PhyloMath
by Hilary McManus
4/13/04

Topics covered
-MCMC (Metropolis algorithm)
-Codon Models
-Robinson et al. (2003) Codon Model

Metropolis Algorithm

Revisiting last week's lecture, we went over simulating a Markov chain and seeing if we could approximate the given posterior probabilities (values in bottom row) of 3 hypotheses (or possible states), A, B and C.

A	*B*	*C*
.80	.15	.05

Simulated chain: …AAABAAAACA…

While this example is trivial, we usually have many more hypotheses than 3 to consider, and thus we generally cannot do an exhaustive search and calculate the posterior probabilities for all of them. Also, we often can relatively easily compute the numerator but not the denominator of the posterior probability. In such cases, MCMC simulation can yield an approximation to these probabilities rapidly.

The following can be estimated based on the small sample (chain) above:

$$A: \ \tfrac{8}{10} \ \ \hat{\pi}_A = 0.8$$
$$B: \ \tfrac{1}{10} \ \ \hat{\pi}_B = 0.1$$
$$C: \ \tfrac{1}{10} \ \ \hat{\pi}_C = 0.1$$

Even with this small sample of 10, the estimates (0.8, 0.1, 0.1) are quite close to the true distribution (0.8, 0.15, 0.05).

To determine what step the chain takes next, A, B, or C, we can set up a transition probability matrix with probability values of moving to a particular state from the current state. (ex. P_{AB} specifies the probability of being in state B after the next step given that you are in state A currently)

Single step transition probability matrix (matrix of all possible values):
$\{P_{ij}\}$ =

	A	B	C
A	P_{AA}	P_{AB}	P_{AC}
B	P_{BA}	P_{BB}	P_{BC}
C	P_{CA}	P_{CB}	P_{CC}

The sum of each row = 1.0

We want to design a chain that is **stationary** (probability of any given state does not change over time) so that we can more accurately estimate the posterior probabilities of different states. If we designed a chain that was not stationary, the probabilities that we seek to estimate would be moving targets!

Note added by Paul:
The point I made about time-reversibility and stationarity is not valid. While it is true that the Metropolis algorithm creates Markov chains that are both time-reversible and stationary, it is possible to find examples of Markov chains that are time-reversible but *not* stationary. For example, consider the simple two-state Markov chain with this transition matrix:

$$\{P_{ij}\} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

When currently at state A, this chain must (with probability 1) go next to state B. When currently at B, the chain must go to state A. It thus alternates between the two states with perfect regularity:

ABABABABABABABABABABABABABA...

We can easily show that this chain is time-reversible. Suppose we start (at time 1) in state A and assume time t is an odd numbered time (t+1 is thus even).

$\pi_A^{(t)} P_{AB} = (1)(1)$ (going forward in time here)
$\pi_B^{(t+1)} P_{BA} = (1)(1)$ (going backward in time here)

The above shows that starting in state A at time t and ending up at state B at time t +1 has the same probability (i.e. 1) of starting at state B at time t +1 and ending up at state A at time t. The same is true if t is an even numbered time, or if we start at B instead of A, etc.

This chain is thus time-reversible but it is *not* stationary. At odd-numbered times the distribution of states is guaranteed to be (1,0) and at even numbered times it is guaranteed to be (0,1) (vice versa if it is started in state B at time 1). Thus, the probability of being in state A (or state B) alternates between 0 and 1 each step – this is not stationarity!

It turns out that to have a limiting distribution, the transition matrix must have the following two features:
1) it must make possible some path between any state and any other state
2) the diagonal elements (i.e. P_{AA}, P_{BB}, etc.) cannot all be zero

The transition probability matrix for the Markov chain above that has alternating states passes requirement (1) but fails requirement (2).

The transition probability matrix used in the Metropolis algorithm is shown below:

$$P_{ij} = \begin{cases} Q_{ij}(1) & \text{if } \pi_j > \pi_i \\ Q_{ij}\dfrac{\pi_j}{\pi_i} & \text{if } \pi_j \le \pi_i \end{cases}$$

What is Q_{ij}? Q_{ij} is an arbitrary transition probability matrix with few restrictions, namely the first condition of stationarity listed above for transition matrices. The second condition (diagonal elements all non-zero) is never a problem because the diagonal elements are always set by subtraction to make the row sums all 1.

Example of a **bad** Q_{ij} proposal probability matrix:

	A	*B*	*C*
A	1.0	0	0
B	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
C	0	0	1

This is a bad matrix because A and C are absorbing states. Since $P_{AA}=P_{CC}=1$, once they are 'hit', the chain will never leave. Paul made the analogy that absorbing states are like being bankrupt when gambling. The matrix is not required to be symmetric, but one is technically using the Metropolis-Hastings algorithm (rather than the Metropolis algorithm) if it is asymmetric.

