

Parsimony I

Genome 373

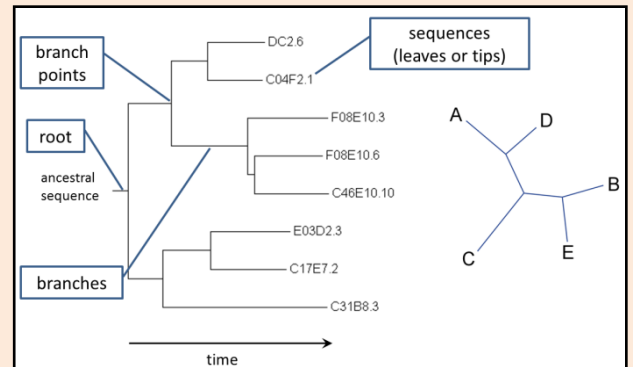
Genomic Informatics

Elhanan Borenstein

A quick review

■ Trees:

- Represent sequence relationships
- A sequence tree has a topology and branch lengths (distances)
- The number of tree topologies grows very fast!



■ Distance trees

- Compute pairwise corrected distances
- Build tree by sequential clustering algorithm (UPGMA or Neighbor-Joining).
- These algorithms don't consider all tree topologies, so they are very fast, even for large trees.

“Maximum Parsimony Algorithm”



A fundamentally different method:

Instead of reconstructing a tree,
we will search for the best tree.

“Pluralitas non est ponenda sine necessitate”

(Maximum) Parsimony Principle

- *“Pluralitas non est ponenda sine necessitate”*
(plurality should not be posited without necessity)
William of Ockham
- Occam’s Razor: Of two equivalent theories or explanations, all other things being equal, the simpler one is to be preferred.



William of Ockham
(c. 1288 – c. 1348)

- "when you hear hoof beats, think horses, not zebras"
Medical diagnosis
- The KISS principle: "Keep It Simple, Stupid!"
Kelly Johnson, Engineer
- “Make everything as simple as possible, but not simpler”
Albert Einstein

Parsimony principle for phylogenetic trees

*Find the tree that requires the
fewest evolutionary changes!*

Consider 4 species

human
chimp
gorilla
orangutan

Consider 4 species

Sequence data:

human	<u>123456</u>
chimp	agtctc
gorilla	agagtc
orangutan	cggcag
	cgggac

positions in alignment
(usually called "sites")

- The same approach would work for any discrete property that can be associated with the various species:
 - Gene content (presence/absence of each gene)
 - Morphological features (e.g., "has wings", purple or white flowers)
 - Numerical features (e.g., number of bristles)

Consider 4 species

Sequence data:

human	123456
chimp	agtctc
gorilla	agagtc
orangutan	cggcag
	cgggac

positions in alignment
(usually called "sites")

