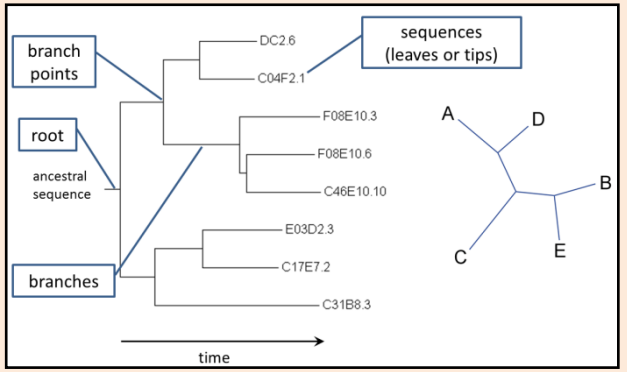
A quick review

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



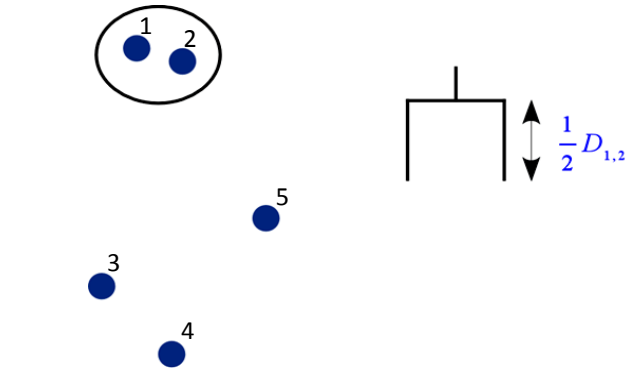
- Distance trees
  - **Aim to find the tree whose distances best match the observed distances**
  - Build tree by sequential clustering algorithm (**UPGMA**).

UPGMA

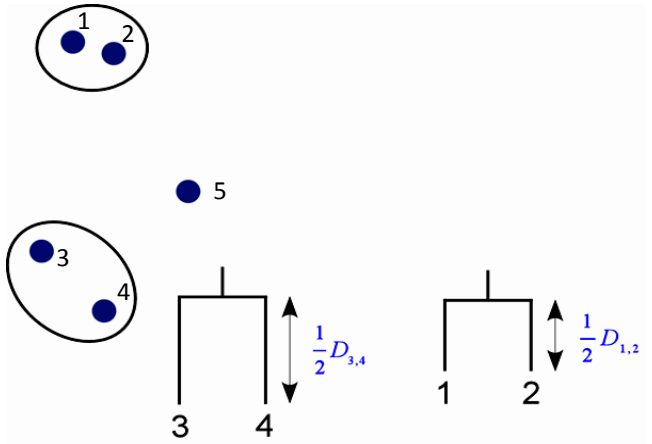
(**U**nweighted **P**air **G**roup **M**ethod with **A**rithmetic Mean)

1) generate a table of pairwise sequence distances and assign each sequence to a list of N tree nodes.
2) look through current list of nodes (initially these are all leaf nodes) for the pair with the smallest distance.
3) merge the closest pair, remove the pair of nodes from the list and add the merged node to the list.
4) repeat until only one node left in list - it is the root.

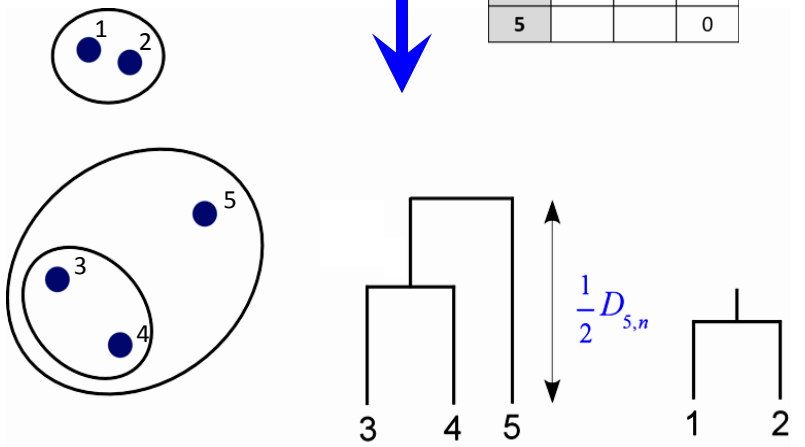
	1	2	3	4	5
1	0	5	18	22	17
2		0	20	24	15
3			0	10	12
4				0	12
5					0



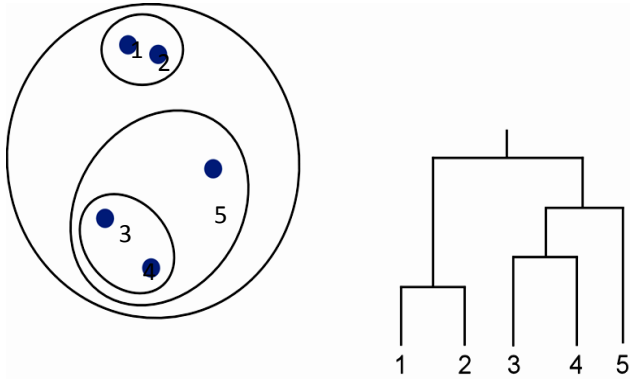
	1,2	3	4	5
1,2	0	19	23	16
3		0	10	12
4			0	12
5				0



