A Very Basic Introduction to Molecular Evolutionary Phylogenetic Inference Software

Lab Eight, Wednesday, February 25, 2009

Author and Instructor: Steven M. Thompson

How to use GCG’s implementation of PAUP* (Phylogenetic Analysis Using Parsimony [and Other Methods], standalone PHYLIP (PHYLogeny Inference Package), and other tools to ascertain and draw phylogenetic trees from multiple sequence alignment datasets. Emphasis is placed on the reliability, congruence, and accuracy of model-based approaches, especially using probabilistic methods, although time constraints will restrict the analyses performed to less computationally intense methods.
Now that you have a great sequence alignment — what next?

Introduction

(Much of Introduction condensed from Gary Olsen’s lecture: *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 text *Evolution* by Dobzhansky, Ayala, Stebbins and Valentine (1977) reads “Nothing in biology makes sense except in the light of evolution.” (quoted from Dobzhansky in 1973). 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 do 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 with Fisher, Haldane, and Wright); however, there was a
dbeeldphasis 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.*

Organisms have a History

![Tree Diagram]

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: inter- versus intra-species.
But, what about molecules? Enter the molecule.

Hemoglobin was perhaps the first molecule used in deciphering phylogeny (Zuckerkandl and Pauling, 1965).

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 following scenario:

```
human α globin
mouse α globin
frog α globin
human β globin
mouse β globin
frog β globin
```

If we did not know the difference between the α and β globins, then we might be very upset by a (historically correct) tree based on fewer sequences:

```
human α globin

frog α globin

mouse β globin
```

In this example, the α globins are orthologous with one another, and the β globins are orthologous with one another. However, the α globins are paralogous to the β globins (they have evolved in parallel, at different locations in the genome, the result of a very 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 concept of Evolutionary Distance: 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 \( X_{01} + X_{02} \), i.e. the sum total of all the evolutionary events since the divergence from 0 along the complete path between 1 and 2.

An interesting mathematical property of additive trees is that it is only necessary to know the pairwise distances between the sequences at 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 this 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 → G → C → 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 →
G → 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 the simple graph shown below. 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:

![Actual versus Observed Sequence Changes](image)

This situation necessitates two things:

1) We must estimate the number of actual substitution events from the number of observed differences, 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 how to define 'best fit.'
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 (easily accomplished by masking in SeqLab) those most highly diverged sequence regions from your analyses. It is in these areas of high ‘homoplasy’ that all assumptions are most violated, and this is also where your alignment is least sure, the primary assumption.
Two often used Distance methods:

Neighbor-Joining

The method is based on clustering similar sequences, but with a correction for the 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 arithmetic method that can only result with one single tree. 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 (Fitch and Margoliash, 1967). The method is somewhat analogous to finding the best line that fits a scattered point dataset.

In all distance methods use only one, most closely related outgroup, if any, in order to decrease the amount of implicit, systematic error in its assumptions.

Parsimony

Historically, the idea of parsimony arose in the analysis of complex morphological features for which it is intuitively appealing to assume that these features were invented only once. In that case, all species in which a given 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 the “cladistic” school of phylogenetics.

The method of inferring the "most parsimonious" (minimum change) tree is purely computational. 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 is referred to as 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’ after its original description (Felsenstein, 1978), guarantees that standard parsimony will always 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 chance that a nucleotide will undergo the same change in each of these two lines. If this happens, then B and D can share the original nucleotide, and A and C will (due to parallel, independent events) share a different nucleotide.

This problem can be minimized by (1) avoiding very diverged sequences, which will have long 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 two shorter branches (only do this in parsimony or maximum likelihood analyses; 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 are incredibly complicated to me; 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/ for MrBayes (Ronquist and Huelsenbeck, 2003), one of these powerful Bayesian statistical methods for phylogenetic inference.

Implementations for allExplicit Optimality Criteria Methods

Exact Methods — Exhaustive versus Branch-and-Bound

Exhaustive

Add every sequence, stepwise, at every possible position. This requires an incredible amount of computer time and is intractable for most datasets, only being 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:

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

In a much easier to fathom manner, a simple table enumerates the increase:

<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\times10^{20})</td>
</tr>
<tr>
<td>50</td>
<td>(&gt;3\times10^{74}) ((&gt;\Sigma) of all the atoms in the universe!)</td>
</tr>
<tr>
<td>10 million</td>
<td>(&gt;5\times10^{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. This eliminates much of the search time, yet guarantees an optimal tree. It is a much faster exact method than exhaustive. However, it is only practical for up to about twenty sequences, thirty at the most, even on very fast, modern computers.

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: 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 illustrate. 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, though now quite dated, 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, 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](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. Either the SeqLab or the SeaView editor can be a big help with these matters. Lab Seven discussed the methods for achieving this in these editors. 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.

As discussed last week, one of the biggest problems in this field is 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. SeaView also has the ability to read and write several different formats, including NEXUS.

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 quite complicated. 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 any phylogenetic inference takes to complete 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, but see parallel implementations like mpiRAxML; others only slow down when a large
number of sequences are being worked with that are relatively long. The individual program documentation is
good about describing how slow each program is. Peruse each program’s documentation prior to running it
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 phylogenetic inference programs 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! PAUP*, RAxML, Garli, MrBayes, and PHYLIP batch mode documentation and scripts are available on the HPC to assist you in operating this way. Review this information prior to running any tree search 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 easy to run simple PAUP* jobs in the background through SeqLab, 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.

**Standard disclaimer**

I write these tutorials from a ‘lowest-common-denominator’ biologist’s perspective. That is, I only assume that you have fundamental molecular biology knowledge, but are relatively inexperienced regarding computers. As a consequence of this they are written quite explicitly. Therefore, if you do exactly what is written, it will work. However, this requires two things: 1) you must read very carefully and not skim over vital steps, and 2) you mustn’t take offense if you already know what I’m discussing. I’m not insulting your intelligence.

I use three writing conventions in the tutorials, besides my casual style. I use **bold** type for those commands and keystrokes that you are to type in at your keyboard or for buttons or menus that you are to click in a GUI. I also use bold type for **section headings**. Screen traces are shown in a ‘typewriter’ style Courier font and “///////////” indicates abridged data. The dollar sign ($) indicates the system prompt and should not be typed as a part of commands. Really important statements may be underlined.

As you’ve learned, specialized X-server graphics communications software is required to use GCG’s SeqLab. I’ll remind you of a few things about X: X Windows are only active when the mouse cursor is in that window, and always close X Windows when you are through with them to conserve system memory. Furthermore, to
activate X items, just <click> on them, rather than holding your mouse button down. Also, X buttons are
turned on when they are pushed in and shaded. Finally, don’t close X Windows with the X-server software’s
close icon in the upper right- or left-hand window corner, rather, always, if available, use the window’s own
“File” menu “Exit” choice, or “Close,” or “Cancel,” or “OK” button.

A ‘Real-Life’ Project Oriented Approach. Molecular Phylogenetic Inference

Activate and log on to the computing workstation you are sitting at and then log onto HPC with an X-tunneled
ssh session. If using a terminal window on Mac OSX or UNIX/Linux, then issue the following command (the X
has to be capitalized and replace “user” with your account name):

$ ssh -X user@submit.hpc.fsu.edu

Preliminary preparations

Change your directory from ‘home’ to last week’s subdirectory. List that directory and check out the files left
over from last week’s tutorial. Look through them and remove any that you don’t want to save. Next, change
directory back to your home directory, create a subdirectory for this week’s tutorial data, and then change
directory into it. After you’ve taken care of file maintenance launch SeqLab with the standard command:

$ seqlab &

Next, it will again be helpful to change your SeqLab working directory to your present location so that
everything you do today will automatically be saved in your new directory rather than last week’s directory.
Do this as before with SeqLab’s “Options” “Preferences...” “Working Dir...” button.

Now verify that you are in SeqLab’s “Main List” “Mode:” and start a new list to contain this week’s data.
Therefore, select ‘New List...’ from the “File” menu and give your new list an appropriate name. It’s not
essential to use the file name extension “.list” but it’s a good idea. Check “OK.”

You should now be in List Mode with an empty window. Go to the “File” menu and select “Add Sequences
From” “Sequence Files...” Use the “Directories” column to move from your present directory over to Lab
Seven’s subdirectory and then replace the text in the “Filter” text box with the name or a wildcard
specification that will identify your final protein alignment RSF file from that tutorial. Press the “Filter” button
and then select the correct entry. Press the “Add” button to add it into your new empty list file. Repeat this
procedure with your aligned DNA FastA format dataset created from within SeaView. “Close” the “Add
Sequences” window afterwards. It is not necessary, but really helps in case of a computer crash to “Save
List” off the “File” menu. That way you won’t need to add your dataset to your Main List again, if systems fail.
Select your protein dataset (only) and switch “Mode:” to “Editor.” “Cut” Lab Seven’s consensus sequence
and any text annotation lines that you may have created from the alignment to simplify matters. You may
even elect to cut sequences from the dataset, probably the most divergent ones, but perhaps based on
taxonomy, to reduce the overall size of the dataset down to between twenty and thirty members, so that you
won’t have to wait so long for some of the programs to run. I cut the plants and Archaea out of my alignment, and left 1EFT to use as an outgroup, but still ended up with 64 sequences. If you do get rid of any sequences, afterwards make sure you select all of your sequences and then use the “Edit” menu “Remove Gaps...” “Columns of gaps” function to ‘repack’ your dataset.

