Chapter 2

Phylogeny

2.1 Introduction

In this chapter we study algoritmic methods to determine a hierarchical relation between a set of objects. We assume here that this hierarchical relation can be presented by a tree. The original objects are placed at the leaves of the tree, while internal nodes are added to explain the relations between the objects. Similar objects are placed close to one another in the tree, where close means that the distance over the edges that lead from one leaf to the other is small.

In biology the objects considered are usually species. Classically the species are compared on the basis of morphological properties (number of legs, type of skin cover, way offspring is born, ...). At present one tends to use molecular data, comparing specific pieces of DNA or functionally equivalent proteins.

Note that the application of the algorithms presented here is not restricted to biological species. The techniques are examples of clustering algorithms. Examples of objects that have been compared using these techniques are chain letters and natural languages (based on precomputed distances).

The data that is used to construct trees can be devided into two classes.

- *Character based*, values in a discrete domain. Such a character can represent a specific property of an animal (which may be morphological) or it can be the value in a column of a multiple alignment of orthologous genes. In the latter case the character is one of the values A, C, G or T (or sometimes $-$). Each object is represented as a string of symbols, and new strings have to be assigned to the internal nodes such that the differences between adjacent strings are as low as possible.
- *Distance based*, where a matrix of distances between the objects is given, and a tree has to be constructed with weighted edges such that the path lengths in the tree correspond (as much as possible) to the original distances.

2.2 Trees

The trees used in phylogeny can be both rooted and unrooted. A rooted tree is given in Example 2.2 while an unrooted tree is depicted in Example 2.1. In both cases it is usually asumed that the internal nodes have degree three. In the rooted case that means the trees are binary (bifurcating): one incoming edge (except for the root) and two outgoing edges.

Leaves are used to represent the given objects, called *taxa* (singular taxon), short for 'taxonomic units'. Internal nodes are postulated as genetic predecessors to explain the genetic differences and agreements between the taxa.

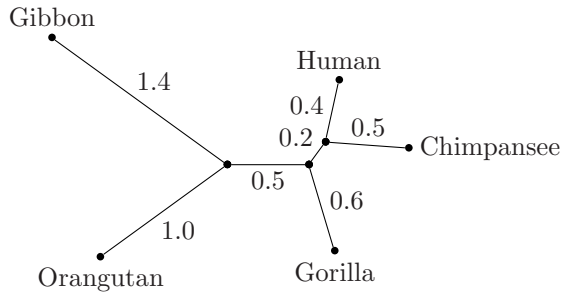
In distance based methods the edges are labelled by numbers. The tree in this way defines a distance between its leaves, adding the edge numbers on the path from one leaf to the another. In the rooted case sometimes a specific property of the distances is required: all leaves are at the same distance from the root.

In phylogenetic tools trees and their branching structure are specified in various formats. A popular one is the *Newick Standard*. Here we start at a node and specify the neighbours as a comma separated list. Each of these neighbours can be a tree itself, hence a list. Edge lengths can be added.

2.1 Example. Consider the tree specification

(Gibbon:1.4, ((Human:0.4, Chimpanzee:0.5):0.2, Gorilla:0.6):0.5, Orangutan:1.0);

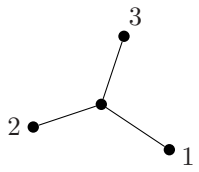
It can be drawn as follows, a typical unrooted tree.



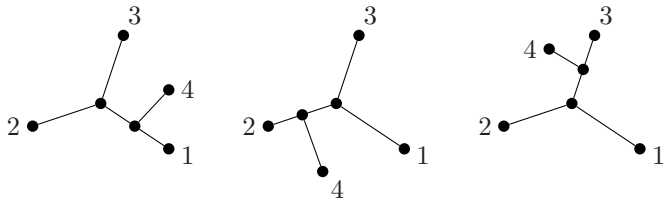
This picture can be obtained in various formats online at
http://www.bioinformatics.nl/tools/plottree.html

Counting Trees. A naive solution to the reconstruction of trees would be to enumerate all trees with the proper number of leaves and stop when one fitting the data is found. In this section we show that the number of (unrooted, leaf-labelled) trees grows very quickly, making this approach infeasible.

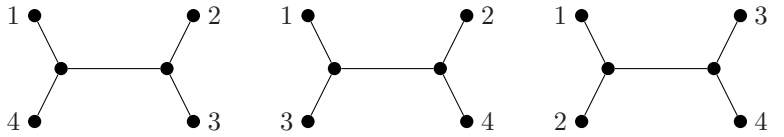
There is a unique labelled tree with three leaves.



