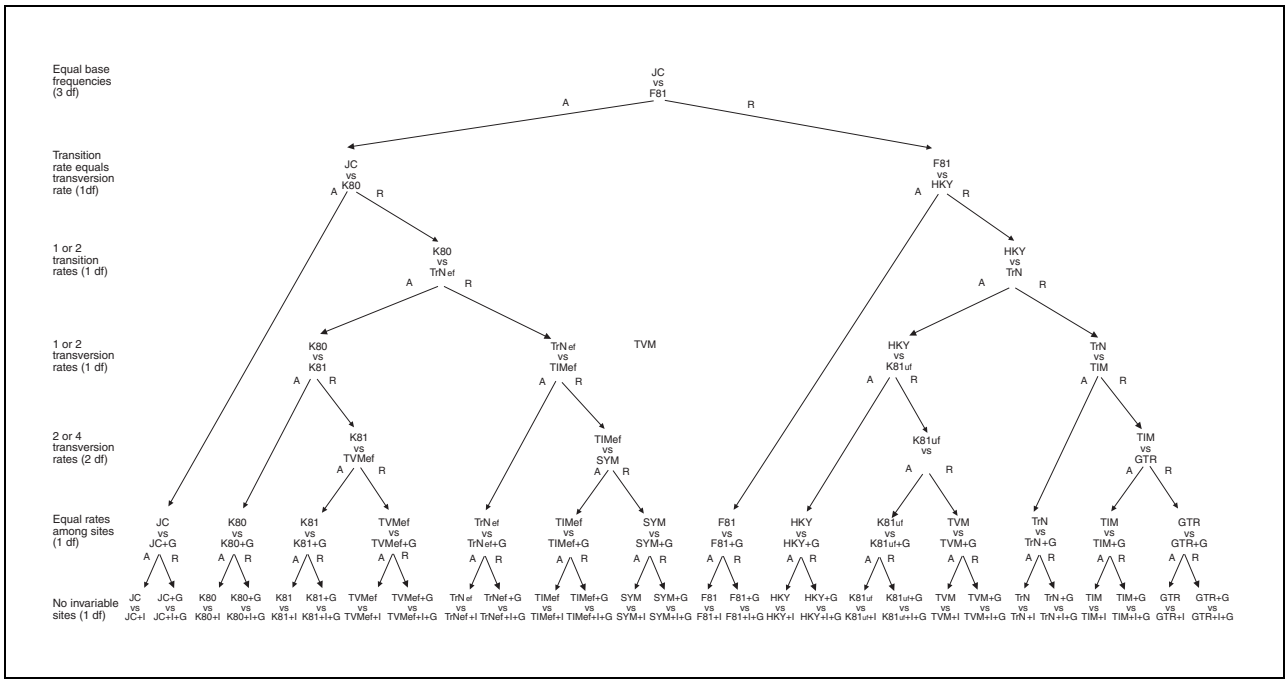


**Figure 6.5.8** MODELTEST hierarchical likelihood ratio tests. At each level the null hypothesis (upper model) is either accepted (A) or rejected (R). The models of DNA substitution are: JC (Jukes and Cantor, 1969), K80 (Kimura, 1980), TrNef (TrN equal base frequencies; see below), K81 (Kimura, 1981), TIMef (TIM with equal base frequencies), TIV (TIV with equal base frequencies), SYM (Zharkikh, 1994), F81 (Felsenstein, 1981), HKY (Hasegawa et al., 1985), TrN (Tamura and Nei, 1993), K81uf (K81 unequal base frequencies; see above), TIM, TIV, and GTR (Rodríguez et al., 1990). Abbreviations: G: shape parameter of the gamma distribution; I: proportion of invariable sites. df: degrees of freedom.

model of evolution or another may change the outcome of the phylogenetic analysis (Leitner et al., 1997; Sullivan and Swofford, 1997; Cunningham et al., 1998; Kelsey et al., 1999; Posada and Crandall, 2001b). The performance of a method is maximized when its assumptions are satisfied, and therefore some indication of the fit of the data to the phylogenetic model is necessary (Huelsenbeck, 1995). In general, phylogenetic methods perform worse when the model of evolution assumed is incorrect (Felsenstein, 1978; Bruno and Halpern, 1999; Huelsenbeck and Hillis, 1993; Huelsenbeck, 1995). Cases where the use of wrong models increases phylogenetic performance (see Yang, 1997; Xia, 2000; Posada and Crandall, 2001c) are exceptional and rather represent a bias towards the true tree associated with violated assumptions (Bruno and Halpern, 1999). However, the relationship between the fit of the model to the data and the ability of the model to correctly predict topology is not completely straightforward (Gaut and Lewis, 1995; Buckley, 2002). Topology estimation by methods such as maximum-likelihood is relatively robust to the model used (Fukami-Kobayashi and Tateno, 1991; Yang et al., 1995; Sullivan and Swofford, 2002). The evaluation of reliability of the estimated trees depends critically on the model; false or simple models tend to suggest that a tree is significantly supported when it cannot be (Yang et al., 1994; Buckley et al., 2001; Buckley and Cunningham, 2002). The use of appropriate models is especially critical for parameter estimation. When a relatively simple model of substitution is assumed, the transition/transversion ratio, branch lengths, and sequence divergence are underestimated, while the shape parameter of the gamma distribution is overestimated (Tamura, 1992; Wakeley, 1994; Yang et al., 1994, 1995; Adachi and Hasegawa, 1995; Buckley et al., 2001). Moreover, the outcome of different tests of evolutionary hypotheses, like the molecular clock, may depend on the model of evolution assumed (Zhang, 1999). Also, simple models of substitution can increase the number of false positives when comparing tree topologies (Buckley, 2002).

## Critical Parameters and Troubleshooting

MODELTEST is a fairly simple program, and the only parameter that can be set is the $\alpha$ level for the individual LRT tests, which by default is set to 0.01. This values aims to provide

a family $\alpha$ level of 0.05, and results from a standard Bonferroni correction considering that there will be 5 or 6 LRT performed (the exact number of LRT cannot be known a priori).

Sometimes problems arise because of the interaction with PAUP*, and the user should check PAUP* Web site in case of problems with the format of the likelihood scores file (`sample.scores`). It is important to distinguish whether the user is running PAUP* or MODELTEST. To keep up to date with problems and or updates of MODELTEST, the user needs to register their copy.

## Literature Cited


Adachi, J. and Hasegawa, M. 1995. Improved dating of the human/chimpanzee separation in the mitochondrial DNA tree: Heterogeneity among amino acid sites. *J. Mol. Evol.* 40:622-628.

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Trans. Autom. Contr.* 19:716-723.

Bruno, W.J. and Halpern, A.L. 1999. Topological bias and inconsistency of maximum likelihood using wrong models. *Mol. Biol. Evol.* 16:564-566.

Buckley, T.R. 2002. Model misspecification and probabilistic tests of topology: Evidence from empirical data sets. *Syst. Biol.* 51:509-523.

Buckley, T.R. and Cunningham, C.W. 2002. The effects of nucleotide substitution model assumptions on estimates of nonparametric bootstrap support. *Mol. Biol. Evol.* 19:394-405.

Buckley, T.R., Simon, C., and Chambers, G.K. 2001. Exploring among-site rate variation models in a maximum likelihood framework using empirical data: The effects of model assumptions on estimates of topology, edge lengths, and bootstrap support. *Syst. Biol.* 50:67-86.

Burnham, K.P. and Anderson, D.R. 1998. Model Selection and Inference: A Practical Information-Theoretic Approach. Springer-Verlag, New York.

Cunningham, C.W., Zhu, H., and Hillis, D.M. 1998. Best-fit maximum-likelihood models for phylogenetic inference: Empirical tests with known phylogenies. *Evolution* 52:978-987.

Felsenstein, J. 1978. Cases in which parsimony or compatibility methods will be positively misleading. *Syst. Zool.* 27:401-410.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.* 17:368-376.

Frati, F., Simon, C., Sullivan, J., and Swofford, D.L. 1997. Gene evolution and phylogeny of the mitochondrial cytochrome oxidase gene in Collembola. *J. Mol. Evol.* 44:145-158.

Fukami-Kobayashi, K. and Tateno, Y. 1991. Robustness of maximum likelihood tree estimation against different patterns of base substitutions. *J. Mol. Evol.* 32:79-91.

Gaut, B.S. and Lewis, P.O. 1995. Success of maximum likelihood phylogeny inference in the four-taxon case. *Mol. Biol. Evol.* 12:152-162.

Goldman, N. 1993. Simple diagnostic statistical test of models of DNA substitution. *J. Mol. Evol.* 37:650-661.

Goldman, N. and Yang, Z. 1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.* 11:725-736.

Hasegawa, M., Kishino, K., and Yano, T. 1985. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.* 22:160-174.

Huelsenbeck, J.P. 1995. Performance of phylogenetic methods in simulation. *Syst. Biol.* 44:17-48.

Huelsenbeck, J.P. 2002. Testing a covariotide model of DNA substitution. *Mol. Biol. Evol.* 19:698-707.

Huelsenbeck, J.P. and Hillis, D.M. 1993. Success of phylogenetic methods in the four-taxon case. *Syst. Biol.* 42:247-264.

Huelsenbeck, J.P. and Crandall, K.A. 1997. Phylogeny estimation and hypothesis testing using maximum likelihood. *Annu. Rev. Ecol. Syst.* 28:437-466.

Huelsenbeck, J.P. and Rannala, B. 1997. Phylogenetic methods come of age: Testing hypothesis in an evolutionary context. *Science* 276:227-232.

Huelsenbeck, J.P. and Nielsen, R. 1999. Variation in the pattern of nucleotide substitution across sites. *J. Mol. Evol.* 48:86-93.

Jukes, T.H. and Cantor, C.R. 1969. Evolution of protein molecules. *In* Mammalian Protein Metabolism (H.M. Munro, eds.) pp. 21-132. Academic Press, New York.

Kelsey, C.R., Crandall, K.A., and Voevodin, A.F. 1999. Different models, different trees: The geographic origin of PTLV-I. *Mol. Phylogenet. Evol.* 13:336-347.

Kimura, M. 1980. A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.* 16:111-120.

Kimura, M. 1981. Estimation of evolutionary distances between homologous nucleotide sequences. *Proc. Natl. Acad. Sci. USA* 78:454-458.

Kullback, S. and Leibler, R.A. 1951. On information and sufficiency. *Ann. Math. Stat.* 22:79-86.

Leitner, T., Kumar S., and Albert, J. 1997. Tempo and mode of nucleotide substitutions in *gag* and *env* gene fragments in human immunodeficiency virus type 1 populations with a known transmission history. *J. Virol.* 71:4761-4770.

Muse, S.V. and Gaut, B.S. 1994. A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. *Mol. Biol. Evol.* 11:715-724.

Posada, D. 2001. The effect of branch length variation on the selection of models of molecular evolution. *J. Mol. Evol.* 52:434-444.

Posada, D. and Crandall, K.A. 1998. Modeltest: Testing the model of DNA substitution. *Bioinformatics* 14:817-818.

Posada, D. and Crandall, K.A. 2001a. Selecting the best-fit model of nucleotide substitution. *Syst. Biol.* 50:1-22.

Posada, D. and Crandall, K.A. 2001b. Selecting models of nucleotide substitution: An application to human immunodeficiency virus 1 (HIV-1). *Mol. Biol. Evol.* 18:897-906.

Posada, D. and Crandall, K.A. 2001c. Simple (wrong) models for complex trees: Empirical bias. *Mol. Biol. Evol.* 18:271-275.

Rodríguez, F., Oliver, J.F., Marín, A., and Medina, J.R. 1990. The general stochastic model of nucleotide substitution. *J. Theor. Biol.* 142:485-501.

Schöniger, M. and von Haeseler, A. 1994. A stochastic model for the evaluation of autocorrelated DNA sequences. *Mol. Phylogenet. Evol.* 3:240-247.

Sullivan, J. and Swofford, D.L. 1997. Are guinea pigs rodents? The importance of adequate models in molecular phylogenies. *J. Mamm. Evol.* 4:77-86.

Sullivan, J. and Swofford, D.L. 2002. Should we use model-based methods for phylogenetic inference when we know that assumptions about among-site rate variation and nucleotide substitution pattern are violated? *Syst. Biol.* 50:723-729.

Swofford, D.L. 2000. PAUP*. Phylogenetic Analysis Using Parsimony (* and Other Methods). Sinauer Associates, Sunderland, Mass.

Swofford, D.L., Olsen, G.J., Waddell, P.J., and Hillis, D.M. 1996. Phylogenetic inference. *In* Molecular Systematics (D.M. Hillis, C., Moritz, and B.K., Mable, eds.) pp. 407-514. Sinauer Associates, Sunderland, Mass.

Tamura, K. 1992. Estimation of the number of nucleotide substitutions when there are strong transition-transversion and G+C content biases. *Mol. Biol. Evol.* 9:678-687.

Tamura, K. and Nei, M. 1993. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.* 10:512-526.

Thorne, J., Kishino, H., and Painter, I.S. 1998. Estimating the rate of evolution of the rate of molecular evolution. *Mol. Biol. Evol.* 15:1647-1657.

Tuffley, C. and Steel, M. 1998. Modeling the covarion hypothesis of nucleotide substitution. *Math. Biosci.* 147:63-91.

Wakeley, J. 1994. Substitution-rate variation among sites and the estimation of transition bias. *Mol. Biol. Evol.* 11:436-442.

Whelan, S. and Goldman, N. 1999. Distributions of statistics used for the comparison of models of sequence evolution in phylogenetics. *Mol. Biol. Evol.* 16:1292-1299.

Xia, X. 2000. Phylogenetic relationships among horseshoe crab species: Effect of substitution models in phylogenetic analysis. *Syst. Biol.* 49:87-100.

Yang, Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *J. Mol. Evol.* 39:306-314.

Yang, Z. 1996. Among-site rate variation and its impact on phylogenetic analysis. *Trends Ecol. Evol.* 11:367-372.

Yang, Z. 1997. How often do wrong models produce better phylogenies? *Mol. Biol. Evol.* 14:105-108.

Yang, Z., Goldman, N., and Friday, A. 1994. Comparison of models for nucleotide substitution used in maximum-likelihood phylogenetic estimation. *Mol. Biol. Evol.* 11:316-324.

Yang, Z., Goldman, N., and Friday, A. 1995. Maximum likelihood trees from DNA sequences: A peculiar statistical estimation problem. *Syst. Biol.* 44:384-399.

Zhang, J. 1999. Performance of likelihood ratio tests of evolutionary hypotheses under inadequate substitution models. *Mol. Biol. Evol.* 16:868-875.

Zharkikh, A. 1994. Estimation of evolutionary distances between nucleotide sequences. *J. Mol. Evol.* 39:315-329.

## Key References

Burnham and Anderson. 1998. See above.

*This book provides a very clear and accessible explanation of different issues around model selection, particularly for the AIC. The book is written by ecologists and it includes many biological examples. A fundamental reference for any biologist doing data analysis.*

Posada and Crandall. 2001a. See above.

*A simulation study of the performance of different strategies for selecting models of substitution. Includes a detailed description of the different selection strategies.*

Swofford et al., 1996. See above.

*This chapter is still the most comprehensive review of phylogenetic inference to date. It provides a detailed description of several substitution models and their use in phylogenetics.*

## Internet Resources

http://www.evolgenics.com/software

*The MODELTEST Web site.*

http://paup.csit.fsu.edu/index.html

*The PAUP* Web site.*

Contributed by David Posada
Universidad de Vigo
Vigo, Spain

# Maximum-Likelihood Analysis Using TREE-PUZZLE

Maximum-likelihood (ML) analysis is a statistically well founded and well known method used in many scientific fields. Edwards and Cavalli-Sforza (1964) have proposed ML for phylogenetics, and Felsenstein (1981) made it applicable for molecular sequences. Although the computation time needed for ML analysis is large, recently, the usage of ML methods has substantially increased and has become an important component in molecular sequence analysis and phylogenetics. The ML approach is appealing because it incorporates explicit models of sequence evolution, and also allows statistical tests of evolutionary hypotheses (Page and Holmes, 1998; Felsenstein, 2004).

The TREE-PUZZLE software (Schmidt et al., 2002) applies the ML principle combined with a fast tree search algorithm called Quartet Puzzling to reconstruct phylogenetic trees from biological sequences. The Quartet Puzzling Algorithm uses quartets, i.e., groups of four sequences, to reconstruct large trees guided by the ML values of the quartet tree topologies (Fig. 6.6.1).

TREE-PUZZLE also offers other algorithmic features, such as Likelihood Mapping (Strimmer and von Haeseler, 1997), a method for visualizing the phylogenetic content of multiple sequence alignments. Likelihood Mapping can also be used to evaluate the quartet support for relationships among groups of sequences.

In addition, TREE-PUZZLE implements several other statistical methods to compare different tree topologies.

In this unit, an amino acid alignment is used to explain the main features of TREE-PUZZLE. The dataset comprises the elongation factors EF-Tu/1α and EF-G/2, two genes that duplicated before the split into the three domains of life, Eukaryota, Archaea, and Bacteria (see Table 6.6.1; similar datasets were first studied by Iwabe et al., 1989).

Although a protein example is used here, TREE-PUZZLE can analyze nucleotide and binary data (e.g., restriction digest data) as well.

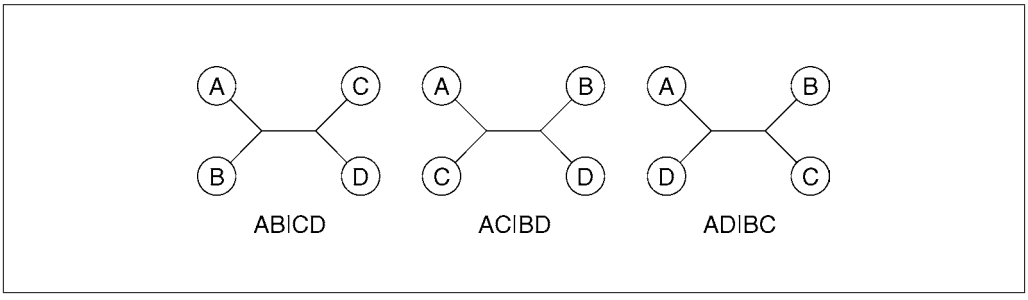


**Figure 6.6.1** The three possible informative quartet tree topologies.



**Contributed by Heiko A. Schmidt and Arndt von Haeseler**
*Current Protocols in Bioinformatics* (2007) 6.6.1-6.6.23

## RECONSTRUCT A PHYLOGENETIC TREE

The main use of TREE-PUZZLE is to reconstruct phylogenetic trees from sequences. The example shows how to use TREE-PUZZLE to construct a tree from amino acid sequences assuming Γ-distributed (Gamma-distributed) rates across sites (*UNIT 6.5*).

### *Necessary Resources*

*Hardware*

TREE-PUZZLE runs on computers with MS Windows, Mac OS, and Unix/Linux operating systems, including workstation clusters and computers using parallel computing

*Software*

TREE-PUZZLE package (see Support Protocols 1 to 3 for information on how to obtain TREE-PUZZLE)

*Files*

Multiple Sequence Alignment file in standard PHYLIP format (see *APPENDIX 1B* and Figure A.1B.3 for a sample PHYLIP format file). The sample data set used here (`EF.phy`) is included with the TREE-PUZZLE software package.

1. Obtain and install TREE-PUZZLE (see Support Protocols 1 to 3).
2. Change to the `data` directory in the TREE-PUZZLE directory and start the program with the command `puzzle EF.phy`.

   *Start* `puzzle` *in a terminal, e.g., Command Prompt (for Windows), Terminal (for Mac OS X; see APPENDIX 1C), or xterm (for Unix/Linux; see APPENDIX 1C & APPENDIX 1D), using the command* `puzzle alignmentfile`*, where* `alignmentfile` *is the name of the file containing the alignment to be analyzed; the example here is* `EF.phy`*. If* `puzzle` *is invoked from a file manager or without a filename, it will search for a file called* `infile` *in the current directory. If* `infile` *does not exist, TREE-PUZZLE will ask for a filename. The* `alignmentfile` *has to be in the current working directory or the full path to its location must be given.*

   IMPORTANT NOTE: *When* `puzzle` *is started by a mouse click, e.g., from the desktop under Windows, Unix/Linux, and Mac OS v. 9.x or lower, the working directory is set to the one in which the executable is located. Under Mac OS X, however, the working directory of a "click-started" instance of* `puzzle` *is set to the user's home folder. When used from a terminal on the command line, the working directory is always the current directory. Thus, the input files should be copied to the working directory or their complete path has to be entered.*
3. Change the type of analysis to `tree reconstruction` (using the `b` key) and the tree search procedure to `quartet puzzling` (using the `k` key), if necessary (Fig. 6.6.2).
4. Adjust the outgroup to the sequence 22 `EFG_MYCGE` (using `o` and the number of the sequence).

   *By default, the first sequence is used to root the resulting tree for output. However, the choice of root has no impact on the log-likelihood.*

   *Note that the natural root lies between EF-α/Tu and EF-2/G (Iwabe et al., 1989). Hence the output tree has to be re-rooted using a phylogeny viewer like TreeView (see UNIT 6.2 and Internet Resources below).*

   *For further discussion of selecting a tree root, see UNIT 6.1.*

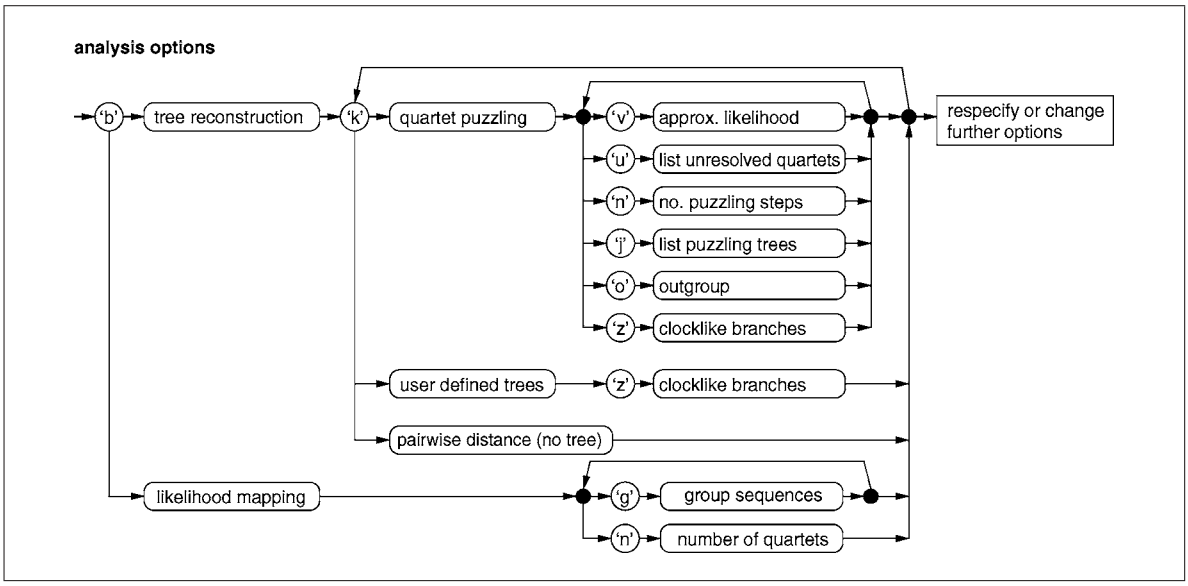


**Figure 6.6.2** Flowchart of analysis type options in the TREE-PUZZLE menu. Options in TREE-PUZZLE are controlled by single letters. The flow chart shows the options that correspond to each letter. For example, entering the letter `b` toggles the analysis between tree reconstruction and likelihood mapping. Similarly, to choose among quartet puzzling, user-defined trees, or pairwise distance matrices, enter the letter `k` until the desired option is shown on the screen.

5. Choose parameter estimation to be performed approximately (with `e`) using neighbor-joining trees (with `x`).

   *Parameters are estimated using tree topologies. These are either inferred by neighbor-joining or given as usertree (usertree evaluation; see Basic Protocol 3). With the* `quartet samples + NJ` *option, the evolutionary parameters are estimated on random quartet samples; neighbor-joining trees are only used for rate parameters.* `Approximate` *estimation uses pairwise distances to fit the branch lengths of the tree topologies, while ML branch lengths are inferred in the* `exact` *estimation.*

### *Choose a model of evolution*

6. Change the type of sequence data to amino acids (using `d`) if the automatically assigned type is not correct (Fig. 6.6.3).

   *Using the character composition of the alignment, TREE-PUZZLE tries to figure out whether the type of data is nucleotide, protein, or binary data.*

7. Choose an appropriate model of sequence evolution to analyze the dataset. For the example alignment, choose the VT model using `m` (Fig. 6.6.3).

   *Several models for protein evolution are implemented in TREE-PUZZLE. While the models by Dayhoff et al. (1978) and Jones et al. (1992) are universal models created from different protein families, more specific models are available, e.g., the mtREV24 model by Adachi and Hasegawa (1996) for mitochondrial protein sequences. Also implemented are the VT (Müller and Vingron, 2000) and the WAG models (Whelan and Goldman, 2001), which are suited to analyze distantly related sequences. The BLOSUM62 matrix (Henikoff and Henikoff, 1992; UNIT 3.5) was designed for database searches and thus should be used with caution for the analysis of evolutionary relationships. For further evolutionary models, refer to the manual. TREE-PUZZLE tries to determine a suitable model by comparing the amino acid frequencies in various models with those of the dataset.*

   *For DNA (Fig. 6.6.4), the HKY (Hasegawa et al., 1985) and TN (Tamura and Nei, 1993) models are available. Those models can be restricted to simpler models like JC (Jukes and Cantor, 1969), K2P (Kimura, 1980), or F84 (Felsenstein, 1984) by setting substitution parameters accordingly. Also the general time-reversible model (GTR; Lanave et al., 1984; Tavare, 1986) is implemented, which can be confined to even more different models by explicitly setting the GTR parameters (refer to the manual and UNITS 6.4 & 6.5 for further*

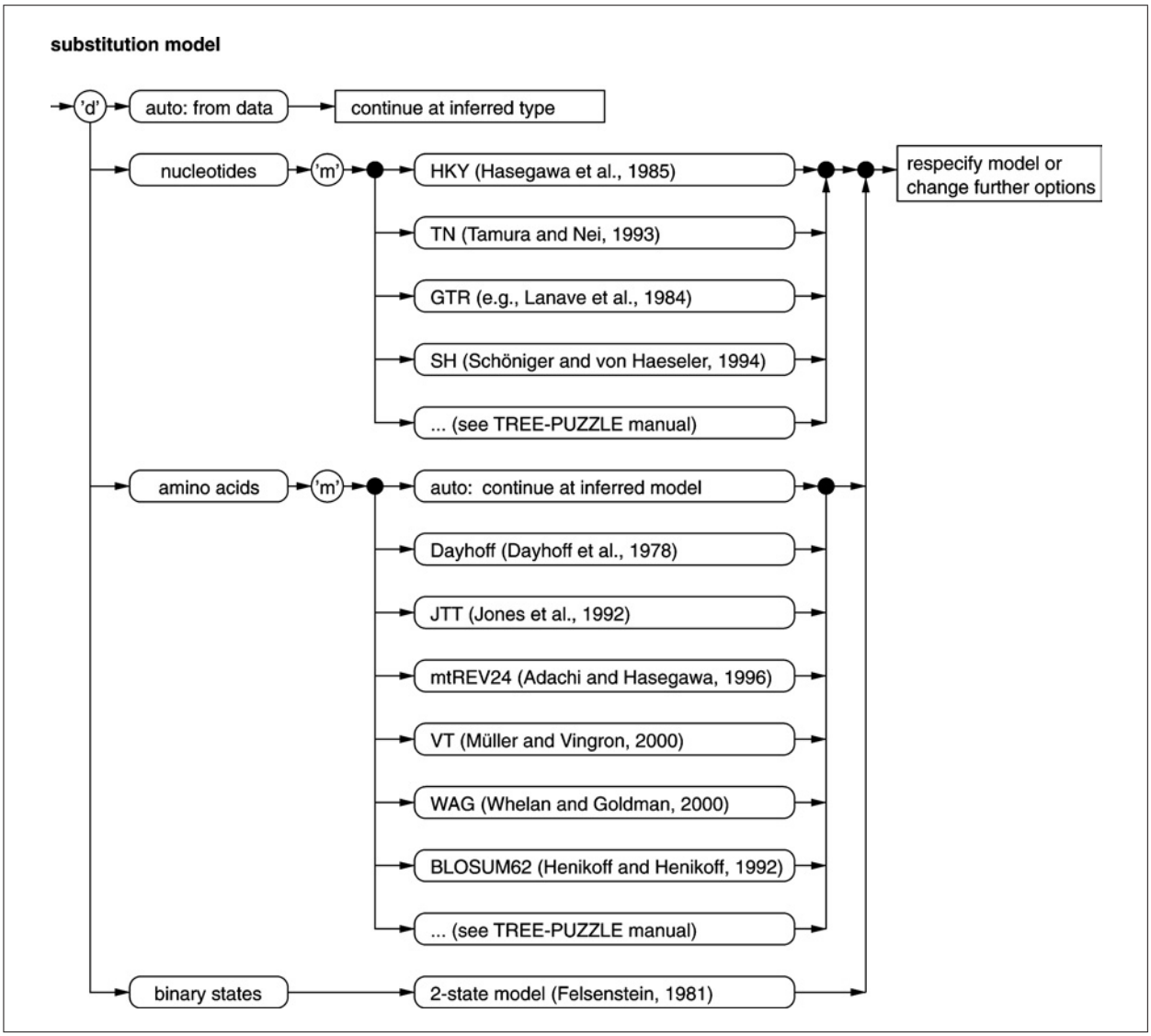


**Figure 6.6.3** Flowchart of substitution model options in the TREE-PUZZLE menu.

*details). Additionally, the SH nucleotide doublet model (Schöniger and von Haeseler, 1994) and a binary model based on the model of Felsenstein (1981) are implemented in TREE-PUZZLE.*

8. Choose Γ-distributed rate heterogeneity by typing `w` (Fig. 6.6.5).

   *It is known that positions in an alignment do not evolve with the same evolutionary rates, typically attributed to selective pressure or other functional constraints acting on positions of the sequence. In such cases, the assumption of rate heterogeneity can improve the estimation of the branch lengths.*

   *Three different models of rate heterogeneity are implemented in TREE-PUZZLE. Besides Γ-distributed rates, there is the two-rates model that assumes a fraction of the positions to be invariable and a mixed model that considers the variable sites to evolve according to a Γ distribution. The amount of rate heterogeneity of the Γ-distributed rates is described by the shape parameter $\alpha$, where $\alpha < 1$ describes strong heterogeneity, while large values describe homogeneity (for more details, refer to Gu et al., 1995; Page and Holmes, 1998; Felsenstein, 2004; UNITS 6.4 & 6.5).*

   *If tree reconstructions with and without the assumption of rate heterogeneity construct different trees, those trees can be compared as described in Basic Protocol 3 to find out whether the resulting tree topologies are significantly different.*

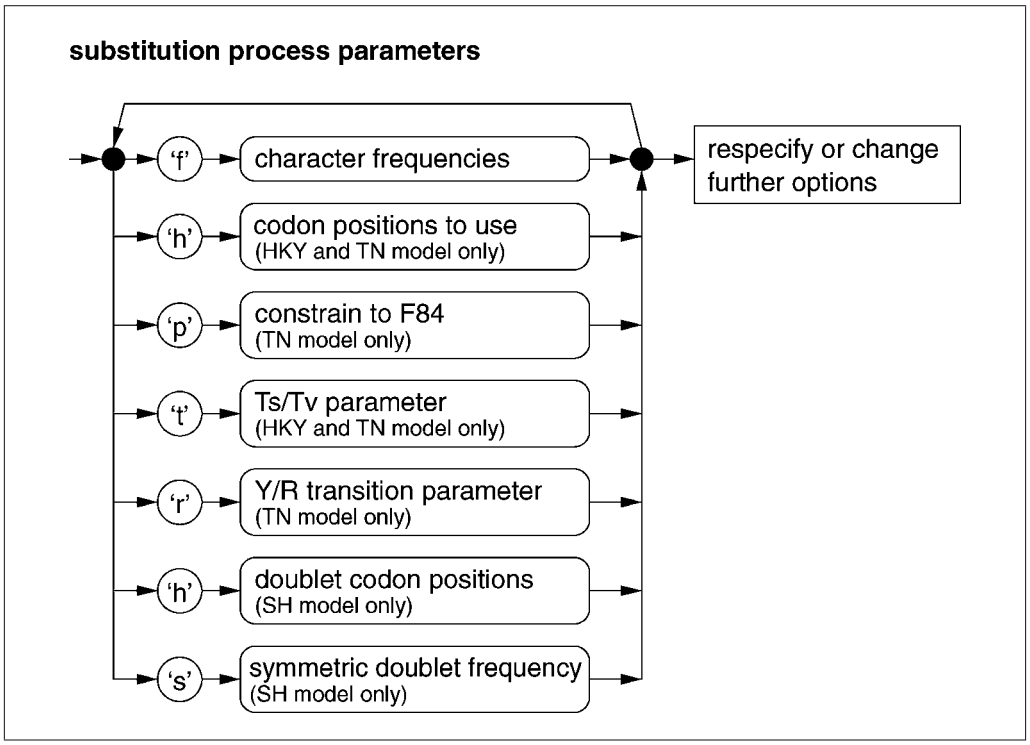


**Figure 6.6.4** Flowchart of further substitution model parameters in the TREE-PUZZLE menu.

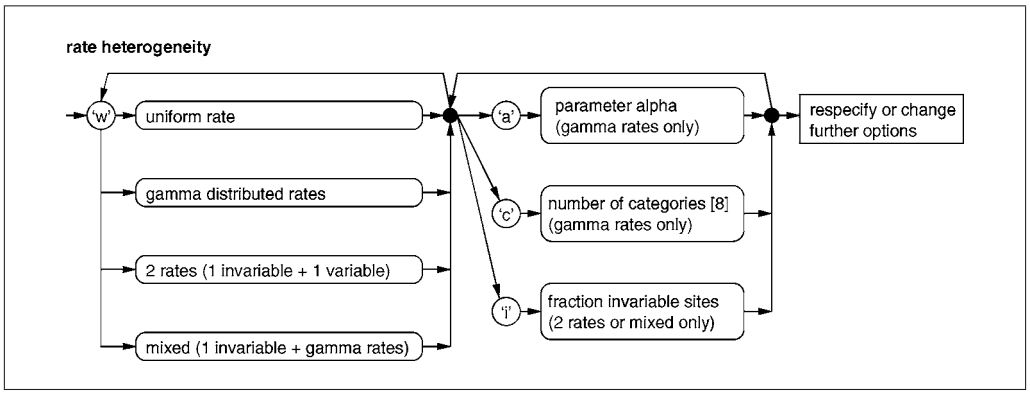


**Figure 6.6.5** Flowchart of rate heterogeneity options in the TREE-PUZZLE menu.

9. Set the `list puzzling step trees` option to `unique topologies` with the `j` key, to make TREE-PUZZLE write all (unique) intermediate tree topologies to file (`EF.phy.ptorder`).

   *When doing one's own analysis, it might be necessary to change other parameters. Many other parameters and options can be set manually. For instance, it is possible to specify the amino acid or nucleotide composition. Figures 6.6.2, 6.6.3, 6.6.4, 6.6.5, and 6.6.6 summarize all options currently available in TREE-PUZZLE. More details are given in the manual.*

10. Start analysis by typing `y`.

    *TREE-PUZZLE will now perform a tree reconstruction. During its run, it will indicate which steps are performed: first the missing parameters are estimated, then all possible quartet maximum-likelihood trees are computed, which are subsequently used to compute intermediate quartet puzzling trees. Finally, the likelihood and the branch lengths of the consensus tree are computed (Fig. 6.6.7).*

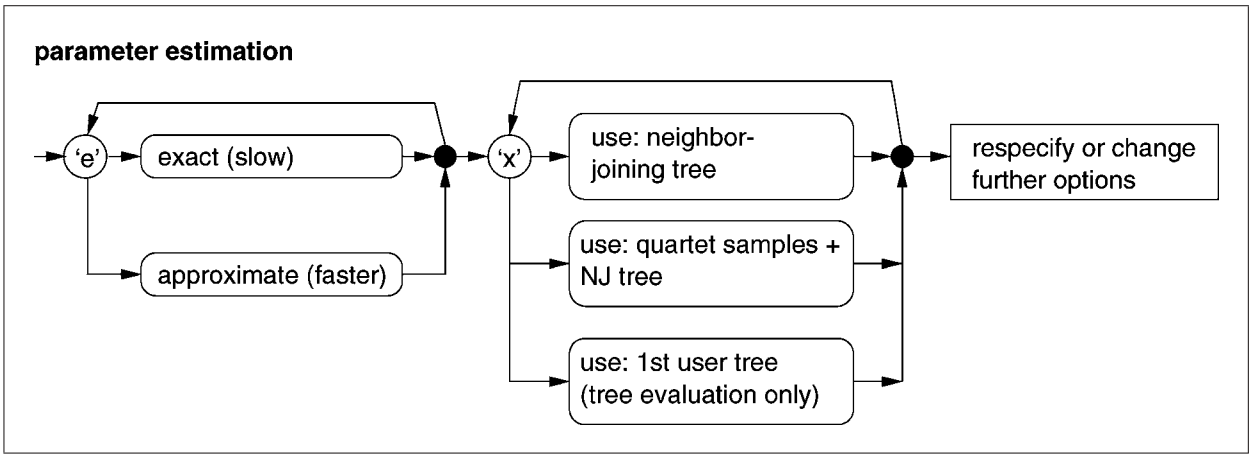


**Figure 6.6.6** Flowchart of parameter estimation options in the TREE-PUZZLE menu.

```
GENERAL OPTIONS
 b                      Type of analysis?  Tree reconstruction
 k                 Tree search procedure?  Quartet puzzling
 v        Approximate quartet likelihood?  Yes
 u               List unresolved quartets?  No
 n               Number of puzzling steps?  1000
 j               List puzzling step trees?  Unique topologies
 o                    Display as outgroup?  EFG_MYCGE
 z      Compute clocklike branch lengths?  No
 e                    Parameter estimates?  Approximate (faster)
 x              Parameter estimation uses?  Neighborjoining tree
SUBSTITUTION PROCESS
 d           Type of sequence input data?  Auto: Amino acids
 m                 Model of substitution?  VT (MuellerVingron 2000)
 f               Amino acid frequencies?  Estimate from data set
RATE HETEROGENEITY
 w           Model of rate heterogeneity?  Gamma distributed rates
 a   Gamma distribution parameter alpha?  Estimate from data set
 c       Number of Gamma rate categories?  8

Quit [q], confirm [y], or change [menu] settings:
Optimizing missing rate heterogeneity parameters
Writing parameters to file EF.phy.puzzle
Writing pairwise distances to file EF.phy.dist
Computing quartet maximum likelihood trees
Computing quartet puzzling tree
Computing maximum likelihood branch lengths (without clock)

All results written to disk:
        Puzzle report file:          EF.phy.puzzle
        Likelihood distances:        EF.phy.dist
        Phylip tree file:            EF.phy.tree
        Unique puzzling step trees: EF.phy.ptorder
```

**Figure 6.6.7** TREE-PUZZLE menu setting and screen output from tree reconstruction.

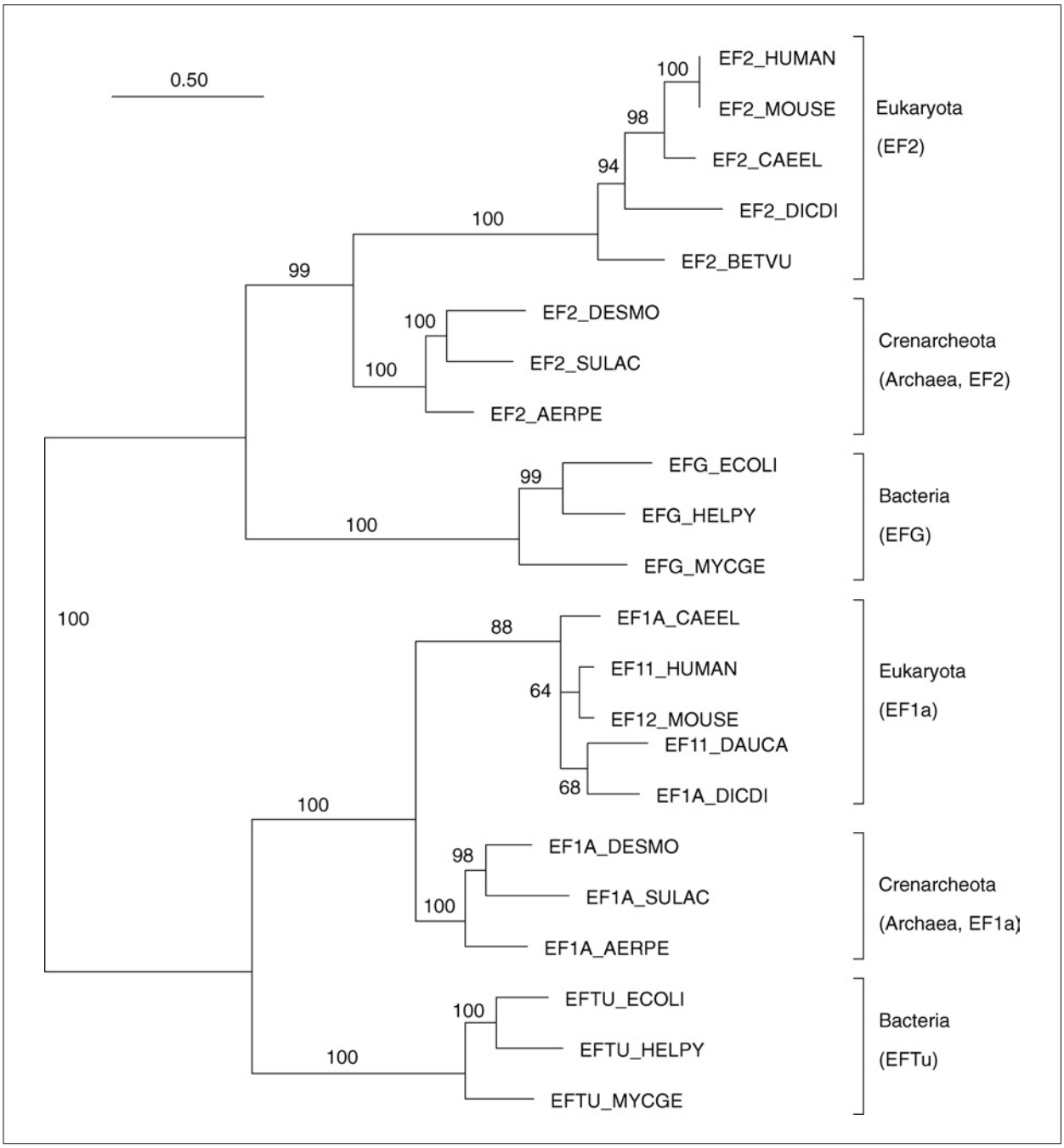


**Figure 6.6.8** Phylogenetic tree reconstructed from the `EF.phy` dataset as described in Basic Protocol 1. The tree is rooted by the duplication event between EF-2/G and EF-1α/Tu.

