A Brief Introduction to Molecular Evolutionary Phylogenetic Inference

Content: The rationale, methodology, and interpretation of molecular phylogenetic inference software. The proper use and interpretation of computational tools for the inference of molecular phylogenies is an extremely complicated subject. Many blatant errors are made in the scientific literature in this field, perhaps more so than in any other aspect of computational molecular biology. This workshop samples and attempts to familiarize you with the basics of several of the techniques available for inferring molecular phylogenies — distance and parsimony based, and maximum likelihood algorithms. Based on a multiple sequence alignment that I provide, you’ll use several methods available in PAUP* as implemented within GCG’s SeqLab, as well as several PHYLIP methods. These programs all analyze aligned molecular sequence datasets to ascertain phylogenies; their suitability and limitations will be discussed and explored while you learn how they run.

Fall 2006/Spring 2007; a GCG\(^{\text{\textregistered}}\) Wisconsin Package™ SeqLab\(^{\text{\textcopyright}}\) tutorial for Florida State University, sponsored by the School of Computational Science (SCS).

Author and Instructor: Steven M. Thompson
GCG is the Genetics Computer Group, part of Accelrys Inc., producer of the Wisconsin Package™ for sequence analysis.
Introduction

(portions of introduction condensed from Gary Olsen’s ‘oldy but goody: *Inference of Molecular Phylogenies*,
University of Illinois at Urbana-Champaign; Thursday, September 3, 1992. Thanks Gary!)

The inscription on the inner cover of the classic *Evolution*, by Dobzhansky, Ayala, Stebbins and Valentine (1977, quoted from Dobzhansky in 1973) reads: “Nothing in biology makes sense except in the light of evolution.” These words ring true to me. Evolution provides the single, unifying, cohesive force that allows all of life to be explained. It is to the life sciences what the long sought holy grail of the unified field theory is to astrophysics.

Since the 19th Century

Similarity is a result of descent from a common ancestor. Species evolve — they are not static; they change over time. Charles Darwin described the process as variation sorted out through drift and selection → lineages diverge; “descent with modification.”

The Modern Synthesis: Mendelism meets Darwinism (starting in the 1930’s, Fisher, Haldane, and Wright); however, *deemphasis on phylogenetics*.

Enter Zimmerman (starting in the 1920’s) and Hennig (in the 1950’s). Their ideas began to form the school of *Phylogenetic Systematics — the Cladists*.

And then came the molecule:

Perhaps the first reference to molecules as a means for deciphering phylogeny is Zuckerkandl and Pauling’s hemoglobin studies (1965).

**Organisms have a History**

Parental strain 0 diverged into two different strains over time, strain 1 and strain 2. How can we discover that History?

**Homology Concepts**

Homologous versus Analogous: e.g. flight — bird and bat wings versus insect wings. And . . .

Orthologous versus Paralogous Homologues: e.g. inter- versus intraspecies.

But what about with molecules? Paralogues are the result of gene duplication and subsequent divergence within a lineage; orthologues result from speciation events between lineages. Therefore, paralogues are different related genes in the same organism; orthologues are the same gene in different related organisms. For example, consider the hemoglobin scenario on the following page:
If we did not know the difference between the $\alpha$ and $\beta$ globins, then we might be very upset by a (historically correct) tree based on fewer sequences, such as the one on the right:

In this example, the $\alpha$ globins are orthologous with one another, and the $\beta$ globins are orthologous with one another. However, the $\alpha$ globins are paralogous to the $\beta$ globins (they have evolved in parallel, at different locations in the genome, the result of an early gene duplication, prior to the divergence of humans and frogs). If one desires to infer relationships among species, it is necessary to study orthologous genes, usually paralogues just ‘mess up’ and overcomplicate the picture, especially if you don’t have all the data.

**Sequence Alignment Concepts**

The concepts of common ancestry and homology extend beyond anatomy, physiology, and genes down to the smallest heritable unit of life, the individual base pairs of DNA. Thus, even after homologous sequences are identified, it is necessary to establish the correspondence between individual sequence positions. A multiple sequence alignment is a phylogenetic hypothesis, and all molecular sequence phylogenetic inference algorithms make the validity of your input alignment their first and most critical assumption. Considerable time and energy need to be devoted toward developing the best alignments possible. Use all available information and understanding to insure this — known enzymatic, regulatory, and structural elements must align — alignments must make biological sense. Meaningful inference results are absolutely dependent on the quality of your alignments. Therefore, insure that you have prepared satisfying ones!

The available software can process any alignment that you feed it of the proper format. Whether or not it is appropriate and should be used, is up for you to determine. Beware of comparing ‘apples and oranges.’ Make sure that the family of sequences that you align are in fact related and that the alignment is meaningful. The programs will work with almost any input sequences but only make sense if they actually do belong to the same gene family. In general, either make paralogous comparisons (i.e. evolution via gene duplication) to ascertain gene phylogenies all within one organism, or orthologous (within one ancestral loci) comparisons to ascertain gene phylogenies between organisms, which should imply organismal phylogenies; try not to mix them up. Lots of confusion can arise and extremely misleading interpretations can result otherwise. Also, be wary of trying to align genomic sequences with cDNA when working with DNA. Similarly, don’t try to align mature and precursor proteins; trim them even if they are different sequences. Otherwise it doesn’t make evolutionary sense to use both of them, as one is not evolved from the other, rather one is the other — these are all easy mistakes to make and can cause all sorts of problems.
The Phylogenetic Inference Algorithms

Distance Methods

In its simplest implementation distance methods are ‘cluster analysis’ but . . .

the concepts of Evolutionary Distance:

\[ X_{01} + X_{02} \]

i.e. the sum total of all the evolutionary events since the divergence from 0 along the complete path between 1 and 2. And Additive Trees:

Evolutionary distances are additive as a consequence of defining distance in terms of events. The evolutionary distance from 1 to 2 above is \( X_{01} + X_{02} \),

An interesting mathematical property of additive trees is that it is only necessary to know the pairwise distances between the tips of the tree branches to uniquely infer the corresponding unrooted tree. Because evolutionary distances are defined to be additive, the evolutionary distances relating a set of present-day sequences will fit exactly one unrooted additive tree (both in branching order and branch lengths). It is not possible to infer the root of the tree without additional assumptions or data. Note: A molecular clock plays no role in the validity of the method, in spite of numerous assertions to the contrary.

Therefore, if one truly knew the actual number of events that have occurred in evolutionary history then one could always get the correct tree, but . . . . This all seems too easy, and it is.

That's because we don't know the actual number of events that contemporary sequences have diverged from one another. Instead, we observe present day sequence differences. Unfortunately, a difference could have arisen by one or more superimposed substitutions (e.g. \( A \rightarrow G \rightarrow C \rightarrow G \), three events yielding one observed substitution). Worse yet, identical residues can be separated by a history of two or more changes (e.g. \( A \rightarrow G \rightarrow A \), two events yielding no observed substitutions). And the longer the divergence has been, i.e. the greater the sequence difference, the worse the whole situation becomes; the harder it is to ‘see’ the actual number of evolutionary events along the route.

The phenomenon can be visualized with a graph. If you plot evolutionary divergence over time, the actual number of sequence changes accumulates at a fairly constant rate, dependent on mutation rate, drift, and selection, yet the observed changes plateau out after some saturation level is reached. This saturation is sometimes referred to as homoplasy and is illustrated in the graph on the following page:
This situation necessitates two things:

1) We must estimate the number of actual substitution events from the number of observed sequence differences using some model of evolution, and

2) Since the above step produces estimates, not additive evolutionary distances, it is necessary to fit these values to an additive tree as best as possible (knowing that the branch lengths will only approximately reproduce the pairwise distance estimates).

The first step requires assumptions. The second step is just arithmetic, once you decide ‘best fit’s’ definition.

Estimating Evolutionary Distances from Sequence Differences

Given a sequence alignment it is simple to count the number of sequence differences. By making assumptions about the distribution of substitutions along the length of the molecule, it is possible to estimate the number of superimposed changes, and hence the total number of events giving rise to the observed differences. This requires the following assumptions:

1) All phylogenetic inference methods assume a correct alignment, so it better be a good one!

2) All methods assume that the probability of a given evolutionary event depends only on the current state of the sequence, not on any previous changes, i.e. sequence evolution is a Markovian process.

3) Furthermore, it is as probable to change from a particular state to another, as it is to change back to the original. That is, all methods are “time-reversible.”

4) Changes at different positions within a molecule, and changes in other lineages, are assumed to be independent of each other in most all methods. This is the independent portion of the I.I.D. (independently and identically distributed) principle and is a potential source of error in most methods.

5) Many methods also assume that every site evolves at the same rate along the length of the sequence. This is the identical portion of I.I.D. and is clearly violated in almost every natural system observed. Therefore, more complex models allow for site rate heterogeneity, both in the percentage of invariant sites, and in the rate distribution among the rest of the sites, but as more free parameters are added to any model, systematic errors become evermore exaggerated.
6) No method satisfactorily treats insertions and deletions (gaps). Parsimony can attempt to count them, but other methods effectively assume that they do not exist at all, or they are treated as unknown states.

Various models of sequence evolution utilize these assumptions in different ways in order to attempt to estimate these superimposed substitutions. The simplest model, from Jukes and Cantor (JC 1969), assumes that all sites have equal base frequencies, all sites are equally subject to change (and therefore, transitions and transversions are equivalent), and all sites change at a uniform rate along the length of the sequence. These assumptions are always violated with ‘real’ data. Therefore, subsequent models relax various assumptions, e.g. the Kimura two-parameter model (K2P 1983) assumes that all transitions and all transversions occur at different rates but are independently equivalent, and Hasegawa, Kishino, and Yano (HKY 1985) add unequal base frequencies to the mix.

Regardless of the ultimate biological validity of these assumptions, do not accept the ‘unrealistic assumptions cop-out!’ All methods will suffer from some deviation from their assumptions at some point. However, . . . in short, if change is sufficiently rare, i.e. if the sequences aren't too diverged for the analysis at hand, none of the methods will suffer from systematic error as a result of deviance from their implicit or explicit assumptions.

The moral of the story: omit (or mask out, as is possible with the GCG SeqLab Editor, and is shown below) those most highly diverged sequence regions from your analyses. It is in these areas of high homoplasy that all assumptions are most violated. Homoplasy is defined as the similarity of organs, other bodily structures, or, in this case, individual base pairs of DNA, between different species, not due to common ancestral origin and development, but rather due to independent evolutionary change: parallelism, reversal, or convergence. Thus, homoplasy is a mistaken hypothesis of homology that confounds phylogenetic analyses. Regions of high homoplasy correspond to areas where your alignment is least sure, that saturation zone mentioned above. Therefore, get rid of 'em.

Two often used Distance methods:

**Neighbor-Joining**

The method is based on clustering similar sequences, but with a correction for the differing relative rates of change in different lineages (NJ Saitou and Nei, 1987). It does not use any type of an ‘optimality criteria,’ rather it is an arithmetical method that comes up with one single tree, where it is impossible to know how good or bad that tree is. It is not an extremely accurate method but is often used as a ‘quick and dirty’ way to get an approximate phylogeny or as a starting point for other methods.

**Least Squares Fit to an Additive Tree**

Given pairwise distance estimates, out of possible trees, find the tree and branch lengths that best, ‘optimally,’ explain these estimates using the least squares method (similar to finding the best line that fits a scattered point dataset). The least squares sum is the method’s optimality criteria (Fitch and Margoliash, 1967).
If you need to root your tree, use only one, most closely related outgroup, in all distance methods to decrease the amount of implicit, systematic error in their assumptions.

**Parsimony**

Historically, the idea of parsimony arose in the analysis of complex morphological features for which it is intuitively appealing to assume were invented only once. In that case, all species in which a given derived feature is found should share a common ancestor with that feature. In terms of methods of phylogenetic analysis, this means that one desires the evolutionary history that would require the minimum amount of reinvention. Mathematically, this is equivalent to the minimum amount of invention. This philosophy is the core of phylogenetic “cladistics.”

The method of inferring the "most parsimonious" (minimum change) tree is a purely computational issue. As with the least squares tree method, it is relatively easy to find out how good (or bad) a given tree is, but it is not so simple to find the best tree. In this case the best tree’s optimality criteria is that tree with the shortest overall branches, i.e. the smallest sum of all of its branch lengths. This called the shortest “tree length.”

Unfortunately, there is no experimental evidence that it is particularly difficult to reinvent a given nucleotide at a given position in a sequence. Consequently, with molecular data parsimony runs a very real risk of confusing residues that are the same due to multiple changes (reinvention) with those that are the same due to conservation.

This problem is particularly evident with long tree branches (long defined as having sufficient numbers of substitutions to have significant superimposed changes). Given ‘long’ enough branches, this ‘long-branch-attraction’ problem, often referred to as being in the ‘Felsenstein Zone,’ guarantees that standard parsimony will infer the wrong tree.

It is illustrated here:

In this ‘true’ tree, A and B are specifically grouped, as are C and D. However, if the branches to A and C are sufficiently long, then there is a significant random chance that a nucleotide will end up the same in each of these two lines. If this happened, then B and D could share the original nucleotide, and A and C will (due to parallel, independent events) share a different nucleotide.

Parsimony analysis, however, would group A with C, and B with D, so that the observed pattern of nucleotides could be explained by a single change. That is, the inferred tree will tend to look like the following:

Even though this is not the historically correct tree.
This problem can be minimized by (1) avoiding very diverged sequences, which will have the longest branches; by (2) avoiding the most variable sequence positions, which will have the greatest chance of multiple substitutions; and by (3) including more than one representative of a distant group, which subdivides the long branch into shorter branches (only do this in parsimony or maximum likelihood analyses; as mentioned above, it can compound problems in the distance methods).

**Maximum Likelihood**

Maximum likelihood yields a statistical ranking of possible trees; it attempts to trace the evolution of each sequence position through an assumed phylogenetic tree, subject to the expected amount of change implied by the tree branch lengths — it chooses those trees that maximize the probability of observing the data. Therefore, evolutionary probability is its optimality criteria. Originally applied to phylogenetics by Felsenstein (1981), it is, in effect, a bridge between distance and parsimony methods. It can provide the ‘best’ answers to most problems since it combines the strongest points of the other methods. The mathematics can be incredibly complicated utilizing n-dimensional matrix integrals and other mind-boggling concepts. However, given ‘enough’ data and an appropriate model of evolution, maximum likelihood will always find the correct tree, even in those cases where parsimony will positively fail. Another advantage is, because it uses probabilities, it can even estimate the most probable evolutionary models to use in its calculations.

Drawback: It is incredibly computationally intensive, especially if you use it to estimate models and calculate trees simultaneously. Therefore, it is often not appropriate for most ‘normal’ computer systems with ‘extremely large’ data sets, though there are lots of tricks to speed things up. A major trick commonly employed by those ‘in the know’ is to use maximum likelihood to iterate between model finding and tree discovery.

Regardless, as computers become faster and faster, and heuristic programs get better and better, it, along with similarly statistics-based Bayesian techniques (that will not be covered in this tutorial), are quickly becoming the methods of choice among serious molecular phylogeneticists. See [http://mrbayes.scs.fsu.edu/](http://mrbayes.scs.fsu.edu/) for MrBayes (Ronquist and Huelsenbeck, 2003), one of these powerful Bayesian statistical methods for phylogenetic inference.

**Implementations** for all Explicit Optimality Criteria Methods

Exact Methods — Exhaustive versus Branch-and-Bound

**Exhaustive**

Add every sequence, stepwise, at every possible position. This is incredibly cpu intensive and is intractable for most datasets, only practical for up to about eight sequences or so. Reason being — the number of possible unrooted, bifurcating trees for a dataset rises as a factorial of the number of sequences. For those of you who like mathematics, the equation can be written thus, where \( T \) is the number of individual sequences:
\[
B(T) = \prod_{i=3}^{T} (2i - 5)
\]

Or in another form:

\[
= \frac{(2T - 5)!}{2^{T-3}(T-3)!}
\]

Or, in a much easier to understand manner, a simple table enumerates the increase. That table follows:

<table>
<thead>
<tr>
<th>Number of sequences</th>
<th>Number of possible bifurcating, unrooted trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>945</td>
</tr>
<tr>
<td>8</td>
<td>10,395</td>
</tr>
<tr>
<td>9</td>
<td>135,135</td>
</tr>
<tr>
<td>10</td>
<td>2,027,025</td>
</tr>
<tr>
<td>20</td>
<td>&gt; 2 \times 10^{20}</td>
</tr>
<tr>
<td>50</td>
<td>&gt; 3 \times 10^{24} (\geq \Sigma of all the atoms in the universe)</td>
</tr>
<tr>
<td>10 million</td>
<td>&gt; 5 \times 10^{68,667,340}</td>
</tr>
</tbody>
</table>

**Branch-and-Bound**

Start with the evaluation of one ‘route, all the way out;’ but only evaluate those routes with better scores than the starting route, and update that starting score as the search proceeds. This eliminates much of the search time, yet guarantees an optimal tree. However, it is only practical for up to about twenty sequences or so.

**Approximate Methods — Heuristics**

Only take the route of ‘lowest resistance;’ this is the only route evaluated. This is analogous to the ‘hill-climbing’ algorithms used in molecular mechanics computations based on energetics. Problem: getting ‘trapped’ on local minima. Therefore, tree rearrangement algorithms are usually incorporated into the programs and should always be taken advantage of (PAUP**’s “tree bisection/reconnection” and PHYLIP’s “global” options).

