




The What the Hell Do I Do with All These Trees Lab

We've generated a lot of trees in the last few weeks. Today we're going to explore different ways to view, compare, and manipulate those trees. First we're going to use *TreeView* to look at a consensus tree generated in *MrBayes*. Then we're going to compare a bunch of trees with the same taxa but different topology using *PAUP** and generate several types of consensus trees. Finally we're going to generate a consensus tree from trees that have overlapping but not identical taxa using Matrix Representation with Parsimony.

We're going to be using many different files for this lab. Find them online at the 200A Syllabus and Handouts page.

TreeViewX

By now you will have noticed that most of the programs that generate trees either don't print trees at all or make really crappy ones. Luckily there are a lot of different ways to view trees. Today we'll be using *TreeViewX* (which you have already used a few times.) It seems to be a little quirky. There are some files I have exported with branch lengths from PAUP that nonetheless do not appear to have branch lengths in *TreeViewX*. You may also want to try plain *TreeView* if you are a PC or MacClassic user. If something isn't working, it is always worth opening it in Mesquite, exporting it in a few different ways, and seeing if any of them work. Mesquite can also export trees as a PDF, they sometimes come out a bit odd but try adjusting all the parameters. But back to today's lab and TreeViewX:

1. Download **MrBayesCephalopodConsensus.nex** from the web page. This is a consensus tree generated by *MrBayes* for the Cephalopod COI dataset we've been using. It has two trees in it with identical topology. The first has branch lengths and node support values. The second has only branch lengths.
2. Open *TreeViewX* and open **MrBayesCephalopodConsensus.nex** in it.
3. Pull down the **style** menu and change the font size, so that you can easily read the names.
4. Push  (the radial tree button) at the top of the page. You will see your tree as an unrooted tree.
5. Push the  phylogram button and the  internal labels button. The tree should now appear as a square phylogram with branch lengths that can be seen in the lengths of the branches and node support values as numbers.
6. Use the arrow buttons to view the other tree in the file, which in this case is just the same tree without internal node labels.
7. Go to the **Tree** menu and select **Define outgroup...** Double click *Alluroteuthis* (this is an arbitrary choice). Now hit **OK**.

8. Pull down the **Tree** menu again and select **Root with outgroup**. This should reroot the tree in the display with *Alluroteuthis* in the outgroup. It will look very different but will still have the same topology.
9. To generate a picture for use in your paper pull down the **File** menu and select **Print Preview**. Then click **Picture** to save that tree as a metafile for use with other programs, or **Copy** to paste it into another program.

Tree Distances

Now we're going to use *PAUP** to generate a number of different tree distance measures on a bunch of trees with the same taxa, like we talked about in class.

10. Download **CephalopodTreesTprobs.nex** from the web site. It contains the first 13 trees from the tprobs file for the cephalopod dataset. These are the most probable 50% of trees that I found during stationarity. It also has the estimated posterior probabilities of those trees.
11. Open PAUP*.
12. Execute CephalopodTreesTprobs.nex in PAUP.

13. Now, we will calculate the differences between the trees in the sample using tree-to-tree differences. For the first run we'll do symmetric differences. This is just a measure of the partitions that the two trees do not share. A branch can be viewed as a partition, because it separates the taxa into two groups. So if branches in both trees separate the taxa into the same two groups, then that partition is shared. Thus more similar trees will disagree on fewer partitions and so have smaller symmetric differences. Type:

```
treedist;
```

This should output a matrix of pairwise differences between the trees, and a frequency distribution of those differences. Are the trees with higher posterior probabilities more like the tree with the highest posterior probability? (Remember these trees are listed in the order of their posterior probabilities, from 1 to 13.) Is the tree with the highest posterior probability more similar to the other trees than they are in general to each other? Why would this be? What trees have the biggest differences?

14. Repeat this analysis, only this time use "Agreement metric d". This is a measure of how many taxa you have to remove to make two trees the same with a correction added on so that if the taxa removed are further apart you get a bigger number, thus more similar trees should have smaller differences. Type:

```
treedist metric = agreement;
```

How do the trees compare under this measure of difference? Do the two metrics produce similar histograms? Which metric is more informative?

