Phylogenetics and phylogenomics: sequences, trees, networks and other combinatorial objects

Céline Scornavacca

ISE-M, Equipe Phylogénie & Evolution Moléculaires

May 13th, 2014
From Aristotle to Darwin

Since Aristotle, naturalists have always tried to classify the abundance of creatures that populate the Earth.

- **Aristote**: the *scala naturae*;
From Aristotle to Darwin

Since Aristotle, naturalists have always tried to classify the abundance of creatures that populate the Earth.

- **Aristote**: the *scala naturae*;
- **Carl von Linné**: classification of living;
From Aristotle to Darwin

Since Aristotle, naturalists have always tried to classify the abundance of creatures that populate the Earth.

- **Aristote**: the *scala naturae*;
- **Carl von Linné**: classification of living;
- **Antoine Laurent de Jussieu**;
From Aristotle to Darwin

Since Aristotle, naturalists have always tried to classify the abundance of creatures that populate the Earth.

- **Aristote**: the *scala naturae*;
- **Carl von Linné**: classification of living;
- **Antoine Laurent de Jussieu**;
- **Leclerc de Buffon**: the first to evoke the possibility that species can evolve;
From Aristotle to Darwin

Since Aristotle, naturalists have always tried to classify the abundance of creatures that populate the Earth.

- Aristote: the *scala naturae*;
- Carl von Linné: classification of living;
- Antoine Laurent de Jussieu;
- Leclerc de Buffon: the first to evoke the possibility that species can evolve;
- Jean-Baptiste Lamarck: first theory of evolution;
From Aristotle to Darwin

Since Aristotle, naturalists have always tried to classify the abundance of creatures that populate the Earth.

- **Aristote**: the *scala naturae*;
- **Carl von Linné**: classification of living;
- **Antoine Laurent de Jussieu**;
- **Leclerc de Buffon**: the first to evoke the possibility that species can evolve;
- **Jean-Baptiste Lamarck**: first theory of evolution;
- **Charles Darwin**: *The Origins of Species* (1859).
It is a truly wonderful fact . . . that all animals and all plants throughout all time and space should be related to each other in groups, subordinate to groups. [...]

The affinities of all the beings of the same class have sometimes been represented by a great tree. [...] The green and budding twigs may represent existing species; and those produced during former years may represent the long succession of extinct species.
Tree and inter-species relationships

Others before Darwin used trees to illustrate species classifications in light of fixism or descent of some species from others.
Tree and inter-species relationships

The originality of Darwin’s tree is the coexistence, in the same figure, of the concepts of time and descent: the bifurcations in the tree follow one another over time.
Phylogenetics aims at clarifying, using molecular and morphological data, the evolutionary relationships that exist among different species. These relationships can be represented through phylogenetic trees or phylogenies.

Woese 1987; Barns et al. 1996; Brown et Doolittle 1997
Applications

TOL – Tree Of Life
Applications

Study character evolution
Applications

Ecology: migrations, host-parasite relationships, ...
Recurrent example: Did Dr David Acer contaminate his patients?

Ou et al. (1992), Page et Holmes (1998)
Rooted phylogenetic trees ...

... are oriented connected and acyclic graphs, where terminal nodes are associated to a set of species:

- the leaves or taxa represent extant organisms
- internal nodes represent hypothetical ancestors
- the only node without ancestor is called root
- each internal node represents the lowest common ancestor of all taxa below it (clade)
- nodes and branches can have several kinds of information associated with them, such as time or amount of evolution estimates.
Unrooted phylogenetic trees …

… connected and acyclic graphs, where terminal nodes are associated to a set of species.
An unrooted binary tree with $n$ leaves, with $n \geq 3$, has:

- $2n - 2$ nodes;
- $2n - 3$ branches;
- $n - 3$ internal branches.
Some numbers

An unrooted binary tree with \( n \) leaves, with \( n \geq 3 \), has:
- \( 2n - 2 \) nodes;
- \( 2n - 3 \) branches;
- \( n - 3 \) internal branches.

A rooted binary tree with \( n \) leaves, with \( n \geq 3 \), has:
Some numbers

An unrooted binary tree with $n$ leaves, with $n \geq 3$, has:
- $2n - 2$ nodes;
- $2n - 3$ branches;
- $n - 3$ internal branches.

A rooted binary tree with $n$ leaves, with $n \geq 3$, has:
- $2n - 1$ nodes;
- $2n - 2$ branches;
- $n - 2$ internal branches.
How many trees?

- How many unrooted trees with \( n \) leaves, with \( n \geq 3 \)?

\[
U(n) = (2^n - 5)!! = 3 \cdot 5 \cdot 7 \cdot ... \cdot (2n - 5).
\]

- How many rooted trees with \( n \) leaves, with \( n \geq 3 \)?

\[
R(n) = (2^n - 3)!! = U(n) \cdot (2^n - 3) = 3 \cdot 5 \cdot 7 \cdot ... \cdot (2n - 3).
\]
How many trees?

- How many unrooted trees with \( n \) leaves, with \( n \geq 3 \)?
  \[
  U(n) = (2n - 5)!! := 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n - 5).
  \]
How many trees?

- How many unrooted trees with $n$ leaves, with $n \geq 3$?
  - $U(n) = (2n - 5)!! := 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n - 5)$.
- How many rooted trees with $n$ leaves, with $n \geq 3$?
How many trees?

- How many unrooted trees with \( n \) leaves, with \( n \geq 3 \)?
  
  \[ U(n) = (2n - 5)!! := 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n - 5). \]

- How many rooted trees with \( n \) leaves, with \( n \geq 3 \)?
  
  \[ R(n) = (2n - 3)!! = U(n) \cdot (2n - 3) = 3 \cdot 5 \cdot \ldots \cdot (2n - 3). \]
How many trees?

- How many unrooted trees with \( n \) leaves, with \( n \geq 3 \)?
  \[ U(n) = (2n - 5)!! := 3 \cdot 5 \cdot 7 \cdot \ldots \cdot (2n - 5). \]

- How many rooted trees with \( n \) leaves, with \( n \geq 3 \)?
  \[ R(n) = (2n - 3)!! = U(n) \cdot (2n - 3) = 3 \cdot 5 \cdot \ldots \cdot (2n - 3). \]

For \( n = 10 \), \( U(n) = 2.027 \times 10^6 \) and \( R(n) = 34.459 \times 10^6 \).
Phylogenetics reconstruction

The first phylogenies were based on:

- the comparison of **morphological** traits (fossils);
- ontogeny;
- the comparison of behaviors;
- geographical repartitions...
Phylogenetics reconstruction

The first phylogenies were based on:

- the comparison of morphological traits (fossils);
- ontogeny;
- the comparison of behaviors;
- geographical repartitions...
Phylogenetics reconstruction

With the discovery of DNA by Watson and Crick in 1953 and the design of sequencing techniques, a new kind of information became available: molecular data.

Today, phylogenies are obtained by studying:

- molecular sequences;
- discrete characters;
- gene frequencies;
- restriction sites;
- microsatellites;
- ...

## Molecular phylogenetics

<table>
<thead>
<tr>
<th></th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balene</td>
<td>AAAATCAATTTAACAACACCAAGCCGCTGCGATATGACAACGAGGAAAAAGAAGCGCGACGCGGCGG</td>
</tr>
<tr>
<td>Homme</td>
<td>CCAATCAATTTAACAACACCAAGCCGCTGCGATATGACAACGAGGAAAAAGAAGCGCGACGCGGCGG</td>
</tr>
<tr>
<td>Lion</td>
<td>CCAATCAATTTAACAACACCAAGCCGCTGCGATATGACGCGACGCGGCGGCGG</td>
</tr>
<tr>
<td>Dauphin</td>
<td>AAAATCAATTTAACAACACCAAGCCGCTGCGATATGACGCGACGCGGCGGCGG</td>
</tr>
<tr>
<td>Vache</td>
<td>CCAATCAATTTAACAACACCAAGCCGCTGCGATATGACGCGACGCGGCGGCGG</td>
</tr>
</tbody>
</table>

Diagram showing evolutionary relationships among species with question mark indicating uncertainty.
Molecular data – Characters

- Gene or protein alignements:
  - example: 3 taxa on 20 characters and 5 states (A, C, G, T, -)

  | Species A    | ATGGCTATTC-TATAGTACG |
  |Species B    | ATCGCT-GTCTTATATTACA |
  |Species C    | TTCACT–ACCTGTGGTCCA   |

- taxa → lines of the matrix, characters → columns.

