Maximum Likelihood and Bayesian Analysis in Molecular Phylogenetics

Peter G. Foster
Natural History Museum, London

Rio de Janeiro, March 2010

1 Modelling sequence change over time

How have gene and protein sequences evolved over time? Of the many forms that mutations can take, here we will focus on nucleotide or amino acid replacements only, and not deal with such things as insertions, deletions, and rearrangements. We will try to model those replacements, that is to attempt to describe the important aspects of the underlying process of evolution that might have generated the sequences that we see.

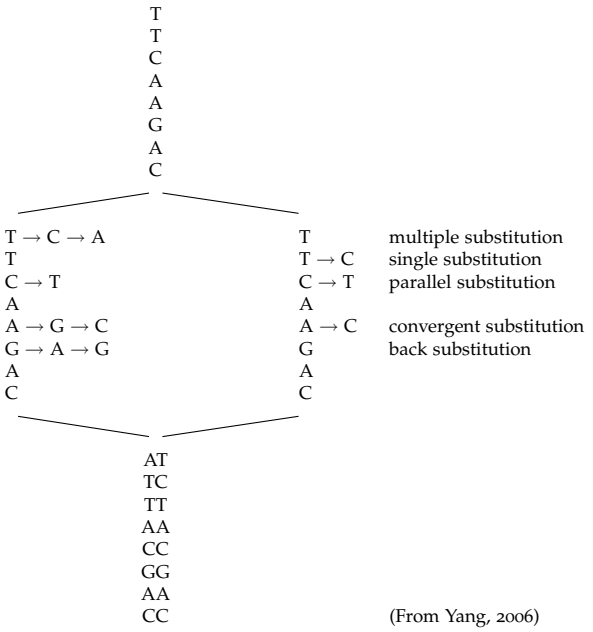
A few decades ago when gene sequences started to appear, the scientific community was comfortable with the idea of Darwinian evolution by random mutation followed by selection of the fittest, and it was assumed that process applied to molecular evolution as well. However, when the genes were examined, there appeared to be more randomness than the sort of selection that had been seen in morphological evolution. One of the earliest observations about sequence evolution was that there was a somewhat linear relationship between evolutionary separation of the organisms and the amount of sequence difference, the "molecular clock" of Zuckerkandl and Pauling. It was an easy extrapolation to imagine, or model, molecular change as a random process, perhaps something like radioactive decay. Soon a picture emerged of the large role of neutral evolution and the small role of selection in molecular evolution.

However, the process is not simple, as we have slow genes and fast genes, and slow and fast sites within genes. To explain these different rates we can recognize that different genes are more or less free to change, and different sites within genes are more or less constrained. At one extreme we have genes that are recognizably homologous throughout the entire tree of life, and at the other extreme we have pseudogenes that are no longer under selection, that are quickly randomized to unrecognizability. Within genes, some sites in proteins are absolutely essential and never change, but third codon positions are free to change rapidly.

2 Hidden mutations and parsimony

Phylogenetic reconstruction using parsimony is excellent when divergences are small. If the divergences are very small, it might even be difficult to fit a model due to lack of variation in the data. However, model-based methods such as ML (maximum likelihood) and Bayesian analysis offer advantages when divergences are large.

How might sequences evolve? If we start with a bit of DNA sequence, duplicate it as in a speciation, and allow each copy to evolve, various things might happen to the nucleotides.



(From Yang, 2006)

At the end of our evolutionary period we have 2 sequences, which we can recognize as being homologous, and we can align them and perhaps try to infer their history. The 2 present-day sequences conceal much of the complexity of their history.

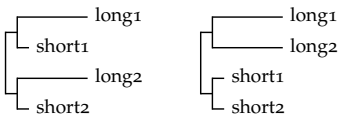
The approach taken by a parsimony analysis is that the history and underlying process of evolution is unknowable, and so we should not try to look for a common mechanism. So, for example, if two sequences in an analysis differ at half of their sites, the parsimony approach is to base conclusions only on these observed data at face value. However, it is reasonable to suspect that if half the sites differ, and mutations happen at random, then more than likely some sites have been hit more than once, and to say that the only changes that have occurred are the ones that we can see would be an underestimate. This other point of view carries the explicit assumption that there is a common mechanism – that mutations happen randomly – and makes inferences based on that, something that par-

simony is not willing to do. Parsimony is not willing to say what the mechanism of evolution is, but it is willing to say that whatever it is, it is not random. Parsimony, disallowing a common mechanism, instead makes the large set of unstated assumptions that each site evolves under its own unknown mechanism. Many people have pointed out that this is not a very parsimonious explanation at all, and allowing a single common mechanism is really the more parsimonious explanation.

A prediction of parsimony is that a character that evolves on a long branch will have the same expectation of change as on a short branch. A prediction of a common mechanism of random change is that a character that evolves on a long branch will have a greater probability of change than on a short branch. Which prediction is borne out in real molecular sequences?

2.1 Long branch attraction in parsimony

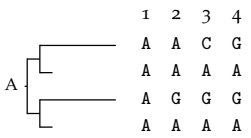
This is a well-known problem in phylogenetics, where unrelated long branches can end up being put together in a parsimony analysis.



True Recovered

Imagine the evolution of one site on the tree shown below. At the root of the tree, the character state is an `A`. Over the short branches, to short1 and short2, it remains an `A`. However, on the longer branches it has had time to be hit by mutations. The ancestral state `A` will be preserved in short1 and short2, but the character states will differ in taxa long1 and long2. Four different patterns might arise in the leaf taxa. Those patterns will be where character states in long1 and long2 are —

1. both the same as short1 and short2
2. one the same and one different
3. both different and different from each other
4. both different but the same as each other



1	`AAAA`	parsimony uninformative
2	`AAGA`	parsimony uninformative
3	`CAGA`	parsimony uninformative
4	`GAGA`	parsimony misinformative

All the possible patterns are either uninformative or misinformative. A parsimony analysis will tend to group the long branches together, and tend to do so more if you add more data (this explanation after SOWH96, in Hillis '96).

3 Simple likelihood calculations

In general, the likelihood is (proportional to) the probability of the data given the model. In phylogenetics, we can say, loosely, that the tree is part of the model, and so the likelihood is the probability of the data given the tree and the model. We call it the likelihood rather than the probability to emphasize that the model is the variable, not the data.

> The likelihood supplies a natural order of preferences among the possibilities under consideration.
>
> *-R.A. Fisher, 1956*

Imagine flipping a coin, and getting a "head". What is the probability of that datum? The probability depends on the model, and if you think it is a fair coin, then the probability of a head is 0.5. However if you think it is a double-headed coin then the probability will then be 1.0. The model that you use can have a big effect on the likelihood.

The models that we use in molecular phylogenetics take into account a few attributes of the underlying process. These include such "loaded dice" aspects as the equilibrium composition of the character states, and the rate of change between character states, and the among-site rate variation that reflects negative selection on the sequence.

We need to know the composition implied by the model, which may or may not be the composition of the data. We also need to know the relative rates of change between character states. If we look at an alignment of sequences such as this,

```
A  acgcaa
B  acataa
C  atgtca
D  gcgtta
```

we can see that for this particular dataset transitions (`a` ↔ `g` and `c` ↔ `t`) appear to occur more often than transversions (`a` or `g` ↔ `c` or `t`). We can have our model accommodate that. Even better, we can have our particular data tell the model how much transition-transversion bias to use.

In complex data the relative rates of change between nucleotide pairs might all be different, and these parameters can be estimated by ML.

The models that we use in molecular phylogenetics allow us to calculate the probability of the data. The simplest model for DNA sequence evolution, is the one formulated by Jukes and Cantor in 1969, and is known as the Jukes-Cantor or JC model. It is not a biologically realistic model, but it is a good place to start. In it, the model composition is equal base frequencies, and the rates of change between all bases are the same. We keep it simple and so have no among-site rate variation — all sites are assumed to be able to vary equally.

We can use this model to calculate probabilities of DNA sequence data even without a tree, and without any evolutionary changes. For example, lets do a first likelihood

calculation. The datum is "a". That's all — one sequence with one nucleotide, and no tree. So we don't even need to know about the rates of change between bases in our model, all we need is the composition part of the model. Its an easy calculation – the likelihood of our a using the JC model is 0.25.

The likelihood depends on the model that we use, and if we had used another model with a different composition, then we would have a different likelihood. If the model had a composition of 100% a, then the likelihood would have been 1. If the composition of the model was 100% c, a model that does not fit our data well at all, the likelihood would be zero.

