How are these trees built from DNA sequences?

- Leaves represent existing species
- Internal vertices represent ancestors
- Root represents the oldest evolutionary ancestor

Today: Compatible characters and perfect phylogeny
Character-Based Tree Reconstruction

Mouth character changes twice.
Both mouth and eyes characters change once.

Contradictory Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Lizard</th>
<th>Frog</th>
<th>Human</th>
<th>Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hair</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Single bone in lower jaw</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Lactation</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Tail</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Hypotheses:
1. Tree is incorrect: Humans are more closely related to frogs than dogs.
2. Tails evolved more than once on tree.
Homoplasy

- **Homoplasy**: Independent (or parallel) evolution of same/similar characters.
- A character exhibits *homoplasy* if it changes to the same state more than once on the tree.

<table>
<thead>
<tr>
<th>Character</th>
<th>Lizard</th>
<th>Frog</th>
<th>Human</th>
<th>Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hair</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single bone in lower jaw</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lactation</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tail</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Compatibility**

**Def**: A character is compatible with a tree if it can evolve on the tree with no homoplasy.

**Def**: A character $c$ with $k$ possible states is *compatible* with a tree $T$ with labeled leaves if and only if there is a labeling of the internal nodes such that the total number of changes of $c$ is exactly $k - 1$.

[Must exhibit all $k$ states but can only change to each state at most once.]

Special case: *binary* (2-state) character $c$, only one change of state. (along a single edge of $T$)
Small Compatibility Problem

**Input:** A tree $T$ with labeled leaves

**Output:** Maximum number of compatible characters, over all possible labelings of internal nodes.

Solve using Fitch’s algorithm from last time.

Large Compatibility Problem

**Input:** Character Matrix $M$.

**Output:** Tree $T$ with maximum number of compatible characters.

```
traits   1 2 3 4 5
species  A 1 1 0 0 0
         B 0 0 1 0 0
         C 1 1 0 1 0
         D 0 0 1 0 1
         E 1 0 0 0 0
```

Binary character state matrix
Character Compatibility

**Def:** Two characters $c_1$ and $c_2$ in a set $S$ of species, are said to be **pairwise compatible** if there exists a tree $T$ such that both $c_1$ and $c_2$ are compatible with $T$.

**Def.** Characters $c_1, \ldots, c_k$ are **mutually compatible** if there exists a tree $T$ such that $c_i$ is compatible with $T$ for all $i$.

Example

- **Character matrix:**

<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>
</table>
  A  | 1 | 0 | 0 | 1 | 1 | 0 |
  B  | 0 | 0 | 1 | 0 | 0 | 0 |
  C  | 1 | 1 | 0 | 0 | 0 | 0 |
  D  | 1 | 1 | 0 | 1 | 1 | 1 |
  E  | 0 | 0 | 1 | 1 | 1 | 0 |
  F  | 0 | 0 | 0 | 0 | 0 | 0 |

- **Diagram:**

  - Character 1 is compatible w/ $T$.
  - Character 2 is compatible w/ $T$.
  - Thus, 1 and 2 are pairwise compatible.

From Felsenstein.
Binary characters

1. Simple test for pairwise compatibility.
2. Pairwise compatibility $\leftrightarrow$ mutual compatibility.

### Binary character state matrix

<table>
<thead>
<tr>
<th>traits</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>1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Binary characters

- A character $i$ partitions the set of species into two sets $i_0$ and $i_1$
- A character is **homogeneous** w.r.t a set $S$ of species, if it has the same value for all species. Otherwise, it is **heterogeneous**.
- Example:
  - $i$ is homogeneous w.r.t $\{A, B, C\}$
  - $i$ is heterogeneous w.r.t $\{A, D, E\}$
Pairwise Compatibility Test
(Wilson 1965)

Binary characters $i$ and $j$ are pairwise compatible if and only if:

- $j$ is homogenous w.r.t $i_0$ or $i_1$.

Equivalently:

- $i_1$ and $j_1$ are disjoint or one contains the other

Equivalently:

- all 4 rows do not exist
- (0,0), (0,1), (1,0), (1,1)

<table>
<thead>
<tr>
<th>$i$</th>
<th>$j$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0</td>
<td>A 0</td>
<td>A 0</td>
</tr>
<tr>
<td>B 0</td>
<td>B 0</td>
<td>B 0</td>
</tr>
<tr>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
</tr>
<tr>
<td>D 1</td>
<td>D 0</td>
<td>D 1</td>
</tr>
<tr>
<td>E 1</td>
<td>E 0</td>
<td>E 0</td>
</tr>
</tbody>
</table>

Proof:

Suppose $i$ and $j$ are pairwise compatible. There is a tree $T$ where $i$ and $j$ change state once.