- Morphological characters:

|          | gills | fins | teeth | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>whale</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>...</td>
</tr>
<tr>
<td>shark</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>...</td>
</tr>
</tbody>
</table>
Molecular data – Distances

- For \( n \) taxa, a matrix \( D \) with \( n \) lines and \( n \) columns;
- case \( d_{i,j} \) contains the estimation of the evolutionary distance between taxa \( i \) et \( j \);
- examples of distances:
  - Hamming distance;
  - edit (PAM matrices, Dayhoff, Blosum);
  - distances in terms of genome rearrangements (inversions, translocations and substitutions);
- loss of information w.r.t. reconstruction based on an alignment, but permits to use fast algorithms.

\[
\begin{array}{ccccccc}
A & B & C & D & E & F \\
A & 0 & 3 & 3 & 5 & 5 & 4 \\
B & 3 & 0 & 3 & 5 & 5 & 4 \\
C & 3 & 3 & 0 & 4 & 4 & 3 \\
D & 5 & 5 & 4 & 0 & 2 & 3 \\
E & 5 & 5 & 4 & 2 & 0 & 3 \\
F & 4 & 4 & 3 & 3 & 3 & 0 \\
\end{array}
\]
The 4 big steps of phylogenetics reconstruction

1. Sélectionner les données
2. Aligner les séquences

Les méthodes basées caractères
- MB
  - Modèle?
- ML
  - Modèle?
- MP
  - Poids? (sites, substitutions)?

Les méthodes de distances
- Calcul de distances (Quel modèle?)
  - Optimisation de critères
    - LS  ME
    - NJ  PGM
- Arbre Unique

3. Choix de la méthode

4. Calculer ou estimer l’arbre qui traduit mieux les données

5. Effectuer des tests statistiques de robustesses

4. Comparer les phylogénies obtenues

Adapté de Hillis et al., (1993)
STEP 1: Choose the data - homology

- We should compare only characters that are similar because they have been inherited from a common ancestor (**homology**) and not characters that are similar because of convergent evolution (**homoplasy**):
  - bat and bird wings (as **wings** – homoplasy, their common ancestor cannot fly);
STEP 1: Choose the data - homology

- We should compare only characters that are similar because they have been inherited from a common ancestor (homology) and not characters that are similar because of convergent evolution (homoplasy):
  - bat and bird wings (as wings – homoplasy, their common ancestor cannot fly);
  - bat and bird wings (as forearms – homology).
STEP 1: Choose the data - homology

- The same thing apply to molecular characters:
  - 2 proteins in two species can be encoded by the same gene inherited from a common ancestor (homology);
  - 2 genes can be similar and have a similar function acquired independently (homoplasy).
STEP 1: Choose the data - homology

- The same thing apply to molecular characters:
  - 2 proteins in two species can be encoded by the same gene inherited from a common ancestor (homology);
  - 2 genes can be similar and have a similar function acquired independently (homoplasy).

- Several genes can belong to a gene family (homologues and not orthologues, because of duplications).
STEP 2: Align the sequences

- In molecular sequence evolution, we not only have substitutions but also insertions and deletions of certain sites. This leads to sequences with different lengths.
- To compare the same character in all sequences, we have to align them.
- Several methods have been proposed to do so (dynamic programmation, heuristics, probabilistic methods...).
STEP 2: Align the sequences

- In molecular sequence evolution, we not only have substitutions but also insertions and deletions of certain sites. This leads to sequences with different lengths.
- To compare the same character in all sequences, we have to align them.
- Several methods have been proposed to do so (dynamic programmation, heuristics, probabilistic methods...).

A bad alignment strongly affects the phylogenetics reconstruction

There is a big effort in producing good alignments
The alignment problem (for two sequences)

- An alignment creates a correspondence between two sequences to identify "homologous" characters.

<table>
<thead>
<tr>
<th>R</th>
<th>D</th>
<th>I</th>
<th>S</th>
<th>L</th>
<th>V</th>
<th>−</th>
<th>−</th>
<th>−</th>
<th>K</th>
<th>N</th>
<th>A</th>
<th>G</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>I</td>
<td>−</td>
<td>L</td>
<td>V</td>
<td>S</td>
<td>D</td>
<td>A</td>
<td>K</td>
<td>N</td>
<td>V</td>
<td>G</td>
<td>I</td>
</tr>
</tbody>
</table>
The alignment problem (for two sequences)

An alignment creates a correspondence between two sequences to identify "homologous" characters.

Three operations are permitted:
  - substitutions;
  - insertions [indels];
  - deletions [indels];
The alignment problem (for two sequences)

- An alignment creates a correspondence between two sequences to identify "homologous" characters.

- Three operations are permitted:
  - substitutions;
  - insertions [indels];
  - deletions [indels];

- More formally:
  - **INPUT**: Two sequences and a cost scheme.
  - **OUTPUT**: Quantify and localize the similarity: score + alignment.
The formal definition of an alignment

For two sequences $X = \{X_1, \cdots, X_n\}$ and $Y = \{Y_1, \cdots, Y_n\}$, a global alignment between $X$ and $Y$ is a matrix with 2 lines and $L$ columns, with $\max(m, n) \leq L \leq n + m$, where:

- each columns is of the type $(X_i, Y_j)$, $(X_i)$ or $(-, Y_j)$, with $1 \leq i \leq n$, $1 \leq j \leq m$;
- for each column with $(X_i, Y_j)$, the next column has to be $(X_{i+1}, Y_j)$, $(X_{i+1})$ or $(-, Y_{j+1})$. 


The formal definition of an alignment

For two sequences \( X = \{X_1, \cdots, X_n\} \) and \( Y = \{Y_1, \cdots, Y_n\} \), a global alignment between \( X \) and \( Y \) is a matrix with 2 lines and \( L \) columns, with \( \max(m, n) \leq L \leq n + m \), where:

- each columns is of the type \((X_i, Y_j)\), \((-\_i)\) or \((-\_j)\), with \(1 \leq i \leq n\), \(1 \leq j \leq m\);
- for each column with \((X_i, Y_j)\), the next column has to be \((X_{i+1}, -\_j)\), \((X_{i+1}, Y_{j+1})\) or \((-\_j)\).

Note that, depending on the number of symbols \(\_\_\), the number of columns may vary.
Identity/substitution: positive /negative scores:
- a similarity scoring matrix \( s \);
- \( s(a, b) \) is the score of aligning \( a \) with \( b \).

Identity/substitution: positive /negative scores:
- elementary scoring: one for each indel;
- complex scoring: affine, logarithmic.

Score of an alignment: sum of all elementary events composing it.
Alignment problem - a naive solution

- Sequences: \( X = ACGCTATC \) and \( Y = ACTGTAATG \), with scores -1, -2, 2, respectively for indel, substitution and identity.
Dynamic programming

- stores the sub solutions in a matrix;
- avoids to recompute several times the same sub solutions;
- sub solutions = score of the alignment between prefixes.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>C</td>
<td>G</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Creation of a table indexed by the two sequences

Case \((i, j)\) : score of aligning the first \(i\) characters of \(ACTGTAATG\) and the first \(j\) characters of \(ACGGCTATC\)
Dynamic programing

If $s$ is the similarity scoring matrix, $g$ is the score of an indel
d. Initial conditions: $M(0, 0) = 0$, $M(0, j) = g \times j$, $M(i, 0) = g \times i$
d. Recursion:

$$M(i, j) = \max \begin{cases} 
M(i - 1, j - 1) + s(X_i, Y_j) & \text{idéntité/substitution} \\
M(i - 1, j) + g & \text{déletion} \\
M(i, j - 1) + g & \text{insertion}
\end{cases}$$
Alignment problem - an example

- Case (9,9) : score of the global alignment between ACGGCTATC and ACTGTAATG, with scores -1,-2, 2, respectively for indel, substitution and identity

```
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
<td>-6</td>
<td>-7</td>
<td>-8</td>
<td>-9</td>
</tr>
<tr>
<td>A</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>C</td>
<td>-2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>T</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>-4</td>
<td>-1</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>-5</td>
<td>-2</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>-6</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>-7</td>
<td>-4</td>
<td>-1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>T</td>
<td>-8</td>
<td>-5</td>
<td>-2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>-9</td>
<td>-6</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
```
Backtracking

- finds the alignment, given the matrix;
- works by:
  - starting at the lowest cell of the last column;
  - moving to a neighbor cell with maximum score such that the score of the corresponding mutation + the score of the cell we want to move in gives the score of the cell where we are;
  - doing this until we arrive at cell (0, 0).
- If we can backtrack in several ways, there exist several optimal solutions.
- Overall, a $O(n \times m)$ complexity (including matrix computation).
The backtracking - an example

- Sequences: $X = ACGCTATC$ and $Y = ACTGTAATG$, with scores -1, -2, 2, respectively for indel, substitution and identity
More on the alignment problem

- **local alignment**: alignment of the best subsequences;
- **different gap** (i.e. / insertions or deletions in a row) scoring schemes
- **heuristics**, e.g. BLAST et FASTA;
- **Multiple Sequence Alignment or MSA** (generalization in \(O(k^22^kn^k)\), NP-complet)
Reconstruction of phylogenies for one dataset

- character-based methods, which retrieve similarities comparing the states taken by species at different characters:
  - without modeling sequence evolution:
    - parsimony methods (e.g. Fitch parsimony)
  - using models of sequence evolution:
    - likelihood methods (e.g. methods implemented in PhyML or RaxML)
    - bayesian methods (e.g. methods implemented in MrBayes or BEAST)

- distance-based methods, which use pairwise distances to quantify the amount of evolution separating species (e.g. BIONJ, FASTME)
Parsimony methods

- **General principle:** Given a set $S$ of $n$ character sequences (one per studied species) $S_1, \cdots, S_n$ of length $m$ and a cost for each transformation $C_{x \rightarrow y}$, find the most parsimonious phylogeny(ies).
  - Fitch parsimony ($C_{x \rightarrow y} = 1$), Sankoff parsimony ($C_{x \rightarrow y} = C_{y \rightarrow x}$), Dollo’s, Camin-Sokal’s ...