We can do a second likelihood calculation, this time where the data are ac — 1 sequence, 2 bases long. We assume that the 2 events (bases) are independent, and so to calculate the probability we multiply. Using the JC model, that would be $1/4 \times 1/4 = 1/16$. That is the likelihood of ac under the JC model. Likelihoods under other models will differ.

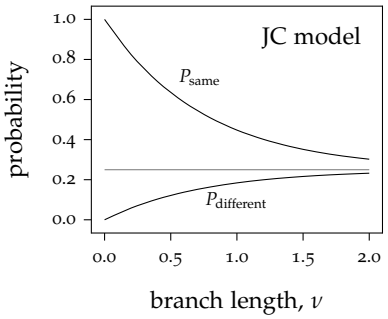
Now we will try to calculate the likelihood of a one-branch tree. The data will be 2 sequences, each 1 base long,

```
one  a
two  c
```

For this calculation we need the part of the model that describes the rate of change between bases, as we need to know how to calculate the probability of base a changing to c. With the models that we use, the probability depends on the branch length, ν. The branch length is measured in mutations ("hits") per site, averaged over the data. We can describe this part of the JC model with 2 equations as shown below, one for the probability of a base staying the same at a given branch length, and the other for the probability of a base changing to some other base at a given branch length. For our data we will need the latter, as we are looking at base a changing to c.

$$P_{\text{same}}(\nu) = \frac{1}{4} + \frac{3}{4}e^{-\frac{4}{3}\nu}$$
$$P_{\text{different}}(\nu) = \frac{1}{4} - \frac{1}{4}e^{-\frac{4}{3}\nu}$$

These curves show that at a branch length of zero, the probability of a base staying the same is 1, and the probability of changing to another base is zero, which seems reasonable. As branch lengths get longer, the probability of staying the same drops, and the probability of changing to something else rises, which again seems reasonable. As the branch length gets very long, the probability of both of these approaches 0.25, so at very long branches there is equal probability of any base changing to any other base, or staying the same; this will randomize any sequence that evolves under this model over very long branches. The random sequence will have the model composition, in this case all equal.



We need a branch length, so lets say that our branch length $\nu = 1$, and so $P_{\text{different}}(1) = 0.184$. In that case, with our one branch tree above, the probability of the base a by itself is 0.25, and the probability of the change to c is 0.184, and so the probability of the whole tree is 0.046. Had we started with c the result would have been the same, as we used a reversible model.

We can try to calculate the likelihood of another one-branch tree, this time with more data. The data alignment that we will use will be

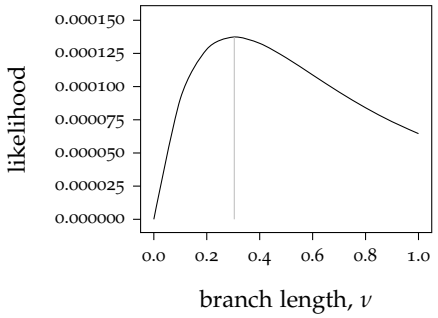
```
one  ccat
two  ccgt
```

and again we will use a branch length of 1. For this calculation we need the probability of a base staying the same at a branch length of 1, and that is $P_{\text{same}}(1) = 0.448$. We can start with sequence one, and using the composition (often notated as π, eg π_c), the P_{same}, and the $P_{\text{different}}$, we can calculate the probability of the tree, as

$$\begin{aligned} &= \pi_c \, \boldsymbol{P}_{c\to c} \, \pi_c \, \boldsymbol{P}_{c\to c} \, \pi_a \, \boldsymbol{P}_{a\to g} \, \pi_t \, \boldsymbol{P}_{t\to t} \\ &= 0.25 \times 0.448 \times 0.25 \times 0.448 \\ &\quad \times 0.25 \times 0.184 \times 0.25 \times 0.448 \\ &= 0.0000645 \end{aligned}$$

Now we have a likelihood of our tree at a branch length of 1 hit/site. Our data matrix has 3 out of 4 sites that are constant, and only 1 out of 4 change, so a branch length of 1 seems long. We can calculate the likelihood of the tree at various branch lengths and plot them.

branch length	likelihood
0.0	0.0000000
0.2	0.0001281
0.4	0.0001326
0.6	0.0001088
0.8	0.0000840
1.0	0.0000645



We can use numerical approximation to find the ML branch length, which is at 0.304099 hits/site, at which the likelihood is 0.000137. There was only 1 of the 4 positions that had an observable change, which would make the branch length 0.25 if there were no hidden changes. This model is telling us that the maximum likelihood is a little more than 0.25, implying that it assumes that there are hidden changes.

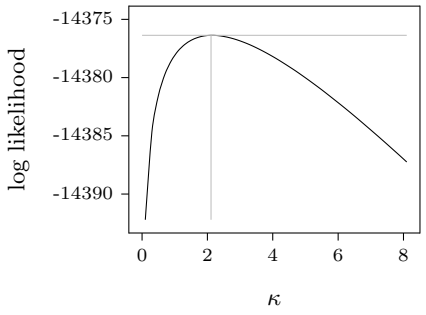
We can check that PAUP gets it right.

```
#NEXUS

begin data;
    dimensions ntax=2 nchar=4;
    format datatype=dna;
    matrix
    A   ccat
    B   ccgt;
end;

begin paup;
    set criterion=distance;
    lset  nst=1
          basefreq = equal;
    dset distance=ml;
    showdist;  [got 0.30410]
end;
```

We can find ML branch lengths described above, and we can optimize other model parameters as well. In this example, the ML estimate for κ is 2.11, at which the log likelihood is -14376.37



In molecular phylogenetics in practise, the data are an alignment of sequences. On every tree, we optimize model parameters and branch lengths to get the maximum likelihood. We are probably most interested in the ML topology, but we need to deal with these extra "nuisance parameters". Each site has a likelihood, and the total likelihood is the product of the site likelihoods. That is usually a very small number! –too small for computers, and so we use the sum of the log of the site likelihoods. The maximum likelihood tree is the tree topology that gives the highest (optimized) likelihood under the given model.

Here is an incomplete list of phylogenetics programs that use ML –

- PAUP*
  - $$, not open source, fast, well tested and debugged, DNA only
- Phylip, especially proml
- puzzle, *aka* Tree-Puzzle. Uses quartet puzzling.
- Phyml, very fast, has a web server
- RAxML, very fast
- TreeFinder, very fast, not open source
- PAML
- p4
- Leaphy
- IQPNNI
- HyPhy

References

- Swofford, Olsen, Waddell, and Hillis, 1996. *in* Hillis *et al*, *Molecular Systematics*.
- Felsenstein 2004. *Inferring Phylogenies*

4 An ML search

As described above, we can optimize branch lengths and model parameters on a given tree to find the ML for that tree. Ideally, to find the ML tree, we would repeat that with all trees, optimizing parameters and branch lengths on each, and looking for the tree that gives the highest value.

However, that is too computationally demanding. Not only is a thorough search of tree space generally impossible, but it is far too expensive to optimize model parameters on each tree. Although optimizing branch lengths is fast, optimizing model parameters is slow. This means that practically, we search tree space with fixed model parameters.

A workable strategy is to start with a good tree and optimize parameters on it, and then fix those parameters before searching tree space. Searching tree space would generally be an heuristic search, and have branch length optimization. At the end of the tree search, having found a better tree, parameters are re-optimized on that current best tree and the tree search is repeated. Alternating between optimizing model parameters on the current best tree and searching tree space with fixed model parameters is continued until no further improvement is seen; this will be evident when the last search does not find a better tree than the second-last search.

This strategy is built-in to `phyml`, but to do the same in `paup` requires explicit instructions. This has been called *successive approximation* in `paup`. Often the model for a search in `paup` will be as suggested by Modeltest (see below), and done with fixed parameters as given by Modeltest. If the `paup` search finds a better tree than the NJ tree used by Modeltest, then it is possible that the `paup` search is not complete.

5 Simple models of evolution

A model is an attempt to describe the important aspects of the underlying process that might have generated the data; a model is a mental picture of the way that you think that things might work. For likelihood, we need

models that allow you to calculate the probability of data. That would include models like JC, F81, GTR, and so on, but it would not include LogDet distances. There is a way to calculate a likelihood for parsimony, although this rarely done, and so our list might rarely include parsimony.

We model sequence change as a continuous time Markov process. For the sort of models that are used in the common phylogenetic programs, models are described in terms of the rate matrix, the composition, and the ASRV (among-site rate variation).

For example, in PAUP, we might describe a model for DNA as

```
lset nst=2 tratio=3.4 basefreq=empirical rates=equal;
```

which says to use a 2-parameter rate matrix, take the composition of the model from the data, and assume that all sites evolve at the same rate.