Trees with four leaves are constructed from this tree by adding a new edge to one of the three original edges. This can be done in essentially three different ways.



After some rearranging we find the following three possible diagrams. When we look at four different leaves in an arbitrary unrooted tree, and number them by $1, 2, 3, 4$, they will form one of these bifurcation diagrams.



Using $n!!$ to denote the double factorial, which for odd n is defined by $n!! = n \cdot (n-2) \cdot (n-4) \ldots 3 \cdot 1$, we have the following result on the number of trees.

Theorem. The number of labelled, unrooted, binary trees with n leaves equals $(2n-5)!!$.

For $n = 3, 4, \ldots, 12$ these numbers equal 1, 3, 15, 105, 945, 10395, 135135, 2027025, 34459425, 654729075, 13749310575, 316234143225.

2.3 Character Based Methods

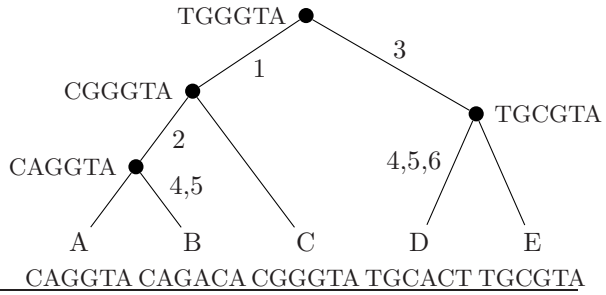
We are given a set of *taxa* together with a set of *characters*. In the small example we have five taxa and six characters. The entries in the matrix indicate the various *character states*. In the rooted tree next to the character matrix four nodes are postulated with their character states. Along the edges the characters are indicated that are changed along the edge. This tree needs a total of eight character switches along the edges to explain the five taxa.

The purpose of character based taxonomy is to construct a tree (rooted or unrooted) that is parsimonous, i.e., has as little character changes as possible. The most general problem is *large parsimony* where both the topology of the tree itself has to be found, and the character states along the tree. Unfortunately large parsimony is an NP complete problem.

If the tree has been obtained by other methods, e.g., by the distance based methods that follow, labelling the tree with appropriate character states is the problem known as *small parsimony*.

2.2 Example.

	1	2	3	4	5	6
A	*C*	*A*	*G*	*G*	*T*	*A*
B	*C*	*A*	*G*	*A*	*C*	*A*
C	*C*	*G*	*G*	*G*	*T*	*A*
D	*T*	*G*	*C*	*A*	*C*	*T*
E	*T*	*G*	*C*	*G*	*T*	*A*



Small parsimony. Given a binary tree topology (branching structure) and character states for each of the leaves (taxa) we have to obtain the character states for the internal nodes. This can be done for each character separately: in order to have a parsimony, each character separately should be optimal.

We generalize the example given above, where each character change was counted. Here there is a weight associated to each change, given as a matrix C, where C_{ij} is the cost of changing from i at the parent to j at the child.

The algorithm given here is originally proposed by Sankoff [10]. It is a fine example of dynamic programming, evaluating the tree bottom-up.

In each node v of the tree we will compute a vector $S(v)$ such that for each character state t the component $S_t(v)$ equals the minimal cost of a tree with the given states in the leaves and with t as character state in the root v of the subtree.

In case v is a leaf we only allow the given character state at that node, so the cost vector is defined by

$$S_t(v) = \begin{cases} 0 & \text{state}(v) = t \\ \infty & \text{otherwise} \end{cases}$$

Assuming state t at a node v, the minimal cost of the subtree rooted at v can be recursively determined by considering all possibilities of characters at the children of v.

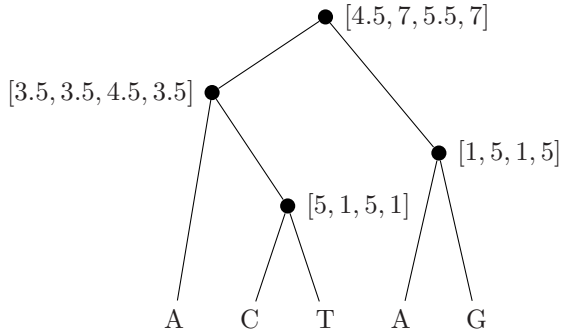
$$S_t(v) = \min_i\{C_{ti} + S_i(u)\} + \min_j\{C_{tj} + S_j(w)\}$$

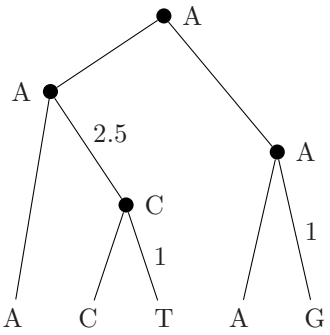
Once all values have been determined, the best choice for the character state in the root is known. The values for the other nodes are found using a backtrace process, finding for which arguments the minima were obtained.

