STATISTICAL ANALYSIS OF SPECIES TREE INFERENCE

By

Andres Dajles

RECOMMENDED: [illegible]
Dr. Ron Barry

Margaret Short
Dr. Margaret Short

[illegible]
Dr. Scott Goddard

[illegible]
Dr. Elizabeth Allman

[illegible]
Dr. John Rhodes
Advisory Committee Chair

[illegible]
Dr. John Rhodes
Chair, Department of Mathematics and Statistics

STATISTICAL ANALYSIS OF SPECIES TREE INFERENCE

A

PROJECT

Presented to the Faculty

of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Andres Dajles, B.A.

Fairbanks, Alaska

May 2016

Abstract

It is known that the STAR and USTAR algorithms are statistically consistent techniques used to infer species tree topologies from a large set of gene trees. However, if the set of gene trees is small, the accuracy of STAR and USTAR in determining species tree topologies is unknown. Furthermore, it is unknown how introducing roots on the gene trees affects the performance of STAR and USTAR. Therefore, we show that when given a set of gene trees of sizes 1, 3, 6 or 10, the STAR and USTAR algorithms with Neighbor Joining perform relatively well for two different cases: one where the gene trees are rooted at the outgroup and the STAR inferred species tree is also rooted at the outgroup, and the other where the gene trees are not rooted at the outgroup, but the USTAR inferred species tree is rooted at the outgroup.

Table of Contents

List of Figures

Chapter 1

Introduction

A common goal in genomics research is to find the evolutionary relationship between different species. Evidence for this relationship arises from the way genes mutate as they are transmitted from one generation to the next. Given the sequences of a particular gene from several species, a phylogenetic gene tree can be inferred using the classical computational methods for constructing phylogenies. However, it is known that the topology of individual gene trees does not always match the topology of the species tree for understood biological reasons. In addition, from a mathematical point of view, we understand gene trees as lineages within a population that form according to the coalescent model, which also explains the topological mis-match phenomenon. Now, if the branch lengths of the species tree are long enough, then the overwhelming majority of gene trees coming from the coalescent model will have the same topology as the species tree, which means that the species tree estimation will simply result in the same topology as most of the gene trees. However, if the branch lengths are short, then many gene trees will not match the species tree topology, and consequently the species tree topology estimation becomes more difficult.

In light of this issue, we find that estimating the topology of the species tree from a set of gene trees requires more elaborate statistical analysis. This is because the standard statistical approaches of maximum likelihood or Bayesian analysis can be too computationally intense. Therefore, we must consider methods that are fast but also statistically consistent. We will be exploring variants of two methods, namely STAR and USTAR, for inferring species tree topologies from gene tree topologies that differ in how the root of the trees is located. Thus, we will simulate gene trees according to the coalescent model, then we will subject these gene trees and inferred species trees to different ways of rooting, and then we will analyze their impact on species tree topology estimation.

In this paper, we begin by introducing the coalescent model in chapter 2, which is key in understanding how the gene trees are simulated. Then in chapter 3, we will explain STAR and USTAR, which are the algorithms used to estimate the species tree topology from a set of gene trees. Chapter 4 is a stand alone chapter where we explain RFsplit and RFclade distances, which are the metrics we use to measure differences in tree topologies. Chapter 5 introduces the framework of our simulations and the plots that are used to present our results. Finally, in chapter 6 we show the results of our simulations and the conclusions we draw from analyzing them.

Chapter 2

The Coalescent Model

Our study involves the simulation analysis of gene trees to infer a species tree. The topologies of these gene trees greatly affect our results, which means that we must understand how these gene trees are formed. The model that describes their formation and underlies their analysis is known as the multi-species coalescent model.

2.1 The Wright-Fisher Model

The coalescent model states that within a single population, viewed backwards in time on a suitable time scale, the rate at which any fixed pair of lineages coalesce into one lineage is constant, and that in addition, these coalescent events are independent and identically distributed for each pair of lineages. To better understand the ideas behind this model, let's begin with a more intuitive, discrete version of it; the Wright Fisher Model.

Imagine that we have a series of generations G_i, where $i \in \{0, 1, 2, 3, \ldots, n\}$, and where each generation G_i has a fixed population of 10 individuals; in other words, $G_i = \{g_{1_i}, g_{2_i} \ldots g_{10_i}\}$, as shown in Figure 2.1.

Now, consider any two individuals, say g_{1_0} and g_{4_0}, from the most recent generation, G_0. We wish to know how a particular gene x was transmitted to these individuals from the previous generation G_1 . Since there are 10 individuals in generation G_1, then we know that g_{1_0} could have inherited gene x from any of those 10 individuals. Without loss of generality, suppose that g_{1_0} inherited gene x from g_{3_1} in G_1. Then, there is a $\dfrac{1}{10}$ probability that g_{4_0} also inherited gene x from g_{3_1}. This occurrence, that a particular gene from various individuals in one generation comes from the same individual in the previous generation, is known as a *coalescent event.*

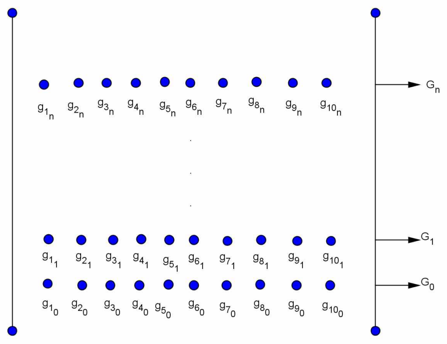

Figure 2.1: A sequence of generations G_i for $i \in \{1, 2, \ldots, n\}$ where each generation has 10 individuals, namely g_{j_i} where $j \in \{1, \ldots, 10\}$ and $g_{j_i} \in G_i$

There is also the case where g_{4_0} may inherit gene x from an individual other than g_{3_1} in the G_1 generation. In this case, the lineages of gene x for g_{1_0} and g_{4_0} do not coalesce in G_1, and the probability of this event is $1 - \frac{1}{10}$. Continuing this line of thought, if the lineages do not coalesce in G_1, then maybe they will coalesce in generation G_2. Thus, the probability that such lineage coalesce in generation G_2 is $\left(1 - \frac{1}{10}\right)\frac{1}{10}$, and the probability that the two lineages for gene x coalesce in at most two generations before the present is given by $\frac{1}{10} + \left(1 - \frac{1}{10}\right)\frac{1}{10}$.

More generally, if we define a random variable X as the number of generations until two distinct lineages coalesce, and we let N be the population size, then

$$P(X \leq n) = \sum_{i=1}^{n} P(X = i) = \sum_{i=1}^{n} \left(1 - \frac{1}{N}\right)^{i-1} \frac{1}{N} = 1 - \left(1 - \frac{1}{N}\right)^{n}.$$

This result indicates how $P(X \leq n)$ depends on N. A larger value of N results in a smaller value of $P(X \leq n)$, which means that the time for the two lineages to coalesce becomes longer. As a matter of fact, we find that the expected time to coalescence is given by

$$\sum_{n=1}^{\infty} nP(X = n) = \sum_{n=1}^{\infty} \frac{n}{N}\left(1 - \frac{1}{N}\right)^{n-1} = N.$$

This result shows that the expected time to coalescence is the value of the population size, which confirms the intuition that faster and slower coalescence occurs for smaller and larger population sizes respectively. This model is known as the Wright-Fisher model, and it is a discrete treatment of genetic coalescence. The coalescent model treats the same idea as the Wright-Fisher model, but in a continuous fashion.

2.2 The Coalescent Model

Although the coalescent model can be derived as a certain limit of the Wright-Fisher model, we will present it here in a more direct way.

The idea that the coalescence of lineages depends on the size of the population naturally leads to the definition of a coalescent unit. As a more useful unit of time, the coalescent time scale is defined by

$$u = \frac{t}{N},$$

where N is the population size, t is time measured in generations and u is time in coalescent units. For the sake of simplicity, we assume that the population size stays constant throughout time.

It is important to notice that this definition indicates that the effect on coalescence of having many generations is indistinguishable from the effect of having a small population size. In other words, having a large t will have the same effect on coalescence as having a small N, a feature we see in the Wright-Fisher model. This allows us to treat the lineages as coalescing at a constant rate.

To clarify this point, suppose that there are two distinct lineages at the present time $u = 0$, and let $h(u)$ be the probability that those two lineages are still distinct at time $u > 0$, where u is in coalescent units. Here, since we want small probabilities to have a small rate of change, then we must have that $\frac{d}{du}h(u) = -h(u)$. This makes sense because as we move backwards in time, the probability of having coalesced must increase, or, in other words, the probability of not having coalesced should decrease, which implies that $\frac{d}{du}h(u)$ must be negative. Moreover, since the two lineages are distinct at time zero, then $h(0) = 1$. A solution to this differential equation is given by $h(u) = e^{-u}$. Therefore, the probability that the lineages did not coalesce between time 0 and time u is $h(u) = e^{-u}$. This means that $P(u) = 1 - e^{-u}$ is the probability that the two lineages did coalesce between time 0 and u. Notice that $P(u)$ can be seen as a cumulative distribution function (CDF); therefore,

$P'(u) = e^{-u}$ is the density function for the time to coalescence. In other words, the time to coalescence (in coalescent units) of two distinct lineages is exponentially distributed with rate 1.

Now, we have that the expected time to coalescence is given by

$$\int_0^\infty uP'(u)du = \int_0^\infty ue^{-u}du = 1.$$

Notice that the expected time to coalescence does not depend on the population size. The definition of coalescent units leads to a model in which population size is no longer relevant.

Thus far, we have considered the coalescence of two lineages. However, analysis of phylogenies typically involves many taxa and many lineages. Therefore, consider the case where we have n different lineages at time $u = 0$. The first coalescent event may occur between any pair of lineages, which means that there are $\binom{n}{2} = \frac{n(n-1)}{2}$ possible lineages that could coalesce. Now, making the assumption that the coalescent events are independent and identically distributed, we have that the rate at which the first coalescent event occurs must be scaled by the number of possible pairs. In other words, if the probability that all of those n lineages are distinct at time u is $h(u)$, then

$$\frac{d}{du}h(u) = -\binom{n}{2}h(u)$$

which implies that

$$h(u) = e^{-\binom{n}{2}u}.$$

Furthermore, using integration by parts, we have that the expected time for n lineages to coalesce to $n-1$ is given by

$$\int_0^\infty \binom{n}{2}ue^{-\binom{n}{2}u}du = \frac{2}{n(n-1)}.$$

Notice that when we have only two lineages, then the expected time to coalescence is 1, as shown in previous calculations. However, if there are 8 lineages coalescing to 7 lineages, then the expected time would be $\frac{2}{8(7)} = \frac{1}{28}$. This is reasonable since the more lineages, the more options for a pair to coalesce, which results in the first coalescent event occurring sooner on average. Finally we can find the expected time for n lineages to coalesce to 1 by adding the expected time for each pair of lineages to coalesce:

$$\begin{aligned}\sum_{i=2}^{n} \frac{2}{i(i-1)} &= 2\sum_{i=2}^{n}\left(\frac{1}{i-1} - \frac{1}{i}\right) \\ &= 2\left(1 - \frac{1}{n}\right).\end{aligned}$$

Finally, notice that $\lim_{n\to\infty} 2\left(1-\frac{1}{n}\right) = 2$, which means that as the number of lineages grows to infinity, the expected time of n lineages to coalesce to 1 lineage is 2 coalescent units.

2.3 Multispecies Coalescent Model

The multispecies coalescent model is an extension of the coalescent model to species trees. In a species tree, each branch represents the population of a particular species; therefore, we represent the branches as pipes from which we draw samples of the population. Also, since we are using coalescent units to measure the branch lengths, then we are making the assumption that the population size has been taken into account, which means that the width of the species tree's pipes are all the same.

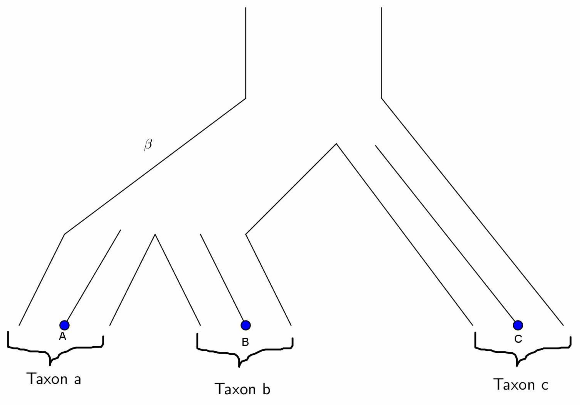

Figure 2.2: A 3-taxon species tree ((a,b),c) showing three sampled gene lineages A,B, and C

Figure 2.2 shows an example of a simple 3 taxon species tree where we sample one gene per taxon; namely genes A, B and C from taxon a, b and c respectively. Notice that these three lineages may coalesce to produce any of three possible topological gene trees: $((A,B),C),((A,C),B)$ or $((B,C),A)$[1].

Consider the gene tree $((B,C),A)$, which has a different topology than the species tree topology $((a,b),c)$. Such tree forms only if the lineages B and C coalesce before meeting A. In other words, lineages A and B must remain distinct as they enter the population of edge β, and once they reach the population ancestral to the root, B and C must coalesce first. Notice that once the lineages reach the ancestral root, the probability of B and C coalescing first is $\frac{1}{3}$ because at this point, each pair of taxa has equal probability of coalescing. Recall that the probability of two distinct lineages not coalescing on edge β is e^{-x}, where x is the length of edge β. This means that the probability of A and B not coalescing on edge β is e^{-x}. Therefore,

$$P((B,C),A) = P(((B,C),A)|\text{no coalescence on }\beta)P(\text{no coalescence on }\beta) = \frac{1}{3}e^{-x}.$$

Similarly

$$P((A,C),B) = \frac{1}{3}e^{-x},$$

and finally, since all probabilities must add to 1, we find

$$P((A,B),C) = 1 - \frac{2}{3}e^{-x}.$$

We can also calculate $P((A,B),C)$ by thinking of this probability as

$$\begin{aligned} &P(((A,B),C)|\text{no coalescence on }\beta)P(\text{no coalescence on }\beta)\\ &+ P(((A,B),C)|\text{coalescence on }\beta)P(\text{coalescence on }\beta)\\ &= \frac{1}{3}e^{-x} + 1\cdot(1-e^{-x}) = 1 - \frac{2}{3}e^{-x}. \end{aligned}$$

As the number of taxa on a tree is increased, the number of edges on which coalescent events occur will also increase, and the number of cases we need to consider for the formation of gene trees grows considerably. Therefore, the calculation shown previously becomes much more complicated.

[1]((A,B),C) is an example of the *Newick* representation of a tree topology, where in this case A and B are joined together first, then followed by C

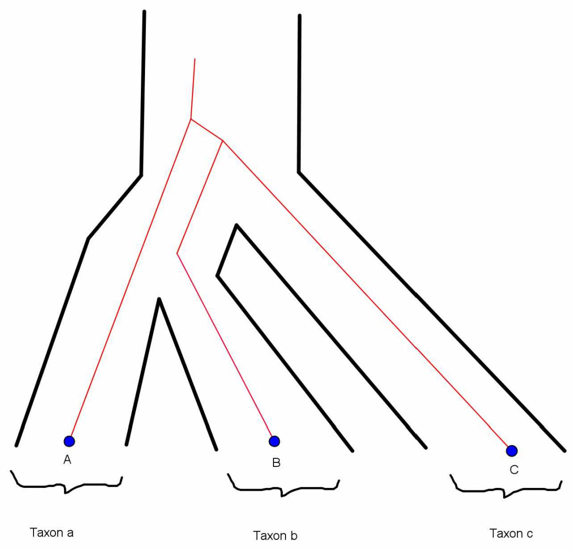

Figure 2.3: Species tree with a near polytomy at the root and a sampled gene tree (red tree) having a different topology than that of the species tree.

Notice that if x is infinitely large, then $P((B,C),A) \approx 0, P((A,C),B) \approx 0, P((A,B),C) = 1$, which means that for a long enough branch, the gene tree topology will most likely match the species tree topology. However, when x is very small, the species tree in Figure 2.2 will have a near polytomy at the root, and in this case each gene tree topology has a probability of approximately $\frac{1}{3}$ of occurring. A topological gene tree not matching the topological species tree is often called *incomplete lineage sorting*. An example of this phenomenon is shown in Figure 2.3.

Now, the coalescent model can be used to infer the species tree from a sample of gene trees. If the branches of the species tree are long enough (no polytomy), then the topology of most gene trees will match the topology of the species tree; in other words, incomplete lineage sorting is unlikely to occur. However, we are particularly interested in exploring how to resolve the case where the species tree has a near polytomy. We will simulate gene trees according to the coalescent model from a species tree that has very short internal branches. This will result in a substantial number of our simulated gene trees not having the same topology as the species tree. Our goal is to explore how different treatments of these gene trees affect the estimation of the species tree topology.

Chapter 3

Inferring Species Trees From Gene Trees Using STAR and USTAR

Since our study analyses the estimation of a species tree topology from a set of gene trees coming from the coalescent model, we must investigate the construction of such a topological estimate. Thus, given a set of gene trees relating different taxa, how can we infer the species tree for these taxa? Maximum Likelihood and Bayesian methods are common statistical frameworks that can be used to infer species trees, but both are computationally intense and limited in the number of taxa they can handle. Therefore, we will introduce STAR [LYPE09] and USTAR, which are algorithms that take collections of gene trees to estimate species trees. These algorithms are fast and they have been proven to be statistically consistent under the multispecies coalescent model [ADR13].

3.1 The STAR Method

STAR begins by assigning a rank to the nodes, or vertices, of the gene trees. This assignment proceeds by first assigning to the roots some value n, which is usually taken to be the number of taxa. Then we assign the value $n - 1$ to the internal nodes that are children of the root. Their children are assigned $n-2$, and so on for all other internal nodes. Consider the example in Figure 3.1.

Notice that the root has a value of 6 for the trees in Figure 3.1 because there are 6 taxa in total. For the tree to the left on Figure 3.1, we proceed by assigning a value of 5 to node a_4 and continue until we reach a_2, which has a value of 2 because the node above a_2 has a value of 3.

Once the nodes have been numbered, we use them to define a metric on the taxa, where the distance between any two leaves (pendant edges) is twice the number of the node that is

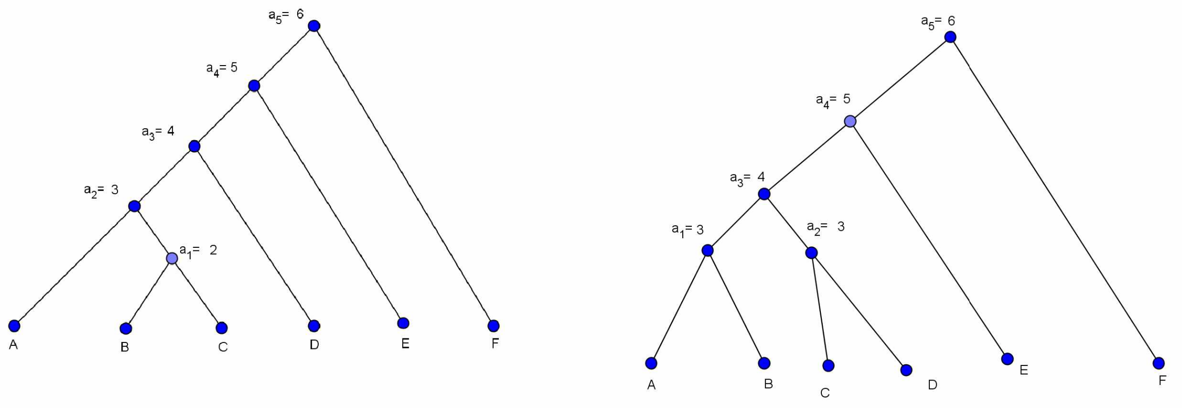

Figure 3.1: Two gene trees on 6 taxa with the STAR node numbering. The root is numbered with 6, and its descendants with numbers decreasing by 1.

their most recent common ancestor. For example, for both trees of Figure 3.1, the distance between lineage A and lineage B is 6, but the distances from B to C is 4 for the tree on the left and 8 for the tree on the right. Since the distance from every leaf to the root is simply the value of the root's label, this scheme produces an *ultrametric* tree, which is defined as a tree with all leaves equidistant from the root.

