Chapter 11

Distance matrix methods

A major family of phylogenetic methods has been the *distance matrix methods*, introduced by Cavalli-Sforza and Edwards (1967) and by Fitch and Margoliash (1967; see also Horne, 1967). They were influenced by the clustering algorithms of Sokal and Sneath (1963). The general idea seems as if it would not work very well: calculate a measure of the distance between each pair of species, and then find a tree that predicts the observed set of distances as closely as possible. This leaves out all information from higher-order combinations of character states, reducing the data matrix to a simple table of pairwise distances. One would think that this must leave out so many of the subtleties of the data that it could not possibly do a reasonable job of making an estimate of the phylogeny.

Computer simulation studies show that the amount of information about the phylogeny that is lost in doing this is remarkably small. The estimates of the phylogeny are quite accurate. Apparently, it is not common for evolutionary processes (at least not the simple models that we use for them) to leave a trace in high-order combinations of character states without also leaving almost the same information in the pairwise distances between the species.

The best way of thinking about distance matrix methods is to consider distances as estimates of the branch length separating that pair of species. Each distance infers the best unrooted tree for that pair of species. In effect, we then have a large number of (estimated) two-species trees, and we are trying to find the n-species tree that is implied by these. The difficulty in doing this is that the individual distances are not exactly the path lengths in the full n-species tree between those two species. They depart from it, and we need to find the full tree that does the best job of approximating these individual two-species trees.

Branch lengths and times

In distance matrix methods, branch lengths are not simply a function of time. They reflect expected amounts of evolution in different branches of the tree. Two

branches may reflect the same elapsed time (as when they are sister lineages in a rooted phylogeny), but they can have different expected amounts of evolution. In effect, each branch has a length that is a multiple r_i of the elapsed time t_i. The product $r_i t_i$ is the branch length. This allows different branches to have different rates of evolution.

The least squares methods

We start by describing the *least squares methods*, which are some of the best-justified ones statistically. The distances themselves also need some discussion, as they must have particular mathematical and statistical properties to work with these methods. We also describe one variant, the minimum evolution methods, and two quicker but more approximate distance matrix methods: UPGMA clustering and the neighbor-joining method.

The fundamental idea of distance matrix methods is that we have an observed table (matrix) of distances (D_{ij}), and that any particular tree that has branch lengths leads to a predicted set of distances (which we will denote the d_{ij}). It does so by making the prediction of the distance between two species by adding up the branch lengths between the two species. Figure 11.1 shows a tree and the distance matrix that it predicts. We also have a measure of the discrepancy between the observed and the expected distances. The measure that is used in the least squares methods is

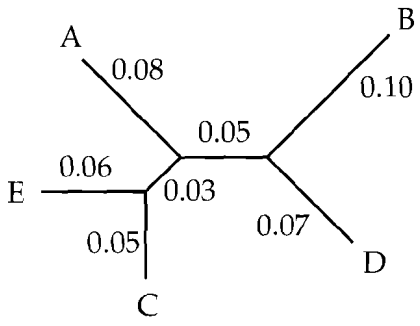
$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(D_{ij} - d_{ij})^2 \tag{11.1}$$

where the w_{ij} are weights that differ between different least squares methods. Cavalli-Sforza and Edwards (1967) defined the unweighted least squares method in which $w_{ij} = 1$. Fitch and Margoliash (1967) used $w_{ij} = 1/D_{ij}^2$, and Beyer et al. (1974) suggested $w_{ij} = 1/D_{ij}$. We are searching for the tree topology and the branch lengths that minimize Q. For any given tree topology it is possible to solve for the branch lengths that minimize Q by standard least squares methods.

The summation in equation 11.1 is over all combinations of i and j. Note that when $i = j$, both the observed and the predicted distances are zero, so that no contribution is made to Q. One can alternatively sum over only those j for which $j \neq i$.

Least squares branch lengths

To find the branch lengths on a tree of given topology using least squares we must minimize Q. The expression for Q in equation 11.1 is a quadratic in the branch lengths. One way that it can be minimized is to solve a set of linear equations. These are obtained by taking derivatives of Q with respect to the branch lengths, and equating those to zero. The solution of the resulting equations will minimize



	A	B	C	D	E
A	0	0.23	0.16	0.20	0.17
B	0.23	0	0.23	0.17	0.24
C	0.16	0.23	0	0.15	0.11
D	0.20	0.17	0.15	0	0.21
E	0.17	0.24	0.11	0.21	0

Figure 11.1: A tree and the distances it predicts, which are generated by adding up the lengths of branches between each pair of species.

Q. In equation 11.1 the d_{ij} are sums of branch lengths. Figure 11.2 shows the same tree with variables for the branch lengths. If the species are numbered in alphabetic order, d_{14} will be the expected distance between species A and D, so that it is $v_1 + v_7 + v_4$. The expected distance between species B and E is $v_{2,5} = v_5 + v_6 + v_7 + v_2$.

Suppose that we number all the branches of the tree and introduce an indicator variable $x_{ij,k}$, which is 1 if branch k lies in the path from species i to species j and 0 otherwise. The expected distance between i and j will then be

$$d_{ij} = \sum_k x_{ij,k}\, v_k \tag{11.2}$$

Equation 11.1 then becomes

$$Q = \sum_{i=1}^{n} \sum_{j:j\neq i} w_{ij} \left(D_{ij} - \sum_k x_{ij,k} v_k \right)^2 \tag{11.3}$$

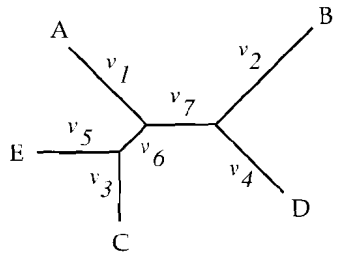


Figure 11.2: The same tree as the previous figure, with the branch lengths as variables.

If we differentiate Q with respect to one of the v's such as v_k, and equate the derivative to zero, we get the equation

$$\frac{dQ}{dv_k} = -2\sum_{i=1}^{n}\sum_{j:j\neq i} w_{ij}\left(D_{ij} - \sum_{k} x_{ij,k}v_k\right) = 0 \tag{11.4}$$

The -2 may be discarded.

