Genomics and Bioinformatics: Week 9

12 November 2013

1 Trees

In this chapter we will review the main approaches to finding a tree that best represents the evolutionary relationships among a set of homologous sequences.

We usually restrict ourselves to bifurcating trees, namely trees in which every internal node connects exactly 3 branches (namely we exclude the case of one species branching out into more than 2 new species).

Theorem 1. *An unrooted bifurcating tree with n leaves has $2n-3$ branches. There are*

$$(2n-5)!! = (2n-5)\cdot(2n-7)\cdots 1$$

such trees. The number of rooted trees with n leaves is

$$(2n-3)!! \ .$$

It is easy to see that there is only one possible tree with $n = 3$, it has 3 branches. Then each new leaf requires to add a new branch that cuts an existing branch: this increases the number of branches by 2.

The total number of trees with n leaves is equal to the number of trees with $n-1$ leaves multiplied by the number of ways to add one branch to a given tree, which is equal to the number of branches in that tree (the new branch can attach to any existing branch). By recursion this is equal to

$$(2(n-1)-3)\cdot(2(n-2)-3)\cdots \ .$$

Rooted trees are like unrooted trees with one additional leaf: the "root leaf".

The order of magnitude of the above formula are indicated below:

n	$(2n-5)!!$
3	1
5	15
7	945
10	$2 \cdot 10^6$
20	$3 \cdot 10^{20}$
30	$9 \cdot 10^{36}$
50	$3 \cdot 10^{74}$

2 Parsimony methods

We have n species with k homologous nucleotides $x_{ij} : i = 1, \ldots, n; j = 1, \ldots, k$. We assume that there exists a tree which represents the evolution of those characters from a common ancestor. The simplest methods to infer this tree is using *parsimony* which follows Cavalli-Sforza's principle (1963): the preferred tree should involve "the minimum net amount of evolution".

For example consider the following situation (3 characters, 5 species):

	1	2	3
a	A	C	A
b	A	G	A
c	A	G	C
d	T	C	C
e	T	C	A

We first take the tree as given and see how to map the data onto it (see Figure 1): in this example there are several consistent ways of assigning nucleotide changes to particular branches of this tree, each yields a total of 5 changes: this is the cost associated with this particular tree. Trying other tree shapes, we can attempt to find the most parsimonious tree, namely the one which has the smallest total number of changes, 5 or less.

Each branch is given a length defined as the average number of changes on that branch for all equivalent reconstructions. The sum of branch lengths must be equal to the total number of changes in the tree.

The trees in the above examples are *unrooted*. There is a classical method to locate the root of the tree by making the hypothesis of a *molecular clock*: the average number of changes in a path from the root to a leaf must be the same for every leaf, because they represent the same amount of time, and if the frequency of changes is constant (like a "clock") the number of changes must be approximately the same.

