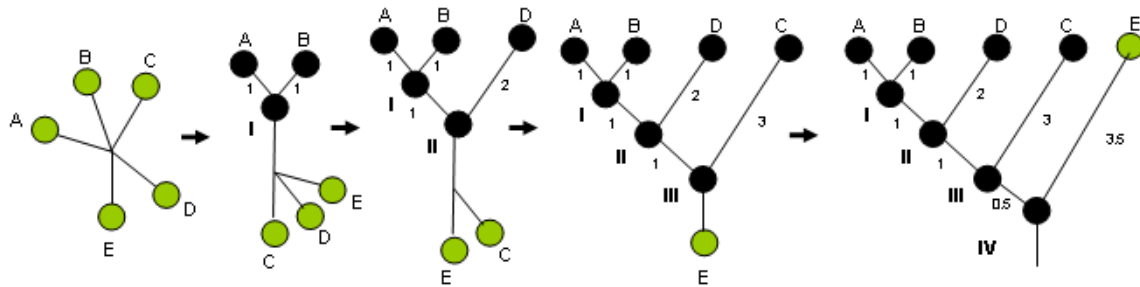


5.2 Local methods — UPGMA (Unweighted Pair Group Method)

5.2.1. The basic idea

UPGMA is a method that uses a *local objective* function to build a rooted bifurcating tree. [Figure 5.2](#) gives an overview of the progressive clustering of five sequences A, B, C, D, E, F into a UPGMA tree. The local objective function clusters the least dissimilar sequences at each step of the procedure. In the example we study below, A and B are clustered first, then D joins them in the cluster, then C and finally E.

5.2



5.2.2 How it works

With five sequences there are 10 pairwise comparisons of distance (the distance between A and B, between A and C, A and D, and so on). The total number of pairwise comparisons in general is given by the formula $n(n-1)/2$, where n is the number of taxa. Suppose we have a d_{ij} matrix of 10 distances as shown in [Figure 5.3](#). We begin our study of UPGMA with this matrix. As clustering proceeds, the matrix is modified to include distance values between clustered and unclustered sequences.

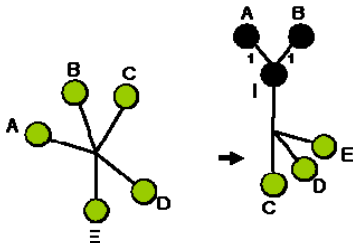
5.3

d_{ij}	
A	
B	2
C	6 6
D	4 4 6
E	7 7 9 5

A B C D

The first step in building a UPGMA tree is to look at the d_{ij} matrix and cluster the two most similar sequences into a partially resolved tree. A and B have the smallest d_{ij} value in the matrix ($d_{AB} = 2$), and so these are clustered as shown in [Figure 5.4](#). This figure shows A and B sharing an ancestral node (I), which is separated from the other sequences by a branch leading to a polytomy of unresolved sequences (C, D, E, F). Note that the newly created ancestral node I has been placed at the midpoint of the path between A and B. Since this pathlength = 2, node I is 1 unit from both A and B.

5.4



The next step in building a UPGMA tree is to calculate the average distance of the members of the cluster AB from all other sequences, and then to include this information in a revised d_{ij} matrix. The average distance from the members of the cluster to a given sequence outside of the cluster is the sum of the distances from each of the clustered sequences to the given sequence divided by the number of paths used in the calculation. That is:

the average distance of the members of the cluster AB from sequence C = $(d_{AC} + d_{BC})/2 = (6 + 6)/2 = 6$

the average distance of the members of the cluster AB from sequence D = $(d_{AD} + d_{BD})/2 = (4 + 4)/2 = 4$

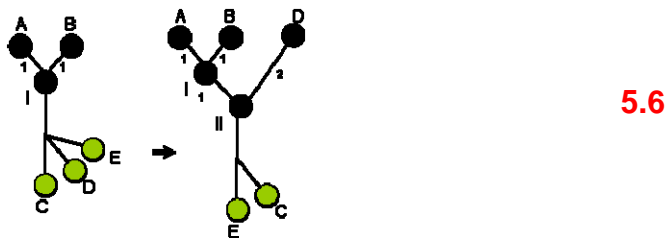
the average distance of the members of the cluster AB from sequence E = $(d_{AE} + d_{BE})/2 = (7 + 7)/2 = 7$

This gives the revised d_{ij} matrix shown in [Figure 5.5](#).

dij **5.5**

AB				
C	6			
D	4	6		
E	7	9	5	
	AB	C	D	

Continuing the clustering, we note that the smallest d_{ij} value in the revised matrix is 4, so we next cluster D and AB to obtain [Figure 5.6](#). The pathlength $d_{D(AB)}$ (average distance of the members of the cluster AB from sequence D) is 4 so we root the path at its midpoint to give the branch lengths as shown.