Parsimony Algorithm

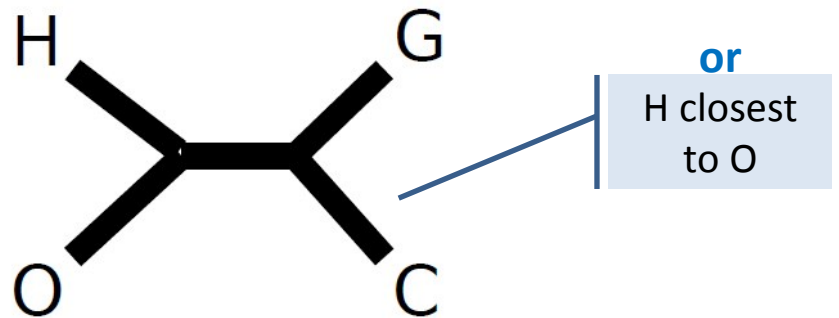
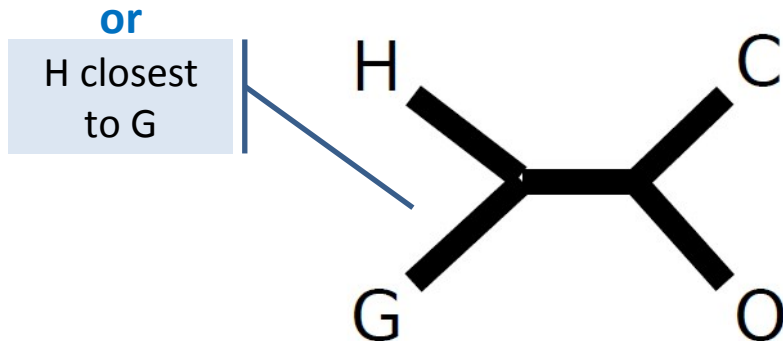
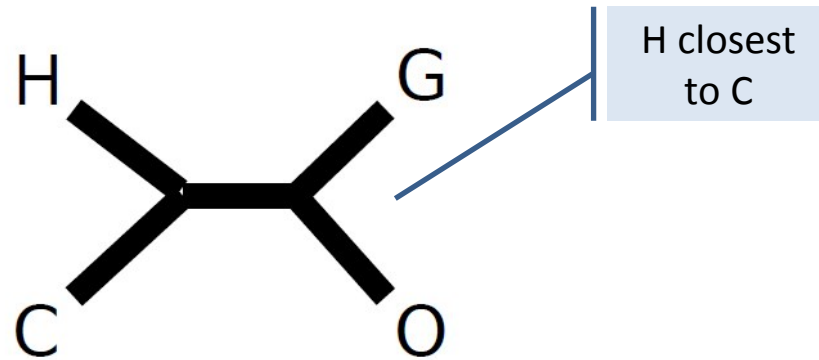
- 1) *Construct all possible trees*
- 2) ***For each site in the alignment and for each tree count the minimal number of changes required***
- 3) *Add all sites up to obtain the total number of changes for each tree*
- 4) *Pick the tree with the lowest score*

Consider 4 species

Sequence data:

	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	a	c	

All possible unrooted trees:



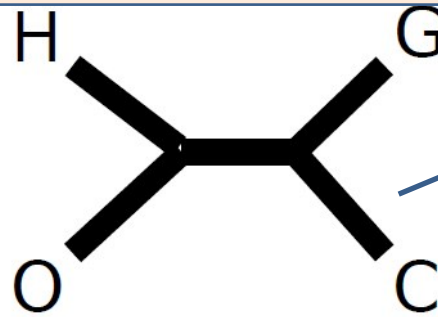
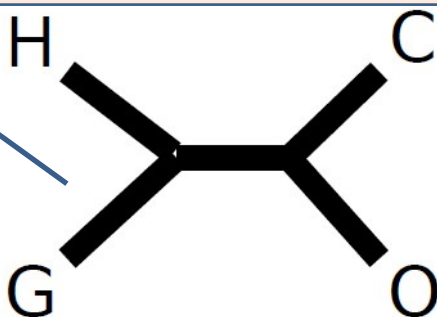
Consider 4 species

Sequence data: human 123456
agtctc

All
uni

*For each site and for each tree
count the minimal number of
changes required:*

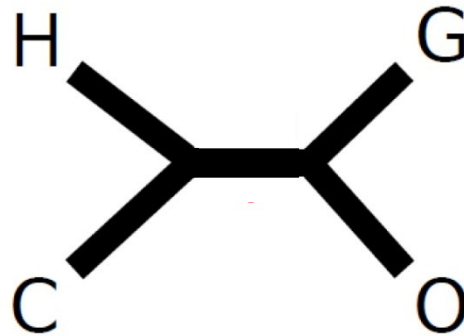
H closest
to G



or
H closest
to O

Consider site 1

	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

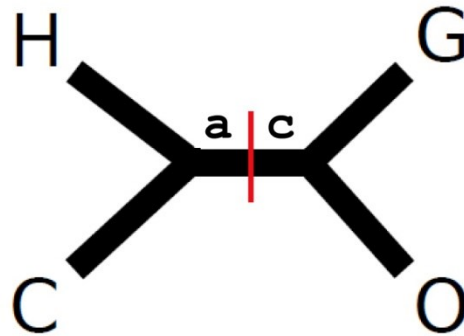


What is the minimal number of evolutionary changes that can account for the observed pattern?

