University of California, Davis

Department of Computer Science

Master Project Report: Tree Estimator Program

Student
Li Li

Advisor
Nina Amenta

July 21, 2017

Contents

1 Introduction

An ultrametric tree is a tree with equal distances from the root to all leaf nodes, which can be generated based on extant taxa from DNA. All ultrametric trees referred in this report have branch lengths representing time. However, when a new fossil is discovered, it has to be added to a given phylogeny based on its landmark coordinates rather than its DNA. It is then necessary to determine an evolutionary tree that can reflect morphological change of all nodes in order to find a best fit for the new fossil. The distance of each branch in this new tree should be proportional to the morphological difference between its two end nodes.

The tree estimator is a program that produces an evolutionary tree with branch lengths reflecting the morphological differences between nodes from a given phylogeny (an ultrametric tree), and fits a fossil to the new evolutionary tree. The weights (landmark coordinates) of the internal nodes in a evolutionary tree are calculated based on [1]. The input data is an ultrametric tree in Newick format, landmark coordinates of all leaf nodes, and the landmark coordinates of the fossil that needs to be inserted to the tree. The first part of the output will be an updated phylogenetic tree with branch lengths representing the morphological change between nodes, and the second part is a phylogenetic tree with the foreign fossil fit to a branch that it is closest to in space.

2 History

This program is built upon Professor Amenta's working notes on Derivation of Weight for Internal Nodes (Appendix 1) and Insertion of Fossils in a Given Tree (Appendix 2).

2.1 Derivation of Weights for Internal Nodes

Professor Amenta's deep background notes on derivation of weights for internal nodes present two methods to compute the weights which control the shape of the skull (landmark coordinates) at the internal nodes of a phylogenetic tree.

2.1.1 Method One: Least Square

Method one justifies the weights using the idea of minimizing the squared Procrustes distance over the entire tree in statistics based on the two phylogeny literature [2, 3]. A covariance matrix describing how nodes that are adjacent to each other are more similar in shape than the ones that are further away can be drawn from an existing evolutionary tree. The least square technique is used to find the root's continuous variable first, and then the internal nodes' continuous variables.

2.1.2 Method Two: Squared-Change parsimony

Method two derives the weight of an internal node by calculating the equilibrium of its neighbors. This project implements the second method to calculate the weights of the internal nodes. This method assumes that the squared change in morphology is proportional to time along any branch in a given tree. This parsimony algorithm [1] minimizes the sum of squared changes along all branches in a tree to reconstruct a phylogenetic tree with continuous-valued characters. When the squared changes are inversely weighted by branch lengths (time), the squared-change parsimony will return an optimal reconstruction of an evolutionary tree. Given a continuous variables x_i as the leaf nodes in a tree, we want to minimize the sum of the squared lengths of branches by doing $min \sum_{edges(s,t)} \left(v^{'}_{s,t}\right)^2$, with $v^{'}_{s,t} = d(x_s - x_t)$. Let $v_{s,t}$ be the fixed branch length (time), node s be a node in a given evolutionary tree, a, b, and c are its three neighbors. The algorithm will weight the squared changes inversely by the branch lengths by doing $\frac{(x_s-x_a)^2}{v_{s,a}} + \frac{(x_s-x_b)^2}{v_{s,b}} + \frac{(x_s-x_c)^2}{v_{s,c}}$ and minimize the value to reconstruct the optimal evolutionary tree.

2.2 Insertion of Fossils in a Given Tree

Professor Amenta's working notes on insertion of fossils in s given tree proposes a process to fit a fossil in an evolutionary tree. This calculation is based on the principal of squared-change parsimony, which states that the "best" tree minimizes the sum of the squares of the morphological difference along each edge. The morphological difference is calculates as the Euclidean distance between two nodes along a branch.

3 Equations, Formulae

This section describes the how the following variables are calculated: the internal node weight, new branch lengths proportional to the morphological difference between nodes, the Euclidean distance from the inserted fossil to any branch of the tree, and its best fit in the tree.

3.1 Computing the Intermediate Shapes at the Internal Nodes

3.1.1 Computing the Internal Node States

The intermediate shapes, also referred to weights (landmark coordinates) of internal nodes in a evolutionary tree are calculated based on the equilibrium of their neighbors introduced in Section 2.1.2. This method assumes that the squared change in morphology is proportional to time along any branch. Although this assumption is not realistic, the goal is to get as close to the internal nodes values as possible, so they are set up to be solved by a linear system through minimization.

The values at leaf taxa x_i are given. Our goal is to set the optimal values x_s at each internal node s (see figure 1 as an example). Let $v_{s,t}$ be the input branch length between s and t, which is time. Considering an internal node s with neighbor a, b, c (or just a, b if s is the root). Squared change weighted by the branch lengths is calculated as:

$$\frac{(x_s - x_a)^2}{v_{s,a}} + \frac{(x_s - x_b)^2}{v_{s,b}} + \frac{(x_s - x_c)^2}{v_{s,c}} \tag{1}$$

The derivative is taken and set to zero for minimization purpose:

$$\frac{x_s}{v_{s,a}} + \frac{x_s}{v_{s,b}} + \frac{x_s}{v_{s,c}} = \frac{x_a}{v_{s,a}} + \frac{x_b}{v_{s,b}} + \frac{x_c}{v_{s,c}} \tag{2}$$

x_s is the weighted average of x_a, x_b, x_c, which is inversely proportional to the branch lengths.

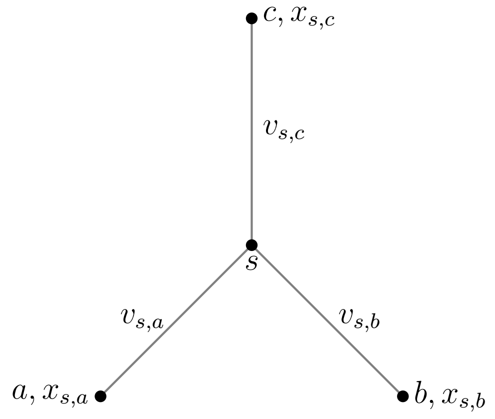

Figure 1: An example of input tree to calculate weight for internal node s

3.1.2 An Example of Computing the Internal Nodes Weights in A Phylogenetic Tree

This section presents a brief example of how to calculate the weights of all internal nodes in an ultrametric tree with the weights of all leaf nodes given.