2.3 Example. Given the following tree with character states in the leaves as given, the S-values are the vectors associated to the internal nodes for the state sequence A, C, G, T.

The transitions $A \leftrightarrow G$ and $C \leftrightarrow T$ cost 1, the other character changes (transversions) cost 2.5 (where the bases have different ring structures).

	A	C	G	T
A	0	2.5	1	2.5
C	2.5	0	2.5	1
G	1	2.5	0	2.5
T	2.5	1	2.5	0





From characters to distances. Consider the DNA as a string (over the alphabet A, C, G, T). A simple distance measure between strings is the number of substitutions to obtain one string from the other. This is like gap-less alignment, ignoring insertions and deletions. For small relative changes this works rather well. However, if we apply random changes to a string over a longer period, then it is possible that a character changes for a second time, and might even change back into its original value. Thus the number of differences will deviate from the number of substitutions that occurred. If distance between strings is to represent the time that passed in changing one string into the other, we have to use a suitable model.

Various *substitution models* have been proposed to account for changes over a longer period of time. The Jukes-Cantor model is one of the simplest. It assumes there is only one parameter μ that determines the relative rate of change from one character (nucleotide) into another. It can be shown that under this model the probability that character i changes into character j during a time period t equals $p_{ij}(t) = \frac{1}{4}(1 - e^{-4\mu t})$, $i \neq j$. The probability that it is not changed after that period equals $p_{ii}(t) = \frac{1}{4}(1 + 3e^{-4\mu t})$.

Distance between two sequences is given by $d = -\frac{3}{4}\ln(1 - \frac{4}{3}p)$ where p is the proportion of sites that differ between the two sequences. Note that for random strings p will be $\frac{1}{4}$. The resulting distance then is infinite.

More sophisticated models have more parameters. They can have different rates depending on the nucleotides, and distinguish the different relative frequencies of the nucleotides.

2.4 Distance based Methods

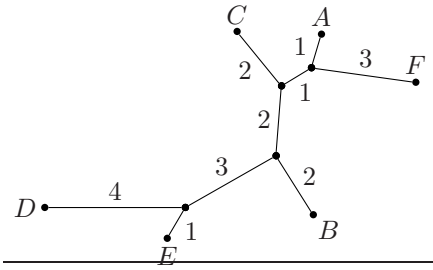
A metric space is a set of objects $\mathcal{O}$ together with a mapping $d : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$, the distance function, which should satisfy four conditions: (i) distance is never negative, (ii) the distance between objects is zero only between a object and itself, (iii) it is symmetric, and (iv) the distance between objects is never decreased by making a detour.

In formulas this reads as follows

i) $d(x, y) \geq 0$
ii) $d(x, y) = 0$ if and only if $x = y$
iii) $d(x, y) = d(y, x)$
iv) $d(x, z) \leq d(x, y) + d(y, z)$ *triangle inequality*

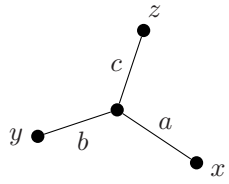
Trees can be used to specify distances. The objects are placed in the leaves of the tree, and edges are labelled with numbers. The tree-distance between two objects is then the total length of the path between the objects. A distance defined by a tree is called *additive.*

2.4 Example. A tree and the distance matrix it defines.



	A	B	C	D	E	F
A	0	6	4	11	8	4
B	6	0	6	9	6	8
C	4	6	0	11	8	6
D	11	9	11	0	5	13
E	8	6	8	5	0	10
F	4	8	6	13	10	0

Not every distance can be represented by a tree. On the other hand, every function defined in this way by a tree automatically satisfies the requirements of a metric, in particular that of the triangle inequality. As in the general picture below, look at the relative position of three nodes x, y and z in a tree. Then $d(x, z) = a + c$, whereas $d(x, y) + d(y, z) = (a + b) + (b + c) > d(x, z)$, assuming branch lengths a, b, c in the tree are positive.