Still — best bet is to start at multiple points, as well as to use tree rearrangement routines; i.e. take advantage of random input order options and repeat the analyses many times, at least ten (PAUP**’s “multiple random additions” and PHYLIP’s “jumble” options)! David Swofford, PAUP**’s author, uses the “blind parachutist” analogy to teach the method. Imagine if you will, a squadron of severely myopic parachutists who all lose their glasses on their jump. Their collective goal is to find the highest peak in the area — they are all equipped with altimeters and walky-talkies. They can’t see the surrounding peaks but they can tell that they are walking uphill versus downhill, so they climb up. Once they reach the top of their respective peak they can read their altimeter and communicate with one-another to collectively identify the highest peak. Ergo, send down lots of parachutists to cover lots of territory, i.e. repeat the analysis as many times as practical and, since the results are order dependent, use several different starting orders.
How Reliable are the Answers? Systematic Errors and Assumptions

Do not accept the ‘unrealistic assumptions’ excuse. In regard to practical consideration always remember the old adage: “garbage in — garbage out!”  Some general guidelines to keep in mind include the following:

1) Avoid the most diverged regions of molecules; these regions are the greatest source of systematic error. Therefore, if the homology of a region is in doubt, then throw that data out (or “mask” it, as shown below).
2) Do not include entire sequences that are more diverged than necessary for the analysis at hand.
3) In pairwise distance techniques, use only one ‘out-group’ species, if any. In parsimony and maximum likelihood methods, consider using a few, if you are using them at all, to subdivide the long out-group branch. In all cases, if you are using an out-group in order to root your tree, try to make your out-group as close to your in-groups as possible without actually being a part of an in-group.
4) When all else fails, use more data! Often the most practical solution to uncertain relationships is to use more sequence, i.e. more molecules, and/or longer sequences. Of course, this isn’t always practical and even when it is, it is guaranteed to increase your run time.

Bootstrapping

To help deal with random errors consider testing with bootstrap techniques. Bootstrap resampling provides a loose confidence limit on the groups that are separated by a given tree branch (Felsenstein, 1985). It gives us a feel for the reliability of a phylogenetic tree, and on how ‘good’ the signal is in the data. The method randomly selects (with replacement) character columns from the original dataset up to the same number of character columns as the original dataset to create ‘pseudo-replicate’ datasets, usually at least 100, better yet, a 1000. Some columns will be over-sampled, others not sampled at all. Whichever inference algorithm you want to use is then run on each of the bootstrapped pseudo-replicate datasets to produce the same number of best trees as the number of datasets. A consensus tree is then calculated from all the different pseudo-replicate dataset trees, and support values, usually as a percentage, are printed for each branch associated with each node. In other words, how often is a particular node’s branch found among all the bootstrapped trees?

A ‘rule of thumb’ that people often use is those nodes that have a bootstrap value of more than ≈70% are probably better than 70% accurate, and those nodes with bootstrap values worse than ≈30% are way worse than 30%. This is why one seldom sees bootstrap values published less than 50%.

Understanding the packages

After you have obtained a satisfying alignment, the next step is to apply any of several molecular phylogenetic inference programs for evolutionary reconstruction to it. Using these programs effectively depends upon your understanding of the software packages’ programs, algorithms, and models. I encourage you to learn about the various evolutionary models and to experiment with the various options. Molecular Systematics (1996) is a very good general reference in this field. The Wisconsin Package also has a phylogenetic inference program section. Two of the GCG programs, Distances and GrowTree, work together to offer a neighbor-joining distance matrix
solution using several different methods for the correction of multiple substitutions at homologous sites. PAUP* version 4.0 (Phylogenetic Analysis Using Parsimony [and other methods] http://paup.scs.fsu.edu/) by David Swofford, formerly of the Smithsonian Institute but now an FSU SCS/Biology endowed chair, is also incorporated into GCG along with two ‘front-end’ interfaces to a limited subset of the package, PAUPSearch and PAUPDisplay. PAUP* is a tremendously powerful, full-featured phylogenetic inference package that incorporates traditional parsimony methods as well as distance methods and maximum likelihood methods. This tutorial will introduce the GCG incorporated programs as well as Joseph Felsenstein’s PHYLIP (PHYLogenetic Inference Package) programs from the University of Washington (http://evolution.genetics.washington.edu/phylip.html). PHYLIP is a comprehensive freeware suite of 35 different programs that can handle both molecular sequence and morphological character data.

PAUP*'s as well as PHYLIP's molecular sequence programs have versions that accept either amino acid or nucleotide sequence data. I strongly encourage you to perform all searching and alignment procedures on an amino acid basis, if dealing with coding sequences. If you want to perform your phylogenetic inference on nucleotide sequence data, which may be desirable, particularly with very similar sequences, it should be converted to nucleotides after alignment. In addition to the much more easily achieved alignment when dealing with amino acid sequences, this also insures that alignment gaps are not placed within codons. The SeqLab Editor interface is a big help with these matters. Lab Seven’s Introduction discussed the method for achieving this in SeqLab. Another general guideline to keep in mind is never initially impose a molecular clock on any phylogenetic inference program. The existence of a molecular clock for your system can be evaluated after the phylogeny has been inferred with statistical tests. Universal molecular clocks are hotly disputed, so don’t base your entire analysis upon this often-false premise. That is the main reason why I strongly argue against ever using UPGMA analysis — it mandates an absolutely perfect clock across your entire dataset. I will only illustrate those programs that do not impose a molecular clock on their analyses.

One of the biggest problems in this field is that of sequence format. Each suite of programs requires a different sequence format. You should be familiar with GCG sequence format by now, both as single sequence and Multiple Sequence Format (MSF), and SeqLab’s native Rich Sequence Format (RSF) that contains both sequence data and reference and feature annotation. Now we have to deal with still more types. PAUP* has a required format called the NEXUS file, and we will be using Bill Pearson’s FastA format as an intermediate on the way to PHYLIP’s own unique input data format requirements. Several different programs are available to allow us to convert formats back and forth between the required standards, but it all can get quite confusing. One very helpful program, ReadSeq by Don Gilbert at Indiana University, allows for the back and forth conversion between several different formats, and will be used in this tutorial to help generate PHYLIP format. The PAUP* interfaces in the GCG system, PAUPSearch and PAUPDisplay, automatically generate their required NEXUS format directly from GCG formatted files, so this is not nearly as much of a hassle.

PAUPSearch and PAUPDisplay can be run through SeqLab, or from the command line, or PAUP* can be launched in its native, “portable” command line format by giving the command “paup.” Documentation on PAUP* is available at the PAUP* WWW site (http://paup.scs.fsu.edu/Cmd_ref_v2.pdf), and as incorporated in GCG's
PAUPSearch and PAUPDisplay programs’ help pages, or directly from the Sinauer Publishing Company. The PHYLIP package’s complete documentation should be on the same computer that has the executables but different system administrators put things in different places. A better bet is to use the html documentation available on the Web through http://evolution.genetics.washington.edu/phylip/phylipweb.html. It’s a good idea to review the available PHYLIP documentation as an aid in selecting the appropriate software to fit your needs. Pay particular attention to the general package documentation, main.html, that describes all the general options available within PHYLIP. Some of them should always be taken advantage of since they can make your analysis much more robust. Some of PHYLIP’s programs require special input files. Others are meant to work on the output data created by another program in the package. Be sure you are familiar with the requirements of each program you wish to use prior to trying it.

Many of the programs also generate information that can be used to create plots of the results. PAUPDisplay uses standard GCG graphics routines so it supports all setplot and SeqLab graphics. The PHYLIP package can also output graphics to many different devices. However, not all are appropriate for displaying images through a network connection. We can try a few options while in the tutorial. Regardless of the graphics device chosen with either package, the highest resolution graphics available in both output PostScript.

To properly use the three software packages in this tutorial, GCG, PAUP*, and PHYLIP, it is necessary to understand some of the philosophy behind them. GCG was created with the user in mind. In spite of common opinion, it is quite user friendly, expects little in the way of background from its users, and provides excellent documentation for its software. GCG provides default parameters for the programs that, while not always ideal, in almost every case will produce meaningful results. And the SeqLab interface makes GCG even easier to use. Most all GCG functions can be launched from SeqLab. It allows full editing control of your alignments, and furthermore, it can parse feature information from database entries so that you can see where important structural and functional sites lay in your alignments. When GCG integrated PAUP* into their package, they attempted to maintain the ease of use issues by building the PAUPSearch and PAUPDisplay programs, although running PAUP* native through the command line “portable” interface can be very complicated indeed. PHYLIP was developed with the accomplished evolutionary biologist in mind. It expects its users to have a great deal of background knowledge and experience — enough to effectively choose rationale options in each program. While extensive documentation is provided, it is written with these expectations in mind.

You need to understand the characteristics of both GCG and PHYLIP to use alignments created by GCG in PHYLIP, especially with regard to the names of sequences, the interpretation of symbols, and file accession. Let’s begin with the manner in which the two systems treat sequence names. PHYLIP restricts sequence names to ten characters. GCG, while it will truncate names, does not do so nearly as drastically; rather, it uses the name itself as the determining factor for the length of the name field. The field will be truncated so that fifty bases or amino acids will be shown on the data line in a MSF sequence file. PHYLIP only lists a name for each sequence once in a data set. GCG’s MSF files display the name of the sequence on nearly every line in the file to avoid user confusion. In all cases spaces and punctuation, other than underscores, periods, or hyphens, should be avoided in sequence names. They’ll just cause problems sooner or later. PHYLIP’s ten character name limitation
can be gotten around just before the final plotting process, but it is a bother. In general, it's a good idea to begin the process with the names that you will want on a final graphic.

GCG and PHYLIP are independent entities; they use alignment symbols differently. Therefore, their symbol usage needs to be clarified. GCG alignment programs insert periods, "."s, to represent gaps, in the alignment and tildes, "~"s, to show placeholder spaces such as uneven end lengths. However, periods mean "the same symbol as the above sequence" to PHYLIP and it doesn't recognize the tildes at all. Therefore, any procedure for changing GCG sequence alignments to PHYLIP datasets will need to convert periods to hyphens (minuses), "-"s, which mean deletions to PHYLIP. ReadSeq does not do this automatically and it must be done either beforehand or afterwards. GCG’s SeqConv+ Importing/Exporting program does make this conversion though, so I encourage people to use FastA format as an intermediate on the way to PHYLIP.

PAUPSearch handles the gap problem by specifying gaps as periods in the NEXUS format file that it creates. However, not all gaps in sequences should be interpreted as deletions. Interior gaps are probably okay to represent this way, as regardless of whether a deletion, insertion or a duplication event created the gap, logically they will be treated the same by the algorithms. These are called indels. However, end gaps, the GCG tildes, should not be represented as indels because a lack of information beyond the length of a given sequence may not be due to a deletion or insertion event. It may have nothing to do with the particular stretch being analyzed at all. It may just not have been sequenced! The tildes are just placeholders for the sequence. Therefore, when preparing datasets, you may want to manually edit the output from any alignment format conversion to change leading and trailing gap symbols to either "x"s ("unknown amino acid") or "n"s ("unknown base") or "?"s ("unknown amino acid or base or deletion") depending on the situation. Even though many phylogenetic inference methods may not discriminate between gaps and unknowns, this will assure that none will make this particular incorrect assumption.

The two systems also treat files differently. PHYLIP uses a set of standard input and output filenames of the format infile, outfile, outtree, and plotfile. Because of this, and because of UNIX's lack of a version number, if you run concurrent PHYLIP analyses from batch scripts, you should run them in separate directories, otherwise the output from one program could overwrite the output of another. Furthermore, even when running PHYLIP programs sequentially, and even though the programs ask you whether you want to overwrite or not:

   the file "outfile" that you wanted to
   use as output file already exists.
   Do you want to Replace it, Append to it,
   write to a new File, or Quit?
   (please type R, A, F, or Q)

It is safest to just rename every output file as it is produced, or the same 'clobbering' effect could destroy your results when the next program is run, if you're not careful. This process can be quite annoying and bothersome but it is vitally important!
The length of time PAUP* or PHYLIP takes to process any particular data set depends on the nature of the data being used and the program doing the processing. Some of the programs are just very slow by nature, e.g., maximum likelihood; others only slow down when a large number of sequences are being worked with that are relatively long (i.e. over around 500 bases or amino acids). The individual program documentation is good about describing how slow each program is. Peruse each program's documentation prior to running for the first time. If you're already familiar with molecular phylogenetic evolutionary analyses techniques, you may only need to read PHYLIP's main.html and the GCG PAUPSearch and PAUPDisplay Help pages. Otherwise, read all of each program's documentation to get the entire breadth available.

Because many programs for phylogenetic inference can be very slow when faced with large data sets, often it is best to run them in background mode. Determining when to do this is based on both the program and the size of your dataset — it's largely a matter of experience. Because sooner or later most all the programs in PAUP* and PHYLIP will become too slow to run interactively with the datasets you'll be wanting to analyze, learning to do these analyses in the background is very important! I provide some batch mode PHYLIP documentation and template parameter and script files to assist you in operating this way. Review this information prior to running any PHYLIP programs in this manner. When background processing is required, copy over the desired file, edit it to reflect the data that you are working with, and then submit the job. It is easiest to run PAUP* jobs in the background through SeqLab, though native PAUP* jobs can be scripted in a similar manner as PHYLIP jobs by incorporating all the necessary commands in a NEXUS file.

I reiterate the most important factor in inferring reliable phylogenies is the accuracy of your multiple sequence alignment. The interpretation of your results is utterly dependent on the quality of your input. In fact, many experts advice against using any parts of the sequence data that are at all questionable. Only analyze those portions that assuredly do align. If any portions of the alignment are in doubt, throw them out. This usually means trimming down or masking the alignment's terminal ends and may require internal trimming or masking as well. SeqLab makes this process much easier with the Mask function, or you can use PHYLIP's Weight option. Either allows the user to differentially weight different parts of their alignment to reflect their confidence in it. It can be a handy trick with some data sets, especially those with both highly conserved and highly variable regions. Just like most computational molecular biology techniques, one is always balancing signal against noise — and it can be quite the balancing act! Too much noise or too little signal, either degrade the analysis to the point of nonsense.

**Molecular Evolution Workshop: A ‘Real-Life’ Project Oriented Approach**

I will use **bold** type in this tutorial for those commands and keystrokes that you are to type in at your console or for buttons that you are to click in SeqLab. I also use bold type for **section headings**. Screen traces are shown in a ‘typewriter’ style Courier font and ‘////////////’ indicates abridged data. The arrow symbol, ‘>’ indicates the system prompt and should not be typed as a part of commands. Really **important statements may be underlined**.
Specialized ‘X-server’ graphics communications software is required to use GCG’s SeqLab Graphical User Interface (GUI). This needs to be installed separately on personal style ‘Wintel’ or Macintosh machines but comes standard with most UNIX operating systems. The details of X and of connecting to the GCG server on campus will not be covered in this tutorial. If you are unsure of these procedures ask for assistance in the computer laboratory. I am also available for individualized personal help in your own laboratories. If you are having difficulties connecting to the GCG server from there, just contact me at stevet@bio.fsu.edu. A couple of tips at this point should be mentioned though. X-windows are only active when the mouse cursor is in that window. Furthermore, rather than holding mouse buttons down, to activate items, just click on them; and buttons are turned on when they are pushed in and shaded. Also do not close windows with the X-server software’s close icon in the upper right- or left-hand window corner, rather, always use GCG’s “Close” or “Cancel” or “OK” button located near the bottom of the window.

This workshop will explore several methods for inferring molecular phylogenies. You will use a sequence alignment of Elongation Factor 1α genes that I have already prepared and refined in order to save you time. If you took my “Introduction to SeqLab” workshop, you’ll recognize the dataset as similar to the one used there. However, for the purpose of this tutorial I have added the corresponding aligned nucleic acid sequences to the aligned peptide sequences.

The alpha subunit of Elongation Factor-1 (EF-1α) in Eukaryota and Archaea, known as Elongation Factor Tu in [Eu]Bacteria (and Euk’ and Arch’ plastids) is ubiquitous to all cellular life. It is crucial to the universal process of protein biosynthesis and promotes the GTP-dependent binding of aminoacyl-tRNA to the A-site of the intact ribosome. GTP is hydrolyzed to GDP in the process. Because of this strong evolutionary pressure resulting in very slow divergence and because of its ubiquity, it should be an appropriate gene on which to estimate early life questions. In fact, a series of papers in the early-90’s, notably those by Iwabe, et al. (1989), Rivera and Lake (1992), and Hasegawa, et al. (1993) all base ‘universal’ trees of life on this gene. Iwabe, et al. used the trick of aligning the α gene parologue EF-1β to their α dataset to root the tree. Elongation Factor 1α/Tu has guanine nucleotide, ribosome, and aminoacyl-tRNA binding sites. There are three distinct types of elongation factors that all work together to help perform the vital function of protein biosynthesis. The other two have the following names (the nomenclature in Archaea has not been completely worked out and is often contradictory):

<table>
<thead>
<tr>
<th>Eukaryota</th>
<th>[Eu]Bacteria</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-1β</td>
<td>EF-Ts</td>
<td>Interacts with EF-1α/EF-Tu to displace GDP allowing the regeneration of GTP-EF-1α/EF-Tu</td>
</tr>
<tr>
<td>EF-2</td>
<td>EF-G</td>
<td>Binds GTP and peptidyl-tRNA and translocates the latter from the A site to the P site.</td>
</tr>
</tbody>
</table>

In EF-1α/EF-Tu, a specific region is involved in a conformational change mediated by the hydrolysis of GTP to GDP. This region is conserved in both EF-1α/EF-Tu and EF-2/EF-G and seems to be typical of GTP-dependent proteins which bind non-initiator tRNAs to the ribosome.

In E. coli EF-Tu is encoded by a duplicated loci, tufA and tufB located about 15 minutes apart on the chromosome at positions 74.92 and 90.02 (ECDC). In humans at least twenty loci on seven different chromosomes
demonstrate homology to the gene. However, only two of them are potentially active; the remainder appear to be retropseudogenes (Madsen, et al. 1990). It is encoded in both the nucleus and mitochondria and chloroplast genomes in eukaryotes and is a globular, cytoplasmic enzyme in all cases.