- If the tree $T$ is given, the problem of computing $P(S|T)$ is easy:
  - each character can be analyze independently, so
    $P(S|T) = \sum_{j=1}^{m} w_j P(c_j|T)$, where $P(c_j|T)$ is the sum of $P(c_j|e)$ over all branches $T$, weighted by the corresponding substitution costs $C_{x \rightarrow y}$;
  - the choice of the root does not change the parsimony value $P(S|T)$ if $C_{x \rightarrow y} = C_{y \rightarrow x}$ (usually true);
  - the number of possible states for a character is limited.
(1) compute $P(S|T)$: an $O(nm)$ algorithm to compute $P(S|T)$ has been proposed by Fitch in 1971.
Parsimony methods - an example

(1) compute $P(S|T)$: an $O(nm)$ algorithm to compute $P(S|T)$ has been proposed by Fitch in 1971.

(2) compute the best tree:

- find the tree minimizing $P(S|T)$ is an NP-hard problem;
- parsimony methods are not consistent.

Figure 1.3: Most parsimonious reconstructions per sites for the set of sequences $S$ given the phylogeny $T$ - Two equally parsimonious reconstructions are possible for site 2. Deduced internal characters are shown between square brackets.

The main drawback of parsimony methods is that they are not consistent [Cavender, 1978; Felsenstein, 1978]. A method is said to be consistent if the probability to obtain the correct tree converges to one as more and more data are analyzed. For example, parsimony methods are not robust to long branch attractions i.e., when rapidly evolving species that had a separated evolution are inferred to be closely related, regardless of their true evolutionary relationships [e.g., Felsenstein, 1978, see Section 2.1.3]. Indeed, when molecular sequences from two species evolve rapidly, the probability that the same nucleotide appears in both two sequences at the same site increases. When this happens, the most parsimonious scenario is a wrong one, where the two species evolved from a common ancestor. As a matter of fact, rapid evolving species accumulate numerous mutations on a single character and contradict the very foundations of the parsimony approach. For a review of other objections to parsimony methods see Sober [1998].
Nucleotide models

Common hypothesis:
- sequences evolve exclusively through nucleotide substitutions, no INDELs;
- each site is a i.i.d. variable;
- substitution process is a first-order Markov model;
- substitution process is homogeneous;
- substitution process is stationary;
- the substitution probability during an infinitesimal time interval $dt$ is proportional to $dt$;
- there is at most one substitution per infinitesimal time interval $dt$.

These models are characterized by a 4x4 $Q$ matrix, where $Q_{xy}$ is the frequency to which base $x$ substitute to base $y$:

$$Q = \begin{pmatrix}
\lambda_A & Q_{AC} & Q_{AG} & Q_{AT} \\
Q_{CA} & \lambda_C & Q_{CG} & Q_{CT} \\
Q_{GA} & Q_{GC} & \lambda_G & Q_{GT} \\
Q_{TA} & Q_{TC} & Q_{TG} & \lambda_T
\end{pmatrix}$$
Nucleotide models

JC

\[ Q = \begin{pmatrix}
-3\alpha & \alpha & \alpha & \alpha \\
\alpha & -3\alpha & \alpha & \alpha \\
\alpha & \alpha & -3\alpha & \alpha \\
\alpha & \alpha & \alpha & -3\alpha
\end{pmatrix} \]

K2P

\[ Q = \begin{pmatrix}
-\alpha - 2\beta & \beta & \alpha & \beta \\
\beta & -\alpha - 2\beta & \beta & \alpha \\
\alpha & \beta & -\alpha - 2\beta & \beta \\
\beta & \alpha & \beta & -\alpha - 2\beta
\end{pmatrix} \]

GTR

\[ Q = \begin{pmatrix}
\lambda_A & \pi C \lambda_{AC} & \pi G \lambda_{AG} & \pi T \lambda_{AT} \\
\pi A \lambda_{AC} & \lambda_C & \pi G \lambda_{CG} & \pi T \lambda_{CT} \\
\pi A \lambda_{AG} & \pi C \lambda_{CG} & \lambda_G & \pi T \lambda_{GT} \\
\pi A \lambda_{AT} & \pi C \lambda_{CT} & \pi G \lambda_{GT} & \lambda_T
\end{pmatrix} \]
Nucleotide models

GTR

Equal base frequencies

TrN

3 substitution types (transversions, 2 transition classes)

Sym

3 substitution types (transitions, 2 transversion classes)

HKY85

Equal base frequencies

F84

2 substitution types (transitions vs. transversions)

K3ST

2 substitution types (transitions vs. transversions)

F81

Single substitution type

K2P

Single substitution type

JC

Equal base frequencies
Nucleotide models

More complex models

- protein models;
- codon models;
- ...

Likelihood methods

- **General principle:** Given a set $S$ of $n$ character sequences (one per studied species) $S_1, \cdots, S_n$ of length $m$, find the most likelyhood phylogeny(ies).

- If the parameter vector $\theta$ (evolutionary model, topology $T$ + branch lengths $\omega$) is given, the problem of computing $\mathbb{P}(S|\theta)$ is easy:
  - each character can be analyze independently, so
    \[
    \mathbb{P}(S|\theta) = \prod_{j=1}^{m} \mathbb{P}(c_j|\theta);
    \]
  - the choice of the root does not change the parsimony value $\mathbb{P}(S|\theta)$ if the model is reversible (usually true);
  - the number of possible states for a character is limited.
Likehood methods - an exemple

(1) compute $\mathbb{P}(S|\theta)$: an $O(nm)$ algorithm to compute $\mathbb{P}(S|\theta)$ has been proposed by Felsenstein in 1981.

where

$$P \left( A_i \right)$$

since $P(x, y)(t)$ is equal to $\frac{1}{4}(1 - e^{-4\alpha t})$ if $x \neq y$ and $\frac{1}{4}(1 + 3e^{-4\alpha t})$ otherwise.
Likehood methods - an exemple

(1) compute $\mathbb{P}(S|\theta)$: an $O(nm)$ algorithm to compute $\mathbb{P}(S|\theta)$ has been proposed by Felsenstein in 1981.

where

$P(A_i | A, C) = P(X | A, C)$

since $P(x, y)(t)$ is equal to $\frac{1}{4}(1 - e^{-4\alpha t})$ if $x \neq y$ and $\frac{1}{4}(1 + 3e^{-4\alpha t})$ otherwise.

$= P(A)P_{e_1}(A, C)P_{e_2}(A, C)P_{e_3}(A, A)P_{e_4}(A, A)P_{e_5}(A, G)$
Likehood methods - an exemple

(1) compute $\mathbb{P}(S|\theta)$: an $O(nm)$ algorithm to compute $\mathbb{P}(S|\theta)$ has been proposed by Felsenstein in 1981.

where $P(A_i|\theta) = P(A_i|E_1(E_2(E_3(A, C)P_{e_3}(A, A)P_{e_4}(A, A)P_{e_5}(A, G)) = P(A)P_{e_1}(A, C)P_{e_2}(A, C)P_{e_3}(A, A)P_{e_4}(A, A)P_{e_5}(A, G)$

$$= \frac{1}{4}(1 - e^{-4\alpha t})$$ if $x \neq y$ and $\frac{1}{4}(1 + 3e^{-4\alpha t})$ otherwise.

$$= \frac{1}{4^6}(1 - e^{-4\alpha \omega_1})(1 - e^{-4\alpha \omega_2})(1 + 3e^{-4\alpha \omega_3})(1 + 3e^{-4\alpha \omega_4})(1 - e^{-4\alpha \omega_5})$$
Limits: to find the most likelihood phylogeny(ies), we need to consider ALL possible parameter vectors $\theta$ (evolutionary model, topology $T + \text{branch lengths } \omega$);

A solution: Heuristics, such as PAUP*, PHYML, IQPNNI, RAxML, GARLI ...

Limits: for some evolutionary models, we cannot find analytical solutions (pour $P_{xy}(\cdot), \pi_x ...$);

A solution: bayesian methods.
Bayesian methods

Based on the MCMC algorithm [Metropolis et al., 1953]:

1. start with a random vector of parameters $\theta_i$;
2. select a new vector $\theta_j$ by modifying $\theta_i$ in some way;
3. compute the acceptance ratio
   \[
   R = \frac{P(\theta_j|S)}{P(\theta_i|S)} = \frac{P(\theta_j)P(S|\theta_j)}{P(\theta_i)P(S|\theta_i)};
   \]
4. accept $\theta_j$ with a probability $\rho = \max(R, 1)$;
5. every $k$ generations, save the current tree and all parameters;
6. return to step 2.

