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

Distance Methods

Due at the end of class:

- **Distance matrices and trees for two different distance measures**
- **Tree from one extra distance-based tree building method**
- **Parsimony reference tree**

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 inferring phylogenies, although clustering methods do have some uses in other areas of statistics. For several reasons, it is important that you learn how to utilize them. First, you should use them as one of the analyses in your final project, as a comparison to other methods. They will almost always give a different tree than the other optimality criteria, since they are searching for total similarity, and not distinguishing between synapomorphy, symplesiomorphy, and homoplasy. Second, some workers people do feel that they are a good way to infer phylogenies. 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.

Distance analysis have two main components: the formulas used to calculate the distances (a.k.a. distance measures) and the algorithms used to compute a tree from the distances.

Get Sequences, Generate a Reference Tree

First, download the cephalopod nexus file from the syllabus page on the 200A website. You may have to right-click on the link and choose save as to download it rather than open it in your browser. Save it to the desktop, or to a folder you create for yourself on the desktop, and open it in PAUP*.

Generate a parsimony tree for comparison to the other trees you will make during this lab:

```
set criterion = parsimony;
hs;
```

The search will retain only one tree. Look at the tree, then save it:

```
showtree;
savetree file=ceph_COI_parsimony.tre;
```

Print out this tree and save it to turn in.

Distance Measures: ways to measure how different things are

Phenetic methods start by measuring how "different" taxa are from one another. They look at the total similarity across many different characters, and analyze these in a statistical framework to come up a measure of the "distance" between each pair of taxa. This works a lot like those timetables that tell you the distance between two cities – like the example at left. Notice that the distance method can only compare two cities at a time – the same is true for distance methods in phylogenetics. Second, consider that you could measure the "distance" between these cities in several different ways. This chart measures it in miles, but it could also have used kilometers, driving time, or the number of In-and-Out Burgers between each city. In phylogenetics, there are also several different ways of calculating how "far apart" taxa are. Some of them are detailed below.

Distance in mi.	Hearst Castle	Los Angles	Monterey
Hearst Castle	-	234	94
Los Angeles	234	-	327
Monterey	94	327	-

Uncorrected (P)

P-distance is the uncorrected number of changes between two sequences. It is called "uncorrected" because 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 transversions. Thus if **P(t)** and **Q(t)** are the chance a base pair has visibly undergone a transition and a transversion respectively and α and β are the instantaneous stochastic transition and transversion rates:

$Q'(t) = 2\beta * (1-Q(t)) - 2\beta * Q(t)$ 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.

Using Different Distance Measures

Each of these different distance measures is implemented in PAUP. You mission is now to try out each one and see what effect it has on our estimate of cephalopod relationships. To test each distance measure do the following things:

1) Set the distance setting you want to use by typing
   dset distance = *distancetype*;
   but instead of *distancetype* type one of the distance codes for each measure discussed above. The codes for each measure are:

Uncorrected P:	P
Jukes-Cantor:	JC
Kimura 2-parameter:	K2P
General Time-Reversible:	*GTR*

   See the Dset entry in the PAUP manual for more options.

2) Generate a distance matrix by typing
   showdist;
   A distance matrix is an estimate of the average number of changes per base pair for each pair of sequences.

3) Generate a neighbor joining tree with branch lengths, and save it:
   nj brlens = yes treefile= *treename*.tre;
   Instead of *treename* type the name you want the file to have (no spaces.) For more details on the other options, see the PAUP manual entry on UPGMA.

4) 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?

You can also make the rates vary from one base-pair to the next along the sequence. To do this, using the distance method General Time Reversible, type:

```
dset distance = GTR rates = gamma;
```

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.

Choose two of these methods and print the distance matrix and the resulting tree to turn in. To copy the distance matrix, go to the Edit menu and choose Edit display buffer. A text version of everything you've done so far will pop up. You can copy and paste into a text file. Don't choose print window in PAUP, as it will print the entire file.

Distance-based Methods for Building Trees

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, once the distance matrix has been generated, there are actually several different algorithms available for using these data to group taxa in a tree.

Many of these analyses are based on the principle of *least squares.* Imagine that you have a pairwise distance matrix and a tree with branch lengths. You can add 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.

Here is a brief description of several distance methods available in PAUP, along with instructions on how to run them. For ease of comparison, as you do these exercises, let's stick with the GTR distance method with rates equal:

```
dset distance = GTR rates = equal;
```

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. The steps in a UPGMA analysis are:

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 type:

```
upgma brlens = yes;
```

Neighbor Joining

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 the most commonly used. 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.

To run Neighbor Joining type:

```
nj brlens = yes;
```

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.

To do a heuristic distance-tree search, first set the objective function, which tells PAUP how to calculate branch lengths and tree scores, allowing comparison among the trees it finds in tree space. There are two different choices, either one is ok for the purposes of this lab:

1) *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. The command for minimum evolution is:
   ```
   dset distance = GTR objective=me;
   ```
2) *Least Squares* uses least squares to assign branch lengths and pick the best tree. The command is:
   ```
   dset distance = GTR objective=lsfit;
   ```

Next, set the search criterion to distance. Type

```
set criterion = distance;
```

then do a heuristic search

```
hs;
```

and look at your trees

```
describetrees /brlens=yes;
```

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?

Choose a tree from one of these methods, print it out and turn it in.