One way to make a least squares estimate of branch lengths is to solve this set of linear equations. There are both exact and iterative methods for doing this. In the case of Cavalli-Sforza and Edwards's original unweighted least squares methods, where the weights w_{ij} are all 1, the equations are particularly simple. This will lead us to a nice matrix form, and the more general case can then be put in that form. (The reader who is prone to panic attacks at the sight of matrices should skip the rest of this subsection and the one on generalized least squares as well.) For the unweighted case, for the tree in Figures 11.1 and 11.2, the equations are:

$$\begin{aligned}
D_{AB} + D_{AC} + D_{AD} + D_{AE} &= 4v_1 + v_2 + v_3 + v_4 + v_5 + 2v_6 + 2v_7 \\
D_{AB} + D_{BC} + D_{BD} + D_{BE} &= v_1 + 4v_2 + v_3 + v_4 + v_5 + 2v_6 + 3v_7 \\
D_{AC} + D_{BC} + D_{CD} + D_{CE} &= v_1 + v_2 + 4v_3 + v_4 + v_5 + 3v_6 + 2v_7 \\
D_{AD} + D_{BD} + D_{CD} + D_{DE} &= v_1 + v_2 + v_3 + 4v_4 + v_5 + 2v_6 + 3v_7 \\
D_{AE} + D_{BE} + D_{CE} + D_{DE} &= v_1 + v_2 + v_3 + v_4 + 4v_5 + 3v_6 + 2v_7 \\
D_{AC} + D_{AE} + D_{BC} & \\
+D_{BE} + D_{CD} + D_{DE} &= 2v_1 + 2v_2 + 3v_3 + 2v_4 + 3v_5 + 6v_6 + 4v_7 \\
D_{AB} + D_{AD} + D_{BC} & \\
+D_{CD} + D_{BE} + D_{DE} &= 2v_1 + 3v_2 + 2v_3 + 3v_4 + 2v_5 + 4v_6 + 6v_7
\end{aligned} \tag{11.5}$$

Now suppose that we stack up the D_{ij}, in alphabetical order, into a vector,

$$\mathbf{d} = \begin{bmatrix} D_{AB} \\ D_{AC} \\ D_{AD} \\ D_{AE} \\ D_{BC} \\ D_{BD} \\ D_{BE} \\ D_{CD} \\ D_{CE} \\ D_{DE} \end{bmatrix} \tag{11.6}$$

The coefficients $x_{ij,k}$ can then be arranged in a matrix, each row corresponding to the D_{ij} in that row of $\mathbf{d}$ and containing a 1 if branch k occurs on the path between species i and j. For the tree of Figures 11.1 and 11.2,

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{11.7}$$

Note that the size of this matrix is 10 (the number of distances) by 7 (the number of branches). If we stack up the v_i into a vector, in order of i, equations 11.5 can be expressed compactly in matrix notation as:

$$\mathbf{X}^T\mathbf{D} = \left(\mathbf{X}^T\mathbf{X}\right)\mathbf{v} \tag{11.8}$$

Multiplying on the left by the inverse of $\mathbf{X}^T\mathbf{X}$, we can solve for the least squares branch lengths:

$$\mathbf{v} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{D} \tag{11.9}$$

This a standard method of expressing least squares problems in matrix notation and solving them. When we have weighted least squares, with a diagonal matrix of weights in the same order as the D_{ij}:

$$\mathbf{W} = \begin{bmatrix} w_{AB} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{AC} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{AD} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{AE} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{BC} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{BD} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{BE} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{CD} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{CE} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{DE} \end{bmatrix} \tag{11.10}$$

then the least squares equations can be written

$$\mathbf{X}^T\mathbf{W}\mathbf{D} = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)\mathrm{v} \tag{11.11}$$

and their solution

$$\mathbf{v} = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{D} \tag{11.12}$$

Again, this is a standard result in least squares theory, first used in least squares estimation of phylogenies by Cavalli-Sforza and Edwards (1967).

One can imagine a least squares distance matrix method that, for each tree topology, formed the matrix 11.10 (or 11.7), inverted it, and obtained the estimates in 11.12 (or 11.9). This can be done, but it is computationally burdensome, even if not all possible topologies are examined. The inversion of the matrix $\mathbf{X}^T\mathbf{W}\mathbf{X}$ takes on the order of n^3 operations for a tree of n tips. In principle, this would need to be done for every tree topology considered. Gascuel (1997) and Bryant and Waddell (1998) have presented faster methods of computation that compute the exact solutions of the least squares branch length equations, taking advantage of the structure of the tree. They cite earlier work by Vach (1989), Vach, and Degens (1991), and Rzhetsky and Nei (1993). For a tree with n tips these fast methods save at least a factor of n (and for the unweighted cases, n^2) operations.

I have presented (Felsenstein, 1997) an iterative method for improving branch lengths. It uses a "pruning" algorithm similar to the one which we will see in the next chapter for likelihood. It computes distances between interior nodes in the tree and tips, and between interior nodes. These distances depend on the current estimates of the branch lengths. Using these new distances, improved estimates of branch lengths can then be obtained. The method is of the "alternating least squares" type, in which least squares estimates of some variables are obtained,

given the values of the others, and this is done successively for different variables (branch lengths, in the present case). They converge fairly rapidly on the correct values. Although they are iterative, they do enable us to constrain the branch lengths to be nonnegative, which may be helpful as negative branch lengths have no biological interpretation.

This algorithm uses, at each node in the tree, arrays of distances from there to each other node. These can play the role that the conditional score arrays play in the Fitch and Sankoff algorithms for computing the parsimony score of a tree. Like those, these arrays can be used to economize on computations when rearranging the tree. This is of less use in the least squares distance matrix methods than it is in the parsimony methods, because the branch lengths in a subtree typically do not remain completely unaltered when other regions of the tree are changed. We will see similar quantities when we discuss likelihood methods.

Finding the least squares tree topology

Being able to assign branch lengths to each tree topology, we need to search among tree topologies. This can be done by the same methods of heuristic search that were discussed in Chapter 4. We will not repeat that discussion here. No one has yet presented a branch-and-bound method for finding the least squares tree exactly. Day (1986) has shown that finding the least squares tree is an NP-complete problem, so that polynomial-time algorithms for it are unlikely to exist.

Note that the search is not only among tree topologies, but also among branch lengths. When we make a small change of tree topology, the branch lengths of the resulting tree should change mostly in the regions that are altered, and rather little elsewhere. This means that the branch lengths from the previous tree provide us with good starting values for the branch lengths on the altered tree. My own iterative algorithm for estimating branch lengths (Felsenstein, 1997) retains partial information at interior nodes of the tree. Thus we not only retain the previous branch lengths, but we do not need to recompute the partial information at the interior nodes, at least not the first time they are used. Another iterative algorithm for estimating branch lengths is described by Makarenkov and Leclerc (1999).

We defer coverage of the highly original least squares method of De Soete (1983) until the next chapter, as it uses quartets of species.

