**PhyloMath Lecture 12**
**by Jadranka Rota, April 20, 2004**

We continued our discussion of the Metropolis algorithm. Before we try applying what we did last time to a phylogeny, we need to remember that we deal here with two matrices: a transition matrix ($P_{ij}$) and a proposal matrix ($Q_{ij}$).

$P_{ij}$ = Pr(proposing and accepting state $j$ | currently in state $i$)

This is a joint probability because the state $j$ has to be both proposed and accepted in order to actually become the state at the next step. It is also conditional on starting with state $i$ at the current time. Another way of writing this probability is this

$P_{ij}$ = Pr(proposing $j$ | currently at $i$) Pr(accept $j$ | propose $j$, currently at $i$)

The first term is the proposal probability ($Q_{ij}$) and the second term is the probability of accepting that state given that it was proposed ($\min\left\{1, \pi_j / \pi_i\right\}$).

Here is our proposal matrix. It is not the only possible proposal matrix, but has the virtue of being simple and promoting mixing by always proposing a different state than the current state. We can have zeros in the diagonal here, but not in the transition matrix (see notes for Lecture 11).

$$\{Q_{ij}\} = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \end{array}$$

Then we calculate the transition matrix following the procedure that was outlined last time.

$$\pi_A = \frac{4}{7} \qquad \pi_B = \frac{1}{7} \qquad \pi_C = \frac{2}{7}$$

$$\{P_{ij}\} = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 5/8 & (1/2)\min\left\{1, \frac{1/7}{4/7}\right\} = 1/8 & (1/2)\min\left\{1, \frac{2/7}{4/7}\right\} = 1/4 \\ (1/2)\min\left\{1, \frac{4/7}{1/7}\right\} = 1/2 & 0 & (1/2)\min\left\{1, \frac{2/7}{1/7}\right\} = 1/2 \\ (1/2)\min\left\{1, \frac{4/7}{2/7}\right\} = 1/2 & (1/2)\min\left\{1, \frac{1/7}{2/7}\right\} = 1/4 & 1/4 \end{bmatrix} \end{array}$$

Every row has to add to one, so the diagonal elements are calculated by subtracting the sum of the other two (off-diagonal) elements from one. Here we can see why the denominator (7) in the calculation of the target distribution (i.e. $\{\pi_A, \pi_B, \pi_C\}$) doesn't matter. It simply cancels out when we do the calculations, so we don't need to know its exact value.

Note added by Paul:
In Bayesian uses of MCMC, the denominator is the marginal probability of the data, and often involves a multiple integral and/or a very large sum. For example, in the case of phylogenies, the denominator would involve a sum over all possible topologies and a multiple integral over all possible branch lengths, kappa values, etc. Ugly, to say the least.

---

If the chain has a limiting distribution, and it has converged to that limiting distribution, then it now exhibits stationarity and the following should be true

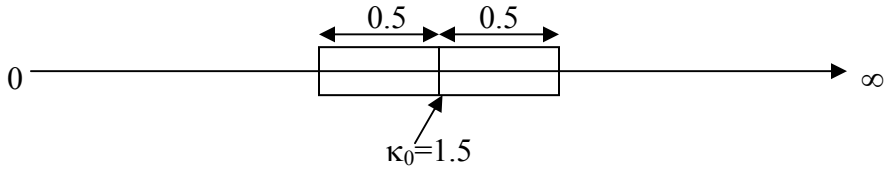
$$\pi_A = \pi_A P_{AA} + \pi_B P_{BA} + \pi_C P_{CA}$$

The $\pi_i$ values on the right-hand side are the frequencies at time $t$, whereas the $\pi_A$ on the left side applies to time $t+1$, but if the chain is stationary, then these relative frequencies stay the same from one step to the next and we can equate the values from time $t$ to those from time $t+1$.

By entering our values from the matrix, we get

$$\frac{4}{7} = \left(\frac{4}{7} \times \frac{5}{8}\right) + \left(\frac{1}{7} \times \frac{1}{2}\right) + \left(\frac{2}{7} \times \frac{1}{2}\right) = \frac{4}{7}$$

So how is this applied to phylogeny?




By choosing a random value from this window, we propose a new value for κ. Every value in this window has the same probability (there is an infinite number of values in this window, all of them with equal weight) and everything outside of it has probability zero of being proposed. All we need to do is calculate where we will go next. MrBayes uses this same procedure for any parameter that has a range $0 \to \infty$.