Here is an **acceptable** Q_{ij} proposal probability matrix:

	A	*B*	*C*
A	0	$\frac{1}{2}$	$\frac{1}{2}$
B	$\frac{1}{2}$	0	$\frac{1}{2}$
C	$\frac{1}{2}$	$\frac{1}{2}$	0

The probability values of 0 encourage the chain to mix well because a different state will always be proposed.

Using the definition of P_{ij} above, it turns out that our Markov chain is time-reversible. This is because we can show that the following equation holds:

$$\pi_i P_{ij} = \pi_j P_{ji}$$

To show that the above equation holds in general, we must show that it holds in two specific situations. Case 1 is when the stationary probability of the ending state *j* is greater than or equal to the probability of the starting state *i*. Case 2 is the opposite: the probability of the starting state *i* is larger than the probability of the ending state *j*.

Case 1 : $\pi_j \geq \pi_i$

$$\pi_i P_{ij} = \pi_i Q_{ij} \min\left\{1, \frac{\pi_j}{\pi_i}\right\}$$
$$= \pi_i Q_{ij} (1)$$

$$\pi_j P_{ji} = \pi_j Q_{ji} \min\left\{1, \frac{\pi_i}{\pi_j}\right\}$$
$$= \pi_j Q_{ji} \left(\frac{\pi_i}{\pi_j}\right)$$
$$= Q_{ji} \pi_i$$

and since the matrix is symmetrical, $Q_{ji} = Q_{ij}$.....

$$Q_{ji} \pi_i = Q_{ij} \pi_i$$

Case 2 : $\pi_j \leq \pi_i$

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

$$\pi_j P_{ji} = \pi_j Q_{ji} \min\left\{1, \frac{\pi_i}{\pi_j}\right\} = \pi_j Q_{ji} (1)$$

$$\pi_j Q_{ij} = \pi_j Q_{ji}$$

Note added by Paul:
With our simple example, there are only three states and it is easy to just calculate the entire transition probability matrix for our Markov chain. Using the Q_{ij} matrix defined above (wherein every off-diagonal cell has the value 1/2) and the known probabilities of the three states (0.8 for A, 0.15 for B, and 0.05 for C), the P_{ij} matrix becomes:

$$\{P_{ij}\} = \begin{array}{c|ccc} & A & B & C \\ \hline A & \frac{7}{8} & \frac{3}{32} & \frac{1}{32} \\ B & \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ C & \frac{1}{2} & \frac{1}{2} & 0 \end{array}$$

You should check to make sure you can get these same values.

Simulating a Markov chain using the Metropolis algorithm

The goal is to simulate a Markov chain using the Metropolis algorithm. Choose state C (arbitrarily) as the starting state and assume the following proposal matrix:

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

First step

Because we are starting in state C, we use only the third row of Qij. This third row defines the proposal distribution conditional on the current state. Set up a table to decide which of the two possible hypotheses (A or B) will be proposed as the next state given a uniform random number:

If uniform falls in this interval...	choose this state...
0.0 to 0.5	A
0.5 to 1.0	B

The random uniform value 0.122 was chosen in class, leading to the choice of A as the proposed state. We must now check to make sure A is accepted before we declare that it is indeed the next state.

$$\Pr(\text{accepting A}) = \min\left\{1, \frac{\pi_A}{\pi_C}\right\} = \min\left\{1, \frac{.8}{.05}\right\} = \min\{1, 16\} = 1$$

The probability of accepting A is 1.0, so A becomes the next state. The Markov chain now looks like this:

$$C \rightarrow A$$

Second step

Now what will be the next proposed step, and will this proposed step be accepted? From A, the proposal distribution dictates the following rule for deciding the state to be proposed next:

If uniform falls in this interval...	choose this state...
0.0 to 0.5	B
0.5 to 1.0	C

The random number drawn in class was 0.512, so C is the next state proposed.

$$\Pr(\text{accepting C}) = \min\left\{1, \frac{\pi_C}{\pi_A}\right\} = \min\left\{1, \frac{.05}{.8}\right\} = \min\{1, 0.0625\} = 0.0625$$

Because this time there is some uncertainty about whether to accept the proposed state, we must draw another uniform random number and use the following table to decide the outcome:

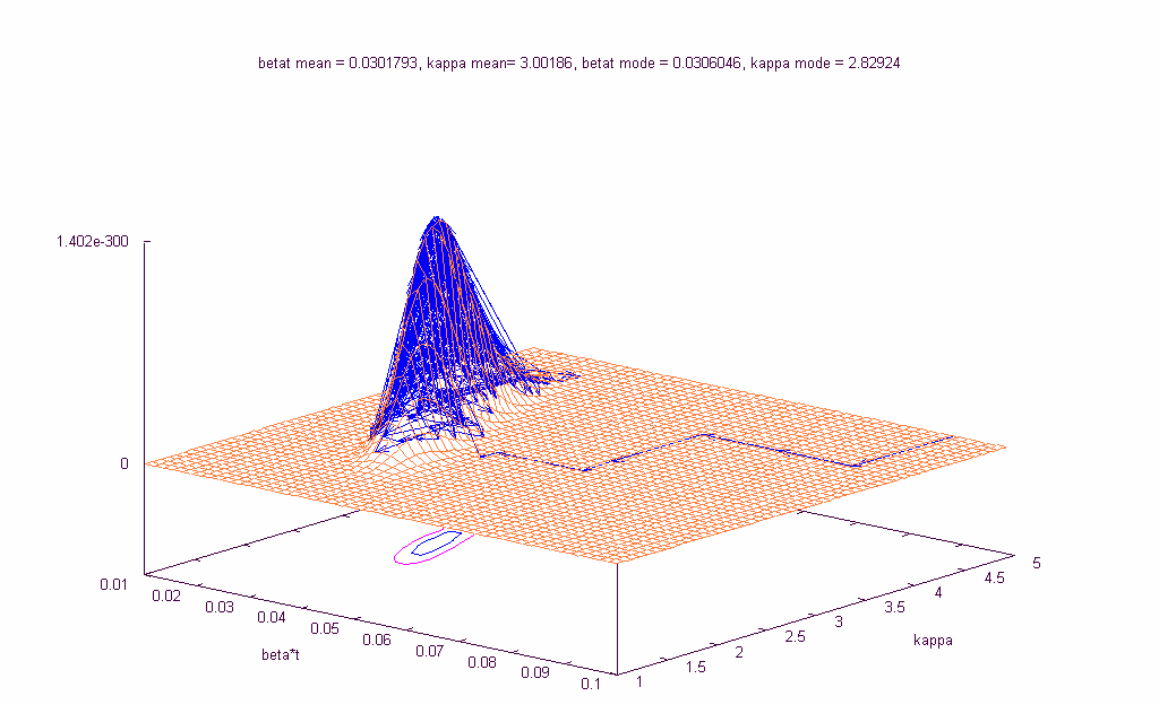
If uniform falls in this interval...	do this...
0.0 to 0.0625	accept proposed state
0. 0625 to 1.0	reject proposed state

The random value chosen was 0.862, so the proposed state C is rejected. This means that the Markov chain must *stay in its current state* A one time step. The Markov chain thus looks like this after the second step:

$$C \rightarrow A \rightarrow A$$

Real example

To visualize how these calculations work given a problem with two parameters and many more possible states, Paul showed us a simulation of the MCMC chain moving through posterior distribution space of two parameters, κ and β. The chain began in the corner where the posterior probability is close to 0.

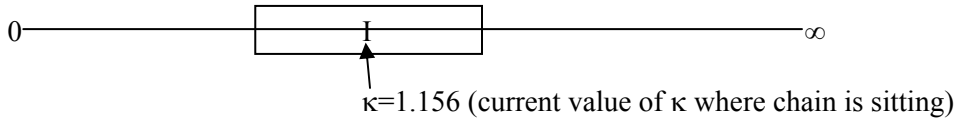


The MCMC chain moved fairly rapidly through the 'flat-land' space, calculating the posterior probability π_i for the current spot, proposing the next step and calculating the posterior probability of that step (π_j), then computing the ratio π_j/π_i (and, when the ratio < 1, also drawing a uniform random number) to decide whether to move on or stay in the same spot.

Note: The 'flat-lands' are not completely flat; if the landscape was completely flat, the chain would wander aimlessly.

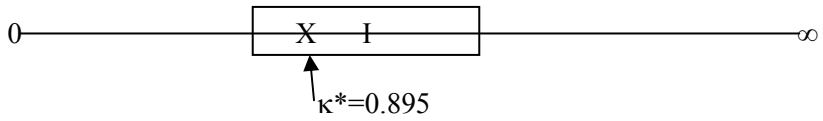
The steps that are proposed fall within a 'sliding window', which surrounds the current value of the parameter where the chain is sitting.

For example, suppose $\kappa = 1.156$ (κ is allowed to range from 0 to ∞)



How do we propose another value?

- choose window size (usually a preset value in program, and stays the same throughout the run; do not want a window that is too large or the chain gets stuck because most proposed steps would be rejected, and a window too small would result in a chain that moves very slowly)
- pick uniform random number between 0 and 1 (for example, 0.3)
- taking this arbitrary number as a proportion, move across this proportion of the window starting from the left end, and the value at that point will be the proposed step.



