


Elementary Sequence Analysis

Brian Golding, Dick Morton and Wilfried Haerty

Department of Biology
McMaster University
Hamilton, Ontario
L8S 4K1

These notes are in Adobe Acrobat format (they are available upon request in other formats) and they can be obtained from the website <http://helix.biology.mcmaster.ca/courses.html>. Some of the programs that you will be using in this course and which will be run locally can be found at <http://evol.mcmaster.ca/p3S03.html>.

The “blue text” should designate links within this document while the “red text” designate links outside of this document. Clicking on the latter should activate your web browser and load the appropriate page into your browser. If these do not work please check your Acrobat reader setup. The web links are accurate to the best of our knowledge but the web changes quickly and we cannot guarantee that they are still accurate. The links designated next to the JAVA logo, , require that JAVA be installed on your computer.

These notes are used in Biology 3S03. The purpose of this course is to introduce students to the basics of bioinformatics and to give them the opportunity to learn to manipulate and analyze DNA/protein sequences. Of necessity only some of the more simple algorithms will be examined.

The course will hopefully cover ...

- databases of relevance to molecular biology.
- some common network servers/sites that provide access to these databases.
- use of the internet to obtain sequence analysis software and data.
- methods of sequence alignment.
- methods of calculating genetic distance.
- methods of phylogenetic reconstruction.
- codon usage.
- methods for detecting gene coding regions.

The formal part of the course will consist of two approximately one hour lectures each week. Weekly assignments will be provided to practice and explore the lecture material. In addition there will be an optional tutorial to help students with these assignments or other problems. These assignments will be 40% of your grade and three, in class quizzes will make up the remainder.

We would appreciate any comments, corrections or updates regarding these notes.

Golding@McMaster.CA

Morton@McMaster.CA

HaertyW@McMaster.CA

Table of Contents in Brief

In order to speed download, I place here links to the individual chapters in pdf format. The contents of these are shown on the following 'Contents' pages but note that the links will function only for the individual chapter included here.

[Preliminaries](#)
[Basic Unix](#)
[Genomics](#)
[Databases](#)
[Sequence File Formats](#)
[Database Searching](#)
[Sequence Alignment](#)
[Distance Measures](#)
[Reconstructing Phylogenies](#)
[Pattern analysis](#)
[Exon analysis](#)

Contents

1	Preliminaries	1
1.1	Resources	1
1.1.1	Electronic Resources	1
1.1.2	Textbooks	2
1.1.3	Journal sources	7
1.2	Biological preliminaries	10
1.2.1	Some notes on terminology	10
1.2.2	Letter Codes for Sequences	11
2	Computer skills preliminaries	13
2.1	UNIX Operating Systems	13
2.1.1	Logging on/off	14
2.1.2	UNIX File System	14
2.1.3	Commands	17
2.1.4	Help	19
2.1.5	Redirection	20
2.1.6	Shells	20
2.1.7	Special 'hidden' files	21
2.1.8	Background Processes	21
2.1.9	Utilities	22
2.1.10	Editors	22
2.2	Exchange among computers	24
2.2.1	ssh	24
2.2.2	Mail	24
2.3	Scripts-Languages	25
2.4	Obtaining LINUX	25
3	Genomics	27
3.1	Where the data comes from	27
3.2	How DNA is sequenced	27

3.3	The reality of sequencing includes errors	30
3.4	From sequence to genome	33
3.5	Yesterday's new sequencing?	39
3.6	Other kinds of biological data	43
3.6.1	Microarrays	43
3.6.2	Mass spectrometry methods	46
3.6.3	Textual information	47
4	Databases	49
4.1	Introduction	49
4.2	N.C.B.I.	52
4.3	E.M.B.L.	56
4.4	D.D.B.J.	58
4.5	SwissProt	58
4.6	Organization of the entries	61
4.7	Other Major Databases	62
4.8	Remote Database Entry retrieval	65
4.8.1	Entrez	65
4.8.2	NCBI retrieve	68
4.8.3	EMBL get	69
4.8.4	Others	69
4.9	Reliability	70
5	Sequence File Formats	73
5.1	Genbank/EMBL	73
5.2	FASTA	75
5.3	FASTQ	76
5.4	Stockholm format	77
5.5	GDE	79
5.6	NEXUS	81
5.7	PHYLIP	82
5.8	ASN	83
5.9	BSML format	86
5.10	PDB file format	86
6	Database Searching	91
6.1	Are there homologues in the database?	91
6.1.1	FASTA	91
6.1.2	BLAST	99

6.1.3	MPsrch	106
6.2	BLOCKS	110
6.2.1	BLOCKS output	111
6.2.2	Getting the Block	112
6.3	SSearch	118
6.4	Why you should routinely check your sequence	118
7	Sequence Alignment	119
7.1	Dot Plots	119
7.1.1	The Exact Way	119
7.1.2	Identity Blocks	121
7.2	Alignments	128
7.2.1	The Needleman and Wunsch Algorithm	128
7.2.2	The Smith-Waterman Algorithm	131
7.3	Testing Significance	132
7.4	Gaps and Indels	135
7.4.1	“Natural” Gap Weights - Thorne, Kishino & Felsenstein	135
7.5	Multiple Sequence Alignments	136
8	Distance Measures	139
8.1	Nucleotide Distance Measures	139
8.1.1	Simple counts as a distance measure	139
8.1.2	Jukes - Cantor Correction	140
8.1.3	Kimura 2-parameter Correction	142
8.1.4	Tamura - Nei Correction	142
8.1.5	Uneven spatial distribution of substitutions	143
8.1.6	Synonymous - nonsynonymous substitutions	144
8.2	Amino acid distance measures	144
8.2.1	PAM Matrices	145
8.2.2	BLOSUM Matrices	147
8.2.3	GONNET Matrix	148
8.3	Gap Weighting	149
9	Reconstructing Phylogenies	151
9.1	Introduction	151
9.1.1	Purpose	151
9.1.2	Trees of what	151
9.1.3	Terminology	153
9.1.4	Controversy	155

9.2	Distance Methods	155
9.3	Parsimony Methods	157
9.4	Other Methods	160
9.4.1	Compatibility methods	160
9.4.2	Maximum Likelihood methods	160
9.4.3	Method of Invariants	161
9.4.4	Quartet Methods	162
9.5	Consensus Trees	164
9.6	Bootstrap trees	164
9.7	Warnings	167
9.8	Available Packages	168
9.9	PHYLIP	172
9.9.1	PHYLIP Contents	172
10	Pattern Analysis	185
10.1	Base Composition: first order patchiness	185
10.1.1	Genome Patchiness	185
10.2	Dinucleotide Composition: second order patchiness	186
10.3	Strand Asymmetry	187
10.3.1	Chargaff's Rules	187
10.3.2	Replication Asymmetry	188
10.3.3	Transcriptional Asymmetry	189
10.3.4	Codon Selection	190
10.4	Simple Sequence Repeats	190
10.5	Sequence Complexity	190
10.5.1	Information Theory	190
10.5.2	Sequence Window Complexity	192
10.6	Finding Pattern in DNA Sequences	193
10.6.1	Consensus Sequences	193
10.6.2	Matrix Analysis of Sequence Motifs	194
10.6.3	Sequence Conservation and Sequence Logos	195
11	Exon Analysis	199
11.1	Open Reading Frames	199
11.2	Gene Recognition	199
11.2.1	Splice Sites	200
11.2.2	Codon Usage	201
11.2.3	Gene Prediction Software	204
11.2.4	Hidden Markov Models (HMM)	205

11.2.5 Comparison of Programs	205
---	-----

Chapter 9

Reconstructing Phylogenies

“The history of the earth is recorded in the layers of its crust; the history of all organisms is inscribed in their chromosomes” — Hitoshi Kihara 1946

9.1 Introduction

9.1.1 Purpose

The purpose of phylogenetic reconstruction is to attempt to estimate the phylogeny for some data. For any collection of data there will be some ancestral relationship between the sampled sequences. The data itself contains information that can be used to reconstruct or to infer these ancestral relationships. This involves reconstructing a branching structure, termed a phylogeny or tree, that illustrates the relationships between the sequences.

The following discussion is based mainly on Molecular Evolutionary Genetics by M.Nei, Genetic Data Analysis by B.Weir, Of URFs and ORFs by R.Doolittle, Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit by H.von Gunnar, Molecular Systematics by Hollis & Moritz and J. Felsenstein (1982, Quart.Rev.Biol.57:379). Refer to these for more detailed information.

9.1.2 Trees of what

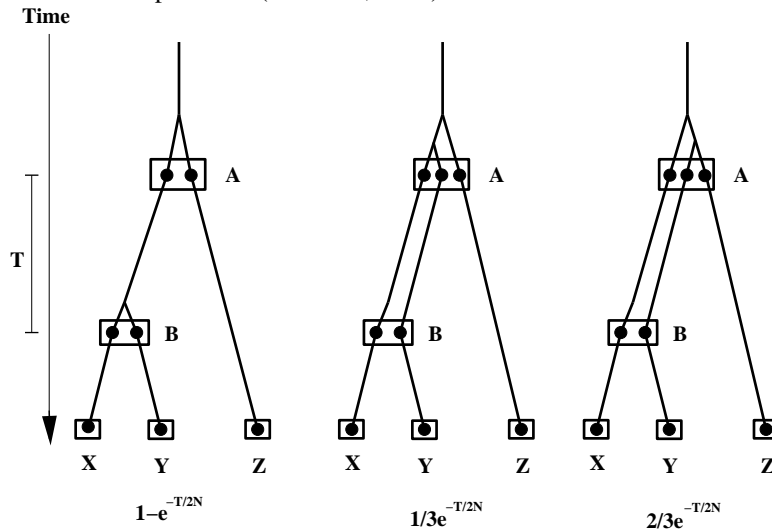
As stated, phylogenetic reconstruction attempts to estimate the phylogeny of some observed data. However, usually people are more interested in using the data to try to infer the species phylogeny and not just the phylogeny of the data. In general, these two are not always the same and estimating the species tree may not be possible. Instead what is estimated has been called a “gene tree” by M.Nei. This is because your data (the sequence of some gene or some other form of data) may not have had the same phylogenetic history as the species within which they are contained.

Consider the species shown in Figure 9.1 (from Nei, 1987). The boxes represent the actual species and the dots represent the genes themselves. In the first example, a reconstructed phylogeny based on these genes would yield something similar to the true species tree. In the second example, the reconstructed phylogeny will provide the same topological tree as the species tree but the branch lengths will all be quite incorrect. In the third example, the reconstructed phylogeny will positively give an incorrect phylogeny. It would suggest that species Y and Z are more closely related when in fact X and Y are more closely related.

All of this stems from the fact that polymorphism can exist within species and the estimated age of many polymorphisms can be quite old. The problem of estimating the wrong topology will be greater when the true distance between speciation events A and B is small.

Even if the first situation applies there may still be errors introduced because the number of changes from one species to

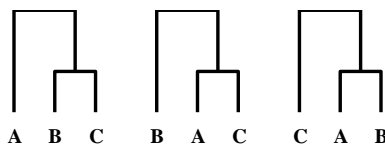
Figure 9.1: Three possible relationships between species (X, Y, Z) and the genes they contain (indicated by dots) when polymorphism is possible at times of speciation (from Nei, 1987).



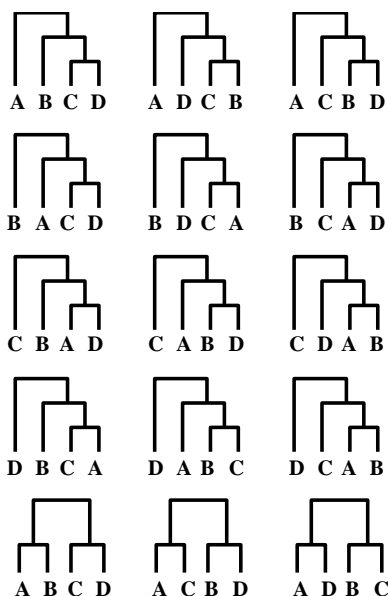
the next is often a stochastic event and subject to sampling error. Hence unless a large number of sites are examined there can be large errors introduced. In addition, if the gene is part of a multigene family it may be difficult to determine the homologous comparable gene in another species. Horizontal gene transfer and gene conversions from unrelated genes are also assumed not to have occurred.