In general we have the following two basic questions related to phylogeny. If we have a matrix representing a distance mapping d on a finite set of objects:

- How do we determine whether d is additive?
- Given that d is additive, how can we reconstruct its tree?

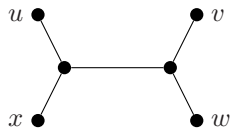
This latter problem is formulated here in the strict mathematical sense. In practice one usually does not have a precise additive function because of errors and imprecisions. Thus, the algorithm that solves the problem is expected to be robust. If the distance matrix (M_{xy}) is 'close' to additive, it should construct a tree for the matrix, such that the distance $d(x, y)$ defined by the tree is close to that given by the matrix. We will not discuss here how the notion 'close' is formalized.

2.5 Unrooted Trees and Neighbour Joining

Additive distances. Every additive metric satisfies the *four point condition*. Every set of four nodes can be ordered such that

$$d(u, v) + d(w, x) = d(u, w) + d(v, x) \geq d(u, x) + d(v, w)$$

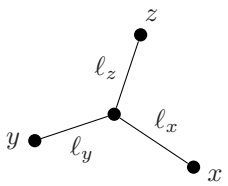
This is easily seen if we realize that the only relative position of four nodes in the tree is as follows. Now if we pair the nodes on both sides of the central branch, then the distances $d(u, x) + d(v, w)$ do not include that branch, thus the sum is less than in the other two cases. The pairing that gives the minimal distance determines how the four nodes are separated. In the figure below there is an edge that separates $\{u, x\}$ from $\{v, w\}$.



2.5 Example. Consider nodes A, B, C and D. $d(A, B)+d(C, D) = 6+11 = 17$, $d(A, C) + d(B, D) = 4 + 9 = 13$, and $d(A, D) + d(B, C) = 11 + 6 = 17$. These four points satisfy the four point condition. Nodes $\{A, C\}$ are separated from $\{B, D\}$ by an edge in the tree.

It can be shown that the four point condition characterizes additive distances. Whenever the metric satisfies the condition, a tree for the metric can be constructed.

Three leaves. In case we have only three leaves, the tree for the given distance matrix can be computed solving a simple set of equations, as follows.

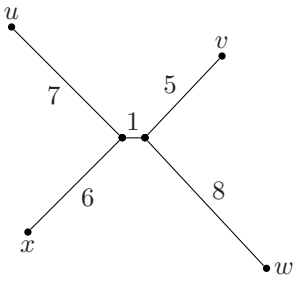


$$\left\{\begin{array}{lcl} d(x,y) &=& \ell_x+\ell_y \\ d(y,z) &=& \ell_y+\ell_z \\ d(z,x) &=& \ell_z+\ell_x \end{array}\right. \iff \left\{\begin{array}{lcl} \ell_x &=& \frac{1}{2}(d(x,y)+d(z,x)-d(y,z)) \\ \ell_y &=& \frac{1}{2}(d(y,z)+d(x,y)-d(z,x)) \\ \ell_z &=& \frac{1}{2}(d(z,x)+d(y,z)-d(x,y)) \end{array}\right.$$

Note that all these lengths will be non-negative if the d-values satisfy the triangle inequality.

Neighbours. The tree is reconstructed from the given distance matrix by determining neighbouring leaves in the tree, leaves that are connected to the same internal node. In the example A and F, as well as D and E are neighbours. Such neighbours are then joined into a single node. Thus $A\&F$ and C are neighbours, as are $D\&E$ and B.

Determining neighbours is not a simple matter of comparing distances. In the diagram below the shortest distance is between v and x. However, it is clear that v and x are not neighbours. This is obvious form the picture, but also follows from the four point condition: $d(u,x)+d(v,w)=26$ which is less than $d(u,v)+d(w,x)=28$. Indeed, there is a edge in the tree that separates $\{u,x\}$ from $\{v,w\}$.



Neighbour-Joining. The algorithm that reconstructs a tree for a given additive distance matrix d determines iteratively a pair of neighbours x and y. These neighbours are 'joined', they are connected to a new node. The distance of the new node to all nodes in the graph is computed, including x and y. The new node replaces the neighbours x and y and the computation continues with one node less. As we have seen, the pair x, y is not simply the pair of closest nodes. Instead a new 'corrected' distance is computed, which includes the total distance of a node to the rest of the nodes.

