Introduction to Algorithms in Phylogeny

Daniel Huson

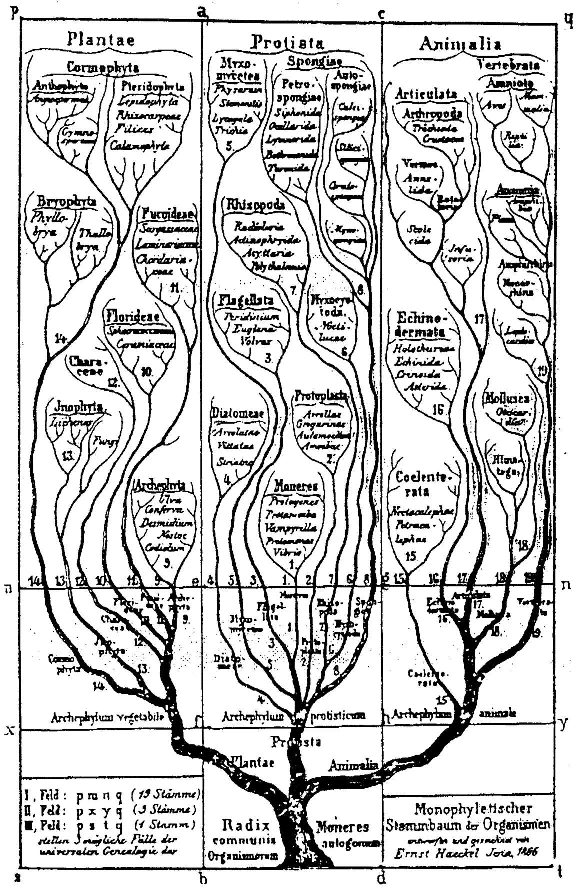
Center for Bioinformatics, Tübingen University
Sand 14, 72075 Tübingen, Germany
www-ab.informatik.uni-tuebingen.de

April 13, 2005

Contents

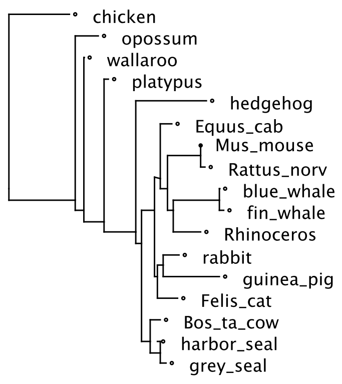
Chapter 1

Phylogenetic trees



Evolutionary tree of organisms Ernst Haeckel, 1866

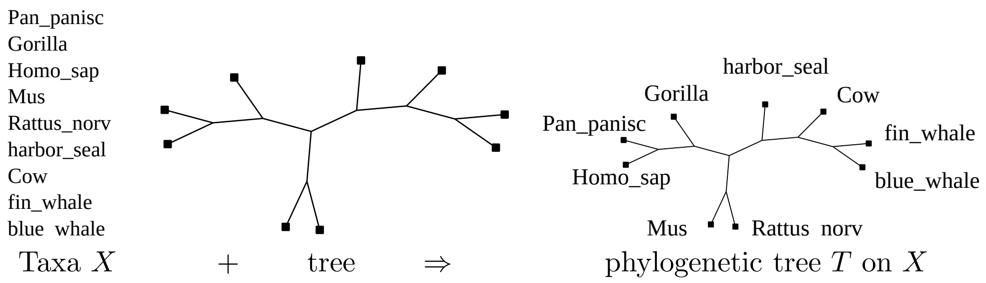
In this chapter we first introduce some basic definitions concerning trees and sequences. We then describe a very simple model of sequence evolution along a tree. We finally discuss some of the methods that are used to reconstruct a phylogenetic tree from a set of extant sequences.



1.1 Rooted and unrooted trees

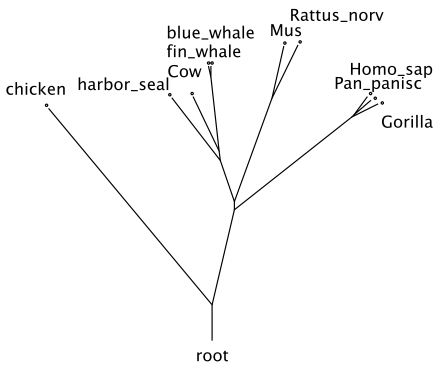
Throughout, let $X = \{x_1, \dots, x_n\}$ denote a set of *taxa*, in which each *taxon* x_i represents some species or organism whose evolutionary history is of interest to us.

For example, $X = \{x_1, x_2, \dots\}$ might denote a set of mammals, with x_1 representing a gorilla, x_2 a seal etc. A *phylogenetic tree* T *on* X (or X*-tree*) is obtained by labeling the leaves of a tree by the set X:

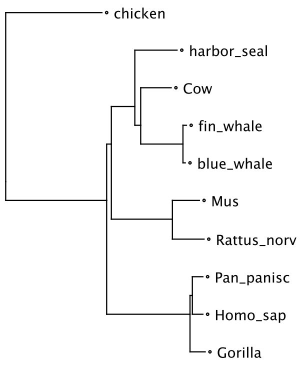


The above is an example of an *unrooted* tree. From a theoretical and algorithmic point of view, unrooted trees are easier to work with than "rooted" trees, In biology the latter are of more interest, as they define *clades* of related taxa.

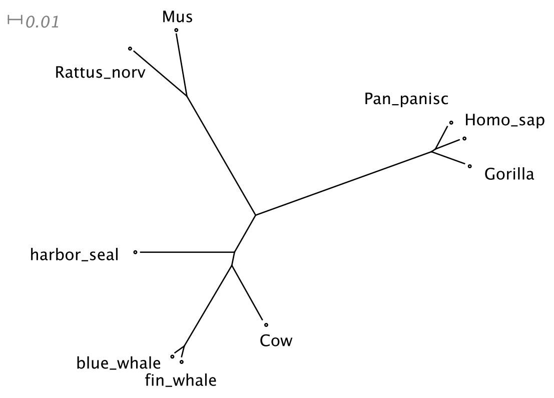
One way to determine where to *root* a tree is to include an appropriate *outgroup* in the analysis and to place the root on the branch leading to the outgroup:



or



Each branch e of a phylogenetic tree T may be scaled to represent $r \times t$, the "rate of evolution" r multiplied by the time t along e:



A phylogenetic tree T is called *bifurcating* or *resolved*, if all its internal nodes (except the root) have degree 3, and *multifurcating* or *unresolved*, else.

1.2 Aligned sequences

In *molecular phylogenetics*, a set of taxa $X = \{x_1, \ldots, x_n\}$ may be given as an alignment of molecular sequences of the form:

$$A = \left\{ \begin{array}{llll} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ & & \ldots & \\ a_{n1} & a_{n2} & \ldots & a_{nm} \end{array} \right.$$

The sequences are usually obtained from some gene or locus that all taxa have in common. One popular sequence is the SSU rRNA molecule, which has proven to carry a robust phylogenetic signal.

The problem of aligning sequences is non-trivial, but this question is beyond the scope of this tutorial.

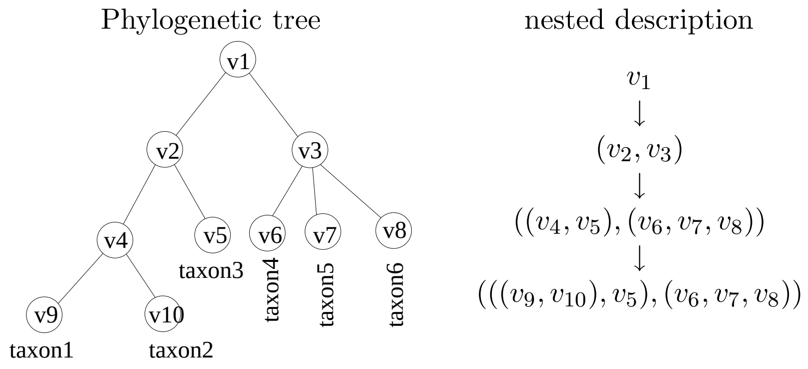
Example:

Homo sap	`fqtpmviilqaimgsatlamtliiftiiiiltvhdtnttvptmitpmllt`
Pan panisc	`fqtpmiiifqaimgsatlaltliiftiiviltvhdtntavpttitpmllt`
Gorilla	`lqtpmviifqaimgsatlamtliiftvimiltvhetnttvptmiapmllt`
harbor seal	`fqlpmviifqaiiggatlalafitftiiiifltvhdtdtstlimilsmilt`
Cow	`fqtpmviifqaiiggatlalalitftiiifmtvhdtdtstltmilsmflt`
fin whale	`lqtfmviifqaimgettlalafitftiaifltvhdtdtsmlltilsmllt`
blue whale	`lqtfmviifqaimgettlvlaiitftiaifltvhdtdtstlltilsmllt`
Rattus norv	`fqismiiifqaimggatlvlatitfiilvfltvhdtdtstfitiissmat`
Mus	`fqismiiifqaimggatlvlatitfiilifltvhdtdtstfitiissmit`

1.3 Nested structure

A rooted phylogenetic tree $T = (V, E, \lambda)$ is a *nested* structure: Consider any node $v \in V$. Let T_v denote the subtree rooted at v. Let $v_1, v_2, \ldots, v_k$ denote the children of v. Then T_v is obtainable from its subtrees $T_{v_1}, T_{v_2}, \ldots, T_{v_k}$ by connecting v to each of their roots.

Such a nested structure can be written using nested brackets:

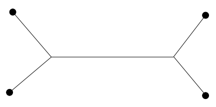


Description: $(((taxon_1, taxon_2), taxon_3), (taxon_4, taxon_5, taxon_6))$

1.4 The number of phylogenetic trees

Let T be an unrooted phylogenetic tree on n taxa, i.e., with n leaves. How many nodes and edges does T have? Let us assume that T is binary. Any non-binary tree on n taxa will have less nodes and edges.

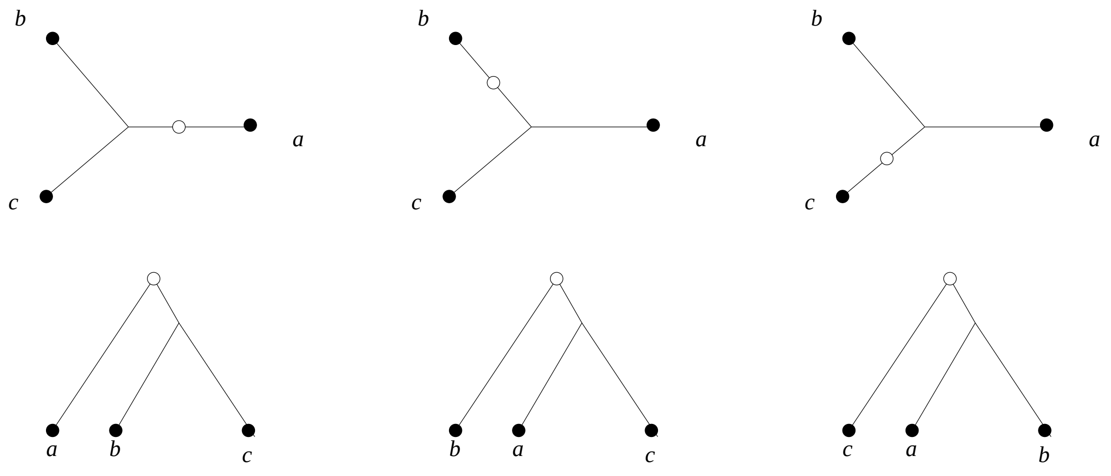
Consider a tree for $n = 4$, it has 6 nodes and 5 edges:



Now inductively, assume $n > 4$. Any tree T' with $n + 1$ leaves can be obtained from some tree T with n leaves by inserting a new node v into some edge e of T and connecting v to a new leaf w. This increases both the number of nodes and the number of edges by 2.

