Maximum Likelihood and Bayesian Analysis in Molecular Phylogenetics

Peter G. Foster
Natural History Museum, London

Rio de Janeiro, March 2009

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.

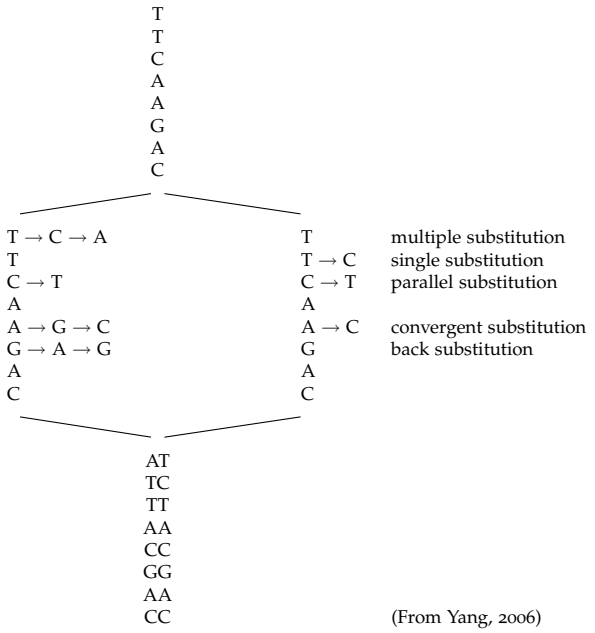
A few decades ago when gene sequences started to appear, the scientific community was comfortable with the idea of Darwinian evolution by random mutation followed by selection of the fittest, and it was assumed that process applied to molecular evolution as well. However, when the genes were examined, 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.

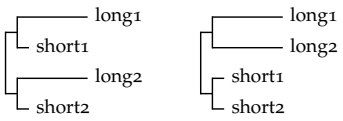
The approach taken by a parsimony analysis is that the history and underlying process of evolution is unknowable, and so we should not try to look for a common mechanism. So, for example, if two sequences in an analysis differ at half of their sites, the parsimony approach is to base conclusions only 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

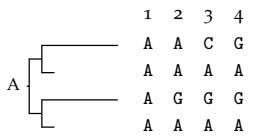
This is a well-known problem in phylogenetics, where unrelated long branches can end up being put together in a parsimony analysis.



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. An analogy is that of a busy car park in a city where there are only 4 colours of cars. Cars park in the car park for varying lengths of time and then leave, to be replaced immediately by another car. Over time, the car park colour composition will reflect the composition of the cars in the city. If there are more red cars in the city than blue ones, then that will happen in the car park as well. If for some reason the car park colour composition started out differently, all blue cars for example, over time the car park car colour composition would tend toward the city car colour composition. If we want to model the process we need to know the city car colour composition.

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". Thats 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.

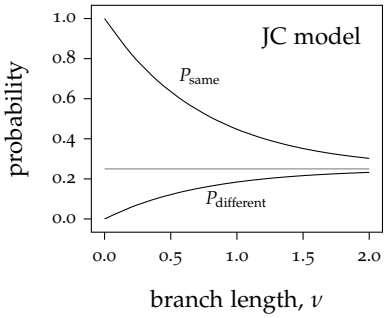
Now we will try to calculate the likelihood of a one-branch tree. The data will be 2 sequences, each 1 base long,

```
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

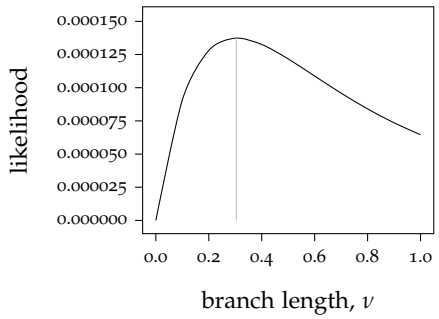
```
one  ccat
two  ccgt
```

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

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

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.

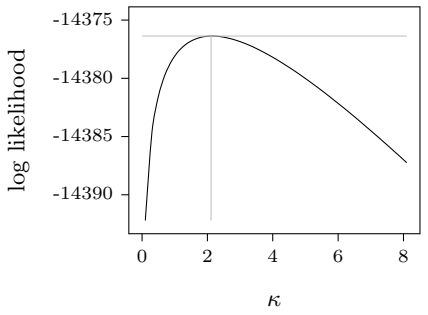
We can check that PAUP gets it right.