Finally, ignoring all of these caveats, people still usually consider a phylogenetic reconstruction program to act like a “black box”. It takes input, churns around for a while and then spits out the actual phylogenetic answer. This is also incorrect. First, the actual phylogenetic answer can not be obtained by any known method. All methods can only provide estimates and educated guesses of what a phylogenetic tree might look like for the current set of data. These estimates are only as good as the data itself and only as good as the algorithm. Some algorithms in common use are actually quite poor methods. Finally, since these algorithms provide just an estimate, most good methods should also provide an indication of how much variation there is in these estimates.

The problem of tree reconstruction is quite difficult. This is particularly true if all potential tree topologies must be scored or otherwise searched. For three species there are only three trees possible. They are ...



While with four species there are a total of fifteen different topologies possible.



For 5 species there are 105 different topologies. More generally, for any strictly bifurcating phylogeny with n species there are

$$(2n - 3)! / (2^{n-2}(n - 2)!)$$

different topologies. This number gets large very quickly¹. With $n = 15$ species there are

$$213, 458, 046, 676, 875$$

and with $n = 20$,

$$8, 200, 794, 532, 637, 891, 559, 375$$

different trees. Obviously if an algorithm must examine all possible trees, then only a handful of species would be permitted. Even given these, there would be an infinite number of branch length combinations that would have to be searched. Indeed this problem belongs to a class of problems that are called **NP-hard** by computer scientists.

These numbers apply to phylogenies that are rooted. That is there is a point of origin for this phylogeny and it appears in the standard classical fashion that you are probably most familiar with. A phylogeny may also be presented in an unrooted fashion in which case it is called an unrooted tree or a network. For n species there are only

$$(2n - 5)! / (2^{n-3}(n - 3)!)$$

different unrooted trees possible (one step behind the number of rooted trees). Any method that purports to provide variable rates of evolution (or substitution) along each branch should generate its output in the form of an unrooted tree. This is because when rates of evolution are free to vary there is no way to determine the location of a root for a tree.

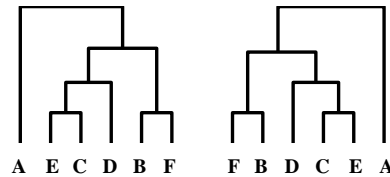
9.1.3 Terminology

There are whole dictionaries that have been created for this field of science (so that people can talk very precisely about what they mean - though this still has not helped to avoid many confusions and useless fights). All of the background

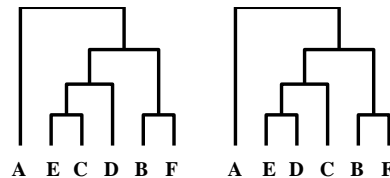
¹A tree calculator 

and terminology necessary can not be provided here. For the present, I simply want to provide you with a subset of the terminology that will enable you to understand some of the problems, some of the methods and to be able to read some of the literature.

The topology of a tree is simply the branching order of the species independent of the branch lengths. Different phylogenies can have the same topology and yet look quite different due to variable branch lengths. If however, branch lengths are all drawn to the same scale then two phylogenies might appear similar whether or not they are identical. Note for example, that



are identical trees while



are not identical.

Because these methods apply equally well to individual samples, to population samples, to species samples and so on, rather than labelling them anyone group it is common practice to label the groups OTU's. These are operational taxonomic units and are meant to represent whatever group of organisms, populations, species, families are under consideration. The individual OTU's or taxa correspond to the terminal nodes of the phylogeny (also called the tips, leaves or external nodes). Places where the interior branches meet are termed internal nodes (also called vertices). An outgroup is an OTU or taxa which is included in the study with the explicit purpose of finding the root of the tree for the remainder of the OTU's. Convergences or parallelisms of a particular character at a site are called homoplasies. These are the characters that usually provide the greatest problems for any tree reconstruction algorithm.

Willi Hennig is the person who began a very systematic approach to phylogeny reconstruction. He was looking mainly at taxonomic characters and was able to show that only shared, derived characters could be used to clearly establish a phylogenetic relationship. He showed that simply having derived characters (i.e. characters that are not ancestral), or having shared ancestral characters are not sufficient to establish a phylogeny. The terminology used for these character states is

- plesiomorphy - an ancestral character state.
- apomorphy - a derived character state.
- synapomorphy - a shared derived character state.
- symplesiomorphy - a shared ancestral character state.
- autapomorphy - a unique derived character state.

Hence only synapomorphic characters are useful for determining a phylogeny. As an example, suppose that there is a genus of plants in which one species develops red petals (with the ancestral form being white petals). Suppose it underwent speciation such that there are now two red-petalled species and that there still exist five white-petalled species. Then white petals is the plesiomorphic character, red petals is an apomorphic character, the white petals among the five species is a symplesiomorphic character, the red petals among the two species is a synapomorphic character (and points to these two

species as being phylogenetically related). If another species arose with purple petals, this would be an autapomorphic character. Note that this depends on being able to identify the primitive or ancestral state. It is generally not possible to unambiguously determine the direction of change for nucleotide characters (hence some strict adherents would claim that you can not produce cladograms from this data).

In addition, multistate characters can be either ordered or unordered. Nucleotide sequences are considered to be unordered since a “C” for example is not necessarily intermediate between “A” and “G”. Again, it is more normal to encounter ordered characters in the analysis of morphological characters. There is also the concept of character polarity which is the assessment of the direction of character change. This most generally involves identifying one character as an ancestral state.

9.1.4 Controversy

Within the field of phylogenetic reconstruction and taxonomy there have, in the past, been two different ways and two different philosophies to the process of reconstructing a phylogeny. The discussions between these groups about the best ways to proceed have often been acrimonious and counter-productive.

One approach is the phenetic approach. In this approach, a tree is constructed by considering the phenotypic similarities of the species without trying to understand the evolutionary pathways of the species. Since a tree constructed by this method does not necessarily reflect evolutionary relationships but rather is designed to represent phenotypic similarity, trees constructed via this method are called phenograms. A phylogenetic tree based on such information is often termed a dendrogram (a branching order that may or may not be the correct phylogeny).

The second approach is called the cladistic approach. Via these methods, a tree is reconstructed by considering the various possible pathways of evolution and choosing from amongst these the best possible tree. Trees reconstructed via these methods are called cladograms.

The phenetic philosophy as a way to do taxonomy is definitely incorrect. However, this does not mean that phenetic methods are necessarily poor estimates of the cladogram. For character data where ancestral forms are known and to construct a taxonomic classification the cladistic approach is almost certainly superior. However, the cladistic methods are often difficult to implement with assumptions that are not always satisfied with molecular data. The phenetic approaches are generally faster algorithms and often have nicer statistical properties for molecular data. Hence, there appears to be a place for both types of methods in the analysis of molecular sequence data.

9.2 Distance Methods

The archetypical phenetic approach uses distance methods. These methods take the input data and derive from them some measure of similarity/difference between species and from this construct a tree that tries to match this data.

Up until the late 1970's the methods by which the art of taxonomy was performed were never explicitly defined and the relationships between species were determined in an unspecified manner (the value of Willi Hennig's work was not clearly recognized at this time). The field of numerical taxonomy was proposed by Sokal and Sneath (1963) as a way to make the common practices of most taxonomists more rigorous. In the emergence of techniques to perform numerical taxonomy many of the methods that were first applied were based on distance matrices. This is in part, because a small group of numbers are easier to handle computationally. To manipulate the complete data set without the aid of more powerful computers than were available at that time would have been too difficult.

The simplest of the distance methods is a type of cluster algorithm that is known as UPGMA (unweighted pair group method using arithmetic averages). This method has gained popularity mostly because of this simplicity and because of its speed (though many other distance methods are as fast).

Cluster methods are a collection of methods that construct the tree by linking the least distant pairs of taxa, followed by successively more distant taxa. When two taxa are clustered they lose their individual identities and new distances are calculated from the original matrix that correspond to the loss of these two taxa and their replacement by a new joint taxa. At each step of the algorithm the total number of OTUs declines by one and the algorithm is finished when the final two OTUs are clustered. In general these methods only permit bifurcating trees. This is not a limitation since branch lengths

can be zero (defining a trifurcation in practice).

This method begins with the construction of a distance matrix (d_{ij}). The two taxa that have the smallest distances are clustered together (assume that this is between the i -th and j -th taxa) and form a new OTU. The branch lengths for the i -th and j -th taxa are taken to be half of the distance between them (hence the depth of the branch between i and j is $d_{ij}/2$). A new distance matrix is constructed that replaces all distances involving the i -th and j -th taxa with the average distance to these two. Thus, for the k -th taxa its distance to the new (i, j) cluster is defined as $(d_{ik} + d_{jk})/2$. The branch length is taken to be the average distance between the OTUs. Then again, the two taxa or OTUs with the smallest distances are clustered together. If the smallest distance were between the k -th taxa and the new (i, j) cluster, the new distance to the l -th taxa is defined as $(d_{il} + d_{jl} + d_{kl})/3$. In general if OTU i and OTU j are to be clustered then the new distance is $d_{k(i,j)} = (T_i d_{ki} + T_j d_{kj}) / (T_i + T_j)$ (where T_i is the number of taxa in OTU i). This process continues until all OTUs have been clustered together.

Some data come naturally in the form of a distance between species. For example measures of DNA homology through DNA hybridization / melting curves and measures from immunological data. For other forms of data, the distances are calculated from sequences of characters. The reduction of this data from sequences to a single number obviously leads to a loss of information. But you can gain a great deal from the speed and simplicity of these distance methods.

With distance methods it is generally assumed (whether intended or not) that the sum of the branch lengths in such trees correlates directly with the expected phenotypic distance between taxa and further more that this corresponds to some proportional measure of time. This is generally not a valid assumption. Hence corrections for distances and accurate measures of the distance become very important.

This method obviously assumes that the taxa are all extant and that all rates of change are equal. This is an explicit assumption of the method and yet we know of many examples where rates of evolution vary between taxa. Violation of this assumption will cause the UPGMA algorithm to perform very poorly.

Another very popular distance method is the Neighbour Joining Method (Saitou and Nei 1987, *Mol. Biol. Evol.* 4:406). This method attempts to correct the UPGMA method for its strong assumption that the same rate of evolution applies to each branch. Hence this method yields an unrooted tree. A modified distance matrix is constructed to adjust for differences in the rate of evolution of each taxon. Similar to the UPGMA method, the least distant pairs of nodes are linked and their common ancestral node is added to the tree, their terminal nodes are pruned from the tree. This continues until only two nodes remain.

The method begins by finding the modified matrix. To do this calculate the net difference of species i from all other taxa as

$$r_i = \sum_k d_{ik}$$

(where $d_{ii} = 0$). Then find the rate-corrected matrix as

$$M_{ij} = d_{ij} - (r_i + r_j) / (n - 2)$$

where n is the number of taxa. Saitou & Nei showed that this equation for M_{ij} (modulo the addition of a constant) is the sum of the least-squares estimates of branch lengths. The next step is to join the two nodes/taxa with the smallest M_{ij} and define the new branch lengths to this node, say u , as

$$l_{iu} = d_{ij}/2 + (r_i - r_j) / (2n - 4)$$

$$l_{ju} = d_{ij} - l_{iu}$$

Next define the new distance from node u to all others as

$$d_{ku} = (d_{ik} + d_{jk} - d_{ij}) / 2$$

Remove nodes i and j , decrease n by one and recalculate r_i , etc. This continues until only two nodes remain and these two are linked with a branch length of $l_{ij} = d_{ij}$.