Change of state for $i$ creates two subtrees of $T$ with leaves $i_0$ and $i_1$.

$j$ changes state only once on $T$, so

- $j_1 \subseteq i_1$ or $j_1 \subseteq i_0$.

Thus,

- $j$ is homogenous w.r.t $i_0$ or $i_1$. 
Pairwise Compatibility Test  
(Wilson 1965)

Proof:
Conversely suppose $j$ is homogenous w.r.t $i_0$. (analogous proof for $i_1$)
$i$ and $j$ are compatible with following tree:

Pairwise Compatibility Theorem  
(Estabrook et al. 1976)

A set $S$ of binary characters is mutually compatible if and only if all pairs $c$ and $c'$ of characters in $S$ are pairwise compatible.

Pairwise compatibility $\iff$ mutual compatibility.
Compatibility Graph

• Compatibility graph $G = (V, E)$
  $V = \{\text{characters}\}$
  $E = \{(u, v) : u \text{ and } v \text{ are pairwise compatible}\}$

• Example:

<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>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Compatibility Graph

• Compatibility graph $G = (V, E)$
  $V = \{\text{characters}\}$
  $E = \{(u, v) : u \text{ and } v \text{ are pairwise compatible}\}$

• A graph is **complete** provided all possible edges are present.

• $m$ pairwise compatible characters give complete graph $K_m$

![Graphs K3, K4, K5]
Graphs and Cliques

• A subgraph $G'$ of a graph $G = (V, E)$ is a graph $G' = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$ is all edges of $E$ that join vertices in $V'$.

• A subgraph that is a complete graph is called a clique.

• By Pairwise Compatibility Theorem:
  Set $S$ of mutually compatible binary characters
  $\rightarrow$ all pairwise compatible.
  $\rightarrow$ form a clique in compatibility graph $G$.

Example

• $\{4, 5, 6\}$ are mutually compatible.
• $\{1, 2, 3, 6\}$ are mutually compatible.

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
A & 1 & 0 & 0 & 1 & 1 & 0 \\
B & 0 & 0 & 1 & 0 & 0 & 0 \\
C & 1 & 1 & 0 & 0 & 0 & 0 \\
D & 1 & 1 & 0 & 1 & 1 & 1 \\
E & 0 & 0 & 1 & 1 & 1 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Maximum Cliques

• By Pairwise Compatibility Theorem:
  Set S of mutually compatible binary characters \( \rightarrow \) all pairwise compatible \( \rightarrow \) form a clique in compatibility graph G.

• Largest set of compatible characters is clique with largest number of vertices: *Maximum Clique Problem*.

• Maximum Clique Problem is NP-hard. No efficient algorithms exist.

*Not same as maximal clique.*

Large Compatibility Problem

• **Input:** Character Matrix M.

• **Output:** Tree T with *maximum* number of compatible characters.

<table>
<thead>
<tr>
<th>traits</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>1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Solving this problem gives the maximum clique in compatibility graph. In fact, problem is NP-hard, even for binary characters.
Perfect Phylogeny

A set of mutually compatible binary characters gives a **perfect phylogeny**:

1. **Evolutionary model**
   - Binary characters \{0,1\}
   - Each character changes state only once in evolutionary history (no homoplasy!).

2. **Tree in which every mutation is on an edge of the tree.**
   - All the species in one sub-tree contain a 0, and all species in the other contain a 1.
   - For simplicity, assume root = (0, 0, 0, 0)

**How do we reconstruct such a tree?**

---

4 Gamete Condition

There exists a perfect phylogeny if and only if for all pair of columns \((i, j)\), \(j\) is homogenous w.r.t \(i_0\) or \(i_1\).

Equivalently,

There exists a perfect phylogeny if and only if for all pairs of columns \((i, j)\), the following 4 rows do **not** exist:

\((0,0), (0,1), (1,0), (1,1)\)

This is the **pairwise compatibility test** for all pairs of columns.
4-gamete condition: proof

(only if) Every perfect phylogeny satisfies the 4-gamete condition

• Depending on which edge the mutation $j$ occurs, either $i_0$, or $i_1$ should be homogenous.

