Program using rjMCMC for exploring network priors

Graham Jones

2011-11-23

1 Introduction

The main difficulty with the network prior is that the normalization constants are needed for different number of parameters.

Suppose the models correspond to the number of hybridizations h. If there are m tetraploids, then there are m models $h = 1, h = 2, \ldots h = m$. Let W be the network topology and node times. The conditional priors $\pi(W|h)$ must be comparable for different values of h. To do this analytically, it is necessary to integrate out the parameters in W, ie calculate for each h the value of

$$\int \pi(W|h) \mathrm{d}W$$

which is a sum over labelled topologies and an integral over node times. That seems to be hard. It is possible to deal with Yule or birth-death priors for trees of different sizes, but I can't find anything analogous for networks.

An alternative is to use some fairly arbitrary prior, and estimate its properties by sampling from it. This note describes a program to do that.

2 Model

The network topology and node times is denoted by W. There are d diploids, m tetraploids, and $h \in \{1, 2, \ldots, m\}$ hybridizations. The network is divided into h tetraploid trees and a single 'diploid history', which is also a tree. Figure 1 shows an example with $d = 3$, $m = 6$, $h = 3$. Note that the diploid history has $d + 2h = 9$ tips. There are d ordinary diploid tips at time 0, and h pairs of tips at nonzero times (b_1, b_2, b_3 in Figure 1). I will call the latter 'hybridization tips'.

2.1 Probability calculation

The formula used (on 2011-11-25) is a Yule density for a tree X which is derived from the network. The hybridization tips are extended past the roots of tetraploid trees to the two children of the root, or to the present. See Figure 2.

The formula is

$$\frac{\lambda^{n-1} 2^{n-1}}{(n-1)!} \exp[-\lambda(2t_1 + t_2 + \cdots + t_{n-1})] \tag{1}$$

where there are n tips in the tree X, where

$$n = d + \sum_{i=1}^{h} \min(2, |T_i|) \tag{2}$$

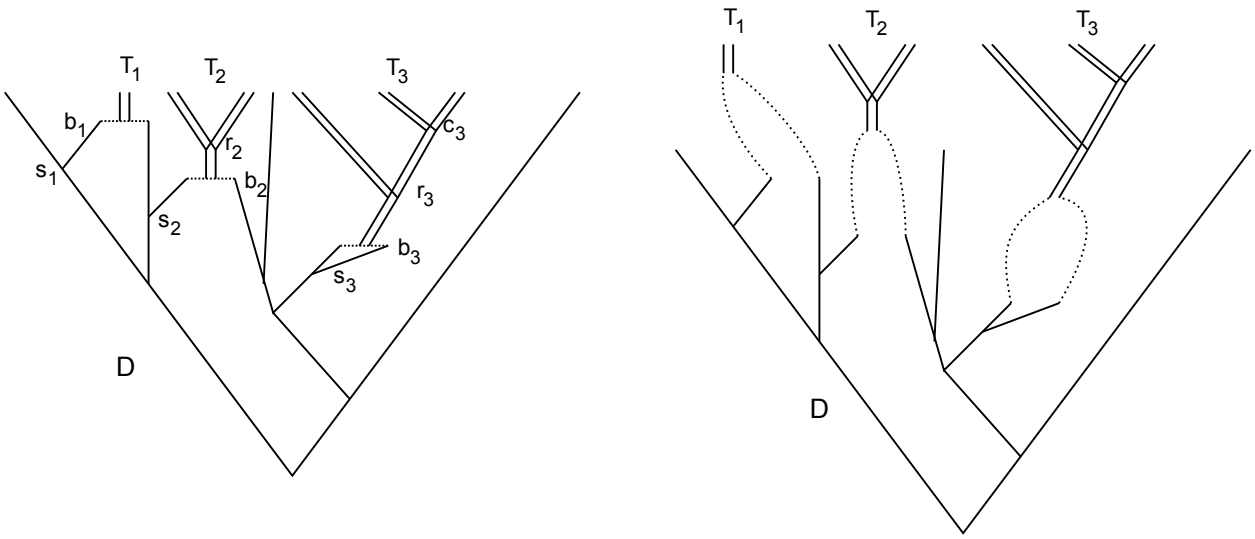

Figure 1: Network notation

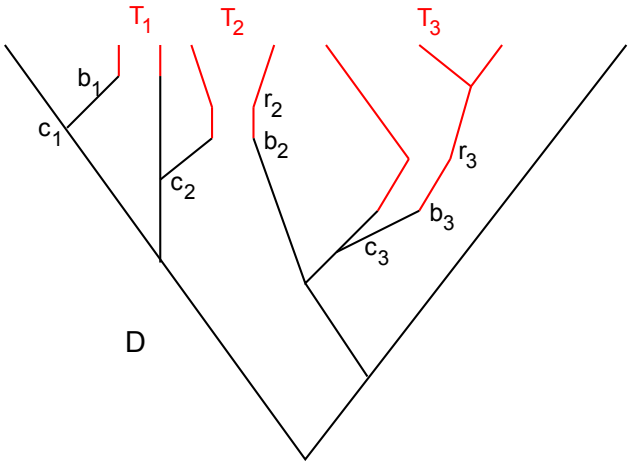

Figure 2: Network to tree

The node heights $t_1, t_2, \ldots, t_{n-1}$ are the internal nodes of the diploid history and the non-root nodes of all of the tetraploid subtrees.

