CHAPTER 12

Inferring phylogenetic relationships from sequence data

Peter G. Foster

1. INTRODUCTION

In molecular phylogenetics, we want to infer an underlying tree of relatedness from our gene sequence and its close relatives. We collect the same gene sequences, DNA, or protein from all of the taxa that we are interested in and align those sequences so that the columns of the alignment represent homologous sites of the gene, and then use that alignment to infer a phylogenetic tree. There are several different approaches to tree building. They have their strengths and weaknesses, and differ widely in the computational time required.

The most important tree-building methods are maximum parsimony (MP) and model-based methods, the latter including distance methods, maximum likelihood (ML), and Bayesian approaches. The distance methods are perhaps the simplest. In this approach, all possible pairwise evolutionary distances between sequences are calculated and these distances are then used to build trees that best explain those distances. The pairwise distances that this method use might simply be the percentage difference between sequences, but because superimposed mutations at the same site can mask the real extent of evolutionary divergence, we correct the observed distances with an explicit model of evolution. The other methods are character-based, and rather than first distilling the sequences to distances, the full information in the sequences is used in judging whether one candidate tree is better than another. The MP approach looks for the trees that can describe the inferred sequence changes over the tree in the smallest number of steps. ML and Bayesian approaches use models to find results that have highest probability. The methods have been tested on simulated data, where the true tree is known, and generally the ML and Bayesian methods perform best, but at a cost of increased computational complexity.

Most methods involve searching 'tree space' – which can be imagined as a landscape in which all possible trees are represented, with similar trees adjacent to each other, and in which the local height of the landscape represents the

Bioinformatics: ***Methods Express*** (Paul H. Dear, ed.)

goodness of the tree at that point. A comprehensive search of tree space is usually an impossible task because the number of trees is astronomical. For example, the number of unrooted trees with 100 taxa is about 1.7×10^{182}. It is not possible to evaluate every tree unless the number of taxa is less than a dozen or so, and so some sort of heuristic search must be used. It is here that compromises and shortcuts are made, and where the quality of the algorithms and the programming come into play. For example, if the tree space has multiple islands of good trees, then a 'greedy' search that only goes uphill and that starts near one of the suboptimal islands will tend to get stuck and never find the globally optimal tree. Better search strategies might use multiple starting points for searching, or have the ability to cross from one island to another, but would be more computationally expensive. Using any heuristic does not guarantee finding the best tree.

The result of our analysis is a phylogenetic tree. The tree is often drawn with parallel lines with the terminal taxa on the right (see *Fig. 1a, b*), which can perhaps lure us into thinking that the present-day taxa on the right evolved from the ancestor nodes on the left. However, in the methods that we generally use, there is no time-line information in the tree because the tree is unrooted, and we do not have the information to infer which taxa evolved from which ancestors. This idea is clearer in the tree shown in *Fig. 1*(*c*), where there is no obvious ancestor to which the eye is erroneously drawn. The usual way to root a tree is to use an 'outgroup' (see *Fig. 2*). In this strategy, we call the taxa that we are interested in the 'ingroup' and we choose several additional taxa that we know are outside of that ingroup to use as an outgroup. Although the analysis as a whole is unrooted, we can infer that the place where the outgroup attaches to the ingroup roots the ingroup. The outgroup should be phylogenetically close to the ingroup, because if it is too distantly related it can distort the analysis.

It has become common to do phylogenetic analysis on more than one gene, perhaps a combination of mitochondrial and nuclear genes, using both ribosomal and protein-coding genes. This trend has arisen not only because of the increased

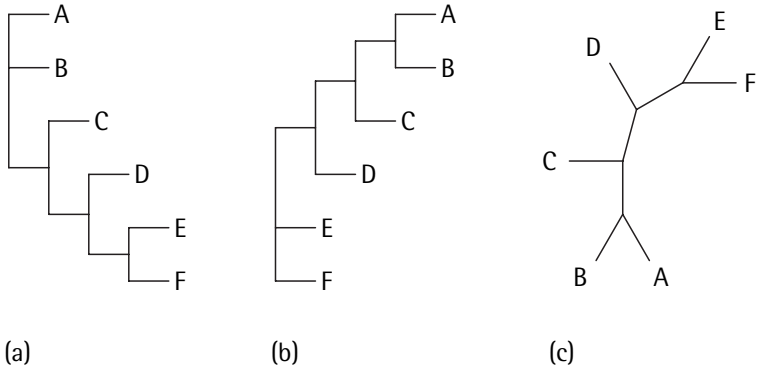

Figure 1. Unrooted trees.
The trees that we infer are generally unrooted and contain no information about ancestor–descendant relationships. The three trees in (*a*), (*b*), and (*c*) have identical topologies and are all unrooted. In particular, we should not infer from (*a*) or (*b*) that the ancestor of all of the taxa is the node on the far left.