	1,2	3,4	5
1,2	0	21	16
3,4		0	12
5			0

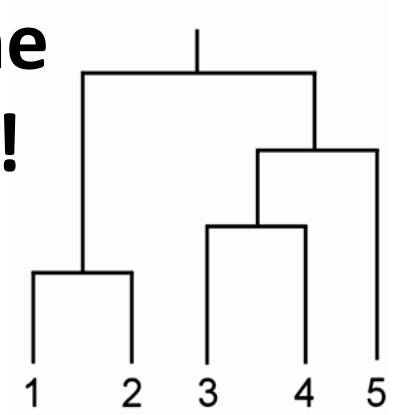


	1,2	3,4,5
1,2	0	19.33
3,4,5		0

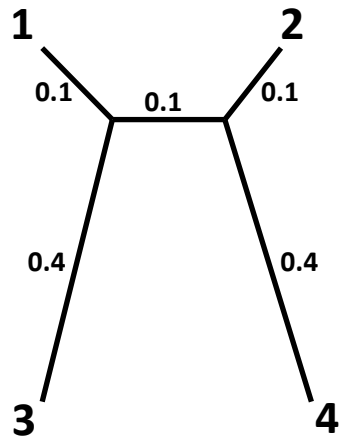


The Molecular Clock

- **UPGMA assumes a constant rate of the molecular clock across the entire tree!**
  - The sum of times down a path to any leaf is the same



- This assumption may not be correct … and will lead to incorrect tree reconstruction.

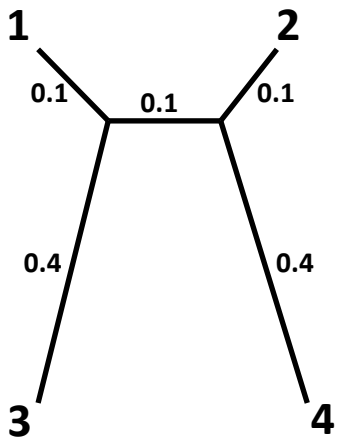


	1	2	3	4
1	0	0.3	0.5	0.6
2		0	0.6	0.5
3			0	0.9
4				0

Neighbor-Joining (NJ) Algorithm

- Essentially similar to UPGMA, but correction for distance to other leaves is made.

- Specifically, for sets of leaves *i* and *j*, we denote the set of all **other** leaves as *L*, and the size of that set as $|L|$, and we compute the corrected distance D_{ij} as:



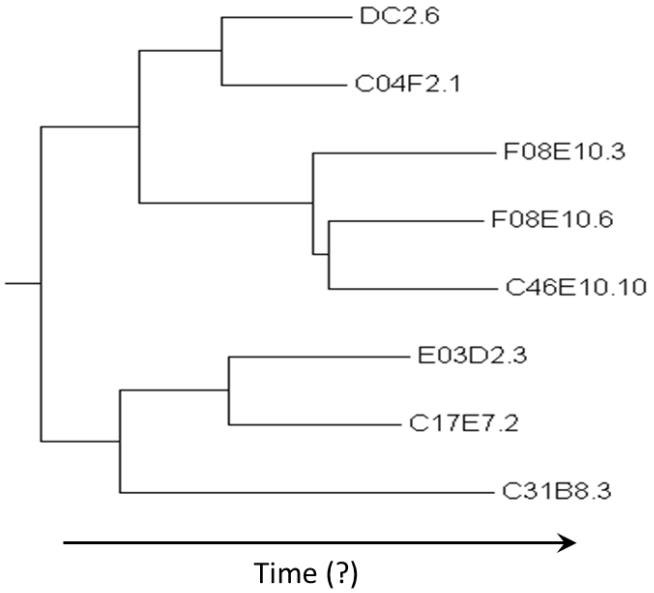
$$D_{ij} = d_{ij} - (r_i + r_j)$$

where

$$r_i = \frac{1}{|L|} \sum_{k \in L} d_{ik}$$

(the mean distance from i to all 'other' leaves)

But wait, there's one more problem



VS

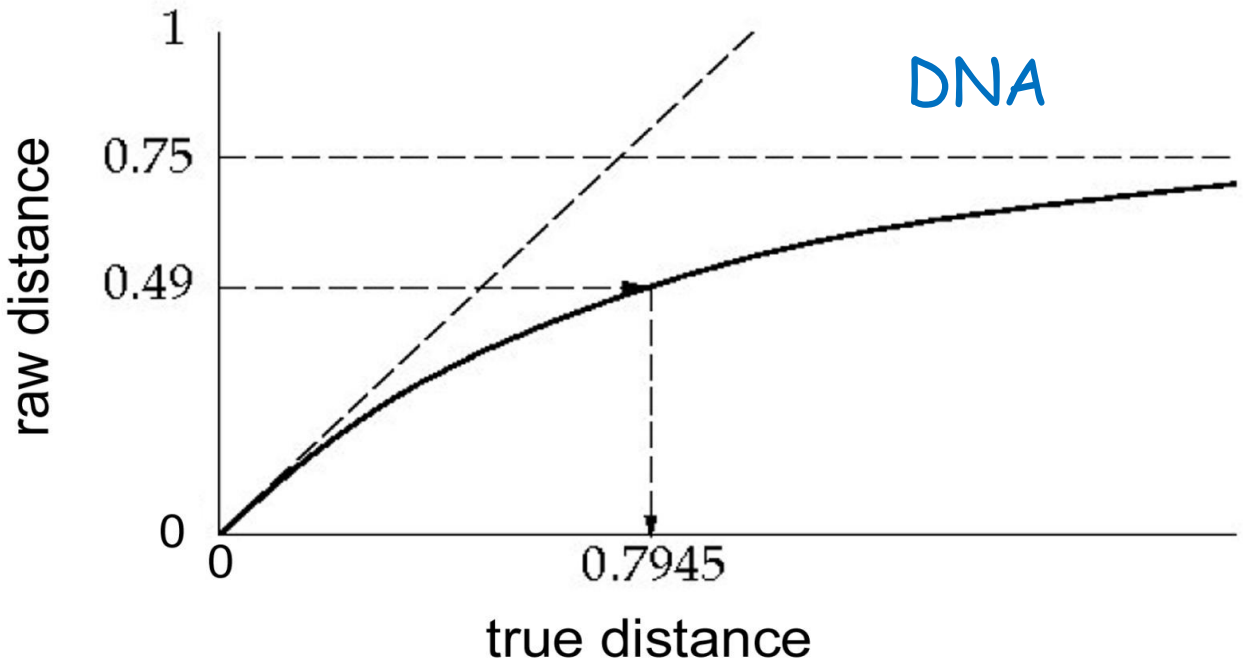
	E03D2.3	C17E7.2	C31B8.3	…
E03D2.3	0	20	33	.
C17E7.2		0	33	.
C31B8.3			0	.
…				0

Raw distance correction

- As two DNA sequences diverge, it is easy to see that their maximum raw distance is ~0.75 (assuming equal nt frequencies, ¼ of residues will be identical even if unrelated sequences).