### *Examine the results*

11. Examine the puzzle report file. The report file is called `EF.phy.puzzle`.

    *The puzzle report file presents details on the quality of the data as well as the reconstructed tree. Hence, it should be thoroughly examined (see Guidelines for Understanding Results below).*

12. Examine the reconstructed tree by viewing the tree file `EF.phy.tree` (Fig. 6.6.8) using a tree-drawing program like TreeView or TreeTool (see *UNIT 6.2* and Internet Resources below).

    *If a program cannot read the trees produced by TREE-PUZZLE, it may be necessary to remove the leading comment (bordered by square brackets). See Guidelines for Understanding Results for help understanding the tree file.*

## ANALYZE THE CONTENT OF PHYLOGENETIC INFORMATION AND THE QUARTET SUPPORT FOR THE RELATIONSHIP OF GROUPS OF SEQUENCES

Likelihood mapping provides the opportunity to either check the content of phylogenetic information in an alignment or to estimate the quartet support of relationships among groups of sequences. The former helps determine whether the data are suitable for phylogenetic analysis by measuring the resolution of the quartet topologies (i.e., trees of four sequences). It is recommended that this check be run especially on large datasets, to avoid spending days or maybe even weeks on phylogenetic analyses with data that contain little phylogenetic information. The latter method partitions a dataset into two to four clusters (i.e., groups of sequences). Likelihood mapping then visualizes which of the possible relationships between these clusters is most supported by the reconstructed quartet tree topologies (Fig. 6.6.1). This method is also useful for reducing the runtime, if the goal is to examine one special bipartition of a tree in a large dataset.

The EF data (Table 6.6.1) will serve as an example of both techniques. First, the suitability of the alignment for phylogenetic analysis is measured (step 4a), then the quartet support for the relationship of four subsets of the dataset is studied (step 4b) in more detail.

**Table 6.6.1** Sequences and Their Accession Numbers Used in the Test Dataset (`EF.phy`)

| Sequence type | Identifier | Accession no. | Species name |
|---|---|---|---|
| Bacterial EF-Tu | EFTU_ECOLI | P02990 | *Escherichia coli* |
| | EFTU_HELPY | P56003 | *Helicobacter pylori* |
| | EFTU_MYCGE | P13927 | *Mycoplasma genitalium* |
| Crenarchaeotic EF-1α (Archaea) | EF1A_DESMO | P41203 | *Desulfurococcus mobilis* |
| | EF1A_SULAC | P17196 | *Sulfolobus acidocaldarius* |
| | EF1A_AERPE | Q9YAV0 | *Aeropyrum pernix* |
| Eukaryotic EF-1α | EF11_HUMAN | P04720 | *Homo sapiens* |
| | EF12_MOUSE | P27706 | *Mus musculus* |
| | EF1A_CAEEL | P53013 | *Caenorhabditis elegans* |
| | EF1A_DICDI | P18624 | *Dictyostelium discoideum* |
| | EF11_DAUCA | P29521 | *Daucus carota* |
| Crenarchaeotic EF-2 (Archaea) | EF2_DESMO | P33159 | *Desulfurococcus mobilis* |
| | EF2_SULAC | P23112 | *Sulfolobus acidocaldarius* |
| | EF2_AERPE | Q9YC19 | *Aeropyrum pernix* |
| Eukaryotic EF-2 | EF2_HUMAN | P13639 | *Homo sapiens* |
| | EF2_MOUSE | P58252 | *Mus musculus* |
| | EF2_CAEEL | P29691 | *Caenorhabditis elegans* |
| | EF2_DICDI | P15112 | *Dictyostelium discoideum* |
| | EF2_BETVU | O23755 | *Beta vulgaris* |
| Bacterial EF-G | EFG_ECOLI | P02996 | *Escherichia coli* |
| | EFG_HELPY | P56002 | *Helicobacter pylori* |
| | EFG_MYCGE | P47335 | *Mycoplasma genitalium* |

***Necessary Resources***

*Hardware*

TREE-PUZZLE runs on computers with MS Windows, Mac OS, and Unix/Linux operating systems, including workstation clusters and computers using parallel computing

*Software*

TREE-PUZZLE package (see Support Protocols 1 to 3 for information on how to obtain TREE-PUZZLE)

*Files*

Multiple Sequence Alignment file in standard PHYLIP format (see *APPENDIX 1B* and Figure A.1B.3 for a sample PHYLIP format file). The sample data set used here (`EF.phy`) is included with the TREE-PUZZLE software download.

1. Obtain and install TREE-PUZZLE (see Support Protocols 1 to 3).
2. Change to the data directory in the TREE-PUZZLE directory and start `puzzle` with the command `puzzle EF.phy`.

   *Start* `puzzle` *in a terminal, e.g., Command Prompt (Windows), Terminal (Mac OS X; APPENDIX 1C), or xterm (for Unix/Linux; APPENDIX 1C & APPENDIX 1D) using the command* `puzzle alignmentfile`*, where* `alignmentfile` *is the name of the file containing the alignment to be analyzed; the example here is* `EF.phy`*. If* `puzzle` *is invoked from a file manager or without a filename, it will search for a file called* `infile` *in the current directory. If* `infile` *does not exist, TREE-PUZZLE will ask for a filename. The* `alignmentfile` *has to be in the current working directory or the full path to its location must be given.*

   *See Basic Protocol 1, step 2, for working directory issues on various platforms.*
3. Change the type of analysis to `Likelihood mapping` (using the `b` key).

4a. Leave the sequences ungrouped for a general likelihood mapping analysis to test the dataset.

4b. Group the sequences into four clusters (using `g`). Assign crenarchaeotic EF-2 to cluster `a`, bacterial EF-G to `b`, eukaryotic EF-2 to `c`, and all EF-1α/Tu sequences to cluster `d` (Table 6.6.1).

   *To analyze the phylogenetic relationship among groups of sequences, define two to four disjoint sets of sequences from the alignment by assigning each sequence the name of the cluster* `a` *through* `d` *(in the case of less than four clusters,* `c` *and/or* `d` *are not valid). Assigning* `x` *will exclude a sequence from the analysis. Each sequence must be labeled* `a`*,* `b`*, (*`c`*,* `d`*), or* `x`*.*

   *A two-cluster analysis will check for the quartet support for bipartition into the two clusters, whereas a four-cluster analysis will infer the quartet support for any of the three possible relationships of the four clusters, namely (ab|cd), (ac|bd), or (ad|bc). where "|" denotes the inner branch that separates the groups (Fig. 6.6.1).*

***Choose a model of evolution (for more information, see Basic Protocol 1, steps 6 to 9)***

5. Change the type of sequence data (using `d`) if the automatically assigned type is wrong.

   *TREE-PUZZLE should have set the data type correctly to amino acids for the example.*
6. Choose an appropriate model of evolution to analyze a dataset. For the example alignment, choose the VT model with the `m` key.
7. Choose Γ-distributed rate heterogeneity by typing `w`.

8. Change other parameters, if necessary. For this example, leave the parameters unchanged.

   *The number of quartets used in the analysis can be set by the* `n` *option. If the number of existing quartets is larger than the specified number, a random subset of all possible quartets is chosen by default, but the size of the sample is also adjustable.*

9. Start analysis by typing `y`.

   *TREE-PUZZLE will now perform a likelihood-mapping analysis. During the run, it will indicate which steps are performed: first the missing parameters are estimated, then the likelihood-mapping analysis is performed, evaluating quartet maximum-likelihood trees. For large datasets, a random subset of quartets is analyzed (Fig. 6.6.9).*

***Examine the results***

10. Examine the puzzle report file. The report file is called `EF.phy.puzzle`, if starting with the alignment file `EF.phy`.

    *The puzzle report file presents the quality of the data as well as the results of the likelihood mapping. Hence, it should be thoroughly examined.*

11. Examine the likelihood-mapping diagram (Figs. 6.6.10, 6.6.11, and 6.6.12), `EF.phy.eps`, using a PostScript browser like Ghostscript/Ghostview (see Internet Resources).

    *See Guidelines for Understanding Results for help in interpreting these diagrams.*

```
GENERAL OPTIONS
 b                      Type of analysis?  Likelihood mapping
 g           Group sequences in clusters?  No
 n                    Number of quartets?  7315 (all possible)
 e                   Parameter estimates?  Approximate (faster)
 x             Parameter estimation uses?  Neighborjoining tree
SUBSTITUTION PROCESS
 d           Type of sequence input data?  Auto: Amino acids
 m                 Model of substitution?  VT (Mueller-Vingron 2000)
 f                Amino acid frequencies?  Estimate from data set
RATE HETEROGENEITY
 w           Model of rate heterogeneity?  Gamma distributed rates
 a   Gamma distribution parameter alpha?  Estimate from data set
 c       Number of Gamma rate categories?  8

Quit [q], confirm [y], or change [menu] settings:
Optimizing missing rate heterogeneity parameters
Writing parameters to file EF.phy.puzzle
Writing pairwise distances to file EF.phy.dist
Performing likelihood mapping analysis

All results written to disk:
        Puzzle report file:          EF.phy.puzzle
        Likelihood distances:        EF.phy.dist
        Likelihood mapping diagram: EF.phy.eps
```

**Figure 6.6.9** TREE-PUZZLE menu setting and screen output from likelihood-mapping analysis.

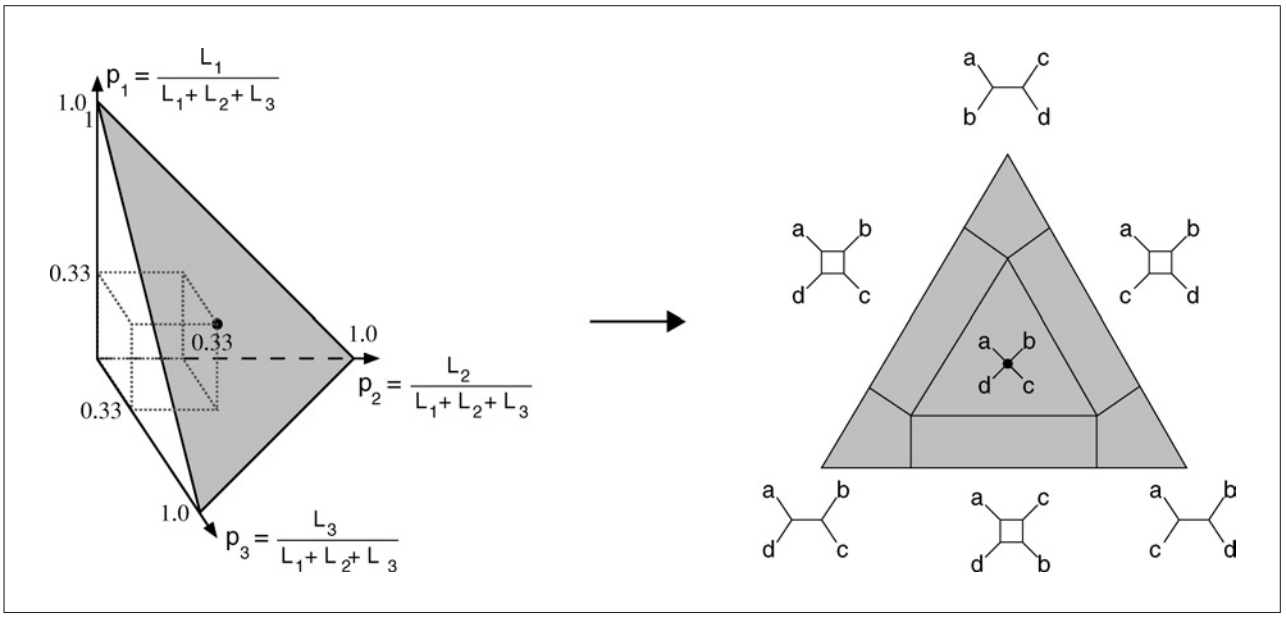


**Figure 6.6.10** How likelihood weights are plotted in a likelihood-mapping diagram. Left side: likelihood weight plotted in a three-dimensional coordinate system. Right side: the simplex and its areas and the corresponding quartet topologies. The gray triangles are identical, only viewed from different angles.

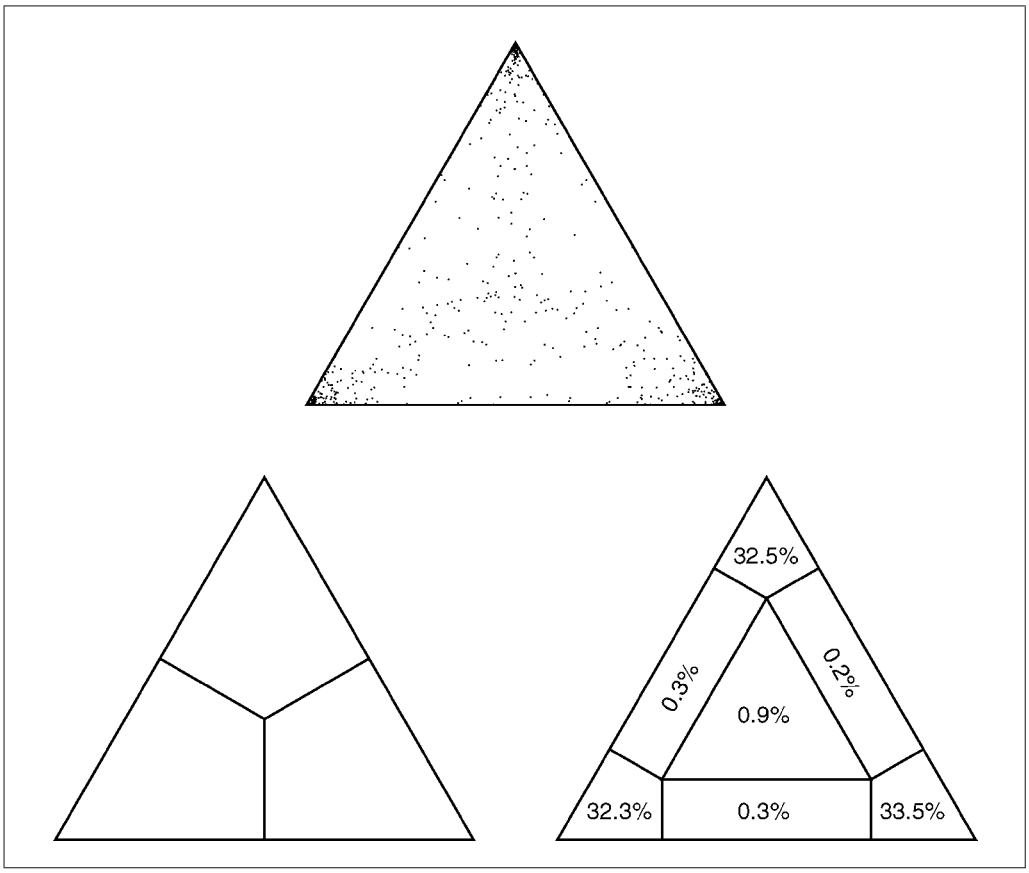


**Figure 6.6.11** Likelihood-mapping diagram visualizing the phylogenetic content of the `EF.phy` dataset performed as described in Basic Protocol 2.

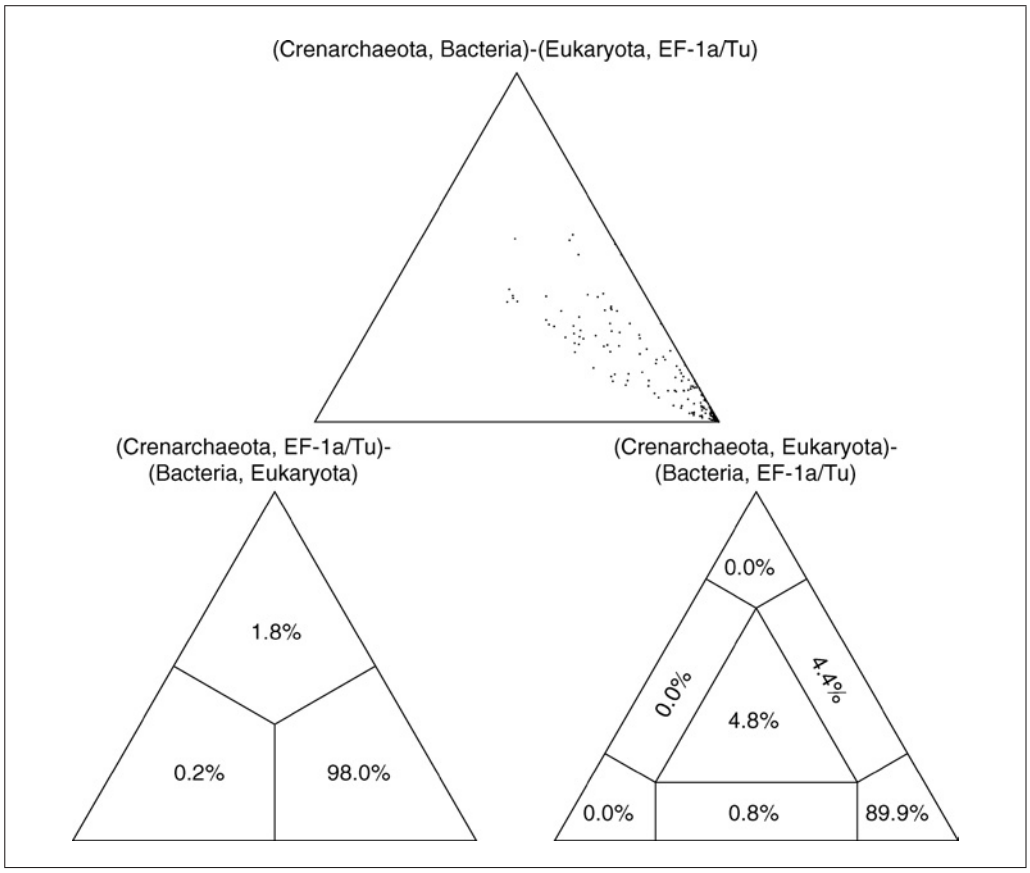


**Figure 6.6.12** Likelihood-mapping diagram visualizing the support for a Crenarchaeota-Eukaryota sister group in the EF-2/G genes of the `EF.phy` dataset as described in Basic Protocol 2.

***BASIC PROTOCOL 3***

## COMPARE TREE TOPOLOGIES

A third type of analysis implemented in TREE-PUZZLE is the likelihood-based comparison of two or more tree topologies using the tests suggested by Kishino and Hasegawa (1989), Shimodaira and Hasegawa (1999), and the so-called expected likelihood weights (Strimmer and Rambaut, 2002). These tests compare different trees to evaluate something like a confidence set of trees. The example used here is a dataset together with a set of trees with different branching patterns, comprising the tree reconstructed in Basic Protocol 1 and two trees with the different possible relationships of Crenarchaeota, Bacteria, and Eukaryota (Fig. 6.6.13).

### *Necessary Resources*

*Hardware*

TREE-PUZZLE runs on computers with MS Windows, Mac OS, and Unix/Linux systems, including workstation clusters and computers using parallel computing

*Software*

TREE-PUZZLE package (see Support Protocols 1 to 3 for information on how to obtain TREE-PUZZLE)

*Files*

Multiple Sequence Alignment file in standard PHYLIP format (see *APPENDIX 1B* and Figure A.1B.3 for a sample PHYLIP format file). A tree file containing the

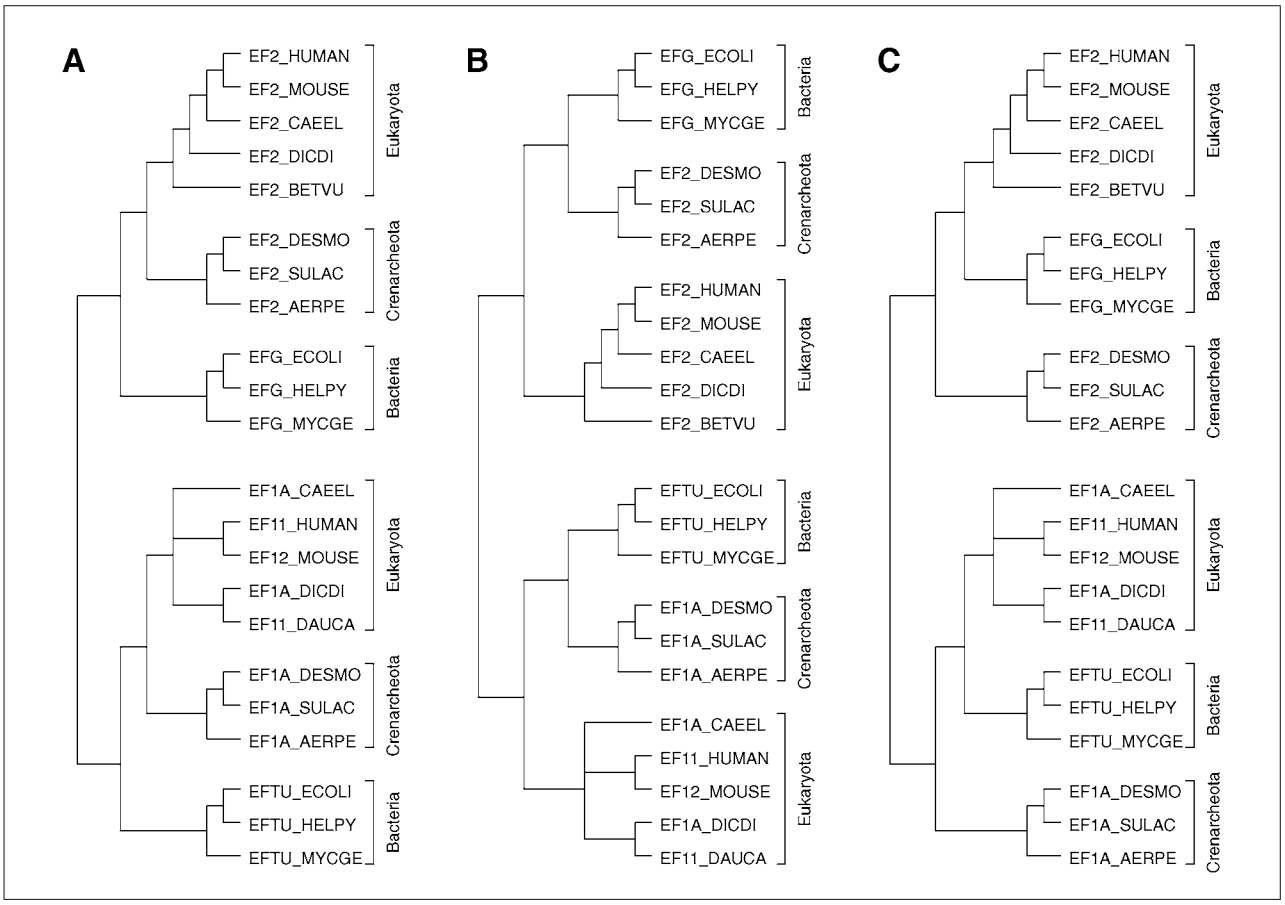


**Figure 6.6.13** The three tree topologies used in the usertree comparison. (**A**) Tree 1: Eukaryota-Crenarchaeota sister groups, (**B**) Tree 2: Bacteria-Crenarchaeota sister groups, (**C**) Tree 3: Eukaryota-Bacteria sister groups. The tree topologies are used without branch lengths.

usertrees in Newick tree format as produced by many programs, e.g., PHYLIP or TREE-PUZZLE. (trees can span several lines and contain comments; for more information see *UNIT 6.2*). The sample data set used here is included with the TREE-PUZZLE software download.

1. Obtain and install TREE-PUZZLE (see Support Protocols 1 to 3).

2. Change to the `data` directory in the `TREE-PUZZLE` directory and start puzzle with the command `puzzle EF.phy EF.3trees`.

   *Start* `puzzle` *in a terminal, e.g., Command Prompt (for Windows), terminal (for Mac OS X; see APPENDIX 1C), or xterm (for Unix/Linux; see APPENDIX 1C & APPENDIX 1D) using the command* `puzzle alignmentfile usertreefile`, *where* `alignmentfile` *is the name of the file containing the alignment to be analyzed and* `usertreefile` *is the name of the file that contains the tree topologies for comparison. If* `puzzle` *is invoked from a desktop or from the command line without filename arguments, it will search for the files* `infile` *and* `intree` *in the current directory. If* `infile` *and/or* `intree` *does not exist, TREE-PUZZLE will ask for a filename. The* `alignmentfile` *and* `usertreefile` *have to be in the current working directory or the full paths to their respective locations must be given.*

   *See Basic Protocol 1, step 2 for working directory issues on various platforms.*

3. Change the type of analysis to `tree reconstruction` (using the `b` key) and the tree search procedure to `user defined trees` (using the `k` key), if necessary.

***Choose a model of evolution (for more information, see Basic Protocol 1, steps 6 to 9)***

4. Change the type of sequence data (using `d`) if the automatically assigned type is wrong. TREE-PUZZLE should have set the data type correctly to amino acids for the example.

5. Choose an appropriate model of evolution to analyze the dataset. For this example alignment, choose the VT model by selecting the `m` option.

6. Choose Γ-distributed rate heterogeneity by typing `w`.

7. Choose neighbor-joining (NJ) tree as the means for the parameter estimation, using the `x` key. Change other parameters, if necessary.

   *For tree evaluation, TREE-PUZZLE uses the first usertree for the parameter estimation by default. This makes sense for the evaluation of single trees, but to test a set of trees, as in this example, an NJ tree should be used to estimate the parameters.*

8. Start the analysis by typing `y`.

   *TREE-PUZZLE will now evaluate and compare the tree topologies in the usertreefile (*`EF.3trees`*). During its run, it will indicate which steps are performed: first, the missing parameters are estimated, then all trees in the usertreefile (*`EF.3trees`*) are evaluated and the results are written to the puzzle report file (Fig. 6.6.14).*

```
GENERAL OPTIONS
 b                      Type of analysis?  Tree reconstruction
 k                  Tree search procedure?  User defined trees
 z     Compute clocklike branch lengths?  No
 e                    Parameter estimates?  Approximate (faster)
 x              Parameter estimation uses?  Neighborjoining tree
SUBSTITUTION PROCESS
 d           Type of sequence input data?  Auto: Amino acids
 m                 Model of substitution?  VT (Mueller-Vingron 2000)
 f               Amino acid frequencies?  Estimate from data set
RATE HETEROGENEITY
 w           Model of rate heterogeneity?  Gamma distributed rates
 a   Gamma distribution parameter alpha?  Estimate from data set
 c      Number of Gamma rate categories?  8

Quit [q], confirm [y], or change [menu] settings:
Optimizing missing rate heterogeneity parameters
Writing parameters to file EF.3trees.puzzle
Writing pairwise distances to file EF.3trees.dist
Computing maximum likelihood branch lengths (without clock) for tree # 1
Computing maximum likelihood branch lengths (without clock) for tree # 2
Computing maximum likelihood branch lengths (without clock) for tree # 3
Performing single sided KH test.
Performing ELW test.
Performing SH test.

All results written to disk:
        Puzzle report file:          EF.3trees.puzzle
        Likelihood distances:        EF.3trees.dist
        Phylip tree file:            EF.3trees.tree
```

**Figure 6.6.14** TREE-PUZZLE menu setting and screen output from usertree evaluation.

```
COMPARISON OF USER TREES (NO CLOCK)

Tree   log L   difference    S.E.     p1sKH      pSH        cELW       2sKH
-------------------------------------------------------------------------------
 1  18965.87      0.00  <---- best    1.0000 +   1.0000 +   0.9123 +    best
 2  18974.43      8.56         5.77    0.0690 +   0.0810 +   0.0846 +      +
 3  18977.24     11.37         4.96    0.0130     0.0130     0.0031        -

The columns show the results and pvalues of the following tests:
1sKH - one sided KH test based on pairwise SH tests (Shimodaira-Hasegawa
       2000, Goldman et al., 2001, Kishino-Hasegawa 1989)
SH   - Shimodaira-Hasegawa test (2000)
ELW  - Expected Likelihood Weight (Strimmer-Rambaut 2002)
2sKH - two sided Kishino-Hasegawa test (1989)

Plus signs denote the confidence sets. Minus signs denote significant
exclusion. All tests used 5% significance level. 1sKH, SH, and ELW
performed 1000 resamplings using the RELL method.
```

**Figure 6.6.15** Results of the comparison of three trees from the `EF.3trees` dataset as described in Basic Protocol 3.

### *Examine the results*

9. Examine the puzzle report file. The report file is called `EF.3trees.puzzle`.

   *The puzzle report file presents the quality of the data as well as the results of the usertree evaluation (Fig. 6.6.15). Hence, it should be thoroughly examined. The file* `EF.3trees.tree` *contains each tree from the usertreefile in NEWICK tree format with estimated branch lengths. The trees can be viewed with tree-drawing programs like TreeView or TreeTool (see UNIT 6.2 and Internet Resources). If a program cannot read such trees, it might be necessary to remove the leading comment (bordered by square brackets).*

   *See Guidelines for Understanding Results for help in interpreting these files.*

## OBTAIN AND INSTALL TREE-PUZZLE FOR UNIX/LINUX AND Mac OS X

***SUPPORT PROTOCOL 1***

This protocol describes how to obtain and install TREE-PUZZLE for Unix/Linux operating systems, including Mac OS X.

### *Necessary Resources*

*Hardware*

Unix/Linux system with TCP/IP Internet connection and a Web browser

*Software*

On Unix systems, an ANSI/ISO C compiler is needed, which is usually delivered with the operating system; otherwise, use the free GNU C compiler (*http://www.gnu.org*)

To use the parallel version of TREE-PUZZLE for supercomputers and workstation clusters, implementation of the MPI library (Message Passing Interface) is needed. There are several free implementations like LAM or MPICH (see *http://www.lam-mpi.org/mpi/implementations* for a list of implementations).

1. Download the current TREE-PUZZLE package for Unix from *http://www.tree-puzzle.de*. It has a name like `tree-puzzle-X.X.tar.gz`, where `X.X` should be the current version.

2. Unpack the package using:

```
gunzip tree-puzzle-X.X.tar.gz
tar -xvf tree-puzzle.X.X.tar
```

This should create a directory `tree-puzzle-X.X`. The subdirectories `doc` and `data` contain the manual and test data, respectively.

3. Change to the `tree-puzzle-X.X` directory.

4. Read the `INSTALL` file and the installation part of the manual carefully. Type the following commands to produce an executable:

```
./configure
./make
```

*The command* `configure` *will determine the system type, and whether all needed software is installed. The* `make` *command will then compile the executable. If* `configure` *finds an MPI library installed,* `make` *will automatically produce the parallel version (*`ppuzzle`*) as well.*

5. To install the executables, run the command:

```
make install
```

*To complete this step, the user will probably need to be logged in as the root user.*

*This will install the executables* `puzzle` *and* `ppuzzle` *(the parallel version).The programs will be installed to* `/usr/local/bin` *by default. If it is necessary to have the programs installed in another directory, change with* `configure` *(see the* `INSTALL` *file for more details) or copy the executables* `src/puzzle` *and/or* `src/ppuzzle` *to the desired location.*

***SUPPORT PROTOCOL 2***

## OBTAIN AND INSTALL TREE-PUZZLE FOR Mac OS X

This protocol describes how to obtain and install TREE-PUZZLE for the Macintosh operating system, Mac OS X.

### *Necessary Resources*

*Hardware*

Macintosh system with TCP/IP Internet connection running Mac OS X and a Web browser

1. Download the current TREE-PUZZLE package for Mac OS X from *http://www.tree-puzzle.de*. It has a name like `tree-puzzle-X.X.tar.gz`, where `X.X` should be the current version.

2. Unpack the package using a program like Stuffit (*http://www.stuffit.com*), which should belong to the Mac OS X release.

*This should create a directory* `tree-puzzle-X.X`*, which contains* `tree-puzzle-YYY` *in its* `src` *folder (*`YYY` *indicating the compiler used). The subdirectories* `doc` *and* `data` *contain the manual and test data, respectively.*

3. Copy the Mac OS X executable to the desired location and renamed to `puzzle` or `tree-puzzle`.

*This location should be in the search* `PATH` *variable (see APPENDIX 1B). For convenience, create a link on the Desktop.*

## OBTAIN AND INSTALL TREE-PUZZLE FOR MS WINDOWS

This protocol describes how to obtain and install TREE-PUZZLE for Windows operating systems.

### *Necessary Resources*

*Hardware*

Windows system with TCP/IP Internet connection and a Web browser

1. Download the current TREE-PUZZLE package for Windows from *http://www.tree-puzzle.de*. It is named `tree-puzzle-X.X.zip`, where `X.X` is the current version.
2. Unpack the package using a program such as WinZip (*http://www.winzip.com*).

   *This should create a directory* `tree-puzzle-X.X`. *The subdirectories* `doc` *and* `data` *contain the manual and test data, respectively. In the* `src` *directory, there is a Windows executable,* `puzzle-windows-YYY.exe`, *where* `YYY` *states the compiler used to prepare the executable.*
3. Copy the Windows executable to the desired location and rename it as `puzzle.exe` or `tree-puzzle.exe`.

   *This location should be in the Windows search path. For convenience, create a link on the Windows Desktop.*

## GUIDELINES FOR UNDERSTANDING RESULTS

### General Aspects

As one can imagine, the outcome of an analysis is highly dependent on the data quality. In an optimal case, the data provide perfect phylogenetic information and no inconsistencies, and hence the resulting tree will show the history of the sequences, which means that the data are perfectly tree-like. Unfortunately, convergent evolution, multiple substitutions, and other processes introduce noise. Thus, careful screening of the data is necessary. TREE-PUZZLE tries to determine if the dataset is suited for phylogenetic analysis.

After running an analysis with TREE-PUZZLE, check the puzzle report file, here called `EF.phy.puzzle` or `EF.3trees.puzzle`. TREE-PUZZLE measures several features of the dataset. In the `SEQUENCE ALIGNMENT` part, it shows the fraction of constant sites as well as how many different columns (site patterns) occur in the alignment. It also checks for identical sequences in the data. Identical sequences should be removed, because they increase computation time and provide no additional information about the phylogeny of the data.

TREE-PUZZLE also estimates the nucleotide composition or amino acid composition of the alignment. It tests if the composition of each sequence (e.g., amino acids or nucleotides) deviates significantly from the average composition. Also, the gaps and ambiguous characters, like `N` in nucleotide sequences and `X` in protein sequences, are counted for each sequence. If a sequence contains many gaps and ambiguous characters, there might not be enough informative characters left to ensure a reliable placement of this sequence in the reconstructed tree.

These features of the data, as well as the resolution of the quartets (in the `QUARTET STATISTICS` part) described below, will help one to find out which of the sequences might have caused inconsistencies in the analysis (see below).

### Tree Reconstruction (see Basic Protocol 1)

To reconstruct phylogenies, TREE-PUZZLE applies a three-step algorithm called Quartet Puzzling (Strimmer and von Haeseler, 1996). In the first step, the maximum-likelihood step (ML step), all possible groups of four sequences, quartets, and their three different topologies (Fig. 6.6.1) are evaluated to create a set of quartet trees supported by the data. This step also determines whether two or even all three tree topologies are almost equally good, i.e., partly resolved or unresolved topologies, respectively (Strimmer et al., 1997). Fully resolved, partly resolved, and unresolved quartets are explained in more detail below (see Likelihood Mapping for Data Quality and Quartet Support of Clusters). In the puzzling step, the supported quartet tree topologies are combined into an overall tree. Since this step is dependent on the input order, it is performed many times for randomized input orders, thus producing a large number of so-called intermediate or puzzle trees. These trees and their frequency can be output to file using the `j` option, as explained in Basic Protocol 1 (see manual for more details; Figure 6.6.2). In the final consensus step, a consensus tree is computed from the intermediate trees, which is then used to infer maximum-likelihood branch lengths and the maximum-likelihood value for the tree, as described in Felsenstein (1981). The percentage of splits (i.e., bipartitions of the dataset induced by an internal edge in a tree) that occurred in the collection of intermediate trees is used as a reliability measure for the splits in the consensus tree. The higher these so-called support values, the more confidence one may place upon the according bipartition. However, never confuse support values with bootstrap values.

If a split does not occur in >50% of the intermediate trees, it is not included in the consensus tree (McMorris and Neumann, 1983). Thus, multifurcations are possible. There is a multifurcation in the eukaryotic EF/1$\alpha$ subtree in Figure 6.6.8.

In the puzzle report file (`EF.phy.puzzle`), all intermediate trees occurring more often than 5% are listed. In addition, the amount of fully resolved, partly resolved, and unresolved quartets for the entire dataset is shown. TREE-PUZZLE also outputs how frequently each sequence occurs in fully resolved, partly resolved, and unresolved quartets. This is another way of displaying phylogenetic information in the data (see Likelihood Mapping below) as well as in any of the sequences. If the reconstructed tree is highly unresolved, the unresolved quartets indicate whether the dataset was not suitable for tree reconstruction (overall fraction of unresolved quartets high) or if there are sequences that should be excluded because they introduce unresolved quartets. If the amount of unresolved quartet for a sequence is high, this sequence should be discarded from the dataset (see below for more details on unresolved quartets).

If the assumption of rate heterogeneity is applied, as in the example, then the report file also displays the site specific rates of each alignment site (`RATE HETEROGENEITY` section in the puzzle report file).

### Likelihood Mapping for Data Quality and Quartet Support of Clusters (see Basic Protocol 2)

Likelihood mapping (Strimmer and von Haeseler, 1997) is based on likelihood values inferred for each of the three possible tree topologies for a quartet (Fig. 6.6.1). Every likelihood value is transferred into a weight (posterior probability) by dividing it by the sum of all three likelihoods (Strimmer et al., 1997). If one of the topologies has a higher likelihood than the others, its weight will be near 1.0 while the other weights are almost zero. If two quartet topologies have similar likelihoods, their weights will be ~0.5, i.e., it is difficult to decide which is the more advantageous topology (partly resolved quartet). For an unresolved quartet, each possible topology has a weight about one-third. The three likelihood weights for a quartet add up to 1.0 and can be plotted in a three-dimensional

coordinate system, one axis for each quartet topology. Each point falls into a triangular surface between (1.0, 0.0, 0.0), (0.0, 1.0, 0.0), and (0.0, 0.0, 1.0), as shown on the left side of Figure 6.6.10. Likelihood mapping plots the likelihood weights directly into such a triangle, also called simplex (Fig. 6.6.10, right side).

The likelihood mapping output (Figs. 6.6.12 and 6.6.14) comprises two different illustrations of the distribution of quartet weights in the simplex. One simplex is divided into three areas. Each area represents the region where a maximum-likelihood reconstruction would reconstruct the tree at the corner of the simplex. The second simplex is partitioned into seven regions. The central region represents the area of unresolved quartets. The three rectangles illustrate partly resolved quartets and the three trapezoids reflect fully resolved quartets, defined by the trees in the corner (Fig. 6.6.10). Each point represents the likelihood of a single quartet.

In an unrestricted likelihood mapping, all quartets are used for analysis, whereas in a grouped analysis, quartets are chosen according to the 2 to 4 assigned clusters:

4 clusters: (a,b,c,d) with $a \in A$, $b \in B$, $c \in C$, $d \in D$, where A, B, C, and D are the clusters
3 clusters: (a,a′,b,c) with $a,a' \in A$, $b \in B$, $c \in C$ $a \in A$, where A, B, and C are the clusters
2 clusters: (a,a′,b,b′) with $a,a' \in A$, $b,b' \in B$, where A and B are the clusters

The 4-cluster analysis is applied to evaluate the support for the phylogenetic relationships of four disjoint groups of sequences (cluster A, B, C, D). The 3-cluster analysis helps, inter alia, to elucidate the reliability of an outgroup by visualizing how well the representatives of a cluster A are separated from a sister group B joined with the outgroup C. If many points are visible in the lower corners of the triangle, the chosen outgroup might be poorly separated from the ingroup; if many points are visible in center of the triangle, its sequences show saturation of the phylogenetic signal. Finally, the 2-cluster analysis reveals how many quartets support the split induced by the two clusters A and B.

The results of the two likelihood mapping analyses for `EF.phy` are given in Figures 6.6.11 and 6.6.12. Figure 6.6.11 shows that the EF dataset is well suited for phylogenetic analysis with 98.3% fully resolved, 0.8% partly resolved, and only 0.9% unresolved quartets. A large percentage of unresolved quartets would indicate that the data are not appropriate for phylogenetic analysis.

The analysis of the branching pattern within the EF-2/G sequences (Fig. 6.6.12) shows a preference for a monophyly of Crenarchaeota and Eukaryota. A percentage of 89.9% of all admissible quartets support this monophyly strongly (lower right simplex) and 98.0% of all quartets would suggest this tree, if the maximum-likelihood values of the quartet trees are considered.

### Comparison of Different Tree Topologies (see Basic Protocol 3)

As mentioned above, the ML framework allows one to test competing hypotheses. Several tests have been proposed to compare phylogenetic trees (for a review, see Goldman et al., 2000). Three of these tests are implemented in TREE-PUZZLE.

The most commonly used is the pairwise KH test (Kishino and Hasegawa, 1989). This test is frequently used to compare the best tree, according to its ML value, to the other trees in the set.

Shimodaira and Hasegawa (1999) proposed a nonparametric test that is applicable if the maximum-likelihood tree, i.e., the tree with the highest likelihood, is among the members

of the set of proposed trees. Note that in a typical application it is not guaranteed that the maximum likelihood tree is present. Contrary to the KH test, which is essentially a pairwise test, the SH test compares all candidate trees simultaneously.

More recently, Strimmer and Rambaut (2002) published a method to infer confidence sets from possibly misspecified trees based on expected likelihood weights (ELW).

Before interpreting the results of the tests for one's own data, the authors of this unit suggest that one carefully study the relevant literature and the limitations of each method. When performing tests on trees, make sure that these tests are applicable. Goldman et al. (2000) explain which tests are valid for a given type of dataset. According to Goldman et al. (2000), KH tests should not be applied if trees were constructed on the basis of the alignment that is then, in turn, used to compare the ML tree against the second and third best tree topology. The Shimodaira-Hasegawa test (1999) is a valid test if the best tree is in the test set and the test can be applied for a collection of trees. KH is a pairwise test, and can be used for testing whether a tree is significantly worse than the best tree. The SH test is typically more conservative. It also has a tendency to depend on the number of trees in the test set, i.e., the larger the test set, the larger the confidence set. For more details about topology testing, especially for KH and SH tests and their applicability, refer to Goldman et al. (2000).

Basic Protocol 3 tests the following trees:

Tree 1: Eukaryota-Crenarchaeota sister groups for EF-2/G and EF-1$\alpha$/Tu (Fig. 6.6.13A)
Tree 2: Bacteria-Crenarchaeota sister groups for EF-2/G and EF-1$\alpha$/Tu (Fig. 6.6.13B)
Tree 3: Eukaryota-Bacteria sister groups for EF-2/G and EF-1$\alpha$/Tu (Fig. 6.6.13C).

The branching orders within the kingdoms are identical to Figure 6.6.8. The test results from the puzzle report file are given in Figure 6.6.15. All tests inferred "confidence sets" comprising trees 1 and 2. Note that tree 2, which groups together Bacteria and Crenarchaeota, got a lower likelihood, but is not significantly worse.