(if) If the 4-gamete condition is satisfied, does a perfect phylogeny exist?

Need to give an algorithm...

Perfect Phylogeny Algorithm

• **Input**: $n \times m$ character matrix $M$ with $m$ mutually compatible binary characters.

• **Output**: Phylogenetic tree $T$ with leaves labeled by characters such that each character is compatible with $T$. 
Perfect Phylogeny Algorithm

- **Idea**: Each character $i$ splits species into two sets $i_0$ and $i_1$.
- Iteratively build tree by repeatedly subdividing species.

Perfect Phylogeny (Compatibility Tree) Algorithm

Iteratively build tree. Each vertex $v$ has associated label set $L_v$ containing species that are descendents.

Initialization: $T \leftarrow \{r\}$. $L_r = \{1, \ldots, n\}$.

While $M$ has at least one column
  - Choose column $i$ of $M$
  - Find a labeled node $v$ in $V(T)$ such that $|L_v \cap i_0| \neq 0$ and $|L_v \cap i_1| \neq 0$.
  - Add vertices $v_0$ and $v_1$ and edges $(v, v_0)$ and $(v, v_1)$ to $T$.
  - Set $L_{v0} = L_v \cap i_0$, $L_{v1} = L_v \cap i_1$.
  - Set $L_v = \text{“empty set”}$.
  - Delete column $i$ from $M$
Example

<table>
<thead>
<tr>
<th>species</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>traits</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$L_v = \{A,C,E\}$
$L_v = \{B,D\}$

Improved Algorithm

Can sort columns of $M$ to improve Find step.

Initialization: $T \leftarrow \{r\}$. $L_r = \{1, \ldots, n\}$.

While $M$ has at least one column

Choose column $i$ of $M$

Find a labeled node $v$ in $V(T)$ such that

$|L_v \cap i_0| \neq 0$ and $|L_v \cap i_1| \neq 0$.

Add vertices $v_0$ and $v_1$ and edges $(v, v_0)$ and $(v, v_1)$ to $T$.

Set $L_{v_0} = L_v \cap i_0$, $L_{v_1} = L_v \cap i_1$.

Set $L_v$ = “empty set”.

Delete column $i$ from $M$
Inclusion Property

• For any pair of columns $i, j$: $i < j$ if and only if $i_1 \subseteq j_1$

• Note that if $i < j$ then the edge containing $i$ is an ancestor of the edge containing $j$

Sort columns

• Sort columns according to the inclusion property: $i < j$ if and only if $i_1 \subseteq j_1$

• This can be achieved by considering the columns as binary representations of numbers (most significant bit in row 1) and sorting in decreasing order
Algorithm Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sankoff’s &amp; Fitch’s Alg.</td>
<td>Characters, T</td>
<td>A, B</td>
</tr>
<tr>
<td>Perfect Phylogeny</td>
<td>Characters</td>
<td>A, B, T</td>
</tr>
<tr>
<td>Felsenstein</td>
<td>Characters, T, B</td>
<td>A</td>
</tr>
</tbody>
</table>

T = tree topology
B = branch lengths
A = ancestral states

Examples

- Want to find characters with no homoplasy.
- Difficult for morphological or behavioral characters; e.g. tails.
- Also difficult for many molecular characters:
  DNA: A nucleotide {A, C, T, G} mutates only once?
- Some molecular characters exhibit low/no homoplasy.
(A) A schematic diagram of syntenic regions in three species, with a repeat insertion in A, B, and C


©2005 by Cold Spring Harbor Laboratory Press

(Retro)transposons

Cell Division and Mutation

DNA Replication → Mitosis → Two Diploid Cells

Unlikely to occur at exact same position twice.
Reversals: Biology

Break and Invert

5’ ATGCCCTGATACTA 3’
3’ TACGGACATGAT 5’

5’ ATGTACAGCTA 3’
3’ TACATGTCGAT 5’

Reversals

Blocks represent conserved genes.
Reversals

Blocks represent conserved genes.
In the course of evolution or in a clinical context, blocks 1, 2, 3, -8, -7, -6, -5, -4, 9, 10 could be incorrectly copied as 1, 2, 3, -8, -7, -6, -5, -4, 9, 10.

Reversals and Breakpoints

The reversion introduced two breakpoints (disruptions in order).
Mammalian Phylogeny from (micro)inversions

Chaisson M. J. et al. PNAS 2006;103:19824-19829
©2006 by National Academy of Sciences