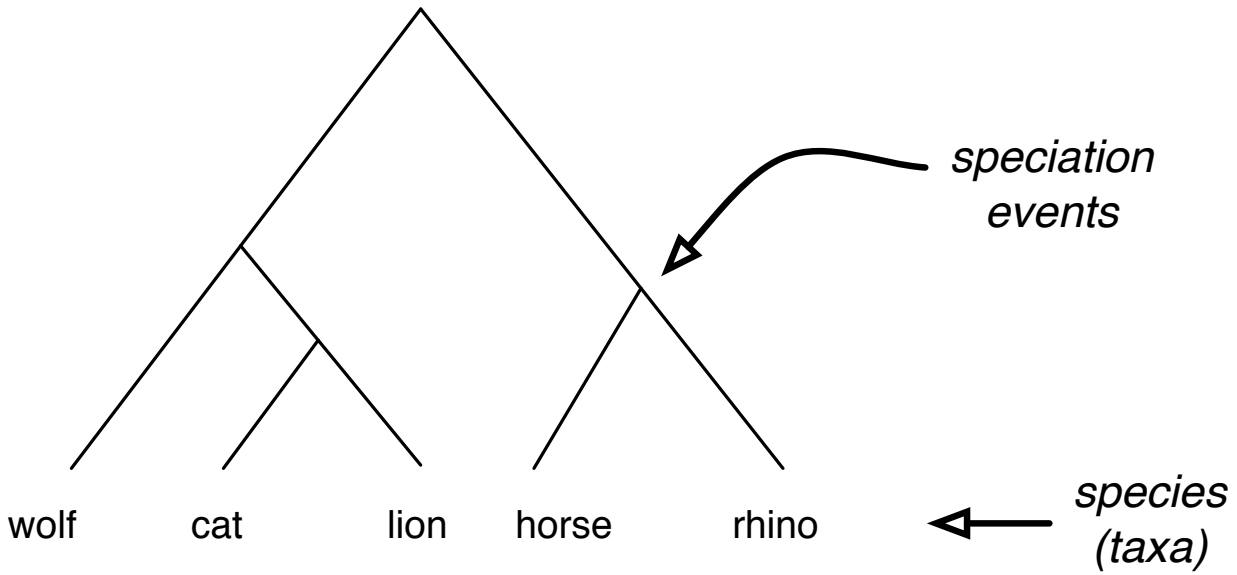
Algorithms for Computational Biology

Zsuzsanna Lipták

Masters in Molecular and Medical Biotechnology
a.a. 2015/16, fall term

Phylogenetics Summary

What is a phylogenetic tree?



Phylogenetic trees display the evolutionary relationships among a set of objects (species). Contemporary species are represented by the leaves. Internal nodes of the tree represent speciation events ($\approx$ common ancestors, usually extinct).

Different types of phylogenetic trees

- rooted vs. unrooted (root on top/bottom vs. root in the middle)
- binary (fully resolved) vs. multifurcating (polytomies)
- are edge lengths significant?
- is there a time scale on the side?

Phylogenetic reconstruction

Goal

Given n objects and data on these objects, find a phylogenetic tree with these objects at the leaves which best reflects the input data.

Phylogenetic reconstruction

Note:

We need to define more precisely

- what kind of input data we have,
- what kind of tree we want (e.g. rooted or unrooted), and
- what we mean by "reflect the data."

Phylogenetic reconstruction

There are two main issues:

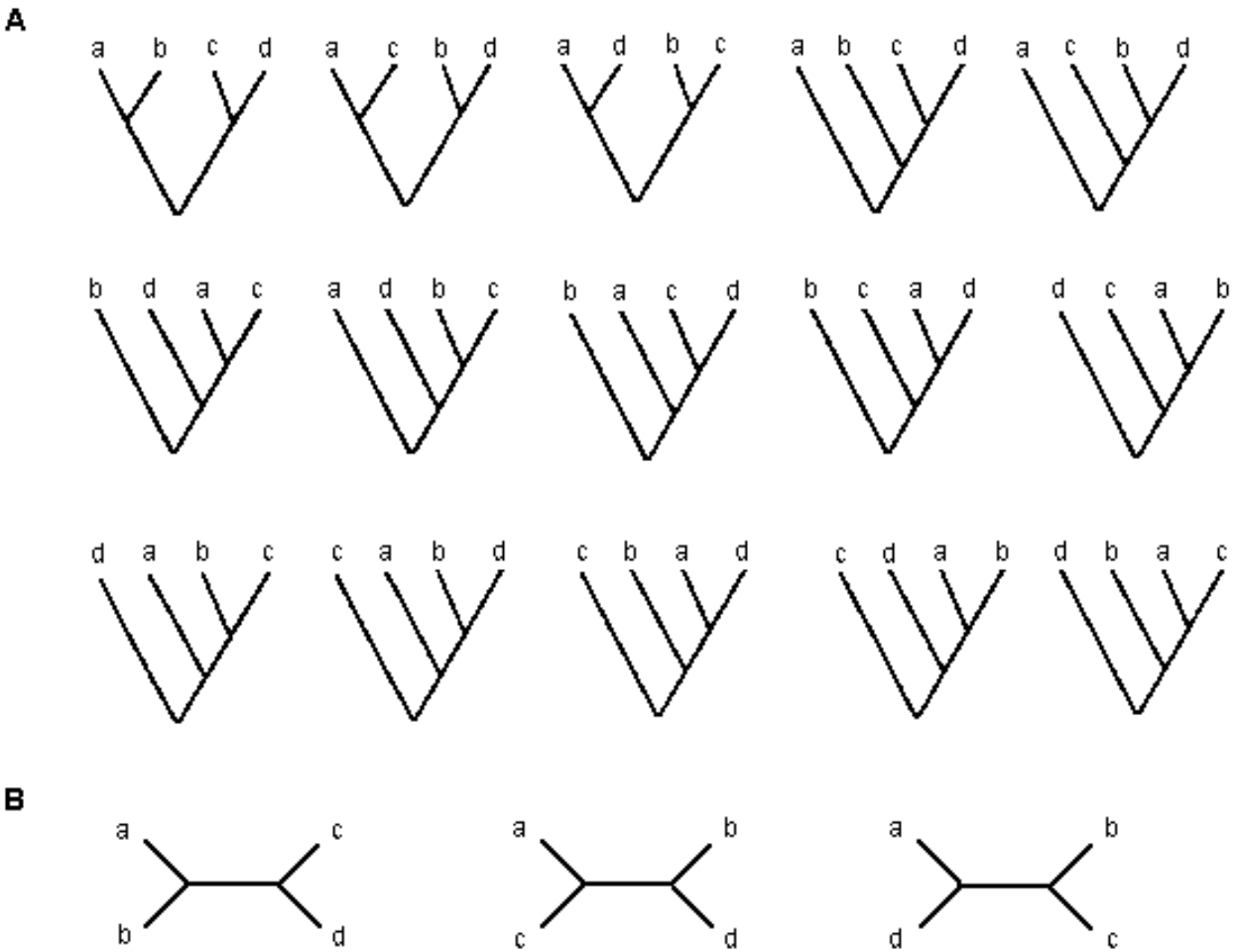
1. How well does a tree reflect my data?
2. How do we find such a tree?