The three-dimensional structure of Elongation Factor 1α/Tu has been solved in more than fifteen cases. Partial and complete E. coli structures have been resolved and deposited in the Protein Data Bank (1EFM, 1ETU, 1DG1, 1EFU, and 1EFC), the complete Thermus aquaticus and Thermus thermophilus structures have been determined (1TTT, 1EFT, and 1AIP), and even cow EF-1α has had its structure determined (1D2E). Most of the structures show the protein in complex with its nucleotide ligand, some show the ternary complex. The Thermus aquaticus structure is shown on the right as drawn by NCBI’s Cn3D molecular visualization tool.

Half of the protein has well defined alpha helices and the rest is rather unordered coils partly defined by beta strands. GTP fits right down in amongst all the helices in the pocket. The Thermus aquaticus structure has six well-defined helices that occur from residue 24 through 38, 86 through 98, 114 through 126, 144 through 161, 175 through 184, and 194 through 207. There are also two short helices at residues 47 to 51 and 54 to 59. The guanine nucleotide binding site involves residues 18 to 25, residues 81 to 85, and residues 136 to 139. Residue 8 is associated with aminoacyl-tRNA binding.

1) Log onto your GCG account and launch SeqLab

Each participant should use a different UNIX account. Either login with your existing account and password, or use your new one for the workshop. Use the appropriate connection commands on the personal computer or terminal that you are sitting at to launch X and log onto the FSU GCG UNIX host computer, “mendel.scs.fsu.edu.” An X-style terminal window should appear on the desktop after a few moments, if it doesn’t, launch one with the appropriate command. Get my assistance for this step if you are unsure of yourself. There are too many variations in method for them all to be described here.

The GCG package should have initialized automatically as soon as your terminal window launched. If it didn’t, type the command “gcg” (without the quotes) at the system prompt in the terminal window to start it up now. This process activates all of the programs within the package and displays the current version of both the software and all of its accompanying databases.

First let’s look at the alignment that I have prepared for your use. Use the GCG command “fetch” to pull the file into your account from the GCG public databases where I put a copy (remember, UNIX is case sensitive):

```
> fetch EF1a-Tu.rsf    that's a one not an 'el'
```
Now issue the command “seqlab &” (again, without the quotes) to fire up the SeqLab interface. The ampersand, “&,” is not necessary but really helps out by launching SeqLab as a background process so that you can retain control of your initial terminal window. This should produce two new windows, the first an introduction with an “OK” box; check “OK.” You should now be in SeqLab’s List mode.

Before beginning the analyses, go to the “Options” menu and select “Preferences...” unless you’ve already seen these in previous sessions. The options there allow SeqLab to run in its most intuitive manner. The defaults are usually fine, but I want you to see what’s available to change. Remember, buttons are turned on when they’re pushed in and shaded.

First notice that there are three different “Preferences” settings that can be changed: “General,” “Output,” and “Fonts;” start with “General.” The “Working Dir. . .” setting will be the directory from which SeqLab was initially launched. This is where all SeqLab’s working files will be stored; it can be changed in your accounts if appropriate, however, leave it as is for now. Be sure that the “Start SeqLab in:” choice has “Main List” selected and that “Close the window” is selected under the “After I push the “Run” button:” choice. Next select the “Output” “Preference.” Be sure “Automatically display new output” is selected. Finally, take a look at the “Fonts” menu. If you are dealing with very large alignments, then picking a smaller Editor font point size may be desirable in order to see more of your alignment on the screen at once. Click “OK” to accept any changes.

Be sure the “Mode:” “Main List” choice is selected in your main window and then go to the “File” menu. Pick “Add sequences from” and select “Sequence Files.” (GCG format compatible sequences or list files are accessible through this route. Use SeqLab’s Editor “Import” function to directly load GenBank format sequences or ABI style binary trace files without the need to reformat.) This will produce an “Add Sequences” window from which you can select sequences to add to your working.list. The “Filter” box can make your life easier here. Add the extension “.rsf” after the “*” wild card, to restrict the listing to only RSF files, and then press the “Filter” button. Select the file entitled “EF1a-Tu.rsf” from the “Files” box, and then check the “Add” button, and then the “Close” button at the bottom of the window to put the file in your working.list. It will appear in the SeqLab “Main List” window. Be sure it is selected and then switch to “Editor” “Mode:” to load the sequences into the SeqLab editor. Drag the window to an appropriate size by ‘grabbing’ the bottom-right corner of its ‘frame’ and ‘pulling’ it out as far as desired. Notice that the sequences now appear in the editor window with the residues and bases color-coded. DNA is colored with four colors for the four bases; amino acids are grouped into nine colors based on a UPGMA clustering of the BLOSUM62 matrix, and roughly correspond to physiochemical classes. The protein sequences are arranged at the top of the display and the corresponding DNA sequences are displayed in the same order at the bottom of the display. Use the vertical scroll bar to see them all. Any portion of or the entire alignment loaded is now available for analysis by any of the GCG programs. The display will look similar to the graphic presented at the top of the following page:
Explore the editor interface for a bit. Nearly all GCG programs are accessible through the "Functions" menu. The scroll bar at the bottom allows you to move through the sequences linearly. You can select sequences in their entirety by clicking on their names or you can select any position(s) within sequences by 'capturing' them with the mouse. You can select a range of sequence names by <shift><clicking> the top-most and bottom-most name desired, or <ctrl><click> sequence entry names to select noncontiguous entries. (However, there is a bug in the Linux version of SeqLab. Use <ctrl><right-click> as a work-around.) Quickly double-click a sequence name (or click on the “INFO” icon with its name selected) to get its full database reference documentation and then “Close” the “Sequence Information” window. The “pos:” and “col:” indicators show you where the cursor is located on a sequence without including and with including gaps respectively. The “1:1” scroll bar near the upper right-hand corner allows you to ‘zoom’ in or out on the sequences; move it to 2:1 and beyond and notice the difference in the display. Go to the “Display:” box and change it from “Residue Coloring” to “Feature Coloring.” The colors are now based on the annotation in the database Feature Table for the entry. Change the “Display:” to “Graphic Features;” now the features are represented using the same colors as before but in a ‘cartoon’ fashion. Use the mouse to move your cursor to one of the colored areas (the protein sequences will have this Feature information much more so than the DNA sequences). Quickly double-click it (or use the “Features” selection under the “Windows” menu). This will produce a new window that describes the features located at the cursor. Click on one of the features to get more information on it and to select it in its entirety. All the features are fully editable through the “Edit” check box in this panel and new features can be added with several desired shapes and colors through the “Add” check box. The display will look similar to my example below. I have used a 16 to 1 zoom factor and I selected one of the GTP binding domains in the E. coli elongation factor Tu protein sequence:
2) GCG’s Mask operation

The SeqLab “Edit” menu allows you to easily create several types of consensus. In addition to standard consensus sequences using various similarity schemes, SeqLab also allows you to create consensus “Masks” that screen specified areas of your alignment from further analyses by specifying 0 or 1 weights for each column. These Masks can even be modified by hand if so desired. Masks can also be created manually through the “New Sequences” menu. They can have position values all the way up to 9, though I doubt anyone would want any column of an alignment to be nine times as important as some other column. Masking can be very helpful for phylogenetic analysis by excluding those less reliable columns in your alignment where you are not confident in the positional homology without actually getting rid of the data. At this point be sure all of the DNA sequences are selected, but not the amino acid sequences at the top of the alignment nor the two Mask sequences that I have already prebuilt for you at the bottom of the alignment, and then create your own Mask style sequence consensus of them by going to the “Edit” “Consensus...” menu and specifying “Consensus type:” “Mask Sequence.” The default mode is to create a consensus at the 2/3’rds plurality level (“Percent required for majority”); however, this is a very high value for phylogenetic analysis and would likely not leave much phylogenetically informative data. Therefore, experiment with different lower plurality values to see the difference that it can make in the appearance of the alignment. Be sure that “Shade based on similarity to consensus” is checked to generate a color mask overlay on the display to help in the visualization process. (When making a normal sequence consensus of a protein alignment, rather than a weight Mask, you can generate a gray intermediate similarity color as well as the black and white representation seen here. This is a nice way to prepare alignment figures for publication.) The screen dump at the top of the following page illustrates my example using a plurality of 15%:
Few areas are excluded by the Mask in this alignment because of the similarity of this group of sequences. This is as it should be, for excluding many more columns in this particular alignment would likely leave nearly identical sequences and it would be impossible to ascertain how they are related. In fact, when dealing with very similar sequences, it is usually best to align the DNA sequences along with their corresponding proteins, as I've done in this example (the “Group” function is very helpful for this). Then perform the phylogenetic analyses on the DNA rather than on the proteins, as you will be doing in this tutorial. This is especially important when dealing with datasets that are quite similar since the proteins may not reflect many differences hidden in the DNA. Furthermore, many people prefer to run phylogenetic analyses on DNA rather than protein regardless of how similar they are — the multiple substitution models are much more robust for DNA, and, in fact, many phylogenetic inference algorithms do not even take advantage of amino acid similarity when dealing with protein sequences, they only count identities! However, the more diverged the dataset becomes, the more random third and eventually first codon positions will become, which will introduce noise (error) into the analysis. Therefore, often third positions are masked out. My second Mask line, entitled “doublet_mask,” does this and can be taken advantage of in the tutorial, if desired.

The logic to this paired protein and DNA alignment SeqLab approach is as follows:

1) The easy case where you can align the DNA directly. If the DNA sequences can be aligned directly because they are quite similar, then merely create your DNA alignment. Next use the “Edit” menu “Translate” function and the “align translations” option to create aligned corresponding protein sequences. Select the region to translate based on the CDS reference in each DNA sequence’s annotation. Be careful of CDS entries that do not begin at position 1 — the GenBank CDS feature annotation “/codon_start=“ identifies which position the
translation begins within the first codon listed. You may also have to trim sequences down to just the relevant exons/gene, especially if they’re genomic. You’ll have to change their protections with the padlock icon if this is the case. Group each protein to its corresponding DNA sequence so that subsequent manipulations will keep them together.

2) The way more difficult case where you need to use the protein sequences to create the alignment because the DNA cannot be directly aligned. In this case, load the protein sequences first, create their alignment, and then load the corresponding DNA sequences. Find the DNA sequence accession codes in the annotation of the protein sequence entries. Next, translate the unaligned DNA sequences into new protein sequences with the Edit-Translate function using the “align translations” option and Group these to their corresponding DNA sequences, just as above. However, this time the DNA along with their translated sequences are not aligned as a set, just the other protein set is aligned. Also, Group all of the aligned protein dataset together, separately from the DNA/aligned translation set. Then rearrange your display to place the DNA, its aligned translation, and the original aligned protein sequence side-by-side. Now comes the manual part — unfortunately GCG does not include an automated way to achieve this in SeqLab — manually slide one set to match the other. Use the “CUT” and “PASTE” buttons to move the sequences around. When pasting realize that the “Sequence clipboard” contains complete sequence entries, whereas the “Text clipboard” only contains sequence data, amino acid residues or DNA bases as the case may be. The translated sequence entries can be “CUT” away after they’re aligned to the rest of the set. Merge the newly aligned sequences into the existing alignment Group as you go and then start on the next one. It sounds difficult, but since you’re matching up two identical protein sequences, the DNA translation and the original aligned protein, it’s really not too bad. The Group function keeps everything together the way it should be so that you don’t lose your original alignment as you space residues apart to match them up to their respective codons. Some codons may become spaced apart in this process and will have to be adjusted afterwards. As usual, save your work often.

Once a Mask has been created in SeqLab, most of the programs available through the “Functions” menu will use that Mask, if the Mask is selected along with the desired sequences, to weight the columns of the alignment data matrix appropriately. This only occurs through the “Functions” menu.

When you’ve found a plurality combination that you like, you can use the “File” “Print. . .” command, changing “Output Format:” to “PostScript,” to prepare a PostScript file of your SeqLab display. You can play around with the other parameters as you like — notice that as you change the font-size the number of pages to be printed varies. In the “Print Alignment” menu specify “Destination. . . File” and give an appropriate filename and then click “OK.” This command will produce a graphics file using the PostScript language of whatever you have displayed in the editor window. Don’t do this here, but it is a great way to prepare presentations for your own research. It will result in a PostScript file of the alignment using the displayed coloring and the specified parameters to be created in the directory that you launched SeqLab from which can then be transferred to another machine for color PostScript printing, or for importing into PostScript savvy programs for further manipulation, or that can be printed to a black and white laser printer that will simulate the colors with gray tones.
Select the consensus Mask entry(ies) (only) that you just created and “CUT” it(Them) from the alignment before proceeding with the remainder of the tutorial. Do not cut any of the sequence data nor the two Masks that I provide. I want you to use my Masks instead of your own. This is because I have carefully modified them to maximize phylogenetic signal and to consider codon positions. I provide two Masks, one including and one excluding third codon positions, named “triplet_mask” and “doublet_mask” respectively, that you can experiment with.

3) Phylogenetic tree estimation — GCG style

3.a. Distance methods: GCG’s Distances and GrowTree

To begin be sure that all of the DNA sequences (only — no proteins!), as well as the first Mask, the one named “triplet_mask” that I provided, are selected. First let’s use GCG’s linked distance matrix and tree drawing combination method for phylogenetic inference. The evolutionary distance between each sequence is first calculated by the program Distances, compensating for multiple substitutions at homologous sites, and then those distances are used in a neighbor-joining algorithm in the program GrowTree to estimate an evolutionary tree for the data. Go to SeqLab’s “Functions” “Evolution” “Distances…” menu to begin the programs. Once the “Distances” program window is displayed, press the “Help” button to read the complete description of this molecular evolution program — it describes problems and considerations in this whole area very well.

Help is always available when using the Wisconsin Package. Every SeqLab program window and SeqLab’s Main window have “Help” buttons. Furthermore, issuing the command “genhelp” in a command line terminal window launches a text driven Help system, and, probably best of all, the URL http://www.scs.fsu.edu/gcg/ links to a Web version of the Help system. Launch a Web browser window now to explore GCG’s Help system. The Web version will request a username and password if connecting from an off-campus location, to comply with GCG’s license restrictions. Give the username “gcguser” and the password “4mendel,” if this is the case. Find the “Distances” help page; pay particular attention to the “ALGORITHM,” “CONSIDERATIONS,” and “SUGGESTIONS” sections of the text.

After looking over the help screens “Close” the “SeqLab Help” window and accept the “Jukes-Cantor distance” default “Distance Correction Method” and then punch the “GrowTree…” button. Be sure “Neighbor joining” is checked and then press GrowTree’s “GrowTree Options…” button. Whatever you do, do not use UPGMA. UPGMA imposes an absolutely uniform molecular clock across all of your data. This is seldom, if ever, the situation. I like to check in “Report negative branch lengths as zero” in the “Options” menu to make reading the tree a bit easier in case the algorithm calculates any nonsensical negative branches. Be sure that “Display Tree As:” “phylogram” is checked to show branch lengths in the output and then “Close” the “GrowTree Options” and main “GrowTree” program menu boxes. Be sure that “Background Job” is checked and then press “Run” to execute the program. Distances will run and then the GrowTree program will automatically be launched by SeqLab to infer the neighbor-joining tree from the distance data.
The results will automatically be displayed when the programs finish. The top-most file shown will be the NEXUS format 'Newick' tree file. This tree file is written in a nested parenthetical connotation known as the Newick standard. Many phylogeny programs, including PHYLIP, can work with this format of nested parentheses to draw graphical representations of evolutionary trees; “Close” the tree file. The “Seqlab Output Manager” window will be next; go ahead and “Close” it. This very important window, always available through the SeqLab “Windows” menu, contains all of the output from your current SeqLab session. Files may be displayed, printed, saved with other names or other locations, and/or deleted from this window. The GCG Figure file graphic of the tree will next be visible. Check it out to see if it makes sense. Here’s my GrowTree phylogram inferred with the default Jukes-Cantor model and no Distances options:

Vertical branch lengths are proportional to evolutionary divergence and have the unit of substitutions per 100 characters. The exact numbers can be seen in the program’s output .trees file. I recognize many problems with the default inference. Among them, most of the protists are forming a discrete clade along with yeast. (A clade is all the members of any lineage that all have a common ancestor, i.e. all share a common node in a phylogenetic tree.) Based on our understanding of the universal 'tree of life,' largely from ribosomal RNA phylogenies, the protists in this dataset should not all be within the same clade and yeast should be completely separate. In fact, fungi are most closely related to animals. The only two protists that should segregate together are falciparum and pyriformis, both classified as Alveolata. In this analysis those two taxa are at opposite ends of a protist clade. And Giardia should not be included with the Archae marismortui, nor should Euglena be in with the plants, but both genera are often problematic. Furthermore, the branching order in the Archaeal and Bacterial clades seems a bit odd. At least the animals, Caenorhabditis, Drosophila, Xenopus, and Homo, all share a common clade as do the two plants, Arabidopsis and Triticum. It appears as if the Jukes-Cantor model was too much of an oversimplification with this data, as it usually is.
To see the difference various evolutionary models and options can make go back to the “Distances” program and change the model. A quick way to repeat an analysis performed in a SeqLab session is to use the shortcut feature saved under the “Windows” menu. This menu keeps track of all programs run in your current SeqLab session.