Putting this together, we see that the number of nodes is $2n - 2$ and the number of edges is $2n - 3$.

An unrooted tree T with n leaves has $2n - 2$ nodes and $2n - 3$ edges. A root can be added in any of the $2n - 3$ edges, thus producing $2n - 3$ different rooted trees from T:



For $n = 3$ there are three ways of adding a root. Similarly, there are 3 different ways of adding an extra edge with a new leaf to obtain an unrooted tree on 4 leaves. This new tree has $(2n - 3) = 5$

edges and there are 5 ways to obtain a new tree with 5 leaves etc.

Continuing this, we see that there are

$$U(n) = (2n - 5)!! := 3 \cdot 5 \cdot 7 \cdot \dots \cdot (2n - 5)$$

unrooted trees on n leaves. Similarly, there are

$$R(n) = (2n - 3)!! = U(n) \cdot (2n - 3) = 3 \cdot 5 \cdot \dots \cdot (2n - 3)$$

rooted trees.

These numbers grows very rapidly with n, for example, $U(10) \approx 2$ million and $U(20) \approx 2.2 \times 10^{20}$.

1.5 Models of evolution

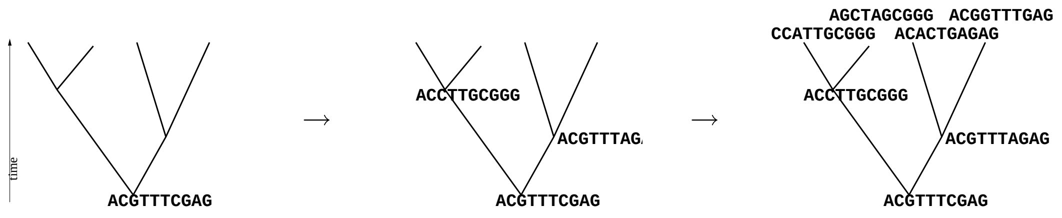
In phylogenetic analysis, a *model of evolution* is given by a rooted tree T, called the *model-*, *true-* or *generating* tree, together with a procedure for generating sequences along the model tree.

Usually, the procedure must determine how to generate an initial sequence at the root of the tree and specify how to "evolve" sequences along the edges of the tree. This involves obtaining intermediate sequences for all internal nodes of the tree, and producing a set of aligned sequences A at the leaves of the tree.

1.6 A simple model of evolution

Start with an ancestor sequence of length n at the root of a given tree. The sequence evolves up the tree, experiencing point-mutations along the way, at a fixed rate:

Start with an ancestor sequence of length n at the root of a given tree. The sequence evolves up the tree, experiencing point-mutations along the way, at a fixed rate:



This model allows for only two types of events, namely *mutations* and *speciation events* (at the nodes of the tree).

1.7 The Jukes-Cantor model of evolution

T. Jukes and C. Cantor [22] formalized such a simple model of DNA sequence evolution:

Definition Let T_0 be a rooted phylogenetic tree. The *Jukes-Cantor* model of evolution makes the following assumptions:

1. The possible states for each site are `A`, `C`, `G` and `T`.
2. The initial sequence length is an input parameter and for each site the state at the root is drawn from a given distribution (typically uniform).
3. The sites evolve identically and independently (i.i.d.) up the branches of the tree from the root at a fixed rate u.
4. With each branch $e \in E$ we associate a duration $t = t(e)$ and the expected number of mutations per site along e is $u\tau(e)$. The probabilities of change to each of the 3 remaining states are equal.

How do we "evolve" a sequence up a branch e under the Jukes-Cantor model?

Let $a = a_1a_2 \dots a_n$ and $b = b_1b_2 \dots b_n$ denote the source and target sequences associated with e. We assume that a has already been determined and we want to determine b.

Under the Jukes-Cantor model, the evolutionary event

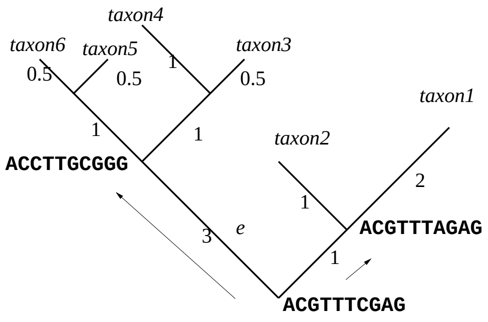
nucleotide changes to one of the other three bases

occurs at a fixed rate u.

From this, we obtain a *probability-of-change formula* for the probability of an *observable* change occurring at any given site in time t:

$$\text{Prob}(\text{change} \mid t) \;=\; \tfrac{3}{4}\left(1 - e^{-\frac{4}{3}ut}\right).$$

This model can be used to "evolve" sequences along a model tree T_0. Consider the following example with $u = 0.1$:



The root node is assigned a random sequence. Then the sequences are evolved up the branches, using the probability-of-change formula to decide whether to "mutate" a given base.

E.g., the probability of change along the branch labeled e is

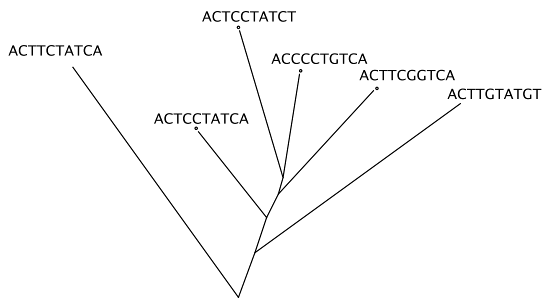
$$0.75(1 - e^{-\frac{4}{3}\times 0.1\times 3}) = 0.75(1 - e^{-0.4}) = 0.247.$$

1.8 The tree reconstruction problem

Given a set of sequences that were generated along some model tree T_0 according to some model, can the model tree be reconstructed?

```
ACTTCTATCA
ACTCCTATCA
ACTCCTATCT
ACCCCTGTCA
ACTTCGGTCA
ACTTGTATGT
```

$\xrightarrow{?}$



More precisely, the challenges are:

- determine the unrooted topology of T_0,
- estimate the branch lengths of T_0, and
- correctly determine the position of the root in T_0.

1.9 Tree reconstruction methods

There exist many different approaches to this problem [11]:

- *Distance-based* methods infer a distance matrix from the input data then *construct* a tree from the matrix, such as:
  - UPGMA [26]
  - Neighbor-Joining [25] and its variants Bio-NJ [13] and Weighbor [2].
- *Sequence-based* methods *search* for a tree that optimally explains the given sequence data, such as:
  - Maximum Parsimony [8],
  - Maximum Likelihood [9], and
  - Bayesian inference [18].

1.10 Distances

Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of taxa. The input to a distance method is a dissimilarity matrix $D : X \times X \to \mathbb{R}_{\geq 0}$ that associates a *distance* $d(x_i, x_j)$ with every pair of taxa $x_i, x_j \in X$. Sometimes we will abbreviate $d_{ij} := d(x_i, x_j)$.

We usually require that

1. the matrix is *symmetric*, that is, $d(x, y) = d(y, x)$ for all $x, y \in X$, and
2. $d(x, x) = 0$ for all $x \in X$.

We call D a *pseudo metric*, if the *triangle inequalities* are satisfied:

$$d(x, z) \leq (x, y) + d(y, z) \text{ for all } x, y, z \in X,$$

and a *metric*, if additionally we have $d(x, y) > 0$ for all $x \neq y$.

1.11 Hamming distance

Let a collection of taxa be given by a set of distinct sequences $A = \{a_1, a_2, \dots, a_n\}$ and assume we are given a multiple sequence alignment A^* of the sequences.

We define *sequence dissimilarity* as the (normalized) *Hamming distance* $Ham(a_i, a_j)$ between two taxa a_i and a_j as the number of mismatch positions in a_i^* and a_j^*, divided by the number of comparisons.

We ignore any column in which both sequences contain a gap. If only one sequence has a gap in a column then we can either ignore the column, or treat it as a match, or as a mismatch, depending on the type of data. Usually, one ignores *all* columns in which any of the n sequences contains the gap.

For protein data, it makes sense to relax the definition of *sequence dissimilarity* to the number of "non-synonymous" residues divided by the number of sequence positions compared.

For example, we may choose to ignore "conservative substitutions" by pooling amino acids with similar properties into six groups: acidic (D,E), aromatic (F,W,Y), basic (H,K,R), cysteine, non-polar (A,G,I,L,P,V), and polar (M,N,Q,S,T). Two residues are considered synonymous, if the are contained in the same group, and non-synonymous, otherwise.

There exist many different methods for computing distances from DNA or protein sequences.

Example:

```
a1  C A A C C C C C A A A A A
a2  T A A T T T - C A A A A A
a3  C G G T T T - - A A A A A
```

Distances:

$$Ham(a_1, a_2) = \frac{4}{12} = 0.\overline{33}$$
$$Ham(a_1, a_3) = \frac{5}{11} = 0.\overline{45}$$
$$Ham(a_2, a_3) = \frac{3}{11} = 0.\overline{27}$$

Hamming distances are only suitable for closely related sequences.

1.12 The Jukes-Cantor distance transformation

The Hamming distance is an *underestimation* of the number of mutations that happened along a path between two leaves a_i and a_j. This is addressed using a transformation based on a model of evolution such as the Jukes-Cantor model:

Lemma The maximum likelihood distance between a pair of sequences a_i, a_j (that is, the most likely ut to have generated the observed sequences) is given by the following formula:

$$JC(a_i, a_j) = -\frac{3}{4} \ln \left(1 - \frac{4}{3} Ham(a_i, a_j)\right).$$

This is called the *Jukes-Cantor distance transformation.*

1.13 UPGMA

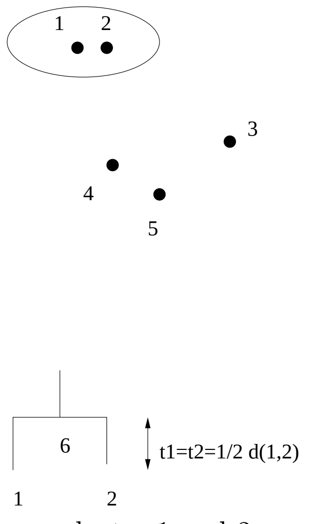
We will now discuss a simple distance method called UPGMA which stands for *unweighted pair group method using arithmetic averages* [26].

Given a set of taxa X and a distance matrix D, UPGMA produces a rooted phylogenetic tree T with edge lengths.

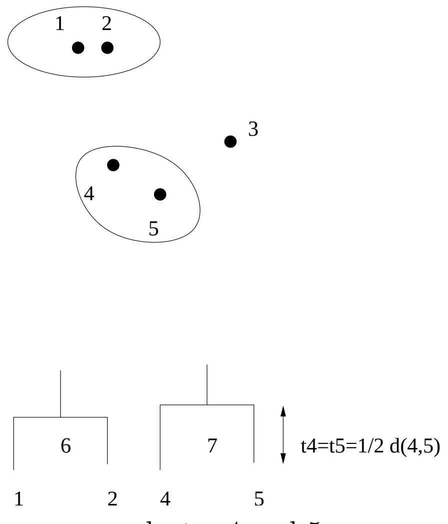
It operates by clustering the given taxa, at each stage merging two clusters and at the same time creating a new node in the tree. The tree is assembled "upwards", first clustering pairs of leaves, then pairs of clustered leaves etc. Each node is given a height and the edge lengths are obtained as the difference of heights of its two end nodes.

