

EÖTVÖS LORÁND UNIVERSITY
FACULTY OF SCIENCE

DÁNIEL HEGEDŰS, MÁRK HUNOR JUHÁSZ

Institute of Mathematics

Methods for pruning phylogenetic trees

37th OTDK Biology Section

Bioinformatics

Supervisor: ESZTER ARI

Institute of Biology, Department of Genetics



Budapest, 2025

Table of Contents

1	Introduction	3
2	Methods	4
2.1	Research goal and key definitions	4
2.2	The TreeShrink algorithm	5
2.3	The IQR algorithm	6
2.4	The Primitive Straight-Forward Approach (PSFA)	7
2.5	The Circular Pruning Algorithm (CPA)	8
2.6	Combined algorithms	10
3	Results and Discussion	11
	References	15

1 Introduction

Phylogenetic trees are fundamental tools in biology, used to describe the evolutionary relationships among various taxons (or Operational Taxonomic Units, OTUs) based on their genetic characteristics. Accurate phylogenetic trees are essential for understanding evolutionary pathways, speciation events, and the distribution of genetic traits across taxa. However, the inference of these trees is fraught with challenges, including the presence of errors due to low sequencing or assembly quality, contamination, and misclassification of taxa [1]. Such inaccuracies can significantly distort the evolutionary inferences drawn from these trees, making robust error detection and correction mechanisms necessary.

One promising approach to enhance the accuracy of phylogenetic trees is through pruning tips that are very distant from the majority of tips, *i.e.* having an extremely long branch. By pruning, we can remove potentially erroneous taxa from the tree. Pruning can help to mitigate the impact of errors by simplifying the tree and eliminating tips on inconsistently long branches. This approach was implemented in the TreeShrink software [1], which relies on identifying leaves (also named as tips or end nodes of the phylogenetic tree, representing the taxa or the OTUs) that are dramatically further away from the center or the root than most of the others. The existence of such leaves would indicate that a small minority of taxa have dramatically different rates of evolution. By removing these problematic leaves and the corresponding branches, the pruned tree may better reflect the true evolutionary relationships among the remaining taxa.

In this study, we introduce new algorithms for pruning phylogenetic trees. This is needed, as the algorithm described in the above-mentioned paper [1], which is basically the only published available algorithm, does not always return the optimal solution, see Figure 1. and Section 3.

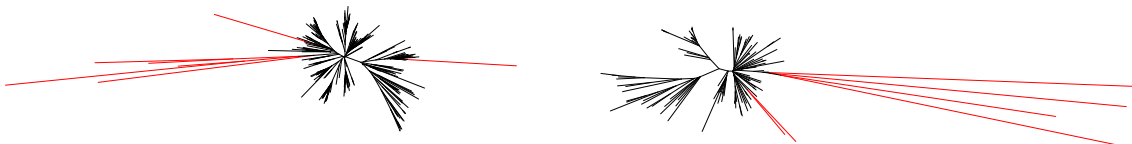


Figure 1. Two examples of phylogenetic trees. In the case of these trees, TreeShrink does not prune anything. The extremely long branches are indicated with red. These should have been pruned in our opinion.

The findings of this research have the potential to significantly improve the reliability of phylogenetic trees, providing a more accurate framework for evolutionary studies.

As the nomenclature is not always clear in the literature, we want to state the definitions we will use in this thesis. The uncleanness of the definitions is understandable, as both the mathematical (graph-theoretical) [2] and biological [3] terminology are often used. In graph-theory, a graph consists of vertices (nodes) and edges: each edge connects two different vertices. The degree of a vertex is the number of edges it has. A treegraph, or shortly a tree, is a connected graph, which has n vertices and $n - 1$ edges (both in the bifurcating and in the multifurcating case). A vertex is a leaf of a tree if its degree is 1. Note, that the evolutionary concept of a phylogenetic tree is the same: the tree consists of nodes and tips connected by branches (or edges). Note that a leaf/tip is also a vertex/node. It is easy to see that for any pair of nodes, there exists a unique path of edges connecting them.

It is important to note, that in phylogenetic trees, the edges usually have lengths (mathematically weights), which hold information about the genetic difference of its endpoints. A tree can be rooted, in which the root is the most recent common ancestor of the leaves. The root-to-tip distance is the sum of the lengths of the edges on the unique path connecting the root and the specified leaf. In the case of unrooted, undirected trees, we do not have information about the common ancestor of the investigated taxa.

