

# Parsimony

## Review

- Three types of tree construction
  - Distance: find tree that accounts for observed distances
  - Parsimony: find tree that requires minimum # of changes
  - Maximum likelihood: tree that maximizes likelihood of data

## Review

- UPGMA
  - This method requires ultrametric property: for three distances, two are equal and third is  $\leq$  the first two
  - Relies on a molecular clock
- Neighbor joining
  - This method requires additive property: distances between two nodes is sum of their edges
  - Often produces a good tree even with non-additive data

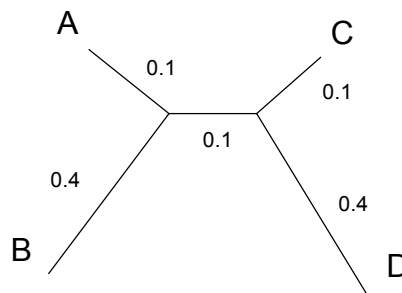
## Four-point condition

- Pairwise distances are additive if and only if for every set of four leaves  $i, j, k, l$ , two of the following three sums are equal and larger than the third:
  - $D_{ij} + D_{kl}$
  - $D_{ik} + D_{jl}$
  - $D_{il} + D_{jk}$

## Idea behind NJ algorithm

- Start with a center star phylogeny where everyone is connected to a single node.
- Choose two sequences to merge based on the mathematically optimal topology under an additive scoring scheme.
- Note this tree starts out unrooted and remains so without an outgroup.

## Example



Neighbor-joining will find the correct tree here

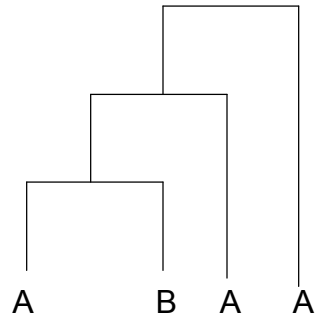
## Intro

- Parsimony is a simple and fast approach, making it popular
- Two distinct subproblems:
  - Find the history of mutations (easy)
  - Given an alignment, infer tree (hard)

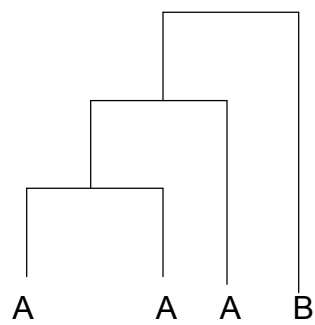
## Parsimony

- Consider the following strings:
  - TTC, TTT, CCT, TCT
- Further, suppose we know the root of the tree is TTT
- What can we do?

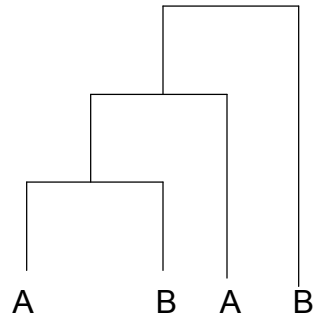
## Illustration



## Illustration



## Illustration



## Parsimony

- Intuitively, we want to measure changes along edges of trees.
- Similar to Okham's razor: find simplest explanation that works

## Overview

- Input:
  - character-based data such as an alignment
- Output:
  - tree that requires minimal number of changes
- Goal:
  - Find right tree topology (how things branch) instead of the actual lengths of edges
  - Hard part is finding optimal topology

## Overview

- As mentioned in the intro, there are two tasks we need to do:
  - Find minimum number of changes needed to explain the data for a given tree
  - Search a bunch of trees using the above

## Definition

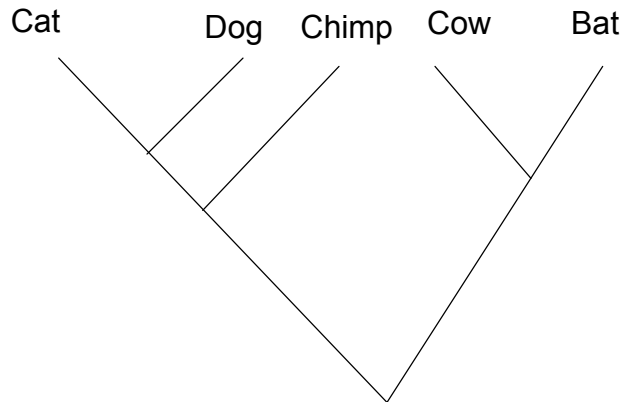
- We want to compute number of times the value of some character changes along edges in a tree.
- Formally:

$$S(t) = \sum_{(v,u) \in E(T)} |\{j : v_j \neq u_j\}|$$

## In-class example

	1	2	3	4	5	6
Cat	A	T	A	C	A	G
Dog	A	T	A	G	T	G
Chimp	A	C	A	C	A	G
Cow	T	C	G	G	T	A
Bat	T	C	G	C	A	G