(Note: This is the “small parsimony” problem)

Consider site 1

	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

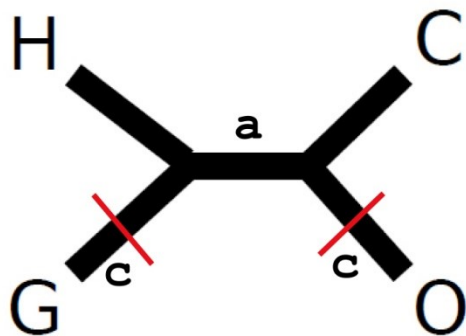
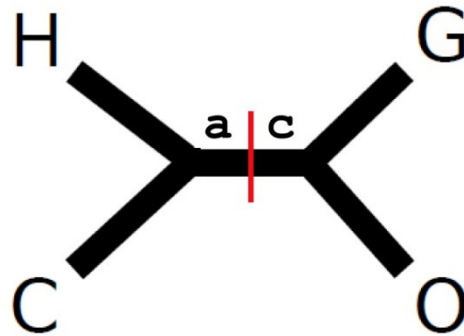


What is the minimal number of evolutionary changes that can account for the observed pattern?

(Note: This is the “small parsimony” problem)

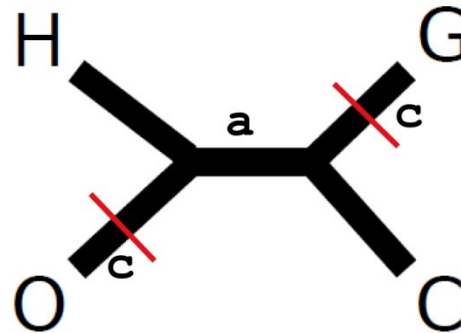
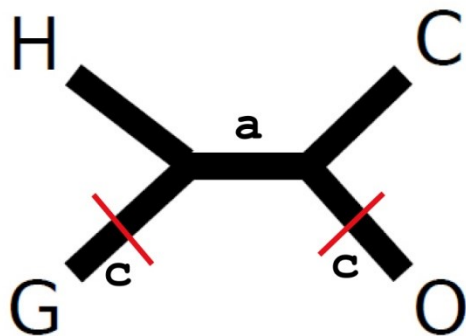
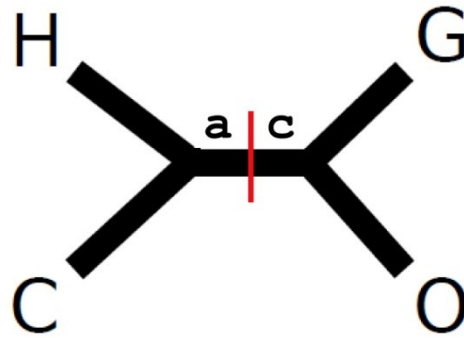
Consider site 1

	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



Consider site 1

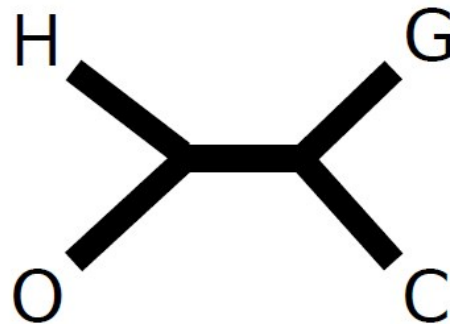
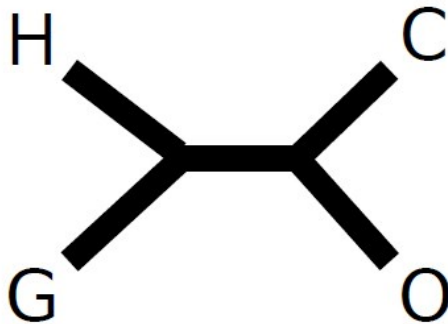
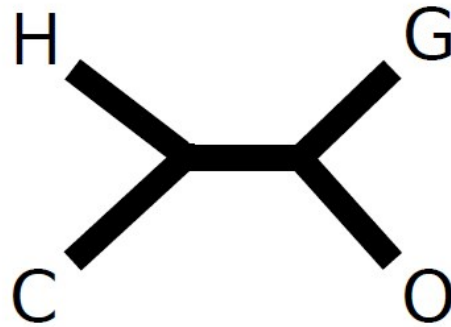
	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c