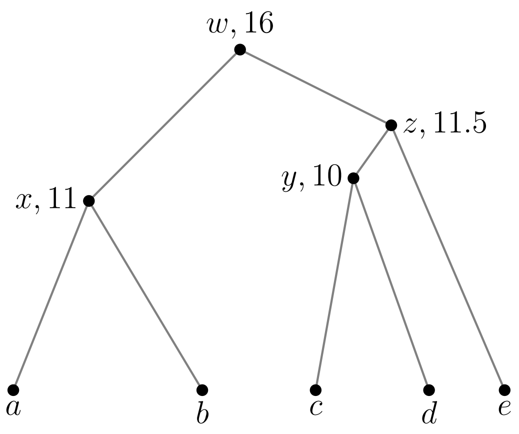

Figure 2: An example of input tree, internal nodes labels are ID, time

With a phylogenetic tree given in figure 2, the linear system to be solved is:

$$\frac{x_x}{11}+\frac{x_x}{11}+\frac{x_x}{5}=\frac{x_a}{11}+\frac{x_b}{11}+\frac{x_w}{5}$$
$$\frac{x_y}{10}+\frac{x_y}{10}+\frac{x_y}{1.5}=\frac{x_c}{10}+\frac{x_d}{10}+\frac{x_z}{1.5}$$
$$\frac{x_z}{1.5}+\frac{x_z}{11.5}+\frac{x_z}{4.5}=\frac{x_y}{1.5}+\frac{x_e}{11.5}+\frac{x_w}{4.5}$$
$$\frac{x_w}{5}+\frac{x_w}{4.5}=\frac{x_x}{5}+\frac{x_z}{4.5}$$

We can get the weights for internal nodes x, y, z, w by solving these above equations:

$$\begin{aligned}
Weight\ for\ x: &\ 0.356a + 0.356b + 0.092c + 0.092d + 0.104e \\
Weight\ for\ y: &\ 0.083a + 0.083b + 0.308c + 0.308d + 0.218e \\
Weight\ for\ z: &\ 0.108a + 0.108b + 0.250c + 0.250d + 0.283e \\
Weight\ for\ w: &\ 0.226a + 0.226b + 0.175c + .0175d + 0.198e
\end{aligned}$$

3.1.3 Verifying Internal Node Weight

In order to verify the results computed previously, the most straight forward way is to check equality of left hand side and the right hand side of equation:

$$\frac{x_s}{v_{s,a}} + \frac{x_s}{v_{s,b}} + \frac{x_s}{v_{s,c}} = \frac{x_a}{v_{s,a}} + \frac{x_b}{v_{s,b}} + \frac{x_c}{v_{s,c}} \tag{3}$$

3.2 Computing New Branch Length

After computing the weight of all internal nodes (including the root) of a given evolutionary tree, the next step is updating the branch lengths. As mentioned previously, $v_{s,t}$ is the branch distance from node s to node t, which represents time. Let $v'_{s,t}$ be the new branch distance between node s and t that represents the morphological difference between these two nodes.

$$v'_{s,t} = d(x_s - x_t) \tag{4}$$

Given that the weight of each node in the tree is landmark coordinates, $v'_{s,t}$ is the Euclidean distance between node s and t in space.

3.3 Computing the Shortest Distance from a Fossil to Any Branch in an Phylogenetic Tree

3.3.1 Dot Product Method

The simplest approach to insert a fossil is to find the branch or node to which it is closest. In the case of landmark coordinates being node weight, we can view this as calculating the distance from a point to a segment in n-dimensional space.

To compute the distance $d(P, L)$ from an arbitrary point P to segment S given by a parametric equation with P_0 and P_1 as endpoints, supposed P_b is the base of the perpendicular dropped from P to S, there are three possible cases to consider when calculating the distance from P to S, since P_b may be outside the range of S.

1. P_b is within the range of segment S, vector P_0P_b is the projection of vector P_0P on to S (P_0P_1), any point P_t on segment S can be represented as:

$$P_t = P_0 + t(P_1 - P_0),\ 0 < t < 1 \tag{5}$$

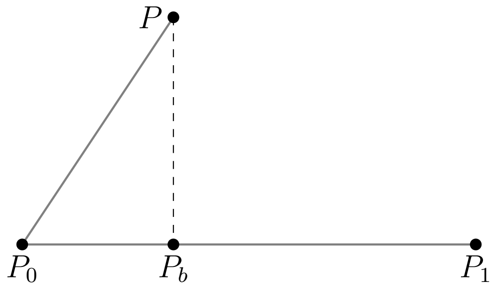

Figure 3: P_b is within the range of P_0P_1,

2. P_b is outside the range of segment S and is closer to end point P_0, the distance from P to S is PP_0, P_b can be represented as:

$$P_b = P_0 + t(P_1 - P_0),\ t < 0 \tag{6}$$

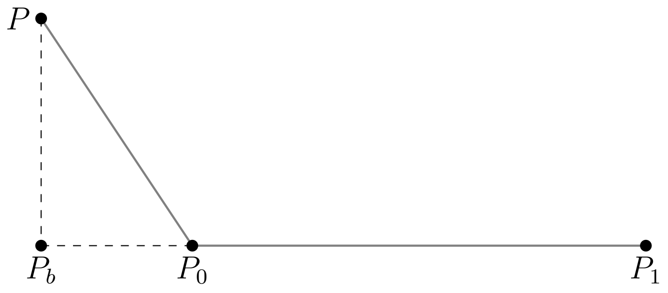

Figure 4: P_b is outside the range of P_0P_1, left to P_0

3. P_b is outside the range of segment S and is closer to end point P_1, the distance from P to S is PP_1, P_b can be represented as:

$$P_b = P_0 + t(P_1 - P_0),\ t > 1 \tag{7}$$

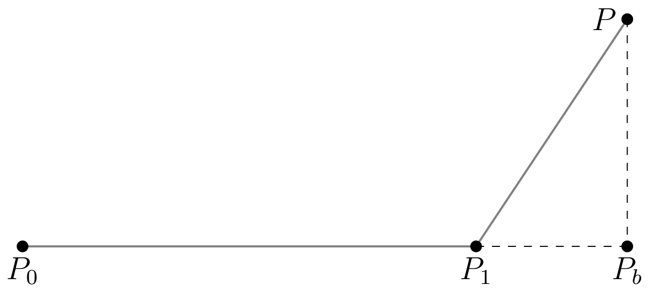

Figure 5: P_b is outside the range of P_0P_1, right to P_1

Algorithm 1 shows how the program calculates the Euclidean distance from a point P to a segment S in n-dimensional space using the dot product method.

Algorithm 1 Pseudo code to compute the Euclidean distance from any point P to segment S in n-dimensional space

```
1: function EUCLIDEAN DISTANCE(point $P$, segment $P_0P_1$)
2:     $v = P_1 - P_0$
3:     $w = P - P_0$
4:     $t = \text{dot}(w \cdot v) \;/\; \text{dot}(v \cdot v)$
5:     if $t \leq 0$ then                          ▷ $P$ to the left of $P_0$
6:         return distance($P, P_0$)
7:     else if $t \geq 1$ then                     ▷ $P$ to the right of $P_0$
8:         return distance($P, P_1$)
9:     else                                        ▷ $P_b$ within the range of $P_0P_1$
10:        $P_b = P_0 + t \cdot v$
11:        return distance($P, P_b$)
```

3.3.2 Heron's Formula

Heron's formula is implemented to compute the distance from any point P to segment S to validate the result computed from the previous dot product method. Heron's formula gives the area of a triangle without requiring a specific side as base and the height from the corresponding vertex to the chosen base. Given a segment S with end points P_0 and P_1, and a point P (which is not on the same line with S), a triangle $\triangle PP_0P_1$ can always be formed with these three points as vertices.

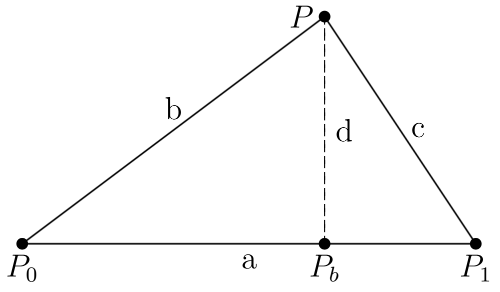

Figure 6: $\triangle PP_0P_1$ formed by point P and segment S

We use a to represent the distance from P_0 to P_1, b to represent the distance from P to P_0, c to represent the distance from P to P_1, d to represent the distance from P to P_0P_1 (hence the height of the triangle from vertex P to base P_0P_1), $s = \frac{a+b+c}{2}$ to represent the semiperimeter of $\triangle PP_0P_1$, and A to represent the area of $\triangle PP_0P_1$. Heron's formula states that:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \tag{8}$$

The definition of the area of a triangle with P as vertex and P_0P_1 as the base is also computed as:

$$A = \frac{1}{2}\, d \cdot a \tag{9}$$

The distance from P to S can be computed as:

$$d = \frac{2\sqrt{s(s-a)(s-b)(s-c)}}{a} \tag{10}$$

Heron's formula is used as an extra step to check if the dot product produces the correct value for distance from P to its projection P_b on segment S. Note that this method will only return the shortest distance from the chosen vertex of the triangle to its corresponding base, regardless of whether it is obtuse or acute.

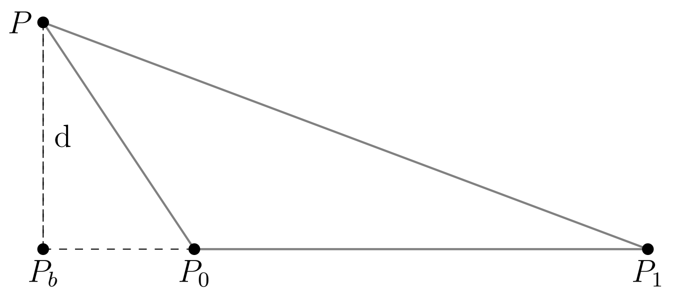

Figure 7: $\triangle PP_0P_1$ (obstuse) formed by point P and segment S

3.4 Computing the intermediate node s to fit foreign fossil to a tree

After finding the edge that fossil f with data x_f is closest to in the evolutionary tree (we will refer that edge as yz), the next step is to add f as a new node. This is done by adding f as a child of an intermediate node s. We introduce two levels of calculation to compute s: level 1 places a node s somewhere on edge yz, which is the projection of f on edge yz; level 2 finds s by minimizing the branch length to f from s

3.4.1 Level 1 Fitting

s lies somewhere on edge yz, which is the projection of f to the edge as computed in Section 3.3.1. By introducing value t, we can determine the exact location of s and its value x_s. If $0 < t < 1$, s lies between the node y and z on the edge; if $t \leq 0$ or $t \geq 1$, s is actually either node y or node z.

3.4.2 Level 2 Fitting

In this case, we place s so as to minimize the length of the subtree connecting s, y, and z. This is achieved by place s at the barycenter of these three points:

$$(x_s - x_f)^2 + (x_s - x_y)^2 + (x_s - x_z)^2 \tag{11}$$

using the linear equation:

$$x_s = \frac{x_f + x_y + x_z}{3} \tag{12}$$

4 Implementation

4.1 Language and Libraries

The program is written in Python, using Numpy, Scipy, and ete3 libraries.