Another common pairwise clustering algorithm is that due to [Fitch and Margoliash \(1967, Science 155: 279\)](#). This method yields an unrooted tree and unlike the two previous methods it does not proceed by adding taxa one at a time to a growing tree. Rather it has an optimum criterion that must be met. This method attempts to find that tree which minimizes the following sum

$$\sum (d - d')^2 / d^2$$

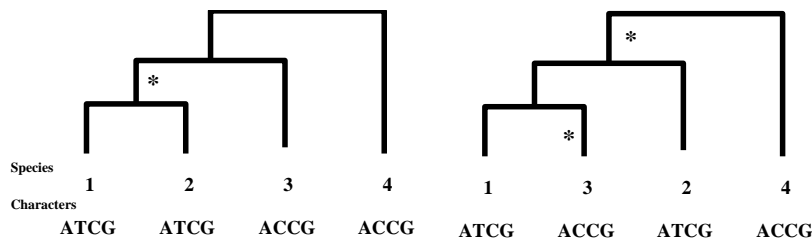
where d is the observed distance and d' is the expected distance given some phylogeny and assuming additivity between all the branch lengths. The details are not given here but are provided in the original paper or in the code and the documentation provided by Dr. Felsenstein.

There are many other methods to reconstruct trees via distance measures. Distance methods are often preferred by people that work in immunology, with frequency data or with data that has some impreciseness in its definition. In addition, almost all of these methods are very rapid and easily permit statistical tests such as bootstraps. These methods loose their accuracy as the number of substitutions goes up and since the correction for multiple substitutions at a single site will loose precision. In this case, the distance methods will increasingly begin to generate less accurate trees. For this reason, with very large trees (where the distance between the most diverged taxa is great) distance methods will do poorly in comparison with methods that are more influenced by local topologies (Rice and Warnow, unpublished).

9.3 Parsimony Methods

Maximum parsimony is perhaps the most popular method for reconstructing ancestral relationships. The method involves evaluating all possible trees (in practice usually only a subset are examined) and giving each a criterion or score that is used to choose between different trees. In maximum parsimony, this criterion is the number of evolutionary changes that need to be postulated in order to explain the observed data with a given tree. The most parsimonious tree is the one with the minimum number of evolutionary changes.

As an example consider the trees shown below.

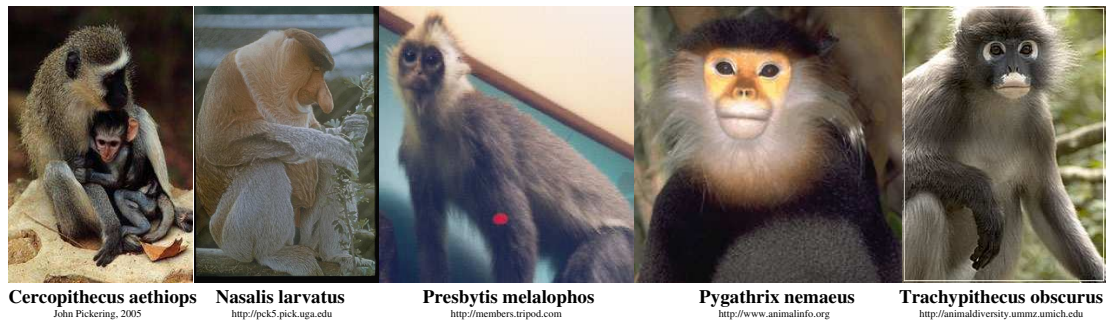


The tree on the left is the most parsimonious tree. It requires only a single evolutionary change (designated by the asterisk) in the second site (a C to T transition). The tree on the right is not as parsimonious. It requires two evolutionary changes. Hence the second tree would be rejected in favour of the first tree.

The principle of the maximum parsimony method is to infer the number of evolutionary events implied by a particular topology and to choose a tree that requires the minimum number of these evolutionary events. In general this means examining a large number of different topologies to search for those that have the minimum changes. For any particular site there are several ways to determine the minimum number of evolutionary events. The Fitch (1971; Syst. Zool. 20:406-416) parsimony criterion is a particularly easy way to count them for nucleotide or amino acid changes. For a particular topology traverse toward the root of the tree. At each node, place the intersection set of the descendant nodes. If this set is empty then place the union set at this node. Continue this for all sites and all nodes. The number of union sets equals the number of events required.

As another example consider the five primate species shown in Table 9.1. These are all old world monkeys. You can collect

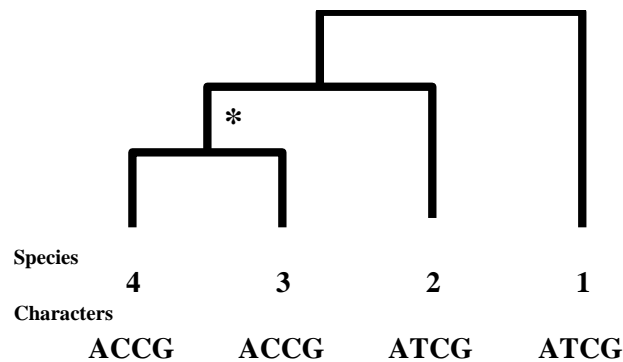
Table 9.1: Results for all fifteen possible trees for five primate taxa.



Phylogeny	Parsimony (dnaps)	MaxLikt (dnaml)	Neighbor Joining (dnadist/neighbor)	Neighbor Joining 1000 Bootstrapped
1 (((Cercopit,Nasalis),Pygathri),Presbyti,Trachypi)	623	-4258	-	5
2 (((Cercopit,Nasalis),Presbyti),Pygathri,Trachypi)	632	-4279	-	0
3 (((Cercopit,Nasalis),Trachypi),Pygathri,Presbyti)	627	-4267	-	1
4 (((Cercopit,Pygathri),Nasalis),Presbyti,Trachypi)	615	-4245	NJ	367
5 (((Cercopit,Pygathri),Presbyti),Nasalis,Trachypi)	611	-4247	-	266
6 (((Cercopit,Pygathri),Trachypi),Nasalis,Presbyti)	623	-4267	-	0
7 (((Cercopit,Presbyti),Nasalis),Pygathri,Trachypi)	631	-4282	-	0
8 (((Cercopit,Presbyti),Pygathri),Nasalis,Trachypi)	617	-4260	-	29
9 (((Cercopit,Presbyti),Trachypi),Nasalis,Pygathri)	613	-4248	-	38
10 (((Cercopit,Trachypi),Nasalis),Pygathri,Presbyti)	631	-4271	-	1
11 (((Cercopit,Trachypi),Pygathri),Nasalis,Presbyti)	634	-4282	-	0
12 (((Cercopit,Trachypi),Presbyti),Nasalis,Pygathri)	617	-4250	-	20
13 (((Nasalis,Trachypi),Cercopit),Pygathri,Presbyti)	629	-4264	-	11
14 (((Pygathri,Trachypi),Cercopit),Nasalis,Presbyti)	643	-4291	-	0
15 (((Presbyti,Trachypi),Cercopit),Nasalis,Pygathri)	619	-4247	-	262

the DNA sequences for their cytochrome oxidase subunit I genes (note that these sequences are too short to provide good phylogenetic information; more sequence data for these species is available but for pedagogical purposes we use only this one gene here). Since there are only five species considered here, there are a total of 15 unrooted trees possible. Parsimony ideally should consider each tree in turn and infer the minimum number of substitutions required to explain the sequence data given the tree. All 15 trees are shown in Table 9.1. The one tree out of all 15 trees with the minimum number is the most parsimonious tree. In this case it is tree #5. Note that in this case, the tree inferred by the neighbor joining method (and the maximum likelihood method; this method as well as bootstrapping will be further discussed below) suggest a different tree is correct. The neighbor joining algorithm suggests that tree #4 is the preferred tree and, unlike parsimony, the algorithm does not consider a metric for all trees but simply returns a single topology.

There are several problems with parsimony methods. First note that the most parsimonious tree may not be unique. Consider the tree



This tree is just as parsimonious as that given above and yet is quite different (so long as only rooted trees are considered). A more serious problem deals with the statistics of these estimators. Suppose that you reconstructed a phylogeny based on a large amount of sequence data and found the most parsimonious tree is one that requires 486 changes. The tree that you prefer (for whatever reason) requires 484 changes. Why and/or when is one phylogeny better than another. This is quite a thorny problem and one of active current research. The best sorts of methods (including methods with algorithms other than parsimony) are those methods that will present you with a whole range of trees that are acceptable via some broad criterion.

Different sites are said to be phylogenetically informative for the parsimony criterion if they provide information that distinguishes between different topologies. Not all sites do this and these sites are, in effect, ignored by the method. Consider characters that are not ordered and can arise via mutation from any other character (such as DNA nucleotides). Then any character that exists uniquely (or locally uniquely) in one OTU is not phylogenetically informative. This is because such a character can always be assumed to have arisen by a single substitution in the immediate branch leading to the OTU in which the character exists. This change is therefore compatible with any topology. A site is phylogenetically informative only when there are at least two different kinds of characters, each represented at least two times. (Remember however, that ALL SITES provide information about the branch lengths - this is true just for the topology).

Note that there are several different kinds of parsimony and the Fitch criterion is only one. As another example, Dollo parsimony is also commonly used. It assumes that derived states are irreversible. That is, a derived character state cannot be lost and then regained. This criterion is most useful when discussing character data other than sequence data. For example if states are complex phenotypes then it is reasonable to assume that these states can evolve only once. Hence, the state can evolve and the state can be lost many times throughout evolution but it cannot be inferred to have evolved twice. An example of such a state in sequence analysis would be restriction sites - these are easier to mutationally lose than to mutationally create. Other parsimony criterion are relaxed Dollo, Wagner, Camin-Sokal, transversion, and generalized.

Many algorithms do not have a series of explicitly stated assumptions required in the derivation of the model and required for its applicability. This is particularly the case with parsimony methods which are often said to be assumption "free". However, the lack of stated assumptions does not mean that no assumptions are necessary for the method to be valid. The assumptions are implicit rather than explicit.

There is a strong bias in parsimony methods when some lineages have experienced rapid rates of change. While this is true of many methods, parsimony methods are particularly sensitive. In general these long branches tend to "attract" each other. Nor do parsimony methods necessarily lead to "correct" trees (nor, for that matter, does any other method). Prof. Felsenstein provides an example of a comparison between four species. The "true" phylogeny is [(A,B),(C,D)], with A and B most closely related and C and D most closely related. If B and D have a more rapid rate of evolution then parsimony will usually generate a tree with [(A,C),(B,D)], with A and C most closely related and B and D related. Indeed after a certain threshold of differential rates is passed, as more and more data are collected (more and more sequences added to the database), parsimony becomes more and more certain that the "correct" tree is [(A,C),(B,D)]. Hence these methods may not be consistent estimators of the phylogeny. Consistency is a term used in statistics that implies convergence of an estimator to the true answer with increasing amounts of data. A maximum parsimony answer will however, converge to a maximum likelihood answer when the rates of evolution along each branch are small (unfortunately this is not true for most data sets). But then, maximum likelihood methods (and Bayesian methods) need not be consistent either. The arguments as to which method is "best" continue (e.g. [Kolaczowski and Thornton, 2004](#) and compare with [Gadagkar and Kumar, 2005](#)).

Parsimony does not require exact constancy of rates of change between branches if the number of substitutions per site is small. If the number of changes per site is large then parsimony methods will make serious errors unless rates are constant between branches. Furthermore, if the total sequence length examined is small and there are a large number of backward and parallel substitutions (as in immunoglobulins) then parsimony has a high probability of producing an erroneous tree even when substitution rates are constant between branches. Also, when the number of substitutions per site is small, a large proportion of the substitutions are autapomorphic and uninformative for constructing a parsimonious tree. In this case, a distance method may perform better since it uses all sites to compute distances.