Rates of change between bases can be described with a rate matrix. PAUP and MrBayes use `nst`, the number of substitution types, to describe the type of rate matrix for DNA. The number of parameters is `nst` minus 1. JC and F81 models are `nst=1`, with no free rate matrix parameters; it is assumed in JC and F81 that the rate of change among bases is equal.

Assuming equal rates of change between bases is not biologically realistic, and a better approach is to allow different rates in transitions and transversions. K2P and HKY models are `nst=2`, with a *kappa* or `tRatio`. For example, in PAUP you might say `lset nst=2 tratio=5.7;`, which describes a 2-parameter rate matrix. You may come across the F84 model; it is similar to the HKY85 model, and both allow different base compositions with a 2-parameter rate matrix.

The "general time-reversible" or GTR rate matrix allows different rates of change between all bases. However, it is symmetrical, and that makes it time-reversible. In PAUP and MrBayes, the GTR is `nst=6`, with 5 parameters. In PAUP you might say `lset nst=6 rmatrix=estimate;`, which tells the program to estimate the parameters of the rate matrix. The GTR matrix, being symmetrical, can be described with 6 numbers, (a to f below), but if you know 5 of those 6 then the 6th can be calculated, and so there are really only 5 free parameters.

$$\boldsymbol{R} = \begin{bmatrix} - & a & b & c \\ a & - & d & e \\ b & d & - & f \\ c & e & f & - \end{bmatrix}$$

Using this notation, we can imagine restrictions of the GTR matrix that use fewer parameters. For example if $a = c = d = f$ and $b = e$, we really only have 2 parameters, and only one free parameter, and it would describe the `nst=2` 2-parameter models such as K2P. There are many possible restrictions of the GTR matrix, some of which have names, and are tested in Modeltest. Only some programs, such as PAUP, are able to use these restrictions.

Composition, the second aspect of how models are described, can be described in several ways. The simplest way is to say that the base frequencies are all equal, 25% each for DNA. In PAUP you can say `lset basefreq=equal`, which would be for the JC or K2P models. Another way to describe the composition, not often used, is to specify it completely as a series of frequencies. In PAUP you can say for example `lset basefreq=(.1 .2 .3)`. The best way, but also the slowest, would be to use ML values of the base frequencies, and in PAUP you can specify this by saying `lset basefreq=estimate`. Using empirical composition is fast, and is usually a good approximation of the ML composition. You can specify it in PAUP with `lset basefreq=empirical`. You would most often use empirical composition in your analysis, or ML if you have the computational time.

Table 1: Simple DNA models.

	`nst=1`	`nst=2`
equal composition	JC	K2P
unequal composition	F81	HKY/F84

The simplest DNA models that we have mentioned so far with `nst=1` and `nst=2` can have equal composition, or have unequal (usually free) composition. Their names are tabulated below. For DNA, if the composition is free then there are only 3 free parameters; this is because if you know 3 of the 4 composition frequencies then the 4th can be calculated.

Among-site rate variation, or ASRV, is the third aspect of how models are described. ASRV can be described in terms the of `pInvar`, gamma-distributed ASRV, or both. Using `pInvar`, the proportion of invariant sites, notated as *eg* GTR+I, allows a proportion of the constant sites in the alignment to be invariant. In PAUP it is set by, for example `lset pinvar=estimate;`. Note that a site may be constant, but not invariant, because it is potentially variable but has not yet varied, and that the ML estimated `pInvar` will often be less than the proportion of constant sites. Discrete gamma-distributed ASRV is notated as G, or Γ, *eg* HKY+G or GTR+I+G. This allows the rates of different sites to be modelled as a range of rates, and is described using only one parameter, α, the `shape` parameter in PAUP, as `lset rates=gamma shape=estimate;`.

Rate variation among data partitions is accommodated by some programs. This is sometimes called site-specific ASRV, but it might better be called among partition rate variation. This would be useful for looking at separate codon positions, or when using concatenated genes when the genes are different enough to model them separately. Each data partition can have a relative rate, *eg* third codon positions are fast, second codon positions are slow. The average relative rate, over all partitions, is generally by definition unity. PAUP has what it calls a site-specific model, but in PAUP other parameters (`RMatrix` *etc*) are homogeneous across all data partitions. The only thing that varies is the partition rate. Other programs, (MrBayes and p4) are more flexible with heterogeneous data, and allow different datatypes, rMatrices, composition, and ASRV in different partitions.

The most common models for amino acid data are em-

Table 2: Number of free parameters for various model components

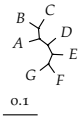
rate matrix		
	JC, F81	none
	K2P, HKY, F84	1
	GTR	5
	Restrictions of GTR	2 – 5
	protein	ignore
composition		
	equal	none
	ML or empirical	DNA 3
		Protein 19
ASRV		
	pInvar	1
	GDASRV	1

pirical protein models. As there are 20 amino acids, these use a 20×20 rate matrix. There are usually too few data with which to reliably estimate the 189 rate matrix parameters needed for an ML rate matrix (and it would be too slow!), so a reasonable compromise is to make an ML rate matrix from a large data set once, and apply it to your data. Such rate matrices include Dayhoff78, JTT, WAG, MTREV, and several others. These models have an inherent composition from the large data set from which they were made, and some programs (notably MrBayes) can only use that composition. Often it is better to use empirical composition, based on your data, if that is allowed by the program that you are using.

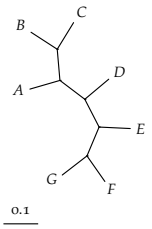
6 Gamma distributed among-site rate variation

Just as there are fast genes and slow genes, there are fast sites and slow sites within genes. Sites differ in how much they are free to vary. A site may be under strong selection and highly constrained; other sites, such as third codon positions, might be relatively unconstrained.

If we could reliably separate the fast sites from the slow sites and analyze the two sets separately, ideally we would get the same tree topology, but the branch lengths would be proportionally bigger in the fast sites. An example might be to separate the first and second codon sites of a protein-coding gene and analyze them separately from the third codon position.



Tree made using slow sites only



Tree made using fast sites only

We could then analyze both sets of sites together in a single analysis using a site-specific ASRV strategy. This strategy is often used with separate codon positions, and with different genes in an analysis with a few genes. It generally forces the branch lengths in the partitions to be proportional. The slow sites partition would have a slow partition rate, and the fast partition would have a fast partition rate. These partition rates can be found from the data by ML. These partition rates can be thought of as branch length multipliers, where the average of the multipliers is 1. Using this strategy gives a much better fit of the model to the data; not using this strategy forces all sites to be analyze with a one-size-fits-all branch length that is a compromise between the slow and fast rates, and fits neither one well.

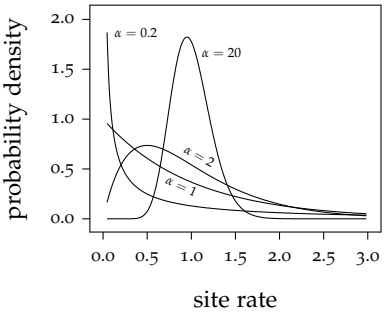


Figure 1: Gamma curves at various α values

The problem with this is that there is usually too much uncertainty in the separation of the sites into slow and fast categories. One very clever strategy that can be used here is to apply a *mixture model*. In this sort of model we do not separate the data into partitions, but instead we analyze every site as if it was in each rate category, and average the results.

If we look at the relative rates of sites in different genes we can notice that for some genes there is extreme ASRV, with many very slow sites, and a few sites that are very fast. In other genes there is a smaller range, where all the sites are more or less close to the average rate. To accommodate this variation in ASRV, it has been proposed that we model ASRV based on a gamma distribution. The gamma (or Γ) distribution is usually described with 2 numbers, α and β, that define the shape and mean of the distribution, but for our purposes we always want the mean to be 1, and so we only need the shape parameter α. That mean of 1 is the average branch length multiplier, and the fast and slow sites are relative to it. The shape of the gamma curve changes widely depending on α. For small values of α the curve is "L"-shaped, and for larger values it is a hill centred on 1. There is nothing compellingly biological about describing ASRV this way, but it does allow a wide range of rate-shapes with only a single parameter. That parameter is usually a free parameter in our models, and so it does not need to be provided, as it can be estimated from the data by ML.