The statistical rationale

The impetus behind using least squares methods is statistical. If the predicted distances are also expected distances, in that each distance has a statistical expectation equal to its prediction on the true tree (equal to the sum of the intervening branch lengths), then we can imagine a statistical model in which the distances vary independently around their expectations and are normally distributed around them. If this were true, the proper least squares estimate would minimize the sum of squares of the standardized normal deviates corresponding to the different distances. The deviation of an individual distance from its expectation would be

$D_{ij} - \mathbb{E}(D_{ij})$, and the variance of this quantity would be Var (D_{ij}). We can make a squared standardized normal variate by dividing the square of the deviation by the variance. The sum of squares would then be

$$Q = \sum_{i=1}^{n} \sum_{j:j \neq i} \frac{[D_{ij} - \mathbb{E}(D_{ij})]^2}{\text{Var } (D_{ij})} \tag{11.13}$$

The expectation $\mathbb{E}(D_{ij})$ is computed from the predicted distance, the result of summing branch lengths between the species. The variance in the denominator depends on the details of the process that produced these distances. In effect, Cavalli-Sforza and Edwards's least squares methods are assuming equal variances for all the distances, and Fitch and Margoliash are assuming that the error (and hence the standard deviation) is proportional to the distance. Fitch and Margoliash approximate the variance (the square of that standard deviation) by using the square of the observed distance.

The problem with this framework is the assumption that the observed distances vary independently around their expectations. If the distances are derived from molecular sequences, they will not vary independently, as random evolutionary events on a given internal branch of the tree can simultaneously inflate or deflate many distances at the same time. The same is true for distances for restriction sites and gene frequencies. DNA hybridization techniques would seem to be likely to satisfy the assumption, however. Their errors have much more to do with experimental error than with random evolutionary events. But alas, DNA hybridization values are computed by standardizing them against hybridizations of a species against its own DNA, and those standards are shared by multiple hybridization values. The result is a lack of independence even in this case.

Fortunately, it can be shown that least squares methods that do not have corrections for the correlations among data items will nevertheless at least make consistent estimates, that they will converge to the true tree as the size of data sets becomes large, even if the covariances are wrongly assumed to be zero and the variances are wrongly estimated. All that is necessary is that the expectations be correct.

I have discussed this approach to justifying distance matrix methods (Felsenstein, 1984), pointing out that it does not require that there be any paths through the data space to the observed data that exactly achieve the estimated branch lengths. For a contrary view see Farris's arguments (Farris, 1981, 1985, 1986) and my reply (1986).

Generalized least squares

The least squares methods as formulated above ignore the correlations between different distances. It is possible to modify the methods, in straightforward fashion, so that they take the correlations into account. This should be statistically

preferable. However, one pays a large computational cost for taking the correlations into account. Chakraborty (1977) presented a least squares method for trees under a molecular clock. He assumed that the covariances of distances were proportional to the shared path length on the paths connecting the two pairs of species. This would be true if evolution were a Poisson process, in which there were random events occurring along the paths, with variances and covariances determined by the number of events. This is approximately true for small amounts of divergence. However, he estimated the divergence times by ordinary unweighted least squares, using the covariances only in the computation of the standard errors of the divergence times.

Hasegawa, Kishino, and Yano (1985) used an explicit model of DNA evolution to derive expressions for the variances and covariances of the distances, and they based a generalized least squares method on this. Bulmer (1991) used the Poisson process approximation, basing a generalized least squares analysis on it.

These methods require more computation than ordinary least squares. The equations are similar to 11.12 and 11.9, but the diagonal array of weights, $\mathbf{W}$, must be replaced by the inverse of the covariance matrix of the distances:

$$\mathbf{X}^T\mathbf{V}^{-1}\mathbf{D} = \left(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\right)\mathbf{v} \tag{11.14}$$

and their solution

$$\mathbf{v} = \left(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{V}^{-1}\mathbf{D} \tag{11.15}$$

The inverse of the covariance matrix $\mathbf{V}$ is inversion of an $n(n+1)/2 \times n(n+1)/2$ matrix. For 20 species, for example, this would be a 190×190 matrix. This must be done for each tree topology examined. Matrix inversion requires an effort proportional to the cube of the number of rows (or columns) of the matrix. Thus the naive cost of finding branch lengths for a least squares tree of given topology would be proportional to n^6. However, Bryant and Waddell (1998) have described a more efficient algorithm that reduces the cost to n^4.

Distances

In order for distances that are used in these analyses to have the proper expectations, it is essential that they are expected to be proportional to the total branch length between the species. Thus, if in one branch a distance X is expected to accumulate and on a subsequent branch a distance Y, then when the two branches are placed end-to-end the total distance that accumulates must be expected to be $X + Y$. It need not be $X + Y$ in every individual case, but it must be in expectation. It is not proper to use any old distance measure, for this property may be lacking. If the distances do not have this linearity property, then wrenching conflicts between fitting the long distances and fitting the short distances arise, and the tree is the worse for them.

We will give an example of how distances may be computed to make them comply with this requirement, using DNA sequences as our example.

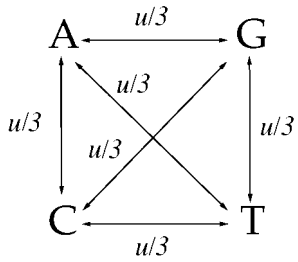


Figure 11.3: The Jukes-Cantor model of DNA change. The rate of change between all pairs of nucleotides is $u/3$ per unit time.

The Jukes-Cantor model—an example

The simplest possible model of DNA sequence evolution is the model of Jukes and Cantor (1969). In this model, each base in the sequence has an equal chance of changing. When one changes, it changes to one of the three other bases with equal probability. Figure 11.3 shows a diagram of this model. The result is, of course, that we expect an equal frequency of the four bases in the resulting DNA. The quantity u that is the rate of change shown on all the arrows is the rate of substitution between all pairs of bases. Note that although this is often miscalled a rate of "mutation," it is actually the rate of an event that substitutes one nucleotide for another throughout a population, or at any rate in enough of the population that it shows up in our sampled sequence. In certain cases of neutral mutation, the rates of substitution and of mutation will be the same.

To calculate distances we need to compute the transition probabilities in this model. Note that this does *not* mean the probabilities of transition rather than transversion; it is much older mathematical terminology, meaning the probability of a transition from one state (say C) to another (say A). The easiest way to compute this is to slightly fictionalize the model. Instead of having a rate u of change to one of the three other bases, let us imagine that we instead have a rate $\frac{4}{3}u$ of change to a base randomly drawn from all four possibilities. This will be exactly the same process, as there then works out to be a probability $u/3$ of change to each of the other three bases. We have also added a rate $u/3$ of change from a base to itself, which does not matter.