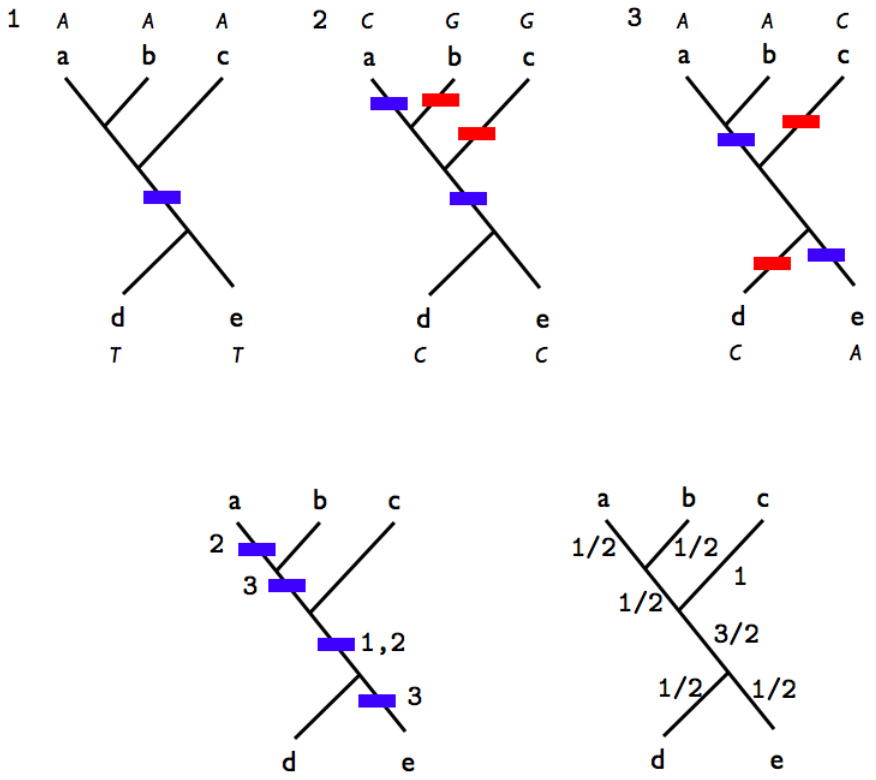

Figure 1: Top: reconstruction of character changes, each blue square indicates one change, red squares are an alternative equivalent reconstruction. Bottom left: One possible complete scenario, numbers indicate which character changes. Bottom right: branch lengths

3 Counting changes

With a large tree, it becomes difficult to count the number of changes on each branch and we need a reliable algorithm for that.

The *Fitch algorithm* works from the leafs to the root of the tree. Every interior node is assigned the intersection of the character set of the two nodes above it. If this intersection is empty, then the union is taken, and a character change ($\star$) is introduced.

The problem with this algorithm is that it is not easy to generalize, for example if we want to count differently some substitutions. A more satisfying solution is to use the

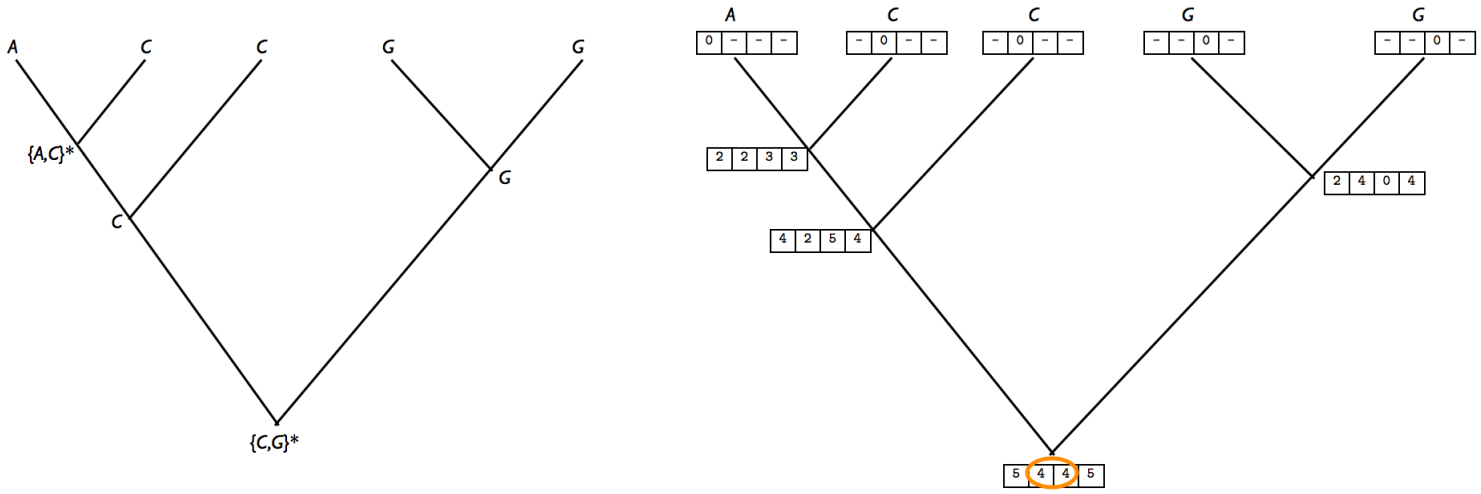

Figure 2: The Fitch and Sankoff algorithms

Sankoff algorithm: we choose a substitution cost matrix, for example

$$M = \begin{array}{c} \\ \mathrm{A} \\ \mathrm{C} \\ \mathrm{G} \\ \mathrm{T} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{T} \end{array} \\ \begin{pmatrix} 0 & 2 & 1 & 2 \\ 2 & 0 & 2 & 1 \\ 1 & 2 & 0 & 2 \\ 2 & 1 & 2 & 0 \end{pmatrix} \end{array} .$$