```
#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 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 analyses, or ML if you have the computational time.

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 modeled as a range of rates, and is described using only one parameter, α, the `shape` parameter in PAUP, as `lset rates=gamma shape=estimate;`.

Table 1: Simple DNA models.

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

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 empirical 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.

5 Gamma distributed among-site rate variation

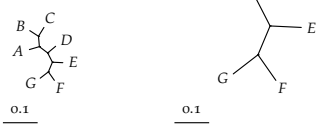
Just as there are fast genes and slow genes, there are fast sites and slow sites within genes. Sites differ in how much they are free to vary. A site may be under strong selection and highly constrained; other sites, such as third codon positions, might be relatively unconstrained.

If we could reliably separate the fast sites from the slow sites and analyse the two sets separately, ideally we would get the same tree topology, but the branch lengths would

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

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 analyse them separately from the third codon position.



Tree made using slow sites only | Tree made using fast sites only

We could then analyse 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 analysed with a one-size-fits-all branch length that is a compromise between the slow and fast rates, and fits neither one well.

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 analyse 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

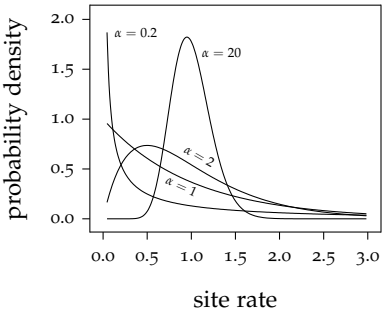


Figure 1: Gamma curves at various α values

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 centered 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 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 analyse 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 analyse these data on a particular tree with a JC model, the log likelihood

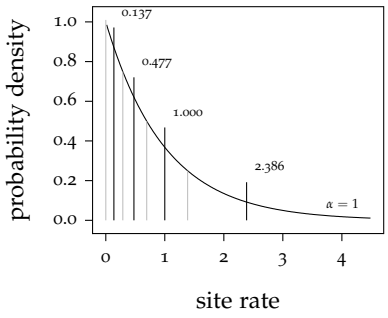


Figure 2: Discrete gamma divisions for $\alpha = 1$

```
 7  50
A         ttctaacgggacagtgcgcccactcacgcacctggtcactgtatgcgagt
B         tgcgaaggtgctattgcgagcattcacgcagatggtaactgtatgtgaga
C         tgcgaaggtgttattgcccacattcgcgcggaaggtaacactatgtgaga
D         tcccatagcgacatggcgcatactgactcctatggatactgtatgcgagt
E         tgcgaaggcgacattgcgcacattcacgcatatggttactgtacgtgagt
F         tgcgaaggcggcattgcgtccattcacccgcatggagactttatgtgaga
G         tgcgaagtcgacattacgctcttttacgcacagggtcactttatgggaca
          *         *
          13131232314212221234222132131332311241123112131122
```

Figure 3: Example data for GDASRV calculation

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.

6 A survey of some other models

6.1 Codon models

- For DNA sequences of codons of protein-coding genes
- Simplifications of 64×64 matrix, so far fewer parameters to estimate
- We have two kinds of change
  - Synonymous, where the aa does not change
  - Non-synonymous, where the aa changes

6.2 Identifying selection using codon models

- d_S is the number of synonymous substitutions per synonymous site
- d_N is the number of non-synonymous substitutions per non-synonymous site
- $\omega = d_N/d_S$ measures selection at the protein level
  $\omega = 1$: neutral evolution
  $\omega < 1$: purifying (negative) selection
  $\omega > 1$: diversifying (positive) selection

6.3 Other mixture models

- Gamma ASRV 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
  - heterogeneity of the rate matrix among sites
- Note that you do not divide the data
  - rather, for each site, you average over the possible states

6.4 Covarion model

- A generalization of the pInvar model
- A site can change from invariable to variable, and back
- Biochemically realistic
- A site at a given node in the tree can be either on or off
  - We cannot know whether it is on or off, so we evaluate assuming both
- Two parameters: off $\rightarrow$ on, and on $\rightarrow$ off

