Parsimony

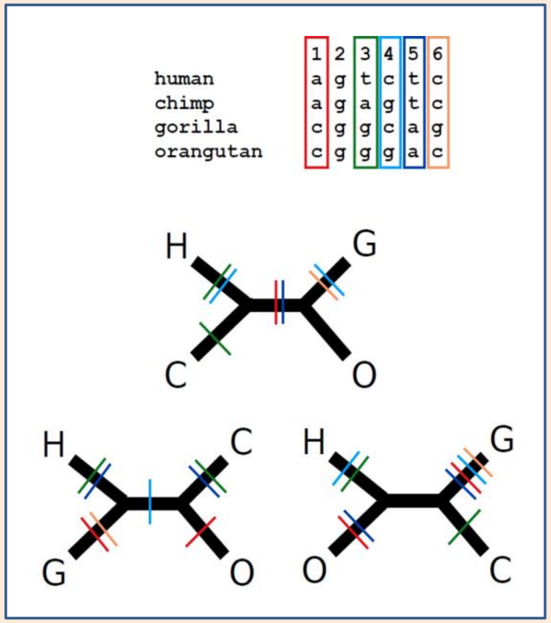
Small Parsimony

Genome 559: Introduction to Statistical and Computational Genomics

Elhanan Borenstein

A quick review

- The parsimony principle:
  - Find the tree that requires the fewest evolutionary changes!
- A fundamentally different method:
  - Search rather than reconstruct
- Parsimony algorithm
  1. Construct all possible trees
  2. For each site in the alignment and for each tree count the minimal number of changes required
  3. Add sites to obtain the total number of changes required for each tree
  4. Pick the tree with the lowest score



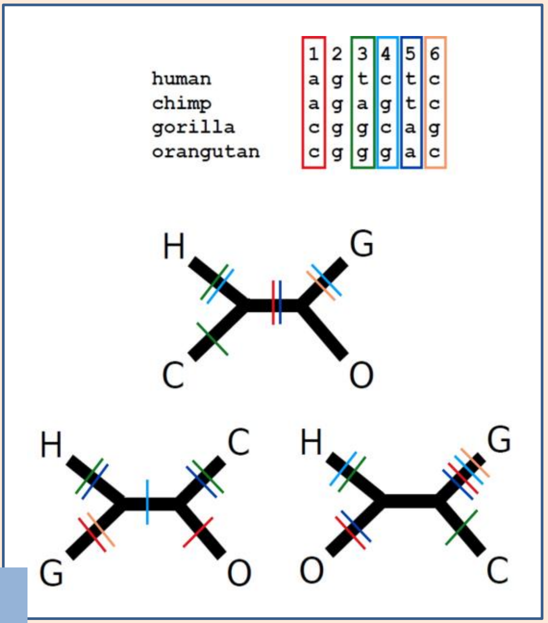
	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c

A quick review

- The parsimony principle:
  - Find the tree that requires the fewest evolutionary changes!
- A fundamentally different method:
  - Search rather than reconstruct
- Parsimony algorithm
  1. Construct all possible trees
  2. For each site in the alignment and for each tree count the minimal number of changes required
  3. Add sites to obtain the total number of changes required for each tree
  4. Pick the tree with the lowest score

Too many!

The small parsimony problem



Large vs. Small Parsimony

- We divided the problem of finding the most parsimonious tree into two sub-problems:
  - **Large parsimony:** Find the topology which gives best score
  - **Small parsimony**: Given a tree topology and the state in all the tips, find the minimal number of changes required

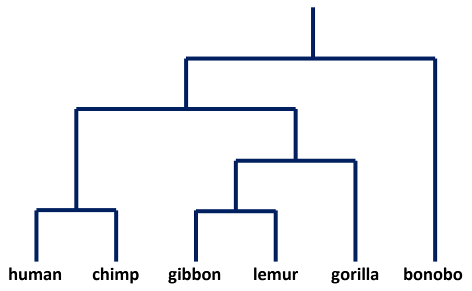
- Divide and conquer. Think functions !!
- Large parsimony is "NP-hard"
- Small parsimony can be solved quickly using Fitch's algorithm

<u>Parsimony Algorithm</u>

1) *Construct all possible trees*
2) *For each site in the alignment and for each tree count the minimal number of changes required*
3) *Add all sites up to obtain the total number of changes for each tree*
4) *Pick the tree with the lowest score*

