Integrative Biology 200A *University of California, Berkeley*
"PRINCIPLES OF PHYLOGENETICS" *Spring 2006*

Lab 8: Distance Methods

Today we're going to use *PAUP** to generate trees using distance methods. We've discussed distance methods in class, and you have learned that they are not the most theoretically justified of methods for finding trees. However, it is important that you learn how to utilize them. First you should use them as one of the analyses in your paper. Also some people do feel that they are a good way to find trees. Finally, they are by far the fastest way to find a tree. Whereas parsimony and likelihood methods have to search through tree space and compare the optimization of the character matrix on many trees, most distance methods use an algorithm to directly generate a tree from the distance matrix. This speed makes it very useful for genomics, where it is often necessary to generate tens of thousands of trees, but getting the exact tree each time is not as important as getting the right tree the vast majority of the time.

There are two different ways that distance analyses may differ. We can use different formulas to calculate the distances, we will cover this first. Once we have a distance matrix we can use different algorithms to generate the tree.

I have included a lot of math in this one. You do not have to memorize it. It is only there for the benefit of those of you who are interested. However, you should understand the assumptions that go into each distance measure.

Download Sequences

First download the nexus file from http://ib.berkeley.edu/courses/ib200a/cephalopod_COI_Clustalw.nex, and open it in PAUP*.

Generate a parsimony tree for comparison to the other trees you are going to generate: Pull down the **Analysis** menu and select **Parsimony**. Then pull down the **Analysis** menu again, select **Heuristic search** then hit **Search**. To see the trees pull down the **Trees** menu and select **Describe Trees** then hit **OK.**

Distance Measures

To chose a new distance measure pull down the **Analysis** menu and select **Distance Settings**. Then select the distance method from the menu next to **DNA/RNA Distances**. For each distance measure that I describe I want you to do the following things:

1) Generate a distance matrix: **Data>Show Pairwise Distances**. A distance matrix is an estimate of the average number of changes per base pair for each pair of sequences.

2) Generate a neighbor joining tree: First pull down the **Analysis** menu and select **Distance**, if it is not already selected. This sets the optimality criterion. Then pull down the **Analysis** menu and select **Neighbor Joining/UPGMA.** Then select **Neighbor Joining**, check the box next to **Show branch lengths** and hit **OK**. We'll discus this in more detail later.

3) Compare the distance matrix and tree for each distance measure. Which measure produces the longest distances? How do the branch lengths compare (A chart of branch lengths can be found above the tree)? How do they compare to parsimony branch lengths? Why? Which tree topology compares best to a tree generated using parsimony?

Uncorrected ("p")

P-distance is the uncorrected number of changes between two sequences. The problem with this method is that it does not take account of the base pairs where multiple changes have occurred. It counts each base pair that has changed as 1 change and all unchanged base pairs as 0, although either one of these situations can mask many other changes. Thus the distance between any sequences will approach ¾ as they get larger. For such distances adding together the distance of two segments of a path gives a distance larger than the entire path. This violates a fundamental assumption of most distance based algorithms that such values are equal.

Jukes-Cantor (JC)

Jukes-Cantor distances are the simplest way to solve this problem. They assume that the chance of any nucleotide changing into any other anywhere in the sequence is equal. Thus, if **P(t)** is the chance that any nucleotide is a different nucleotide at time **t**, and **u** is the instantaneous stochastic rate at which a nucleotide changes into any other nucleotide:

$$P'(t) = u * (1-P(t)) - 1/3\ u * P(t)$$

The distance **D**, will now be the average number of changes per base pair or **ut**.

$$D = ut = -3/4 \ln(1- 4/3\ P(t))$$

where **P(t)** is the fraction of changed nucleotides.

Kimura 2-parameter (K2P)

The Kimura two-parameter model essentially solves the problem in the same way as Jukes-Cantor, except that there are different rates for transitions and trasversions. Thus if **P(t)** and **Q(t)** are the chance a base pair has visibly undergone a transition and a transverion respectively and α and β are the instantaneous stochastic transition and transversion rates:

$$Q'(t) = 2\beta * (1-Q(t)) - 2\beta * Q(t) \text{ and}$$

$$P'(t) = \alpha * (1-P(t)-Q(t)) + \beta * Q(t) - (\alpha+2\beta) * P(t)$$

The distance can now be calculated as,

$$D = (\alpha + 2\beta) * t = -1/4 \ln [(1-2Q(t)) (1-2P(t)-Q(t))^2]$$

Where **P(t)** and **Q(t)** are the fraction of transitions and transversions observed.