For each node x compute the 'average' distance to all other nodes

$$r(x) = \frac{1}{N-2}\sum_{z\in V} d(x,z)$$

Then, the matrix given by $D(x,y) = d(x,y) - r(x) - r(y)$ is computed. It can be shown that the pair of nodes x, y for which $D(x,y)$ is minimal is indeed a neighbouring pair (they are connected by edges to the same node). The nodes x, y are joined into a new node z.

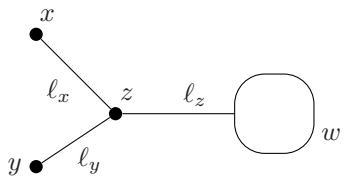
The distance of the new node z to all other nodes in the set, including the original x and y, is given by the formulas

- $d(z,x) = \frac{1}{2}(d(x,y) + r(x) - r(y))$
- $d(z,y) = \frac{1}{2}(d(x,y) + r(y) - r(x))$
- $d(z,w) = \frac{1}{2}(d(x,w) + d(y,w) - d(x,y))$, $w \neq x, y$.

The computation is repeated until three nodes are left. For this final triple the distances are computed as in the three leaves case.

Sometimes one wants to see a clear direction of evolution in the tree. Obviously this is not present in the unrooted tree. It is customary to add an outlier to the set of taxa, one that is so different from the rest that it must be the first object that splits from the other objects. This fixes the position of the root

2.6 Remark. The formulas can be understood when we try to solve a situation similar to the three node case, where $(N-1) \cdot r(x)$ is the total distance to all other nodes, and ℓ_z is the average distance to the other nodes (excluding x and y) from the newly inserted node z.



$$\begin{cases} d(x,y) &= \ell_x + \ell_y \\ r(x) &= \frac{(N-1)\cdot\ell_x + \ell_y + (N-2)\cdot\ell_z}{N-2} \\ r(y) &= \frac{(N-1)\cdot\ell_y + \ell_x + (N-2)\cdot\ell_z}{N-2} \end{cases}$$

From this $\ell_x - \ell_y = r(x) - r(y)$ follows. □

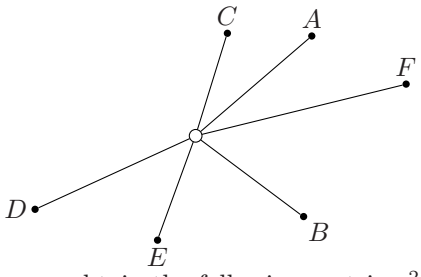
2.7 Example. We start with the distance matrix $d(\cdot,\cdot)$ from a previous example. We compute the sum of distances $r(\cdot)$ for each node, and the 'corrected' distance $D(x,y) = d(x,y) - r(x) - r(y)$. [1]

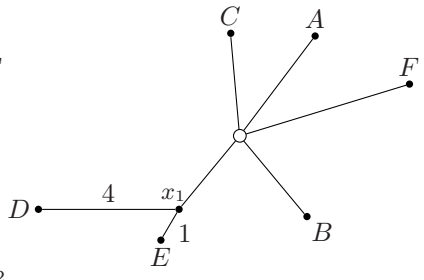
d	A	B	C	D	E	F	$4 \cdot r$
A	–						33.0
B	6.0	–					35.0
C	4.0	6.0	–				35.0
D	11.0	9.0	11.0	–			49.0
E	8.0	6.0	8.0	5.0	–		37.0
F	4.0	8.0	6.0	13.0	10.0	–	41.0

$-4 \cdot D$	A	B	C	D	E	F
A	–					
B	44.0	–				
C	52.0	46.0	–			
D	38.0	48.0	40.0	–		
E	38.0	48.0	40.0	66.0	–	
F	58.0	44.0	52.0	38.0	38.0	–

Step 1. The minimal value is $D(D,E) = \frac{66}{4}$. Hence nodes E and D are neighbours. They are joined to form x_1 (distances 1.0 and 4.0 for the edges).

[1] Sorry for having D both as taxon and as distance matrix.





We now obtain the following matrices[2].

d	A	B	C	x_1	F	$3 \cdot r$
A	–					21.0
B	6.0	–				25.0
C	4.0	6.0	–			23.0
x_1	7.0	5.0	7.0	–		28.0
F	4.0	8.0	6.0	9.0	–	27.0

$-3 \cdot D$	A	B	C	x_1	F
A	–				
B	28.0	–			
C	32.0	30.0	–		
x_1	28.0	38.0	30.0	–	
F	36.0	28.0	32.0	28.0	–

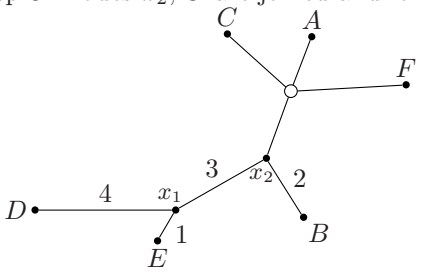
Step 2. Nodes x_1 and B are joined into the new node x_2 (with distances 3.0 and 2.0).