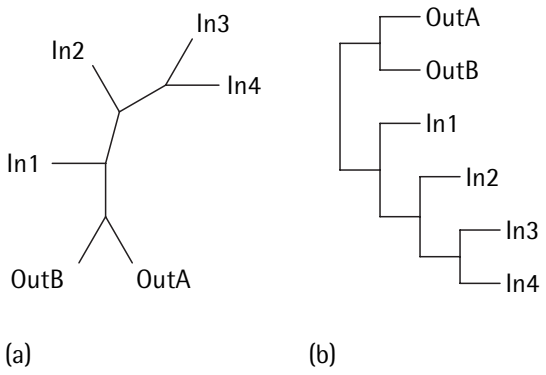

Figure 2. Outgroup rooting.
Here we use outgroup rooting, where taxa OutA and OutB are known to be outside the ingroup (taxa In1–In4) and so we can infer the root of the ingroup. As with most analyses, as a whole it is unrooted, as emphasized in (*a*). It is often drawn as in (*b*), where by convention a graphically introduced (but otherwise meaningless) bifurcating basal node is drawn to divide the ingroup from the outgroup.

availability of data, but also because it has been noted that the results of an analysis from one gene often differ from the results of another. Assuming that these differences are due to noise or different biases, it is the assumption and hope that in a combined analysis the effects of noise and bias will cancel out, but the true phylogenetic signal, perhaps weak in any one gene, will be the same in each gene and so will sum to give a well-supported tree. We might take any one of a few approaches in a multi-gene analysis. We might analyze each gene separately and then reconcile any differences between the respective trees afterwards; or we might simply concatenate the data and analyze it as if it were a single gene. However, the different genes may well have different evolutionary characteristics, and in a simultaneous analysis those differences can, in some software, be accommodated in a multi-partition model.

We want to find the best tree, but we also want to show how well supported our results are. Perhaps our single best tree is much better than any other tree, or perhaps, if our data are not decisive, then our best tree will not be significantly different from other similar trees. We may even find that we have many optimal trees, all equally good. Furthermore, some parts of the best tree might be more reliable than other parts. The usual way to determine these supports is with the bootstrap. In the bootstrap, we make pseudoreplicate datasets by resampling columns of the original alignment with replacement and then repeat the entire analysis on each. This process is repeated usually hundreds of times and the results summarized with a majority-rule consensus tree showing bootstrap support for the internal branches (see *Fig. 3*).

The methods should not be treated as black boxes that we throw our data into, from which a tree then emerges. Different methods have different assumptions that may or may not be met by your data. For example, in most common methods, there is an assumption that different sites in the alignment evolve independently. This may not be a good assumption, as for example in the

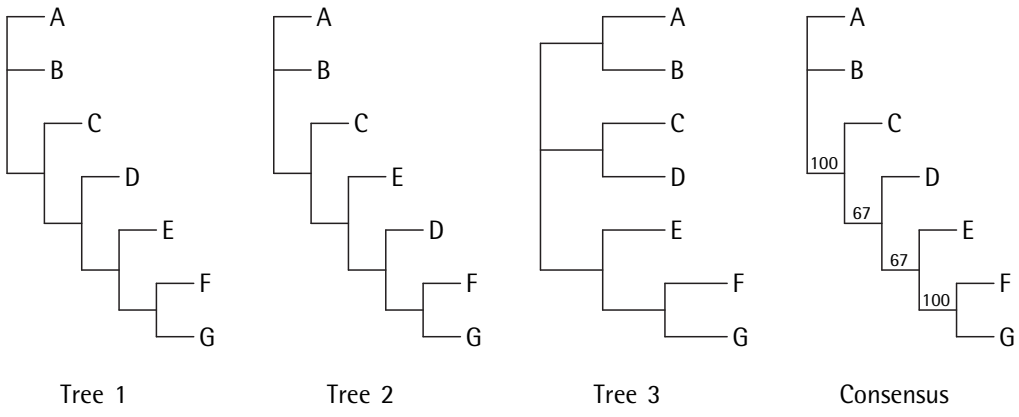

Figure 3. Consensus trees and support.
The input trees might be bootstrap replicates or samples from a Bayesian Markov chain Monte Carlo. Here we make a majority-rule consensus tree from three input trees. The consensus shows the proportion of input trees that have the given split or tree bipartition (internal branches). Some splits, for example the grouping of taxa C and D together, were seen in only one of the three input trees and so do not appear in the consensus. Some supports are higher than others, and so we have 100% support for the groupings of A with B, and of F with G, as these splits were found in all of the input trees. However, we would have less confidence in other parts of the tree.

stem regions of structural RNA genes. Again, most methods assume compositional homogeneity, meaning that the sequences all have the same composition. If this assumption is not met, then the methods can fail and the recovered tree can be distorted. The MP method, in particular, is susceptible to long-branch attraction. Here, highly diverged but unrelated taxa tend to group together erroneously in the inferred trees. This artifact arises from the failure of MP to deal correctly with sequences that have many superimposed, and therefore hidden, mutations. You might think that using several different approaches would be safest, but this may not be so; if you get the same result from several different methods, it may be because you are getting the correct answer to an easy problem with clear data, or it may be because all the results are distorted in the same way by an unknown bias in your data.

There are, of course, many aspects of phylogenetics that are beyond the scope of this short chapter and many complex areas are glossed over. In particular, I will not be discussing molecular clocks and molecular dating, nor the identification of sites in a gene that are under selective pressure, both of which are important emerging applications of phylogenetics. These methods are changing quickly, and any attempt to make a recipe would immediately be out of date in these rapidly moving and often controversial fields (1). To get to grips with these, you need to dive into the primary literature in journals such as *Molecular Biology and Evolution* and *Systematic Biology*.

The field of phylogenetics has matured to a point that there is now a comprehensive textbook by Joe Felsenstein (2) (see also http://evolution.genetics.washington.edu/[12.1]). The classic chapter on phylogenetic inference by Dave Swofford and colleagues has not gone out of date in the decade since its writing

(3). The reviews written or co-written by Paul Lewis are especially readable (4, 5). The course material for the annual Workshop on Molecular Evolution is written by experts in the field and is a very useful resource, available online (http://workshop.molecularevolution.org/ [12.2]).