In this thesis, we list four different algorithms for pruning phylogenetic trees: the TreeShrink, the IQR algorithm by Eszter Ari, a new Primitive Straight-Forward Approach (PSFA), and the new intuition-based complex algorithm, called the Circular Pruning Algorithm (CPA). Besides describing them, we compare their effectiveness using different approaches.

2 Methods

2.1 Research goal and key definitions

Throughout this paper the input of the algorithms is always an unrooted phylogenetic tree given in Newick-format [4]. The aim of this research is to improve the existing algorithm for tree pruning. Since we have not properly defined our goal yet, we define it now before moving on to the algorithms. In the above-mentioned paper [1], the goal was to reduce the diameter as much as possible while only removing a certain proportion of leaves. (Unfortunately, this certain number cannot be changed as a user in the case of

TreeShrink, but we want to provide an algorithm, in which the user can choose their best parameters.) We aim to do something similar: we want to remove the least amount of leaves to achieve such a tree, where reducing the radius of the tree significantly is only possible by removing a lot of other leaves. Throughout this paper we use the following definitions.

Definition. The diameter of a phylogenetic tree is defined as the longest path between any two leaves in the tree. Formally, if T is a tree with a set of leaves L , then the diameter $d(T)$ is given by the formula

$$d(T) = \max_{\ell_i, \ell_j \in L} d(\ell_i, \ell_j), \quad (1)$$

where $d(\ell_i, \ell_j)$ represents the distance between leaves ℓ_i and ℓ_j .

Definition. The radius of a rooted phylogenetic tree is defined as the longest path connecting the root of the tree and a leaf. Formally, if T is a rooted tree with root vertex v and a set of leaves L , then the radius $r(T)$ is given by the formula

$$r(T) = \max_{\ell \in L} d(v, \ell), \quad (2)$$

where $d(v, \ell)$ represents the distance between the root vertex v and a leaf ℓ . Note that $d(v, \ell)$ is also called root-to-tip distance in evolutionary biology. For any two nodes u, v , the distance $d(u, v)$ is measured as the sum of the branch lengths (or evolutionary distances) along the unique path from u to v .

2.2 The TreeShrink algorithm

First, review the k -shrink problem: "Given a tree on n leaves with branch lengths and a number $1 \leq k \leq n$; for every $1 \leq i \leq k$, find a set of i leaves that should be removed to reduce the tree diameter maximally" [1]. The TreeShrink is an algorithm that solves this problem in $O(k^2r + n)$ time, where r is the radius of the tree after being rerooted at the centroid edge. Note that this algorithm makes extra, unnecessary steps if there is an extremely long branch with more leaves, since in this case, the algorithm cuts every leaf from that branch one by one in many steps, instead of simply removing the long branch with all the leaves on it. Such a tree is shown in Figure 2. We also have to mention that the TreeShrink software fails to prune any of the branches of the tree shown in Figure 2.



Figure 2. A phylogenetic tree with a long branch with leaves on both ends.

2.3 The IQR algorithm

The IQR algorithm was produced in 2023 by Eszter Ari [5]. First, we describe two definitions, which are used in the algorithm.

Definition. Let H be a vector of n numbers, ordered from smallest to largest. The lower quartile (or first quartile) of the data is the number which is the $\frac{n}{4}$ -th in the vector, meaning that 25% of the data is smaller than the number. We can similarly define the median (or second quartile), and the upper quartile (or third quartile). They are denoted by $Q1, Q2$ and $Q3$ respectively. It should be noted, that if there is no number precisely at the 25%, then we take the average of the two adjacent numbers. The interquartile range of the vector is $Q3 - Q1$, denoted by IQR . The upper fence of the interquartile range is $Q3 + 3(Q3 - Q1)$. The multiplier 3 can be changed to other constants. The interquartile range is discussed for example in the paper of F.M. Dekking [6].

Definition. A midpoint rooted tree is a rooted tree, which is rooted from the midpoint of the farthest two leaves. In other words, this can be seen as rooting from the midpoint of a diameter.

The algorithm includes two steps. The first step prunes the tree based on only the upper fence of the IQR of branch lengths. The second step iteratively midpoint roots the tree and removes leaves based on the IQR of root-to-tip distances.