1.14 UPGMA example

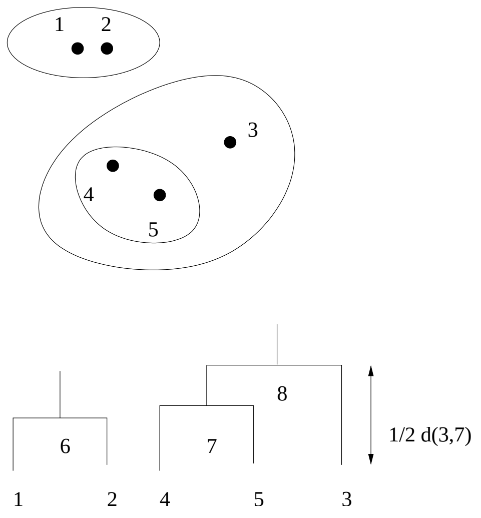
Example $X = \{1, 2, 3, 4, 5\}$, distances given by distance in the plane:



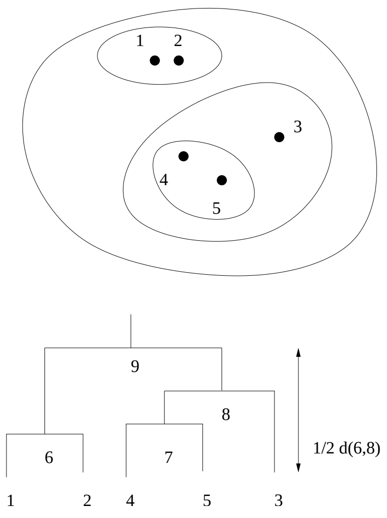
cluster 1 and 2



cluster 4 and 5



cluster 7 and 3



cluster 6 and 8

UPGMA produces a rooted, binary phylogenetic tree.

1.15 The distance between two clusters

Initially, we are given a distance $d(x, y)$ between any two taxa, i.e. leaves, x and y.

We define the distance $d(i, j) := d(C_i, C_j)$ between two clusters $C_i \subseteq X$ and $C_j \subseteq X$ to be the average distance between pairs of taxa from each cluster:

$$d(i, j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x, y).$$

Note that, if C_k is the union of two clusters C_i and C_j, and C_l is any other cluster, then

$$d(k, l) = \frac{d(i, l)|C_i| + d(j, l)|C_j|}{|C_i| + |C_j|}.$$

This is a useful *update* formula, because using it in the algorithm, we can obtain the distance between two clusters in constant time.

1.16 The UPGMA algorithm

The UPGMA algorithm is very straight-forward:

```
**Algorithm** UPGMA
Input: A set of taxa $X$ and a corresponding distance matrix $D$
Output: A binary, rooted phylogenetic UPGMA tree on $T$
**Initialization**
        Assign each taxon $x_i$ to its own cluster $C_i$
        Define one leaf of $T$ for each taxon, placed at height zero
**Iteration**
```

```
    Determine two clusters $C_i$ and $C_j$ for which $d(i,j)$ is minimal
    Define a new cluster $k$ by $C_k = C_i \cup C_j$
    Define $d(k,l)$ for all existing clusters $l$ using the update formula
    Define a node $k$ with daughter nodes $i$ and $j$, and place it at height $\frac{d(i,j)}{2}$
    Add $C_k$ to the set of current clusters and remove $C_i$ and $C_j$.
Termination
    When only two clusters $C_i$ and $C_j$ remain, place the root at height $\frac{d(i,j)}{2}$.
```

(Problem: show that this algorithm produces well-defined edge lengths, i.e. a parent node always lies above its daughters.)

Finally, for each edge e, set the edge length $\omega(e)$ equal to the difference of the heights of the two incident nodes.

Example of UPGMA applied to 5S rRNA data:

Original distances:

Abbreviations:
Bsu: *Bacillus subtilis*
Bst: *Baclillus stearothermophilus*
Lvi: *Lactobacillus viridescens*
Amo: *Acholeplasma modicum*
Mlu: *Micrococcus luteus*

	Bsu	Bst	Lvi	Amo	Mlu
Bsu	–	$\underline{0.1715}$	0.2147	0.3091	0.2326
Bst		–	0.2991	0.3399	0.2058
Lvi			–	0.2795	0.3943
Amo				–	0.4289
Mlu					

$\longrightarrow$

	Bsu + Bst	Lvi	Amo	Mlu
Bsu + Bst	–	0.2569	0.3245	$\underline{0.2192}$
Lvi		–	0.2795	0.3943
Amo			–	0.4289
Mlu				–

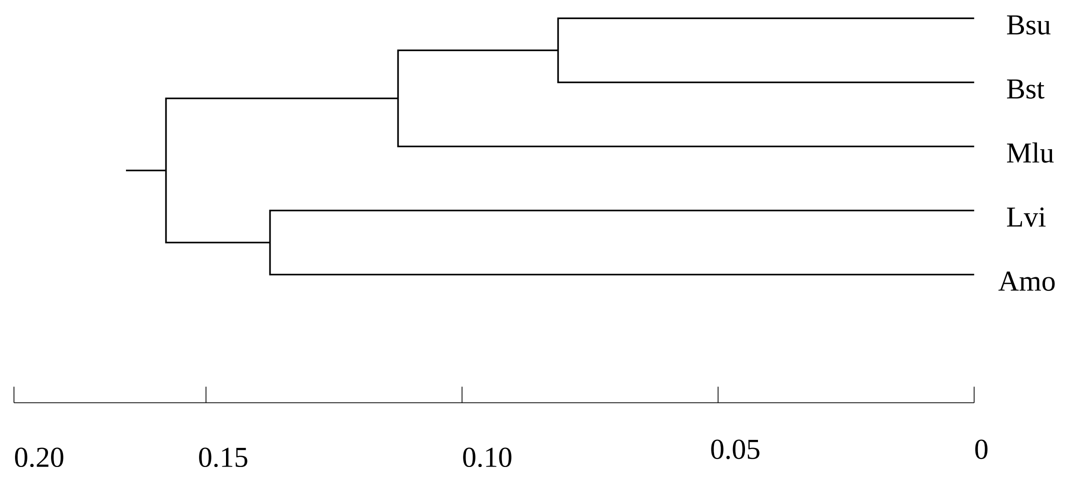
$\longrightarrow$

	Bsu + Bst + Mlu	Lvi	Amo
Bsu + Bst + Mlu	–	0.3027	0.3593
Lvi		–	$\underline{0.2795}$
Amo			–

$\longrightarrow$

	Bsu + Bst + Mlu	Lvi + Amo
Bsu + Bst + Mlu	–	$\underline{0.3310}$
Lvi + Amo		–

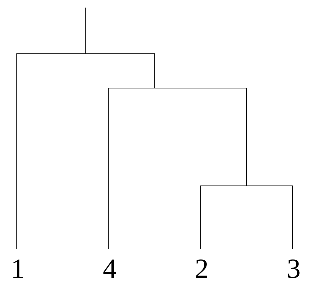
The resulting tree:



This tree is biologically incorrect, as we will see later.

1.17 The molecular clock hypothesis

Given a distance matrix D, the UPGMA method aims at building a rooted tree T with the property that all leaves have the same distance from the root ρ:

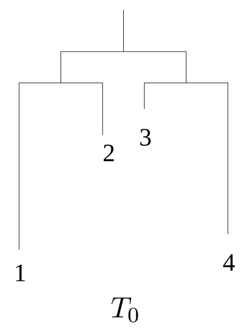


This approach is suitable for sequence data that has evolved under circumstances in which the rate of mutations of sequences is constant over time and for all lineages in the tree.

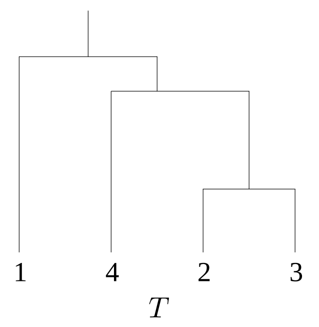
Definition The assumption that evolutionary events happen at a constant rate is called the *molecular clock* hypothesis.

1.18 UPGMA and the molecular clock

If the input distance matrix D was directly obtained from a generating phylogenetic tree T_0 that adheres to the molecular clock assumption, then the tree T reconstructed by UPGMA from D will equal T_0. Otherwise, if T_0 does not do so, then UPGMA may fail to reconstruct the tree correctly, for example:



$\overrightarrow{\text{UPGMA}}$



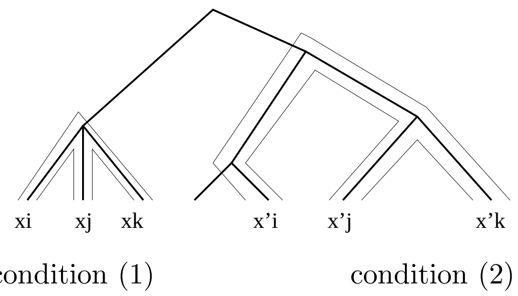
The problem here is that the closest leaves in T_0 are not neighboring leaves: they do not have a common parent node.

1.19 The ultrametric property

A distance matrix D is called an *ultrametric*, if for *every triplet* of taxa $x_i, x_j, x_k \in X$, the three distances $d(x_i, x_j)$, $d(x_i, x_k)$ and $d(x_j, x_k)$ have the property that either:

1. all three distances are equal, or
2. two are equal and the remaining one is smaller.

Note that if D was directly obtained from some tree T that satisfies the molecular clock hypothesis, then D is an ultrametric:



We say that a rooted phylogenetic tree T is *ultrametric*, if every leaf has the same distance from the root.

One can show the following results:

Theorem A distance matrix D is directly obtainable from some ultrametric tree T, if and only if D is an ultrametric.

Theorem If D is a distance matrix directly obtainable from some ultrametric tree T, then UPGMA applied to D will produce that tree T.

1.20 Additivity and the four-point condition

Given a set of taxa X. A distance matrix D on X is called *additive*, if D is directly obtainable from some phylogenetic tree T.

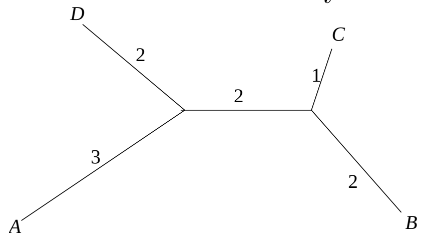
Given an arbitrary distance matrix D. Can we determine whether D is additive without attempting to obtain a suitable tree? The answer is yes, using the following result due to Peter Buneman [4]:

Theorem A distance matrix D on X is additive, iff for *every quartet* of (not necessarily distinct) taxa $x_i, x_j, x_k, x_l \in X$ the so-called *four-point condition* holds:

$$d(x_i, x_j) + d(x_k, x_l) \leq \max\left(d(x_i, x_k) + d(x_j, x_l), d(x_i, x_l) + d(x_j, x_k)\right).$$

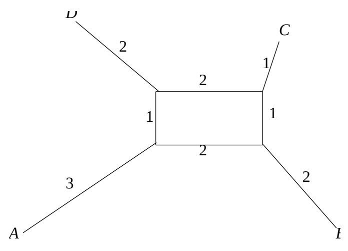
(That is, two of the three expressions $d(x_i, x_j) + d(x_k, x_l)$, $d(x_i, x_k) + d(x_j, x_l)$, and $d(x_i, x_l) + d(x_j, x_k)$ are equal and are larger than the third.)