If all puzzling step trees occurring in Basic Protocol 1 are evaluated and tested together with the tree from Figure 6.6.8, the best tree found has a log-likelihood of −18958.52 compared to a log-likelihood of −18965.87 for the tree in Figure 6.6.8. The increase in likelihood is due to the fact that the best tree is fully resolved. This increase in the number of parameters (branches in the tree) leads to a higher likelihood. However, both statistical tests (KH and SH) indicate that the Figure 6.6.8 tree is not worse than the best tree. Incidentally, the best tree is the most frequent tree among all intermediate trees.

## COMMENTARY

### Background Information

Most of the background information needed to understand the results, as well as to interpret the data, are discussed in Guidelines for Understanding Results, above.

Programs that aim to reconstruct large phylogenetic trees have to contend with the enormous number of possible trees (Felsenstein, 1978). TREE-PUZZLE tries to cope with that problem by dividing the task into small fractions, the quartets (Strimmer and von Haeseler, 1996). For four sequences, only three informative topologies exist (Fig. 6.6.1) and the ML evaluation of each quartet is fast. Although there is still a large number of quartets to evaluate, this is often faster than computing likelihoods for a large number of large trees. From all quartet topologies, those chosen are the ones that are best supported by the data. TREE-PUZZLE takes into account that two or even all three topologies may be equally good (Strimmer et al., 1997). The set of quartet topologies is then "puzzled" together into so-called intermediate trees repeatedly with

different orders of taxa. The set of intermediate trees offers two important advantages. The frequency of bipartitions found in the intermediate trees gives a reliability measure for the internal branches in the final tree without the necessity of running a large number of initial analyses. In addition, this set of somehow biologically reasonable trees gives insight into the set of trees that is supported by the data (see Suggestions for Further Analysis).

The use of quartets also serves other purposes. The quartets are used to visualize the "tree-likeness" and subsequently the quality of the dataset for phylogenetic analysis. The number of unresolved quartets also helps to identify problematic sequences in the data sets.

Another advantage of TREE-PUZZLE is the broad variety of evolutionary models it implements. Besides DNA and binary sequence models, TREE-PUZZLE offers several general and specialized models to reconstruct phylogenies from amino acid sequences, which are supported only by a very limited number of phylogenetic software.

## Critical Parameters

### *Number of sequences and length of the alignment*

As previously mentioned, the example dataset contains 22 sequences, and thus the reliability of the reconstructed topology depends heavily on the selection of species, which is very small for the evolutionary span it covers. Several researchers (e.g., Hillis, 1996) suggest that an increased number of sequences will increase the accuracy of the reconstructed tree. Another crucial point that deserves attention is the length of the alignment. The authors' sample alignment is 915 amino acids long. If longer sequences were available, the accuracy of the reconstructed tree would increase, and the estimation of the parameters of evolution would be more precise.

### *Model selection*

All phylogenetic methods rely on assumptions about the process of DNA or amino acid substitutions. The confidence one puts into a phylogenetic analysis depends on the goodness of fit, i.e., how appropriate is the model to describe the data? In a statistical framework, the goodness of fit is typically explored applying a likelihood ratio statistics. When the models are nested, (the null model is a special case of the alternative model), the differences in the log-likelihood between both models is typically $\chi^2$-distributed (Posada and Crandall, 1998; *UNIT 6.5*). More recently, the Akaike Information Criterion and Bayesian approaches have been found to be better choices to compare and select an appropriate model of evolution (Posada and Buckley, 2004). To select the best model, a variety of programs are available, e.g., ModelTest (Posada and Crandall, 1998; Posada and Buckley, 2004; *UNIT 6.5*), which is applicable for DNA sequences. This program can be used to find the best model for a fixed tree topology. If, however, the tree topologies are different (the models are not nested), one needs to apply Monte Carlo simulations as suggested by Goldman (1993a,b).

## Suggestions for Further Analysis

As previously noted, all methods for reconstructing large phylogenetic trees, i.e., trees with more than 10 to 15 taxa, try to contend with the enormous number of possible trees (Felsenstein, 1978) by heuristic search methods (Swofford et al., 1996). Because of this, none of these methods are guaranteed to find the overall best tree. Each method has its advantages and drawbacks, which influence the result in a way that is not fully understood. Hence, "the one and only" method to reconstruct trees is not available. The authors suggest applying different methods to reconstruct trees, including maximum-likelihood, maximum-parsimony, and distance-based methods. If the methods provide the same tree topologies, then one may have some confidence in the results. If all these methods produce different tree topologies, then one should interpret the data with great care and perform further analyses to find out what is going on (Sanderson and Shaffer, 2002).

In this context, TREE-PUZZLE can be used as a generator for data-driven plausible trees. For example, one can analyze the intermediate trees, which may be different from the consensus tree, to study the distribution of different trees, thereby obtaining an indication of noise in the data. Alternatively, one may use the set of intermediate trees to apply the tests outlined in the section about comparison of trees.

In conclusion, there is no one ideal method for phylogenetic analysis. Each dataset deserves its own careful analyses guided by the results of the rich collection of tree-building methods (Swofford et al., 1996). Finally, it is good to remember that, sometimes, a tree is simply not the best way to visualize the data.

## Literature Cited


Adachi, J. and Hasegawa, M. 1996. Model of amino acid substitution in proteins encoded by mitochondrial DNA. *J. Mol. Evol.* 42:459-468.

Dayhoff, M.O., Schwartz, R.M., and Orcutt, B.C. 1978. A model of evolutionary change in proteins. *In* Atlas of Protein Sequence Structure, Vol. 5 (M.O. Dayhoff, ed.) pp. 345-352. National Biomedical Research Foundation, Washington, D.C.

Edwards, A.W.F. and Cavalli-Sforza, L.L. 1964. Reconstruction of evolutionary trees. *In* Phenetic and Phylogenetic Classification (V.H. Heywood and J. McNeill, eds.) pp. 67-76. Systematics Association, London.

Felsenstein, J. 1978. The number of evolutionary trees. *Syst. Zool.* 27:27-33.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.* 17:368-376.

Felsenstein, J. 1984. Distance methods for inferring phylogenies: A justification. *Evolution* 38:16-24.

Felsenstein, J. 2004. Inferring Phylogenies. Sinauer Associates, Sunderland, Mass.

Goldman, N. 1993a. Statistical tests of models of DNA substitution. *J. Mol. Evol.* 36:182-198.

Goldman, N. 1993b. Simple diagnostic statistical tests of models for DNA substitution. *J. Mol. Evol.* 37:650-661.

Goldman, N., Anderson, J.P., and Rodrigo, A.G. 2000. Likelihood-based tests of topologies in phylogenetics. *Syst. Biol.* 49:652-670.

Gu, X., Fu, Y.-X., and Li, W.-H. 1995. Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. *Mol. Biol. Evol.* 12:546-557.

Hasegawa, M., Kishino, H., and Yano, T. 1985. Dating the human-ape split by a molecular clock of mitochondrial DNA. *J. Mol. Evol.* 22:160-174.

Henikoff, S. and Henikoff, J.G. 1992. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.* 89:10915-10919.

Hillis, D.M. 1996. Inferring complex phylogenies. *Nature* 383:130-131.

Iwabe, N., Kuma, K.-I., Hasegawa, M., Osawa, S., and Miyata, T. 1989. Evolutionary relationship of Archaebacteria, Eubacteria, and Eukaryotes inferred from phylogenetic trees of duplicated genes. *Proc. Natl. Acad. Sci. U.S.A.* 86:9355-9359.

Jones, D.T., Taylor, W.R., and Thornton, J.M. 1992. The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.* 8:275-282.

Jukes, T.H. and Cantor, C.R. 1969. Evolution of protein molecules. *In* Mammalian Protein Metabolism (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.

Kishino, H. and Hasegawa, M. 1989. Evolution of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. *J. Mol. Evol.* 29:170-179.

Lanave, C., Preparata, G., Saccone, C., and Serio, G. 1984. A new method for calculating evolutionary substitution rates. *J. Mol. Evol.* 20:86-93.

McMorris, F.R. and Neumann, D.A. 1983. Consensus functions defined on trees. *Math. Soc. Sci.* 4:131-136.

Müller, T. and Vingron, M. 2000. Modeling amino acid replacement. *J. Comput. Biol.* 7:761-776.

Page, R.D. and Holmes, E.C. 1998. Molecular Evolution: A Phylogenetic Approach. Blackwell Science, Oxford.

Posada, D. and Crandall, K.A. 1998. MODELTEST: Testing the model of DNA substitution. *Bioinformatics* 14:817-818.

Posada, D. and Buckley, T. 2004. Model selection and model averaging in phylogenetics: Advantages of akaike information criterion and Bayesian approaches over likelihood ratio tests. *Syst. Biol.* 53:793-808.

Sanderson, M.J. and Shaffer, H.B. 2002. Troubleshooting molecular phylogenetic analyses. *Annu. Rev. Ecol. Syst.* 33:49-72.

Schmidt, H.A., Strimmer, K., Vingron, M., and von Haeseler, A. 2002. TREE-PUZZLE: Maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics* 18:502-504.

Schöniger, M. and von Haeseler, A. 1994. A stochastic model for the evolution of autocorrelated DNA sequences. *Mol. Phyl. Evol.* 3:240-247.

Shimodaira, H. and Hasegawa, M. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Mol. Biol. Evol.* 16:1114-1116.

Strimmer, K. and von Haeseler, A. 1996. Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.* 13:964-969.

Strimmer, K. and von Haeseler, A. 1997. Likelihood mapping: A simple method to visualize phylogenetic content of a sequence alignment. *Proc. Natl. Acad. Sci. U.S.A.* 94:6815-6819.

Strimmer, K. and Rambaut, A. 2002. Inferring confidence sets of possibly misspecified gene trees. *Proc. R. Soc. Lond. B* 269:137-142.

Strimmer, K., Goldman, N., and von Haeseler, A. 1997. Bayesian probabilities and quartet puzzling. *Mol. Biol. Evol.* 14:210-213.

Swofford, D.L., Olsen, G.J., Waddell, P.J., and Hillis, D.M. 1996. Phylogeny reconstruction. *In* Molecular Systematics, 2nd ed. (D.M. Hillis, C. Moritz, and B.K. Mable, eds.) pp. 407-514. Sinauer Associates, Sunderland, Mass.

Tamura, K. and Nei, M. 1993. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol. Biol. Evol.* 10:512-526.

Tavare, S. 1986. Some probabilistic and statistical problems on the analysis of DNA sequences. *Lec. Math. Life Sci.* 17:57-86.

Whelan, S. and Goldman, N. 2001. A general empirical model of protein evolution derived from multiple protein families using a maximum likelihood approach. *Mol. Biol. Evol.* 18:691-699.

## Key References

Felsenstein, 2004. See above.
*A comprehensive textbook covering almost all areas of phylogenetic inference.*

Goldman et al., 2000. See above.
*A comprehensive review discussing tests for tree topologies and their applicability.*

Page and Holmes, 1998. See above.
*A well written textbook about phylogenetics and its applications.*

Sanderson and Shaffer, 2002. See above.
*A good review on problems in phylogeny reconstruction.*

Strimmer and von Haeseler, 1996. See above.
*An original publication of the Quartet Puzzling method.*

Strimmer and von Haeseler, 1997. See above.
*A more detailed description of Likelihood Mapping.*

Swofford et al., 1996. See above.
*An excellent introduction to the rich collection of phylogenetic methods.*

## Internet Resources

http://www.tree-puzzle.de
*TREE-PUZZLE Web site.*

http://rdp8.cme.msu.edu/download/programs/TreeTool/
*TreeTool Web site (tree-drawing program).*

http://taxonomy.zoology.gla.ac.uk/rod/treeview.html
*TreeView Web site (tree-drawing program, see UNIT 6.2).*

http://evolution.genetics.washington.edu/phylip/software.html
*Joe Felsenstein's list of tree-reconstruction and -drawing programs.*

http://www.ghostscript.com/
*GhostScript Web page (PostScript viewer and converter).*

---

Contributed by Heiko A. Schmidt and Arndt von Haeseler
Center for Integrative Bioinformatics Vienna (CIBIV), Max F. Perutz Laboratories (MFPL), Vienna, Austria
University of Vienna, Austria
Medical University Vienna, Austria
University of Veterinary Medicine, Vienna, Austria

# What If I Don't Have a Tree?: Split Decomposition and Related Models

Evolutionary relationships between taxa are most often represented as phylogenetic trees, and this is, of course, justified by the assumption that evolution is a branching or tree-like process. However, a set of real data often contains a number of different and sometimes conflicting signals, and thus does not always clearly support a unique tree.

To address this issue, Andreas Dress and Hans-Jürgen Bandelt developed the theory of split decomposition that uses a type of phylogenetic network called a *splits graph* to represent evolutionary relationships (Bandelt and Dress, 1992a,b). For ideal data, this graph is a tree, whereas less-ideal data will give rise to a tree-like network that can be interpreted as possible evidence for different and conflicting phylogenies.

The Basic Protocol describes how to use the software program SplitsTree interactively to compute splits graphs from sequence or distance data, and discusses many of the commonly used features of the program. An Alternate Protocol describes how to use the command-line version of the program—e.g., within a pipeline. The Support Protocol describes how to obtain the program. In Guidelines for Understanding Results, discussion is presented on how to interpret the results obtained by the program, and in Background Information there is a brief description of the underlying mathematical theory.

The current implementation of SplitsTree (version 3.2) is written in C++ and runs under Windows, Linux, and Solaris, requiring version 8.0.5 of Tcl-Tk. There also exists an older version of the program (version 2.4) that runs under old MacOS versions. At present, the program is being reimplemented in Java. The new version 4.0, to be released in 2003, will run on any platform that supports Java and will contain new methods that are not described in this unit. SplitsTree is freely available from *http://www-ab.informatik.uni-tuebingen.de/software/splits/welcome_en.html*, where links to Web servers that run the program can also be found.

## USING SplitsTree INTERACTIVELY

***BASIC PROTOCOL***

There are three different SplitsTree binaries that provide a graphical user interface: one older version (2.4) runs under older MacOS systems, and a more recent version (3.2) runs under Windows, Linux, and Solaris. The following protocol is based on the more recent version. A description of version 2.4 can be found in Huson (1998).

### *Necessary Resources*

*Hardware*

A computer running Windows, Linux, or Solaris

*Software*

The SplitsTree program, downloadable from the SplitsTree Web site:
*http://www-ab.informatik.uni-tuebingen.de/software/splits/welcome_en.html*

*Files*

SplitsTree uses the NEXUS file format for input and output of data. Unfortunately, there are two variants of this format in use, an old version and the "official" format defined in Maddison et al. (1997). A number of existing programs use the old format or can parse both formats. SplitsTree, however, only supports the "official" format.

**Contributed by Daniel H. Huson**
*Current Protocols in Bioinformatics* (2003) 6.7.1-6.7.13

For identification purposes, the first line of a file in NEXUS format must consist of the statement `#NEXUS`. For a file containing input data, this is followed by a `taxa` block that lists the names of all occurring taxa. The `taxa` block is then followed by a `characters` block containing a set of aligned character sequences for the named taxa, or by a `distances` block, if the input consists of a distance matrix.

It is usually very easy to convert a file containing aligned sequences or distances into the NEXUS format because both the `characters` and the `distances` block contain a `format` command that one can use to describe the precise format in which the data is given. For example, in the `characters` block, one can specify whether the sequences are interleaved, not interleaved, or transposed. Moreover, one can specify whether the sequences are DNA, RNA, or protein and what the gap, missing, and match characters are. For `distances`, one can specify, e.g., whether the upper, lower, or both triangles of the matrix are given, with or without the diagonal. An example of an input file in NEXUS format is given in Figure 6.7.1. Note that the program does not distinguish between upper- and lowercase letters. Moreover, a taxon label must be a single word without spaces or special characters (see Maddison et al., 1997, for details), unless it is surrounded by single quotes, e.g., `'Example One'`.

A formal description of the input format used by SplitsTree is given in Figure 6.7.2.

In practice, the most tedious and error-prone step in formatting input data in NEXUS format is producing a list of all taxa names to place in the `taxa` block. To avoid this, a special label, `_detect_`, can be used in the `taxlabels` command of the taxa block instead of a list of taxon names. In this case, SplitsTree will read the labels from the supplied `distances` or `characters` block.

The sample data used below are shown in Figure 6.7.1 and can be downloaded from the Current Protocols Web site (*http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm*).

***Activate program and open file***

1. Download SplitsTree (see Support Protocol). Start the SplitsTree program by double-clicking on the SplitsTree icon (under Windows) or typing `./xsplits` in the `splitstree3.2` directory (under Linux or Solaris).

   *The first window to appear will be the main window in which the splits graph will be drawn and in which one can interact with the graph, e.g., to modify its layout. Messages generated by the program are written to a second log window.*

2. Select the Open item in the Files menu file (Fig. 6.7.3A) and then use the file selector dialog to open a file.

   *By convention, the name of a NEXUS file should end on* `.nex` *or* `.nxs`*, but this is not enforced by the program.*

   *The program will read in the data and will then immediately apply the split decomposition method to compute a splits graph for the given input data, first using the Hamming distance transformation (based simply on the "observed" distances) to obtain a distance matrix, if aligned sequence data are given. The computed graph is then displayed in the main window (Fig. 6.7.4).*

   *Messages generated by the program during these computations are printed to the log window. If errors are encountered in the input file, then no computation will be performed and an error message will be written to the log window. Perhaps the most common error is that the NEXUS file was produced by a program that does not adhere to the format published in Maddison et al. (1997). In this case, only a few changes are necessary to modify the input so that it can parsed by SplitsTree (Fig. 6.7.1).*

```
#NEXUS [The first token in the file must be #NEXUS]
[! Comments come in square brackets. A comment starting with "!" is
echoed to the user when read]
BEGIN taxa;
     DIMENSIONS ntax=8;
     TAXLABELS Lemur_catta Homo_sapiens Pan Gorilla Pongo M.mulatta
     M.fascicularis Saimiri_sciureus
;

BEGIN characters;
     DIMENSIONS nchar=898;
     FORMAT datatype=DNA gap=- labels interleave;

     MATRIX
     Lemur_catta         AAGCTTCATAGGAGCAACCATTCTAATAATCGCAC
     Homo_sapiens        AAGCTTCACCGGCGCAGTCATTCTCATAATCGCCC
     Pan                 AAGCTTCACCGGCGCAATTATCCTCATAATCGCCC
     Gorilla             AAGCTTCACCGGCGCAGTTGTTCTTATAATTGCCC
     Pongo               AAGCTTCACCGGCGCAACCACCCTCATGATTGCCC
     M.mulatta           AAGCTTTTCTGGCGCAACCATCCTCATGATTGCTC
     M.fascicularis      AAGCTTCTCCGGCGCAACCACCCTTATAATCGCCC
     Saimiri_sciureus    AAGCTTCACCGGCGCAATGATCCTAATAATCGCTC

          [Middle section removed]

     Lemur_catta         ACCCTAAAACTATCTATTAGCTT
     Homo_sapiens        ACAACCCAGCTCTCCCTAAGCTT
     Pan                 ACAACCCAGCTCTCCCTAAGCTT
     Gorilla             ACAATTCAACTCTCCCTAAGCTT
     Pongo               ACACTACAACTCTCACTAAGCTT
     M._mulatta          ACACTAGACCTAACACTAAGCTT
     M._fascicularis     ACATTAGACCTAACACTAAGCTT
     Saimiri_sciureus    ACCATCAAACTATCCCTAAGCTT
     ;
END;

BEGIN distances;
     DIMENSIONS ntax=8;
     FORMAT triangle=LOWER diagonal labels;
     MATRIX
     'Lemur_catta'       0
     'Homo_sapiens'      0.3062 0
     'Pan'               0.307 0.08908 0
     'Gorilla'           0.2917 0.1035 0.1057 0
     'Pongo'             0.2906 0.161 0.1714 0.1670 0
     'M._mulatta'        0.2884 0.2327 0.2505 0.2349 0.2461 0
     'M._fascicularis'   0.2984 0.248 0.2672 0.2616 0.2605 0.09242 0
     'Saimiri_sciureus'  0.2850 0.2739 0.2850 0.2706 0.2861 0.2939 0.2873 0
     ;
END;
```

**Figure 6.7.1** Example of an input file in NEXUS format. Usually, an input file will contain either a `characters` or `distances` block, but not both.

```
#NEXUS

BEGIN TAXA;
     DIMENSIONS NTAX=number-of-taxa;
     TAXLABELS taxon_1 taxon_2 ... taxon_ntax;
END;

BEGIN CHARACTERS;
     DIMENSIONS [NTAX=number-of-taxa]      NCHAR=number-of-
characters;
     [FORMAT
          [DATATYPE={STANDARD|DNA|RNA|PROTEIN}]
          [GAP=symbol]
          [MATCHCHAR=symbol]
          [MISSING=symbol]
          [RESPECTCASE]
          [SYMBOLS="symbol symbol ..."]
          [[NO] INTERLEAVE]
          [[NO] LABELS]
          [[NO] TOKENS]
          [[NO] TRANSPOSE]
     ;]
     [CHARWEIGHTS wgt_1 wgt_2 ... wgt_nchar;]
     MATRIX
          sequence data in specified format
     ;
END;

BEGIN DISTANCES;
     [DIMENSIONS NTAX=number-of-taxa;]
     [FORMAT
          [TRIANGLE={LOWER|UPPER|BOTH}]
          [[NO] DIAGONAL]
          [[NO] LABELS]
          [MISSING=symbol]
     ;]
     MATRIX
          distance data in specified format
     ;
END;
```

**Figure 6.7.2** A depiction of the input format used by SplitsTree. Here, square brackets (e.g., `[NO]`) indicate optional statements and vertical lines (e.g., `{LOWER|UPPER|BOTH}`) indicate alternative choices. Note that the `FORMAT` is not case-sensitive, unless the `RESPECTCASE` label is present in the `FORMAT` command of the `CHARACTERS` block, in which case the program will distinguish between upper- and lowercase characters in the input matrix.

### *Set parameters for analysis*

3. If the input data consist of a set of aligned character sequences, then one can select between a number of different distance transformations (*UNIT 6.4*) that are listed in the Options menu (Fig. 6.7.3D). After any change of parameters, all dependent data and the resulting graph are immediately updated. Hence, the displayed graph always reflects the current choice of method and options.

   *For example, selecting Hamming, Kimura-3ST, Jukes-Cantor, or LogDet requests that the program compute the distance matrix and splits graph using either the Hamming distances (the default), the Kimura-3ST transformation (Kimura, 1981), the-Jukes Cantor correction (Jukes and Cantor, 1969), or the LogDet transformation (Steel, 1994). Note that the Kimura-3ST and Jukes-Cantor transformations can only be applied to nucleotide data, whereas the Hamming and LogDet transformations apply to protein sequences as well.*

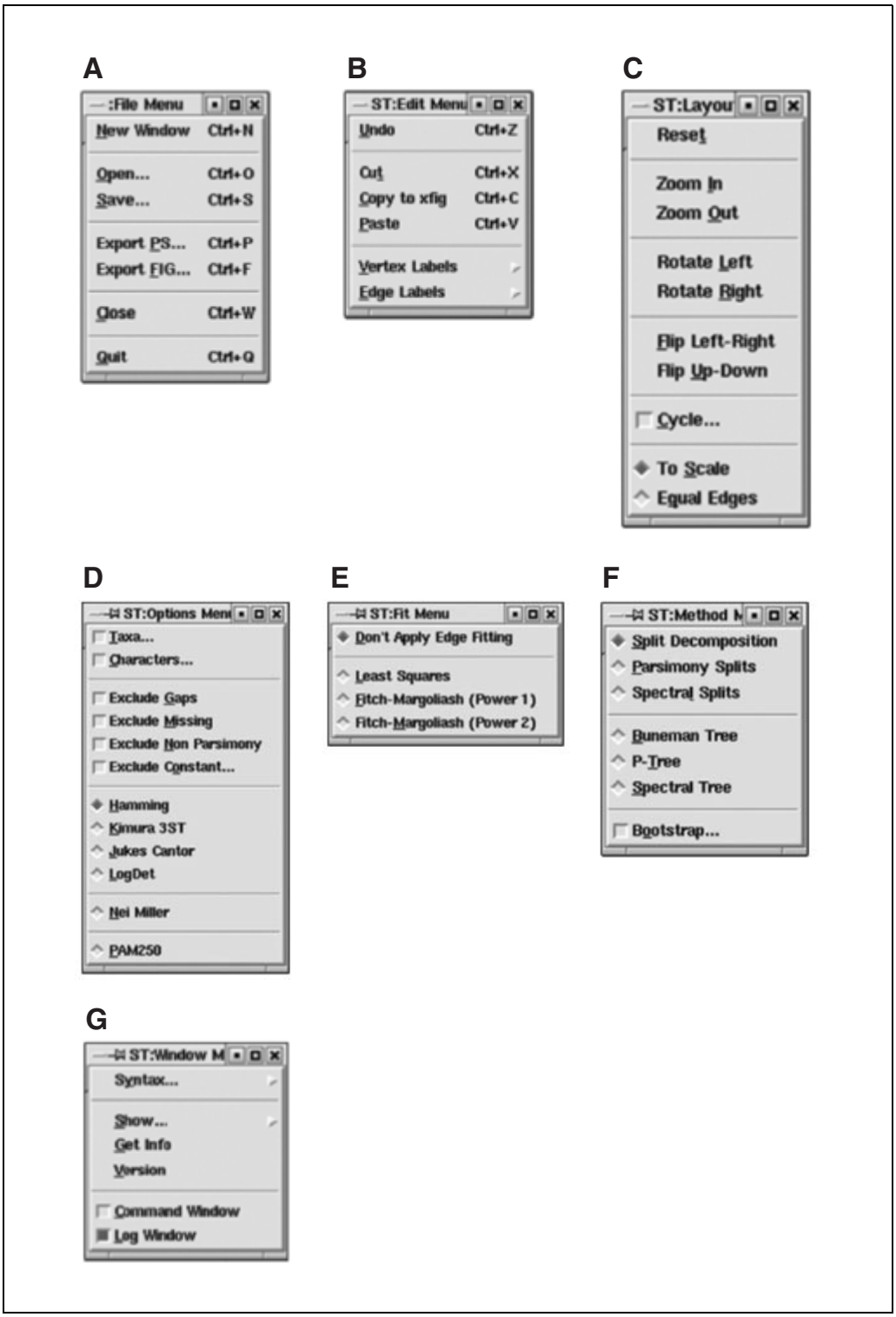


**Figure 6.7.3** The menus provided by SplitsTree.

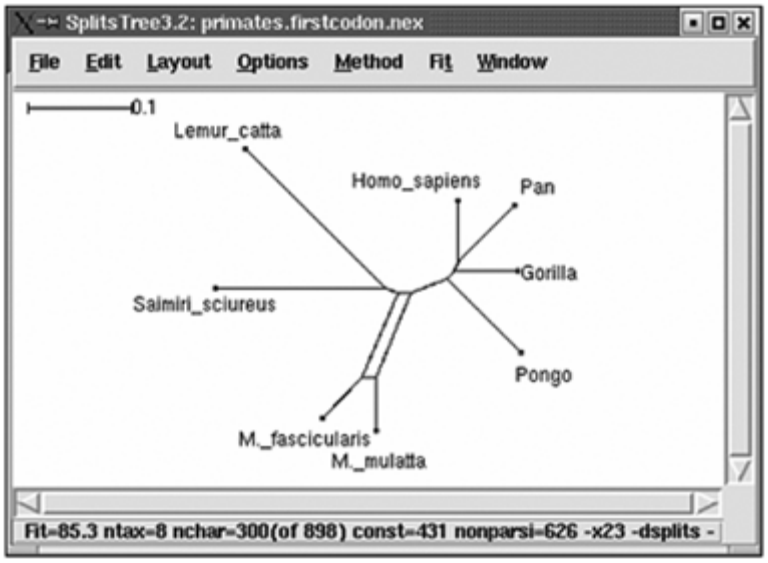


**Figure 6.7.4** Splits graphs are displayed in the main window. In this graph, a strong split groups *M. fascicularis* and *M. mulatta* together, whereas a second split, incompatible with the first, groups *M. fascicularis*, *Lemur catta*, and *Samimiri sciureus* together, against the rest. Notice the Fit value of 85.3 in the lower lefthand corner (see Guidelines for Understanding Results). This graph was obtained from the data in Figure 6.7.1, using split decomposition applied to the first position of each codon.

4. The Methods menu (Fig. 6.7.3F) is used to determine which algorithm is employed to compute the splits graph. By default, the Split Decomposition method is selected. Other implemented methods are Parsimony Splits, Spectral Splits, Buneman Tree, P-Tree, and S-Tree.

   *The Split Decomposition and Buneman Tree methods compute a network or tree, respectively, from distances (Bandelt and Dress, 1992a; see UNITS 6.3 & 6.4 for distance calculation). All other methods compute a graph directly from sequences. Parsimony Splits and P-Tree operate by considering the most parsimonious trees on each quartet and fitting these together to obtain a splits graph or tree, respectively (Bandelt and Dress, 1993). The Spectral Splits and Spectral Tree methods operate by performing spectral analysis (Hendy and Penny, 1993) on the given DNA sequences and then greedily selecting a set of splits as to obtain a splits graph or tree, respectively, using a "greedy" heuristic that attempts to maximize the total weight of the selected splits.*

5. Split decomposition and related methods work best on small sets of closely related sequences, and thus one will often want to analyze different subsets of the original data set. This is accomplished by selecting the Taxa item in the Options menu to obtain a dialog box that one can use to include and exclude specific taxa (Fig. 6.7.5).

   *As described below, one can often obtain a more resolved graph by removing certain problematic taxa from the analysis.*

6. Given a set of aligned character sequences as input, often one would like to restrict the analysis to a subset of positions. SplitsTree offers a number of mechanisms for doing this. In the Options menu (Fig. 6.7.3D), the Exclude Gaps, Exclude Missing, Exclude Non Parsimony, and Exclude Constant... items should be selected to exclude all positions from consideration that contain gaps or missing characters, that are non-parsimony-informative, or that are constant, respectively. Additionally, selecting

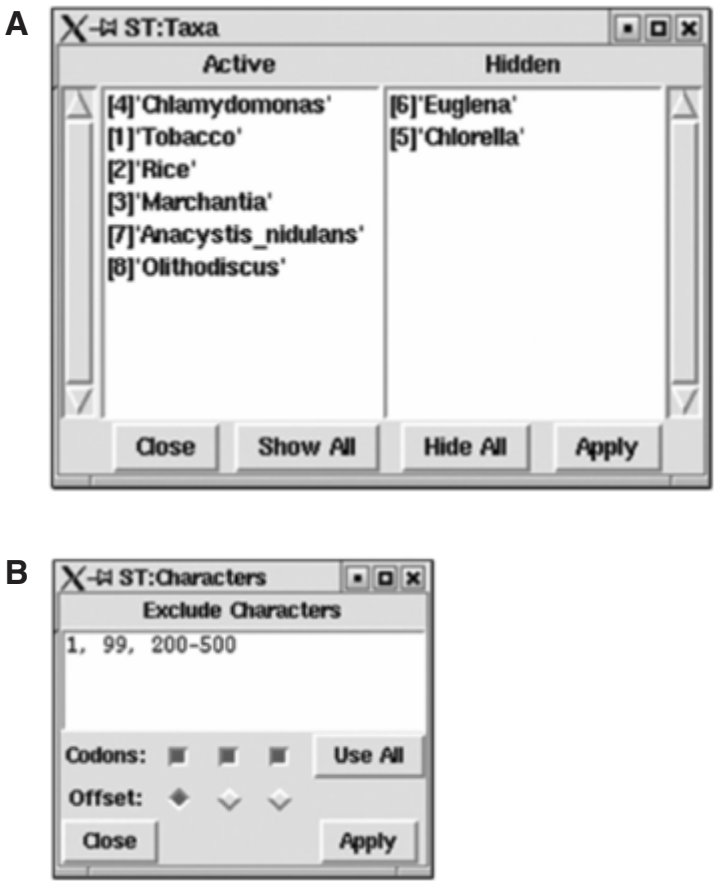


**Figure 6.7.5** (**A**) Taxa listed on the left are "active" and included in all computations. All taxa listed on the right are "hidden." Clicking on a list item will move it to the other list. (**B**) Specific sites in aligned character sequences can be excluded from consideration; for example, here the sites 1, 99, and 200-500 will be ignored. Additionally, one can specify which codon positions (with which offset) are to be considered.

the Characters... item from the Options menu will open a window that one can use to specifically exclude single positions or ranges of positions (by typing them into the input box in the format `1, 99, 200-500`, to exclude positions 1, 99, and 200 to 500), or to select specific codons (Fig. 6.7.5B).

*A position is called constant if the character at the given position is the same for all sequences, and it is called non-parsimony-informative if it is constant except for precisely one sequence. For constant sites, one can distinguish abstractly between sites that are free to vary, but simply have not, and sites that are constant due to functional constraints. For the LogDet transformation to work best, one must exclude the number of constant sites of the latter type (Lockhart et al., 1994) using the Exclude Constant dialog box.*

***Set layout and save results***

7. The layout of the splits graph is modified either by using the Layout menu (Fig. 6.7.3C), by using the arrow keys, or by clicking the mouse in the main window. Use the items in the Layout menu to zoom, rotate, or flip the splits graph. Moreover, the To Scale and Equal Edges items toggle between a to-scale representation and a representation in which all edges are drawn at the same length. The up and down arrow keys zoom in and out of the splits graph, whereas the left- and right arrow keys

rotate left and right. Clicking and dragging a taxon label will reposition it. Clicking on a edge of the graph will select that edge and all other edges representing the same split. Clicking and dragging one end of a selected edge will rotate the selected edges without changing any edge lengths.

8. The computed graph is saved in NEXUS format using the Save item in the File menu. As the complete state is saved, no recomputation will take place upon reopening of the file. A picture of the graph is saved in PostScript format using the Export PS... item.

   *The Linux and Solaris versions additionally provide an Export FIG... item (Fig. 6.7.3A) that produces a FIG file; these versions also provide a Copy to xfig item in the Edit menu (Fig. 6.7.3B) that copies the graph into the xfig copy buffer. Unfortunately, the current Windows version does not support copy and paste between SplitsTree and other Windows programs. This will change with version 4.0. As a work-around, save a picture of the graph as a PostScript file and then use a program such as PSTOEDIT to convert the PostScript file to a WMF file that can then be imported into a word processing or graphics program in such a way that all edges and labels of the graph appear as individual objects, each of which can be modified separately.*

9. The Window menu has items for displaying the syntax or contents of any of the NEXUS blocks (Fig. 6.7.3G). For example, selecting the Show... item from that menu and then the Taxa item in the submenu that appears will print the current taxa block to the log window, whereas selecting the Syntax... item and then Characters from the submenu that appears will produce a description of the syntax of the characters block in the log window.

   *Additionally, under the Window menu, the Get Info item will list some general information about the current data, and the Version command will report the version of the program. The Command Window and Log Window items are used to show or hide the command and log windows. The command window is used to enter commands in the same way as for the command-line version of the program (see Alternate Protocol).*

***ALTERNATE PROTOCOL***

## USING THE COMMAND-LINE VERSION OF SplitsTree

This protocol describes how to use the command-line version of SplitsTree that is available for Linux and Solaris. The main application here is to use the program in a scripted pipeline of programs. In the command-line version of the program, commands are typed at the SplitsTree prompt or piped to the program. Entries are assumed to be single lines, unless a backslash is entered, in which case the program goes into multiline input model until a second backslash is entered.

### *Necessary Resources*

*Hardware*

A computer running Windows, Linux, or Solaris

*Software*

The command-line version of the SplitsTree program, downloadable from the SplitsTree Web site: *http://www-ab.informatik.uni-tuebingen.de/software/splits/welcome_en.html*

*Files*

SplitsTree uses the NEXUS file format for input and output of data. Unfortunately, there are two variants of this format in use, an old version and the "official" format defined in Maddison et al. (1997). A number of existing programs use the old format or can parse both formats. SplitsTree, however, only supports the "official" format.

For identification purposes, the first line of a file in NEXUS format must consist of the statement `#NEXUS`. For a file containing input data, this is followed by a `taxa` block that lists the names of all occurring taxa. The `taxa` block is then followed by a `characters` block containing a set of aligned character sequences for the named taxa, or by a `distances` block, if the input consists of a distance matrix.

It is usually very easy to convert a file containing aligned sequences or distances into the NEXUS format because both the `characters` and the `distances` block contain a `format` command that one can use to describe the precise format in which the data is given. For example, in the `characters` block, one can specify whether the sequences are interleaved, not interleaved, or transposed. Moreover, one can specify whether the sequences are DNA, RNA, or protein and what the gap, missing, and match characters are. For `distances`, one can specify, e.g., whether the upper, lower, or both triangles of the matrix are given, with or without the diagonal. An example of an input file in NEXUS format is given in Figure 6.7.1. Note that the program does not distinguish between upper- and lowercase letters. Moreover, a taxon label must be a single word without spaces or special characters (see Maddison et al., 1997, for details), unless it is surrounded by single quotes, e.g., `'Example One'`.

A formal description of the input format used by SplitsTree is given in Figure 6.7.2.

In practice, the most tedious and error-prone step in formatting input data in NEXUS format is producing a list of all taxa names to place in the `taxa` block. To avoid this, a special label, `_detect_`, can be used in the `taxlabels` command of the taxa block instead of a list of taxon names. In this case, SplitsTree will read the labels from the supplied `distances` or `characters` block.

The sample data used below are shown in Figure 6.7.1 and can be downloaded from the Current Protocols Web site (*http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm*).

1. Download the command-line version of SplitsTree (Support Protocol). Launch the command-line version of SplitsTree by typing `./splits` in the `splitstree3.2` directory.

2. To open a file, type `open [filename]` at the prompt, where `[filename]` is the name of the input file. If the file name contains any spaces, surround the whole file name string with single quotes. The program will immediately compute a splits graph, as in the Basic Protocol. After any change of parameters, all dependent data and the resulting graph are immediately updated. Hence, the displayed graph always reflects the current choice of method and options.

   *Entered input lines are edited using the arrow keys; the left and right arrows are used to move the cursor within the current input line, whereas the up and down arrows are used to navigate the input history.*

3. To obtain a picture of the graph, type `ps [filename]` at the command line to write a PostScript picture to the named file, or enter `draw` to open Ghostview with a picture of the graph.

   *By default, the* `draw` *command will attempt to execute the command* `ghostview tmp.ps`. *However, if the environment variable* `SPLITSVIEWER` *is set to a different PostScript display program, then it will be used instead.*

4. Specify which distance transformation (*UNIT 6.4*) is used by typing `assume transform=[name]`, where `[name]` is one of the following: `hamming`, `k3st`, `jc`, or `logdet`.

5. Specify which method is used for computing the splits graph by typing `assume method=[name]`, where `[name]` is one of the following: `dsplits`, `psplits`, `spectral`, `dtree`, `ptree`, or `stree`, indicating split decomposition, parsimony splits, spectral splits, Buneman tree, p-tree, and s-tree, respectively.
6. To exclude taxa from the analysis, type `assume extaxa = [which]`, where `[which]` is a list of space-separated taxon labels.
7. There are a number of additional assumptions that are set interactively. Type `help assume` to obtain a complete description of how to use the `assume` command.
8. The computed data are saved to a file in NEXUS format by entering `save [file-name]`.

   *Use* `save [blockname] [filename]` *to save an individual NEXUS block, e.g.,* `save characters out.nex` *will save the* `characters` *block to the file* `out.nex`.
9. Entering `help` produces an overview of all commands known to the program and `help [name]` will give specific help for the named command or NEXUS block.

***SUPPORT PROTOCOL***

## OBTAINING SplitsTree

The SplitsTree program, example files, and brief documentation are available from the SplitsTree Web site at *http://www-ab.informatik.uni-tuebingen.de/software/splits/welcome_en.html*, which also contains links to online servers running SplitsTree.

The Windows version of SplitsTree is installed by downloading and unzipping the file `Splits32.zip` and installing the version of Tcl-Tk specified on the SplitsTree Web site.

The command-line and GUI executables targeting Linux are installed by downloading the file `splitstree32.linux.tgz` and extracting the contents using the command

`tar xzf splitstree32.linux.tgz`.

The command-line and GUI executables targeting Solaris are installed by downloading the file `splitstree32.solaris.tgz` and extracting the contents using the command `tar xzf splitstree32.solaris.tgz`.

All source files are contained in the file `splitstree32.src.tgz`.

Both the Linux and Solaris GUI versions require that the correct version of Tcl-Tk be installed, as specified on the SplitsTree Web site.

## GUIDELINES FOR UNDERSTANDING RESULTS

The standard model of evolution states that species evolve in a tree-like pattern produced by successive speciation events. Hence, given a matrix of evolutionary distances, the goal of phylogenetic analysis is usually to construct a phylogenetic tree that represents the given distances as well as possible.

Any matrix of evolutionary distances $d$ that fulfills the mathematical condition of *additivity* possesses an exact representation by a phylogenetic tree $T$, in the sense that for any two taxa $i$ and $j$, the distance $d_{ij}$ equals the distance between $i$ and $j$ obtained as the sum of lengths of edges along the path in $T$ that joins the two nodes labeled $i$ and $j$, and vice versa.

Evolutionary distance matrices rarely fulfill the condition of additivity, so any computed phylogenetic tree will provide only an approximation of the given distances. Split-decomposition-type methods use a more general type of graph, a so-called splits graph, to represent phylogenetic distances, and are therefore often able to provide a better approximation of the evolutionary distances, while usually producing a graph that is still very tree-like. Indeed, if the matrix is additive (i.e., a tree metric), then split decomposition will produce the exact tree representation of the matrix, while less ideal matrices will give rise to graphs that are more network-like.

Like a tree, such a splits graph represents the distance $d_{ij}$ between any two taxa $i$ and $j$ by the length of a shortest path of edges that connects the node labeled $i$ to the node labeled $j$.

The method of split decomposition has a very nice mathematical foundation, which was worked out by Bandelt and Dress (1992a). In particular they proved that the distance between two taxa given in a distance matrix is always greater than or equal to the distance represented in the splits graph. Based on this, the *fit* of a splits graph is defined as the sum of all graph distances divided by the sum of all distances given in the matrix, as a percentage.

A splits graph with a high fit value, e.g., 80% or higher, provides a very good representation of the given evolutionary distance matrix, whereas a low fit indicates that the computed graph is less useful. Evolutionary data often have a good fit value.

SplitsTree should be viewed as a data exploration tool that is used to determine how "tree-like" a given data set is. If the split-decomposition method produces a fairly resolved tree with a high fit value, then one can place confidence in the resulting tree, as split decomposition is a very conservative method and only produces splits that are supported by all taxa present in the data set. On the other hand, if split decomposition produces a network with many pairs of incompatible splits, then this is an indication that the data set contains conflicting signals. In this case further analysis is necessary to determine the exact nature of the conflicts. The development of algorithmic methods to address this problem is an area of active research. Finally, if the resulting splits graph has a very low fit value, then, again, the data are probably very far from tree-like. However, in this case, the actual graph is of little use as it provides a very poor representation of the distance matrix.