Number of phylogenetic trees

Say we have answered these questions, then: Could we just list all possible trees and then choose the/a best one?

# taxa n	# unrooted trees $(2n-5)!!$	# rooted trees $(2n-3)!!$
1	1	1
2	1	1
3	1	3
4	3	15

Number of phylogenetic trees



All phylogenetic trees (rooted and unrooted) on 4 taxa.

Number of phylogenetic trees

Theorem

There are $U_n = (2n-5)!! = \prod_{i=3}^{n}(2i-5)$ unrooted binary phylogenetic trees on n objects, and $R_n = (2n-3)!! = \prod_{i=2}^{n}(2i-3)$ rooted binary phylogenetic trees on n objects.

Proof

By induction on n, using that (1) we can get every unrooted tree on $n+1$ objects in a unique way by adding the $(n+1)$st leaf to an unrooted tree on the first n objects; (2) an unrooted binary tree with n leaves has $2n-3$ edges, (3) every unrooted tree on n objects can be rooted in (number of edges) ways, yielding a rooted tree on n objects.

Number of phylogenetic trees

#taxa n	#unrooted trees $(2n-5)!!$	#rooted trees $(2n-3)!!$
1	1	1
2	1	1
3	1	3
4	3	15
5	15	105
6	105	945
7	945	10,395
8	10,395	135,135
9	135,135	2,027,025
10	2,027,025	34,459,425

Number of phylogenetic trees

So there are super-exponentially many trees:
We cannot check all of them!

Types of input data

We can have two kinds of input data:

- distance data: $n \times n$ matrix of pairwise distances between the taxa, or
- character data: $n \times m$ matrix giving the states of m characters for the n taxa

Distance data

Distance data is given as an $(n \times n)$ matrix M with the pairwise distances between the taxa.

Ex.

	a	b	c
a	0	5	2
b	5	0	4
c	2	4	0

E.g., $M_{a,b} = 5$ means that the distance between a and b is 5. Often, this is the edit distance (between two genomic sequences, or between homologous proteins, ...).

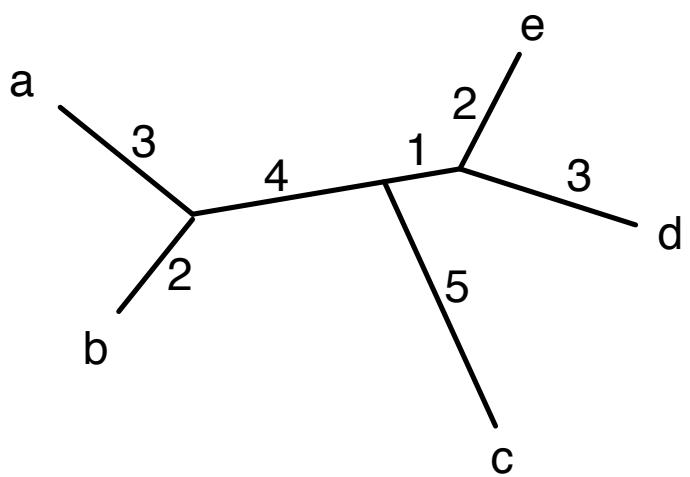
We want to find a tree with a, b, c at the leaves s.t. the distance in the tree (the path metric) between a and b is 5, between a and c is 2, etc.

Distance data

Path metric of a tree

Given a tree T, the path-metric of T is d_T, defined as: $d_T(u, v) =$ sum of edge weights on the (unique) path between u and v.

Example



$$d_T(a, b) = 5,$$
$$d_T(a, d) = 11,$$
$$d_T(c, d) = 9, \ldots$$

Note

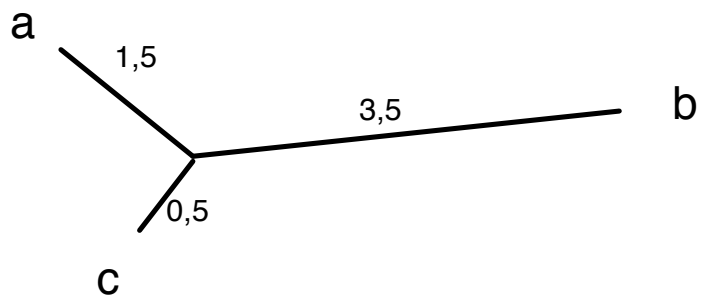
$d(u, v)$ is also defined for inner nodes u, v, but we only need it for leaves.