If we have a branch along which elapsed time is t, the probability in this fictionalized model that there are no events at all at a site, when the number expected to occur is $\frac{4}{3}ut$, is the zero term of a Poisson distribution. We can use that distribution because we take time to be continuous, and the branch of length t consists then of a vast number of tiny segments of length dt each, each having the small probability $\frac{4}{3}u\ dt$ of an event. The probability of no event is then

$$e^{-\frac{4}{3}ut}$$

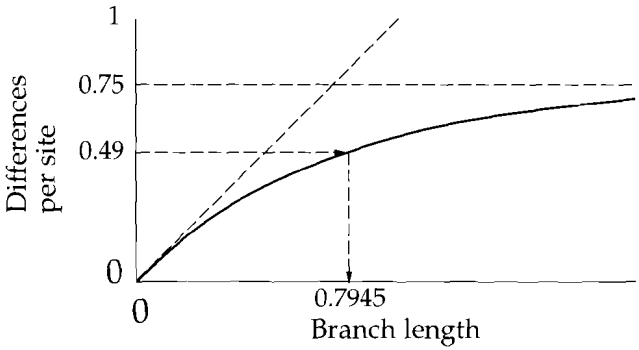


Figure 11.4: The expected difference per site between two sequences in the Jukes-Cantor model, as a function of branch length (the product of rate of change and time). The process of inferring the branch length from the fraction of sites that differ between two sequences is also shown.

The probability of at least one event is the complement of this,

$$1 - e^{-\frac{4}{3}ut}$$

If there is an event, no matter how many there are, the probability that the last one resulted in a particular nucleotide is then 1/4. So, for example, the probability of C at the end of a branch that started with A is

$$\text{Prob}(C|A,u,t) = \frac{1}{4}\left(1 - e^{-\frac{4}{3}ut}\right) \tag{11.16}$$

As there are three other nucleotides to which the A could have changed, the probability that this site is different at the two ends of the branch is the sum of three such quantities, being

$$D_S = \frac{3}{4}\left(1 - e^{-\frac{4}{3}ut}\right) \tag{11.17}$$

Figure 11.4 shows this curve of difference against ut. Note that it plateaus at 3/4. This is what we expect; when a sequence changes by so much that it is unrelated to its initial sequence, there are still 1/4 of the sites at which it happens to end up in the same state as when it started.

Note that if we try to use the difference per site, which is the vertical axis of Figure 11.4, it will certainly rise linearly with branch length. As it flattens out at 3/4, it will accumulate less and less difference with successive branches traversed. If we have two branches that each would, individually, lead us to expect 0.20 difference from one end of the branch to the other, when combined they will in fact only lead to an expected distance of 0.34666. The differences will not be additive at all.

The easiest way to find an additive distance is simply to use the difference per site to estimate ut itself. The value ut is the product of the rate of change and the time. It is the expected number of changes along the branch, counting both those that end up being visible to us, and those that do not. We call the value of ut for a branch the *branch length*. The values of ut on each branch will, by definition, add perfectly. Figure 11.4 shows this estimation. Starting with a value 0.49 of the difference, the dashed arrows show the process of estimating ut. The resulting estimate, 0.7945, is in effect the difference corrected for all the events that are likely to have occurred, but could not be seen. They include changes that overlay others, or even reversed their effects.

The formula for this estimation is easily derived from equation 11.17. It is:

$$D = \widehat{ut} = -\frac{3}{4}\ln\left(1 - \frac{4}{3}D_S\right) \tag{11.18}$$

This is actually a maximum likelihood estimate of the distance, it turns out. Its one tiresome aspect is that it becomes infinite when the difference between sequences becomes greater than 3/4. That cannot occur in the data if infinitely long sequences follow the Jukes-Cantor model, but it can certainly happen for finite-length sequences, simply as a result of random sampling.

Although the result of these calculations is called a distance, it does not necessarily satisfy all the requirements that mathematicians make of a distance. One of the most important of these is the Triangle Inequality. This states that for any three points A, B, and C,

$$D_{AC} \leq D_{AB} + D_{BC} \tag{11.19}$$

A simple example of violation of the Triangle Inequality is three DNA sequences 100 bases in length, with 10 differences between the A and B, and 10 differences between B and C, those being at different sites. Thus A and C differ at 20 sites. Using equation 11.18, $D_{AB} = D_{BC} = 0.107326$ and $D_{AC} = 0.232616$, which violates the inequality. Thus we can call the number a distance in a biological sense, but not in a mathematical sense. Fortunately, most distance matrix methods do not absolutely require the Triangle Inequality to hold.

Why correct for multiple changes?

The Jukes-Cantor distance does not simply compute the fraction of sites that differ between two sequences. Like all the distances we will encounter, it attempts a correction for unobserved substitutions that are overlain or reversed by others. Why is this necessary? The first impulse of many biologists is to use the uncorrected differences as distances. This is dangerous.

An example is given in Figure 11.5. The original tree is shown on the left. Under the Jukes-Cantor model, the uncorrected fractions of sequence difference predicted from this tree are shown in the table in the center. If these are used with the unweighted least squares method, the tree on the right is obtained. It has the

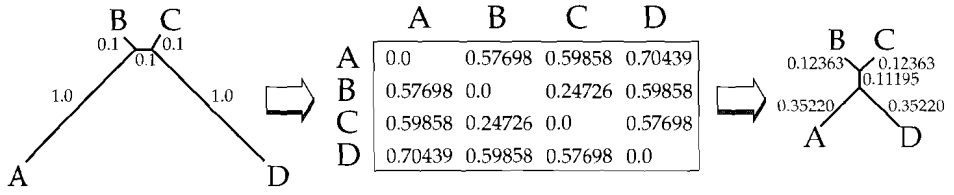


Figure 11.5: An example of distortion of tree topology when uncorrected distances are used. The true tree is on the left, the expected uncorrected sequence differences under a Jukes-Cantor model are shown in the table in the center. The least squares tree from those differences is shown on the right. It incorrectly separates B and C from A and D.

wrong topology, most likely because the tips A and D are trying to get close to each other harder than either is to get close to B or C. There is a battle between the long and short distances, with the lack of correction making the long distances try harder to shorten the corresponding branches.

The example is what we would see if we used infinitely long sequences, but without correction of the distances for multiple changes. Despite the infinitely long sequences, we get an incorrect topology. Of course, if the corrected Jukes-Cantor distance were used, there would be a perfect recovery of the true tree, as the distances would be the sums of branch lengths along that tree. By contrast, if we use the Jukes-Cantor correction, we approach the true branch lengths as more and more DNA is sequenced, and the correct left-hand tree is found.

One case in which correction is unnecessary is when the trees are clocklike. The deeper is the common ancestor of two species, the greater will be the expected difference between their sequences. Correction for multiple changes will not alter the ranking of the distances, and distance matrix methods that assume a clock will tend to find the same topology whether or not there is correction of the distances. Rzhetsky and Sitnikova (1996) show this in simulations, where failure to correct distances has serious consequences in nonclocklike cases, but does not cause serious problems when there is a molecular clock.

Minimum evolution

Having seen the computational methods and biological justification of the least squares methods, we now look at distance matrix methods that do not use the least squares criterion. Some use other criteria; others are defined by an algorithm for constructing the tree and do not use an explicit criterion.