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 less reliable columns in your alignment where you are unsure of positional homology, without actually getting rid of the data. At this point be sure all of your protein sequences are selected and create a Mask style sequence consensus of them by going to the “Edit” “Consensus...” menu and specifying “Consensus type:” “Mask Sequence.” The default protein mode is to create a consensus at the 2/3’rds plurality level (“Percent required for majority”) with a threshold of 5 (“Minimum score that represents a match”) and an identity scoring matrix; however, this is an incredibly high value for phylogenetic analysis, and would likely leave very little phylogenetically informative data. Therefore, experiment with lower plurality and threshold values as well as different scoring comparison matrices to see the difference that it can make in the appearance of your 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. The following screen illustrates my example using the “BLOSUM30” matrix, a “Percent required for majority” (plurality) of “15%,” and a “Minimum score that represents a match” (threshold) cutoff value of “4.” Much of the length of the alignment is shown below with a two to one zoom factor:

Few areas are excluded by the Mask in this alignment because of the high similarity of this group of sequences. This is as it should be, for excluding many more columns in this particular alignment would leave
nearly identical sequences making it difficult to ascertain how they are related. As discussed in the Lab Seven Introduction last week, when dealing with very similar sequences, it is usually best to align DNA sequences along with their corresponding proteins. This also assures that gaps are not placed in the middle of codons, yet retains the sensitivity of amino acid alignment. SeqLab and SeaView require DNA datasets to do this. We’ve got both — protein from SeqLab, and DNA from SeaView. Several Perl scripts, other programs, and the rationale for dealing with this situation, were discussed further in last week’s tutorial.

Phylogenetic analyses can be performed on DNA rather than on protein, when this approach is used. This is especially important when dealing with datasets that are extremely 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 have a longer history 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 become, which will introduce noise (error) into the analysis. Therefore, third codon positions are sometimes excluded. As with so much in bioinformatics, the factors regarding protein versus DNA phylogenetic inference are quite complicated, often balancing available signal against confounding noise.

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.

Phylogenetic tree estimation — GCG style
Distance methods: GCG’s Distances and GrowTree

To begin, be sure that all of your protein sequences, as well as your Mask, are selected. We’ll use GCG’s linked distance matrix and tree drawing combination method for phylogenetic inference first. The evolutionary distance between each sequence is first calculated by the program Distances, compensating for multiple substitutions at homologous sites with a specified model, and then those distances are used in a neighbor-joining algorithm by GrowTree to estimate a phylogenetic tree for the data. You’ll use SeqLab’s “Functions” “Evolution” “Distances…” menu to begin the programs. However, before doing that use a Web browser to read the complete description of this molecular evolution program in the GCG GenManual documentation (http://www.scs.fsu.edu/gcg/distances.html) — it describes problems and considerations in this whole area very well. Pay particular attention to the “ALGORITHM,” “CONSIDERATIONS,” and “SUGGESTIONS” sections of the manual. Note that all of the GCG Distance protein models included only count identities, not similarities!

Once the “Distances” program window is displayed and you’ve read over the manual, accept the “Kimura” default “Distance Correction Method” and then punch the “GrowTree…” button. Be sure “Neighbor joining” is checked and then punch 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. 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 Kimura protein model:

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 1EFT from Thermus aquaticus. (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 what we think we understand of the universal ‘tree of life,’ largely from ribosomal RNA phylogenies, the protists in this dataset should not all be within the same clade, they are polyphyletic, but protists are often problematic. The animals correctly share a common clade, and the fungi are properly placed as the sister clade to the animals. However, one frog sequence, EF11_Xenla, somehow ends up mixed in with the protists. The Kimura model was too great of an oversimplification, as it often is.

To see the difference various evolutionary models and options can make go back to the “Distances” program and change the model. Remember that 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 an even simpler model, “Jukes-Cantor distances.” Press “Run” in the main “Distances” program window. My Jukes-Cantor tree follows below on the next page. Notice the differences between this tree and the former Kimura tree:

In particular, check out the extremely long branch on the 1EFT lineage in my example. Otherwise it is quite similar to the Kimura tree, with the same mistakes and truths. Parameters and models can make a huge difference, particularly in a simple method like neighbor-joining.

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 neighbor-joining trees. If your sequences are too divergent and you are using the Jukes-Cantor DNA correction 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 causing interpretive and reliability problems, when many sequence pairs have greater than 100 substitutions per 100 residues. 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. You’ll sometimes see the word “homoplasy” used to describe these “saturation” situations where so many multiple substitutions have occurred in an area of sequence that it is impossible to ascertain relationships based on that area. As explained in the introduction, this phenomenon increasingly confounds evolutionary reconstruction as divergence between the members of a dataset increases.

GCG’s Distance/GrowTree programs are not a very robust method, providing very limited model flexibility and only utilizing neighbor-joining, 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 today.
GCG’s interface to PAUP* — PAUPSearch and PAUPDisplay

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

```
$ typedata paup-license.txt
```

PAUP was originally written to only perform parsimony analysis with either DNA sequences or morphological character data, and it only worked on personal Apple Macintosh computers. The name meant “Phylogenetic Analysis Under Parsimony.” Its latest incarnation, version 4+, changed the package’s name by adding the asterisk, which means “and other methods,” referring to the addition of distance and the maximum likelihood methods 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 “Classic” Macintosh GUI. PAUP* is seldom employed to analyze protein datasets though, as the current production release does not incorporate any protein models of evolution other than a crude like/not-like model. More sophisticated protein models can be used by embedding the necessary commands and matrices in the NEXUS file used as input to the package, and prerelease versions have it all built in. Furthermore, as I discussed previously, many people prefer to perform evolutionary inference with DNA sequences regardless. PAUP*’s DNA models are among the most sophisticated available in any molecular phylogenetic inference software, and I, therefore, heartily recommend using it for DNA datasets. Here, due to a lack of time, we’ll be using your protein alignment during the lab session, and your DNA alignment will be a part of your homework.

The PAUPSearch and PAUPDisplay program pair is GCG’s interface to the PAUP* package. These programs can provide an 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’ll 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 production 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.

Using PAUPSearch and PAUPDisplay for phylogenetic inference

I encourage you to learn how to run the most robust PAUP* searches possible, before accepting any of its output as valid. However, here, for the sake of time, 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 tutorial. If you are more interested in this package, I would recommend first carefully reading all of GCG’s documentation on their PAUP* interfaces, and then, for all of the nitty-gritty details, a complete PAUP*
command reference manual is available (see http://paup.scs.fsu.edu/Cmd_ref_v2.pdf). Finally, if you’re really interested in this stuff, I heartily recommend applying for the annual Workshop on Molecular Evolution (http://workshop.molecularevolution.org/) at the Woods Hole Marine Biological Laboratory.

Begin the process by again being sure that all of your protein sequences and your weight Mask are selected, and then go to the “Functions” “Evolution” menu. Select “PaupSearch...” to launch the dialogue box.

Accept the default “Tree Optimality Criterion” “maximum parsimony” setting 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, leaving them at their default settings, until you get to the section 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.” Scroll through the rest of the options but do not use any of the others at this point; “Close” the “Options” menu.

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 if you want, though it will not change the tree topology, only the way it looks. I’ll set it to 64 to force the *Thermus aquaticus* 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. In my example’s case I know that the Bacteria *Thermus aquaticus* is definitely not a part of the ingroup of eukaryotic sequences. Often this external data relates to fossil evidence. Another trick that is occasionally done when no appropriate outgroup exists (other than a rock) is to use a paralogue to the system in study (Iwabe, et al. 1989), if the gene duplication event that produced the paralogue 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. 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.

After a few minutes, the results will be displayed. The top-most file will be the text output file from PAUPDisplay that describes the tree along with several statistical measures. Look it over, and then “Close” the window. The next window will be the “SeqLab Output Manager,” “Close” it. The next window will be the
phylogram graphic drawn from the PAUPDisplay output figure file; check it out and then “Close” it. As is often the case with parsimony many best trees may be discovered. Parsimony often finds more than one shortest tree. I found more than one hundred! The graphics window allows you to “Page” through them. A screenshot graphic of the first of my 100 PAUPSearch/PAUPDisplay parsimony trees is shown below:

Notice the differences between this tree and what we've previously seen, and this is only one of 100 most parsimonious saved. The most obvious difference is the *Thermus aquaticus* outgroup placed there by option. The protists are not all in a discrete clade anymore; that’s good. And the animals and fungi are in separate, discrete sister clades, also good. This analysis clearly has some other problems though, e.g. the red algae *Porphyra* sharing a clade with the Stramenopile Protist *Blastocystis* and the Mycetozoa ‘slime mold’ *Dictyostelium*, perhaps caused by ‘long branch attraction’ — it’s hard to say without further testing — but in some ways it’s more satisfying than what we’ve seen previously.

Relaunch the “Output Manager” window from SeqLab’s “Windows” menu. The PAUPtrees file contains the NEXUS formatted alignment as well as the Newick format trees discovered by the search. “Display” the PAUPtrees file and then “Close” it and the “Output Manager.”

Let’s see a PAUP* minimum evolution distance method next. Assure that all the protein sequences and your Mask are still selected and then relaunch “PAUPsearch. . .”. This time change the default “Tree Optimality Criterion” from “maximum parsimony” to “distance (minimum evolution)” but again 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. Notice that very few models
are listed for protein systems, only “mean” or “total” distances. This is what I warned you about earlier. I’ll run it with the “mean distance” model. Notice the “How to Treat Negative Branch Lengths” panel below the model choice panel. It may help to turn on “reset branch lengths to zero.” “Close” the “Options” menu.