The first step starts with a technical calculation. For each branch, it finds the direction with the fewer leaves. Then it calculates the IQR of branch lengths and collects the extreme outlier branches. If there are any outlier branches with less than 5% of leaves from one of its endpoints, the algorithm removes it from the tree with the clade which contains few leaves.

The second step iteratively makes the following. It midpoint roots the tree, then removes the most extreme outlier leaf based on the IQR of root-to-tip distances. If there

are no leaves with root-to-tip distance greater than the upper fence, then the algorithm terminates and unroots the tree.

Intuitively, the algorithm first instantly prunes those long branches, which have only a very small proportion of leaves on one of its clades. Then it recursively prunes leaves until there are no extremely large root-to-tip distances. This algorithm intuitively fulfils the above-described goal, as the pruned tree does not have leaves with exceptional midpoint-to-tip distances.

2.4 The Primitive Straight-Forward Approach (PSFA)

The phenomenon on Figure 2. led us to a primitive and intuitive pruning algorithm, which we will describe in this section. The input of this algorithm is an unrooted phylogenetic tree, the number of leaves on the (original) tree and a threshold, which specifies the maximum percentage of leaves that can be cut. (At this point, the number of leaves of the original tree could be calculated, but when combining the algorithms, it will be important to use the parameters calculated from the original tree.)

First, the algorithm calculates the average branch length of the tree and selects the longest branch. These actions can be done in linear time. If the average branch length is much shorter than the longest branch, then the algorithm removes this branch from the tree, otherwise, it terminates. But we cannot simply delete a branch, because if we remove a branch from the middle of the tree, it will split into two subtrees, thus it is not going to be a tree anymore. To solve this problem, we have to delete one of the subtrees, but only if we do not exceed the threshold by doing so. The algorithm checks if this condition is satisfied, and if we do not exceed the threshold by deleting one of the subtrees, then we delete it, but if we exceed it, the algorithm terminates. Then we repeat this process until the algorithm terminates, meaning that there are no more extremely long branches left, or we cannot delete such a branch without exceeding the threshold.

Note that there are loosely defined steps in the algorithm, for example, we did not define "much shorter", and we did not say anything about the threshold. This is because the algorithm (like most of the others) is intuitive, and there are no specific values for these ratios that work all the time. Regardless, we have to use some ratios in order to run the algorithm, so in these cases, we use the ratios that have the highest success rate when we run the algorithm on phylogenetic trees (which were calculated for an independent

research project by other researchers). We discuss these success rates and the testing of the algorithms in Section 3. For the threshold we used 10%, which means that the algorithm keeps at least 90% of the leaves. By increasing the threshold, one can achieve better results on some tests (for example, on the R^2 -test, see in Section 3), but it will result in the loss of a lot of information. For the other ratio (average/longest), we used $1/9$, meaning that the algorithm deletes only if the longest branch is at least 9 times longer than the average. (We could use median instead of average, but usually it does not change anything on the result.) One may adjust these ratios to find the one that works the best for their tree. It is easy to see that the runtime of the algorithm is $O(n^2)$ in the worst case.

In some cases, this simple method provides a good solution, but note that the existence of an extremely long branch is not necessary, even if we have a leaf very far from the root, as shown on the left part of Figure 3.