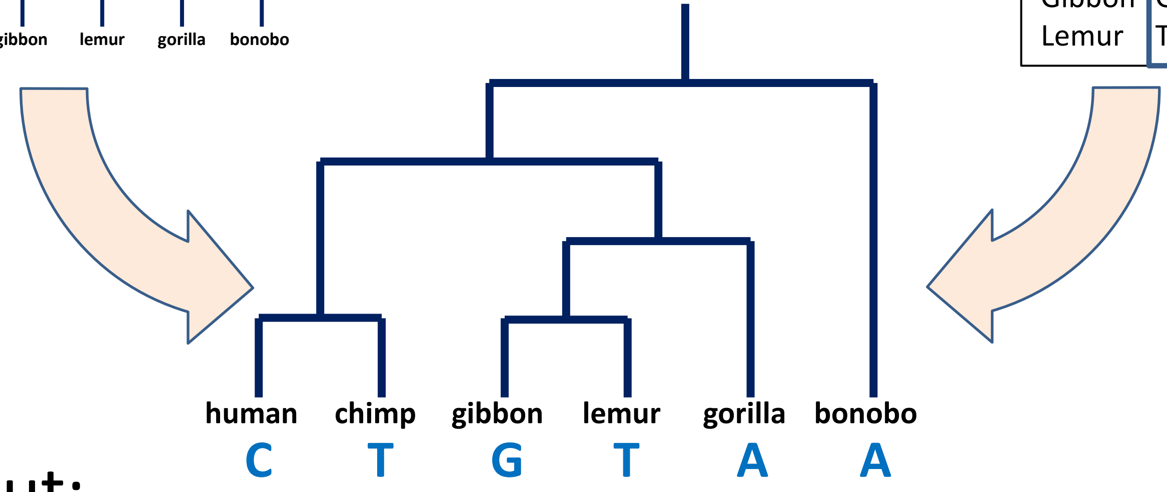
The Small Parsimony Problem

- Input:
  1. A tree topology:



  2. State assignments for all tips:

Human	C	A	C	T
Chimp	T	A	C	T
Bonobo	A	G	C	C
Gorilla	A	G	C	A
Gibbon	G	A	C	T
Lemur	T	A	G	T



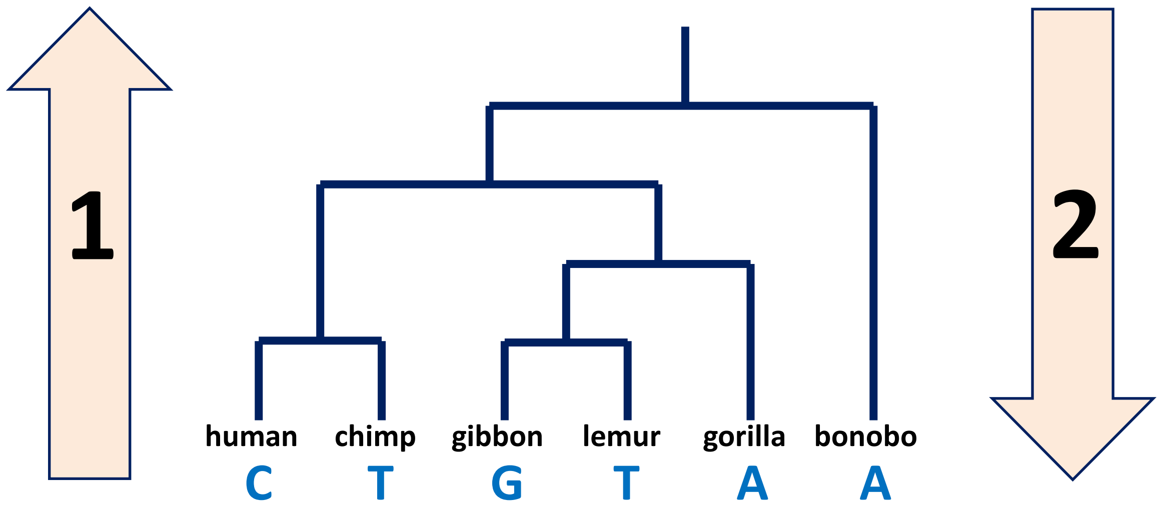
- Output:

The minimal number of changes required: ***parsimony score***

(*but in fact, we will also find the most parsimonious assignment for all internal nodes*)