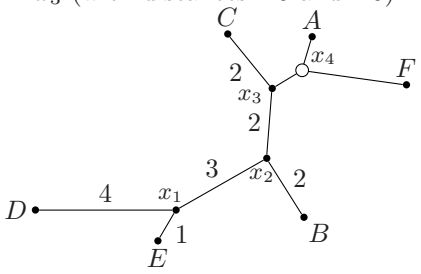
Present distance matrix

d	A	C	x_2	F	$2 \cdot r$
A	–				12.0
C	4.0	–			14.0
x_2	4.0	4.0	–		14.0
F	4.0	6.0	6.0	–	16.0

$-2 \cdot D$	A	C	x_2	F
A	–			
C	18.0	–		
x_2	18.0	20.0	–	
F	20.0	18.0	18.0	–

Step 3. Nodes x_2, C are joined and form x_3 (with distances 2.0 and 2.0).

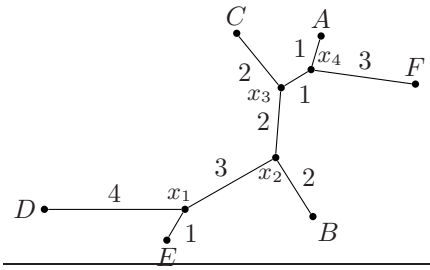




d	A	x_3	F	r
A	–			6.0
x_3	2.0	–		6.0
F	4.0	4.0	–	8.0

Now that we have three remaining nodes the joining phase ends. The three nodes we have connected to the 'central' node with edges the length of which can be computed as in the three node case.

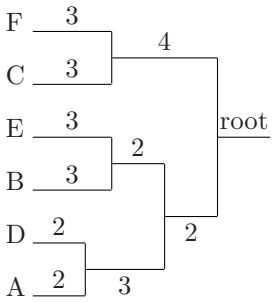
[2]The position of x_1 in the matrix is due to the program we wrote, where the new node replaces E.



	A	B	C	D	E	F
A	0	6	4	11	8	4
B	6	0	6	9	6	8
C	4	6	0	11	8	6
D	11	9	11	0	5	13
E	8	6	8	5	0	10
F	4	8	6	13	10	0

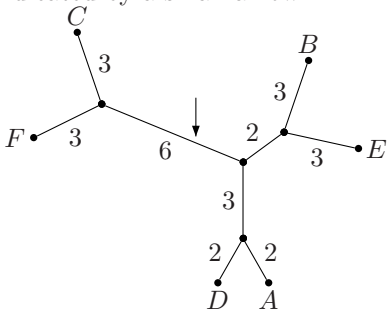
2.6 Rooted Trees and UPGMA

In this section it is assumed that the distances between the objects can be represented by a rooted tree, where all leaves have equal distance to the root. This assumption holds if we believe that species are descending from a common ancestor with a constant molecular clock, i.e., the genetic changes along all branches are in direct (linear) correspondence with the time that has passed.



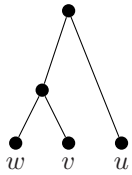
	A	B	C	D	E	F
A	0	10	14	4	10	14
B	10	0	14	10	6	14
C	14	14	0	14	14	6
D	4	10	14	0	10	14
E	10	6	14	10	0	14
F	14	14	6	14	14	0

Note that a rooted tree can be transformed into an unrooted tree by removing the root, and joining the two branches of the root into a single edge. Hence rooted trees can be seen as a special case of additive distances. The position of the old root has been indicated by a small arrow.



Utrametric distance. We consider a such a 'clock tree' where all leaves have the same distance to the root. Looking at three leaves we get the following situation, the only possible branching diagram for three leaves in a rooted tree.

It is clear that we have one short distance, and two longer ones, the longer ones being equal: $d(v,w) \leq d(u,v) = d(u,w)$. Sometimes this is called the *three point condition*.



