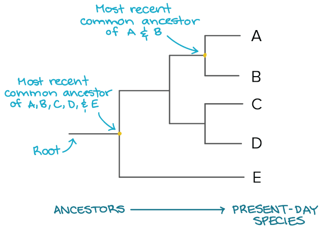
Phylogenetics



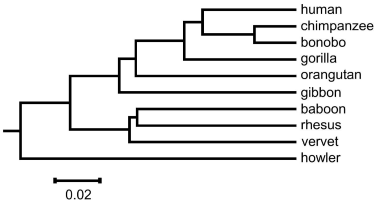
Most recent common ancestor of A & B
A
B
C
D
E
Most recent common ancestor of A, B, C, D, & E
Root
ANCESTORS
PRESENT-DAY SPECIES

Taxonomy:	the science of classifying organisms
Phylogenetics:	describes the evolutionary relationship between species
Speciation:	A population of organisms becomes separated. Over time, these evolve into separate species that do not cross-breed.

Robert Bear via Khan Academy

UPGMA

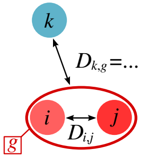
Unweighted Pair Group Method with Arithmetic Mean



$$\frac{1}{|\mathbb{A}| \cdot |\mathbb{B}|} \sum_{x \in \mathbb{A}} \sum_{y \in \mathbb{B}} d(x, y)$$

- average linkage: mean distance between elements of each group
- generates ***rooted*** trees
- generates ***ultrametric*** trees:
  - distances from the root to every branch tip are equal
- $O(n^3)$ unoptimized

UPGMA algorithm



Consider a distance matrix D, and n groups containing one item/leaf each:

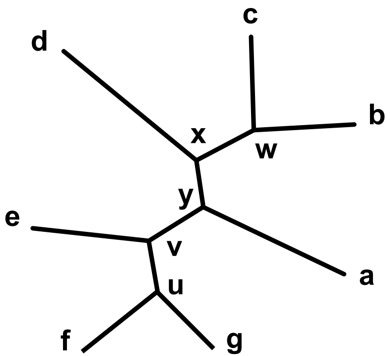
1. Choose the i and j with the smallest D_{ij}
2. Create a new group ij
3. Connect i and j to a new node in the tree that correspond to the new group
4. Set the branch length to $\frac{D_{ij}}{2}$ (ultrametric)
5. Calculate the distance between the group and all existing groups (n_i = number of elements):

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

6. Replace the i and j columns with the new group
7. If there is only one item left stop, otherwise go to 1

Thibaut Jombart, Introduction to phylogenetics

Neighbour joining



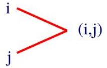
Tomfy via WikiMedia Commons

- 🌲 most widely-used distance based method for phylogenetic reconstruction
- 🌲 trees are unrooted
- 🌲 does not assume a molecular clock
- 🌲 does not produce ultrametric trees

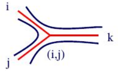
- 🌲 UPGMA: constructs a larger cluster C by merging two nearest clusters A and B
- 🌲 neighbour joining: distance from A and B to other clusters should be as large as possible
    - look for nodes that are ***close to each other*** and ***far from everything else***
        - subtract the averaged distances to all other leaves
        - compensate for long edges
- 🌲 $O(n^3)$ unoptimized

Neighbour joining algorithm

$$u_i = \sum_{j:j\neq i}^{n} \frac{D_{ij}}{n-2}$$



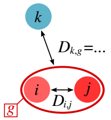
$$v_i = \frac{1}{2}D_{ij} + \frac{1}{2}(u_i - u_j)$$
$$v_j = \frac{1}{2}D_{ij} + \frac{1}{2}(u_j - u_i)$$



$$D_{(ij)k} = \frac{D_{ik} + D_{jk} - D_{ij}}{2}$$

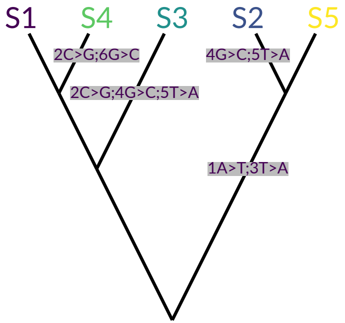
Consider a distance matrix D:

1. Calculate the “average” distance to other nodes/clusters for each leaf
2. Choose i and j to minimize $D_{ij} - u_i - u_j$
   (Nodes that are close to each other, and far from everything else)
3. Join i and j to create a new node (i, j) and calculate the new branch lengths
4. Compute distance between leaves and the new group
5. Replace the i and j leaves with the new node (i, j)
6. Continue until two nodes remain



Evaluating trees

S1	ACTGTG
S2	TCACAG
S3	AGTCAG
S4	AGTGTC
S5	TCAGTG



- 🌲 Label internal nodes, *e.g.* Hamming distance (number of changes)

A candidate tree:

- 🌲 $L(T) = 9$ changes
- 🌲 How can we make this tree more parsimonious?