Example

For our earlier example, we can find such a tree:

Ex. 1 (from before)

	a	b	c
a	0	5	2
b	5	0	4
c	2	4	0

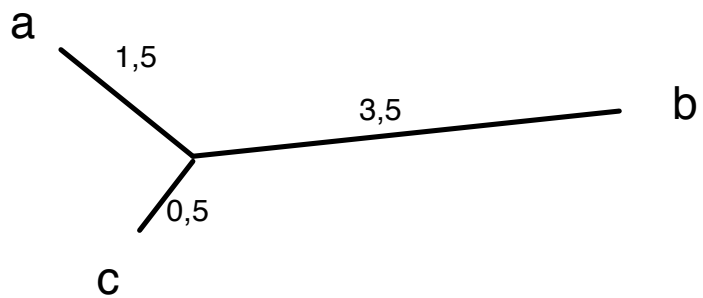


Example

For our earlier example, we can find such a tree:

Ex. 1 (from before)

	a	b	c
a	0	5	2
b	5	0	4
c	2	4	0



Question

Is it always possible to find a tree s.t. its path-metric equals the input distances? I.e. does such a tree exist for any input matrix M?

Distance data

First of all, the input matrix M has to define a metric (= a distance function), i.e. for all x, y, z,

Distance data

First of all, the input matrix M has to define a metric (= a distance function), i.e. for all x, y, z,

- $M(x, y) \geq 0$ and $(M(x, y) = 0$ iff $x = y)$ (positive definite)
- $M(x, y) = M(y, x)$ (symmetry)
- $M(x, y) + M(y, z) \geq M(x, z)$ (triangle inequality)

For example, the edit distance is a metric (on strings), the Hamming distance (on strings of the same length), the Euclidean distance (on $\mathbb{R}^2$).

Conditions on distance matrix

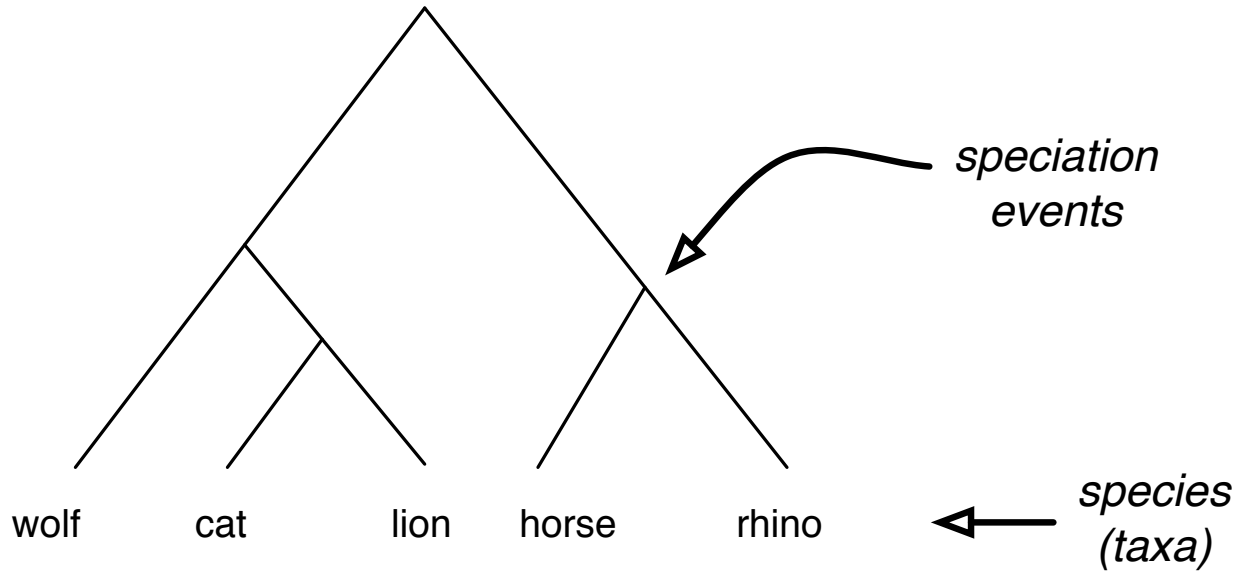
Question:

When does a tree exist whose path metric agrees with a distance matrix M?

Answer:

- if we want a rooted tree: M needs to be ultrametric
- if we want an unrooted tree: M needs to be additive

Rooted trees and the molecular clock



In a rooted phylogenetic tree, the molecular clock assumption holds: that the speed of evolution is the same along all branches, i.e. the path distance from each leaf to the root is the same. Such a tree is also called an ultrametric tree.

Ultrametrics and the three-point condition

Three point condition

Let d be a metric on a set of objects O, then d is an ultrametric if $\forall\ x, y, z \in O$:

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

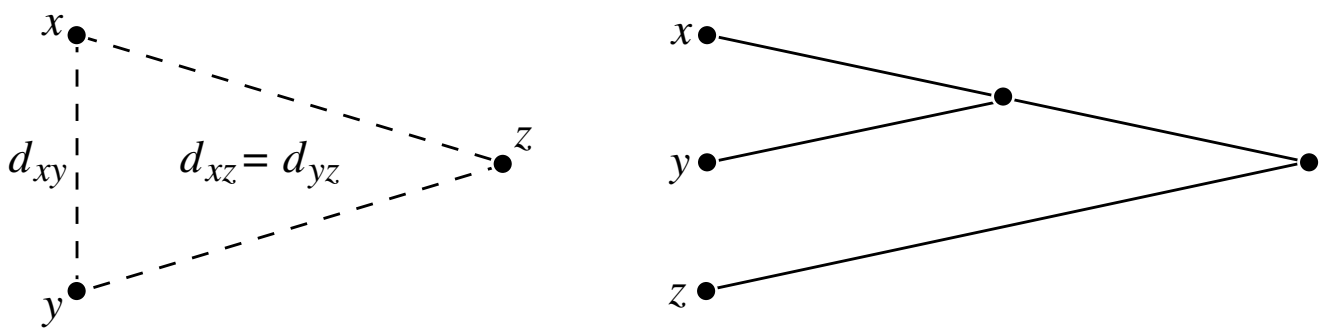


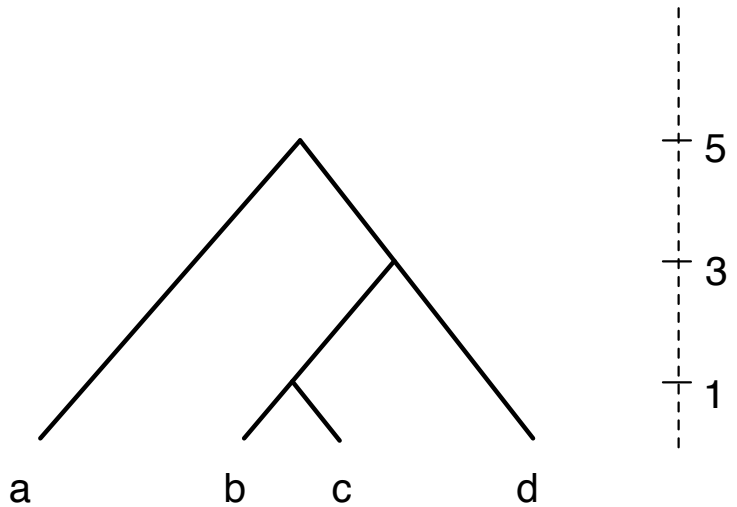
Figure : Three point condition. It implies that the path metric of a rooted tree is an ultrametric.

In other words, among the three distances, there is no unique maximum.

Example

Ex. 2

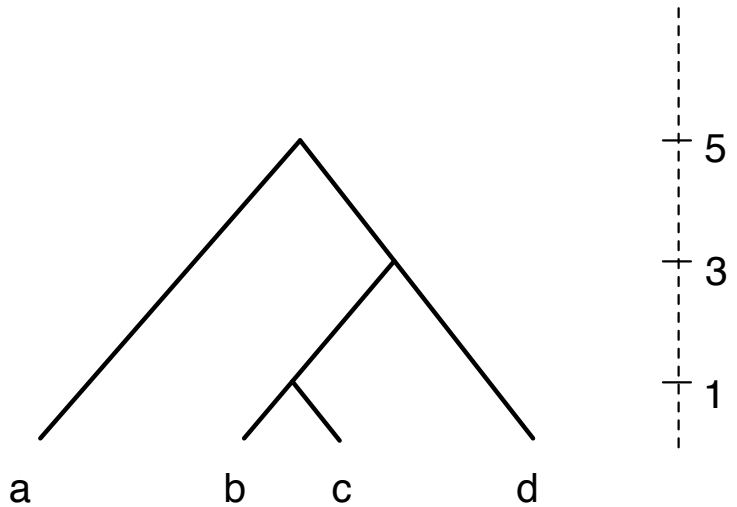
	a	b	c	d
a	0	10	10	10
b	10	0	2	6
c	10	2	0	6
d	10	6	6	0



Example

Ex. 2

	a	b	c	d
a	0	10	10	10
b	10	0	2	6
c	10	2	0	6
d	10	6	6	0



Checking the ultrametric condition, we see that:

- for a, b, c we get $2, 10, 10$ — okay
- for a, b, d we get $6, 10, 10$ — okay
- for a, c, d we get $6, 10, 10$ — okay
- for b, c, d we get $2, 6, 6$ — okay

Example

Compare this to our earlier example. There the matrix M does not define an ultrametric!

Ex. 1 (from before)

	a	b	c
a	0	5	2
b	5	0	4
c	2	4	0

For the triple a, b, c (the only triple), we get: $2, 4, 5$, and there is a unique maximum: 5.

Example

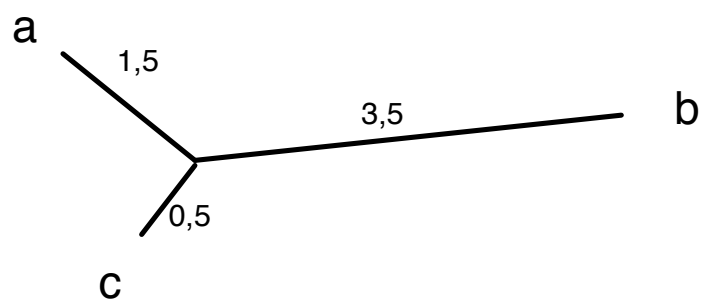
Compare this to our earlier example. There the matrix M does not define an ultrametric!

Ex. 1 (from before)

	a	b	c
a	0	5	2
b	5	0	4
c	2	4	0

For the triple a, b, c (the only triple), we get: $2, 4, 5$, and there is a unique maximum: 5.

Indeed, the only tree we found was not rooted:



Ultrametrics and the three-point condition

Theorem

Given an $(n \times n)$ distance matrix M. There is a rooted tree whose path metric agrees with M if and only if M defines an ultrametric (i.e. if and only if the 3-point-condition holds). This tree is unique[1].

[1] i.e. there is only one such tree

Ultrametrics and the three-point condition

Theorem

Given an $(n \times n)$ distance matrix M. There is a rooted tree whose path metric agrees with M if and only if M defines an ultrametric (i.e. if and only if the 3-point-condition holds). This tree is unique[1].

Algorithm

The algorithm UPGMA (*unweighted pair group mtheod using arithmetic averages*, Michener & Sokal 1957), a hierarchical clustering algorithm, constructs such a tree, given an input matrix which is ultrametric. Its running time is $O(n^2)$.

[1] i.e. there is only one such tree

Additive metrics and the four-point condition

So what is the condition on the matrix M for unrooted trees?

Four point condition.

Let d be a metric on a set of objects O, then d is an additive metric if $\forall\ x, y, u, v \in O$:

$$d(x, y) + d(u, v) \leq \max\{d(x, u) + d(y, u), d(x, v) + d(y, u)\}$$

In other words, among the three sums of two distances, there is no unique maximum.