2. METHODS AND APPROACHES

2.1 Alignments

We start our phylogenetic analysis with an alignment of molecular sequences. We want to compare homologous positions in the genes, and the alignment is our statement of homology. Homologous sites in genes are kept in line in spite of insertions and deletions in the gene sequences by introducing alignment gaps, usually with the '–' character. As we do not know whether the gaps are due to deletions in one sequence or insertions in another, they are referred to as indels. You can identify a group of related sequences from public databases using BLAST. For some closely related genes, alignment is trivially easy and can be done by hand, often with few or no introduced gaps. However, for more diverged sequences, you will need to use multiple sequence alignment software. The standard program for making alignments is CLUSTALW (6), which has several web servers including one at the European Bioinformatics Institute (http://www.ebi.ac.uk/clustalw/ [12.3]). Another recommendable alignment program is MUSCLE, which also has a web server (http://www.drive5.com/muscle/ [12.4]). Making multiple alignments is a difficult computational problem and often a CLUSTALW alignment can be, and should be, improved by eye. This should be no surprise, as the human brain is very good at recognizing patterns. Use a graphic user interface multiple sequence editor with colored characters to help you to fix up the alignment. Often, it is not obvious what gap pattern to impose on the sequences to get the best alignment, and if that is the case those parts that you are not sure of should be masked out and not used in subsequent analyses. An attempt at automating the process of identifying unreliable parts of an alignment has been made (7). A comment should be made here concerning the alignment of DNA sequences for protein-coding genes: with the assumption that homology is determined by the protein sequence, you should translate the DNA into its protein sequence and align the protein first, and then back-align the DNA to the protein so that the gap pattern between codons is preserved.

2.2 File formats

There are many file formats for sequence data (see the section on formats at http://workshop.molecularevolution.org/ [12.5]), and most phylogenetic software is limited in the formats that it is able to read. In many areas of bioinformatics, the simple FASTA format is used, and it is commonly used as the input format for multiple-alignment programs. It is used occasionally in phylogenetics, even though

it does not imply aligned sequences. A particularly common format for aligned sequence data is PHYLIP format, which might be either sequential (one sequence after another, like FASTA) or, more usually, interleaved, showing the alignment more clearly. Another important format is Nexus format, which is used by PAUP*, MRBAYES, and a few other programs (8). This extremely flexible format is able to accommodate information in addition to the sequence data, such as information about partitioning the data, about trees, and even commands for the programs that use this format. There is software that can convert formats, including READSEQ (http://iubio.bio.indiana.edu/soft/molbio/readseq/java/ [12.6]), with a web server (http://iubio.bio.indiana.edu/cgi-bin/readseq.cgi [12.7]). Trees can also be described in text files, almost always in Newick format (see http://iubio.bio.indiana.edu/cgi-bin/readseq.cgi [12.7]). Nexus format trees are Newick trees embedded in a Nexus structure. Newick format shows the tree topology using nested parentheses, and can incorporate branch lengths and labeling of interior nodes (often labeled with support values), as well as terminal taxa. Different programs more or less adhere to file format standards, but it is often the case that incompatibilities arise that force the user to make adjustments. If you are lucky, this can be automated or scripted, but often it cannot be changed easily and the data must be edited by hand. Format definitions are often modified and extended by software authors, which is an invitation to incompatibility when going from one program to another. The practicing phylogeneticist needs to be aware of this and to be able to debug problems.

In the case of a multi-gene analysis, if the data are simply concatenated and not kept separate, then there is no format problem. However, if we want to keep the different data partitions separate, perhaps to allow different overall rates in the different partitions, or if different model characteristics are to be applied to the different data partitions, then some way of indicating how the data are divided up into those partitions is needed. In Nexus files, there is a standard way of partitioning the data and that way is used by PAUP*. Other programs like MRBAYES and TREEFINDER that can do multi-partition data use their own ways of partitioning that are unfortunately not compatible with other programs.

2.3 Software

Joe Felsenstein maintains a valuable web page of all of the phylogenetic software that he knows about (http://evolution.genetics.washington.edu/phylip/software.html [12.8]), and many of the programs listed have web servers. The classic phylogenetic software includes the PHYLIP suite of programs by Joe Felsenstein, and PAUP*, written by Dave Swofford. Both are enormously capable and have been improved and debugged through wide use, and both are very well documented. PHYLIP includes a great many methods, but it can be slow and often other programs are able to do similar analyses faster. Many other programs use the PHYLIP data format and imitate the PHYLIP command-line menu-driven interface. PAUP* (http://paup.csit.fsu.edu/ [12.9]) is one of the few mainstream programs in phylogenetics that is not free – the source code is not available and there is a small license fee.

Nevertheless, it is worthwhile having, as it is excellent for parsimony, distance methods, and for ML of DNA.

PAUP* is in wide use for ML analysis of DNA. It is excellent for this, but is slow compared with newer ML programs such as PHYML (http://atgc.lirmm.fr/phyml [12.10]) and TREEFINDER (http://www.treefinder.de/ [12.11]). TREE-PUZZLE (previously called just PUZZLE; http://www.tree-puzzle.de/ [12.12]) is an ML program with good models for both DNA and protein that uses quartet puzzling to find ML trees. In this strategy, the problem is decomposed into finding the best trees from many different quartets of taxa sampled from the data and then combining (puzzling) the quartets to make the full tree. Although it is faster than PAUP* for ML, the resulting trees are generally not as good as more-thorough tree-based methods, and with the advent of newer ML programs, its niche is smaller.