Then starting from the leaves, we compute a score for each base at each node according to

$$S(\alpha) = \min_{\beta} \left(S_{<}(\beta) + M(\alpha, \beta) \right) + \min_{\gamma} \left(S_{>}(\gamma) + M(\alpha, \gamma) \right) ,$$

where $S_{<}$ (resp. $S_{>}$) represents the nodes above to the left (resp. to the right) of the current node and the leaves are initialized with a score 0 for the observed base and a dummy value for the other bases. Then the total tree cost is the minimum score at the root node.

4 Distance-based methods

Another approach that is more quantitative in nature and directly uses the combined information about all the characters in the dataset is based on the matrix of distances between sequences. Every pair i, j of sequences has some distance D_{ij} and we would like to find a tree where the sum of all branch lengths connecting the leaves i and j is close to D_{ij}

4.1 UPGMA

1. Find the pair i, j with smallest distance D_{ij}.
2. Create a new group (ij) with $n_{(ij)} = n_i + n_j$ members.
3. Connect i and j in the tree to a new node (ij), each new branch has length $D_{ij}/2$.
4. Compute the new distances $D_{(ij),k} = \frac{n_i}{n_{(ij)}} D_{ik} + \frac{n_j}{n_{(ij)}} D_{jk}$, and delete entries for i and j in the distance matrix.
5. Iterate until the matrix has only one item left.

4.2 Neighbor-joining

1. For each leaf, compute $u_i = \sum_{j \neq i} D_{ij}/(n-2)$.
2. Choose i, j with smallest $D_{ij} - u_i - u_j$.
3. Connect i and j in the tree to a new node (ij), with branch lengths $d_{(ij),i} = \frac{1}{2}(D_{ij} + u_i - u_j)$ and $d_{(ij),j} = \frac{1}{2}(D_{ij} + u_j - u_i)$.
4. Compute the new distances $D_{(ij),k} = (D_{ik} + D_{jk} - D_{ij})/2$, delete entries for i and j in the distance matrix, and consider node (ij) as a new leaf in the tree.
5. Iterate until there are only 2 leaves left, which are connected by a branch of length D_{12}.

5 Likelihood

All the tree reconstruction methods described above were based on heuristic principles. We would like to describe the evolution as a random process which generates (with a certain probability) the observed characters.

5.1 The Jukes-Cantor model

We first introduce the simplest model of evolution: the *Jukes-Cantor* model. It is defined by the following probabilities of switching from a nucleotide α to a nucleotide β during a time interval of length t:

$$\begin{aligned} P(\beta|\alpha,u,t) &= \frac{1}{4}(1-e^{-4ut}) \ , \qquad \alpha \neq \beta \ , \\ P(\alpha|\alpha,u,t) &= \frac{1}{4}(1+3e^{-4ut}) \ , \end{aligned}$$

where u is the rate of mutation and is the only free parameter in the model.

This model has the following properties:

$$\begin{aligned}
\text{Markov property:} \quad & P(\beta|\alpha,u,t+s) \;=\; \sum_{\gamma} P(\beta|\gamma,u,s)P(\gamma|\alpha,u,t) \;, \\
\text{Small time asymptotics:} \quad & P(\beta|\alpha,u,\varepsilon) \;=\; u\varepsilon \;, \quad P(\alpha|\alpha,u,\varepsilon) \;=\; 1-3u\varepsilon \;, \qquad \varepsilon \to 0 \;, \\
\text{Large time asymptotics:} \quad & P(\beta|\alpha,u,t) \;\to\; \frac{1}{4} \;, \quad P(\alpha|\alpha,u,t) \;\to\; \frac{1}{4} \;, \qquad t \to \infty \;.
\end{aligned}$$

This model is the only solution with the first 2 properties and a single rate. The small time limit justifies the interpretation that u is the mutation rate. More general models exist:

1. Kimura model: Transitions ($C \leftrightarrow T, A \leftrightarrow G$) are more probable then transversions (purine↔pyrimidine). This has 2 parameters.