Additive metrics and the four-point condition

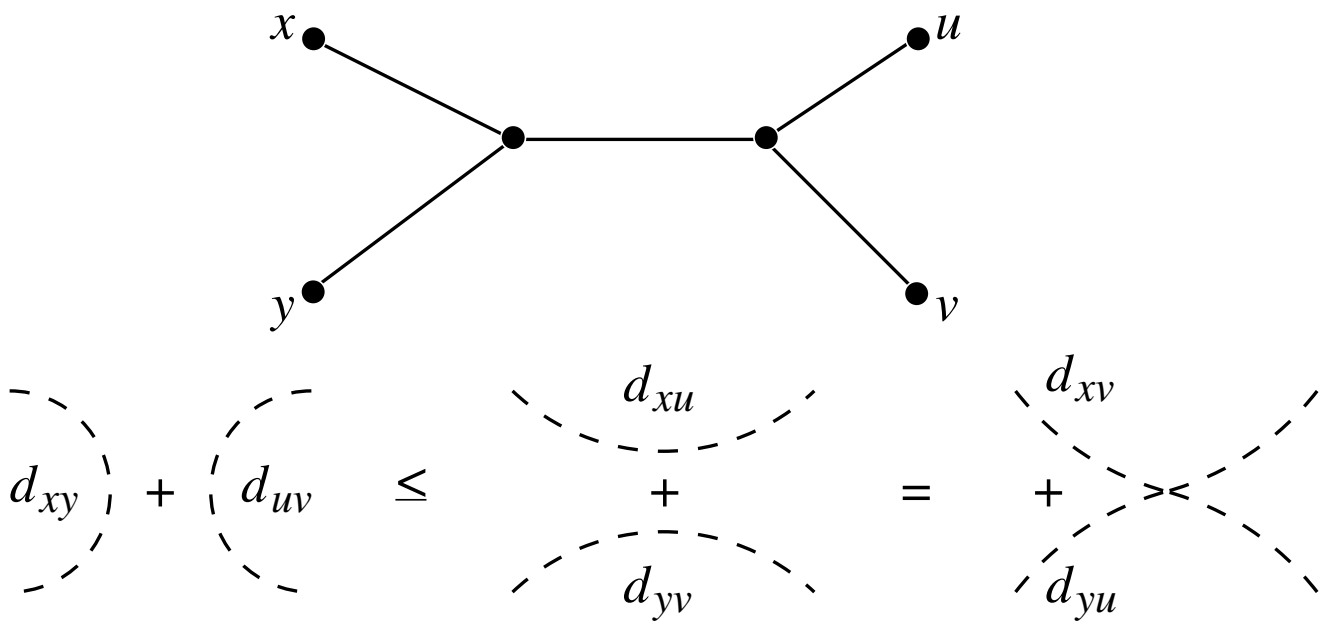
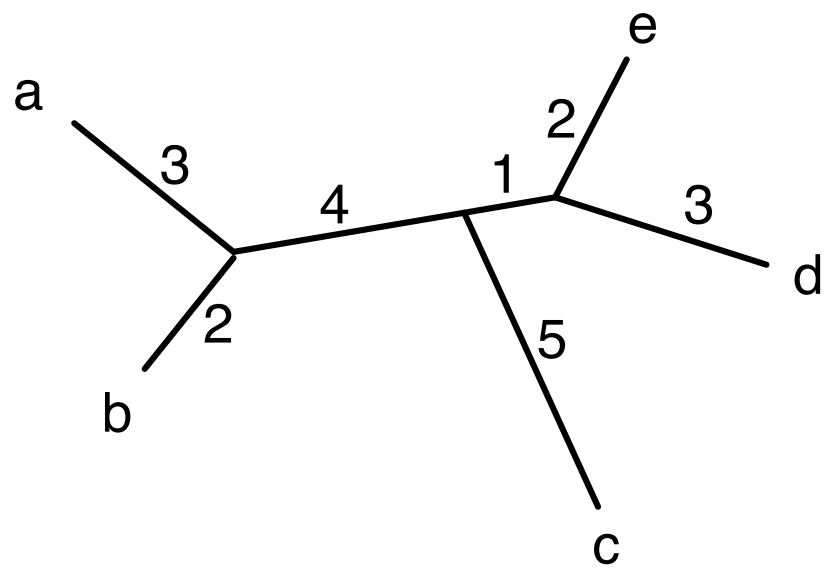


Figure : The four point condition. It implies that the path metric of a tree is an additive metric.

Example



For ex., choose these 4 points: a, b, c, e. Then we get the three sums: $d(a,b) + d(c,e) = 5 + 8 = 13$, $d(a,c) + d(b,e) = 12 + 9 = 21$, and $d(a,e) + d(b,c) = 10 + 11 = 21$. Among $13, 21, 21$, there is no unique maximum—okay. (Careful, this has to hold for all quadruples; how many are there?)

Additive metrics and the four-point condition

Theorem

Given an $(n \times n)$ distance matrix M. There is an unrooted tree whose path metric agrees with M if and only if M defines an additive metric (i.e. if and only if the 4-point-condition holds). This tree is unique.

Algorithm

There are algorithms which, given M, compute this unrooted tree in $O(n^3)$ time (e.g. Neighbor Joining, Saitu & Nei, 1987).

Additive metrics and the four-point condition

Theorem

Given an $(n \times n)$ distance matrix M. There is an unrooted tree whose path metric agrees with M if and only if M defines an additive metric (i.e. if and only if the 4-point-condition holds). This tree is unique.

Algorithm

There are algorithms which, given M, compute this unrooted tree in $O(n^3)$ time (e.g. Neighbor Joining, Saitu & Nei, 1987).

In fact, it is even possible to compute a "good" tree if the matrix is not additive but "almost" *(all this needs to be defined precisely, of course).*

Summary for distance data

- When the input is a distance matrix, then we are looking for a tree whose path metric agrees with M.

Summary for distance data

- When the input is a distance matrix, then we are looking for a tree whose path metric agrees with M.
- A rooted tree agreeing with M exists if and only if the distance matrix M defines an ultrametric.

Summary for distance data

- When the input is a distance matrix, then we are looking for a tree whose path metric agrees with M.
- A rooted tree agreeing with M exists if and only if the distance matrix M defines an ultrametric.
- This tree can then be computed efficiently (i.e. in polynomial time), with UPGMA - which we studied!
- An unrooted tree agreeing with M exists if and only if the distance matrix M defines an additive metric. It can then computed efficiently (i.e. in polynomial time), with Neighbor Joining - which we did not study.

Character data

Now the input data consists of states of characters for the given objects, e.g.

- morphological data, e.g. number of toes, reproductive method, type of hip bone, . . . or
- molecular data, e.g. what is the nucletoide in a certain position.