Next press the “PaupDisplay . . .” button to get its corresponding menu. Change “describe and plot tree(s)” to “plot tree(s) only,” since we already saw how PAUPDisplay describes trees above. An “Optimality Criterion” choice will no longer be listed. PAUPDisplay’s “Options . . .” should still specify the outgroup and “phylogram” representation that you previously chose. “Close” the “PAUPDisplay” menus and then press “Run” on the main “PAUPSearch” menu. My example follows below:

The tree from this run is different still from all the previous ones. At least all the animals are still sharing a clade and the fungi are forming a sister clade to that. But there are still some problems. Clearly we have again violated some systematic assumptions about the data in this analysis. Realize that the mean distance model we used (or the total distance model) is very crude and only counts amino acid identities, not similarities, and does not compensate for superimposed substitutions at all. But that’s all you can do with the PAUPSearch/PAUPDisplay interfaces through GCG on protein datasets. You either need to run PAUP* native, or install the program on your own computer to take advantage of more sophisticated protein models and associated options. It really pays in this field to do your homework and learn about the various models and the importance of their parameters.

**NEXUS format**

I have only briefly shown you PAUP* and encourage you to explore the package at length with your own data. But, before moving on to PHYLIP, let’s finish up this section on GCG’s implementation of PAUP* by looking at how SeqLab can be used to merely generate a NEXUS format file. If you end up running PAUP* exterior to GCG, PAUPSearch can be used as a very handy and reliable tool for directly generating NEXUS format files. Plus, remember from last week that SeaView is another way to easily create NEXUS format.
Begin the NEXUS conversion process by again being sure that all the protein sequences and your weight Mask are selected in your “Main Window” Editor display, and then return to the “Functions” “Evolution” menu (or relaunch “PaupSearch” through the “Windows” menu). Select “PaupSearch. . .” to launch the dialogue box. Because you merely want to generate a NEXUS file, you will run PAUPSearch in its fastest mode here without actually performing a search. The “Tree Optimality Criterion” and “Method for Obtaining Best Tree(s)” doesn’t matter — you can leave them at whatever setting they are. Again 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 check in the top box to save the PAUPscript file. This is not required for running the programs but since here we are just generating NEXUS format, it is essential. You can change or leave the file name as you wish. The PAUPscript output file results from the automatic conversion of your alignment to NEXUS format and contains embedded PAUP commands as well as your alignment. (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 ‘when it really matters' to avoid otherwise frustrating troubleshooting.) Uncheck the next box, “Perform the analysis.” This makes the program do the conversion to generate the NEXUS script but prevents it from performing the heuristic search for the best tree (equivalent to the command line option –NoRun). “Close” the “Options” menu. Normally PAUPSearch and PAUPDisplay are linked to each other when you run them from the SeqLab interface, just like when you ran Distances and GrowTree earlier. Therefore, since we don’t want to run PAUPDisplay, uncheck the “PaupDisplay. . .” button in “PaupSearch”’s main window. Be sure that “How:” “Background Job” is specified on the main “PAUPSearch” menu and then press “Run” there. After a brief moment the output PAUPscript file will be displayed.

The PAUPscript file is critical. It is the NEXUS format file generated by GCG to run PAUP*. Notice that columns of your alignment with zeroes in their Mask are excluded from the NEXUS alignment. This file can be used to run PAUP* on a different machine, if desired, by transferring the file. Using a Macintosh may be desirable in order to take advantage of PAUP**'s very friendly Classic Macintosh GUI. Since GCG’s PAUPSearch automatically creates this file for you, correctly encoding all of the data in the required format, 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 will want 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.

Phylogenetic Inference using the PHYLIP package

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. Since we want to take advantage of the Mask data for subsequent phylogenetic analyses, we’ll use another method to export our alignment. Therefore, after being sure that all of the protein sequences as well as the Mask sequence are selected, 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 non-GCG format 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:

```
>EF1A_TETPY Elongation factor 1-alpha (EF-1-alpha) (14-nm filament-associated protein).
MARGKVHNLVGHVDGSGKTSTGHLIYKCGIDKRVIEKFEKESAEGQKSFKYAWVL
DKLKAERGIDISLWKFETAYHTPIDAPGHRDFIKNMTGTSQADVAILMAISPQ
GEEFAISKQTRHALAFFALTGQMVNLKNMDEKTNFSEERYEIKKELSDYLLK
VQYKPDTPFPIFISGFDNMHSTNAPIWYK-----------------------------GQILVEALDALEPPKR
PVDKPLRPLQDVYKIGGITVPVFVGETVGVKPSNQIPAPKIAECKSSEMLHEQLP
EA-VPGDVFNIKVGVSODIRGNAVDASKMDPKEAEATFSQVIINHPQIQAGYTP
VLDCHTAHIAACKFETIKDIDRRTGKSQEENPKFIRKDQAALTVLITPALKCVVPQYEYP
PLGRYAARMDKQTVAVGVIKKE----------------------------------KDK
>EF1A_ENTHI Elongation factor 1-alpha (EF-1-alpha).
MPKEKTHTINLVGHVDGSGKTSTGHLIYKCGIDQRTIEKFEKESAEMGQSKFAMYWVL
DNLKAERGIDISLWKFETAYHTPIDAPGHRDFIKNMTGTSQADVAILVAAGT
GEEFAISKQTRHEILLSTLTVGKDQIVGVNMD-AIQYKQERYEIKKEISAPLKK
TGYNPDKIPFPIFISGFDNMIPSTNMPWYK-----------------------------GPTLIGALDSYTPPER
PVDKPLRPLQDVYKISGQTTVPVFVGETVGVKPSNQIPAPKIAECKSSEMLHEQLP
QA-IPDINQFNVRNLTVKDIKRGNVSDAKNOQPAGCEDPTAQVIVNHPPQIQKGYTP
VLDCHTAHIAACKFETIKDIDRRTGKSMEGQPIEKNGDAALTVLITPALKCVVPQYEYP
PLGRYAARMDKQTVAVGVIKKE----------------------------------TP
```

I suggest renaming this file with a name that makes more sense to you. We won’t need SeqLab anymore in today’s session, so exit it with the “File” menu “Exit” choice and 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 HPC. Next, we can convert the FastA format file just produced to PHYLIP compatible format.

To do this we’ll run Don Gilbert’s public-domain program ReadSeq. This program can be used to change your FastA format file into a format acceptable to PHYLIP. A ReadSeq limitation is it doesn’t recognize that you want to just use portions of your alignment, nor does it automatically convert dots and tildes to hyphens. However, since SeqLab has taken care of these points, it’ll work just fine for us here. The command line version of ReadSeq runs a bit backward from what most people are used to, though. Begin the program by typing “readseq” at your command prompt in your 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 while running the program, 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 FastA format sequence. It will not be named “Lab8.fa” as mine is, unless you renamed it that. It’ll have the cryptic default name “seqconv+.fa” if you didn’t rename it. (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. A sample screen trace is shown below; as usual, responses are shown in bold:

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

Name of output file (?=help, defaults to display):
Lab8.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. MFP
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:
Lab8.fa
Sequences in Lab8.fa (format is  8. Pearson/Fasta)
1)  EF1A_TETPY Author:        HMMER 2.1.1
2)  EF1A_ENTHI Author:        HMMER 2.1.1
3)  EF1A_TRYBB Author:        HMMER 2.1.1
4)  EF1A_EUGGR Author:        HMMER 2.1.1
5)  EF1A_STYLE Author:        HMMER 2.1.1
6)  EF1A_DICDI Author:        HMMER 2.1.1
7)  EF1A_PODAN Author:        HMMER 2.1.1
8)  EF1A_PODCU Author:        HMMER 2.1.1
54)  EF1A_HYDAT Author:       HMMER 2.1.1
55)  EF1C_PORPU Author:       HMMER 2.1.1
56)  EF1A_CRYPV Author:       HMMER 2.1.1
57)  EF1A_PLAER Author:       HMMER 2.1.1
58)  EF11_EUPCR Author:       HMMER 2.1.1
59)  EF1A_BLAHO Author:       HMMER 2.1.1
60)  EF11_XENLA Author:       HMMER 2.1.1
61)  EF1A_EIMBO Author:       HMMER 2.1.1
62)  EF1A_GIALA Author:       HMMER 2.1.1
63)  EF12_EUPCR Author:       HMMER 2.1.1
64)  1EFT Author:             HMMER 2.1.1

Choose a sequence (# or All):
all
This format requires equal length sequences.
Sequence truncated or padded to fit.