6.5 Tree-heterogeneous models

- Most models are homogeneous over the tree
- The process of evolution can and does differ over the tree
  - This is easy to see when homologous genes differ in composition
- It can lead to recovery of erroneous trees if you use a tree-homogeneous model (as most are)
- The heterogeneity over the tree can be modelled
  - Better fit of the model to the data
  - Better trees

6.6 Recoding data

- Some parts of sequences become saturated faster than others
- Saturated sequences are noisy and often biased and difficult to model
- One approach is to recode the data
- RY-recoding: A & G → R, C & T → Y
  - A 2-state model, transversions only
- Grouped aa's: C, STPAG, NDEQ, HRK, MILV, FYW
  - This allows a 6×6 rate matrix with free parameters

6.7 Modeling RNA stems

- We have `A:U`, `G:C`, and `G:U` stable pairs
- Also mismatches (`A:G` *etc*)
- Also reversals (*eg* `A:U` ≠ `U:A`)
- Recode into N-state models
  - 16: all combinations
  - 6: only stable pairs, ignore mismatches
  - 7: mismatches are lumped together as one state

6.8 Clock models and molecular dating

- There is obviously some truth in a clock-like behaviour of evolutionary change
  - Can we use that to date divergences?
  - Calibrate it with fossils
- The strict molecular clock rarely applies
- Relaxed clock models work better
- Many sources of error
- Widely used, but there is vocal debate over its validity

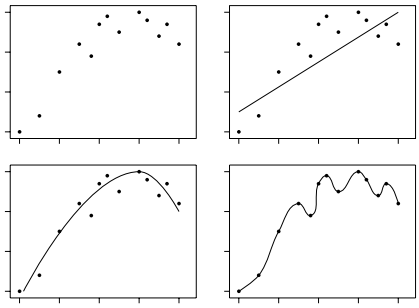
7 Choosing a model

- Generally, don't "assume" a model
- Rather, find a model that best fits your data.

7.1 Parameters

- Models differ in their free, *ie* adjustable, parameters
- More parameters are often necessary to better approximate the reality of evolution
- The more free parameters, the better the fit (higher the likelihood) of the model to the data. (Good!)
- The more free parameters, the higher the variance, and the less power to discriminate among competing hypotheses. (Bad!)
- We do not want to "over-fit" the model to the data

7.2 What is the best way to fit a line (a model) through these points?



7.3 Choosing a phylogenetic model by ML

- Start with a reasonably good tree
  - The neighbor-joining tree will do fine
- Evaluate the likelihood (ML for that tree) for all the models that you want to test
- No tree search involved, so it is fast
- Choose the best model
  - With enough parameters, but not too many
  - This may be the ML model, or not

7.4 Choosing a model

- 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.
  - So if adding a parameter increases the log likelihood by more than 1, it is not useless.

7.5 A table of AIC values

	$\ln L$	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	

- We penalize each likelihood by the number of free parameters
- By the AIC, GTR+G is the best model, even though the GTR+IG had a higher likelihood

7.6 How many parameters?

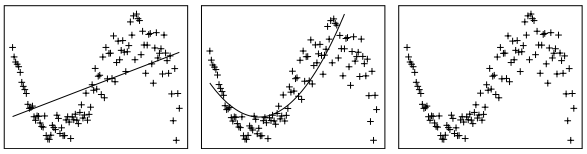
Model	rate matrix parameters	composition parameters	ASRV parameters	total parameters
JC69	0	0	0	0
F81	0	3	0	3
HKY+G	1	3	1	5
TN93	2	3	0	5
GTR+IG	5	3	2	10
JTT+F	0	19	0	19
WAG+IG	0	0	2	2

7.7 Modeltest

D. Posada and K. A. Crandall 1998. "MODELTEST: testing the model of DNA substitution" *Bioinformatics* 14: 817-818.

- Automates the process of choosing a model
- Uses PAUP to do the likelihood calculations
- Uses the AIC and the likelihood ratio test
- Site-specific rate variation and clock models are *not* covered
- See also Prottest and Modelgenerator

7.8 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 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 Simulating evolution

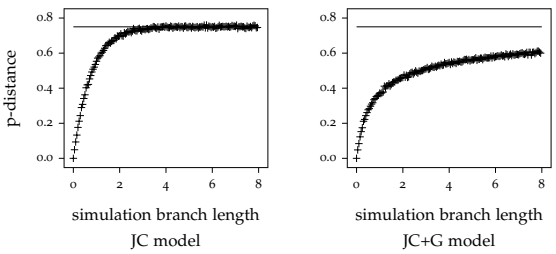
A model is your idea of the way that you think that things work. Armed with this it is possible to simulate sequences and evolve those on a tree to simulate evolution. This can be useful to test the methods and models. When you simulate data you simulate on a given model and on a given tree, and since 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.