A better tree:

- 🌲 $L(T) = 8$ changes

An equally good tree:

- 🌲 $L(T) = 8$ changes

Small parsimony: computational problems

Small parsimony

- given a tree T, calculate $L(T)$
- for small trees we can calculate by hand
- impractical for larger trees with many leaves

Algorithmically:

- iterate over positions in the alignment
- at each position, find internal nodes that require a mutation to explain the data found in the children

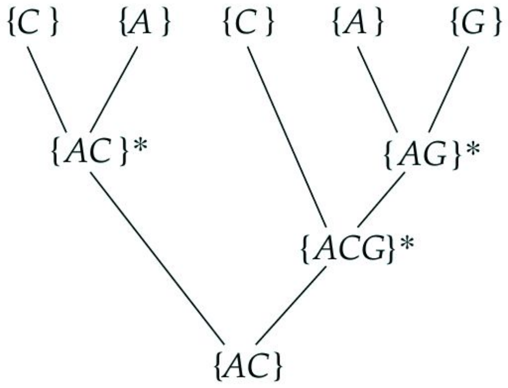
Fitch algorithm

- dynamic programming
- compute parsimony score for a column of the sequence alignment
- repeat the process for each column
- substitutions have the same cost

Sankoff algorithm

- dynamic programming
- allows us to calculate the cost of changes in a given tree

Fitch algorithm: example 1



For each leaf v:

$$S_v = \{v_c\}$$

For any internal node v:

$$S_v = \{S_u \cap S_w \text{ if } S_u \cap S_w \neq 0 \quad S_u \cup S_w \text{ otherwise}\}$$

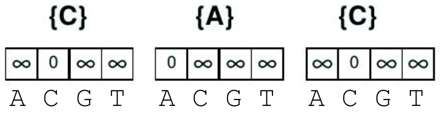
- $L(T) = 3$
- Repeat the process for each column
- Changes have the same cost

Sankoff algorithm

Count the smallest number of possible (weighted) changes needed on a given tree

Cost for the leaves

- 0 for the observed letter
- infinity otherwise



Calculate costs for internal nodes

- for each node, compute the minimum cost S_a for each character i to occur at that node

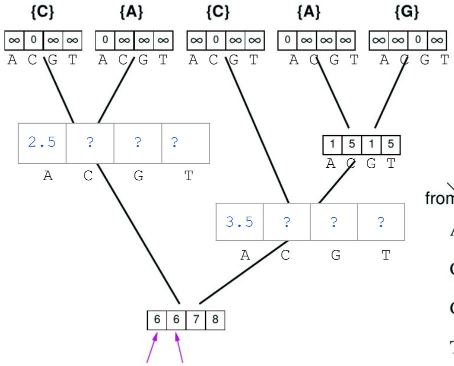
$$S_a(i) = \min[c_{ij} + S_L(j)] + \min[c_{ik} + S_R(k)]$$

- L and R are left and right children nodes
- c_{ij} is the cost for changing from state i to j

Use a cost matrix

- we used a fixed cost for Fitch's algorithm
- for Sankoff, we use a cost matrix

Sankoff example



cost matrix:

from \ to	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

S_i?

$$
\begin{aligned}
i &= A \\
j &= A, C, G, T \\
k &= A, C, G, T
\end{aligned}
$$

$$
\begin{aligned}
S_a(i) = \min[c_{ij} + S_L(j)] \\
+ \min[c_{ik} + S_R(k)]
\end{aligned}
$$

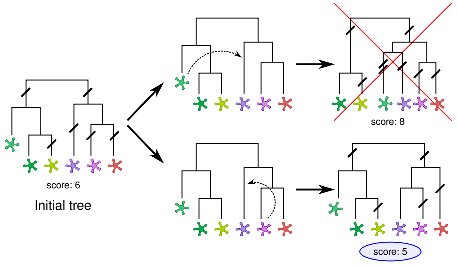
- Limitation: implicitly assumes that rate of change along branches is similar

Parsimony:· computational problems

- 🌲 we know how to score a tree for parsimony (***small parsimony***)
- 🌲 how can we find the best tree? (***large/maximum parsimony***)
  - optimization problem
- 🌲 enumerating trees is unfeasible
  - $O(n!)$: factorial growth with the number of leaves (*e.g.* sequences)
  - not feasible to score all of them
  - heuristic approach
  - tree searching methods

Sequences	Unrooted trees
3	1
4	3
5	15
10	$>$ 2 000 000

Exploring tree space



Exact methods

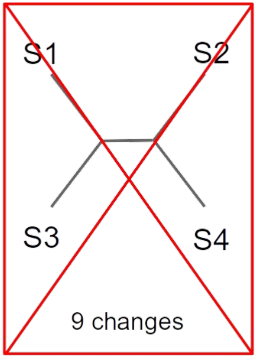
- 🌲 exhaustive search
- 🌲 branch and bound algorithms
  - reduce search space
  - eliminate candidate solutions that will not reach an optimal solution

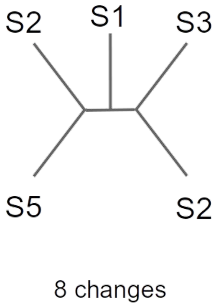
Heuristics

- 🌲 sequential/stepwise addition
- 🌲 branch swapping methods
  - we can rearrange trees by breaking and reattaching branches
  - efficient to re-score because Sankoff algorithm is recursive