2. Tamura-Nei, F84, HKY models: All have 5 parameters to account for different nucleotide frequencies (3 parameters) and Kimura's 2 rates.

3. General time-reversible (GTR) model: The most complete model, with different rates for every substitutions.

All these models satisfy the time-reversal symmetry: if π_α is the frequency of base α ($\frac{1}{4}$ in Jukes-Cantor's model), then

$$\pi_\beta P(\alpha|\beta,t) \;=\; \pi_\alpha P(\beta|\alpha,t) \;.$$

meaning that the direction of time along an internal branch of the tree is irrelevant!

Given such a probabilistic model, we can now evaluate the Likelihood of a tree T given the observed characters X_i at each site i:

$$\begin{aligned}
\mathcal{L}(T) \;&=\; \prod_i P(X_i|T) \;=\; \prod_i \mathcal{L}_i(T) \;, \\
\mathcal{L}_i(T) \;&=\; \sum_u \sum_v \sum_w \sum_x \mathcal{L}(A,C,C,G,G,u,v,w,x|T) \\
&=\; \sum_u \sum_v \sum_w \sum_x P(x)P(w|x,t_{xw})P(v|x,t_{xv}) \\
&\quad\; P(u|w,t_{wu})P(C|w,t_{wC})P(G|v,t_{vD})P(G|v,t_{vE}) \\
&\quad\; P(A|u,t_{uA})P(C|u,t_{uB}) \\
&=\; \sum_x P(x) \left(\sum_v P(x|v,t_{xv})P(G|v,t_{vD})P(G|v,t_{vE}) \right) \times \\
&\quad\; \left(\sum_w P(x|w,t_{xw})P(C|w,t_{wC}) \left(\sum_u P(w|u,t_{wu})P(A|u,t_{uA})P(C|u,t_{uB}) \right) \right) \;.
\end{aligned}$$

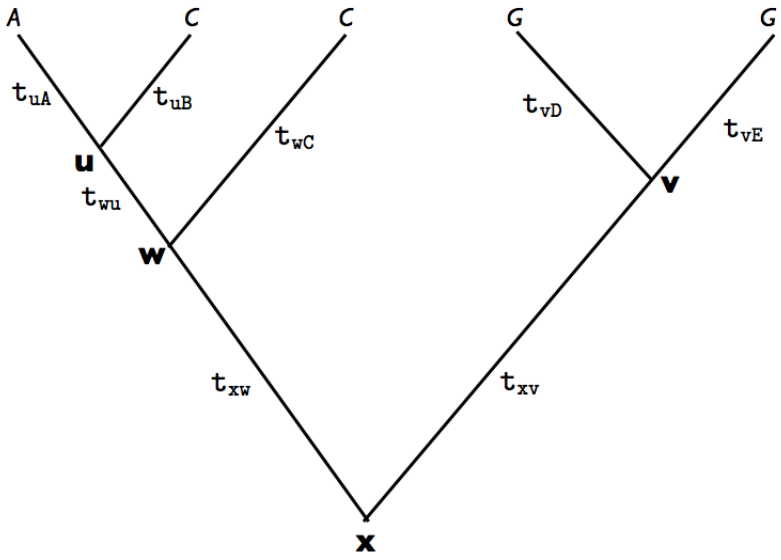

Figure 3: Calculation of the likelihood

The most important property of this formula is that it can be easily split into subtrees:

$$\mathcal{L}(T_1 \times T_2) \;=\; \mathcal{L}(T_1, x)\mathcal{L}(T_2, x)P(x) \;,$$

Therefore there exists efficient recursive algorithms that can evaluate small subtrees and combine them into a larger tree.