## COMMENTARY

### Background Information

A *phylogenetic tree*, $T$, is a tree whose leaves (and possibly some internal nodes, too) are labeled by a set, $X$, of taxa and whose remaining nodes are of degree at least 3 (in the unrooted case). Any edge $e$ of $T$ defines a *split*, $S = \{A, A'\}$ of $X$, that is a partitioning of $X$ into two nonempty sets $A$ and $A'$, consisting of all taxa on the one side, or the other, of $e$ in $T$. Such a system of splits, $\Sigma$, is called *compatible*, if for any two splits, $S_1 = \{A_1, A'_1\}$ and $S_2 = \{A_2, A'_2\}$ in $\Sigma$, one of the four intersections

$$A_1 \cap A_2, \quad A_1 \cap A'_2, \quad A'_1 \cap A_2, \quad \text{or} \quad A'_1 \cap A'_2$$

is empty. Any phylogenetic tree $T$ gives rise to a compatible split system $\Sigma$. Vice versa, any compatible split system $\Sigma$ corresponds to a unique phylogenetic tree $T$ (Buneman, 1971). Therefore, tree reconstruction for a given set of taxa $X$ is equivalent to computing a compatible system of splits $\Sigma$ for $X$ and then determining a weight for each split $S$, equal to the length of the associated edge.

Hence, to obtain more general graphs, one must consider less restricted systems of splits. Let $X$ be a set of taxa. A system of splits $\Sigma$ of $X$ is called *weakly compatible*, if for any three splits $S_1 = \{A_1, A'_1\}$, $S_2 = \{A_2, A'_2\}$, $S_3 = \{A_3, A'_3\}$, at least one of the four intersections

$$A_1 \cap A_2 \cap A_3, \quad A_1 \cap A'_2 \cap A'_3, \quad A'_1 \cap A_2 \cap A'_3$$

$$\text{or} \quad A'_1 \cap A'_2 \cap A_3$$

and at least one of the four intersections

$A_1' \cap A_2' \cap A_3'$, $A_1' \cap A_2 \cap A_3$, $A_1 \cap A_2' \cap A_3$,

or $A_1 \cap A_2 \cap A_3'$

is empty (Bandelt and Dress, 1992a). So, in particular, any two splits are permitted to be incompatible. Intermediately, $\Sigma$ is called *circular*, if there exists an ordering $x_1, x_2, ..., x_m$ of the taxa such that for every split $S \in \Sigma$, there exists $A \in S$ with $A = \{x_{p(S)}, x_{p(S)+1}, ..., x_{q(S)}\}$ and $1 < p(S) = q(S) = m$. It has been proven that a compatible split system is always circular and a circular split system is always weakly compatible. Circular split systems are of particular interest because they can be represented by planar split graphs, whereas this is not true in general for noncircular split systems. In biological applications, the arising split systems are often either circular or only mildly noncircular.

A splits graph representing a weakly compatible split system $\Sigma$ is a graph $G(S) = (V,E)$ whose nodes $v \in V$ are labeled by the set of taxa $X$ and whose edges $e \in E$ are straight line segments that represent the splits in $\Sigma$ (see Fig. 6.7.4). More precisely, each split $S = \{A,A'\} \in \Sigma$ is represented by a band of parallel edges of equal length in such a way that deleting all edges in such a band partitions the graph into precisely two components, one containing all nodes labeled by taxa in $A$ and the other containing all nodes labeled by taxa in $A'$. The length of the edges representing a given split $S$ indicates its weight or support and is calculated as the so-called *isolation index* of $S$. For an algorithm that computes splits graphs, see Dress et al. (1996).

Split decomposition is a method for obtaining a system of weakly compatible splits with weights from a given set of evolutionary data. Assume a given set of taxa, $X$, and a distance map $d: X \times X \rightarrow \mathbb{R}^{\geq 0}$ on $X$, i.e., a matrix representing evolutionary distances between pairs of taxa. Bandelt and Dress (1992a) showed that such a distance map $d$ has the following unique decomposition:

$$d = \sum_S \alpha_s \delta_s + d0$$

Here, a sum is calculated over all possible splits $S$ of $X$ with positive isolation index $\alpha_s$. The map $\alpha_s: X \times X \rightarrow \mathbb{R}^{\geq 0}$ is the split metric on $S$ with $\alpha_s(x,y) = 1$ if the two taxa $x$ and $y$ lie on different sides of the split $S$, and 0 otherwise. The number $\alpha_s \geq 0$ is the weight or *isolation-index* of the split $S$, and the map $d_0: X \times X \rightarrow \mathbb{R}^{\geq 0}$ is the so-called *split prime residue* and cannot be decomposed further. A split $S$ with positive isolation index $\alpha_s > 0$ is called a *d-split*, and the system $\Sigma$ of all $d$-splits is weakly compatible and can be computed efficiently (see Bandelt and Dress, 1992a) for details.

If there is no split prime residue, then the distance between any two taxa $x$ and $y$ is precisely equal to the sum of weights of all $d$-splits that separate $x$ and $y$, and thus proportional to the sum of all edge lengths along a shortest path from $x$ to $y$ in the splits graph. In other words, the splits graph provides an exact representation of the given distance matrix.

However, in general, the split prime residue will be positive and so the sum of weights will only give an approximation (from below) of the original distances. The fit of the approximation is measured by the sum of all approximated distances divided by the sum of all original distances, as a percentage. In biological applications, the fit is often quite high and a small split prime residue can be considered as "noise."

## Critical Parameters and Troubleshooting

Split decomposition and related methods are global methods in the sense that any given split must be supported by all quartets of taxa in the input set. As a consequence, when larger numbers of taxa or greater evolutionary distances are involved, the method will often lose resolution simply because individual quartets of taxa will occur that contradict otherwise well supported splits. Depending on the evolutionary distances involved, SplitsTree works well for tens of taxa, but rarely for more than 50 to 100. Often, a more resolved graph can be obtained by excluding a number of taxa from the analysis.

As mentioned above, in the case of split decomposition (or the Buneman tree), the fit value associated with the splits graph is important. If the fit value drops below, e.g., 80%, then this indicates that the displayed graph is not a good representation of the given distances. If the original input data are character sequences and the fit value is low, then one should reconsider the choice of distance transformation that one has made, as this is often an important parameter for the method.

## Suggestions for Further Analysis

When using SplitsTree, in most cases the best results are obtained using the split decomposition method. Given character sequences,

the parsimony splits method (Bandelt and Dress, 1992b) usually produces a very similar result. For small data sets, spectral analysis (Hendy and Penny, 1993) often gives a more resolved picture. The tree-producing variants Buneman-tree, P-tree, and S-tree, respectively, are seldom useful, as they produce very unresolved trees.

For character sequences, a number of different distance transformations should be tried. In most cases, this will show that the analysis is quite robust with respect to the choice of distance transformation.

As an exploration tool for phylogenetic data, SplitsTree should be used side-by-side with other phylogenetic analysis tools described in this chapter.

## Literature Cited


Bandelt, H.-J. and Dress, A.W.M. 1992a. A canonical decomposition theory for metrics on a finite set. *Adv. Math.* 92:47-105.

Bandelt, H.-J. and Dress, A.W.M. 1992b. Split decomposition: A new and useful approach to phylogenetic analysis of a distance data. *Mol. Phylogenet.Evol.* 1:242-252.

Bandelt, H.-J. and Dress, A.W.M. 1993. A relational approach to split decomposition. *In* Information and Classification: Concepts, Methods and Applications. *(*O. Opitz, B. Lausen, and R. Klar, eds.) Heidelberg, Germany.

Buneman, P. 1971. The recovery of trees from measures of dissimilarity. *In* Mathematics and the Archeological and Historical Sciences (F.R. Hodson, D.G. Kendall, and P. Tautu, eds.), pp. 387-395. Edinburgh University Press, Edinburgh.

Dress, A.W.M., Huson, D.H., and Moulton, V. 1996. Analyzing and visualizing sequence and distance data using SplitsTree. *Discrete Appl. Math.* 71:95-109.

Hendy, M.D. and Penny, D. 1993. Spectral analysis of phylogenetic data. *J. Classif.* 10:5-24.

Huson, D.H. 1998. SplitsTree: A program for analyzing and visualizing evolutionary data. *Bioinformatics* 14:68-73.

Jukes, T.H. and Cantor, C.R. 1969. Evolution of protein molecules. *In* Mammalian Protein Metabolism (H.N. Munro, ed.), p. 21-132. Academic Press, London.

Kimura, M. 1981. Estimation of evolutionary distances between homologous nucleotide sequences. *Proc. Natl. Acad. Sci. U.S.A.* 78:454-458.

Lockhart, P.J., Steel, M.A., Hendy, M., and Penny, D. 1994. Recovering the correct tree under a more realistic model of evolution. *Mol. Biol. Evol.* 11:605-612.

Maddison, D.R., Swofford, D.L., and Maddison, W.P. 1997. NEXUS: An extensible file format for systematic information. *Syst. Biol.* 46:590-621.

Steel, M.A. 1994. Recovering a tree from the leaf colorations it generates under a markov model. *Appl. Math. Lett.* 7:19-24.


## Key References


Bandelt and Dress, 1992a,b, 1993. See above.


*The theory of split decomposition and related methods was introduced by Hans-Jürgen Bandelt and Andreas Dress.*

## Internet Resources

http://www-ab.informatik.uni-tuebingen.de/software/splits/welcome_en.html

*This Web site contains additional information on SplitsTree, has links to Web servers running the program, and provides downloads of the different versions of the program.*

---


Contributed by Daniel H. Huson
Center for Bioinformatics Tübingen
Tübingen University
Tübingen, Germany

# Using PEBBLE for the Evolutionary Analysis of Serially Sampled Molecular Sequences



The PEBBLE (Phylogenetics, Evolutionary Biology, and Bioinformatics in a moduLar Environment) application is a relative newcomer to the field of phylogenetic applications. Although designed as a customizable generalist application, PEBBLE was initially developed to implement procedures for the analysis of sequences associated with different sampling times, e.g., rapidly evolving viral genes sampled over the course of infection, or ancient DNA sequences.

One aspect of PEBBLE is the implementation of the serial Unweighted Pair-Group Method using Arithmetic Averages (sUPGMA) algorithm described by Drummond and Rodrigo (2000) for the construction of phylogenies with serially sampled homologous molecular sequence data under the assumption of a molecular clock. The sUPGMA algorithm also estimates substitution rates and intra-sample divergence. Further serial-sample functionality implemented by PEBBLE involves maximum likelihood estimation of substitution rates (Rambaut, 2000; Drummond et al., 2001; *UNIT 6.4*), and tools for simulating serially sampled sequences under the coalescent model (Rodrigo and Felsenstein, 1999). A number of other phylogenetic functions not dealing directly with serially sampled data is also provided by PEBBLE, such as alignment editing and general tree construction. PEBBLE incorporates functionality provided by the Phylogenetic Analysis Library (PAL; Drummond and Strimmer, 2001).

The Basic Protocol describes the use of PEBBLE to infer a phylogenetic tree using the sUPGMA algorithm, and the inference of substitution rate parameters using maximum likelihood. Note that, because one cannot claim that the substitutions are neutral, a substitution rate can only be estimated. If the sequences have evolved neutrally, then the substitution rate is equal to the mutation rate. The Alternate Protocol describes the simulation capabilities of PEBBLE, while the Support Protocol discusses the general use of the PEBBLE application.

## USING PEBBLE FOR ANALYSIS

*BASIC PROTOCOL*

This protocol details the steps involved in performing the sUPGMA procedure, and the maximum likelihood estimation of rate parameters that can be inferred with serially sampled sequence data (see Background Information). The protocol begins with the importing of a sequence alignment and a phylogenetic tree into PEBBLE, and then continues with steps for specifying substitution model parameters and sample information.

The term **ceblet** is used to describe the individual functional modules of PEBBLE (such as the different forms of phylogenetic analysis). A ceblet is represented in a fashion similar to the "wizard" concept (see Support Protocol), and involves input selection, processing, and user interaction, as well as result annotation. The **object store** is the term used to describe the workspace of PEBBLE. The object store is where imported data, and the results of PEBBLE functions, reside until exported by the user. Inputs to ceblets are taken from the object store. The object store is graphically represented on the left side of the main window of PEBBLE (Fig. 6.8.1). The **object view area** situated on the right side of the main window is where objects selected in the object store can be viewed. Individual objects (such as alignments and phylogenetic trees) can be viewed by clicking on the related object. For a more in-depth discussion of the ideas behind the general interface of PEBBLE, the user is strongly urged to read the Support Protocol in this unit.

**Contributed by Matthew Goode and Allen G. Rodrigo**
*Current Protocols in Bioinformatics* (2004) 6.8.1-6.8.26

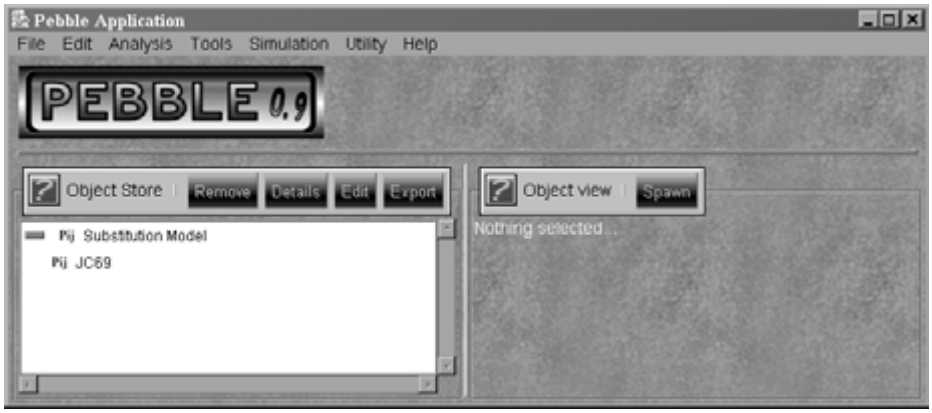


**Figure 6.8.1** The main window of the PEBBLE application, running under the Microsoft Windows operating system. Running PEBBLE under other operating systems may produce minor differences. On the left is the object store, on the right is the object view area.

### *Necessary Resources*

*Hardware*

Any computer capable of running Java 1.1. This includes Apple Macintosh computers (running MacOS 8 or higher) and personal computers running Microsoft Windows or Linux.

*Software*

PEBBLE main application, and the Java Runtime Environment (JRE) version 1.1 or higher. The PEBBLE application can be downloaded from *http://www.cebl.auckland.ac.nz* (by following the Software link). Notification of updates and milestone releases of the PEBBLE application can be obtained via E-mail by sending an E-mail to *pebble_notice-subscribe@yahoogroups.com*. Bug reports regarding the PEBBLE application may be sent to *pebble-bugs@yahoogroups.com*.

*Files*

PEBBLE accepts sequence alignment files in Nexus (*UNIT 6.4*), PHYLIP (*UNIT 6.3*), Clustal (*UNIT 2.3*), or FASTA (*APPENDIX 1B*) formats. PEBBLE accepts tree files in Nexus (*UNIT 6.4*) and Newick (*UNIT 6.2*) formats.

1. Download and install PEBBLE (*http://www.cebl.auckland.ac.nz*).

### *Import data files into PEBBLE*

2. Start the PEBBLE application by double-clicking the PEBBLE executable icon.

   *The PEBBLE application begins by first showing a "splash" screen to show startup progress. Following startup, the main window will appear, similar to that shown in Figure 6.8.1 (the contents of the object store may differ).*

   *The left side of the main window represents the object store. The initial object store will contain default objects such as an example distance matrix, tree, and alignment. Upon selecting an object from the object store, a representation of that object will appear on the right side of the window.*

3. Open the File menu at the top of the PEBBLE window.

*There are six items in the file menu: the New submenu, the Import submenu, Export, Open Results, Save Results, and Exit. The New submenu contains options for constructing certain phylogenetic objects (such as substitution models). The Import submenu is for opening data files from external sources. The Export menu item allows the exporting of the currently selected object, or objects, in the object store (the same result as clicking the Export button just above the object store). The Open and Save Results items are used for loading and storing objects in PEBBLE's internal format.*

*The internal format of PEBBLE should be used only as temporary storage, as newer versions of PEBBLE may be incompatible with PEBBLE files from older versions of PEBBLE.*

4. From the Import submenu under the File menu, select either the Nexus Alignment or General Alignment option to open an alignment file. Select a suitable alignment file, and then click the Open button.

   *PEBBLE does not attribute any special information to file name suffixes (e.g., `.nxs`), so the user is free to attempt to open any file.*

   *The General Alignment reader of PEBBLE is very lenient and will attempt to read the input file even if it is not an alignment. Care should therefore be taken that the imported alignment looks as intended. PHYLIP and Clustal sequential and interleaved files can be read. The general reader will report two possible parsing errors: (1) if the file starts with `#Nexus`, in which case the Nexus Alignment file reader should be used; and (2) if a sequence appears to be repeated.*

   *The General Alignment reader will prompt the user for the residue type of the alignment. After selecting the residue type, the user should click the Next button.*

   *The Nexus Alignment reader is for reading Nexus files. The Nexus Alignment reader does not prompt the user for the residue type of the alignment, as that information is part of the Nexus format. Sequential and interleaved Nexus alignment files are supported.*

5. Annotate the resulting alignment object.

   *When a ceblet is completed in PEBBLE, the user is required to annotate the results before they are placed in the object store. The name is mandatory, while the description is optional. When the annotation has been entered, the user should click on the Finished button. If the given name matches something already in the object store, the Finished button will not be selectable; changing the name will enable the Finished button.*

   *For a detailed discussion on ceblets and the object store, the reader is again referred to the Support Protocol.*

   *In Figure 6.8.2 the example alignment from the file `examples/ExampleAlignment.nxs` has been imported (as a Nexus file) and is being viewed in the object-view area.*

6. Select either Newick Tree or Nexus Tree from the Import submenu under the File menu.

   *This step is only required if an externally generated tree is to be used for maximum likelihood rate estimation, or for simulation. It is not required for sUPGMA analysis.*

   *PEBBLE currently only supports the importing of Nexus format trees, or trees represented in parenthetical notation (i.e., Newick formatted trees; UNIT 6.2). Nexus support allows for trees with a "Labels" block (UNIT 6.4).*

   *The resulting tree object should be annotated as in step 5.*

***Enter substitution model information***

7. Open the File menu, then open the New submenu.

   *The New submenu is the access point for the fresh creation of a number of phylogenetic objects.*

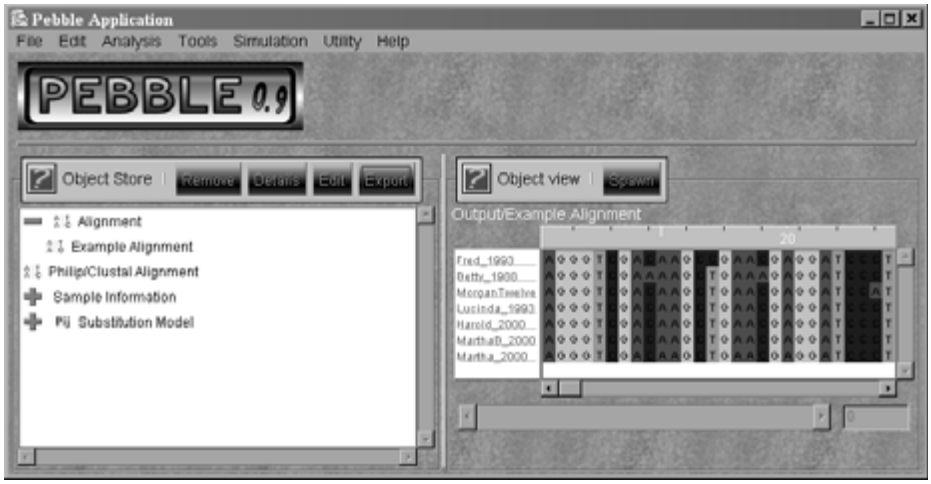


**Figure 6.8.2** The main window after having imported an example alignment, with that alignment now selected. A view of the example alignment is shown in the "object view" window.

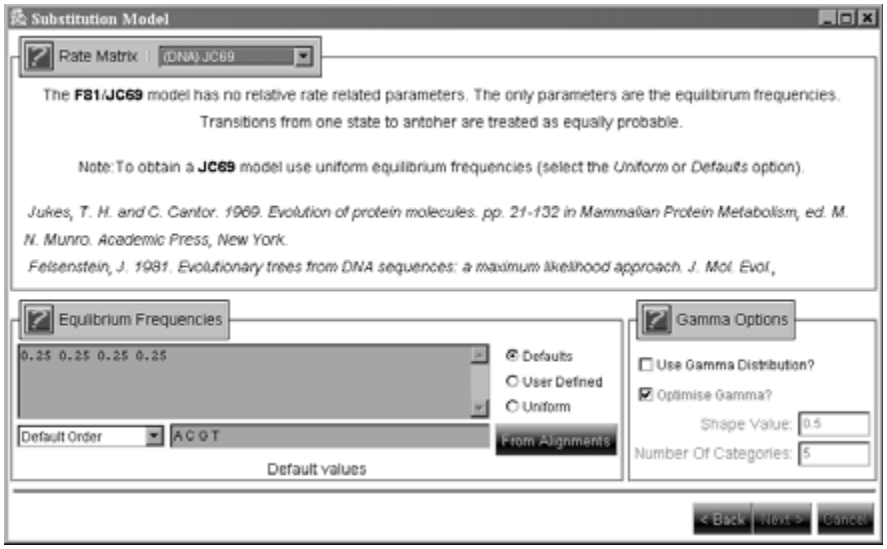


**Figure 6.8.3** The "new substitution model" ceblet. The rate matrix is chosen by using the drop-down menu in the top left of the window. The rate matrix parameters are entered in the upper center of the window. The equilibrium frequencies are entered in the lower left area, and the gamma distribution is defined in the lower right of the window.

8. Select the Substitution Model option to create a new substitution model.

   *The substitution model input ceblet widow will appear as in Figure 6.8.3.*

9. Select the required underlying rate matrix from the drop-down menu in the title area of the Rate Matrix area (Fig. 6.8.3).

   *This ceblet allows the user to specify the appropriate model of sequence evolution. Both nucleotide and amino acid substitution models are available, and future versions of PEBBLE should also include codon-based models. The residue type of the model (for example, nucleotide) does not have to match the residue type of any alignment with which*

*it is associated, as PEBBLE will automatically do the required conversion. However, users are strongly encouraged to keep residue types consistent.*

*Upon selecting a model, an associated description and reference will appear in the main area of the Rate Matrix area (the user may need to enlarge the window to see the full information). If the model has relative rate parameters that require user specification, then input fields will also appear for each parameter.*

*Under the likelihood estimation procedure, PEBBLE allows model parameters to be estimated. If the model is the basis of a substitution model optimization, the user may wish to set the values to an initial estimate. If the model is to be held constant in an analysis, the user should enter the predetermined values for each parameter.*

10. Enter the equilibrium frequencies in the Equilibrium Frequencies area at the bottom left of the window (Fig. 6.8.3).

    *There are three different equilibrium frequencies options. The first option is to use uniform frequencies, the second is to use the default value for the model (quite often uniform), and the third option is to enter them manually. Users can make their particular choice by selecting the appropriate option (above the From Alignments button). If User Defined is selected, the frequencies must be entered in the appropriate fields. Internally, PEBBLE works with equilibrium probabilities, but general frequency information will be accepted and scaled automatically.*

    *To select frequencies based on one or more alignments, select the From Alignments button, available only when the selected option is User Defined. PEBBLE will then prompt the user for one or more alignments from the object store, as the basis to estimate the frequencies. Multiple alignments may be selected by holding down the Shift key.*

    *The user should be aware of the order in which the user-defined frequencies are given. The text field below the frequency input area holds the expected order of frequencies, as described by single characters for each residue state. For example, to state that the order of nucleotide frequencies is in order of adenine, cytosine, guanine, and then thymine, the ordering "A C G T" is used. Using the drop-down menu to the left of the Order text field, users can select to use the default ordering, a typical alternative order, or their own ordering. If the user chooses to specify the order, it should be entered in the Order text field.*

11. If a gamma rate distribution is required, enter related information in Gamma Information area at the bottom-right of the window.

    *To use a gamma rate distribution to model varying rates across sites with an analysis, select Use Gamma. The user must enter the number of rate categories. During an analysis, the continuous gamma distribution is approximated by grouping rates into a finite number of categories (Yang, 1994). The more categories, the more accurate the modeling of the gamma distribution, but the more computational time required. The user should also enter an initial alpha value.*

### *Enter time/order information*

12. From the New submenu under the File menu, select Sample Information.

    *A ceblet will appear prompting the user for a source of label information. The source could be a tree, an alignment, or a previously defined Sample Information object. The user should select the source of interest and then click the Next button.*

    *A **sample information** object describes the ordering relationship of sequences in terms of what time they were obtained. This information may be relative, specifying that sequence "A" was obtained before sequence "B," without information on the time difference between sequences "A" and "B." Alternatively, the information given may also be absolute, specifying exactly the time difference between obtaining sequence "A" and sequence "B."*

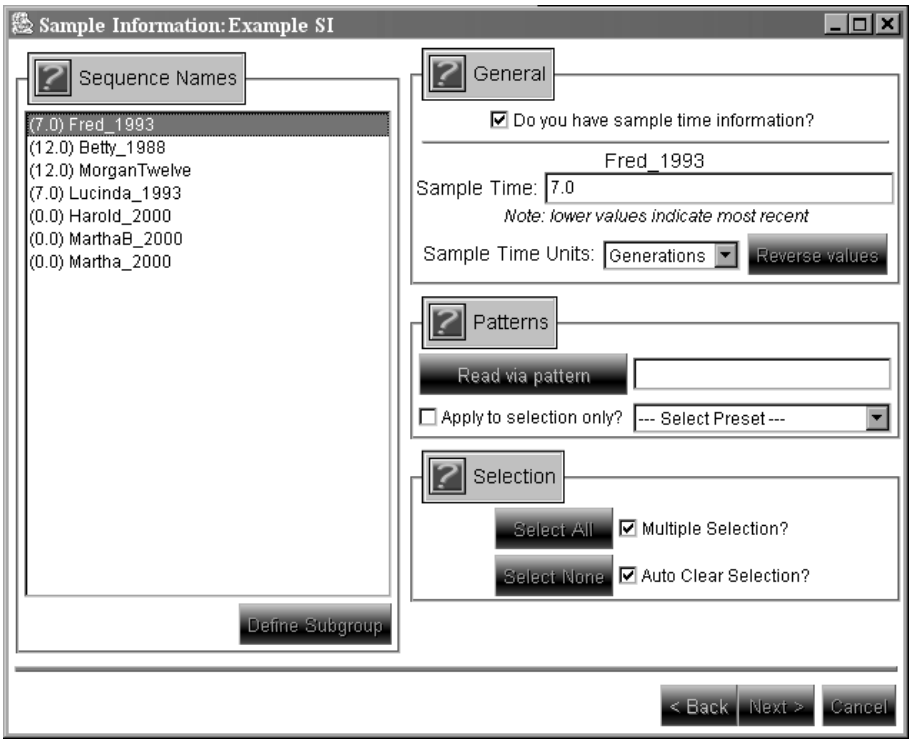


**Figure 6.8.4** The "sample information input" ceblet. Sequence names are listed on the left, along with any time/order information attributed to each sequence. The right side contains controls for selecting time or order information.

*The sample information input ceblet is shown in Figure 6.8.4, with the time information for the example alignment already entered.*

13. If only order information is available, unselect "Do You Have Time Information?" (Fig. 6.8.4).

    *Sample or time information is given by relating numbers to sequence. Such numbers signify either the chronological times associated with the sequences (relative to the most recent sample), or the order in which the sequences were sampled.*

    *Sample times are taken as time from the most recent time point. For example, time "0" is the most recent time point and time "10" is ten chronological units back in time from the most recent time point.*

    *Sample order is given by arbitrary numbers; ordering is imposed by the relationship whereby higher numbers signify time points further back in time than lower numbers. In general, the user would enter "0" as the ordering for the most recent sample, "1" for the next most recent sample, "2" for the sample before that, and so on.*

    *The time/order information is displayed in brackets before the sequence name, in the sequence list. If sample time/order information is in the opposite form, such that higher values represent more contemporaneous samples, step 17 describes performing the necessary corrections.*

14. If time information is to be used, select the related units.

    *While setting the units of time correctly is not vital for a correct analysis, it is allows the computer to correctly report time and rate information using the correct units at later points in the analysis. The units may be changed by using the Units drop-down menu. This menu will not be selectable if only order information is available.*

15. If the sequence names contain time or order information, the user can choose to extract this information by entering an extraction pattern and selecting the Read Pattern button.

    *PEBBLE has a limited ability to read in time or order information directly from the label names. This function reduces data entry considerably if the sequences have well formed names. The patterns use a limited regular expression language with* `%N` *to signify where the time or order value is given. For example, the pattern* `pC_%N` *instructs PEBBLE to extract time information from all sequences containing* `pC_` *in their label, with the time or order value appearing directly after the underscore. The user should enter the pattern in the Pattern text field, and then click the Read Pattern button. If no sequences are selected, the pattern extraction will be applied to all sequences; if one or more sequences are selected, the pattern extraction will only be applied to the selected sequences. To find out the latest information on using the pattern matching, the user should click on the question mark icon next the Pattern heading.*

    *To extract the time information from most of the sequences in the example alignment previously imported, the pattern* `_%N` *can be used. The sequence named MorganTwelve does not follow the pattern of other sequences, and will be left without time information by the pattern reader.*

16. If the sequence names do not embed time or order information, enter time/order information directly.

    *Sequences that cannot be allocated time or order values from their labels will need to be entered manually. Select sequences that have the same time or order value and then enter the related value in the text field labeled either Sample Time or Sample Ordering (the exact text changes depending on use).*

    *For the example alignment sequences, the sequence MorganTwelve needs to have time information entered directly. The time for this sequence is 1988.*

17. Reorder temporal information.

    *If the embedded time/order information in the label is arranged such that the highest time/order represents the most recent sample, then highlight the related sequences (use the Select All button to highlight all sequences), and click the Reverse Values button. This action will result in reversing all highlighted values such that what was the highest value is now zero and what was the lowest value is now the highest. The aforementioned operation will also affect all sequences if no sequences are selected.*

    *In the case of the example alignment, the ordering is such that the highest value represents the most recent sample. After reversing the values, the Sequence Names area should display:*

```
(12.0)MorganTwelve
(12.0)Betty_1988
(7.0)Fred_1993
(7.0)Lucinda_1993
(0.0)MarthaB_2000
(0.0)Martha_2000
(0.0)Harold_2000
```

18. Continue steps 15 to 17 until all sequences are accounted for.

    *The user will not be able to finish until every sequence has a time or order value attached to it. The Next button will become selectable once data entry is complete. The final step will be annotating the resulting sample information object.*

    *All sequences for the example alignment should be accounted for without the need to repeat any steps.*

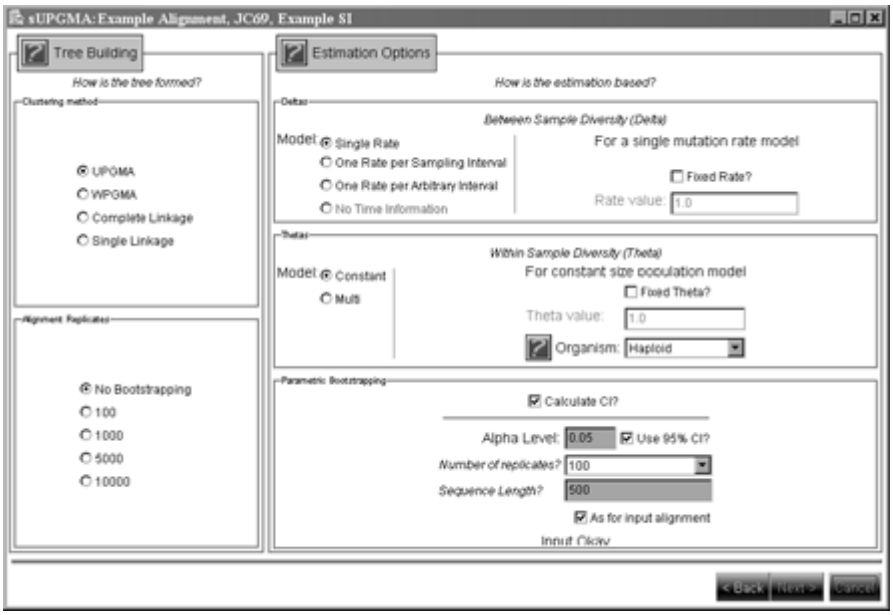


**Figure 6.8.5** The sUPGMA ceblet, waiting for various options to be selected. The clustering method and number of nonparametric bootstraps are selected on the left. The rate models and the parametric bootstrapping parameters are selected on the right.

### *Deriving a sUPGMA tree*

19. Open the Analysis menu at the top of the PEBBLE window (Fig. 6.8.1).

    *The Analysis menu is the access point for analysis-based ceblets. The majority of ceblets in PEBBLE are accessed through three menus, Analysis, Tools, and Simulation.*

20. From the Cluster submenu select "sUPGMA."

    *The user will be required to select one alignment, one substitution model, and one sample information object. The sample information object must be appropriate for the given alignment (if it is not, PEBBLE will inform the user). Once the information has been selected, the user should click the Next button to continue. This will present the user with a display similar to that shown in Figure 6.8.5.*

21. Select clustering method in the Tree Building area (Fig. 6.8.5).

    *PEBBLE can perform clustering using the UPGMA (Average Linkage), WPGMA, Single Linkage, or Complete Linkage methods. While the user is free to experiment with all clustering methods, Drummond and Rodrigo (2000) only examined the use of UPGMA clustering.*

22. Select parameter estimation options in the Estimation Options area.

    *There are two classes of parameters that may be estimated: within-sample divergence (the theta rates, see Background Information), and between-sample divergence (substitution rate). Within-sample divergence may be estimated by a single value common to all samples, or, alternatively, a different value may be estimated for each sample.*

    *For substitution rate there are four options: Single Rate, One Rate per Interval, One Rate per Arbitrary Interval, and No Time Information. The first three options are selectable when time information is available, and only the last option is available when ordinal information alone is provided. All four models are analogous to the rate models described in step 32.*

23. Select the parametric bootstrapping options in the Estimation Options area.

*Confidence intervals can be estimated when a constant between-sample divergence and a single substitution rate is assumed. If the parameterization has been set appropriately, the Calculate CI option will be available, and the user may select the option. The level of confidence for the confidence interval must be indicated by either entering an alpha value or selecting Use 95% CI for a 95% confidence interval.*

*Confidence intervals are estimated using parametric bootstrapping (Huelsenbeck et al., 1996). Parametric bootstrapping involves creating a set of simulated alignments by first generating a set of trees based on the estimated parameters using a serial coalescent-based simulation assuming a constant population size and constant substitution rate. Sequences are simulated across each generated tree using the specified model of evolution, creating the bootstrap replicates. The options that need to be set are sequence length and number of bootstrap replicates. To enter these values, use the drop-down menu labeled Number of Replicates and the text field labeled Sequence Length. To use the same sequence length as the input alignment, select the "As for input alignment" check box. In general, a good option is to have sequence length the same as the input alignment, and to select the "1000" replicates option (at the very least).*

24. Select the alignment bootstrapping options in the Tree Building area (Fig. 6.8.5).

*Nonparametric or alignment-based bootstrapping may also be performed on trees generated by sUPGMA. The nonparametric bootstrapping method utilized in PEBBLE forms replicate alignments by selecting sites of the original alignment randomly and with replacement, so that the lengths of the resulting replicate alignments are the same as the original. Nonparametric bootstrapping for sUPGMA is provided only to allow the user to get an indication of clade support in the resulting tree. Nonparametric bootstrapping was not discussed in Drummond and Rodrigo (2000).*

*The full sUPGMA analysis is performed on every bootstrap replicate to generate a tree. The returned tree is the originally inferred tree, and not a consensus tree.*

25. Perform the analysis.

*Once all options have been finalized, the user should then click on the Next button to perform the analysis. If no bootstrapping is performed, the analysis will usually take a matter of seconds to complete. If bootstrapping is enabled, the total time taken for the analysis will increase proportionally to the number of replicate alignments generated. PEBBLE will show a progress bar to inform the user of current progress.*

26. Examine and make a note of the parameter estimates.

*Upon completion of the sUPGMA algorithm, the user will be presented with a summary of the parameter estimation and related confidence interval information (if it was requested). The user should make a note of the results and then click Next.*

27. Save the result in the object store.

*Once the analysis is complete, the user will have the option to name the resulting tree so that it can be placed in the object store. Analysis results, including parameter estimate information, may be obtained at a later date by clicking on the question mark icon that appears with the tree when viewed from the main PEBBLE window.*

*Figure 6.8.6 shows a complete analysis for the previously mentioned example alignment, with a single substitution rate, a single theta, and a Jukes-Cantor substitution model (Jukes and Cantor, 1969).*

### *Estimating substitution rates using maximum likelihood, with a fixed tree topology*

28. From the Analysis menu under Maximum Likelihood, select Serial ML Estimate.

*A ceblet will appear prompting the user to select a template. A template, as applied to a ceblet, is a style of input. If an example of the substitution model to be used is already in the object store, select the Predefined Model template; otherwise select the Normal*

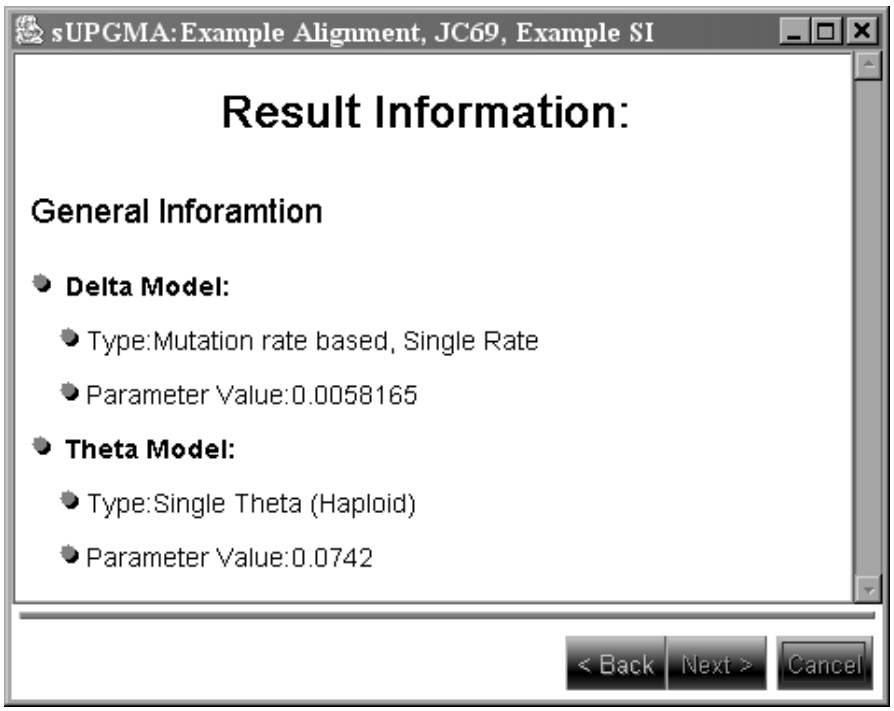


**Figure 6.8.6** The result of using the sUPGMA analysis on the example data, described in the text. A single substitution rate was assumed as well as a single within-sample divergence parameter.

*template. The difference between the choices of template is simply whether the substitution model is taken from the object store or created during the ceblet execution.*

*For general information on templates, see Support Protocol.*

29. Select the appropriate phylogenetic data. Click Next to continue.

*Regardless of whether the Predefined Model or Normal template is used, a tree, alignment, and sample information objects must be selected from the object store. The labels of the alignment, tree, and sample information must be identical. PEBBLE will inform the user if the labels do not match, and the analysis will not continue. If the user selects the Predefined Model template, a substitution model must also be selected from the object store.*

*Upon clicking Next, the user will be presented with a display similar to Figure 6.8.7. If a predefined substitution model was supplied, the Model area will not appear.*

*If a substitution model is selected from the object store, it will be used by the likelihood-estimation tool only as a means to describe model type and initial parameters. The substitution model in the object store will never be altered in any way, as a new substitution model object will be created if the analysis involves substitution model estimation.*

30. If no substitution model was supplied, select a substitution model using the Model area (Fig. 6.8.7).

*If a JC69 model is used, select the "JC69" option. If a more complex model is required, select the "Other" option, and click the Select Model button. A ceblet dialogue box will appear with the Substitution Model Input ceblet, and the model should be selected as described in steps 7 through 11.*

31. Select the subject of estimation from the Subject area.

*PEBBLE will allow the user to either estimate just the branch lengths of the tree or estimate both the model of evolution and the tree simultaneously. Select either Tree or Tree and*

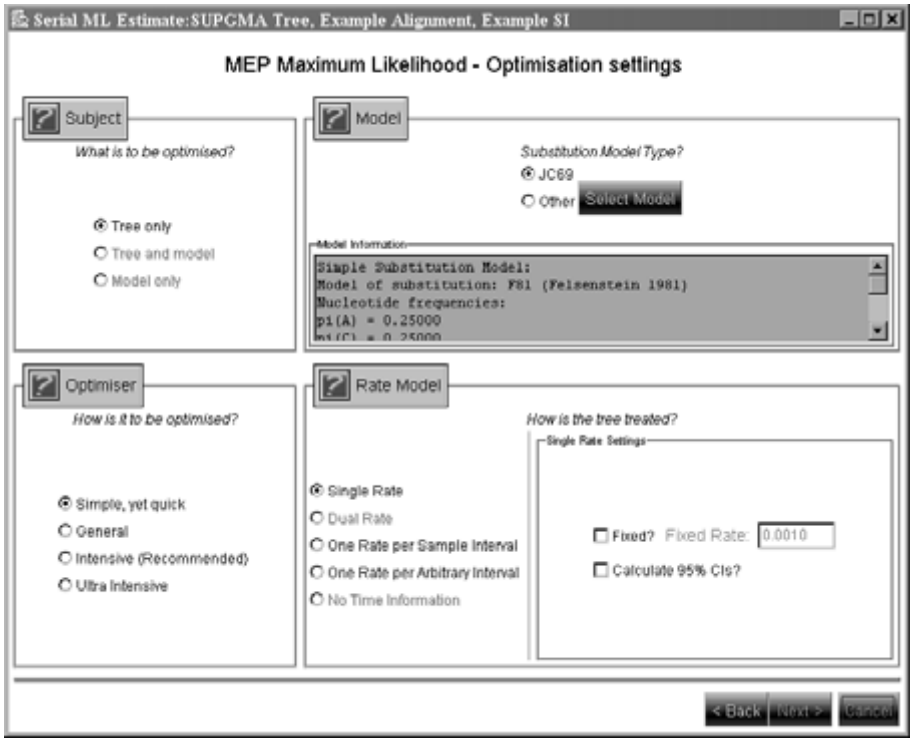


**Figure 6.8.7** The maximum likelihood estimation ceblet for serially sampled data. The top left area in the window allows the user to select the subject of optimization; only Tree is available, as the model selected (JC69) does not have any free parameters. The top right area is for selecting a substitution model (this does not appear if a model was given from the object store). The lower left area allows the user to select the sophistication of the optimizer method, and the lower right area is where the assumptions regarding substitution rate are selected.