Choices for analyzing multiple data sets simultaneously are limited. The major Bayesian program in wide use, MRBAYES, has a comprehensive set of models and is one of the few programs that is able to model multiple data partitions well. PAUP* is able to model separate DNA partitions with its site-specific model, by allowing the different data partitions to have their own overall rates; however, PAUP* otherwise assumes that the other model characteristics, rate matrix, and so on, are the same in all of the partitions. Modeling of different data partitions in MRBAYES is much more flexible, allowing combinations of datatypes and fitting of different complete models to the different partitions. TREEFINDER is also able to model different data partitions well under ML.

As larger analyses can be very computationally intensive, it would sometimes be appropriate to be able to run the analyses in parallel on clusters of computers. MRBAYES and TREE-PUZZLE have parallel versions. A bootstrap analysis is naturally amenable to running on a cluster, where different machines are each given a part of the task. Another up-and-coming strategy is cycle scavenging, where phylogenetic jobs flexibly migrate among idle desktop computers (http://www.cs.may.ie/distributed/multiphyl.php [12.13]).

The end result of a phylogenetic analysis is usually a tree with support values on the nodes, and often we will want to draw that tree. We will want to show the branch lengths accurately, perhaps with a scale bar, and we will want to show the support values on the nodes. We will want to use vector rather than bitmap graphics, and we will probably want the output of the tree-drawing program to be in a form that will allow further editing with your favorite graphics program. Unfortunately, software to draw trees can be frustrating and any given program may only satisfy some of these desiderata. Perhaps the best combination of ease of use and capabilities is TREEVIEW (see Rod Page's TREEVIEW page at http://taxonomy.zoology.gla.ac.uk/rod/treeview.html [12.14]). The PHYLIP programs DRAWTREE and DRAWGRAM can also be useful.

2.4 Tree-building methods

The main tree-building methods are MP and model-based methods. Parsimony is fast and able to handle many sequences and is excellent if divergences are small.

Indeed, if there is very little divergence, then attempting to fit a model will be difficult through lack of variation in the data. A search might find one MP tree, but often there are many such MP trees, each with the same minimum number of sequence changes over each tree; this might be considered an indication of the amount of ambiguity in the particular analysis. We usually will want to do a bootstrap analysis, in which case the result would be a consensus of the MP trees from each bootstrap replicate. A given bootstrap replicate might also have more than one MP tree, and these are weighted by the inverse of the number of these trees when the consensus is made.

Protocol 1

MP search using PAUP*

1. Save your data in Nexus format. (An example dataset is given in the Protocol_1 folder for this chapter on the book's web site, as data.nex[12.15].)
2. Make and save a text file containing the following:

```
#nexus

begin paup; [ Comments inside square brackets ]

[ Keep a log file. I assume the analysis will need to be repeated,
  so I set replace=yes to overwrite the file ]

log file=log replace=yes;

set maxtrees=1000 increase=auto; [ appropriate for parsimony]

execute data.nex; [ read in your data ]

[ Do a parsimony bootstrap, with default settings, except that 1000
  bootstrap replicates are used. ]

bootstrap nreps=1000;

[ The following line saves the resulting consensus tree to a nexus
  tree file. Using the from/to should not be needed, but it is a
  workaround to a known bug. Bootstrap support values are saved
  with the tree. If you repeat the analysis, the file is over-written. ]

savetrees from=1 to=1 file=bootTree.nex savebootp=nodelabels

maxdecimals=1 replace=yes;

log stop;

quit;

end;
```

   (A copy of this text file is given in the Protocol_1 folder for this chapter on the book's web site, as commands.nex[12.16].)
3. Execute the file. On Mac Classic or Windows versions of PAUP*, you start the program and then use the execute command. On Unix-like machines, including Mac OS X and Linux, you can run it by saying, to your command line:

```
paup commands.nex
```

4. The example data file contains only 12 taxa and will run quickly. An alignment of 100 taxa might take a few hours to run.
5. At the end of the analysis, you will have a tree in a Nexus file, where internal nodes are decorated with bootstrap support values. The program will also make a log file as requested. Copies of the expected files (bootTree.nex[12.17] and log[12.18]) are in the Protocol_1 folder.
6. You can then import the result file into a tree-drawing program. If you want to import it into TREEVIEW X (http://taxonomy.zoology.gla.ac.uk/rod/treeview.html [12.14]), the file needs to be modified slightly – the zero just before the final semi-colon in the tree description needs to be removed. (A worked example is in the Protocol_1 folder: Treeview/bootTreeM.nex[12.19].)
7. In this analysis, branch lengths are meaningless and so a 'cladogram' format, where all of the leaf nodes line up on the right, is appropriate. TREEVIEW X can place the bootstrap support values on the internal nodes. You can print it to a file for further manipulation. (A copy of the output is in the Protocol_1 folder: Treeview/bootTreeM.pdf[12.20].)

Model-based methods tend to do better than MP methods when divergences are large. Models take into account the possibility of superimposed, and therefore hidden, mutations when calculating evolutionary distances, and branch lengths are more meaningfully expressed in terms of mutations per site. Model-based methods include distance methods, ML, and Bayesian methods. Of these, the distance-based methods do not do as well as the full tree-based ML and Bayesian methods, but distance methods are fast and so might be suitable for bootstrap analyses of many taxa. (However, newer ML implementations are getting faster (9, 10) and can now handle many taxa, so the reason for using distance methods is somewhat less compelling.) A distance analysis is a two-step process. First, a matrix of pairwise distances is made and then these distances are somehow made into a tree. The pairwise distances might be a very simple measure such as the number of changes between the sequences; these are uncorrected or p-distances and so are not really model based. Better distances can be obtained with models that take into account superimposed mutations. Maximum-likelihood distances can also be made, which would be recommended. A tree can then be made from the distance matrix using neighbor joining (11) or its relatives WEIGHBOR and BIONJ (12, 13). Neighbor joining is an algorithm and does not involve tree searching, so it is very fast. Another alternative to treeing distance matrices is to search tree space using something like the minimum tree length as the function for choosing the best tree.