Variants: Metropolis-Hasting, Metropolis Coupled MCMC (called also MC$^3$).
Distance methods

Given a matrix $S$ of $m$ characters for $n$ sequences:

- we compute a distance matrix $D_{nxn}$ where the distance $D_{s,z}$ between two sequences $s$ and $z$ is defined as the number of average mutations for site that occurred since the divergence of $s$ and $z$ (we use the information contained in $S$ and a model of evolution $M$ to correct the observed distances);

- we want to reconstruct a phylogeny $T$ such that the distances in $D$ are identical to that in $T$;

- not always possible (if the matrix is not additive);

- in this case, we want to get the tree that is closer to $D$ (i.e. a tree optimizing a given criterion);

- algorithms based on clustering (e.g. UPGMA, WPGMA, NJ, BIONJ) or not (e.g. FASTME).
Distance methods based on clustering

Different methods, based on different answers to the following questions:

- How to choose the 2 groups to fusion?
- How to replace 2 groups by one in the distance matrix?
- How to evaluate the branch lengths?
  - Least squared methods.

Figure 1.4: The clustering process to build a phylogenetic tree

-(i) initial situation. (ii) the first clustering groups. (c) the final situation.

1.6 Likelihood methods

Likelihood methods were first introduced in phylogeny by Edwards and Cavalli-Sforza [1964] to deal with gene frequency data. The first applications to molecular sequences was proposed by Neyman [1971] and improved by Kashyap and Subas [1974] and Felsenstein [1981].

In this section and in the following one, we denote as $\mathbf{x}$ the vector of all the parameters of an evolutionary model, where here an evolutionary model is the combination of a substitution model $M$ (see Section 1.4), a topology and its branch lengths.

Given a sequence alignment $S$ of $n$ character sequences (one per studied species) of length $m$ and a vector of parameters of an evolutionary model $\mathbf{x}$, the likelihood of $\mathbf{x}$, denoted by $P(S|\mathbf{x})$, is defined as the probability to observe the data set $S$, given $\mathbf{x}$. The likelihood can be viewed as a function of $\mathbf{x}$.

The hypothesis of the independence of the evolution of each site, already evoked in Section 1.4.1, implies that $P(S|\mathbf{x}) = \prod_{j=1}^{m} P(c_j|\mathbf{x})$ (1.12)

This simplifies a lot the calculation of the likelihood. When the vector $\mathbf{x}$ is given, the topology of $T$ is known. In such a case, to compute the likelihood of a site $c_j$, we associate at each node $u$ of $T$ a likelihood vector $L_{c_j,u} = (L_{c_j,u,A}, L_{c_j,u,T}, L_{c_j,u,G}, L_{c_j,u,C})$, where $L_{c_j,u,x}$ is the probability of observing the state $x$ at the node $u$, with $x \in \{A, T, G, C\}$. The reversibility hypothesis, assumed by most models of sequence evolution, implies that the likelihood of $T$ does not depend on the position of the root [Felsenstein, 1981, the “pulley principle”]. We can then compute the likelihood of an unrooted phylogeny rooting it on whatever branch or node. Once the tree is rooted, the algorithm starts initializing the likelihood vectors associated to each leaf of $T$ in the following way: $L_{c_j,u,x} = 1$ if the leaf $u$ has state $x$ at the site $c_j$, otherwise $L_{c_j,u,x} = 0$. If the state of site $c_j$ is unknown, then $L_{c_j,u,x} = 1$ [Felsenstein, 2004, page 255]. Internal nodes are considered.
Confidence values – bootstrap

Given a tree $T$ obtained with an inference method $I$ from a sequence matrix $M$ with $n$ rows (one per species) and $m$ columns (one per site):

- a set of bootstrap replicates $M = \{M_1, \cdots, M_k\}$ is derived from $M$ by sampling, with replacement, columns of $M$ until obtaining a matrix with $m$ columns;
- from each bootstrap replicate $M_i$ a tree $T_i$ is inferred, employing the inference method $I$;
- the so-obtained forest $T = \{T_1, \cdots T_k\}$ is used to estimate the reliability of each branch $e$ of $T$, with the percentage of trees in $T$ containing $e$. This value is called the bootstrap value of $e$.

Variant: (delete-half) jackknife: it consists in obtaining a set of pseudo matrices randomly by sampling without replacement half of the columns of $M$. 
Gene trees

- Gene trees are built by analyzing a gene family, i.e., homologous molecular sequences appearing in the genome of different organisms.
Gene trees

- Gene trees are built by analyzing a gene family.

- Often used to estimate species trees.
Gene trees

- Gene trees are built by analyzing a gene family.

- Often used to estimate species trees.
Gene trees

- Gene trees are built by analyzing a gene family.

- Often used to estimate species trees.

Gene trees can significantly differ from the species tree for:

- methodological reasons
- biological reasons
Gene trees are built by analyzing a gene family.

Often used to estimate species trees.

Gene trees can significantly differ from the species tree for:
- methodological reasons
- biological reasons

We usually use several gene families...
Gene trees are built by analyzing a gene family.

Often used to estimate species trees.

Gene trees can significantly differ from the species tree for:

- methodological reasons
- biological reasons

We usually use several gene families...
Reconstruction of phylogenies for multiple datasets

- from sequences

from alignments or distance matrices

- super matrix (total evidence) methods, i.e. character methods
- distance methods (e.g. SDM)
Reconstruction of phylogenies for multiple datasets

- from sequences

**from alignments or distance matrices**
- super matrix (total evidence) methods, i.e. character methods
- distance methods (e.g. SDM)

- from trees

**via triplets, quartets, clusters, splits etc.**
- consensus methods (e.g. majority/strict consensus)
- super tree methods (e.g. MRP)
Interest of supertrees

Supertrees are useful for:

- Combining heterogeneous data
- Obtaining a phylogeny using several genes:
  - Avoids having to deal with too much missing data
  - Evolutionary models adapted for each gene sequence
- Pointing out problematic areas of the phylogeny
  - agreement and disagreement among input trees.
  - measuring taxon overlap
Consensus methods on unrooted trees (bipartitions)

All source trees have the same taxa

(1) strict consensus, majority consensus

(2) semistrict consensus

(3) greedy consensus

(4) Nelson consensus

(5) Nelson-Page consensus
A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition.

Consensus methods on rooted trees (clusters)
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. 
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.

3. Sort the characters in lexicographical order.
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.

3. Sort the characters in lexicographical order. $O(kn^2)$: radix sort (linear in the input)
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.

3. Sort the characters in lexicographical order. $O(kn^2)$: radix sort (linear in the input)

4. Find the characters that are present $k$ times in a row.
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.

3. Sort the characters in lexicographical order. $O(kn^2)$: radix sort (linear in the input)

4. Find the characters that are present $k$ times in a row. $O(kn^2)$: a simple traversal of the matrix $M$
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.

3. Sort the characters in lexicographical order. $O(kn^2)$: radix sort (linear in the input)

4. Find the characters that are present $k$ times in a row. $O(kn^2)$: a simple traversal of the matrix $M$

5. Find the tree corresponding to the compatible characters.
Consensus methods on rooted trees (clusters)

A $O(kn^2)$ algorithm for the strict consensus

1. Code each branch in the tree forest as a bipartition. $O(kn^2)$: postfix traversal of each of the $k$ trees, gathering a $O(n)$ information for each of the $O(n)$ nodes.

2. Collect the characters of all branches of the forest in a matrix $M$. $O(kn^2)$: each of the $O(kn)$ characters is created in $O(n)$ from the list of leaves below the node.