The *minimum evolution method* (ME) uses a criterion, the total branch length of the reconstructed tree. It is not to be confused with the "minimum evolution" method of Edwards and Cavalli-Sforza (1964) which was the first parsimony method. One might think that the minimum evolution tree should simply be a

tree all of whose branches are of length 0. That would be the case if the tree were unconstrained by the data. In the minimum evolution method the tree is fit to the data, and the branch lengths are determined, using the unweighted least squares method. The least squares trees are determined for different topologies, and the choice is made among them by choosing the one of shortest total length. Thus this method makes partial use of the least squares criterion. In effect it uses two criteria at the same time, one for choosing branch lengths, another for choosing the tree topology.

This minimum evolution method was first used by Kidd and Sgaramella-Zonta (1971), who used the sum of absolute values of branch lengths. Its present-day use comes from its independent invention by Rzhetsky and Nei (1992, 1993, 1994). They used the sum of branch lengths. Trees with negative branches thus tend to attract the search, and heuristic tree rearrangement may spend considerable time among them. Kidd and Sgaramella-Zonta suggested that if there is any tree topology that has all positive estimated branch lengths, then the best solution by their method would also have no negative branch lengths.

Rzhetsky and Nei showed that if the distances were unbiased estimates of the true distance (many distances are not unbiased) then the expected total length of the true tree was shorter than the expected total length of any other. However, this is not the same as showing that the total length is always shortest for the true tree, as the lengths vary around their expectation. It would be impossible for it to be true that the total length is always shorter for the true tree, as that would establish that this particular criterion always triumphs over statistical noise! Their result is meaningful if one reduces all the information in the data to one quantity, the estimated length of the tree. Even then, having its expectation be least for the true tree is not the same as showing that the use of the minimum evolution criterion makes a maximum likelihood estimate given the tree length. For that we would need to know that this quantity was normally distributed, and had equal variances for all tree topologies. It is not clear whether minimum evolution methods always have acceptable statistical behavior. Gascuel, Bryant, and Denis (2001) have found cases where minimum evolution is inconsistent when branch lengths are inferred by weighted least squares or by generalized least squares.

Minimum evolution requires an amount of computation similar to least squares, since it uses least squares to evaluate branch lengths for each tree topology. The methods of Bryant and Waddell (1998) for speeding up least squares calculations will thus speed up minimum evolution methods as well. Kumar (1996) has described search methods that improve on Rzhetsky and Nei's. Rzhetsky and Nei (1994) describe the use of bootstrap support of branches (which I describe in Chapter 20) to guide the search for branches where the tree topology should be reconsidered. Desper and Gascuel (2002) have found that using a "greedy" search of tree topologies and a somewhat approximate version of minimum evolution led to great increases in speed with good accuracy of the resulting trees.

An early variant on Minimum Evolution that did not use least squares to infer the branch lengths was given by Beyer et al. (1974; Waterman et al., 1977). They instead required that the path lengths between all pairs of species remain longer than, or equal to, the observed distances. This makes the inference of branch lengths a linear programming problem. Their inequality is justified in the case of closely related molecular sequences, where the total branch length will approximate a parsimony criterion. Like a parsimony criterion, their method may fit branch lengths that are substantially shorter than is plausible when the sequences are quite different.

Clustering algorithms

The methods mentioned so far optimize a criterion such as the sum of squares, searching among all trees for the tree with the best value. Another class of distance matrix methods does not have an explicit criterion, but instead applies a particular algorithm to a distance matrix to come up with a tree more directly. This can be quite a lot faster, but it has the disadvantage that we are not sure that the distance information is being used fully, and we are not sure what are the statistical properties of the method.

These methods are derived from clustering algorithms popularized by Sokal and Sneath (1963). Chief among them is the UPGMA method, whose name is an acronym for its name in their classification of indexclusteringclustering methods. UPGMA can be used to infer phylogenies if one can assume that evolutionary rates are the same in all lineages.

UPGMA and least squares

One can constrain the branch lengths so that they satisfy a "molecular clock." Trees that are clocklike are rooted and have the total branch length from the root up to any tip equal. They are often referred to as being *ultrametric*. When a tree is ultrametric, it turns out to be extremely simple to find the least squares branch lengths. The total branch length from a tip down to any node is then the average of the distances between all the pairs of species whose most recent common ancestor is that node. Thus if a node leads to two branches, one of which leads on upwards to all mammals and the other on upwards to all birds, the estimate of the total branch length down to the node is the average of the distances between all (bird, mammal) pairs. The weights w_{ij} are used to weight this average.

The branch lengths are then the differences between these total branch lengths. If they give a negative branch length, it may be necessary to set that branch length to zero, which combines two nodes, and recompute the associated sums of branch lengths. Farris (1969a) was the first to note this relationship between averages and least squares branch lengths.

A clustering algorithm

Done this way, finding ultrametric trees has the same search problems as other phylogeny methods. However, there is a simple algorithm that can be used to quickly construct a clocklike phylogeny—the *UPGMA* or average linkage method. It is not guaranteed to find the least squares ultrametric phylogeny, but it often does quite well. This algorithm was introduced by Sokal and Michener (1958) — it belongs to the class of phenetic clustering methods that were predecessors of most modern phylogeny methods. It has been rather extensively criticized in the phylogeny literature, but if a clock is thought to be a reasonable assumption (and it often is if the species are closely related), then UPGMA is a well-behaved method.

The algorithm works on a distance matrix and also keeps track, for each species or group, of the number, n_i, of species in the group. These are initially all 1. The steps in the algorithm are:

1. Find the i and j that have the smallest distance, D_{ij}.
2. Create a new group, (ij), which has $n_{(ij)} = n_i + n_j$ members.
3. Connect i and j on the tree to a new node [which corresponds to the new group (ij)]. Give the two branches connecting i to (ij) and j to (ij) each length $D_{ij}/2$.
4. Compute the distance between the new group and all the other groups (except for i and j) by using:

$$D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$$

5. Delete the columns and rows of the data matrix that correspond to groups i and j, and add a column and row for group (ij).
6. If there is only one item in the data matrix, stop. Otherwise, return to step 1.

This method is easy to program and takes about n^3 operations to infer a phylogeny with n species. Each time we look for the smallest element in the distance matrix, we need a number of operations proportional to n^2, and we do this $n - 1$ times. However, we can speed things up by a large factor by simply retaining a list of the size and location of the smallest elements in each row (or column). Finding the smallest element in the matrix then requires a number of operations proportional to n rather than n^2. With each clustering, this list of minima can be updated in a number of operations proportional to n, so that the whole algorithm can be carried out in a number of operations proportional to n^2. It can be shown never to give a negative branch length.