9 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 practice.

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.

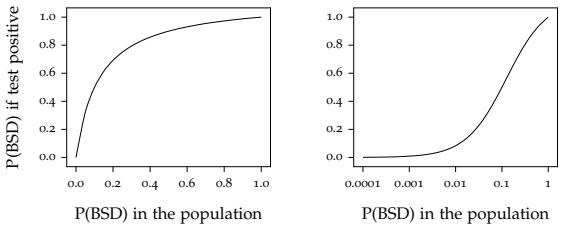
Another difference is that Bayesian analysis explicitly relies on *prior probabilities*. The prior probability might be known with confidence, but for many problems we have only a vague idea of what the prior probabilities are, and since a Bayesian analysis forces us to be explicit about them we may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

9.1 Rare diseases and imperfect tests

Lets say that we are testing for a disease – Bad Spelling Disease, or BSD, and we know that 1% of the population suffer from it. We have a test for BSD that is fairly accurate – if you suffer from BSD 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 BSD. 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 BSD sufferers in those 108 people. So the probability of having BSD if you test positive is only 9/108, about 8%. Thats all!

The surprisingly low probability depends mostly on the low background frequency of BSD in the population. That is the *prior probability* of BSD, that is the probability that you would expect somebody to have BSD *before* you see the result of the test. The probability of somebody having BSD 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.

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 BSD 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 spelling 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 BSD 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.

$P(BSD|\oplus)$ = $\frac{P(\oplus|BSD)P(BSD)}{P(\oplus)} = \frac{0.009}{0.108}$

= The posterior probability of having BSD given a positive test.

$P(\oplus|BSD)$ = The likelihood. The probability of a positive test if you have BSD. The probability of true positive tests, 0.9

$P(BSD)$ = The prior probability of BSD, 0.01

$P(\oplus)$ = The *marginal likelihood*. The probability of getting the data, a positive test, under all circumstances. That would include true positives and false positives.

= $P(\oplus|BSD)P(BSD)+$
$P(\oplus|healthy)P(healthy)$

= $(0.9 \times 0.01) + (0.1 \times 0.99)$

= 0.108

$P(\oplus|healthy)$ = The probability of a positive test if you are healthy. The probability of a false positive, 0.1

$P(healthy)$ = The prior probability of not having BSD, $1 - P(BSD) = 0.99$

9.2 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 BSD in the population. While most studies place the incidence of BSD 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 spelling 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 BSD, 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 BSD — hardly a satisfactory answer.

The test for BSD might involve nuisance parameters such as the age or background of the patient that might affect their ability to spell, and we can formulate a model involving these parameters to allow us to calculate the likelihood of the data in the BSD 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.

10 Bayesian phylogenetic analysis in practice

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. 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.

10.1 Reference

Mark Holder and Paul O. Lewis. 2003. Phylogeny estimation: Traditional and Bayesian approaches. Nature Reviews Genetics 4: 275–284.

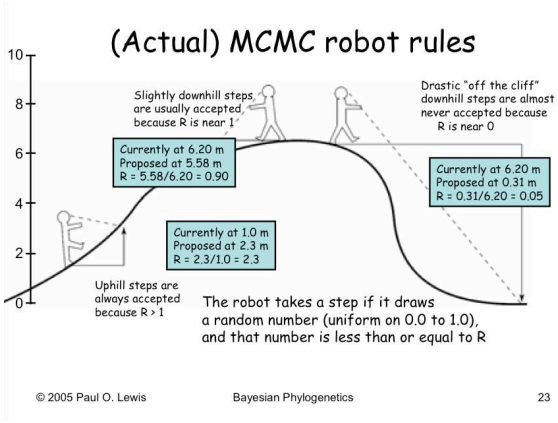
10.2 History of Bayesian analysis in phylogenetics