As tested with simulated data, the best-quality tree-building methods are the full tree-based ML and Bayesian approaches, which both use the likelihood function. The likelihood is proportional to the probability of the data given the model and the tree. The likelihood of a tree is usually a very small number and so is expressed in its log form (and should not be confused with the probability of a tree). When we talk of the likelihood of a tree, we usually mean the maximized likelihood, where all of the nuisance parameters are optimized, maximizing the likelihood for that particular tree. The nuisance parameters are things like the branch lengths and model rate matrix parameters. The ML tree is the tree, found in a search, that has the highest maximized likelihood. As maximizing the likelihood

uses numerical optimization and so is computationally intensive, searching for the ML tree can be slow. Because of this, phylogenetic programmers have, with some success, been pushing for increased speed via improved algorithms and workable compromises.

2.5 Choosing a model

Choosing a model of evolution is important, as it can greatly affect your analysis. You should not guess or assume a model, but rather choose the best available model, the one that best fits your data. Generally, we use ML to choose a model; the strategy is to evaluate the same tree with many different models and then to choose the best one. However, we do not simply choose the model that gives the highest likelihood. The process is analogous to fitting a polynomial curve to a scatter plot; we want our model to describe the important trends in the data, but without overfitting. Models differ in the number of parameters, and generally the more parameters there are in the model, then the better the fit of the model to the data and the higher the likelihood. On the other hand, with more and more parameters, the likelihood will rise, but we will see diminishing returns, and simple data described by an overly complex model might be modeling noise in the data – it would be overparameterized. We want to avoid that, and so when we evaluate our available models, we choose the one that gives the highest likelihood, but penalize by the number of parameters. This is formalized in the Akaike information criterion (AIC) (14). The AIC of a model is defined as $-2\log L + 2n$, where n is the number of free parameters in the model. We make a table of AIC values and the best choice of model is the one with the lowest AIC value.

Models for a single data partition can be described in terms of (see below):

- The rate matrix (e.g. GTR or HKY for DNA, JTT or WAG for protein)
- The composition (e.g. empirical or ML, +F in protein)
- The ASRV (none, +G, +I, +IG)

Models differ from each other in their free (i.e. adjustable) parameters. The parameters might be optimized by ML, or in a Bayesian analysis they can be free to be varied, or sometimes they can be fixed to reasonable numerical values. Models with variable numerical parameters can therefore be considered to be families of models. The rate matrix describes the rates of change between character states (for example, between any two of the four bases, A, C, G, and T). The most general rate matrix for DNA models in wide use is the GTR (general time reversible) matrix, and matrices for other models can be considered to be simplifications of it. Superimposed on the rate matrix is the composition of the model. Generally, we use the observed average composition of the data, i.e. the empirical composition, or we optimize the composition by ML. If the data are fairly homogeneous, the empirical and ML compositions will be about the same. However, sometimes whether you use empirical or ML compositions will make a difference to the analysis. If you have the computational time for it, it is recommended that ML-optimized compositions be used. On the other hand, optimization of parameters

like this is computationally expensive and one reasonable way to save time is to use fixed empirical compositions. The third aspect of model descriptions is among-site rate variation (ASRV). This is an accommodation of the effects of selection on different sites and so would be especially important with biological sequences. It should definitely be taken into account (15). ASRV is usually modeled by using discrete Γ (gamma)-distributed among-site rates (+G), or a proportion of invariant sites (+I), or both together (+IG). The former strategy, using Γ-distributed rates, allows different sites to have different rates, where the relative rates can be described by the shape parameter of the Γ-distribution family. There is no biological basis for assuming that the ASRV is Γ distributed; it is simply a computational convenience to allow a wide range of different distributions all described by a single parameter. The other major strategy to accommodate ASRV is to allow a proportion of sites to be invariant (i.e. constant sites that are not allowed to be varied, in contrast to constant sites that have the potential to vary but have not yet done so). Such a proportion can be estimated by ML. There is some overlap (nonidentifiability) in the two strategies. For example, a Γ distribution with a low α value has an L-shaped distribution of site rates, implying many very slow sites. This would be describing approximately the same thing as a model with a sizable proportion of invariant sites, and so your estimate of one will affect your estimate of the other.

In protein models, we generally do not optimize the rate matrix parameters (as we would do with the DNA GTR model). Rather, we use one of the 'off-the-shelf' empirical models such as the JTT or WAG models. These have been made with large data sets and are meant to be generally applicable. It is assumed that your small alignment will behave in approximately the same way as the large data set. This is a good strategy because your small data set probably does not have enough information for an accurate estimation of all 189 rate matrix parameters. These large datasets have their own inherent composition, which may well differ from the composition of your data. Imposing your empirical composition on the chosen rate matrix is often called the +F model and is certainly recommended if available.

The strategy for choosing a model by ML is to optimize the same tree with different combinations of rate matrix, composition, and ASRV. The tree used for this purpose need not be the optimal tree; any reasonably good tree, such as a neighbor-joining tree, can be used. After evaluating the tree with different models, a table is made with the AIC (or its variants AICc or AIC2, or the Bayesian information criterion) and used to choose the best model. This can be done by hand, but it is tedious and so has been automated. The original automated model choice program is MODELTEST (16), which tests DNA models available in PAUP*, using PAUP* to do the likelihood calculations. The theory and practice of model selection can be found in the documentation accompanying MODELTEST. A version for protein models, PROTTEST (17), uses its own modified version of PHYML to do the likelihood calculations. A version of PROTTEST is available as a web server (http://darwin.uvigo.es/ [12.21]). A similar program, MODELGENERATOR (http://bioinf.may.ie/software/modelgenerator/ [12.22]), for both DNA and protein models, does not require external software for the likelihood calculations. Choosing a model tells you what model family to use (e.g.