An example

Using immunological distances from the work of Sarich (1969) we can show the steps involved in inferring a tree by the UPGMA method. The amount of work

needed is so small that we can carry out the clustering by hand. Here is the original distance matrix, which has the distances corrected logarithmically to allow for a presumed exponential decay of immunological similarity with branch length. We start by looking for the smallest distance. In this table it is marked by a box, and the elements of those rows and columns are indicated in boldface and by asterisks at the borders of the table.

						*	*		
		dog	bear	raccoon	weasel	**seal**	**sea lion**	cat	monkey
	dog	0	32	48	51	**50**	**48**	98	148
	bear	32	0	26	34	**29**	**33**	84	136
	raccoon	48	26	0	42	**44**	**44**	92	152
	weasel	51	34	42	0	**44**	**38**	86	142
*	**seal**	**50**	**29**	**44**	**44**	**0**	**24**	**89**	**142**
*	**sea lion**	**48**	**33**	**44**	**38**	**24**	**0**	**90**	**142**
	cat	98	84	92	86	**89**	**90**	0	148
	monkey	148	136	152	142	**142**	**142**	148	0

Combining the rows for seal and sea lion, we average their distances to all other species.

						*	*		
		dog	bear	raccoon	weasel	**seal**	**sea lion**	cat	monkey
	dog	0	32	48	51	**50**	**48**	98	148
	bear	32	0	26	34	**29**	**33**	84	136
	raccoon	48	26	0	42	**44**	**44**	92	152
	weasel	51	34	42	0	**44**	**38**	86	142
*	**seal**	**50**	**29**	**44**	**44**	**0**	**24**	**89**	**142**
*	**sea lion**	**48**	**33**	**44**	**38**	**24**	**0**	**90**	**142**
	cat	98	84	92	86	**89**	**90**	0	148
	monkey	148	136	152	142	**142**	**142**	148	0

After we do this, we infer the immediate ancestor of seal and sea lion to be 12 units of branch length from each, so that the distance between them is 24. The new combined row and column (marked SS) replaces the seal and sea lion rows and columns. This reduced table has its smallest element marked by a box. It involves

bear and raccoon, and those rows and columns are boldfaced and indicated by asterisks:

		dog	* **bear**	* **raccoon**	weasel	SS	cat	monkey
	dog	0	**32**	**48**	51	49	98	148
*	**bear**	**32**	**0**	**26**	**34**	**31**	**84**	**136**
*	**raccoon**	**48**	**26**	**0**	**42**	**44**	**92**	**152**
	weasel	51	**34**	**42**	0	41	86	142
	SS	49	**31**	**44**	41	0	89.5	142
	cat	98	**84**	**92**	86	89.5	0	148
	monkey	148	**136**	**152**	142	142	148	0

Again, we average the distances from bear and from raccoon to all other species, and we infer their common ancestor to have been 13 units of branch length below each of them. We replace their rows and columns by a new one, BR:

		dog	* **BR**	weasel	* **SS**	cat	monkey
	dog	0	**40**	51	**49**	98	148
*	**BR**	**40**	**0**	**38**	**37.5**	**88**	**144**
	weasel	51	**38**	0	**41**	86	142
*	**SS**	**49**	**37.5**	**41**	**0**	**89.5**	**142**
	cat	98	**88**	86	**89.5**	0	148
	monkey	148	**144**	142	**142**	148	0

The smallest element in this table was 37.5, between BR and SS. The ancestor of these two groups is inferred to be 18.75 units of branch length below these four species. It is thus 5.75 below the ancestor of bear and raccoon, and 6.75 below the ancestor of seal and sea lion. You should refer to Figure 11.6 to see the branches and branch lengths that are added to the tree by each step. Each of the groups BR and SS is a group with two species, so the proper average is again a simple average of their distances to other species:

		dog	* **BRSS**	* **weasel**	cat	monkey
	dog	0	**44.5**	**51**	98	148
*	**BRSS**	**44.5**	**0**	**39.5**	**88.75**	**143**
*	**weasel**	**51**	**39.5**	**0**	**86**	**142**
	cat	98	**88.75**	**86**	0	148
	monkey	148	**143**	**142**	148	0

Now the smallest distance, 39.5, is between BRSS and weasel. One is a group of four species, the other a single species. In averaging their distances to all other species, we do not do a simple average, but weight the distance to BRSS four times as much as the distance to weasel. For example, the distance of the new group to dog is $(4 \times 44.5 + 51)/5 = 45.8$. The new row and column are called BRSSW and replace BRSS and weasel.

		* dog	* BRSSW	cat	monkey
*	**dog**	**0**	**45.8**	**98**	**148**
*	**BRSSW**	**45.8**	**0**	**88.2**	**142.8**
	cat	**98**	**88.2**	0	148
	monkey	**148**	**142.8**	148	0

Now dog joins BRSSW, and the average of those rows and columns is again a weighted average, weighting BRSSW five times more heavily than dog.

		* DBRWSS	* cat	monkey
*	**DBRWSS**	**0**	**89.833**	**143.66**
*	**cat**	**89.833**	**0**	**148**
	monkey	**143.66**	**148**	0

With only three groups left, cat joins up next. Finally, we have only two groups which must, of course, join one another.

	DBRWSSC	monkey
DBRWSSC	0	144.2857
monkey	144.2857	0

The final tree is shown in Figure 11.6. It is fairly close to biological plausibility.

UPGMA on nonclocklike trees

The main disadvantage of UPGMA is that it can give seriously misleading results if the distances actually reflect a substantially nonclocklike tree. Figure 11.7 shows a small set of imaginary distances, which are derived from a nonclocklike tree by adding up branch lengths. Also shown is the resulting UPGMA tree. It first clusters species B and C, which creates a branch (the one separating them from A and D) that is not on the true tree. For this problem to arise, evolutionary rates

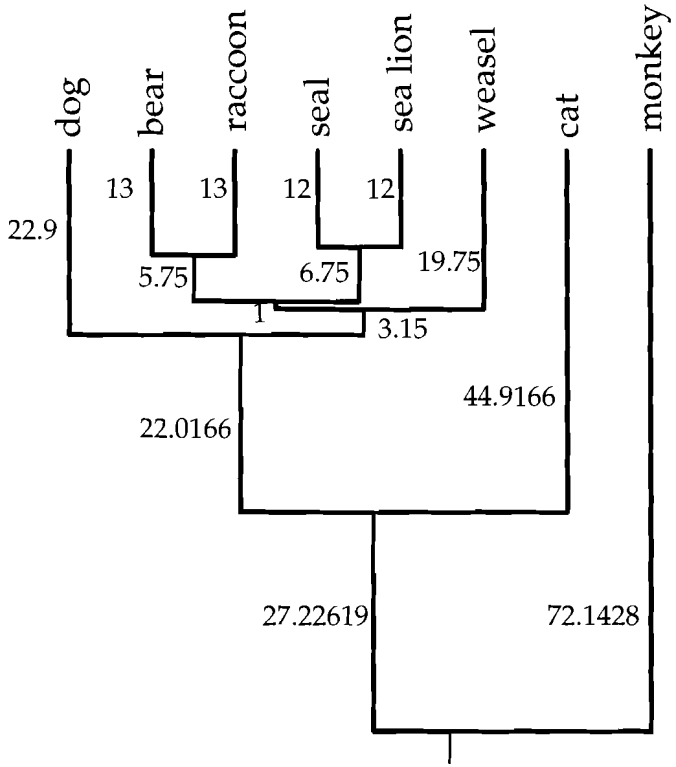


Figure 11.6: The tree inferred by UPGMA clustering of the Sarich (1969) immunological distance data set.

on different branch lengths must differ by at least a factor of two. Note that this is not long branch attraction. In fact, it is short branch attraction: B and C are put together because they are similar in not having changed.