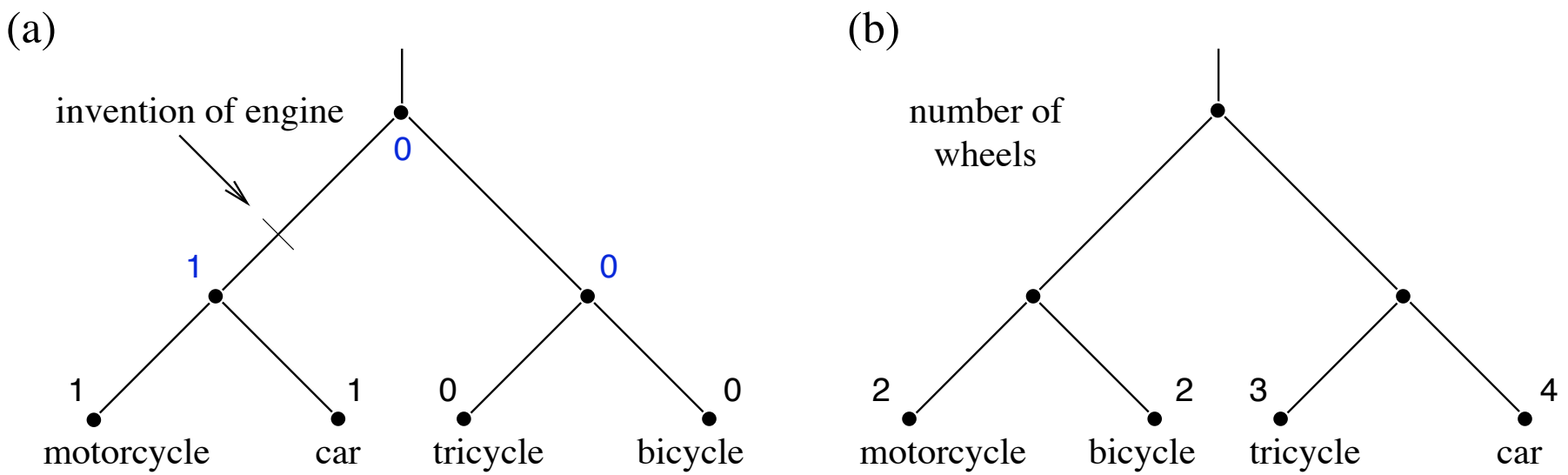
Character data

Example

	C_1 : # wheels	C_2 : existence of engine
bicycle	2	0
motorcycle	2	1
car	4	1
tricycle	3	0

- objects (species): Bicycle, motorcycle, tricycle, car
- characters: number of wheels; existence of an engine
- character states: $2, 3, 4$ for C_1;
  $0, 1$ for C_2 ($1 =$ YES, $0 =$ NO)
- This matrix M is called a character-state-matrix, of dimension $(n \times m)$, where for $1 \leq i \leq n, 1 \leq j \leq m$: $M_{ij} =$ state of character j for object i. (Here: $n = 4, m = 2$.)

Character data



Two different phylogenetic trees for the same set of objects.

Character data

We want to avoid

- parallel evolution (= convergence)
- reversals

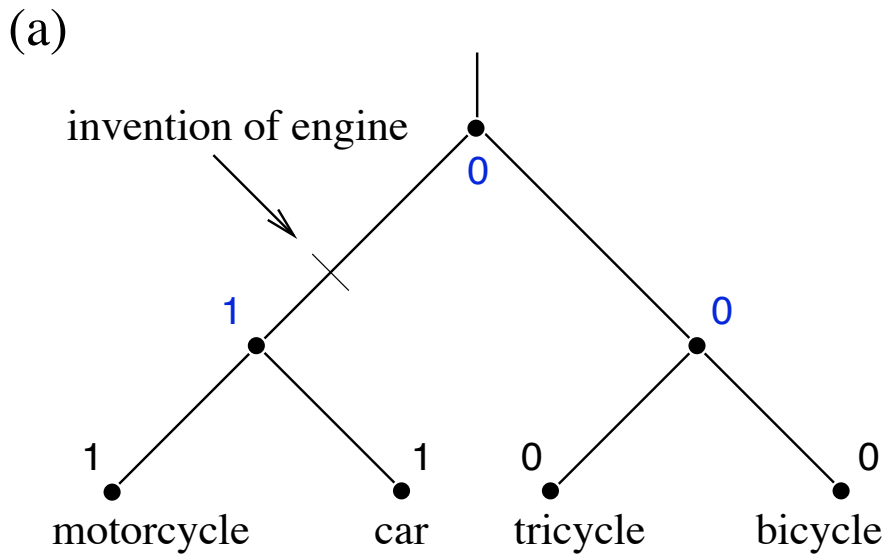
Together these two conditions are also called homoplasies.

Mathematical formulation: compatibility.

Compatibility

Definition

A character is compatible with a tree if all inner nodes of the tree can be labeled such that each character state induces one connected subtree.



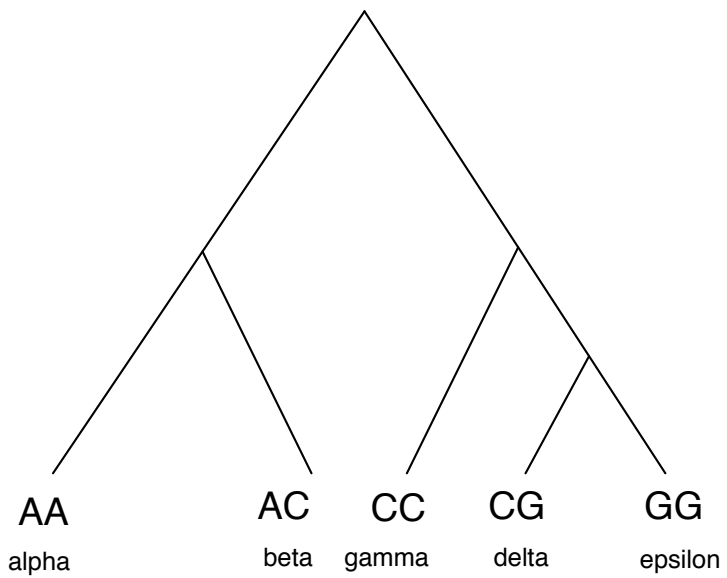
This tree is compatible with C_2, one possibility of labeling the inner nodes is shown.