Name an input sequence or -option: <rtn>

Never mind if you happen to get the “... 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:

```
$ tr \~\ . \- < infile.phy > outfile.phy  but you should not need to use it in this tutorial!
```

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

```
$ more Lab8.phy
```

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

```plaintext
64 463
EF1A_TETPY  MARGKVHINL VVIGHVDSGK STTTGHLIYK CGGIDKRVIE KFEKESAEQG
EF1A_ENTHI  MPEKKTHINLI VVIGHVDSGK STTTGHLIYK CGGIDQRTIE KFEKESAEMG
EF1A_TRYBB  MGKKEKMNML VVGHVDSGK STATGHLIYK CGGIDKRITIE KFEKEAADIG
EF1A_EUGGR  MGKEKVFISIL VVIGHVDSGK STTTGHLIYK CGGIDKRITIE KFEKEASEMG
EF1A_STYLE  MPEKKINHLNL VVIGHVDSGK STTGHHLIYK CGGIDKRTIE KFEKEAAELG
EF1A_SORNK  MGKKEHINIVL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAEELG
EF1A_NEUCR  MGKKEKINIVL VVIGHVDSGK STTTGHLIYQ CGGIDKRITIE KFEKEAELG
EF1A_TRIRE  MGKKEKINIVL VVIGHVDSGK STTTGHLIYQ CGGIDKRITIE KFEKEAELG
EF1A_AJICA  MGKKEKINIVL VVIGHVDSGK STTGHHLIYK CGGIDKRTIE KFEKEAELG
EF1A_COCON  MGKKEKINIVL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_ASMOR  MGKKEKINIVL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_AREPU  MGKKEKINIVL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF12_SCHPO  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF15_SCHPO  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF13_SCHPO  MGKKEKHHVNL VVIGHVDSGK FTTTGHHLIYK CGGIDKRTIE KFEKEAELG
EF11_SCHPO  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_YEAST  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_MASTER  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_YRLLI  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_ARXAD  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF11_RHAR  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF12_RHAR  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF13_RHAR  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_ABSOL  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
EF1A_SCHCO  MGKKEKHHVNL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAELG
```

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 the sequence data itself.

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, “?”s, 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. Don’t bother performing this question mark editing procedure today though.

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 limited punctuation and mixed capitalization, and can be up to ten characters in length.

**Using PHYLIP for phylogenetic inference**

**Another estimate of the distance matrix: PHYLIP’s ProtDist**

Let’s use the PHYLIP package now to see how its results differ from the previous analyses. I’ll show the use of a couple of the distance methods in the package first. All the distance methods require a distance matrix to be prepared first 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 uses default filenames that need to be dealt with. A menu allows you to pick desired options within the program. 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 protein distance matrix program, ProtDist, allows the use of four alternate multiple substitution correction models. The default JTT correction model is more powerful than any of the GCG’s Distances protein options or the simple protein models included in PAUP*. Notice it also allows a gamma distribution site rate variation parameter, although we won’t be taking advantage of that today. Begin the program by typing “protdist.” When the program asks you for an input file, specify your converted PHYLIP format alignment. At the “Are these settings correct?” prompt type “y” for yes. Issue the following command in your terminal window and respond with the appropriate bold-faced text commands to produce the following screen trace:

```
$ protdist
protdist: can't find input file "infile"
Please enter a new file name> Lab8.phy

Protein distance algorithm, version 3.65

Settings for this run:
P  Use JTT, PMB, PAM, Kimura, categories model?  Jones-Taylor-Thornton matrix
G  Gamma distribution of rates among positions?  No
C  One category of substitution rates?  Yes
W  Use weights for positions?  No
M  Analyze multiple data sets?  No
I  Input sequences interleaved?  Yes
O  Terminal type (IBM PC, ANSI)?  ANSI
```
1       Print out the data at start of run  No
2       Print indications of progress of run  Yes

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

y

Computing distances:
EF1A_TETPY          .
EF1A_ENTHI          ..
EF1A_TRYBB          ...
EF1A_EUGGR          ...
EF1A_STYLE          ....
EF1A_DICDI          .....  
EF1A_PODAN          ......  
EF1A_PODCU          ........  
EF1A_SORMA          ........
EF1A_NEUCR          ........
EF1A_TRIRE          .......
EF1A_AJECA          ........
EF1A_COCIM          ..........  
EF1A_ASPOR          ..........  
EF1A_AURPU          ..........  
EF12_SCHPO          ........
EF13_SCHPO          ........
EF11_SCHPO          ........
EF1A_YEAST          ........
EF1A_ASHGO          ........  
EF1A_CANAL          ........
EF1A_ARXAD          ........
EF11_RHIRA          ........
EF12_RHIRA          ........

///////////-----------------------------------

Output written to file "outfile"

The distance matrix output file is named outfile by default. Immediately rename this file with the move command, "mv," so that the next program doesn’t complain about it!

$ mv outfile JTT.phydist

The distance matrix is very long, but you’re welcome to take a look with the “more” command if you want.

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-squares fit is a powerful way of estimating a tree from distance data but it is somewhat 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 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 *Thermus aquaticus* as the outgroup to add directionality to the tree, but do remember that designating an outgroup in no way changes the topology of a phylogenetic tree, it only changes its visual impression (which can sometimes be very powerful).
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 also be taken advantage of. It is a very good idea to jumble multiple times, at least ten, but do it only once here for run time sake. Supply jumble with any odd number as a random number seed. The Fitch screen trace follows; as always, follow the bold prompts:

```
$ fitch
fitch: can't find input file "inFILE"
Please enter a new file name> JTT.phyDIST

Fitch-Margoliash method version 3.65

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
J   Randomize input order of species?  No. Use input order
M   Analyze multiple data sets?  No
1   Terminal type (IBM PC, ANSI, none)?  ANSI
2   Print out the data at start of run  No
3   Print indications of progress of run  Yes
4   Print out tree  Yes
5   Write out trees onto tree file?  Yes

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

Type number of the outgroup:
64

Fitch-Margoliash method version 3.65

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?  Yes, at species number 64
L   Lower-triangular data matrix?  No
R   Upper-triangular data matrix?  No
S   Subreplicates?  No
G   Global rearrangements?  No
J   Randomize input order of species?  No. Use input order
M   Analyze multiple data sets?  No
1   Terminal type (IBM PC, ANSI, none)?  ANSI
2   Print out the data at start of run  No
3   Print indications of progress of run  Yes
4   Print out tree  Yes
5   Write out trees onto tree file?  Yes

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

32
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?  Yes, at species number 64
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)?  ANSI
1  Print out the data at start of run  No
2  Print indications of progress of run  Yes
3  Print out tree  Yes
4  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)?
543
Number of times to jumble?
10  (but you should probably only do one because of time issues!)

Adding species:
1. EF1A_TETPY
2. EF1A_ENTHI
3. EF1A_TRYBB
4. EF1A_EUGGR
5. EF1A_STYLE
6. EF1A_DICDI
7. EF1A_PODAN
8. EF1A_PODCU
9. EF1A_SORMA
10. EF1A_NEUCR
11. EF1A_TRIRE
12. EF1A_AJJECA
13. EF1A_COCIM
14. EF1A_ASPOR
15. EF1A_AURPU
16. EF12_SCHPO
17. EF15_SCHPO
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 JTT.fitc
$ mv outtree JTT.fitchtree
```

The output text file, called outfile by default, shows the least-squares fit formula used, an ASCII representation of the tree, and it lists all branch lengths. Unfortunately it can have very wide lines, if you’ve got a wide range of divergence in your data, producing inappropriate line wraps making it difficult to read. The text output from my Fitch run is shown below:

64 Populations

Fitch-Margoliash method version 3.65

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

Negative branch lengths not allowed

global optimization

```
+---------EF12_EUPCR
  +25
  !  +---EF11_EUPCR
  !  !  +-----EF1A_BLAHO
  !  !  +35
  !  !  !  +-----EF1A_GIALA
  !  !  !  !  +-----EF1A_ENTHI
  !  !  !  !  !  +-----EF1A_EIMBO
  !  !  !  !  !  !  +21
  !  !  !  !  !  !  !  +-----EF1A_PLAFK
  !  !  !  !  !  !  !  +62
  !  !  !  !  !  !  !  !  +-----EF1A_CRYPV
  !  !  !  !  !  !  !  !  !  +---EF1A_PUCGR
  !  !  !  !  !  !  !  !  !  !  +29
  !  !  !  !  !  !  !  !  !  !  !  +---EF1A_CRYNE
  !  !  !  !  !  !  !  !  !  !  !  !  +61
  !  !  !  !  !  !  !  !  !  !  !  !  !  +EF1A_PIRIN
  !  !  !  !  !  !  !  !  !  !  !  !  !  !  +40
```
remember: (although rooted by outgroup) this is an unrooted tree!