*Model, depending on choice. The user is advised that estimation of the model can dramatically increase the total time to maximize the likelihood, depending on the model used. If the substitution model chosen has no free parameters (such as a JC69 model) then the Tree and Model option will not be selectable.*

32. Select the rate model from the Rate Model area (Fig. 6.8.7).

*When time information is available, PEBBLE is able to model substitution rate in three different ways: a single rate over all intervals, one rate per sampling interval, and different rates over prespecified intervals that do not necessarily match the sampling intervals. When only ordinal information is available, PEBBLE estimates only the divergence between samples in each sampling interval.*

*The required model should be selected from the list in the Rate Model area. A discussion of the various rate models can be found in Drummond et al. (2001). The different types of models, as named in the Rate Model area, are described as follows.*

a. The Single Rate model is the SRDT model first described by Rambaut (2000). This model assumes a constant rate from the time of the first sample to the time of the last sample.

b. The One Rate per Interval model is the basic MRDT model described by Drummond et al. (2001). This model assigns an individual rate to each interval between samples.

c. The One Rate per Arbitrary Interval model is the more general MRDT model where rate-change times do not necessarily coincide with sampling times.

d. The No Time Information model is the basic MRDT model for ordinal information. The difference between the No Time Information model and the One Rate per Interval model is that, instead of inferring a substitution rate between samples, the expected number of substitutions per site within each sampling interval is inferred (essentially a combination of time and substitution rate which are both unknown).

*An additional rate model called the Dual Rate model is available and operates in a similar fashion to the Single Rate model, except that one rate is used for a particular clade of the tree and the another rate is for the remainder of the tree. As of PEBBLE 1.0, the effectiveness of this rate model has not been investigated, and therefore is only provided for the user's curiosity. The Dual Rate model is only available when the sample information has a defined subgroup. A subgroup can be defined when the sample information object is being created by clicking on the Define Subgroup button.*

33. Select the rate-model parameters.

*Upon selecting a rate model, related controls will appear in the area to the right of the rate model list, if applicable.*

*The Single Rate model has options for fixing the substitution rate and for calculating a 95% confidence interval. The rate is fixed to a particular value by selecting the Fixed Value option and entering the fixed value in the nearby text field. If Fixed Value is not selected, PEBBLE will infer the rate. Calculation of a 95% confidence interval for the value of the substitution rate is enabled by selecting the Calculate CI check box; this is only available when the rate is not fixed. The asymptotic confidence interval is approximated by finding the two substitution rates that correspond to likelihoods on either side of the maximum likelihood value that give differences equal to $\chi^2/2$ (with one degree of freedom) less than the maximum likelihood. In essence, this approach is finding the set of substitution rate parameters that would not be rejected under a likelihood ratio test. See Rambaut (2000), Rambaut and Bromham (1998), and Kalbfleisch (1985) for more information.*

*The One Rate per Interval, and No Time Information models have no user specified parameters.*

*The One Rate per Arbitrary Interval requires the user to enter the rate-change times. The times (in the same time units as the sample information) at which the rate can change should be entered in the large text area that appears. The user will not be allowed to continue if the times are illegal. The times are illegal if more than one rate change occurs within one sample interval.*

34. Select the optimization method in the Optimiser area (Fig. 6.8.7).

*Maximum likelihood estimation is done by finding the set of model parameters (including branch lengths) that has the highest probability of giving the observed data (the likelihood), subject to the correct topology being specified. The process of finding such a set of parameters is an optimization problem.*

*The problem of maximum likelihood optimization is not trivial. It is more difficult when working with clock-constrained trees. All optimization techniques used by PEBBLE suffer from the problem of local maxima. The likelihood surface can be likened to a hilly terrain, where the heights of the hills and peaks are the likelihoods and the position locators (e.g., latitude and longitude) are the possible parameter values. Optimizers, in general, function by traversing this terrain searching for the highest point (the maximum likelihood estimate). Sometimes an optimizer may find what appears to be the maximum point because it cannot "see" any better points to move to. If that point is not the true maximum likelihood estimate (the global maximum) the optimizer will have settled upon what is called a local maximum. For this reason, PEBBLE cannot guarantee finding the maximum likelihood estimate. To give the user a good chance at obtaining the global maximum, a number of optimization methods are available. The trade-off between methods that are better at finding the global maximum over those that are not as good is in terms of speed.*

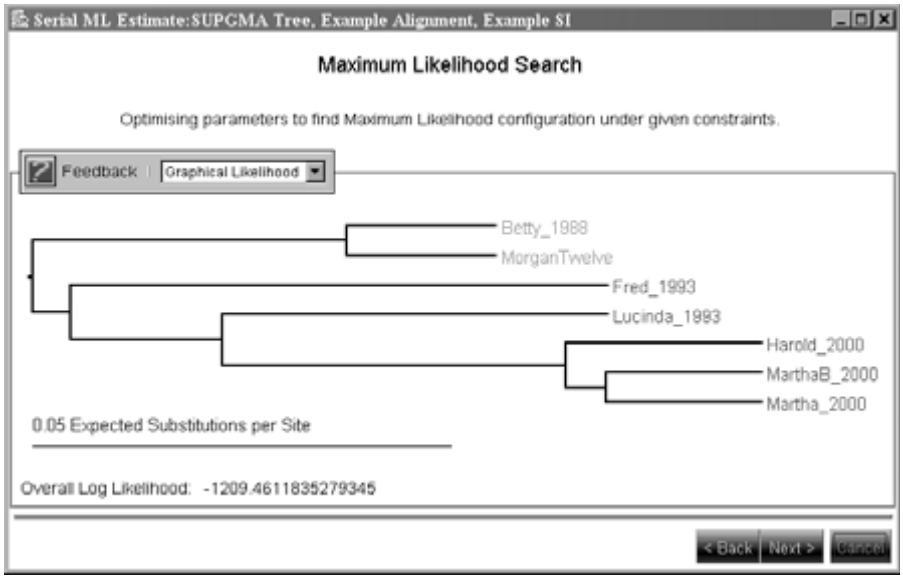


**Figure 6.8.8** The maximum likelihood visualization tool in mid-optimization. The branch lengths of the tree are updated in "real time" as the optimization algorithm searches for the maximum likelihood estimates.

*If time is not an issue, the most comprehensive optimizer should be selected. If time is an issue, the simpler-but-faster optimizers should be utilized. Multiple optimizations should be executed to detect if local maxima are a problem. On very large data sets, the simpler but quicker optimizers may be the only viable choice.*

*For more information on the specific properties of each optimizer, click the question mark icon next to the Optimization Style heading.*

35. Begin the optimization by clicking Next (at the bottom right of the window; Fig. 6.8.7), once all options have been set.

    *This will move the ceblet to the optimization stage. No user interaction is required during this stage. A message will appear informing the user that maximum likelihood optimization is being performed. The optimization process may take anywhere between a few seconds and an indefinite amount of time, depending on the size of the tree, the size of the alignment, the complexity of the model, and the speed of the computer. The ceblet will automatically move to the result annotation phase when the optimization is complete.*

36. Select the optimization view.

    *For the user's viewing pleasure, and for possible educational purposes, PEBBLE can give visual feedback on the current progress of the optimization process. Placed at the top left of the optimization display (Fig. 6.8.8) is a drop-down menu that allows the user to choose a dynamically updated view of the optimization progress. By selecting General Likelihood, the user can view the current maximum likelihood value obtained, and by selecting Graphical View, the user can view the current maximum likelihood tree. Both displays update automatically as the optimization process continues.*

    *The extra computational time used in the dynamic updating process can extend the time that an optimization requires to complete. To gain maximum optimization speed, the No Information view should be used.*

37. Examine and make a note of the parameter estimates.

*Upon completion of the optimization process, a summary of the parameter estimation and related confidence interval information (if it was requested) will appear. The user should make a note of the results and then click Next. The summary is available at a later point when the tree object has been stored in the object store by clicking on the question mark icon that appears with the tree when the tree object is viewed.*

38. Annotate results.

*Depending on whether the subject of optimization was Tree Only or Tree and Model, there may be more than one result. In the case of Tree and Model, the first result will be the estimated tree. Clicking Next after annotating the tree will move the ceblet to annotating the estimated substitution model. Give the substitution model object a name and then click Finish. Both the estimated tree and substitution model will be added to the object store after Finish has been clicked.*

***SUPPORT PROTOCOL***

## GENERAL USE OF PEBBLE

This protocol describes the basic use and design of the CEBL Application Framework, from a user's point of view. In particular it provides an introduction to the idea of a ceblet. It is provided as a more thorough introduction to the CEBL framework then given throughout the Basic Protocol.

### General Overview

The PEBBLE application is built around the notion of a set of modular components called **ceblets**, and an **object store**. A ceblet is a mini application module, or function that may or may not interact with the user. The object store is an internal store of data objects that are either imported from external sources (such as a Nexus file) or are the result of operation of PEBBLE. The user should approach PEBBLE as an experimental laboratory. The object store holds the materials of the scientist, and the ceblets represent experiments performed upon such materials.

### A General Introduction to Ceblets

Ceblets are based around the "wizard" concept made popular in the Microsoft Windows series of operating systems, and involve moving through a number of steps to achieve a final outcome. At the bottom of any ceblet are three buttons: the Back button, the advancement button (labeled either Next or Finished depending on the state of the ceblet), and the Cancel button.

The advancement button allows the user to step through the different steps in the ceblet. The Back button may be used to move the ceblet back a step in the ceblet. In general, moving back will negate the effects of moving forward. The Back button will not be enabled in cases where the computer is busy, or where it is not possible to move back through the ceblet. The user may also click the Cancel button at any point. Clicking Cancel signifies that the user is no longer interested in the results of that particular ceblet, and any current analysis being performed, or previously performed by the ceblet, will be lost.

Ceblet execution is broken into three phases, where each phase may have zero or more steps. The three phases of ceblet execution are: the object input phase, the user interaction phase, and the result annotation phase.

The **object input phase** allows for selecting input. A ceblet may require input objects, such as an alignment or a substitution model, which must come from the object store, or it may only require user interaction. If input is required from the object store, it is acquired through the object input phase. In the object input phase, the user will be presented with a window similar to that shown in Figure 6.8.9. The display is divided up into the title

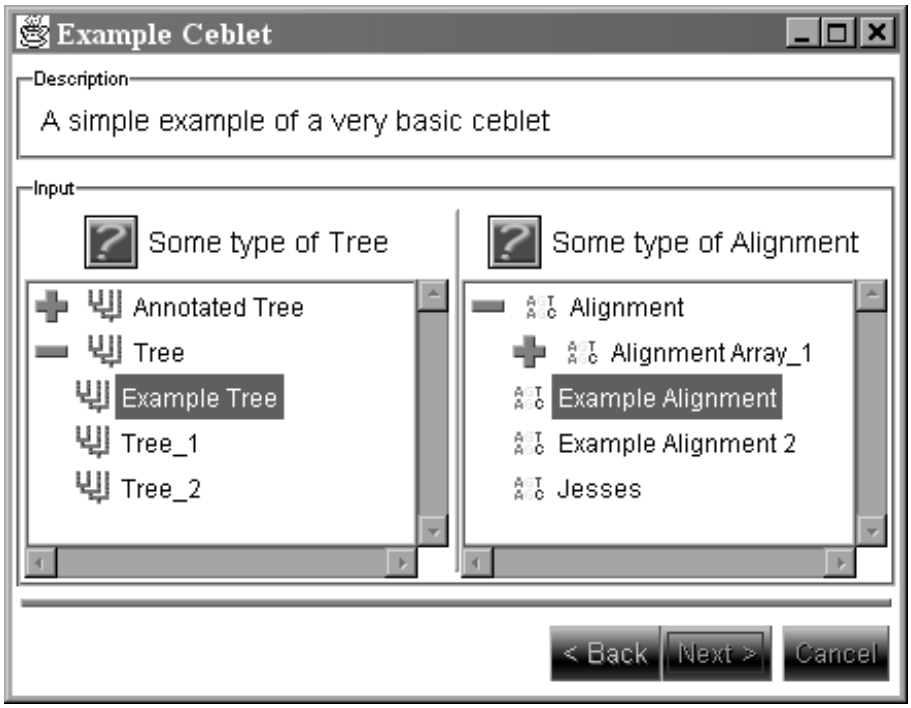


**Figure 6.8.9** For ceblets that require one or more input objects, the first phase of ceblet execution will be object input. In this example the user is prompted for a Tree object and an Alignment object, which must be selected from the object store.

section at the top and separate areas for each input required. The number of objects, and type of objects requested, will depend on the ceblet. A user requiring more in-depth information about a particular input requirement should click on the question mark icon closest to the input of interest. The user should select the desired inputs using the mouse, and then click the advancement button (labeled Next) to continue.

In general, a ceblet will have one or more templates. A **template** is a description of the type of object store requirements. In the case of ceblets that have multiple templates, the user must first select which particular template is to be used before proceeding. A ceblet's behavior may vary slightly depending on the template, to suit the style of input. In cases where a ceblet has more than one template, the user should select the template of choice, during the object input phase, by using the drop-down menu at the top left of the window. Information on the particular choice of template is available by clicking on the question mark icon in the title area towards the top of the window.

The **user interaction phase** is dependent on the particular ceblet. In some cases, no interaction will be required, and the user interaction phase will only involve internal computation by the ceblet (which in many cases may take a short period of time). In other cases, the ceblet will prompt the user to input data or to adjust settings. The user should do as requested, and then click the advancement button (labeled Next) to continue. If the requested information is not given completely, or correctly, the ceblet will prevent the user from continuing by disabling the advancement button. The advancement button will also be disabled while the ceblet is executing any internal computation. An example of the user interaction phase is given in Figure 6.8.10.

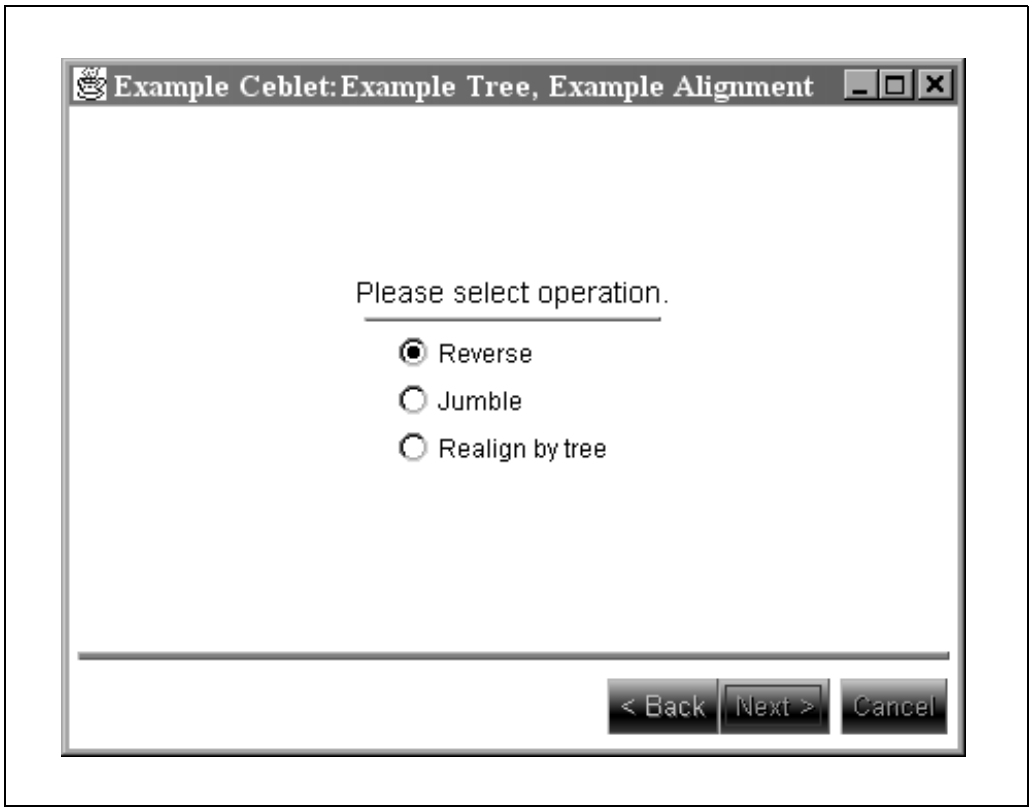


**Figure 6.8.10** A simple example of user interaction within a ceblet. The user is being prompted to select the requested operation from a short list. To continue to the next phase of ceblet execution the user would click on the Next button.

The **result annotation phase** is the final phase of ceblet execution. In this phase, the user is expected to provide annotation information regarding each object created by ceblet (i.e., each result). The user will be presented with a display similar to that in Figure 6.8.11. At this phase, the user is expected to give a name to the newly created result, and, if desired, to give a brief description. Assuming that all results have been annotated correctly, the advancement button will change to show the word Finish. Clicking the Finish button signifies the user's acceptance of the result, and the result object, or objects, will be added to the object store. The Finish button will not be selectable if the chosen name for the result object matches the name of an object already in the object store.

In certain cases, the ceblet may not actually generate any result. Ceblets that export objects are an example of those that do not generate any result (for the object store). In such cases, the final phase becomes a message detailing that the operation was a success, and the Finish button simply closes the window.

### Using the Object Store

The object store is a hierarchical "tree" of objects that appears in the main window of the PEBBLE application. Result objects created by ceblets, or data imported from external sources, appear in the object store. Such objects are grouped according to type (such as alignment, tree, or substitution model). Within the general type groupings there may also be subgroups to specify subtypes, as well as groups of objects that were generated at the same time.

The graphical view of the object store allows the user to manage, export, and view objects. In the title area (marked Object Store) are four control buttons: the Remove button, for removing objects, the Details button, for changing object annotation, the Edit button, for

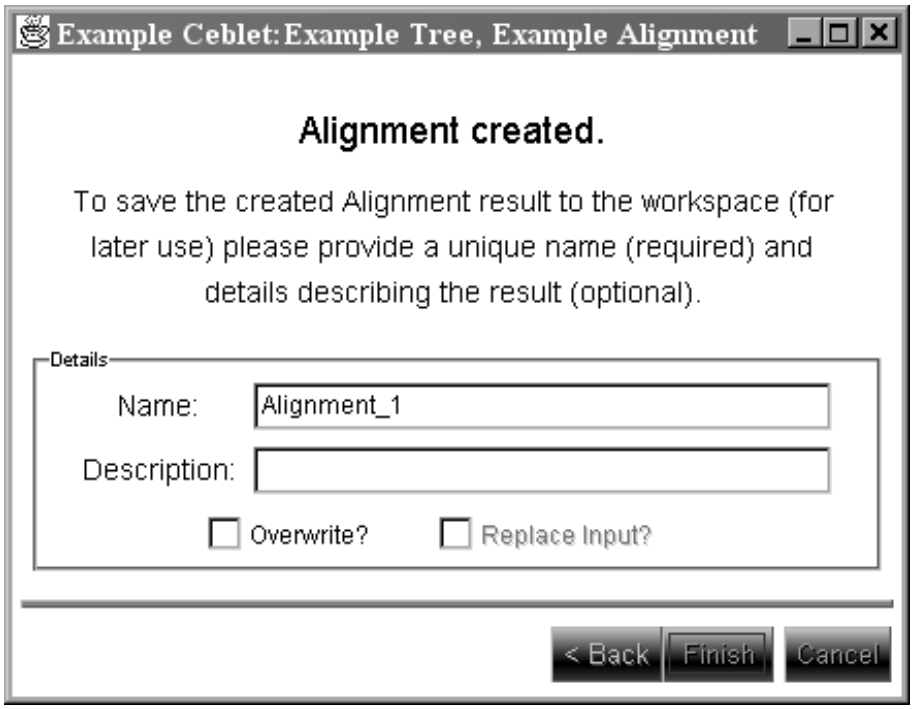


**Figure 6.8.11** The last phase of Ceblet execution prompts the user to annotate any results generated. In this case, an alignment has been created, and a default name is presented to the user for editing. When the user has supplied a name, and, optionally, a description, clicking the Finish button will result in the Alignment object being added to the object store for later use.

editing selected objects, and the Export button, for exporting an object to disk. Objects selected by the user, using the computer's mouse, appear in the object view area to the right of the object store.

The object store is stored on disk between subsequent sessions. When the user closes the PEBBLE application (by closing the main window or using the Quit option in the File menu), the contents of the object store are saved to disk. Upon beginning the next session of PEBBLE, the saved object store will be reloaded. In PEBBLE version 1.0, the object store will not be saved if the PEBBLE application is forcibly stopped (for example, by using the Task Manager under Windows), or if the computer is shut down while PEBBLE is running. This is an issue with using Java and is not specific to the PEBBLE application. Future versions will fix this problem (dependent on the version of Java used).

*To export objects*

1a. Select the object or objects to export from the object store.

*To select more than one item, hold down the Shift key while selecting objects with the mouse.*

*If the selected object or objects cannot be exported, the Export button will not be enabled. Often, a single object of a certain type (for example, an alignment) may be exported while multiple objects of the same type may not.*

2a. Click the Export button.

*The user will be prompted for a filename for the exported file. In some cases, after the user has selected a filename, a ceblet dialogue will appear prompting the user for additional*

*information. For example, tree saving functions will request if branch lengths should be stored with the file.*

*An alternative to the Export button is the Export menu option in the File menu at the top of the PEBBLE main window (Fig. 6.8.1).*

*To edit objects*

1b. Select the object to edit from the object store.

*If the selected object cannot be edited, the Edit button will not be enabled.*

2b. Click the Edit button.

*An appropriate ceblet for editing the selected object will appear. For example, if the user has selected a sequence alignment file, clicking Edit will show the Alignment Editor ceblet, editing the selected Alignment object. In the Result Annotation phase of an editing ceblet, the option to overwrite the input will be available. Otherwise, editing can be used to produce an new edited object based on, but not replacing, the original.*

*To save data using PEBBLE's internal format*

1c. Select the object or objects to save from the object store.

*As with exporting data, to select more than one item, hold down the Shift key while selecting objects with the mouse. Any object in the object store, and any combination of objects, may be saved using the internal format of PEBBLE. This is useful for moving large amounts of data between computers, but the user is warned that the internal format is not compatible with any other phylogenetic applications, nor is it guaranteed to be compatible with future versions of PEBBLE.*

*The internal format will save object annotation as it appears in the object store.*

*Future versions of PEBBLE may utilize XML as the primary format, bypassing some of the shortcomings of the current format.*

2c. Select Save Results from the File menu.

*A File Requestor will appear as for exporting data, but no further interaction will be required.*

*To alter an object's annotation*

1d. Select the object to reannotate from the object store.

*The annotation of only one object can be altered at a time.*

2d. Click the Details button.

*A dialogue will appear prompting the user to select the name and description of the object; the user may change either or both aspects of the object. The Cancel button can be used to leave the object's annotation unchanged. An object cannot be renamed with the same name as another object already in the object store.*

*An alternative to the Details button is the Edit Details menu option in the Edit menu.*

*To remove an object from the object store*

1e. Select the object or objects to remove from the object store.

*Again, to select more than one item, hold down the Shift key while selecting objects with the mouse. Removal only applies to the object store of the PEBBLE application, and saved copies of the removed objects will not be affected.*

2e. Click the Remove button.

*A window will appear prompting the user to confirm the decision to remove the selected objects. The user may click Cancel to keep the selected objects or Okay to remove the selected objects from the object store.*

*An alternative to the Remove button is the Remove from Store menu option in the Edit menu at the top of the PEBBLE main window (Fig. 6.8.1).*

## USING PEBBLE FOR SIMULATION

*ALTERNATE PROTOCOL*

This protocol describes the two components available in PEBBLE for simulation of phylogenetic data. The primary component is the serial coalescent simulator that is used for generating phylogenetic tree data based on population data, using the coalescent, and assuming either a constant population or an exponentially growing/declining population. The second simulation tool is the alignment simulator that can simulate an alignment across a given tree, assuming a particular substitution model.

The serial coalescent is described in Rodrigo and Felsenstein (1999) and in Drummond et al. (2002, 2003). Readers are directed to these papers for technical descriptions of the serial coalescent. A brief description is provided in the Commentary.

### *Using the serial coalescent simulator*

1. From the Simulation menu at the top of the PEBBLE main window (Fig. 6.8.1), select Serial Coalescent Simulator.

   *The user will be presented with a display similar to that in Figure 6.8.12.*

2. Enter sample information parameters in the Sample Information area (Fig. 6.8.12).

   *The sample information can be described in two different ways, the "basic" style and the "advanced" style. The basic style allows a fixed sample size with a fixed number of samples and a fixed time interval between samples. The advanced style allows explicit description of each sample, with independent size and time information.*

   *For the basic style, select Basic from the left side of the Sample Information area, and then fill in the appropriate text fields on the right side. Sample size and the number of samples must all be at least one, and the time between samples must be greater than zero.*

   *For the advanced style select Advanced from the left side of the Sample Information area. The user will be presented with a single text area, into which a list of number pairs should be supplied. The first value in a pair is the sample size and the second value is the sample time. For example,* `(10,0), (5,1), (10,3)` *would indicate a sample of size 10 at*

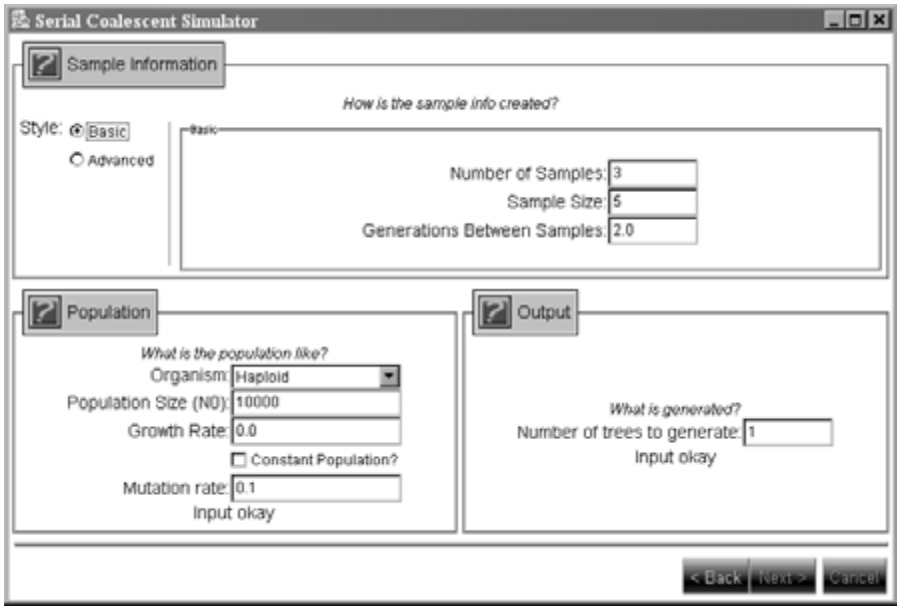


**Figure 6.8.12** The Serial Coalescent Simulator ceblet is depicted in this figure. With the options as shown, the result will be one tree with 15 taxonomic units.

*time 0, a sample of size 5 at time 1, and a sample of size 10 at time 3. The format is quite general and non-numerical components may be omitted as long as there is white space between each number. Therefore, the previously given example may be written* `10 0 5 1 10 3`.

3. Set the population parameters in the Population area. Also set the mutation rate in the Mutation Rate text field (the mutation rate is interpreted as the expected number of substitutions between each generation, and is used to scale the resulting trees).

   *For a constant population, select the Constant Population checkbox; otherwise select the growth rate in the Growth Rate text field. The mutation rate will also need to be set, with the units being the expected substitutions per site per generation. The population size must be at least 1 (although 1 is not, in general, a sensible value), and, in the case of exponential growth, is the size of the population at the most recent time point. The user must also select if the population is haploid or diploid, using the related drop-down menu.*

4. Set the number of trees to generate in the Output area, then click Next.

   *The serial coalescent generator can be set to generate any number of trees at one time. The user must request at least one tree.*

5. Annotate the results.

   *If only one tree was requested, the user will be able to name that one tree, as usual. If more than one tree was requested, the user will only be able to name the group of trees, not individual trees. This group will appear in the object store as a single object that may be expanded (by clicking on the + symbol) to view individual trees.*

***Simulating alignment data***

6. From the Simulation menu, select the Alignment Simulator ceblet.

   *The Alignment Simulator ceblet requires two inputs: a set of phylogenetic trees and a substitution model. To select more than one tree from the object store, the Shift key is held down while selecting Tree objects. The substitution model determines how to evolve sequences over time.*

   *Clicking Next to continue will present the user with a display similar to that shown in Figure 6.8.13.*

7. Set the sequence generation parameters in the Root Sequence area (Fig. 6.8.13).

   *The user has two options. The first is to supply the ancestral sequence directly; the second is simply to specify the length of all sequences and let the computer generate the ancestral sequence.*

   *To provide an ancestral sequence, select the Provided radio button and enter the sequence into the Sequence text field. The sequence is entered using the character matching the residue type of the substitution model used. The sequence should not contain any gap characters or characters representing ambiguity. A list of acceptable characters will appear in the Acceptable Characters text field. The number of sites in the resulting alignment, or alignments, will match the number of sites in the provided sequence.*

   *To instruct the computer to generate a random ancestral sequence based on a specified number of sites, select the Generated radio button and enter the number of sites of the generated sequences in the Sequence Length text field. The sequence will be generated based on the equilibrium frequencies of the substitution model.*

8. If required, select the mutation rate in the Mutation Rate text field.

   *If some or all of the input trees are not represented using expected substitutions per site as units, the user will be required to enter a mutation rate in order to scale the offending trees.*

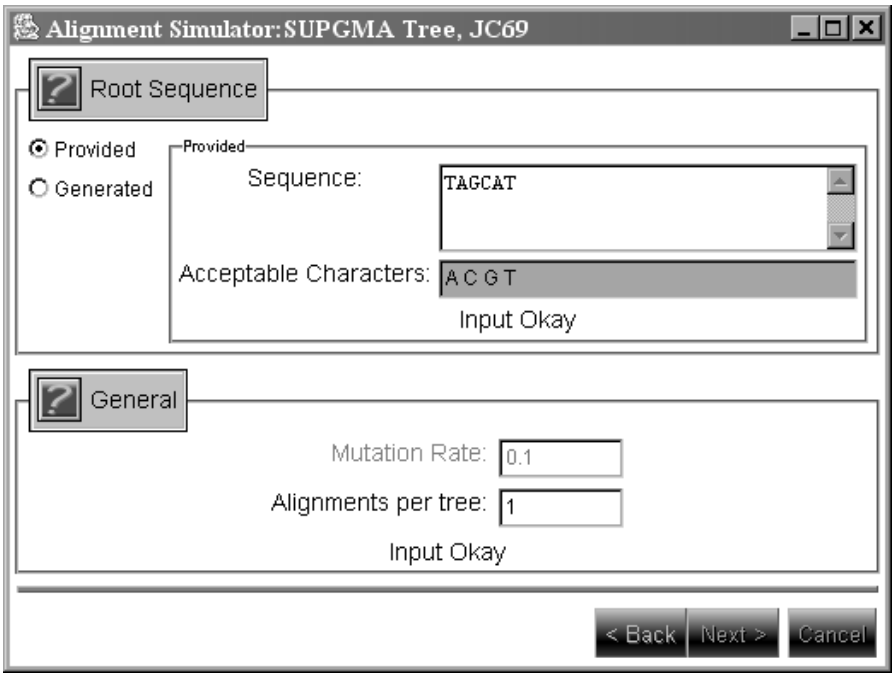


**Figure 6.8.13** The Alignment Simulator ceblet allows the user to simulate an alignment across one or more input trees. In this particular use of this ceblet, the root sequence has been selected as TAGCAT.

## GUIDELINES FOR UNDERSTANDING RESULTS

Please refer to steps 19 through 38 of the Basic Protocol, including the annotations, for a discussion of interpreting the results.

## COMMENTARY

### Background Information

A very readable exposition of the analysis of serially sampled sequences is given by Drummond et al. (2003). Their paper describes the most recent methods used for these analyses and outlines the types of applications for which they are appropriate. Here, a very brief description is given of the approach taken by the authors of this unit; this description is meant to complement the two methods discussed above, sUPGMA and maximum likelihood estimation of substitution rates.

Suppose that a population of genes is sampled several times over the course of a study period, and at each sampling occasion a number of sequences are obtained. If these sequences have accumulated substitutions at a clock-like rate, sequences from the same time point will terminate at identical times. One method for reconstructing phylogenies when sequence evolution is clock-like is UPGMA (Unweighted Pair-Group Method using Arithmetic Averages; Sneath and Sokal, 1973). However, in a UPGMA tree, all tips terminate at the same time (i.e., the tree is ultrametric). Recently there has been an increased interest in the analysis of serial sequence samples that are gathered from the same population with each sample obtained at a different time. These includes samples from rapidly evolving viral populations such as HIV (Leitner and Albert, 1999; Forsberg et al., 2001), as well as ancient DNA (Leonard et al. 2000; Barnes et al., 2002; Lambert et al., 2002). Several methods have been developed to estimate the value of some

evolutionary parameter when serially sampled sequences are available. Rodrigo et al. (1999), and later Fu (2001), developed methods to estimate the generation time of a population when serial samples are available. Maximum-likelihood and least-squares estimators of single or multiple substitution rates have also been developed (Rambaut, 2000; Drummond and Rodrigo, 2000; Drummond et al., 2001). Drummond and Rodrigo (2000) also described a method to reconstruct serial genealogies using serial sample UPGMA (sUPGMA). The serial UPGMA method consists of four distinct steps. This procedure has been described in Drummond and Rodrigo (2000), and is only briefly described here.

***a. Estimation of* δ*s***

This step simply involves estimating the expected number of substitutions per site that accumulate between sampling times. A number of others (Shankarappa et al., 1999; Fu, 2001) have indicated how this may be done for pairs of samples. The expected distance between a pair of sequences, one from a later timepoint and the other from an earlier timepoint is:

$$E\left[\text{dist}\left(S_{\text{early}}, S_{\text{late}}\right)\right] = E\left[\text{dist}\left(S^{\text{i}}_{\text{early}}, S^{\text{j}\neq\text{i}}_{\text{early}}\right)\right] + \delta_{\text{early}\rightarrow\text{late}}$$

**Equation 6.8.1**

where the first term on the right-hand side is simply the expected average pairwise distance between two sequences from the earlier timepoint. To obtain an estimate of δ, the average pairwise distance between early and late sequences calculated from the sample is substituted for the term on the left, the average pairwise distance between pairs of early sequences is substituted for the first term on the right, and the equation is then solved.

The problem becomes tricky when there are more than two time points, because now it becomes possible to calculate δs for every possible pair of sampling times. The problem with this approach is that, for any three timepoints A, B, and C (where C is earlier than B, which is earlier than A), it is possible that $\hat{\delta}_{CA} \neq \hat{\delta}_{CB} + \hat{\delta}_{BA}$ (where $\hat{\delta}$ signifies the estimated value), when this inequality is indefensible under any reasonable model. To overcome this problem, the authors have adopted a general regression approach to estimate δ, such that the condition $\hat{\delta}_{CA} = \hat{\delta}_{CB} + \hat{\delta}_{BA}$ holds with one constraint: any value of δ that has been estimated as a negative value in the regression is set to zero.

When estimating δs, it is only necessary to know the order in which the samples were drawn. However, if the actual sampling times are known, then an alternative approach is to estimate a single constant, ω, the number of substitutions per unit time, and multiply this by the time interval between two sampling occasions:

$$E\left[\text{dist}\left(S_{\text{early}}, S_{\text{late}}\right)\right] = E\left[\text{dist}\left(S^{\text{i}}_{\text{early}}, S^{\text{j}\neq\text{i}}_{\text{early}}\right)\right] + \omega\left(t_{\text{late}} - t_{\text{early}}\right)$$

**Equation 6.8.2**

Once again, ω is estimated using a regression procedure. Note that ω is not the mutation rate per generation, unless time is expressed in generation units. However, ω can be converted to the mutation rate (i.e., number of substitutions per site per generation) if the generation time is known. Alternatively, estimating ω makes it possible to estimate generation time if mutation rate is known.

***b. Correction of pairwise distances***

Each pairwise distance $d_{ij}$ in the distance matrix is transformed to a corrected distance, $c_{ij}$ as follows:

$$c_{ij} = d_{ij} + \delta_{t(i)\to 0} + \delta_{t(j)\to 0}$$

**Equation 6.8.3**

where *t(i)* and *t(j)* are the time points from which the *i*th and *j*th sequences are obtained, and $\delta_{t(i)\to 0}$ and $\delta_{t(j)\to 0}$ are the δs associated with the divergence between *t(i)* and *t(j)* and the most recent sampling occasion (labeled "0"). What this does, in effect, is extend the distances of sequences sampled earlier to a value that approximates the expected divergences of sequences obtained most recently.

***c. Cluster using UPGMA***

UPGMA or WPGMA (Weighted PGMA; Sneath and Sokal, 1973) is applied to the corrected distance matrix.

***d. Trim back branches***

Once the UPGMA tree has been constructed, for a given terminal lineage which extends to sequence *i*, $\delta_{t(i)\to 0}$ is subtracted from the branch length. The sUPGMA tree has the topology recovered by UPGMA (on corrected distances) with tips terminating in the appropriate order of sampling.

Maximum-likelihood estimation of substitution rates, given a topology, is also reasonably straightforward. Consider the case of sequences sampled serially from a single population for which there is exact information on sampling times and a known phylogeny. In a model where it is assumed that there is a uniform rate of substitution—i.e., the single rate with dated tips (SRDT) model (Rambaut, 2000), total branch lengths from the root of the tree to the tips are no longer required to be equal. Instead branch lengths are determined by the number and lengths (in time units) of sampling intervals that the branches traverse, and the substitution rate. The parameters of the tree are the substitution rate, ω, the vector of times, τ, corresponding to the dated tips, and the ($n - 1$ for a bifurcating tree) internal node heights ($h$) measured in units of substitutions (following Rambaut, 2000; note that the tip times may be measured either in generations or some calendar unit, and a simple rescaling allows one to move between the two). As described previously by Drummond et al. (2001), ω is only estimated within the interval bounded by the first and last samples. Specifically, no assumptions are made with regard to the rate between the earliest sampling time and the root of the tree. Setting $\omega = 0$ is equivalent to terminating all tips an equal distance from the root, as is done under a standard molecular clock model.

For a given phylogeny, $T$, for which only the topology is known, one may estimate the joint likelihood of ω and **H**, the vector of internal node heights on $T$, as the conditional probability of obtaining the sequence data, **S**, given ω, $T$, **H**, and τ, the vector of times, as well as the instantaneous substitution rate matrix, **M** (also assumed to be known):

$$L(\omega, \mathbf{H}) = \text{Prob}\left(\mathbf{S} \middle| \omega, T, \mathbf{H}, \tau, \mathbf{M}\right)$$

**Equation 6.8.4**

Since $T$, τ, and **M** are fixed, $L(\omega, \mathbf{H})$ = Prob(**S** | ω **H**) will be written without loss of generality. This likelihood is calculated in the standard manner (Felsenstein, 1981; Goldman, 1990; Rodriguez et al., 1990) for phylogenetic trees; the addition of ω and τ

enters the calculations as constraints on the branch tip positions. The MLEs of the rate, $\hat{\omega}$ and elements of the vector of node heights, $\hat{\mathbf{H}}$, are constrained to be greater than, or equal to, zero, and is chosen such that $L(\hat{\omega}, \hat{\mathbf{H}})$ is maximized. It is worth noting, at this point, that since the only parameter of interest is $\omega$, $\mathbf{H}$ is a nuisance parameter and is estimated only because it is necessary to do so.

Note that it is further possible to allow different substitution rates to be defined over the sampling intervals. This model, first introduced by Drummond et al. (2001), is called the Multiple Rates Dated Tips (MRDT) model. The MRDT model is flexible enough to allow different rates to be assigned to different intervals on the phylogeny such that the intervals corresponding to different rates do not coincide with the sampling intervals. The MRDT model can be implemented with both sUPGMA and maximum likelihood estimation.

## Critical Parameters and Troubleshooting

When interpreting parameters obtained using serially sampled sequences, it is important to be aware of the following issues.

### *a. Phylogenies versus genealogies*

Although both phylogenies and genealogies may be represented as trees, a genealogy is really a representation of history within a population, whereas a phylogeny may represent the evolutionary history of species or higher taxa. In terms of the point estimation of substitution rates, as described here, this distinction is not important. However, when the variance of these values (or their confidence intervals) is estimated, the distinction is particularly important. This is because for sUPGMA the confidence interval (CI) is estimated using parametric bootstrapping. This involves simulating a tree (and attached sequences) using the estimated parameters. The only way in which it is possible to simulate a tree is by using a model; the model used by the authors derives from the coalescent. The coalescent (Kingman, 1982) is a mathematical description of a genealogy drawn from a Fisher-Wright population. It is not a description of the branching process of phylogenetic trees in general (although it equivalent to the Yule generating process for species trees; see Steel and McKenzie, 2000). In particular, the coalescent specifies the distribution of times between coalescent events (i.e., the internodal distances in chronological units on a clock-constrained tree). Because the authors use a coalescent specifically defined for a constant sized population to estimate their CIs, they make the assumption that their sequences are drawn from a population and that their tree is a representation of a genealogy. The effective size of the population is obtained from the parameter $\Theta$ (theta), which is equal to $2N\mu$ for a haploid population and $4N\mu$ for a diploid population. Use of the coalescent also requires that the genes evolve neutrally. If these conditions are not met, then any estimate of the variance based on parametric bootstrapping will be incorrect. Note that this is not an issue with confidence interval estimation when likelihood is used to obtain the estimates of substitution rates. In this case, the CI is based on the curvature of the likelihood function in the neighborhood of the ML estimate, and is an asymptotic estimator of variance.

### *b. Negative branch lengths in sUPGMA trees*

It is possible to obtain negative branch lengths with sUPGMA trees because step "d" described above (see Background Information) involves trimming back the branches by an estimate of the $\delta$ value. Negative branch lengths are often a sign of nonadditivity. Trees are additive if distances between any pair of sequences can be expressed as the sum of the lengths of branches connecting the two sequences in the pair. With clock-constrained trees, negative branch lengths may also be a sign that the sequences are not accumulating substitutions at a constant rate. To deal with this, the simplest approach is to obtain the

maximum likelihood of a tree constrained to fit a molecular clock, and compare this with the likelihood of an unconstrained tree of the same topology (i.e., without a molecular clock but using the same model of evolution). This comparison can be performed as a likelihood ratio test (see Drummond et al., 2001 for a description of such a test). If the molecular clock is rejected, then sUPGMA (or any method that requires sequences to evolve in a clock-like manner) is inappropriate.

### *c. Time-stamped versus ordered samples*

Serial sequence samples typically have associated sampling dates, but sometimes only information about the order of sampling occasions is known. If only ordinal information on times is available, the types of parameters that can be estimated are restricted. In particular, it is not possible to estimate a single substitution rate over all intervals. In fact, it is only possible to estimate the expected number of substitutions that occur along the branches of each interval. These are composite variables equal to the product of the unknown substitution rate and the time between samples.