Back in the “Distances” program window change the “Distance Correction Method” to “Jin-Nei gamma distance” and then press “Options...” to see what else is available. An important parameter here describes the base substitution rate heterogeneity across sites in your sequences. This is called gamma. The default gamma distribution shape parameter \( \alpha \) has a value of 1.0 in GCG’s Jin-Nei Distances option; this may or may not be appropriate. Another very important consideration, mentioned previously, and available in this program by option, without using a separate mask, is to ignore the third codon position along every sequence. This is because third positions vary tremendously due to the redundancy of the DNA code resulting from the “Wobble” hypothesis and quickly reach saturation where they contribute nothing but noise, which causes error, to the analysis. Therefore, change the “Base Position(s) To Analyze” selection to “First and second positions of each triplet.” Next “Close” the “Distances Options” window and then press “Run” in the main “Distances” program window. My Jin-Nei tree where only first and second positions of each codon (as long as they are not masked out) is scored follows below. Notice the differences between this tree, shown on the following page, and the former Jukes-Cantor tree where all unmasked positions contributed equally to the analysis:

The main difference that I notice is the yeast sequence now has a lineage of its own in the more accurate Jin-Nei tree although it is still improperly rooted within the protists. Whether this improvement is due to the model or because we ignored third positions would have to be tested. Most protists are also now spread out along the base of the tree as they should be rather than forming a cohesive clade, though Euglena continues to present a problem. The Archae marismortui is also now in the Archael clade where it belongs. Parameters and models can make a huge difference.
The text output files from Distances can be displayed with the "Output Manager" if so desired. These files contain the distance matrices of all pairwise evolutionary distances between all of the sequences. They were automatically fed to GrowTree in order to calculate the trees. If your sequences are too divergent and you are using the Jukes-Cantor DNA model, then you will get a warning message at the conclusion of the run stating how many of your sequence pairs have a distance greater than 100. All models tend to break down when many sequence pairs have greater than 100 substitutions per 100 residues. This will lead to interpretive and reliability problems with the analysis. Reliability can be increased in these cases by further editing of the initial alignment to exclude the most diverse regions and/or sequences. Multiple substitutions at homologous sites are always the problem. As explained in the introduction, this phenomenon increasingly confounds evolutionary reconstruction as divergence between the members of a dataset increases. Neighbor-joining is not the best of methods, but they give a good, quick estimate. Several PHYLIP alternatives to this GCG program as well as GCG's version of PAUP* will also be explored in this workshop.

3.b. GCG's interface to PAUP* — PAUPSearch and PAUPDisplay

Next I’ll introduce GCG’s implementation of David Swofford’s PAUP* (pronounced ‘pop star’) package. Use the following command in a terminal window to read the GCG PAUP* license agreement, if you’re curious:

```
> typedata paup-license.txt
```

The PAUP package was originally written to only do parsimony analysis with either DNA sequences or morphological character data using a Macintosh. It latest incarnation, version 4.0+, changed the package's name by adding the asterisk which means “and other methods” referring to the incorporation of distance methods and the maximum likelihood method to the package. It was also expanded into a “portable” package capable of being run on many different platforms using a command line interface in addition to its original Mac pre-OS X graphical user interface. PAUP* should generally not be used with protein sequences as it does not incorporate any protein models of evolution other than a crude amino acid identity model. However, more sophisticated protein models can be used by embedding the necessary commands and matrices in the NEXUS file used as input to the package. Though, as I discussed previously, many people prefer to perform evolutionary inference with DNA sequences anyway. Furthermore, PAUP*’s DNA models are perhaps the most sophisticated available in any molecular phylogenetic inference software, and I, therefore, heartily recommend using it for DNA datasets.

GCG’s interface to the PAUP* package is the paired programs PAUPSearch and PAUPDisplay. These paired programs provide easy to use access to a subset of PAUP*, and their use for evolutionary inference will be demonstrated here. However, for serious phylogenetic analysis you may want to consider running PAUP* exterior to GCG in native mode, directly at the command line, to take advantage of the complete package’s substantial power. Alternatively, you can get the latest version directly from Sinauer Associates, the publishing company that distributes the software, and install it on your own system. Sinauer, can be found on the World Wide Web at http://www.sinauer.com/, or by e-mail at publish@sinauer.com. However, if you do run PAUP* in command line mode, then PAUPSearch can be used as a very handy tool for generating NEXUS format files.
3.b.1. NEXUS format and using PAUPSearch and PAUPDisplay for phylogenetic inference

Before accepting any of PAUP*’s output as valid phylogenetic inference, regardless of the version you use, you should learn how to run the most robust searches possible. Here we will start by using PAUP* in a very simple fashion in its default parsimony mode and with very few of the more sophisticated options. We will then briefly check out some of the more powerful algorithms and options available, but will in no way fully explore the PAUP* package. My intent is merely to expose you to the package as it is incredibly powerful, sophisticated, and complex. Unfortunately learning all PAUP* has to offer is completely beyond the scope of the present workshop. If you are more interested in this package, I would recommend first carefully reading all of GCG’s documentation on their PAUP* interface, and then, for all of the nitty-gritty details, a PAUP* command reference manual is available (http://paup.scs.fsu.edu/Cmd_ref_v2.pdf).

Begin the process by being sure all of the DNA sequences and one of the weight Masks are selected in your “Main Window” display, and then go to the “Functions” “Evolution” menu. Select “PaupSearch…” to launch the dialogue box. (In my examples I’ve used the “triplet_mask,” but you are welcome to see the difference that masking third positions makes by using the “doublet_mask” instead.) First accept the default “Tree Optimality Criterion” “maximum parsimony” and the “heuristic tree search (fast)” “Method for Obtaining Best Tree(s).” Be sure that the “perform bootstrap replications…” button is not pressed and then launch the “Options” menu by pressing the appropriate button. In the “PaupSearch Options” menu scroll down through the options until you get to the portion that talks about “How to Create Initial Tree” “stepwise addition.” Specify “choose sequences at random” and then supply any random number “Initial seed;” do not change any of the other stepwise addition parameters including the “10” “random addition sequence replications.” “Close” the options menu. The PAUPSearch options screen should look like the following graphic:

[Diagram of PAUPSearch options screen]

27
Next, be sure that the “PaupDisplay. . .” button is activated and then press it to get its corresponding menu. “describe and plot tree(s)” should be checked there as well as “maximum parsimony” “Optimality Criterion.” Press the “Options. . .” button to see PAUPDisplay’s options menu. You can specify an outgroup for your analysis in this menu; I’ll set it to “13” to force the Giardia sequence into an outgroup position. This is often the most diverged sequence in your dataset and can be used to root the inferred tree. Its designation should be based on external knowledge, i.e. if you know, based on other studies, that a particular sequence is not of the same clade as the rest of your sequences, then you can designate it as an outgroup. Often this external data relates to fossil evidence. Another trick is to use a parologue as an outgroup, if the gene duplication event that produced the parologue preceded the speciation event that produced the taxa. In all cases, your designated outgroup should be as close to your ingroup as possible without actually being a part of it. Giardia will be used here since several other molecular phylogenies have placed it in the Eukaryote lineage very near the base of the universal tree of life. It’s not a true outgroup since it is a member of the ‘tree of life,’ but it will add directionality to the tree. Be sure to change the default “cladogram” representation to a “phylogram.” This enables branch lengths to reflect evolutionary divergence based on the optimality criterion being used, here parsimony. Leave all other PAUPDisplay options at their default and then “Close” the options and main PAUPDisplay menus. Be sure that “How: “Background Job” is specified on the main PAUPSearch menu and then press “Run” there. To check on the progress of the job you can go to SeqLab’s “Windows” menu and choose “Job Manager.” Select the “PaupSearch/PaupDisplay” entry to see its progress and then close the window.

The results will be displayed shortly. You should see the “SeqLab Output Manager” window and a phylogram graphic drawn from the PAUPDisplay output figure file; check it out and then “Close” it. Notice the differences between this tree and what we’ve previously seen. The most obvious difference is the Giardia outgroup placed there by option. However, also notice that, with the exception of Giardia, and Euglena, which is again segregating with the plants, the protists are again forming a discrete clade along with yeast as they did in the initial Jukes-Cantor analysis. As stated previously, we do not believe this is true. Furthermore, marismortui is an Archae, not a Bacteria. Clearly, this analysis also has many problems, perhaps third position saturation, perhaps ‘long branch attraction’ — it’s hard to say without further testing. Compare your tree to mine, if you chose my doublet_mask for your analysis. Is it any better? My PAUPSearch/PAUPDisplay parsimony tree follows below:
Use the “Output Manager” to “Display” the PAUPtrees file. This is the text output from PAUPDisplay that describes the tree with several statistical measures and contains the NEXUS formatted alignment as well as the Newick format trees discovered by the search. Parsimony can and often does find more than one shortest tree. The graphics window allows you to “Page” through them when more than one is discovered. “Close” the PAUPtrees file and the graphics window and the “Output Manager.”

We’ll use a minimum evolution distance method next. Assure that all the DNA sequences and whichever Mask you want to use is selected and then relaunch “PAUPsearch.” Change the default “Tree Optimality Criterion” from “maximum parsimony” to “distance (minimum evolution)” and use the default “heuristic tree search (fast)” “Method for Obtaining Best Tree(s).” Also be sure that the “perform bootstrap replications. .” button is still not pressed and then launch the “Options” menu. In the “PaupSearch Options” menu scroll down to “How to Select Next Sequence for Stepwise Addition to Tree;” “choose sequences at random” should still be selected from the parsimony run. Supply another random number as the “Initial seed” and keep “10” “random addition sequence replications.” Next scroll down to “Distance Options” “Distance Correction Method” and pick a model. I’ll run it with the “Felsenstein 1984 distance (nucleotide only)” model but you can try something else. “Close” the options menu.

Next press the “PaupDisplay. .” button to get its corresponding menu. “describe and plot tree(s)” should still be checked but change the “Optimality Criterion” to “distance (minimum evolution).” It makes the most sense to use the same tree optimality criteria in PAUPDisplay as you use in PAUPSearch. PAUPDisplay’s “Options. .” should still specify the outgroup and “phylogram” representation that you previously chose. Scroll down to the “Distance Options” “Distance Correction Method” section and specify the same model there as you used in the PAUPSearch options window. “Close” the PAUPDisplay menus and then press “Run” on the main PAUPSearch menu. My parameters should create a figure similar to the following (otherwise, how does yours compare?):

![Diagram of phylogenetic tree]

My tree from this run is different from all the previous ones; yet there are still problems. At least all the animals are still sharing a clade as are the two plants. However, protists, with the exception of Giardia and Euglena, are
again being forced into a clade, and yeast is again joining it. A divided Archae clade further raises suspicion. Clearly we have again violated some systematic assumptions about the data in this analysis. Realize that even though we chose a Felsenstein 1984 (F84) model here, we did not explicitly set any of its variables. In fact the PAUPSearch/PAUPDisplay interfaces through SeqLab don’t even allow it. You need to run PAUP* exterior to GCG in order to take advantage of many of the options. Perhaps the default settings for the two F84 parameters transition/transversion ratio and/or base frequencies were the culprit or those saturated third positions caused my problem. If you ran the analysis with third positions masked out, i.e. my doublet_mask, was your inference any more satisfying? It really pays in this field to do your homework and learn about the various models and the importance of their parameters.

As mentioned above, PAUPSearch can also be used as a quick and easy way to generate NEXUS format for running PAUP* exterior to GCG. To do this, take advantage of the top check box in PAUPSearch’s Options window to save the PAUPScript file. Designate an appropriate file name. The PAUPScript output file results from the automatic conversion of your alignment to NEXUS format and contains default PAUP commands as well as your alignment. It contains the NEXUS format file that was generated by GCG to run PAUP*. Columns of your alignment with zeroes in their Mask are excluded from the NEXUS file. This file can be used to run a native version of PAUP* exterior to GCG if desired. Since PAUPSearch automatically creates this file for you, correctly encoding all of the required format data, there is no need to hassle with a later conversion of your alignment to NEXUS. As I stated in the introduction, file format conversion can be the biggest headache of this whole area and here GCG has done all of that work for you. When using this file as input to native PAUP* you need to comment out or remove any inappropriate commands within the command block with a simple text editor. Likewise, this file can be greatly expanded by encoding any desired commands within its command block. (If needed, the PAUPLog file keeps track of all that happened during the program run and is a good place to look for any error messages. It is, therefore, a handy file to save to avoid otherwise frustrating troubleshooting.)

Before leaving PAUP* for PHYLIP, let me briefly mention its maximum likelihood (ML) implementation (not available for amino acid sequence data without customization in native mode)— one of the best around. ML is arguably the most accurate and certainly the most powerful molecular phylogenetic inference algorithm available. But, before starting any turf wars, let’s get it straight right up front. ML used in its most accurate method, i.e. letting it estimate all of its own model parameters and estimating the best tree simultaneously, is largely impossible to achieve with most datasets and most computers! In spite of that, I consider ML to be the method of choice; however, it requires several tricks to enable analyses to be completed in the researcher’s lifetime. The biggest trick is to use it to estimate parameters on a user defined ‘reasonable’ tree, perhaps a neighbor-joining tree or a parsimony tree, and then specify those estimated parameters to find a better tree. You can iterate this process over and over, refining your estimates with each pass and testing the results against one another with likelihood ratio tests and other statistical methods. The cpu savings can be significant — analyses that would be impossible, if you were estimating all the parameters and the best tree simultaneously, can be realistically run. These parameters can make a big difference in the outcome of the inference. As we’ve already seen, the model really does matter. As mentioned previously, one parameter that can make a huge difference is the gamma rate
heterogeneity distribution parameter $\alpha$ so be sure to estimate a valid one. A very helpful Web resource with this process is found at: http://inbio.byu.edu/Faculty/kac/crandall_lab/modeltest.htm.

Unfortunately this iterative process of specifying a user defined tree to estimate parameters and then using those parameters to find a better tree is not practical using the GCG PAUPSearch interface. You’ll need to use PAUP*’s native mode. There are some real advantages to learning PAUP*’s native command language. As mentioned in the Introduction, to launch the native version of PAUP* use the “paup” command and then use the built in help system to learn about some of these commands. I can also provide more individual help in this area. I have only briefly shown you PAUP* and encourage you to explore the package at length with your own data.

4) Phylogenetic Inference using the PHYLIP package
   4.a. PHYLIP format

In the “SeqLab Main Window” go to the “File” “Export” menu; click “Format” in the new window and notice that several different formats are available for saving a copy of your RSF file. But do not export any of these formats at this point, and “Cancel” the window. Realize that using this export route does not use the Mask data to include or exclude columns from your alignment. In fact exporting sequences this way in GCG version 11 has proven problematic, and I’ve reported several bugs to the company. Since we want to take advantage of the Mask data for subsequent phylogenetic analyses, we will export our alignment using another method. Therefore, after being sure that all of the DNA sequences (but not the protein sequences!), as well as whichever Mask sequence that you wish to experiment with (again, my examples do not take advantage of the doublet_mask but you are welcome to experiment with it) are selected. Next, go to the “Functions” menu, where choices will be affected by the Mask, and choose “Importing/Exporting” “SeqConv+. . .” “Set the output format to: FastA” and press “Run” to convert those portions of the alignment that are not masked out into FastA format. FastA is a great intermediate format on our way to PHYLIP’s required format because it is so simple. However, the new file is not automatically displayed by SeqLab and is not listed in the Output Manager. The file will appear in your working directory with the name “seqconv+.fa.” Use your terminal window to look at it. Notice that it excludes those positions that were masked with zero and that it now follows all FastA format conventions including the automatic conversion of all GCG style gap periods and tildes to the more universal gap hyphen representation. This step, therefore, circumvents the common ‘dot to dash’ problem often encountered in sequence format conversion. The very first part of my FastA format output file is shown below:

```
> A.NIDULANS X17424 A.nidulans DNA for a region containing ORF150, rps12, rps7, f
ATGGCACGCCGCCACCAAGCTTCAACGCAAACATCGGGACCATCGGTCACGTTGACCACGGT
AAAACGACGCTGACG---------------------GCAGCTATC-----ACCACT----GTG
TTG----GCCAAGGCT----GCC--ATGGCTAAA----GCC----GCC----GCTTACGTGAT
--------ATC----GACCCAGCTTCGGGAAGAAAACTCTGTTGTTTACGATCAACAAGC
GCTCAGCTTGAAATACGAACCCGCAACCGTCATACGTTCACGAGCTTGTTGACCTCCGGTCAC
GCTGACTAGCTGAAAACATGATACGCTGCTGCGAGATGACGGCCATCTCCTGCTG
GTGCTGGCTGTCAGGCC------CCC----------ATGCCC--------CAAACTCGCC
GAGGACACTTCTCGTCGGGAAAACAGGGTGCTGCCGTCCCAACATCGTGTTTTTCTTTGACT
AAGAACAGCTTT---------GA------GACCGTGAACGTGGTGGATGTGAGATGC
GCTGAGCTCGTGAGCTCTGCCATGTTCTCGCTGCCGATGC---------ATGCCCAAT
GTG------GCTGCTGGCTGTCGCTGCGAAGCTTGAAGCTCAGGATGTTGAGGTC
AGCATACGGGCTGCTCAGGCTGCGGGCCTCTCAGGCTGCGAGTTGTTGAGGTC
---------------------------------------------------------------------------------------------------
```
Rename this file with a name that makes more sense to you ("mv" in your terminal window). We won't need SeqLab anymore in today's session, so exit it with the "File" menu "Exit" choice; save your RSF file and any changes in your list with appropriate responses. Accept the suggested changes and designate names that make sense. SeqLab will close. Do not logoff Mendel. Next, we can convert the FastA format file just produced to PHYLIP compatible format with ReadSeq.