It is possible to do the analysis by integrating over the site rates of the continuous gamma density, but it is just as

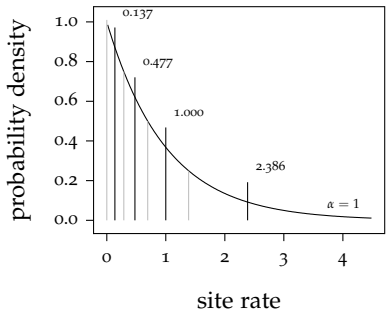


Figure 2: Discrete gamma divisions for $\alpha = 1$

```
 7  50
A         ttctaacgggacagtgcgcccactcacgcacctggtcactgtatgcgagt
B         tgcgaaggtgctattgcgagcattcacgcagatggtaactgtatgtgaga
C         tgcgaaggtgttattgcccacattcgcgcggaaggtaacactatgtgaga
D         tcccatagcgacatggcgcatactgactcctatggatactgtatgcgagt
E         tgcgaaggcgacattgcgcacattcacgcatatggttactgtacgtgagt
F         tgcgaaggcggcattgcgtccattcacccgcatggagactttatgtgaga
G         tgcgaagtcgacattacgctcttttacgcacagggtcactttatgggaca
          *         *
          1313123231421222123422213213133231124112311213112 2
```

Figure 3: Example data for GDASRV calculation

good and much faster to use a discrete approximation to the continuous curve. These strategies were developed by Ziheng Yang in the mid-1990's. The idea is that we can approximate the continuous curve by dividing up the curve into a number of discrete categories, and then we only need to average over those categories rather than integrating over the continuous curve. Four categories is usually considered to be sufficient.

Paup will tell you the borders and the means of the categories with the `gammaplot` command. For example, for the gamma curve where $\alpha = 1$, the output from `gammaplot` is

```
category      lower        upper       rate (mean)
--------------------------------------------------
    1      0.00000000   0.28768207      0.13695378
    2      0.28768207   0.69314718      0.47675186
    3      0.69314718   1.38629436      1.00000000
    4      1.38629436     infinity      2.38629436
```

which we can plot as in Figure 2. The mean of the mean rates is 1.0. We can show how the calculation works using an example, which we can analyze using a JC+G model.

In the data in Figure 3, the last line shows the number of different character states in the alignment column. This should imperfectly reflect the site rate. If we analyze these data on a particular tree with a JC model, the log likelihood is -307.57; under the JC+G model, with a shape of $\alpha = 1$, it is -305.70, a slight improvement.

We will look at site 1 (all t) as an example of a slow site, and site 11 (all 4 nt) as an example a fast site. With no gamma model, these site likelihoods are 0.0848452 and 0.0000111865, respectively, and with the gamma model they are 0.117246 and 0.0000150721.

site	rate category	site rate	likelihood
1	1	0.137	0.215112
	2	0.477	0.148608
	3	1.000	0.084845
	4	2.386	0.020420
		mean	0.117246
11	1	0.137	0.0000000628
	2	0.477	0.0000019392
	3	1.000	0.0000111865
	4	2.386	0.0000471000
		mean	0.0000150721

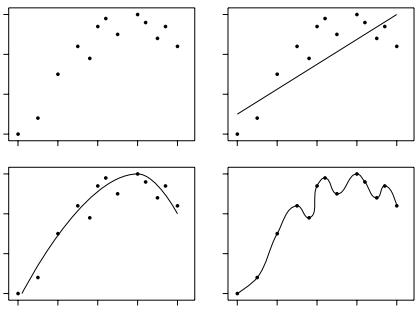
Here we average over the states of our uncertainty, and estimate a site rate that gives a better fit of the model to the data. The slow category contributes most to the slow site, and the fast category contributes most to the fast site. This strategy comes at a cost of 4 times the likelihood calculations, and 4 times the memory requirement.

7 Choosing a model

Last century, it was common to assume a model without justification. However, we should choose a model that fits our data, and be able to justify that choice.

Models differ in their free, *ie* adjustable, parameters. More parameters are often necessary to better approximate the complex reality of evolution. Usually, the more free parameters in the model, the better the fit (as measured with a higher likelihood) of the model to the data; this would be a good thing. However, the more free parameters, the higher the variance, and the less power we have to discriminate among competing hypotheses; this would be a bad thing. Also, some parameters might not be important, or might only model meaningless noise in the data; generally we do not want that: we do not want to "over-fit" the model to the data.

By analogy, we can ask What is the best way to fit a line (a model) through these points?



A linear fit would capture some of the trend. A quadratic fit would be better. It would be possible for a higher-order fit to go through every point – but then you would only be fitting noise, *ie* "over-fitting" the data, not capturing extra important trends in the data.

We take a similar approach to choosing a model in phylogenetics. We want to choose a model that describes im-

portant trends in the data, but without describing noise. While it is easy and intuitive to see over-fitting in a plot such as that above, it is less easy in the phylogenetic context. The way we choose among available models in phylogenetics is to start with a reasonably good tree, such as the NJ or MP tree, and then evaluate the likelihood (ML for that tree) for all the models that you want to test. Since the same tree is used over and over, there is no tree search involved, so it is relatively fast. Then we choose as the best model the model that has enough parameters, but not too many. This may be the ML model, or it might not be.

We want to choose a model that gives the highest likelihood, but penalized by the number of free parameters. This is formalized in the AIC, the Akaike Information Criterion, $-2\log L + 2n$, 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. Informally, the AIC says that adding a useless parameter generally increases the log likelihood by about 1 unit, and so if adding a parameter increases the log likelihood by more than 1, it is not useless.

A table of AIC values might be like this —

	$\ln L$	n	$-2\ln L + 2n$	
JC	-5211.7	0	10423.4	
F81	-5166.6	3	10339.2	
HKY85	-5125.0	4	10258.0	
GTR	-5092.5	8	10201.0	
GTR+I	-4946.1	9	9910.2	
GTR+G	-4937.8	9	9893.6	⇐
GTR+IG	-4937.2	10	9894.4	

In ranking the models by the AIC, we penalize each likelihood by the number of free parameters. In the example above, the GTR+G is the best model, even though the GTR+IG had a higher likelihood.

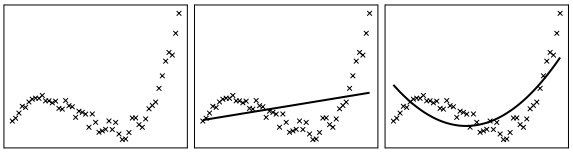
It is important to know the number of free parameters in models. Here are some examples —

Model	rate matrix parameters	composition parameters	ASRV parameters	total parameters
JC69	0	0	0	0
F81	0	3	0	3
HKY+G	1	3	1	5
TN93	2	3	0	5
GTR+IG	5	3	2	10
JTT+F	0	19	0	19
WAG+IG	0	0	2	2

Choosing a model in phylogenetics is done often, and so attempts have been made to automate that choice. One of the most common is Modeltest (D. Posada and K. A. Crandall 1998. "MODELTEST: testing the model of DNA substitution" *Bioinformatics* 14: 817-818.) It uses PAUP to do the likelihood calculations, and uses both the AIC and the likelihood ratio test to evaluate those likelihoods. However, it only tests the models that it tests, and it does not test clock models or "site-specific" models included in PAUP, nor does it test models that are not in PAUP. Other programs to automate model choice include ProtTest, Modelgenerator, and MrModelTest.

Besides the suggested model, an interesting output of Modeltest is the table if Akaike weights, which can give some idea of the ambiguity surrounding model choice.

7.1 Does the model fit?



Imagine that we have some complex data, but all we have available are two models – linear and quadratic. We can choose the quadratic model as the better of the two available, but even though it is the best of the two it does not fit well. What is really needed is a better model. In the case of this scatterplot it is easy to see the inadequacy, but it would be less obvious with phylogenetic data. When you choose a model, you often "max out" and choose the most complex model available – often the GTR+IG model. That should make you suspect that you might have chosen a better model had one been available.

We should apply statistical methods such as *posterior predictive simulation* to assess the fit of the model to the data. However, the question of whether the data fits is rarely even asked, let alone answered.

8 A survey of some other models

The simple DNA models described above are standard and widely used, but there are many other models and variations. A few are described below.

Identifying selection using codon models. Codon models can be used to model the evolution of DNA sequences of codons of protein coding genes. This does not use a 64×64 rate matrix — simplifications are used and far fewer parameters are estimated. Using these models we can have two kinds of change — synonymous changes where the amino acid would not change, and non-synonymous where the amino acid does change.

Expressing these as rates, we have d_S as the number of synonymous substitutions per synonymous site, and d_N as the number of non-synonymous substitutions per non-synonymous site. The ratio $\omega = d_N/d_S$ measures selection at the protein level. For neutral evolution we would expect $\omega = 1$. However, mostly we find $\omega < 1$, indicating purifying (negative) selection. Occasionally we see $\omega > 1$, showing diversifying (positive) selection. PAML implements these models.