The second step of STAR is to record these intertaxon distances in a matrix. In the case of the trees in 3.1, with taxa ordered alphabetically, we have the matrices D_{T_L} and D_{T_R} for the tree on the left and the right respectively. Keep in mind that the entries of these matrices are labeled; in other words, the first row and column correspond to A, second B and so on. This allows for interpreting the entry on the first row and third column to be the distance between lineage A and lineage C, or the entry on the fourth row and second column to be the distance between D and B. Hence, these matrices encode the rooted tree topologies.

$$D_{T_L} = \begin{bmatrix} 0 & 6 & 6 & 8 & 10 & 12 \\ 6 & 0 & 4 & 8 & 10 & 12 \\ 6 & 4 & 0 & 8 & 10 & 12 \\ 8 & 8 & 8 & 0 & 10 & 12 \\ 10 & 10 & 10 & 10 & 0 & 12 \\ 12 & 12 & 12 & 12 & 12 & 0 \end{bmatrix} \text{ and } D_{T_R} = \begin{bmatrix} 0 & 6 & 8 & 8 & 10 & 12 \\ 6 & 0 & 8 & 8 & 10 & 12 \\ 8 & 8 & 0 & 6 & 10 & 12 \\ 8 & 8 & 6 & 0 & 10 & 12 \\ 10 & 10 & 10 & 10 & 0 & 12 \\ 12 & 12 & 12 & 12 & 12 & 0 \end{bmatrix}$$

The third step is to build a new matrix by averaging the distance matrices for all gene trees. With a large number of gene trees, produced in accord with the multispecies coalescent model, this new matrix approximates the distances on some ultrametric version of the species tree [ADR13]. Therefore, the new matrix can be used to find the species tree topology, though the distances themselves are not those on the original species tree. The two main algorithms that are used to build the species tree topology from a distance matrix are UPGMA and Neighbor Joining. Using its ultrametric properties, UPGMA will estimate the rooted species tree, while Neighbor Joining will estimate the unrooted species tree. UPGMA and Neighbor Joining will be discussed in Section 3.3 and Section 3.4 respectively.

3.2 USTAR

USTAR is a method similar to STAR, except that it considers unrooted gene trees instead. As in STAR, we first have to metrize the gene trees in order to build a distance matrix characterizing each tree. However, instead of numbering the nodes, USTAR assigns the length 1 to all edges of the tree, and the distance between any two taxa is simply the number of edges in the path connecting the two taxa. In other words, the distance matrix is built by considering the *graph-theoretic* distance between the taxa [ADR16]. Consider Figure 3.2.

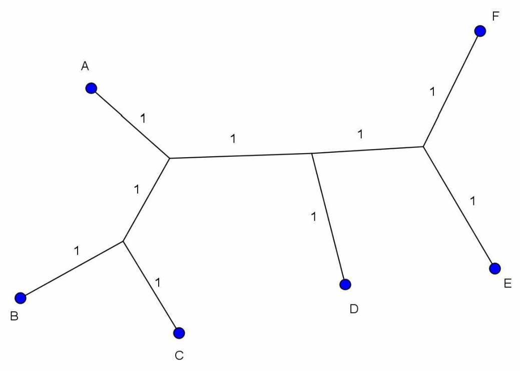

Figure 3.2: A 6-taxon unrooted tree with all edges of length 1.

With taxa ordered alphabetically, the tree T on Figure 3.2 will produce the following matrix:

$$D_T = \begin{bmatrix} 0 & 3 & 3 & 3 & 4 & 4 \\ 3 & 0 & 2 & 4 & 5 & 5 \\ 3 & 2 & 0 & 4 & 5 & 5 \\ 3 & 5 & 5 & 3 & 0 & 2 \\ 4 & 5 & 5 & 3 & 0 & 2 \\ 4 & 5 & 5 & 3 & 2 & 0 \end{bmatrix}$$

We obtain a distance matrix like D_T for all unrooted gene trees, and we average them as done in STAR. Finally, because this average must fit a species tree, but not necessaries ultrametrically, we use Neighbor Joining to estimate the species tree. Consequently, this results in an unrooted species tree topology estimate.

3.3 Unweighted Paired Group Method with Arithmetic Means (UPGMA)

The Unweighted Paired Group Method with Arithmetic Means is a method of constructing an ultrametric tree from intertaxon distances. The best way to understand how UPGMA works is by working through an example. Consider the ultrametric tree of Figure 3.3 with corresponding distance matrix D.

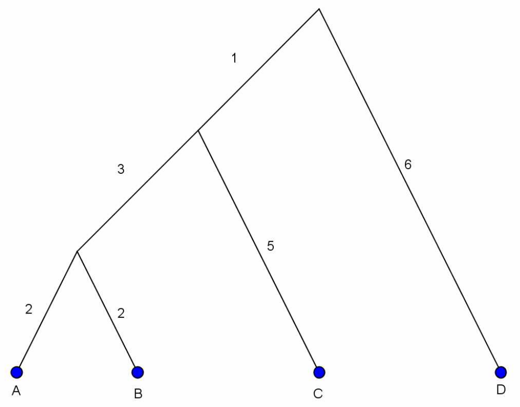

Figure 3.3: Tree with 4 taxa rooted at taxon "D". This tree is the model tree for a simple UPGMA example.

$$D = \begin{bmatrix} 0 & 4 & 10 & 12 \\ 4 & 0 & 10 & 12 \\ 10 & 10 & 0 & 12 \\ 12 & 12 & 12 & 0 \end{bmatrix}$$

The first step of UPGMA is to look for the smallest nonzero distance in the matrix and group the corresponding taxa together. Since UPGMA maintains ultrametricity, we divide such distance by two in order to obtain the distance to the most recent common ancestor. For example, notice that in matrix D the smallest distance is 4, which is the distance from A to B. Hence, the distance from A and B to their most recent common ancestor is 2, as shown in Figure 3.4 :

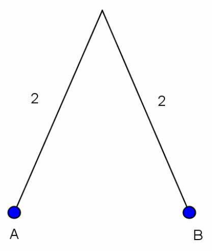

Figure 3.4: First step of UPGMA is to group the taxa corresponding to the smallest entry in the distance matrix.

Next, we collapse taxa A and B into a group, which we name AB and build another matrix by averaging the distance from A and B to all other taxa. For example, the distance from AB to C is given by

$$d(AB, C) = \frac{d(A, C) + d(B, C)}{2} = \frac{10 + 10}{2} = \frac{20}{2} = 10.$$

This produces a new distance matrix with AB as a new "taxon". With ordering AB, C, D we have

$$D' = \begin{bmatrix} 0 & 10 & 12 \\ 10 & 0 & 12 \\ 12 & 10 & 0 \end{bmatrix}.$$

D' now has a new shortest distance, namely $d(AB, C) = 10$. Again, we divide this value in half to obtain the most recent common ancestor and graft in the tree joining A and B. This leads to the tree in Figure 3.5:

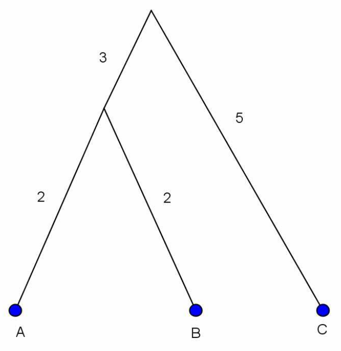

Figure 3.5: Tree where taxa A,B,C have been collapsed and their distance to their most common recent ancestor is 5

We then average the distances by referring to the initial D matrix. Hence we have that

$$d(ABC, D) = \frac{d(A, D) + d(B, D) + d(C, D)}{3} = \frac{15 + 16 + 14}{3} = 15$$

and this produces the final matrix

$$D'' = \begin{bmatrix} 0 & 15 \\ 15 & 0 \end{bmatrix}$$

which leads to an estimate of the initial tree, as shown in Figure 3.6

Notice that the estimated tree in Figure 3.6 matches exactly the actual tree shown in Figure 3.3. This is because UPGMA assumes that the tree to be estimated is ultrametric. However, if the distance matrix used in the UPGMA calculation does not exactly fit an ultrametric tree, UPGMA may still predict the right topology, but edge lengths of the estimated tree will not match the original tree. Furthermore, if the distance matrix is far from fitting an ultrametric tree, then even the topology estimation of UPGMA may not be accurate.

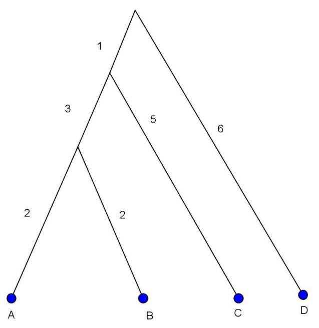

Figure 3.6: Final tree produced by UPGMA

3.4 Neighbor Joining

Now, consider an unrooted version of the tree in Figure 3.3 and let's modify its branch lengths so that we obtain the tree shown in Figure 3.7

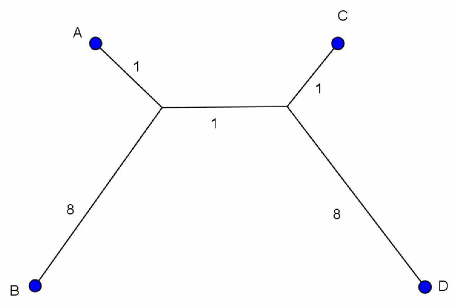

Figure 3.7: 4-taxon tree for which UPGMA fails to recover the correct topology.

This tree leads to the following distance matrix:

$$\begin{bmatrix} 0 & 9 & 3 & 10 \\ 9 & 0 & 10 & 17 \\ 3 & 10 & 0 & 9 \\ 10 & 17 & 9 & 0 \end{bmatrix}.$$

From the distance matrix, we have that $d(A, C) = 3$ and $d(A, B) = 9$, which means that $d(A, C) < d(A, B)$. Therefore, UPGMA would group lineages A and C first. However, this would result in the wrong topological tree because A and C are not joined in Figure 3.7.

Neighbor Joining is an algorithm that attempts to fix this problem. To understand the main idea behind Neighbor Joining, consider again the tree in Figure 3.7. Note that $d(A, B) = 9$ and $d(C, D) = 9$, while $d(A, C) = 3$ and $d(B, D) = 17$. Since $9 + 9 < 3 + 17$, then we have that $d(A, B) + d(C, D) < d(A, C) + d(B, D)$. This inequality actually holds for any value assigned to the branch lengths because this inequality is based on a topological feature rather than a metric one. In more detail, for any generic 4-taxon tree with topology as shown in Figure 3.7, the quantity $d(A, B) + d(C, D)$ only counts the edge lengths of the four edges on which taxa A, B, C and D lie. On the other hand, the quantity $d(A, C) + d(B, D)$ counts the same edges as described previously, plus twice the central edge connecting the four taxa. By the same line of reasoning, we have that $d(A, C) + d(B, D) = d(A, D) + d(B, C)$. From this analysis, we conclude that for any 4-taxon tree with positive edge lengths and with topology as in Figure 3.7, the following inequalities/equalities hold:

$$d(A, B) + d(C, D) < d(A, C) + d(B, D) = d(A, D) + d(B, C).$$

This is known as the *4-point condition*. By computing the three terms in the inequalities above and determining the smallest value among them, we can determine the topology of a 4-taxon tree. Using an "average" version of the 4-taxon condition coupled with similar steps as in UPGMA, Neighbor Joining allows us to construct trees from a distance matrix that may not fit an ultrametric tree. The detailed steps of the algorithm are omitted in this paper, but Neighbor Joining runs computationally as fast as UPGMA. For further details on Neighbor Joining, refer to [SK88].

As this paper involves the estimation of species tree topologies by STAR and USTAR, we will perform both UPGMA and Neighbor Joining. Since the average distance matrix for STAR is approximately ultrametric, either UPGMA or Neighbor Joining can lead to statistically consistent inference. And, since the average distance matrix for USTAR is generally not ultrametric, then only Neighbor Joining can be used with USTAR to obtain statistical consistency.

Chapter 4

Metrics On Tree Space

In our study of the performance of variants of STAR and USTAR methods, we must measure the error in the inferred species trees, which means that we must choose a metric to determine how far away phylogenetic trees are from each other. One of the most intuitive and easily calculated comparisons is known as the Robinson-Foulds distance.

4.1 Robinson-Foulds Split Distance

By a *split* on a tree, we refer to the bipartition of taxa that results when deleting an edge. More generally, if we let X be the set of taxa, then a *split* of X is defined to be any partition of X into two non-empty disjoint subsets. We write $X_0|X_1$ to denote the split with subsets X_0, X_1. For example, consider the tree on the left side of Figure 4.1. This tree contains three internal edges, namely α, β and ϵ. If we remove the edge α, then there are two sets of taxa that split apart, which are $\{A, B\}$ and $\{C, D, E, F\}$. Therefore, the split induced by α is $\{A, B\}|\{C, D, E, F\}$.

Robinson-Foulds split distance, or RFsplit distance, between trees simply counts the number of splits that are different in each tree. It is important to mention that this paper only considers the deletion of internal edges. This is because the deletion of terminal edges results in trivial splits, which appear on all trees and thus do not provide useful information for determining the difference between trees.

For example, consider again Figure 4.1.

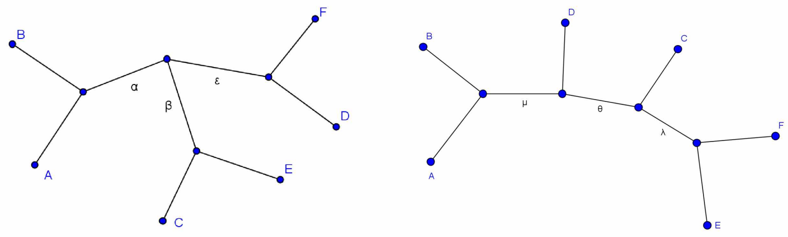

Figure 4.1: Two 6-taxon unrooted trees with RFsplit distance of 4.

For the tree on the left hand side of Figure 4.1, we have the following non-trivial splits:

$$\alpha : \{A, B\}|\{C, E, D, F\}$$
$$\beta : \{C, E\}|\{A, B, D, F\}$$
$$\epsilon : \{D, F\}|\{A, B, C, E\}$$

while for the tree on the right we have:

$$\mu : \{A, B\}|\{C, E, D, F\}$$
$$\theta : \{A, B, D\}|\{C, E, F\}$$
$$\lambda : \{E, F\}|\{A, B, C, D\}$$

Now notice that the only non-trivial split that these trees have in common is $\{A, B\}|\{C, E, D, F\}$. This means that there are four different splits that belong to either tree but not both; hence, the RFsplit distance between the trees in Figure 4.1 is 4.

4.2 Robinson-Foulds Clade Distance

The analysis of splits for the RFsplit distance must be modified when considering rooted trees. Instead of counting splits, we count clades, where a clade on a rooted tree is all the taxa descended from a specific node. We call this the RFclade distance.

There are two types of trivial clades: one element clades, which are induced by considering each taxon as a vertex, and the clade containing all taxa, which is induced by considering the root of the tree. Again, we need not consider trivial clades when determining the dis-

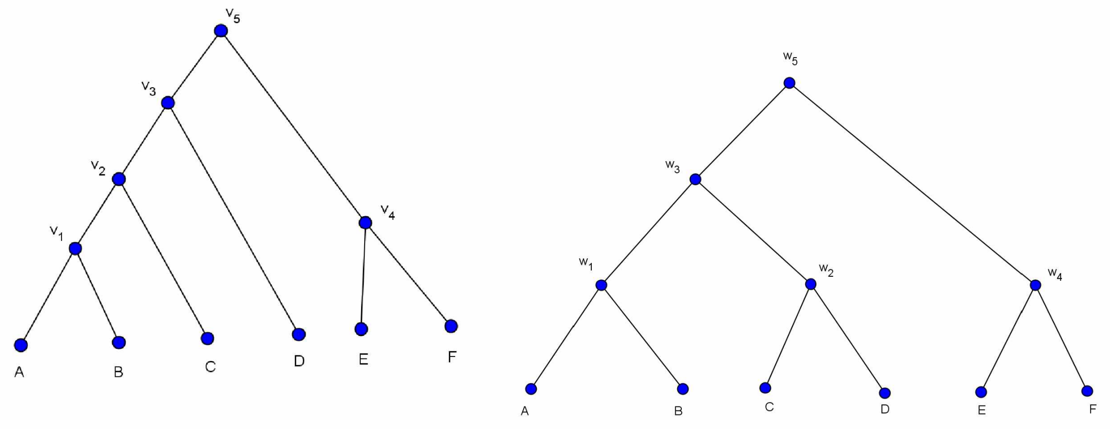

Figure 4.2: Two 6-taxon rooted trees with RFclade distance of 4

tance between trees because these clades do not provide useful information on the difference between rooted trees.

For example, consider the trees in Figure 4.2. For the tree on the left, under vertices v_1, v_2, v_3 and v_4, we have the clades

$$\{A, B\}$$
$$\{A, B, C\}$$
$$\{A, B, C, D\}$$
$$\{E, F\}$$

respectively. On the other hand, for the tree on the right, we have that under vertices w_1, w_2, w_3 and w_4, the clades are

$$\{A, B\}$$
$$\{C, D\}$$
$$\{A, B, C, D\}$$
$$\{E, F\}$$

respectively. From these clades, we see that both trees share the clades $\{A, B, C, D\}$ and

$\{E, F\}$, which means that there are four clades that are different between them. Therefore, the RFclade distance between these two rooted trees is 4.

4.3 Clade and Split Distances When an Outgroup is Present

To understand the construction of our tree simulations, we must introduce the concept of an outgroup. An *outgroup* is a taxon known to be genetically distant enough to all other taxa on the tree that we can use it identify a root. For example, suppose we are interested in finding a phylogenetic tree relating four species of frogs, say f_1, f_2, f_3 and f_4. After analyzing the data, we hypothetically obtain the tree shown in Figure 4.3.

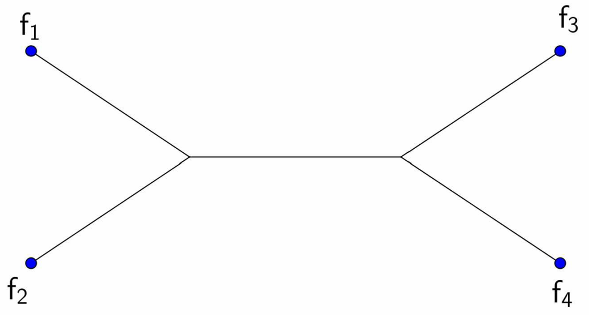

Figure 4.3: Unrooted topological tree relating four species of frogs, f_1, f_2, f_3 and f_4.