We can now run Don Gilbert's ReadSeq program. A limitation of ReadSeq is it does not allow you to choose only a portion of an alignment, nor does it automatically convert dots and tildes to hyphens. However, since we've taken care of these points while in SeqLab, it'll work just fine for us here. ReadSeq runs a bit backward from what most people are used to. Begin the program by typing "readseq" at your command prompt in the terminal window. ReadSeq first prompts you for an appropriate output file name, not an input file. Do not make a mistake in this step by giving the name of your input file first. If you do, you will overwrite the input file, and then when it tries to read it, there will be nothing left to read! Next choose "12" off of the ReadSeq menu for the current PHYLIP format and then designate the input sequence. It most likely will not be named "EF1a-Tu.tfa" as mine is, unless you renamed it that. It may still have the cryptic default name "seqconv+.fa." (Do not use the GCG {*} designator; this is not a GCG program.) Finally, after the program has read all of the input sequences, specify "All" the sequences by typing the word "all." When the program again asks for an input sequence, press return to inform it that you are done, and let it do its thing. An abridged screen trace is shown below; as usual, user responses are shown in bold:

```
> readseq
readSeq (1Feb93), multi-format molbio sequence reader.

Name of output file (?=help, defaults to display):
EFla-Tu.phy
  1. IG/Stanford           10. Olsen (in-only)
  2. GenBank/GB            11. Phylip3.2
  3. NBRF                   12. Phylip
  4. EMBL                   13. Plain/Raw
  5. GCG                    14. PIR/CODATA
  6. DNAStrider            15. MSF
  7. Fitch                  16. ASN.1
  8. Pearson/Fasta         17. PAUP/NEXUS
  9. Zuker (in-only)        18. Pretty (out-only)

Choose an output format (name or #):
  12

Name an input sequence or -option:
EFla-Tu.tfa (or whatever SeqLab named your file, if you did not rename it with the Output Manager earlier!)
Sequences in EFla-Tu.tfa (format is 8. Pearson/Fasta)
  1)  A.NIDULANS X17442 A.nidulans DNA for a region containing ORF150, rps12, rps 7, fus, tufA. 6/
  2)  T.AQUATICUS X66322 T.aquaticus tufA and fus genes for elongation factor Tu and elongation fa
  3)  E.COLI X57091 E.coli tufB gene for translation elongation factor EF-Tu. 5/1 993
  4)  M.TUBERCULOSIS X63539 M.tuberculosis tuf gene for elongation factor TU. 7/1 993
  5)  M.GALLISEPTICUM X16462 Mycoplasma gallisepticum tuf gene for elongation fac
tor tu. 9/1993
  6)  T.MARITIMA M27479 T.maritima, elongation factor Tu (EF-Tu) gene, complete c
ds. 4/1993
```

32
20) A. THALIANA X16430 Arabidopsis thaliana EF-1 alpha A1 gene for elongation factor 1-alpha. 9
21) S. CEREVISIAE X00779 Yeast gene for elongation factor 1-alpha (EF-1-alpha). 9/1993
22) C. ELEGANS U51994 Caenorhabditis elegans cosmid R03G5. 4/1996
23) D. MELANOGASTER M1744 D. melanogaster 50 kDa protein F1 gene, complete cds. 4/1993
24) X. LAEVIS X55324 X. laevis EF-1alpha mRNA for elongation factor 1-alpha. 3/1991
25) H. SAPIENS X03558 Human mRNA for elongation factor 1 alpha subunit (EF-1 alpha). 9/1993

Choose a sequence (# or All):
all

Name an input sequence or -option: <rtn> (in other words, just press the return key)

Never mind if you happen to get a “... padded to fit” error message — the program is just doing what it is supposed to do. Do realize, though, that had we not used ReadSeq on the output from SeqConv+ to convert to PHYLIP, and had rather used a GCG MSF file as input, then an essential change would have to be made before it would be correct for PHYLIP. As mentioned before, periods and tildes will not work to represent indels (gaps); they must all be changed to hyphens. The following, rather strange, UNIX command works very well for this step from the command line, but you should not need to use it in this workshop:

> tr \-\. \- < infile.phy > outfile.phy

Run “more” on your new file to see what PHYLIP format looks like:

> more EFla-Tu.phy

The first part of that PHYLIP output file is displayed below:

25 1373
A. NIDULANS ATGGCACGCC GCACCAAGCC TCACGCAAAC ATCGGGACCA TCGGTCACGT
T. AQUATICU ATGGCGAAGC GGACGAAGCC CCACGTGAAC GTGGGGACGA TTGGGCACGT
E. COLI ATGTCTAAAC GTACAAAACC GCACGTTAAC GTCGGTACTA TCGGCCACGT
M. TUBERCUL GTGGCGAAGC GGACCAAGCC CCACGTCAAC ATCGGGACCA TCGGTCACGT
M. GALLISEP ATGGCAAAAC GTAGTAAACC TCACGTTAAT ATTGGAACAA TCGGTCATAT
T. MARITIMA ATGGCGAAGA GAACAAAACC GCATGTTAAC GTTGGAACGA TTGGACATAT
C. TRACHOMA ATGTCTAAAC GTAAATGGCC TCATATGCAA ATGAGGCCA TCGGCCACGT
S. SOLFAPAR ATG-------T CTACAAAGCC TCACCTTAAT TTAATAGTAA TAGGTCACGT
P. WOESEI ATG-------A AGGACAAGCC CCACGTTAAC ATTGTATTTA TTGGACACGT
H. MARISMOR ATG-------A GCGACGAACA ACACCGAAGC CTGGCCATTA TCGGCCACGT
M. VANNIELI ATGGCA-------A AAACAAAAAC ATTCCTTAAT GTGCTTTTTA TCGGCCACGT
T. ACIDOPHI ATGCCA-------A GTGACAAACC ACAGCTGAAT CTTGTCGTTA TCGGCCACGT
G. LAMBLIA NNN-------N NNNNNNNNN NNNNNNNNN NNNNNNNNN NNNNNNNNN
P. FALCIAP ATG-------A AGGAAAACAC ATATATTAC TTAATGTTTA TCGGCCACGT
D. DISCOIDE ATG-------T CGAACAAGAC ATATATATAC ATGTGTCGTTA TCGGCCACGT
T. PYRIFORM ATGGCTAGA-------GATAATGT CTGATTAAAC TTTGTCGTTA TCGGCCACGT
E. HISTOLYT ATG-------A AGGAAAAGAC ATATATATAC ATGTGTCGTTA TCGGCCACGT
E. GRACILIS ATG-------A AGGAAAAGAC GCACCTGATA CTGGTCGTTA TCGACCACGT
T. ABELIUM ATG-------A AGGAAAAGAC GCACCTGATA CTGGTCGTTA TCGACCACGT
A. THALIANA ATG-------A AGGAAAAGAC GCACCTGATA CTGGTCGTTA TCGACCACGT
S. CEREVISI ATG-------A AGGAAAAGCT TCATATTAAC ATGGTCGTTA TCGACCACGT
C. ELEGANS ATG-------A AGGAAAAGCT TCATATTAAC ATGGTCGTTA TCGACCACGT
D. MELANOGA ATG-------A AGGAAAAGCT TCATATTAAC ATGGTCGTTA TCGACCACGT
X. LAEVIS ATG-------A AGGAAAAGCT TCATATTAAC ATGGTCGTTA TCGACCACGT
H. SAPIENS ATG-------A AGGAAAAGCT TCATATTAAC ATGGTCGTTA TCGACCACGT

TGACCACGGT AAAAAAGACG TGACG------- -------GCAATGACG
GGACCACGCG AAAAAAGACG TGACG------- -------GCAATGACG

////////////////////////////////////////////////////////////////////

33
Notice that the file begins with two numbers; the first shows the number of sequences in the matrix and the second lists the length of the matrix including any gaps and ambiguities. The next section lists the names of the sequences truncated to ten characters, if necessary, along with all the sequences printed in an ‘interleaved’ fashion. Only the first sequence block lists the names, all others just give sequence data.

Regardless of how you go from GCG format to acceptable PHYLIP format, one more technicality should be looked at. As discussed in the introduction, you should evaluate the terminal ends of your data matrix. If any of the implied indels are uncertain (especially true if sequence lengths were different), then question marks, “?”, are usually more appropriate than hyphens. Leaving them hyphens could be misleading. As discussed earlier, gaps in the data are represented by deletion symbols, “-,” which is logically correct in most cases. However, gaps at the ends and beginnings of sequences probably should not have hyphens unless you really know that a deletion/insertion is responsible for the length discrepancy. Therefore, it is a good idea to edit the output from ReadSeq to replace leading and trailing hyphens in your alignment with question marks or the unknowns characters “n” or “x” depending on which is more appropriate, DNA or protein sequence respectively. Be very careful when changing these characters so that the alignment doesn’t shift out of phase. This is also an excellent point at which to verify that the sequence names are exactly as you wish them to appear in the final PHYLIP plots.

As mentioned in the introduction, PHYLIP sequence names can contain some punctuation and mixed capitalization, and can be up to ten characters in length. I have taken the bother to perform this question mark editing exercise with my PHYLIP input file for the rest of the tutorial; you needn’t bother. The file is now ready for PHYLIP analysis.

4.b.1. Another estimate of the distance matrix: PHYLIP’s DNADist

Let’s use the PHYLIP package now to see how its results differ. I will show the use of a couple of the distance methods in the package first. All the distance methods require that a distance matrix first be prepared from the alignment matrix. The GCG distance matrix is not suitable as input to PHYLIP; the formats are totally different. The programs are all launched with their name. If you have a file in the present directory named “infile,” the program will automatically read that, otherwise it will request an input file name. As discussed in the introduction, PHYLIP doesn’t allow you to specify output file names. A menu does allow you to pick desired options within the program. Several will be taken advantage of below. After you’ve made all your changes to the default settings, you respond with a “y” for yes at the prompt, and the process begins.

PHYLIP’s DNA distance matrix program, DNADist, allows the use of four alternate multiple substitution models. The default F84 model is much more powerful than the Jukes-Cantor model used earlier with GCG’s Distances program, and will contrast well with the last PAUP* F84 minimum evolution run, so let’s try it again. I’ll specify the same gamma $\alpha$ value of 1.0 (though PHYLIP programs use the “Coefficient of variation, 1/square root of $\alpha$) as Distances used with the Jin-Nei model by default, and I’ll accept the default PHYLIP transition/transversion ratio of 2.0, even though these values may not be correct. Begin the program by typing “dnadist.” When the program asks for an input file, specify your converted alignment. At the “Are these settings correct?” prompt type
"y" for yes. Issue the following command line in your terminal window and respond with the bold-faced text commands to produce the following screen trace:

```plaintext
> dnadist
dnadist: can't find input file "infile"
Please enter a new file name> EFla-Tu.phy

Nucleic acid sequence Distance Matrix program, version 3.66

Settings for this run:
D  Distance (F84, Kimura, Jukes-Cantor, LogDet)?  F84
G  Gamma distributed rates across sites?  No
T  Transition/transversion ratio?  2.0
C  One category of substitution rates?  Yes
W  Use weights for sites?  No
F  Use empirical base frequencies?  Yes
L  Form of distance matrix?  Square
M  Analyze multiple data sets?  No
I  Input sequences interleaved?  Yes
0  Terminal type (IBM PC, ANSI, none)?  (none)
1  Print out the data at start of run  No
2  Print indications of progress of run  Yes

Y to accept these or type the letter for one to change

g

Nucleic acid sequence Distance Matrix program, version 3.6a3

Settings for this run:
D  Distance (F84, Kimura, Jukes-Cantor, LogDet)?  F84
G  Gamma distributed rates across sites?  Yes
T  Transition/transversion ratio?  2.0
W  Use weights for sites?  No
F  Use empirical base frequencies?  Yes
L  Form of distance matrix?  Square
M  Analyze multiple data sets?  No
I  Input sequences interleaved?  Yes
0  Terminal type (IBM PC, ANSI, none)?  (none)
1  Print out the data at start of run  No
2  Print indications of progress of run  Yes

Y to accept these or type the letter for one to change

y

Coefficient of variation of substitution rate among sites (must be positive)
In gamma distribution parameters, this is 1/(square root of alpha)
1.0

Distances calculated for species
A.NIDULANS  ................
T.AQUATICU  .................
E.COLI  ....................
M.TUBERCUL  ..................
M.GALLISEP  ................
T.MARITIMA  ................
C.TRACHOMA  ..............
S.SOLFATAR  ..............
P.WOESEI  ................
H.MARISMOR  ..............
M.VANNIELI  ..............
T.ACIDOPHI  ..............
G.LAMBLIA  ..............
P.FALCIPAR  ..............
D.DISCOIDE  ..............
T.PYRIFORM  ..............
E.HISTOLYT  ..............
E.GRACILIS  ..............
T.AESTIVUM  ..............
A.THALIANA  .......

35
The distance matrix output file is named "outfile" by default. Immediately rename this file with the move command, "mv," so that the next program does not overwrite it!

> mv outfile F84.phydist

The distance matrix may be wider than your screen and not look so hot when displayed, but you’re welcome to take a look with the “more” command if you want.

4.b.2. Fit the best tree to the distance data: PHYLIP’s Fitch

Next we can pass the distance matrix to one of several distance tree inference programs in PHYLIP. I’ll start with Fitch, a least-squares fit algorithm. Least-square fit is a powerful way of estimating a tree from distance data but it is a bit computationally intense. Notice the time it takes to run in the following session. When asked for an input file you need to specify the input distance matrix. Note the options I choose below. One option that I will show, since it’s somewhat appropriate in this dataset, is the designation of an “outgroup;” specify “o” and designate the number of your outgroup sequence based on its order in the input file. I’ll again designate Giardia as the outgroup to add directionality to the tree.

You should always improve tree reliability by taking advantage of the “global” rearrangement and “jumble” (randomize) sequence order options by using the “g” and “j” switches respectively. The global rearrangement option adds the extra step of breaking up and recombining the tree found after all species have been added in an attempt to find an even better tree. This optimization procedure helps prevent you from getting caught on a local ‘tree-space’ optima and roughly triples the run time of each pass through the algorithm. Since the results of most of the tree construction algorithms are dependent on the order of sequence input, the jumble option should usually be taken advantage of also. It is a very good idea to jumble multiple times, at least ten (but only do one here, to save time). Supply jumble with any odd number as a random number seed. The Fitch screen trace follows:

> fitch
fitch: can't find input file "infile"
Please enter a new file name> F84.phydist

Fitch-Margoliash method version 3.66

Settings for this run:
D Method (F-M, Minimum Evolution)? Fitch-Margoliash
U Search for best tree? Yes
P Power? 2.00000
- Negative branch lengths allowed? No
O Outgroup root? No, use as outgroup species 1
L Lower-triangular data matrix? No
R Upper-triangular data matrix? No
S Subreplicates? No
G Global rearrangements? No
<table>
<thead>
<tr>
<th></th>
<th>Randomize input order of species?</th>
<th>No. Use input order</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Analyze multiple data sets?</td>
<td>No.</td>
</tr>
<tr>
<td>0</td>
<td>Terminal type (IBM PC, ANSI, none)?</td>
<td>(none)</td>
</tr>
<tr>
<td>1</td>
<td>Print out the data at start of run</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Print indications of progress of run</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Print out tree</td>
<td>Yes.</td>
</tr>
<tr>
<td>4</td>
<td>Write out trees onto tree file?</td>
<td>Yes.</td>
</tr>
</tbody>
</table>

Y to accept these or type the letter for one to change

Type number of the outgroup:

13

Fitch-Margoliash method version 3.66

---

**Settings for this run:**

- **D**: Method (F-M, Minimum Evolution)?
- **U**: Search for best tree? Yes
- **P**: Power? 2.00000
- **O**: Outgroup root? Yes, at species number 13
- **L**: Lower-triangular data matrix? No
- **R**: Upper-triangular data matrix? No
- **S**: Subreplicates? No
- **G**: Global rearrangements? Yes
- **J**: Randomize input order of species? No. Use input order
- **M**: Analyze multiple data sets? No
- **0**: Terminal type (IBM PC, ANSI, none)? (none)
- **1**: Print out the data at start of run No
- **2**: Print indications of progress of run Yes
- **3**: Print out tree Yes
- **4**: Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

---

Fitch-Margoliash method version 3.66

---

**Settings for this run:**

- **D**: Method (F-M, Minimum Evolution)?
- **U**: Search for best tree? Yes
- **P**: Power? 2.00000
- **O**: Outgroup root? Yes, at species number 13
- **L**: Lower-triangular data matrix? No
- **R**: Upper-triangular data matrix? No
- **S**: Subreplicates? No
- **G**: Global rearrangements? Yes
- **J**: Randomize input order of species? No. Use input order
- **M**: Analyze multiple data sets? No
- **0**: Terminal type (IBM PC, ANSI, none)? (none)
- **1**: Print out the data at start of run No
- **2**: Print indications of progress of run Yes
- **3**: Print out tree Yes
- **4**: Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

---

Random number seed (must be odd)?
5431

Number of times to jumble?
10 **but only do it one time here, in the interest of saving time!**

---

Fitch-Margoliash method version 3.66

---

**Settings for this run:**

- **D**: Method (F-M, Minimum Evolution)?
- **U**: Search for best tree? Yes
- **P**: Power? 2.00000
Negative branch lengths allowed? No

Outgroup root? Yes, at species number 13

Lower-triangular data matrix? No

Upper-triangular data matrix? No

Subreplicates? No

Global rearrangements? Yes

Randomize input order of species? Yes (seed = 5431, 10 times)

Analyze multiple data sets? No

Terminal type (IBM PC, ANSI, none)? (none)

Print out the data at start of run No

Print indications of progress of run Yes

Print out tree Yes

Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

Adding species:

1. D.MELANOGA
2. E.COLI
3. A.THALIANA
4. A.NIDULANS
5. P.FALCIPAR
6. T.PYRIFORM
7. S.CEREVISI
8. P.WOESEI
9. T.AQUATICU
10. E.GRACILIS
11. X.LAEVIS
12. T.AESTIVUM
13. T.MARITIMA
14. E.HISTOLYT
15. H.SAPIENS
16. C.ELEGANS
17. G.LAMBLIA
18. M.TUBERCUL
19. M.VANNIELI
20. T.ACIDOPHI
21. C.TRACHOMA
22. S.SOLFATAR
23. D.DISCOIDE
24. M.GALLISEP
25. H.MARISMO

Doing global rearrangements

Output written to file "outfile"

Tree also written onto file "outtree"

Done.

Fitch produces both a tree file called "outtree" and an output text file called "outfile." Again, immediately rename them. Use names that make sense and that identify the process used to create them; it's very important to keep track of what all the files are.

> mv outfile F84.fitch
> mv outtree F84.fitchtree

The output text file shows the least-squares fit formula used, an ASCII representation of the tree, and it lists all branch lengths. The text output from this Fitch run is shown below:

25 Populations
Fitch-Margoliash method version 3.66

\[
\text{Sum of squares} = \sum_{i}^{\text{Obs}} \sum_{j}^{\text{Exp}} \frac{(\text{Obs} - \text{Exp})^2}{\text{Obs}}
\]

Negative branch lengths not allowed

global optimization