9.4 Other Methods

9.4.1 Compatibility methods

Another class of methods are known as compatibility methods. The compatibility method assumes that the criterion for choosing between phylogenies should be the number of individual characters /sites that are strictly compatible with a given tree. Two characters are compatible if there exists some phylogeny on which both of these characters could evolve without any state having to arise more than once (no homoplasies).

With multi-state characters there have been methods developed to recode the data and to include knowledge of the ancestral states of characters and from this to determine what changes are compatible. Again compatibility methods are more accurate when there are slow rates of evolutionary change. Both compatibility and parsimony, in effect assume that homoplasies will be rare. If you expect homoplasies to be scattered at random throughout the sequence data, then a parsimony method will perform best. If homoplasies are expected to be concentrated in a few characters, whose identities are known in advance, then compatibility will perform better than parsimony. Nei (1987) notes that the compatibility method and parsimony will give the same answer when the number of OTUs is 5 or less.

9.4.2 Maximum Likelihood methods

The method of maximum likelihood attempts to reconstruct a phylogeny using an explicit model of evolution. Certainly, for this given model of evolution, no other method will perform as well nor provide you with as much information about the tree. Unfortunately, this is computationally difficult to do and hence, the model of evolution must be a simple one. Even with simple models of evolutionary change the computational task is enormous and this is the slowest of all methods.

As a typical model of simple evolutionary change consider a single site in a sequence of nucleotides. Let all sites be selectively neutral and let them spontaneously mutate at a rate μ per gamete per generation. For simplicity, let the mutation rates to and from each nucleotide be equal. Generations are assumed to be discrete and the evolution of each site is assumed to be independent of all other sites. (This may seem like a lot of assumptions but in reality the other methods will not work very well without them either).

Given this model an explicit statement can be made about the probability of change from one nucleotide to another within a specified time period. The probability that a site initially with nucleotide i will change to nucleotide j within time t is P_{ij}^t . The value of P_{ij}^t can be found easily as

$$P_{ij}^t = \delta_{ij}e^{-\mu t} + (1 - e^{-\mu t})g_j$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise and g_j is the equilibrium frequency of nucleotide j .

The likelihood that some site is in state i at the k -th node of an evolutionary tree can be designated by $L_i^{(k)}$. This likelihood can be calculated in a recursive fashion. As an example, consider a simple bifurcating tree with two branches #1 and #2, and with one root node, #3. The time between node #3 and node #1 is t and the time separating node #3 and #2 is t' . With these definitions the likelihood of having the i -th nucleotide at node #3 in an evolutionary tree can be found as

$$L_i^{(3)} = \left(\sum_{j=1}^4 P_{ij}^t L_j^{(1)} \right) \left(\sum_{k=1}^4 P_{ik}^{t'} L_k^{(2)} \right)$$

(Felsenstein, 1981). The terms $L_j^{(1)}$, $L_k^{(2)}$ designate the likelihoods of states j and k in the nodes or taxa #1 and #2. If nodes #1 and #2 designate extant species then these likelihoods are known explicitly. The likelihoods are either 1 or 0 depending on whether the extant species does or does not have that nucleotide at that particular site. In words, you calculate the probability that the descendant would end up having nucleotide j given that t generations in the past it had nucleotide i and multiply this by the likelihood that the descendant had nucleotide j . Sum this for all possible nucleotides and do the same for the other branch on the tree. Take the product of the two to give you the likelihood of the tree up to this point.

This information determines $L_i^{(3)}$. In more complicated phylogenies with more than two species the likelihoods of interior nodes can be calculated in a similar fashion, recursively. In this case identify node #3 with a more ancient bifurcation and nodes #1 and #2 with bifurcations that in turn give rise to more species. Begin at the tips of the phylogeny and move down the tree one node at a time. Each successive step uses the likelihoods just calculated (such as the value determined for $L_i^{(3)}$) to find the likelihood of the next node. The likelihood of every state is calculated for every node using those likelihoods calculated for the previous nodes. This continues until the root of the tree is reached and then the overall likelihood is found by summing the products of the root likelihoods with the prior probabilities of each state. Without any further information, the prior probabilities of each state are usually taken to be their equilibrium frequencies.

Since each site evolves independently, the likelihood of a phylogeny can be calculated separately for each site. The product of the likelihoods for each site provides the overall likelihood of the observed data. To maximize the likelihood different values of u are analyzed until a set of branch lengths/substitution rates are found which provide the highest likelihood of observing the actual sequences. Finally different tree topologies are searched to find the best one.

Note that a likelihood is not quite the same thing as the probability of observing the given sequences and nor are likelihoods the same thing as a probability. For example, a set of maximum likelihoods need not sum to one. In general, you would normally have the probability of some observed data as a function of some parameter (here the parameters are the branch lengths/substitution rates). The likelihood function turns this relationship around. Instead of considering this to be a set of probabilities for alternative observations given some parameter, it considers the data as fixed and the likelihood as a function of the parameters. For more information on the powerful abilities of likelihood methods consult a text on probability.

9.4.3 Method of Invariants

The method of invariants was originally suggested by Lake (1987) under the name “evolutionary parsimony”. This is probably a poor use of terminology since it holds little relationship to parsimony and is a distinct method. Others have termed the approach “invariants”.

The idea behind this method is quite simple. Basically there are some patterns which are not functions of branch lengths and depend only on the topology of the tree. These are therefore invariant to the difficulties caused by different rates of evolution along each branch. Lake considered only transversions (since he was originally trying to decipher an ancient branch point and transversions occur less frequently than do transitions) and derived a set of linear invariants for transversions. These are equations for each site that should be zero for the incorrect tree topology. The values for every site are summed and he used a Chi-square to determine those values that differ significantly from zero. This defines the most probable tree topology. The method was originally proposed for four species with just two states (R or Y) per site. This is what will be discussed here. See PHYLIP below for a more general discussion.

With four species there are only three unrooted topologies that need be considered

$$((A, B), (C, D)), \quad ((A, C), (B, D)) \quad \text{and} \quad ((A, D), (B, C)).$$

If say the first tree was the correct tree then a pattern of nucleotides that would support this tree and be phylogenetically informative, would be $((R, R), (Y, Y))$. Due to the stochastic nature of mutations not all sites will have this pattern. Some sites will not support the correct tree. For example $((R, Y), (R, Y))$ supports the second topology above. Lake reasoned that some functions of the “support” for each tree might be found that would depend only on topology and not on branch lengths. The number of sites that have any particular pattern are tabulated. Let X and Y denote two different purines and Z and W denote two different pyrimidines. Then the following equations are Lake’s (1987) invariants.

$$(X X Z Z + X Y Z W) - (X X Z W + X Y Z Z)$$

$$(X Z X Z + X Z Y W) - (X Z X W + X Z Y Z)$$

$$(XZZX + XZWY) - (XZWX + XZZY)$$

For whatever the correct topology is, one of these equations should be different from zero and the other two should be equal to zero (or close to it due to random events). The significance of all of the scores can be tested via a Chi-square or via an exact binomial test.

For the topology $((A, B), (C, D)) \dots$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ x \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ y \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ y \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ x \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ x \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ y \end{array} \right) \neq 0$$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} y \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ y \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} y \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \right) = 0$$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ y \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} w \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ y \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} w \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ x \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ y \end{array} \right) = 0$$

For the topology $((A, C), (B, D)) \dots$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} y \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ w \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} y \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \right) = 0$$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ x \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ y \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ x \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ y \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \right) \neq 0$$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ y \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ w \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ x \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ y \end{array} \right) = 0$$

For the topology $((A, D), (B, C)) \dots$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} y \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ w \end{array} \begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} y \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \right) = 0$$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ y \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ w \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ x \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ y \end{array} \right) = 0$$

$$\left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ x \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad z \\ \diagup \quad \diagdown \\ w \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ y \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ w \end{array} \right) - \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ w \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad x \\ \diagup \quad \diagdown \\ x \end{array} \right) + \left(\begin{array}{c} x \\ \diagdown \quad \diagup \\ \quad w \\ \diagup \quad \diagdown \\ z \end{array} \begin{array}{c} z \\ \diagdown \quad \diagup \\ \quad y \\ \diagup \quad \diagdown \\ z \end{array} \right) \neq 0$$

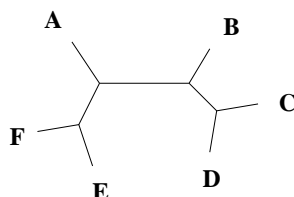
While this method was suggested for only four species there have been several extensions suggested to develop it further and apply it to more than four species. Similarly there are extensions to consider not only transversions but to consider in complete generality all four nucleotides. [Steel and Fu \(1995 J Comput Biol 2:39\)](#), [Fu \(1995 J. Theo. Biol. 173: 339\)](#), [Cavender \(1991 Math Biosci 103:69\)](#), [Felsenstein \(1991 J. Theo. Biol. 152:357\)](#), and [Sankoff \(1990 Mol. Biol. Evol. 7:255\)](#), have developed quadratic and higher order invariants (or in Sankoff's words "made to order invariants"). These extensions promise that invariants will be a very useful tool in the future since these methods are dependent only on the branching order.

9.4.4 Quartet Methods

In principle if the taxa from a tree are reduced to four taxa trees, the original tree can be reconstructed from these quartets. This is the idea behind a collection of quartet methods. Quartet puzzling was suggested in a paper by Strimmer and von Haeseler (1996; Mol. Biol. Evol. 13:964-969). Their method made use of a maximum likelihood algorithm to construct the individual quartets but any algorithm that is preferred can be used for this step. Because there are only three possible

trees for four species, the total number of trees that need be constructed are only $3 \times \binom{n}{4}$ for n taxa. This step is therefore quite feasible even for large n . For example, with $n = 20$ there are only $3 \times \binom{20}{4} = 14535$ trees that need be constructed.

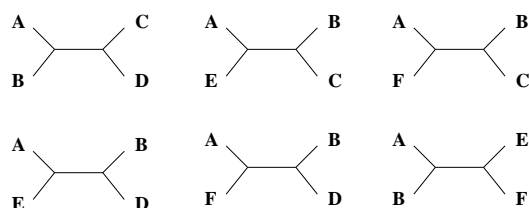
In principle, these quartets should uniquely determine the topology of the tree. For a tree with six taxa



there are $\binom{6}{4} = 15$ quartets. These are with taxa

ABCD	ABCE	ABCF	ABDE	ABDF
ABEF	ACDE	ACDF	ACEF	ADEF
BCDE	BCDF	BCEF	BDEF	CDEF

The quartets corresponding to the above tree are



and so on.

In practice the various quartets may not agree with other and some method must be chosen to weight their suggested topologies. Strimmer and von Haeseler (1996) used a method that weighted the three topologies (1,0,0), (0,1,0) or (0,0,1). They then choose four taxa at random, and began adding taxa one at a time to this four taxa tree according to the quartets. As an example, if there are four taxa (A, B, C, D) initially and if taxa E has a quartet such that ((A,B),(C,E)) then E should not be placed on branch leading to A or B. If it has a quartet ((A,D),(B,E)) then it should not be placed on a branch leading to A or D. Running through all quartets containing E a score is kept for all branches and the branch point with the minimum score is chosen as the branch point to place taxa E. The next taxa is chosen and treated in the same way and then the next taxa.

The order in which the taxa are added and the initial taxa chosen to start the process will critically influence the resulting tree. To prevent any bias due to the order, this whole process is done multiple times with random choices for the order of taxa. A majority rule consensus tree is then chosen as the final tree. This also means that a measure of variability is immediately available in the form of how many times a particular group of taxa branched together. Note that this measure is not the same as a bootstrap value and does not necessarily have the same statistical properties.

The quartet methods are useful for their comparative speed. A maximum likelihood algorithm can be applied with this algorithm to problems that would otherwise not be feasible. As a result, Strimmer and von Haeseler were able to show that this method obtained results as good as neighbor joining when the data was well behaved and results better than neighbor joining when the data had large variations in branch length (a situation where likelihoods are known to do better). The method performed only slightly worse than Felsenstein's complete maximum likelihood method.