Consensus Trees

Now we're going to generate several different consensus trees from that same tree file using *PAUP**. I want to emphasize that this is not the appropriate way to generate a consensus tree from *MrBayes*. It is much better to use the **sumt** command in *MrBayes*, because that will consider the trees based on their estimated posterior probabilities and will also calculate branch lengths. However, there are many other situations when you would want to use this method, such as if you generate several most parsimonious trees. I'm just using this tree file, because it is convenient.

Last week, when we practiced bootstrap, jackknife, and decay index, we got a bit of experience generating consensus trees. Now we will explore this command in a bit more detail:

15. First let's generate a strict consensus of all the trees in memory. Type:

```
contree all /strict=yes;
```

This will output a tree that only contains nodes present in all your input trees.

16. When generating consensus trees, *PAUP** will not hold the consensus trees in memory. If you want to keep the consensus trees, you have to save them to a file. Let's practice this now:

```
contree all /strict=yes treefile=mytree.tre;
```

This will save a consensus tree file to your working directory.

17. Now let's generate a Majority Rule tree with a cut off at 50%. This will output a tree with all the nodes that appear in more than 50% of the tree. It will also tell you in what percentage of those trees the nodes occurred.

```
contree all /strict=no majrule=yes percent=50;
```

Does this have the same topology as the strict consensus? Are they compatible?

18. Generate another Majority Rule tree, only this time up the cut off, so that you eliminate some clades. The cut off point is always kind of arbitrary, but can not be less than 50%. If it were less than 50%, then you couldn't be sure that all the clades are compatible. How high would the cut off have to be to guarantee that you are going to get the same tree as strict consensus?

Matrix Representation with Parsimony (MRP)

So it's easy to generate consensus trees if they all have exactly the same taxa, but what do you do if all the trees have different taxa? For example how would you put together a bunch of trees from different studies with overlapping but not identical taxa? Well, it is a matter of big debate. Maybe you shouldn't even do it at all. Maybe it is best to take the data matrices from all those studies and combine them into one supermatrix for analysis. If you do decide to combine trees it is not at all clear what the best method is. The mostly commonly used method is Matrix Representation with Parsimony (MRP). Here we're going to do a made up easy example of it.

We are going to do MRP on the three trees of rays that you will find on the next page. I just took a single data set of rays, randomly deleted two taxa from it three times, and used those

reduced data sets to make trees by parsimony. This is a totally unrealistic situation for several reasons. The taxa have a lot of overlap. If these were three trees picked from the literature they would have very little overlap. This would mean that the MRP matrix would have a lot of question marks. Also the trees are all generated from the same data set, so that you know that you won't have a contradictory signal from two different data sets, which you are likely to have in reality. However, I didn't have a time to find a more realistic set of trees, and these will make filling out the matrix easier.

19. Download the Ray matrix from the web site. Open it in Mesquite. This is just an empty matrix with the taxa names on it (and apparently lots of spelling mistakes), so that you don't have to bother filling them all in. You are just going to fill in the data.
20. Now code the three trees into the matrix. You do this by treating each interior branch from each tree as a separate character. Remember that every branch separates the taxa into two groups, one on each side of the branch. You can assign each of these partitions a separate character state, so that all the taxa on one side of a branch get a 0 and the other side a 1 for that character. All the taxa that don't appear in that tree should get a ?. Every tree should have 6 internal branches.
21. For example the branch that I marked as A in the first tree should be coded:

Raja polystigma	1
Raja montagui	1
Raja brachyura	1
Raja microocellata	1
Raja asterias	0
Raja undulate	0
Raja radula	?
Raja clavata	?
Leucoraja meitensis	0
Leucoraja naevus	0
Leucoraja fullonica	0

22. When you're done with the matrix save it and close Mesquite.
23. Open PAUP* and open the MRP matrix in it.
24. Run an exhaustive search for the most parsimonious tree. Type:
`alltrees;`
25. Did you get one tree? Was it fully resolved? Is there any homoplasy (which in this case would indicate a disagreement between the trees)?

Well that looks very pretty. It wouldn't be so pretty if I hadn't cheated.

26. **You should save a copy of this tree, print it out, and turn it into me, now or next week.**