Neighbor-joining

The *neighbor-joining* (NJ) algorithm of Saitou and Nei (1987) is another algorithm that works by clustering. It does not assume a clock and instead approximates the minimum evolution method. (It may also be thought of as a rough approximation to least squares.) The approximation is in fact quite good, and the speed advantage of neighbor-joining is thus not purchased at much cost. It is practical well into the hundreds of species.

Neighbor-joining, like the least squares methods, is guaranteed to recover the true tree if the distance matrix happens to be an exact reflection of a tree. Thus for

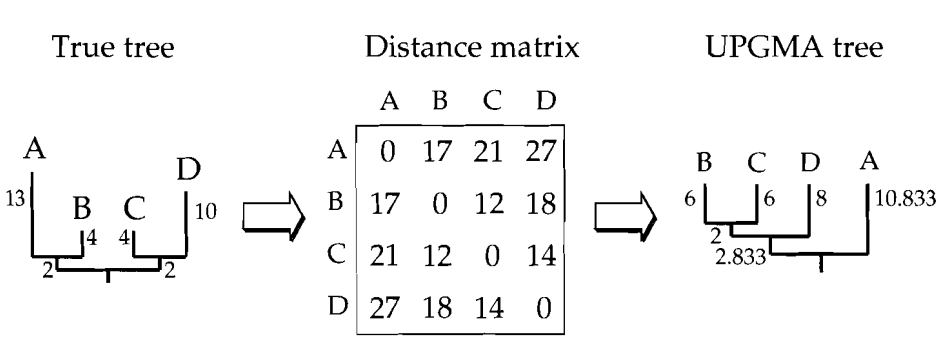


Figure 11.7: A four-species, nonclocklike tree and the expected data matrix it yields, when distances are the sums of branch lengths. The tree estimated by applying the UPGMA method to this distance matrix is shown—it does not have the correct tree topology. In both trees the branch lengths are proportional to the vertical length of the branches.

the data matrix of Figure 11.7 it simply recovers the true tree. The algorithm is (as modified by Studier and Keppler, 1988):

1. For each tip, compute $u_i = \sum_{j:j\neq i}^{n} D_{ij}/(n-2)$. Note that the denominator is (deliberately) not the number of items summed.

2. Choose the i and j for which $D_{ij} - u_i - u_j$ is smallest.

3. Join items i and j. Compute the branch length from i to the new node (v_i) and from j to the new node (v_j) as

$$\begin{aligned} v_i &= \tfrac{1}{2}D_{ij} + \tfrac{1}{2}(u_i - u_j) \\ v_j &= \tfrac{1}{2}D_{ij} + \tfrac{1}{2}(u_j - u_i) \end{aligned}$$

4. Compute the distance between the new node (ij) and each of the remaining tips as

$$D_{(ij),k} = (D_{ik} + D_{jk} - D_{ij}) \Big/ 2$$

5. Delete tips i and j from the tables and replace them by the new node, (ij), which is now treated as a tip.

6. If more than two nodes remain, go back to step 1. Otherwise, connect the two remaining nodes (say, ℓ and m) by a branch of length $D_{\ell m}$.

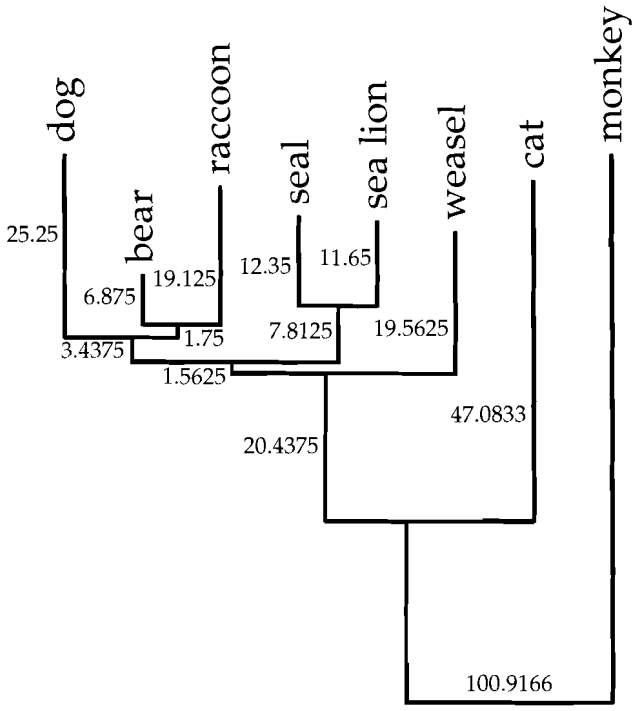


Figure 11.8: The neighbor-joining tree for the data set of Sarich (1969) rooted at the midpoint of the longest path between species. It may be compared with Figure 11.6.

We will not show the steps in detail for the Sarich data set but only the result of applying neighbor-joining to that data set, in Figure 11.8. The midpoint rooting method used is due to Farris (1972). Unlike the UPGMA algorithm, the neighbor-joining algorithm is not carried out in a number of operations proportional to n^2. Current algorithms use a number of operations proportional to n^3, owing to the necessity of updating the u_i and subtracting them from the distances.

Performance

Computer simulation studies have shown that neighbor-joining performs quite well. Although it has sometimes been claimed to perform better than the Fitch-Margoliash method, this seems not to be the case, although the difference is not great (Kuhner and Felsenstein, 1994). When the decision is made as to which pair of tips to cluster, ties are possible. Backeljau et al. (1996) have raised this issue and examined how well various implementations of neighbor-joining cope with ties. Farris et al. (1996) noted that when neighbor-joining is used together with bootstrap resampling, an arbitrary resolution of ties can produce the appearance

of strong support for a grouping when there is none. Takezaki (1998) finds that this problem is serious only for short sequences or closely related species and can be avoided if one randomly chooses among tied alternatives. This can be done by ensuring that each bootstrap replicate is analyzed with a different input order of species.