Fitch's algorithm

- Execute independently for each character:
- Two phases:
  1. **Bottom-up phase**: Determine the set of possible states for each internal node
  2. **Top-down phase**: Pick a state for each internal node

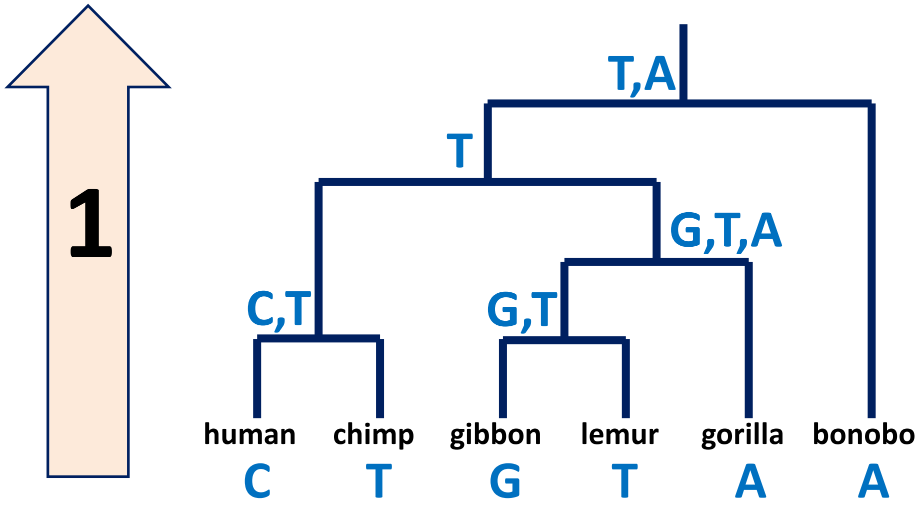


1. Fitch's algorithm: Bottom-up phase

(Determine the set of possible states for each internal node)

1. Initialization: $R_i = \{s_i\}$ for all tips
2. Traverse the tree from leaves to root ("post-order")
3. Determine R_i of internal node *i* with children *j, k:*

$$R_i = \begin{cases} if\ \ R_j \cap R_k \neq \phi \rightarrow R_j \cap R_k \\ otherwise \rightarrow R_j \cup R_k \end{cases}$$



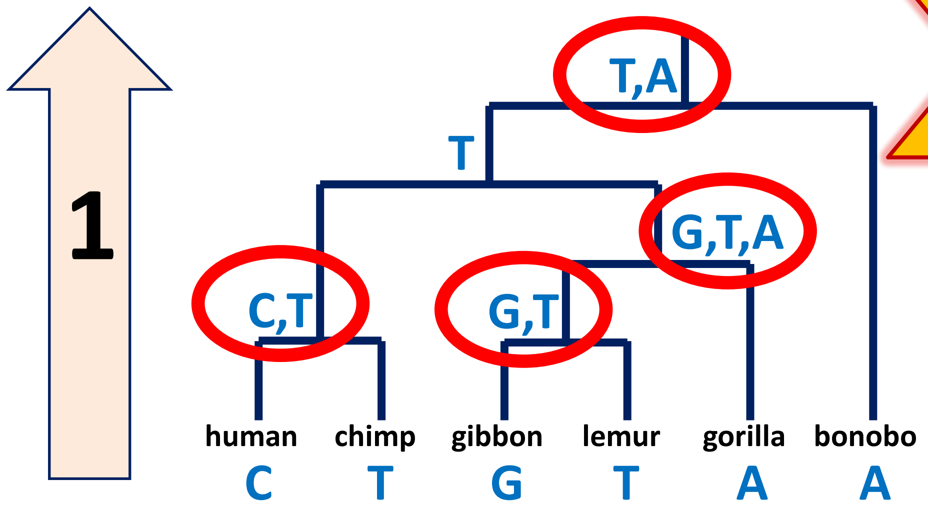
Let s_i denote the state of node *i* and R_i the set of possible states of node *i*

1. Fitch’s algorithm: Bottom-up phase

(Determine the set of possible states for each internal node)

1. Initialization: $R_i = \{s_i\}$
2. Traverse the tree from leaves to root (“post-order“)
3. Determine R_i of internal node i with children j, k:

This image cannot currently be displayed.