Distances obtained directly from a phylogenetic tree:



Check the four-point condition with:
$d(A, B) + d(C, D) = 7 + 5 = 12$
$d(A, C) + d(B, D) = 6 + 6 = 12$
$d(A, D) + d(B, C) = 5 + 3 = 8$
$\Rightarrow$ the four-point condition holds.

A phylogenetic network:



Check the four-point condition with:
$d(A,B) + d(C,D) = 7 + 5 = 12$
$d(A,C) + d(B,D) = 7 + 7 = 14$
$d(A,D) + d(B,C) = 6 + 4 = 10$
$\Rightarrow d(A,C) + d(B,D) \nleq$
$\max(d(A,B) + d(C,D), d(A,D) + d(B,C))$
$\Rightarrow$ 4-point condition doesn't hold.

1.21 Neighbor-Joining

The most widely used distance method is *Neighbor-Joining (NJ)*, originally introduced by Saitou and Nei [25], and modified by Studier and Keppler [27].

Given a distance matrix D, Neighbor-Joining produces an unrooted phylogenetic tree T with edge lengths.

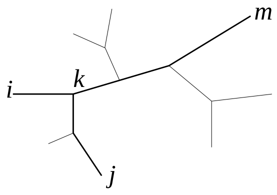
It is more widely applicable than UPGMA, as it does not assume a molecular clock.

Let D be a distance matrix directly obtainable from some (unknown) tree T. Assume that we are building a tree based on D by repeatedly pairing "neighboring" taxa. The following step reduces the number of leaves by one and we can repeatedly apply it until we arrive at a single pair of leaves:

Let i and j be two *neighboring leaves* that have the same parent node, k. Remove i, j from the list of nodes and add k to the current list of nodes, defining its distance to any given leaf m by

$$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij}).$$

By additivity of D, the distances d_{km} defined in this way are precisely those between equivalent nodes in the original tree:



In other words, for any three leaves i, j, m there is a node k where the paths to them meet. By additivity,

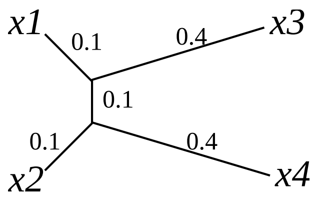
$$d_{im} = d_{ik} + d_{km},\ d_{jm} = d_{jk} + d_{km} \text{ and } d_{ij} = d_{ik} + d_{jk},$$

which implies $d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$.

How to determine which nodes are neighbors?

The Neighbor-Joining method is based on the fact that we can decide which nodes are neighbors, using only the distance matrix.

However, it does *not* suffice simply to pick the two closest leaves, i.e. a pair i, j with d_{ij} minimal, for example:



Given distances generated by this tree. Leaves x_1 and x_2 have minimal distance, but are not neighbors.

To avoid this problem, the trick is to subtract the "averaged distances" to all other leaves, thus compensating for long edges. We define:

$$N_{ij} := d_{ij} - (r_i + r_j),$$

where

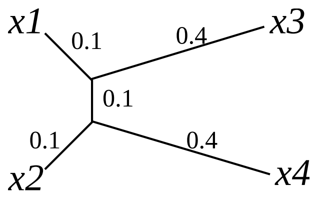
$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik},$$

and L denotes the set of leaves. (Note that this is not precisely the average, as the number of summands is $|L|$, not $|L - 2|$. However, this expression is correct and is necessary for the proof of the following result.)

Lemma If D is directly obtainable from some tree T, then the 2 leaves x_i and x_j for which N_{ij} is minimal are neighbors in T.

This result ensures that the Neighbor-Joining algorithm will correctly reconstruct a tree from its additive distances.

Let us illustrate this result using the previous example:



Here, $r_1 = 0.7$, $r_2 = 0.7$, $r_3 = 1.0$ and $r_4 = 1.0$. And so,

$$N = \left\{ \begin{array}{ccccc} & x_1 & x_2 & x_3 & x_4 \\ x_1 & - & -1.1 & -\mathbf{1.2} & -1.1 \\ x_2 & & - & -1.1 & -\mathbf{1.2} \\ x_3 & & & - & -1.1 \\ x_4 & & & & - \end{array} \right.$$

The matrix N attains a minimum value for the pair $i = 1$ and $j = 3$ and for the pair $i = 2$ and $j = 4$, as required.

1.22 The Neighbor-Joining algorithm

Algorithm (Neighbor-Joining)
Input: Distance matrix D
Output: Phylogenetic tree T

```
Initialization:
        Define $T$ to be the set of leaf nodes, one for each taxon.
        Set $L = T$.
Iteration:
        Pick a pair $i, j \in L$ for which $N_{ij}$ is minimal.
        Define a new node $k$ and
            set $d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$, for all $m \in L$.
        Add $k$ to $T$ with edges of lengths $d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$ and
            $d_{jk} = d_{ij} - d_{ik}$, joining $k$ to $i$ and $j$, respectively.
        Remove $i$ and $j$ from $L$ and add $k$.
Termination:
        When $L$ consists of two leaves $i$ and $j$, add the remaining
            edge between $i$ and $j$, with length $d_{ij}$.
```

1.23 Application of Neighbor-Joining

Given an additive distance matrix D directly obtained from a phylogenetic tree T, Neighbor-Joining is guaranteed to reconstruct T correctly.

However, in practice we are never given a matrix that was "directly obtained" from the generating tree, but rather the distance matrix is usually obtained very indirectly by a comparison of finite sequence data generated along the tree. Such data is rarely additive. Nevertheless, the Neighboring-Joining method is often applied to such data and has proved to be a fast, useful and robust tree reconstruction method.

1.24 Example

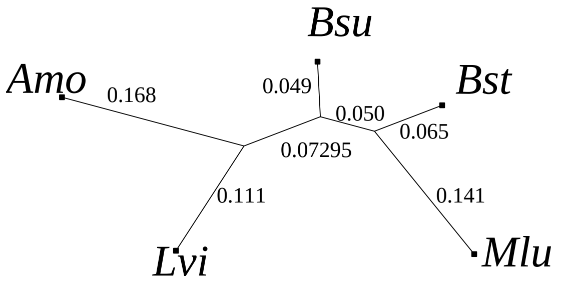
Example of Neighbor-Joining applied to 5S rRNA data:

Original distances:

Abbreviations:
Bsu: *Bacillus subtilis*
Bst: *Baclillus stearothermophilus*
Lvi: *Lactobacillus viridescens*
Amo: *Acholeplasma modicum*
Mlu: *Micrococcus luteus*

	Bsu	Bst	Lvi	Amo	Mlu
Bsu	–	0.1715	0.2147	0.3091	0.2326
Bst		–	0.2991	0.3399	0.2058
Lvi			–	0.2795	0.3943
Amo				–	0.4289
Mlu					

The resulting tree:



1.25 Maximum Parsimony

The Maximum Parsimony method is by far the most used sequence-based tree reconstruction method.

In science, the principal of *Maximum Parsimony* is well known: always use the simplest, *most parsimonious* explanation of an observation, until new observations force one to adopt a more complex theory.

In phylogenetic analysis, the *Maximum Parsimony problem* is to find a phylogenetic tree that explains a given set of aligned sequences using a minimum number of "evolutionary events".

1.26 The Parsimony score of a tree

The *difference* between two sequences $x = (x_1, \ldots, x_L)$ and $y = (y_1, \ldots, y_L)$ is simply their non-normalized Hamming distance

$$\mathrm{diff}(x, y) = |\{k \mid x_k \neq y_k\}|.$$

Given a multiple alignment of sequences $A = \{a_1, a_2, \ldots, a_n\}$ and a corresponding phylogenetic tree T, leaf-labeled by A.

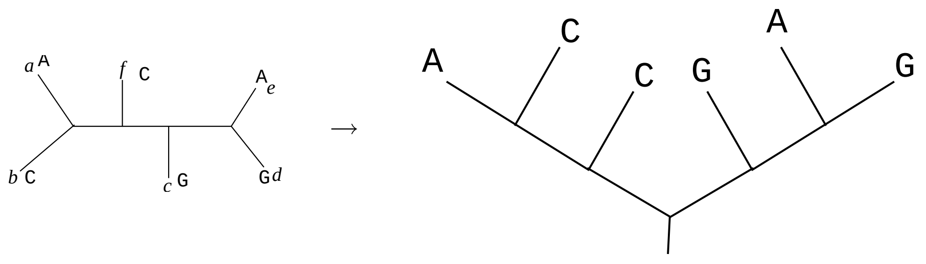
If we assign a hypothetical ancestor sequence to every internal node in T, then we can obtain a score for T together with this assignment, by summing over all differences $\mathrm{diff}(x, y)$, where x and y are any two sequences labeling two nodes that are joined by an edge in T.

The minimum value obtainable in this way is called the *Parsimony score* $PS(T, A)$ of T and A.

1.27 The Small Parsimony problem

The *small* parsimony problem is to compute the Parsimony score for a given tree T. Can it be solved efficiently?

As the Parsimony score is obtained by summing over all columns, the columns are independent and so it suffices to discuss how to obtain an optimal assignment for one position:



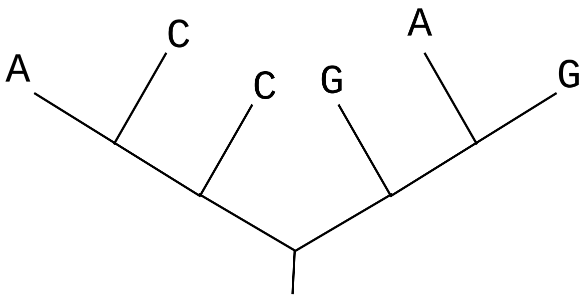
unrooted tree rooted tree

1.28 The Fitch algorithm

The following algorithm computes the Parsimony score for T and a fixed column in the sequence alignment. It modifies a global score variable and is repeatedly run to obtain the total score. Initially it is called with $e = null$ and v the root node.

```
**Algorithm** ParsimonyScore($e, v$) [12]
Input: A phylogenetic tree $T$ and a character $c(v)$ for each leaf $v$
Output: The Parsimony score for $T$ and $c$
if $v$ is a leaf node then
  Set $R(v) = \{c(v)\}$
else
  for each edge $f_1, f_2 \neq e$ adjacent to $v$ do
      Let $w_i$ be the opposite node of $f_i$
      Call ParsimonyScore($f_i, w_i$) // to compute $R(w_i)$
  if $R(w_1) \cap R(w_2) \neq \emptyset$ then
    Set $R(v) = R(w_1) \cap R(w_2)$
  else
    Set $R(v) = R(w_1) \cup R(w_2)$ and increment the global score
end
```

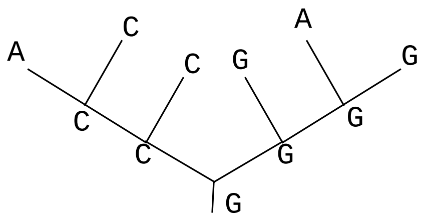
We can use the Fitch algorithm to show that the Parsimony score for the following labeled tree is 3:



In total, the algorithm requires $O(nL)$ steps.