Now, we are interested in trying to locate a root for this tree. A common technique is to choose a species known to be distant from all these frogs, say a chameleon, which we will call c_1. A priori, we must know that as species, the frogs are genetically closer to each other than to the chameleon. Hence, we expect c_1 to coalesce on an edge that leads to the most recent ancestral frog, thus providing a place for the root, as shown in Figure 4.4. The resulting rooted tree is shown in Figure 4.5

The simulations conducted in our analysis are built to have an outgroup which we call "O". However, when this outgroup is present, the RFclade distance and the RFsplit distance become indistinguishable, or equal in value. Consider the trees in Figure 4.6.

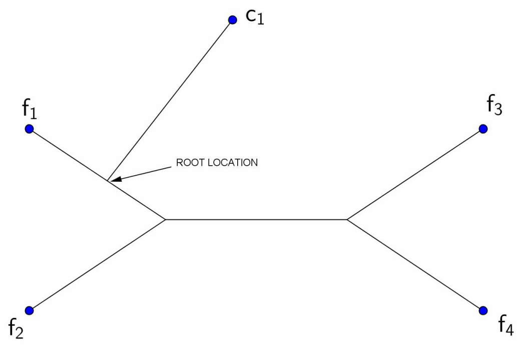

Figure 4.4: Location of a root using an outgroup.

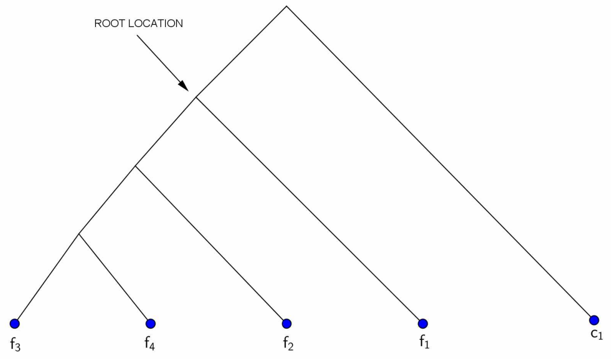

Figure 4.5: Resulting tree after rooting at the outgroup.

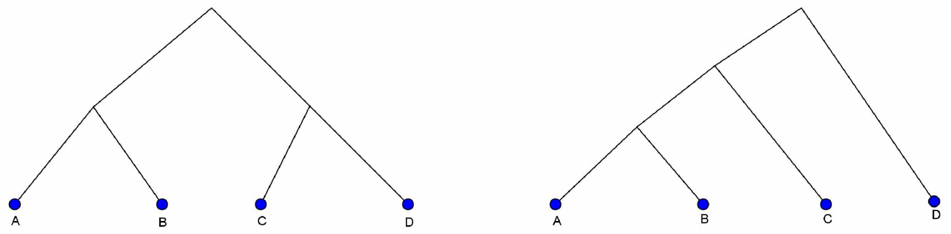

Figure 4.6: Trees where the clade distance is 2 and the split distance is 0.

Note that the clades for the tree on the left are $\{A, B\}$,$\{C, D\}$, while the clades for the tree on the right are $\{A, B\}$, $\{A, B, C\}$. Thus, the RFclade distance between these two trees is 2. On the other hand, as unrooted trees, both trees have the splits $\{A, B\}|\{C, D\}$, which means the RFsplit distance is 0. Now, suppose that we add an outgroup "O", producing the picture in Figure 4.7.

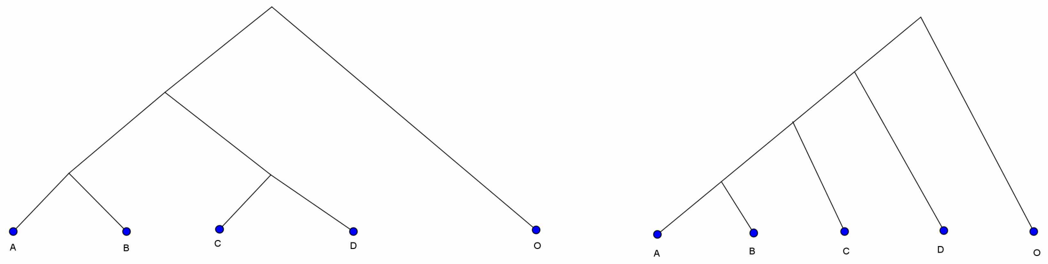

Figure 4.7: Trees Rooted at "O"

The splits for the tree on the left of Figure 4.7 are

$$\{A, B\}|\{C, D, O\}$$
$$\{C, D\}|\{A, B, O\}$$

while the tree on right has splits

$$\{A, B\}|\{C, D, O\}$$
$$\{A, B, C\}|\{D, O\}.$$

This means that the RFsplit distance is 2. However, we have clades $\{A, B\}, \{C, D\}$ and $\{A, B\}, \{A, B, C\}$ respectively, which results in a RFclade distance of 2. This is problematic because although we added an outgroup, the underlying topologies are different as rooted trees, yet the RFclade distance is not capturing this feature. The reason this occurs is because adding an outgroup introduces a new clade, but such clade is shared by both trees because it is the clade that includes all the taxa except the outgroup. On the other hand, introducing an outgroup also introduces a new internal edge, which means we have a new split. This split is different depending on the underlying topologies, which means our RFsplit distance would increase.

Therefore, our simulation analysis must take this observation into account. The way we incorporate the RFclade distance into our analysis is by first removing the outgroup from both the actual species tree topology and the estimated species tree topology. The result is a rooted topology among all taxa except the outgroup. In other words, instead of considering Figure 4.7, we consider trees as shown in Figure 4.6. The RFclade distance on these outgroup-free topologies will result in the more useful clade difference.

Chapter 5

Methods of Simulation Study

This study focuses on analyzing different variants of species trees inferred by STAR and USTAR. By variants, we mean different ways in which we can root the gene trees simulated from a species tree according to the coalescent model, and different ways in which we can root the resulting inferred species trees. We perform such gene tree simulations from species trees that have very short branch lengths, which induce errors as a result of incomplete lineage sorting. Our final goal is to determine which of these variants results in the best inference of the initial species tree.

5.1 Types of Gene Trees Simulated And Species Tree Estimations

The simulations in our study were conducted using the programming language R. In particular, we use the statistical packages "ape" [ape15], "phangorn" [pha16], and "phybase" [phy14].

In our study, we simulated gene trees according to the coalescent model. We did this by feeding a particular metric species tree, which we call the *true species tree*, into the phybase function "sim.coal.sptree" which then outputs simulated rooted gene trees. This species tree is designed to have taxon "O" as the outgroup. Once we obtain the gene trees from the coalescent model, we unroot them using the ape function "unroot." We do this because even if there is an outgroup present, the inference of gene trees from sequence data results in unrooted trees. Since incomplete lineage sorting may occur, especially if the species tree branch lengths are short, then it is possible that rooting by the outgroup would be erroneous. Therefore, our methods vary in the rooting by the outgroup "O". In other words, since we are interested in exploring the effects on species tree estimation produced by rooting gene trees, we consider different variants of the gene trees that deal with how the root is treated. The following are the different variants and the names we have assigned to the corresponding species tree estimation:

A-rooted: Species trees built by rooting the gene trees at the outgroup “O”. We then apply STAR with Neighbor Joining, which produces an unrooted species tree. Finally, we root the resulting species tree by “O”.

A-unrooted: Species trees built by rooting the gene trees at the outgroup “O” and applying STAR with Neighbor Joining, which produces an unrooted species tree (Notice this is the same as A-rooted, except we do not root the estimated species tree at the outgroup).

B-rooted: Species trees built by applying USTAR with Neighbor Joining to the unrooted gene trees, which produces an unrooted species tree. We then root the resulting species tree at the outgroup “O”.

B-unrooted: Species trees built by applying USTAR with Neighbor Joining, which produces an unrooted species tree (Notice this is the same as B-rooted, except we do not root the species tree at the outgroup).

C-rooted: Species trees built by rooting the gene trees by the outgroup “O” and applying STAR with UPGMA, which produces a rooted species tree.

C-unrooted: Species trees built by simply unrooting the species tree produced from *C-rooted*.

D-unrooted: Species trees built by rooting the gene trees by “O”, and then “label rooting” them with taxon “ROOT”. This label root procedure simply involves attaching an extra taxon called “ROOT” to the root of the rooted gene trees, which makes the gene trees have one extra taxon, but we can technically view them as unrooted trees. Finally, we apply USTAR with Neighbor Joining to produce an unrooted species tree, and remove the taxon “ROOT” from this inferred species tree.

D-rooted: Species trees built by taking the species trees produced from *D-unrooted* and rooting them at “ROOT” before removing such taxon.

5.2 Choosing Topologies

Analyzing phylogenetic trees can be a daunting task because of the many possible topologies that can arise with relatively few taxa. In fact, for $n \geq 3$ taxa, there are $(2n-5)!! =$

$1 \cdot 3 \cdot 5 \cdots (2n-5)$ different topologies. For a simulation study, we hope to choose a small number of trees to analyze such that the results capture the overall behavior for any topology. Also, we must clarify how the simulations in our study are constructed and how they will be displayed in our results. In this paper, we will arbitrarily choose trees with 8 taxa plus an additional outgroup. This means we should consider 5198 different tree topologies. However, we want to capture the most important features of all these topologies by analyzing a small subset of them. We will consider three different topologies as shown in Figures 5.1, 5.2, 5.3: a caterpillar tree, a balanced tree and a mixed tree that is perhaps more typical of a "random" topology.

We choose the caterpillar and the balanced tree because they represent the most extreme features, and we choose the mixed tree because it lies somewhere in between. A *measure of balance* quantifies this "extremeness" by taking the absolute value of the difference between the number of taxa to the left and to the right of each node, and then adding the result over all nodes. For example, consider the tree topologies of Figures 5.1, 5.2 and 5.3.

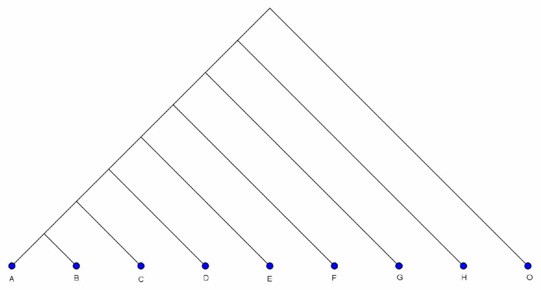

Figure 5.1: Caterpillar

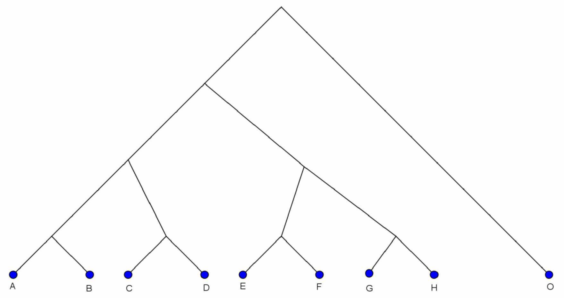

Figure 5.2: Balanced

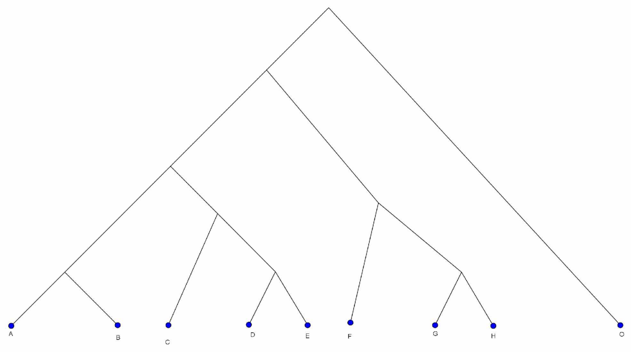

Figure 5.3: Mixed

Since "O" is the outgroup and we are mainly interested in the underlying topology, we ignore "O" in our measure of balance. For the balanced tree, each node has equal number of taxa to the left and to the right, which means that the measure of balance is 0. For the caterpillar, beginning at the node corresponding to the most recent common ancestor of A and H, we have

$$|1-7|+|1-6|+|1-5|+|1-4|+|1-3|+|1-2|+|1-1|=21,$$

and for the mixed tree we have

$$|3-5|+|2-1|+|1-1|+|3-2|+|2-1|+|1-1|+|1-1|=5.$$

Notice that the smallest measure is 0, for the balanced tree, the biggest is 21, for the caterpillar, and it is easy to see that there is no tree with more extreme measure of balance than these.

5.3 The True Species Tree Branch Lengths

Now, consider a species tree with a caterpillar topology, as shown in Figure 5.1. Notice that this tree has 7 internal edges, which means that there are 7 places where lineages can coalesce to form the caterpillar tree. However, if these edges are very small, then it is very likely that incomplete lineage sorting will occur. As indicated before, we are interested in answering

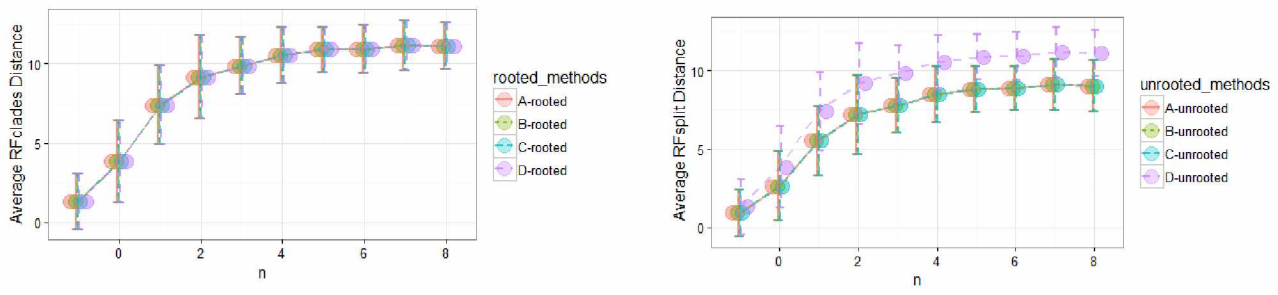

Figure 5.4: Sample plot showing the average RFclade distance and average RFsplit distance for rooted and unrooted methods respectively.

the following question: if we collect a set of gene trees simulated from a true species tree with small internal edges, which variant of STAR or USTAR best estimates the topology of the true species tree?

We will perform simulations with internal branch lengths of equal size, and ranging from 2^{-8} up to 2^{1} coalescent units by factors of 2. We choose these branch lengths because for all the 8 methods above, if the internal branches of the true species tree are long enough (e.g., if they are as long as 4 coalescent units), then incomplete lineage sorting is very unlikely to occur, which means that almost all gene trees will match the topology of the species tree. In turn, STAR and USTAR will estimate a species tree with the right topology. As the branch lengths decrease towards 0, however, the probability of incomplete lineage sorting increases, and it should become increasingly difficult to infer the correct species tree.

5.4 Plot Structure

Since our goal is to compare the estimated species trees we obtain from the 8 methods above to the true species tree, we must use RFsplit and RFclade distances to quantify their difference. We present our results through plots that show these distances, as displayed in Figure 5.4.

Each plot is constructed as follows: first, we choose a specific true species tree topology, which can be either a caterpillar, balanced, or mixed topology. Second, we specify the number of gene trees we wish to simulate, which we will refer to as *sample size*; these are the gene trees that will be used in the 8 methods discussed previously to infer a species tree. Third, we specify the number of *repetitions*, which is the number of times we wish to repeat the first and second steps. In other words, the repetitions are the number of times we wish to simulate gene trees, vary their rooting, and use them to estimate the true species tree. Our code calculates the RFsplit or RFclade distance between the inferred and the true species tree for each repetition, and then averages these distances over all repetitions. Therefore, for each indicated sample size and number of repetitions, we obtain two plots, one showing the average RFclade distance and the other showing the RFsplit distance, for the rooted and unrooted methods A,B,C, and D respectively.

Now, for each individual plot, we consider different true species tree internal branch lengths that vary from 2^{-8} to 2^{1} coalescent units. In order to make the interpretations of the plots more reasonable, we label the horizontal axis, from left to right, with the integers $n \in \{-1, \ldots, 8\}$ that correspond to branch lengths 2^{-n}. In other words, the tick marks, from left to right, on the horizontal axis correspond to branch lengths of $2^1, 2^2, \ldots, 2^{-8}$ coalescent units. The vertical axis corresponds to the average RFsplit or RFclade distance, over all repetitions, between the true species tree and the STAR/USTAR estimated species tree.

Another important aspect of the plots is the variance. Specifying the number of repetitions allows us to determine how many data points we wish to obtain for the average RFclade and RFsplit distance corresponding to each size of internal branch lengths. Using the function "var" from R, we calculate the variance of the RFclade and RFsplit distance for each size of internal branch lengths, and then we take the square root to obtain the standard deviation. Therefore, the vertical lines on each point of the plot correspond to the average RFclade or RFdistance plus or minus its standard deviation.

Finally, in order to make the plots cleaner, we horizontally shifted the points on the graph corresponding to the methods A,B,C and D. Therefore, if four dots have the same vertical value, but appear to have different horizontal values, then that means all those four dots are on top of each other on the same point.

5.5 Sample Size and Repetitions

As mentioned previously, if the internal branch lengths of the true species tree are long enough, we expect most of the simulated gene trees to have the same topology as the true species tree, which means that STAR/USTAR will estimate such topology as well. Therefore, if we consider the case where all internal edges are very small, then we would expect lineage sorting to occur frequently, which means that STAR/USTAR will not necessarily estimate the right true species tree topology. However, even with small internal edges, if we simulate a large number of gene trees, then STAR/USTAR will estimate the correct species tree topology. The evidence for this claim is shown in Figure 5.5 , where we use the 8 taxon caterpillar tree with 1 repetition in each trial, and we show the results for sample sizes of 5, 50, 500 and 5000. Figure 5.5 has two columns of plots: the left column corresponds to plots showing the average RFclade distance, while the right column corresponds to plots showing the average RFsplit distance. Also, the rows correspond to the sample sizes; in this case, the first row corresponds to sample size 5 and the last row to sample size 5000. Notice that each time, the curve flattens to the right, indicating that the STAR/USTAR estimate for very short internal branches becomes more accurate as we increase the number of gene trees.

Since we know that for a large number of gene trees, STAR predicts the right topology, we will investigate the effects of a small number of gene trees on species tree estimation. In this paper, we will explore sample sizes of 1, 3, 6 and 10 gene trees. Now, since we will be varying the number of gene trees, we must choose a particular number of repetitions. Consider the plots on Figure 5.6. Notice that the plots using 50, 70 and 1000 repetitions do not really have a significant difference in their shape; they all seem to flatten at about $n = 4$. Therefore, we will set the number of repetitions to 50 for all of our plots.