Note added by Paul:
While this may seem different than applying the Qij matrix to determine the proposal distribution in the case of our discrete example, it is in fact quite similar except for the fact that in the case of proposing a new κ, the number of possibilities is infinite whereas we only had three possible values in the discrete example. Consider the third row of the Qij matrix (i.e. pretend that the chain is currently sitting on state C). The "window" in this case corresponds to the set {A,B}, and all values in this set have an equal chance of being proposed, just like all values in the window around κ have an equal chance of being proposed. Also, there are some values (namely C) that have zero chance of being

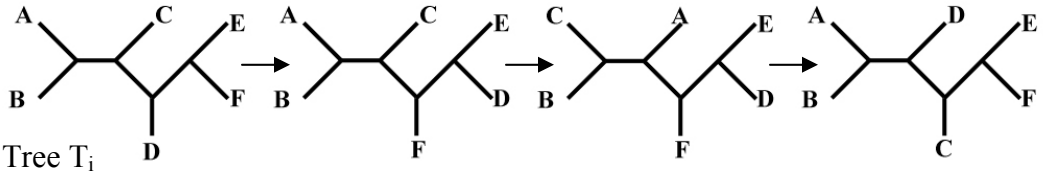
proposed, just like values outside the κ window have no chance of being proposed. The method used for proposing new k values is analogous to having a Qij matrix with an infinite number of rows and columns. We only need to concentrate on one row at a time because the proposal distribution is conditional on the current value of κ.

---

The width of the window somewhere close to 1 would be good for κ. (Note from Paul: you can find the sliding window size for any parameter that uses this method in MrBayes by typing "props" at the MrBayes prompt). If we are proposing values very far from where we are, we will be wasting a lot of time because most of our values will not be accepted, so we will be staying at the same place most of the time. We want there to be a lot of mixing, meaning that we are moving over the parameter space. This procedure makes it possible to choose every possible value if we run this for an infinite amount of time.

This method works with any ordinary continuous parameter, but how do we go about proposing and accepting a tree?

In this case our Markov chain will be a sequence of trees and we measure the posterior probability of clades.

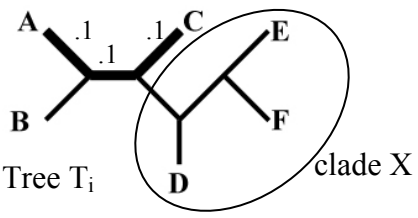



Tree $T_i$

1 0 0 1

For example we are looking at the clade that contains taxa E and F in the tree $T_i$. When that is true, we mark it with 1 and when it isn't, we mark it with 0. Estimate of the posterior probability of that clade in this case is 50% because it is found in two of our four trees. Of course, in a real analysis we would run this for millions of steps, not just four.

How do we go about proposing and accepting changes like this to a tree arrangement? Larget and Simon (Larget, B., and D. L. Simon. 1999. Markov chain monte carlo algorithms for the Bayesian analysis of phylogenetic trees. Molecular Biology and Evolution 16:750-759) came up with a method that allows us to do that. They called this their LOCAL move, and this is the name that MrBayes uses to describe it.




Tree $T_i$

Let's look at the tree $T_i$. A three-branch segment of the tree is chosen randomly. In this tree we have chosen the segment that is bold. Each of those three branches is 0.1 in length, so the segment length is 0.3. We are calling this quantity $v_0$ ($v_0$=0.3). The new length of our segment is $v$.

$$v = v_0 e^{\lambda(u-1/2)}$$

$\lambda$ in this equation is a tuning parameter: the larger it is, the more drastic the proposed change is. We will use $\lambda = 1$ in the example that follows.

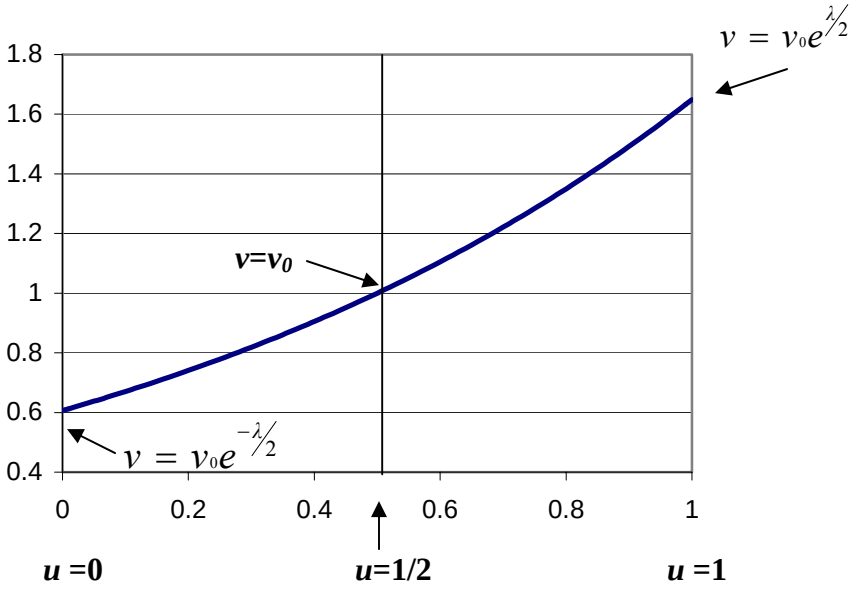
Note from Paul:
Note how the new length depends on the current length. This keeps us from routinely proposing values that are rejected. Baby steps are the rule in MCMC.

---

$u$ is a uniform(0,1) random variable.

The tree $T_i$ is our current position. Now we are going to make a "local move."

