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.

- "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:

<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>ctc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chimp</td>
<td>ag</td>
<td>a</td>
<td>gtc</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)

positions in alignment (usually called "sites")
Consider 4 species

Sequence data:

<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>agtctc</td>
<td>agagtc</td>
<td>cgccag</td>
<td>cgggac</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></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>

**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

positions in alignment (usually called "sites")
Consider 4 species

Sequence data:

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

All possible unrooted trees:

1. H closest to C
2. H closest to G
3. H closest to O
Consider 4 species

Sequence data: [Diagram showing two trees with labels H, C, G, O]

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

- H closest to G
- H closest to O
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

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>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>g</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>g</td>
<td>g</td>
<td>c</td>
<td>a</td>
<td>g</td>
</tr>
<tr>
<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>agtcttc</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>

Diagram:

- Human: HGC
- Chimpanzee: HGC
- Gorilla: HGC
- Orangutan: HGC

Evolutionary tree with mutations indicated.
Consider site 2

- Human: agtctc
- Chimp: agagtc
- Gorilla: cggcag
- Orangutan: cgggac

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

Too many!
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

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

- 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:

```
<table>
<thead>
<tr>
<th></th>
<th>Human</th>
<th>Chimp</th>
<th>Bonobo</th>
<th>Gorilla</th>
<th>Gibbon</th>
<th>Lemur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>A</td>
<td>G</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>T</td>
<td>A</td>
<td>G</td>
<td>C</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>
```

- **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: $R_i = \{s_i\}$
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 $R_{root}$ 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 $j$:

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

---

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 $j$:

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

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