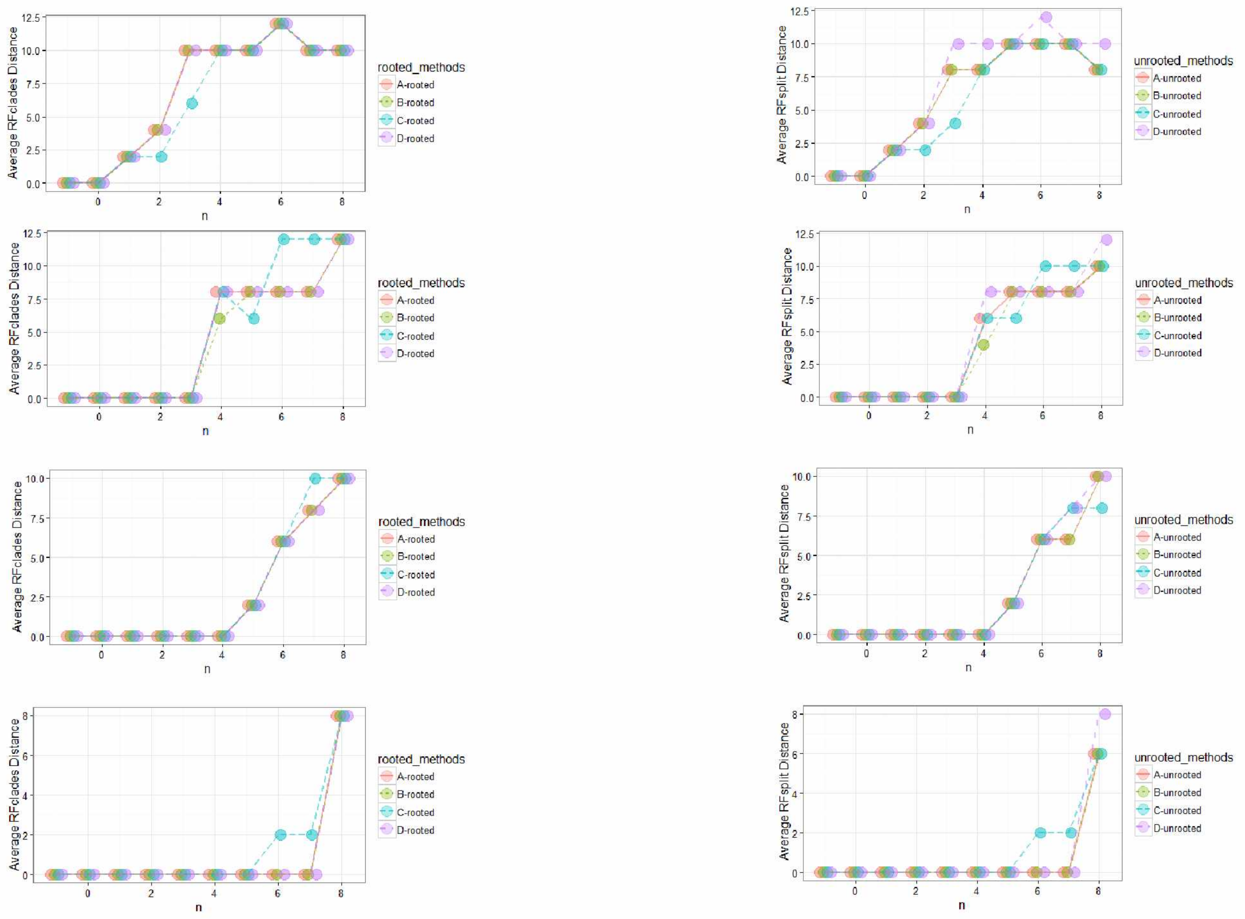

Figure 5.5: Species tree inference error plot showing increasing accuracy of species tree estimation as the sample size (number of gene trees simulated) changes by 5, 50, 500, and 5000 and the number of repetitions is chosen to be 1. The plots are organized into 4 rows and 2 columns; the four rows correspond to the sample sizes, the first row being the sample size of 5 and the fourth being the sample size of 5000. The left and right columns correspond to the rooted and unrooted versions respectively.

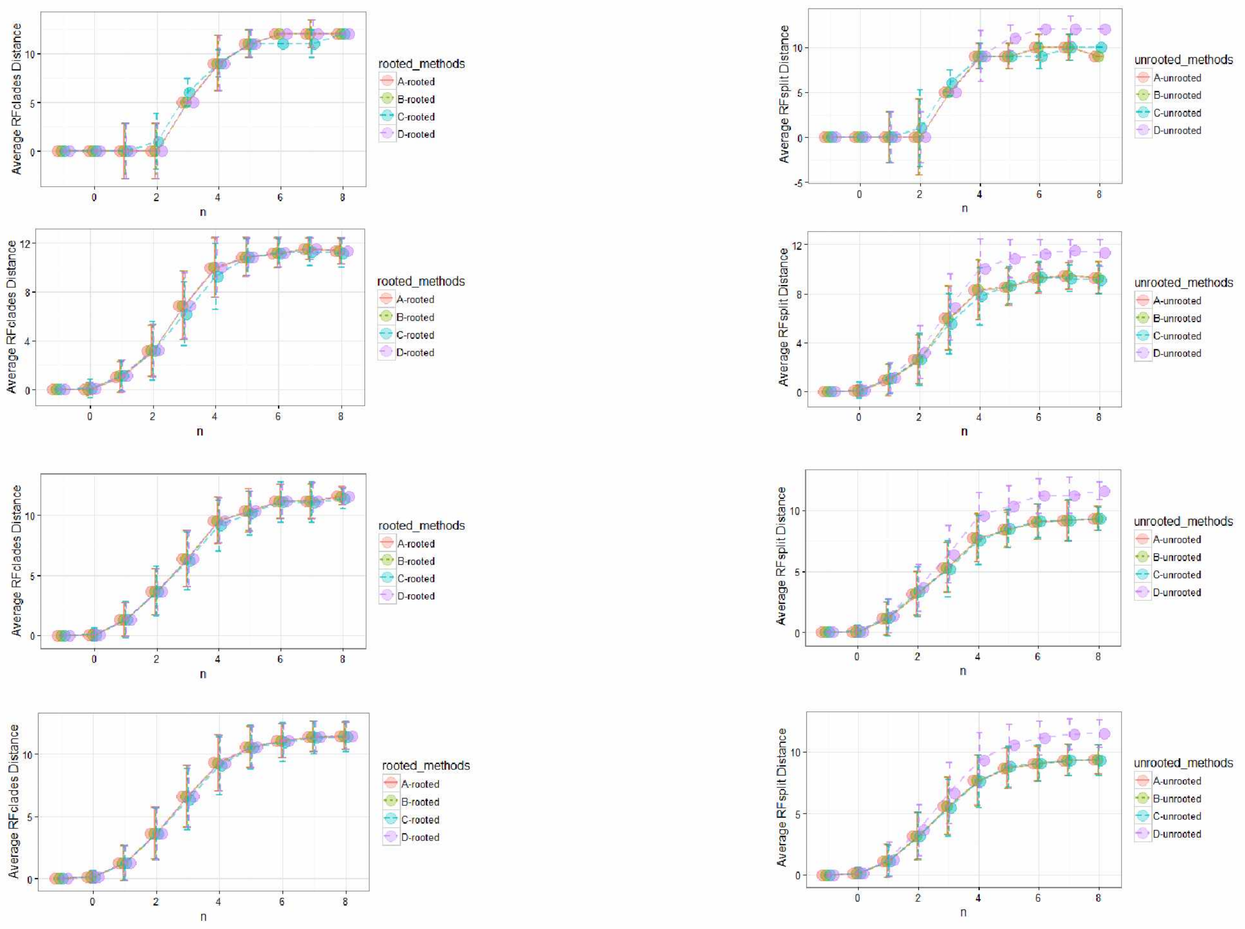

Figure 5.6: Species tree inference error plot showing the effect of increasing repetitions on species tree estimation as the number of repetitions changes by 2, 50, 70 and 1000. All of the plots are constructed with a sample size of 10 and a caterpillar species tree. The plots are organized into 4 rows and 2 columns; the four rows correspond to the sample sizes, the first row being the sample size of 2 and the fourth being the sample size of 1000. The left and right columns correspond to the rooted and unrooted versions respectively.

Chapter 6

Analysis of Simulations

Now that we understand the general framework of the plots, we will make simulations to explore the effects of our proposed methods on the estimation of the true species tree topology.

6.1 Simulations and Observations

Figures 6.1, 6.2 and 6.3 show the results of a simulation for the caterpillar, balanced and mixed trees respectively. It is clear that, with the exception of D-unrooted, there are not large differences in the performance between methods. However, analysis of their minor differences can provide useful insights about the behavior of these methods.

Consider the case for the caterpillar tree as shown in Figure 6.1, and focus first on the rooted methods. We can see that if we only have 1 gene tree, all variants perform the same. However, as we increase the sample size, we see that C-rooted performs slightly better for mid-size species tree branch lengths. In fact, it is clear to see the out-performance of C-rooted with a sample size of 10, where there is a distinct line connecting values of C-rooted from $n = 2$ to $n = 5$. Now, for the unrooted methods, we see that C-unrooted does better on average than all the other methods. However, D-unrooted performs significantly worse, regardless of the sample size. Finally, for both, rooted and unrooted methods, the variance is larger for values between $n = 1$ and $n = 4$, and then it decreases for values larger than $n = 5$. Hence, when the internal branches are very small, we do not have a highly accurate estimate of the caterpillar species tree topology, but the variance of the estimation is small compared to cases for the relatively larger internal branch lengths.

Consider the case for the balanced tree as shown in Figure 6.2. Here, we see that the

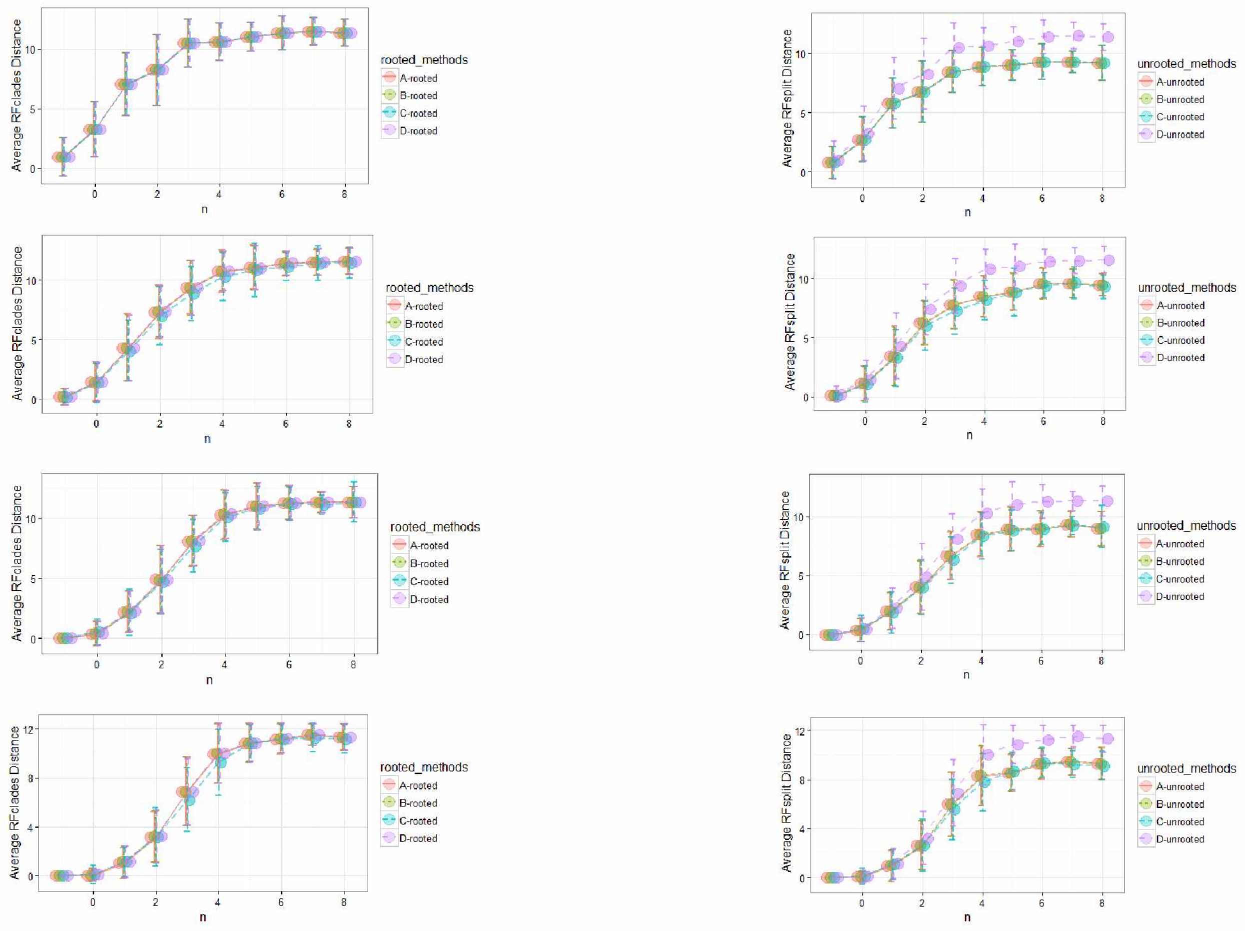

Figure 6.1: Species tree inference error plots for a caterpillar species tree. Plots show results for 50 repetitions and sample sizes of 1, 3, 6 and 10 gene trees . The plots are organized into 4 rows and 2 columns; the four rows correspond to the sample sizes, the first row being the sample size of 1 and the fourth being the sample size of 10. The left and right columns correspond to the rooted and unrooted versions respectively.

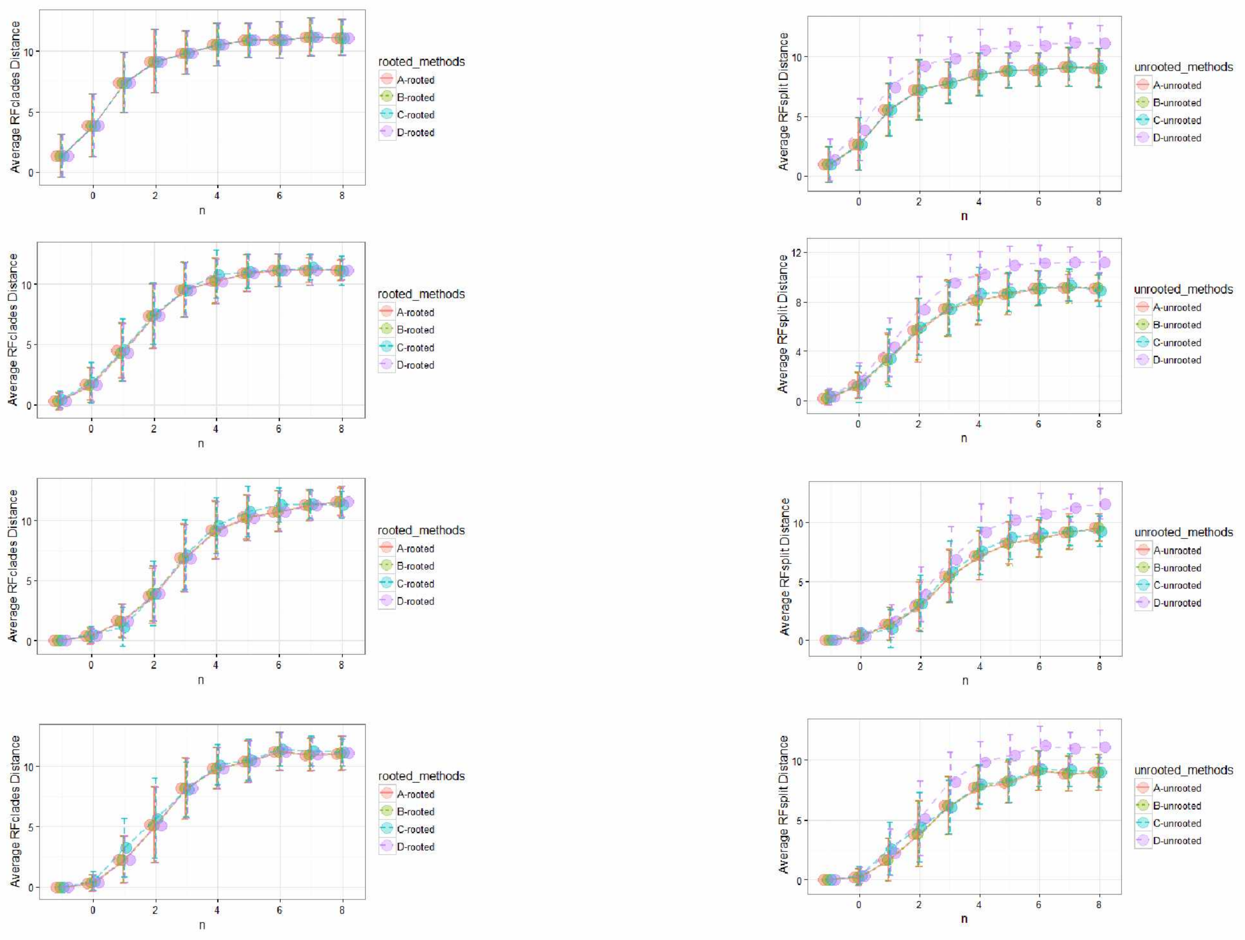

Figure 6.2: Species tree inference error plots for a balanced species tree. Plots show results for 50 repetitions and sample sizes of 1, 3, 6 and 10 gene trees. The plots are organized into 4 rows and 2 columns; the four rows correspond to the sample sizes, the first row being the sample size of 1 and the fourth being the sample size of 10. The left and right columns correspond to the rooted and unrooted versions respectively.