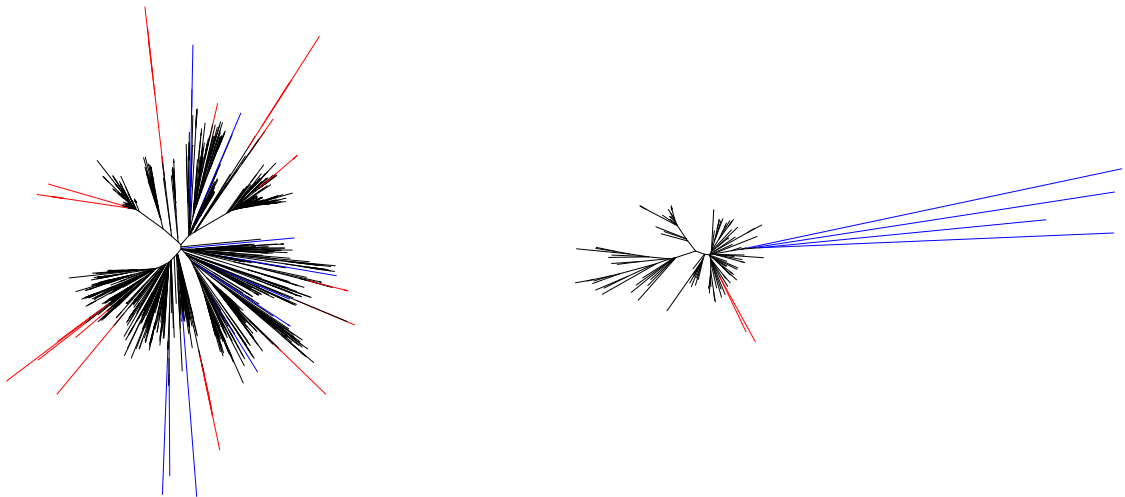


Figure 3. Two examples of a phylogenetic trees pruned by the PSFA. Blue indicates the edges pruned by the PSFA and red indicates the edges we think also should have been pruned. Note, that the tree on the right is the same as the tree on the right part of Figure 1.

2.5 The Circular Pruning Algorithm (CPA)

The Circular Pruning Algorithm was produced to prune the tree into a "circular" shape, as we wanted. First, we will present the underlying intuition that drives this algorithm, then we will go into the details and describe every step precisely.

Consider the tree as if it were a physical object, where the edges (branches) are sticks with some mass, and the nodes are mechanical hinge joints. Now imagine that we hang

it by one of its nodes, meaning that we grab one node, and let the gravity pull down the rest of the tree. If the tree was originally "circular" and we hung it by the center node, then now there are a lot of leaves right at the bottom. This is because these leaves were originally on the perimeter of the circle, so the distances between them and the root were approximately the same. If the original tree was not in a "circular" shape, then we want to prune it to be nearly circular. This means that we have to cut down the bottom of the tree, so that the remaining hanging tree has a lot of leaves right at the bottom. Another criterion was that we could not reduce the radius significantly without deleting a lot of leaves from the pruned tree. We will deal with both criteria in our algorithm. Now we present our algorithm in detail, but first, we have to define hanging precisely.

Definition. Let T be a phylogenetic tree and let v be one of its vertices. The hanging of T by v is calculating the distances from v to all leaves ℓ_i within the tree. In this context, let us call v the hanging point of T .

By the intuitive description, it is clear that the pruned tree highly depends on the hanging point. Throughout the algorithm, we use the root of the tree as the hanging point, but since we are working with unrooted trees, we have to choose the root very carefully. In fact, instead of choosing only one root, we pick different roots randomly and use the one which gives the best result. (The measurement of goodness will be discussed later.) The number of randomly chosen roots is predetermined, we used 10% of the non-leaf nodes, but always at least 15 nodes. In addition to these randomly chosen roots, we also hang the tree by its midpoint, since midpoint rooting gives good results in most of the cases.

Once we have these roots, we iterate through them and perform the following steps on every root one by one. First, we hang the tree from the root, let us call it v , and store the distance of each leaf from v (root-to-tip distance). Then we create $M(n)$ equally long intervals $I_1, \dots, I_{M(n)}$, which do not overlap and cover the whole range of the dataset (where n is the number of leaves in the tree, and $M(n)$ is a function of n). Precisely speaking, we partition the range of the dataset into a disjoint union of equally long intervals. Now we create $M(n)$ brackets, so that every bracket corresponds to exactly one interval and put every leaf into exactly one bracket, based on its distance from the root. A leaf ℓ goes into the bracket B_{I_j} , if and only if $d(v, \ell) \in I_j$, where B_{I_j} corresponds to the interval I_j . Once we put every leaf into the brackets, we can calculate the distribution