Consider site 2

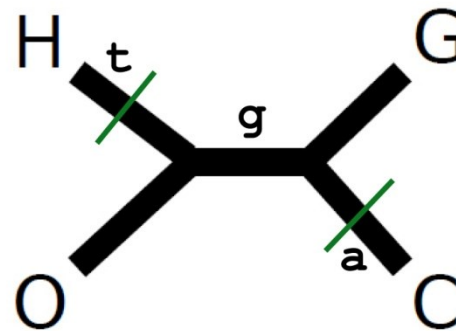
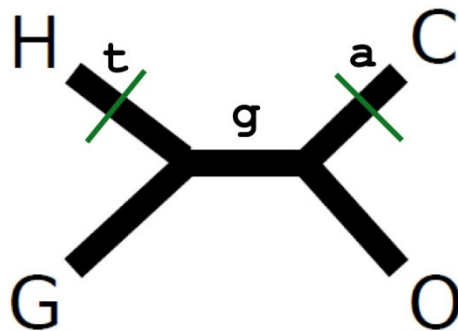
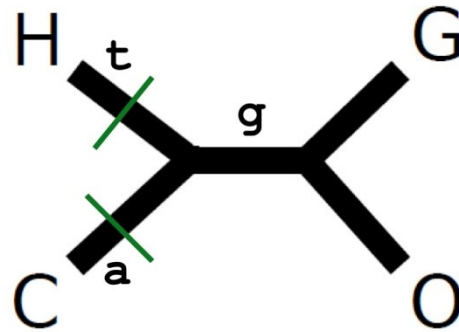
	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c

Uninformative
(no changes)



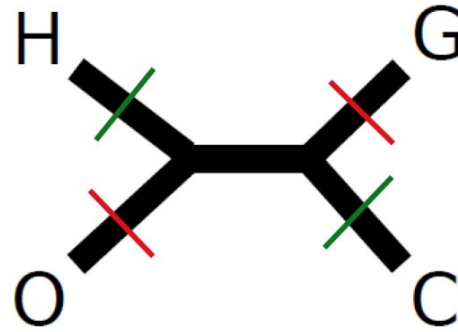
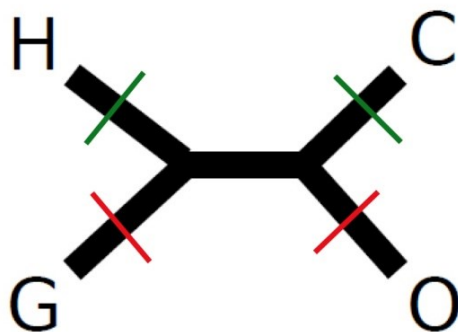
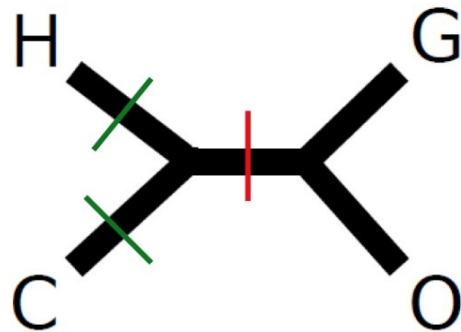
Consider site 3