Parsimony-score = # union operations

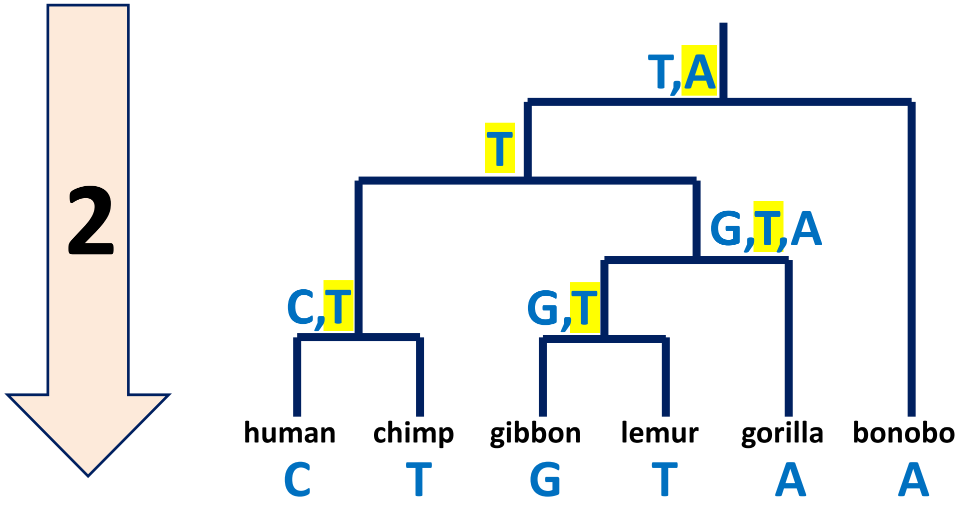
Parsimony-score = 4

2. Fitch's algorithm: Top-down phase

(Pick a state for each internal node)

1. Pick arbitrary state in R_{root} to be the state of the root ,s_{root}
2. Traverse the tree from root to leaves ("pre-order")
3. Determine s_i of internal node i with parent j:

$$s_i = \left\{ \begin{array}{l} if \quad s_j \in R_i \rightarrow s_j \\ otherwise \rightarrow arbitrary \;\; state \in R_i \end{array} \right\}$$



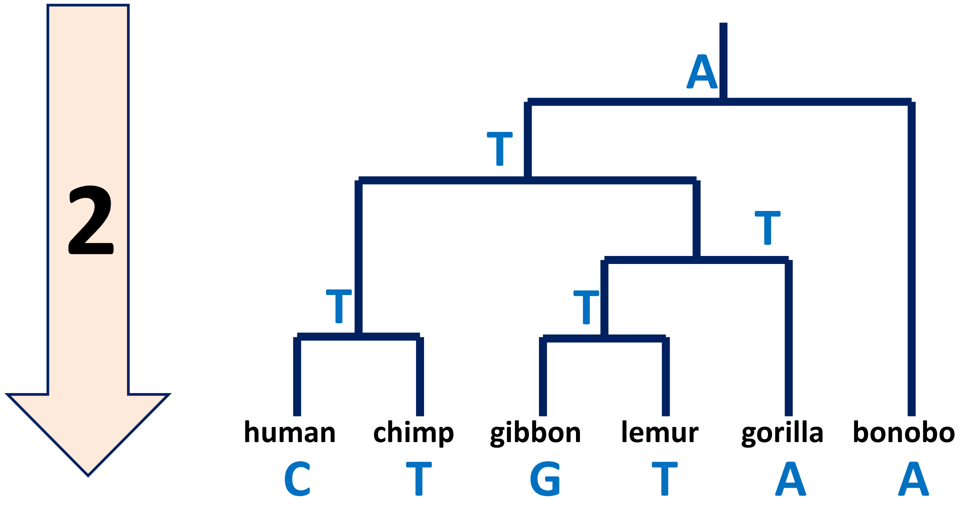
Parsimony-score = 4

2. Fitch's algorithm: Top-down phase

(Pick a state for each internal node)

1. Pick arbitrary state in R_{root} to be the state of the root ,s_{root}
2. Traverse the tree from root to leaves ("pre-order")
3. Determine s_i of internal node *i* with parent *j*:

$$\boxed{s_i = \begin{cases} if \quad s_j \in R_i \rightarrow s_j \\ otherwise \rightarrow arbitrary \;\; state \in R_i \end{cases}}$$



Parsimony-score = 4

And now back to the “big” parsimony problem …

*How do we find the most parsimonious tree amongst the **many** possible trees?*