Other mixture models. Gamma ASRV described above is one kind of mixture model, made to accommodate heterogeneity of rates among sites. We can have other mixture models, to accommodate heterogeneity of composition among sites, and heterogeneity of the rate matrix among sites. As usual you do not divide the data — rather, for each site, you average over the possible states. Such models are implemented in PhyloBayes and BayesPhylogenies.

Covarion model We can con consider the covarion model to be a generalization of the pInvar model. The covarion allows a site can change from invariable to variable and back over time. This appears to be biochemically realistic as it may describe the way that paralogs evolve. It is a mixture model – a site at a given position in the tree can be either on or off, but since we cannot know which, we evaluate assuming both. It has only two extra parameters: the rates $r_{\text{off}\to\text{on}}$, and $r_{\text{on}\to\text{off}}$. It is implemented in MrBayes.

Tree-heterogeneous models Most models are homogeneous over the tree. However, we know that the process of evolution can and does differ over the tree. This is easy to see when homologous genes differ in composition. To describe this evolution we can allow the characteristics of the model, for example the model composition, to change over the tree. Such models are implemented in PAML, `p4`, PhyloBayes (with the CAT-BP model) and `phase`.

Recoding data Some parts of sequences become saturated faster than others. Saturated sequences are noisy, often biased, and difficult to model. One approach to lessening the effects of saturation is to recode the data. An old method is to use RY-recoding, where A and G are recoded as R, and C and T are recoded as Y The recoded data would use a 2-state model, describing transversions only. A newer approach is to use grouped amino acids, where for example the amino acids are recoded into the six Dayhoff groups (C, STPAG, NDEQ, HRK, MILV, FYW) where the amino acids tend to exchange faster within the groups than between the groups. This allows a GTR-like 6×6 rate matrix with free parameters. This is implemented in `p4` and PhyloBayes.

Modelling RNA stems In structural RNA stems we have 3 kinds of stable pairs — `A:U`, `G:C`, and `G:U`. We also see mismatches (`A:G` *etc*). It seems reasonable to model these pairs as a unit, and use a 16×16 rate matrix for all possible combinations. These models are implemented in `phase`.

Clock models and molecular dating There is obviously some truth in the clock-like behaviour of evolutionary change, and it is common to attempt to date divergences with molecular data, calibrated with fossil dates. The strict molecular clock model, where the tips of the tree are constrained to line up at the present, rarely applies; relaxed clock models seem to work better. Such models are implemented in BEAST, and PAML.

8.1 Using multi-partition data models

What programs can do multi-partition models?

ML		
	PAUP	yes, but limited
	Phyml	no
	RAxML	yes
	p4	yes, but no search
Bayesian		
	MrBayes	yes
	p4	yes

9 Simulating evolution

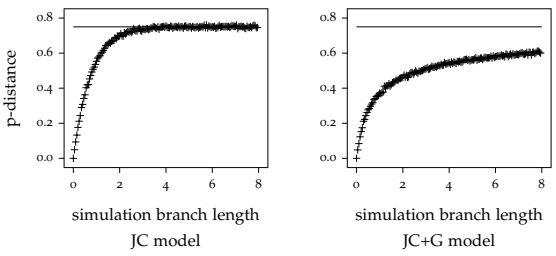
A model is your idea of the way that you think that things work. Armed with this it is possible to simulate sequences and evolve those on a tree to simulate evolution. This can be useful to test the methods and models. When you simulate data you simulate on a given model and on a given tree, and since you know these you can use the simulated data to test methods. You can also use simulations to test models; if you simulate data on a model that you are using for your real data and the simulated data are not similar to the original data, you can conclude that the model does not fit.

Simulation can help us to visualize evolutionary ideas. For example we can look a the problem of sequence saturation. Saturation is loss of phylogenetic signal due to superimposed mutations. When sequences become saturated phylogenetic signal has been randomized and trees that are made from those data are not reliable. One way to visualize saturation is to plot p-distances *vs* model-based simulation or inferred distances between all sequence pairs. P-distances are the simple face-value differences between sequences, calculated by the number sequence differences divided by the sequence length. With the JC model, the maximum p-distances will be 0.75, which we will see if we compare two random sequences with equal base frequencies. It is 0.75 and not 1.0 because even in random sequences 1/4 of the bases will happen to be the same in both sequences compared.

If we make random sequences and evolve those sequences based on the JC model for ever increasing distances, the p-distances between the original and the evolved sequences increase at first, but eventually level off when the sequences become saturated, and the p-distances can no longer increase. With the JC model saturation happens when the sequences have been mutated by about 2.5 or 3 hits per site.

If we evolve the sequences under a JC+G model, that is with among-site rate variation, the onset of saturation is delayed. Here the mutations are being concentrated in the fast sites, and they would become well saturated early on, but the slow sites are relatively untouched.

You can visualize saturation in real data in the same way, by plotting pairwise p-distances *vs* model-based distances (such as the sum of the branch lengths on the tree path between the taxon pairs in a ML tree). If the plot shows the tell-tale plateau then you have saturation.



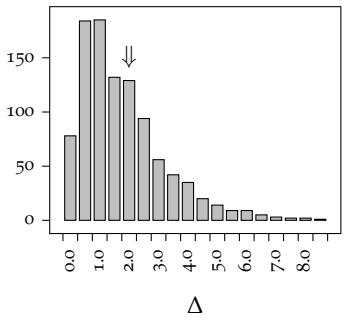
10 Simulation and the likelihood ratio test

Previously we have used the AIC to compare models. Another way to compare models is to use the likelihood ratio test. It will be demonstrated using simulations.

If we have some data, a tree, and a model, it will have a certain likelihood. If we use the same data and tree, but measure the likelihood under a more complex model, we will find that we get a higher likelihood. The question is whether that increase is significant, which will tell us whether the more complex model is the better model.

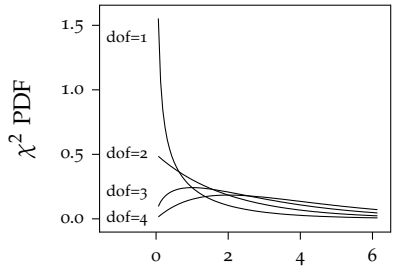
An increase in likelihood under the more complex model will be expected regardless of whether the extra parameters are important; we will expect a small increase due to noise alone, and a larger increase if the more complex model really is better. So how much of an increase is big enough? To answer that, we need to know how much we would expect the likelihood to increase due to noise alone. Then, if our increase is more than that, it is significant.

Lets use an example — we have some data and a tree, and under the HKY model we get a log likelihood of -1328.04985. Using the same data and tree, we optimize parameters under the GTR model and find a log likelihood of -1325.67926. That difference, 2.37059, is the *likelihood ratio*, and we ask whether it is significant. To assess its significance, we simulate data the same size as our original data, using our tree and the HKY model. Using the newly simulated data, and our same tree, we optimize parameters and branch lengths using both the HKY and the GTR models. For example, on one simulated data set, we get an increase of 2.67, and we know that is due only to noise. To determine if that is typical, we can simulate and analyze 1000 data sets, and plot the log of the likelihood ratio (Δ)

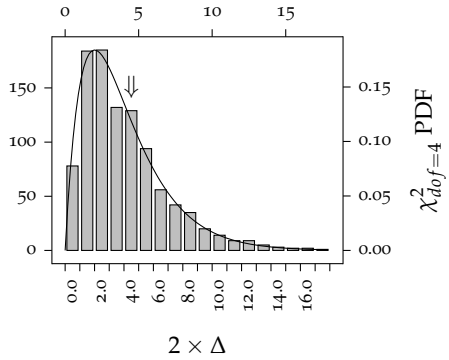


It looks like the increase that we got with the original data (2.37, shown with ⇓) was about the same as we generally got with the simulated data. So in our case, with the original data, the GTR was not a significant improvement.

We need not do simulations, because twice the log likelihood ratios are χ^2 distributed, and so we can use the χ^2 curve with the degrees of freedom given by the difference in the number of free parameters to assess significance.



In our example above, the GTR model has 5 free parameters and the HKY model has 1 free parameter, and so twice the log likelihood ratio is expected to be $\chi^2_{dof=4}$ distributed, as shown here



Using twice the log likelihood ratio from above ($2 \times 2.37 = 4.74$), we can assess its significance using $\chi^2_{dof=4}$, where $P = 0.31$, which is not significant at the $P = 0.05$ level.

The likelihood ratio test only works well for nested models; nested models are models where one model is a special case of another, *eg* HKY is a special case of GTR. Sometimes the LRT does not work well even for nested models, *eg* comparing models with and without GDASRV.