Thibaut Jombart, Introduction to phylogenetics

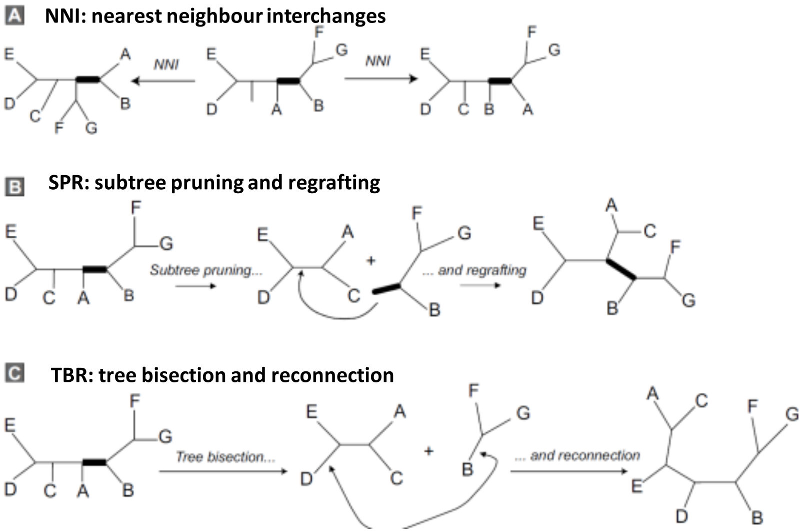
Branch and bound





- this was our best five-tip tree $L(T) = 8$ (8 changes)
- once we find that, we don't have to look at trees based on the four-tip tree with 9 changes
- reduces search space
- always finds the optimal

Heuristic: branch Swapping

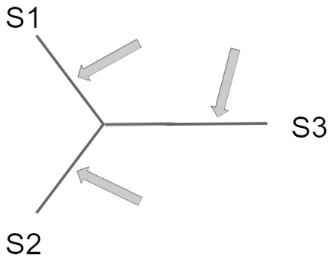

A
NNI: nearest neighbour interchanges
NNI
NNI
E
A
D
C
F
G
B
B
SPR: subtree pruning and regrafting
Subtree pruning...
... and regrafting
+
C
TBR: tree bisection and reconnection
Tree bisection...
... and reconnection
+

Kai Müller via WWU Münster

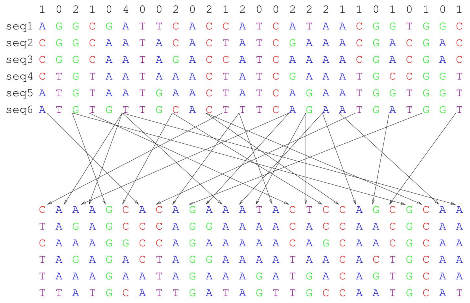
Heuristic: sequential addition

- 🌲 assume the tree is unrooted for simplicity
- 🌲 we can add S4 in three places

S1	ACTGTG
S2	TCACAG
S3	AGTCAG
S4	AGTGTC
S5	TCAGTG



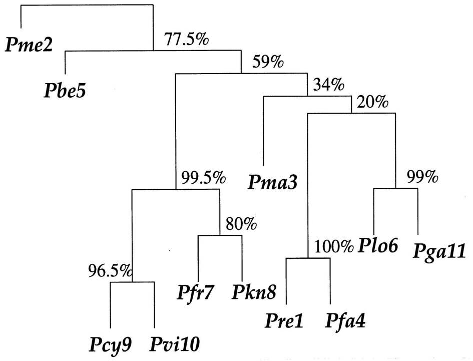
Bootstrapping



1 0 2 1 0 4 0 0 2 0 2 1 2 0 0 2 2 2 1 1 0 1 0 1 0 1
seq1 A G G C G A T T C A C C A T C A T A A C G G T G G C
seq2 C G G C A A T A C A C T A T C G A A A C G A C G A C
seq3 C G G C A A T A G A C C A T C A A A A C G A C G A C
seq4 C T G T A A T A A A C T A T C G A A A T G C C G G T
seq5 A T G T A A T G A A C T A T C A G A A T G G T G G T
seq6 A T G T G T T G C A C T T T C A G A A T G A T G G T
C A A A G C A C A G A A A T A C T C C A G C G C A A
T A G A G C C C A G G A A A A C A C C A A C G C A A
C A A A G G C C A G A A A A A C A G C A A C G C A A
T A G A G A C T A G G A A A A T A A C A C T G C A A
T A A A G A A T A G A A A G A T G A C A G T G C A A
T T A T G C A T T G A T A G T T G C C A A T G C A T

- 🌲 **general approach:** assess accuracy of an estimator using simulated data
- 🌲 re-sample columns in an alignment of sequences to create new alignments
- 🌲 re-apply the same phylogeny reconstruction method

Bootstrapping



- repeat bootstrapping (at least 100 times)
- count occurrence of nodes in bootstrap trees
- if we see the branching point often, it is more reliable
- ***rule of thumb***: accept bootstrap values from 90–100 %