- We would like to use the "true" distance, rather than raw distance.

- This graph shows evolutionary distance related to raw distance:



Jukes-Cantor model

Jukes-Cantor model:

$$D = -\frac{3}{4}\ln\left(1 - \frac{4}{3}D_{raw}\right)$$

D_{raw} is the raw distance (what we directly measure)
D is the corrected distance (what we want)

- Convert each pairwise raw distance to a corrected distance using Jukes-Cantor model.
- Build tree as before (UPGMA/NJ algorithm).

Distance trees – Summary notes

- **Note 1:** Notice that these methods only consider pairwise distances. All other information is discarded.

- **Note 2:** Notice that these methods don't need to enumerate all tree topologies - they are therefore very fast, even for large trees.

Parsimony I

Genome 373

Genomic Informatics

Elhanan Borenstein

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

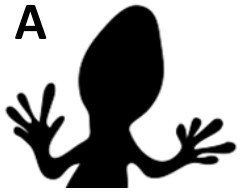
Parsimony principle for phylogenetic trees:

Find the tree that can explain the current states with the fewest evolutionary changes!

Lizard Island

Lizard Island

A

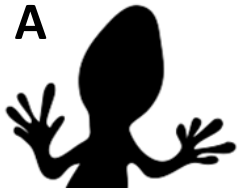


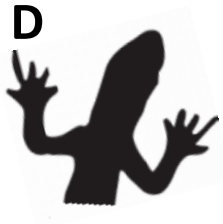
B

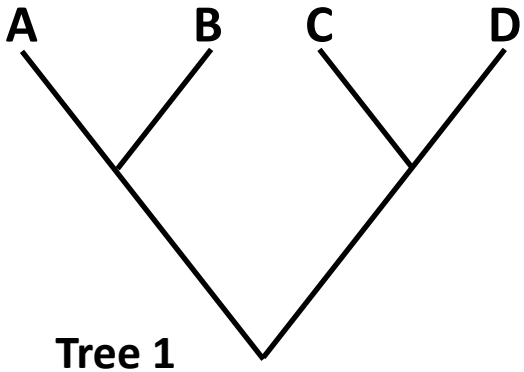
C

D

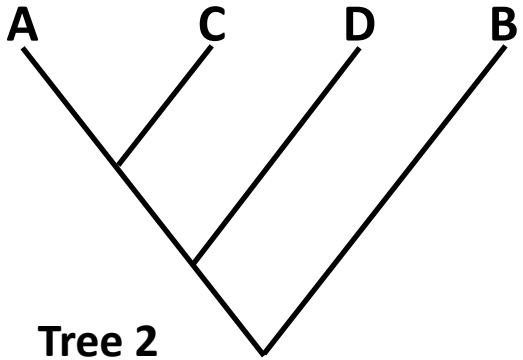
Lizard Island



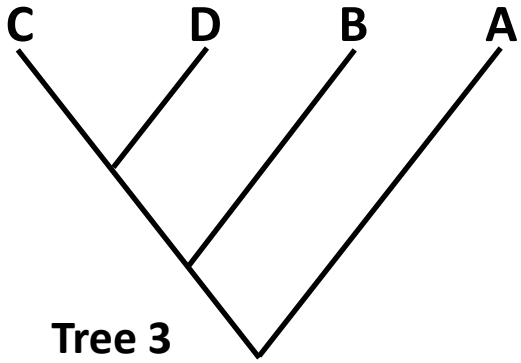




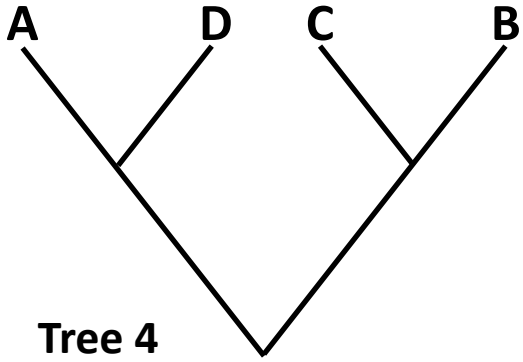
Tree 1



Tree 2



Tree 3



Tree 4

Consider 4 species

human
chimp
gorilla
orangutan

Consider 4 species

Sequence data:

```
          123456
human     agtctc
chimp     agagtc
gorilla   cggcag
orangutan 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:

```
          123456
human     agtctc
chimp     agagtc
gorilla   cggcag
orangutan cgggac
```

positions in alignment (usually called "sites")

Parsimony Algorithm

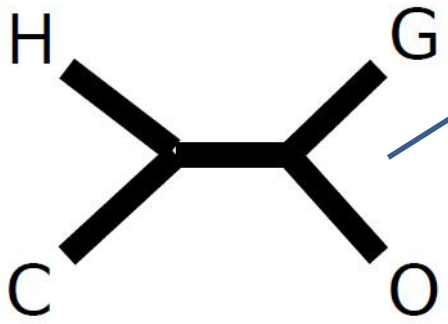
1) *Construct all possible trees*
2) ***For each site in the alignment and for each tree*** *count the minimal number of changes required*
3) *Add all sites up to obtain the total number of changes for each tree*
4) *Pick the tree with the lowest score*

Consider 4 species

Sequence data:

	123456
human	agtctc
chimp	agagtc
gorilla	cggcag
orangutan	cgggac

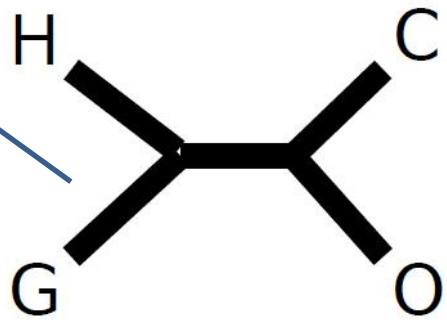
1) Construct all possible trees

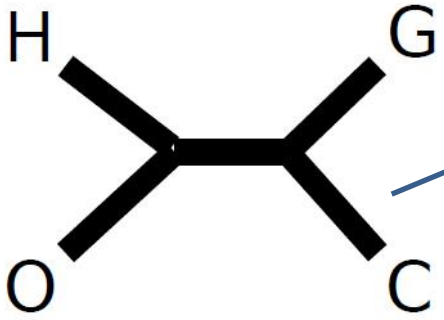


H closest to C

or

H closest to G





or

H closest to O

Consider 4 species

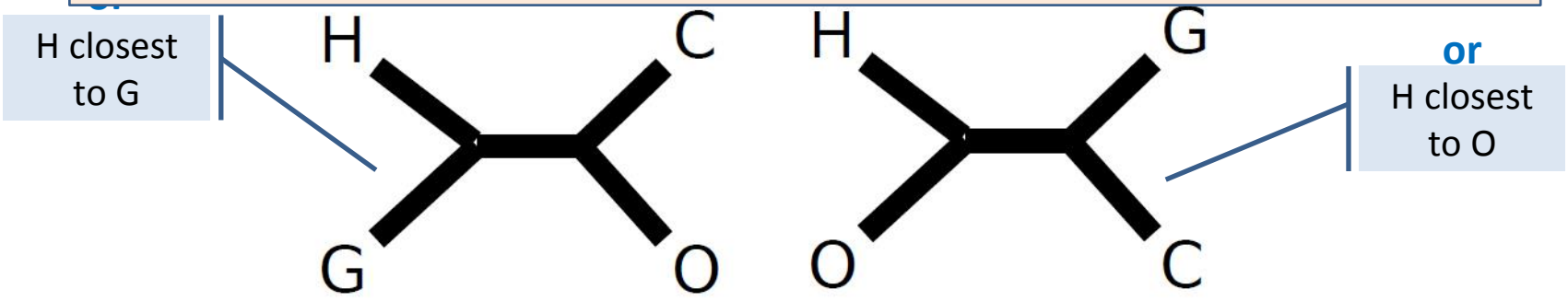
Sequence data:

```
          123456
human     agtctc
```

All

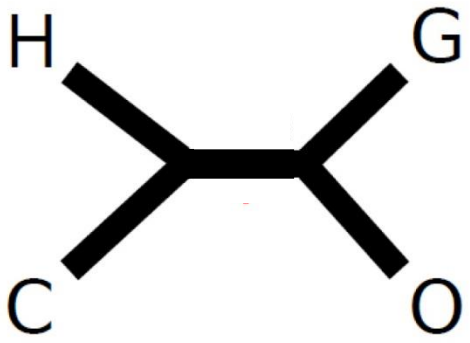
un

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



Consider site 1

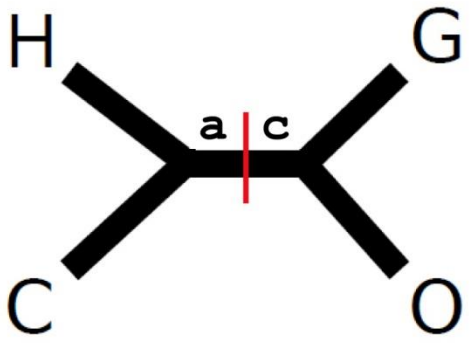
	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c



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

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

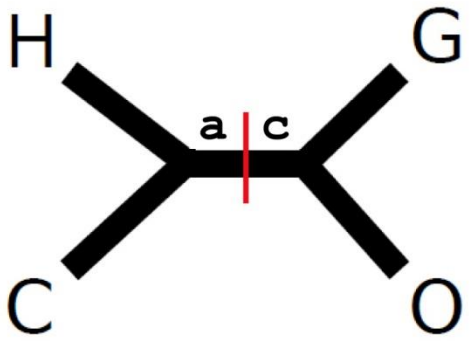


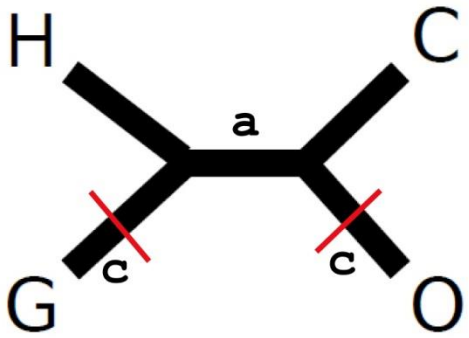
What is the minimal number of evolutionary changes that can account for the observed pattern?

(Note: This is the "small parsimony" problem)

Consider site 1

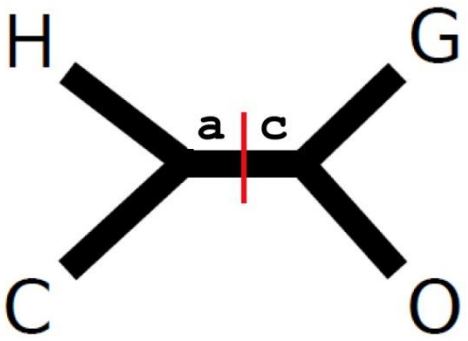
	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c

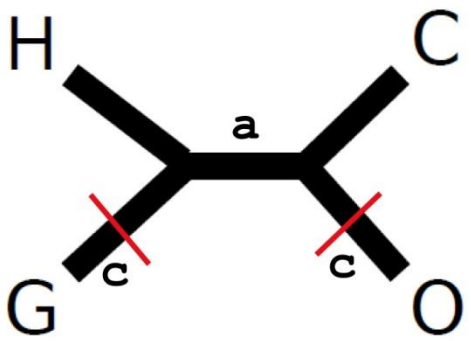


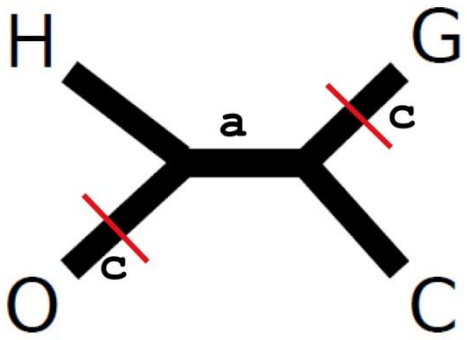


Consider site 1

	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



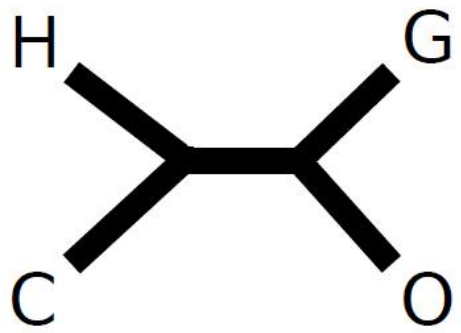


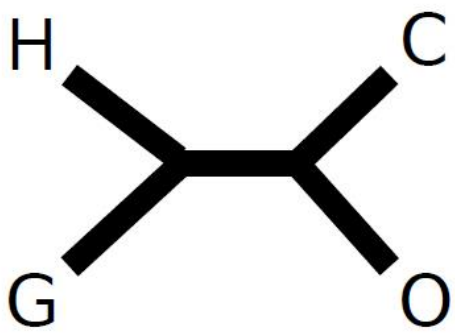


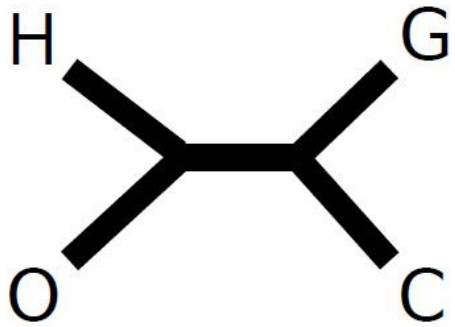