3. Sort the characters in lexicographical order. $O(kn^2)$: radix sort (linear in the input)

4. Find the characters that are present $k$ times in a row. $O(kn^2)$: a simple traversal of the matrix $M$

5. Find the tree corresponding to the compatible characters. $O(n^2)$: apply a linear algorithm to construct a tree from a compatibility matrix on the $O(n)$ characters (each of size $O(n)$).
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
Consensus methods on rooted trees (clusters)

A \( O(n) \) algorithm for the strict consensus of 2 trees

1. Number the leaves of \( T_1 \) in the left-right order of appearance;
2. For each node \( u \) of \( T_1 \), compute \( M(u) \), \( m(u) \), \( nb(u) \);
3. Construct the clade list \( L_1 \) for \( T_1 \), in lexicographical order;
4. Construct the clade list \( L_2 \) for \( T_2 \), with clades such that \( M(u) - m(u) \leq nb(u) - 1 \), in lexicographical order;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
4. Construct the clade list $L_2$ for $T_2$, with clades such that $M(u) - m(u) \leq nb(u) - 1$, in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
Consensus methods on rooted trees (clusters)

A \(O(n)\) algorithm for the strict consensus of 2 trees

1. Number the leaves of \(T_1\) in the left-right order of appearance;
2. For each node \(u\) of \(T_1\), compute \(M(u), m(u), nb(u)\);
3. Construct the clade list \(L_1\) for \(T_1\), in lexicographical order;
4. Construct the clade list \(L_2\) for \(T_2\), with clades such that \(M(u) - m(u) \leq nb(u) - 1\), in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
4. Construct the clade list $L_2$ for $T_2$, with clades such that $M(u) - m(u) \leq nb(u) - 1$, in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
4. Construct the clade list $L_2$ for $T_2$, with clades such that $M(u) - m(u) \leq nb(u) - 1$, in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
4. Construct the clade list $L_2$ for $T_2$, with clades such that $M(u) - m(u) \leq nb(u) - 1$, in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
4. Construct the clade list $L_2$ for $T_2$, with clades such that $M(u) - m(u) \leq nb(u) - 1$, in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
6. Mark the nodes to keep in $T_1$;
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the strict consensus of 2 trees

1. Number the leaves of $T_1$ in the left-right order of appearance;
2. For each node $u$ of $T_1$, compute $M(u)$, $m(u)$, $nb(u)$;
3. Construct the clade list $L_1$ for $T_1$, in lexicographical order;
4. Construct the clade list $L_2$ for $T_2$, with clades such that $M(u) - m(u) \leq nb(u) - 1$, in lexicographical order;
5. Joint traverse the two lists to keep only the clades present in both;
6. Mark the nodes to keep in $T_1$;
7. Collapse the other nodes to obtain the strict consensus.
Consensus methods on rooted trees (clusters)

A $O(n)$ algorithm for the Adams consensus of 2 rooted trees

---

**Figure 3.8:** Example of Adams consensus tree (Section 3.2.2.1) for a forest comprised of two trees $T_1$ and $T_2$.

The computation of the Adams consensus tree for this forest requires 3 recursive steps. This type of consensus tree is useful for identifying rogue taxa, i.e., taxa whose position greatly differs from one input tree to another. For example, the rooted trees $T_1 = ((a, b), (c, d), e, f, g)$ and $T_2 = ((a, g), (c, d), e, f, b)$ have the same shape (i.e., they are equivalent if leaf labels are not taken into account) but differ in the positions of taxa $b$ and $g$. The Adams consensus tree puts these taxa at the most inclusive position that each occupies in any of the input trees. Since each of the taxa was positioned at the basis of $T_1$ or $T_2$, both are moved to the basis of the Adams consensus tree $T_A = ((a, c), (d, e), f, b, g)$. Note that the Adams consensus method interprets polytomies as Adams polytomies (see Section 3.1.3). Several properties of the MinCut supertree are defined with respect to the Adams consensus (see Section 3.3.1.2 for more details). Some recent methods are also useful for identifying rogue taxa both in the consensus setting, i.e., the afore-descrived MAST method (see Section 3.2.1.7), and in the supertree setting, i.e., the SMAST and PhySIC_IST methods, respectively presented in sections 3.3.4.1 and 4.3. For the afore-described forest, those methods...
Consensus methods on rooted trees (MAST)

The MAST (Maximum Agreement SubTree) problem consists in finding the maximum agreement subset tree for a forest $\mathcal{F}$. Given a forest of trees $\mathcal{F}$, an agreement subtree $T$ is a tree such that $L(T) \subseteq L(\mathcal{F})$ and $T = T_i|_{L(T)}$, $\forall T_i \in \mathcal{F}$.
Consensus methods on rooted trees (MAST)

The MAST (Maximum Agreement SubTree) problem consists in finding the maximum agreement subset tree for a forest $\mathcal{F}$. Given a forest of trees $\mathcal{F}$, an agreement subtree $T$ is a tree such that $L(T) \subseteq L(\mathcal{F})$ and $T = T_i|_{L(T)}$, $\forall T_i \in \mathcal{F}$.

- an $O(\sqrt{dn\log(n)})$ algorithm [Przytycka, 1997] and an $O(\sqrt{dn\log^2(\frac{n}{d})})$ algorithm [Kao et al., 2001] for two rooted trees, where $n = |L(\mathcal{F})|$ and $d$ is the maximum degree of the input tree nodes;
- an $O(n^{1.5})$ algorithm for two unrooted trees [Kao et al., 1999].
- NP-hard for more than two trees
- can be solved in polynomial time for forests with bounded degree [e.g., Bryant, 1997; Guillemot and Nicolas, 2006].
- an FPT algorithm has been proposed for unbounded degree forests [Guillemot and Nicolas, 2006].
Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $R(T)$.

![Diagram of a tree and its triplets]

{ $ab|c$, $ab|d$, $cd|a$, $cd|b$ }
Triplet information

Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $R(T)$.

$\{a b | c, a b | d, c d | a, c d | b\}$
Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $\mathcal{R}(T)$.

$\mathcal{R}(T) = \{ ab|c, ab|d, cd|a, cd|b \}$
Triplet information

- Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $R(T)$.

$$\{ab\mid c, ab\mid d, cd\mid a, cd\mid b\}$$

- The set of triplets of a forest of trees $T$ is denoted by $R(T)$.

- A conflict of $F$ is a set of three leaves $\{a, b, c\}$ such that $T_i, T_j \in F, T_i\mid \{a, b, c\} \neq T_j\mid \{a, b, c\}$.

- Property: Two binary rooted trees $T_1$ and $T_2$ are isomorphic iff $R(T_1) = R(T_2)$.

- Corollary: Two binary rooted trees $T_1$ and $T_2$ are isomorphic iff there are no conflicts of $F$.

- Conclusion: if $\{a, b, c\}$ is a conflict of $F$, then any MAST cannot contain these three leaves at once (thus we have to remove at least one of these leaves).
Any rooted tree \( T \) can be decomposed in the set of its triplets, denoted by \( \mathcal{R}(T) \).
Triplet information

- Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $\mathcal{R}(T)$.
- The set of triplets of a forest of trees $\mathcal{T}$ is denoted by $\mathcal{R}(\mathcal{T})$.
Triplet information

- Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $\mathcal{R}(T)$.
- The set of triplets of a forest of trees $\mathcal{T}$ is denoted by $\mathcal{R}(\mathcal{T})$.
- A conflict of $\mathcal{F}$ is a set of three leaves $\{a, b, c\}$ such that $T_i, T_j \in \mathcal{F}$, $T_i|\{a, b, c\} \neq T_j|\{a, b, c\}$.

**Property**

Two binary rooted trees $T_1$ and $T_2$ are isomorphic iff $\mathcal{R}(T_1) = \mathcal{R}(T_2)$

**Corollary**

Two binary rooted trees $T_1$ and $T_2$ are isomorphic iff there are no conflicts of $\mathcal{F}$.
Triplet information

- Any rooted tree $T$ can be decomposed in the set of its triplets, denoted by $\mathcal{R}(T)$.
- The set of triplets of a forest of trees $\mathcal{T}$ is denoted by $\mathcal{R}(\mathcal{T})$.
- A conflict of $\mathcal{F}$ is a set of three leaves $\{a, b, c\}$ such that $T_i, T_j \in \mathcal{F}$, $T_i|\{a, b, c\} \neq T_j|\{a, b, c\}$.

**Property**

Two binary rooted trees $T_1$ and $T_2$ are isomorphic iff $\mathcal{R}(T_1) = \mathcal{R}(T_2)$

**Corollary**

Two binary rooted trees $T_1$ and $T_2$ are isomorphic iff there are no conflicts of $\mathcal{F}$.

- Conclusion: if $\{a, b, c\}$ is a conflict of $\mathcal{F}$, then any MAST cannot contain these three leaves at once (thus we have to remove at least one of these leaves).
A FPT algorithm in $\rho$ for the rooted MAST
A FPT algorithm in \( \rho \) for the rooted MAST

The search tree has \( O(3^p) \) nodes, so the overall complexity is \( O(3^p kn) \).
For rooted forests, terminal taxa not included in a MAST tree can be regrafted to the tree.

Data: A forest of rooted trees \( F = \{T_1, T_2, \cdots, T_k\} \).

Result: The set of clusters \( C \) induced by the CPRT of \( F \), as defined in Gordon [1980],

\[
\begin{align*}
1 & C \leftarrow \emptyset; \\
2 & T_M \leftarrow \text{MAST}(F); \\
3 & L' \leftarrow L(F) - L(T_M); \\
4 & \text{foreach (cluster } C_i \in C(T_M) \text{ do} \\
5 & \quad A_i \leftarrow C_i; \\
6 & \quad \text{foreach (taxon } l_j \in L' \text{ do} \\
7 & \quad \quad T_S \leftarrow \text{strict consensus tree for } F_{|L(T_M)\cup l_j}; \\
8 & \quad \quad \text{if } (\{C_i \cup l_j\} \in C(T_S)) \text{ then} \\
9 & \quad \quad \quad A_i \leftarrow A_i \cup l_j; \\
10 & \quad C \leftarrow C \cup A_i; \\
11 & T_{SC} \leftarrow \text{strict consensus tree for } F; \\
12 & \text{return } C \cup C(T_{SC});
\end{align*}
\]
The common pruned-and-regrafted tree (CPRT)

For rooted forests, terminal taxa not included in a MAST tree can be regrafted to the tree.