Perfect Phylogeny

Definition

A tree T is called a perfect phylogeny (PP) for $\mathcal{C}$ if all characters $C \in \mathcal{C}$ are compatible with T.

Example



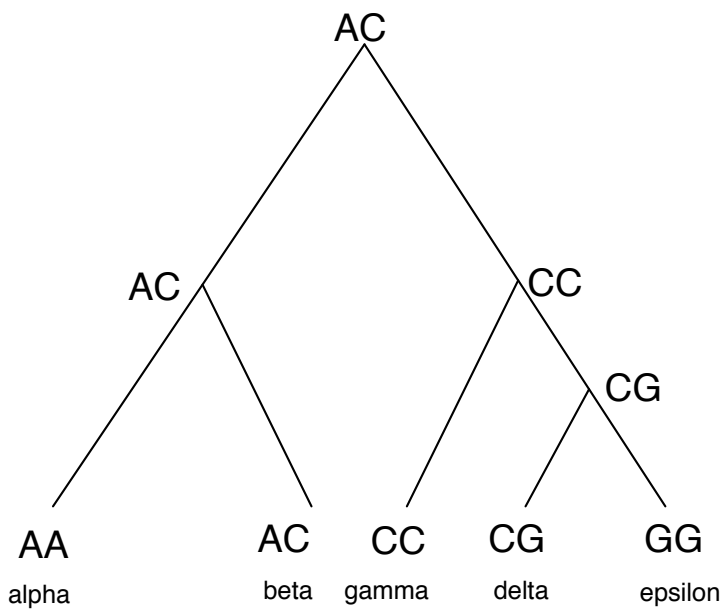
Why? We have to find a labeling of the inner nodes s.t. for both characters C_1 and C_2, each state induces a subtree.

Perfect Phylogeny

Definition

A tree T is called a perfect phylogeny (PP) if all characters are compatible with T.

Example



Our first tree for the vehicles was also a PP.

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.
- Deciding in general whether a PP exists is NP-hard.

Perfect Phylogeny

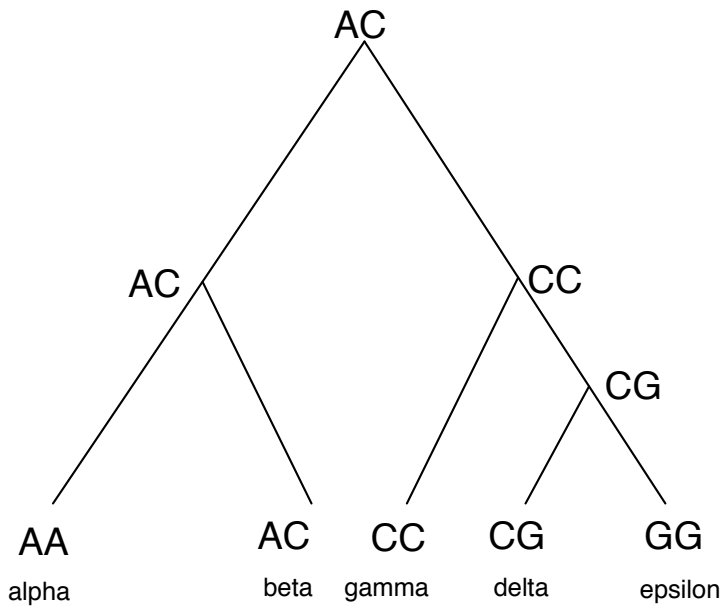
- Ideally, we would like to find a PP for our input data.
- Deciding in general whether a PP exists is NP-hard.
- This is not really a problem, since most of the time, no PP exists anyway. Why: due to homoplasies; because our input data has errors; our evolutionary model may has errors; and, and, and ...

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.
- Deciding in general whether a PP exists is NP-hard.
- This is not really a problem, since most of the time, no PP exists anyway. Why: due to homoplasies; because our input data has errors; our evolutionary model may has errors; and, and, and ...
- Therefore we usually want to find a best possible tree.

Parsimony

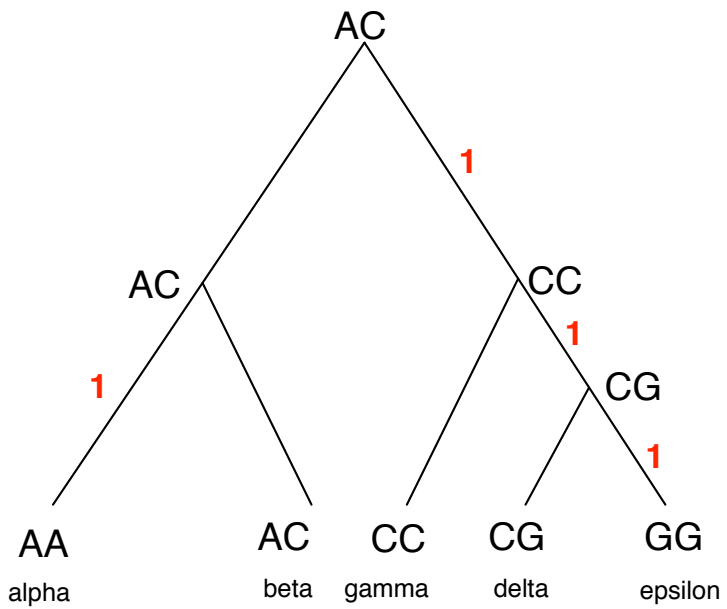
What is a best possible tree?



Why is this tree "perfect"?

Parsimony

What is a best possible tree?



Why is this tree "perfect"?