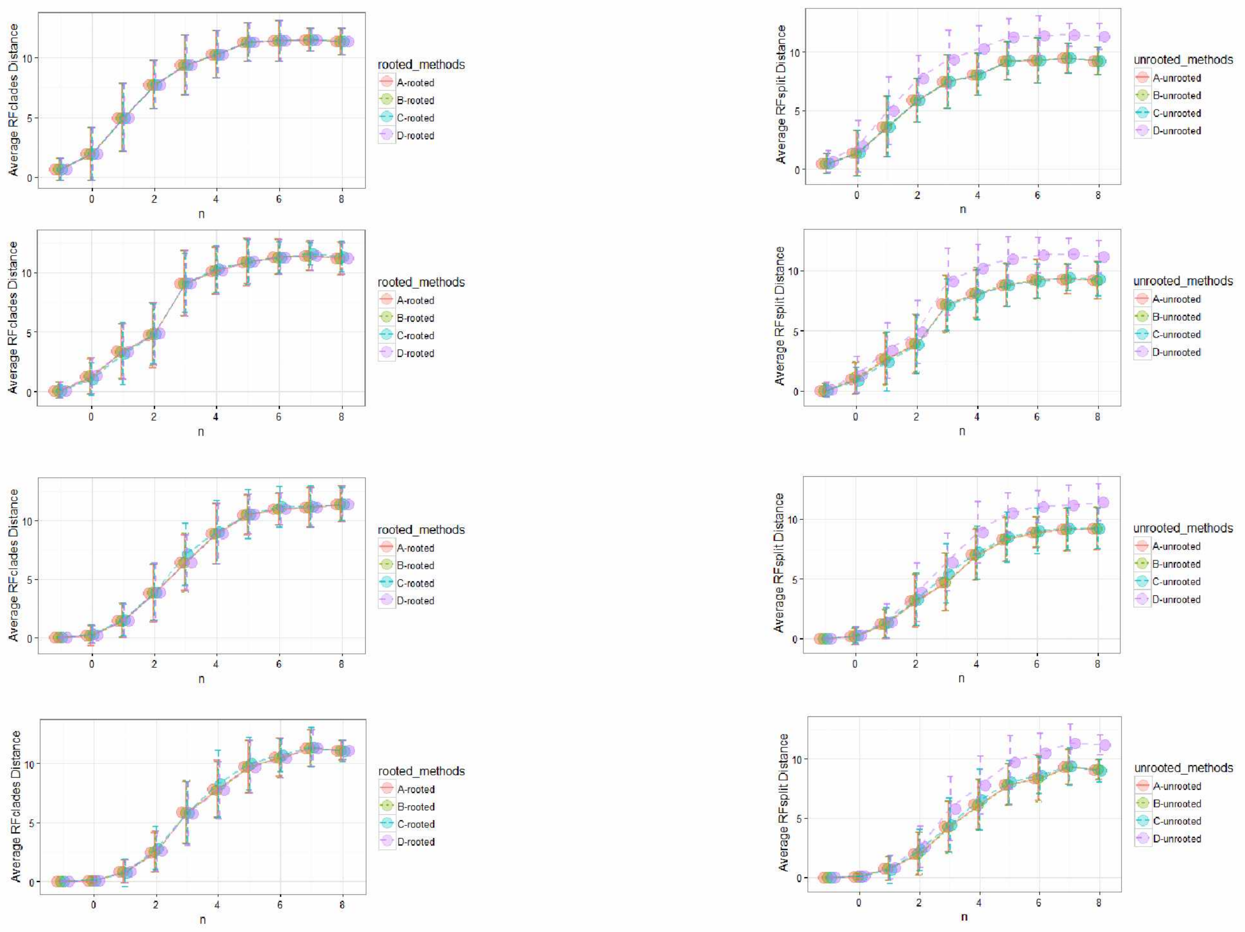

Figure 6.3: Species tree inference error plots for a mixed species tree. Plots show results for 50 repetitions and sample sizes of 1, 3, 6 and 10 gene trees. The plots are organized into 4 rows and 2 columns; the four rows correspond to the sample sizes, the first row being the sample size of 1 and the fourth being the sample size of 10. The left and right columns correspond to the rooted and unrooted versions respectively.

opposite behavior of that observed with the caterpillar occurs; the C-rooted variant does worse than all the other rooted methods, and additionally, D-rooted performs best. For the unrooted methods, we see again that D-unrooted performs significantly worse than all other variants. On the other hand, unlike the caterpillar, C-unrooted does not perform as strongly; in fact, variants A-unrooted and B-unrooted perform equally and better than all other variants. As noted in the case for the caterpillar species tree, these simulations also show the same pattern with variances. They are large for mid-size branch lengths, but as the internal branches become smaller, the estimate becomes more inaccurate and the variance decreases significantly.

Finally, consider the case for the mixed tree as shown in Figure 6.3. For the rooted methods, C-rooted does slightly worse on average. It is difficult to see the behavior of C-rooted for a sample size of 3, but for a sample size of 10 gene trees, it is clear that C-rooted does worst, and in addition, it is clear that D-rooted outperforms all other variants. For the unrooted versions, we still see that D-unrooted does significantly worse than other variants, and again A-rooted and B-rooted seem to perform equally and better than the rest. Therefore, the behaviors of the variants observed in the simulations where the species tree is a balanced tree are the same as those observed where the species tree is a mixed tree.

From these observations, we can outline some general conclusions. We see that the relative performance of C-rooted shows an apparent dependence on the topology of the species tree. By relative performance we mean performance in comparison to the other methods. In this case, relative to methods A-rooted, B-rooted and D-rooted, method C-rooted performs best if the true species tree is a caterpillar; otherwise, it performs the worst. Because of such variability in performance, this indicates that C-rooted is not a reliable method for inferring a true species tree topology.

To clarify this point further, consider Figure 6.4. Notice that the average RFclade distance for each method differs according to the species tree topology. Therefore, we cannot make any generalizations about the overall performance of each method across all topologies; we can only compare the methods relative to each other. It is also worth noticing that in Figure 6.4, the mixed tree topology seems to have the best results for methods A,B, and C. In other words, the average RFclade and RFsplit distance for methods A,B, and C are the smallest if the species tree topology is mixed. This result is reassuring in the sense that the majority of the gene tree topologies from real data would result in a mixed tree topology,

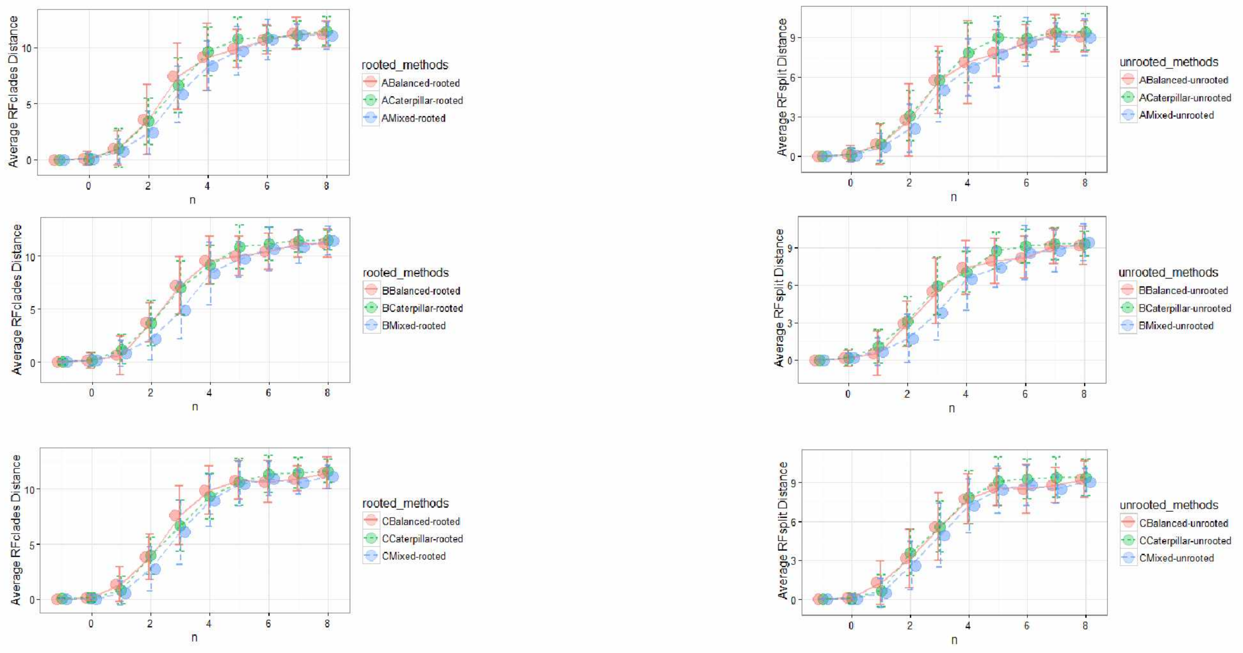

Figure 6.4: Species tree inference error plots for methods A, B and C on balanced, caterpillar and mixed species trees. The plots are organized into 3 rows and 2 columns; each row correspond to a method, the first being method A and the last being method B. The left and right columns are rooted and unrooted versions respectively. Each plot was constructed with 50 repetitions and a sample size of 10 gene trees.

which means that the analysis of real data could potentially have optimal results.

Another obvious observation is the under-performance of D-unrooted on the different simulation studies. It is clear from the plots that regardless of the true species tree topology, D-unrooted does poorly compared to other variants. This may be caused by the extra error that we added to this variant by attaching an extra taxon "ROOT".

Additionally, the relative performance of A-rooted/A-unrooted and B-rooted/B-unrooted is uniformly good throughout all simulations. There does not seem to be a difference in the performance of A variants compared to B variants, regardless of the species tree topology being inferred. Therefore, we can conclude that if we root a set of gene trees at the outgroup "O," infer a species tree from such set of gene trees using STAR with Neighbor Joining, and then root the species tree by "O," this would have the same effect on the accuracy of the true species tree estimation as if we instead do not root the gene trees at the outgroup "O," infer a species tree from such set of gene trees using USTAR with Neighbor Joining, and then root the resulting species tree by "O." We can draw the same conclusion if we do not root the inferred species trees by "O;" in other words, the unrooted versions of method A and B work comparatively well. In contrast, if we root a set of gene trees at the outgroup "O" and then infer a species tree from such set of gene trees using STAR with UPGMA, then the accuracy of the species tree estimation relative to the accuracy of A and B will vary depending on the true species tree topology. Finally, introducing more error by attaching extra taxa beyond the outgroup only worsens the species tree estimation, as shown in the D-unrooted methods.

6.2 Future Work

In this paper, all the gene trees we obtained came from the coalescent model using a known species tree topology. In other words, these gene trees had no computational error in them, which means we had a perfectly designed sample. However, in practice, DNA sequences do not always fit a phylogenetic tree; therefore, inferring gene trees from sequences of DNA data results in gene trees with errors. An important follow up study will consist of simulating gene trees and DNA sequences evolving on such gene trees. These DNA sequences can be used to infer gene trees by standard phylogenetic methods. This set of inferred gene trees will contain errors, and we would want to observe the behavior of our proposed methods when inferring a species tree from such set of gene trees. The performance of the variants of STAR and USTAR on this simulation study will have a stronger meaning for evaluating the usefulness of our proposed methods on real data.