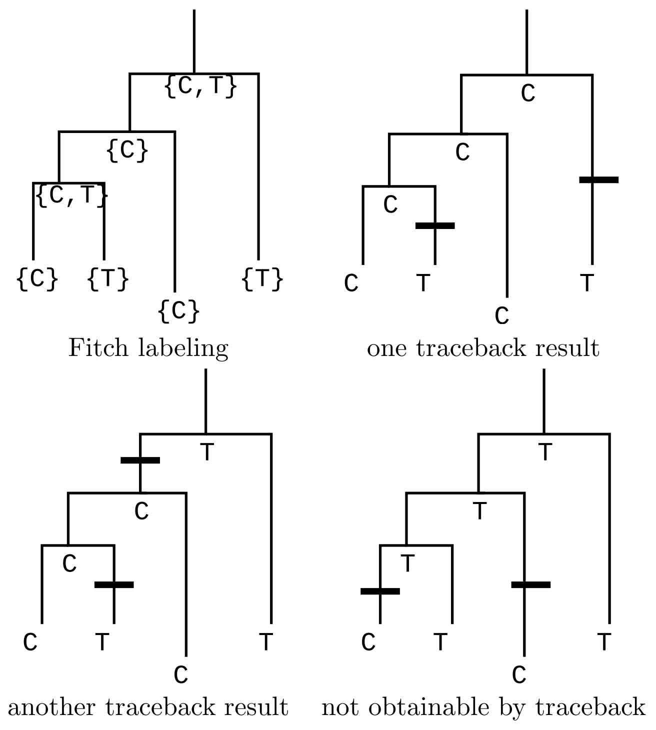
1.29 Traceback

The above algorithm computes the Parsimony score. An optimal labeling of the internal nodes is obtained via traceback: starting at the root node r, we label r using any character in $R(r)$. Then, for each child w, we us the same letter, if it is contained in $R(w)$, otherwise we us any letter in $R(w)$ as label for w. We then visit the children von w etc:



Again, the algorithm requires $O(nL)$ steps, in total.

Example:

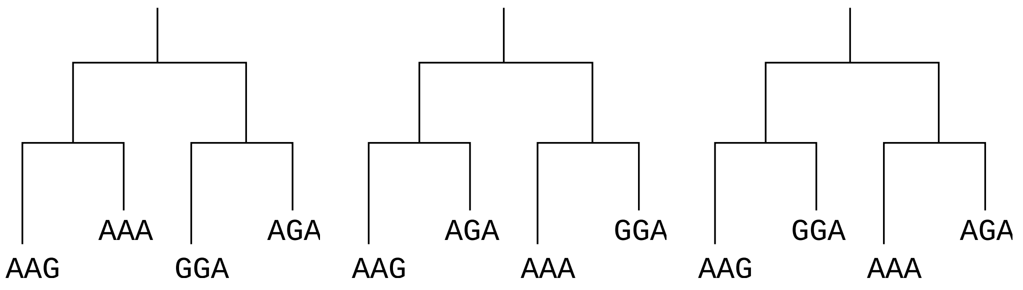


1.30 A simple example

Assume we are given the following four aligned sequences:

```
A A G
A A A
G G A
A G A
```

There are three possible binary topologies on four taxa:



In each tree, label all internal nodes with sequences so as to minimize the score obtained by summing all mismatches along edges. Which tree minimizes this score?

1.31 The Large Parsimony problem

Given a multiple alignment $A = \{a_1, \ldots, a_n\}$, it's *Parsimony score* is defined as

$$PS(A) = \min\{PS(T, A) \mid T \text{ is a phylogenetic tree on } A\}.$$

The *Large Parsimony problem* is to compute $PS(A)$.

Potentially, we need to consider all $(n-5)!!$ possible trees. Unfortunately, in general this can't be avoided and the Maximum Parsimony problem is known to be NP-hard.

Exhaustive enumeration of all possible tree topologies will only work for $n \leq 10$ or 11, say.

Thus, we need more efficient strategies that either solve the problem exactly, such as the branch and bound technique, or return good approximations, such as heuristic searches.

Remark: As with most biological problems, we are not only interested in the optimal solution, but would also like to know something about other near optimal solutions as well.

1.32 Branch and bound

Recall how we obtained an expression for the number $U(n)$ of unrooted phylogenetic tree topologies on n taxa:

For $n = 3$ there are three ways of adding an extra edge with a new leaf to obtain an unrooted tree on 4 leaves. This new tree has $(2n-3) = 5$ edges and there are 5 ways to obtain a new tree with 5 leaves etc.

To be precise, one can obtain any tree T_{i+1} on $\{a_1, \dots, a_i, a_{i+1}\}$ by adding an extra edge with a leaf labeled a_{i+1} to some (unique) tree T_i on $\{a_1, \dots, a_i\}$.

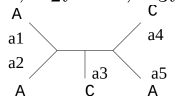
In other words, we can produce the set of *all* possible trees on n taxa by adding one leaf at a time in all possible ways, thus systematically generating a complete *enumeration tree.*

A simple, but crucial observation is that adding a new sequence a_{i+1} to a tree T_i to obtain a new tree T_{i+1} cannot lead to a smaller Parsimony score.

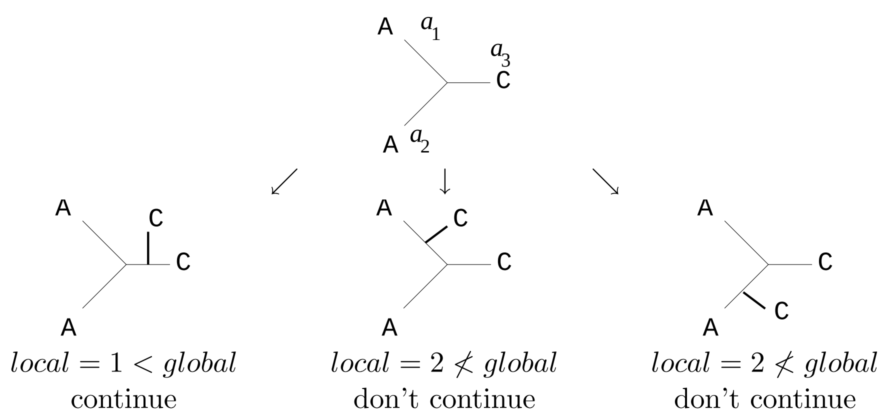
This gives rise to the following bound criterion when generating the enumeration tree: if the *local* Parsimony score of the current incomplete tree T' is larger or equal to the best *global* score for any complete tree seen so far, then we do not generate or search the enumeration subtree below T'.

In practice, using branch and bound one can obtain exact solutions for data sets of twenty or more sequences, depending on the sequence length and the messiness of the data.

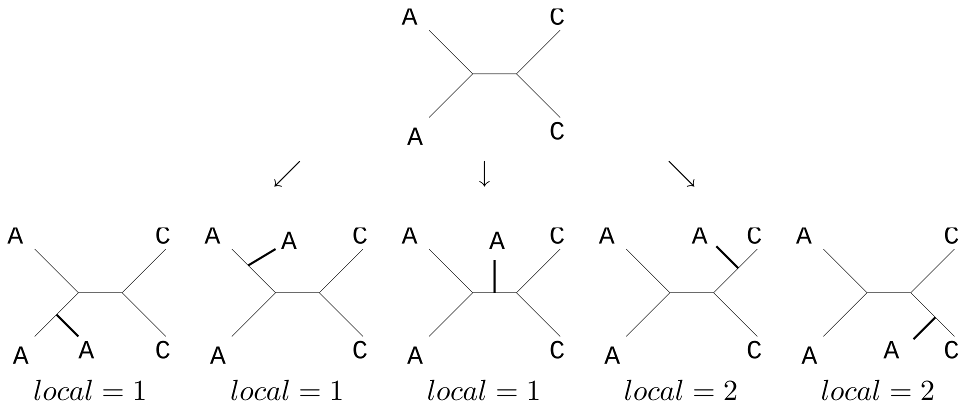
A good starting strategy is to first compute a tree T_0 for the data, e.g. using Neighbor-Joining, and then to initialize the *global* bound to the Parsimony score of T_0.

Example Assume we are given an msa $A = \{a_1, a_2, \dots, a_5\}$ and at a given position i the characters are: $a_{1i} = \texttt{A}$, $a_{2i} = \texttt{A}$, $a_{3i} = \texttt{C}$, $a_{4i} = \texttt{C}$ and $a_{5i} = \texttt{A}$. Assume that the Neighbor-Joining tree on A looks like this: 

and thus gives a global upper bound of *global* $= 2$ for position i. The first step in generating the enumeration tree is this:



Only the first of the three trees fulfills the bound criterion and we do not pursue the other two trees. The second step in generating the enumeration tree looks like this:



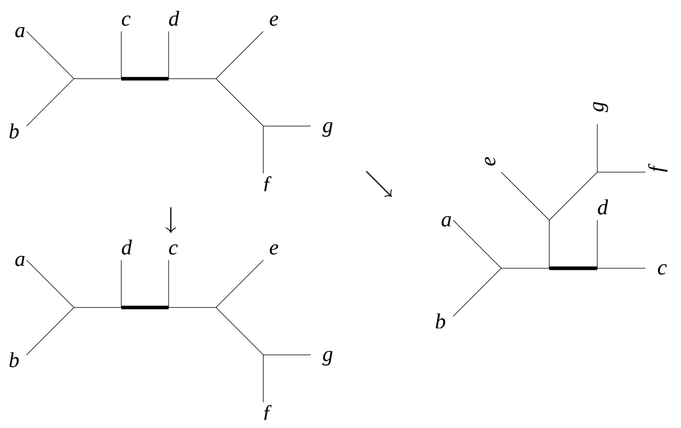
The first three trees are optimal. Note that the bound criterion in the first step reduced the number of full trees to be considered by two thirds.

Application of branch-and-bound to evolutionary trees was first suggested by Mike Hendy and Dave Penny (1982).

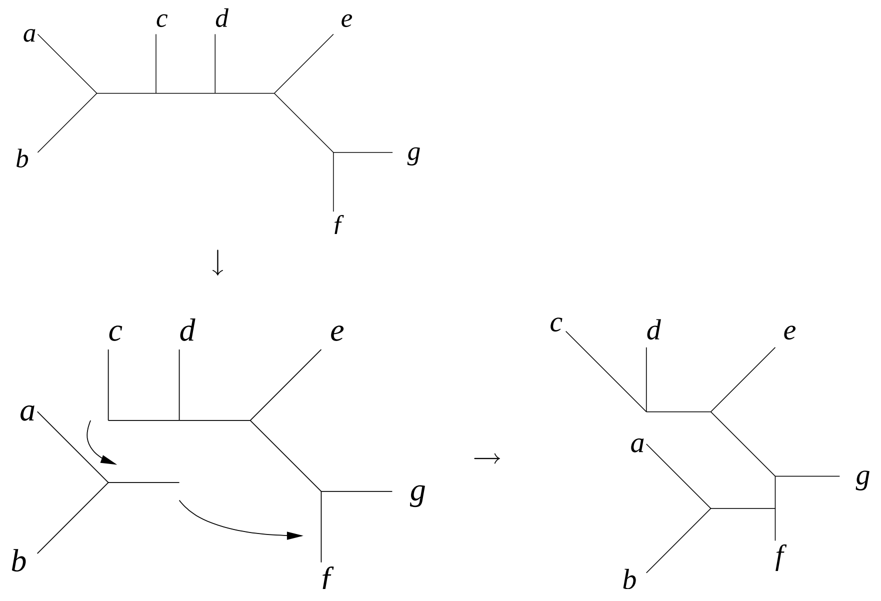
1.33 Branch swapping methods

The two heuristics just described are both very susceptible to entrapment in local optima. We now discuss a number of *branch-swapping operations* that one can use to move through the space of all trees, hopefully jumping far enough to escape from local optima.