6 Finding the best tree

Knowing how to evaluate a tree, the ultimate goal is to find the tree that optimizes the chosen cost (minimum total length, maximum likelihood, etc.). As shown above, the number of possible trees is huge and it is not possible to just enumerate them.

Practically all existing strategies use some kind of sampling of the tree space to find a local optimum: it starts from an initial tree (for example found by a distance-based method) and then tries to improve the tree score by making small changes, evaluating the score of each new tree, and following the path(s) which improve the tree score. Typical tree modifications used are *NNI* (nearest-neighbor interchange) and *SPR* (subtree pruning and re-grafting)

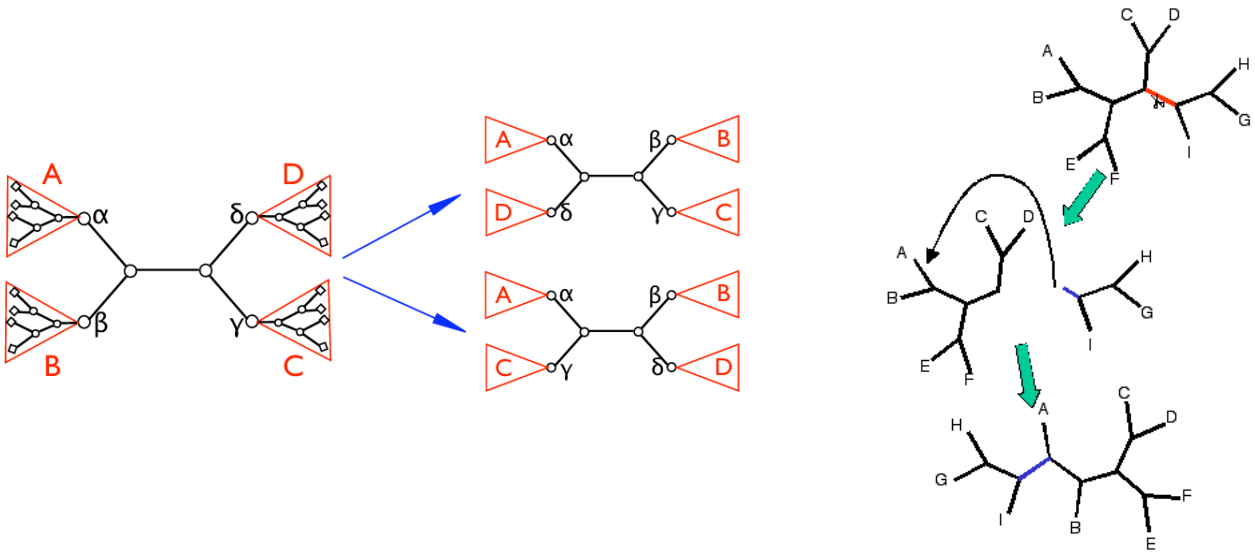

Figure 4: NNI and SPR moves

7 Bootstrap

In statistics, in general, bootstrap methods are techniques to generate a probability distribution from repeatedly sampling the same data source in a randomized way. Then a particular observation in the data can be given a p-value (probability of being observed in the random samples).

In phylogenetics, the data source is the multiple sequence alignment. Each column in the alignment is viewed as an independent replicate of the same evolutionary process. Therefore each of them carries equal information about the underlying tree. In particular if the columns of a MSA are shuffled, then all the tree reconstruction techniques described above will generate the same tree.

Phylogenetic bootstrap consists in the following procedure:

1. Reconstruct a tree T_0 based on an MSA with N characters (columns).
2. Randomly choose N columns of the MSA, where each column can be picked any number of times.
3. Reconstruct a tree as before with the re-sampled MSA.
4. Repeat the previous two steps K times.
5. For each *split* in the tree T_0, count how many of the K bootstrap trees contain the same split. The support value of the split is the corresponding proportion (the higher the better).