multiFaAcceleration: A program for the measurement of mutation velocity and acceleration from a four-species multiple alignment

Riley J. Mangan

August 21, 2021

1 Usage

multiFaAcceleration - Performs velocity and acceleration on a four way multiple alignment in multiFa format.

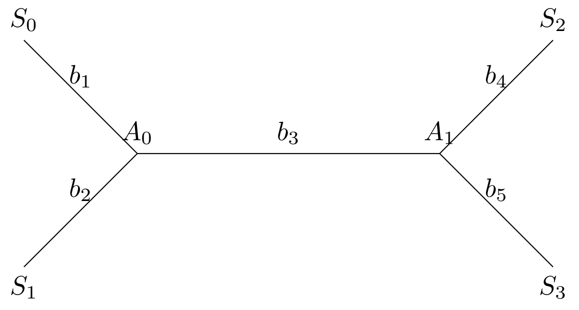
A four way multiple alignment must contain four species (index 0 to 3) in the topology that species 1 to 3 are successive outgroups of species 0.

Three bed files are returned. The first produces the velocity score, the second returns the acceleration score, and the third returns the initial velocity score for each window of the genome for aln[0].

multiFaAcceleration chromName in.fa velocity.bed acceleration.bed initialVelocity.bed

2 Distance-based phylogenetic inference with the Fitch-Margoliash method

Consider a phylogenetic tree with extant species $S_0 : S_3$, extinct ancestors $A_0 : A_1$, and branch lengths $b_1 : b_5$ with the following topology.



Consider that we can measure the pairwise mutation distance between any two extant species on this tree, represented by D_{ij}. The distance defined by the sum of branch lengths, also known as the patristic distance, is represented as d_{ij}. It follows that the pairwise distance between two extant species is equal to the sum of branch lengths separating those species on the phylogenetic tree shown above. Thus, we are able to produce the following system of linear equations.

$$\begin{aligned}
D_{01} &\approx d_{01} = b_1 + b_2 \\
D_{02} &\approx d_{02} = b_1 + b_3 + b_4 \\
D_{12} &\approx d_{12} = b_2 + b_3 + b_4 \\
D_{03} &\approx d_{03} = b_1 + b_3 + b_5 \\
D_{13} &\approx d_{13} = b_2 + b_3 + b_5 \\
D_{23} &\approx d_{23} = b_4 + b_5
\end{aligned} \tag{1}$$

If our interest is to study the genome evolution of S_0, we can define the mutation distance as b_1, the distance between that extant species and its most recent common ancestor with S_1. We can then define the initial mutation distance as b_3, the distance along the previous branch between A_0 and its ancestor A_1.

We can compute branch lengths using the method of Fitch and Margoliash [Fitch and Margoliash, 1967] with an alternating least squares optimization algorithm developed by Felsenstein [Felsenstein, 1997]. While the full details and derivations for these methods can be found in these two papers, I will briefly explain the method below.

Due to such phenomenon as back mutation and discrepancies in INDEL-sensitive distance metrics, it is not always possible to find a set of branch lengths for the above system of equations such that $D_{ij} = d_{ij} \; \forall \, i, j \in S$, where S is the set of all extant species. Thus, in the Fitch-Margoliash method, we aim to find a set of branch lengths B that minimizes the squared difference between the pairwise and patristic distances. In symbolic terms:

$$Q = \sum_{i \in S} \sum_{j \in S} w_{ij} (D_{ij} - d_{ij})^2$$

In the above expression, w_{ij} represents a weight for error terms, and serves to place more of the overall error in the estimated branch lengths on longer branches. This expression is of the form:

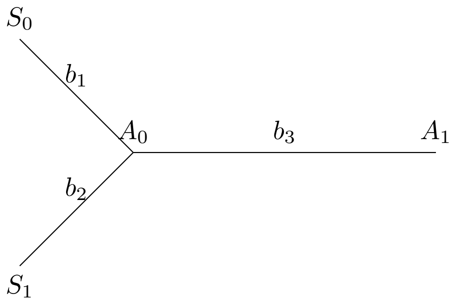
$$w_{ij} = \frac{1}{D_{ij}^2} \mid D_{ij} \neq 0$$

When $D_{ij} = 0$, the weight term w_{ij} approaches positive infinity. In this program, we approximate this limit by assigning an arbitrarily large number to w_{ij}. This value is 10^{20} by default, but can be controlled by the user with the option $zeroDistanceWeightConstant$.