4.1.1 Numpy and Scipy Libraries

The Numpy library is used to solve the linear system to compute the weights of internal nodes, and the distance matrix computation is imported from Scipy library to compute the Euclidean distance between two 1-D arrays. The detailed instructions on downloading and installing the libraries can be found on the official website: `https://www.scipy.org/scipylib/download.html` .

4.1.2 ETE Toolkit

ETE (environment for tree exploration) toolkit is a Python framework for phylogenetic tree construction, analysis, and visualization. ete3 library is imported in the program to read the evolutionary tree in Newick format, traverse the tree, and write the updated tree with new branch lengths to a Newick file. The detailed instructions on downloading and installing the library can be found on `http://etetoolkit.org/download/`, it is recommend to install ETE using Conda for Mac and Linux users. Listing 1 is an example of a phylogeny in Newick format, figure 7 is a visualization of the sample Newick tree using the ete3 library. Note that the leaf nodes are not placed on the same vertical line due to the discretization error in the drawing package of ETE library.

Listing 1: A phylogenetic tree in Newick format

```
(((((mandrillus:3.1,cercocebus:3.1)a:4,(lopocebus:4.1,(papio:4,
theropithecus:4)c:0.1)b:3)d:2,macaca:9.1)e:2.5,
(allenopithecus:9.3,(chlorocebus:8.1,(cercopithecus:8,
miopithecus:8)h:0.1)g:1.2)f:2.3)k:4.5,((presbytini:8.8,
pygathrix:8.8)i:1.2,colobini:11)j:5)l;
```

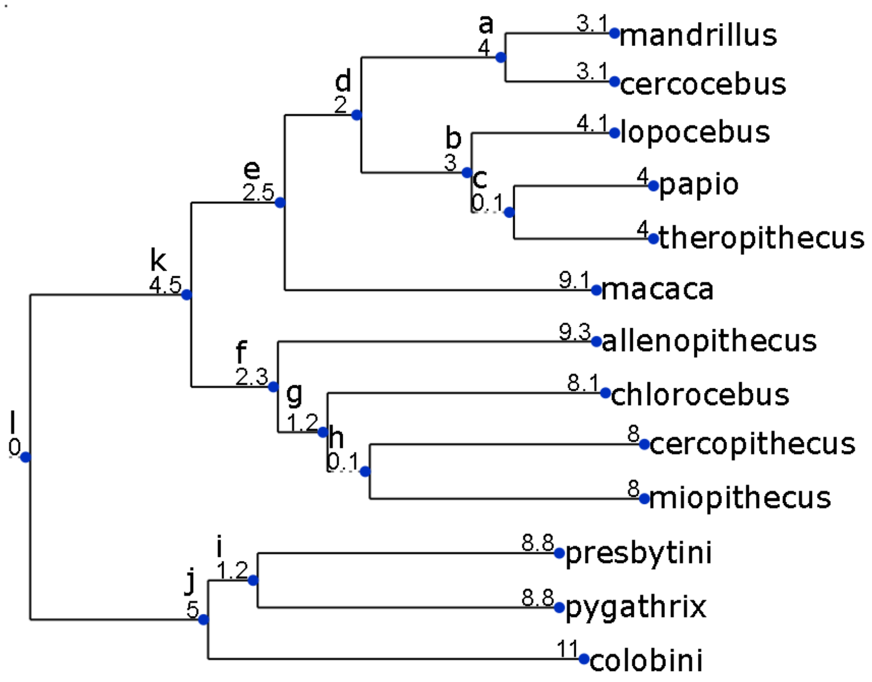

Figure 8: Evolutionary tree visualized by ETE toolkit

4.2 Software

4.2.1 Software Functionality

The tree estimator software *fossil.py* takes an ultrametric tree in Newick format, landmark coordinates of the input tree's leaf nodes, and optionally the landmark coordinates of the fossil to be fit to the tree as input data, the last input will be optional depending on the input data. There are a total of three commands for the user to choose: *n*, *v*, and *f*.

4.2.2 Input File Type

Evolutionary Tree The evolutionary tree should be a Newick file which is similar to the tree provided in Listing 1, Section 4.1.2 previously.

Landmark Coordinates The landmark coordinates should be in a text file with each data point separated either by comma or space.

4.2.3 Command Line Options

-n The *-n* command line option requires two input files from user, which are the original phylogenetic tree and a directory containing the landmark coordinates of its leaf nodes. The output will be an updated phylogenetic tree with branch lengths representing morphological change named "new_tree.nw" and a directory of landmark coordinates including the ones for internal nodes named "new_landmarks". The program runs like:

```
$ python fossil.py original_tree.nw landmark -n
```

-v The *-v* command line option requires two input files from user, which are the original phylogenetic tree and a directory containing the landmark coordinates of its leaf nodes. The output will be a list of values printed on user screen showing the differences between the left hand side and right hand side of equation $\frac{x_s}{v_{s,a}} + \frac{x_s}{v_{s,b}} + \frac{x_s}{v_{s,c}} = \frac{x_a}{v_{s,a}} + \frac{x_b}{v_{s,b}} + \frac{x_c}{v_{s,c}}$ in Section 3.1. The program runs like:

```
$ python fossil.py original_tree.nw landmark -v
```

-f The *-f* command line option requires three input files from user, which are the original phylogenetic tree, a directory containing the landmark coordinates of its leaf nodes, and the landmark coordinates. The output will be an updated phylogenetic tree with branch lengths representing morphological change named "new_tree.nw", a directory of landmark coordinates including the ones for internal nodes named "new_landmarks", a new tree "t_level1.nw" with the foreign fossil inserted using level 1 method, and a new tree "t_level2.nw" with the foreign fossil inserted using level 2 method. The program runs like:

```
$ python fossil.py original_tree.nw landmark foreign.txt -f
```

5 Results

5.1 Evolutionary Tree with Updated Branch Length

5.1.1 Computing and Updating Weights of Internal Nodes

A phylogenetic tree, referred to as $t1$, with branch lengths representing time is used as the input tree. Landmark coordinates of all leaf nodes of the tree is given to compute the weights of internal nodes (including the root).