Next we calculate the average distance of the members of the cluster ABD from each of the unclustered sequences, and include this information in a revised d_{ij} matrix.

The average distance of the members of the cluster ABD from sequence C = $(d_{AC} + d_{BC} + d_{CD})/3 = 6$

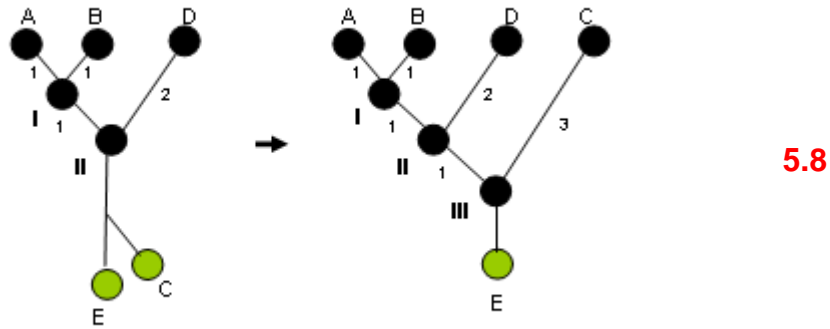
The average distance of the members of the cluster ABD from sequence E = $(d_{AE} + d_{BE} + d_{DE})/3 = 6.3$

The revised matrix is shown in [Figure 5.7](#).

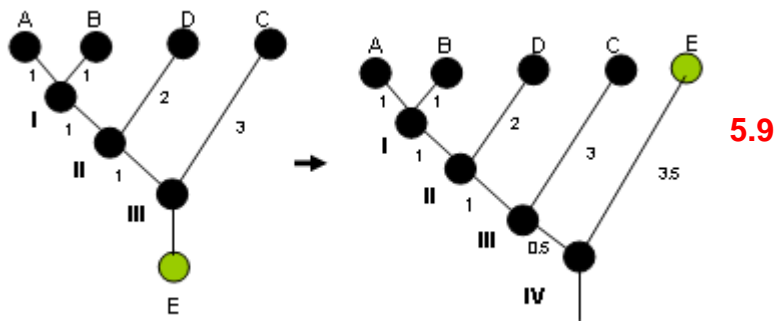
dij **5.7**

ABD			
C	6		
E	6.3	9	
	ABD	C	

The smallest value in this matrix indicates that we next cluster ABD with C. Placing the midpoint root, we obtain **Figure 5.8**.



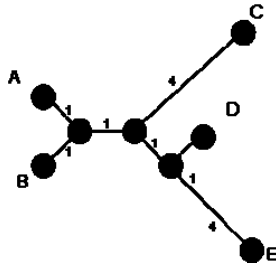
Since there is now only one sequence that remains unclustered (E), we calculate the length of the path from this sequence to the clustered sequences. That is, the average distance of the members of the cluster ABDC from sequence E = $(d_{AE} + d_{BE} + d_{DE} + d_{CE})/4 = 7$. The midpoint of this path identifies the root placement for the universal ancestor of E and ABDC, as shown in **Figure 5.9**.



5.2.3 Discussion

We have been a bit sneaky with this example, because the distances we chose for the example have the property of being *additive* but not *ultrametric* — that is, the sequences have not evolved under a molecular clock (see **Section 4.8**). The distances we chose were actually taken from the unrooted tree shown in **Figure 5.10**. Notice that the pathlengths

in this tree correspond exactly to the distances in the d_{ij} matrix shown in [Figure 5.3](#). This contrasts with the pathlengths in the tree actually recovered by UPGMA.



5.10

You can see that the distances are additive — i.e. the length of the branches that make up individual paths sum to the same value as their corresponding distances in the d_{ij} matrix. However, they are not ultrametric. In practice, it is rare that distances obtained from sequences are ultrametric. Thus, because UPGMA **makes this assumption**, it is seldom used for building phylogenetic trees. Nevertheless, it is useful for introducing other distance methods.