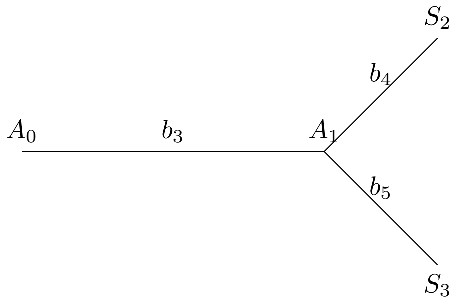
3 Tree reduction and subtree optimization

In the alternating least squares approach top optimize Q described by Felsenstein, we must first reduce the above tree, which contains two internal nodes A_0 and A_1, to subtrees containing only one internal node. These subtrees, which we refer to as the left and right subtrees, are shown below.

Left subtree:



Right subtree:



In the *multiFaAcceleration* program, these subtrees are produced with the *pruneLeft* and *pruneRight* helper functions, respectively. To find the optimal branch lengths for these subtrees, we must find the pairwise distances between the extant species nodes and the ancestral node which is now a leaf in the subtree. For the left subtree, Felsenstein demonstrates that we can calculate these distances as:

$$D_{0A_1} = \frac{w_{02}(D_{02} - b_4) + w_{03}(D_{03} - b_5)}{w_{02} + w_{03}} \mid w_{02} + w_{03} \neq 0$$
$$D_{1A_1} = \frac{w_{12}(D_{12} - b_4) + w_{13}(D_{13} - b_5)}{w_{12} + w_{13}} \mid w_{12} + w_{13} \neq 0$$

Similarly, for the right subtree:

$$D_{2A_0} = \frac{w_{02}(D_{02} - b_4) + w_{12}(D_{12} - b_5)}{w_{02} + w_{12}} \mid w_{02} + w_{12} \neq 0$$
$$D_{3A_0} = \frac{w_{03}(D_{03} - b_4) + w_{13}(D_{13} - b_5)}{w_{03} + w_{13}} \mid w_{03} + w_{13} \neq 0$$

As distance is necessarily a non-negative quantity, and the weight constant is set to an arbitrarily large number when $D_{ij} = 0$, the denominator in the above equations can never be zero, and does not need to be otherwise constrained.

A three-species subtree, which can be generalized to contain leaves a, b, and c with the corresponding branch lengths v_a, v_b, and v_c has the following optimal branch lengths:

$$v_a = \frac{(D_{ab} + D_{ac} - D_{bc})}{2}$$
$$v_b = \frac{(D_{ab} + D_{bc} - D_{ac})}{2}$$
$$v_c = \frac{(D_{ac} + D_{bc} - D_{ab})}{2}$$

4 Constraint for non-negative branch lengths

In some cases, the optimal set of branch lengths for the minimization of Q will include negative branch lengths. While this can be allowed by the user with the option *allowNegative*, this program constrains branch lengths to non-negative values by default. Briefly, when one or more branches from the 3-species subtrees are evaluated as negative values, they are set to 0. From Felsenstein 1997, if branch lengths v_b or v_c are set to zero in this fashion, the new optimal branch length for v_a, given that either of these terms are set to zero, can be approximated as:

$$\hat{v}_a = \frac{w_{ab}(D_{ab} - v_b) + w_{ac}(D_{ac} - v_c)}{w_{ab} + w_{ac}}$$