Step 1: expand or contract $v_0$ to give us new length for segment $v$.




Example: we picked a random number $u$=0.897. From this we can calculate $v$.
$v = 0.3e^{1(0.897-0.5)} = 0.446$. Since $v$ is greater than $v_0$, our segment has expanded. Each one of the three branches grows proportionately.

Step 2: randomly choose B or clade X to "prune" (euphemism for "chop off") from $T_i$. We will draw another uniform(0,1) random variable $u$ to make the decision.

If $u \leq 1/2$, choose B.

If $u \geq \frac{1}{2}$, choose X.

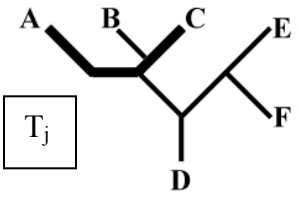
Random number drawn is $u$=0.222, so we chose B.

Step 3: reattach B uniformly on the (recently expanded) bold segment (uniformly means that we will choose a spot somewhere along that segment randomly).

The probability that the tree topology will change is 0.333 because the tree topology will change only if the new branch gets attached somewhere on the final third of our segment. Remember the three branches composing the bold segment were originally equal in length, and they maintain their relative proportions when expanded or contracted.

We will use a random number to find out how far we need to go down that segment.

Random number is $u$=0.775, so we are attaching B 77.5% of the way from A to C and thus we have indeed changed the tree topology. Let's call the new tree $T_j$.




This is the proposed tree. Now we need to decide whether we are going to accept it or not.

$$\Pr(\text{accept } T_j) = \min\left\{1, \frac{\pi_j}{\pi_i}\right\}$$

When using the LOCAL move algorithm, if there is a change in the topology, it is always equivalent to a "nearest neighbor interchange" type of rearrangement (or NNI).

MrBayes uses this algorithm for proposing and accepting new trees about 30-40% of the time. (Note from Paul: you can increase or decrease the rate using the "props" command in MrBayes). Some of the time it uses the so-called worm move, which differs from the LOCAL move in that the bold segment stretches all the way across the tree. That way a much greater change to the tree is often proposed. That can help us explore the whole landscape better, but one would probably not want to use the worm move exclusively because it is expected to have a higher rejection rate.

The LOCAL proposal algorithm that we used in this calculation is not symmetric, so the probability of going from $T_i$ to $T_j$ is not the same as going from $T_j$ to $T_i$. To fix this problem, Hastings (1970) made an addition to the Metropolis algorithm. He made it possible to use an asymmetric proposal matrix and still have it work correctly.

For example, here is an asymmetric proposal matrix:

$$\{Q_{ij}\} = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \end{array}$$

Original Metropolis algorithm: $P_{ij} = Q_{ij} \min\left\{1, \frac{\pi_j}{\pi_i}\right\}$

Metropolis-Hastings algorithm: $P_{ij} = Q_{ij} \min\left\{1, \frac{\pi_j \times Q_{ji}}{\pi_i \times Q_{ij}}\right\}$. The quantity $\frac{Q_{ji}}{Q_{ij}}$ is called the Hastings ratio.

Simple example: $\{Q_{ij}\} = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{bmatrix} 1/3 & 2/3 \\ 1/3 & 2/3 \end{bmatrix} \end{array}$

First we are going to show that the following is not true if we employ the Metropolis algorithm.

$$\pi_A = \pi_A P_{AA} + \pi_B P_{BA}$$

$$\pi_A = 4/5$$
$$\pi_B = 1/5$$

$$\{P_{ij}\} = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{bmatrix} 5/6 & (2/3)\min\left\{1, \frac{1/5}{4/5}\right\} = 1/6 \\ (1/3)\min\left\{1, \frac{4/5}{1/5}\right\} = 1/3 & 2/3 \end{bmatrix} \end{array}$$

$$\frac{4}{5} = \frac{4}{5} \times \frac{5}{6} + \frac{1}{5} \times \frac{1}{3} = \frac{11}{15}$$
$$\frac{4}{5} \neq \frac{11}{15}$$

With this we have shown that if $Q$ matrix is not symmetrical, the Metropolis algorithm does not work.

However, using the Metropolis-Hastings algorithm fixes the problem.

$$\{P_{ij}\} = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{bmatrix} 11/12 & (2/3)\min\left\{1, \frac{1/5 \times 1/3}{4/5 \times 2/3}\right\} = 1/12 \\ (1/3)\min\left\{1, \frac{4/5 \times 2/3}{1/5 \times 1/3}\right\} = 1/3 & 2/3 \end{bmatrix} \end{array}$$

$$\frac{4}{5} = \frac{4}{5} \times \frac{11}{12} + \frac{1}{5} \times \frac{1}{3} = \frac{11}{15} + \frac{1}{15} = \frac{4}{5}$$

At the end of the class Paul demonstrated his MC Robot program. You can download the MCRobot program from http://lewis.eeb.uconn.edu/lewishome/software.html (note that it is a Windows program, however, so Mac users will need VirtualPC or some other Windows emulator if you want to play with it on a Mac).