Sum of squares =  15.23711
Average percent standard deviation =  6.14892

<table>
<thead>
<tr>
<th>Between</th>
<th>And</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
<td>0.02768</td>
</tr>
<tr>
<td>25</td>
<td>EF12_EUPCR</td>
<td>0.31840</td>
</tr>
<tr>
<td>25</td>
<td>EF11_EUPCR</td>
<td>0.14398</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>0.01651</td>
</tr>
<tr>
<td>14</td>
<td>35</td>
<td>0.01754</td>
</tr>
<tr>
<td>35</td>
<td>EF1A_BLAHO</td>
<td>0.17806</td>
</tr>
<tr>
<td>35</td>
<td>EF1A_GIALA</td>
<td>0.30195</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>0.00350</td>
</tr>
<tr>
<td>6</td>
<td>EF1A_ENTHI</td>
<td>0.14227</td>
</tr>
<tr>
<td>6</td>
<td>27</td>
<td>0.00323</td>
</tr>
<tr>
<td>27</td>
<td>45</td>
<td>0.00778</td>
</tr>
<tr>
<td>45</td>
<td>21</td>
<td>0.03798</td>
</tr>
<tr>
<td>21</td>
<td>EF1A_EIMBO</td>
<td>0.18293</td>
</tr>
<tr>
<td>21</td>
<td>62</td>
<td>0.02191</td>
</tr>
<tr>
<td>62</td>
<td>EF1A_PLAFK</td>
<td>0.18613</td>
</tr>
<tr>
<td>62</td>
<td>EF1A_CRYPV</td>
<td>0.10103</td>
</tr>
<tr>
<td>45</td>
<td>19</td>
<td>0.00687</td>
</tr>
<tr>
<td>19</td>
<td>12</td>
<td>0.00645</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>0.01378</td>
</tr>
<tr>
<td>8</td>
<td>31</td>
<td>0.03539</td>
</tr>
<tr>
<td>31</td>
<td>3</td>
<td>0.00486</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0.00727</td>
</tr>
</tbody>
</table>
Notice the *Thermus aquaticus* sequence is again being forced to lie to the outside of everything else by its designation as an outgroup, and it is sharing the base of the tree with several protists. Surprisingly, but reassuringly, they are not forming a discrete clade. As mentioned earlier, we believe this is actually the case. Perhaps the more accurate JTT model did a much better job estimating the proper phylogeny of these sequences. Take a look at the tree file to get a feel for the Newick format. My corresponding Fitch tree file follows:

```
(((EF1A_PLAFK:0.19419,EF1A_CRYPV:0.10141):0.03376,((EF12_EUPCR:0.33365,
EF11_EUPCR:0.15305):0.04555,((EF1A_TETPY:0.15301,(EF1A_BLAHO:0.22008,
((EF1A_TRYBB:0.12681,EF1A_STYLE:0.14702,(EF1A_EUGGR:0.11838):0.01095):0.02369,
(((EF1A_HYDAT:0.15106,((EF1A_ONCVO:0.08492,EF1A_CAEE:0.06938):0.04485,
(((EF11_DROME:0.03950,EF1A_BROOMO:0.02530):0.02599,((EF12_DROME:0.05444,
EF1A_API:0.00576):0.00889):0.05301,(EF11_XENLA:0.31632,
(EF1A_OYLA:0.07025,(((EF1A_CHICK:0.00368,(EF11_HUMAN:0.00008,
(EF11_MOUSE:0.00105,EF11_CRIGR:0.00115):0.00107):0.00302):0.01608,
EF10_XENLA:0.02210):0.02164,((EF12_HUMAN:0.00148,EF12_MOUSE:0.00072):0.07020):0.0
0377,
EF1A_BRARE:0.04436):0.00746,(EF13_XENLA:0.00759,EF12_XENLA:0.00831):0.05704):0.0
0879):0.00275):0.00822):0.01069):0.01138,
(((EF1A_YEAST:0.03689,EF1A_ASHGO:0.03031):0.02206,EF1A_CANAL:0.03834):0.0104
5,
EF1A_YARLI:0.06042):0.00123,EF1A_ARXAD:0.05643):0.01436,
(EF1A_AURPU:0.05033,((EF1A_ASPOR:0.05461,EF1A_AJEC:0.03228,
EF1A_COCCIN:0.02592):0.02984):0.01125),((EF1A_PODAN:0.01014,
EF1A_PODCU:0.01486):0.02327,((EF1A_SORMA:0.01401,EF1A_NEUCR:0.01289):0.01552,
EF1A_TRIRE:0.03899):0.00790):0.02124):0.00868):0.00879,((EF1A_CRYNE:0.08762,
EF1A_PIRIN:0.03809,EF1A_SCHCO:0.03021):0.01908):0.01237,
EF1A_PUCGR:0.08722):0.0086,(EF1A_ABSGL:0.03242,(EF13_RHIRA:0.00204,
(EF11_RHIRA:0.00220,((EF12_RHIRA:0.00000):0.00246):0.03866):0.03065):0.00780):0.0
0565,
(EF12_SCHPO:0.00000,((EF13_SCHPO:0.00328,EF11_SCHPO:0.00122):0.00063,
EF12_SCHPO:0.00059):0.00161):0.08493):0.03664):0.01535,((EF1A_EIMBO:1.09305,
(EF1A_RHYAM:15.57566,EF1A_GIALA:0.00000):0.053897):5.93415,
(EF1A_ARTSA:20.23175,EF1A_SPOFR:0.00000):0.00250,EF1A_HELVI:0.00000):5.66385):1
.7304,
EF1A_DICDI:0.00000):0.15860,(EF1C_PORPU:0.15828):0.01748):0.00778):0.00875):0.009
05):0.00301,
EF1A_ENTHI:0.15352):0.00496):0.01854,1EFT:1.11067);
```

Many PHYLIP programs generate evolutionary tree Newick format data files. Two PHYLIP drawing routines, DrawTree and DrawGram, can produce graphical tree representations from them. However, we will delay using them until we’ve run through all the PHYLIP tree inference programs that I want to introduce.

**Bootstrapping neighbor-joining techniques**

Next I'll show how to run a bootstrapped neighbor-joining analysis. As described in the Introduction, 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 consensus 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:

```
$ seqboot
seqboot: can't find input file "infile"
Please enter a new file name> Lab8.phy

Bootstrapping algorithm, version 3.65

Settings for this run:
  D  Sequence, Morph, Rest., Gene Freqs?  Molecular sequences
  J  Bootstrap, Jackknife, Permute, Rewrite?  Bootstrap
  %  Regular or altered sampling fraction?  regular
  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)?  ANSI
  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)? 9875

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 Lab8.seqboot

The output alignment data matrix now contains 100 randomly selected sequence data matrices. To generate
100 distance matrices, launch ProtDist, being very careful to specify "multiple" datasets with the "m" option.
I’ll again use the JTT model without any rate heterogeneity specification. Issue the following command line, to
see the accompanying, much abridged screen trace (This will take a while to run!):
$ protdist
protdist: can't find input file "infile"
Please enter a new file name> Lab8.seqboot

Protein distance algorithm, version 3.65

Settings for this run:
P Use JTT, PMB, PAM, Kimura, categories model? Jones-Taylor-Thornton matrix
G Gamma distribution of rates among positions? No
C One category of substitution rates? Yes
W Use weights for positions? No
M Analyze multiple data sets? No
I Input sequences interleaved? Yes
O Terminal type (IBM PC, ANSI)? ANSI
1 Print out the data at start of run No
2 Print indications of progress of run Yes

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

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

How many data sets?
100

Protein distance algorithm, version 3.65

Settings for this run:
P Use JTT, PMB, PAM, Kimura, categories model? Jones-Taylor-Thornton matrix
G Gamma distribution of rates among positions? No
C One category of substitution rates? Yes
W Use weights for positions? No
M Analyze multiple data sets? Yes, 100 data sets
I Input sequences interleaved? Yes
O Terminal type (IBM PC, ANSI)? ANSI
1 Print out the data at start of run No
2 Print indications of progress of run Yes

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

Data set # 1:

Computing distances:
EF1A_TETPY
EF1A_ENTHI .
EF1A_TRYBB ..
EF1A_EUGGR ... 
EF1A_STYLE ....
EF1A_DICDI ......
EF1A_PODAN .......
EF1A_PODCU ........ 
EF1A_SORMA ..........
EF1A_NEUCR ..........
EF1A_TRIRE ..........
EF1A_AJECA ..........
EF1A_COCIM ..........
EF1A_ASPOR ..........
EF1A_AURPU ..........
EF12_SCHPO ............
EF15_SCHPO ............
EF13_SCHPO ............
EF11_SCHPO ............
EF1A_YEAST ............
EF1A_ASHGO ................
EF1A_EIMBO ................................................
EF1A_GIALA ................................................
EF12_EUPCR .............................................

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

39
Rename the outfile to something that identifies it:

```bash
$ mv outfile JTT.boot.dist
```

Now submit the 100 matrix distance file to Neighbor. Neighbor-joining algorithms aren’t the greatest for tree inference, but they are very fast, which we need here because we’re dealing with 100 datasets and a limited amount of lab time. 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. Issue the following command line to see the drastically shortened screen trace below:

```bash
$ neighbor
neighbor: can't find input file "infile"
Please enter a new file name> JTT.boot.dist
```

**Neighbor-Joining/UPGMA method version 3.65**

**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
- 0 Terminal type (IBM PC, ANSI, none)? ANSI
- 1 Print out the data at start of run No
- 2 Print indications of progress of run Yes
- 3 Print out tree Yes
- 4 Write out trees onto tree file? Yes

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

Type number of the outgroup:

64

**Neighbor-Joining/UPGMA method version 3.65**

**Settings for this run:**

- N Neighbor-joining or UPGMA tree? Neighbor-joining
- O Outgroup root? Yes, at species number 64
- 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
- 0 Terminal type (IBM PC, ANSI, none)? ANSI
- 1 Print out the data at start of run No
- 2 Print indications of progress of run Yes
- 3 Print out tree Yes
- 4 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)?

97
Neighbor-Joining/UPGMA method version 3.65

Settings for this run:
N Neighbor-joining or UPGMA tree? Neighbor-joining
O Outgroup root? Yes, at species number 64
L Lower-triangular data matrix? No
R Upper-triangular data matrix? No
S Subreplicates? No
J Randomize input order of species? Yes (random number seed = 97)
M Analyze multiple data sets? No
0 Terminal type (IBM PC, ANSI, none)? ANSI
1 Print out the data at start of run No
2 Print indications of progress of run Yes
3 Print out tree Yes
4 Write out trees onto tree file? Yes

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

How many data sets?
100
Random number seed (must be odd)?
975

Neighbor-Joining/UPGMA method version 3.65

Settings for this run:
N Neighbor-joining or UPGMA tree? Neighbor-joining
O Outgroup root? Yes, at species number 64
L Lower-triangular data matrix? No
R Upper-triangular data matrix? No
S Subreplicates? No
J Randomize input order of species? Yes (random number seed = 975)
M Analyze multiple data sets? Yes, 100 sets
0 Terminal type (IBM PC, ANSI, none)? ANSI
1 Print out the data at start of run No
2 Print indications of progress of run Yes
3 Print out tree Yes
4 Write out trees onto tree file? Yes

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

Cycle  2: node 45 ( 0.00742) joins node 8 ( 0.03119)
Cycle  1: node 45 ( 0.01041) joins node 13 ( 0.01881)
last cycle:
species 15 ( 0.05845) joins node 22 ( 0.02222) joins node 45 ( 0.00022)

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

Data set # 100:

Cycle  61: species 52 ( 8.76261) joins species 53 ( 4.17269)
Cycle  60: species 62 ( -2.73369) joins node 52 (10.27304)
Cycle  59: species 61 ( 1.80554) joins node 62 ( 5.93753)
Cycle  58: species 46 ( 0.00000) joins species 47 ( 0.00000)
Cycle  57: node 61 ( 8.34951) joins species 60 ( -2.22450)
Cycle  56: species 46 ( 5.56029) joins species 64 ( -0.13869)

Output written on file "outfile"
Tree written on file "outtree"
node 14 ( 0.00720) joins node 31 ( 0.00485) joins node 51 ( 0.00701)

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

Done.

The output text file, “outfile,” 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 JTT.neighbor.boot.tree

To condense all this information down into a single consensus tree with branch lengths proportional to bootstrap confidence levels run the PHYLIP program Consense. Launch the following command line; specify the input tree file and designate an 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, 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 Consense screen trace as a guide:

$ consense
consense: can't find input tree file "intree"
Please enter a new file name> JTT.neighbor.boot.tree

Consensus tree program, version 3.65

Settings for this run:
C  Consensus type (MRe, strict, MR, ML): Majority rule (extended)
0  Outgroup root: No, use as outgroup species 1
R  Trees to be treated as Rooted: No
T  Terminal type (IBM PC, ANSI, none): ANSI
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:
64

Consensus tree program, version 3.65

Settings for this run:
C  Consensus type (MRe, strict, MR, ML): Majority rule (extended)
0  Outgroup root: Yes, at species number 64
R  Trees to be treated as Rooted: No
T  Terminal type (IBM PC, ANSI, none): ANSI
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)
42
Consensus tree written to file "outtree"
Output written to file "outfile"

Done.

Rename both output files:

$ mv outfile JTT.neighbor.boot.consense
$ mv outtree JTT.neighbor.boot.contree

Check out the text file output from the Consense program. The numbers at the branch forks indicate the bootstrap value. Branch lengths may again be too long for your display — adjust your terminal accordingly. The following abridged display show the results of my Consense run:

Consensus tree program, version 3.65
Species in order:
1. EF12 EUPCR
2. EF1A TETPY
3. EF1A STYLE
4. EF1A EUGGR
5. EF1A BLAHO
6. EF1A TRYBB
7. EF1A PUCGR

Sets included in the consensus tree
Set (species in order) How many times out of 100.00

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
remember: (though rerooted by outgroup) this is an unrooted tree!

Notice some nodes are very well resolved with bootstrap values as high as 100%. In particular I am impressed that all of the fungi are held together as a sister group to the animals with a bootstrap value of 99%. This is a great indication of the reliability of the phylogenetic inference of those nodes based on the prepared dataset and the inference model. However, several others, in particular those that have had trouble with other techniques, especially EF11_XENLA, reflect very low confidence values with numbers ranging from about 50% on down. In general bootstrap values above about 70% tend to under-represent the actual confidence level of the branching pattern, whereas numbers below 30% over-represent confidence levels. A ‘rule-of-thumb’ that many people accept is bootstrap values above around 70% indicate the cohesiveness of a particular clade; whereas values from around 60% to 70% 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.

```
(((((((((EF13_XENLA:100.0,EF12_XENLA:100.0):100.0,((((EF11_HUMAN:100.0,
(EF11_MOUSE:100.0,EF11_CRIGR:100.0):78.0):87.0,EF1A_CHICK:100.0):100.0,EF10_XENLA:100.0):100.0,
EF1A_BRARE:100.0):55.0,(EF12_HUMAN:100.0,EF12_MOUSE:100.0):100.0):33.0):52.0,
EF1A_GYRLA:100.0):44.0,(((EF1A_SPOFR:100.0,EF1A_HELVI:100.0):42.0,EF1A_BOMHO:100.0):39.0,
(EF1A_RHYAM:100.0,EF1D_DROME:100.0):61.0):98.0,(EF1A_APIME:100.0,EF12_DROME:100.0):64.0):60.0,
EF1A_ARTSA:100.0):100.0):52.0,(EF1A_ONCVO:100.0,EF1A_CAEEL:100.0):100.0):48.0,
EF1A_HYDAT:100.0):50.0,(((EF11_SCHPO:100.0,EF13_SCHPO:100.0):54.0,EF15_SCHPO:100.0):42.0,
EF12_SCHPO:100.0):100.0,(((EF1A_ARXAD:100.0,EF1A_YARLI:100.0):58.0,(EF1A_ASHGO:100.0,
EF1A_YEAST:100.0):99.0,EF1A_CANAL:100.0):83.0):77.0,(((EF1A_NEUCR:100.0,
EF1A_SORMA:100.0):100.0,EF1A_TRIRE:100.0):90.0,(EF1A_PODCU:100.0,EF1A_PODAN:100.0):99.0):100.0,
((EF1A_ASPOR:100.0,(EF1A_COCIM:100.0,EF1A_AJECIA:100.0):100.0):86.0,EF1A_AURPU:100.0):70.0):63.0):58.0,
(((EF1A_CRYNE:100.0,(EF1A_SCHCO:100.0,EF1A_PIRIN:100.0):99.0):94.0,EF1A_PUCGR:100.0):65.0,
(((EF12_RHIRA:100.0,EF11_RHIRA:100.0):63.0,EF13_RHIRA:100.0):100.0,EF1A_ABSGL:100.0):98.0):59.0):67.0):9
9.0):41.0,
EF11_XENLA:100.0):15.0,((EF1A_BLAHO:100.0,EF1A_GIALA:100.0):30.0,(EF1C_PORPU:100.0,
EF1A_DICDI:100.0):44.0):7.0):3.0,(((EF1A_PLAK:100.0,EF1A_CRYPV:100.0):55.0,
EF1A_EINBO:100.0):54.0,((EF1A_STYLE:100.0,(EF1A_EUGGR:100.0,EF1A_TRYBB:100.0):51.0):57.0,
EF1A_TETPY:100.0):18.0):3.0,(EF1A_ENTHI:100.0,EF11_EUPCR:100.0):9.0):2.0):50.0,
EF12_EUPCR:100.0):100.0,1EFT: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.
Finally I will show how to run one of PHYLIP’s parsimony programs, ProtPars. 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?” In ProtPars’ case those symbol changes are based on how many implied nucleotide changes each amino acid change represents based on the genetic code. ProtPars uses an approximate, heuristic, tree-rearrangement solution, although a much slower branch-and-bound, exact solution program, DNAPenny, is available in PHYLIP for DNA sequences. 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 in parsimony is the reconstruction of ancestral node sequences; choose option 5 to designate, if this interests you. My interactive screen trace follows:

```
$ protpars
protpars: can't find input file "infile"
Please enter a new file name> Lab8.phy

Protein parsimony algorithm, version 3.65

Setting for this run:
  U  Search for best tree? Yes
  J  Randomize input order of sequences? No. Use input order
  O  Outgroup root? No, use as outgroup species 1
  T  Use Threshold parsimony? No, use ordinary parsimony
  C  Use which genetic code? Universal
  W  Sites weighted? No
  M  Analyze multiple data sets? No
  I  Input sequences interleaved? Yes
  0  Terminal type (IBM PC, ANSI, none)? ANSI
  1  Print out the data at start of run No
  2  Print indications of progress of run Yes
  3  Print out tree Yes
  4  Print out steps in each site No
  5  Print sequences at all nodes of tree No
  6  Write out trees onto tree file? Yes

Are these settings correct? (type Y or the letter for one to change)
j
Random number seed (must be odd)?
321
Number of times to jumble?
10  (but you should only do it once to save time.)

Protein parsimony algorithm, version 3.65

Setting for this run:
  U  Search for best tree? Yes
  J  Randomize input order of sequences? Yes (seed = 321, 10 times)
  O  Outgroup root? No, use as outgroup species 1
  T  Use Threshold parsimony? No, use ordinary parsimony
  C  Use which genetic code? Universal
  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
```
Are these settings correct? (type Y or the letter for one to change)

Type number of the outgroup:
64

Protein parsimony algorithm, version 3.65

Setting for this run:
U  Search for best tree? Yes
J  Randomize input order of sequences? Yes (seed = 321, 10 times)
O  Outgroup root? Yes, at sequence number 64
T  Use Threshold parsimony? No, use ordinary parsimony
C  Use which genetic code? Universal
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

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

Adding species:
1. EF1A_ARTSA
2. EF11_CRIGR
3. EF15_SCHPO
4. EF1A_BLAHO
5. EF1A_PIRIN
6. EF11_XENLA
7. EF1A_ABSGL
8. EF1A_BOMMO
9. EF1A_PODAN
10. EF12_SCHPO
62. EF1A_COCIM
63. EF1A_ABSGL
64. EF1A_ASHGO

Don't forget to rename the output files:

$ mv outfile phylip.pars
$ mv outtree phylip.parstree

The outfile shows how many trees were found and illustrates them with a text schematic. My output showed that I had found three most parsimonious trees. As mentioned previously, 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 — no likelihoods are used in parsimony analysis.