Consider site 2

	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)

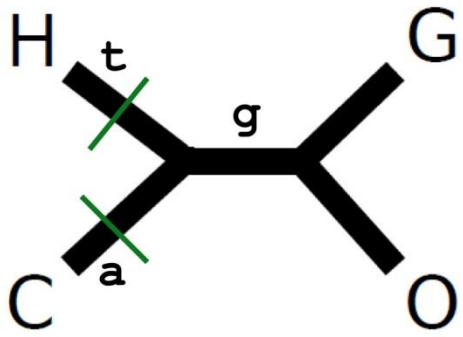


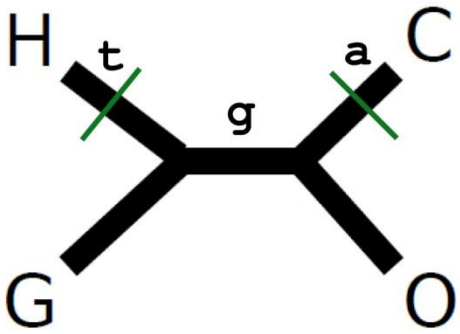


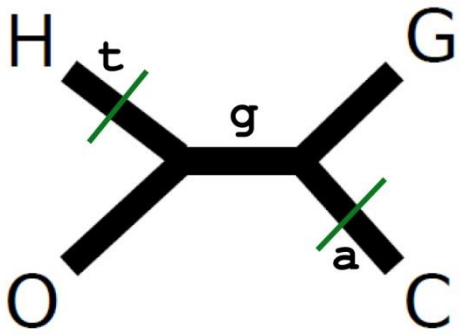


Consider site 3

	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

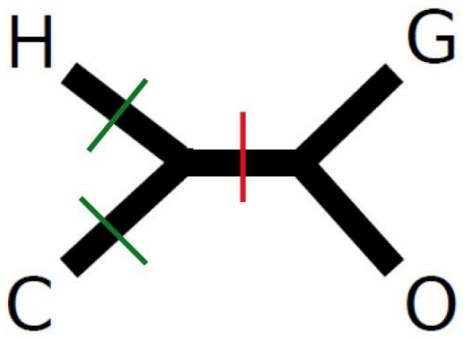


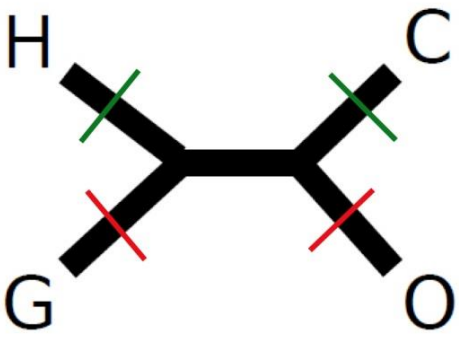


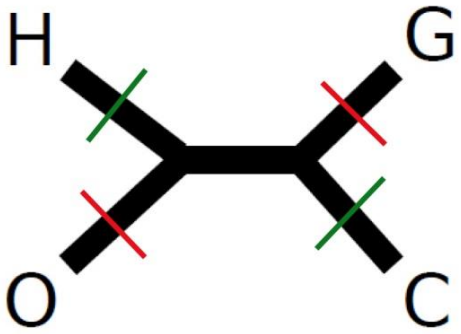


Put sites 1 and 3 together

	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



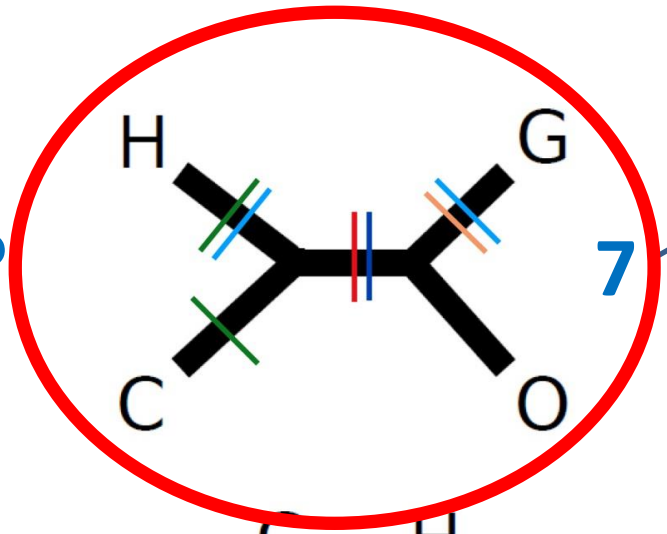




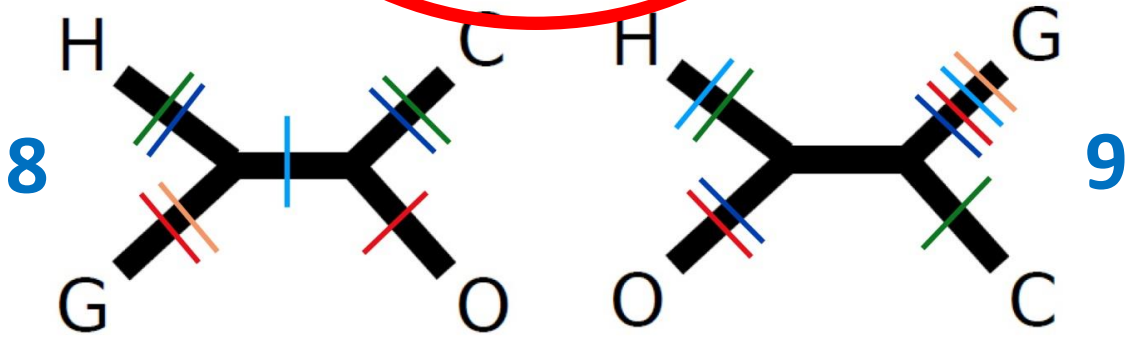
Now put all of them together

	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?



parsimony 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*

The parsimony algorithm

Too many!

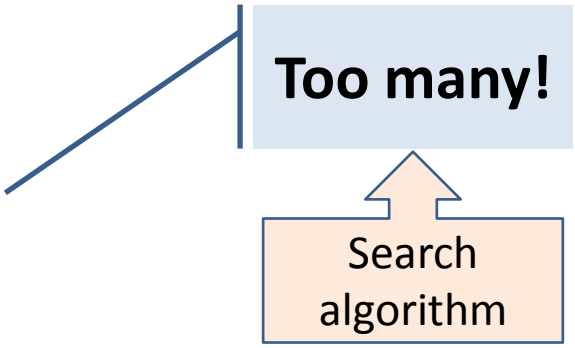
1) *Construct all possible trees*

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*

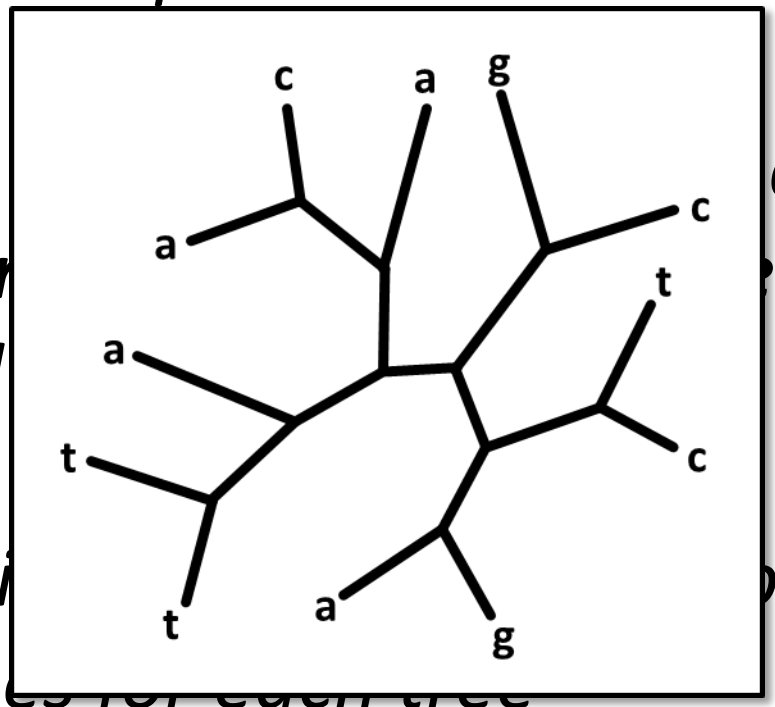
4) *Pick the tree with the lowest score*

The parsimony algorithm

Too many!