HKY+G, or GTR+IG), but it also may tell you the optimized numerical parameters that were obtained. Whether you should use those numerical parameters depends on the context. Generally, for an ML search or a Bayesian analysis, the parameters should all be free (although we make an exception for empirical protein rate matrix parameters) and so we would not use those optimized parameter values. (Using PAUP* for ML searching is a special case, where we would use a successive approximation strategy for the best searches; 18.)

In a multi-gene analysis, the genes might be quite different in their evolutionary dynamics, in which case they should be modeled separately. If they are different data types, then they must be modeled separately. One strategy that can be used is to analyze the different genes completely separately. For this, you could use any ML program. You would need to evaluate (without searching) all of the possible candidate trees with all of the genes, and the ML tree is the tree that gives the highest sum. In this strategy, the analyses are completely separate and the branch lengths are unrelated. This has the disadvantage that there is a huge increase in the number of parameters because of all of the free branch lengths. Another reasonable strategy is to analyze the data together, but in separate data partitions, modeled separately. Available software for this is more limited. The partitions would be able to have overall rates, and so we would have fast genes and slow genes. These rates could be considered branch length multipliers, forcing the corresponding branch lengths in the different partitions to be proportional to each other. This has far fewer parameters.

Protocol 2

Automated model choice

The program MODELGENERATOR will be used here because it is applicable to both DNA and protein, and it does not require additional software to do the likelihood calculations (19). MODELGENERATOR will recognize whether the data are DNA or protein and test a full complement of rate matrices and ASRV. It constructs a neighbor-joining tree to make model comparisons. For protein models, both the inherent model composition and your empirical data composition (the +F model) are tested. For DNA models, the composition is estimated by ML, where applicable. The models can be compared and chosen by the AIC (or its variant AICc, used for short data; 20).

1. Get the software from its website (http://bioinf.may.ie/software/modelgenerator/[12.22]) and unpack it.
2. Install the program on Mac OS X or Linux by making the following text file:

   ```
   java -jar /path/to/your/modelgenerator.jar $1 $2
   ```

   Name the file modelgenerator[12.23], make it executable, and put it in your path.
3. Put the data in FASTA or PHYLIP format (data for a worked example is given in the Protocol_2 folder for this chapter on the book's web site, named data.phy[12.24]).
4. Run the following command:

   ```
   modelgenerator data.phy 4
   ```

Here the '4' is the number of discrete Γ-distributed among-site rate categories; four is generally sufficient. The results with the suggested model are saved in a log file.

5. The output for the worked example is given in the Protocol_2 folder as modelgenerator0.out[12.25]; a README[12.26] file is also provided.

Protocol 3

ML with PHYML

PHYML (9) is a fast implementation of ML with a good set of models for both DNA and protein. It is fast enough that bootstrapping can be done in a reasonable amount of time. A local installation can be run using command line options (for a complete list, use the command '`phyml -h`'), but the PHYLIP-like interface allows more options. PHYML also has a web interface, which we will use in this protocol.

1. Choose a model for your data (see *Protocol 2*).
2. Put your data into PHYLIP format. A worked example is given in the Protocol_3 folder for this chapter on the book's web site as data.phy[12.24]; note that this file is the same as the one used in *Protocol 2*).
3. Call up the web page (http://atgc.lirmm.fr/phyml[12.10]) and, in the upper-right part of the screen, change the setting for the input file from **example file** to **file** and use the **choose file** button[a] to upload the data.phy[12.24] data file. Also check the **Perform bootstrap** box, and set the number of bootstrap replicates ('Number of data sets') required (200 for the worked example; generally 100 at a minimum and usually more if time allows).
4. The model suggested by the AIC in *Protocol 2* for these data was the GTR+G model, so choose the **GTR** substitution model. Make the proportion of invariant sites **fixed** and set to zero, but turn on Γ-distributed ASRV by setting the number of substitution rate categories to 4, leaving the Γ-distribution parameter in its default setting of **estimated**. Leave the other settings at their default values, fill in your e-mail address, and submit the job.
5. When the job finishes, the results are e-mailed to you. A copy of the expected results is given in the Protocol_3 folder as phyml_online_results[12.27].
6. The final line of the results gives the tree structure with its bootstrap values. To view the tree in TREEVIEW X, put this line into a new text file (the worked example file in the Protocol_3 folder is Treeview/tree.phy[12.28]).
7. Use this as an input into TREEVIEW X to create a graphical file (the expected result is given in the Protocol_3 folder as Treeview/tree.pdf[12.29]). Note that bootstrap values are given as numbers out of 200 (the number of bootstraps used in this example – see step 3 above), rather than as percentages.

Note

[a]Depending on which web browser you are using, this button may instead be called **Browse**.