Quartet methods are also interesting in their ability to separate each of the individual steps and to then easily permit the incorporation of improvements in each step. For example, the original quartets can be constructed via any algorithm. The algorithm know to give the best results for a particular data set can then be chosen (or one known to be most robust under the broadest variety of circumstances). The construction of the tree from the quartets is a completely separate step that can be optimized as well. In a subsequent paper Strimmer, Goldman and von Haeseler (1997; Mol. Biol. Evol. 14:210-211) study the influences of different weights for each quartet and develop a discrete weighting which is both efficient

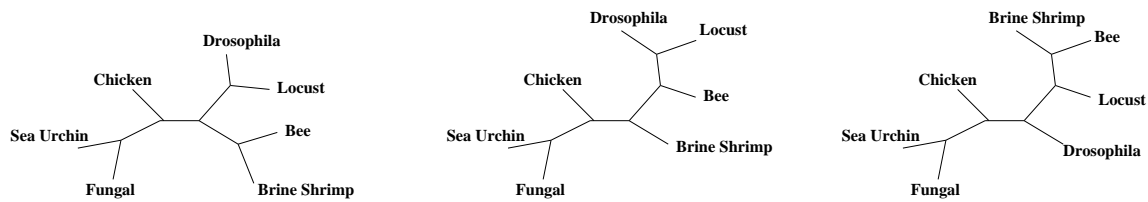


Figure 9.2: A quartet puzzling tree for eight NAD5 mitochondrial proteins. These are three of the 15 trees that quartet puzzling considers unresolved for this data.

and improves that accuracy of the trees reconstructed. There are similarly a variety of methods possible to reconstruct consensus trees from multiple trees.

9.5 Consensus Trees

The goal of phylogenetics is to reconstruct the true tree reflecting the genetic history of groups of organisms. The true history, however, is unattainable. All we can hope to find are a collection of more or less likely trees and then only for the single gene(s) sequence under consideration. Many methods of reconstructing evolutionary relationships will generate multiple possibilities for this history. Parsimony methods, for example, usually obtain several or even many trees of the same, minimum length. Minimum spanning networks connecting a set of haplotypes regularly define many, often hundreds of equivalent trees.

Uncertainty in phylogenetic relationships also results in multiple trees. An example of multiple trees generated by quartet puzzling is shown in Figure 9.2. Statistical uncertainty arises because any set of sequences are a sample of only a finite number of sites and only one of a number of possible evolutionary events. It is inconceivable that the same process starting from the same ancestral sequence would lead twice to exactly the same result. Methods of reconstructing history must account for all conceivable paths. Reconstruction is therefore inherently probabilistic, leading only to a set of possible histories and the single true history.

Faced with conflicting alternatives, our response (in true Canadian fashion) is usually to reach some sort of consensus. It is perhaps no simpler in phylogenetics than it is in human affairs. There are not one, but many ways to find consensus trees. The PHYLIP package provides a program `consense` that will do strict consensus (a group of species must be present in all species) as well as a family of majority rule consensus tree methods labeled the M_l (M-sub-L) methods. These allow the majority rule consensus tree to consist of any percentage level between 50% and 100%. Thus, groups that occur frequently are merged into a consensus tree until uncertainties in this tree are no longer resolved.

In addition, the default method of consensus in PHYLIP is an extended majority method that finds the 50% majority rule consensus tree and then continues to add groups with a lower frequencies as long as they do not conflict higher frequency groups.

9.6 Bootstrap trees

Assessing the significance of phylogenetic trees has been a controversial problem. One method that has proved useful is the bootstrap. This method of statistical inference was invented by Bradley Efron in 1977. A popular account is given in the article by P. Diaconis and B. Efron *Scientific American* 248: 116-130, 1983. Bootstrap statistics do not require assumptions about the underlying sample distribution. Further, they can be applied to a variety of different properties of samples beyond the traditional mean, variance and correlation.

The concept of a bootstrapped statistic depends on the concept that repeated samples are assumed to produce an accurate distribution of the data if a new data set were collected from the population. Hence, the data itself is assumed to accurately reflect the variation that might be present in the population. By repeatedly sampling (with replacement) the sample itself, you can obtain an understanding of the effect that this level of variation might have on any statistic that you might be

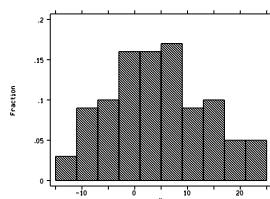


Figure 9.3: A sample of 100 data points from an underlying normal distribution

Table 9.2: Ten samples of bootstrapped data ($n = 100$ in each case)

Sample	Mean	Std.Dev.	5%	95%
1	5.5	9.71	-11.3	21.2
2	4.5	9.39	-9.0	21.2
3	5.1	9.78	-10.5	24.5
4	4.4	9.22	-9.0	22.4
5	4.6	8.92	-9.6	19.0
6	5.5	8.36	-9.6	17.7
7	4.6	9.34	-9.5	22.2
8	3.0	9.45	-11.3	20.8
9	4.0	9.71	-8.5	22.4
10	5.4	8.42	-8.6	20.8

interested in. If theoretical knowledge of the statistics is available then it should be used in preference to a bootstrap. If the data itself is biased, then bootstraps tend to exaggerate this bias and again bootstraps should either not be used or corrected for the suspected bias. The bootstrap is advantageous when there is no knowledge of the true statistical properties such as when the underlying distribution is unknown. This is the case in phylogenetic studies.

To illustrate, consider the problem of estimating 95% confidence limits on the mean. Suppose we draw 100 samples from a population with a underlying normal distribution with mean $\mu = 5$ and standard deviation $\sigma = 10$ (a graph of this data is given in Figure 9.3). Due to sample effects, the mean of this data is $\bar{x} = 4.3$ and its standard deviation is $s = 9.15$. To estimate 95% confidence limits for the mean, we then sample with replacement each value of x until a hundred new samples have been drawn. The statistic of interest (mean, median, confidence limits, or whatever else is of interest) is recalculated for this data set and then the whole process is repeated (here we will repeat it ten times but in practice bootstraps **must** be repeated with much larger sample numbers; on the order of 1000 or more). The statistics from ten potential samples from the original $n = 100$ data set are shown in the accompanying Table 9.2. For each repeated sample statistics are calculated just as one would traditionally with the original data set. These can then be combined to yield bootstrapped estimates. Bootstrap statistics allow an alternate estimate of the 95% confidence interval. For example, in this case, the 10% confidence limit on the mean would be calculated from the values in Table 9.2 and would be 3.0 and the 90% confidence limit on the mean would be 5.5 (of course based on such a limited sample size of 10, neither is a useful estimate).

Felsenstein first recommended applying the bootstrap method to phylogenies. A useful review of this and other methods of assessing the reliability of phylogenies is given in J. Felsenstein *Annu. Rev. Genet.* 22: 521-565, 1988. Again, the idea is resample the sequence data (with replacement) site by site to construct a new sequence data set of the same length as the original and then to estimate a set of bootstrap trees. The original sequences (x) are used to estimate a distance matrix (D) by some method that measures differences between sequence pairs. This distance matrix is converted into a tree by an algorithm that connects sequence pairs into an unrooted, bifurcating tree (T). Alternatively the tree (T) can be obtained directly from the sequences by parsimony (or your method of choice). Felsenstein's method is to randomly sample sites

Original data set (x)	
Species	Sites
	abcdefghijklmnop . . .
1	ATACCAGCAC . . .
2	ATACCAACAC . . .
3	ATACCGGGAT . . .
4	ATACCCGAAA . . .

Bootstrap data set (x^1)	
Species	Sites
	abbjigccfb . . .
1	ATTCAGAAAT . . .
2	ATTCAAAAAT . . .
3	ATTTAGAAGT . . .
4	ATTAAGAACT . . .

Bootstrap data set (x^2)	
Species	Sites
	aafegghiaa . . .
1	AAACGGCAAA . . .
2	AAACAACAAA . . .
3	AAGCGGGAAA . . .
4	AACCGGAAAA . . .

Figure 9.4: Bootstrapped data sets are made by randomly sampling nucleotide (or amino acid) sites with replacement. Some sites may, therefore, be omitted altogether from some of the bootstrap samples.

(columns of sequence set x) from the sequence data with replacement to form a bootstrap data set (x^y ; Figure 9.4). The original algorithm is then applied to this data to yield a bootstrap tree (T^y). Repeated bootstrap samples yield a set of bootstrap trees T^1, T^2, \dots, T^n . These trees are derived from sequences containing representative sites sampled from the actual data. The assumption of this method is that the sampled sites are independent of one another and representative of what the evolutionary process would produce if repeated.

There are many ways in which the bootstrap set of trees could be used to answer questions about significance. Felsenstein suggested that the significance of phylogenetic relationships could be assessed from their frequency of occurrence in the bootstrap set T^1, T^2, \dots, T^n . More specifically, suppose one wanted to know if a subgroup of taxa (called G) were monophyletic (exclusively comprised of descendants of a common ancestor). We determine the fraction of trees in the bootstrap set T^1, T^2, \dots, T^n in which G is, in fact, monophyletic (call this F_G). Obviously if F_G is small there is little support from monophyly while if F_G is close to 100% we feel that a monophyletic grouping is more likely. Felsenstein recommended that $F_G \geq 95\%$ be considered as significant support of a monophyletic relationship.

Analysis of NAD5 mitochondrial genes illustrates this method of using bootstrap trees to determine phylogenetic significance. NAD5 is one of the larger proteins encoded within the mitochondria and hence it is easy to obtain the DNA and to sequence this gene. Phylogenetic relationships for four diverged examples of the NAD5 sequences were determined. Bootstrap DNA sequences (PHYLIP: [seqboot](#)) were used to determine pairwise distance matrices (PHYLIP: [dnadist](#), Kimura 2-parameter, $T_s/T_v = 2.0$) and the Fitch-Margoliash, least squares method (PHYLIP: [fitch](#)) for merging taxa was then

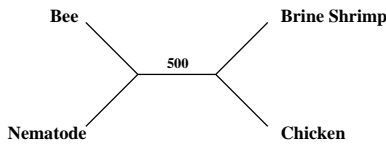


Figure 9.5: The consensus of 500 bootstrapped trees based on *NAD5* gene sequences

used to make a set of 500 bootstrap trees. The consensus of all 500 of these trees grouped the brine shrimp with the chicken and grouped the bee with the nematode (Figure 9.5). Hence, in this case the support is clear and unequivocal for this one of the three possible relationships.

Although bootstrap methods are widely used, there is considerable debate over the measurement of the actual level of statistical validity. Many have criticized that the 95% criterion is too conservative and accept smaller F_G values (e.g. $\geq 80\%$) as indicating significant monophyletic grouping. Support for this less conservative interpretation of F_G comes from simulation and theoretical work. However, some considerations in favor of a more conservative approach are the following.

1. The consensus tree itself is often used to suggest monophyletic groups. Thus, these are not tested a priori. It is possible that phylogenetically uninformative data may sometimes generate groupings by chance. Since it is not clear how often this could happen, it is best to be conservative and demand a larger value of F_G .
2. The consensus tree effectively tests all possible groups formed from the set of taxa. A few such groups could reach high F_G just by chance.
3. If the group G is actually monophyletic, the value of F_G estimates the probability that it would appear as monophyletic in sequence data of the type obtained. It is not a confidence interval. It determines the stability of the phylogenetic relationship if more sequence data of the same type were to be accumulated.
4. Bootstrap statistics are assumed to vary smoothly and continuously in sample space. However, a taxonomic group can either be monophyletic or not monophyletic. Methods to extend bootstrap methods to bivariate statistics have not been developed.
5. The bootstrap method assumes that sites are independent and that a sufficient number have been sampled to give a complete representation of the evolutionary process.