Search algorithm

1) *Construct all possible trees*



2) For each and for each tree cour r of changes required

3) Add all si otal number of change

4) Pick the tree with the lowest score

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?

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*

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

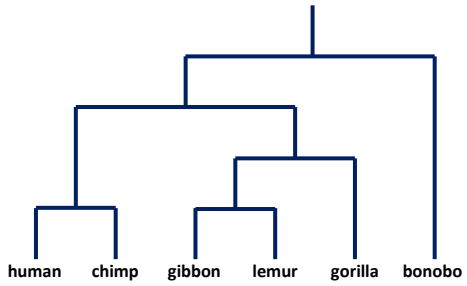
- Divide and conquer. (Think functions !!)
- Large parsimony is "NP-hard"
- Small parsimony can be solved quickly using Fitch's algorithm

Parsimony Algorithm

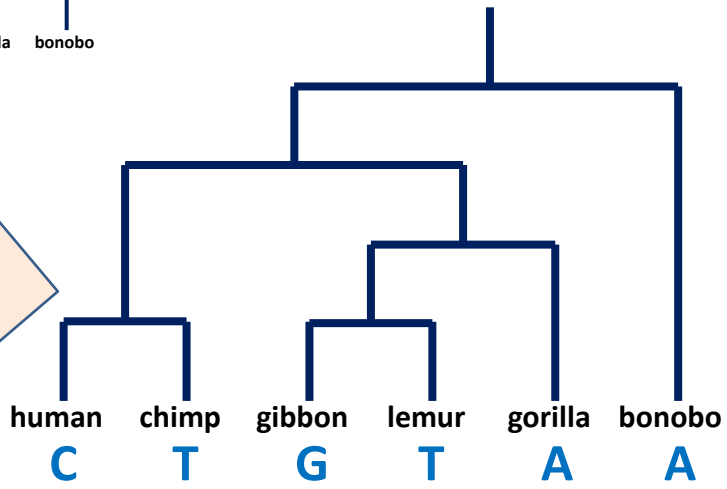
1) *Construct all possible trees*
2) *For each site in the alignment and for each tree count the minimal number of changes required*
3) *Add all sites up to obtain the total number of changes for each tree*
4) *Pick the tree with the lowest score*

The 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



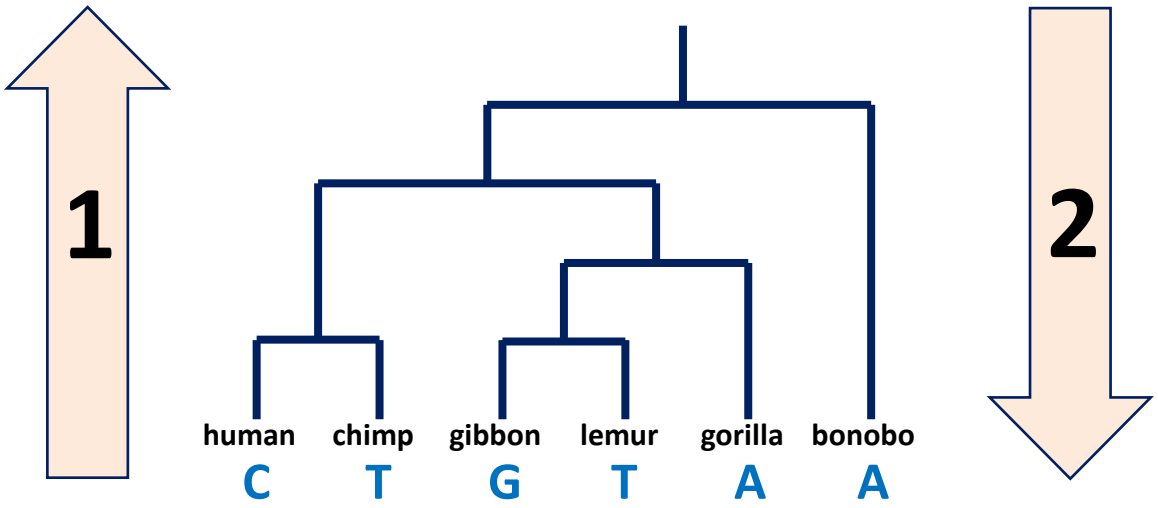
- **Output:**

  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