Appendix

Listing 6.1: Codes Required to Plot RFsplit and RFclade Distances

```
#The following set of functions are compiled so that we can use the function
#"distplotnshort," which is the function that plots the average RClade and RFsplit distance
    for the method we discussed in Chapter 5 of this paper.


`spsimulation` <-
function(sptree,Nsamples, Nsimulations)
#This code simulates gene trees using the coalescent model.
#Nsample is the number of lineages per taxon.
#This code requires loading package 'ape'.

{
genetrees<- rep("", Nsimulations)

for (j in 1:Nsimulations)
{
#Inputs required by sim.coaltree.sp
spname<-species.name(sptree)
name<-spname
nspecies<- length(spname)
nodematrix<-read.tree.nodes(sptree, spname)$nodes
rootnode<-2*nspecies - 1
seq<-rep(Nsamples,nspecies)

# Simulate gene trees

genetrees[j]<- sim.coaltree.sp(rootnode,nodematrix,nspecies,seq,spname)$gt
}

#Inputs required by star.sptree
speciesname<-species.name(genetrees[1])
taxaname<-speciesname
matrixnumber= length(speciesname)
species.structure<-matrix(0,ncol= matrixnumber,nrow= matrixnumber)
diag(species.structure)<-1

# Build Species tree using STAR

infsptree.nj<-paste( "Neighbor␣Joining:", star.sptree(genetrees, speciesname, taxaname,
    species.structure, outgroup="0", method="nj"), sep="")
infsptree.upgma<-paste("UPGMA:", star.sptree(genetrees, speciesname, taxaname, species.
    structure, outgroup="0", method="upgma"), sep="")
tree1= star.sptree(genetrees, speciesname, taxaname, species.structure, outgroup="0", method
    ="nj")
```

```
}




#The following are functions that are required by "spsimulation" and "countmethods.RF".

`sim.coaltree.sp` <-
function(rootnode,nodematrix,nspecies,seq,name)
{
theta<-nodematrix[rootnode,5]

if(rootnode<=nspecies){
{if(seq[rootnode] == 1){
z<-list(gt="", height=as.matrix, node=as.matrix)
z$gt<-name[rootnode]
z$height<-0
z$node<-nodematrix
return(z)}
else{
treestr<-paste(name[rootnode],"s",1:seq[rootnode],sep="")
i<-seq[rootnode]
height<-rexp(1,rate=i*(i-1)/theta)
brlens<-rep(0,i)
father<-nodematrix[rootnode,1]
fatherheight<-node.height(father,nodematrix,nspecies)

while(height<fatherheight){
nodematrix[rootnode,6]<-nodematrix[rootnode,6]+1
##randomly choose two nodes
b<-sample(1:i,2)

##update groups
newname<-paste("(",treestr[b[1]],sep="")
newname<-paste(newname,":",sep="")
newname<-paste(newname,round(height-brlens[b[1]],6),sep="")
newname<-paste(newname,",",sep="")
newname<-paste(newname,treestr[b[2]],sep="")
newname<-paste(newname,":",sep="")
newname<-paste(newname,round(height-brlens[b[2]],6),sep="")
newname<-paste(newname,")",sep="")
treestr[b[1]]<-newname
brlens[b[1]]<-height

##update dist,treestr, and branch length
index<-1:i
index[b[2]]<-0
index<-index[index>0]
treestr<-treestr[index]
brlens<-brlens[index]
if(i==2)
break
i<-i-1
height<-height+rexp(1,rate=i*(i-1)/theta)
}
z<-list(gt="", height=as.matrix,node=as.matrix)
z$gt<-treestr
z$height<-brlens
z$node<-nodematrix
return(z)
}
}

}
if(rootnode>nspecies){
son1<-nodematrix[rootnode,2]
son2<-nodematrix[rootnode,3]
leftstr<-sim.coaltree.sp(rootnode=son1,nodematrix=nodematrix,nspecies,seq,name)
nodematrix<-leftstr$node
rightstr<-sim.coaltree.sp(rootnode=son2,nodematrix=nodematrix,nspecies,seq,name)
nodematrix<-rightstr$node
i<-length(leftstr$gt)+length(rightstr$gt)

treestr<-1:i
treestr[1:length(leftstr$gt)]<-leftstr$gt
treestr[(length(leftstr$gt)+1):i]<-rightstr$gt

brlens<-1:i
brlens[1:length(leftstr$height)]<-leftstr$height
brlens[(length(leftstr$height)+1):i]<-rightstr$height



height<-rexp(1,rate=i*(i-1)/theta)+node.height(rootnode,nodematrix,nspecies)
father<-nodematrix[rootnode,1]
if(father == -9 | father == -8)
fatherheight<-100000 else
fatherheight<-node.height(father,nodematrix,nspecies)
brlens


while(height<fatherheight){
nodematrix[rootnode,6]<-nodematrix[rootnode,6]+1
##randomly choose two nodes
b<-sample(1:i,2)
```

```
##update groups
newname<-paste("(",treestr[b[1]],sep="")
newname<-paste(newname,":",sep="")
newname<-paste(newname,round(height-brlens[b[1]],6),sep="")
newname<-paste(newname,",",sep="")
newname<-paste(newname,treestr[b[2]],sep="")
newname<-paste(newname,":",sep="")
newname<-paste(newname,round(height-brlens[b[2]],6),sep="")
newname<-paste(newname,")",sep="")
treestr[b[1]]<-newname
brlens[b[1]]<-height

##update dist,treestr, and branch length
index<-1:i
index[b[2]]<-0
index<-index[index>0]
treestr<-treestr[index]
brlens<-brlens[index]
if(i==2)
break
i<-i-1
height<-height+rexp(1,rate=i*(i-1)/theta)
}
if(nodematrix[rootnode,1]==-9 | nodematrix[rootnode,1]==-8)
treestr<-paste(treestr,";",sep="")
z<-list(gt="", height=as.matrix,node=as.matrix)
z$gt<-treestr
z$node<-nodematrix
z$height<-brlens
return(z)
}

}


`node.height` <-
function(inode,nodematrix,nspecies)
{
if(inode<=nspecies)
height<-0
if(inode>nspecies)
{
son1<-nodematrix[inode,2]
height<-node.height(son1,nodematrix,nspecies)+nodematrix[son1,4]
}
return(height)
}

`star.sptree` <-
```

```
function (trees, speciesname, taxaname, species.structure,outgroup, method="nj")
{
ntree <- length(trees)
nspecies <- length(speciesname)
ntax<-length(taxaname)
dist <- matrix(0, nrow=ntree, ncol=ntax*ntax)

for (i in 1:ntree)
{
ranktree1 <- read.tree.nodes(trees[i])
thistreetaxa<-ranktree1$names
ntaxaofthistree<-length(thistreetaxa)
ranktree<-ranktree1$nodes
thistreenode<-rep(-1,ntaxaofthistree)

for(j in 1:ntaxaofthistree)
{
thistreenode[j]<-which(taxaname == thistreetaxa[j])# these names, taxanme and thistreetaxa
    arent the same
if(length(thistreenode[j])==0)
{
print(paste("wrong␣taxaname",thistreetaxa[j],"in␣gene",i))
return(0)
}
}


if(!is.rootedtree(ranktree))
{
root<-which(thistreetaxa==outgroup)
if(length(root) == 0)
warnings(paste("outgroup␣is␣missing␣at␣tree",i))
else
ranktree<-root.tree(ranktree,root)
}


a<-rep(0,2*ntaxaofthistree-1)
ranknode<-rank.nodes(ranktree,rootoftree(ranktree),ntaxaofthistree, ntax,a)

dist1 <- matrix(0, ntax, ntax)

for(j in (ntaxaofthistree+1):(2*ntaxaofthistree-1))
{
son1<-offspring.nodes(ranktree[j,2],ranktree,ntaxaofthistree)
son1<-son1[son1<=ntaxaofthistree]
son2<-offspring.nodes(ranktree[j,3],ranktree,ntaxaofthistree)
son2<-son2[son2<=ntaxaofthistree]
for(k in 1:length(son1))
for(l in 1:length(son2))
{
dist1[thistreenode[son1[k]],thistreenode[son2[l]]]<-ranknode[j]*2
}
}
dist[i,]<-as.numeric(dist1+t(dist1))

}
dist[dist==0]<-NA
dist2<-matrix(apply(dist,2,mean,na.rm=TRUE),ntax,ntax)
diag(dist2)<-0
if(sum(is.nan(dist2))>0)
{
print("missing␣species!")
dist2[is.nan(dist2)]<-10000
}


dist3 <- pair.dist.mulseq(dist2, species.structure)
dist <- dist3



{
if(method == "upgma") sptree2 <- upgmaR(dist,speciesname)$treestr
if(method == "nj")
{
sptree<-write.tree(nj(dist))
sptree1<-read.tree.nodes(sptree,speciesname)$nodes
root<-which(taxaname==outgroup)
root<-species.structure[,root]
root<-which(root==1)
sptree2<-root.tree(sptree1,root)
a<-sptree2[,4]
b<-which(a<0)
if(length(b) > 1)
sptree2[b[1:(length(b)-1)],4]<-0
sptree2<-write.subtree(rootoftree(sptree2),sptree2,speciesname,rootoftree(sptree2))
}

#else print("choose either upgma or nj")
}

sptree<-sptree2
return(sptree)
}
```

```
read.tree.nodes <-function(str,name="")
{
#eliminate the content between []
str<-gsub("\\[.*\\]","",str)
nobrlens <- 0
if(length(grep(":",str))==0)
{
nobrlens <- 1
str<-gsub(",",":1.0,",str)
str<-gsub(")",":1.0)",str)
str<-gsub(";",":1.0;",str)
}
string <- unlist(strsplit(str, NULL))
leftpar<-which(string=="(")
rightpar<-which(string==")")
if(length(leftpar) != length(leftpar))
stop("The␣number␣of␣left␣parenthesis␣is␣NOT␣equal␣to␣the␣number␣of␣right␣␣parenthesis")

speciesname<-sort(species.name(str))
nspecies<-length(speciesname)

{if(length(leftpar) == (nspecies-1))
rooted<-TRUE
else if(length(leftpar) == (nspecies-2))
rooted<-FALSE
else
stop("The␣number␣of␣comma␣in␣the␣tree␣string␣is␣wrong!")}

if(length(name)>1 & (nspecies != length(name)))
stop("Wrong␣number␣of␣species␣names!")

if(length(name)>1)
speciesname<-name
{if(rooted)
nNodes<-2*nspecies-1
else
nNodes<-2*nspecies-2
nodes<-matrix(-9,nrow=nNodes,ncol=7)}

str1<-str
if(length(grep("[a-z]",speciesname,ignore.case=TRUE)))
str1<-name2node(str1,speciesname)
father<-nspecies+1

while(father < (nNodes+1))
{
```

```
string <- unlist(strsplit(str1, NULL))
leftpar<-which(string=="(")
rightpar<-which(string==")")
colon<-which(string==":")

{if(length(leftpar) == 1) substr <- paste(string[leftpar[sum(leftpar < rightpar[1])]:
    rightpar[1]],
sep = "", collapse = "")
else substr <- paste(string[leftpar[sum(leftpar < rightpar[1])]:(colon[which(colon>rightpar
    [1])[1]]-1)],
sep = "", collapse = "")}

substring<-unlist(strsplit(substr, NULL))
colon<-which(substring==":")
comma<-which(substring==",")
pound<-which(substring=="#")
percent<-which(substring=="%")
combine<-which(substring=="," | substring==")" | substring=="#" | substring=="%")

node1<-as.integer(paste(substring[2:(colon[1]-1)],sep="",collapse=""))
node2<-as.integer(paste(substring[(comma[1]+1):(colon[2]-1)],sep="",collapse=""))

if(length(comma)>1)
node3<-as.integer(paste(substring[(comma[2]+1):(colon[3]- 1)],sep="",collapse=""))

#branch length
if(length(colon)==0)
{
node1Branch<--9;
node2Branch<--9;
}
if(length(colon)>0)
{
x1<-combine[sum(combine<colon[1])+1]-1
x2<-combine[sum(combine<colon[2])+1]-1
if(length(colon)==3)
{
x3<-combine[sum(combine<colon[3])+1]-1
nodes[node3,4]<-as.double(paste(substring[(colon[3] +1):x3],sep="",collapse=""))
}
node1Branch<-as.double(paste(substring[(colon[1] +1):x1],sep="",collapse=""))
node2Branch<-as.double(paste(substring[(colon[2] +1):x2],sep="",collapse=""))
}

#mutation rates
if(length(percent)==0)
{
node1mu<--9
node2mu<--9
```

```
}
if (length(percent)==1)
{
if(percent[1]<comma[1])
{
node1mu<-as.double(paste(substring[(percent[1]+1):(comma[1]- 1)],sep="",collapse=""))
node2mu<--9
}
else
{
node2mu<-as.double(paste(substring[(percent[1]+1):(length (substring)-1)],sep="",collapse=""
    ))
node1mu<--9
}
}
if(length(percent)==2)
{
node1mu<-as.double(paste(substring[(percent[1]+1):(comma[1]- 1)],sep="",collapse=""))
node2mu<-as.double(paste(substring[(percent[2]+1):(length(substring)- 1)],sep="",collapse=""
    ))
}
if(length(percent)==3)
{
node1mu<-as.double(paste(substring[(percent[1]+1):(comma[1]- 1)],sep="",collapse=""))
node2mu<-as.double(paste(substring[(percent[2]+1):(comma[2]- 1)],sep="",collapse=""))
node3mu<-as.double(paste(substring[(percent[3]+1):(length(substring)- 1)],sep="",collapse=""
    ))
nodes[node3,5]<-node3mu
}

#population size
if(length(percent) == 0)
{
if(length(pound)==0)
{
node1theta<--9
node2theta<--9
}
if (length(pound)==1)
{
if(pound[1]<comma[1])
{
node1theta<-as.double(paste(substring[(pound[1]+1):(comma[1]- 1)],sep="",collapse=""))
node2theta<--9
}
else
{
node2theta<-as.double(paste(substring[(pound[1]+1):(length (substring)-1)],sep="",collapse="
    "))
node1theta<--9
}
}
if(length(pound)==2)
{
node1theta<-as.double(paste(substring[(pound[1]+1):(comma[1]- 1)],sep="",collapse=""))
node2theta<-as.double(paste(substring[(pound[2]+1):(length(substring)- 1)],sep="",collapse="
    "))
}
if(length(pound)==3)
{
node1theta<-as.double(paste(substring[(pound[1]+1):(comma[1]- 1)],sep="",collapse=""))
node2theta<-as.double(paste(substring[(pound[2]+1):(comma[2]- 1)],sep="",collapse=""))
node3theta<-as.double(paste(substring[(pound[3]+1):(length(substring)- 1)],sep="",collapse="
    "))
nodes[node3,5]<-node3theta
}
}
if(length(percent)>0)
{
if(length(pound)==0)
{
node1theta<--9
node2theta<--9
}
if (length(pound)==1)
{
if(pound[1]<comma[1])
{
node1theta<-as.double(paste(substring[(pound[1]+1):(percent[1]- 1)],sep="",collapse=""))
node2theta<--9
}
else
{
node2theta<-as.double(paste(substring[(pound[1]+1):(percent[2]-1)],sep="",collapse=""))
node1theta<--9
}
}
if(length(pound)==2)
{
node1theta<-as.double(paste(substring[(pound[1]+1):(percent[1]- 1)],sep="",collapse=""))
node2theta<-as.double(paste(substring[(pound[2]+1):(percent[2]- 1)],sep="",collapse=""))
}
if(length(pound)==3)
{
node1theta<-as.double(paste(substring[(pound[1]+1):(percent[1]- 1)],sep="",collapse=""))
node2theta<-as.double(paste(substring[(pound[2]+1):(percent[2]- 1)],sep="",collapse=""))
node3theta<-as.double(paste(substring[(pound[3]+1):(percent[3]- 1)],sep="",collapse=""))
nodes[node3,5]<-node3theta
```

```
}
}
nodes[node1,1]<-father
nodes[node1,4]<-node1Branch
nodes[node1,5]<-node1theta
nodes[node1,6]<-node1mu

nodes[node2,1]<-father
nodes[node2,4]<-node2Branch
nodes[node2,5]<-node2theta
nodes[node2,6]<-node2mu

if(length(comma)>1)
{
nodes[node3,1]<-father
nodes[father,4]<-node3
}

nodes[father,2]<-node1
nodes[father,3]<-node2

rightpar1 <- which(substring == ")")
if (rightpar1 < length(substring))
{
postprob <- paste(substring[(rightpar1+1):length(substring)],sep = "", collapse = "")
nodes[father, 7] <- as.numeric(postprob)
}

substr<-gsub("[(]","[(]",substr)
substr<-gsub("[)]","[)]",substr)
substr<-gsub("\\+","",substr)
str1<-gsub("\\+","",str1)
str1<-gsub(substr,father,str1)
father<-father+1
}

if(length(grep("%",str1)))
nodes[nNodes,6]<-as.double(gsub(";","",gsub(".*\\%","",str1)))
if(length(grep("#",str1)))
{
if(length(grep("%",str1)))
nodes[nNodes,5]<-as.double(gsub(".*\\#","",gsub("\\%.*","",str1)))
else
nodes[nNodes,5]<-as.double(gsub(";","",gsub(".*\\#","",str1)))
}
if(!rooted)
nodes[nNodes,1]<--8
if(nobrlens == 1) nodes[,4]<--9
```

```
z <- list(nodes = matrix(0, nNodes, 5), names ="", root=TRUE)

z$nodes<-nodes
z$names<-speciesname
z$root<-rooted
z
}

`offspring.nodes` <-
function(inode,nodematrix,nspecies)
{
string<-offspring.nodes.string(inode,nodematrix,nspecies)
offspringnodes<-as.numeric(unlist(strsplit(string,split="␣")))
if(nodematrix[inode,1] == -8)
offspringnodes<-dim(nodematrix)[1]:1
return(as.numeric(unlist(strsplit(string,split="␣"))))
}




`species.name` <-
function(str)
{
str<-gsub("[␣]", "", str)
str<-gsub("[;.]","",str)
str<-gsub(":[0-9e-]*","",str)
str<-gsub("#[0-9e-]*","",str)
str<-gsub(")[0-9e-]*","",str)
str<-gsub("[()]","",str)
str<-gsub("[␣]", "", str)
name<-sort(unlist(strsplit(str,split=",")))

return(name)
}



`rootoftree` <-
function (nodematrix)
{
if(sum(nodematrix[,1]<0)>1)
stop("more␣than␣two␣roots!")
nodes<-sum((nodematrix[,1]<0)*(1:length(nodematrix[,1])))
return(nodes)
}
```

```
‘root.tree‘ <-
function(nodematrix,outgroup)
{


node<-change.root(nodematrix,nodematrix[outgroup,1])
nodes<-node$nodes
oldroot<-node$rootnode

rootnode<-matrix(-9,1,dim(nodematrix)[2])
rootnode[1,2]<-outgroup
rootnode[1,3]<-oldroot


if(nodes[oldroot,2]==outgroup)
nodes[oldroot,2]<-nodes[oldroot,4]
if(nodes[oldroot,3]==outgroup)
nodes[oldroot,3]<-nodes[oldroot,4]

nodes[oldroot,4]<-nodes[outgroup,4]/2
nodes[outgroup,4]<-nodes[outgroup,4]/2

nodes[oldroot,1]<-dim(nodes)[1]+1
nodes[outgroup,1]<-dim(nodes)[1]+1

roottree<-rbind(nodes,rootnode)

return(roottree)

}


'spstructure'<-
function(numsgenenodes)
{

nspecies<-length(numsgenenodes)
ntaxa<-sum(numsgenenodes)
sp<-matrix(0,nrow=nspecies,ncol=ntaxa)
index<-matrix(0,nrow=ntaxa,ncol=2)
index[,2]<-1:ntaxa

b<-1
for(i in 1:nspecies)
{
for(j in 1:numsgenenodes[i])
{
index[b,1]<-i
b<-b+1
}
}
sp[index]<-1
return(sp)
}


`is.rootedtree` <-
function(tree)
{
if(is.character(tree))
{
string <- unlist(strsplit(tree, NULL))
leftpar<-which(string=="(")
rightpar<-which(string==")")
comma<-which(string==",")
if(length(leftpar) != length(leftpar))
stop("The␣number␣of␣left␣parenthesis␣is␣NOT␣equal␣to␣the␣number␣of␣right␣parenthesis")
if(length(leftpar) == length(comma))
rooted<-TRUE
else if(length(leftpar) == (length(comma)-1))
rooted<-FALSE
else
stop("The␣number␣of␣comma␣in␣the␣tree␣string␣is␣wrong!")
}
else
{
rooted<-TRUE
if(tree[rootoftree(tree),1] == -8)
rooted<-FALSE
}
return(rooted)
}

`rank.nodes` <-
function(treenode, inode, ntaxa, start, rank)
{
if(inode > ntaxa)
{
left<-treenode[inode,2]
right<-treenode[inode,3]
rank[inode]<-start
rank[left]<-start-1
rank[right]<-start-1
rank<-rank.nodes(treenode,left,ntaxa,rank[left],rank)
rank<-rank.nodes(treenode,right,ntaxa,rank[right],rank)
return(rank)
}
```

```
else
{
return(rank)
}
}




`node2name` <-
function(treestr, name="")
{
str<-treestr
speciesname<-species.name(str)


old<-paste(1:length(name),":",sep="")
new<-paste(name,"$",sep="")
for(i in length(name):1)
str<-gsub(old[i],new[i],str)
str<-gsub("\\$",":",str)

str

}




`name2node` <-
function(treestr,name="")
{
str<-treestr
if(length(name)<2)
speciesname<-sort(species.name(str))
else
speciesname<-name

new<-1:length(speciesname)
old<-speciesname
strlength<-nchar(old)
inputorder<-order(rank(strlength,ties.method="random"),decreasing=TRUE)
for(i in 1:length(speciesname))
str<-gsub(old[inputorder[i]],new[inputorder[i]],str)

str

}
```

```
`offspring.nodes.string` <-
function(inode,nodematrix,nspecies)
{
if(inode<1 | inode>dim(nodematrix)[1]){
a<-paste("The␣node␣number␣should␣be␣between␣1␣and",dim(nodematrix)[1])
stop(a)
}
if(inode<=nspecies)
return(paste(inode))
if(inode>nspecies){
son1<-nodematrix[inode,2]
son2<-nodematrix[inode,3]
str1<-offspring.nodes.string(son1,nodematrix,nspecies)
str2<-offspring.nodes.string(son2,nodematrix,nspecies)
str<-paste(inode)
str<-paste(str,str1)
return(paste(str,str2))
}
}




`pair.dist.mulseq` <-
function(dist,species.structure)
{
for(i in 1:dim(species.structure)[1]){
species.structure[i,]<-species.structure[i,]/sum(species.structure[i,])
}

dis<-round((species.structure)%*%dist%*%t(species.structure),8)
diag(dis)<-0
dis
}



`change.root` <-
function(nodematrix, newroot)
{

root<-rootoftree(nodematrix)

if(newroot != root){
##unroot the tree
if(nodematrix[root,1] == -9)
nodes<-unroottree(nodematrix)   else
nodes<-nodematrix
nspecies<-dim(nodes)[1]/2+1
##new root should be the internodes
if(newroot<=nspecies)
stop("new␣root␣should␣be␣the␣internodes!")

anc<-ancestor(newroot,nodes)
nanc<-length(anc)
##update old root
a<-nodes[dim(nodes)[1],2:4]
a[a==anc[nanc-1]]<-a[3]
nodes[dim(nodes)[1],2:4]<-a

##reverse the ancestral history
if(nanc==2){
##reverse son, father, and branch length
brlens<-nodes[anc[1],4]
nodes[anc[1],4]<-anc[2]
nodes[anc[2],1]<-anc[1]
nodes[anc[2],4]<-brlens
}

if(nanc>2){
##reverse son
brlens<-nodes[anc[1],4]
nodes[anc[1],4]<-anc[2]
for(i in 2:(nanc-1)){
if(nodes[anc[i],2]==anc[i-1])