```
+-------D.DISCOIDE
! \\
! +-------T.PYRIFORM
+-20 +23
! ! ! +-------S.CERESI
! ! !
! ! ! +-------E.GRACILIS
! +22 +17
! ! ! +--T.AESTIVUM
! ! ! +--5
! ! ! +--A.THALIANA
! +--18
+-2

! ! +-------C.ELEGANS
! ! +--4
! ! ! +--D.MELANOGA
! ! +--3
! ! ! +--H.SAPIENS
! ! +--12
! ! +--X.LAEVIS
! !
! ! +-------P.FALCIPAR
! +--14
! +-------E.HISTOLYT
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
Notice that the *Giardia* sequence is again being forced to lie to the outside of everything else by its designation as the outgroup, but it is sharing the base of the tree with several other protists. Surprisingly, they are not forming a discrete clade. As mentioned earlier, we believe this is actually the case, but yeast is still in their midst. *Euglena* also remains problematic, congregating every time with the green plants, and *marismortui* is again causing a problem by joining the Bacterial clade rather than the Archae (although it is separated from the Bacteria by a very long branch). The rest of the tree doesn’t look too bad. Take a look at the tree file to get a feel for the Newick format. The corresponding Fitch tree file follows:

```

((D.DISCOIDE:0.24269,((T.PYRIFORM:0.22541,S.CEREVISI:0.18229):0.01619,((E.GRACILIS:0.25780,(T.AESTIVUM:0.07999,A.THALIANA:0.07001):0.04360,((C.ELEGANS:0.19896,D.MELANOGA:0.15584):0.02539,((M.GALLISEP:0.37234,((T.MARITIMA:0.32220,((C.TRACHOMA:0.42169),(T.AQUATICU:0.29272,((H.MARISMOR:0.42164,((P.WOESEI:0.30722,((T.ACIDOPHI:0.43675,((T.ACI
```
Many PHYLIP programs generate tree file data for an evolutionary tree. Two PHYLIP drawing routines, DrawTree and DrawGram, can produce graphical tree representations from tree files. However, we will delay using them until we’ve run through all the PHYLIP tree inference programs that I want to introduce.

4.b.3. Bootstrapping neighbor-joining techniques.

Next I will show how to run a bootstrapped neighbor-joining analysis. Bootstrapping is a statistical method for ascertaining input data reliability. It randomly selects column subsets of your alignment data matrix up to the same overall size as the original to create as many test sets as specified. A consensus is then made of all the resultant analyses; those branches best resolved have the highest bootstrap value. PHYLIP displays these as the longest branch lengths in its tree representation. Normally bootstrapping is done with a minimum of 100 replicates. Because of this, the neighbor-joining distance based method is often used to estimate the best tree of the results because it is so much faster than all other methods even though it can also be less reliable. All PHYLIP methods accept the multiple dataset input option — they all have tradeoffs, one way or another. Use SeqBoot to bootstrap your original PHYLIP compatible input data set using the following command line and screen trace as a guide:

```bash
> seqboot
seqboot: can't find input file "infile"
Please enter a new file name> EF1a-Tu.phy

Bootstrapping algorithm, version 3.66

Settings for this run:
D  Sequence, Morph, Rest., Gene Freqs?  Molecular sequences
J  Bootstrap, Jackknife, Permute, Rewrite?  Bootstrap
B  Block size for block-bootstrapping?  1 (regular bootstrap)
R  How many replicates?  100
W  Read weights of characters?  No
C  Read categories of sites?  No
F  Write out data sets or just weights?  Data sets
I  Input sequences interleaved?  Yes
0  Terminal type (IBM PC, ANSI, none)?  (none)
1  Print out the data at start of run  No
2  Print indications of progress of run  Yes

Y to accept these or type the letter for one to change
y
Random number seed (must be odd)?
6521

completed replicate number  10
completed replicate number  20
completed replicate number  30
completed replicate number  40
completed replicate number  50
completed replicate number  60
completed replicate number  70
completed replicate number  80
completed replicate number  90
completed replicate number 100
```
Output written to file "outfile"
Done.

Rename the “outfile” before doing anything else!

> mv outfile EFla-Tu.seqboot

The output alignment data matrix now contains 100 randomly selected sequence data matrices. To generate 100 distance matrices, launch DNADist, being very careful to specify “multiple” datasets with the “m”, “d”, and “100” specifiers. I will again use the F84 model with a transition/transversion ratio of 2.0 and a rate heterogeneity gamma parameter $\alpha$ value of 1.0. Issue the following command line, to see the accompanying, much abridged screen trace (This will take a while to run!):

> dnadist
dnadist: can’t find input file "infile"
Please enter a new file name> EFla-Tu.seqboot

Nucleic acid sequence Distance Matrix program, version 3.66

Settings for this run:
D Distance (F84, Kimura, Jukes-Cantor, LogDet)? F84
G Gamma distributed rates across sites? No
T Transition/transversion ratio? 2.0
C One category of substitution rates? Yes
W Use weights for sites? No
F Use empirical base frequencies? Yes
L Form of distance matrix? Square
M Analyze multiple data sets? No
I Input sequences interleaved? Yes
0 Terminal type (IBM PC, ANSI, none)? (none)
1 Print out the data at start of run No
2 Print indications of progress of run Yes

Y to accept these or type the letter for one to change g

Nucleic acid sequence Distance Matrix program, version 3.66

Settings for this run:
D Distance (F84, Kimura, Jukes-Cantor, LogDet)? F84
G Gamma distributed rates across sites? Yes
T Transition/transversion ratio? 2.0
W Use weights for sites? No
F Use empirical base frequencies? Yes
L Form of distance matrix? Square
M Analyze multiple data sets? No
I Input sequences interleaved? Yes
0 Terminal type (IBM PC, ANSI, none)? (none)
1 Print out the data at start of run No
2 Print indications of progress of run Yes

Y to accept these or type the letter for one to change m

Multiple data sets or multiple weights? (type D or W) d

How many data sets?
100

Nucleic acid sequence Distance Matrix program, version 3.66

Settings for this run:
D Distance (F84, Kimura, Jukes-Cantor, LogDet)? F84
Gamma distributed rates across sites?  Yes
Transition/transversion ratio?  2.0
Use weights for sites?  No
Use empirical base frequencies?  Yes
Form of distance matrix?  Square
Analyze multiple data sets?  Yes, 100 data sets
Input sequences interleaved?  Yes
Terminal type (IBM PC, ANSI, none)?  (none)
Print out the data at start of run  No
Print indications of progress of run  Yes

Y to accept these or type the letter for one to change

Coefficient of variation of substitution rate among sites (must be positive)
In gamma distribution parameters, this is 1/(square root of alpha)
1.0

Data set # 1:
Distances calculated for species
A.NIDULANS ........................
T.AQUATICU .......................
E.COLI ........................
M.TUBERCUL  .................
M.GALLISEP  ....................
T.MARITIMA  ...................
C.TRACHOMA  ..................
S.SOLFATAR  .................
P.WOESEI  ....................
H.MARISMOR  .................
M.VANNIELI  .................
T.ACIDOPHI  .................
G.LAMBLIA  ................
P.FALcipar  .................
D.DISCOIDE  .................
T.PYRIFORM  .................
E.HISTOLYT  .................
E.GRACILIS  .................
T.AESTIVUM  .................
A.THALIANA  .................
S.CEREVISI  .................
C.ELEGANS  .................
D.MELANOGA  .................
X.LAEVIS  .............
H.SAPIENS  .................

Distances written to file "outfile"

Data set # 2:
Distances calculated for species
A.NIDULANS  ......................

Data set # 100:
Distances calculated for species
A.NIDULANS  ......................
T.AQUATICU  .......................
E.COLI  .........................
M.TUBERCUL  ........................
M.GALLISEP  ........................
T.MARITIMA  ........................
C.TRACHOMA  ........................
S.SOLFATAR  ........................
P.WOESEI  ........................
H.MARISMOR  ........................
M.VANNIELI  ........................
T.ACIDOPHI ............
G.LAMBLIA ............
P.FALCIPAR ............
D.DISCOIDE ............
T.PYRIFORM ............
E.HISTOLYT ............
E.GRACILIS ............
T.AESTIVUM ............
A.THALIANA ............
S.CEREVISI ............
C.ELEGANS ............
D.MELANOGA ............
X.LAEVIS ............
H.SAPIENS ............

Distances written to file "outfile"

Done.

Rename the "outfile:"

> mv outfile F84.bootdist

Now submit the 100 matrix distance file to Neighbor. Neighbor-joining algorithms aren't the greatest, as we saw earlier, but they are very fast, which we need here because we're dealing with 100 datasets. The program will produce a huge output text file and an output tree file, each containing 100 trees. Be sure to use the "multiple" option and specify "100" multiple datasets. Always randomize the input order with the "jumble" option and if it really matters make multiple runs with different seed numbers and compare results. Also specify your "outgroup" if you have one. Follow the drastically shortened screen trace below:

> neighbor
neighbor: can't find input file "infile"
Please enter a new file name>F84.bootdist

Neighbor-Joining/UPGMA method version 3.66

Settings for this run:
N   Neighbor-joining or UPGMA tree? Neighbor-joining
O   Outgroup root? No, use as outgroup species 1
L   Lower-triangular data matrix? No
R   Upper-triangular data matrix? No
S   Subreplicates? No
J   Randomize input order of species? No. Use input order
M   Analyze multiple data sets? No
1   Print out the data at start of run No
2   Print indications of progress of run Yes
3   Print out tree Yes
4   Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

Type number of the outgroup:
13

Neighbor-Joining/UPGMA method version 3.66

Settings for this run:
N   Neighbor-joining or UPGMA tree? Neighbor-joining
O   Outgroup root? Yes, at species number 13
L   Lower-triangular data matrix? No
R   Upper-triangular data matrix? No
S   Subreplicates? No
Randomize input order of species? No. Use input order

Analyze multiple data sets? No

Terminal type (IBM PC, ANSI, none)? (none)

Print out the data at start of run No

Print indications of progress of run Yes

Print out tree Yes

Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

Random number seed (must be odd)?

631

Neighbor-Joining/UPGMA method version 3.66

Settings for this run:

Neighbor-_joining or UPGMA tree? Neighbor

Outgroup root? Yes, at species number 13

Lower-triangular data matrix? No

Upper-triangular data matrix? No

Subreplicates? No

Randomize input order of species? Yes (random number seed = 631)

Analyze multiple data sets? No

Terminal type (IBM PC, ANSI, none)? (none)

Print out the data at start of run No

Print indications of progress of run Yes

Print out tree Yes

Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

How many data sets?

100

Random number seed (must be odd)?

753

Neighbor-Joining/UPGMA method version 3.66

Settings for this run:

Neighbor-joining or UPGMA tree? Neighbor-joining

Outgroup root? Yes, at species number 13

Lower-triangular data matrix? No

Upper-triangular data matrix? No

Subreplicates? No

Randomize input order of species? Yes (random number seed = 753)

Analyze multiple data sets? Yes, 100 sets

Terminal type (IBM PC, ANSI, none)? (none)

Print out the data at start of run No

Print indications of progress of run Yes

Print out tree Yes

Write out trees onto tree file? Yes

Y to accept these or type the letter for one to change

Data set # 100:

CYCLE 22: OTU 7 ( 0.44357)ジョINS OTU 1 ( 0.26673)
CYCLE 21: NODE 7 ( 0.01624)ジョINS OTU 5 ( 0.37186)
CYCLE 20: NODE 7 ( 0.03961)ジョINS OTU 3 ( 0.21084)
CYCLE 19: NODE 7 ( 0.06114)ジョINS OTU 2 ( 0.32846)
CYCLE 18: OTU 4 ( 0.53544)ジョINS NODE 7 ( 0.05356)
CYCLE 17: NODE 4 ( 0.13156)ジョINS OTU 6 ( 0.36471)
CYCLE 16: OTU 8 ( 0.53544)ジョINS OTU 11 ( 0.32416)
CYCLE 15: OTU 12 ( 0.41768)ジョINS OTU 10 ( 0.52902)
CYCLE 14: NODE 8 ( 0.12299)ジョINS NODE 12 ( 0.03541)
CYCLE 13: NODE 8 ( 0.02564)ジョINS OTU 9 ( 0.27351)
CYCLE 12: NODE 4 ( 0.92409) JOINS NODE 8 ( 0.03048)
CYCLE 11: OTU 14 ( 0.29867) JOINS OTU 17 ( 0.23853)
CYCLE 10: NODE 4 ( 0.22004) JOINS OTU 13 ( 0.46936)
CYCLE  9: OTU 20 ( 0.06109) JOINS OTU 19 ( 0.07201)
CYCLE  8: OTU 25 ( 0.11269) JOINS OTU 24 ( 0.06811)
CYCLE  7: NODE 14 ( 0.10203) JOINS OTU 15 ( 0.21402)
CYCLE  6: NODE 14 ( 0.01495) JOINS OTU 21 ( 0.17643)
CYCLE  5: OTU 16 ( 0.22688) JOINS NODE 14 ( 0.03052)
CYCLE  4: NODE 20 ( 0.09730) JOINS OTU 18 ( 0.27620)
CYCLE  3: OTU 22 ( 0.14438) JOINS NODE 22 ( 0.20939)
CYCLE  1: NODE 23 ( 0.02288) JOINS OTU 22 ( 0.09057)
LAST CYCLE:
    NODE 16 ( 0.07142) JOINS NODE 20 ( 0.03698) JOINS NODE 23 ( 0.00943)

Output written on file "outfile"
Tree written on file "outtree"
Done.

The output text file is huge. You may want to look at it, but then delete it with the "rm" command as it takes up an awful lot of space in your account. While deleting things it's a good idea to get rid of the SeqBoot'ed alignment and distance matrices also since they are also very large and they will not be needed any more in the tutorial. Be sure not to delete the tree file output from this last run of Neighbor; instead, rename it immediately:

> mv outtree F84.neighbor.boottree

Run the PHYLIP program Consense to condense all this information into a single consensus tree with branch lengths proportional to bootstrap confidence levels. Launch the following command line; specify the input tree file and designate the outgroup if you have one. Be careful with this designation. If your outgroup was not the last member of the dataset in your initial input file, as it was in my case, then it probably changed its order after the above neighbor run because you specified it as an outgroup there. Most likely it is now either the first or the last member of the dataset. Check the output tree file from the above step to be sure. Each tree is separated from the next by a semicolon, ";". Use the following abridged screen trace as a guide:

> consensus
consense: can't find input tree file "intree"
Please enter a new file name> F84.neighbor.boottree

Consensus tree program, version 3.66

Settings for this run:
C  Consensus type (MRe, strict, MR, ML):  Majority rule (extended)
O  Outgroup root:  No, use as outgroup species  1
R  Trees to be treated as Rooted:  No
T  Terminal type (IBM PC, ANSI, none):  (none)
  1   Print out the sets of species:  Yes
  2   Print indications of progress of run:  Yes
  3   Print out tree:  Yes
  4   Write out trees onto tree file:  Yes

Are these settings correct? (type Y or the letter for one to change)
  o
Type number of the outgroup:
    25

Consensus tree program, version 3.66

Settings for this run:
Consensus type (MRe, strict, MR, MI): Majority rule (extended)
Outgroup root: Yes, at species number 25
Trees to be treated as Rooted: No
Terminal type (IBM PC, ANSI, none): (none)
Print out the sets of species: Yes
Print indications of progress of run: Yes
Print out tree: Yes
Write out trees onto tree file: Yes

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

Consensus tree written to file "outtree"
Output written to file "outfile"

Done.