General Time-Reversible (GTR)

The general time reversible model relaxes the assumptions about the correlation among rates of change from one nucleotide to another as much as possible while still having the same probability of change in both directions. Thus there are six different rates that have to be set (A to G / G to A is one rate, and A to C / C to A another) and an equilibrium frequency for each nucleotide. The total rate has to average out to one and all the equilibrium frequencies have to ad up to one, so that there are eight total parameters in this model, in addition to all the pairwise distances. The distances can not be directly calculated, but instead all the parameters have to be estimated using Maximum Likelihood. Because there are so many parameters, you have to have a lot of data to make a good ML estimate. We'll talk a lot more about these types of models, when we do Maximum Likelihood.

You can also make the rates vary from one base-pair to the next along the sequence. Open the **Distance Settings** window. Leave the distance method as general time reversible, and select **Gamma, shape** at the bottom of the window this makes the program calculate the Likelihood of the data as if the rates for each base are chosen at random from a gamma distribution.

Distance Methods

All methods of generating a tree from a distance matrix depend on the pairwise distances between the sequences, and thus depend critically on the distance measure used. However, they can all derive different trees, as they analyze the data in different ways. Let's stick with the GTR for these calculations.

Many of these analyses are based on the principle of *least squares.* Imagine that you have a pairwse distance matrix and a tree with branch lengths. You can ad up the branches on the tree that connect any two taxa, and take the difference between that sum and the pairwise distance between the taxa. Least squares assigns branch lengths to a tree by minimizing the sum of the square of that difference for all the pairwise distances. It can further be used to pick among the trees by choosing the tree with the least sum of squares.

UPGMA

UPGMA is a clustering algorithm for generating trees from a distance matrix. It assumes that the trees are ultrametric, meaning that the branch lengths obey the molecular clock. It approximates the least squares tree. It approximates an ultrametric, least squares tree and is well behaved if the molecular clock is followed.

It works like this:

1) It takes the two OTUs with the smallest pairwise distance between them.

2) It joins them together at a node and gives the distance between each of those OTUs and that node as one half of the distance between the two OTUs.

3) It calculates the distance between that node and all the other OTUs as the average of the difference between each of the OTUs that make up this new node and the other OUT.

4) Eliminate the OTUs included in the new node, and replace them with the node.

5) Repeat.

To run UPGMA:

1) **Analysis > Distance**

2) **Analysis > Neighbor Joining/UPGMA**

4) Select **UPGMA**

5) Check box next to **Show branch lengths**

You can also change the distance measure from this window by clicking the Distance Options button.

6) Press **OK**

Neighbor Joining and BioNJ

Neighbor joining is another clustering algorithm, but it does not assume the molecular clock. It does a very good job of approximating Minimum Evolution. In fact it is guaranteed to get the right tree if the distance matrix is an exact reflection of the tree (which it never is). NJ is currently the distance method with the best reputation and is thus the one most commonly used, although UPGMA is still used in a lot of genomics studies for some reason beyond my comprehension.

The clustering algorithm of NJ is similar to that of UPGMA in that both replace pairs of taxa OTUs with composite OTUs one after another. However, it makes much more complicated calculations that I won't go over here. If you're interested, I can point you in the right direction.

To run Neighbor Joining:

1) **Analysis > Distance**

2) **Analysis > Neighbor Joining/UPGMA**

4) Select **Neighbor Joining**

5) Check box next to **Show branch lengths**

6) Press **OK**

BioNJ is a modification of the NJ algorithm that weights the component branches differently when combining them in order to account for the variances and covariances of the distances. It thus improves the approximation of least squares.

To run BioNJ, check box next to **Use BioNJ Method**, before running an NJ analysis.

Heuristic Distance Methods

It is also possible to find a tree using a distance matrix by comparing the fit of many trees to the distance matrix according to some optimality criterion, as we do when we use parsimony or maximum likelihood to find the best tree. These methods are better justified than any of the clustering algorithms, but they lose much of the time saved by using clustering algorithms, as they still have to search through tree space.

1) **Analysis > Distance**

2) **Analysis>Distance Settings**

3) Select **Objective Function** from **Other Options** menu

4) Pick one of the optimality criteria below.

Minimum Evolution uses least squares to assign branch lengths to the trees, but then chooses the tree with the least total branch length as the best.

Unweighted Least Squares uses least squares to assign branch lengths and pick the best tree.

Weighted Least Squares uses different waiting criteria to calculate the least squares. It uses this calculation to assign branch lengths and pick a tree.

5) **Analysis>Heuristic search** then hit **Search**

6) To see the trees pull down the **Trees** menu and select **Describe Trees** then hit **OK.**

Compare the trees from each distance method. How do the branch lengths compare (A chart of branch lengths can be found above the tree)? How do they compare to parsimony branch lengths? Which tree topology compares best to a tree generated using parsimony?