Finally, the bootstrap trees are only as good as the method that generates them. Again, if the method is biased, the trees will not be representative examples of evolution. In particular, the trees generated for the *NAD5* genes are certainly biased (and were purposely used to illustrate this point – no, bees and nematodes are not close relatives). The DNA distance method not only did not account for rate variation among sites (e.g. first, second and third codon positions), but also ignored large base composition differences between sequence pairs. The substitution model for DNA distance assumes that all sequences are subject to the same mutational forces leading to equilibrium nucleotide frequencies of 25% for each of the four nucleotides. This is certainly not the case for the *NAD5* genes. Thus the method artificially makes sequences of similar nucleotide composition (e.g. bee and nematode) closer because the expected number of substitutions is underestimated. The significant grouping of these taxa is entirely a result of this bias. Indeed, otherwise this would provide strong evidence that the brine shrimp and the bee do not form a monophyletic grouping commonly known as arthropods. Steel *et al.* Nature 364: 440-442, 1993 have discussed a randomization test that corrects for such nucleotide biases and can be used to show that the evidence for the association displayed in Figure 9.5 is due to the nucleotide bias and is not an accurate phylogenetic reconstruction.

9.7 Warnings

Remember that each of these methods have their advantages and their disadvantages. They provide estimates of what the phylogenetic history of the sample may be like - they do not provide “truth”. When you run an algorithm for your data set, consider this simply as the starting point of your analysis.

There are several approaches that can be taken to begin an in depth phylogenetic analysis of your data. These are a few suggestions but they are not exhaustive - for different data sets additional steps should be taken.

1. The first rule that should be followed is to apply several different algorithms to your data set. Each one will provide a different picture of the phylogenetic history reflecting the assumptions of the methods.
2. Your data should be bootstrapped or jackknifed to sample your data. These are techniques to create new data sets either by sampling with replacement from the original set or, in the case of a jackknife, by successively dropping individual data points. They will help to determine how sensitive the phylogenetic history is to changes in the data set. The actual statistics for these cases are non-standard and difficult to calculate (ie: 95% bootstrap support does not necessarily imply that one should expect a 95% probability that that clade is correct) but it will provide a rough measure of variability.
3. If the data and tree inference technique were ideal, analysing any two subsets of taxa would yield congruent trees (i.e., the trees would be identical after pruning taxa absent from one or both trees). Try this and see what happens for different subsets.
4. In this regard, if the tree changes dramatically when a single OTU is dropped this is usually an indication that that OTU is causing systematic errors (such as would be caused by a significantly different rate of change).
5. Worry about long unbranched lineages and any subtrees on either side of long branches. Long branches tend to attract each other !!!
6. Remember that these are gene trees and hence the trees from different genes may or may not be the same. If your taxa are each sufficiently diverged then the trees should be similar. If not then check for non-orthologous genes, check for lateral gene transfer or for other events that would cause systematic errors.
7. Always include more than one outgroup taxa. In this way you can check that the outgroups are indeed "out".
8. If possible choose your outgroup species such that they are evenly spaced on the tree. You will obtain more reliable information from these. Two outgroups that are closely related to each other will not add much information.
9. Even if you are interested in the relationships of just a few taxa it is best to include as many intermediate taxa as possible. These will help to highlight the multiple substitutions that confound any analysis.
10. Others have suggested that because large branch lengths confound many methods, one should limit an analysis to those sequence regions that exclude the most variable positions. (I personally disagree with this rule of thumb but hey ..!)

9.8 Available Packages

The following section of these notes will provide you with some background to the package of programs distributed by Dr. Felsenstein - the **PHYLIP** package. These are excellent programs, work on any platform, they are free (!) and they are easily obtained.

They are not however unique. A list of other phylogenetic reconstruction programs is maintained by Dr. Felsenstein and parts of it are reproduced here. A listing from Prof. Felsenstein's PHYLIP (see <http://evolution.genetics.washington.edu/phylip.html>) homepage includes

- General-purpose packages
 - PHYLIP
 - PAUP*
 - MEGA
 - VOSTORG
 - Fitch programs
 - Phylo_win
 - ARB
 - DAMBE
 - PAL
 - Bionumerics
- Parsimony programs
 - PAUP*

- Hennig86
- MEGA
- Tree Gardener
- RA
- Nona
- PHYLIP
- TurboTree
- Freqpars
- Fitch programs
- CAFCA
- Phylo_win
- sog
- gmaes
- LVB
- GeneTree
- TAAR
- ARB
- DAMBE
- MALIGN
- POY
- DNASEP
- SEPAL
- Gambit
- TNT
- GelCompar II
- Bionumerics
- TCS

- Distance matrix methods

- PHYLIP
- PAUP*
- MEGA
- MacT
- ODEN
- Fitch programs
- ABLE
- TREECON
- DISPAN
- RESTSITE
- NTSYSpc
- METREE
- TreePack
- TreeTree
- GDA
- Hadtree, Prepare and Trees
- GCG Wisconsin Package
- SeqPup
- PHYLTEST
- Lintre
- WET
- Phylo_win
- njbafd
- Gambit
- gmaes
- DENDRON

- Molecular Analyst Fingerprinting

- BIONJ
- TFPGA
- MVSP
- SOTA
- ARB
- BIOSYS-2
- Darwin

- T-REX
- sendbs
- nneighbor
- DAMBE
- weighbor
- QR2
- DNASIS
- minspnet
- PAL
- Arlequin
- vCEBL
- HY-PHY
- Vanilla
- GelCompar II
- Bionumerics
- qclust
- TCS

- Computation of distances

- PHYLIP
- PAUP*
- RAPDistance
- MULTICOMP
- MARKOV
- RSVP
- Microsat
- DIPLOMO
- OSA
- DISPAN
- RESTSITE
- NTSYSpc
- TREE-PUZZLE
- Hadtree, Prepare and Trees
- GCG Wisconsin Package
- AMP
- GCUA
- DERANGE2
- POPGENE
- TFPGA
- REAP
- MVSP
- SOTA
- RSTCALC
- Genetix
- BIOSYS-2
- RAPD-PCR package
- DISTANCE
- Darwin
- sendbs
- K2WuLi
- GeneStrut
- Arlequin
- DAMBE
- DnaSP
- PAML
- puzzleboot
- MATRIX
- PAL
- Sequencer
- Vanilla
- GelCompar II
- Bionumerics
- qclust

- Maximum likelihood and related methods

- PHYLIP

- PAUP*
 - fastDNAMl
 - MOLPHY
 - PAML
 - Spectrum
 - SplitsTree
 - PLATO
 - TREE-PUZZLE
 - Hadtrees, Prepare and Trees
 - SeqPup
 - Phylo_win
 - PASSML
 - ARB
 - Darwin
 - BAMBE
 - DAMBE
 - Modeltest
 - TreeCons
 - VeryfastDNAMl
 - PAL
 - dnrates
 - TrExMI
 - HY-PHY
 - Vanilla
 - MEGA
 - Bionumerics
 - fastDNAMlRev
 - RevDNARates
 - rate-evolution
 - MrBayes
 - Hadtrees, Prepare and Trees
 - CONSEL
- Quartets methods
 - TREE-PUZZLE
 - STATGEOM
 - SplitsTree
 - PHYLTEST
 - GEOMETRY
 - PICA95
 - Darwin
 - PhyloQuart
 - Willson quartets programs
 - Gambit
 - Artificial-intelligence methods
 - SOTA
 - Invariants (or Evolutionary Parsimony) methods
 - PHYLIP
 - PAUP*
 - Interactive tree manipulation
 - MacClade
 - PHYLIP
 - PDAP
 - TreeTool
 - ARB
 - WINCLADA
 - TreeEdit
 - UO
 - TreeExplorer
- TreeThief
 - RadCon
 - Mavric
- Looking for hybridization or recombination events
 - PLATO
 - Bootscanning Package
 - TOPAL
 - reticulate
 - RecPars
 - partimatrix
 - homoplasy test
 - LARD
 - Network
 - TCS
 - Bootstrapping and other measures of support
 - PHYLIP
 - PAUP*
 - PARBOOT
 - ABLE
 - Random Cladistics
 - AutoDecay
 - TreeRot
 - RASA
 - DNA Stacks
 - OSA
 - DISPAN
 - TreeTree
 - PHYLTEST
 - Lintre
 - sog
 - njbafd
 - MEGA
 - PICA95
 - ModelTest
 - TAXEQ2
 - BIOSYS-2
 - RAPD-PCR package
 - TreeCons
 - BAMBE
 - DAMBE
 - puzzleboot
 - CodonBootstrap
 - DNASEP
 - SEPAL
 - Gambit
 - MEAWILK
 - TrExMI
 - Sequencer
 - PAL
 - PHYCON
 - MrBayes
 - CONSEL
 - Compatibility analysis
 - COMPROB
 - PHYLIP
 - PICA95
 - reticulate
 - partimatrix
 - SECANT
 - CLINCH
 - MEAWILK

- Consensus trees and distances between trees
 - COMPONENT
 - TREEMAP
 - NTSYSpc
 - PHYLIP
 - PAUP*
 - REDCON
 - TAXEQ2
 - TreeCons
 - QUARTET2
 - RadCon
- Tree-based sequence alignment
 - TreeAlign
 - ClustalW
 - MALIGN
 - GeneDoc
 - GCG Wisconsin Package
 - TAAR
 - Ctree
 - DAMBE
 - POY
 - ALIGN
 - DNASIS
- Biogeographic analysis and host-parasite comparison
 - COMPONENT
 - TREEMAP
- Comparative method analysis
 - PHYLIP
 - CAIC
 - COMPARE
 - PA
 - CMAP
 - CoSta
 - PDAP
 - ACAP
 - ANCML
 - RIND
 - MacroCAIC
 - Fels-Rand
 - Phylogenetic Independence
- Simulation of trees or data
 - COMPONENT
 - Bi-De
 - SEQEVOLVE
 - TheSiminator
 - Seq-Gen
 - Treevolve and PTreevolve
 - PSeq-Gen
 - COMPARE
 - ROSE
 - PAML
 - ProSeq
 - PAL
 - Vanilla
- Examination of shapes of trees
 - End-Epi
 - MacroCAIC
- Genie
- PAL
- Vanilla
- RadCon
- BRANCHLENGTH
- Clocks, dating and stratigraphy
 - StratCon
 - QDate
 - Diversi
 - K2WuLi
 - Modeltest
 - PAML
 - TipDate
 - RRTree
 - vCEBL
 - TreeEdit
 - HY-PHY
 - PAL
 - rate-evolution
 - BRANCHLENGTH
- Description or prediction of data from trees
 - CONSERVE
 - TreeDis
- Tree plotting/drawing
 - PHYLIP
 - PAUP*
 - TreeTool
 - TreeView
 - Fitch programs
 - NJplot
 - DendroMaker
 - Tree Draw Deck
 - Phylodendron
 - ARB
 - unrooted
 - DAMBE
 - TREECON
 - Mavric
 - TreeExplorer
 - TreeThief
 - Bionumerics
- Sequence management/job submission
 - PARBOOT
 - Random Cladistics
 - Tree Gardener
 - GDE
 - MUST
 - DNA Stacks
 - SeqPup
 - ARB
 - BioEdit
 - Singapore PHYLIP web interface
 - PHYCON
 - Bionumerics
- Teaching about phylogenies
 - Phylogenetic Investigator

9.9 PHYLIP

This is the package of programs distributed by Professor Felsenstein. It is distributed free and Joe is a very friendly character and can help with whatever problem you might have (but carefully read the documentation before contacting him). I have reproduced parts of the documentation here but I urge you to get your own copy of the programs so that Dr. Felsenstein can know how many copies are out there and can update/modify programs etc. Also if you do use these programs in a publication you must quote Dr. Felsenstein (the same applies for any other program obtained from the file servers). Remember that the value of a scientist's work is often measured by quotations and if you use someone's programming work you should quote it just as you would quote their experimental work.