Rename both output files:

> mv outfile F84.neighbor.bootconsense
> mv treefile F84.neighbor.bootconsensetree

Check out the text file output from the Consense program. The numbers at the branch forks indicate the bootstrap value. The following abridged display show the results of this Consense run:

Consensus tree program, version 3.66
Species in order:
1. D.MELANOGA
2. C.ELEGANS
3. H.SAPIENS
4. X.LAEVIS
5. E.HISTOLYT
6. P.FALCIPAR
7. D.DISCOIDE
8. S.CEREVISI
9. T.PYRIFORM
10. A.THALIANA
11. T.AESTIVUM
12. E.GRACILIS
13. P.WOESEI
14. S.SOLFATAR
15. T.ACIDOPHI
16. M.VANNIELI
17. E.COIL
18. A.NIDULANS
19. M.GALLISEP
20. C.TRACHOMA
21. T.AQUATICU
22. M.TUBERCUL
23. T.MARITIMA
24. H.MARISMOR
25. G.LAMBLIA

Sets included in the consensus tree

Set (species in order) How many times out of 100.00

**************.* 100.00
******       100.00
............ 100.00
............ 100.00
............ 100.00
************ 89.00
************ 89.00
************ 89.00

---

47
Sets NOT included in consensus tree:

Set (species in order)     How many times out of 100.00

........... * **........ 81.00
........... **.*....... 79.00
.............. **.**.... 77.00
.............. **.**.... 75.00
.............. **.**.... 74.00
.............. **.**.... 73.00
**.............. ***.... 64.00
**.............. ***.... 64.00
**.............. ***.... 64.00
**.............. ***.... 63.00
**.............. ***.... 63.00
**.............. ***.... 62.00
**.............. ***.... 62.00
**.............. ***.... 61.00
**.............. ***.... 59.00
**.............. ***.... 58.00
**.............. ***.... 57.00
**.............. ***.... 56.00
**.............. ***.... 55.00
**.............. ***.... 54.00
**.............. ***.... 53.00
**.............. ***.... 52.00
**.............. ***.... 51.00
**.............. ***.... 50.00
**.............. ***.... 49.00
**.............. ***.... 48.00
**.............. ***.... 47.00
**.............. ***.... 46.00
**.............. ***.... 45.00

Extended majority rule consensus tree

CONSENSUS TREE:
the numbers on the branches indicate the number
of times the partition of the species into the two sets
which are separated by that branch occurred
among the trees, out of 100.00 trees

+-------P.FALCIPAR
      |       +-------E.HISTOLYT
      |       |       +-------D.DISCOIDE
      |       |       +-------S.CEREVISI
      |       |       +-------T.PYRIFORM
      |       +----------89.0-
      |                    +-------T.AESTIVUM
      |                    +----------81.0-
      |                    |       +-------A.THALIANA
      |                    |       +-------E.GRACILIS
      |                    +----------D.MELANOGA
      |                    |       +--------X.LAEVIS
      |                    |       +-------H.SAPIENS
      |                    |       +-------C.ELEGANS
      |                    +-------C.TRACHOMA
      |                     +------54.0-
      |                     +-------M.GALLISEP
      |                     +-------A.NIDULANS
      |                     +-------E.COLI
      |                     +--------52.0-
      +-------------------+-------M.TUBERCUL
                              +-------T.AQUATICU

+-------T.AQUATICU
      |       +-------M.TUBERCUL
      |       +-------T.AQUATICU

48
Notice some nodes are very well resolved with bootstrap values as high as 100%. This is a great indication of the reliability of the phylogenetic inference of those nodes based on the prepared dataset, the inference method, and the evolutionary model used. However, several others, in particular those that have had trouble with other techniques, reflect very low confidence values with numbers ranging around 50%. In general bootstrap values above about 60% tend to under-represent the actual confidence level of the branching pattern, whereas numbers below 40% over-represent confidence levels. A ‘rule-of-thumb’ that many people accept is bootstrap values above around 75% definitely indicate the cohesiveness of a particular clade; whereas values from around 60% to 75% get a bit ‘iffy;’ and values below about 60% argue against a particular clade’s distinctiveness. These less cohesive clades should probably be thought of as a polytomy degrading down to the next node with an acceptable bootstrap value.

The bootstrapped neighbor-joining Consense output tree file follows. Notice that the branch lengths are now representing bootstrap values, not evolutionary distances. We will plot this data later.

```
((((((P.FALCIPAR:100.0,E.HISTOLYT:100.0):100.0,D.DISCOIDE:100.0):74.0,
S.CEREVISI:100.0):75.0,T.PYRIFORM:100.0):89.0,((T.AESTIVUM:100.0,A.THALIANA:100
.0):100.0,E.GRACILIS:100.0):81.0,((D.MELANOGA:100.0,(X.LAEVIS:100.0,H.SAPIENS:100.0):100
.0):64.0,(S.SOLFATAR:100.0,M.VANNIELI:100.0):79.0):64.0,P.WOESEI:100.0):63.0):89.0,(((T.ACIDOPHI:100.0,(((C.TRACHOMA:100.0,(M.GALLISEP:100.0,
(A.NIDULANS:100.0,E.COLI:100.0):77.0):45.0):54.0,(M.TUBERCUL:100.0,T.AQUATICU:10
0.0):52.0):52.0,
T.MARITIMA:100.0):100.0,H.MARISMOR:100.0):46.0,((T.ACIDOPHI:100.0,(S.SOLFATAR:10
0.0,
M.VANNIELI:100.0):79.0):64.0,P.WOESEI:100.0):63.0):89.0):100.0,G.LAMBLIA:100.0));
```

You may notice that those nodes of low bootstrap confidence correspond to those that seem to bounce about depending on the inference method used. One possible solution is to exclude them from your analysis.

4.b.4. PHYLIP’s parsimony methods: DNAPars.

Finally I will show how to run one of PHYLIP’s parsimony programs, DNAPars. Parsimony analysis directly uses the aligned sequence data matrix as input. The program explores ‘tree space’ in an attempt to answer the question: “Which trees lead to the observed alignment with the least number of symbol changes?” DNAPars uses an approximate, heuristic, tree-rearrangement solution, although a much slower branch-and-bound, exact solution program, DNA Penny, is also available in PHYLIP. Begin the program with the command line below; again take advantage of the “jumble” option. In your run you may want to only use one jumble to save time, but in actual
research you should probably jumble a minimum of ten times. Also designate your “outgroup,” if you have one. Another option available is the reconstruction of ancestral node sequences; choose option 5 to designate, if this interests you. My interactive screen trace follows:

```bash
> dnapars
 dnapars: can't find input file "infile"
Please enter a new file name> EF1a-Tu.phy

DNA parsimony algorithm, version 3.66

Setting for this run:
U  Search for best tree?  Yes
S  Search option?  More thorough search
V  Number of trees to save?  10000
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
N  Use Transversion parsimony?  No, count all steps
W  Sites weighted?  No
M  Analyze multiple data sets?  No
I  Input sequences interleaved?  Yes
0  Terminal type (IBM PC, ANSI, none)?  (none)
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

Y to accept these or type the letter for one to change
j
Random number seed (must be odd)?
753
Number of times to jumble?
10  but only do it one time here, in the interest of saving time!

DNA parsimony algorithm, version 3.66

Setting for this run:
U  Search for best tree?  Yes
S  Search option?  More thorough search
V  Number of trees to save?  10000
J  Randomize input order of sequences?  Yes (seed = 753, 10 times)
O  Outgroup root?  No, use as outgroup species 1
T  Use Threshold parsimony?  No, use ordinary parsimony
N  Use Transversion parsimony?  No, count all steps
W  Sites weighted?  No
M  Analyze multiple data sets?  No
I  Input sequences interleaved?  Yes
0  Terminal type (IBM PC, ANSI, none)?  (none)
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

Y to accept these or type the letter for one to change

Type number of the outgroup:
13

DNA parsimony algorithm, version 3.66

Setting for this run:
U  Search for best tree?  Yes
S  Search option?  More thorough search
V  Number of trees to save?  10000
J  Randomize input order of sequences?  Yes (seed = 753, 10 times)
O  Outgroup root?  Yes, at sequence number 13
T  Use Threshold parsimony?  No, use ordinary parsimony
N  Use Transversion parsimony?  No, count all steps
W  Sites weighted?  No
M  Analyze multiple data sets?  No
I  Input sequences interleaved?  Yes
0  Terminal type (IBM PC, ANSI, none)?  (none)
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

Y to accept these or type the letter for one to change

y

Adding species:
D.MELANOGA
M.VANNIELI
H.MARISMOR
G.LAMBLIA
E.HISTOLYT
M.TUBERCUL
T.ACIDOPHI
M.GALLISEP
C.ELEGANS
S.SOLFATAR
T.AQUATICU
H.SAPIENS
C.TRACHOMA
S.CERESI
T.AESTIVUM
A.NIDULANS
A.THALIANA
T.PYRIFORM
P.FALCIPAR
P.WOESEI
E.GRACILIS
D.DISCOIDE
T.MARITIMA
E.COLI
X.LAEVIS

Doing global rearrangements
!-----------------------------------------------------!

Output written to file "outfile"

Tree also written onto file "outtree"

Done.

Don't forget to rename the output files:

  > mv outfile phylip.pars
  > mv outtree phylip.parstree

The output text file follows below:

DNA parsimony algorithm, version 3.66

One most parsimonious tree found:
requires a total of 6848.000

between and length

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>21</td>
<td>0.054933</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>0.049157</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>0.094111</td>
</tr>
<tr>
<td>23</td>
<td>H.SAPIENS</td>
<td>0.077313</td>
</tr>
<tr>
<td>23</td>
<td>X.LAEVIS</td>
<td>0.057148</td>
</tr>
<tr>
<td>22</td>
<td>D.MELANOGA</td>
<td>0.082210</td>
</tr>
<tr>
<td>21</td>
<td>C.ELEGANS</td>
<td>0.101902</td>
</tr>
<tr>
<td>12</td>
<td>G.LAMBLIA</td>
<td>0.206184</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>0.049000</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>0.074849</td>
</tr>
<tr>
<td>19</td>
<td>A.THALIANA</td>
<td>0.059758</td>
</tr>
<tr>
<td>19</td>
<td>T.AESTIVUM</td>
<td>0.057554</td>
</tr>
<tr>
<td>18</td>
<td>E.GRACILIS</td>
<td>0.134692</td>
</tr>
<tr>
<td>17</td>
<td>15</td>
<td>0.074410</td>
</tr>
<tr>
<td>15</td>
<td>T.PYRIFORM</td>
<td>0.103382</td>
</tr>
</tbody>
</table>
The output tree file is the simple nested Newick parenthetical form. We will use DrawGram to plot graphics of these trees later. Notice that only one 'shortest' trees was found with this parsimony run. As mentioned previously, this is not always the case as there can be more than one most parsimonious tree. The output will list all equally parsimonious trees, not an ordered ranking of the most likely trees. The drawing programs in the PHYLIP system will only use the first tree given in such a file. If you really want to draw graphics of all the trees in a PHYLIP multiple tree treefile, you would need to copy the file and create a series of tree files with editing so that you can produce output plots of each possible tree that was found. In all cases though, the PHYLIP program Consense can be run to generate the consensus tree from a tree file with multiple trees.

As described in the introduction, lineages that evolve much faster than others tend to confound many of the algorithms. This causes 'the long branch attraction effect' that I've already discussed and leads to the dreaded "Felsenstein Zone" where many parsimony algorithms are guaranteed to always find the wrong answer! I am always tempted to exclude sequences that appear to be causing this problem due to their extreme divergence but it is very hard to recognize. Other common problems in phylogenetic inference include sequencing and/or initial alignment mistakes, totally incorrect multiple substitution models and/or parameters, third (and sometimes first) position saturation in coding sequences, compositional biases, the occurrence of active lateral transfer of genes between the groups being examined, and the fact that some of the organisms may have improper traditional taxonomies leading to a discrepancy between your inferred phylogeny and the normally accepted one.

5) Maximum likelihood techniques
An algorithm that I am not going to demonstrate, but that I heartily recommend is maximum likelihood. I described PAUP*'s implementation of maximum likelihood earlier in the tutorial and will discuss the concept in general a bit further here. These algorithms are arguably the most powerful available for ascertaining phylogenies from sequence data; however, they are also computationally the most involved. They can take a prohibitive amount of time to run depending on the size of your dataset. Generally attributed to Felsenstein, for the use of phylogenetic reconstruction using sequence data, maximum likelihood is a probabilistic statistical procedure that combines the best features of distance and parsimony methods. Unfortunately it is very slow, although recent improvements such as those in PAUP* have sped up the process considerably. Datasets the size that we are dealing with could take several hours to evaluate, though, even with the tricks I described earlier, so I am not going to have you run it here, although I do provide the result of an ML run that I performed in the graphical tree section following the conclusion.

The PHYLIP maximum likelihood program is DNAML, but it is quite slow. Gary Olsen has recoded DNAML to produce fastDNAML (1994), considerably speeding up the algorithm. A protein implementation of maximum likelihood, ProtML is a part of the MOLPHY 2.2 package by J. Adachi and M. Hasegawa (1994) available by ftp from sunnh.ism.ac.jp. An even better protein maximum likelihood program is now in the PHYLIP 3.6 release. It is named ProML and it allows for site rate heterogeneity. Another relatively new implementation is the Tree-Puzzle (Strimmer and von Haeseler, 1996, and available at http://www.tree-puzzle.de/) maximum likelihood phylogenetic inference program. It works very quickly on either nucleic or amino acid sequences and performs very well, although some people argue that quartet puzzling is an inefficient heuristic searching technique. In approximation mode with uniform rate heterogeneity it is much, much faster than DNAML (perhaps 100-fold); allowing the software to estimate rate heterogeneity with the maximum number of categories allowed makes it only slightly faster than DNAML. A real advantage of using ProML, ProtML, or Tree-Puzzle is they can be used with protein datasets since they incorporate several ‘realistic’ models of protein evolution, modern variations of the famous Dayhoff PAM matrix. As mentioned before, in spite of PAUP*'s very good DNA maximum likelihood models, it does not work on protein sequences without considerable customization.

An advantage of Tree-Puzzle, in addition to the indisputable robustness of any maximum-likelihood approach, is it automatically assigns node support without the separate need to perform bootstrap analysis. Tree-Puzzle’s resultant node-support values can be thought of as, and substituted for, bootstrap values in most instances. Therefore, one program run gives you all three results, the most likely tree topology, the best-fit branch lengths, and node support confidence values. It accepts standard PHYLIP input conventions, though it converts all ambiguous and nonstandard sequence symbols, including “-” gaps and “?” missing info’ symbols, to “unknown” base “N” or amino acid “X” as the case may be. Dots “.” are treated as they are in PHYLIP, “the same as above,” so be sure not to have any left in your dataset from GCG’s usage where they represent gaps. One warning: the PHYLIP tree drawing programs, DrawTree and DrawGram, don’t know how to cope with the dual parameter, branch length and node support, values in the Tree-Puzzle output tree. Try TreeView (available at http://taxonomy.zoology.gla.ac.uk/rod/treeview.html) on your own desktop machine instead. It knows how to deal
with this situation and allows you to visualize and manipulate trees with their branches proportional to evolutionary
divergence and their node support values displayed concurrently.

Another somewhat quicker utilization of maximum likelihood statistics is to use it to test a user defined tree. Most
programs will accept a user defined tree for evaluation and branch length calculations by appending tree data to
the input data. See the documentation files for the particular details of achieving this in the different programs.
PHYLIP’s ‘nonclock’ parsimony programs require rooted trees while the ‘nonclock’ maximum likelihood and Fitch
programs require their user defined trees to be unrooted. PHYLIP’s ReTree program can be used to switch these
formats back and forth.

Refer to the documentation available online through the Web for assistance with all of these maximum likelihood
programs. If you have need to use any of these programs, contact me for further assistance.

6) Running PHYLIP in the background

Unlike GCG programs in SeqLab that run in the background automatically, PHYLIP requires manual background
submission. You will not need to run any PHYLIP jobs in this manner during today’s workshop, but you should
know how to do it for when the need arises. As mentioned in the introduction, I have prepared some simple script
files that you can modify in order to make the process easier. More complicated scripts and help documentation
are also available for passing the output of one program on to the input of another. These can all be found at
http://bio.fsu.edu/~stevet/scripts/. If the data set is small enough, you can run most PHYLIP programs
interactively. If not, you should use a script to run the programs in the background.

An example of a script that could be used for a Fitch run is given below. This file must contain all the same
parameters that the program would ask you, if you were running it interactively. It uses many of the same options
as given in the previous interactive sessions. The user needs to change the lines in the file to reflect their own
data filename and their chosen options. Pound signs, “#,” indicate comments, which are not a required part of the
parameter script file.