### *d. User-defined change points for rates*

In some instances, there is a prior reason to define a change point for substitution rate that does not coincide with any particular sampling occasion. This can done when time-stamped information is available. However, it is not possible to specify more than a single change point between any two consecutive sampling occasions.

## Literature Cited


Barnes, I., Matheus, P., Shapiro, B., Jensen, D., and Cooper, A. 2002. Dynamics of Pleistocene population extinctions in Beringian brown bears. *Science* 295:2267-2270.

Drummond, A. and Rodrigo, A.G. 2000. Reconstructing genealogies of serial samples under the assumption of a molecular clock using serial-sample UPGMA (sUPGMA). *Mol. Biol. Evol.* 17:1807-1815.

Drummond, A. and Strimmer, K. 2001. PAL: An object-oriented programming library for molecular evolution and phylogenetics. *Bioinformatics* 17:662-663.

Drummond, A., Forsberg, R., and Rodrigo, A.G. 2001. Estimating stepwise changes in substitution rates using serial samples. *Mol. Biol. Evol.* 18:1365-1371.

Drummond, A., Nicholls, G.K., Rodrigo, A.G., and Solomon, W. 2002. Estimating mutation parameters, population history and genealogy simultaneously from temporally spaced sequence data. *Genetics* 161:1307-1320.

Drummond, A.J., Pybus, O.G., Rambaut, A., Forsberg, R., and Rodrigo, A.G. 2003. Measurably evolving populations. *Trends Ecol. Evol.* 18:481-488.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.* 17:368-376.

Forsberg, R., Oleksiewicz, M.B., Petersen, A.M.K., Hein, J., Botner, A., and Storgaard, T. 2001. A molecular clock dates the common ancestor of European-type porcine reproductive and respiratory syndrome virus at more than 10 years before the emergence of disease. *Virology* 289:174-179.

Fu, Y.X. 2001. Estimating mutation rate and generation time from longitudinal samples of DNA sequences. *Mol. Biol. Evol.* 18:620-626.

Goldman, N. 1990. Maximum likelihood inferences of phylogenetic trees, with special reference to the Poisson process model of DNA substitutions and to parsimony analysis. *Syst. Zool.* 39:345-361.

Huelsenbeck, J.P., Hillis, D.M., and Jones, R. 1996. Parametric bootstrapping in molecular phylogenetics: Applications and performance. *In* Molecular Zoology: Advances, Strategies and Protocols (J. D. Ferraris and S. R. Palumbi, eds.) pp. 19-45. John Wiley & Sons, New York.

Jukes, T. and Cantor, C. 1969. Evolution of protein molecules. *In* Mammalian Protein Metabolism, Volume III (H. Munro, ed.) pp. 21-132. Academic Press, New York.

Kalbfleisch, J.G. 1985. Probability and Statistical Inference. Springer-Verlag, New York.

Kingman, J.F.C. 1982. The coalescent. *Stochastic Process. Appl.* 13:235-248.

Lambert, D.M., Ritchie, P.A., Millar, C.D., Holland, B., Drummond, A.J., and Baroni, C. 2002. Rates of evolution in ancient DNA from Adelie penguins. *Science* 295:2270-2273.

Leitner, T. and Albert, J. 1999. The molecular clock of HIV-1 unveiled through analysis of a known transmission history. *Proc. Natl. Acad. Sci. U.S.A.* 96:10752-10757.

Leonard, J.A, Wayne, R.K., and Cooper, A. 2000. Population genetics of Ice Age brown bears. *Proc. Natl. Acad. Sci. U.S.A.* 97:1651-1654.

Rambaut, A. 2000. Estimating the rate of molecular evolution: Incorporating non-contemporaneous sequences into maximum likelihood phylogenies. *Bioinformatics* 16:395-399.

Rambaut, A. and Bromham, L. 1998. Estimating divergence rates from molecular sequences. *Mol. Biol. Evol.* 15:442-448.

Rodrigo, A.G. and Felsenstein, J. 1999. Coalescent approaches to HIV population genetics. *In* The Evolution Of HIV (K.A. Crandall, ed.) pp. 223-271. Johns Hopkins University Press, Baltimore.

Rodrigo, A.G., Shpaer, E.G., Delwart, E.L., Iverson, A.K.N., Gallo, M.V., Brojatsch, J., Hirsch, M.S., Walker, B.D., and Mullins, J.I. 1999. Coalescent estimates of HIV-1 generation time in vivo. *Proc. Natl. Acad. Sci. U.S.A.* 96:2187-2191.

Rodriguez, F., Oliver, J.F., Marin, A., and Medina, J.R. 1990. The general stochastic model of nucleotide substitution. *J. Theor. Biol.* 142:485-501.

Shankarappa, R., Margolick, J.B., Gange, S.J., Rodrigo, A.G., Upchurch, D., Farzadegan, H., Gupta, P., Rinaldo, C.R., Learn, G.H., He, X., Huang, X.L., and Mullins, J.I. 1999. Consistent viral evolutionary dynamics associated with the progression of HIV-1 infection. *J. Virol.* 73:10489-10502.

Sneath, P.H.A. and Sokal, R.R. 1973. Numerical Taxonomy. W.H. Freeman, San Francisco.

Steel, M. and McKenzie, A. 2000. Properties of phylogenetic trees generated by Yule-type speciation models. *Math. Biosci.* 170:91-112.

Yang, Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *J. Mol. Evol.* 39:306-314.

## Internet Resources

http://www.cebl.auckland.ac.nz

*The PEBBLE application can be downloaded from the above URL by following the Software link. Notification of updates and milestone releases of the PEBBLE application can be obtained via E-mail by sending an E-mail to pebble_notice-subscribe@yahoogroups.com. Bug reports regarding the PEBBLE application may be sent to pebble-bugs@yahoogroups.com.*

http://www.cebl.auckland.ac.nz/pal-project

*The home page of the PAL project can be found at the above URL.*

---

Contributed by Matthew Goode and Allen G. Rodrigo
Bioinformatics Institute and The Allan Wilson Centre for Molecular Ecology and Evolution
University of Auckland
Auckland, New Zealand

# Phylogenomic Inference of Protein Molecular Function

One of the fundamental paradigms in computational biology is function prediction by homology. In this framework, a gene or protein is compared against other genes or proteins in a database, and, if a sequence can be detected whose similarity is statistically significant, the function of the unknown gene or protein is inferred based on the known (or presumed) function of the homolog.

Homology-based predictions are used to gain a first-order approximation of the molecular function of the proteins encoded in a genome, and to prioritize experimental investigation. While computationally efficient methods for pairwise sequence comparison—notably BLAST (Altschul et al., 1990)—have been developed to make this approach feasible in high-throughput, homology-based function prediction is not without its dangers. Systematic errors associated with this paradigm have become increasingly apparent (Bork and Koonin, 1998; Eisen, 1998; Galperin and Koonin, 1998). Biological processes such as gene duplication (Fitch, 1970), domain shuffling (Doolittle and Bork, 1993; Doolittle, 1995), and speciation (Galperin and Koonin, 1998; Gerlt and Babbitt, 2001) produce families of related genes whose gene products can have vastly different molecular functions. Finally, existing database errors can be propagated through function prediction by homology (Brenner, 1999; Devos and Valencia, 2001; Gilks et al., 2002).

This unit presents a workflow (Fig. 6.9.1) for phylogenomic inference of protein molecular function (Eisen, 1998; Sjölander, 2004) for a sequence of interest. The first step involves identification of homologs and construction of a multiple sequence alignment. For this task, the FlowerPower Web server is presented in Basic Protocol 1.

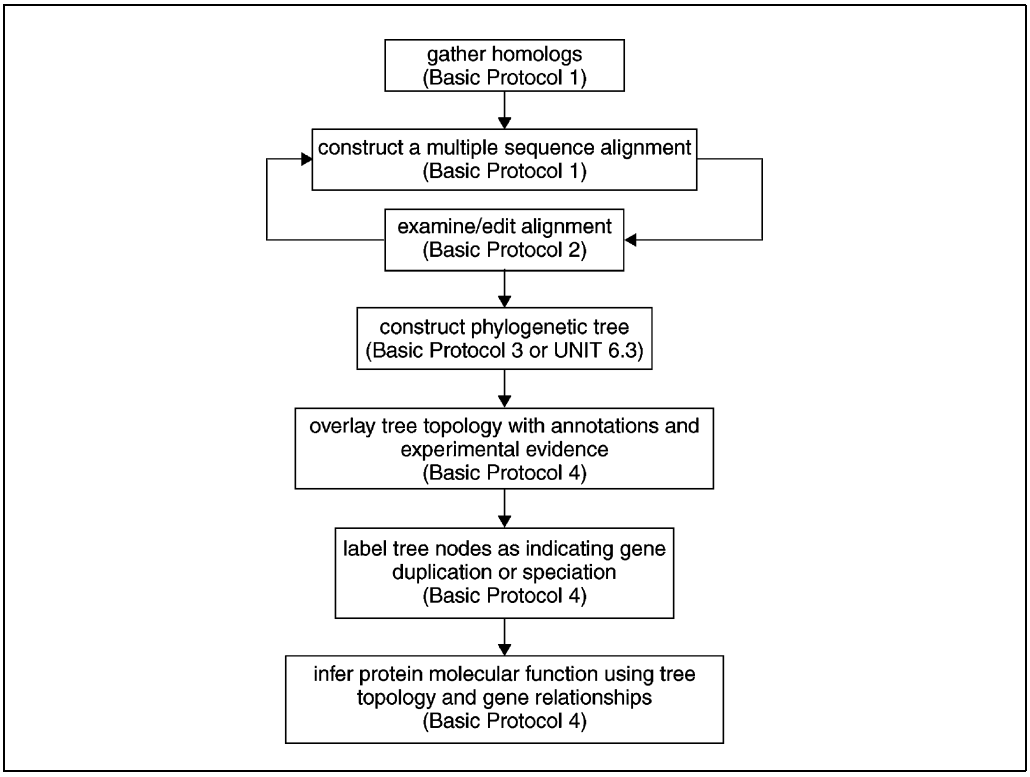


**Figure 6.9.1** Workflow for phylogenomic analysis. For every step in the workflow, the basic protocol in this unit that describes it is given in parentheses.

**Contributed by Nandini Krishnamurthy and Kimmen Sjölander**
*Current Protocols in Bioinformatics* (2005) 6.9.1-6.9.19

Next, the sequence alignment is analyzed and edited using the Belvu software (Basic Protocol 2). A phylogenetic tree can then be constructed using Neighbor Joining or the BETE software (Basic Protocol 3). The phylogenetic tree is displayed with annotations culled for the members of the family using the annotation software TreeNotator (Basic Protocol 4). Finally, the tree topology is analyzed to label branch points as indicative of either speciation or gene-duplication events, enabling the discrimination of orthologs from paralogs. Changes in biochemical function (and sometimes structure) can be traced along the evolutionary tree. For proteins whose functions are unknown, consistency of database annotations within subtrees containing the protein can be used as the basis for function prediction (a process termed "subtree neighbors"; Zmasek and Eddy, 2002).

The reliability of a phylogenomic inference of protein molecular function obviously depends on the accuracy of the phylogenetic tree, which depends in turn on the accuracy of the multiple sequence alignment and on a representative and thorough sampling of the protein family under analysis. Accuracy of annotation data is an additional critical dependency.

*BASIC PROTOCOL 1*

## IDENTIFYING HOMOLOGS AND CONSTRUCTING A MULTIPLE SEQUENCE ALIGNMENT USING FlowerPower AND MUSCLE

The first step in phylogenetic inference of protein function involves identifying homologs to a given protein and constructing a multiple sequence alignment as input to tree construction. The primary protocol for this stage, as detailed below, presents the use of the Berkeley Phylogenomics Group FlowerPower Web server for clustering and aligning protein sequence homologs to a user-supplied seed sequence. FlowerPower constructs a multiple sequence alignment during the homolog clustering process; users can download both the FlowerPower alignment and the MUSCLE (Edgar, 2004)

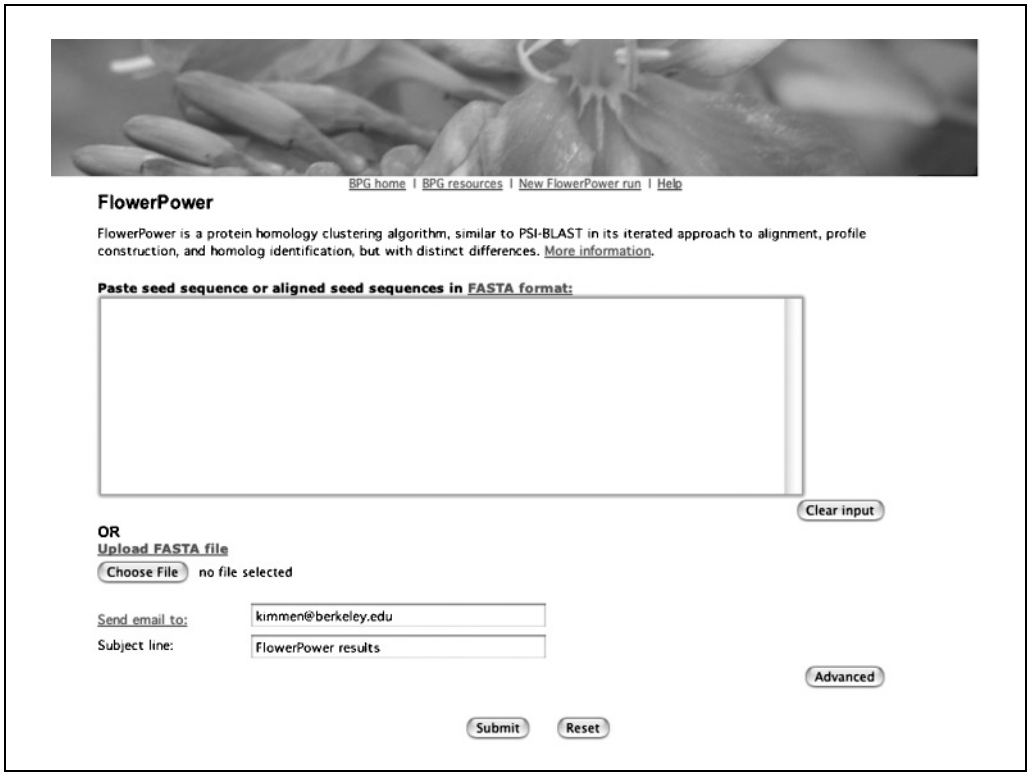


**Figure 6.9.2** FlowerPower submission page. Paste the seed sequence in the box provided, in FASTA format. Type a valid E-mail address in the box provided. Results will be sent to that address.

realignment of the same sequences, as well as the raw (unaligned) sequences. Users who prefer to use a different method for multiple alignment would naturally download the raw sequences.

***Necessary Resources***

*Hardware*

Any computer with an Internet connection

*Software*

Web browser

*Files*

*Protein sequence*: The sequence to be used as a seed for FlowerPower should be in FASTA format (see *APPENDIX 1B*). An example FASTA format file is also shown in Figure 6.9.4.

1. Point the browser at the FlowerPower Web server (*http://phylogenomics.berkeley.edu/flowerpower/*).
2. Paste the seed sequence in FASTA format into the box provided (see Fig. 6.9.2).
3. Enter a functioning E-mail address as directed. Results will be sent to this address.

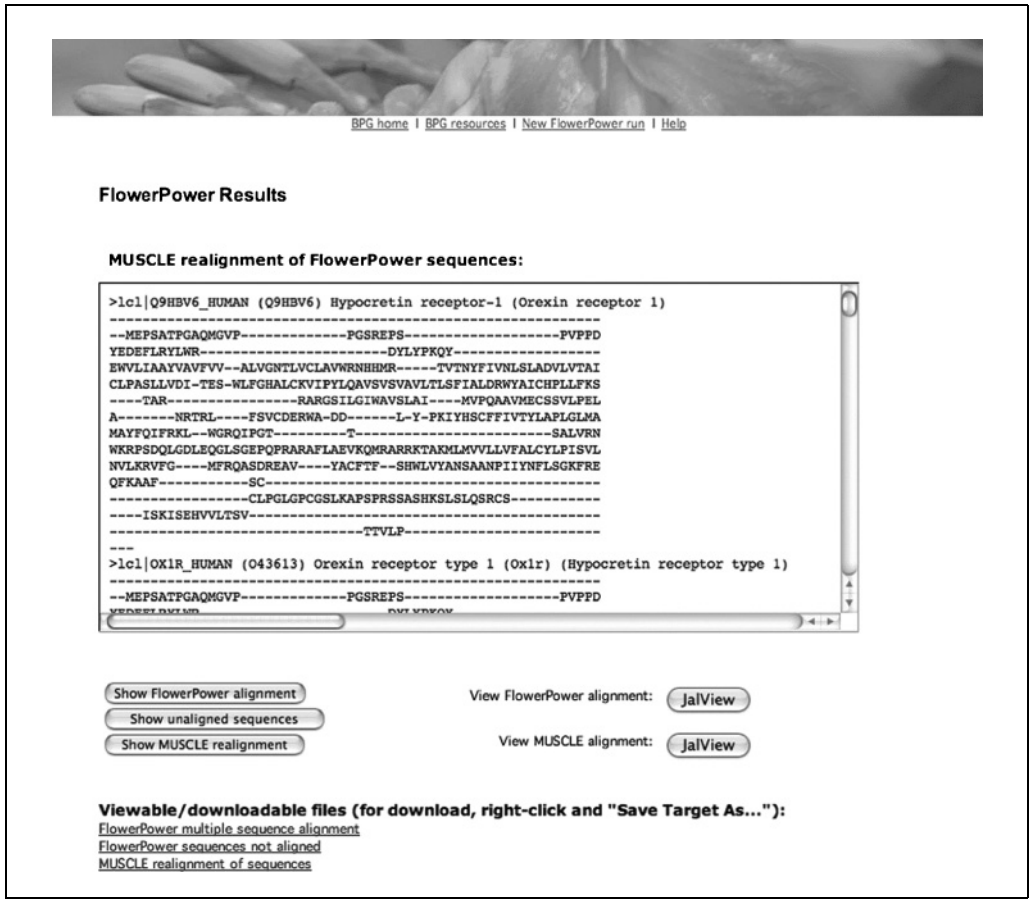


**Figure 6.9.3** FlowerPower results page. The FlowerPower and MUSCLE alignments can be downloaded by clicking on the hyperlinks at bottom left. The FlowerPower cluster (unaligned sequences) can be downloaded using the second hyperlink from the bottom. The MUSCLE and FlowerPower alignments can be viewed directly on the Web page by clicking on the JalView buttons at right.

4. Use the default parameters to gather global homologs to the seed from the UniProt database.

   *Users can override default parameters by clicking on the Advanced button. Advanced settings make it possible to select a different sequence database and to adjust the inclusion cutoff parameters (e.g., minimum length coverage and pairwise identity). Note that overriding default inclusion cutoff parameters can result in nonglobal homologs being included in the final result, producing inaccuracies in the phylogenomic analysis.*

5. Click Submit.

6. Retrieve results sent by E-mail.

   *The Web page includes two alignments of the sequences retrieved by FlowerPower (Fig. 6.9.3); the first alignment is constructed by FlowerPower and the second is a realignment of these sequences using the MUSCLE software. If a different multiple alignment program is desired, download the unaligned sequences to the local computer and construct the alignment separately.*

7. Download the MUSCLE alignment to the local computer.

***BASIC PROTOCOL 2***

## MULTIPLE SEQUENCE ALIGNMENT ANALYSIS AND EDITING USING Belvu

The accuracy of a phylogenetic tree is dependent on the accuracy of the input multiple sequence alignment (MSA). Alignment masking is often used to increase the phylogenetic signal in the MSA (see Commentary). In practice, masking is accomplished by editing the alignment to be used as the basis for phylogenetic inference. For this task, an alignment viewer/editor is critical. The authors' method for accomplishing this task employs the Belvu alignment editing software from the Karolinska Institute (Stockholm, Sweden) to remove selected columns from the alignment. Other alignment viewer/editors may provide the same functionality, but with a different interface.

### *Necessary Resources*

*Hardware*

Unix system with X Windows

*Software*

Alignment editor (e.g., Belvu; see the Support Protocol)

*Files*

*Protein sequence*: The multiple sequence alignment in aligned FASTA format (Fig. 6.9.4).

### *Open the multiple sequence alignment file in Belvu*

1. Download and install Belvu (see Support Protocol).

2. To use Belvu on a Unix/Linux machine, type `belvu <MSA>` (where `<MSA>` is the name of the multiple sequence alignment file) at the prompt. This will start up the program and make it possible to view and edit the alignment.

### *Edit the alignment*

The FlowerPower algorithm is designed to restrict clusters to sequences that are globally alignable. Steps 3 to 5 may therefore not be necessary when FlowerPower is used to gather homologs (but should still be confirmed). All manipulations explained below can

```
>gi|2494987|sp|P79292|OPRX_PIG/1-370
----MESLFPAPFWEVLYGSPLQGNLSLLSPNHSLLPPHLLLNASHG--------------------AFLPLGLK-VTIV
GLYLAVCVGGLLGNCLVMYVILRHTKMKTATNIYIFNLALADTAVLLTLPFQGTDVLLGFWPFGNALCKAVIAIDYYNMF
TSAFTLTAMSVDRYVAICHPIRALDVRTSSKAQAVNVAIWALASIVGVPVAIMGSAQVEDEE--IECLVEIPAPQDYWGP
VFA-VCIFLFSFVIPVLIISVCYSLMVRRLRGVRLLS-----------GSREKDRNLRRITRLVLVVVAVFVGCWTPVQV
FVLVQGLGVQP-GSETAVAVLRFCTALGYVNSCLNPILYAFLDENFKACFRKFCCAPT----------------------
------------RRREMQVSDRVRSIA-KDVALACKTSETVPRPA--------------------
>gi|730228|sp|P41144|OPRK_CAVPO/1-380
--MGRRRQGPAQPASELPARN----ACLLPNGSAWLPGWAEPDGNGS----------AGPQDEQLEPAHISPAIP-VIIT
AVYSVVFVVGLVGNSLVMFVIIRYTKMKTATNIYIFNLALADALVTTTMPFQSTVYLMNSWPFGDVLCKIVISIDYYNMF
TSIFTLTMMSVDRYIAVCHPVKALDFRTPLKAKIINICIWLLSSSVGISAIILGGTKVREDVDIIECSLQFPDDDYSWWD
LFMKICVFVFAFVIPVLIIIVCYTLMILRLKSVRLLS-----------GSREKDRNLRRITRLVLVVVAVFIICWTPIHI
FILVEALGSTS-HSTAALSSYYFCIALGYTNSSLNPILYAFLDENFKRCFRDFCFPIK----------------------
------------MRMERQSTSRVRNTV--QDPAYMRNVDGVNKPV--------------------
```

**Figure 6.9.4** Aligned FASTA format. The aligned FASTA format displays aligned residues in uppercase and gaps as dashes.

be done using the pull-down menu options at the top of the window, using right-click in Windows or Apple-click on the Macintosh.

3. *Remove any sequences appearing not to be globally alignable with the seed:* Ideally, one wants to exclude any sequence having significant inserts or deletions relative to the seed and to obtain a bidirectional overlap between the seed and included sequences of ≥70% (i.e., at least 70% of the amino acids in each database hit should align to corresponding residues in the seed, and vice versa). Identify sequences having large contiguous inserts (e.g., >50 amino acids) or deletions relative to the seed; these may represent structural domains conferring changes in function relative to the seed. Delete selected sequences using the Edit pull-down menu option "Remove highlighted line." Alternatively, one can select the "Remove many sequences" option and then double-click on each sequence to be removed from the alignment.

4. *Remove any sequences having very low pairwise identity with the seed:* To accomplish this using the Belvu viewer, first select the seed sequence by clicking on its identifier, then select the Sort pull-down menu option "Sort by identity to highlighted sequence." This will sort the sequences in the alignment, placing the seed (or sequences identical to the seed) at the top. A general rule of thumb in predicting homology based on amino acid identity is to require ≥30% identity with the seed over at least 80 amino acids, with lower levels of identity being permitted for longer sequences (Sander and Schneider, 1991). Experienced users may choose to retain more remotely related sequences with lower percent identities and alignment coverage (but caution is advised in these cases). Delete sequences using the Edit pull-down menu as described in step 1.

5. *Remove sequences with no close homologs in the alignment:* Choose the Edit pull-down menu option "Remove outliers." This option removes any sequence having less than a minimum percent identity to at least one other sequence in the MSA. The default minimum percent identity required to retain a sequence in the alignment is 20%, but can be reset by the user (the authors recommend using the default).

6. *Alignment masking—remove highly variable columns and columns containing a significant number of gap characters:* This step is designed to increase the signal-to-noise ratio in the alignment.

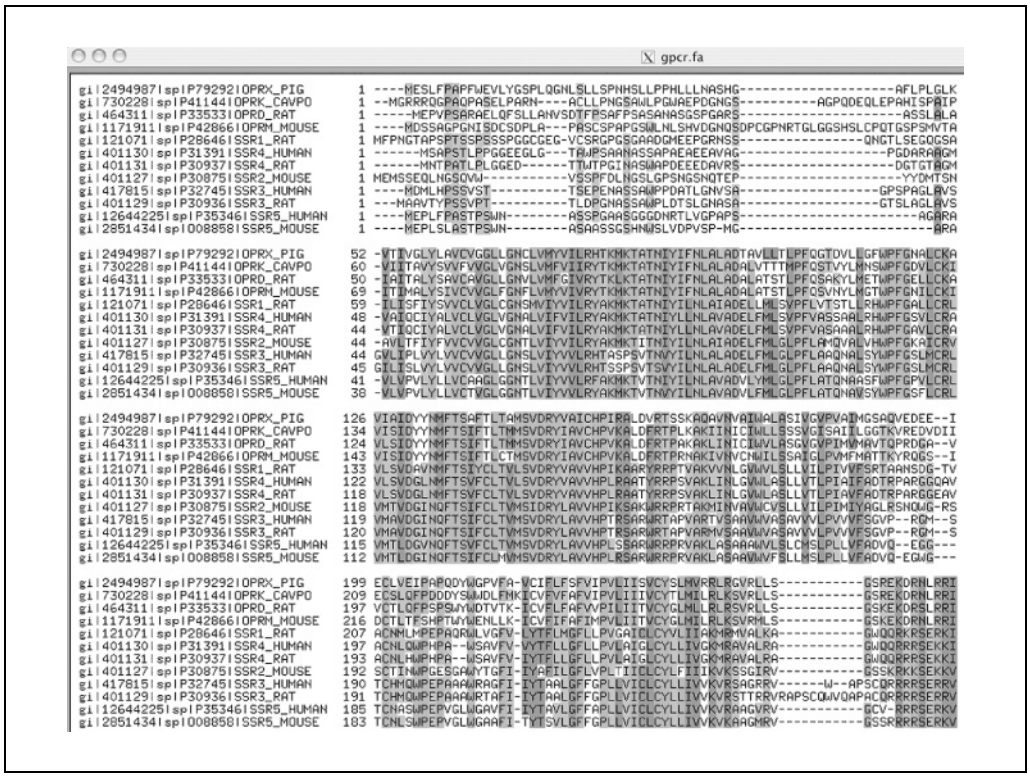


**Figure 6.9.5** Belvu display of an MSA. The alignment is displayed using the Belvu wrap-around mode option. This displays a region of variability at the amino-terminus, and another region (at bottom) where only one of the sequences aligns, causing the remaining sequences to have gap characters at these positions. The variable amino-terminus and gappy columns can be deleted during alignment masking, prior to use of the alignment as the basis for phylogenetic tree construction.

a. Remove variable N- and C-terminal regions (Fig. 6.9.5). Variable N- and C-termini can be deleted by selecting an amino acid in the alignment at the edge of a variable region and then using the Edit menu options "Remove columns to the left of cursor (inclusive)" and "Remove columns to the right of cursor (inclusive)."

b. Remove gappy columns. Columns with many gap characters can be removed using the Edit pull-down menu option "Remove gappy columns." The authors recommend setting the threshold to 80%, although lower values (down to 50%) can be used.

c. Remove regions with low sequence similarity. These columns can be identified by visual inspection of the alignment (see step 7 below for coloring scheme), and deleted using the Edit menu option "Remove columns" (setting the start and end indices manually). Alternatively, the user can use the Edit menu option "Remove columns according to conservation," setting the maximum conservation to 0.2 or less.

7. *Remove sequences not matching at family-defining motifs:* The default Belvu alignment coloring uses the Blosum62 scores to color columns; cyan for columns having high Blosum62 scores (i.e., very similar amino acids), dark blue for somewhat less conserved, gray for still less conserved, and no coloring for residues with poor Blosum62 scores (Fig. 6.9.5). Use this coloring to identify key motifs and delete sequences not matching at these motifs. Information from experimental investigation about key residues should be included at this stage.

8. *Save the alignment.* When finished, save the alignment using the File pull-down menu option "Save alignment as..." and select Aligned Fasta. Be sure to give the saved alignment a new name, so that the original alignment will not be overwritten.

## DOWNLOADING AND INSTALLING THE Belvu SOFTWARE

*SUPPORT PROTOCOL*

This protocol describes how to obtain and install the Belvu software used for multiple sequence alignment analysis and editing in Basic Protocol 2.

### *Necessary Resources*

*Hardware*

Unix system with X Windows

*Files*

Belvu executables for Linux/Unix platforms Sun, SGI, and Dec Alpha are available via anonymous FTP at *ftp.cgb.ki.se* in the directory `/pub/prog/belvu/`. Documentation for the Belvu software is available at *http://www.cgb.ki.se/cgb/groups/sonnhammer/Belvu.html*.

1. Go to *ftp://ftp.cgb.ki.se/pub/prog/belvu/*.
2. Download file `belvu.LIN_2.28`.
3. Rename file `belvu.LIN_2.28` to `belvu`.
4. Move the belvu file to a location on the local computer's path where executables are stored (e.g., `/usr/bin or /usr/local/bin`).
5. Make sure that the file is an executable by typing:

```
chmod 755 belvu
```

## CONSTRUCTING A PHYLOGENETIC TREE USING BETE

*BASIC PROTOCOL 3*

The authors' recommended approach for this task involves constructing a Neighbor-Joining tree (Saitou and Nei, 1987) and performing bootstrap analysis, as described in *UNIT 6.3*. As an alternative, the following protocol describes the use of the BETE Web server to construct a phylogenetic tree. It is assumed that a large number of proteins (>50) are included in the alignment, so computationally efficient methods for tree construction such as BETE (Sjölander, 1998) or Neighbor-Joining must be employed. For further confidence in an estimated phylogeny, the authors recommend the use of more than one tree method, followed by derivation of a consensus tree from the set of estimated phylogenetic trees (using the Consense software from the PHYLIP suite).

### *Necessary Resources*

*Hardware*

Any computer with an Internet connection

*Software*

Web browser

*Files*

*Multiple Sequence Alignment*: The alignment produced in Basic Protocol 2 can be used for tree construction. The alignment should be in aligned FASTA format or in SAM A2M format. Examples of alignments in aligned FASTA format and A2M format are presented in Figures 6.9.4 and 6.9.6 respectively.

```
>gi|1083836|pir||A55259
mgrrrqgpaqpas------------ELPARN.ACLLPNGSAWLPGWAEPDGNGSagpqdeqlepAHISPAIPVIITAVYS
VVFVVGLVGNSLVMFVIIRYTKMKTATNIYIFNLALADALVTTTMPFQSTVYLMNSWPFGDVLCKIVISIDYYNMFTSIF
TLTMMSVDRYIAVCHPVKALDFRTPLKAKIINICIWLLSSSVGISAIILGGTKVREDVdiIECSLQFPDDDYSWWDLFMk
ICVFVFAFVIPVLIIIVCYTLMILRLKSVRLLSGSREKDRNLRRITRLVLVVVAVFIICWTPIHIFILVEALGSTSHSTA
ALSSYYFCIALGYTNSSLNPILYAFLDENFKRCFRDFCFPIKMRMERQSTSRVRNTVQDPAYMRNVDGVNKPv
>gi|20379020|gb|AAM21070.1|
.............MESPIQIFRGEPGPTCAPsACLPPNSSAWFPGWAEPDSNGSagsedaqlepAHISPAIPVIITAVYS
VVFVVGLVGNSLVMFVIIRYTKMKTATNIYIFNLALADALVTTTMPFQSTVYLMNSWPFGDVLCKIVISIDYYNMFTSIF
TLTMMSVDRYIAVCHPVKALDFRTPLKAKIINICIWLLSSSVGISAIVLGGTKVREDVdvIECSLQFPDDDYSWWDLFMk
ICVFIFAFVIPVLIIIVCYTLMILRLKSVRLLSGSREKDRNLRRITRLVLVVVAVFVVCWTPIHIFILVEALGSTSHSTA
ALSSYYFCIALGYTNSSLNPILYAFLDENFKRCFRDFCFPLKMRMERQSTSRVRNTVQDPAYLRDIDGMNKPv
```

**Figure 6.9.6** UCSC A2M format. The A2M (for "align2model", i.e., the alignment of a sequence to a hidden Markov model) format is designed to indicate the states used by an HMM to generate a sequence. HMM states include Match states (representing the consensus structure for a family), Delete states (used to skip over a consensus position), and Insert states (used to generate additional amino acids not contained in the consensus). The A2M format displays aligned residues in uppercase and "unaligned" characters (emitted in HMM insert states) in lowercase. Gaps in the aligned regions are indicated as dashes (-) and are emitted in HMM skip/delete states. By contrast dots (.) are inserted post hoc in columns containing lowercase letters emitted in Insert state, so that all sequences have the same length.

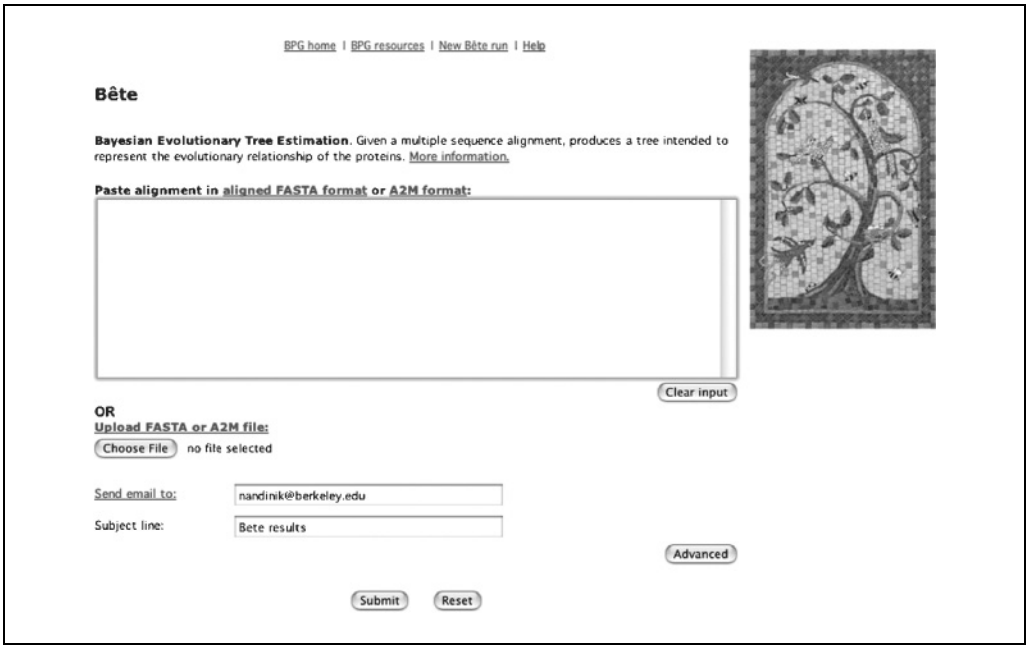


**Figure 6.9.7** BETE Web server submission page. The multiple sequence alignment is pasted into the box provided. Results are sent by E-mail.

1. Point the browser to *http://phylogenomics.berkeley.edu/bete* (Fig. 6.9.7).
2. Paste the multiple sequence alignment in the window provided or upload an alignment file (in FASTA or A2M format) by clicking the Browse button.
3. Enter a working E-mail address and click Submit.
4. Retrieve results sent by E-mail. The E-mail body will include a hyperlink for retrieval of results.
5. Download subfamily identification and tree files.

   *The results page (Fig. 6.9.8) contains the alignment file separated by subfamilies and a tree file in Newick format (UNIT 6.2), which can be viewed using any tree viewer, or annotated with the TreeNotator utility.*


**Figure 6.9.8** BETE results page. Download the Newick-format tree for annotation by the TreeNotator software. To view the BETE subfamily decomposition, download the "Alignment organized by subfamily" file.

## PHYLOGENOMIC INFERENCE OF MOLECULAR FUNCTION USING TreeNotator

***BASIC PROTOCOL 4***

Phylogenomic inference of protein function requires a cluster of related proteins with associated experimental and annotation data, along with a phylogenetic tree displaying their predicted evolutionary relationships. To facilitate the tree analysis and prediction of molecular function in a phylogenetic context, the use of the Berkeley Phylogenomics Group TreeNotator Web server is presented. TreeNotator enables phylogenetic tree annotation for proteins drawn from the GenBank and UniProt databases. Annotations are retrieved for sequences in the tree and used to label the leaves; the annotated tree is displayed online using the Java-based ATV software (Zmasek and Eddy, 2001), and can also be downloaded for analysis and display using other tree viewer software tools. Function prediction of a sequence (or sequences) of interest can then be performed by integrating experimental data and annotations in an evolutionary context.

### *Necessary Resources*

*Hardware*

Any computer with an Internet connection

*Software*

Web browser

*Files*

Phylogenetic tree in Newick format (*UNIT 6.2*), generated in Basic Protocol 3. To enable annotation retrieval from sequence databases such as UniProt and GenBank, sequence identifiers must be in a prespecified Newick format. An example Newick file format is shown in Figure 6.9.9.

*NOTE:* This protocol assumes that sequences in the tree are drawn from either the UniProt database (Apweiler et al., 2004) or the Genbank (Benson et al., 2004) database. If other sequence databases are used as a source for sequences, TreeNotator will not be able to retrieve annotations. In these cases, the authors recommend the use of TreeView (*UNIT 6.2*) to display and print the tree(s), and manual annotation of the trees to predict molecular function.

### *Display the phylogenetic tree overlaid with annotations*

1. Point the browser at the TreeNotator site (*http://phylogenomics.berkeley.edu/treenotator*).

```
(((((((401124:100.0,401125:100.0):78.0,121071:100.0):100.0,((401131:100.0,1351119:100.0):100.0,401130:100.0):100.0):100.
0,((((401128:100.0,401129:100.0):100.0,417815:100.0):100.0,((730838:100.0,2851434:100.0):100.0,12644225:100.0):100.0):89
.0,(((401126:100.0,464813:100.0):85.0,464812:100.0):81.0,(401127:100.0,267008:100.0):100.0):100.0):96.0):100.0,(((((464311:
100.0,417418:100.0):94.0,32363499:100.0):100.0,2494985:100.0):100.0,(((2494986:100.0,6093615:100.0):71.0,(2851402:100.
0,20139232:100.0):100.0):74.0,(1171911:100.0,464314:100.0):100.0):100.0):82.0,(((464313:100.0,464312:100.0):99.0,730229:
100.0):87.0,730228:100.0):100.0):88.0):100.0,((548427:100.0,548426:100.0):94.0,1352647:100.0):48.0):54.0,2494987:100.0):1
00.0,730230:100.0);
```

**Figure 6.9.9** Newick file format. The Newick format is a standard tree file format readable by most tree viewers. The format uses nested parentheses to indicate the join order of the nodes. In the above example, the sequences in the tree are represented by their GenBank identifier, and the bootstrap values are indicated at each node. For example `((401124:100.0,401125:100.0):78.0,121071:100.0)` means that the sequences 401124 and 401125 are joined at a node with bootstrap value of 78.0, and this node is joined to sequence 121071 with a bootstrap value of 100.

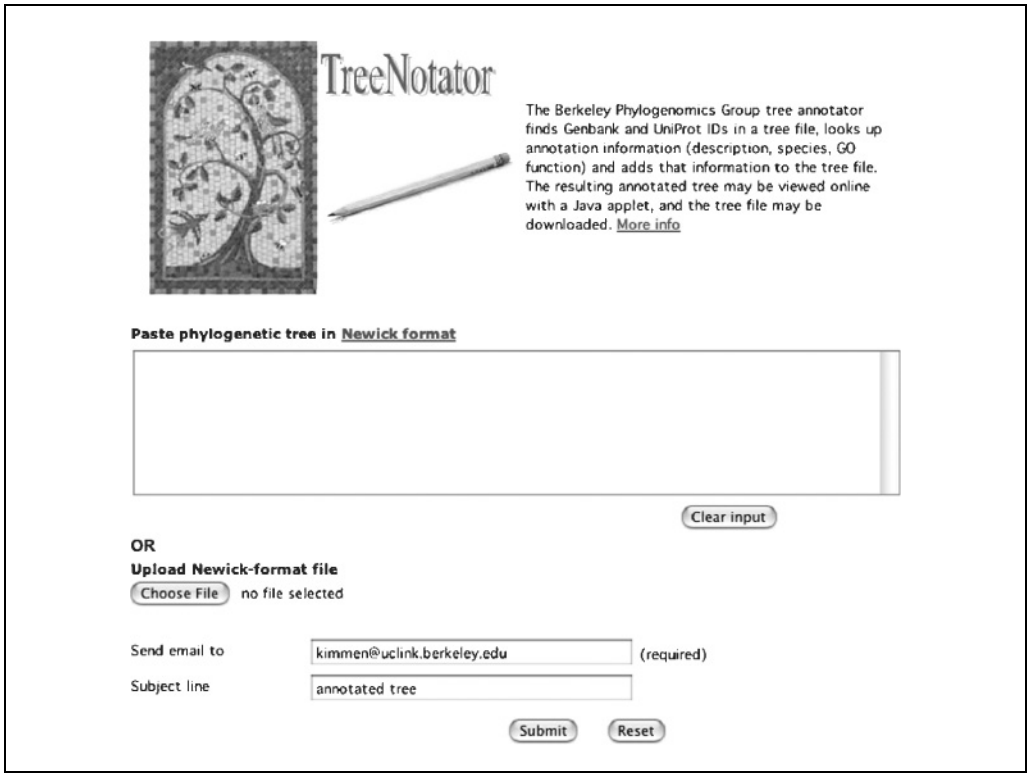


**Figure 6.9.10** TreeNotator submission page. The tree is pasted in the input box in Newick format. Results are returned by E-mail to the address provided.

2. Paste the tree file in Newick format in the box provided or upload a tree file by clicking the "Choose file" button (Fig. 6.9.10).
3. Enter a working E-mail address as directed. Results will be sent to this address.
4. Click Submit.
5. Retrieve results by clicking on the hyperlink provided in the E-mail. The TreeNotator results Web page (Fig. 6.9.11) provides a tool to view the tree with annotations using the ATV software. It is also possible to download annotated tree files for use with other tree-viewing software applications such as TreeView (*UNIT 6.2*).

