The Neighbor-Joining Method

Neighbor-joining (Saitou and Nei, 1987) is a method that is related to the cluster method but does not require the data to be ultrametric. In other words it does not require that all lineages have diverged by eaqual amounts. The method is especially suited for datasets comprising lineages with largely varying rates of evolution. It can be used in combination with methods that allow correction for superimposed substitutions.

The following programs are available

- Neighbor of the Phylip package (Jo Felsentein, Univ. Washington),
- ClustalW (D. Higgins, EMBL) ,
- Distnj in the Protml package (Adachi and Hasegawa, Univ. Tokyo)

The neighbor-joining method is a special case of the star decomposition method. In contrast to cluster analysis neighbor-joining keeps track of nodes on a tree rather than taxa or clusters of taxa. The raw data are provided as a distance matrix and the initial tree is a star tree. Then a modified distance matrix is constructed in which the separation between each pair of nodes is adjusted on the basis of their average divergence from all other nodes. The tree is constructed by linking the least-distant pair of nodes in this modified matrix. When two nodes are linked, their common ancestral node is added to the tree and the terminal nodes with their respective branches are removed from the tree. This pruning process converts the newly added common ancestor into a terminal node on a tree of reduced size. At each stage in the process two terminal nodes are replaced by one new node. The process is complete when two nodes remain, separated by a single branch.

Negative branch lengths
As the neighbor-joining algorithm seeks to represent the data in the form of an additive tree, it can assign a negative length to the branch. Here the interpretation of branch lengths as an estimated number of substititions gets into difficulties. When this occurs it is adviced to set the branch length to zero and transfer the difference to the adjacent branch length so that the total distance between an adjacent pair of terminal nodes remains unaffected. This does not alter the overall topology of the tree (Kuhner and Felsenstein, 1994).

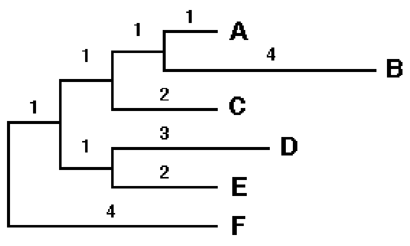
Advantages and disadvantages of the neighbor-joining method

- Advantages
  - is fast and thus suited for large datasets and for bootstrap analysis
  - permist lineages with largely different branch lengths
  - permits correction for multiple substitutions
- Disadvantages
  - sequence information is reduced
  - gives only one possible tree
  - strongly dependent on the model of evolution used.

NB: *especially its suitability to handle large datasets has led to the fact that the method is widely used by molecular evolutionists. With the rapid growth of sequence databases it is still one of the few methods that allows the rapid inclusion of all homologous sequences present in the database in a single tree. A good example can be found in the Ribosomal Database Project that maintains a tree of life based on all available ribosomal RNA sequences.*

Example of the method

Suppose we have the following tree:



Since B and D have accumulated mutations at a higher rate than A. The Three-point criterion is violated and the UPGMA method cannot be used since this would group together A and C rather than A and B. In such a case the neighbor-joining method is one of the recommended methods.

The raw data of the tree are represented by the following distance matrix:

	A	B	C	D	E
B	5				
C	4	7			
D	7	10	7		
E	6	9	6	5	
F	8	11	8	9	8

We have in total 6 OTUs (N=6).

Step 1: We calculate the net divergence r (i) for each OTU from all other OTUs

```
r(A) =  5+4+7+6+8=30
r(B) = 42
r(C) = 32
r(D) = 38
r(E) = 34
r(F) = 44
```

Step 2: Now we calculate a new distance matrix using for each pair of OUTs the formula:

M(ij)=d(ij) - [r(i) + r(j)]/(N-2) or in the case of the pair A,B:

M(AB)=d(AB) -[(r(A) + r(B)]/(N-2) = -13

	A	B	C	D	E
B	-13				
C	-11.5	-11.5			
D	-10	-10	-10.5		
E	-10	-10	-10.5	-13	
F	-10.5	-10.5	-11	-11.5	-11.5

Now we start with a star tree:

```
      A
F    |    B
  \  |  /
   \ | /
    \|/
    /|\
   / | \
  /  |  \
E    |    C
      D
```

Step 3: Now we choose as neighbors those two OTUs for which Mij is the smallest. These are A and B and D and E. Let's take A and B as neighbors and we form a new node called U. Now we calculate the branch length from the internal node U to the external OTUs A and B.

```
S(AU) =d(AB) / 2 + [r(A)-r(B)] / 2(N-2) = 1
S(BU) =d(AB) -S(AU) = 4
```

Step 4: Now we define new distances from U to each other terminal node:

```
d(CU) = d(AC) + d(BC) - d(AB) / 2 = 3
d(DU) = d(AD) + d(BD) - d(AB) / 2 = 6
d(EU) = d(AE) + d(BE) - d(AB) / 2 = 5
d(FU) = d(AF) + d(BF) - d(AB) / 2 = 7
```

and we create a new matrix:

	U	C	D	E
C	3			
D	6	7		
E	5	6	5	
F	7	8	9	8

The resulting tree will be the following:

```
                            C
                         D  |
                          \ |     A
                           \|___/ 1
                           /|    \
                          / |     \ 4
                         E  |      \
                            F       \
                                      B
```

N= N-1 = 5
The entire prodcedure is repeated starting at step 1

```
Last updated: 8 August 1997.
```

---

created by :Fred Opperdoes