3 Moves

I think it can be done with four types of move:

- 1. Change a tetraploid subtree, tipwards of the allopolyploidization
- 2. Change an allopolyploidization time
- 3. Change the diploid history, rootwards of allopolyploidizations
- 4. Change the number of tetraploid subtrees

3.1 MCMC moves in tetraploid subtrees

Many MCMC moves for trees have been developed. *BEAST uses a MCMC move for the species tree based on the ideas of [1], and this move also also seems appropriate here.

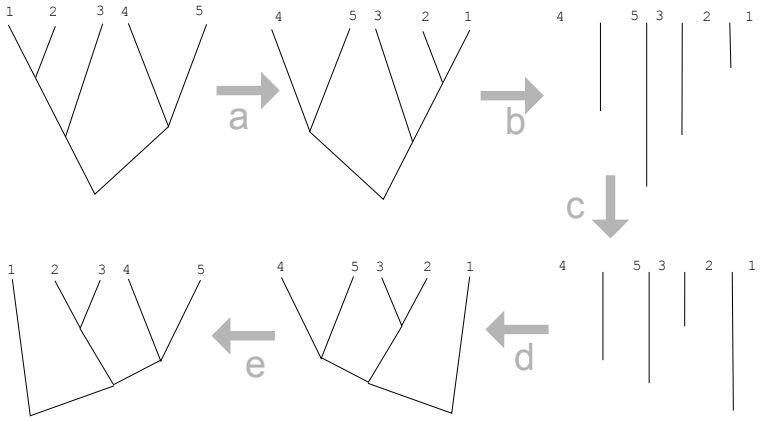

Figure 3: MCMC move based on Mau et al 1999.

a. Randomly orient the tree, giving each branch at each internal node a left/right label.

b. Convert to a point process representation (a 'drawing'), with every node getting an (x,y) position.

c. Change one or more of the node heights.

d. Then, forgetting the identity of the internal nodes, but keeping the left-right order of the tips, construct a new tree by joining up clades in order of node height.

e. (Not necessary, but it makes the move more symmetric.) Forget the left-right ordering.

This will change the topology if two nodes which were originally parent and child get heights which reverses the order. Step (c) should be reversible, so that the probability of going from one set of heights A to another set of heights B is the same as going from B back to A.

3.2 MCMC move for the allopolyploidization time

This is straightforward. In Figure 1, it means moving b_1 in $[0, s_1]$, b_2 in $[r_2, s_2]$, or b_3 in $[r_3, s_3]$. There is no change of topology.

3.3 MCMC move in the diploid history

The 'Mau move' described above can be adapted to deal with the diploid history. The diploid history is an ordinary tree with some tips having nonzero times. So in step (c) the heights must be changed in such a way that no node gets a height smaller than the tip height to its immediate left or right.

To keep this simple, I change only one height at a time, and use a uniform kernel.

3.4 MCMC move for the number of allopolyploidizations

NOTE: THIS DESCRIBES WHAT THE PROGRAM DOES (ON 2011-11-25), BUT IT IS NOT A RIGOROUS PROOF THAT THIS IS WHAT IT SHOULD DO.

Sampling all values of h can be done by repeatedly changing h to $h-1$ or $h+1$, and that can be done by splitting one tetraploid subtree into two and merging two into one. The difficult part is making the moves reversible, so that the probability of a move going from one state A to another B is balanced by a reverse move.

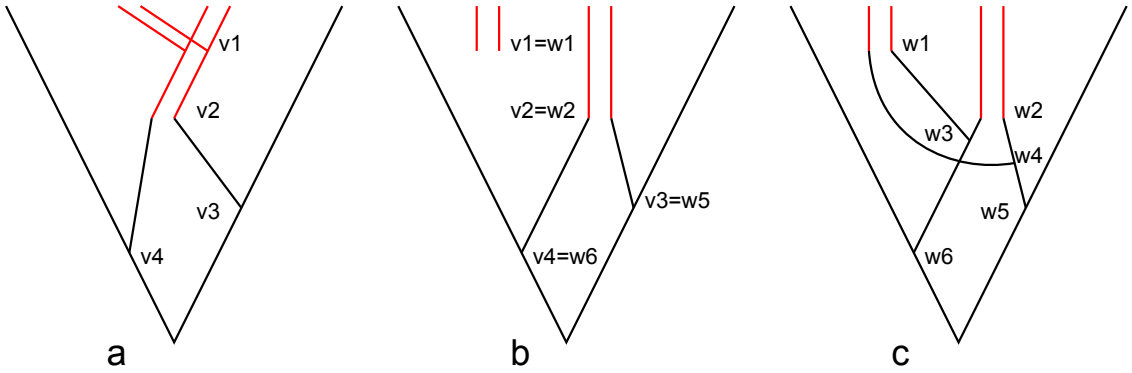

Figure 4: MCMC move to change number of tetraploid subtrees.