2.6 A Bayesian approach to phylogenetics

The past decade has seen the emergence of Bayesian methods into the repertoire of the phylogeneticist (4, 21). Bayesian methods are also based on the likelihood of trees, but the likelihood is not optimized as in ML. Rather the result is expressed in terms of the posterior probability distribution of the trees or splits, and takes into account the variance of all of the nuisance parameters. The posterior probability distribution cannot be calculated directly, but it can be approximated by means of the Markov chain Monte Carlo (MCMC) method. This is a computational process that samples tree topologies, branch lengths, and model parameters in proportion to their posterior probability; those trees with a higher posterior probability are sampled more often. It is a chain that goes from state to state, typically for many thousands of state changes or generations. At the end, when we have collected enough samples, we need to summarize those samples for our result, and generally that summary will be the consensus tree topology from the sampled trees. Typically, we start our MCMC from a random tree topology, with arbitrary model parameters and, after a while, the chain converges to the posterior probability distribution. Because of this delay in reaching convergence, samples taken from the beginning of the chain are not representative of the posterior and are discarded as 'burn-in'. Assessing whether the chain has converged is difficult, and the investigator needs to pay careful attention to the available diagnostics. The posterior distribution is proportional to the likelihood of trees with their models, but it is also proportional to prior probabilities, which is what you think the results should be before you see the data. Generally, the prior is chosen so that it does not influence the result much and so, for example, the prior probability of any one tree topology is the same as any other. Generally, when there is a sizable amount of data, the influence of the prior will be small and the posterior will mostly be driven by the likelihood, and so ML and Bayesian analyses will be quite similar.

A valid MCMC involves many practical considerations. We want the MCMC to show good mixing, i.e. we want our samples from the MCMC to cover the whole of the posterior distribution, and there are strategies to promote that. One simple strategy is to sample the chain rarely, for example every 1000 generations, rather than every generation. Another strategy uses Metropolis-coupled MCMC, or MCMCMC; the interested reader is invited to consult the MRBAYES manual for an explanation (Huelsenbeck and Ronquist's MRBAYES web page at http://mrbayes.csit.fsu.edu/ [12.30]). We also want some assurance that our MCMC has reached convergence. We cannot have absolute assurance, but there are diagnostics that we can use. A reasonable assumption is that if we run the analysis more than once, with each run starting from a different random starting tree, and if we find that each run converges to the same consensus topology, then we can have more confidence that they have converged. Doing more than one run is highly recommended and two runs is now the default in MRBAYES. Perhaps the simplest diagnostic is to plot the likelihood of a run and check that it has reached the more or less noisy plateau that is consistent with convergence. This is a widely used quick-and-dirty method, but it is not reliable. Certainly, if the likelihood

plot has not reached a plateau, then we can say that the chain has not reached convergence, but we cannot argue the reverse. Other parameters besides the likelihood can be examined, and the '`sump`' command in MRBAYES aids this. Generally, we are especially interested in the topology, and we want to make sure that the topology has converged. To aid this, MRBAYES now examines how similar the sampled trees are from different simultaneous runs in the form of the average standard deviation of split frequencies. When this reaches a low level, such as 0.01, then the two runs are similar to each other and we can have more confidence that the chains have converged. Whilst many investigators use a fairly short burn-in and so get a few more samples, it is recommended that a long burn-in be used to give a better chance of sampling truly converged chains.

Protocol 4

Simple Bayesian analysis with MRBAYES

MRBAYES has a comprehensive wiki manual and tutorial introduction on its web site (http://mrbayes.csit.fsu.edu/[12.30]) and this should be consulted. There is also extensive online help within the program itself and there are example files that come with the software. It is recommended that more than one run be made, and doing two simultaneous runs is the default in MRBAYES. Instructions for doing multi-partition analyses are given in the wiki manual.

1. Install MRBAYES from its web site (http://mrbayes.csit.fsu.edu/[12.30]). MRBAYES is generally installed under the name '`mb`[12.31]', and you can start the program for interactive use by typing '`mb`' at your prompt.
2. Choose a model for your data[a].
3. Put your data into Nexus format, using a Nexus data block (8). (Data for this worked example is given in the Protocol_4 folder for this chapter, as data.nex[12.15].)
4. Run the analysis for a short time, perhaps 10 000 generations, interactively (as in the tutorial). That way you can get some idea of how long a longer analysis will take. When running interactively, you can continue running more generations after the first lot has finished.
5. When you have an idea of how long to run the MCMC, you can collect your commands in a Nexus file. Such a commands file might contain:

```
#nexus
begin mrbayes;
log filename=mbout.log append start;
execute data.nex;
lset nst=6 rates=invgamma;
mcmc ngen=1000000 samplefreq=500 printfreq=10000 filename=mbout;
sump filename=mbout burnin=1001;
sumt filename=mbout burnin=1001;
log stop;
quit;
end;
```

(A copy of a similar worked example is given in the Protocol_4 folder as commands.nex[12.16].)

The '`lset`' line, which specifies the model, should specify your chosen model. Here, I have specified a million generations, which might take overnight for a medium-sized data

set. I collect samples every 500 generations ('`samplefreq`'), which means that I collect 2001 samples for each of the two runs. The burn-in is given by the number of samples, not the number of generations (in this case, 1001 samples, or half of the run)[b]. The '`sump`' command summarizes the likelihood and parameter values, and the '`sumt`' command summarizes the sampled trees and makes a consensus tree; these commands also specify the names for the respective output files.

6. Run the command file from the command line as '`mb commands.nex`'.
7. At the end of the run, examine the output to determine whether topological convergence has been reached, and whether you have convergence of the model parameters, good chain swapping, and good proposal acceptance rates. (The output files for this worked example are given in the Protocol_4 folder as mbout.*[12.32], along with a README[12.26] file describing their interpretation.) If everything went well (as in this worked example), then the analysis is finished, but, if not, then you need to make adjustments and run it again, perhaps for a longer run.
8. Bayesian analysis results are very straightforward to interpret. The split supports, given in the consensus tree, are the posterior probability that the split is true, given the data and model.

