Phylogenetic trees

The only figure from Darwin’s “The Origin of Species”

Summary of Darwin's theory of evolution:

1. Species are not fixed
2. Common descent
3. Multiplication of species
4. Gradualism
5. Natural selection

In 1866, Ernst Haeckel coined the word *phylogeny* and presented phylogenetic trees for most known groups of living organisms.
Tree of Life

Doolittle, 19999

About 1.7 million species described.
What we have so far:

- TreeBASE database (15,000 taxa)
- Ribosomal Database Project (RDP II) (20,000 sequences)
- The Tree of Life Project (11,000 taxa)

NSF $10 million to construct a phylogeny for the 1.7 million described species of Life
Tree building algorithms

Distance-based methods

- Input: distance data such as sequence edit distance
- Output: weighted tree with pairwise distances matching evolutionary distance
- We will consider data that is:
  - Ultrametric (section 17.1)
  - Additive but not ultrametric (section 17.2)

Maximum-parsimony methods

- Character-based methods
- Input: character data (often aligned sequences)
- Output: tree with
  - input taxa at leaves
  - Inferred taxa at internal nodes
- Goal: minimize the total cost of mutations
  - maximize parsimony.
  - seek a tree that has the minimum cost over all possible trees

Definition. Let $A$ be a set of taxa. Let $d : A \times A \to \mathbb{R}$ be a function. Then, $d$ is a **metric** on $A$ if it satisfies the following properties:

- (i) For all $a, b \in A$, $d(a, b) \geq 0$ (**non-negativity**).
- (ii) For all $a, b \in A$, $d(a, b) = 0$ holds if and only if $a = b$.
- (iii) For all $a, b \in A$, $d(a, b) = d(b, a)$ (**symmetry**).
- (iv) For all $a, b, c \in A$, $d(a, b) \leq d(a, c) + d(c, b)$ (**triangle inequality**).

**Definition.** Let $A$ be a set of taxa. Let $d : A \times A \to \mathbb{R}$ be a function. Then, $d$ is **ultrametric** if it satisfies the following properties:

- (i) For all $a, b \in A$, $d(a, b) \geq 0$ (**non-negativity**).
- (ii) For all $a, b \in A$, $d(a, b) = 0$ holds if and only if $a = b$.
- (iii) For all $a, b \in A$, $d(a, b) = d(b, a)$ (**symmetry**).
- (iv) For all $a, b, c \in A$, $d(a, b) \leq \max(d(a, c), d(c, b))$ (**strong triangle** or **ultrametric inequality**).

Every ultrametric $d$ is a metric.

**Theorem.** [Peter Bunemann] A metric $d$ is ultrametric if and only if any three points $a, b, c$ can be renamed such that

$$d(a, b) \leq d(a, c) = d(b, c).$$
Ultrametric Tree

- $T$ has $n$ leaves corresponding to rows and columns
- Each internal node has at least two children
- Internal nodes are labeled and each parent has larger label than any child
- $D(i, j)$ is the label of least common ancestor of $i$ and $j$

\[
\begin{array}{cccccc}
A & B & C & D & E \\
\hline
A & 0 & 8 & 8 & 5 & 3 \\
B & 0 & 3 & 8 & 8 & \\
C & 0 & 8 & 8 & \\
D & 0 & 5 & \\
E & 0 & \\
\end{array}
\]

- $T$ has $n$ leaves corresponding to rows and columns
- Each internal node has at least two children
- Edges are weighted
- The sum of the weights on every path from the root to any leaf is the same

$D(i, j)$ is the total weight of edges on the path between $i$ and $j$.

How to test for an ultrametric tree

**Definition.** A symmetric matrix $D$ defines ultrametric distance if for every 3 indices $i, j, k$ the maximum of $D(i, j)$, $D(i, k)$, $D(j, k)$ is not unique.

**Theorem.** A symmetric matrix $D$ has an ultrametric tree if and only if $D$ is ultrametric matrix.

**Theorem.** Every ultrametric matrix has unique ultrametric tree. It can be found in $O(n^2)$ time.

Ultrametric Tree Construction

- Sort the first row $A : 3, 5, 8$. These are the labels on the path from $A$ to the root.
- Group the nodes using these labels.
- Recursively solve the problem in each group.
- Time is $O(n^2 \log n)$. There is a better algorithm with $O(n^2)$ runtime:
Let $N$ be the set of taxa and $i$ be any taxon. 
Let $L$ be an empty list of taxa.
While $N \neq \emptyset$: remove $i$,
find $j \in N$ minimizing $D(i, k)$, $k \in N$,
add $j$ to $L$ and set $i$ to $j$.

Additive-distance trees

- $T$ has $n$ leaves corresponding to rows and columns
- Each edge has a label
- $D(i, j)$ is the total weight of the path between $i$ and $j$

Tree construction

- Given a symmetric matrix with diagonal entries equal 0, find an additive tree if any.
- Two versions. Rows and columns correspond to a) leaves, b) all nodes of $T$. There are algorithms for a) with $O(n^2)$ time.
- We consider case b), compact additive tree.