	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



Put sites 1 and 3 together

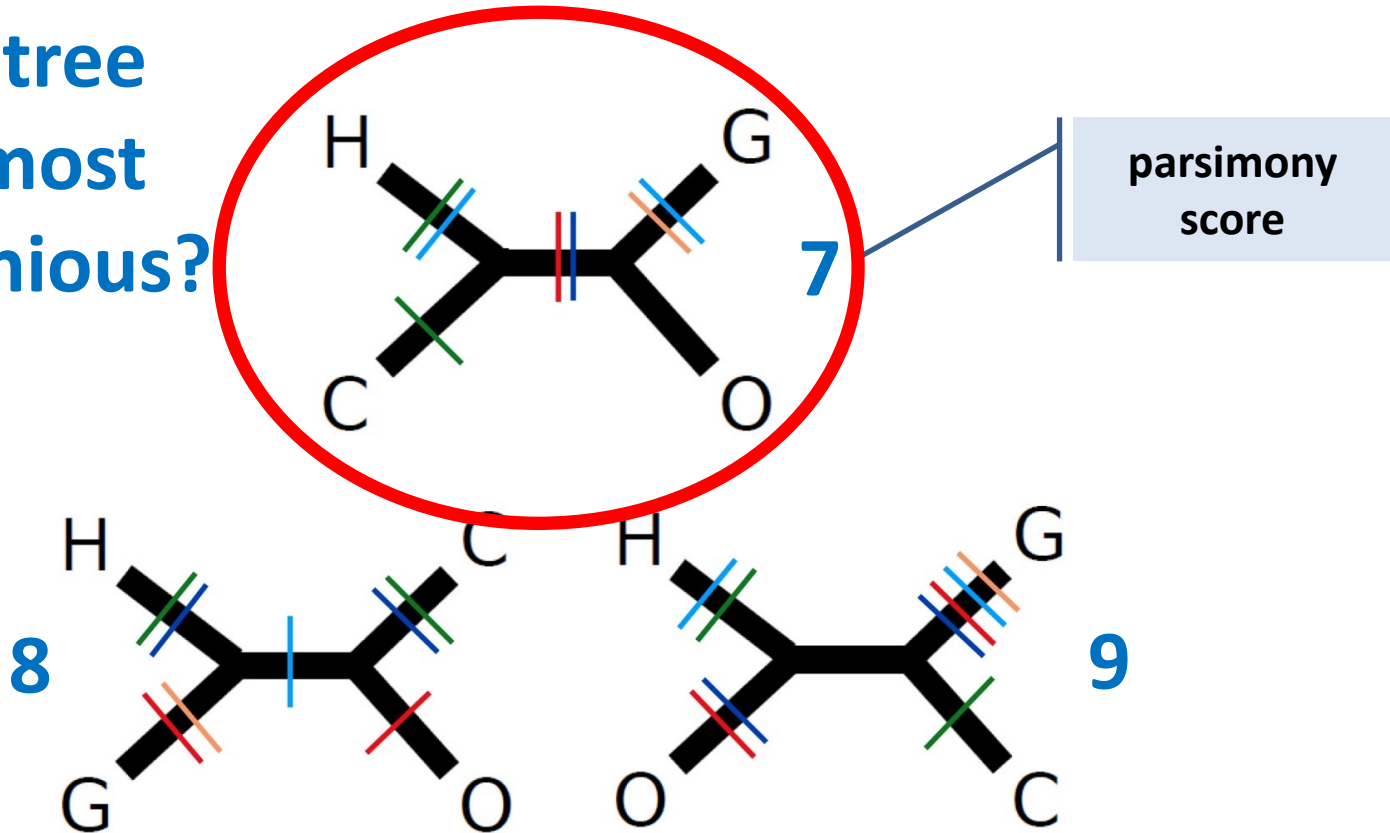
	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c



Now put all of them together

	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

Which tree is the most parsimonious?



The parsimony algorithm


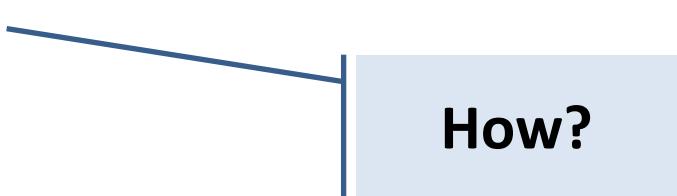
- 1) *Construct all possible trees*
- 2) *For each site in the alignment and for each tree count the minimal number of changes required*
- 3) *Add all sites up to obtain the total number of changes for each tree*
- 4) *Pick the tree with the lowest score*

The parsimony algorithm

Too many!

- 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 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 all sites up to obtain the total number of changes for each tree*
- 4) *Pick the tree with the lowest score*

Too many!

How?

The parsimony algorithm

1) *Construct all possible trees*

Too many!

Search
algorithm

2) *For each site in the alignment and for each tree count the minimal number of changes required*

How?