of the leaves among the brackets. We will prune the tree by deleting the leaves in the last couple brackets, so we have to find an index k and delete every leaf which is in a bracket with an index greater than k . This will be exactly the same thing as we did in the intuition, where we cut down the bottom parts. Similarly as in the first Primitive Straight-Forward Approach, deciding where to cut and where to stop is a difficult task, and there is no perfect ratio for this that works all the time. So first, we satisfy our two conditions of cutting and then we set the ratio that results in the best pruned tree. In the case of defining $M(n)$, we proceed similarly. To satisfy the first condition (not losing too much information), the algorithm finds the threshold index j , where the first j brackets together contain $\alpha\%$ of the leaves, but the first $j - 1$ does not. To satisfy the second condition (to not be able to reduce the radius significantly without deleting a lot of leaves), the algorithm will delete everything after the bracket B_{I_k} only if there are at least $\beta\%$ less leaves in the bracket $B_{I_{k+1}}$ than in the bracket B_{I_k} . Similarly to $M(n)$ and α , we will set the value of β during the testing. For the first attempts, we used $\alpha = 90$, $\beta = 25$ and $M(n) = n$.

The algorithm now finds the smallest index p , which satisfies both conditions, and prunes the tree in the above mentioned way. In order to decide which root is the best, we have to come up with a measurement of goodness. For this purpose, we will use a variable p_v , such that if we calculate the p value for all the roots v , the best root will be the one with the smallest $p_v = p$ value. We use p as a measure of goodness for intuitive reasons. We use the same number of intervals (brackets) for every root, so if p_{v^*} is the smallest among the p_v -s, it means that we achieve the smallest radius (while satisfying the conditions) when we prune the tree using the root v^* . With this, our algorithm unroots the tree and returns the pruned tree. It is easy to see that the runtime of the algorithm is $O(n^2 \log(n))$.

2.6 Combined algorithms

Now we will present some algorithms we created by combining the Primitive Straight-Forward Approach with the Circular Pruning Algorithm, focusing on the underlying intuition. The algorithms are implemented in Python and can be checked out on [github](#). To make working with phylogenetic trees easier, we used the ete3 Python package [7].

First, one can notice, that the Circular Pruning Algorithm does not guarantee that

the resulting tree will not have any branches that are much longer than the average. Indeed, suppose we have a tree where the root-to-tip distance of every leaf is r , but every root-to-tip path contains one node in the middle. Now, if we put an r long branch on the root, then that branch is two times longer than the average, but the CPA still will not cut it down. This scenario is not rare, so one may want to prune the tree with the PSFA after pruning it with the Circular Pruning Algorithm. We will see in Section 3 that this improves the pruning in some cases. For this, the first algorithm must keep track of how many more leaves the second algorithm can cut after its execution, in order not to exceed the predetermined limit.

The reader may question why we consider the visibly outlier sections during the optimization process of the Circular Pruning Algorithm, given that we intend to remove them regardless. It is also possible that accounting for these outlier sections may negatively impact the runtime. Although optimizing for runtime is not our primary goal, but as it will be evident during testing, the algorithm is also practically fast. This led us to another intuitive solution: run the PSFA before and after the Circular Pruning Algorithm (running it after the other algorithm is not necessary, we simply combined the two observations). The algorithms have to keep track of the number of removable leaves as before to make sure we do not exceed the limit. As we will see in Section 3, this algorithm also provides great results.

3 Results and Discussion

Tests were run on the 25 trees from [8]. In this study, the authors constructed 25 phylogenetic trees corresponding to the 25 primary clades, referred as sequence types (STs), of *Escherichia coli*. After downloading ~ 9000 *E. coli* genomes, they utilized the MLST software [9] to determine the STs. Recombinant genomic regions were subsequently excluded, and phylogenetic trees were inferred using the maximum likelihood method. When visually investigating the reconstructed trees, we see some moderately or even extremely long branches that should be pruned. The trees shown in Figure 1., 2., and 3. are all belonging to the above study, ST 38 and ST 156, ST 655, ST 10 and ST 156 correspondingly.

We measured the goodness of the algorithms by calculating the R^2 values of the root-to-tip regressions of the trees with dated tips. We know the sampling date for $\sim 90\%$ of

the genomes used in the above study. First, we rooted the trees using the given dates of the tips (sample collection dates) [10], and then we checked how much the dates align with the root-to-tip distances in the case of the original tree and the trees pruned with various methods. We say that the algorithm works well if the R^2 value of the pruned tree was increased compared to the R^2 value of the original tree. Note that one can achieve R^2 values arbitrarily close to 1, if they delete a lot of branches. Indeed, if we only leave one node on the tree, the R^2 value will be exactly 1. But we aim to achieve good results without removing too many branches from the tree, that is we want to leave at least 90% of the tips untouched. The algorithm that calculated the R^2 values based on the dated tips was given to us by our supervisor. The second and third row of Figure 4. were also produced by this algorithm.