Mathematically the three point condition is equivalent to a so-called *ultrametric distance*, where condition iv) is replaced by

iv′ $d(x,z) \leq \max\{\, d(z,y), d(y,z) \,\}$

This is a stronger requirement than the original one.

Perfect distance matrix. When the matrix perfectly meets the conditions of an ultrametric it is rather simple (at least in theory) to determine the tree representing the matrix, basically by clustering nodes having the same distance to an arbitrary fixed node. The subtree for each cluster is then determined recursively.

2.8 Example. Start with the following distance matrix. It is ultrametric.

	A	B	C	D	E	F	G
A	–						
B	15.7	–					
C	28.4	28.4	–				
D	8.0	15.7	28.4	–			
E	36.4	36.4	36.4	36.4	–		
F	15.7	1.0	28.4	15.7	36.4	–	
G	15.7	12.5	28.4	15.7	36.4	12.5	–

Take an arbitrary taxon, for instance G. Its distances to other taxa are 12.5, 15.7, 28.4, and 36.4. Hence, if we follow the path from leaf G upwards to the root, at distance 6.25 we find a branch to the subtree with nodes B and F, at distance 7.85 we find a branch to the subtree with nodes A and D, at distance 14.2 we find a branch to the subtree with node C, and finally at distance 18.2 we find the root, with a branch to the subtree with node E.

At this moment we have practically found the full tree. It suffices to see that taxa B and F are 1.0 apart, which means their common anester in the tree is 0.5 above both nodes. A similar computation is done for A and D.

UPGMA Algorithm. The UPGMA (for *unweighted pair-group method using arithmetic mean*) is a general data clustering technique. We consider each initial taxon as a cluster consisting of a single point. The algorithm repeatedly joins the pair of clusters having minimal distance into a new larger cluster. At each moment the distance between clusters is defined as the arithmetic mean of all distances between points in the two clusters:

$$d(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{\substack{x \in C_1 \\ y \in C_2}} d(x, y)$$

If two (disjoint) clusters C_1 and C_2 are amalgamated into a new cluster $C_1 \cup C_2$ its distance to each other cluster has to be computed. Fortunately we do not have to compute the mean distance from scratch. It can be simply obtained from the distances of the original clusters (and their sizes):

$$d(C_1 \cup C_2, C) = \frac{|C_1|}{|C_1| + |C_2|} d(C_1, C) + \frac{|C_2|}{|C_1| + |C_2|} d(C_2, C)$$

2.9 Example. Start with the following initial distance matrix, which does not perfectly conform to an ultrametric.

	A	B	C	D	E	F	G
A	–						
B	19.0	–					
C	27.0	31.0	–				
D	8.0	18.0	26.0	–			
E	33.0	36.0	41.0	31.0	–		
F	18.0	1.0	32.0	17.0	35.0	–	
G	13.0	13.0	29.0	14.0	28.0	12.0	–

Step 1. The minimal distance equals 1.0. The clusters to amalgamate are B and F. We obtain the following distance matrix.

	A	BF	C	D	E	G
A	–					
BF	18.5	–				
C	27.0	31.5	–			
D	8.0	17.5	26.0	–		
E	33.0	35.5	41.0	31.0	–	
G	13.0	12.5	29.0	14.0	28.0	–

Step 2. The minimal distance equals 8.0. The clusters to amalgamate are A and D.

Step 3. The minimal distance equals 12.5. The clusters to amalgamate are BF and G.

	AD	BF	C	E	G
AD	–				
BF	18.0	–			
C	26.5	31.5	–		
E	32.0	35.5	41.0	–	
G	13.5	12.5	29.0	28.0	–

	AD	BFG	C	E
AD	–			
BFG	15.8	–		
C	26.5	30.2	–	
E	32.0	31.8	41.0	–

Step 4. The minimal distance equals 15.8. The clusters to amalgamate are AD and BFG.