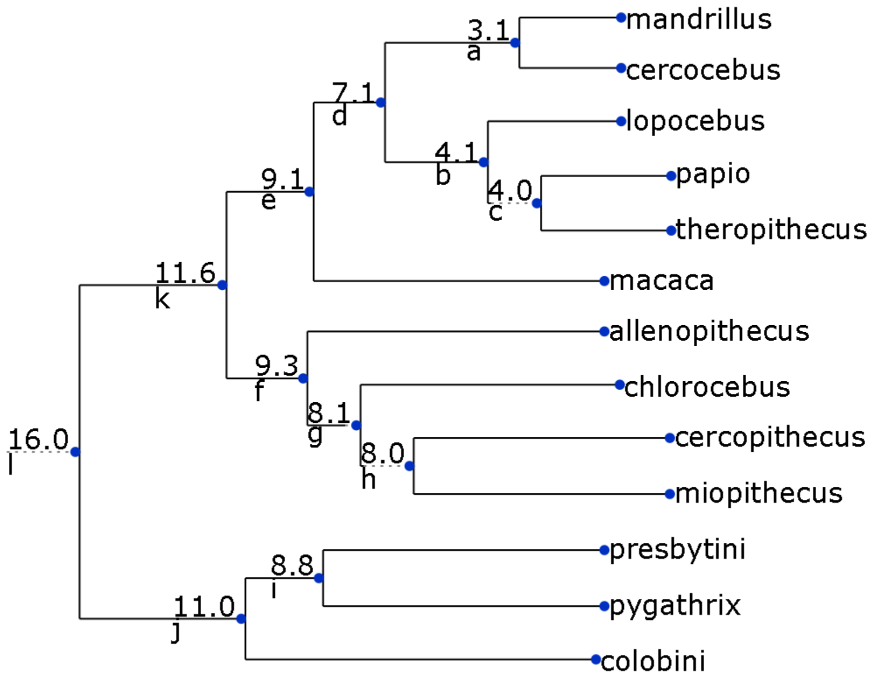

Figure 9: Original evolutionary tree

5.1.2 Computing Branch Length Proportional to Morphological Change

After calculating the weights for all internal nodes, the next step is to update all branch lengths to represent the morphological change between nodes and save all these changes to a new tree $t2$.

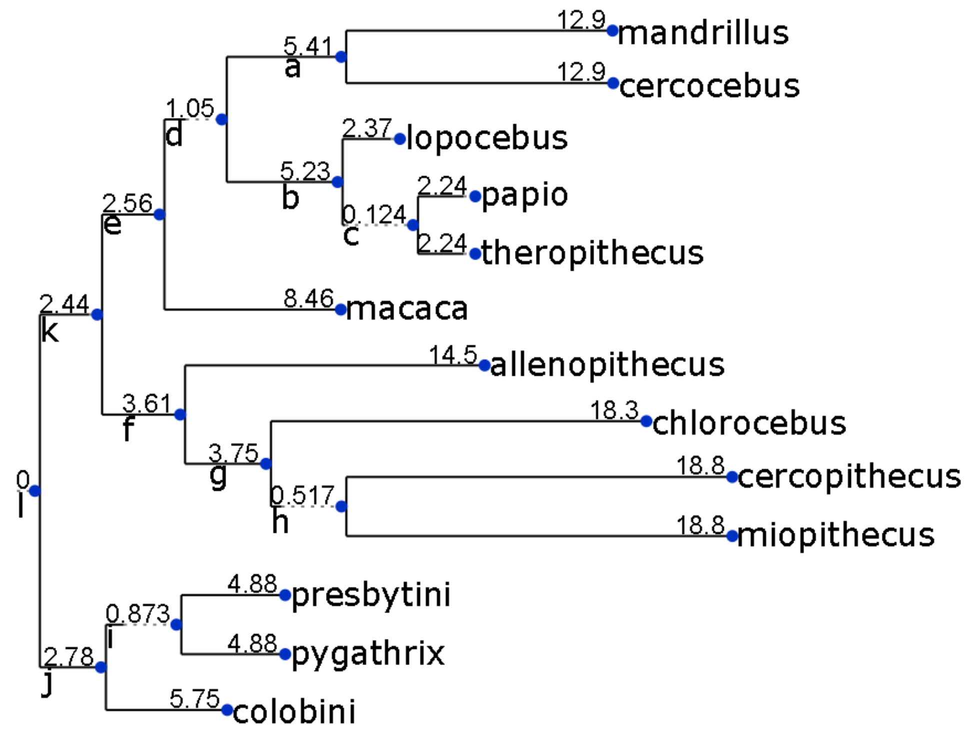

Figure 10: New tree with branch lengths representing morphological change

5.1.3 Verifying Internal Node Weight

As mentioned in Section 3.1.3, it is necessary to make sure that the weights of internal nodes are computed correctly. For each internal node x, we need to prove that equation $\frac{x_x}{v_{x,a}} + \frac{x_x}{v_{x,b}} + \frac{x_x}{v_{x,c}} = \frac{x_a}{v_{x,a}} + \frac{x_b}{v_{x,b}} + \frac{x_c}{v_{x,c}}$ always holds. x_i represents the weights of node i, a, b, and c are the neighboring nodes of x. $v_{x,i}$ represents the length of branch $x - i$, which is time. When running command:

```
$ python fossil.py original_tree.nw landmark -v
```

All internal nodes return the same value on both sides of the equation, hence the landmark coordinates are calculated properly. The detailed result of this verification step is included in Appendix 3 Section 3.

5.2 Fitting a Fossil

5.2.1 Dataset Description

Due to the limited size of dataset, all leaf nodes are used to test the level 1 and level 2 refit of fossil insertion. The goal is to examine if the leaf nodes are placed relatively close to their original location in the evolutionary tree. The landmark coordinates used as test data are from the leaf nodes (name of primate species) listed below: *Mandrillus, Cecocebus, Lophocebus, Papio, Theropithecus, Macaca, Allenopithecus, Chlorocebus, Cercopithecus, Miopithecus, Presbytini, Pygathrix, Colobini*, a total of 13 leaf nodes.