- The first papers and demonstration programs for 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
- p4, BEAST, BayesPhylogenies, Phycas, PhyloBayes

10.3 Bayesian analysis in practice

- It is practically impossible to do Bayesian calculations analytically
- Rather, the posterior probability density is approximated by the *Markov chain Monte Carlo* (MCMC) method



10.4 Markov chain Monte Carlo

- 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 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 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

10.5 How the MCMC works

- Start somewhere
  - with defined model, topology, branch lengths
  - That "somewhere" will have a likelihood and a prior
  - Not the optimized, maximum likelihood
- Randomly propose a new state
  - Maybe adjust one of the branch lengths
  - If the new state has a better likelihood $\times$ prior, the chain goes there

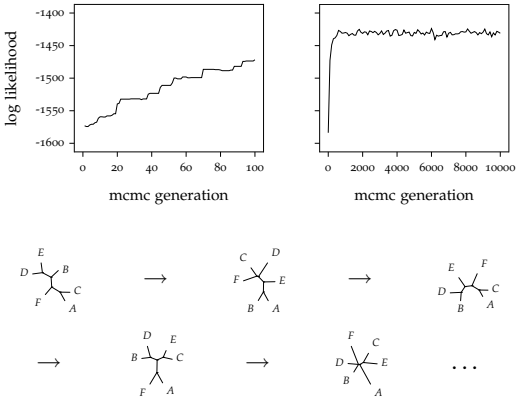
10.6 If the proposed state has a worse probability

- Calculate the posterior probability ratio between the current and the proposed states. That ratio will be between 0 and 1.
- Choose a random number between 0 and 1. If the random number is less than the the likelihood ratio of the two states, then the proposed state is accepted.
- If the likelihood of the proposed state is only a little worse, it will sometimes be accepted
- This means that the chain can cross likelihood valleys

10.7 State → state → state → ...

- Proposals are made, sometimes accepted, often rejected
- Branch lengths, topology, and model parameters change
- We save samples to digest later
- The start of the chain is random and poor, so the first proposals tend to make the probability much better quickly.

10.8 MCMC likelihood plot



10.9 MCMC burn-in

- The chain only works properly after it has converged or equilibrated
- 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
  - Only use the samples from the plateau
  - This is a widely-used but unreliable way of assessing convergence

10.10 MCMC result

- Sampled trees are written to a file during the chain
- 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 (after burn-in) 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

10.11 Splits, *aka* tree bipartitions

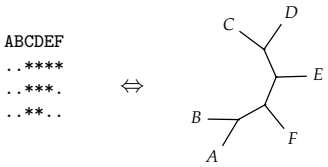
- We can represent trees as splits, in 'dot-star' notation.

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

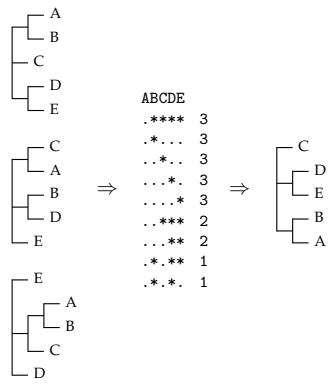
- By convention the first position is always a dot
- Terminal branches may or may not be included

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

10.12 Splits ⇔ trees



10.13 Making a consensus tree from splits



10.14 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 distribution
- Or perhaps the proposals are too near the current state, causing *autocorrelation*, which decreases the *effective sample size*
- Rather than sampling the chain at every generation, the chain is sampled more rarely, *eg* every 10 or every 100 generations.
- These sampled states will more likely be different from each other, and so be more useful.