In summary, the method is *heuristic* and *greedy* because the best solution (the smallest d_{ij} value) is chosen at each cycle of clustering. Because of this approach, the method is computationally *efficient* when there are large numbers of sequences to compare. However, the method will be *inconsistent* when distances are not ultrametric. This means that we can expect the method to converge more and more strongly to an incorrect tree as we have longer and longer sequences. The requirement for data being ultrametric is strict, which means that the method is not very *robust* to this type of model misspecification. Computer simulations have suggested that distance methods, including UPGMA, have low *reconstruction accuracy* in comparison to methods that analyse site patterns.

5.3 Neighbor Joining

5.3.1 The basic idea

A distance method that is much more widely used than UPGMA is Neighbor Joining (NJ).

This also uses a greedy algorithm to cluster sequences progressively. It produces a fully resolved *weighted unrooted bifurcating tree*. To reconstruct an accurate tree NJ requires that the distances be additive but not necessarily ultrametric. NJ differs from UPGMA in three important respects:

1. The order of clustering with NJ is based on identifying the most negative value in an S_{ij} matrix, the values for which are calculated from the d_{ij} values.
2. Neighbor Joining uses algorithms that more precisely calculate the lengths of branches in the tree being reconstructed.
3. The resolved bifurcating NJ tree is unrooted, which means that no assumption is made concerning the direction of evolution or location of the oldest node (universal ancestor) on the graph.

5.3.2 How it works

We can study NJ in more detail by analyzing the same data set used for UPGMA. To begin, we need to calculate an S_{ij} matrix for the d_{ij} values shown in [Figure 5.3](#). S_{ij} values measure not only the degree of difference between sequences i and j , they also measure how different these sequences are from all other sequences in the data set.

S_{ij} values are defined as follows:

$$S_{ij} = (N-2) d_{ij} - R_i - R_j$$

where

N = the number of sequences in the d_{ij} matrix

R_i = the sum of the pairwise distances of sequence i from other sequences in the d_{ij} matrix

R_j = the sum of the pairwise distances of sequence j from other sequences in the d_{ij} matrix

The S_{ij} matrix corresponding to the d_{ij} matrix used in the UPGMA example is given in [Figure 5.11](#). For example, $S_{AB} = (5 - 2) 2 - 19 - 19 = -32$. The most negative S_{ij} value in this matrix identifies the sequences to be clustered.

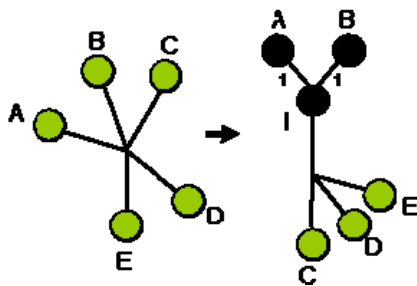
5.11

dij					
a	0				
b	2	0			
c	6	6	0		
d	4	4	6	0	
e	7	7	9	5	0
	a	b	c	d	e

sij					
a	0				
b	-32	0			
c	-28	-28	0		
d	-26	-26	-28	0	
e	-26	-26	-28	-32	0
	a	b	c	d	e

Earlier figures didn't have zeroes on diagonals.

In our example there is a tie. S_{AB} and S_{DE} both have the same most negative number (-32). Where there is a tie, an arbitrary choice needs to be made as to which sequences are to be clustered (A and B, or D and E). In this case choosing either one will still result in the same topology. So, we will cluster A and B. Once the cluster is chosen, the branch lengths between node I and sequence A and between node I and sequence B need to be calculated ([Figure 5.12](#)).



5.12

To do this, the NJ algorithm adds and subtracts d_{ij} values to obtain values for the lengths of these individual branches. The equations are:

$$branch_{I_i} = 1/(2N-4) [(N-2)d_{ij} + R_i - R_j]$$

$$branch_{I_j} = 1/(2N-4) [(N-2)d_{ij} + R_j - R_i]$$

where

N = the number of sequences in the d_{ij} matrix

R_i = the sum of the pairwise distances of sequence i from other sequences in the d_{ij} matrix

R_j = the sum of the pairwise distances of sequence j from other sequences in the d_{ij} matrix

