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 or Neighbor-Joining).
  - These algorithms don't consider all tree topologies, so they are very fast, even for large trees.
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)
Raw distance correction
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 (observed) distance:
Jukes-Cantor model

Jukes-Cantor model:

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

\( D_{\text{raw}} \) is the raw distance (what we directly measure)

\( D \) is the corrected distance (what we want)
Distance trees - summary

- Convert each pairwise raw distance to a corrected distance.

- Build tree as before (UPGMA/NJ algorithm).

- Notice that these methods don't need to consider all tree topologies - they are 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.
- "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
Consider 4 species

human
cchimp
gorilla
orangutan
Consider 4 species

Sequence data:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>agtctc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chimp</td>
<td>agagtc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gorilla</td>
<td>cggcag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orangutan</td>
<td>cgggac</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>agtctc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chimp</td>
<td>agagtca</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gorilla</td>
<td>cgcccccg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orangutan</td>
<td>cgggac</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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:

- human: 123456
- chimp: agtctc
- gorilla: agagtc
- orangutan: cgccag
- orangutan: cgggac

All possible unrooted trees:

- H closest to C
- H closest to G
- H closest to O
Consider 4 species

Sequence data:

All possible unrooted trees:

For each site and for each tree, count the minimal number of changes required:
Consider site 1

What is the minimal number of evolutionary changes that can account for the observed pattern?
Consider site 1

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>chimp</td>
<td>a</td>
<td>g</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>gorilla</td>
<td>c</td>
<td>g</td>
<td>g</td>
<td>c</td>
<td>a</td>
<td>g</td>
</tr>
<tr>
<td>orangutan</td>
<td>c</td>
<td>g</td>
<td>g</td>
<td>g</td>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>
Consider site 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>ag</td>
<td>t</td>
<td>c</td>
<td>t</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>chimp</td>
<td>ag</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>gorilla</td>
<td>cg</td>
<td>g</td>
<td>c</td>
<td>a</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>orangutan</td>
<td>cg</td>
<td>g</td>
<td>g</td>
<td>a</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>
Consider site 2

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>chimp</td>
<td>a</td>
<td>g</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>gorilla</td>
<td>c</td>
<td>g</td>
<td>g</td>
<td>c</td>
<td>a</td>
<td>g</td>
</tr>
<tr>
<td>orangutan</td>
<td>c</td>
<td>g</td>
<td>g</td>
<td>g</td>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>

Uninformative (no changes)
Consider site 3

<table>
<thead>
<tr>
<th></th>
<th>human</th>
<th>chimp</th>
<th>gorilla</th>
<th>orangutan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ag</td>
<td>ag</td>
<td>cggc</td>
<td>cggac</td>
</tr>
<tr>
<td>2</td>
<td>g</td>
<td>g</td>
<td>g</td>
<td>g</td>
</tr>
<tr>
<td>3</td>
<td>t</td>
<td>c</td>
<td>cag</td>
<td>gac</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>t</td>
<td>cag</td>
<td>gac</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>g</td>
<td>g</td>
<td>g</td>
</tr>
<tr>
<td>6</td>
<td>c</td>
<td>it</td>
<td>cag</td>
<td>gac</td>
</tr>
</tbody>
</table>
Put sites 1 and 3 together

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>chimp</td>
<td>a</td>
<td>g</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>gorilla</td>
<td>c</td>
<td>g</td>
<td>g</td>
<td>c</td>
<td>a</td>
<td>g</td>
</tr>
<tr>
<td>orangutan</td>
<td>c</td>
<td>g</td>
<td>g</td>
<td>g</td>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>

Diagram:

```
  H   G
 /|
C | O
 /|
G   H
```
Now put all of them together

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

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

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
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:

```
          human
         /     \
      chimp   gibbon
        / \    /  \
  lemur gorilla bonobo
```

- **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

![Diagram of phylogenetic tree with characters and nucleotides]
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 \( j, k \):

\[
R_i = \begin{cases} 
\text{if } R_j \cap R_k \neq \emptyset \rightarrow R_j \cap R_k \\
\text{otherwise } \rightarrow R_j \cup R_k
\end{cases}
\]

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 \( j, k \):

\[
R_i = \begin{cases} 
\text{if } R_j \cap R_k \neq \emptyset \rightarrow R_j \cap R_k \\
\text{otherwise } \rightarrow R_j \cup R_k 
\end{cases}
\]

**Example:**

<table>
<thead>
<tr>
<th>Gene</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C,T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>G</td>
<td>G,T,A</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

**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 $j$:

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

Parsimony-score = 4
2. Fitch’s algorithm: Top-down phase

(Pick a state for each internal node)

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

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

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