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
(Unweighted Pair Group Method with Arithmetic 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.
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.

![Diagram of a molecular clock tree]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.6</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
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
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:

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

\( D_{\text{raw}} \) 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 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
Lizard Island

Tree 1
A → B → C → D

Tree 2
A → C → D → B

Tree 3
C → D → B → A

Tree 4
A → D → C → B

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>

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>human</th>
<th>123456</th>
</tr>
</thead>
<tbody>
<tr>
<td>agtctc</td>
<td></td>
</tr>
<tr>
<td>chimp</td>
<td></td>
</tr>
<tr>
<td>agagtc</td>
<td></td>
</tr>
<tr>
<td>gorilla</td>
<td></td>
</tr>
<tr>
<td>cggcag</td>
<td></td>
</tr>
<tr>
<td>orangutan</td>
<td></td>
</tr>
<tr>
<td>cgggac</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 (cggcag)

1) Construct all possible trees

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

Sequence data: 123456

All possible unrooted trees:

2) 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>ag t c t c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chimp</td>
<td>ag a g t c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gorilla</td>
<td>cg g c a g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orangutan</td>
<td>cg g g a c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Diagram of site 1: [Diagram Image]
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></td>
<td>cg</td>
<td>g</td>
<td>c</td>
<td>a</td>
<td>g</td>
</tr>
<tr>
<td>orangutan</td>
<td></td>
<td>cg</td>
<td>g</td>
<td>g</td>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>
Consider site 2

Human: agttctc
Chimp: agagtc
Gorilla: cggcag
Orangutan: cgggac

Uninformative
(no changes)
Consider site 3
Put sites 1 and 3 together

<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>agttc</td>
<td>agttc</td>
<td>cggcac</td>
<td>cgggac</td>
</tr>
<tr>
<td>2</td>
<td>agttc</td>
<td>agttc</td>
<td>cggcac</td>
<td>cgggac</td>
</tr>
<tr>
<td>3</td>
<td>cggcac</td>
<td>cggcac</td>
<td>cgggac</td>
<td>cgggac</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Human sequence: agttc
- Chimpanzee sequence: agttc
- Gorilla sequence: cggcac
- Orangutan sequence: cgggac

Diagram:

```
H   G
C   O
```

```
H   C   G
C   O
```

```
H   C
G   O
```

```
H   C
G   O
```

```
H   G
C   O
```

```
H   G
C   O
```
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
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 minimum number of changes required

3) Add all site information 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
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:

- **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 $j, k$:

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

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

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

   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 $R_{\text{root}}$ 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 $j$:

$$s_i = \begin{cases} 
  s_j & \text{if } s_j \in R_i \\
  \text{arbitrary state} & \text{otherwise}
\end{cases} \quad i \neq j \quad i \in R$$

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 $j$:

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