## Tree



## Fitch's algorithm

- Published in 1971
- Relies on the following assumptions:
  - Any state can convert to any other state
  - Positions in an input are independent
  - Cost is uniform, i.e.,  $A \rightarrow T = G \rightarrow T$

## Fitch's algorithm

- We will do two traversals of the tree
- Bottom - up (leaves to root)
  - Determine set of possible states for each node
- Top - down (root to leaves)
  - Pick the ancestral state for each node from the set of possibilities

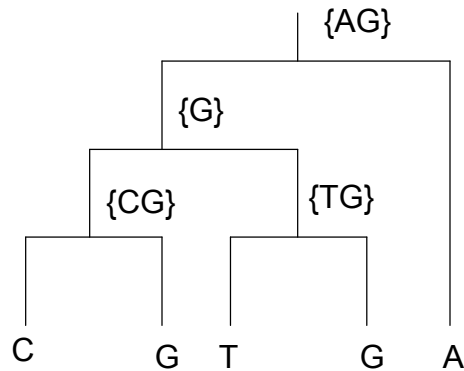
## Step 1

- Perform a post-order traversal of the tree
- Compute:

$$R_i = \begin{cases} R_j \cup R_k, & \text{if } R_j \cap R_k = \emptyset \\ R_j \cap R_k & \text{otherwise} \end{cases}$$

- # union operations = # of changes

## Example

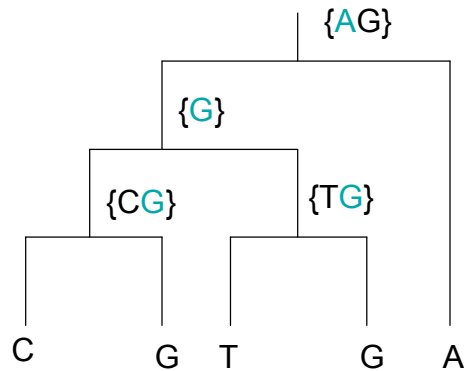


## Step 2

- Now do a preorder traversal of tree
- Select state  $r_j$  for an internal node  $j$  with a parent node  $i$  as follows:

$$r_j = \left\{ \begin{array}{l} r_i, \text{if } r_i \in R_j \\ \text{any} \in R_j, \text{otherwise} \end{array} \right\}$$

## Example



## Weighted Parsimony

- Sankoff & Cedergren (1983)
- Rather than assume all state changes are equally likely, use different costs for different changes
- We'll need to propagate costs up during first step of approach

## Details

- Lets define a cost of assigning a character  $c$  to a node  $i$
- This is defined to be 0 for a leaf if the leaf is that character, infinity otherwise

## Internal nodes

- Suppose we are looking at a node  $i$  with children  $x$  and  $y$ . This is  $R_i$
- Cost is defined as:
  - $\text{Min} (R_x(b) + S(a,b)) + \text{min} (R_y(b) + S(a,b))$

## Step 2 for weighted case

- We will still do a pre-order traversal
- At root, select minimum cost character
- At leaves, select character that resulted in minimum cost value at parent

## Methods for exploring tree space

- Consider any single internal edge in a tree
- There are 3 ways the four subtrees can be grouped:
  - AB - CD; AC - BD; AD - BC
- Using this rule to look at trees is called nearest neighbor interchange.

## Exact method

- There is a branch and bound approach that can be used to calculate the best tree more efficiently.
- In short, if a tree you build is worse than a previously discovered tree, you stop
- Keep track of all partial trees such that you can reuse information as much as possible

## Probabilistic methods

- Input:
  - Given an alignment and a mutation model (e.g., Jukes-Cantor)
- Problem:
  - Compute the likelihood of a tree as a product of the individual likelihoods from the alignment
  - Assumes columns are independent

## Conclusions

- Many algorithms exist for these problems
- Parsimony generally does pretty well for most applications
- Likelihood, however, is becoming popular due to increased computational power.