Using neighbor-joining with other methods

Neighbor-joining is also useful to rapidly search for a good tree that can then be improved by other criteria. Pearson, Robins, and Zhang (1999) use it while retaining nearly-tied trees, choosing among them by minimum evolution or least squares criteria. Ota and Li (2000, 2001) use neighbor-joining and bootstrapping to find an initial tree and identify which regions of it are candidates for rearrangement (as Rzhetsky and Nei, 1994, did for minimum evolution). They then use maximum likelihood (which is described in Chapter 16) for further search. This results in a substantial improvement in speed over pure likelihood methods.

Relation of neighbor-joining to least squares

There is a relationship between neighbor-joining and other methods, though this is not immediately obvious from the algorithm. It belongs to the class of clustering methods that are defined by the precise algorithm rather than by a criterion. If we made a change in the neighbor-joining algorithm that resulted in somewhat different trees, we could not argue that we were doing better at accomplishing the objectives of neighbor-joining. By definition, anything that results in a different tree is not neighbor-joining. In comparison, methods that minimize a criterion always allow us the possibility that we could find an algorithm that does a better job of searching for the tree that achieves the minimum value.

The relation of neighbor-joining to other methods may not be clear from its presentation in the papers of Saitou and Nei (1987) and Studier and Keppler (1988). In their equations, observed distances are used as if they were equal to the sums of branch lengths, when in fact only their expectations are. There is acknowledgment in these papers that observed distances do differ from these expectations, but little more than that. Nevertheless, Saitou and Nei (1987) do establish one connection. In an extensive appendix they show that, for a given pair of species that is to be indexclusteringclustered, the branch lengths inferred are those that would be assigned by unweighted least squares. At first sight this appears to establish neighbor-joining as a more approximate version of least squares. In Chapter 4 I mentioned star-decomposition search, in which an unresolved starlike tree is gradually resolved by clustering pairs of species. We might think that neighbor-joining is simply a star-decomposition search using the unweighted least squares criterion.

It would be, if it used least squares to choose which pair of tips to join. But although it computes their branch lengths by least squares, it does not use the sum of squares as the criterion for choosing which pair of species to join. Instead it

uses the total length of the resulting tree, choosing that pair that minimizes this length. This is a form of the minimum evolution criterion. But we also cannot identify neighbor-joining as a star-decomposition search for the minimum evolution tree. Neighbor-joining allows negative branch lengths, while minimum evolution bans them. Thus neighbor-joining has some relation to unweighted least squares and some to minimum evolution, without being definable as an approximate algorithm for either. It would be of interest for someone to see whether a star-decomposition algorithm for least squares, or one for minimum evolution, could be developed that was comparable in speed to neighbor-joining.

Gascuel (1994) has pointed out a relationship between neighbor-joining and the quartets method of Sattath and Tversky (1977).

Weighted versions of neighbor-joining

Two modifications of neighbor-joining have been developed to allow for differential weighting in the algorithm to take into account differences in statistical noise. Gascuel (1997) has modified the neighbor-joining algorithm to allow for the variances and covariances of the distances, in a simple model of sequence evolution. This should correct for some of the statistical error. Gascuel's method, called *BIONJ*, thus comes closer to what generalized least squares would give, though it is, of course, still an approximation. The weights are applied at step 3 of the neighbor-joining algorithm given above.

Subsequently, Bruno, Socci, and Halpern (2000) developed weighted neighbor-joining (the *weighbor* method) which uses weights in the formulas at somewhat different steps, and in a different way. They are used in steps 2 and 3. The weighbor method is justified by appeal to a likelihood argument, but it is not the full likelihood of the data but a likelihood calculated separately for each of a series of overlapping quartets of species, and under the assumption that distances are drawn from a Gaussian (normal) distribution.

I will not attempt a detailed justification of the terms in either BIONJ or weighbor, both for lack of space and because I believe that both are approximate. It would be better to start with a likelihood or generalized least squares method, and then show that a weighted version of neighbor-joining is an approximation to it. This has not been done in either case.

Nevertheless, both methods seem to improve on unweighted neighbor-joining. In Gascuel's BIONJ method, the variances and covariances of the distances are taken to be proportional to the branch lengths. The variance of D_{ij} is taken as proportional to branch length between species i and j. The covariance of D_{ij} and $D_{k\ell}$ is taken to be proportional (with the same constant) to the total shared branch length on the paths i–j and k–ℓ. This is usually a good approximation provided the branch lengths are not too long. Gascuel (2000) presents evidence that BIONJ can outperform minimum evolution.

Bruno, Socci, and Halpern's weighbor method uses the exact formula for the variance of a Jukes-Cantor distance instead. This is approximate for other models

of DNA change, but more correctly copes with the very high variances of distances when tips are far apart on the tree. The cost paid for this greater accuracy is that some additional approximations are needed to keep calculation to order n^3. These authors argue that weighbor can find trees more accurately than BIONJ because it is less affected by noise from very large distances. BIONJ should do well when no distances are large, and both should do better than neighbor-joining.

There seems more left to do in developing weighted versions of neighbor-joining that properly reflect the kinds of noise that occur in biological sequence data.

Other approximate distance methods

Before the neighbor-joining method, a number of other approximate distance methods were proposed. Like it, they were defined by their detailed algorithms, not by an explicit criterion. The earlier ones have since largely been superseded by neighbor-joining.

Distance Wagner method

The earliest is Farris's (1972) *distance Wagner method*. This is closely related to his earlier WISS (weighted invariant shared steps) method (Farris, Kluge, and Eckardt, 1970) and his "Wagner method" algorithm (Kluge and Farris, 1969; Farris, 1970) for approximate construction of a most parsimonious tree. The distance Wagner method is intended as an approximation to construction of a most parsimonious tree. Species are added to a tree, each in the best possible place. This is judged by computation of the increase in the length of the tree caused by each possible placement of that species.

The intention is thus similar to that of minimum evolution, but the details are different. Instead of using a least squares reconstruction of the branch lengths, the lengths are computed from distances between pairs of nodes. The distances between the tip species are given, but those between a tip and an interior node, or between two interior nodes, are also computed approximately. These approximate distances between interior nodes, and between interior nodes and tips, determine the branch lengths. The approximation used assumes the Triangle Inequality. Unlike many other distance matrix methods, this restricts the use of the distance Wagner method to distances that satisfy the Triangle Inequality, which many biological distance measures do not.

An exposition of the details of the distance Wagner method will be found in the book by Nei (1987, pp. 305-309). Modifications of the distance Wagner method have also been proposed (Tateno, Nei, and Tajima, 1982; Faith, 1985).

A related family

Another family of approximate distance matrix methods (Farris, 1977b; Klotz et al., 1979; Li, 1981) uses a reference species to correct distances between species