nodes[anc[i],2]<-anc[i+1]
else
nodes[anc[i],3]<-anc[i+1]
}
##reverse son and branch length
nodes[anc[2:nanc],1]<-anc[1:(nanc-1)]
nodes[anc[2:nanc],4]<-nodes[anc[1:(nanc-1)],4]
nodes[anc[2],4]<-brlens
}
nodes[newroot,1]<--8
}else{
nodes<-nodematrix
}

z <- list(nodes = matrix(0, dim(nodes)[1], 5), rootnode=as.integer)

z$nodes<-nodes
z$rootnode<-newroot
z
}
```

```
‘write.subtree‘ <-
function (inode, nodematrix, taxaname, root)
{
if(nodematrix[inode,2] > 0)
{
son1 <- nodematrix[inode,2]
son2 <- nodematrix[inode,3]
x <- paste("(",write.subtree(son1,nodematrix,taxaname, root),":",nodematrix[son1,4],",",
    write.subtree(son2,nodematrix,taxaname,root),":",nodematrix[son2,4],")",sep="")
}
else x <- taxaname[inode]
if(inode == root) x <- paste(x,";",sep="")
return(x)
}



‘ancestor‘ <-
function(inode,nodematrix)
{
if(!is.rootedtree(nodematrix))
{
warnings("The␣tree␣is␣not␣rooted!")
}
rootnode<-rootoftree(nodematrix)
nnodes<-dim(nodematrix)[1]

if(inode == rootnode)
ancestor<-rootnode
else
{
ancestor<-rep(0,nnodes)
ancestor[1]<-inode
i<-1
while(ancestor[i] != rootnode)
{
ancestor[i+1]<-nodematrix[ancestor[i],1]
i<-i+1
}
}
return(ancestor[ancestor>0])
}


’NJst’<-
function(genetrees, taxaname, spname, species.structure)
{

ntree<-length(genetrees)
```

```
ntaxa<-length(taxaname)
dist <- matrix(0, nrow = ntree, ncol = ntaxa * ntaxa)

for(i in 1:ntree)
{
genetree1 <- read.tree.nodes(genetrees[i])
thistreetaxa <- genetree1$names
ntaxaofthistree <- length(thistreetaxa)
thistreenode <- rep(-1, ntaxaofthistree)
dist1<-matrix(0,ntaxa,ntaxa)
for (j in 1:ntaxaofthistree)
{
thistreenode[j] <- which(taxaname == thistreetaxa[j])
if (length(thistreenode[j]) == 0)
{
print(paste("wrong␣taxaname", thistreetaxa[j],"in␣gene", i))
return(0)
}
}
dist1[thistreenode, thistreenode]<-nancdist(genetrees[i],thistreetaxa)$dist
dist[i,]<-as.numeric(dist1)
}

dist[dist == 0] <- NA
dist2 <- matrix(apply(dist, 2, mean, na.rm = TRUE), ntaxa, ntaxa)
diag(dist2) <- 0
if (sum(is.nan(dist2)) > 0)
{
print("missing␣species!")
dist2[is.nan(dist2)] <- 10000
}
speciesdistance <- pair.dist.mulseq(dist2, species.structure)

tree<-write.tree(nj(speciesdistance))
node2name(tree,name=spname)
}

'nancdist'<-
function(tree, taxaname)
{
ntaxa<-length(taxaname)
nodematrix<-read.tree.nodes(tree,taxaname)$nodes
if(is.rootedtree(nodematrix)) nodematrix<-unroottree(nodematrix)
dist<-matrix(0, ntaxa,ntaxa)
for(i in 1:(ntaxa-1))
for(j in (i+1):ntaxa)
{
anc1<-ancestor(i,nodematrix)
anc2<-ancestor(j,nodematrix)
```

```
n<-sum(which(t(matrix(rep(anc1,length(anc2)),ncol=length(anc2)))-anc2==0, arr.ind=TRUE)[1,])
    -3
if(n==-1) n<-0
dist[i,j]<-n
}
dist<-dist+t(dist)
z<-list(dist=as.matrix, taxaname=as.vector)
z$dist<-dist
z$taxaname<-taxaname
z
}

'upgmaR' <-

# This function takes a distance matrix and produces an ultrametric tree using UPGMA.
# NOTE: method "min" from phybase has been REMOVED

function(dist,name,method="average")
{
nspecies<-length(name)        # Number of species
treestr<-name                        # Names to be used and modified to build Newick format
    tree.
brlens<-rep(0,nspecies)         # Vector of distance from leaves to MRCA of clade (initially
    zero for 1-clades)
cladesize<- rep(1, nspecies) # Vector of counts of number of taxa in each clade already
    grouped (initially all 1s).
b<-rep(0,2)                          # This will be location of the smallest positive
    entry in dist


for(i in 1:(nspecies-1)){
## Find the minimum distance, determining clades to be amalgamated
diag(dist)<-NA                                                              # Change the
    diagonal of the distance matrix to NA so that the minimum number is not 0
position <- which(dist == min(dist, na.rm = TRUE))[1]  # Find the the first occurrence of
    the smallest number, searching column-wise.
b[1] <- floor(position/length(dist[1, ]))                        # This will be the
    column index.
b[2] <- position - b[1] * length(dist[1, ])                      # This will be the row
     index.
if (b[2] == 0)                                                        # If the
    position is at the bottom of the column, then we must adjust
b[2] <- length(dist[1, ])
else b[1] <- b[1] + 1

## Amalgamate clades and collapse to smaller distance matrix
halfdist12= dist[ b[2], b[1] ]/2    # Save half the smallest value in the matrix

if(method == "average"){
```

```
dist[b[1],]= ( dist[b[1],]*cladesize[b[1]] + dist[b[2],]*cladesize[b[2]])/(cladesize[b[1]] +
    cladesize[b[2]]) # Compute weighted average using clade size for new row/column of
    distance matrix
dist[,b[1]]= t(dist[b[1],])
}
if (method == "min") {
bdist <- dist[b[2], b[1]]
dist[, b[1]] <- pmin(dist[, b[1]], dist[, b[2]])
dist[b[1], ] <- pmin(dist[b[1], ], dist[b[2], ])
}
# Remove a column and row
index <- 1:(nspecies + 1 - i)
index[b[2]]<-0
index<-index[index>0]
dist<-dist[index,index]

##Build a Newick tree
newname<-paste("(",treestr[b[1]],sep="")
newname<-paste(newname,":",sep="")
newname<-paste(newname,round((halfdist12-brlens[b[1]]),5),sep="")
newname<-paste(newname,",",sep="")
newname<-paste(newname,treestr[b[2]],sep="")
newname<-paste(newname,":",sep="")
newname<-paste(newname,round((halfdist12-brlens[b[2]]),5),sep="")
newname<-paste(newname,")",sep="")
treestr[b[1]]<-newname
brlens[b]<-halfdist12

##Update clade sizes.
cladesize[b[1]]<- cladesize[b[1]] + cladesize[b[2]]
cladesize[b[2]]<- 0
cladesize<- cladesize[cladesize>0]

##
treestr<-treestr[index]
brlens<-brlens[index]
}

## Display results
treestr<-paste(treestr,";",sep="")
z<-list(nodes=matrix,treestr="",name="")
node<-read.tree.nodes(treestr,name)
z$nodes<-node$nodes
z$name<-node$name
z$treestr<-treestr
z
}
```

```
'labeltoporoot'<- function(tree)

#This function takes a NON-METRIC rooted tree in Newick, adds a label called ROOT and
    produces a rooted
#tree with the label ROOT in Newick format. New edges have arbitrary lengths.


{ tree1<- read.tree(text=tree)

#Make sure the tree is rooted
if(is.rooted(tree1)==TRUE)
{
tree<- sub(";", "", tree)
tree<- paste("(", tree, sep="")
tree<- paste(tree,",ROOT);", sep="")
}
else
{
print("Error:␣Can't␣place␣ROOT␣on␣unrooted␣tree")
}

tree
}




'RFclades'<-
function (tree1, tree2)
#This function takes two rooted trees in Newick format, attaches a label called "ROOT" and
#performs a Robinson Fould's Distance. This distance is interpreted as the clade distance.
    Note: trees cannot have
# any edge lengths

{
#Attach label "ROOT" to the trees
tree1<- labeltoporoot(tree1)
tree2<- labeltoporoot(tree2)

#Cnvert trees to phylo format ( neccesary for RF.dist)
tree1<- read.tree(text= tree1)
tree2<- read.tree(text=tree2)
#Compute Robinson Foulds Distance.
distance<- RF.dist(tree1, tree2)
distance

}
```

```
'labelroot'<-
function(tree)

#This function takes a METRIC rooted tree in Newick, adds a label called ROOT and produces a
     rooted
#tree with the label ROOT in Newick format. New edges have arbitrary lengths.


{ tree1<- read.tree(text=tree)

if(is.rooted(tree1)==TRUE)
{
tree<- sub(";", "", tree)
tree<- paste("(", tree, sep="")
tree<- paste(tree,":0.1,ROOT:0.001);", sep="")
}
else
{
print("Error:␣Can't␣place␣ROOT␣on␣unrooted␣tree")
}

tree
}





`meancountmethods.RF` <-
function(sptree,topology,Nsamples, Nsimulations, repetition)
## Purpose: This function  simulates gene trees according to the coalescent model, then
    takes these gene trees and estimates
#the species tree using STAR, and it computes the mean Robinson Fould's distance as well as
    its variance.

##This function requires loading packages 'ape' and 'phangorn'

## Due to sim.coal.sptree, species trees must be ultrametric.

##Nsample is the number of lineages per taxon.

## repetition is  the number of data points you want to collect.

##Nsimulations indicates the number of simulated gene trees you want from the coalescent
    code.

# Aunrooted :unroot gene  trees, reroot at O and run STAR to produce an unrooted species
    tree.
```

```
# Arooted: unroot gene  trees, reroot at O, run STAR and then reroot the resulting species
    tree at O.
# Bunrooted: unroot gene trees and run USTAR to produce an unrooted species tree.
# Brooted: unroot hene trees, run USTAR and then reroot the resulting species tree at O.
# Crooted: unroot gene  trees, reroot at O, run STAR using UPGMA to produce a rooted species
     tree.
# Cunrooted: unroot Crooted.
#Dunrooted:  USTAR on gene trees rerooted at "O" and then root labeled: reroot at o, then
    label root and then unroot to run USTAR.


{
#Inputs required by sim.coaltree.sp
spname<-species.name(sptree)
name<-spname
nspecies<- length(spname)
nodematrix<-read.tree.nodes(sptree, spname)$nodes
rootnode<-2*nspecies - 1
seq<-rep(Nsamples,nspecies)


#Label ROOT  the topological tree and convert it to phylo format

labeltopology<- sub(";", "", topology)
labeltopology<-paste("(", labeltopology, sep="")
labeltopology<- paste(labeltopology,",ROOT);", sep="")
labeltopology<- read.tree(text=labeltopology)


#Convert the original topological tree ( no label ROOT) from Newick format into phylo format
    .
rootedtopology<-read.tree(text=topology)

#Unroot the topological tree (phylo)
unrootedtopology<-unroot(rootedtopology)

#Unroot the label ROOT topological tree (phylo)
unrootedlabeltopology<- unroot(labeltopology)

#Allocate space for vectors in the following loop
genetrees<- rep("", Nsimulations)
labelrootedgenetrees<-rep("", Nsimulations)
rerooted=rep("", Nsimulations)
unrootedtruetree=rep("", Nsimulations)
rerootedlabelgenetrees=rep("", Nsimulations)

#Allocate space for count vectors.
Arooted.count<- c()
Aunrooted.count<- c()
```

```
Brooted.count<- c()
Bunrooted.count<- c()
Crooted.count<- c()
Cunrooted.count<- c()
Drooted.count<-c()
Dunrooted.count<-c()
for (i in 1:repetition)

{

# SIMULATE GENE TREES

for (j in 1:Nsimulations)
{



#simulate genee trees according to coalescent model
genetrees[j]<- sim.coaltree.sp(rootnode,nodematrix,nspecies,seq,spname)$gt
labelrootedgenetrees[j]<-labelroot(genetrees[j])



#Trees used in A and C: unroot genetrees and reroot them at "O" (outgroup).
phylorooted<-read.tree(text=genetrees[j])# Convert the gene trees from the coalescent model
    into phylo format.
phylorooted<-root(phylorooted, "O", resolve.root=TRUE)# Unroot the phylo formatted gene
    trees and reroot them at the outgroup "O".
rerooted[j]<-write.tree(phylorooted) # Convert the phylo rerrooted trees back to Newick
    format and store them in a vector.



#Trees used in B: Unroot the gene trees from the coalescent model.
unrooted<- read.tree(text = genetrees[j])# conert gene trees to phylo
unrooted<- unroot(unrooted) # Unroot the gene trees
unrootedtruetree[j]<-write.tree(unrooted) # Convert  back to Newick.
# print(unrootedtruetree)



#Trees used in D :USTAR on gene trees rerooted at "O" and then root labeled: reroot at o,
    then label root and then unroot to run USTAR.:
Z<- labelroot(rerooted[j])
Z<-read.tree(text=Z)
A<-unroot(Z)
rerootedlabelgenetrees[j]<- write.tree(A)



}



# INPUTS REQUIRED BY "star.sptree" AND NJst.
```

```
#For trees without label ROOT.
speciesname<-species.name(genetrees[1])
taxaname<-speciesname
matrixnumber= length(speciesname)
species.structure<-matrix(0,ncol= matrixnumber,nrow= matrixnumber)
diag(species.structure)<-1

#For trees with label ROOT.
speciesname2<-species.name(labelrootedgenetrees[1])
nspecies2=length(speciesname2)
taxaname2<-speciesname2
matrixnumber2= length(speciesname2)
species.structure2<-matrix(0,ncol= matrixnumber2,nrow= matrixnumber2)
diag(species.structure2)<-1


# BUILD SPECIES TREES USING STAR AND NJst (USTAR). THE TREES ARE BUILT WITH NEIGHBOR JOINING
     AND UPGMA.

#The following are comparisons among rooted gene trees




# Arooted:
Arooted<-star.sptree(rerooted, speciesname, taxaname, species.structure, outgroup="0",
    method="nj")
Arooted<-read.tree(text=Arooted)

# Aunrooted:
Aunrooted<-unroot(Arooted)
Arooted<-drop.tip(Arooted, c("0"), trim.internal=FALSE) #Drop 0 from Arooted tree
# Bunrooted
Bunrooted<-NJst(unrootedtruetree, taxaname, speciesname, species.structure)
Bunrooted<- read.tree(text=Bunrooted)
#Brooted
Brooted<- root(Bunrooted, "0", resolve.root=TRUE)
#Crooted
Crooted<-star.sptree(rerooted, speciesname, taxaname, species.structure, outgroup="0",
    method="upgma")
Crooted<- read.tree(text=Crooted)
#Cunrooted
Cunrooted<- unroot(Crooted)
Crooted<-drop.tip(Crooted, c("0"), trim.internal=FALSE) #Drop 0 from Crooted tree
#Dunrooted: USTAR on gene trees rerooted at "0" and then root labeled: reroot at o, then
    label root and then unroot to run USTAR.
Dunrooted<-NJst(rerootedlabelgenetrees,taxaname2, speciesname2, species.structure2 )
Dunrooted<-read.tree(text=Dunrooted)
Drooted<-root(Dunrooted, "ROOT", resolve.root=TRUE)
```

```
#Arooted RF distance (Using RFclades)
topology1<-read.tree(text=topology)
topology1<-drop.tip(topology1, c("O"), trim.internal=FALSE) # Drop "O" from topology
Arooted1=Arooted
Arooted1$edge.length<-NULL #Remove the edge lengths
Arooted=write.tree(Arooted1)
topology1=write.tree(topology1)
Arooted.dist<-RFclades(Arooted, topology1)

#Aunrooted RF distance
Aunrooted<-drop.tip(Aunrooted, c("O"), trim.internal=FALSE)
unrootedtopology1<-drop.tip(unrootedtopology, c("O"), trim.internal=FALSE) #Unrootedtopology
     without 'O'
Aunrooted.dist<-RF.dist(Aunrooted,unrootedtopology1)


#Brooted RF distance (Using RFclades)
Brooted= drop.tip(Brooted, c("O"), trim.internal=FALSE)
Brooted1<-Brooted
Brooted1$edge.length<-NULL #Remove the edge lengths
Brooted<- write.tree(Brooted1)
Brooted.dist<- RFclades(Brooted, topology1)

#Bunrooted RF distance
Bunrooted<- drop.tip(Bunrooted, c("O"), trim.internal=FALSE)
Bunrooted<- unroot(Bunrooted)###   NOTE :Not sure why R requires this
unrootedtopology1<-unroot(unrootedtopology1)###
Bunrooted.dist<- RF.dist(Bunrooted, unrootedtopology1)

#Crooted RF distance (Using RFclades)
Crooted1<-Crooted
Crooted1$edge.length<-NULL#Remove the edge lengths
Crooted<- write.tree(Crooted1)
Crooted.dist<- RFclades(Crooted, topology1)


#Cunrooted RF distance
Cunrooted<-drop.tip(Cunrooted, c("O"), trim.internal=FALSE)
Cunrooted<-unroot(Cunrooted)
Cunrooted.dist<- RF.dist(Cunrooted, unrootedtopology1)

#Dunrooted RF distance #Should it be RFclades(rerooted) instead?
Dunrooted1<-Dunrooted
Dunrooted1$edge.length<-NULL
Dunrooted1<-drop.tip(Dunrooted1, c("ROOT"), trim.internal=FALSE)
Dunrooted1<-unroot(Dunrooted1)
```

```
Dunrooted<-Dunrooted1
Dunrooted.dist<-RF.dist(Dunrooted,unrootedtopology )

#Drooted
Drooted1<-Drooted
Drooted1$edge.length<-NULL
Drooted1<-drop.tip(Drooted1, c("ROOT"), trim.internal=FALSE)
Drooted1<-drop.tip(Drooted1, c("0"), trim.internal=FALSE)
Drooted<-Drooted1
Drooted<-write.tree(Drooted)
Drooted.dist<- RFclades(Drooted,topology1)


#Build a vector of  distance counts from Robinson Foulds

Arooted.count[i]<- Arooted.dist
Aunrooted.count[i]<- Aunrooted.dist
Brooted.count[i]<- Brooted.dist
Bunrooted.count[i]<- Bunrooted.dist
Crooted.count[i]<- Crooted.dist
Cunrooted.count[i]<- Cunrooted.dist
Drooted.count[i]<- Drooted.dist
Dunrooted.count[i]<- Dunrooted.dist

}

mean1=mean(Arooted.count)

mean2=mean(Aunrooted.count)

mean3=mean(Brooted.count)

mean4=mean(Bunrooted.count)

mean5=mean(Crooted.count)

mean6=mean(Cunrooted.count)

mean7= mean( Drooted.count)

mean8=mean( Dunrooted.count)


var1=var(Arooted.count)
sd1=sqrt(var1)

var2=var(Aunrooted.count)
sd2=sqrt(var2)
```

```
var3=var(Brooted.count)
sd3=sqrt(var3)

var4=var(Bunrooted.count)
sd4=sqrt(var4)

var5=var(Crooted.count)
sd5=sqrt(var5)

var6=var(Cunrooted.count)
sd6=sqrt(var6)

var7= var( Drooted.count)
sd7=sqrt(var7)

var8=var( Dunrooted.count)
sd8=sqrt(var8)

vector1=c( mean1,mean2,mean3,mean4,mean5,mean6, mean7, mean8)

vector2=c( sd1,sd2,sd3,sd4,sd5,sd6,sd7,sd8)
B=list(vector1, vector2)
return(B)

}



"buildtree8general"<- function(popsize,branchlength1, branchlength2, branchlength3,
    branchlength4,branchlength5,branchlength6,branchlength7)

#This function takes the tree with Newick notation given by ((((((A,B), C),(D,E)),F),(G,H)),
     O) and adds the indicated branchlengths. It also ensures the tree is ultrametric.
{

N=popsize
X=branchlength1
Y=branchlength2
Z=branchlength3
K=branchlength4
L=branchlength5
M=branchlength6
P=branchlength7
tree= sprintf("((((((A:%g#%g,B:%g#%g):%g#%g,␣C:%g#%g):%g#%g,(D:%g#%g,E:%g#%g):%g#%g):%g#%g,F
    :%g#%g):%g#%g,(G:%g#%g,H:%g#%g):%g#%g):%g#%g,O:%g#%g)#%g;", X, N, X,N,Y,N,X+Y,N,Z,N,X,N,
    X,N, Y+Z,N,K,N,X+Y+Z+K,N,L,N,X,N, X, N,Y+Z+K+L,N, M,N,X+Y+Z+K+L+M,N,N )
topology3<- "((((((A,B),␣C),(D,E)),F),(G,H)),␣O);"
bothtrees=c(tree, topology3)
```

```
}



"buildtree8B"<- function(popsize,branchlength1, branchlength2, branchlength3,branchlength4)
#This function takes the tree with Newick notation given by ((((A,B),(C,D)),((E,F),(G,H))),O
    ) and adds the indicated branchlengths. It also ensures the tree is ultrametric.
{

N=popsize
X=branchlength1
Y=branchlength2
Z=branchlength3
K=branchlength4
tree= sprintf("((((A:%g#%g,B:%g#%g):%g#%g,(C:%g#%g,D:%g#%g):%g#%g):%g#%g,((E:%g#%g,F:%g#%g)
    :%g#%g,(G:%g#%g,H:%g#%g):%g#%g):%g#%g):%g#%g,O:%g#%g)#%g;", X,N,X,N,Y,N,X,N,X,N,Y,N,Z,N,
    X,N,X,N,Y,N,X,N,X,N,Y,N,Z,N,K,N, X+Y+Z+K,N,N )
topology3<- "((((A,B),(C,D)),((E,F),(G,H))),O);"
bothtrees=c(tree, topology3)

}

"buildtree8C"<- function(popsize,branchlength1, branchlength2, branchlength3,branchlength4,
    branchlength5, branchlength6, branchlength7, branchlength8)
#This function takes the tree with Newick notation given by ((((((((A,B),C),D),E),F),G),H),O
    ) and adds the indicated branchlengths. It also ensures the tree is ultrametric.

{

N=popsize
X=branchlength1
Y=branchlength2
Z=branchlength3
K=branchlength4
L=branchlength5
M=branchlength6
O=branchlength7
P=branchlength8
tree= sprintf("((((((((A:%g#%g,B:%g#%g):%g#%g,C:%g#%g):%g#%g,D:%g#%g):%g#%g,E:%g#%g):%g#%g,F
    :%g#%g):%g#%g,G:%g#%g):%g#%g,H:%g#%g):%g#%g,O:%g#%g):#%g;",X,N,X,N,Y,N,X,N,Z,N ,X,N,K,N,
    X,N,L,N,X,N,M,N,X,N,O,N,X,N,P,N,X+Y+Z+K+L+M+O+P,N,N )
topology3<- "((((((((A,B),C),D),E),F),G),H),O);"
bothtrees=c(tree, topology3)

}



#Te following are the functions that plot the images displayed in the paper.