Notes

[a] MRBAYES does not implement the many subsets of the GTR model such as the TN and K3P models, and so if those models are chosen, you can use the GTR model. In this case, it is probably better to err on the side of slightly overparameterizing rather than underparameterizing.

[b] A long burn-in is recommended. Some diagnostics, such as a likelihood plot, might reach apparent convergence quickly, but other aspects, such as topology, might be slower to converge. You can use AWTY ('Are we there yet?') for more convergence diagnostics (http://king2.csit.fsu.edu/CEBProjects/awty/awty_start.php [12.33]).

3. TROUBLESHOOTING

The process of evolution has been complex, and the present-day gene sequences that we use might easily contain biases or other aspects that can confound our phylogenetic methods; a few will be mentioned here. For example, if there is compositional heterogeneity in the sequences, and there are unrelated sequences that have a similar composition, those sequences might erroneously be attracted to each other in the recovered tree. When we model ASRV, one assumption is that the rate of individual sites is constant over the tree. However, there is evidence that the rate of sites can change in diverged taxa, and even the spectrum of invariant sites can change over time. These covarion and heterotachy effects are difficult to model. In early evolution, horizontal (or lateral) gene transfer appears to have been more common and of course that will cause gene trees to be different from organism trees. Perhaps such effects might better be described by networks rather than trees. Long-branch attraction in parsimony has long been appreciated, and model-based methods are less susceptible to this. However, we can also see long-branch effects in model-based methods when the model does not fit well. Such problems are not easily identified. At the least, you might want to identify very long branches or compositionally divergent taxa, and as an

experiment remove them for a repeat of your analysis to see whether it affects your conclusions. A similar problem with difficult data occurs when the outgroup is highly diverged from the ingroup to the extent that its presence distorts the ingroup. As a workaround, you can remove the outgroup to get a better supported but unrooted ingroup topology, and then as a separate analysis attempt to see where the outgroup attaches.

Phylogenetic software can be frustrating. It is often written by academics whose main expertise is not programming, and it is often user-hostile and buggy. Something as simple as differences in line endings in files (Unix versus old Mac versus DOS line endings) might cause a program to fail. These otherwise invisible characters can be seen with the '`od -c`' command in Unix. Perhaps you have long names for taxa and you intend to do an analysis with a program that only takes short names. It might be worthwhile changing the names to arbitrary short names to do the analysis and then changing the names back to the long versions at the end. A script using a dictionary or hash in Perl or Python can facilitate that. You can save time debugging methods if you use small datasets. Debugging is often helped by example files provided with the program, and trial and error debugging is made easier when you use very small files.

Bayesian analysis requires more expertise than other methods in order to adjust the MCMC and to assess convergence. It also has its own problems (see, for example, (39)). You will often find that, when the same data are analyzed by both ML and Bayesian approaches, posteriors are higher than bootstraps; these inflated posteriors should be kept in mind when interpreting results. You should be especially suspicious of high support on short internal branches.

4. REFERENCES

1. **Graur D & Martin W** (2004) *Trends Genet.* **20**, 80–86.
★ 2. **FelsensteinJ** (2004) *Inferring Phylogenies.* Sinauer Associates, Sunderland, MA. – *Standard comprehensive textbook.*
3. **Swofford DL, Olson GJ, Waddell PJ & Hillis DM (**1996) In *Molecular systematics.* Edited by DM Hillis, G Moritz & BK Mable. Sinauer Associates, Sunderland, MA, pp. 407–514.
★ 4. **HolderM & Lewis PO** (2003) *Nat. Rev. Genet.* **4**, 275–284. – *A very readable explanation of Bayesian methods and the MCMC.*
★ 5. **Lewis PO** (2001) *Trends Ecol. Evol.* **16**, 30–37. – *A paper describing some of the new directions in phylogenetics.*
6. **Chenna R, Sugawara H, Koike T, *et al.*** (2003) *Nucleic Acids Res.* **31**, 3497–3500.
7. **Castresana J** (2000) *Mol. Biol. Evol.* **17**, 540–552.
8. **Maddison DR, Swofford DL & Maddison WP** (1997) *Syst. Biol.* **46**, 590–621.
9. **Guindon S & Gascuel O** (2003) *Syst. Biol.* **52**, 696–704.
10. **Jobb G, von Haeseler A & Strimmer K** (2004) *BMC Evol. Biol.* **4**, 18.
11. **Saitou N & Nei M** (1987) *Mol. Biol. Evol.* **4**, 406–425.
12. **Bruno WJ, Socci ND & Halpern AL** (2000) *Mol. Biol. Evol.* **17**, 189–197.
13. **Gascuel O** (1997) *Mol. Biol. Evol.* **14**, 685–695.
14. **Akaike H** (1974) *IEEE Trans. Autom. Contr.* **19**, 716–723.
15. **Sullivan J & Swofford DL** (1997) *J. Mamm. Evol.* **4**, 77–86.
16. **Posada D & Crandall KA** (1998) *Bioinformatics*, **14**, 817–818.
17. **Abascal F, Zardoya R & Posada D** (2005) *Bioinformatics*, **21**, 2104–2105.
18. **Sullivan J, Abdo Z, Joyce P & Swofford DL** (2005) *Mol. Biol. Evol.* **22**, 1386–1392.

19. **Keane TM, Creevey CJ, Pentony MM, Naughton TJ & McInerney JO** (2006) *BMC Evol. Biol.* **6**, 29.
20. **Posada D & Buckley TR** (2004) *Syst. Biol.* **53**, 793–808.
21. **Huelsenbeck JP, Ronquist F, Nielsen R & Bollback JP** (2001) *Science*, **294**, 2310–2314.