Two methods are used to test the program. The first method does not make any change to the original tree, each node is used as a “test node” to be fit to the updated tree. Results are examined by how close the test node is from its original position in the tree after the insertion. The second method will make changes to the original tree, this is done so by removing a leaf node before computing the tree’s internal nodes weights, and then compute the weights of internal nodes to update the tree’s branch lengths. At the step of fitting a node to the tree, the removed leaf node will be inserted to the tree. This node should be relatively close to its original position in the tree, or have a relatively short distance to the edge of its closest neighbor in the original tree.

Each leaf node is viewed as a “foreign fossil” (also referred to as the test node) to be inserted to the new evolutionary tree computed from the original ultrametric tree with branch distance representing the morphological difference between two nodes. For the level 1 refit, the program should be able to attach the test node to its corresponding leaf node in the evolutionary tree, and the distance from this node to the corresponding leaf node in space should be 0. As for the level 2 refit, node s should be the parent node for the test node and its corresponding leaf node, the distance from s to the two noes should be the same in space.

A full list of the test results can be found in Appendix 3.

5.2.2 Method One

Level 1 Fitting This is an example of inserting node *macaca* to $t2$ with level 1 fitting. The node is attached to it’s original location the tree, which is the optimal position for fitting.

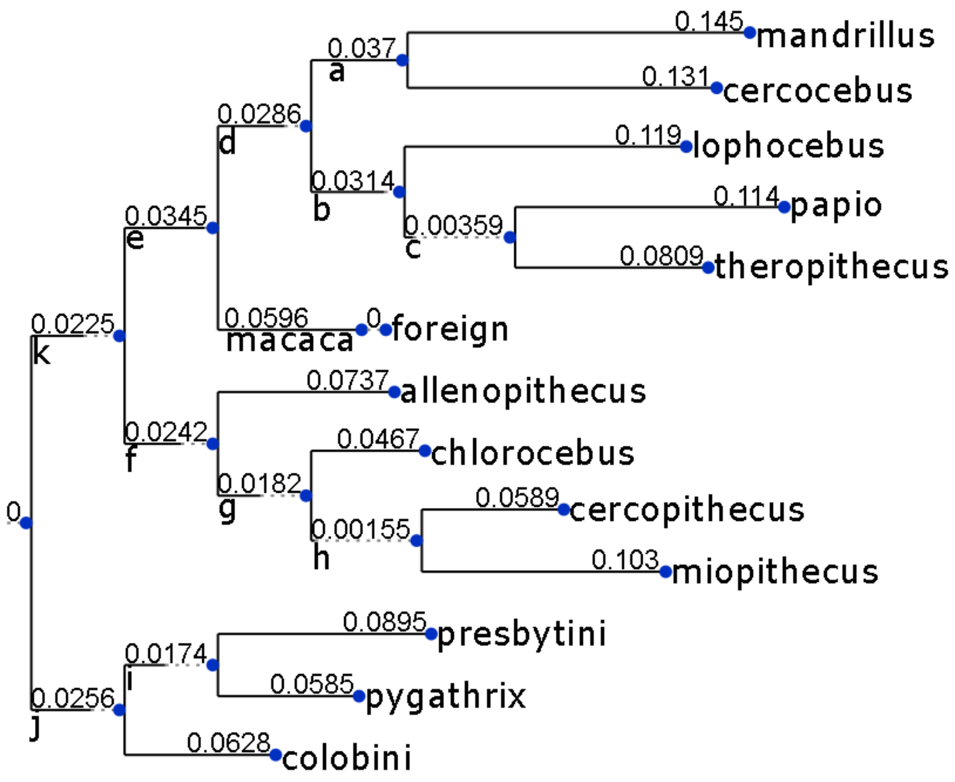

Figure 11: Level 1 fitting of *macaca*

Level 2 Fitting This is an example of inserting node *macaca* to $t2$ with level 2 fitting. The node is attached to an intermediate node s together with the original *macaca* node in the tree with equal distance to s, which is the optimal position for fitting.