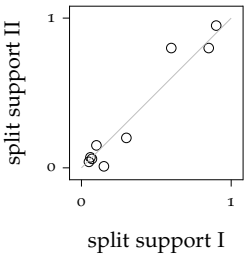
10.15 Assessing convergence

- Commonly:
  - plot the likelihood
  - plot other parameters
  - Can be an unreliable indicator of convergence
- Do multiple runs, starting from different random trees and assess agreement
  - PSRF, potential scale reduction factor
  - ASDOSS, averaged standard deviation of split support (or split frequency)

10.16 Two runs

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



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

10.17 ASDOSS

- MrBayes now does 2 separate runs by default to encourage this strategy
- 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.
- This is a good topology convergence diagnostic. But how low is low enough?
- Where it stabilizes depends on the data

10.18 Prset

- The prior probability should usually be a distribution
  - Uniform
  - Exponential
- Can also be fixed to single values
  - Generally not a good idea
  - Generally we fix the rate matrix for proteins, *eg* `prset aamodelpr=fixed(wag)`

10.19 Acceptance rates

```
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*
- 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
- Rule of thumb— aim for 10% – 70% acceptance for mixing.
- Can change tuning parameters in MrBayes with `props`

10.20 MCMCMC

- MrBayes introduced Metropolis-coupled MCMC
- Several chains are run in parallel
- All but one is "heated"
  - Increases the acceptance probabilities
  - 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
- Only the cold chain is sampled

11 Bayes factors

11.1 Bayes' theorem

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

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalizing constant}}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

11.2 Bayes Factor

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

But marginal likelihoods are difficult to compute. They need to be estimated.

11.3 The harmonic mean estimator

- The HME is a reasonable, easily computed, somewhat controversial way of estimating the marginal likelihood.
- The harmonic mean is

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

- *eg* the harmonic mean of 2 and 4 is

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

11.4 The harmonic mean estimator

- HME is the harmonic mean of likelihoods sampled from the posterior distribution.
  - likelihoods, *not* log likelihoods
- Then converted to log form
- Tricky to calculate because of numerical overflow
- MrBayes and p4 will calculate it

11.5 Using Bayes factors

- Bayes factor (B_{10}) is ratio of marginal likelihoods
- Log Bayes factor is difference of log marginal likelihoods

$2\log_e(B_{10})$	$\log_e(B_{10})$	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

- Use this to compare models in a Bayesian framework

12 Model fit

12.1 How to assess whether the model fits?

- Composition – often easy to show that the model does not fit
- Other aspects – can use simulation

12.2 Composition fit

- We ask whether the composition part of the model fits the data
- Eg the chi-square test in the `basefreq` command in PAUP
  - Read in some data
  - `basefreq`
  - Does a chi-square test for homogeneity of composition in the sequences
  - This test assumes that the sequences are not related
  - suffers from a high probability of Type-II error.

12.3 Composition fit

```
paup> basefreq

Base frequencies:

Taxon              A         C         G         T     # sites
-------------------------------------------------------------------
A            0.11000   0.21500   0.25500   0.42000         200
B            0.11000   0.22500   0.28000   0.38500         200
C            0.11000   0.22000   0.27000   0.40000         200
D            0.15000   0.21500   0.26000   0.37500         200
E            0.12500   0.23500   0.27000   0.37000         200
F            0.12500   0.21500   0.29000   0.37000         200
G            0.12500   0.22500   0.26000   0.39000         200
H            0.13000   0.23000   0.25500   0.38500         200
-------------------------------------------------------------------
Mean         0.12312   0.22250   0.26750   0.38688      200.00


Chi-square test of homogeneity of base frequencies across taxa:

Taxon                  A         C         G         T
-----------------------------------------------------------
A           O         22        43        51        84
            E      24.62     44.50     53.50     77.38
B           O         22        45        56        77
            E      24.62     44.50     53.50     77.38
C           O         22        44        54        80
            E      24.62     44.50     53.50     77.38
D           O         30        43        52        75
            E      24.62     44.50     53.50     77.38
E           O         25        47        54        74
            E      24.62     44.50     53.50     77.38
F           O         25        43        58        74
            E      24.62     44.50     53.50     77.38
G           O         25        45        52        78
            E      24.62     44.50     53.50     77.38
H           O         26        46        51        77
            E      24.62     44.50     53.50     77.38

Chi-square = 4.320890 (df=21), P = 0.99996131
Warning: This test ignores correlation due to phylogenetic structure.
```

12.4 Fit of other aspects of the model

- You can use simulations
- Pick some aspect of your data that you measure
  - in your original data
  - in your simulations
- Ask whether the original data point fits in the range generated by the simulations

12.5 Using simulations to assess model fit