The output tree file is the simple nested Newick parenthetical form; it follows below:

```
(1EFT,(EF1A_GIALA,(EF1A_ENTHI,((EF1A_BLAHO,((EF12_EUPCR,EF11_EUPCR),
(EF1A_PLA FK,EF1A_CRYPV)),(EF1C_PORPU,(EF11_XENLA,(EF1A_HYDAT,
(EF1A_CA EEL,EF1A_ONCVO)),(((EF1A_ARTSA,((EF1A_APIME,EF12_DROME),
(EF11_DROME,(EF1A_BOMMO,((EF1A_EIMBO,EF1A_RHYAM),(EF1A_HELVI,
EF1A_SPOFR)))))),EF1A_ORYLA),EF1A_BRA RE),((EF10_XENLA,((EF1A_CHICK,
EF11_MOUSE),EF11_CRIGR),EF11_HUMAN)),((EF12_MOUSE,EF12_HUMAN),
(EF12_XENLA,EF13_XENLA))),(E F15_SCHPO,((EF11_SCHPO,EF13_SCHPO),
EF12_SCHPO)),((EF1A_PIRIN,EF1A_SCHCO),((EF1A_PUCGR,(EF1A_CRYNE,
(EF1A_ABSGL,((EF13_RHIRA,((EF12_RHIRA,EF11_RHIRA))))),(EF1A_YARLI,
((EF1A_ARXAD,((EF1A_CANAL,(EF1A_ASHGO,EF1A_YEAST)),(EF1A_ASPOR,
(EF1A_CO CIM,EF1A_AJEC A)),(EF1A_AURPU,(EF1A_TRIRE,((EF1A_NEUCR,
EF1A_SORMA),EF1A_PODCU,EF1A_PODAN))))),EF1A_DICDI)),
(EF1A_STYLE,(EF1A_EUGGR,EF1A_TRYBB))),EF1A_TETPY)))[0.3333];
```

```
1EFT,(EF1A_GIALA,(EF1A_ENTHI,((EF1A_BLAHO,((EF12_EUPCR,EF11_EUPCR),
(EF1A_PLA FK,EF1A_CRYPV)),(EF1C_PORPU,(EF11_XENLA,(EF1A_HYDAT,
(EF1A_CA EEL,EF1A_ONCVO)),(((EF1A_ARTSA,((EF1A_APIME,EF12_DROME),
(EF11_DROME,(EF1A_BOMMO,((EF1A_EIMBO,EF1A_RHYAM),(EF1A_HELVI,
EF1A_SPOFR)))))),EF1A_ORYLA),EF1A_BRA RE),((EF10_XENLA,((EF1A_CHICK,
EF11_MOUSE),EF11_CRIGR),EF11_HUMAN)),((EF12_MOUSE,EF12_HUMAN),
(EF12_XENLA,EF13_XENLA))),(E F15_SCHPO,((EF11_SCHPO,EF13_SCHPO),
EF12_SCH PO)),((EF1A_PIRIN,EF1A_SCHCO),((EF1A_PUCGR,(EF1A_CRYNE,
(EF1A_ABSGL,((EF13_RHIRA,((EF12_RHIRA,EF11_RHIRA))))),(EF1A_YARLI,
((EF1A_ARXAD,((EF1A_CANAL,(EF1A_ASHGO,EF1A_YEAST)),(EF1A_ASPOR,
(EF1A_CO CIM,EF1A_AJEC A)),(EF1A_AURPU,(EF1A_TRIRE,((EF1A_NEUCR,
EF1A_SORMA),EF1A_PODCU,EF1A_PODAN))))),EF1A_DICDI)),
(EF1A_STYLE,(EF1A_EUGGR,EF1A_TRYBB))),EF1A_TETPY)))[0.3333];
```

We will use DrawGram to plot graphics of these trees later. The drawing programs in the PHYLIP system will only read the first tree in a file containing more than one tree. If you really want to draw graphics of all the trees in a PHYLIP multiple tree outtree, you would need to copy the file and edit each copy to create a series of tree files, each containing one tree apiece. That way you could produce output plots of each separate tree.
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 parsimony is 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 hard to recognize beforehand. 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) DNA codon position saturation in coding sequences, compositional biases, recombination, 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.

Maximum likelihood techniques

An optimality criteria that we will not be using today, but that I mentioned in the Introduction and heartily recommend is maximum likelihood (ML). These algorithms, along with Bayesian methods, 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 (1981), 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 have sped up the process considerably. Datasets the size that we are dealing with would take longer to evaluate than we have, so I am not going to have you run it here. Regardless, let’s discuss some of the implementations.

The PHYLIP DNA maximum likelihood program is DNAML, but it is the slowest of the group. Gary Olsen recoded DNAML to produce fastDNAML (1994), considerably speeding up the original algorithm. PHYLIP also has a very powerful protein maximum likelihood implementation called ProML. ProML allows you to choose between three different amino acid transition models, and to model site rate heterogeneity in a variety of ways. Another protein implementation of maximum likelihood, ProtML is a part of the MOLPHY 2.2 package (Adachi and Hasegawa, 1994) available by ftp from sunmh.ism.ac.jp. ProtML doesn’t allow site rate variation though. PAUP*’s ML implementation was a pioneer in the field, particularly in the area of DNA model availability, and is used the world over, although current production versions do not incorporate amino acid models. Another implementation is Tree-Puzzle (Strimmer and von Haeseler, 1996). It works very quickly on either nucleic or amino acid sequences and performs well, although some people argue that quartet puzzling is an inefficient heuristic searching technique. Quartet puzzling automatically assigns node support values though, which can be thought of as, and substituted for, bootstrap values in most instances. Two other, quite recent ML programs available on FSU’s HPC are RAxML (Stamatakis, 2006) and Garli (Zwickl, 2006). Both are incredibly faster than older ML implementations, and parallel versions allow ML
bootstrapping in reasonable time frames. A real advantage of using any of these newer implementations 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.

One warning: the PHYLIP tree drawing programs that I’ll show below, DrawTree and DrawGram, can’t cope with the dual parameter, branch length and node support, values that some of the newer program’s output tree files can contain. Try TreeView (http://taxonomy.zoology.gla.ac.uk/rod/treview.html) or FigTree (http://tree.bio.ed.ac.uk/software/figtree/) instead. They know how to deal with this situation and allow you to visualize and manipulate trees with their branches proportional to evolutionary divergence and their node support values displayed concurrently.

ML (along with Bayesian techniques) is arguably the most accurate and powerful molecular phylogenetic inference algorithm available. But, before starting any turf wars, let’s get it straight right up front: in general using ML to estimate all of its own model parameters, and to estimate the best tree simultaneously, is not a good idea on most computers! It takes far too long. In spite of that, I consider ML to be the method of choice; it just requires a few 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, and then specify those estimated parameters to find an even better ML tree. You can iterate this process, refining your estimates with each pass, and testing the results against one another with likelihood ratio tests and other statistical methods. This is the process used automatically by the program ModelTest (Posada and Crandall, 1998). The timesaving can be significant — analyses that would be impossible, if you were estimating all the parameters and the best tree simultaneously, can be realistically run. As we’ve already seen, the models really do matter; parameters can make a huge difference. One assumption mentioned in the Introduction that can make an enormous difference is a lack of rate heterogeneity along the sequences. Often this is modeled with a gamma distribution parameter $\alpha$, so be sure to take advantage of models that allow this, and estimate a valid one. A couple of nice World Wide Web resources can be very helpful in this overall process. They can be found at:

http://darwin.uvigo.es/software/modeltest.html and
http://workshop.molecularevolution.org/resources/models/

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 possible using the GCG PAUPSearch interface. You can only do it exterior to GCG. As mentioned in the Introduction, to launch the native version of PAUP* use the command “paup” and then use the built in help system and the Command Reference Manual to learn further. I can provide more individual help in this area, and encourage you to check these, and Bayesian, methods out!

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 and through the World Wide Web for assistance with all of these maximum likelihood programs. If you need to use any of these programs, or the Bayesian method MrBayes, contact me for further assistance.

Running PHYLIP in the background

Unlike GCG programs in SeqLab that run in the background automatically, PHYLIP requires manual background submission, and the HPC requires MOAB job submission. You will not run any PHYLIP jobs in this manner during today’s tutorial, 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, and submit the procedure through MOAB.

An example of a script that could be used for a Fitch run is shown 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 you would 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. All responses to the usual interactive menu need to be in your input script file and you need to tell the program to use that script as input.
To run this sort of job in the background on many UNIX systems you would launch the name of the PHYLIP program that you want to use followed by appropriate file redirection and an ampersand, "&," e.g. "fitch < fitch_parameter_script.file > terminal &." However on FSU's HPC, whenever possible you need to use the MOAB job submission mechanism. MOAB is fully documented on the HPC Web pages, but I'll give you a brief example here. Therefore, using the same parameter script as above, a corresponding MOAB script could look like this:

```
#!/bin/bash
#MOAB -j oe
/opt/Bio/phylip/exe/fitch < fitch_parameter_script.file > terminal
```

And then the MOAB script is submitted to HPC with the msub command. This process will launch the job in the background when a queue is available, and you can go about whatever other computing tasks you would like to do. You can even log out of your current session and the job will run to completion. The programs still produce their standard outfile and outtree 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. Do not want submit any MOAB PHYLIP jobs in this tutorial, but the method may be helpful for you in the future. Please see the MOAB HPC Web pages for further details.