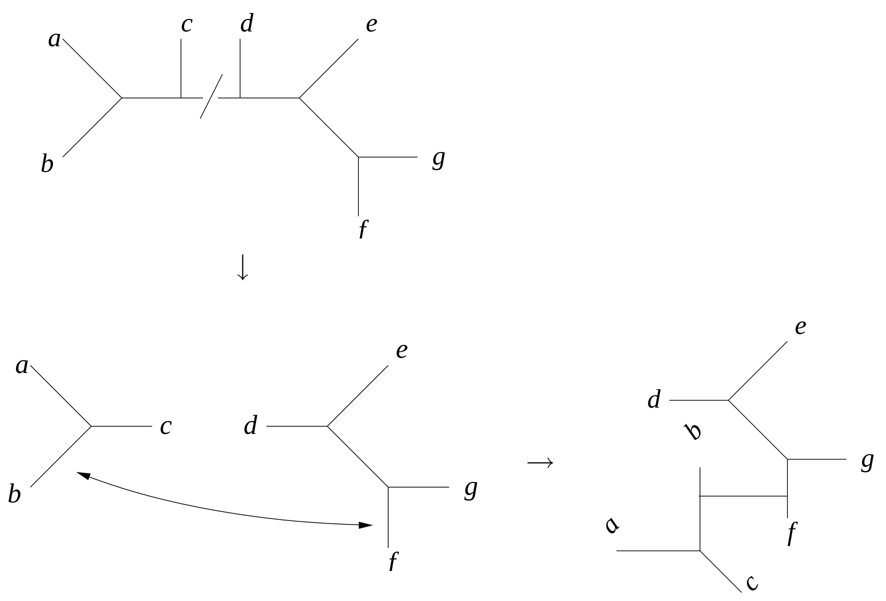
In a *nearest-neighbor interchange (NNI)*, two of the four subtrees around an edge are swapped, in two different ways:



In branch swapping by *subtree pruning and re-grafting*, a subtree is pruned from the tree and re-grafted to a different location of the tree:



In branch swapping by *tree bisection and reattachment*, the tree is bisected at an edge, yielding two subtrees. The two subtrees are then reconnected at two new positions:



1.34 Heuristic search

If the data set $A = \{a_1, \ldots, a_n\}$ is too big to be solved exactly via branch and bound, then we can use a heuristic search method in an attempt to find or approximate the optimal solution.

This involves searching through the space of all unrooted phylogenetic trees on n labels and trying to proceed toward a globally optimal one. We "move" through tree space using one or more branching-swapping techniques.

Heuristic searches employ *hill-climbing techniques*: we imagine that the "goodness" $-PS(T)$ of the solution as a landscape along which we move during the search. The general strategy is to always move upwards in the hope of reaching the top of the highest peak.

Even using the above described branch-swapping techniques, any heuristic search is in danger of "climbing the wrong mountain" and getting stuck in a local optimum. Different strategies have been developed to avoid this problem.

1.35 Simulated annealing

The *simulated annealing* method employs a *temperature* that cools over time [23]. At high temperatures the search can move more easily to trees whose score is less optimal than the score of the current tree. As the temperature decreases, the search becomes more and more directed toward better trees.

I.e., let T_i denote the current tree at step i and let $z(T_i)$ denote the goodness of T_i (e.g., $-PS(T)$). In hill climbing, a move to T_{i+1} is acceptable, if $z(T_{i+1}) \geq z(T_i)$. In simulated annealing, *any* new solution is accepted with a certain probability:

$$Prob(\text{accepting solution } T_{i+1})$$
$$= \begin{cases} 1 & \text{if } z(T_{i+1}) \geq z(T_i) \\ e^{-t_i(z(T_{i+1})-z(T_i))} & \text{otherwise,} \end{cases}$$

where t_i is called the *temperature* and decreases over time.

1.36 The Great Deluge method

The *Great Deluge* method, introduced by Gunter Dueck and Tobias Scheuer[7], employs a slowly rising water level and the search accepts any move that stays above the water level.

The probability of accepting a new solution T_{i+1} is 1, if $z(T_{i+1}) > w_i$, where w_i is a bound that increases slowly with time.

If T_{i+1} is accepted, then we update the water level by setting

$$w_{i+1} = c \times (z(T_{i+1}) - z(T_i)).$$

Typically, the constant c is usually about 0.01 to 0.05.

Another of the many heuristics is *tabu search* method that maintains a *tabu list* of $5 - 10$ solutions recently visited and refrains from revisiting them.

1.37 Maximum Likelihood and Bayesian methods

Any Maximum Likelihood or Bayesian method is based on an explicit model of evolution, such as the Jukes-Cantor model.

In the maximum-likelihood approach, one computes the "likelihood" $P(T \mid A)$ that the true tree is T, given that the alignment A was observed. The method returns:

$$T_{ML} = \max_T P(T \mid A).$$

More desirable is the tree T that maximizes the probability of generating the data A (computed using Bayes' Theorem):

$$T_{Bayesian} = \max_T P(A \mid T).$$

Both approaches are computationally very expensive.

1.38 Maximum Likelihood Estimation (MLE)

Given a multiple alignment $A = \{a_1, \dots, a_n\}$. Assuming a specific model of evolution M, one may attempt to *estimate* a phylogenetic tree T with edge lengths ω that *maximizes the likelihood*

$$P(A \mid T)$$

of generating the sequences $a_1, \dots, a_n$ at the leaves of T.

A main attraction of Maximum Likelihood estimation (MLE) is that it provides a systematic framework for explicitly incorporating assumptions and knowledge about the process that generated the given data.

One potential draw-back is that any given model of evolution is only a rough estimation of real-world biological evolution. Fortunately, in practice, Maximum Likelihood methods have proved to be quite robust to many violations of the assumptions formulated in the models. MLE methods work very well on small data sets.

Similar to Maximum Parsimony, an optimal MLE tree is determined by a search in tree space. One can attempt to find an exact solution, using branch and bound techniques, or one can attempt to find a good approximate solution using a heuristic search technique...

A main draw-back of MLE appears to be that it is very expensive to compute, even more so that Maximum Parsimony.

In the case of Maximum Parsimony, we are able to solve the Small Parsimony problem in polynomial time using the Fitch algorithm.

In the case of MLE, no such fast method of evaluating a single tree is known.

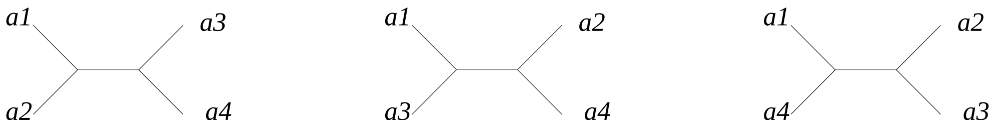
Given a multiple alignment $A = \{a_1, \dots, a_n\}$, MLE seeks to determine the topology (evolutionary branching order) and branch lengths of the true or generating tree.

This is done under the assumption of a model of evolution, such as the Jukes-Cantor model, described above, or more general models. Such models for biological sequences are usually *time reversible* and thus the likelihood of a tree is generally independent of the location of the root.

Example Assume that we are given the following msa $A = \{a_1, \dots, a_4\}$:

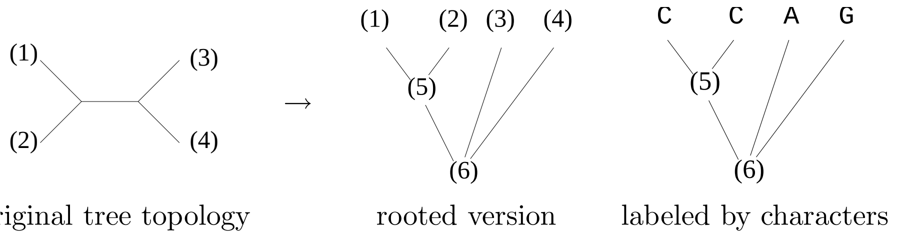
	1	2			i			N
sequence a_1	A	C	...	G	C	G	...	A
sequence a_2	A	C	...	G	C	C	...	G
sequence a_3	A	C	...	T	A	C	...	G
sequence a_4	A	C	...	T	G	G	...	A

There are three possible unrooted tree topologies on four taxa:



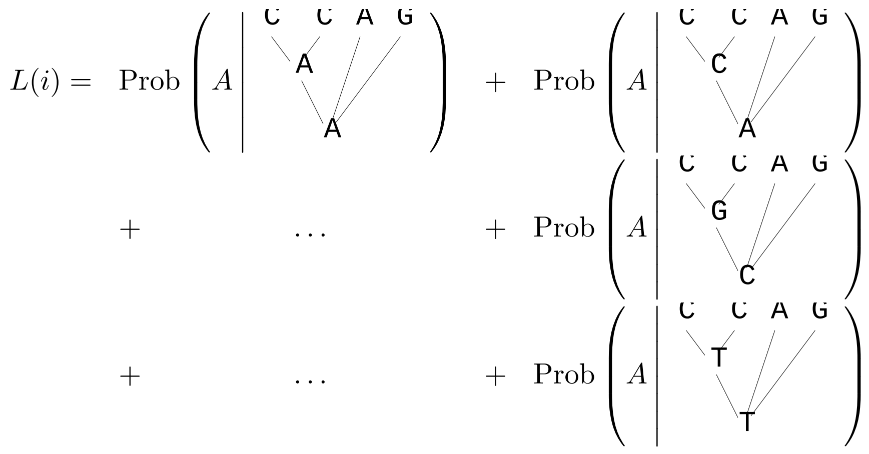
We now discuss how to compute the Maximum Likelihood for the first tree. The other trees are processed similarly.

For simplicity, we root the tree at an arbitrary internal node and then consider each position $i = 1, 2, \dots, N$ in the sequence:



original tree topology rooted version labeled by characters at position i

Schematically, we obtain the likelihood that *this* tree generated the characters seen at position i of the multiple alignment by summing over *all possible* labelings of the internal nodes by characters:


L(i) = Prob(A | C C A G, A, A) + Prob(A | C C A G, C, A)
+ … + Prob(A | C C A G, G, C)
+ … + Prob(A | C C A G, T, T)

We then multiple the likelihoods obtained for each position:

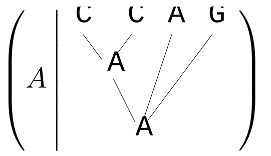
$$L = L(1) \cdot L(2) \cdot \dots \cdot L(N) = \prod_{i=1}^{N} L(i).$$

Obviously, the individual probabilities will often be very small and so we add their logarithms instead of using multiplication:

$$\ln L = \ln L(1) + \ln L(2) + \dots + \ln L(N) = \sum_{i=1}^{N} \ln L(i).$$

Actually, the situation is more complicated as we must determine the choice of edge lengths for the given tree that produces the highest likelihood.

How do we compute e.g. Prob



?

Let us look at this under the Jukes-Cantor model of evolution with a fixed mutation rate u, as discussed above. For any edge e, let P and Q denote the labels at the two opposite ends of e. The probability of $P = Q$ is given by

$$\text{Prob}(Q = P \mid T) = \frac{1}{4}(1 + 3e^{-\frac{4}{3}ut}),$$

and the probability that the two characters differ is

$$1 - \text{Prob}(Q = P \mid T) = \frac{3}{4}(1 - e^{-\frac{4}{3}ut}),$$

where $t = \omega(e)$.

The total probability that *this* tree with *this* labeling of internal nodes generated the observed data at the leaves of the tree is obtained by multiplication of the probabilities for each edge.

1.39 Software

Here is a small selection of software that build phylogenetic trees:

- PAUP* [28], a program for performing phylogenetic analysis using parsimony, Maximum Likelihood and other methods,
- Phylip [10], a package for phylogenetic inference,
- MrBayes [17], a program for Bayesian inference of trees,
- Mesquite [24], a modular system for evolutionary analysis,
- PAL [6], an object-oriented programming library for molecular evolution and phylogenetics, and
- SplitsTree4 [19, 20], an integrated program for estimating phylogenetic trees and networks.