```
sample.phydist
o                       # designate outgroup option
6                       # number of outgroup based on order in input file
g                       # global rearrangement option turned on
j                       # jumble (randomize) order option turned on
8765                    # supply odd random number seed
10                      # how many jumbles?
y                       # Are these all the options you want? Yes.
```

To use a parameter script like this change the file name given on line one to that for your desired input file. Next,
options are indicated in the body of the file in the same order as they would be answered interactively. In this
example lines two through six specify options change similar to those seen the previous interactive sessions. If
you had wanted to just use the default parameters, then only the final “y” for “yes” is necessary, but you must
have at least the name of your input file and the “y” prompt for the program to run.
To run this sort of job in the background, launch the name of the PHYLIP program that you want to use followed by appropriate file redirection and an ampersand, “&.” All responses to the usual interactive menu need to be in your input script and you need to tell the program to use that script as input. The following command line illustrates the technique:

```bash
> fitch < fitch_parameter_script.file > terminal &
```

This process will immediately launch the job in the background and then you can go about whatever other computing tasks you would like to do. You can even log out of your current session and the background job will continue to run to completion. A handy UNIX command to check on the status of all of your currently running jobs is a variation of the process command that searches for your particular user name:

```bash
> ps -U your_account_name
```

The programs still produce their standard out file and tree file containing the usual program output. These can cause problems if you are running other PHYLIP jobs in the same directory since the file names will overwrite one-another. Therefore, be sure to devote separate directories to each concurrent PHYLIP job that you run. The corresponding terminal output file will contain the screen trace that you would normally see on screen and list potential problems if the job bombs. I do not want you submitting any background PHYLIP jobs in this workshop, but the method may be helpful for you in the future.

After a PHYLIP background job stops, look at the terminal file created by the run. If it stopped very quickly, check to see if it at least got into the program and that it was loading in sequences. If it wasn’t doing that, go back and fix your parameter script file until it does. If it didn’t work the first time, once everything has been corrected, restart the job. It is hard to have accurate ideas about how long a PHYLIP task might take. Some processes can run for days depending on the parameters selected, the size of the data set, and the user load on the computer. Experience will help. When the job finishes the output outfile and treefile will end up in the same directory where the job was started. Remember to rename them. Be sure to delete the terminal file after everything has gone well. They are only for troubleshooting and serve no other purpose. There’s no need to fill your account with unnecessary files.

7) Plotting phylogenetic trees with PHYLIP

Two PHYLIP programs plot PHYLIP generated Newick tree files — DrawTree and DrawGram. DrawTree produces an unrooted ‘network’ representation; DrawGram produces several varieties of rooted style ‘grams.’ In both cases, branch lengths are proportional to the evolutionary distance separating the entries, if they are provided, or the level of consensus, if it is a Consense tree. Both programs require that the user designate the input file and the font set. The PHYLIP system has six fonts to work with. These are the so-called Hershey fonts; they have the file names font1 through font6. Since the package is designed to be graphics device independent, it doesn’t use text characters in plots by default; rather the Hershey fonts give detailed drawing instructions for each character. These fonts must either be copied into the directory that the drawing programs are run in or you must specify the full path name to them each time that you specify one. This will vary with the system that you are
running PHYLIP on. For the purpose of this workshop, and to simplify the following procedures, I have placed the six font files in a GCG public data directory accessible with the Fetch command (this is not standard practice at all GCG sites!). Therefore, issue the following command line:

\[ \text{fetch genmoredata:font?} \]

to, copy the fonts to your current directory. Notice to use the question mark, “?” rather than an asterisk, “*,” in this command so that you only get the PHYLIP fonts and not all the GCG system font files that end in .bin.

If you need to change taxa (gene) names, you should do it before the final plotting stage. It’s just much easier that way. If you haven’t gotten the names in your data matrix the way that you want them to end up, then either change them with an editor in the Newick file or use PHYLIP’s ReTree program to do it interactively.

DrawTree and DrawGram operate in a similar fashion. They can produce two different visualizations — one a high resolution, publication quality PostScript file of the plot, and the other a bitmapped, graphics image that can be viewed on the monitor. Typically you’ll produce both outputs in an X environment. PHYLIP’s internal PostScript driver does a very good job of sizing the tree to the page to produce the final graphic; the on-screen graphic is handy for modifying the run parameters until you get just what you want. When either drawing program, DrawTree or DrawGram, is launched it looks for a file in the current directory entitled “intree,” if it doesn’t find one, it returns a request for the required input file and then it asks for a font file. A number of options are then listed governing the actual plot itself. There are a whole slew of these. To begin try the defaults, if they don’t give you what you want, change the parameters until you see what you prefer. One option worth exploring is “F,” the “Font” parameter. Because we’re outputting PostScript we can specify any PostScript supported font as long as the PostScript printer that you are using recognizes the font that you specify. PostScript font names are case sensitive. Most printers can generate a test page of all their supported fonts to help you choose.

Follow the sample screen trace below using the results from the first Fitch tree data set to get a feel for running DrawGram. Specify the name of the input tree and then the font file. To begin, I’ll use the default drawing parameters. Follow the screen trace:

```bash
> drawgram
drawgram: can't find input tree file "intree"
Please enter a new file name> F84.fitchtree
DRAWGRAM from PHYLIP version 3.6a3
Reading tree ...
Tree has been read.
Loading the font ....
drawgram: can't find font file "fontfile"
Please enter a new file name> font2
Font loaded.

Rooted tree plotting program version 3.6a3

Here are the settings:
0 Screen type (IBM PC, ANSI): (none)
P Final plotting device: Postscript printer
V Previewing device: X Windows display
H Tree grows: Horizontally
S Tree style: Phenogram
B Use branch lengths: Yes
```
After typing "y" and pressing <return> a new window will appear on your desktop containing a plot of the PHYLIP tree. That window will look something like the graphic on the following page:

Use the “File” button at the top of the window to either “Plot,” “Change Parameters,” or “Quit.” “Change Parameters” allows you to modify the tree’s appearance by changing the program’s run parameters. One that I’ll often change is option “T” “Stem-length/tree-depth” by setting it to zero; this suppresses the drawing of an implied root on the tree. After you’ve gotten the plot to look the way you want, use the “File” menu “Plot” choice to save a publication quality PostScript file of the plot to your account:

> Writing plot file ...
> Plot written to file "plotfile"
> Done.

As always, when running PHYLIP programs, immediately rename the output, this time called “plotfile.” Use an extension that identifies the content as a PostScript graphics file:

> mv plotfile F84.fitchtree.ps

This PostScript file can now be sent to any printer that interprets PostScript. This may involve transferring the file from the GCG server to a local computer. If you have access to the default GCG system printer, then you can use the lpr command to print the PostScript on that printer. Printing on any UNIX printer may require a special print queue; you may need some variation of the following lpr command (ask your system administrator):

> lpr -PPostScript_que PHYLIP_plot_filename.ps

A copy of the bootstrapped neighbor-joining tree as a phenogram, as well as a curvogram of the parsimony tree, and a Fitch eurogram, is attached to the end of this tutorial, following the conclusion.
Finally I will prepare a consensus tree from the three previous PHYLIP trees. The three separate tree files need to be merged into one to do this. You can either use the UNIX cat command or a text editor to do this. A sample cat command line follows. The second greater than symbol, “>,” is essential (the first one is just the system prompt); it redirects the output into the file combined.trees:

```bash
> cat Jin-Nei.fitchtree Jin-Nei.neighbor.bootconsenstree phylip-parsimony.tree > combined.trees
```

After combining the files, run Consense on the combined file. I’m not going to show a screen trace of this Consense run as I have already shown you one. Don’t forget to rename the output files afterward:

```bash
> mv outfile combined.consense
> mv treefile combined.consensetree
```

The final consensus tree will show how well the three methods — Fitch least-squares distance fit, bootstrapped distance neighbor-joining, and parsimony — agree with each other. Their branch lengths will be proportional to the level of agreement between the three methods. A handy thing to do with this type of tree is to impose actual branch lengths on the consensus topology. This can be done by taking advantage of PHYLIP’s user tree option in many of the programs. Merely append the desired Newick tree topology below the expected input data file for the desired program along with a number identifier telling how many user trees you are providing and specify that you want to use the tree that you are supplying once in the program. Details for doing this are available in the PHYLIP documentation.

As a grand finale, I will take the above consensus tree and load it into PHYLIP’s tree manipulation tool ReTree. This program enables one to change the names, root placement, appearance, branch lengths, and topology of trees. The appearance of a tree can be drastically changed without changing its topology by flipping and rerooting. Naturally, if you actually change the branch lengths or orders in your tree, then you are changing the topology and no longer accepting the conclusion of the inference software. I will use ReTree to change the appearance of the tree to more closely match our preconceived idea of what it should look like without changing the topology of the tree at all. A much abridged ReTree screen trace follows below; use the question mark key to see the help screen:

```bash
> retree
Tree Rearrangement, version 3.66
Settings for this run:
   U    Initial tree (arbitrary, user, specify)? User tree from tree file
   N    Use the Nexus format to write out trees? No
   0    Graphics type (IBM PC, VT52, ANSI)? ANSI
   W    Width of terminal screen, of plotting area? 80, 80
   L    Number of lines on screen? 24

Are these settings correct? (type Y or the letter for one to change)
Y
Reading tree file ...
```

retrree: can’t read intree
Please enter a new filename>combined.consensetree

```
*******1:D.MELANO
****33
```
First I will reroot the tree by specifying branch tip 13, the Giardia sequence, as the outgroup and check out the results. You can also use entire clades as outgroups by specifying an interior node. This is a very nice feature and not available within the other PHYLIP programs. After looking at the tree with Giardia as the outgroup, I decided to reroot it again using the node at the base of the two prokaryote superclasses as the outgroup.
Another nice feature is the ability to easily change the names at each tip. This is done through the “n” option. I used names right from the start that I wanted, although PHYLIP did truncate them to ten characters and ReTree will allow you more than that. Regardless, I am not going to bother changing them, but it can be very helpful.

Next, experiment with flipping, the “f” option, and transposing, the “t” option, nodes. Neither of these operations change the topology of the tree, yet they can make a drastic difference in the relationships implied by the tree. The psychological effect of this can be enormous — don’t overlook the potential. After getting your tree into exactly the form that you want, exit the program with the “x” option and write out the new tree. If you have good faith in your outgroup, go ahead and write the tree out in a rooted format. I’m going to write out mine as unrooted.

```
** 12 lines above screen **

**********5:E.GRACILIS

**********30

**********10:P.FALCIPAR

**********40

**********9:E.HISTOLYT

**********8:D.DISCOIDE

**********11:S.CEREVISI

**********12:T.PYRIFORM

**********13:G.LAMBLIA

**********17:P.WOESEI

**********41

**********14:T.ACIDOPHI

**********42

**********16:S.SOLFATAR

**********43

** 18 lines below screen **
```

NEXT? (Options: R . U W O T F B N H J K L C + ? X Q) (? for Help) x
Do you want to write out the tree to a file? (Y or N) y
Enter R if the tree is to be rooted
OR enter U if the tree is to be unrooted: u

Tree written to file
Be sure that you write out your new tree before quitting the program! The new file will have the name outtree. Rename it immediately:

> mv outtree combined.finaltree

This final tree can then have a PHYLIP PostScript graphic made of it with either DrawTree or DrawGram. When running DrawTree, I usually switch option “2,” “Angel of Labels,” in the settings menu to “a” for “Along” or “r” for “Radial;” there’s less chance of them overwriting one another that way. A PostScript DrawTree graphic of this final consensus tree is attached to the end of the tutorial. Do realize that this tree is still not in agreement with commonly held opinion of the relationships of these taxa — undoubtably due to the great amount of error introduced into the analyses by the saturation of the third codon positions in this dataset. If you used my doublet_mask, were your inferences any more satisfying? If this analysis really mattered, I would definitely repeat all procedures with third positions masked out to see the difference this would make, and, in fact, the ML tree following the PHYLIP trees below used my doublet_mask. The one GCG program that was able to automatically exclude third positions by option, Distances, did yield a more acceptable tree, even with just a neighbor-joining interpretation. I would also definitely use maximum likelihood methods, if I was planning on publishing the results.

8) Some concluding remarks

One vital point that can’t be repeated often enough, is the importance of your multiple sequence alignments. All subsequent analyses are absolutely dependent upon them. Another point that needs to be considered in phylogenetic inference is do not base an organism’s phylogeny on just one gene. There can be many complicating factors that make interpretation difficult. Weird phylogenies can be the result of several factors: bad alignments, insufficient data, abjectly incorrect models, saturated positions (homoplasy), compositional biases, and/or horizontal gene transfer. Use several genes — the Ribosomal Database Project (RDP) (http://rdp.cme.msu.edu/html/) provides a good, largely accepted, alignment and phylogenetic framework with which other phylogenies can be compared. The complete RDP can be installed on the local GCG server in aligned GCG format, given sufficient interest, which could then be used in the same manner as the sequences explored in this tutorial. Let me know if this interests you. Anytime the phylogenies of organisms based on two different genes do not agree, there is either some type of problem with the analysis or you have found a case of lateral transfer of genetic material. Paralogous gene phylogenies are another story altogether and should be based, if at all possible, on sequences all from the same organism. Please contact me at stevet@bio.fsu.edu for further information about using RDP and other tools for molecular phylogenetic inference not covered in this workshop.

Some analyses can take quite a while to run, especially if you use maximum likelihood methods, and I highly recommend using maximum likelihood as it is guaranteed to give you the most accurate answers possible, as long as you use appropriate models and rate parameters. However, I cannot stress the importance of the quality of your multiple sequence alignment enough — everything is dependent on it!
9) PHYLIP PostScript graphics of phylogenetic trees from the workshop's dataset

The F84 model, Fitch derived tree drawn by PHYLIP's DrawGram eurogram option. Vertical distance is meaningless in this representation; horizontal distance is directly proportional to evolutionary divergence in the number of substituted bases per site.
The PHYLIP bootstrapped neighbor-joining tree (distances by F84) phenogram. Here the vertical component of the line is again meaningless. The horizontal component of the line is now proportional to the amount of agreement between the 100 neighbor-joining trees found from the 100 bootstrapped datasets fed to the procedure, i.e. the bootstrap value.
The most parsimonious PHYLIP tree drawn as a curvogram. Again the vertical distance component is meaningless, but this time the horizontal component is also meaningless. Note the distinct differences between this tree and the two previous distance based methods’ trees.
The final consensus tree of the three above PHYLIP methods used in the tutorial drawn by DrawTree is present below. Now overall branch lengths indicate the amount of agreement between the three methods used. Three different relative branch lengths can be seen: one of three, where all three methods disagree; two of three, where there’s some agreement; and three of three, where all methods consistently gave the same answer. For instance, the Animalia and Bacteria clades are consistently separated from all others by long branches, while the protists have several short branches amongst them indicating that their branching order is pretty arbitrary.
TreeView (Page, 1996) graphic of PAUP* Maximum Likelihood, ModelTest (Posada and Crandall, 1998) optimized, phylogenetic tree from workshop’s dataset using my doublet_mask so that third codon positions are ignored. Note the differences between this and all previous trees.

General Time Reversible model: rates — A to C = 2.52, A to G = 2.05, A to T = 1.08, C to G = 1.34, C to T = 2.05, G to T = 1.00; base frequencies — A = 0.33, C = 0.19, G = 0.27, T = 0.21; rate heterogeneity — proportion of invariable sites = 0.18 and Gamma distribution shape parameter = 1.25. Natural log Likelihood = -11432.96. Vertical distance is again meaningless; horizontal distance is directly proportional to evolutionary divergence in the number of substituted bases per site as per the scale bar at the bottom of the graphic.
10) Bibliography

Key Suggested Readings


References


Hasegawa, M., Kishino, H. and Yano, T.A. (1985) Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA. *Journal of Molecular Evolution 22:* 160-74


**Software Cited**


Gilbert, D.G. (1990) ReadSeq, public domain software, Biology Department, Indiana University, Bloomington, IN, U.S.A. (see http://iubio.bio.indiana.edu/soft/molbio/readseq/)


I am indebted to Gary Olsen from the University of Illinois, Urbana Campus, whose lectures provided the gist and heart of this tutorial. Thank you Gary. I also wish to acknowledge Susan Jean Johns, my former colleague and supervisor at the Center for Visualization, Analysis and Design at Washington State University, now at the University of California, San Francisco — thank you for teaching me so very much over the years.

11) Appendix: The GCG Figure program and selected GCG graphics command switches

Figure files can be edited to change and enhance GCG graphics. A Figure input file contains one instruction per line. Each type of instruction has a special code. Those that you most likely will want to change or use as reference points are listed below. There are, of course, many others; refer to the GCG Program Manual for more information on the Figure program and its complete instruction set.

.d xx.xx yyy.yy draw to location. Draws a line to the location on the device given by the xx.xx yyy.yy platten coordinates.
.m xx.xx yyy.yy move to location. Move the pen to the location on the device given by the xx.xx yyy.yy platten coordinates.
.nc x new color. Changes the pen color or sets it to a new value.
.pt string plot text string. The command to plot the text string shown on the line

Another instruction that you won’t often find is that to designate the linestyle for drawn lines. This may be something that you might want to change to reflect a particular branch of a tree that you feel is very important or uncertain. It is also very helpful for differentiating between lines in multiple parameter plots without the use of color, e.g. black and white PostScript.

.ls x x.xx linestyle. Changes the nature of the line drawn. A linestyle line of .ls 1 0.10 would produce a solid line of width .1 platten units. A line with .ls 2 0.50 would produce a dotted line with the dots .5 platten units apart.

Instructions that may be useful to modify files for controlling text font, orientation, character size, and text angle:

/.fo x font to be used. Out of 22 available fonts listed in the GCG Program Manual.
/to x text orientation. Determines the relationship of the text to the location given in the previous move command. A value of 2 means that the text starts just to the right of the location.
/ch x.x character height. Sets the height of the characters used.
/td xx.x text angle. Set the angle at which the text is drawn. The default is horizontal.

Finally, to add entire figure legends you may want to experiment with the paragraph commands.

/wd x and .sp y These commands specify the width and spacing of your paragraph.

The following is a listing of some of the command switches that all GCG graphics routines will respond to. A brief description of what happens to the data is given with each command switch. The required part of the command is shown in UPPER-CASE type. “I” denotes an integer number and “x” a real one.
-COLor=I  The entire structure is drawn in the color of pen I for the graphics device regardless of the designations within the file being worked with.

-FIgure=yyyy.yyy  Writes the contents of the plot into a text file called yyyy.yyy that can be used as an input file for the program Figure. Figure files are editable and device independent. See the Program Manual for more information on this program.

-FOnt=I  Draws all the text characters in the plot with font I. GCG software will work with 22 different character sets. But, remember, the name labels within PHYLIP plots are not text characters unless you use PHYLIP’s PostScript driver and specify PostScript fonts.

-PLot=yyyy.yyy  Redirects the graphics output to file yyyy.yyy using whatever graphics language you have previously initialized.

-PORtrait  Rotates the plot 90 degrees.

-SCAle=x  Scales the size of the plot by the factor x. The factor can either expand or contract the size of the plot depending on the value of the number used, i.e. x=2.0 would twice as big, x=.5 would be half as small.

-XSCAle=x  Scales the size of the x axis by the factor x.

-YSCAle=x  Scales the size of the y axis by the factor x.

-XPAN=xx.xx  Moves the plot to the right along the x axis xx.xx platen units

-YPAN=xx.xx  Moves the plot to up along the y axis up xx.xx platen units

-SPEed=x  Controls the speed of the pen, the default is 10.0 and is quite fast. A value of 1.0 or 2.0 slows down the plotting process and produces better lines on a HP pen plotter for a higher quality plot. This only applies to HPGL pen plotters.

-DENsity=x  Not available for all GCG graphics programs, however, where available, this function controls the number of pages on which the graphic will be drawn. By default graphics are calculated to fit one page. If you want to superimpose the graphics from sequences of different length, this parameter allows the scale to remain identical. Another use is to zoom in on a portion of a graphic without losing the rest of the information — it will spread out onto more pages.