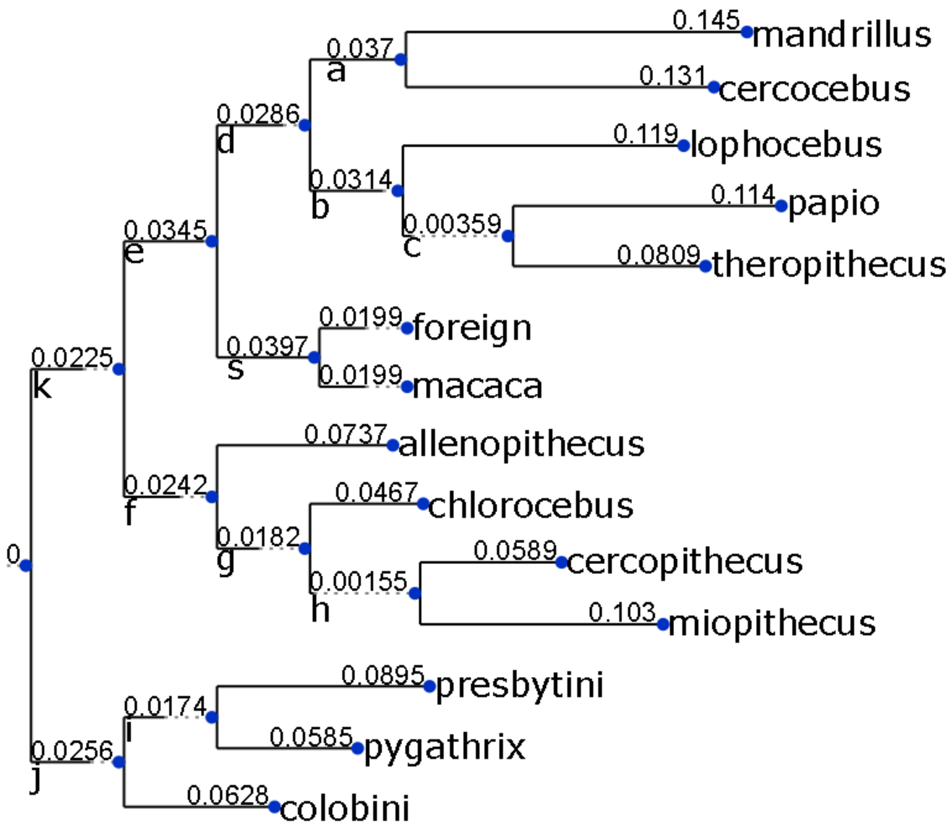

Figure 12: Level 1 fitting of *macaca*

5.2.3 Method Two

Removing a Leaf Node from Original Tree Figure 12 shows $t1$ with node *theropithecus* removed, figure 13 shows the updated $t2$ computed from $t1$ with *theropithecus* removed.

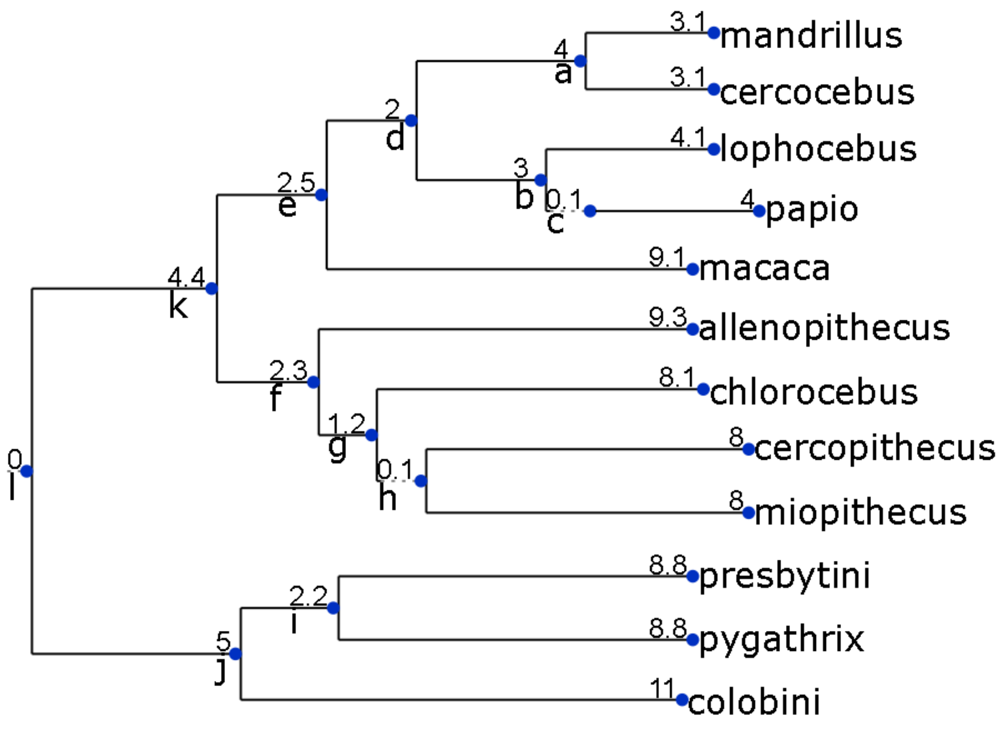

Figure 13: Ultrametric tree with node *theropithecus* removed

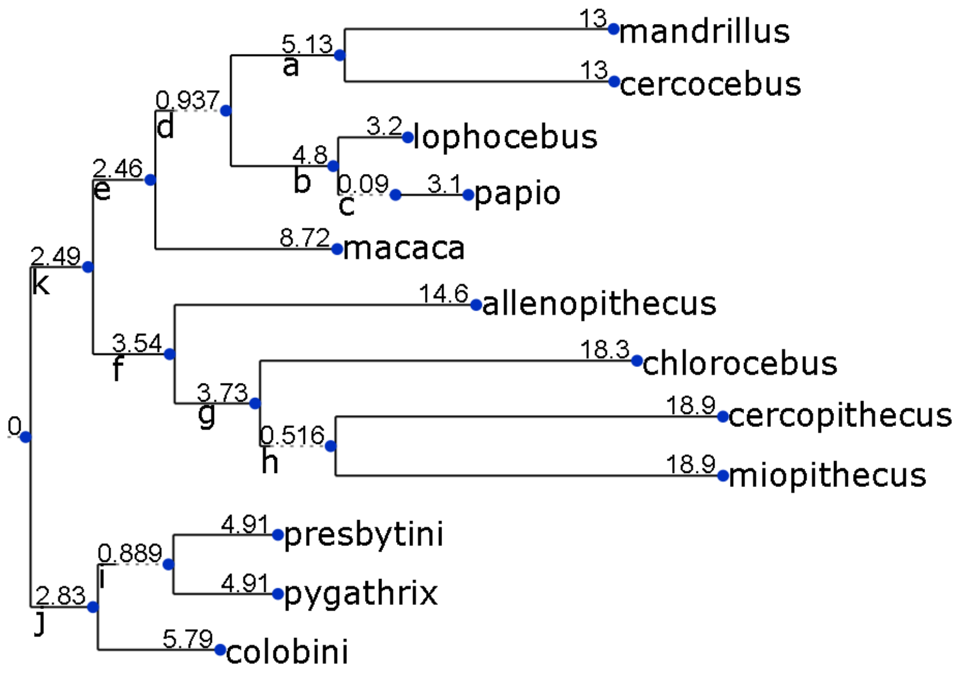

Figure 14: Ultrametric tree with node *theropithecus* removed

Level 1 Fitting This is an example of inserting *theropithecus* with level 1 after removing it from the original tree, this node is added to the position that is closest to *papio*, which is the optimal location. *papio* is the sister node of *theropithecus* is removed in the original tree.