Chapter 2

Consensus networks and super networks

In this chapter we first discuss additional evolutionary events that are not considered in simple models such as the one proposed by Jukes and Cantor.

This will lead us to the fundamental observation that:

gene trees differ.

Hence, it may not be adequate to represent a set of gene trees by a single consensus tree, as is sometimes done, and we will discuss how to represent the conflicting signals using a "consensus network" or "super network".

Finally, we will briefly look at some other methods that use a network to represent conflicting signals.

2.1 Additional evolutionary events

Models such as the Jukes-Cantor one are usually understood to represent the evolution of a *single* gene. They don't consider insertions and deletions, or more complicated events.

If one studies more than one gene simultaneously, additional evolutionary events must be taken into account. E.g.:

- individual genes may be *born*, *duplicated* or *lost*.

Moreover, biological mechanisms such as

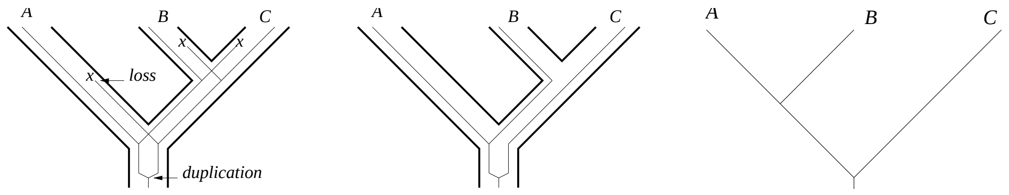
- *recombination*,
- *hybridization*, or
- *horizontal gene transfer*

may be involved.

2.2 Gene trees can differ

Now, suppose we are given one or more genes for X. Consider a model in which the sequence of a gene evolves via mutations, but we also allow gene *duplication* or *loss*.

The true phylogeny of a gene can differ from the model phylogeny. here we depict a species phylogeny using bold parallel lines and the history of a single gene by thin lines:



So, true “gene trees” can differ from the true “species phylogeny” and also from each other.

2.3 The split encoding of a tree

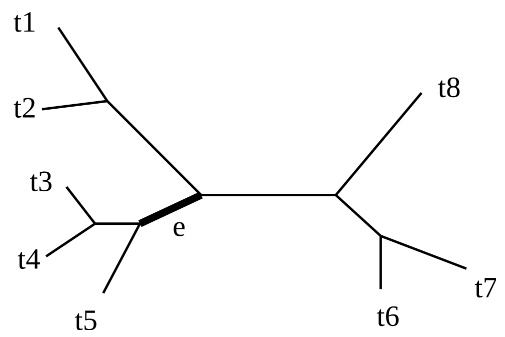
Let $X = \{x_1, \dots, x_n\}$ be a set of taxa and $g_1, \dots, g_k$ a set of genes that are present in all taxa. For each gene g_i we are given a sequence alignment A_i.

Assume that we have reconstructed a phylogenetic tree T_i for each gene g_i. The goal is to compute a *consensus* of these trees. To this end, we introduce the following concepts.

An X-*split* $S = \frac{A}{B} (= \frac{B}{A})$ is a bipartitioning of X with [1]:

$$A, B \neq \emptyset, \; A \cap B = \emptyset \text{ and } A \cup B = X.$$

Any edge e of T defines a split $S = \frac{A}{B}$, where A and B are the sets of taxa contained in the two sub-trees defined by e. E.g.:

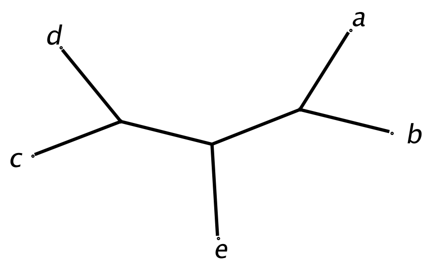


For the edge labeled e we get:

$$A = \{t_3, t_4, t_5\} \text{ and } B = \{t_1, t_2, t_6, t_7, t_8\}.$$

Let $\Sigma(T)$ denote the *split encoding of* T, i.e. the set of all splits obtained from T.

Consider the tree T:



The split encoding $\Sigma(T)$ contains 5 *trivial* splits and 2 *non-trivial* ones. The trivial splits are:

$$\frac{\{a\}}{\{b,c,d,e\}}, \frac{\{b\}}{\{a,c,d,e\}}, \frac{\{c\}}{\{a,b,d,e\}}, \frac{\{d\}}{\{a,b,c,e\}} \text{ and } \frac{\{e\}}{\{a,b,c,d\}},$$

and the non-trivial ones are:

$$\frac{\{a,b\}}{\{c,d,e\}} \text{ and } \frac{\{a,b,e\}}{\{c,d\}}.$$

2.4 Trees and splits

Two different X-splits $S = \frac{A}{B}$ and $S' = \frac{A'}{B'}$ are *compatible*, if "one is a refinement of the other", that is, if one of the four following inclusions holds:

$$A \subset A',\ A \subset B',\ B \subset A',\ \text{or } B \subset B'.$$

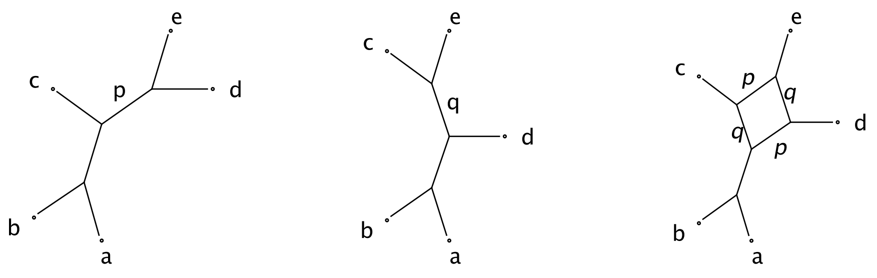
This is an important concept, as we have:

Lemma Let Σ be a set of X-splits. Then there exists an unique X-tree T with $\Sigma(T)$ iff Σ is compatible [4].

2.5 Representing incompatible splits

Any *compatible* set of X-splits can be represented by a phylogenetic tree. What about incompatible splits sets?

Consider the following two trees T_1 and T_2, for which the splits $S_p = \frac{\{a,b,c\}}{\{d,e\}} \in \Sigma(T_1)$ and $S_q = \frac{\{a,b,d\}}{\{c,e\}} \in \Sigma(T_2)$ are incompatible:



T_1 $\quad$ T_2 $\quad$ $SN(\Sigma)$

The "splits network" $SN(\Sigma)$ represents the incompatible set of splits $\Sigma := \Sigma(T_1) \cup \Sigma(T_2)$, using "bands of parallel edges" to represent splits that are incompatible with others [5].

2.6 Consensus of trees

A collection of trees $\mathcal{T} = \{T_1, \ldots, T_K\}$ is often summarized using a consensus tree.

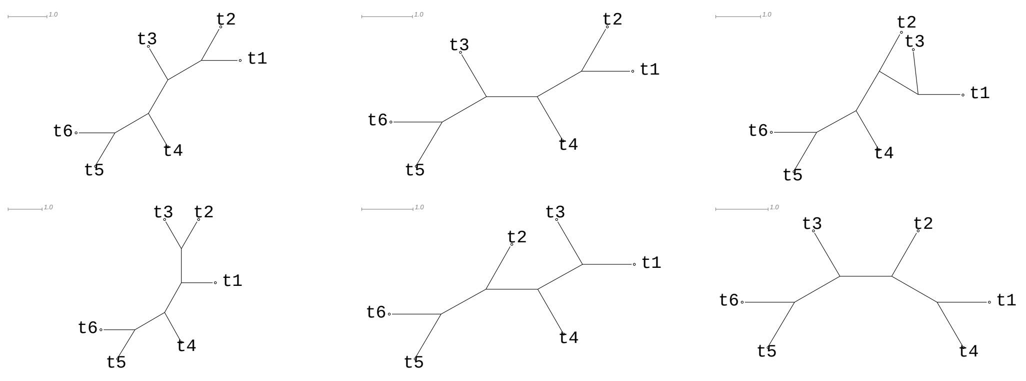
Let $\Sigma_{all} = \cup_{T \in \mathcal{T}} \Sigma(T)$ be the set of all present splits.

Let $\Sigma(p) = \{S \in \Sigma_{all} : |\{T \in \mathcal{T} : S \in \Sigma(T)\}| > pK\}$ be the set of splits that occur in more than a proportion p of all trees. Then,

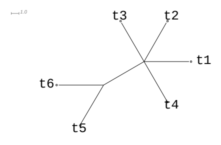
- $\Sigma^*(1) := \bigcap_i \Sigma(T_i)$ defines the *strict consensus*,
- $\Sigma(\frac{1}{2})$ defines the *majority consensus*, and, more generally,
- $\Sigma(\frac{1}{d+1})$ $(d \geq 0)$ defines a set of *consensus splits.*

Note that both $\Sigma^*(1)$ and $\Sigma(\frac{1}{2})$ are always compatible and thus correspond to trees, whereas $\Sigma(\frac{1}{d+1})$ with $d \geq 0$ may be incompatible and thus is usually represented by a network.

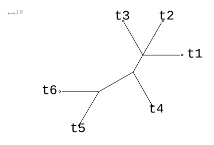
For example, given these trees as input:



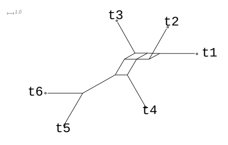
We get these consensus trees and networks:



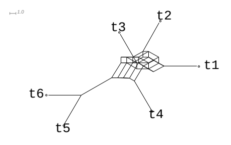
$\Sigma(\frac{1}{2}) = \Sigma^*(1)$



$\Sigma(\frac{1}{3})$



$\Sigma(\frac{1}{6})$



$\Sigma(0)$

2.7 Consensus networks

Often, a set of trees $\mathcal{T} = \{T_1, \ldots, T_K\}$ is summarized using a consensus *tree.*

This may not always be appropriate, as gene trees are not necessarily different estimations of the same true phylogeny, but may differ substantially for biological reasons.

A *consensus network* is obtained by computing the consensus splits $\Sigma(\frac{1}{d+1})$ for some fixed value $d \geq 0$.

The parameter d sets the *maximum dimensionality* of the corresponding network: for $d = 1$ the network will be 1-dimensional, hence a tree, for $d = 2$ the network may contain parallelograms, and in general it may contain cubes of dimension $\leq d$ [15, 14].

2.8 Consensus super networks

Consider a set of taxa $X = \{x_1, \dots, x_n\}$ and a set of genes $G = \{g_1, \dots, g_t\}$.

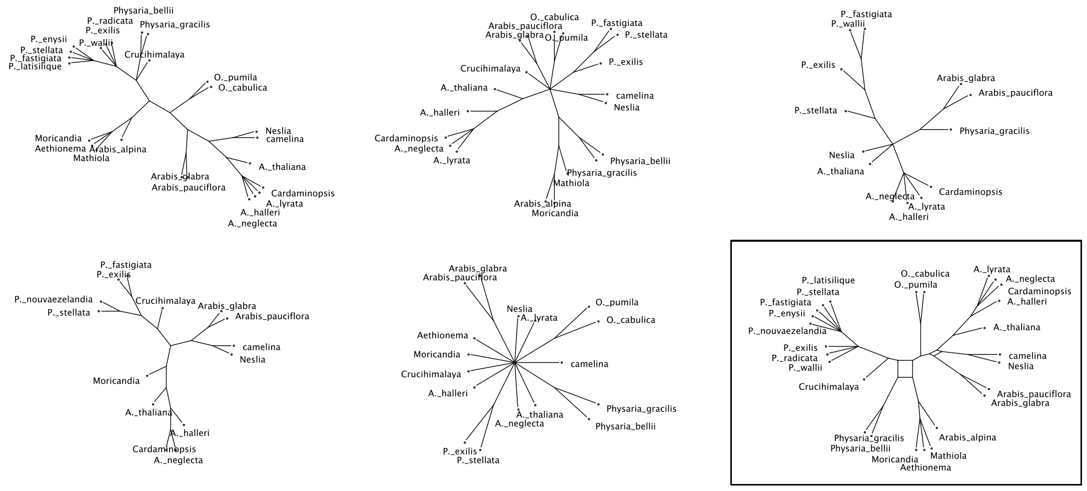
It is often the case that a given gene is not available for all taxa and the alignment A_i associated with some gene g_i only contains sequences for a subset $X' \subset X$. Then, any X'-tree inferred from A_i is called a *partial* X-tree.