Data: A forest of rooted trees \( \mathcal{F} = \{ T_1, T_2, \cdots, T_k \} \).

Result: The set of clusters \( \mathcal{C} \) induced by the CPRT of \( \mathcal{F} \), as defined in Gordon [1980],

1. \( C \leftarrow \emptyset \);
2. \( T_M \leftarrow \text{MAST}(\mathcal{F}) \);
3. \( L' \leftarrow L(\mathcal{F}) - L(T_M) \);
4. \textbf{foreach (cluster} \( C_i \in \mathcal{C}(T_M) \text{ do}
5. \qquad A_i \leftarrow C_i \);
6. \textbf{foreach (taxon} \( l_j \in L' \text{ do}
7. \qquad T_S \leftarrow \text{strict consensus tree for} \ \mathcal{F}|_{(L(T_M) \cup l_j)}
8. \qquad \text{if} \ \{ C_i \cup l_j \} \in \mathcal{C}(T_S) \text{ then}
9. \qquad \qquad A_i \leftarrow A_i \cup l_j \);
10. \qquad C \leftarrow C \cup A_i \);
11. \( T_{SC} \leftarrow \text{strict consensus tree for} \ \mathcal{F} \);
12. \textbf{return} \ \mathcal{C} \cup \mathcal{C}(T_{SC}) \);

The CPRT might not be unique. It can be computed only for collections of rooted trees since its computation requires the use of clusters.
Supertree methods – DECISION methods

Supertree methods can be classified depending on the way they deal with incongruent data:
Supertree methods – DECISION methods

Supertree methods can be classified depending on the way they deal with incongruent data:

1. The first suite of methods cannot handle incompatible source trees, e.g., Build (Aho et al., 1981).
Supertree methods – DECISION methods

Supertree methods can be classified depending on the way they deal with incongruent data:

1. The first suite of methods cannot handle incompatible source trees, e.g., Build (Aho et al., 1981).

Since phylogenies usually conflict with one another, those methods are of limited use.

---

Figure 3.10: Example of the common pruned-and-regrafted consensus tree - the input forest is comprised of two rooted trees (i) and (ii). The MAST and the CPRT are depicted respectively in (iii) and (iv).

- A graph may be **undirected**, meaning that there is no distinction between the two nodes associated with each edge, or **directed**, if some edges may be directed from one node to another. Note that, as mentioned in Section 3.1, a tree is a graph in which any two nodes are connected by exactly one path.

3.3.1 The OneTree supertree method and its variants

This set of supertree methods encodes topological relationships contained in the source trees in a graph introduced by Aho et al. [1981], hence is known as the Aho graph. These supertree methods are defined only for rooted trees.

3.3.1.1 The OneTree supertree

The OneTree supertree method, proposed by Ng and Wormald [1996] and then modified by Bryant [1997], is based on the Build algorithm [Aho et al., 1981]. The Build algorithm is a yes-or-no algorithm that tells whether a collection of triplets $R$ on a leaf set $L$ is compatible or not. To achieve its goal, Build tries to build a tree displaying all triplets in $R$, i.e., to find a tree such that $L(T) = L(F)$ and $R(F) \sqsubseteq R(T)$; if the process is blocked at some step, this means that the input triplets are not compatible. The OneTree supertree method for a rooted forest $F$ consists in applying the Build algorithm to the triplet set $R(F)$, to obtain a tree $T_B$ such that $L(T_B) = L(F)$ and $R(F) \sqsubseteq R(T_B)$. If such a tree does not exist, i.e., if $F$ is not compatible, this method does not return a tree, so the OneTree supertree method does not handle incompatible source trees. In practice since phylogenies usually conflict with one another (see Chapter 2), this method is of limited use.
Algorithm 3: Build($R, L$)

Data: A triplet set $R$ on a leaf set $L$.
Result: A tree $T : L(T) = L$ and $R \subseteq R(T)$ or a statement that no such a tree exists.

1. if $|L| = 1$ then return a single node labeled by the label of $L$;
2. else
3.   if $|L| = 2$ then
4.     return a tree with two leaves respectively labeled by the labels of $L$;
5.   else
6.     create a new tree $T$ composed by a single unlabeled node $r$;
7.     construct $G(R, L)$;
8.     if $|CC(G(R, L))| = 1$ then return “no such a tree exists”;
9.     else
10.    foreach (connected component $C_i \in CC(G(R, L)))$ do
11.      if ($\text{Build}(R|_{V(C_i)}, V(C_i))$ returns a tree $T_{C_i}$) then
12.        add the root node of $T_{C_i}$ as son of $r$ in $T$;
13.      else
14.        return “no such a tree exists”;
15.    return $T$;

The OneTree supertree algorithm for a forest $F$ runs in $O(|R(F)| \cdot |L(F)|)$ time [Bryant, 1997]. There exists a faster implementation of this method in the case of binary trees that runs in $O(m \cdot |L(F)|)$ time, where $m = \sum_{T_i \in F} |I(T_i)|$ [Henzinger et al., 1999], where, recall, $I(T)$ is the set of interior nodes in $T$. This algorithm can be improved to $O(m \cdot \log_2 |L(F)|)$ by changing the dynamic connectivity algorithm it resorts to [Berry and Semple, 2006]. Obtaining a tree $T$ such that $R(F) \sqsubseteq R(T)$ for a compatible forest $F$ can also be done using the Ancestral Build algorithm [Berry and Semple, 2006; Daniel and Semple, 2004]. This method accepts as input trees where some internal nodes can be labeled and is not based on the Aho graph but on a graph called the descendancy graph. Its running time for input trees of unbounded degree is $O(\log_2 |L(F)| \cdot \sum_{T_i \in F} d^2)$ where $d(u)$ denotes the degree of the node $u$.

Note that for a compatible triplet set $R$ on a leaf set $L$, there are often more than one tree displaying all triplets in $R$. Moreover, the number of rooted phylogenetic trees with this property may be exponential in $|R|$. Semple [2003] presented the method AllMinTrees that returns all trees $F_{\min} R$.
### Build or OneTree algorithm

**Data:** A triplet set $\mathcal{R}$ on a leaf set $L$.

**Result:** A tree $T : L(T) = L$ and $\mathcal{R} \subseteq \mathcal{R}(T)$ or a statement that no such a tree exists.

1. **if** $(|L| = 1)$ **then** return a single node labeled by the label of $L$;
2. **else**
   3. **if** $(|L| = 2)$ **then**
      4. return a tree with two leaves respectively labeled by the labels of $L$;
   5. **else**
      6. create a new tree $T$ composed by a single unlabeled node $r$;
      7. construct $G(\mathcal{R}, L)$;
      8. **if** $(|CC(G(\mathcal{R}, L))| = 1)$ **then** return “no such a tree exists”;
      9. **else**
         10. **foreach** (connected component $C_i \in CC(G(\mathcal{R}, L)))$ **do**
            11. **if** $(\text{Build}(\mathcal{R}|_{V(C_i)}, V(C_i))$ returns a tree $T_{C_i}$) **then**
                12. add the root node of $T_{C_i}$ as son of $r$ in $T$;
            13. **else**
                14. return “no such a tree exists”;
      15. return $T$;

**Complexity:**
- $O(\mathcal{R}(\mathcal{F}))L(\mathcal{F})$
- $m\log^2(L(\mathcal{F}))$, with $m$ number of interior vertices in $\mathcal{F}$ for binary trees

1. $(((a, c), b), (e, f))$ and $(((a, d), b), c)$ is none
2. $(((a, c), b), (e, f))$ and $((a, d), b, c)$ is $(((a, d), c), b), (e, f))$. 

---

**Examples of Aho graphs**

(a) $C_1$ and $C_2$ don't exist, the method outputs a tree (the minimal threshold tree) with interesting properties [Willson, 2004].

(b) A compatible forest $\mathcal{F}$ of a set of rooted triples $\mathcal{R}$ is the set of interior nodes in $\mathcal{F}$.

(c) If $\mathcal{F}$ is a compatible forest, then its number of interior vertices is $\log_2|\mathcal{F}|$.

(d) A star tree is a phylogenetic tree where all leaves are at distance one from the root.

(e) An ancestral tree of $\mathcal{F}$ is a tree that displays the OneTree supertree is returned for which some polytomies may have degree is $\log_2|\mathcal{F}|$.
Vote methods resolve conflicts, opting for the resolution that maximizes their optimization criteria.

Supertree methods – VOTE methods
Vote methods resolve conflicts, opting for the resolution that maximizes their optimization criteria.

**Pros:** usually, highly resolved and accurate supertrees.
Supertree methods – VOTE methods

Vote methods resolve conflicts, opting for the resolution that maximizes their optimization criteria.