Fitch's algorithm

3) *Add all sites up to obtain the total number of changes for each tree*

4) *Pick the tree with the lowest score*

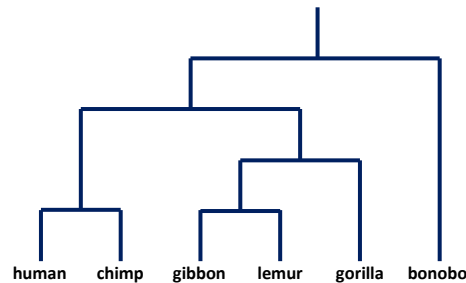
Large vs. Small Parsimony

- 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
- Large parsimony is “NP-hard”
- Small parsimony can be solved quickly using Fitch’s algorithm

The Small Parsimony Problem

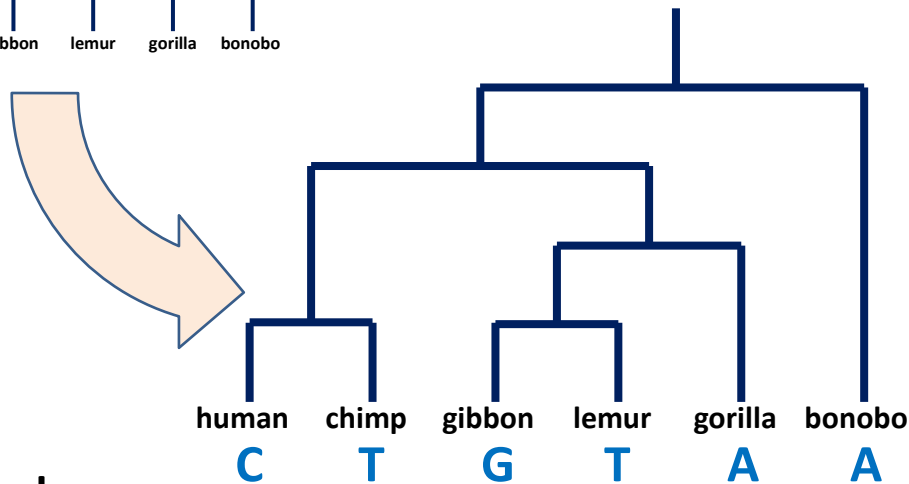
- Input:

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

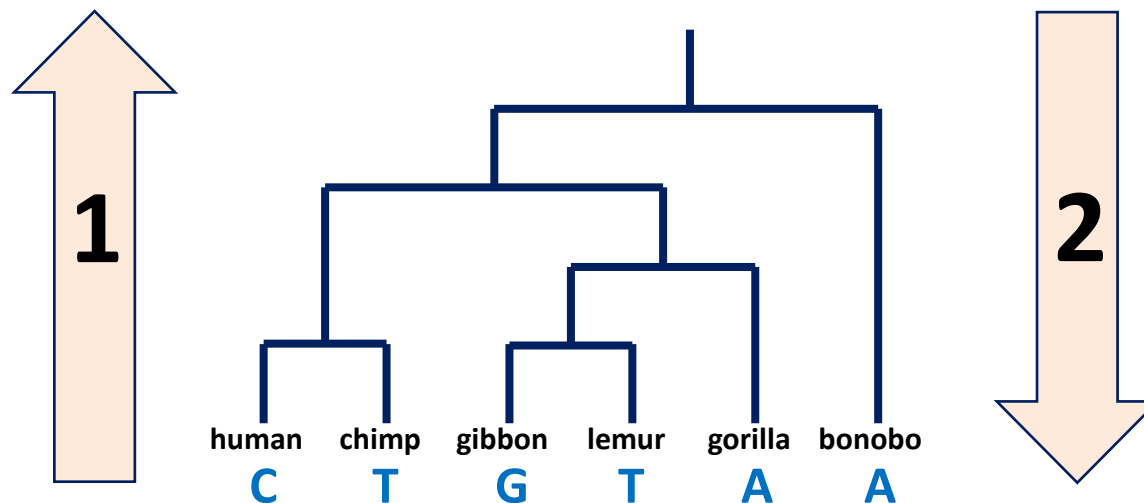


- Output:

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

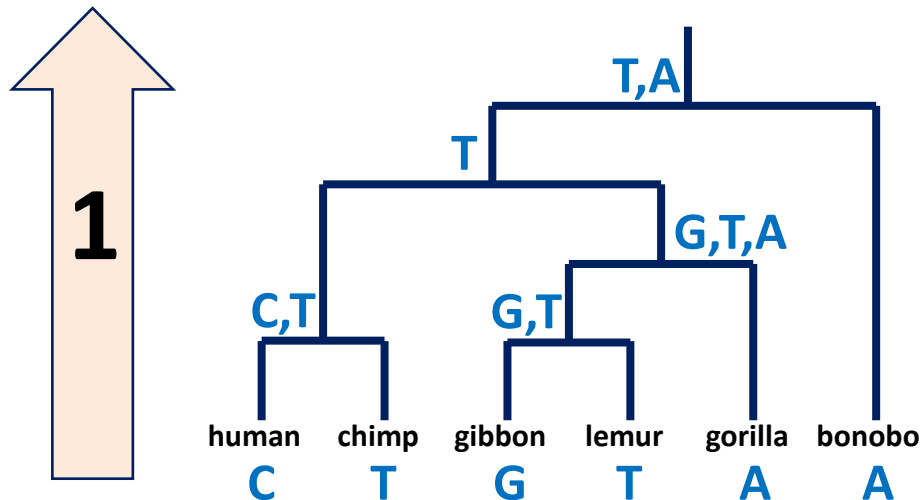


1. Fitch's algorithm: Bottom-up phase

(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 :

$$R_i = \left\{ \begin{array}{l} \text{if } R_j \cap R_k \neq \phi \rightarrow R_j \cap R_k \\ \text{otherwise} \rightarrow R_j \cup R_k \end{array} \right\}$$



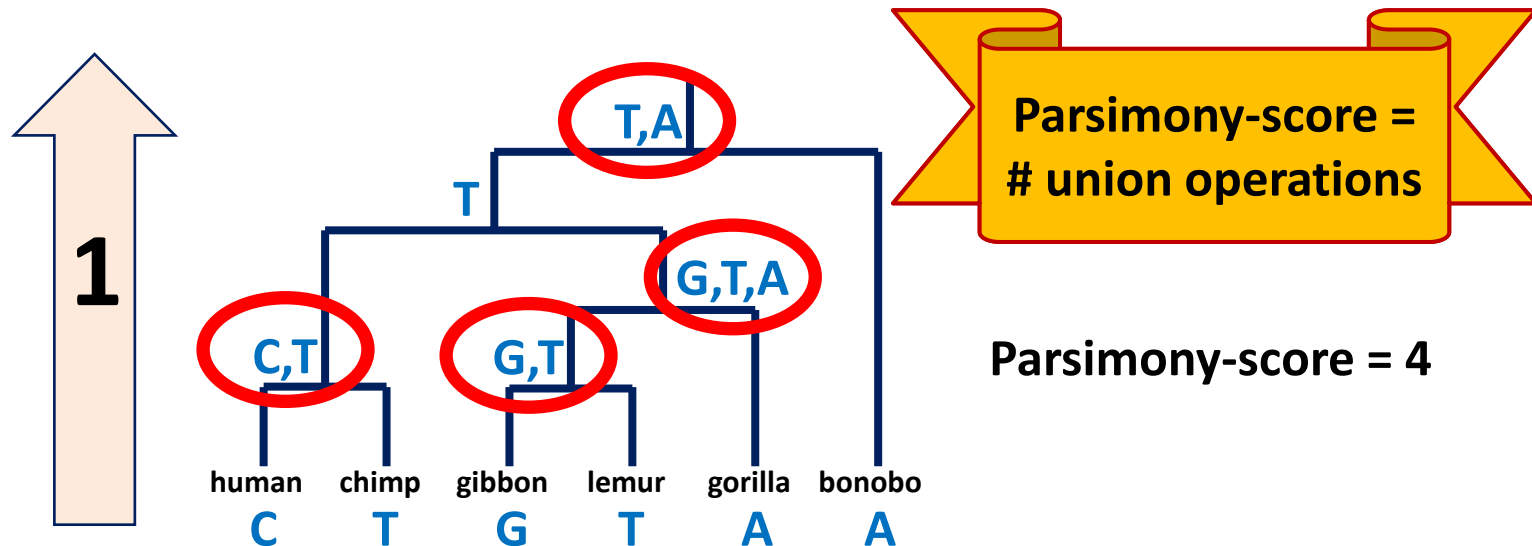
Let s_i denote the state of node i and R_i the set of possible states of node i

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 :

$$R_i = \left\{ \begin{array}{l} \text{if } R_j \cap R_k \neq \phi \rightarrow R_j \cap R_k \\ \text{otherwise} \rightarrow R_j \cup R_k \end{array} \right\}$$

