Homework #3 – (due Friday, Sep, 10)

The split equivalence theorem states that: if we have a set of splits then pairwise compatibility between all members of the set guarantees that there is a tree that displays all of the splits.

Because binary characters can be viewed as representations of a split, we can use this property of splits when finding a tree that is compatible with as many characters as possible (which is the tree preferred by the maximum compatibility theorem). See Chapter 8 of Felsenstein for a more thorough discussion of the compatibility approach to tree inference.

Table 1 shows two matrices. Each has 5 taxa (A-E) and 3 characters. These matrices demonstrate that we must have binary (two-state) characters without missing data if we want to use the trick "pairwise compatibility implies the existence of a tree that is compatible with all members of a set." In both of these examples, all of the characters are "pairwise compatible" with each other. However, if we try to map all three onto the same tree, we find that we have to use homoplasy to explain the distribution of character states.

For each matrix:

1. Show the three trees that demonstrate pairwise compatibility. In other words, show:
   (a) a tree that fits characters 1 and 2 perfectly,
   (b) a tree that fits character 1 and 3 perfectly, and
   (c) a tree that fits characters 2 and 3 perfectly.
2. For each of these trees, show the most parsimonious character reconstruction *all three* of the characters in the matrix. I know that we have not covered parsimony reconstructions yet, but I think you can find the reconstruction by experimenting. Some points to note:
   (a) for each tree, two of the characters should fit perfectly, but one character should require homoplasy.
   (b) When working through the first matrix (the one with missing data) make sure to indicate what state is present at the leaf of the tree (usually the data gives us the states at the leaves, but in this example three states are unscored).
   (c) I think that it is easiest to show the reconstruction by listing the states present in the ancestor in a table. But you can draw tick marks on the branches of the tree if you prefer.
3. You should end up with 6 trees (three for the first matrix and three for the second matrix). The trees for first and second matrix may not be distinct (it is possible that the same tree will show up for the first matrix and the second matrix) – but the problem is *not* asking you to consider both matrices at the same time.

Table 1: Data matrices

	Characters		
Taxon	1	2	3
A	0	0	0
B	?	0	1
C	1	?	0
D	0	1	?
E	1	1	1

	Characters		
Taxon	1	2	3
A	0	0	0
B	0	1	2
C	1	1	1
D	1	2	0
E	2	2	2