- **Pros**: usually, highly resolved and accurate supertrees.
- **Cons**: sometimes clades contradicting all source trees (Goloboff and Pol, 2002).
Variants of the Build algorithm

- MinCut (MC), Modified MC and strict consensus supertrees

Algorithm 4: $\text{MC}(\mathcal{F}, w)$

**Data:** A set of rooted trees $\mathcal{F}$ and a weighed function $w : \{1, \ldots, |\mathcal{F}|\} \rightarrow (\mathbb{Q}^+)^{|\mathcal{F}|}$.

**Result:** A tree $T_{\text{MC}}$ that is the MC supertree for the pair $(\mathcal{F}, w)$.

1. if $(|L(\mathcal{F})| = 1)$ then return a single node labeled by the label of $L(\mathcal{F})$;
2. else
3. otherwise if $(|L(\mathcal{F})| = 2)$ then
4. return a tree with two leaves respectively labeled by the labels of $L(\mathcal{F})$;
5. else
6. create a new tree $T_{\text{MC}}$ composed by an unlabeled node $r$;
7. construct $\mathcal{G}(\mathcal{R}(\mathcal{F}), L(\mathcal{F}))$;
8. if $|\mathcal{CC}(\mathcal{G}(\mathcal{R}(\mathcal{F}), L(\mathcal{F})))| = 1$ then
9. construct $\mathcal{G}(\mathcal{F}, w)$;
10. $\mathcal{G} \leftarrow \mathcal{G}(\mathcal{F}, w)$;
11. construct the set $E'$ of edges of $\mathcal{G}$ that lie in at least one minimum-weight cut set of $\mathcal{G}$;
12. $C \leftarrow \mathcal{CC}(\mathcal{G} \setminus E')$;
13. else
14. $C \leftarrow \mathcal{CC}(\mathcal{G}(\mathcal{R}(\mathcal{F}), L(\mathcal{F})))$;
15. **foreach** (connected component $C_i \in C$) **do**
16. $T_{C_i} \leftarrow \text{MC}(\mathcal{F}_{\mathcal{V}(C_i)}, w)$;
17. add the root node of $T_{C_i}$ as son of $r$ in $T_{\text{MC}}$;
18. return $T_{\text{MC}}$;
Variants of the Build algorithm

- MinCut (MC), Modified MC and strict consensus supertrees

---

**Algorithm 4: MC(\(F, w\))**

Data: A set of rooted trees \(F\) and a weight \(w: \{1, \ldots, |F|\} \rightarrow (Q^+)^{|F|}\).

Result: A tree \(T_{MC}\) that is the MC super tree.

1. if \(|L(F)| = 1\) then return a single node
2. else
3. if \(|L(F)| = 2\) then
4. return a tree with two leaves respect \(L(F)\);
5. else
6. create a new tree \(T_{MC}\) composed by
7. construct \(G(R(F), L(F))\);
8. if \(|CC(G(R(F), L(F)))| = 1\) then
9. construct \(G(F, w)\);
10. \(G \leftrightarrow G(F, w)\);
11. construct the set \(E'\) of edges of minimum-weight cut set of \(G\);
12. \(C \leftarrow CC(G \setminus E')\);
13. else
14. \(C \leftarrow CC(G(R(F), L(F)))\);
15. foreach (connected component \(C_i \in C\)) do
16. \(T_{C_i} \leftarrow MC(F_{V(C_i)}, w))\);
17. add the root node of \(T_{C_i}\) as son of \(r\) in \(T_{MC}\);
18. return \(T_{MC}\);

---

**Algorithm 5: GMMC(F, \(\hat{G}(F, w)\))**

Data: A set of rooted trees \(F\) and a weighted graph \(\hat{G}(F, w)\).

Result: A weighted graph \(G_{MMC}\).

1. \(G_{MMC} \leftarrow \hat{G}(F, w)\);
2. \(E' \leftarrow \emptyset\);
3. compute the set \(C\) of contradicted edges in \(G_{MMC}\);
4. if \((G_{MMC} \setminus C\) is disconnected) then
5. find the connected components of \(G_{MMC} \setminus C\);
6. \(E' \leftarrow\) edges connecting two nodes of the same connected component;
7. else
8. build the set \(AC\) of all edges in \(G_{MMC}\) adjacent to a contradicted edge;
9. if \((G_{MMC} \setminus (AC \cap C)\) is disconnected) then
10. find the components of \(G_{MMC} \setminus (AC \cap C)\);
11. \(E' \leftarrow\) edges connecting two nodes of the same connected component;
12. return \(\hat{G}(F, w) \odot E'\).
Variants of the Build algorithm

- MinCut (MC), Modified MC and strict consensus supertrees

<table>
<thead>
<tr>
<th>Algorithm 4: ( \text{MC}(\mathcal{F}, w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data:</strong> A set of rooted trees ( \mathcal{F} ) and a weight ( w : {1, \cdots,</td>
</tr>
<tr>
<td><strong>Result:</strong> A tree ( T_{\text{MC}} ) that is the MC supertree for ( \mathcal{F} ).</td>
</tr>
<tr>
<td>1. if ( (</td>
</tr>
<tr>
<td>2. else</td>
</tr>
<tr>
<td>3. if ( (</td>
</tr>
<tr>
<td>4. return a tree with two leaves respecting ( L(\mathcal{F}) );</td>
</tr>
<tr>
<td>5. else</td>
</tr>
<tr>
<td>6. create a new tree ( T_{\text{MC}} ) composed by</td>
</tr>
<tr>
<td>7. constructing ( G(\mathcal{R}(\mathcal{F}), L(\mathcal{F})) );</td>
</tr>
<tr>
<td>8. if ( \left</td>
</tr>
<tr>
<td>9. construct ( G(\mathcal{F}, w) );</td>
</tr>
<tr>
<td>10. ( G \leftarrow G(\mathcal{F}, w) );</td>
</tr>
<tr>
<td>11. construct the set ( E' ) of edges of minimum-weight cut set of ( G );</td>
</tr>
<tr>
<td>12. ( C \leftarrow \text{CC}(G \setminus E') );</td>
</tr>
<tr>
<td>13. else</td>
</tr>
<tr>
<td>14. ( C \leftarrow \text{CC}(G(\mathcal{R}(\mathcal{F}), L(\mathcal{F}))) );</td>
</tr>
<tr>
<td>15. foreach (connected component ( C_i \in \mathcal{C}(\mathcal{T}) )) do</td>
</tr>
<tr>
<td>16. ( T_{C_i} \leftarrow \text{MC}(\mathcal{F}</td>
</tr>
<tr>
<td>17. add the root node of ( T_{C_i} ) as son</td>
</tr>
<tr>
<td>18. return ( T_{\text{MC}} );</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 5: ( G_{\text{MMC}}(\mathcal{F}, \hat{G}(\mathcal{F}, w)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data:</strong> A set of rooted trees ( \mathcal{F} ) and a weighted graph ( \hat{G}(\mathcal{F}, w) ).</td>
</tr>
<tr>
<td><strong>Result:</strong> A weighted graph ( G_{\text{MMC}} ).</td>
</tr>
<tr>
<td>1. ( G_{\text{MMC}} \leftarrow \hat{G}(\mathcal{F}, w); )</td>
</tr>
<tr>
<td>2. ( E' \leftarrow \emptyset; )</td>
</tr>
<tr>
<td>3. compute the set ( C ) of contradicted edges in ( G_{\text{MMC}} );</td>
</tr>
<tr>
<td>4. if ( (G_{\text{MMC}} \setminus C ) is disconnected) then</td>
</tr>
<tr>
<td>5. find the connected components of ( G_{\text{MMC}} \setminus C );</td>
</tr>
<tr>
<td>6. ( E' \leftarrow ) edges connecting two nodes of the same connected component;</td>
</tr>
<tr>
<td>7. else</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 6: ( \text{SCS}(\mathcal{F}, w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data:</strong> A set of rooted trees ( \mathcal{F} ).</td>
</tr>
<tr>
<td><strong>Result:</strong> A tree ( T_{\text{SCS}} ) that is the strict consensus supertree for ( \mathcal{F} ).</td>
</tr>
<tr>
<td>1. ( BC \leftarrow \emptyset ) // Bad Clusters set;</td>
</tr>
<tr>
<td>2. ( T \leftarrow \text{Build}(\mathcal{R}(\mathcal{F}), L(\mathcal{F})); )</td>
</tr>
<tr>
<td>3. foreach (cluster ( C_i \in \mathcal{T}(\mathcal{F}) )) do</td>
</tr>
<tr>
<td>4. ( x \leftarrow ) a leaf of ( C_i );</td>
</tr>
<tr>
<td>5. foreach ( (a, b) \in L(\mathcal{F}) ) such that ( a \in C_i \setminus {x}, b \in L(\mathcal{F}) \setminus L(C_i) ) do</td>
</tr>
<tr>
<td>6. if ( !(G(\mathcal{R}(\mathcal{F}) \cup {ab</td>
</tr>
<tr>
<td>7. ( BC \leftarrow BC \cup C_i; )</td>
</tr>
<tr>
<td>8. build the tree ( T_{\text{SCS}} ) such that ( \mathcal{C}(T_{\text{SCS}}) = \mathcal{C}(T) - BC );</td>
</tr>
<tr>
<td>9. return ( T_{\text{SCS}} );</td>
</tr>
</tbody>
</table>

\( O(n^6 \cdot \log^2(n)) \)
MRP (Matrix Representation with Parsimony)

The clades of source trees (a) are coded as a matrix $M$ of binary characters (plus question marks).
We apply a maximum parsimony algorithm to $M$ to obtain a supertree (c).