### *The TreeNotator display*

6. Click on View Tree in the results page. This will open a window that displays the tree and sequence annotations using the ATV software (Fig. 6.9.12). It may be necessary

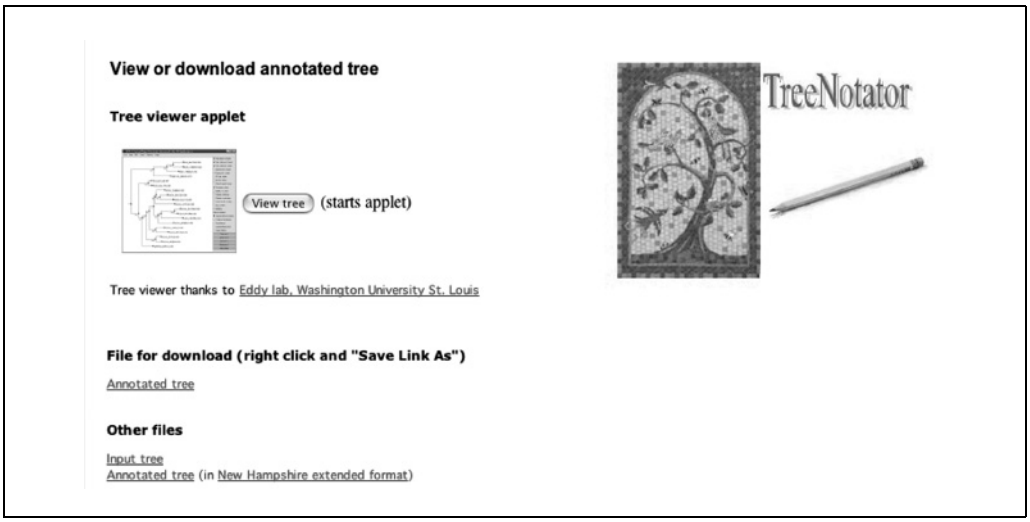


**Figure 6.9.11** TreeNotator results page. Click on the "View tree" button to display the annotated tree using the ATV software, or download the annotated tree (immediately under "File for download") for display using other phylogenetic tree visualization software.

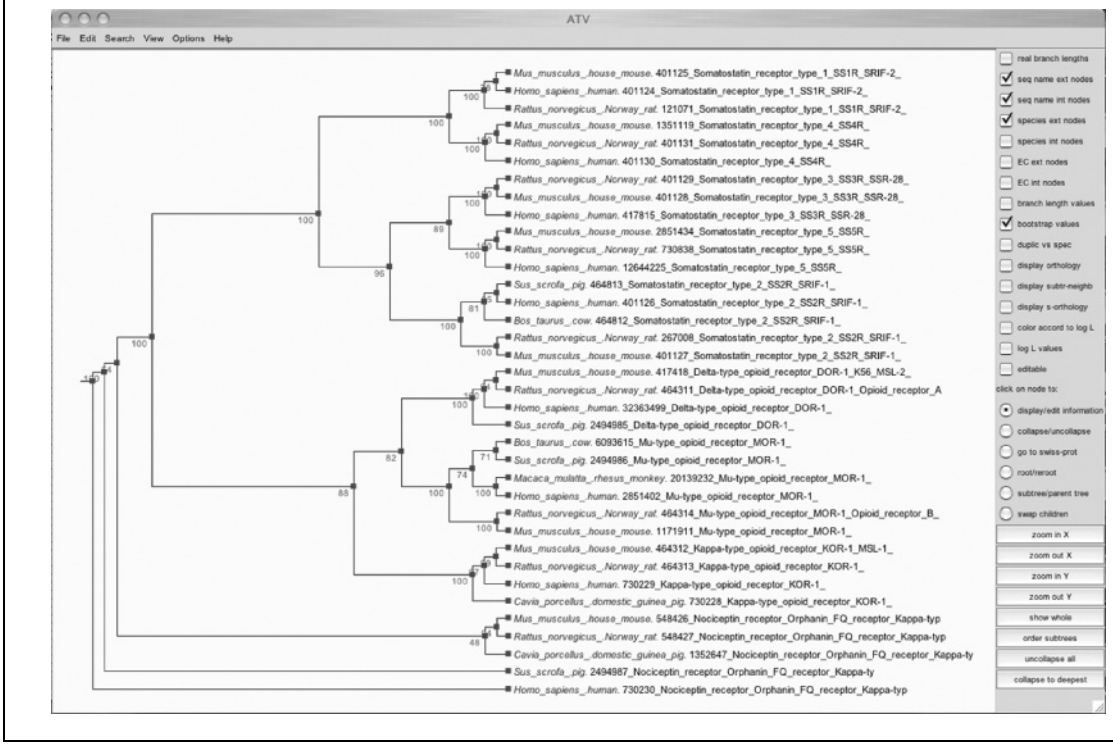


**Figure 6.9.12** Annotated tree displayed with ATV viewer. For each sequence, the following data are retrieved from the corresponding database (UniProt or GenBank) and displayed at the leaves: Species, GenBank, or UniProt idenitifier and the annotation. The bootstrap values are displayed in green at internal nodes.

to resize the window to display the tree topology more accurately, as the initial display tends to compress the tree.

7. Options for manipulating the tree display are provided in the panel on the right. Although there are several options, only those relevant for this unit are discussed. The user is encouraged to explore the other options.

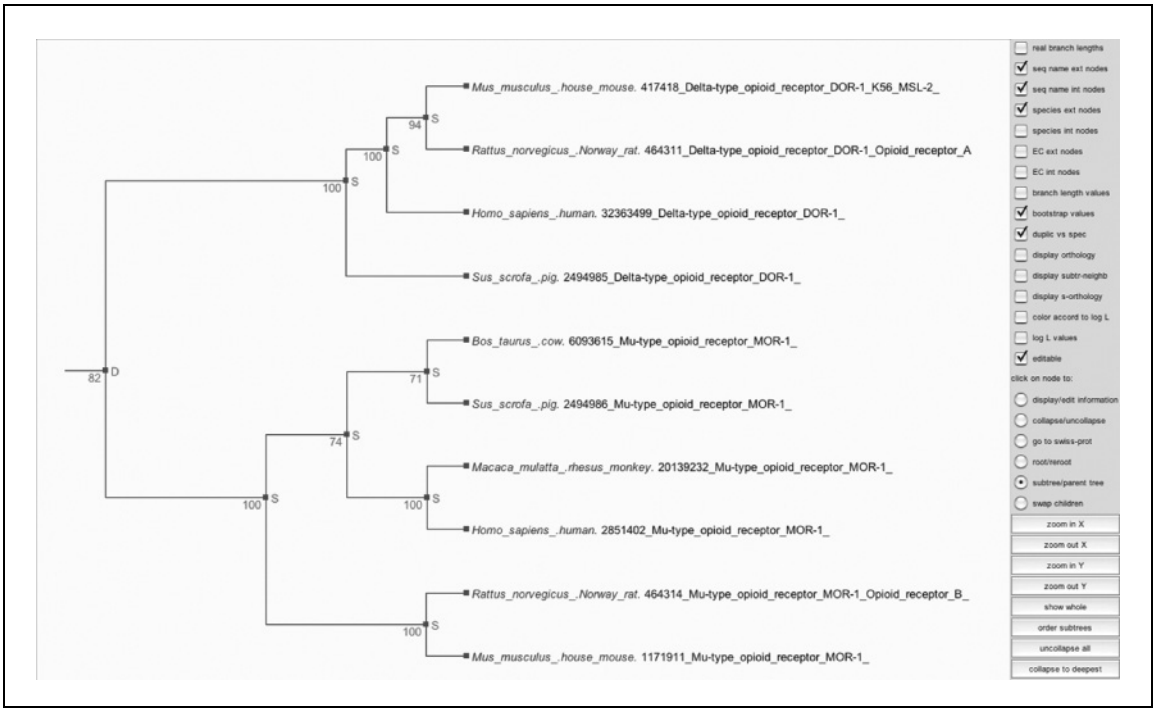


**Figure 6.9.13** In the tree shown above, the root is labeled D to indicate a duplication event, joining two subtrees, each of which contains a group of putative orthologs. All other internal nodes are labeled S, to indicate speciation events. Nodes for subtrees containing only one representative of a species can safely be labeled with an S. While labeling nodes as indicative of speciation is straightforward, labeling nodes as indicative of duplication is less so, since multiple protein sequences encoded by the same gene can be present in a dataset (e.g., due to splice variants or simple database duplicates).

8. All the following steps are to be performed using the options panel on the right.

   a. If a large number of sequences are used, the initial display may be compressed vertically, so that sequence identifiers on the leaves are not legible and subtree structure is not obvious. To view subtrees clearly, click on "zoom in Y." To show the full tree, click "show whole."

   b. To display the organism name, select "species ext nodes."

   c. To display the sequence ID and annotation, select "seq name ext nodes."

   d. To display bootstrap values, select "bootstrap values." To hide bootstrap values, deselect this box. Bootstrapping is a popular resampling method and bootstrap values indicate statistical support for subtrees in the MSA used as input. For further explanation of bootstrapping, see *UNIT 6.1*.

   e. To edit tree information, select "editable" and click on the node to edit. This will open a separate window with options for annotating the node with Name, Taxonomy ID, EC number, etc.

   f. To restrict the display to a subtree, select "subtree/parent tree" and click on the node of interest. This will display only the selected node and the subtree descending from that node.

***Label tree nodes as indicating gene duplication or speciation***

Label subtree nodes as indicating either speciation or duplication events, based on analysis of the tree topology (Fig. 6.9.13). To label nodes as indicating speciation or duplication events, select "editable," click on the node of interest, and, in the window that opens, select "duplication" or "speciation" or "not assigned," click Write to Tree, and close the

window. To display this on the tree, select "duplic vs spec." This will display duplication at a node with a "D" (in red) and speciation with an "S" (also shown in red).

9. To revert to the original tree and discard any changes to the tree display, click File in the menu bar on the upper left corner and select Reload.

***Identifying functionally consistent subtrees***

10. Identify three sets of "basis" subtrees:

    a. Inspect the phylogenetic tree to identify subtrees containing orthologous proteins. Nodes whose subtrees contain only single representatives from individual species can be labeled as indicative of speciation; the sequences in these subtrees are putative orthologs. Subtrees containing multiple (nonidentical) sequences from the same species should not be considered orthologous. (See Figs. 6.9.13 and 6.9.14).

    b. Inspect the phylogenetic tree to identify subtrees with consistent annotation. Many sequences may have uninformative annotations (e.g., hypothetical, unknown), while others may have simply been annotated by a hit to a PFAM HMM or some other domain analysis (e.g., a match to a Conserved Domain in the NCBI CDD, or to a COG). Attempt to identify any sequence with experimental support for the assigned function.

    c. If bootstrap values have been computed, identify subtrees with significant bootstrap support. A good rule of thumb for this is 70%.

11. The three "basis" subtrees identified in step 10 (groups a, b and c) should now be examined. Sequences in these "basis" subtrees can be assumed to share a similar function, with orthologous groups (group a) having the highest likelihood of functional similarity, followed by consistently labeled subtrees having high bootstrap support (the intersection of groups b and c).

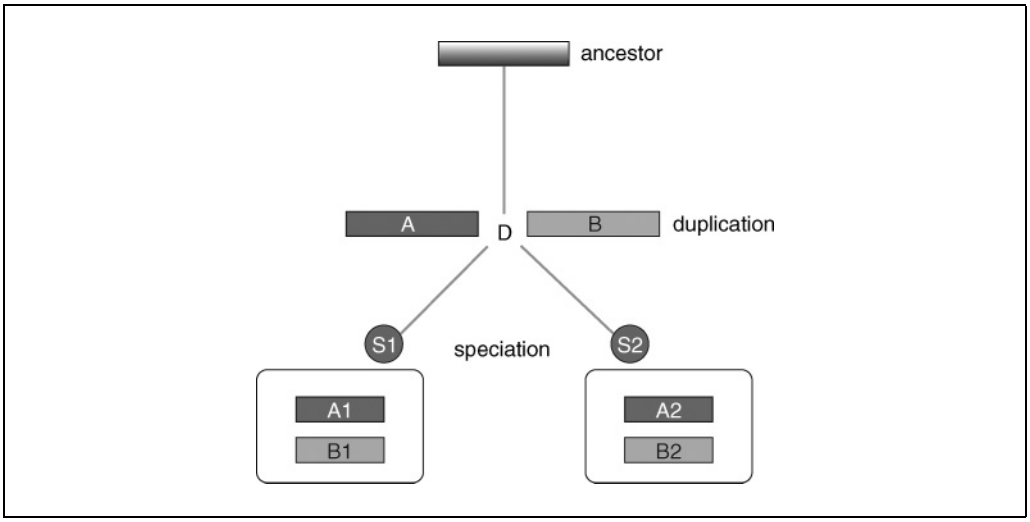


**Figure 6.9.14** Discriminating orthologs and paralogs. This figure represents the evolution of a protein family through the joint processes of gene duplication and speciation. The ancestral gene, shown at top, undergoes duplication in the ancestral genome (at the node labeled D), producing two paralogous genes, A and B. A subsequent speciation event produces two species, S1 and S2, each having a copy of the A and B genes. A1 and B1 are clearly paralogs, as are A2 and B2. If the definition of ortholog used is "the same gene in different species," then according to this tree, A1 and A2 are clearly orthologs, as are B1 and B2. By contrast, it cannot be asserted that A1 and B2 are orthologs, nor can it be asserted that B1 and A2 are orthologs. The A and B genes are related by a duplication event in the ancestral genome where they are paralogous. This figure is adapted from Koonin (2001).

### *Infer molecular function*

12. Label these basis subtrees with the molecular function assigned to sequences in the subtree (assuming credible experimental evidence supports the assigned function) using the "editable" feature in the ATV software as explained above (in step 8e, above). Sequences in these subtrees can then be assigned as potentially having the molecular function of their subtree neighbors.

## COMMENTARY

### Background Information

#### *Clustering homologs for phylogenomic inference*

Several factors are critical when clustering homologs for phylogenomic inference. The most important point concerns whether the phylogenomic inference is being performed on the whole-chain level (i.e., including all domains in all proteins gathered) or only along a single conserved domain found in proteins with otherwise divergent folds. This distinction is critical for accurate prediction of molecular function, due to the impact of domain fusion and fission processes in the evolution of protein families.

Domain shuffling in protein families requires special attention to avoid errors in prediction by homology. This is particularly problematic, as standard methods of homolog detection typically ignore whether two proteins align globally or only locally. For example, plant receptor-like kinases (RLKs) are composed of extracellular leucine-rich repeat (LRR) regions followed by a transmembrane domain and a cytoplasmic kinase domain. Receptor-like proteins (RLPs) are very similar to RLKs, but lack the cytoplasmic kinase domain. Polygalacturonase-inhibiting proteins (PGIPs) are similar to both RLPs and RLKs in containing an extracellular LRR region, but lack the transmembrane and cytoplasmic domains. PGIPs, RLPs, and RLKs have distinct molecular functions and overall architectures, but each has significant sequence similarity to the others along their LRR region, which can be hundreds of amino acids long.

The protocols presented here assume that a whole-chain analysis is being performed, and FlowerPower default parameters are designed to match this assumption. If one prefers to perform a phylogenomic analysis on the domain level, one can either adjust the FlowerPower options (using the Advanced settings page) or use another method, such as PSI-BLAST, to retrieve homologs.

Ideally, the set of proteins gathered should include all identifiable homologs sharing the same overall architecture. Practically speaking, this can be difficult. First, the overall architecture (i.e., sequence of structural domains) is known for only a small fraction of proteins; differentiating globally homologous proteins from those sharing only a local similarity is not always straightforward. Second, most bioinformatics methods for identifying proteins homologous to a query (or "seed") protein will identify all proteins sharing any significant *local* similarity to the query. This is true for the pairwise methods such as BLAST (Altschul et al., 1990), as well as the iterative profile methods such as PSI-BLAST (Altschul et al., 1997). This stage therefore requires the investigator to combine homolog identification and analysis in order to produce a final set of reliable homologs.

The FlowerPower algorithm is designed explicitly for phylogenomic analysis of protein molecular function. Like PSI-BLAST, FlowerPower employs an iterative approach to clustering, but instead of using a single HMM or profile to expand the cluster, it identifies subfamilies using the BETE algorithm and then selects and aligns new homologs using HMMs constructed for each subfamily (Brown et al., 2005). Subfamily HMMs compete for new sequences, enabling improved alignment quality (particularly in regions of structural variability across the family as a whole), and prevent drifting of the profile away from the seed sequence (profile drift). This is accomplished by the persistent representation of the seed sequence (and close homologs) by subfamily HMMs. FlowerPower employs alignment analysis following each iteration to reduce the intrusion of non-homologs.

#### *Multiple sequence alignment construction and analysis*

The accuracy of a phylogenetic tree depends on the accuracy of the multiple sequence alignment on which it is based. Studies have shown that multiple sequence alignment methods are sensitive to attributes characterizing many protein superfamilies—i.e., large numbers (in the hundreds and thousands) of

sequences, high sequence variability, and length differences (McClure et al., 1994; Thompson et al., 1999). To the degree that phylogenomic analysis is restricted to reasonable numbers of closely related sequences, alignment accuracy can be expected to be high, with corresponding increased likelihood of accuracy in the resulting tree topology. However, when large numbers of divergent sequences are included in a cluster, alignment and tree topology accuracy can be expected to decrease. The protocol presented here includes the use of the MUSCLE multiple sequence alignment program to align the proteins retrieved by FlowerPower.

Of the available multiple sequence alignment software tools, the MUSCLE algorithm is computationally efficient and has solid performance on a number of benchmark datasets (Edgar, 2004). Users who have elected to use a different program to align their sequences should download the unaligned sequences instead of the MUSCLE realignment, and then use the program of choice to construct the alignment.

### *Masking the multiple sequence alignment*

One of the fundamental assumptions of phylogenetic inference is positional homology, i.e., all the residues in a column of an MSA descend from a residue in the ancestral protein. However, protein sequence and structural variability can cause this assumption to be violated. Alignment *masking* is designed to maximize the phylogenetic signal in the multiple sequence alignment by deleting (or downweighting) columns corresponding to either misaligned or structurally divergent regions in the molecules. These high-structural-divergence regions are more often found at the amino- and carboxy-terminus of the proteins, but may also be found at the exposed surface of the proteins (in particular, in loop regions) between conserved blocks. Two basic approaches to alignment masking have been proposed. The first approach (presented here) involves deleting columns that appear unreliable or that contain large numbers of gap characters. The multiple sequence alignment created is analyzed to discriminate the conserved structural features of the family as a whole. This analysis is used to determine which regions of the alignment should be used for phylogenetic tree construction and to identify and remove potentially spurious database hits. Both alignment rows and columns can be edited independently to produce a final multiple sequence alignment containing only the conserved structural features of highly credible homologs. An alternative approach (Wheeler et al., 1995) involves the construction of a concatenated superalignment of several separate multiple alignments, varying parameters and using the concatenated alignment as input to phylogenetic tree construction.

### *Tree construction and analysis*

As noted earlier, the accuracy of a phylogenomic analysis of protein function depends on the accuracy of phylogenetic tree topology. Unfortunately, ensuring phylogenetic tree accuracy is far from straightforward. Detailed phylogenomic analyses of protein families using different tree-estimation tools have shown a lack of consistency across methods (see, for example, Citerne et al., 2003). The lack of robustness of phylogenetic tree estimation under different conditions is supported by simulation studies (Felsenstein, 1988; Hasegawa and Fujiwara, 1993; Kuhner and Felsenstein, 1994), which show that most phylogenetic inference methods are sensitive to conditions that are commonly observed in protein superfamilies, e.g., large numbers of sequences, high levels of divergence, lack of positional homology, and nonuniformity of conservation across character states (columns) in the alignment. Importantly, while some methods are more robust with respect to these issues than others, no method is robust under all circumstances.

The potential uncertainty in a phylogenetic tree topology can be identified in several ways. Bootstrap analysis is traditionally performed to identify subtrees with good statistical support. This analysis can be supplemented through the use of different phylogenetic tree-estimation software tools and/or different multiple sequence alignments as input. The different topologies produced by these different methods and alignments will support a different set of hypotheses about the relationships between proteins and their molecular functions. Because of the inherent inconsistencies between tree-construction methods and their dependencies on the input multiple sequence alignment, the authors of this unit recommend that biologists use a combination of different alignment and tree-construction methods, followed by the identification of a consensus tree topology across all the derived trees.

It is assumed that a large number (>50) of proteins are included in the alignment, so that computationally efficient methods for tree construction such as BETE (Sjölander, 1998) or Neighbor-Joining (Saitou and Nei, 1987)

must be employed. If fewer sequences are being analyzed, users can employ more computationally intensive approaches. Primary sources of phylogenetic tree construction software include the PHYLIP Web site (*http://evolution.genetics.washington.edu/phylip.html*), MrBayes (Huelsenbeck and Ronquist, 2001), and PAUP (Swofford, 2002; *UNIT 6.4*). Additional information on these programs can be found in Chapter 6 of this manual (Inferring Evolutionary Relationships).

This unit recommends the use of Neighbor-Joining (NJ) and bootstrap analysis to construct a phylogenetic tree and identify subtrees with statistical support. Neighbor-Joining is one of the most commonly used methods for phylogenetic tree construction. Its computational efficiency enables NJ to be used to estimate phylogenies for large datasets and for estimation of bootstrap support. The alternative to this is the use of the Bayesian Evolutionary Tree (BETE) algorithm (Basic Protocol 3). The main advantage of BETE for phylogenomic analysis lies in its automatic prediction of functional subfamilies. BETE estimates a phylogenetic tree and automatically identifies functional subfamilies (Sjölander, 1998) given an input multiple sequence alignment. BETE uses relative entropy between profiles constructed using Dirichlet mixture densities (Sjölander et al., 1996) to build a tree, and minimum encoding cost principles to cut the tree into subtrees to define the subfamily decomposition. BETE subfamilies often correspond to orthologous groups, but may instead contain ultra-paralogs from the same species, or, simply, very similar proteins (including both paralogs and orthologs).

***Using phylogenetic trees to predict function***

As described earlier, gene-duplication events produce families of related proteins having different functional specificities. Phylogenomic inference is designed to enable users to differentiate orthologs (related by speciation events) from paralogs (genes in the same species related by duplication events). Orthologs are expected to share a common function (although this is not always the case), while paralogs are expected to be somewhat more divergent in function.

Two new methods for differentiating orthologs and paralogs have been developed very recently: Resampled Inference of Orthologs (RIO; Zmasek and Eddy, 2002) and Orthostrapper (Storm and Sonnhammer, 2002). Both methods involve analysis of a phylogenetic tree to label nodes as indicative of either speciation or duplication events.

Complicating matters, orthology does not always suffice for correct inference of molecular function (Eisen and Wu, 2002; Zmasek and Eddy, 2002). Paralogs can occasionally have greater functional similarity than orthologs, particularly in cases where the gene duplications are recent, or in the case of subfunctionalization (e.g., paralogs may perform the same function but specialize for different tissue types). Similarly, orthologous genes in distantly related species might perform different functions. The authors look for consistent annotation within subtrees (an approach called *subtree neighbors*; Zmasek and Eddy, 2002). When available, relevant biological information should be incorporated along with annotations. The use of the term "ortholog" differs among investigators in the field; some use this term to refer to homologous genes in different species, whereas others restrict the term ortholog to indicate only those genes or proteins in different organisms that are clearly "the same gene in different species" and that can thus be assumed to share an identical molecular function (Koonin, 2001). See Figure 6.9.14 for an illustration of these issues.

In practice, it can be very difficult to unambiguously label subtree nodes as corresponding to speciation or duplication events. The presence of multiple copies of the same sequence in the database, as well as allelic or splice variants of the same gene, can be interpreted incorrectly as entirely different genes and would consequently appear to be paralogs in the tree. If it is not possible to remove any such sequences from the alignment used to estimate the tree, detailed scrutiny of the tree will be necessary to avoid errors. For this reason, it may be simpler to use a "subtree neighbors" approach when predicting molecular function. An automatic tool for identifying putative orthologs is the Orthostrapper program, available at *http://orthostrapper.cgb.ki.se/* (Hollich et al., 2002).

The accuracy of sequence annotations should be confirmed. A large fraction of sequence annotations are based on homology and may be suspect. The UniProt database includes information about the annotation source, using Evidence Codes (e.g., Traceable Author Statement (TAS), for annotations supported by scientific publications). The authors strongly recommend checking biological literature for descriptions of experiments

performed for the original assignment of molecular function, as not all experiments are equally well designed.

## Critical Parameters and Troubleshooting

### *Clustering and alignment*

The sequences included by FlowerPower are selected according to the user-specified parameter settings. The default settings (particularly for fractional coverage of the seed sequence and of database hits) have been determined empirically and are designed to maximize inclusion of homologous proteins with identical architectures while reducing the number of sequences with different architectures. Occasionally, these default parameters may be overly conservative, resulting in the retrieval of very few sequences. There are two ways to overcome this problem. To include additional homologs, one can either change the default database for homolog retrieval from, the smaller but higher-quality UniProt database to NR, or adjust the inclusion parameters to make them less restrictive (e.g., by reducing the query and hit coverage parameters). In the latter case, it will be necessary to pay special attention to the alignment-editing step, to ensure that any non-homologs are rejected from the alignment used as the basis for tree construction.

As an alternative to FlowerPower, biologists can use BLAST or PSI-BLAST to retrieve sequences and then manually remove sequences with significant length differences. A multiple sequence alignment for the remaining sequences can be derived using the alignment program of choice. This alignment will need to be examined to confirm that all sequences included in the cluster can be aligned along their entire lengths (i.e., that all sequences appear to have the same domain architecture and are globally alignable).

### *Alignment masking*

Protein superfamilies can have regions of structural divergence that introduce noise in a phylogenetic estimation. To improve the expected accuracy of the phylogenetic tree, these regions should be identified and removed (or masked) prior to phylogenetic tree estimation. A detailed discussion of the various masking methods used is beyond the scope of this unit. The authors recommend that users construct and compare several phylogenetic trees using different masking protocols. Recommended masking protocols include: (i) removing columns with many gaps (e.g., where $<30\%$ of sequences align) and (ii) removing columns with average pairwise Blosum62 scores below zero. Trees estimated using masked alignments as input should be compared with trees based on unmasked alignments.

### *Tree construction and analysis*

One of the seldom-noted issues in phylogenetic reconstruction involves the uncertainty of a phylogenetic analysis. If the same alignment is used as input to three different tree-estimation programs, three (or even more) distinct trees can be produced. Small changes to the alignment algorithm, or different methods for masking a multiple sequence alignment, can also produce unexpected changes in the tree topology.

For these reasons, it is strongly encouraged that users construct phylogenetic trees via two or more distinct methods, rather than rely on a single method. For even greater robustness in phylogenetic reconstruction, the authors of this unit recommend constructing several multiple alignments using different alignment methods and using each (following judicious alignment masking) as input to the phylogenetic tree methods selected. To avoid the inherent bias of the guide tree used by most progressive methods, it is recommended that the alignment methods selected include at least one iterative (i.e., nonprogressive) approach. The final set of trees can be used to derive a consensus tree using the PHYLIP Consense software (see *UNIT 6.3*).

The authors strongly encourage users to retrieve experimentally confirmed data on molecular function, interactions, biological processes, cellular localization, and other aspects of a protein's overall function and role. These data are often difficult to separate from annotations supplied entirely by homology, which are likely to contain errors.

Users of the PHYLIP software should take note of the following caution. Sequence IDs are often truncated by PHYLIP to the first 10 characters following the > sign in the FASTA sequence. Truncated IDs will cause errors in TreeNotator. To avoid these errors, the authors recommend that the tree file be manually edited to repair any truncated IDs, or that a different tree-reconstruction program be used.

## Acknowledgments

This work was supported in part by Grant #0238311 from the National Science Foundation and by Grant #R01 HG002769-01 from the National Institutes of Health.

## Literature Cited

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic local alignment search tool. *J. Mol. Biol.* 215:403-410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25:3389-3402.

Apweiler, R., Bairoch, A., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O'Donovan, C., Redaschi, N., and Yeh, L.S. 2004. UniProt: The Universal Protein knowledgebase. *Nucl. Acids Res.* 32:D115-D119.

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., and Wheeler, D.L. 2004. GenBank: Update. *Nucl. Acids Res.* 32:D23-D26.

Bork, P. and Koonin, E.V. 1998. Predicting functions from protein sequences—where are the bottlenecks? *Nature Genet.* 18:313-318.

Brenner, S.E. 1999. Errors in genome annotation. *Trends Genet.* 15:132-133.

Brown, D., Krishnamurthy, N., Dale, J.M., Christopher, W., and Sjölander, K. 2005. Subfamily HMMs in functional genomics. *Pac. Symp. Biocomput.* 322-333.

Citerne, H.L., Luo, D., Pennington, R.T., Coen, E., and Cronk, Q.C. 2003. A phylogenomic investigation of CYCLOIDEA-like TCP genes in the Leguminosae. *Plant Physiol.* 131:1042-1053.

Devos, D. and Valencia, A. 2001. Intrinsic errors in genome annotation. *Trends Genet.* 17:429-431.

Doolittle, R.F. 1995. The multiplicity of domains in proteins. *Annu. Rev. Biochem.* 64:287-314.

Doolittle, R.F. and Bork, P. 1993. Evolutionarily mobile modules in proteins. *Sci. Am.* 269:50-56.

Edgar, R.C. 2004. MUSCLE: A multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* 5:113.

Eisen, J.A. 1998. Phylogenomics: Improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Res.* 8:163-167.

Eisen, J.A. and Wu, M. 2002. Phylogenetic analysis and gene functional predictions: Phylogenomics in action. *Theor. Popul. Biol.* 61:481-487.

Felsenstein, J. 1988. Phylogenies from molecular sequences: Inference and reliability. *Annu. Rev. Genet.* 22:521-565.

Fitch, W.M. 1970. Distinguishing homologous from analogous proteins. *Syst. Zool.* 19:99-113.

Galperin, M.Y. and Koonin, E.V. 1998. Sources of systematic error in functional annotation of genomes: Domain rearrangement, non-orthologous gene displacement and operon disruption. *In Silico Biol.* 1:55-67.

Gerlt, J.A. and Babbitt, P.C. 2001. Divergent evolution of enzymatic function: Mechanistically diverse superfamilies and functionally distinct suprafamilies. *Annu. Rev. Biochem.* 70:209-246.

Gilks, W.R., Audit, B., De Angelis, D., Tsoka, S., and Ouzounis, C.A. 2002. Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics* 18:1641-1649.

Hasegawa, M. and Fujiwara, M. 1993. Relative efficiencies of the maximum likelihood, maximum parsimony, and neighbor-joining methods for estimating protein phylogeny. *Mol. Phylogenet. Evol.* 2:1-5.

Hollich, V., Storm, C.E., and Sonnhammer, E.L. 2002. OrthoGUI: Graphical presentation of Orthostrapper results. *Bioinformatics* 18:1272-1273.

Huelsenbeck, J.P. and Ronquist, F. 2001. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 17:754-755.

Koonin, E.V. 2001. An apology for orthologs—or brave new memes. *Genome Biol.* 2:COMMENT1005.

Kuhner, M.K. and Felsenstein, J. 1994. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol. Biol. Evol.* 11:459-468.

McClure, M.A., Vasi, T.K., and Fitch, W.M. 1994. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.* 11:571-592.

Sander, C. and Schneider, R. 1991. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins* 9:56-68.

Saitou, N. and Nei, M. 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* 4:406-425.

Sjölander, K. 1998. Phylogenetic inference in protein superfamilies: Analysis of SH2 domains. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* 6:165-174.

Sjölander, K. 2004. Phylogenomic inference of protein molecular function: Advances and challenges. *Bioinformatics* 20:170-179.

Sjölander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I.S., and Haussler, D. 1996. Dirichlet mixtures: A method for improved detection of weak but significant protein sequence homology. *Comput. Appl. Biosci.* 12:327-345.

Storm, C.E. and Sonnhammer, E.L. 2002. Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics* 18:92-99.

Swofford, D. 2002. PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4. Sinauer Associates, Sunderland, Mass.

Thompson, J.D., Plewniak, F., and Poch, O. 1999. A comprehensive comparison of multiple sequence alignment programs. *Nucl. Acids Res.* 27:2682-2690.

Wheeler, W.C., Gatesy, J., and DeSalle, R. 1995. Elision: A method for accommodating multiple molecular sequence alignments with alignment-ambiguous sites. *Mol. Phylogenet. Evol.* 4:1-9.

Zmasek, C.M. and Eddy, S.R. 2001. ATV: Display and manipulation of annotated phylogenetic trees. *Bioinformatics* 17:383-384.

Zmasek, C.M. and Eddy, S.R. 2002. RIO: Analyzing proteomes by automated phylogenomics using resampled inference of orthologs. *BMC Bioinformatics* 16:3(1):14.

## Key References

Bork and Koonin, 1998. See above.

*The authors of this paper identify common problems associated with function prediction by homology and present ways to avoid these errors.*

Eisen, 1998. See above.

*Jonathan Eisen's cogent presentation of the raison d'etre behind phylogenomic analysis for improving prediction of gene function.*

Sjölander, 2004. See above.

*A detailed view of the challenges in phylogenomic analysis, with a description of new methods for key tasks in a phylogenomic pipeline.*

## Internet Resources

http://phylogenomics.berkeley.edu/resources

*The BPG resources Web site includes a variety of user-friendly resources for phylogenomic inference of protein molecular function. A description of all the available tools can also be found on the Web site.*

Contributed by Nandini Krishnamurthy and Kimmen Sjölander
University of California
Berkeley, California

# Using OrthoCluster for the Detection of Synteny Blocks Among Multiple Genomes

**Ismael A. Vergara[1] and Nansheng Chen[1]**

[1]Department of Molecular Biology and Biochemistry, Simon Fraser University, Burnaby, British Columbia, Canada

## ABSTRACT

Synteny blocks are composed of two or more orthologous genes conserved among species, resulting from speciation from their last common ancestor. OrthoCluster (Zeng et al., 2008) is a fast and easy-to-use program for the identification of synteny blocks among multiple genomes. It allows users to identify synteny blocks that contain different types of mismatches, and to decide whether they require conservation of gene orientation and conservation of gene order within the blocks. OrthoCluster can also be used to find duplicated blocks within genomes. Although genes and their correspondence are usually used as input for OrthoCluster, in fact, OrthoCluster can be applied using any type of markers as input as long as their relationships can be established. OrthoClusterDB provides a Web interface for running OrthoCluster with user-defined datasets and parameters, as well as for browsing and downloading precomputed synteny blocks for different groups of genomes. *Curr. Protoc. Bioinform.* 27:6.10.1-6.10.18. © 2009 by John Wiley & Sons, Inc.

Keywords: OrthoCluster • OrthoClusterDB • InParanoid • MultiParanoid • genome painter • GBrowse • segmental duplication

## INTRODUCTION

A synteny block is defined as corresponding genomic segments in two or more related genomes that contain two or more orthologous genes. The presence of synteny blocks among multiple genomes suggests that they may be functionally important. Thus, identification of synteny blocks may provide clues regarding gene and regulatory element arrangements that are essential for normal biological processes and that may be disrupted in disease conditions. OrthoCluster (Zeng et al., 2008) is a gene-based program designed for the detection of synteny blocks among multiple genomes. Given an arbitrary number of genomes and the orthologous relationships existing among all genomes, OrthoCluster outputs all those synteny blocks that satisfy constraints set by the user to control block size, mismatches, and conservation of order and orientation of the genes conforming the block. Additionally, OrthoCluster can be used for the detection of duplicated blocks within one genome by providing as input the genome of interest and the paralogous relationships existing in that genome. In this work, we focus on the detection of synteny blocks among multiple genomes using the OrthoCluster stand-alone version and OrthoClusterDB (*http://genome.sfu.ca/orthoclusterdb/*; Ng et al., 2009), a Web interface for running OrthoCluster with user-defined parameters and datasets, as well as for visualizing precomputed synteny blocks for different species. To run OrthoCluster, the user needs to provide as input $n$ ($n \geq 2$) genome files and one correspondence file with their corresponding orthologous relationships. For the detection of synteny blocks involving only two genomes (pairwise comparison), tools like InParanoid (Remm et al., 2001) can be used to generate the correspondence file. For the detection of synteny blocks among three or more genomes, a correspondence file can be generated based on tools such as OrthoMCL (Li et al., 2003) or MultiParanoid (Alexeyenko et al., 2006).



*Current Protocols in Bioinformatics* 6.10.1-6.10.18, September 2009
Published online September 2009 in Wiley Interscience (www.interscience.wiley.com).
DOI: 10.1002/0471250953.bi0610s27

## FILES

OrthoCluster receives two types of input files: the genome file and the correspondence file.

### Genome File

The user should provide one genome file as input for each genome (species) analyzed. The structure of each line composing a genome file is:

```
GENE_ID CHROMOSOME START END STRAND
```

Fields are separated by spaces or tabs, and no header is necessary. The GENE_ID field has to be unique, and the STRAND field has to be either `1` or `-1`. For gene-based analysis, only one isoform per gene should be included (usually the longest isoform). It is not necessary to sort the genes within or between chromosomes. In this protocol, we refer to chromosomes for the location of genes, but OrthoCluster can be applied to contig-level genomes or genomes in any other assembly state.

An example of part of the genome file for *C. elegans* (`genome_ele.txt`) is shown in Figure 6.10.1.

```
Y74C9A.3        I       4221 10148 -1
74C9A.2         I       11641 16585 1
74C9A.4a        I       17911 26778 -1
Y74C9A.5        I       28280 32482 -1
Y74C9A.1        I       43733 44677 1
Y48G1C.12       I       47472 49416 1
Y48G1C.4        I       49921 54360 1
Y48G1C.5        I       55337 64021 -1
C01B12.4        II      6664 9233 1
C01B12.5        II      9808 11826 -1
C01B12.3        II      12986 15858 1
C01B12.2        II      19538 21842 1
C01B12.1        II      23347 24347 1
C01B12.8        II      24771 25381 1
C01B12.7        II      25517 26961 -1
F23F1.4         II      27611 28158 -1
F23F1.5         II      30116 31574 -1
F23F1.3         II      31825 33559 1
F23F1.2         II      34488 35150 1
F23F1.6         II      35301 37302 -1
F23F1.7         II      38605 39487 -1
.
.
.
```

**Figure 6.10.1** Part of the genome file for *C. elegans* (`genome_ele.txt`).

### Correspondence File

The correspondence file includes orthologous relationships among genes in all input genomes. One and only one correspondence file is required for each run, independent of the number of genomes being analyzed. The structure for each line is:

```
GENE_1 GENE_2 ... GENE_N
```

Different fields are separated by spaces or tabs, and no header is necessary. Genes located in the correspondence file are called in-map genes. Thus, by definition, in-map genes are genes with orthologous relationships in the dataset. In contrast, genes that are not included in the correspondence file are called out-map genes. By definition, then, out-map genes are genes without orthologous relationships in the dataset. Orthologous relationships for some genomes are available at public databases such as Ensembl (*http://www.ensembl.org/*; Flicek et al., 2008; *UNIT 1.15*) and InParanoid (*http://inparanoid.sbc.su.se/cgi-bin/index.cgi*; O'Brien et al., 2005). For genomes whose orthologous relationships are not available, users can generate correspondence files using programs such as InParanoid (Remm et al., 2001) for pairwise orthologous relationships and OrthoMCL (Li et al., 2003) for multiple orthologous relationships.

Part of the correspondence file between *C. elegans* and *C. briggsae* (`mapping_ele_bri.txt`) is shown in Figure 6.10.2.

Note that in the example (Fig. 6.10.2), `ZK617.1a` appears twice. This is because the gene `ZK617.1a` in *C. elegans* has two orthologs in *C. briggsae* (one-to-many orthologous relationship). OrthoCluster takes both one-to-one relationships and one-to-many relationships for identifying synteny blocks. Users need to ensure that orthologous relationships in the correspondence file are not duplicated. Otherwise, all instances of the same orthologous relationship will be reported within the same synteny block. No score associated with the orthologous relationships is required by OrthoCluster.

```
ZK617.1a        CBG06205
ZK617.1a        CBG06206
C09D1.1a        CBG12070
C41A3.1         CBG14683
K11C4.5         CBG19042
F29D11.1        CBG11882
F33D4.2a        CBG05938
.
.
.
```

**Figure 6.10.2** Part of the correspondence file between *C. elegans* and *C. briggsae* (`mapping_ele_bri.txt`).

## USING OrthoCluster VIA WEB INTERFACE WITH USER-DEFINED DATASETS AND PARAMETERS

***BASIC PROTOCOL 1***

OrthoClusterDB (*http://genome.sfu.ca/orthoclusterdb/*) provides Web access to the program OrthoCluster and to precomputed synteny blocks. It consists of two main pages: (1) the Run OrthoCluster page, which provides easy access to running OrthoCluster with user-defined datasets and parameters, and (2) the View Synteny page, which provides access to precomputed synteny blocks with predefined parameters. Users who have a large number of genomes to analyze are encouraged to use the stand-alone version of

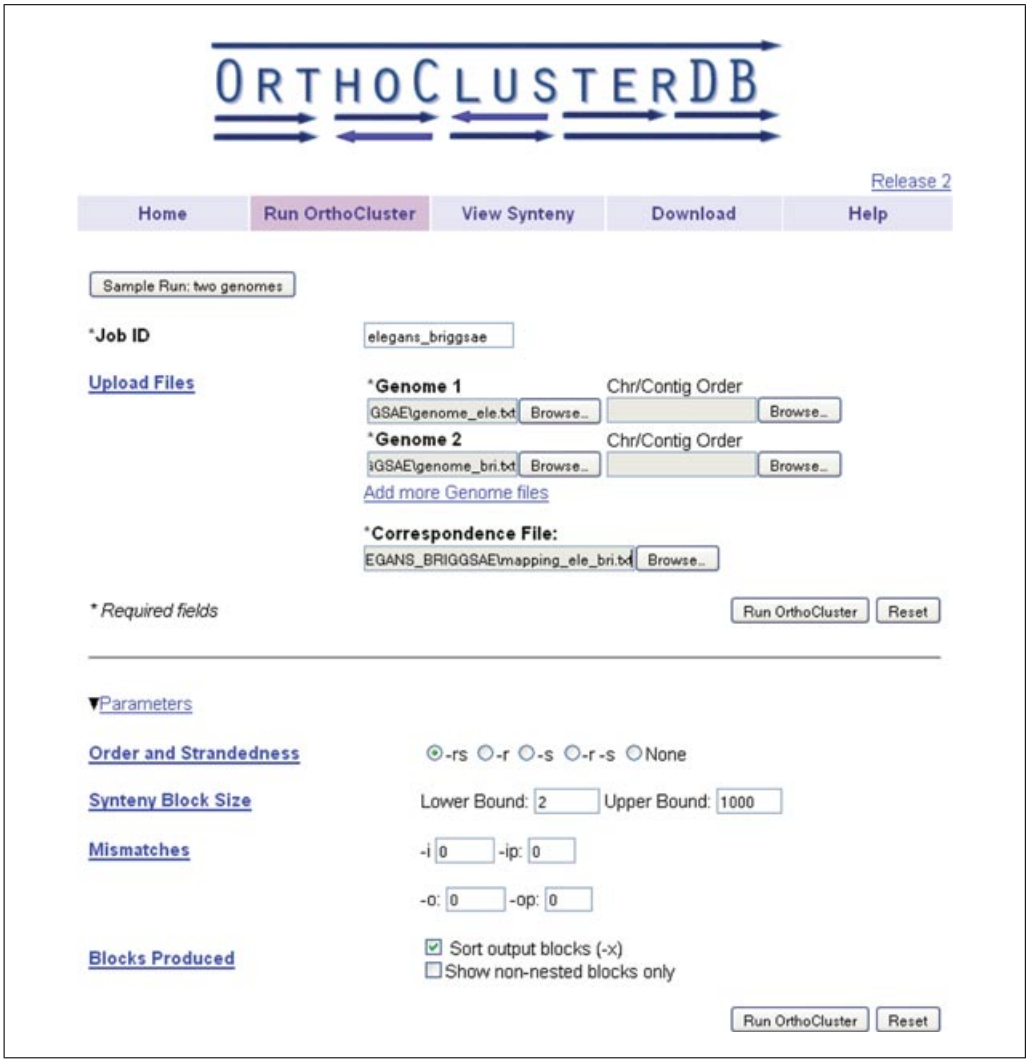


**Figure 6.10.3** User-modifiable parameters available in the Run OrthoCluster page of OrthoClusterDB. Parameters can be divided into those that handle order and strandedness, synteny block size, mismatches, and the output of blocks generated by OrthoCluster. The default values of the parameters are for detecting perfect synteny blocks.