11 Bayesian phylogenetic analysis

Phylogenetic analysis using a Bayesian approach has become very widely used in the few years since its introduction. It has a acceptable speed, and can handle big datasets with parameter-rich models. Here we will look at differences between ML and Bayesian approaches in general, and in the next section we will look at Bayesian phylogenetic methods in practise.

One of the main ways that the Bayesian approach differs from ML is that the Bayesian approach deals with uncertainty in a more explicit way. For example, nuisance parameters such as branch lengths and model parameters will each have some uncertainty associated with them. The ML approach is to find the ML values for all of those nuisance parameters, but the Bayesian approach is to retain that uncertainty in the result. This means that the result of a Bayesian analysis is usually not a single point, but rather is a probability density or distribution. Being a distribution might mean that it is awkward to summarize the result to pick out the message that you want to get from it, but even so it can be considered an advantage over ML.

While in ML a picture of the uncertainty involved is often done after the analysis, often laboriously, as in for example the bootstrap, in a Bayesian analysis the uncertainty is part of the initial result.

Another difference is that Bayesian analysis explicitly relies on *prior probabilities*. The prior probability might be known with confidence, but for many problems we have only a vague idea of what the prior probabilities are, and since a Bayesian analysis forces us to be explicit about them we may have to make something up. Requiring a prior probability can be considered both a strength and a weakness, and is certainly controversial.

Of course the implementation details differ between Bayesian and ML analysis. ML uses hill-climbing algorithms to get the result to any required precision, while Bayesian analyses generally require an approximating algorithm such as the MCMC. From a computational point of view, the Bayesian MCMC can handle more parameters than ML, which means that you can solve bigger problems with more realistic models.

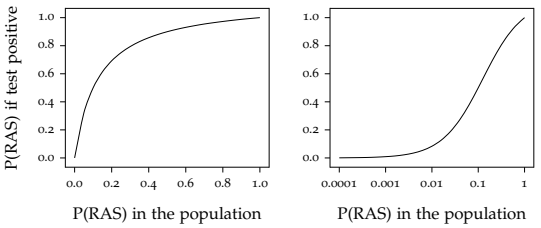
While ML expresses itself in terms of the probability of the data given the model, the Bayesian approach expresses the result as the probability of the model given the data. The probability of the model, or hypothesis, is more likely what the investigator wants, and this directness is one of the main attractions of Bayesian analysis.

11.1 Rare diseases and imperfect tests

Lets say that we are testing for a disease – Really Awful Stinkfoot, or RAS, and we know that 1% of the population suffer from it. We have a test for RAS that is fairly accurate – if you suffer from RAS then the test will tell you so 90% of the time. The test sometimes gives false positives – if you do not suffer from the disease, the test will tell you that you do suffer from it 10% of the time. Lets say that one of your patients tests positive for RAS. What is the probability that they actually have the disease?

It is perhaps easiest to explain it if we imagine giving the test to 1000 people. Of those, 10 will have the disease, and 9 will test positive. The remaining 990 do not have the disease, but 99 of them will test positive anyway. So after testing 1000 people we get 108 positives, but we know we only have 9 true RAS sufferers in those 108 people. So the probability of having RAS if you test positive is only 9/108, about 8%. That's all!

The surprisingly low probability depends mostly on the low background frequency of RAS in the population. That is the *prior probability* of RAS, that is the probability that you would expect somebody to have RAS *before* you see the result of the test. The probability of somebody having RAS if they test positive, 8%, is the probability *after* you see the test result – it is the *posterior probability*. We have a prior opinion, and we modify our opinion in the light of new information. Our prior opinion is a major player in the calculation; we cannot base our calculation only on the test results. Our prior opinion is not *replaced* by the new information provided by the test, rather it is *adjusted* by it.



$$
\begin{aligned}
P(RAS|\oplus) &= \frac{P(\oplus|RAS)P(RAS)}{P(\oplus)} = \frac{0.009}{0.108} \\
&= \text{The posterior probability of having RAS given a positive test.} \\
P(\oplus|RAS) &= \text{The likelihood. The probability of a positive test if you have RAS. The probability of true positive tests, 0.9} \\
P(RAS) &= \text{The prior probability of RAS, 0.01} \\
P(\oplus) &= \text{The } \textit{marginal likelihood}\text{. The probability of getting the data, a positive test, under all circumstances. That would include true positives and false positives.} \\
&= P(\oplus|RAS)P(RAS) + P(\oplus|healthy)P(healthy) \\
&= (0.9 \times 0.01) + (0.1 \times 0.99) \\
&= 0.108 \\
P(\oplus|healthy) &= \text{The probability of a positive test if you are healthy. The probability of a false positive, 0.1} \\
P(healthy) &= \text{The prior probability of not having RAS, } 1 - P(RAS) = 0.99
\end{aligned}
$$

This is formalized in Bayes' Theorem. Bayes' Theorem says that the relative belief that you have in some hypothesis given some data is the support that the data provide for the hypothesis times the prior belief in the hypothesis, divided by the support times the prior for all hypotheses.

Before you give a test to somebody, you think that they might have RAS with a probability of 1%. If they test positive, you use that information to adjust your estimate to 8%. But you want to be more sure, so you give them another test. Then, starting from your current estimate of 8%, you incorporate the result of the new test to adjust your estimate up or down. If they test positive on that second test, then you will be somewhat more sure that they are a RAS sufferer, but you still won't be *completely* sure based on the results of only 2 imperfect tests.

As you do more tests on your patient, the accumulated test results tend to overwhelm the original prior of 1%. If you have a lot of test results then even if your original prior had been somewhat wrong it will not matter much. If you do not have many data then the result may be noticeably influenced by the prior, but if you have a lot of data then the result will be mostly data-driven.

11.2 Prior distributions and nuisance parameters

In the analysis above, lets say that we are not really sure about our point estimate of the incidence of RAS in the population. While most studies place the incidence of RAS at 1–2%, one study places it at 5%, while another controversial study places it at over 10%. We should expect some uncertainty in those estimates — after all, those estimates are based on tests like ours, and we know they are imperfect. So our prior probability is no longer a single point at 1% — it now becomes a *prior distribution*. To do the calculations we need to be explicit about it, and describe the distribution completely. This may mean choosing a uniform range, or perhaps, if it seems more suitable, a curve such as the beta distribution. If we do that, then when we calculate the posterior probability it will also be a distribution. Now if you were a doctor and one of your patients tested positive for RAS, what would you tell them? The complete answer, the whole posterior probability distribution, might be not be welcome by the patient, who really just wants a simple answer. You might instead choose to give your answer as a range or an average taken from the distribution.

If you are not sure of the prior, you might think that to be fair and objective you should assume a uniform prior probability from 0 – 100%. However, that might not be satisfactory, as then the posterior distribution will also be from 0 – 100%. If you then choose to state the posterior as a range, you might find yourself telling your patient that based on their positive test they have a 0 – 100% probability of having RAS — hardly a satisfactory answer.

The test for RAS might involve nuisance parameters such as the age, gender, or background of the patient that might affect their susceptibility to RAS, and we can formulate a model involving these parameters to allow us to calculate the likelihood of the data in the RAS test under various conditions. The probability of the data given the model is the likelihood, and it will be a multidimensional distribution. The nuisance parameters will all have prior distributions, and so the prior probability will be a multidimensional distribution as well.

12 Bayesian phylogenetic analysis in practise

Phylogenetic applications of a Bayesian approach will use complex models and have many parameters, and are too big to calculate analytically. The likelihoods and the prior probabilities at various points in the distribution are relatively easy to calculate, but the marginal likelihood of these multidimensional distribution problems becomes complex and intractable. Fortunately there are ways to approximate the posterior distribution that do not require calculation of the marginal likelihood, and the most common way uses a Markov chain Monte Carlo (MCMC) approach using the Metropolis-Hastings algorithm. This approach depends only on making posterior probability *ratios*, and so while the likelihoods and priors need to be calculated, the marginal likelihoods in the ratio cancel out, and need not be calculated.

The MCMC is a computational machine that takes samples from the posterior distribution. The more samples you let it take, the better it's approximation, like pixels building up a picture until you can recognize it. It is able to handle complex models and lots of parameters, and so we can make our models realistic. The result of an MCMC is a large number of samples, and that leaves us with the easily surmountable problem of how to digest and summarize those samples to extract some meaning. A bigger and more difficult problem is to find out whether it has run well, and whether it has run long enough.

We can interpret the results in a very direct way. For example, the highest posterior probability tree is the one that gets sampled most often, and the posterior probability of a split is simply its frequency in the samples.