When the MCMC move for changing h is chosen, a split or a merge is chosen with equal probabilty. If a split is chosen, but no splits are possible, no move is made; the same state is sampled again. Likewise, if a merge is chosen, but no merges are possible, the same state is sampled again.

Splitting (going left to right in Fig 4). Any tetraploid subtree with more than one tip can be split. One, T, is chosen at random. The two child nodes of the root of the tetraploid subtree become the roots of the two new tetraploid subtrees T_1, T_2. In Fig 4, the root of T is at v_1, and the two new roots are also tips. The allopolyploidization time v_2 becomes the allopolyploidization time of one of the new tetraploid subtrees (w_2 in T_2), and the root time v_1 becomes the allopolyploidization time of the other, T_1. This produces the situation labelled (b).

Then two new tips with time w_1 are added to the diploid history. They are joined into the two external branches of the diploid history which led to the tips at w_2. One new node is created in each of these, at w_3 and w_4. The new nodes could be chosen anywhere in the diploid history earlier than w_1. The restriced choice is to mkae things simpler.

Merging (going right to left in Fig 4) must be the reverse of splitting. So two tetraploid subtrees can only merge if they have a configuration like that labelled (c) in the figure. It is required that the hybridization time of one tree is earlier than the root time of the other tree, and later than the hybridization time of the other ($0 \le w_1 \le w_2$ in the figure). It is also required that there are two nodes in the diploid history (the ones at w_3 and w_4 in the figure) which each have one child as a hybridization tip for T_1, and the other child as a hybridization tip for T_2. It is not necessary that the ancestors of these two nodes at w_3 and w_4 be different: the nodes at w_5 and w_6 could be identical.

A list of possible pairs of tetraploid subtrees is made, and if there any suitable pairs, one pair is chosen at random, and the merge is carried out. The most recent hybridization time (w_1) becomes the root time of

the merged tetraploid subtree T. The tree (T_2) with the earlier hybridization time provides the hybridization tips for T, and the hybridization tips of T_1 are removed.

There are various Hastings ratios to be worked out. I will describe what I have implemented on 2011-11-24. I don't think it is complete, and it is not tested. I am using [2] as my main reference. In particular, equations (7) and (8) are applicable.

When new node times (w_3 and w_4) are created, this is implemented by sampling two values u_1, u_2 from the uniform distribution on $[0, 1]$ and then using $w_3 = w_2 + u_1(w_6 - w_2)$ and $w_4 = w_2 + u_2(w_5 - w_2)$. All other node heights are set equal to existing values (they just acquire new interpretations). This means the determinant of the Jacobian when splitting is

$$\left| \frac{\partial(w)}{\partial(v, u)} \right| = (w_6 - w_2)(w_5 - w_2) \tag{3}$$

Here v is a vector of node times for the network with the split case, w is a vector of node times for the network with the merged case, and $u = (u_1, u_2)$. The inverse of equation (3) is used when merging. Since the density of u_1 and u_2 are both 1, the 'q' terms in equations (7) and (8) of [2] can be omitted.

This leaves the probabilities of the moves being chosen, namely $j(\mathrm{M}, w)$ and $j(\mathrm{S}, w)$ to be accounted for, where M and S stand for 'merged' and 'split', which are Green's cases 1 and 2.

For splitting, if there are N tetraploid subtrees with more than one tip, there are $N_s = 2N$ possible splitting moves, since either child of the root can play the role of T_1, the tree that gets new hybridization tips. The probability that one tetraploid subtree is chosen to be split is $1/N_s$. For merging, the number of ordered pairs N_m of tetraploid subtrees which are suitable for merging is found. The condition on the hybridization times means only one ordering of each pair is possible, and there are no further choices in how to merge two trees, so there are N_m merges to choose from.

This results in a ratio $(1/N_m(M))/(1/N_s(S)) = N_s(S)/N_m(M)$ when splitting, that is, moving from state M to state S.

4 Results

The results are odd, so far. This is the result for zero diploids ($d = 0$) and twenty tetraploids ($m = 20$).

h	count
1	21375
2	6490
3	4516
4	4638
5	5857
6	7670
7	8714
8	9187
9	7355
10	7086
11	11388
12	4267
13	1407
14	51
15	0
17	0
18	0
19	0
20	0

Note the local maxima at 1,8, and 11.

References

[1] Bob Mau, Michael A. Newton, Bret Larget, *Bayesian Phylogenetic Inference via Markov Chain Monte Carlo Methods*, Biometrics, Vol. 55, No. 1 (Mar., 1999), pp. 1–12

[2] Peter J Green, *Reversible jump Markov chain Monte Carlo computation and Bayesian model determination*, Biometrika (1995), Vol. 82, 4, No. 1 pp. 711–32