Theorem. Compact additive tree $T$ for $D$ is the unique minimum spanning tree of $G(D)$ (complete graph whose edge weighs are from from $D$).
We assume that $D(x, y) > 0$ if $x \neq y$.

**Proof.** Let $T$ be a compact additive tree for $D$, and let $e = (x, y)$ be an edge of $G(D)$ not in $T$. Let $x_1 = x, x_2, \ldots, x_k = y$ be the $xy$-path in $T$. Then $w(x, y) = w(x_1, x_2) + \cdots + w(x_{k-1}, x_k)$ and $w(x, y) > w(x_i, x_{i+1})$ for any $i$.

We show that $e$ is not in any minimum spanning tree of $G(D)$. Suppose to the contrary that $e$ is in a MST $T'$.

Removal $e$ from $T'$ produces 2 trees $T'_x$ and $T'_y$ so that $x \in T'_x$ and $y \in T'_y$. There is an $i$ such that $x_i \in T'_x$ and $x_{i+1} \in T'_y$. Add $(x_i, x_{i+1})$ and new tree has weight

$$w(T') - w(x, y) + w(x_i, x_{i+1}) < w'(T).$$

This contradicts that $T'$ is a MST.

**Algorithm.** Given $D$, construct $G(D)$ and compute MST with Prim’s algorithm. We want to verify that tree distances $d(u, v) = D(u, v)$.

In Prim’s iteration, a new vertex $y$ is discovered and a new edge $(x, y)$ is added to the current tree $T$.

For each vertex $i \in T$, we check that $D(i, y) = D(i, x) + D(x, y)$.

This takes $O(n)$ time per iteration and $O(n^2)$ time overall.

### Additive and ultrametric trees

Consider version a) where rows and columns of $D$ correspond to the leaves of unknown tree.

- Let $D$ be additive matrix and $T$ be its additive tree. Let $m_v$ be the max entry of $D$. Root $T$ at node $v$.
  
  We “stretch” leaf edges so that $v$ is equidistant to each leaf, i.e. add $m_v - D(v, i)$ edge distance at leaf $i$.

- Can we do it without knowing $T$?

  - Yes. $2D'(i, j) = D(i, j) + m_v - D(v, i) + m_v - D(v, j)$.

  - $D'(i, j) = m_v + (D(i, j) - D(v, i) - D(v, j))/2$.

**Theorem.** $D$ is an additive if and only if $D'$ is ultrametric.

From additive matrix $D$ to ultrametric matrix $D'$
• Let $T''$ be the ultrametric tree of $D'$.
  
  • Label every edge $(p, q)$ by the difference of the numbers at $p$ and $q$ (the numbers at leaves are 0).
  
  • For every leaf $i$, subtract $m_v - D(v, i)$ from its incident edge.

UPGMA

UPGMA stands for Unweighted Pair Group Method with Arithmetic Means.

  • A simple method for hierarchical clustering
• UPGMA has \( n - 1 \) iterations for given \( n \) taxa
• Iteration: find two closest clusters and join them
• Distance between clusters \( A \) and \( B \) is the mean distance
\[
d(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)
\]

Good news
• It computes ultrametric tree if it exists
• There is an \( O(n^2) \)-time algorithm

Bad news: it may not find an additive tree even for \( n = 4 \)

Neighbor-joining algorithm

Main iteration
1. Given \( n \) clusters (rooted trees) and the current distances \( d_{i,j} \),
calculate \( r_i = \sum_{k=1}^{n} d_{i,k} \) and matrix \( Q_{i,j} = (n - 2)d_{i,j} - r_i - r_j \).
2. Find the pair of taxa \((i, j)\) in \( Q \) with the lowest value.
   Create a node \( u \) joining \( T_i \) and \( T_j \).
   Set edge weights \( w_{i,u} = \frac{1}{2}d_{i,j} + \frac{1}{2(n - 2)}(r_i - r_j) \) and \( w_{j,u} = d_{i,j} - w_{i,u} \).
3. Update $d$ by adding $u$ with $d_{u,k} = \frac{1}{2}(d_{i,k} + d_{j,k} - d_{i,j})$ and removing $i$ and $j$.

Example: http://www.utdallas.edu/~sxb027100/NJ.pdf

Advantages of Neighbor-joining method

- is fast and thus suited for large datasets and for bootstrap analysis
- permits lineages with largely different branch lengths
- permits correction for multiple substitutions

Disadvantages

- sequence information is reduced
- gives only one possible tree
- strongly dependent on the model of evolution used

Parsimony

- Based on attributes or characters that objects may possess.
- **Definition.** $M$ is $n \times m$ binary matrix representing $n$ objects in terms of $m$ traits. Phylogenetic tree for $M$:
  - Each of $n$ objects labels exactly one leaf
  - Each of $m$ traits labels exactly one edge
  - The characters of an object $p$ (row in $M$) are the characters on the path from leaf $p$ to the root.