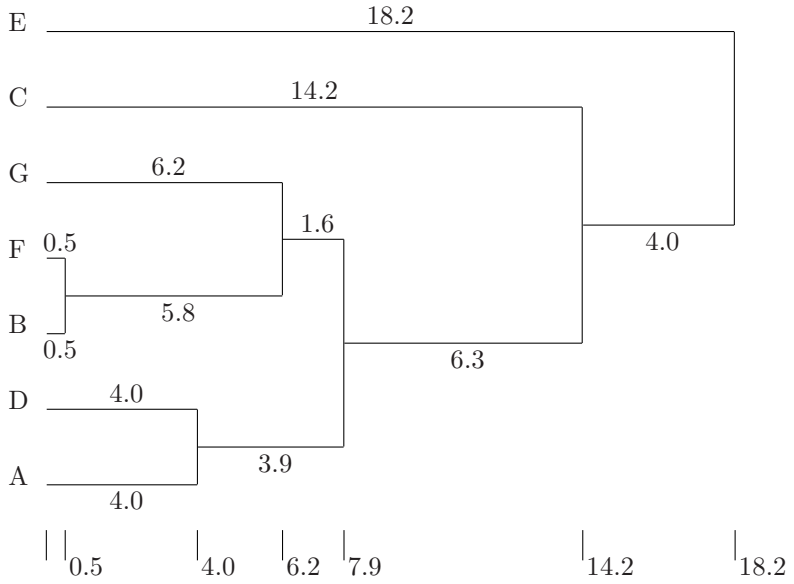
Step 5. The minimal distance equals 15.8. The clusters to amalgamate are $ABDFG$ and C.

	$ABDFG$	C	E
$ABDFG$	–		
C	28.4	–	
E	31.9	41.0	–

	A	E
A	–	
E	36.4	–

Step 6. The final clusters that are amalgamated are A and E. The height of the resulting tree is 18.2.

The final phylogeny looks as follows.



Here we repeat the original distance matrix, and the distance matrix that is

defined by the resulting tree.

	A	B	C	D	E	F	G	A	B	C	D	E	F	G
A	–							–						
B	19	–						15.8	–					
C	27	31	–					28.4	28.4	–				
D	8	18	26	–				8.0	15.8	28.4	–			
E	33	36	41	31	–			36.4	36.4	36.4	36.4	–		
F	18	1	32	17	35	–		15.8	1.0	28.4	15.8	36.4	–	
G	13	13	29	14	28	12	–	15.8	12.4	28.4	15.8	36.4	12.4	–

The example was taken from http://www.nmsr.org/upgma.htm on the web. The actual data are from the paper by Fitch and Margoliash [3]. The taxa correspond to the species (A) Turtle, (B) Man, (C) Tuna, (D) Chicken, (E) Moth, (F) Monkey, and (G) Dog.

Any distance that is 'close' to an ultrametric distance can be reconstructed by the UPGMA: the tree that is found will have the same topology as the one it is close to. In general however, even when we start with an additive distance, the tree that is obtained can have a branching structure different from the original one.

Bibliography

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman (1990). Basic local alignment search tool. Journal of Molecular Biology 215 (3): 403–410. doi:10.1006/jmbi.1990.9999
▷ alignment heuristics, BLAST, Section 1.5

[2] K.S. Booth, G.S. Lueker: Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms, Journal of Computational Systems Science, Vol. 13 (1976), pp. 335-379.
▷ physical mapping, Section 3.5

[3] Fitch and Margoliash, Construction of Phylogenetic Trees, Science Vol. 155, 20 Jan. 1967.

[4] A.P. Gultyaev, Computational Molecular Biology, Application-oriented view, Leiden University, 2009.

[5] D.S. Hirschberg. Algorithms for the Longest Common Subsequence Problem, Journal of the ACM, 24 (1977) 664–675.
▷ linear space alignment, Section 1.4

[6] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710.
▷ global alignment, edit distance, Section 1.2

[7] D.J. Lipman, W.R. Pearson, Rapid and sensitive protein similarity searches. Science. 1985 Mar 22;227(4693):1435-41.
▷ alignment heuristics, FASTA, Section 1.5

[8] S.B. Needleman, C.D. Wunsch. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4
▷ global alignment, Section 1.2

[9] N. Saitou and M. Nei, (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4(4):406-425
▷ phylogeny, unrooted trees, Section 2.5

[10] D. Sankoff (1975). Minimal mutation trees of sequences. SIAM Journal of Applied Mathematics 28: 35-42.
▷ character based, small parsimony, Sankoff algorithm, Section 2.3

[11] R. Shamir, Algorithms in Molecular Biology, lecture notes, 2001-2002, Tel Aviv University School of Computer Science. www.cs.tau.ac.il/ rshamir/algmb/01/algmb01.html

[12] J. Setubal, J. Meidanis. *Introduction to Computational Molecular Biology*, PWS Publishing Company, 1997.

[13] T.F. Smith, M.S. Waterman (1981). Identification of Common Molecular Subsequences. Journal of Molecular Biology 147: 195–197. doi:10.1016/0022-2836(81)90087-5
▷ local alignment, Section 1.3