As with any Bayesian analysis, we absolutely need to have prior probabilities for all our parameters. Everything needs priors, including branch lengths, composition parameters, rate matrix parameters, ASRV parameters, APRV parameters, and tree topologies. However, usually we have so much data that the prior gets overwhelmed by the likelihood – so generally we don't need to worry about priors much. Rarely, if the data are few or not decisive, then the prior may have an influence and we may need to pay closer attention to it.

A good reference for Bayesian analysis is Mark Holder and Paul O. Lewis. 2003. Phylogeny estimation: Traditional and Bayesian approaches. Nature Reviews Genetics 4: 275–284.

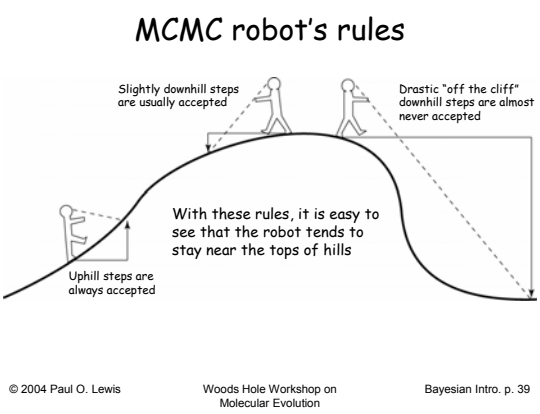
The first papers and demonstration programs for Bayesian analysis in phylogenetics were in the mid-1990's. The first practical program was BAMBE, by Larget and Simon, in 1999. In 2000 MrBayes was released, a program by John Huelsenbeck and Fredrik Ronquist. Other programs that do Bayesian analysis in phylogenetics are p4, BEAST, BayesPhylogenies, Phycas, and PhyloBayes.

As described above, the Markov chain Monte Carlo is a computational machine; it can be thought of as a chain of phylogenetic states. After the chain equilibrates, it visits tree space and parameter space in proportion to the posterior probability of the hypotheses, *ie* the tree and parameters. We let the chain run for many thousands or millions of cycles so that it builds up a picture of the most probable trees and parameters. We sample the chain as it runs and save the tree samples and parameters to a file. The result of the MCMC is a sampled representation of the parameters and tree topologies. The samples mostly come from regions of highest posterior probability.

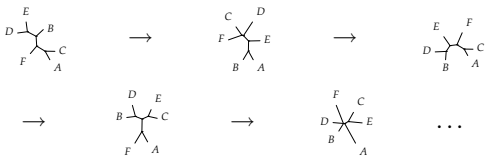
To get the MCMC going, we need to start somewhere, with a defined model, tree topology, and branch lengths. That "somewhere" will have a likelihood and a prior; here the likelihood is not the optimized, maximum likelihood. To take the first step we randomly propose a new state. In that proposal, perhaps we might adjust one of the branch lengths, or adjust the model parameters, or perhaps change the topology slightly. Without actually committing to going to the new state, the likelihood (unoptimized) and prior of the new state is then calculated. If the new state has a better likelihood $\times$ prior, the chain goes there.

However, if the proposed state has a worse probability, the chain does not automatically go there. A decision is made by the following process – First we need to calculate the posterior probability ratio between the current and the proposed states. That ratio will be between 0 and 1. Then we choose a random number between 0 and 1. If the random number is less than the the probability ratio of the two states, then the proposed state is accepted. So if the probability of the proposed state is only a little worse, it will sometimes be accepted. This means that the chain can cross probability valleys.

This process has been described in a cartoon by Paul Lewis.



The chain goes from state to state to state. Proposals are made, sometimes accepted, but often rejected. Branch lengths, topology, and model parameters change.



We save samples to digest later. Sampled trees are written to a file during the chain. At the end of the run we can summarize those samples. Trees in the file can be analyzed for tree partitions, from which a consensus tree can be made. The proportion of a given tree partition in the trees is the posterior probability of that partition. The proportion of a given tree topology in these trees is the posterior probability of that tree topology. Other parameters are written to a different file. These continuous parameters may be averaged, and the variance calculated.

Making a majority-rule consensus tree

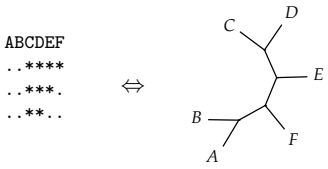
We can represent any tree as a list of splits, in 'dot-star' notation.

```
ABCDEF
..****
..***.
..**..
```

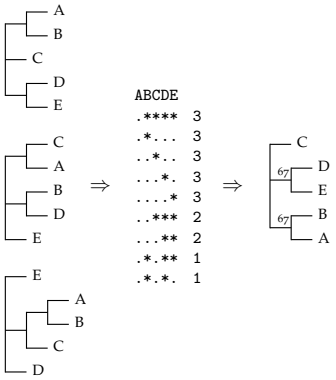
By convention the first position is always a dot. Terminal branches may or may not be included —

```
ABCDEF
.*****
..*...
```

Trees and lists of splits can be inter-converted. They are equivalent.

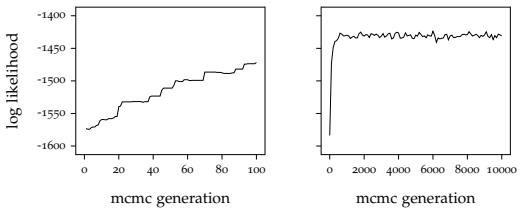


We can combine lists of splits from several trees, and then make a consensus tree from the most common (majority rule) splits in the combined list. This is typically done to summarize the sampled trees of an MCMC. The internal nodes of the consensus tree are usually decorated with the frequency of the splits in the combined list, showing the posterior probability of those splits.



Running the MCMC

Burn-in. The start of the chain is random and poor, so the first proposals tend to make the probability much better quickly. If we plot the likelihood of the chain at 2 different x-axis scales —



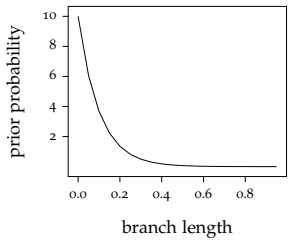
we see the typical fast rise in likelihood, and eventual noisy plateau. The chain really only works properly after it has converged or equilibrated. Since it generally starts out poorly, the first samples (100? 1000?) are discarded as "burn-in". You can plot the likelihood of the chain to see if it has reached a plateau, and only use the samples from the plateau; this is a widely-used but unreliable way of assessing convergence — there are better ways.

Thinning the chain Often proposed states are not accepted, so the chain does not move; this is not good for getting a good picture of the probability distribution. Or

perhaps the proposals are too near the current state, causing *autocorrelation*, which decreases the *effective sample size*. To ameliorate these, rather than sampling the chain at every generation, the chain is sampled more rarely, *eg* every 100 or every 500 generations. These sampled states will more likely be different from each other, and so be more useful.

Adjusting the prior The prior probability should usually be a distribution, such as a uniform or exponential distribution. The prior can also be fixed to single values, but this generally is not a good idea, as it does not give a good picture of the posterior probability. An exception is that we generally do fix the rate matrix for proteins, *eg* in MrBayes we might say `prset aamodelpr=fixed(wag)`.

In MrBayes, the default prior on branch lengths is exponential, $\theta e^{-\theta v}$, where v is the branch length. The mean is θ^{-1}, and by default, $\theta = 10$. One effect of this is that if the data are indecisive, then branch lengths will tend to 0.1 because of the prior.



Acceptance rates. If we look at the output from a MrBayes run, we can see a table of proposal acceptances like this —

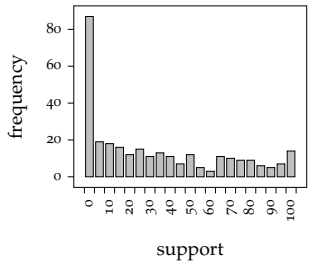
```
Acceptance rates for the moves in the "cold" chain of run 1:
   With prob.  Chain accepted changes to
     53.40 %   param. 1 (revmat) with Dirichlet proposal
      8.42 %   param. 2 (state frequencies) with Dirichlet proposal
     97.94 %   param. 4 (prop. invar. sites) with sliding window
     84.28 %   param. 5 (rate multiplier) with Dirichlet proposal
     26.85 %   param. 6 (topology and branch lengths) with extending TBR
     18.15 %   param. 6 (topology and branch lengths) with LOCAL
Acceptance rates for the moves in the "cold" chain of run 2:
   With prob.  Chain accepted changes to
     54.08 %   param. 1 (revmat) with Dirichlet proposal
      7.12 %   param. 2 (state frequencies) with Dirichlet proposal
     98.07 %   param. 4 (prop. invar. sites) with sliding window
     84.12 %   param. 5 (rate multiplier) with Dirichlet proposal
     28.47 %   param. 6 (topology and branch lengths) with extending TBR
     17.33 %   param. 6 (topology and branch lengths) with LOCAL
```

We want good *mixing*. We can get some idea of the mixing from the proposal acceptance rates. Proposals that take baby steps that are too close to the current state will tend to have high acceptances. Proposals that take giant steps that are too far from the current state will tend to have low acceptances. We want medium steps for good mixing. A rule of thumb is to aim for 10% – 70% acceptance for good mixing. If there are problems in the acceptances one can sometimes change the tuning parameters of the MCMC, for example using the `propset` command in MrBayes.