After the testing, we came to the conclusion that the following parameters work the best (for our trees). In the case of the CPA we set $M(n) = n$, $\alpha = 90$, $\beta = 20$. When we ran the PSFA after the CPA, we set $\alpha = 95$, $\beta = 20$ and let the PSFA delete at least 5% of the leaves. When we ran the PSFA both before and after the CPA, we let the first PSFA to prune at most 5%, then the CPA at most another 5% (more precisely we set $\alpha = 95$ and $\beta = 20$), then the final pruning can delete the rest of the removable 10%.

Our results are summarized in Table 1. Note that the table does not contain all 25 trees, since we omitted the trees with too many leaves. In the case of those trees, the R^2 values do not carry any significant information, since those values will always be very small, and even if we pruned the tree randomly, the results would improve. The main conclusion from the table is that the best pruning algorithms are the IQR and the CPA. However, we should also take into account that we used only 10% of the nodes (but at least 15) as roots in the case of the CPA to optimize runtime. One can achieve better results with the CPA by setting this percentage higher. We also show an example for the statistical analysis of a tree in Figure 4.

BactDating results for some trees

Algorithm \ ST		16	32	38	69	101	117	127	155	156	167	405	410	648	2332
Unpruned	R^2	0.11	0.09	0.02	0.21	0.20	0.18	0.12	0.06	0.49	0.18	0.18	0.51	0.22	0.07
	tips %	100	100	100	100	100	100	100	100	100	100	100	100	100	100
TreeShrink	R^2	0.11	0.09	0.02	0.31	0.28	0.19	0.12	0.12	0.49	0.18	0.29	0.51	0.29	0.07
	tips %	100	100	100	99.5	99.1	99.5	100	98.6	100	100	99.5	100	99.4	100
IQR	R^2	0.24	0.33	0.29	0.35	0.29	0.20	0.12	0.13	0.60	0.62	0.32	0.74	0.09	0.56
	tips %	96	94.3	96.1	96.3	95.4	97	98.3	98.2	94.7	90	90.2	90.8	93.8	89.4
PSFA	R^2	0.22	0.29	0.23	0.24	0.24	0.18	0.11	0.11	0.49	0.47	0.21	0.75	0.09	0.55
	tips %	96.7	96.1	97.7	99.8	99.5	100	98.9	98.6	96.5	95.3	95.1	95.4	96.9	90.4
CPA	R^2	0.20	0.29	0.27	0.47	0.23	0.42	0.40	0.20	0.61	0.52	0.22	0.76	0.33	0.54
	tips %	95.3	95.6	96.5	91.8	92.6	90.5	90.3	91.7	93.8	90.0	90.6	90.4	90.1	91.3
CPA+PSFA	R^2	0.20	0.29	0.27	0.40	0.24	0.20	0.11	0.17	0.58	0.47	0.21	0.72	0.12	0.55
	tips %	96.0	95.2	96.5	95.1	99.1	95.0	96.0	95.8	95.6	94.0	96.1	95.0	93.8	90.4
PSFA+CPA +PSFA	R^2	0.22	0.29	0.28	0.36	0.25	0.28	0.11	0.18	0.69	0.51	0.22	0.81	0.12	0.55
	tips %	92.0	91.3	93.1	95.1	96.3	95.0	94.3	94.4	92.0	90.0	93.6	91.2	93.8	90.4

Table 1. Each row corresponds to an algorithm and each column corresponds to a tree. We listed the R^2 values and also the percentages of the remaining tips compared to the original number of tips. One can see that TreeShrink does not always prune many tips, even though a large increase in the R^2 value is possible. Darker green indicates the highest values, while lighter green indicates the second highest values in each column.

