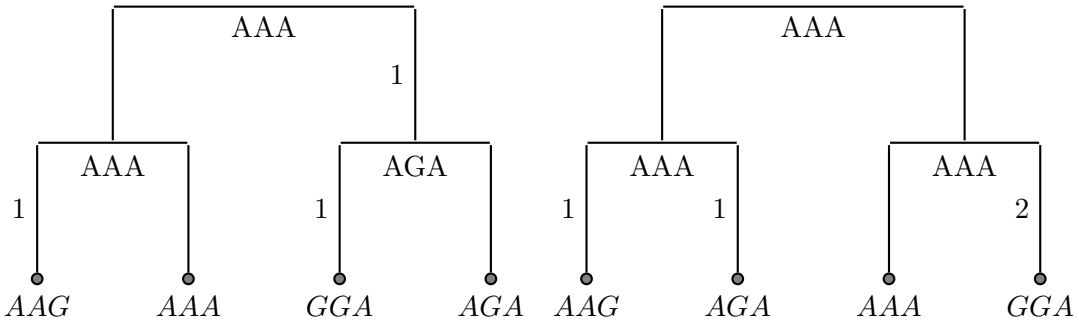
20.5 Parsimony

Parsimony is form of Occam's razor. It postulates that the best tree is the one that requires the fewest changes along it to explain all sequences. This best tree is called the most parsimonious tree or the *(maximum) parsimony tree.*

The main algorithm that we discuss here is not actually a method for constructing the maximum parsimony tree but rather provides a way of calculating the cost of any given tree. We must then search over trees to find the tree of minimal cost.

Example: suppose we have four sequences AAG, AAA, GGA and AGA. Consider the two trees given below.



The number of mutations on each branch is shown to the left of the branch. The tree topology on the left has requires 3 mutations to explain the given sequences, while the tree on the right requires 4 mutations to explain the same sequences. The more parsimonious tree is therefore the one on the left. The sequences given at the internal vertices and the positions of the mutations could be altered in these examples but the total parsimony score for each tree would remain the same. □

An algorithm to compute the parsimony cost of a tree is given below. This finds the minimum number of substitutions to explain given sequences and tree. It assumes that all changes have equal cost. A similar algorithm accounts for the case where different substitutions have different costs. The algorithm is given in terms of rooted trees but the parsimony score is independent of the position of the root so this algorithm applies to unrooted trees.

Parsimony (Fitch 1971)

Number the nodes, in descending order, so that the root node is $2n-1$. Let u be the site for which we are considering the cost. Let B be the parsimony cost.

Initilise Set $B_u = 0$ and $k = 2n-1$.

Recursion To obtain the set R_k:

If k is a leaf node: Set $R_k = x_u^k$.

If k is not a leaf: compute R_i and R_j for child nodes of k. Set $R_k = R_i \cap R_j$ if $R_i \cap R_j \neq \emptyset$. Otherwise, set $R_k = R_i \cup R_j$ and set $B_u = B_u + 1$.

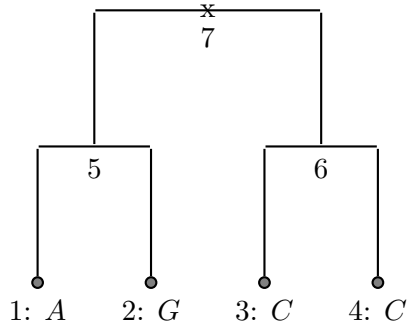
Stop Return B_u, the minimal cost of the tree at site u.

A traceback procedure can be used to construct possible ancestral states. Starting at the root, choose a residue from R_{2n-1} and go to the daughter nodes. Having chosen a residue at R_k, pick the same residue from the child set R_i if possible, otherwise choose a random reside of R_i.

The total cost for a tree and sequences is the sum of costs over all positions in the sequence. That is, if we have sequences of length L and B_i is the parsimony score for site i, then the total parsimony score for the tree and sequences is

$$B = \sum_{i=1}^{L} B_i.$$

Example: Given the following tree with just a single site at the 4 leaves we want to calculate the parsimony cost. Label the nodes as shown. There is just a single site so set $u = 1$.