MCMCMC. MrBayes introduced Metropolis-coupled MCMC. In this strategy, several chains are run in parallel; all but one is "heated". Heated chains are adjusted so that acceptance probabilities are increased, which allows easier crossing of likelihood valleys. Heated chains act as "scouts" for the cold chain (thanks to Paul Lewis for this analogy). Chains are allowed to swap with each other, so the cold chain may exchange identity with a heated chain. Only the cold chain is sampled.

The star tree paradox

If we simulate data on a star tree with 4 taxa, where each branch has a length of 0.05, then there will be about 1 mutation every 20 characters. If we simulate datasets of 100,000 characters, there will be plenty of information, however since the simulation used a star tree, in an ideal analysis there should be no clear favourite, and any favoured resolution that we might see must be an artifact, presumably due to noise. We can do Bayesian runs with MrBayes and look at the support for all 3 possible trees. Whereas you might expect that the 3 possible trees would be supported about equally, about 1/3 each, that is not what is found. The following is from 100 simulations and Bayesian runs, and shows the support for all 300 trees. Note that if one or two trees in an analysis have high support, then the other tree or trees must have low support, and since we see quite a few high supports, we also see many very low supports.



Priors on trees MrBayes assumes equal prior probability for all trees. On the surface, this might seem like the obvious thing to do. However, MrBayes only considers fully resolved trees. For a given number of taxa, about half of all possible trees are not fully resolved. It turns out that the MrBayes "equal" prior on tree topologies is not equal at all, as it gives polytomous trees a prior of zero.

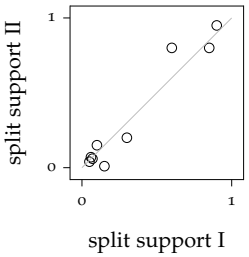
It is possible to have an MCMC that allows polytomies. (See Lewis, Holder, and Holsinger, 2005. Polytomies and Bayesian phylogenetic inference, *Syst Biol* **54**(2): 241–253. Their polytomy prior is implemented in p4 and in phycas). We might give equal prior probability to any tree, resolved or not. Alternatively we might decide to have a higher prior probability for unresolved trees — then if you do get resolution, driven by the data, it will be more convincing. Allowing polytomies seems to eliminate the star tree paradox, and can reduce artefactually high support for some poorly-supported nodes in real data sets.

Assessing convergence

In the early days of Bayesian analysis in phylogenetics, it was common to plot the likelihood or other parameters of the run and so identify the burn-in and the typical plateau area, and if you had an obvious plateau that was taken

split	support I	support II	standard deviation
..****	0.90	0.95	0.035
...***	0.85	0.80	0.035
....**	0.60	0.80	0.141
..**.*	0.30	0.20	0.071
..*.**	0.15	0.01	0.099
.*.***	0.10	0.15	0.035
.**..*	0.07	0.06	0.007
.*.*..	0.06	0.07	0.007
.***..	0.05	0.04	0.007

average standard deviation = 0.049



to indicate convergence. However, in time it became evident that this was not a reliable indicator of convergence. Now it is more common to do more than one independent run, perhaps each starting from different random trees, and to assess convergence based on agreement between runs. This agreement is measured with the PSRF (potential scale reduction factor) between numerical parameters and branch lengths, and with the average standard deviation of split frequencies (or split supports – ASDOSS). (The PSRF will not be described here.)

If we look at various splits from 2 runs, each run will have the split at similar but slightly different frequency. Each average frequency has a standard deviation.

The average standard deviation in split support (split frequency) summarizes the topological agreement between 2 runs in the form of a single number.

MrBayes now does 2 separate runs by default to encourage this strategy of assessing convergence based on agreement between runs. The cumulative average standard deviation of the difference in split supports (or split frequencies) (ASDOSS) between the two runs is calculated periodically and printed out. Since it is cumulative, it should approach zero. This is a good topology convergence diagnostic.

13 Bayes factors

Recall our version of Bayes' Theorem

$$P(RAS|\oplus) = \frac{P(\oplus|RAS)P(RAS)}{P(\oplus|RAS)P(RAS) + P(\oplus|healthy)P(healthy)}$$

The denominator here is the support times the prior for all hypotheses. It is the *marginal likelihood*. The Bayes factor B_{10} can be defined as the ratio of marginal likelihoods of 2 hypotheses $H1$ and $H0$. We can use Bayes factors to compare hypotheses (for us, models or trees).

$$\text{Bayes factor} = \frac{\text{marginal likelihood A}}{\text{marginal likelihood B}}$$

The problem is that for complex problems marginal likelihoods are difficult to compute; they need to be estimated. The "harmonic mean estimator" is a reasonably good, easily computed, but somewhat controversial way of estimating the marginal likelihood.

The harmonic mean of a series of numbers is

$$m_h = \frac{1}{[\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3} + \ldots + \frac{1}{x_n}] \times \frac{1}{n}}$$

eg the harmonic mean of 2 and 4 is

$$\frac{1}{[1/2 + 1/4] \times 1/2} = \frac{1}{3/8} = \frac{8}{3}$$

The harmonic mean estimator is the harmonic mean of likelihoods sampled from the posterior distribution. Because it is calculated from likelihoods, not log likelihoods, it is a little tricky to calculate because of numerical underflow (MrBayes and p4 will calculate it). It is generally presented in log form. We should be clear that Bayes factor (B_{10}) is ratio of marginal likelihoods, while the log Bayes factor is the difference between log marginal likelihoods. We can use the following table for interpretation of log Bayes factors (after Kass and Raftery, 1995, based on Jeffreys 1961).

$2\log_e(B_{10})$	$\log_e(B_{10})$	Evidence against H_0
0 to 2	0 to 1	Not worth more than a bare mention
2 to 6	1 to 3	Positive
6 to 10	3 to 5	Strong
>10	>5	Very strong

We can compare models and trees in a Bayesian framework using Bayes factors. To compare models, it is perhaps simplest to use a fixed tree topology in the Bayesian run, but allowing the branch lengths to be free. Turning off topology moves and turning on branch length moves is possible in MrBayes (v 3.1.2 using the `props` command interactively, and in v 3.2 using the `propset` command) and in p4. Comparing models this way might be faster and easier than comparing models using ML, for some models such as those with multiple data partitions. Runs can be short, as tree space is not explored. One run is done for each model being compared, or for each tree being compared, after which the estimated marginal likelihoods are compared.

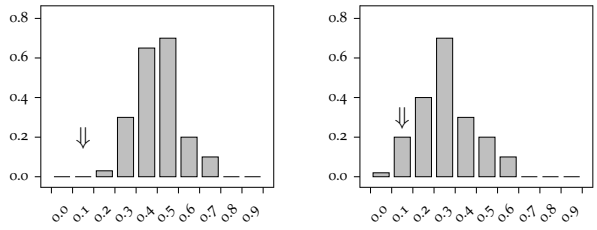
14 Model fit

In the section on model choice above, we touched on assessment of model fit, motivated using a scatterplot of X-Y data. With scatterplots and fitted lines, it is easy to

see if the model fits badly, but with phylogenetic data and models, it is not so obvious.

Assessing whether the composition part of the model fits the data is the easiest. One very easy way to do that is with the χ^2 test in the `basefreq` command in PAUP. This is a test for homogeneity of composition in the sequences. One problem with this test is that it assumes that the sequences are not related — and of course in phylogenetic data they certainly are related. Due to this, this test suffers from a high probability of Type-II error.

To assess the fit of other aspects (or of composition) of the model to the data, you can use simulations. The strategy is to find some aspect of your data that you measure, that is affected by your model. Then you measure that in your original data and in your simulations, and ask whether the original data point fits in the range generated by the simulations. In a Bayesian setting this is called *posterior predictive simulation*. Choosing a good test quantity is not trivial. Below the (hypothetical) test quantity is measured in the original data and in simulations using two different models. The test quantity for the original data is shown by the arrow, and of course is the same for both models, while the simulated values differ depending on the model. The test on the left shows a model that does not fit, while the model in the test on the right appears to fit for this test quantity.



❄