For a collection of *partial* trees $\mathcal{T} = \{T_1, \dots, T_K\}$, the consensus methods above do not apply.

One alternative is to compute an optimal *super tree* T that "optimally" summarizes the set of input trees.

A second approach is to summarize the input trees in terms of a *super network* that attempts to represent as many of the input "partial" splits as possible.

The *Z-closure method* [21] takes as input a set of partial X-trees $\mathcal{T} = \{T_1, \dots, T_K\}$ and produces as output a set of X-splits Σ. Here is an example of five partial gene trees and a summarizing super network:

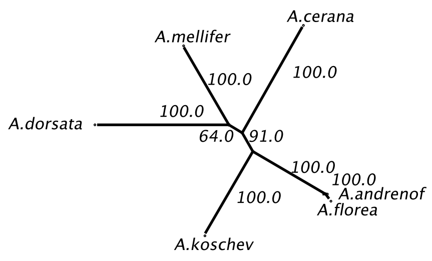


super network

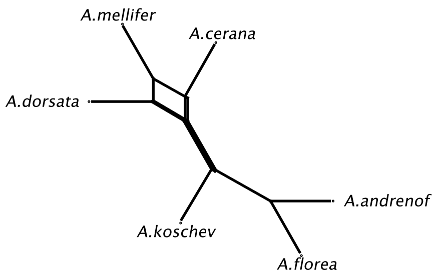
2.9 Bootstrap network

One popular way to study how robust the different branches of an inferred tree, is to generate many "bootstrap replicates" by randomly resampling from the given alignment A. Then, every branch of the originally inferred tree is labeled by the percentage of replicates that yield the corresponding split.

We propose to construct a *bootstrap network* [20] by collecting all splits that are present in any of the replicates and displaying them in a splits network:



NJ tree with bootstrap values



bootstrap network

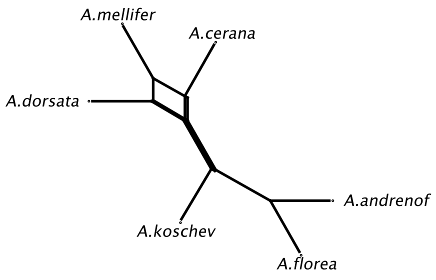
2.10 Distance-based network methods

So, incompatible splits arise naturally in the context of consensus. There also exist a number of methods that generate incompatible splits directly from a distance matrix.

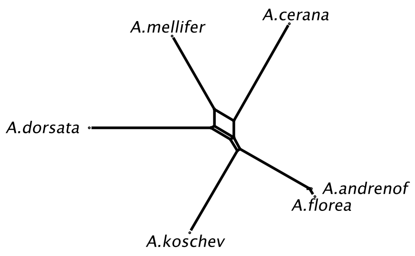
The *split decomposition* [1] takes as input a distance matrix D on X and produces a set of *weighted* X-splits Σ_{decomp}, where the sum of weights of all splits that "separate" two taxa $x, y \in X$ is an approximation of the given distance $D(x, y)$.

This method has the nice property that it produces a tree, whenever the distance matrix fits a tree, and otherwise it produces a tree-like splits network that potentially displays different and conflicting signals in a given data set.

To illustrate this, compare the bootstrap network with the network produced using the split decomposition method:



bootstrap network

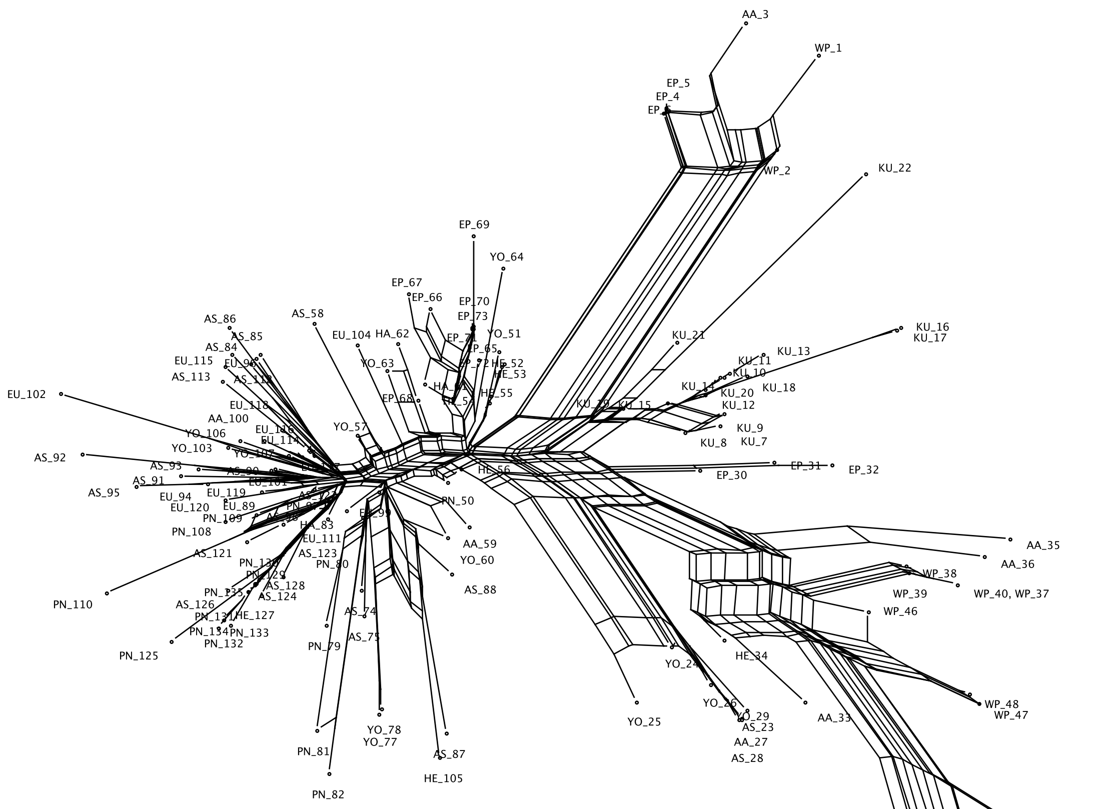


split decomposition

Here, both the bootstrap analysis and split decomposition indicate that the input sequences contain two different and incompatible signals.

The split decomposition is a useful for visualizing conflicting signals in a data set. However, it is sensitive to noise and only has good resolution for data sets of up to about 20 taxa.

The *Neighbor-Net* method [3] is a hybrid of Neighbor-Joining and split decomposition. It is applicable to data sets containing hundreds of taxa. Here is an example based on human mtDNA:



2.11 Software

- SplitsTree4 [20] provides implementations of *all* methods described in this chapter, including a number of different algorithms for constructing networks from splits.
- SpectroNet [16] provides an algorithm for constructing a splits network (a special case, namely the median network) and some related methods

Bibliography

[1] H.-J. Bandelt and A. W. M. Dress. A canonical decomposition theory for metrics on a finite set. *Advances in Mathematics*, 92:47–105, 1992.

[2] W. J. Bruno, N. D. Socci, and A. L. Halpern. Weighted Neighbor Joining: A likelihood-based approach to distance-based phylogeny reconstruction. *Molecular Biology and Evolution*, 17(1):189–197, 2000.

[3] D. Bryant and V. Moulton. NeighborNet: An agglomerative method for the construction of planar phylogenetic networks. In R. Guigó and D. Gusfield, editors, *Algorithms in Bioinformatics, WABI 2002*, volume LNCS 2452, pages 375–391, 2002.

[4] P. Buneman. The recovery of trees from measures of dissimilarity. In F. R. Hodson, D. G. Kendall, and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1971.

[5] A. W. M. Dress and D. H. Huson. Constructing splits graphs. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 1(3):109–115, 2004.

[6] A. Drummond and K. Strimmer. PAL: An object-oriented programming libary for molecular evolution and phylogenetics. *Bioinformatics*, 17:662–663, 2001.

[7] G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175, 1990.

[8] A.W.F. Edwards and L.L. Cavalli-Sfroza. The reconstruction of evolution. *Annals of Human Genetics*, 27:105–106, 1963.

[9] A.W.F. Edwards and L.L. Cavalli-Sfroza. Reconstruction of evolutionary trees. In V.H. Heywood and J. NcNeill, editors, *Phenetic and Phylogenetic Classification*, volume 6, pages 67–76. Systematics Association, London, 1964.

[10] J. Felsenstein. PHYLIP – phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[11] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Inc., 2004.

[12] W. Fitch. Toward defining the course of evolution: minimum chnage for a specific tree topology. *Syst. Zool.*, 20:406–416, 1971.

[13] O. Gascuel. BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, 14:685–695, 1997.

[14] B. Holland, K. Huber, V. Moulton, and P. J. Lockhart. Using consensus networks to visualize contradictory evidence for species phylogeny. *Molecular Biology and Evolution*, 21:1459–1461, 2004.

[15] B. Holland and V. Moulton. Consensus networks: A method for visualizing incompatibilities in collections of trees. In G. Benson and R. Page, editors, *Proceedings of "Workshop on Algorithms in Bioinformatics"*, volume 2812 of *LNBI*, pages 165–176. Springer, 2003.

[16] K. T. Huber, M. Langton, D. Penny, V. Moulton, and M. Hendy. Spectronet: A package for computing spectra and median networks. *Applied Bioinformatics*, 1:159–161, 2002.

[17] J.P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

[18] J.P. Huelsenbeck, F. Ronquist, R. Nielsen, and J.P. Bollback. Bayesian inference of phylogeny and its impact on evolutionary biology. *Science*, 294:2310–2314, 2001.

[19] D. H. Huson. SplitsTree: A program for analyzing and visualizing evolutionary data. *Bioinformatics*, 14(10):68–73, 1998.

[20] D. H. Huson and D. Bryant. Estimating phylogenetic trees and networks using SplitsTree 4. Manuscript in preparation, software available from `www.splitstree.org`, 2005.

[21] D. H. Huson, T. Dezulian, T. Kloepper, and M. A. Steel. Phylogenetic super-networks from partial trees. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 1(4):151–158, 2004.

[22] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, 1969.

[23] P.J.M. Van Laarhoven and E.H.L. Aarts. *Simulated annealing: theory and applications.* 1987.

[24] W. Maddison and D. Maddison. Mesquite- a modular system for evolutionary analysis. version 1.05. `http://mesquiteproject.org`, 2005.

[25] N. Saitou and M. Nei. The Neighbor-Joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[26] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

[27] J. A. Studier and K. J. Keppler. A note on the neighbour-joining algorithm of Saitou and Nei. *Mol. Biol. Evol.*, 5:729–731, 1988.

[28] D. L. Swofford. PAUP*: Phylogenetic analysis using parsimony ($*$: and other methods), version 4.2, 2000.