```

```
"distplotnshort"<- function(branch1, branch2 ,branch3, branch4, branch5, branch6, branch7,
    exponentshort, Nsimulations, repetitions)
{
#Requires the following packages: ape, phangorn, ggplot2, gridExtra
#branch1, branch2...branch7 are the lenghts of the internal branches
#exponentshort is  the range of sizes one wishes make in order to hae a short edge.
#We choose the species tree topology by uncommenting the tree of interest. The code
    presented here has the Balanced tree uncommented.

n=length(exponentshort)
shortbranch=2^(-exponentshort)
Arooted<-rep(NA,n)
Aunrooted<-rep(NA,n)
Brooted<-rep(NA,n)
Bunrooted<-rep(NA,n)
Crooted<-rep(NA,n)
Cunrooted<-rep(NA,n)
Drooted<-rep(NA,n)
Dunrooted<-rep(NA,n)
std1<-rep(NA, n)
std2<-rep(NA, n)
std3<-rep(NA, n)
std4<-rep(NA, n)
std5<-rep(NA, n)
std6<-rep(NA, n)
std7<-rep(NA, n)
std8<-rep(NA, n)
for (i in 1:n)
{
#1) Mixed Tree Species Tree

#sptree<-buildtree8general(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[
    i], shortbranch[i], shortbranch[i], shortbranch[i])[1]
#topology<-buildtree8general(2,  shortbranch[i],  shortbranch[i],  shortbranch[i],
    shortbranch[i], shortbranch[i], shortbranch[i], shortbranch[i])[2]

#2) Balanced Tree Species Tree

sptree<-buildtree8B(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i])[1]
topology<-buildtree8B(2, shortbranch[i],  shortbranch[i],  shortbranch[i], shortbranch[i])
    [2]

#3) Caterpilar Tree Species Tree

#sptree<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[1]
#topology<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[2]
```

```
temp1=meancountmethods.RF(sptree, topology, 1, Nsimulations, repetitions)[[1]]
temp2= meancountmethods.RF(sptree, topology, 1, Nsimulations, repetitions)[[2]]


Arooted[i]= temp1[1]
Aunrooted[i]= temp1[2]
Brooted[i]= temp1[3]
Bunrooted[i]= temp1[4]
Crooted[i]= temp1[5]
Cunrooted[i]= temp1[6]
Drooted[i]=temp1[7]
Dunrooted[i]= temp1[8]

std1[i]= temp2[1]
std2[i]= temp2[2]
std3[i]= temp2[3]
std4[i]= temp2[4]
std5[i]= temp2[5]
std6[i]= temp2[6]
std7[i]= temp2[7]
std8[i]= temp2[8]


}
#Build a data frame that will be included in the ggplot commands.
L= length(Arooted)
m=max(Arooted, Aunrooted, Brooted, Bunrooted, Crooted, Cunrooted)+1
short_branch= rep(exponentshort, 8)
rooted_values= c(Arooted, Brooted, Crooted, Drooted)
print(length(rooted_values))
rooted_methods= c("A-rooted",  "B-rooted",  "C-rooted", "D-rooted")
rooted_methods=rep(rooted_methods, each=L)
stdrooted= c( std1, std3, std5, std7)
maxrooted= rooted_values + stdrooted
minrooted= rooted_values- stdrooted
rooted_data=data.frame(short_branch, rooted_values, rooted_methods,minrooted, maxrooted)
unrooted_values= c(Aunrooted, Bunrooted, Cunrooted, Dunrooted)
unrooted_methods= c("A-unrooted",  "B-unrooted",  "C-unrooted", "D-unrooted")
unrooted_methods=rep(unrooted_methods, each=L)
stdunrooted= c(std2, std4, std6, std8)
minunrooted= unrooted_values-stdunrooted
maxunrooted= unrooted_values+stdunrooted
unrooted_data=data.frame(short_branch, unrooted_values, unrooted_methods, minunrooted,
    maxunrooted)
print(rooted_data)

#Compare Rooted Distances
rootplot=ggplot(rooted_data, aes(x=short_branch, y=rooted_values, color=rooted_methods,
    linetype=rooted_methods, ymin=minrooted, ymax=maxrooted))
rootplot= rootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
rootplot= rootplot +geom_line(size=0.9,alpha=0.5)
rootplot= rootplot + geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
    theme_bw() + xlab("n")+ ylab("Average␣RFclades␣Distance")
#Compare Unrooted Distances
unrootplot= ggplot(unrooted_data, aes(x=short_branch, y=unrooted_values, color=unrooted_
    methods, linetype=unrooted_methods, ymin=minunrooted, ymax=maxunrooted))
unrootplot= unrootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
unrootplot= unrootplot + geom_line(size=0.9,alpha=0.5)
unrootplot= unrootplot+ geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
     theme_bw()+xlab("n")+ ylab("␣Average␣RFsplit␣Distance")
grid.arrange(rootplot, unrootplot)
}





"distplotimposedA"<- function(branch1, branch2 ,branch3, branch4, branch5, branch6, branch7,
    exponentshort, Nsimulations, repetitions)
{

#This code plots the results for method A on caterpillar, balanced and mixed species trees.
#Requires the following packages: ape, phangorn, ggplot2, gridExtra
#branch1, branch2...branch7 are the lenghts of the internal branches
#exponentshort is  the range of sizes one wishes make in order to hae a short edge.

n=length(exponentshort)
shortbranch=2^(-exponentshort)
ACrooted<-rep(NA,n)
ACunrooted<-rep(NA,n)
ABrooted<-rep(NA,n)
ABunrooted<-rep(NA,n)
AMrooted<-rep(NA,n)
AMunrooted<-rep(NA,n)
std1AC<-rep(NA, n)
std2AC<-rep(NA, n)
std1AB<-rep(NA, n)
std2AB<-rep(NA, n)
std1AM<-rep(NA, n)
std2AM<-rep(NA, n)

for (i in 1:n)
{
sptreeM<-buildtree8general(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[
    i], shortbranch[i], shortbranch[i], shortbranch[i])[1]
topologyM<-buildtree8general(2,  shortbranch[i],  shortbranch[i],  shortbranch[i],
    shortbranch[i], shortbranch[i], shortbranch[i], shortbranch[i])[2]
```

```
sptreeB<-buildtree8B(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i])[1]
topologyB<-buildtree8B(2, shortbranch[i],  shortbranch[i],  shortbranch[i], shortbranch[i])
    [2]

sptreeC<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[1]
topologyC<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[2]


temp1AC=meancountmethods.RF(sptreeC, topologyC, 1, Nsimulations, repetitions)[[1]]
temp2AC=meancountmethods.RF(sptreeC, topologyC, 1, Nsimulations, repetitions)[[2]]
temp1AB=meancountmethods.RF(sptreeB, topologyB, 1, Nsimulations, repetitions)[[1]]
temp2AB=meancountmethods.RF(sptreeB, topologyB, 1, Nsimulations, repetitions)[[2]]
temp1AM=meancountmethods.RF(sptreeM, topologyM, 1, Nsimulations, repetitions)[[1]]
temp2AM=meancountmethods.RF(sptreeM, topologyM, 1, Nsimulations, repetitions)[[2]]


ACrooted[i]= temp1AC[1]
ACunrooted[i]= temp1AC[2]
ABrooted[i]= temp1AB[1]
ABunrooted[i]= temp1AB[2]
AMrooted[i]= temp1AM[1]
AMunrooted[i]= temp1AM[2]

std1AC[i]= temp2AC[1]
std2AC[i]= temp2AC[2]
std1AB[i]= temp2AB[1]
std2AB[i]= temp2AB[2]
std1AM[i]= temp2AM[1]
std2AM[i]= temp2AM[2]



}
L= length(ACrooted)
short_branch= rep(exponentshort, 6)
rooted_values= c(ACrooted, ABrooted, AMrooted)
rooted_methods= c("ACaterpillar-rooted",  "ABalanced-rooted",  "AMixed-rooted")
rooted_methods=rep(rooted_methods, each=L)
stdrooted= c( std1AC, std1AB, std1AM)
maxrooted= rooted_values + stdrooted
minrooted= rooted_values- stdrooted
rooted_data=data.frame(short_branch, rooted_values, rooted_methods,minrooted, maxrooted)
unrooted_values= c(ACunrooted, ABunrooted, AMunrooted)
unrooted_methods= c("ACaterpillar-unrooted",  "ABalanced-unrooted",  "AMixed-unrooted")
unrooted_methods=rep(unrooted_methods, each=L)
stdunrooted= c(std2AC, std2AB, std2AM)
minunrooted= unrooted_values-stdunrooted
```

```
maxunrooted= unrooted_values+stdunrooted
unrooted_data=data.frame(short_branch, unrooted_values, unrooted_methods, minunrooted,
    maxunrooted)
#print(rooted_data)

#Compare Rooted Distances
rootplot=ggplot(rooted_data, aes(x=short_branch, y=rooted_values, color=rooted_methods,
    linetype=rooted_methods, ymin=minrooted, ymax=maxrooted))
rootplot= rootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
rootplot= rootplot +geom_line(size=0.9,alpha=0.5)
rootplot= rootplot + geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
    theme_bw() + xlab("n")+ ylab("Average␣RFclades␣Distance")
#Compare Unrooted Distances
unrootplot= ggplot(unrooted_data, aes(x=short_branch, y=unrooted_values, color=unrooted_
    methods, linetype=unrooted_methods, ymin=minunrooted, ymax=maxunrooted))
unrootplot= unrootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
unrootplot= unrootplot + geom_line(size=0.9,alpha=0.5)
unrootplot= unrootplot+ geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
     theme_bw()+xlab("n")+ ylab("␣Average␣RFsplit␣Distance")
grid.arrange(rootplot, unrootplot)
}



"distplotimposedB"<- function(branch1, branch2 ,branch3, branch4, branch5, branch6, branch7,
    exponentshort, Nsimulations, repetitions)
{

#This code plots the results for method B on caterpillar, balanced and mixed species trees.
#Requires the following packages: ape, phangorn, ggplot2, gridExtra
#branch1, branch2...branch7 are the lenghts of the internal branches
#exponentshort is  the range of sizes one wishes make in order to hae a short edge.

n=length(exponentshort)
shortbranch=2^(-exponentshort)
BCrooted<-rep(NA,n)
BCunrooted<-rep(NA,n)
BBrooted<-rep(NA,n)
BBunrooted<-rep(NA,n)
BMrooted<-rep(NA,n)
BMunrooted<-rep(NA,n)
std1BC<-rep(NA, n)
std2BC<-rep(NA, n)
std1BB<-rep(NA, n)
std2BB<-rep(NA, n)
std1BM<-rep(NA, n)
std2BM<-rep(NA, n)

for (i in 1:n)
```

```
{
sptreeM<-buildtree8general(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[
    i], shortbranch[i], shortbranch[i], shortbranch[i])[1]
topologyM<-buildtree8general(2,  shortbranch[i],  shortbranch[i],  shortbranch[i],
    shortbranch[i], shortbranch[i], shortbranch[i], shortbranch[i])[2]

sptreeB<-buildtree8B(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i])[1]
topologyB<-buildtree8B(2, shortbranch[i],  shortbranch[i],  shortbranch[i], shortbranch[i])
    [2]

sptreeC<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[1]
topologyC<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[2]



temp1BC=meancountmethods.RF(sptreeC, topologyC, 1, Nsimulations, repetitions)[[1]]
temp2BC=meancountmethods.RF(sptreeC, topologyC, 1, Nsimulations, repetitions)[[2]]
temp1BB=meancountmethods.RF(sptreeB, topologyB, 1, Nsimulations, repetitions)[[1]]
temp2BB=meancountmethods.RF(sptreeB, topologyB, 1, Nsimulations, repetitions)[[2]]
temp1BM=meancountmethods.RF(sptreeM, topologyM, 1, Nsimulations, repetitions)[[1]]
temp2BM=meancountmethods.RF(sptreeM, topologyM, 1, Nsimulations, repetitions)[[2]]



BCrooted[i]= temp1BC[3]
BCunrooted[i]= temp1BC[4]
BBrooted[i]= temp1BB[3]
BBunrooted[i]= temp1BB[4]
BMrooted[i]= temp1BM[3]
BMunrooted[i]= temp1BM[4]

std1BC[i]= temp2BC[3]
std2BC[i]= temp2BC[4]
std1BB[i]= temp2BB[3]
std2BB[i]= temp2BB[4]
std1BM[i]= temp2BM[3]
std2BM[i]= temp2BM[4]




}
L= length(BCrooted)
short_branch= rep(exponentshort, 6)
rooted_values= c(BCrooted, BBrooted, BMrooted)
rooted_methods= c("BCaterpillar-rooted",  "BBalanced-rooted",  "BMixed-rooted")
rooted_methods=rep(rooted_methods, each=L)
stdrooted= c( std1BC, std1BB, std1BM)
maxrooted= rooted_values + stdrooted
minrooted= rooted_values- stdrooted
```

```
rooted_data=data.frame(short_branch, rooted_values, rooted_methods,minrooted, maxrooted)
unrooted_values= c(BCunrooted, BBunrooted, BMunrooted)
unrooted_methods= c("BCaterpillar-unrooted", "BBalanced-unrooted", "BMixed-unrooted")
unrooted_methods=rep(unrooted_methods, each=L)
stdunrooted= c(std2BC, std2BB, std2BM)
minunrooted= unrooted_values-stdunrooted
maxunrooted= unrooted_values+stdunrooted
unrooted_data=data.frame(short_branch, unrooted_values, unrooted_methods, minunrooted,
    maxunrooted)
#print(rooted_data)

#Compare Rooted Distances
rootplot=ggplot(rooted_data, aes(x=short_branch, y=rooted_values, color=rooted_methods,
    linetype=rooted_methods, ymin=minrooted, ymax=maxrooted))
rootplot= rootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
rootplot= rootplot +geom_line(size=0.9,alpha=0.5)
rootplot= rootplot + geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
    theme_bw() + xlab("n")+ ylab("Average␣RFclades␣Distance")
#Compare Unrooted Distances
unrootplot= ggplot(unrooted_data, aes(x=short_branch, y=unrooted_values, color=unrooted_
    methods, linetype=unrooted_methods, ymin=minunrooted, ymax=maxunrooted))
unrootplot= unrootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
unrootplot= unrootplot + geom_line(size=0.9,alpha=0.5)
unrootplot= unrootplot+ geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
     theme_bw()+xlab("n")+ ylab("␣Average␣RFsplit␣Distance")
grid.arrange(rootplot, unrootplot)
}






"distplotimposedC"<- function(branch1, branch2 ,branch3, branch4, branch5, branch6, branch7,
    exponentshort, Nsimulations, repetitions)
{

#This code plots the results for method C on caterpillar, balanced and mixed species trees.
#Requires the following packages: ape, phangorn, ggplot2, gridExtra
#branch1, branch2...branch7 are the lenghts of the internal branches
#exponentshort is  the range of sizes one wishes make in order to hae a short edge.

n=length(exponentshort)
shortbranch=2^(-exponentshort)
CCrooted<-rep(NA,n)
CCunrooted<-rep(NA,n)
CBrooted<-rep(NA,n)
CBunrooted<-rep(NA,n)
CMrooted<-rep(NA,n)
```

```
CMunrooted<-rep(NA,n)
std1CC<-rep(NA, n)
std2CC<-rep(NA, n)
std1CB<-rep(NA, n)
std2CB<-rep(NA, n)
std1CM<-rep(NA, n)
std2CM<-rep(NA, n)

for (i in 1:n)
{
sptreeM<-buildtree8general(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[
    i], shortbranch[i], shortbranch[i], shortbranch[i])[1]
topologyM<-buildtree8general(2,  shortbranch[i],  shortbranch[i],  shortbranch[i],
    shortbranch[i], shortbranch[i], shortbranch[i], shortbranch[i])[2]

sptreeB<-buildtree8B(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i])[1]
topologyB<-buildtree8B(2, shortbranch[i],  shortbranch[i],  shortbranch[i], shortbranch[i])
    [2]

sptreeC<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[1]
topologyC<-buildtree8C(2,  shortbranch[i], shortbranch[i],  shortbranch[i], shortbranch[i],
    shortbranch[i],shortbranch[i], shortbranch[i], shortbranch[i])[2]


temp1CC=meancountmethods.RF(sptreeC, topologyC, 1, Nsimulations, repetitions)[[1]]
temp2CC=meancountmethods.RF(sptreeC, topologyC, 1, Nsimulations, repetitions)[[2]]
temp1CB=meancountmethods.RF(sptreeB, topologyB, 1, Nsimulations, repetitions)[[1]]
temp2CB=meancountmethods.RF(sptreeB, topologyB, 1, Nsimulations, repetitions)[[2]]
temp1CM=meancountmethods.RF(sptreeM, topologyM, 1, Nsimulations, repetitions)[[1]]
temp2CM=meancountmethods.RF(sptreeM, topologyM, 1, Nsimulations, repetitions)[[2]]


CCrooted[i]= temp1CC[5]
CCunrooted[i]= temp1CC[6]
CBrooted[i]= temp1CB[5]
CBunrooted[i]= temp1CB[6]
CMrooted[i]= temp1CM[5]
CMunrooted[i]= temp1CM[6]

std1CC[i]= temp2CC[5]
std2CC[i]= temp2CC[6]
std1CB[i]= temp2CB[4]
std2CB[i]= temp2CB[6]
std1CM[i]= temp2CM[5]
std2CM[i]= temp2CM[6]
}
L= length(CCrooted)
short_branch= rep(exponentshort, 6)
rooted_values= c(CCrooted, CBrooted, CMrooted)
rooted_methods= c("CCaterpillar-rooted",  "CBalanced-rooted",  "CMixed-rooted")
rooted_methods=rep(rooted_methods, each=L)
stdrooted= c( std1CC, std1CB, std1CM)
maxrooted= rooted_values + stdrooted
minrooted= rooted_values- stdrooted
rooted_data=data.frame(short_branch, rooted_values, rooted_methods,minrooted, maxrooted)
unrooted_values= c(CCunrooted, CBunrooted, CMunrooted)
unrooted_methods= c("CCaterpillar-unrooted",  "CBalanced-unrooted",  "CMixed-unrooted")
unrooted_methods=rep(unrooted_methods, each=L)
stdunrooted= c(std2CC, std2CB, std2CM)
minunrooted= unrooted_values-stdunrooted
maxunrooted= unrooted_values+stdunrooted
unrooted_data=data.frame(short_branch, unrooted_values, unrooted_methods, minunrooted,
    maxunrooted)
#print(rooted_data)

#Compare Rooted Distances
rootplot=ggplot(rooted_data, aes(x=short_branch, y=rooted_values, color=rooted_methods,
    linetype=rooted_methods, ymin=minrooted, ymax=maxrooted))
rootplot= rootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
rootplot= rootplot +geom_line(size=0.9,alpha=0.5)
rootplot= rootplot + geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
    theme_bw() + xlab("n")+ ylab("Average␣RFclades␣Distance")
#Compare Unrooted Distances
unrootplot= ggplot(unrooted_data, aes(x=short_branch, y=unrooted_values, color=unrooted_
    methods, linetype=unrooted_methods, ymin=minunrooted, ymax=maxunrooted))
unrootplot= unrootplot +geom_point(size=5, alpha=0.3, position=position_dodge(width=0.5))
unrootplot= unrootplot + geom_line(size=0.9,alpha=0.5)
unrootplot= unrootplot+ geom_errorbar(size=1,alpha=0.5, position=position_dodge(width=0.1))+
     theme_bw()+xlab("n")+ ylab("␣Average␣RFsplit␣Distance")
grid.arrange(rootplot, unrootplot)
}
```

References

[ADR13] E. S. Allman, J. H. Degnan, and J. A. Rhodes. Species tree inference by the STAR method, and generalizations. *J. Comput. Biol.*, 20(1):50–61, 2013.

[ADR16] E. S. Allman, J. H. Degnan, and J. A. Rhodes. Species tree inference from gene splits by unrooted star methods. 2016.

[ape15] *ape: Analyses of Phylogenetics and Evolution*, 2015.

[LYPE09] L. Liu, L. Yu, D. K. Pearl, and S. V. Edwards. Estimating species phylogenies using coalescence times among sequences. *Syst. Biol.*, 58:468–477, 2009.

[pha16] *phangorn: Phylogenetic Analysis in R*, 2016.

[phy14] *phybase: Basic Functions for Phylogenetic Analysis*, 2014.

[SK88] J.A. Studier and K.J. Keppler. A note on the neighbor-joining algorithm of Saitou and Nei. *Mol.Biol.Evol*, 6:729–731, 1988.