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Interpretation of the tree based on biological assumptions:

- The root represents an ancestor without traits
- Each trait status changes from NO to YES exactly once and never changes later.

**Perfect phylogeny**

- Given $M$, find a phylogenetic tree $T$ for $M$ if any.
- Sort columns in decreasing order.
- Let $O_k$ be the set of taxa with 1 in column $k$ (set of taxa that have $k$th trait).

**Theorem.** Matrix $M$ has a phylogenetic tree if and only if for any pair $i, j$ either $O_i$ and $O_j$ are disjoint or one contains the other.

**Proof.** $\Rightarrow$ Let $T$ be a phylogenetic tree for $M$ and let $e_i$ and $e_j$ be two edges of $T$ with characters $i$ and $j$. There are 4 cases.

1. $e_i = e_j$. Then $O_i = O_j$.
2. $e_i$ is on the path from $e_j$ to the root. Then $O_j \subseteq O_i$.
3. $e_j$ is on the path from $e_i$ to the root. Then $O_i \subseteq O_j$.
4. The paths from $e_i$ and $e_j$ to the root meet at some vertex. Then $O_i \cap O_j = \emptyset$.

**Proof.** $\Leftarrow$ Sort traits so that $|O_i|$ is increasing sequence.

Let $x$ be a taxon and $i_1, \ldots, i_k$ be its traits.

The sets $O_{i_1}, \ldots, O_{i_k}$ contain $x$ (no two sets are disjoint). Then

$$O_{i_1} \subset O_{i_2} \subset \cdots \subset O_{i_k}.$$
Algorithm

- Sort columns using radix sort making $M'$.
- For each row $p$, construct the string of characters it possesses (in increasing order).
- Build the keyword tree $T$ for $n$ strings.
- Test whether $T$ is a perfect phylogeny for $M$.

Runtime $O(mn)$. 

Tree Compatibility

We assume more general trees: a leaf can be labeled by many taxa. “Reduced form” of a tree: binary tree and only root can have one child.

Definition. A phylogenetic tree $T'$ is a refinement of $T$ if $T$ can be obtained by a series of contractions of edges of $T'$.

If $T'$ refines $T$ then $T'$ agrees with all the evolutionary history displayed in $T$.

Definition. Trees $T_1$ and $T_2$ are compatible is there exists a phylogenetic tree $T_3$ refining both $T_1$ and $T_2$. 
Tree compatibility problem. Given two trees $T_1$ and $T_2$, determine whether the two trees are compatible, and if so, produce a refinement tree $T_3$.

Let $M_1$ be a 0-1 matrix with one row for each object and one column for each internal node $j$ in $T_1$. Entry $(i, j)$ of $M_1$ has value one if and only if the leaf for object $i$ is found below node $j$.

Matrix $M_2$ is similarly defined for $T_2$. $M_3$ is the union of columns of $M_1$ and $M_2$.

**Theorem.** $T_1$ and $T_2$ are compatible if and only if there is a phylogenetic tree for $M_3$. Further, a phylogenetic tree for $M_3$ is a refinement of both $T_1$ and $T_2$.

Theorem reduces tree compatibility to perfect phylogeny.
The algorithm takes $O(n^2)$ time.

Can we detect tree compatibility faster? Yes. Idea: find common subtree $T'_1 = T'_2$.

For each node find smallest object.
Find largest common subtree. Check that the sets at leaves are the same.

**Theorem.** Tree compatibility can be detected in $O(n)$ time. A refinement of two trees can be computed in $O(n)$ time.
Perfect phylogeny and ultrametric problem

Perfect phylogeny can be solved using ultrametric tree.

**Definition.** Let $n$ be the number of taxa and $m$ be the number of traits. Given $n \times m$ matrix $M$, define $n \times n$ matrix $D_M$: $D_M(p, q)$ is the number of characters both $p$ and $q$ possess. $D_M(p, q)$ equals the number of columns $i$ such that $M(p, i) = M(q, i) = 1$.

**Lemma.** If $M$ has a perfect phylogeny then $D_M$ is a min-ultrametric matrix.

**Proof.** Let $T$ be a perfect phylogeny for $M$. Label nodes of $T$ in top-down manner: label root as 0 and label a node $v$ with the number of its parent $u$ plus the number of characters labeling $(u, v)$. The lowest common ancestor of $p$ and $q$ has label $D_M(p, q)$. □

Is it true that $M$ has a perfect phylogeny if $D_M$ is a min-ultrametric matrix? No.

**Algorithm**

1. Create $D_M$ from $M$.
2. Test if $D_M$ is min-ultrametric. If not then print “$M$ has no perfect phylogeny” and return. Otherwise compute a min-ultrametric tree $T'$ for $D_M$.
3. Try to label the edges of $T'$ with $m$ characters, converting $T'$ into a perfect phylogeny for $M$. If this fails then $M$ has no perfect phylogeny; otherwise $T''$ is the perfect phylogeny.