The **PHYLIP** package is distributed for free. Programs are written in a standard subset of "C" and the source code is provided with the package. You can reach Dr. Felsenstein at joe@genetics.washington.edu and the complete package can be obtained via anonymous ftp from [evolution.genetics.washington.edu](ftp://evolution.genetics.washington.edu).

9.9.1 PHYLIP Contents

On the following pages you will find extracts of the documentation for the **PHYLIP** package of programs. The complete documentation is not reproduced - you should get your own official copy.

What The Programs Do

Here is a short description of each of the programs. For more detailed discussion you should definitely read the documentation file for the individual program and the documentation file for the group of programs it is in. In this list the name of each program is a link which will take you to the documentation file for that program. Note that there is no program in the PHYLIP package called PHYLIP.

PROTPARS

Estimates phylogenies from protein sequences (input using the standard one-letter code for amino acids) using the parsimony method, in a variant which counts only those nucleotide changes that change the amino acid, on the assumption that silent changes are more easily accomplished.

DNAPARS

Estimates phylogenies by the parsimony method using nucleic acid sequences. Allows use the full IUB ambiguity codes, and estimates ancestral nucleotide states. Gaps treated as a fifth nucleotide state. Can use 0/1 weights, reconstruct ancestral states, and infer branch lengths.

DNAMOVE

Interactive construction of phylogenies from nucleic acid sequences, with their evaluation by parsimony and compatibility and the display of reconstructed ancestral bases. This can be used to find parsimony or compatibility estimates by hand.

DNAPENNY

Finds all most parsimonious phylogenies for nucleic acid sequences by branch-and-bound search. This may not be practical (depending on the data) for more than 10 or 11 species.

DNACOMP

Estimates phylogenies from nucleic acid sequence data using the compatibility criterion, which searches for the largest number of sites which could have all states (nucleotides) uniquely evolved on the same tree. Compatibility is particularly appropriate when sites vary greatly in their rates of evolution, but we do not know in advance which are the less reliable ones.

DNAINVAR

For nucleic acid sequence data on four species, computes Lake's and Cavender's phylogenetic invariants, which test alternative tree topologies. The program also tabulates the frequencies of occurrence of the different nucleotide patterns. Lake's invariants are the method which he calls "evolutionary parsimony".

DNAML

Estimates phylogenies from nucleotide sequences by maximum likelihood. The model employed allows for unequal expected frequencies of the four nucleotides, for unequal rates of transitions and transversions, and for different (prespecified) rates of change in different categories of sites, with the program inferring which sites have which rates. It also allows different rates of change at known sites.

DNAMLK

Same as DNAML but assumes a molecular clock. The use of the two programs together permits a likelihood ratio test of the molecular clock hypothesis to be made.

PROML

Estimates phylogenies from protein amino acid sequences by maximum likelihood. The PAM or JTT models can be employed. The program can allow for different (prespecified) rates of change in different categories of amino acid positions, with the program inferring which positions have which rates. It also allows different rates of change at known sites.

DNADIST

Computes four different distances between species from nucleic acid sequences. The distances can then be used in the distance matrix programs. The distances are the Jukes-Cantor formula, one based on Kimura's 2-parameter method, Jin and Nei's distance which allows for rate variation from site to site, and a maximum likelihood method using the model employed in DNAML. The latter method of computing distances can be very slow.

PROTDIST

Computes a distance measure for protein sequences, using maximum likelihood estimates based on the Dayhoff PAM matrix, Kimura's 1983 approximation to it, or a model based on the genetic code plus a constraint on changing to a different category of amino acid. Rate variation from site to site is also allowed. The distances can be used in the distance matrix programs.

RESTDIST

Distances calculated from restriction sites data or restriction fragments data. The restriction sites option is the one to use to also make distances for RAPDs or AFLPs.

RESTML

Estimation of phylogenies by maximum likelihood using restriction sites data (not restriction fragments but presence/absence of individual sites). It employs the Jukes-Cantor symmetrical model of nucleotide change, which does not allow for differences of rate between transitions and transversions. This program is very slow.

SEQBOOT

Reads in a data set, and produces multiple data sets from it by bootstrap resampling. Since most programs in the current version of the package allow processing of multiple data sets, this can be used together with the consensus tree program CONSENSE to do bootstrap (or delete-half-jackknife) analyses with most of the methods in this package. This program also allows the Archie/Faith technique of permutation of species within characters.

FITCH

Estimates phylogenies from distance matrix data under the "additive tree model" according to which the distances are expected to equal the sums of branch lengths between the species. Uses the Fitch-Margoliash criterion and some related least squares criteria. Does not assume an evolutionary clock. This program will be useful with distances computed from molecular sequences, restriction sites or fragments distances, with DNA hybridization measurements, and with genetic distances computed from gene frequencies.

KITSCH

Estimates phylogenies from distance matrix data under the "ultrametric" model which is the same as the additive tree model except that an evolutionary clock is assumed. The Fitch-Margoliash criterion and other least squares criteria are assumed. This program will be useful with distances computed from molecular sequences, restriction sites or fragments distances, with distances from DNA hybridization measurements, and with genetic distances computed from gene frequencies.

NEIGHBOR

An implementation by Mary Kuhner and John Yamato of Saitou and Nei's "Neighbor Joining Method," and of the UPGMA (Average Linkage clustering) method. Neighbor Joining is a distance matrix method producing an unrooted tree without the assumption of a clock. UPGMA does assume a clock. The branch lengths are not optimized by the least squares criterion but the methods are very fast and thus can handle much larger data sets.

CONTML

Estimates phylogenies from gene frequency data by maximum likelihood under a model in which all divergence is due to

genetic drift in the absence of new mutations. Does not assume a molecular clock. An alternative method of analyzing this data is to compute Nei's genetic distance and use one of the distance matrix programs. This program can also do maximum likelihood analysis of continuous characters that evolve by a Brownian Motion model, but it assumes that the characters evolve at equal rates and in an uncorrelated fashion, so that it does not take into account the usual correlations of characters.

GENDIST

Computes one of three different genetic distance formulas from gene frequency data. The formulas are Nei's genetic distance, the Cavalli-Sforza chord measure, and the genetic distance of Reynolds et. al. The former is appropriate for data in which new mutations occur in an infinite isoalleles neutral mutation model, the latter two for a model without mutation and with pure genetic drift. The distances are written to a file in a format appropriate for input to the distance matrix programs.

CONTRAST

Reads a tree from a tree file, and a data set with continuous characters data, and produces the independent contrasts for those characters, for use in any multivariate statistics package. Will also produce covariances, regressions and correlations between characters for those contrasts. Can also correct for within-species sampling variation when individual phenotypes are available within a population.

PARS

Multistate discrete-characters parsimony method. Up to 8 states (as well as "?") are allowed. Cannot do Camin-Sokal or Dollo Parsimony. Can reconstruct ancestral states, use character weights, and infer branch lengths.

MIX

Estimates phylogenies by some parsimony methods for discrete character data with two states (0 and 1). Allows use of the Wagner parsimony method, the Camin-Sokal parsimony method, or arbitrary mixtures of these. Also reconstructs ancestral states and allows weighting of characters (does not infer branch lengths).

MOVE

Interactive construction of phylogenies from discrete character data with two states (0 and 1). Evaluates parsimony and compatibility criteria for those phylogenies and displays reconstructed states throughout the tree. This can be used to find parsimony or compatibility estimates by hand.

PENNY

Finds all most parsimonious phylogenies for discrete-character data with two states, for the Wagner, Camin-Sokal, and mixed parsimony criteria using the branch-and-bound method of exact search. May be impractical (depending on the data) for more than 10-11 species.

DOLLOP

Estimates phylogenies by the Dollo or polymorphism parsimony criteria for discrete character data with two states (0 and 1). Also reconstructs ancestral states and allows weighting of characters. Dollo parsimony is particularly appropriate for restriction sites data; with ancestor states specified as unknown it may be appropriate for restriction fragments data.

DOLMOVE

Interactive construction of phylogenies from discrete character data with two states (0 and 1) using the Dollo or polymorphism parsimony criteria. Evaluates parsimony and compatibility criteria for those phylogenies and displays reconstructed states throughout the tree. This can be used to find parsimony or compatibility estimates by hand.

DOLPENNY

Finds all most parsimonious phylogenies for discrete-character data with two states, for the Dollo or polymorphism parsimony criteria using the branch-and-bound method of exact search. May be impractical (depending on the data) for more than 10-11 species.

CLIQUE

Finds the largest clique of mutually compatible characters, and the phylogeny which they recommend, for discrete character data with two states. The largest clique (or all cliques within a given size range of the largest one) are found by a very fast branch and bound search method. The method does not allow for missing data. For such cases the T (Threshold) option of PARS or MIX may be a useful alternative. Compatibility methods are particular useful when some characters are of poor quality and the rest of good quality, but when it is not known in advance which ones are which.

FACTOR

Takes discrete multistate data with character state trees and produces the corresponding data set with two states (0 and 1). Written by Christopher Meacham. This program was formerly used to accomodate multistate characters in MIX, but this is less necessary now that PARS is available.

DRAWGRAM

Plots rooted phylogenies, cladograms, and phenograms in a wide variety of user-controllable formats. The program is interactive and allows previewing of the tree on PC or Macintosh graphics screens, and Tektronix or Digital graphics terminals. Final output can be to a file formatted for one of the drawing programs, on a laser printer (such as Postscript or PCL-compatible printers), on graphics screens or terminals, on pen plotters (Hewlett-Packard or Houston Instruments) or on dot matrix printers capable of graphics (Epson, Okidata, Imagewriter, or Toshiba).

DRAWTREE

Similar to DRAWGRAM but plots unrooted phylogenies.

TREEDIST

Computes the Robinson-Foulds symmetric difference distance between trees, which allows for differences in tree topology (but does not use branch lengths).

CONSENSE

Computes consensus trees by the majority-rule consensus tree method, which also allows one to easily find the strict consensus tree. Is not able to compute the Adams consensus tree. Trees are input in a tree file in standard nested-parenthesis notation, which is produced by many of the tree estimation programs in the package. This program can be used as the final step in doing bootstrap analyses for many of the methods in the package.

RETREE

Reads in a tree (with branch lengths if necessary) and allows you to reroot the tree, to flip branches, to change species names and branch lengths, and then write the result out. Can be used to convert between rooted and unrooted trees.

OVERVIEW OF THE INPUT AND OUTPUT FORMATS

When you run most of these programs, a menu will appear offering you choices of the various options available for that program. The data that the program reads should be in an input file called (in most cases) "infile". If there is no such file the programs will ask you for the name of the input file. Below we describe the input file format, and then the menu.

Input File Format

I have tried to adhere to a rather stereotyped input and output format. For the parsimony, compatibility and maximum likelihood programs, excluding the distance matrix methods, the simplest version of the input file looks something like this:

```
6 13
Archaeopt CGATGCTTAC CGC
HesperornicGTTACTCGT TGT
BaluchitheTAATGTTAAT TGT
B. virginITAATGTTCGT TGT
BrontosaurCAAAACCCAT CAT
B.subtilisGGCAGCCAAT CAC
```

The first line of the input file contains the number of species and the number of characters, in free format, separated by blanks (not by commas). The information for each species follows, starting with a ten-character species name (which can include punctuation marks and blanks), and continuing with the characters for that species. In the discrete-character, DNA and protein sequence programs the characters are each a single letter or digit, sometimes separated by blanks. In

the continuous-characters programs they are real numbers with decimal points, separated by blanks:

```
Latimeria 2.03 3.457 100.2 0.0 -3.7
```

The conventions about continuing the data beyond one line per species are different between the molecular sequence programs and the others. The molecular sequence programs can take the data in "aligned" or "interleaved" format, with some lines giving the first part of each of the sequences, then lines giving the next part of each, and so on. Thus the sequences might look like this:

```
6 39
Archaeopt CGATGCTTAC CGCCGATGCT
HesperornicGTTACTCGT TGTCGTTACT
BaluchitheTAATGTTAAT TGTTAATGTT
B. virginiaTAATGTTTCGT TGTTAATGTT
BrontosaurCAAAACCCAT CATCAAAACC
B.subtilisGGCAGCCAAT CACGGCAGCC
```

```
TACCGCCGAT GCTTACCGC
CGTTGTCGTT ACTCGTTGT
AATTGTTAAT GTTAATTGT
CGTTGTTAAT GTTCGTTGT
CATCATCAAA ACCCATCAT
AATCACGGCA GCCAATCAC
```

Note that in these sequences we have a blank every ten sites to make them easier to read: any such blanks are allowed. The blank line which separates the two groups of lines (the ones containing sites 1-20 and ones containing sites 21-39) may or may not be present, but if it is, it should be a line of zero length and not contain any extra blank characters (this is because of a limitation of the current versions of the programs). It is important that the number of sites in each group be the same for all species (i.e., it will not be possible to run the programs successfully if the first species line contains 20 bases, but the first line for the second species contains 21 bases).