5 Algorithm for branch length calculations

With these equations in hand, we can now describe the algorithm for computing optimal branch lengths for the four-species tree, which is performed by the helper function *alternatingLeastSquares* in *multiFaAcceleration*. First, the set of output branch lengths $b1 : b5$ are initialized such that each branch length is equal to 1. In each iteration, the tree is first pruned to the left subtree, and the branch lengths $b_1 : b_3$ are then set to the optimal branch lengths for this subtree. Next, the tree is pruned into the right subtree, and the branches $b_3 : b_5$ are then set to the optimal values for this subtree. Once both optimizations have occured, the value of Q is calculated for the current branch lengths and compared to the value of Q observed in the previous iteration. If the difference between the estimates of Q falls below a user-specified level of error ϵ, the answer is returned. Otherwise, a new iteration is initiated. This heuristic will converge on a local minimum for Q and achieve stationarity in finite iterations. Due to the approximation used for non-negative branch lengths, it is possible stationarity will be achieved as the oscillation between two sets of branch lengths. In this case, we take the set of branch lengths associated with the lowest value of Q.

6 Genome-wide acceleration calculation

For a given four-way alignment in multiFa format, $gonomics : multiFaAcceleration$ calculates b_1 and b_3 using pairwise mutation distance (defined as the number of SNPs and INDELs, where each INDEL counts as one mutation regardless of length) for each window of a user-specified window size. Windows may be every possible window of the genome, or may be restricted to a particular subset of the genome using the option $-searchSpaceBed$, which enables the input of a bed file which specifies the regions that should be considered. The option $-searchSpaceProportion$ enables the user to consider all windows in which at least a user-specified proportion of bases are within the searchSpace.

We define $\mathbf{v}$ as the normalized mutation velocity, or the normalized rate of mutation over the branch b_1. To calculate $\mathbf{v}$, we calculate the average b_1 length $\overline{b_1}$ across all windows. For each window:

$$\mathbf{v} = \frac{b_1}{\overline{b_1}}$$

Similarly, the normalized initial rate of mutation, or the normalized rate of mutation over the branch b_3, can be calculated as:

$$\mathbf{v}_0 = \frac{b_3}{\overline{b_3}}$$

Where $\overline{b_3}$ is the average value of b_3 over all windows.

$\mathbf{v}$ and $\mathbf{v}_0$ have intuitive numerical interpretations. If $\mathbf{v} = 1$ for a particular window, the mutation rate in the branch b_1 is equal to the chromosome-wide average mutation rate. $\mathbf{v} = 2$ would be found in a region evolving twice as quickly, and $\mathbf{v} = 0.5$ in a region evolving at half the average rate. The same interpretations apply for $\mathbf{v}_0$, the rate of evolution along the branch b_3.

Finally, we define the quantity $\mathbf{a}$, for acceleration, as the normalized change in mutation rate between branches branches $b1$ and $b3$:

$$\mathbf{a} = \mathbf{v} - \mathbf{v}_0$$

The quantity $\mathbf{a}$ is equal to zero when the mutation rate along b_1 is equal to the mutation rate along b_2. As both $\mathbf{v}$ and $\mathbf{v}_0$ are normalized, this holds true even if b_1 and b_3 are not equal in absolute length, which will be the case when the extant species S_0, S_1, and S_2 are not separated by equal amounts of evolutionary time. Positive values for $\mathbf{a}$ indicate accelerated regions, and negative values suggest regions under negative acceleration, in which a region evolved at a slower rate along b_1 than b_3.

References

[Felsenstein, 1997] Felsenstein, J., 1997. An alternating least squares approach to inferring phylogenies from pairwise distances. *Syst Biol*, **46**(1):101–111.

[Fitch and Margoliash, 1967] Fitch, W. M. and Margoliash, E., 1967. Construction of phylogenetic trees. *Science*, **155**(3760):279–284.