OrthoCluster (see Basic Protocol 3 and Alternate Protocol). Two other pages are the Download page, which hosts all input files used to precompute the synteny blocks, and the Help page, which provides further information on the server functionality, input and output data formats, and other useful information.

This protocol illustrates how OrthoClusterDB is applied for identifying synteny blocks between two genomes—those of the nematode *Caenorhabditis elegans* and its sister species *Caenorhabditis briggsae*. The procedure for running OrthoClusterDB for identifying synteny blocks among multiple genomes is similar. Two types of input files are needed: the genome file and the correspondence file. Each input genome is represented in a separate genome file, whereas orthologous relationships are represented in a single file. For this example, we have prepared two genome files: `genome_ele.txt` and `genome_bri.txt`, representing the *C. elegans* and *C. briggsae* genomes, respectively. The correspondence file, `mapping_ele_bri.txt`, contains orthologous relationships between these two genomes.

Note that the order of input genomes for running OrthoCluster should be consistent with the order of columns of the correspondence file. Since we will use *C. elegans* as reference

**Table 6.10.1** Description of Parameters

| Parameter/flag | Description | Default value | Range[a] | Optional? |
|---|---|---|---|---|
| `-u` | The upper bound on the number of ortholog genes in each block | 1000 | [1-∞] | Yes |
| `-l` | The lower bound on the number of ortholog genes in each block | 1 | [1-u] | Yes |
| `-m` | Correspondence file name | NA | NA | No |
| `-gi` | Sequence file corresponding to the $i^{th}$ column in the correspondence file | NA | NA | No |
| `-i` | Maximal number of in-map mismatch genes | 0 | [0-∞ u) | Yes |
| `-o` | Maximal number of out-map mismatch genes | 0 | [0-∞) | Yes |
| `-ip` | Maximal percentage of in-map mismatch genes | 0 | [0-100) | Yes |
| `-op` | Maximal percentage of out-map mismatch genes | 0 | [0-100) | Yes |
| `-d` | Job id | NA | NA | No |
| `-r` | Finds order-preserving blocks | OFF | [ON,OFF] | Yes |
| `-s` | Finds strandedness-preserving blocks only | OFF | [ON,OFF] | Yes |
| `-rs` | Finds strandedness- and order-preserving blocks only | OFF | [ON,OFF] | Yes |
| `-f` | Finds non-overlapping blocks only | OFF | [ON,OFF] | Yes |
| `-x` | Sorts the blocks according to its size | OFF | [ON,OFF] | Yes |
| `-v` | Prints warning information in the STDOUT | OFF | [ON,OFF] | Yes |
| `-h` | Prints help information | OFF | [ON,OFF] | Yes |

[a]The infinity sign indicates that there is essentially no up limit to that parameter.

(it will be entered as the first input genome) and *C. briggsae* as target (the second input genome) in the correspondence file `mapping_ele_bri.txt`, the first column lists the *C. elegans* genes and the second column the orthologous *C. briggsae* genes. If the user wants to run OrthoCluster using *C. briggsae* as reference and *C. elegans* as target, then the file `mapping_bri_ele.txt` should be used as the correspondence file. Nevertheless, the orthologous relationships contained in both correspondence files are exactly the same. Users can download the genomes and correspondence files for *C. elegans* and *C. briggsae* at *http://genome.sfu.ca/cgi-bin/orthoclusterdb/download*. Additionally, genome files for five groups of species (*Pseudomonas*, *Plasmodium*, *Caenorhabditis*, *Drosophila*, and Mammals), as well as their correspondence files for each pair of genomes, are available from the OrthoClusterDB Web site.

### *Necessary Resources*

*Hardware*

Computer with Internet connection

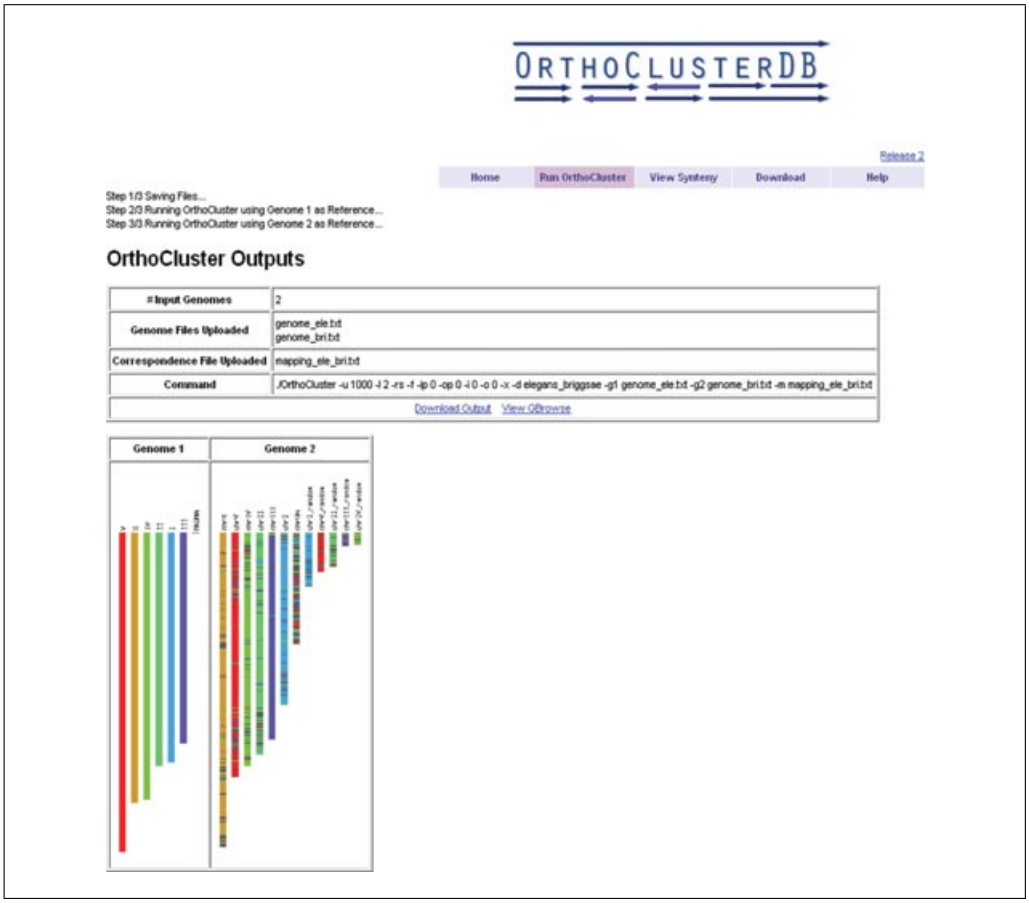


**Figure 6.10.4** Genome Painter image output after running OrthoCluster from the Run OrthoCluster page. The main content of this page is a summary table with information about the OrthoCluster run, and the Genome Painter image displaying the location of synteny blocks in the target genome(s) with respect to the reference genome. Additional information describing the steps done by OrthoCluster to generate the results is also available. In this example, the information displayed corresponds to the result of running OrthoCluster between *C. elegans* and *C. briggsae* for the detection of perfectly conserved synteny blocks.

*Software*

Internet browser, e.g., Internet Explorer (*http://www.microsoft.com/ie*), Firefox (*http://www.mozilla.org/firefox*), or Safari (*http://www.apple.com/safari*)

*Files*

The input files required in this protocol are described in the Files section at the beginning of the unit

1. From the OrthoClusterDB homepage (*http://genome.sfu.ca/orthoclusterdb/*), click on the Run OrthoCluster link at the top of the page.

2. On the Run OrthoCluster page which then appears (see Fig. 6.10.3), first enter a Job ID. If the Job ID includes spaces, these spaces will be substituted with underscores (we recommend that users provide Job IDs composed of alphanumeric characters). Next, upload the genome annotation files for each species (illustrated in Fig. 6.10.3) by clicking on the Browse buttons. The first genome entered (Genome 1, *C. elegans* in this example) is interpreted by OrthoCluster as the reference genome, while the rest of the genomes are regarded as the target genomes (*C. briggsae* in this example).

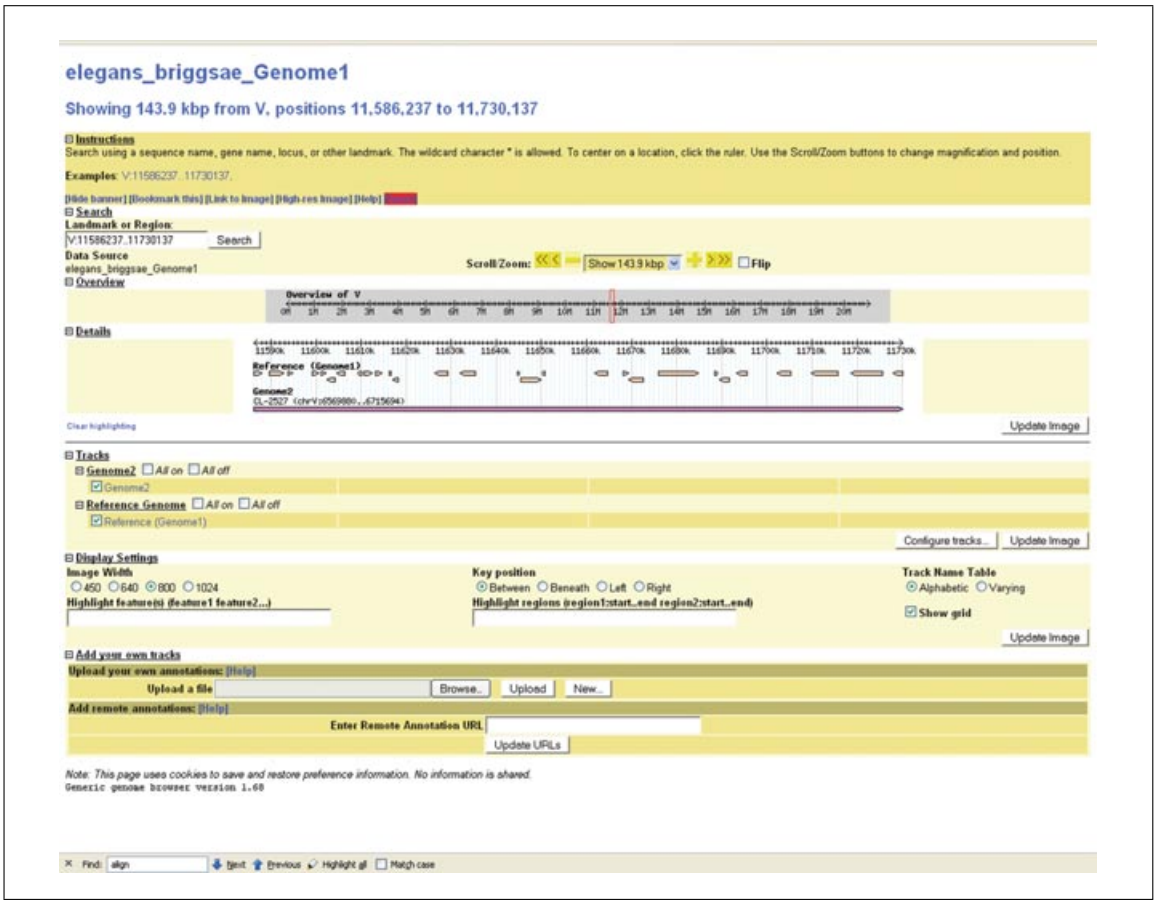


**Figure 6.10.5** GBrowse view of a synteny block. The region displayed corresponds to the largest perfect synteny block found between the genomes of *C. elegans* and *C. briggsae*.

3. *Optional:* For each genome, the user can upload an additional file specifying the order of the chromosomes. This file will be used to sort the chromosomes for display in the Genome Painter image (see step 6 below). Within this file, all chromosomes are listed on only one line, separated by spaces. Once an order file is provided, only chromosomes listed in that file will be displayed in the Genome Painter image.

4. Upload the correspondence file in the Correspondence File field (Fig. 6.10.3).

5. Click the Run OrthoCluster button to submit the job and run OrthoCluster using default settings.

   *The default parameters are specified in order to detect perfectly conserved synteny blocks, i.e., blocks of genes in which the order and strandedness of genes are conserved and no mismatches are allowed. The Run OrthoCluster page also allows users to run OrthoCluster using modified settings. It has a separate section, which is shown when users click on the Parameters link. Here, users can modify OrthoCluster parameters and flags before submitting the job (Fig. 6.10.3; Table 6.10.1). In general, we refer to an imperfect synteny block as one that is not generated by using the default parameters.*

6. OrthoCluster output is usually shown within 1 min, and includes a summary table and a genome-level view of detected synteny blocks (the Genome Painter; Fig. 6.10.4). In the Genome Painter, each chromosome of the reference genome is assigned a different color. Each chromosome in the target genome is displayed and colored according to the corresponding color of the synteny block in the reference chromosome. The Genome Painter output page also provides two links: (1) Download Output, which provides access to raw output data generated by OrthoCluster, and (2)

View GBrowse, which provides access to chromosome-level visualization of synteny blocks. Additionally, each synteny block in the target genome is cross-linked to a GBrowse (*UNIT 9.9*; Stein et al., 2002) view of the corresponding region (Fig. 6.10.5).

7. The user can browse synteny blocks in detail by accessing a GBrowse view via the View GBrowse link (see Fig. 6.10.4) or by clicking the synteny blocks in the Genome Painter image. Once the user clicks either one of these links, a genome browser page (Fig. 6.10.5) is brought up.

***BASIC PROTOCOL 2***

## USING OrthoCluster VIA WEB INTERFACE WITH PREDEFINED DATASETS AND PARAMETERS

Precomputed synteny blocks are available on the View Synteny page. The advantage of this page is that the user does not need to prepare input files for identifying and visualizing synteny blocks. In the current release (Release 2), five groups of species are available for visualization: *Plasmodium*, *Pseudomonas*, *Caenorhabditis*, *Drosophila*, and Mammals. Users are allowed to select up to three genomes (including the reference genome) for visualization. To visualize synteny blocks among more than three genomes, users can generate their own correspondence files with tools like OrthoMCL locally, and then use the Run OrthoCluster page (see Basic Protocol 1). Alternatively, if the user does not require visualization of synteny blocks, OrthoCluster can be used locally (see Basic Protocol 3).

### *Necessary Resources*

*Hardware*

Computer with Internet connection

*Software*

Internet browser, e.g., Internet Explorer (*http://www.microsoft.com/ie*), Firefox (*http://www.mozilla.org/firefox*), or Safari (*http://www.apple.com/safari*)

1. From the OrthoClusterDB homepage, click on View Synteny at the top of the page.
2. From the Select Group drop-down menu on the View Synteny page, select one of the five groups available (Fig. 6.10.6): *Plasmodium* (6 genomes), *Pseudomonas* (14 genomes), *Caenorhabditis* (2 genomes), *Drosophila* (12 genomes), and Mammals (20 genomes). For this demonstration, select *Caenorhabditis*.
3. From the Select Reference Genome drop-down menu, select *Caenorhabditis elegans* as the reference genome.
4. Select *Caenorhabditis briggsae* as the target genome by checking the box to the left of Select Target Genome(s) (up to 2 selections).
5. Select one of the two options (radio buttons) to visualize precomputed synteny blocks: (1) Perfect for perfectly conserved synteny blocks, i.e., blocks of genes in which the order and strandedness of genes composing the block are conserved and in which no mismatches are allowed (see Critical Parameters for details) *or* (2) Imperfect for synteny blocks generated by allowing a maximum of 5% in-map gene mismatches and 10% out-map gene mismatches. See Critical Parameters section for a detailed explanation of these and additional parameters.
6. Click Submit to get the results for the specified genomes and parameters.

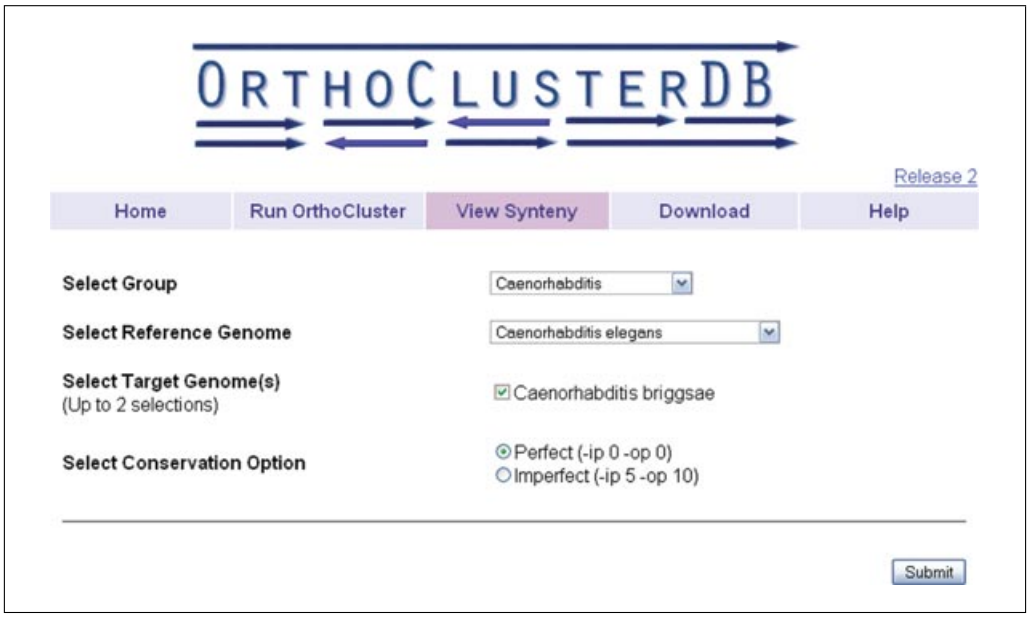


**Figure 6.10.6** View Synteny page on the OrthoClusterDB Web site. Five groups of species are available for visualization of precomputed synteny blocks. In this example, the group selected is *Caenorhabditis*, the reference genome is *C. elegans*, and the target genome is *C. briggsae*. Perfect and imperfect precomputed synteny blocks are available (see text). For this example, visualization of perfect synteny blocks is selected.

7. The results page with the OrthoCluster output is shown. The results page is essentially the same as that described in Basic Protocol 1 (see steps 6 and 7 of Basic Protocol 1 for details on the output generated).

## USING OrthoCluster LOCALLY FOR THE DETECTION OF SYNTENY BLOCKS

***BASIC PROTOCOL 3***

This protocol illustrates how the stand-alone version of OrthoCluster is applied for identifying synteny blocks between two genomes, those of the nematodes *Caenorhabditis elegans* and its sister species *Caenorhabditis briggsae*. The procedure for running OrthoCluster for identifying synteny blocks among multiple genomes is similar.

### *Necessary Resources*

*Hardware*

Any Unix/Linux, Windows, or MacOS workstation

*Software*

OrthoCluster: available at *http://genome.sfu.ca/projects/orthocluster/orthocluster.tar.gz* (the precompiled versions for Unix/Linux, Windows, and MacOS systems are available in this package)

*Files*

The input files required in this protocol are described in the Files section at the beginning of the unit

### *Run OrthoCluster*

We explain here how to run OrthoCluster on a Linux workstation. The steps described here are identical to those necessary for running OrthoCluster on a Windows or Mac system.

1. For this example, make a working directory called EXAMPLE in the home directory and then copy OrthoCluster and the input files into this directory. For this example, type:

```
[ivergara@grouse ~]$ mkdir EXAMPLE
[ivergara@grouse ~]$ cd EXAMPLE
```

2. Download OrthoCluster to your working directory:

```
[ivergara@grouse EXAMPLE]$ curl —O
http://genome.sfu.ca/projects/orthocluster/
orthocluster.tar.gz
```

3. Extract the tar-gzipped file as follows:

```
[ivergara@grouse EXAMPLE]$ tar xvfz orthocluster.
tar.gz
```

   This command will uncompress and unpack the different precompiled versions (for Unix/Linux, Windows, and Mac) of OrthoCluster to a directory called ORTHOCLUSTER.

4. *Optional:* Copy the OrthoCluster executable file from the ORTHOCLUSTER directory to your working directory. The program is precompiled and ready to be used. In the case of Linux system, copy the file as follows:

```
[ivergara@grouse EXAMPLE]$ cp ORTHOCLUSTER/LINUX/
OrthoCluster ./
```

   *Users can run the program directly from the folder in which the program is located. This step is not mandatory, but it is cleaner for users to keep the program and files in a working directory.*

5. Change the permission of OrthoCluster:

```
[ivergara@grouse EXAMPLE]$ chmod u+x OrthoCluster
```

6. *Optional:* Place your genome files and correspondence file in your working directory.

   In this example, we assume that the user has the genome files and correspondence file in the folder with path /home/ivergara/2009/CURRENT_PROTOCOLS/DATASETS/.

   The user should modify this path according to the location of the input files:

```
[ivergara@grouse EXAMPLE]$ cp /home/ivergara/2009/
CURRENT_PROTOCOLS/DATASETS/genome_ele.txt ./
[ivergara@grouse EXAMPLE]$ cp /home/ivergara/2009/
CURRENT_PROTOCOLS/DATASETS/genome_bri.txt ./
[ivergara@grouse EXAMPLE]$ cp /home/ivergara/2009/
CURRENT_PROTOCOLS/DATASETS/mapping_ele_bri.txt ./
```

   *The user can access the files directly from the folder in which they are located. This step is not mandatory, but it might be easier for the user to keep the program file and data files in the same working directory.*

7. Use the command below to run OrthoCluster within your working directory. The command provided here is a generic one, and the user should revise the options described in Table 6.10.1 for best results. Example 1, below, shows how to run OrthoCluster to detect perfect synteny blocks (blocks of genes in which the order and strandedness of genes are conserved, and no mismatches are allowed) between *C. elegans* and *C. briggsae*. The command is:

```
[ivergara@grouse EXAMPLE]$ ./OrthoCluster -g1
<genome_1_file> -g2 <genome_2_file> ... —gn
<genome_n_file> -m <correspondence_file> -d
<job_name> [OPTIONS]
```

where:

<genome_1_file>: path to the genome file for species 1.
<genome_2_file>: path to the genome file for species 2.
.
.
.
<genome_n_file>: path to the genome file for species *n*.
<correspondence_file>: path to the correspondence file.

See Table 6.10.1 for details about [OPTIONS].

Three output files are generated. These files are named with the prefix <job_name>.

a. A .cluster file, that shows all the synteny blocks detected among genomes,
b. A .stat file, that summarizes information related to the size distribution of the synteny blocks.
c. A .log file, which keeps a record of the OrthoCluster run.

*See Anticipated Results for further explanation on the format of these files.*

***Example 1: Running OrthoCluster for detecting perfect synteny blocks between C. elegans and C. briggsae***

Next, we provide an example of running OrthoCluster between *C. elegans* and *C. briggsae* to detect perfect synteny blocks. We call a perfect synteny block one that preserves order and strandedness of genes within the block, and for which no mismatches are allowed.

After going through steps 1 to 5 as specified above in this protocol, run OrthoCluster as follows:

```
[ivergara@grouse EXAMPLE]$ ./OrthoCluster -g1
genome_ele.txt -g2 genome_bri.txt -m mapping_ele_bri.txt
-d perfect —rs —f —i 0 —ip 0 —o 0 —op 0
```

For an explanation of the different parameters, see Table 6.10.1, as well as Critical Parameters.

***Scripts for converting to GFF format***

Once the OrthoCluster output is available, the user can convert the .cluster file to a GFF (General Feature Format) format. This format is generally used for displaying sequence features in a genome browser (in this case, the synteny blocks). We provide two additional scripts for this purpose (available at *http://genome.sfu.ca/projects/orthocluster/*):

```
orthclu2gff3-perfect.pl
```

This script is useful for parsing the OrthoCluster output only for the detection of perfect synteny blocks (as shown in the above example).

To download:

```
[ivergara@grouse EXAMPLE]$ curl —O
http://genome.sfu.ca/projects/orthocluster/
orthclu2gff3-perfect.pl
```

To make it executable:

```
[ivergara@grouse EXAMPLE]$ chmod u+x orthclu2gff3-
perfect.pl
```

Run the script as follows:

```
[ivergara@grouse EXAMPLE]$ ./orthclu2gff3-perfect.pl
<.cluster output> <genome_1_file> <genome_2_file>
... <genome_N_file> <avoid_nested_blocks>
```

where:

`<.cluster output>`: path to the OrthoCluster output.
`<genome_1_file>`: path to the genome file for species 1.
`>genome_2_file<`: path to the genome file for species 2.
.
.
.
`<genome_n_file>`: path to the genome file for species n.
`<avoid_nested_blocks>`: 1 if nested blocks should be avoided, 0 if not.

This will generate one `gff3` file for each pair of reference and target genomes in the dataset.

Following Example 1, the user should run the following command:

```
[ivergara@grouse EXAMPLE]$ ./orthclu2gff3-perfect.pl
perfect.cluster genome_ele.txt genome_bri.txt 0
```

`0` means that nested blocks should not be avoided when parsing the OrthoCluster output. If `1` is used instead, then nested blocks will not be reported in the `gff` file.

```
orthclu2gff3-imperfect.pl
```

This script is useful for parsing the OrthoCluster output for any set of parameters that does not generate perfect synteny blocks. In general, we refer to an imperfect synteny block as one that is not generated by using the parameters that yield perfect blocks (see Example 1). It is installed and executed in the same manner as the `orthclu2gff3-perfect.pl` program.

***ALTERNATE PROTOCOL***

## USING OrthoCluster FOR THE DETECTION OF SEGMENTAL DUPLICATIONS

This protocol illustrates how OrthoCluster (stand-alone version) is applied for identifying duplicated blocks within the genome of the nematode *Caenorhabditis elegans* (Vergara et al., 2009). Two types of input files are needed here: the genome file and the correspondence file. For this example, we have prepared the genome file, `genome_ele.txt`, representing the *C. elegans* genome, and the correspondence file, `paralogs_elegans.txt`, containing paralogous relationships within this genome.

***Necessary Resources***

*Hardware*

Unix/Linux, Windows, or MacOS workstation

*Software*

OrthoCluster: available at *http://genome.sfu.ca/projects/orthocluster/orthocluster.tar.gz* (the precompiled versions for Unix/Linux, Windows and MacOS systems are available from this package)

*Files*

The input files required in this protocol are described in the Files section at the beginning of the unit; in contrast to Basic Protocol 1, paralogous relationships instead of orthologous relations should be included in the correspondence file

***Run OrthoCluster***

We explain here how to run OrthoCluster on a Linux workstation. The steps described are identical to those necessary for running OrthoCluster on a Windows or Mac system. Follow the steps described in Basic Protocol 3 for downloading and installing OrthoCluster.

The command provided here is a generic one, and the user should revise the options described in Table 6.10.1 for best results. Example 2, below, shows how to run OrthoCluster to detect perfect duplicated blocks (blocks of genes within the genome in which the order and strandedness of genes are conserved, and no mismatches are allowed) within *C. elegans*.

Use the following command to run OrthoCluster:

```
[ivergara@grouse EXAMPLE]$ ./OrthoCluster -g1
<genome_file> -m <correspondence_file> -d
<job_name> [OPTIONS]
```

where:

`<genome_file>`: path to the genome file for the species under analysis.
`<correspondence_file>`: path to the correspondence file containing the paralogous relationships.

See Table 6.10.1 for details about `[OPTIONS]`.

***Example 2: Running OrthoCluster for detecting perfect duplicated blocks within C. elegans***

We provide an example of running OrthoCluster within *C. elegans* to detect perfect duplicated blocks. We call a perfect duplicated block one that preserves order and strandedness of genes within the block, and for which no mismatches are allowed.

After going through steps 1 to 5 of Basic Protocol 3, run OrthoCluster as follows:

```
[ivergara@grouse EXAMPLE]$ ./OrthoCluster -g1
genome_ele.txt -m paralogs_elegans.txt -d perfect —rs
—f —i 0 —ip 0 —o 0 —op 0
```

***Output files***

When running OrthoCluster within a single genome, three output files are generated. These files are named with the prefix `<job_name>`.

a. A `.cluster` file, which shows all detected duplicated blocks,

b. A `.stat` file, which summarizes information related to the size distribution of the duplicated blocks.

c. A `.log` file, which keeps a record of the OrthoCluster run.

See Anticipated Results for a detailed explanation on the format of these files.

For an explanation of the different parameters, see Table 6.10.1, as well as Critical Parameters.

## COMMENTARY

### Background Information

The problem that OrthoCluster tackles is the following. Given a set of genomes and a correspondence between orthologous genes in different genomes, find *synteny blocks*, i.e., chromosomal regions in which orthologous genes are clustered, with the following properties:

a. The total number of orthologous genes in each block is less than or equal to −`u`;

b. The total number of orthologous genes in each block is greater than or equal to −`l`;

c. The number of orthologous genes that have correspondence in other regions of the genome, but not within the synteny block, is less than or equal to −`i`;

d. The percentage of orthologous genes within a block (with respect to the total number of in-map genes within the synteny block) that have correspondence in other regions of the genome but not within the synteny block is less than or equal to −`ip`;

e. The number of out-map genes (genes without any orthologous relation among genomes) within a block is less than or equal to `o`;

f. The percentage of genes that are out-map genes with respect to the total number of genes within the block is less than or equal to −`op`,

where −`u`, −`l`, −`i`, −`ip`, −`o`, and −`op` are user-defined parameters (Table 6.10.1).

The user may also require that the order/strandedness of genes in the block be conserved among all the genomes under analysis. This is achieved with the −`r`, `-s`, and −`rs` parameters (Table 6.10.1) described in Critical Parameters.

### Critical Parameters

#### *The* −`i`, `-ip`, `-o`, *and* −`op` *parameters*

The −`ip` parameter works as follows. If this parameter is set to a certain value *ip*, with $0 \leq ip < 100$, then the constraint is exceeded if $100 \times$ (in-map mismatches/in-map genes in the block) is $> ip$. Also, a dependency between the −`i` and −`ip` parameters exists. A block must satisfy at least one of the −`i` and −`ip` constraints. A block violating both constraints is not considered as valid, and hence is not reported.

The −`op` parameter works as follows. If this parameter is set to a certain value *op*, with $0 \leq op < 100$, then the constraint is exceeded if $100 \times$ (out-map mismatches/total number of genes in the block) is $> op$. The dependency between the −`o` and −`op` parameters is the same as the one existing between −`i` and −`ip`.

#### *The* −`r`, `-s`, *and* −`rs` *parameters*

When the parameter −`r` is used, the program will search for synteny blocks that have either *consistent* order, i.e., the order of orthologous genes in a synteny block is the same, or *inverted* order, i.e., the order of the orthologous genes in a synteny block is inverted with respect to the order of the genes in the second block.

When the −`s` parameter is used, the program will search for synteny blocks that have either *consistent* strandedness, i.e., for each pair of orthologs within the synteny block, the orientation is the same, or *reversed* strandedness, i.e., for each pair of orthologs within the synteny blocks the orientation is reversed.

Based on how −`r` and −`s` work, when the user provides −`r` −`s` in the same run of OrthoCluster, four different types of synteny blocks can be reported:

a. Synteny blocks with consistent order and consistent strandedness.

b. Synteny blocks with consistent order and reversed strandedness.

c. Synteny blocks with inverted order and consistent strandedness.

d. Synteny blocks with inverted order and reversed strandedness.

If the user sets −`rs` instead of −`r` −`s`, then only cases (a) and (d) will be considered by the algorithm when searching for synteny

blocks. Hence, using —rs is a more stringent criterion for searching blocks than using —r —s separately.

## Anticipated Results

For Basic Protocol 3:

OrthoCluster yields as output three files:

a. A `.cluster` file, which shows all the synteny blocks detected among genomes.

b. A `.stat` file, that summarizes information related to the size distribution of the synteny blocks.

c. A `.log` file, which keeps a record of the OrthoCluster run.

The name of each file is the Job ID assigned by the user.

Each file has a common header with the structure shown in Figure 6.10.7.

Figure 6.10.8 is an example of this header for comparing two genomes, for finding nonoverlapping blocks (`-f`) with a minimum block size of 1 (`-l 1`), a maximum block size of 1000 (`-u 1000`), with no mismatches (`—i0 —ip0 —o0 —op0`), and preserving the four different types of order (`-r`) and strandedness (`-s`).

### *The* `.cluster` *file*

Given *N* genomes, the first line of each block has the format shown in Figure 6.10.9. Then, each gene in each genome conforming a block is described by the format line shown in Figure 6.10.10.

If there are more two genomes being analyzed, then column 5 will suppress the display of `'++'` and `'+'`.

An example of a block is shown in Figure 6.10.11. Following the format described above for the header, it is easy to see that:

```
Column 1 : CL-61
Column 2 : ' '
Column 3 : 8
Column 4 : 12
Column 5 : 1
Column 6 : 2
Column 7 : 0
Column 8 : 2
```

```
Time and date of the current running.
Settings & Parameters:
No. of sequence                          : number of -gi parameters
Max group size                           : -u
Min group size                           : -l
Max out-map-mismatches                   : -o
Max out-map-mismatch percentage          : -op
Max in-map-mismatches                    : -i
Max in-map-mismatch percentage           : -ip
Find order preserving clusters           : -r
Find strand preserving clusters          : -s
Find non-overlapping clusters            : -f
Mapping file name                        : -m
Sequence 1 file name                     : -g1
Sequence 2 file name                     : -g2
.
.
.
Sequence N file name                     : -gN
Job ID                                   : -d
```

**Figure 6.10.7** Structure of the common header of OrthoCluster files.

```
Sat Jul 14 17:16:44 2007

Settings & Parameters:

No. of sequence                                          : 2

Max group size                                           : 1000

Min group size                                           : 1

Max out-map-mismatches                                   : 0

Max out-map-mismatch percentage                          : 0%

Max in-map-mismatches                                    : 0

Max in-map-mismatch percentage                           : 0%

Find order preserving clusters (-r)                      : Yes

Find strand preserving clusters (-s)                     : Yes

Find order and strandedness preserving clusters (-rs)    : No

Find non-overlapping clusters                            : Yes

Mapping file name                                        : G1_G2_orthologs.txt

Sequence 1 file name                                     : G1_genes.list

Sequence 2 file name                                     : G2_genes.list

Output file name                                         : output
```

**Figure 6.10.8** Example of this header for comparing two genomes, for finding nonoverlapping blocks (`-f`) with a minimum block size of 1 (`-l 1`), a maximum block size of 1000 (`-u 1000`), with no mismatche, and preserving the four different types of order (`-r`) and strandedness (`-s`).

***The* `.stat` *file***

In this file, the total number of synteny blocks, number of non-nested synteny blocks, and the number of nested synteny blocks are reported.

Then, for each genome, the following values are displayed:

```
Smallest cluster size
Largest cluster size
Median of nested cluster size
Median of non-nested cluster size
```

Also, the size distribution for the nested and non-nested synteny blocks is reported as follows:

```
SIZE NUMBER_OF_CLUSTERS
1 <number of blocks of size 1>
2 <number of blocks of size 2>
.
.
.
K <number of blocks of size K>
```

Finally, the syntenic correlation value between two genomes and the corresponding contingency table is reported, according to this equation (Housworth and Postlethwait, 2002).

$$\rho = \frac{\sum_{i=1}^{r}\sum_{j=1}^{c}\left(n_{ij} - e_{ij}\right)^2}{n \cdot \min\left\{r-1, c-1\right\} e_{ij}}$$

$$0 \le \rho \le 1$$

$\rho = 1$ if and only if, for one of the two species, knowing which chromosome an ortholog belongs to in that species determines which chromosome the ortholog is on in the other species.

$\rho = 0$ if and only if the counts of orthologs are perfectly independently scattered on the chromosomes of the two species.

To compute this value, an $r \times c$ contingency table is created and displayed, summarizing the number of ortholog genes on each of the chromosomes 1 through $r$ of the first genome that are on each of the chromosomes 1 through $c$ of the second genome.

$n_{ij}$ is the observed number of genes on species $A$ chromosome $i$ with an ortholog on species $B$ chromosome $j$.

$e_{ij}$ is the expected number of genes in each cell assuming that the genes are scattered independently in the two genomes.

```
Column  1       : cluster ID.
Column  2       : 'N' if it is a nested cluster; empty otherwise.
Column  1+2     : cluster size for genome 1.
Column  2+2     : cluster size for genome 2.
.
.
.
Column  N+2     : cluster size for genome N.
Column  1+(N+2) : in-map mismatch for genome 1.
Column  2+(N+2) : in-map mismatch for genome 2.
.
.
.
Column  N+(2+N) : in-map mismatch for genome N.
Column  1+(2+2N): out-map mismatch for genome 1.
Column  2+(2+2N): out-map mismatch for genome 2.
.
.
.
Column  N+(2+2N): out-map mismatch for genome N.
```

**Figure 6.10.9** Format of the first line of each block (`.cluster` file) given *N* genomes.

```
Column 1: gene ID within the block. The corresponding orthologous genes in
the other genomes will have the same gene ID. If this gene is an in-map or
out-map mismatch, then no ID is assigned.
Column 2: position of the gene within the genome (obtained from internal
OrthoCluster sorting).
Column 3: strand.
Column 4: chromosome.
Column 5:
        '*' if the gene is an in-map mismatch.
        '++' if the gene has multiple matches in the other genome.
        '+' if the gene is one of multiple matches of another gene in the other genome.
        ' ' otherwise.
Column 6: gene name.
```

**Figure 6.10.10** Format line for each gene conforming a block.

```
CL-61          8     12           1    2              0    2
 1:     162    -1          I            Y48G8AL.11          7:     1782     1     chrI           CBG08484
 2:     163     1          I            Y48G8AL.6           7:     1783     1     chrI           CBG08485
 3:     164     1          I            Y48G8AL.5           6:     1784    -1     chrI     +     CBG08486
        165     1          I      *     Y48G8AL.1           6:     1785    -1     chrI     +     CBG08487
        166    -1          I            Y48G8AL.12          7:     1786    -1     chrI     *     CBG08488
 4:     167    -1          I            Y48G8AL.13          5:     1787     1     chrI           CBG08489
 5:     168     1          I     ++     Y48G8AL.15          4:     1788     1     chrI           CBG08490
 6:     168     1          I     ++     Y48G8AL.15                 1789    -1     chrI           CBG08491
 7:     169    -1          I            Y48G8AL.1                  1790     1     chrI     *     CBG08492
                                                             3:     1791    -1     chrI           CBG08493
                                                             2:     1792    -1     chrI           CBG08494
                                                             1:     1793     1     chrI           CBG08495
```

**Figure 6.10.11** A synteny block generated by running the standalone OrthoCluster program.


## Acknowledgments

We would like to thank Christian Frech and Maja Tarailo-Graovac for critical reviewing of the manuscript. Ismael A. Vergara is supported by a Weyerhaeuser Molecular Biology and Biochemistry Fellowship. This work is supported by a Natural Sciences and Engineering Research Council (NSERC) of Canada support to Nansheng Chen, who is also a Michael Smith Foundation for Health Research (MSFHR) Scholar.


## Literature Cited


Alexeyenko, A., Tamas, I., Liu, G., and Sonnhammer, E.L. 2006. Automatic clustering of orthologs and in paralogs shared by multiple proteomes. *Bioinformatics* 22:e9-e15.

Flicek, P., Aken, B.L., Beal, K., Ballester, B., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cunningham, F., Cutts, T., Down, T., Dyer, S.C., Eyre, T., Fitzgerald, S., Fernandez-Banet, J., Gräf, S., Haider, S., Hammond, M., Holland, R., Howe, K.L., Howe, K., Johnson, N., Jenkinson, A., Kähäri, A., Keefe, D., Kokocinski, F., Kulesha, E., Lawson, D., Longden, I., Megy, K., Meidl, P., Overduin, B., Parker, A., Pritchard, B., Prlic, A., Rice, S., Rios, D., Schuster, M., Sealy, I., Slater, G., Smedley, D., Spudich, G., Trevanion, S., Vilella, A.J., Vogel, J., White, S., Wood, M., Birney, E., Cox, T., Curwen, V., Durbin, R., Fernandez-Suarez, X.M., Herrero, J., Hubbard, T.J., Kasprzyk, A., Proctor, G., Smith, J., Ureta-Vidal, A., and Searle, S. 2008. Ensembl 2008. *Nucleic Acids Res* 36:D707-D714.

Housworth, E.A. and Postlethwait, J. 2002. Measures of synteny conservation between species pairs. *Genetics* 162:441-448.

Li, L., Stoeckert, C.J. Jr., and Roos, D.S. 2003. OrthoMCL: Identification of ortholog groups for eukaryotic genomes. *Genome Res.* 13:2178-2189.

Ng, M.P., Vergara, I.A., Frech, C., Chen, Q., Zeng, X., Pei, J., and Chen, N. 2009. OrthoClusterDB: An online platform for synteny blocks. *BMC Bioinform.* 10:192.

O'Brien, K.P., Remm, M., and Sonnhammer, E.L. 2005. Inparanoid: A comprehensive database of eukaryotic orthologs. *Nucleic Acids Res.* 33:D476-D480.

Remm, M., Storm, C.E., and Sonnhammer, E.L. 2001. Automatic clustering of orthologs and inparalogs from pairwise species comparisons. *J. Mol. Biol.* 314:1041-1052.

Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A., and Lewis, S. 2002. The generic genome browser: A building block for a model organism system database. *Genome Res.* 12:1599-1610.

Vergara, I.A., Mah, A.K., Huang, J.C., Tarailo-Graovac, M., Johnsen, R.C., Baillie, D.L., and Chen, N. 2009. Polymorphic segmental duplication in the nematode *Caenorhabditis elegans*. *BMC Genomics* 10:329.

Zeng, X., Pei, J., Vergara, I.A., Nesbitt, M., Wang, K., and Chen, N. 2008. OrthoCluster: A new tool for mining synteny blocks and applications in comparative genomics. *In* 11th International Conference on Extending Database Technology (EDBT), March 25-30, 2008, Nantes, France. Association for Computer Machinery, New York.