Alternatively, an option can be selected to take the data in "sequential" format, with all of the data for the first species, then all of the characters for the next species, and so on. This is also the way that the discrete characters programs and the gene frequencies and quantitative characters programs want to read the data. They do not allow the "interleaved" format.

In the sequential format, the character data can run on to a new line at any time (except in a species name or in the case of continuous character and distance matrix programs where you cannot go to a new line in the middle of a real number). Thus it is legal to have:

```
Archaeopt 001100
1101
```

or even:

```
Archaeopt
```

0011001101

though note that the FULL ten characters of the species name MUST then be present: in the above case there must be a blank after the "t". In all cases it is possible to put internal blanks between any of the character values, so that

Archaeopt 0011001101 0111011100

is allowed.

If you make an error in the input file, the programs will often detect that they have been fed an illegal character or illegal numerical value and issue an error message such as "BAD CHARACTER STATE:", often printing out the bad value, and sometimes the number of the species and character in which it occurred. The program will then stop shortly after. One of the things which can lead to a bad value is the omission of something earlier in the file, or the insertion of something superfluous, which cause the reading of the file to get out of synchronization. The program then starts reading things it didn't expect, and concludes that they are in error. So if you see this error message, you may also want to look for the earlier problem that may have led to this.

The other major variation on the input data format is the options information. Many options are selected using the menu, but a few are selected by including extra information in the input file. Some options are described below.

The Options Menu

The menu is straightforward. It typically looks like this (this one is for DNAPARS):

DNA parsimony algorithm, version 3.5c

Setting for this run:

```

U           Search for best tree?  Yes
J  Randomize input order of sequences?  No. Use input order
O           Outgroup root?  No, use as outgroup species 1
T           Use Threshold parsimony?  No, use ordinary parsimony
M           Analyze multiple data sets?  No
I           Input sequences interleaved?  Yes
0  Terminal type (IBM PC, VT52, ANSI)?  ANSI
1  Print out the data at start of run  No
2  Print indications of progress of run  Yes
3           Print out tree  Yes
4           Print out steps in each site  No
5  Print sequences at all nodes of tree  No
6           Write out trees onto tree file?  Yes

```

Are these settings correct? (type Y or the letter for one to change)

If you want to accept the default settings (they are shown in the above case) you can simply type "Y" followed by a carriage-return (Enter) character. If you want to change any of the options, you should type the letter shown to the left of its entry in the menu. For example, to set a threshold type "T". Lower-case letters will also work. For many of the options the program will ask for supplementary information, such as the value of the threshold.

Note the "Terminal type" entry, which you will find on all menus. It allows you to specify which type of terminal your screen is. The options are an IBM PC screen, an ANSI standard terminal (such as a DEC VT100), a DEC VT52-compatible terminal, such as a Zenith Z29, or no terminal type. Choosing "0" toggles among these four options in cyclical order, changing each time the "0" option is chosen. If one of them is right for your terminal the screen will be cleared before the menu is displayed. If none works the "none" option should probably be chosen. Keep in mind that VT-52 compatible terminals can freeze up if they receive the screen-clearing commands for the ANSI standard terminal! If this is a problem it may be helpful to recompile the program, setting the constants near its beginning so that the program starts up with the VT52 option set.

The other numbered options control which information the program will display on your screen or on the output files. The option to "Print indications of progress of run" will show information such as the names of the species as they are successively added to the tree, and the progress of global rearrangements. You will usually want to see these as reassurance that the program is running and to help you estimate how long it will take. But if you are running the program "in background" as can be done on multitasking and multiuser systems such as Unix, and do not have the program running in its own window, you may want to turn this option off so that it does not disturb your use of the computer while the program is running.

The Output File

--- -----

Most of the programs write their output onto a file called (usually) "outfile", and a representation of the trees found onto a file called "treefile".

The exact contents of the output file vary from program to program and also depend on which menu options you have selected. For many programs, if you select all possible output information, the output will consist of (1) the name of the program and its version number, (2) the input information printed out, (3) a series of phylogenies, some with associated information indicating how much change there was in each character or on each part of the tree. A typical rooted tree looks like this:

```

                                     +-----Gibbon
      +-----2
      !
      !
      !
+----3
!    !
                                     +-----Orang
                                     +-----4
                                     ! +-----Gorilla
                                     +--6
                                     ! +-----Chimp

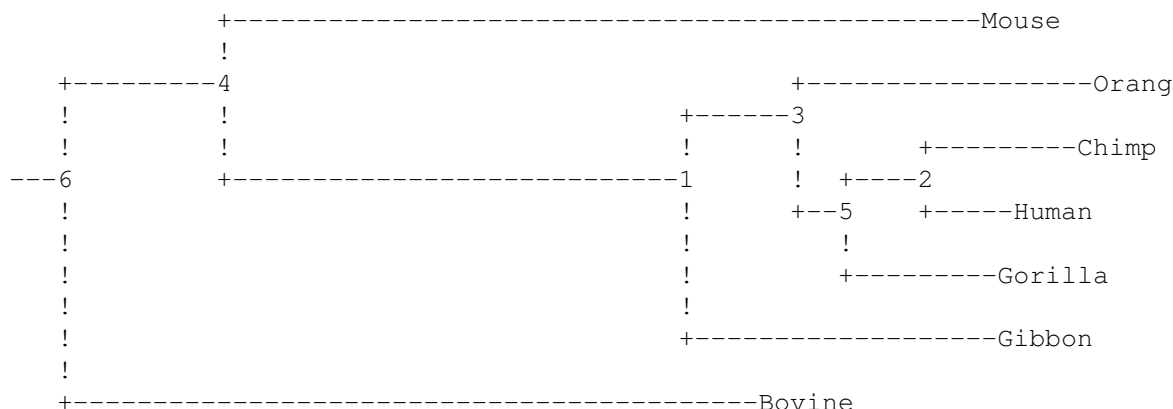
```



The interpretation of the tree is fairly straightforward: it "grows" from left to right. The numbers at the forks are arbitrary and are used (if present) merely to identify the forks. In some of the programs asterisks ("*") are used instead of numbers. For many of the programs the tree produced is unrooted. It is printed out in nearly the same form, but with a warning message:

remember: this is an unrooted tree!

The warning message ("remember: ...") indicates that this is an unrooted tree (mathematicians still call this a tree, though some systematists unfortunately use the term "network". This conflicts with standard mathematical usage, which reserves the name "network" for a completely different kind of graph). The root of this tree could be anywhere, say on the line leading immediately to Mouse. As an exercise, see if you can tell whether the following tree is or is not a different one from the above:



remember: this is an unrooted tree!

(it is NOT different). It is IMPORTANT also to realize that the lengths of the segments of the printed tree may not be significant: some may actually represent branches of zero length, in the sense that there is no evidence that the branches are nonzero in length. Some of the diagrams of trees attempt to print branches approximately proportional to estimated branch lengths, while in others the lengths are purely conventional and are presented just to make the topology visible. You will have to look closely at the documentation that accompanies each program to see what it presents and what is known about the lengths of the branches on the tree. The above tree attempts to represent branch lengths approximately in the diagram. But even in those cases, some of the smaller branches are likely to be artificially lengthened to make the tree topology clearer. Here is what a tree from DNAPARS looks like, when no attempt is made to make the lengths of branches in the diagram proportional to estimated branch lengths:

```

          +--Human
            +--5
              +--4  +--Chimp
                !  !
                +--3  +-----Gorilla
                  !  !
                  +--2  +-----Orang
                    !  !
                    +--1  +-----Gibbon
                      !  !
--6  +-----Mouse
      !
      +-----Bovine

```

remember: this is an unrooted tree!

Some of the parsimony programs in the package can print out a table of the number of steps that different characters (or sites) require on the tree. This table may not be obvious at first. A typical example looks like this:

```

steps in each site:
      0  1  2  3  4  5  6  7  8  9
*-----
0!      2  2  2  2  1  1  2  2  1
10!     1  2  3  1  1  1  1  1  2
20!     1  2  2  1  2  2  1  1  2
30!     1  2  1  1  1  2  1  3  1
40!     1

```

The numbers across the top and down the side indicate which site is being referred to. Thus site 23 is column "3" of row "20" and has 2 steps in this case.

The Tree File

--- ----

In output from most programs, a representation of the tree is also written into the tree file (usually named "treefile"). The tree is specified by the nested pairs of parentheses, enclosing names and separated by commas. If there are any blanks in the names, these must be replaced by the underscore character "_". Trailing blanks in the name may be omitted. The pattern of the parentheses indicates the pattern of the tree by having each pair of parentheses enclose all the members of a monophyletic group. The tree file for the above tree would have its first line look like this:

```
((Mouse,Bovine),((Orang,(Gorilla,(Chimp,Human))),Gibbon));
```

In the above tree the first fork separates the lineage leading to Mouse and Bovine from the lineage leading to the rest. Within the latter group there is a fork separating Gibbon from the rest, and so on. The entire tree is enclosed in an outermost pair of parentheses. The tree ends with a semicolon. In some programs such as DNAML, FITCH, and CONTML, the tree will be completely unrooted

and specified by a bottommost fork with a three-way split, with three "monophyletic" groups separated by two commas:

```
(A, (B, (C,D)), (E,F));
```

The three "monophyletic" groups here are A, (B,C,D), and (E,F). The single three-way split corresponds to one of the interior nodes of the unrooted tree (it can be any interior node). The remaining forks are encountered as you move out from that first node, and each then appears as a two-way split. You should check the documentation files for the particular programs you are using to see in which of these forms you can expect the user tree to be in. Note that many of the programs that estimate an unrooted tree produce trees in the treefile in rooted form! This is done for reasons of arbitrary internal bookkeeping. The placement of the root is arbitrary.

For programs estimating branch lengths, these are given in the trees in the tree file as real numbers following a colon, and placed immediately after the group descended from that branch. Here is a typical tree with branch lengths:

```
((cat:47.14069, (weasel:18.87953, ((dog:25.46154, (raccoon:19.19959, bear:6.80041):0.84600):3.87382, (sea_lion:11.99700, seal:12.00300):7.52973):2.09461):20.59201):25.0, monkey:75.85931);
```

Note that the tree may continue to a new line at any time except in the middle of a name or the middle of a branch length, although in trees written to the tree file this will only be done after a comma.

These representations of trees are a subset of the standard adopted on June 24, 1986 at the annual meetings of the Society for the Study of Evolution at an meeting (the final session in a local lobster restaurant) of an informal committee consisting of Wayne Maddison (MacClade), David Swofford (PAUP), F. James Rohlf (NTSYS-PC), Chris Meacham (COMPROB and plotting programs), James Archie (character coding program), William H.E. Day, and me. This standard is a generalization of PHYLIP's format, itself based on a well-known representation of trees in terms of parenthesis patterns which has been around for almost a century. The standard is now employed by most phylogeny computer programs but unfortunately has yet to be described in a formal published description.