In our example:

$$branch_{I_A} = 1/(2N-4) [(N-2)d_{AB} + R_A - R_B] = 1/6*((3*2)+(2+6+4+7)-(2+6+4+7)) = 1$$

$$branch_{I_B} = 1/(2N-4) [(N-2)d_{AB} + R_B - R_A] = 1/6*((3*2)+(2+6+4+7)-(2+6+4+7)) = 1$$

For the next cycle of NJ, the d_{ij} matrix and S_{ij} matrix must be revised. To update the d_{ij} matrix, the length of the path from the newly added internal node I to each of the unclustered sequences is calculated. The pathlengths are given by the equation:

$$path_{I_k} = \frac{1}{2} * (d_{ik} + d_{jk} - d_{ij})$$

In our example:

$$path_{I_C} = \frac{1}{2} * (d_{AC} + d_{BC} - d_{AB}) = \frac{1}{2} * (6+6-2) = 5$$

$$path_{I_D} = \frac{1}{2} * (d_{AD} + d_{BD} - d_{AB}) = \frac{1}{2} * (4+4-2) = 3$$

$$path_{I_E} = \frac{1}{2} * (d_{AE} + d_{BE} - d_{AB}) = \frac{1}{2} * (7+7-2) = 6$$

The d_{ij} matrix is revised with these values and a corresponding S_{ij} matrix is calculated ([Figure 5.13](#)).

5.13

dij				
I				
c		0		
d		5	0	
e		3	6	0
		6	9	5
				0
		I	c	d
				e

sij				
I				
c		0		
d		-24	0	
e		-22	-22	0
		-22	-22	-24
				0
		I	c	d
				e

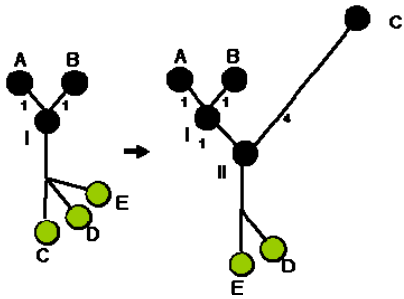
Again there is a tie for the smallest S_{ij} value. An arbitrary choice is made to cluster sequence C with AB ([Figure 5.14](#)). The branch lengths between node II and sequence C and between node II and node I are next calculated.

$$branch_{II_C} = 1/(2N-4) [(N-2)d_{IC} + R_C - R_I] = 4$$

$$branch_{II_I} = 1/(2N-4) [(N-2)d_{IC} + R_I - R_C] = 1$$

No space between II and C etc in previous formulae

Note that C is the new sequence being added to the tree while I is already in the tree, which determines where R_C and R_I appear in these two formulae.



5.14

To update the d_{ij} matrix for the next cycle, the length of the path from the newly added internal node II to each of the unclustered sequences is calculated:

$$path_{II_D} = \frac{1}{2} * (d_{ID} + d_{CD} - d_{IC}) = 2$$

$$path_{II_E} = \frac{1}{2} * (d_{IE} + d_{CE} - d_{IC}) = 5$$

The revised d_{ij} and corresponding S_{ij} matrix are shown in [Figure 5.15](#).

5.15

d _{ij}				
II			0	
d			2	0
e			5	5
		II	d	e

s _{ij}				
II			0	
d			-12	0
e			-12	-12
		II	d	e

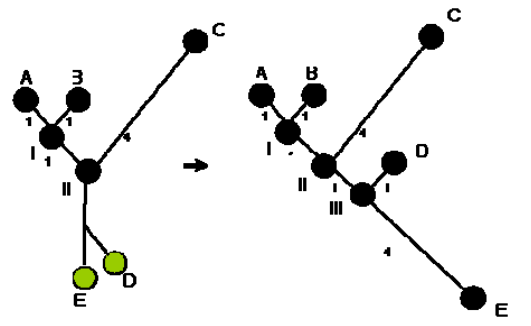
Finally, we create a node III that will allow us to join the last two sequences to the cluster ABC. We then calculate the branch lengths between node III and node II, between node III and D and between node III and E:

$$branch_{III_D} = 1/(2N-4) [(N-2)d_{IID} + R_D - R_{II}] = 1$$

$$branch_{III_II} = 1/(2N-4) [(N-2)d_{IID} + R_{II} - R_D] = 1$$

$$branch_{III_E} = 1/(2N-4) [(N-2)d_{IIE} + R_E - R_{II}] = 4$$

This gives us the tree shown in [Figure 5.16](#).



5.16