<table>
<thead>
<tr>
<th>Taxon</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A B C D E F</td>
</tr>
<tr>
<td>outgroup</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1 1 0 ? ? ?</td>
</tr>
<tr>
<td>2</td>
<td>1 1 0 ? ? ?</td>
</tr>
<tr>
<td>3</td>
<td>1 0 0 1 0 0</td>
</tr>
<tr>
<td>4</td>
<td>0 0 1 1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>0 0 1 0 1 1</td>
</tr>
<tr>
<td>6</td>
<td>? ? ? 0 1 1</td>
</tr>
<tr>
<td>7</td>
<td>? ? ? 0 1 0</td>
</tr>
</tbody>
</table>

(b) 

(c)
We can obtain an exponential number of most parsimonious trees. In this case, MRP returns the strict consensus of the most parsimonious trees (to avoid to take arbitrary decisions).
We can obtain an exponential number of most parsimonious trees. In this case, MRP returns the strict consensus of the most parsimonious trees (to avoid to take arbitrary decisions).

Properties

If $\mathcal{T}$ is compatible, we obtain the strict consensus supertree. We can take into account confidence values associated to the clades of the trees in $\mathcal{T}$ (ex, bootstrap proportions).
We can obtain an exponential number of most parsimonious trees. In this case, MRP returns the strict consensus of the most parsimonious trees (to avoid to take arbitrary decisions).

**Problems**

- The MRP supertree may contain new clades, not present in any of source trees.
- Bias Bigger clades/ less equilibrated trees.
This method [Ranwez et al. 2009] aims at finding the asymmetric median supertree according to triplet dissimilarity. It consists of four steps:

1. input trees are encoded as a set of weighted triplets;
2. a starting binary supertree is proposed by an agglomerative procedure;
3. the candidate binary supertree is iteratively improved using small topological changes;
4. unsupported edges of the binary supertree are collapsed.
Other VOTE methods

- The Matrix Representation with Flipping (MRF) supertree
- The Matrix Representation using Compatibility (MRC) supertree
- The t-MRP method
- The average consensus supertree or MRD
- Median supertrees
- Maximum Likelihood supertree
- Bayesian supertree
- Quartet supertrees
- ...

...
Supertree methods – VETO methods

3 Veto methods do not allow the resulting supertree to contain clades that a source tree would vote against.
Supertree methods – VETO methods

Veto methods do not allow the resulting supertree to contain clades that a source tree would vote against.

- proposing multifurcations
Veto methods do not allow the resulting supertree to contain clades that a source tree would vote against.

- proposing multifurcations
  OR
- pruning some taxa
**Supertree methods – VETO methods**

3. Veto methods do not allow the resulting supertree to contain clades that a source tree would vote against.
   - proposing multifurcations
   - OR
   - pruning some taxa
   - pros: more reliable supertrees

![Diagram showing supertree methods and VETO process](image-url)
Supertree methods – VETO methods

Veto methods do not allow the resulting supertree to contain clades that a source tree would vote against.

- proposing multifurcations
  - OR
- pruning some taxa

Pros: more reliable supertrees
Cons: unresolved supertrees.
PhySIC_IST – A VETO method with desirable properties

- The resulting supertree does not contain relationships that conflict either directly with a source tree or indirectly with a combination of them.

non-contradiction property, denoted by PC
PhySIC_IST – A VETO method with desirable properties

- The resulting supertree does not contain relationships that conflict either directly with a source tree or indirectly with a combination of them.

\{ABIC, BCID\} \models \text{ACID}

non-contradiction property, denoted by PC
PhySIC_IST – A VETO method with desirable properties

- The resulting supertree only contains relationships that are present in a source tree or collectively induced by several source trees.

induction property, denoted by PI
PhySIC_IST – A VETO method with desirable proprieties

- PhySIC_IST is a method that:
PhySIC_IST – A VETO method with desirable proprieties

- PhySIC_IST is a method that:
  - maximizes the information contained in the produced supertree;
PhySIC_IST – A VETO method with desirable proprieties

PhySIC_IST is a method that:

- maximizes the information contained in the produced supertree;
- returns a supertree $T$ that still respects PC and PI by:
PhySIC_Ist – A VETO method with desirable properties

- PhySIC_Ist is a method that:
  - maximizes the information contained in the produced supertree;
  - returns a supertree $T$ that still respects PC and PI by:
    - allowing multifurcations;
PhySIC_IST – A VETO method with desirable properties

PhySIC_IST is a method that:

- maximizes the information contained in the produced supertree;
- returns a supertree $T$ that still respects PC and PI by:
  - allowing multifurcations;
  - pruning rogue taxa:
PhySIC IST – A VETO method with desirable properties

PhySIC IST is a method that:
- maximizes the information contained in the produced supertree;
- returns a supertree $T$ that still respects PC and PI by:
  - allowing multifurcations;
  
  AND

- pruning rogue taxa:

$$T_1$$

$$T_2$$

$$ST_1$$
PhySIC_IST – A VETO method with desirable properties

- PhySIC_IST is a method that:
  - maximizes the information contained in the produced supertree;
  - returns a supertree $T$ that still respects PC and PI by:
    - allowing multifurcations;
    - pruning rogue taxa:
PhySIC IST (PHYlogenetic Signal with Induction and non-Contradiction Inserting a Subset of Taxa) is an algorithm that operates successive insertions of taxa on a backbone tree.
We order taxa in decreasing priority order

The first taxa to be inserted are those present in as much source trees as possible and involved in as few contradictions as possible

We build the backbone tree
Within which region of the backbone tree can a taxon $s$ be inserted without contradicting $T_1$ and $T_2$?
one best supported position (PI) and all trees agree (PC)
- one best supported position (PI) and all trees agree (PC)
- more than one best supported position and/not all trees agree (PI and PC???)
- one best supported position (PI) and all trees agree (PC)
- more than one best supported position and/not all trees agree (PI and PC???)
- one best supported position (PI) and all trees agree (PC)
- more than one best supported position and/not all trees agree (PI and PC???)
We need to compare trees with different taxa sets.

We use a variant of the CIC criterion (Thorley, Wilkinson, Charleston 1998) that also takes into account missing taxa and we define it as:

\[
CIC(T, n) = -\lg \frac{\text{number of permitted binary trees with } n \text{ taxa}}{\text{number of possible binary trees with } n \text{ taxa}}
\]
Limits of veto methods

As the amount of available information continues to increase, the number of conflicts between source trees increases.

MRP △, PhySIC ○, PhySIC_INST □
Vote VS veto methods?

51% 49%

IDEA: a flexible liberal (voting) preprocessing of the input trees before applying a veto approach.
Vote VS veto methods?

IDEA: a flexible liberal (voting) preprocessing of the input trees before applying a veto approach.
IDEA: a flexible liberal(voting) preprocessing of the input trees before applying a veto approach.
Source Tree Correction (STC) preprocess

We want to drop the statistically less supported alternative(s), if any exists.

(Chi-square based test)
After that, the STC preprocess modifies the source trees, forcing them not to contain the dropped resolutions.
After that, the STC preprocess modifies the source trees, forcing them not to contain the dropped resolutions.

Each modified tree may contain either new multifurcations, or lack some of its former taxa.
After that, the STC preprocess modifies the source trees, forcing them not to contain the dropped resolutions.

Each modified tree may contain either new multifurcations, or lack some of its former taxa.

The reliability threshold $\alpha$ is chosen by the user.
Large-scale simulations ($\alpha = 0.95$)

MRP $\triangle$, PhySIC $\bigcirc$, PhySIC_IST $\square$, STC + PhySIC $\bullet$ and STC + PhySIC_IST $\blacksquare$
Large-scale simulations ($\alpha = 0.95$)

MRP $\triangle$, PhySIC $\circ$, PhySIC_IST $\square$, STC + PhySIC $\bullet$ and STC + PhySIC_IST $\blacksquare$
Other VETO methods

- The semi-strict supertree
- PhySIC supertree
- The SMAST supertrees
- ...
Merci à V Berry et S Berard pour leur transparents, source d’inspiration de certains de miens :)

THANKS!
A *new* approach: building phylogenetic networks

Why do we need them? Due to reticulate evolutionary phenomena (hybridization, recombination, horizontal gene transfer) the evolution of a set of species sometimes cannot be described using a *single* phylogenetic tree.
Networks...

IF TIME...!