Look at the terminal file and MOAB log file created by the run after a PHYLIP background job stops. If it stopped very quickly, check to see that it got into the program and was loading sequences. If it wasn't doing that, go back and fix your MOAB script or PHYLIP parameter script. Once everything has been corrected, restart the job. It's hard to know how long a PHYLIP job might take. Some processes can run for days depending on the parameters selected, the size of the data set, and the user load on HPC. Experience will help. When the job finishes outfile and outtree will end up in the same directory where the job was started. Remember to rename them. 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.

**Plotting phylogenetic trees with PHYLIP**

Once you have generated PHYLIP Newick format tree files, you can create plots of the results. Two PHYLIP programs draw plots of Newick format tree file data, 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 to be used. The PHYLIP system has six fonts to work with. These are the so-called Hershey fonts; they have the file names font1 through font6. 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 this tutorial copy them all to your current directory with the following command:

```
$ cp /opt/Bio/phylip/exe/font* .
```

Since the package is meant 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 used. This makes changing the names on the generated plots very challenging, so the names should be changed, if desired, at earlier points in the work. If you haven’t gotten the names in your initial data matrix the way that you want them to end up, then you can either change them with a text editor in the Newick tree file or use PHYLIP’s ReTree program to do it interactively.

DrawTree and DrawGram operate in a similar fashion. They can typically be run in an X11 environment to preview the plot on the screen for visualization purposes and to create a PostScript graphics file of that plot.

### Publication quality PHYLIP PostScript graphics

PHYLIP can produce publication quality PostScript plots. PHYLIP’s internal PostScript driver does a very good job of sizing the tree to the page and producing a legible final graphic. 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 asks for the required input file and then for a font file. You then specify the final file format for the graphic and the preview screen. A number of options govern the plot itself. There are a whole slew of these. To begin, try the defaults, if they don’t give you what you want, repeat the process and select different ones. I suggest running them many, many times until you see what you prefer. To produce a PostScript graphic file of your tree, launch either DrawTree or DrawGram and accept the default “Postscript printer” as your “Final plotting device.”

Follow the sample screen trace below using the results from your first Fitch analysis to get a feel for running DrawGram. Specify the name of the input tree and then the Hershey font file name. However, because we’re using PostScript, we can use any non-bitmapped, PostScript supported, font once within the program, such as the PostScript default Times-Roman. Be sure that 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. 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. For this example, I will draw a default phenogram of the Fitch tree that I estimated:

```
$ drawgram
drawgram: can't find input tree file "intree"
Please enter a new file name> JTT.fitch.tree
DRAWGRAM from PHYLIP version 3.65
Reading tree ...
Tree has been read.
Loading the font ....
drawgram: can't find font file "fontfile"
Please enter a new file name> font2
```
Here are the settings:

0 Screen type (IBM PC, ANSI): ANSI
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
L Angle of labels: 90.0
R Scale of branch length: Automatically rescaled
D Depth/Breadth of tree: 0.53
T Stem-length/tree-depth: 0.05
C Character ht / tip space: 0.3333
A Ancestral nodes: Weighted
F Font: Times-Roman
M Horizontal margins: 1.65 cm
M Vertical margins: 2.16 cm
# Pages per tree: one page per tree

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

After pressing “y” for yes, a preview window is opened and the plot is drawn there. If it is acceptable, use the
“File” button to “Plot” the results, otherwise use the “Change Parameters” button and adjust things until you
like the plot. Once you are satisfied, and do press the “Plot” button, the preview screen goes away and the
PostScript plot is produced:

Writing plot file ...

Plot written to file "plotfile"

Done.

As always when running PHYLIP programs, immediately rename the output file:

$ mv plotfile JTT.fitch.tree.ps

The output PostScript file can now be sent to any printer that interprets PostScript. This may involve
transferring the file from the GCG server to a Mac or PC network for printing. If you have access to a system
printer, then you can use the lpr command to print the PostScript on that printer. However, it may require a
special print queue, so you may need to use some variation of the following type of lpr command (discuss this
with your system administrator):

$ lpr –PPostScript_que PHYLIP_plot_filename.ps

Another option is to transfer the file to another computer that can further process PostScript and embed them
in manuscripts or otherwise modify them. The PostScript graphic from this Fitch tree phenogram, as well as a
boot-strapped neighbor-joining tree eurogram, and a curvogram of the parsimony consensus tree, have been
embedded into this document in Appendix C and follow the tutorial’s Conclusion.
Finally I’ll 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; it redirects the output into the file combined.trees:

```bash
$ cat JTT.fitch.tree JTT.neighbor.boot.contree phylip.pars.contree > 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 outtree combined.consensetree
```

The final consensus tree will show how well the three methods — Fitch least-squares distance fit, bootstrapped distance neighbor-joining, and parsimony — all agree with each other. Their branch lengths will be proportional to the level of agreement between the three methods. This final tree can then have a PHYLIP PostScript graphic made of it with either DrawTree or DrawGram. A PostScript DrawTree graphic of this final consensus tree is also included in the Appendix after the Conclusion of the tutorial. Do realize that this tree is still not in agreement with commonly held opinion of the relationships of all these taxa — undoubtedly due to error introduced into the analyses by the use of overly simple protein models with this dataset. If this analysis really mattered, I would definitely repeat it with a corresponding DNA alignment. I would also definitely use maximum likelihood methods, if I were planning on publishing the results.

A handy thing to do with this type of tree is to impose actual branch lengths on the consensus topology. Take advantage of PHYLIP’s user tree option, available in many of the programs, to achieve this. 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.

PHYLIP also has a great tree manipulation tool called ReTree. This program enables you 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. However, you can use ReTree to change the appearance of your tree to more closely match your preconceived idea of what it should look like without actually changing the topology of the tree at all. I encourage you to play with ReTree, but we won’t be taking the time today.

That’s it for now. Log out of your current UNIX session on HPC and on the workstation that you are using.

**Homework assignment**

To begin, try to repeat those experiments that you performed today, that you thought were worthwhile with the protein data, with your DNA dataset. This will take some time, so get started early! The Lab Report Form will
cover the following topics: Which, of all the methods used, both today, and from your DNA dataset homework, seemed to give the most ‘satisfying’ answer? By this I mean, which method’s results seemed most consistent with your preconceived notion of what the phylogeny should be, based on your sequences’ descriptions. I don’t need any quantification, just your feeling. This includes all the GCG methods including PAUP* from within GCG, and all the PHYLIP methods too. What major problem seemed common to all of the inference methods using protein alignment data as a starting point in this tutorial? Did you see this with the DNA dataset as well? Describe this problem, why it occurs, where it comes from, and how to deal with it.

**Conclusion**

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, recombination, and/or horizontal gene transfer. Use several genes — the Ribosomal Database Project (RDP) ([http://rdp.cme.msu.edu/index.jsp](http://rdp.cme.msu.edu/index.jsp)) 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 recombination or 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 tutorial. Regardless, I cannot stress the importance of the quality of your multiple sequence alignment enough — everything is dependent on it!

Some analyses can take quite a while to run, especially if you use maximum likelihood methods, or Bayesian methods, which I have not discussed at all. However, I highly recommend using maximum likelihood and/or Bayesian methods, as they will produce the most accurate answers possible, as long as you use appropriate models and rate parameters.
Appendix A: The GCG Figure program

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 to be 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 being .5 platten units apart.

Instructions that may be useful to modify files for controlling text font, orientation, character size, and 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.
Appendix B: GCG graphics command switches

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 on all GCG graphics programs, however, on those where it is available, this function controls how many pages the graphic will be drawn on. By default graphics are calculated to best 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 merely be spread out onto more pages.
Appendix C: PHYLIP PostScript graphics of phylogenetic trees from my example dataset.

The JTT model, Fitch derived tree drawn by PHYLIP’s DrawGram phenogram option. Vertical distance is meaningless in this representation; horizontal distance is directly proportional to evolutionary divergence in the number of substituted amino acids per site.
The PHYLIP bootstrapped neighbor-joining tree (distances by JTT) euragram is shown next. 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.
A consensus tree of the three most parsimonious PHYLIP trees drawn as a curvogram follows. Again the vertical distance component is meaningless and the horizontal component reflects agreement between the three trees. Note the distinct differences between this tree and the two previous distance based methods’ trees. In many ways this tree is much closer to the ‘truth.’
The final consensus tree of the three PHYLIP methods used in the tutorial as presented by DrawTree is shown 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 agreement between two of them; and three of three, where all methods consistently gave the same answer. For instance, the fungi 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.
And to show you that maximum likelihood is possible, here’s the ProML tree. I used the Dayhoff rate matrix, with 0.07% invariant sites, and an alpha shape parameter of 0.6 (\(-\ln L = 14846\)). This was the best combination of parameters available in ProML as determined with ProtTest (Abascal, et al. 2005).
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 greatly indebted to Gary Olsen from the University of Illinois, Urbana Campus, whose lectures provided the gist and heart of this tutorial’s Introduction. Thank you Gary.