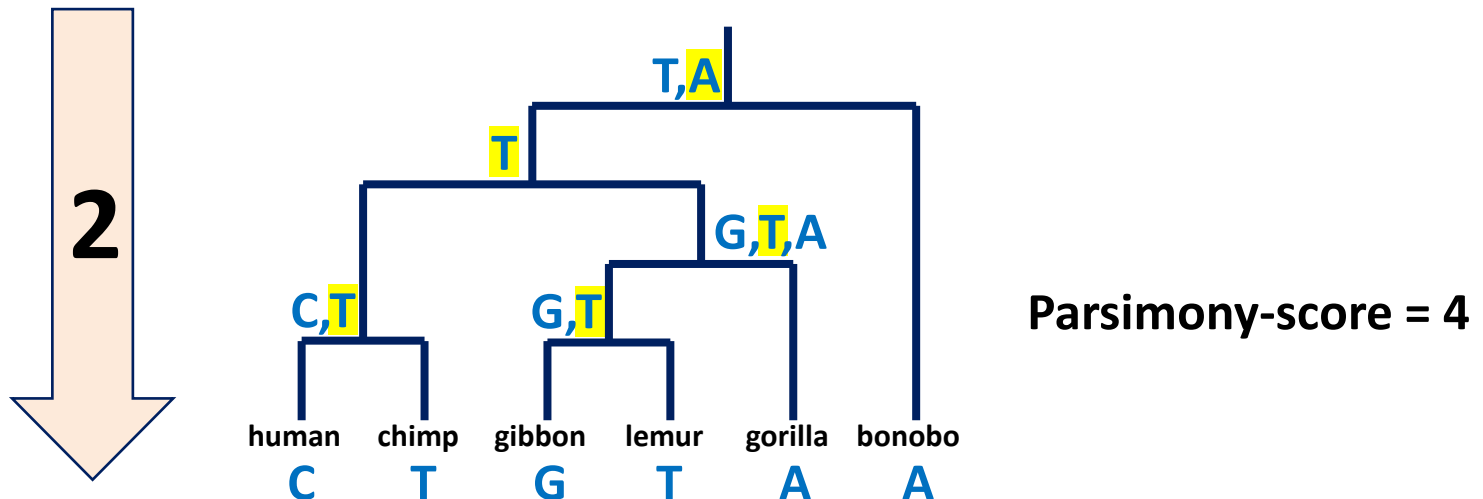


2. Fitch's algorithm: Top-down phase

(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 :

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

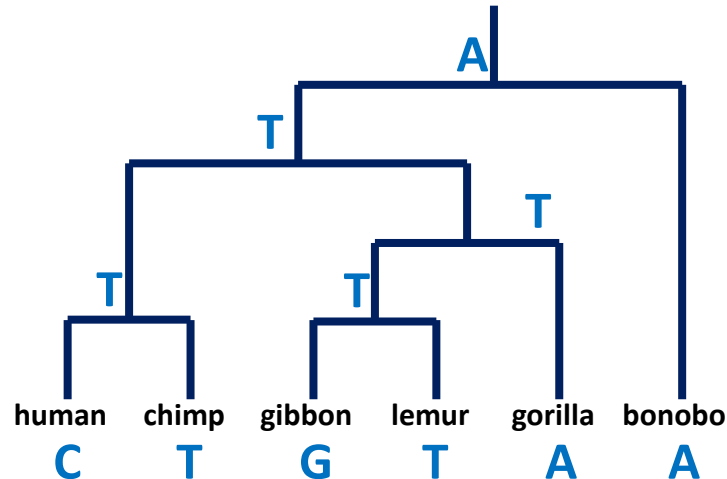
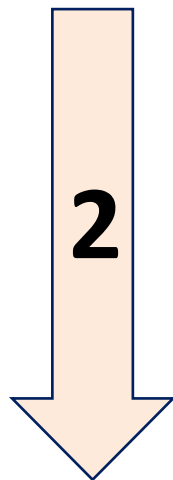


2. Fitch's algorithm: Top-down phase

(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 :

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



Parsimony-score = 4

The 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 **using Fitch's algorithm***
- 3) *Add all sites up to obtain the total number of changes for each tree*
- 4) *Pick the tree with the lowest score*