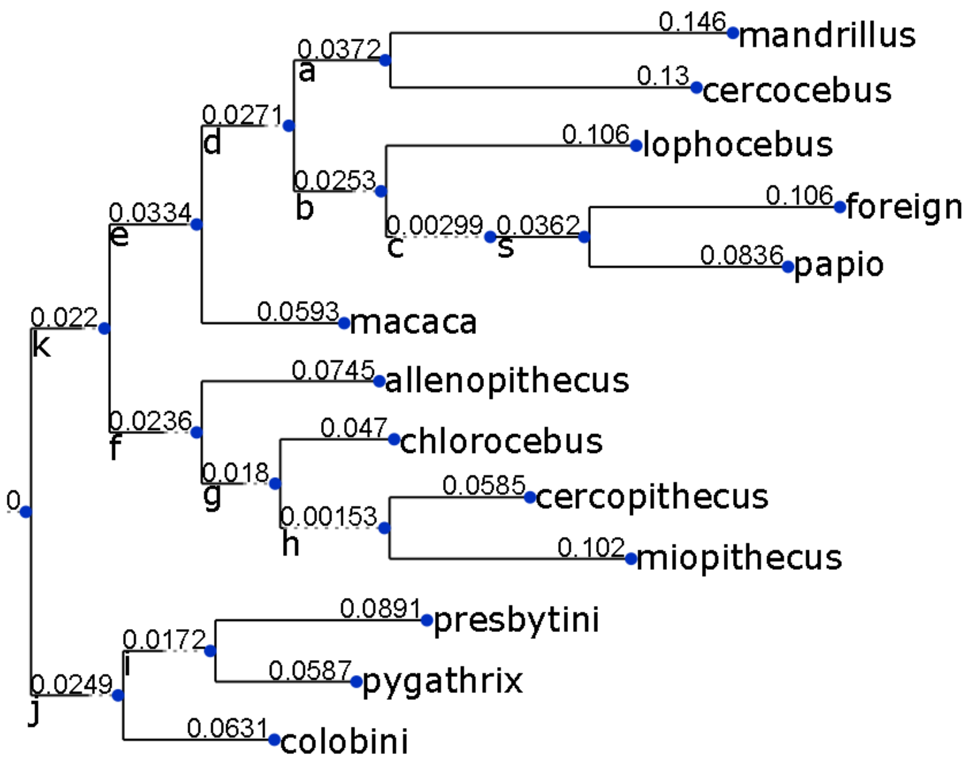

Figure 15: Ultrametric tree with node *theropithecus* removed

Level 2 Fitting This is an example of inserting *theropithecus* with level 12 after removing it from the original tree, the results is very similar to the level 1 fitting, with *theropithecus* placed in the optimal location.

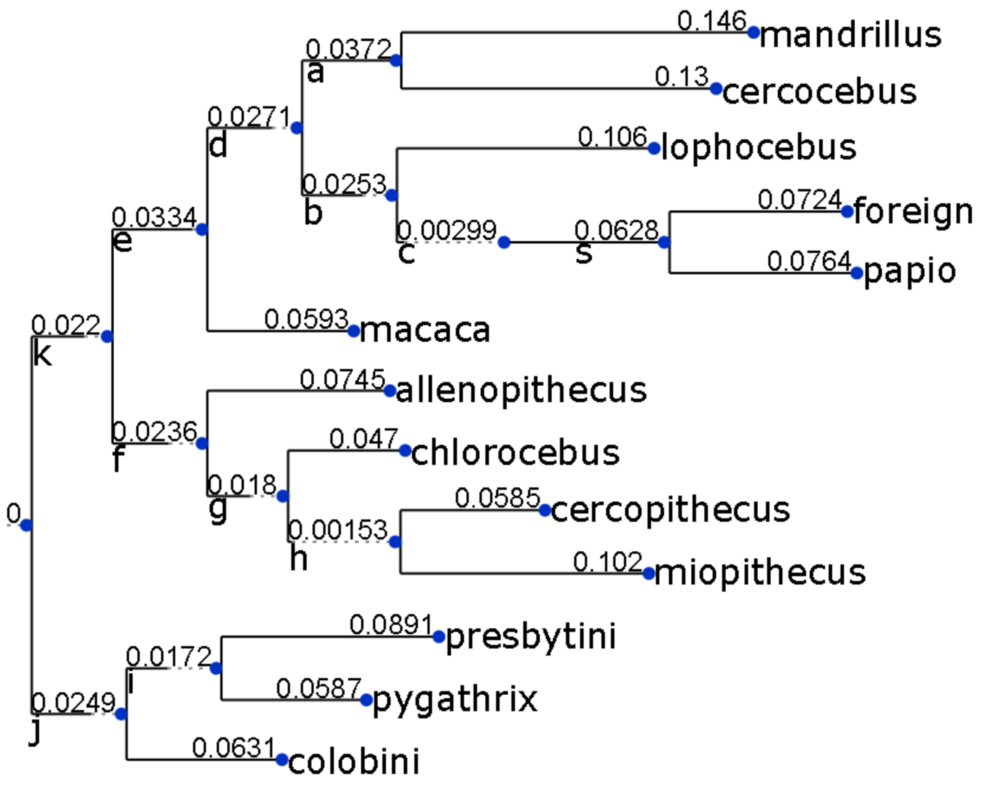

Figure 16: Ultrametric tree with node *theropithecus* removed

References

[1] Wayne P Maddison. Squared-change parsimony reconstructions of ancestral states for continuous-valued characters on a phylogenetic tree. *Systematic Biology*, 40(3):304–314, 1991.

[2] Emilia P Martins and Thomas F Hansen. Phylogenies and the comparative method: a general approach to incorporating phylogenetic information into the analysis of interspecific data. *The American Naturalist*, 149(4):646–667, 1997.

[3] F James Rohlf. Comparative methods for the analysis of continuous variables: geometric interpretations. *Evolution*, 55(11):2143–2160, 2001.