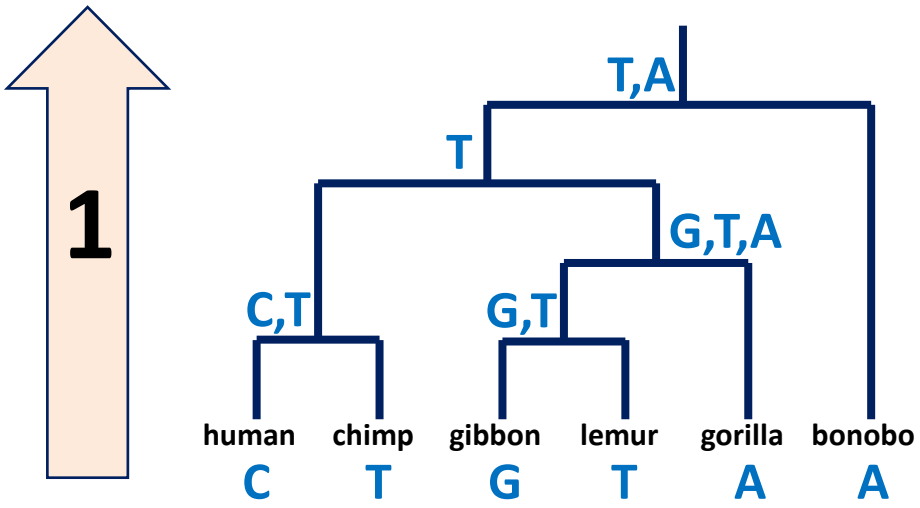


1. Fitch's algorithm: Bottom-up phase

(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:

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



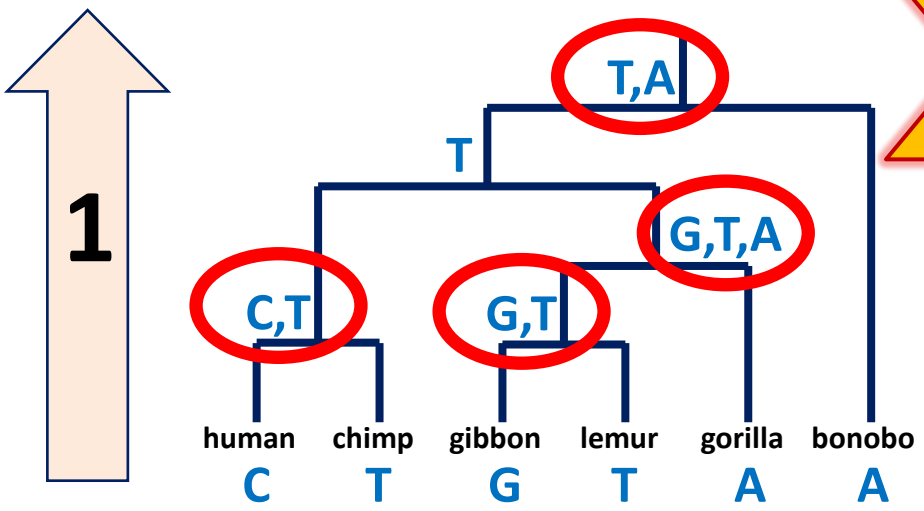
Let s_i denote the state of node i and R_i the set of possible states of node i

1. Fitch's algorithm: Bottom-up phase

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

1. Initialization: $R_i = \{s_i\}$ 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:

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



Parsimony-score = # union operations

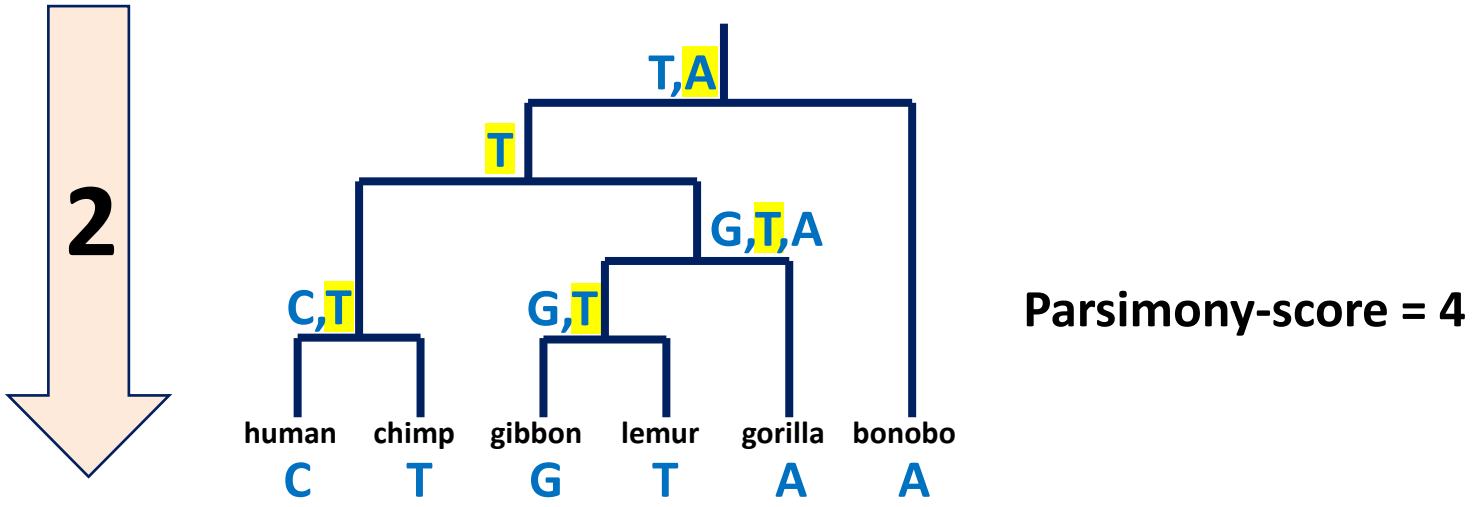
Parsimony-score = 4

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:

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



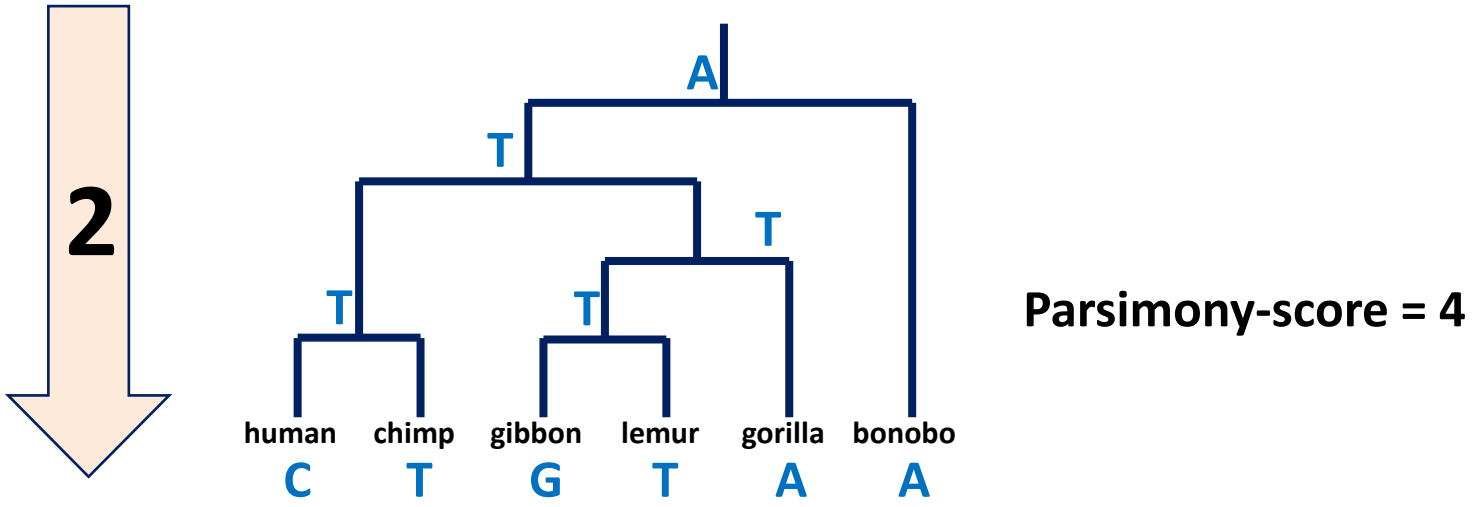
Parsimony-score = 4

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:

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



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*