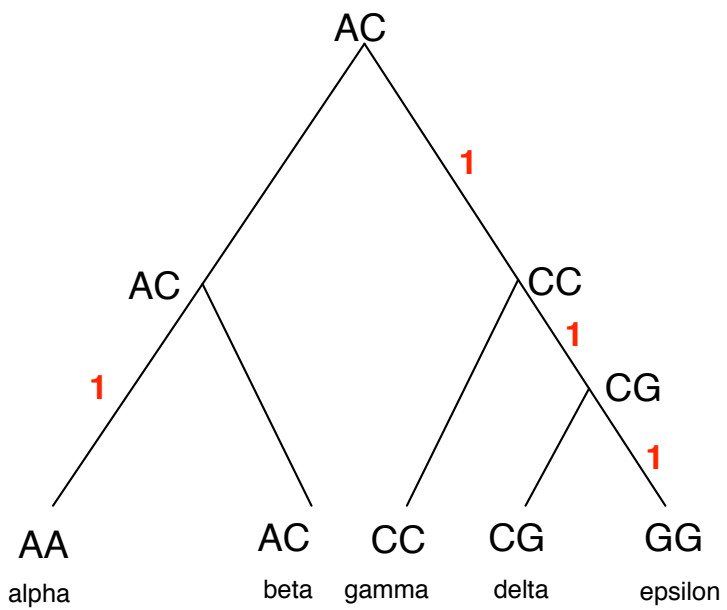
Because it has few changes of states!

In red, we marked the edges where there are state changes (an evolutionary event happened), and how many (in this case, always 1).

Parsimony

Definition

The parsimony cost of a phylogenetic tree with labeled inner nodes is the number of state changes along the edges (i.e. the sum of the edge costs, where the cost of an edge = number of characters whose state differs between child and parent).

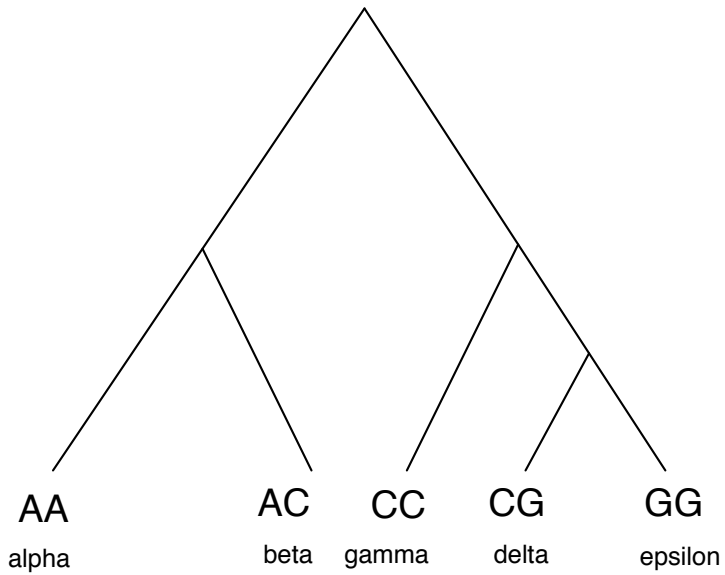


The parsimony cost of this labeled tree is 4.

Parsimony

Definition

The parsimony cost of a phylogenetic tree (without labels on the inner nodes) is the minimum of the parsimony cost over all possible labelings of the inner nodes.



The parsimony cost of this tree is 4, because the best labeling has cost 4.

Parsimony

Phylogenetic Reconstruction with Character Data

Given a character-state matrix M, our goal is to find a phylogenetic tree which minimizes the parsimony cost.

Given the character-state matrix M, we split the problem in two sub-problems:

1. Small Parsimony: Given a phylogenetic tree, find its parsimony cost, i.e. find a most parsimonious labeling of the inner nodes. This problem can be solved efficiently.
2. Large Parsimony or Maximum Parsimony: Find a tree with minimum parsimony cost. This problem is NP-hard.

Small Parsimony

Small Parsimony Problem

Given: a phylogenetic tree T with character-states at the nodes.
Find: a labeling of the inner nodes with states with minimum parsimony cost.

Algorithm

This problem can be solved using Fitch' algorithm, which runs in time $O(nmr)$, where $n =$ number of species, $m =$ number of characters, and $r =$ maximum number of states over all characters.

Maximum Parsimony

Maximum Parsimony Problem

The maximum parsimony problem is, given a character-state matrix, find a phylogenetic tree with lowest parsimony cost (= a "most parsimonious tree").

- When a PP exists, then it is also the most parsimonious tree.
- In general, this problem is NP-hard.

Algorithms for Maximum Parsimony

- Since problem NP-hard, we cannot hope to find an algorithm that solves it efficiently.
- We have seen two algorithms for this problem:
  1. Greedy Sequential Addition Algorithm - heuristic algorithm: guaranteed polynomial running time but no guarantee on the quality of the solution
  2. Branch-and-Bound for Parsimony - runtime heuristic: guarantee on exact solution, but no guarantee on the running time (may or may not be fast)

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- Usually, no PP exists, therefore in general ...

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- Usually, no PP exists, therefore in general ...
- We are looking for a most parsimonious tree (a tree with lowest parsimony cost).

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- Usually, no PP exists, therefore in general ...
- We are looking for a most parsimonious tree (a tree with lowest parsimony cost).
- The parsimony cost is defined as the minimum number of the state changes on the edges over all possible labelings of the inner nodes.

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- Usually, no PP exists, therefore in general ...
- We are looking for a most parsimonious tree (a tree with lowest parsimony cost).
- The parsimony cost is defined as the minimum number of the state changes on the edges over all possible labelings of the inner nodes.
- Recall: There are super-exponentially many trees on n taxa (both rooted and unrooted), so we cannot try them all.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.
- Small Parsimony can be solved efficienly, e.g. by Fitch' algorithm.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.
- Small Parsimony can be solved efficienly, e.g. by Fitch' algorithm.
- Maximum Parsimony is NP-hard, so probably no efficient algorithms exist.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.
- Small Parsimony can be solved efficienly, e.g. by Fitch' algorithm.
- Maximum Parsimony is NP-hard, so probably no efficient algorithms exist.
- We saw two algorithms for Maximum Parsimony: one heuristic (Greedy Seq. Add. Alg.) and one exact algorithm which is a runtime heuristic algorithm (Branch-and-Bound for Parsimony).