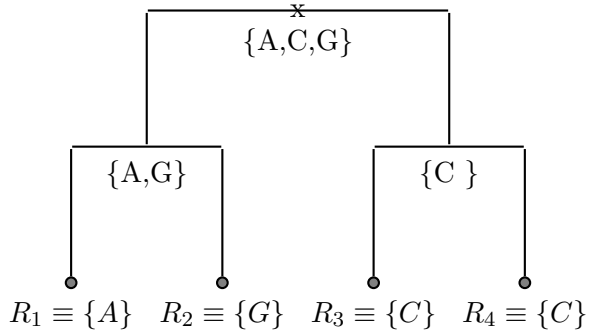
We set $B = 0$ and $k = 7$. Now try to find R_7.

7 is not a leaf node, so recurse down to its children. Want to find R_5 and R_6. 5 is not a leaf node so recurse down to its children. 1 is a leaf node so set $R_1 = \{A\}$. Similarly, $R_2 = \{G\}$. Now, $R_1 \cap R_2 = \emptyset$ so we set $R_5 = R_1 \cup R_2 = \{A, G\}$ and $B = B+1 = 0+1 = 1$.

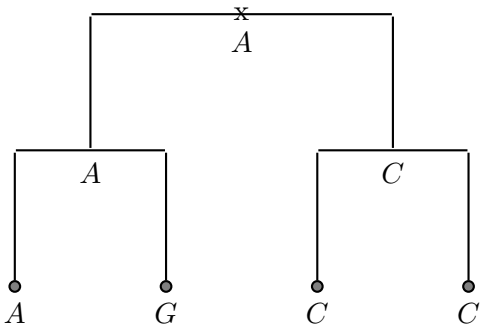
In a similar manner we get $R_3 = \{C\}$ and $R_4 = \{C\}$ so $R_6 = R_3 \cap R_4 = \{C\}$.

Now, since $R_5 \cap R_6 = \emptyset$ we set $R_7 = R_5 \cup R_6 = \{A, C, G\}$ and $B = B + 1 = 2$.

Thus we have the sets R_k as follows:



From these sets, we can traceback from the root, picking possible ancestral states that would give us the parsimony score for the tree. For example, at the root choose A, which forces us to choose A at node 5. Clearly, at node 6, we only have the choice of C.



□

20.5.1 Weighted parsimony

The basic parsimony idea easily extends to the case where instead of counting each mutation equally, different costs apply to different mutations.

Let $S(a, b)$ be the cost of mutating from a to b and again calculate the parsimony score at a single site u.

Initilise Set $k = 2n - 1$.

Recursion If k is a leaf node: Set $S_k(a) = 0$ when $a = x_u^k$ and $S_k(a) = \infty$ otherwise.

If k is not a leaf: Compute $S_i(a)$ and $S_j(a)$ for all a and children i and j of k. Set

$$S_k(a) = \min_b \left(S_i(b) + S(a, b)\right) + \min_b \left(S_j(b) + S(a, b)\right).$$

Stop Return

$$B_u = \min_a S_{2n-1}(a).$$

The total cost of the tree is

$$B = \sum_{u=1}^{L} B_u$$

Weighted parsimony reduces to the standard parsimony algorithm when $S(a, a) = 1$ and $S(a, b) = 0$ if $a \neq b$.

A traceback procedure to recover the ancestral states is again available by keeping track of which residue, b, gave the minimum at each step. For exact details of the traceback, see Durbin et al.

20.5.2 Parsimony informative sites

Many sites in an alignment will have the same parsimony score on every tree. For example, consider sites that have the same residue for all taxa (an invariant site). This will have a parsimony score of 0 regardless of the tree. Sites that have different scores on different trees are known as *parsimony informative*. It is easy to show that parsimony informative sites have at least two characters that each occur in two or more taxa.

In the detailed example given above, the site studied (which would be written as the column AGCC in a multiple sequence alignment for the 4 taxa) is not parsimony informative since there is only one site that appears more than once. If we instead considered the site corresponding to the column $AACC$, it would be parsimony informative.

20.6 Finding the maximum parsimony tree

The number of substitutions on a tree (the parsimony score) is sometimes called the *length* of a tree. This corresponds to the molecular clock idea where, under a constant rate mutation model, we will only see more substitutions if we wait for a longer time.

Thus finding the maximum parsimony tree is equivalent to finding the shortest tree. We'll consider a number of methods for finding the maximum parsimony tree for a given set of sequences.

20.6.1 Exhaustive search

Finding the maximum parsimony tree is a very hard problem computationally. Naive methods which attempt to score all possible unrooted trees fail when the number of sequences is even moderate due the huge number of possible trees.

We therefore need to resort to more clever methods and heuristic search algorithms.

The simplest of the smarter search algorithms are based on the idea of branch-and-bound.