- after a step is proposed, the probability of the proposed step is calculated, and a decision is made about accepting or rejecting it (as demonstrated above).

Note from Paul:
An exercise[1] to test your understanding: the window used above starts at what value? ends at what value? is how wide?

If the window overlaps 0, and if the proposed value is less than 0, just negate the value to produce a positive (valid) proposed value. Although this rule seems like it would make the proposal probability distribution asymmetrical, it doesn't: the probability of proposing κ^* given that you are currently at κ is exactly the same as proposing κ given that you are currently at κ^*.

Codon Models

[1] Answers: the window starts at 0.5035, ends at 1.8085 and thus its width is 1.305.

Classical models (ex. K80 model) consider nucleotide sites independent and calculate the likelihood for one site without reference to other sites, and calculate the overall likelihood by taking the product of all site likelihoods ($L=L_1L_2L_3…..L_n$)

Though classical models can account for differences in transition rates versus transversion rates (ts vs. tv), they are not able to account for non-independence between sites, and therefore cannot account for the genetic code (i.e. the non-independent evolution of individual nucleotides within a codon).

Therefore, a codon models were invented to recognize codons as the independent states rather than nucleotide sites. To do this, 61x61 codon rate matrices are used, representing all possible codons (actually 64x64 matrix, less the 3 stop codons). An example of such a rate matrix can be downloaded from the PhyloMath web site, and corresponds to a simplified version of the matrix presented by:

> Goldman, N., and Z. Yang. 1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. Molecular Biology and Evolution 11:725-736.
>
> (Also see: Muse, S. V., and B. S. Gaut. 1994. A likelihood approach for comparing synonymous and nonsynonymous substitution rates, with application to the chloroplast genome. Molecular Biology and Evolution 11:715-724.)

In this rate matrix, the rate formulas are similar to those found in the HKY-model rate matrix, however ω's are incorporated to account for synonymous vs. non-synonymous changes in addition to transitions and transversions. 0's in the rate matrix result from the stipulation that changes in more than one nucleotide are not allowed in 1 unit of time.

Synonymous vs. Non-synonymous transitions and transversions:

Synonymous transition (nucleotide transition results in no change in amino acid)
GG**G** → GG**A** (Gly → Gly)

Synonymous transversion (nucleotide transversion results in no change in amino acid)
GG**G** → GG**C** (Gly → Gly)

Non-synonymous transition (nucleotide transition results in change in amino acid)
G**G**G → G**A**G (Gly → Glu)

Non-synonymous transversion (nucleotide transversion results in change in amino acid)
G**G**G → G**C**G (Gly → Ala)

The following assume that the transition or transversion results in a codon differing from the starting codon by having an A (adenine). Otherwise, π_C, π_G, or π_T would appear in place of the π_A.

Synonymous rates: ts. = $\pi_A\kappa\beta$
tv. = $\pi_A\beta$

Non-synonymous rates : ts. = $\pi_A\kappa\beta\omega$

$$tv. = \pi_A \beta \omega$$

If ω=1.0, synonymous transitions are occurring at the same rate as non-synonymous transitions (and the same goes for transversions). This would indicate there is no selection on that particular gene and therefore there is no penalty for changing the protein (ex. pseudogenes)

If ω<1, synonymous changes are occurring at a higher rate than corresponding non-synonymous changes, and this indicates that some or all of the protein is evolutionarily conserved (under stabilizing selection).

If ω>1, non-synonymous changes are occurring at an increased rate compared to the corresponding synonymous changes, and thus the amino acid sequence is changing a lot (at least in some part of the protein). High ω values correspond to positive selection.

In this general codon model, it is assumed that codons are evolving independently, however we know this is not the case. Proteins have a conserved tertiary (3º) structure, therefore every codon has some effect on the evolution of other codons.

Robinson et al. (2003) constructed a model to account for the 3º structure (non-independence of codons). The rate matrix is "All Possible Sequences" by "All Possible Sequences" and contains a lot of 0's (more than one nucleotide substitution not possible in 1 unit of time). It is similar to the general codon matrix illustrated above, but calculates rates accounting for 3º structure, solvent accessibility, etc. The model distinguishes among kinds of non-synonymous changes and penalizes those changes that would result in structural changes.

Jeff Thorne's seminar will go into more detail regarding this codon model, and we will continue discussing it next week.

Robinson DM, Jones DT, Kishino H, Goldman N, Thorne JL (2003) Protein evolution with dependence among codons due to tertiary structure. Molecular Biology and Evolution 20(10): 1692-1704.