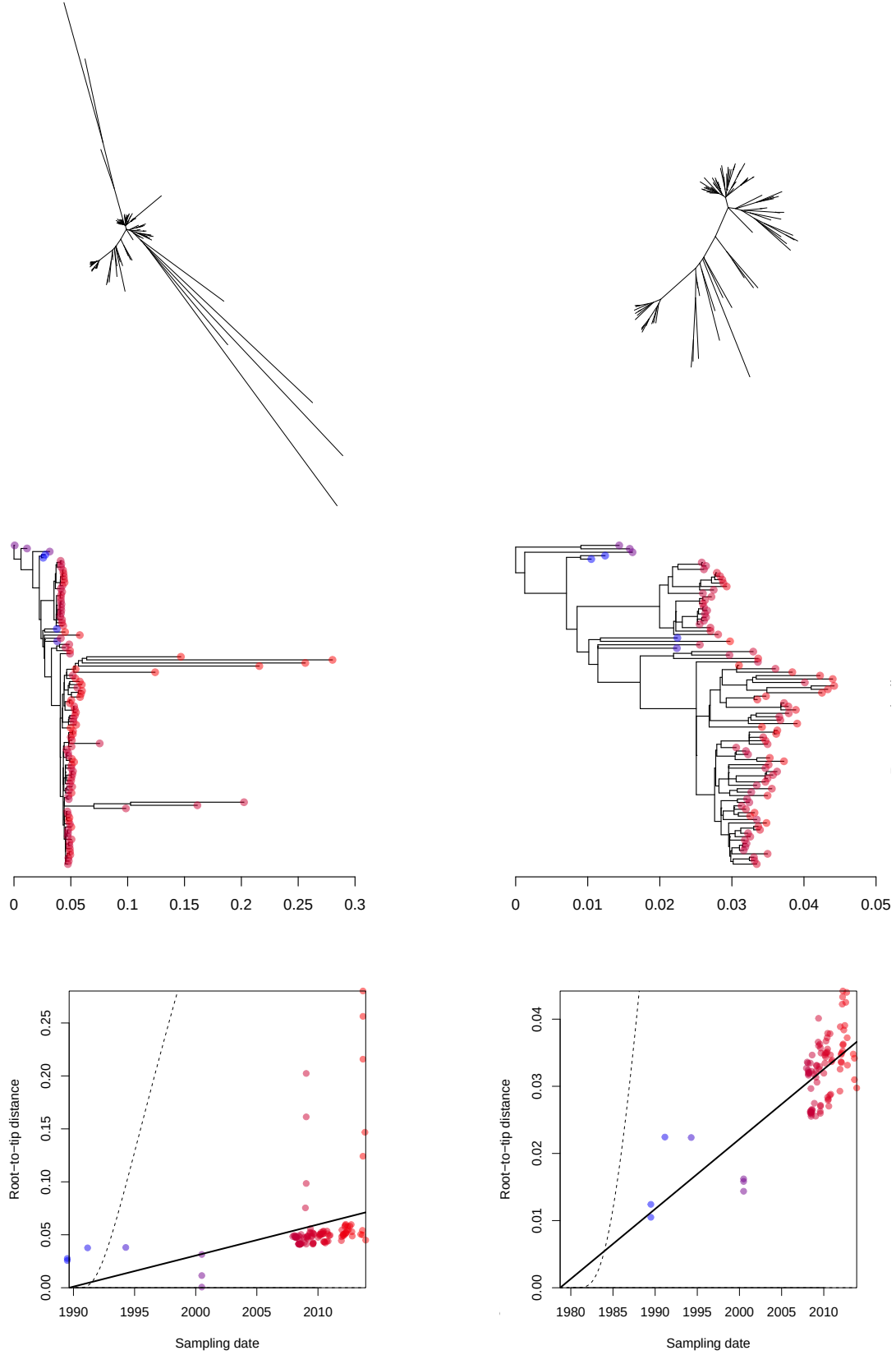


Figure 4. An example of a tree (ST 2332) before and after pruning with the CPA. The left column is the unpruned and the right column is the pruned tree. In the top row, the tree is visualized. In the middle, the root-to-tip distances are measured on the x -axis. At the bottom, the linear regression is visualized between the sampling dates (x -axis) and root-to-tip distances (y -axis).

Acknowledgments

We would like to thank Erika Bérczi-Kovács and Petra Csomós for organizing the research. Furthermore, we would like to thank Dorottya Farkas and Lőrinc Kóródi for partially working with us during the work. We are also grateful to our supervisor, Eszter Ari, who introduced us to the topic and provided great help throughout the project.

References

- [1] Uyen Mai and Siavash Mirarab. “TreeShrink: fast and accurate detection of outlier long branches in collections of phylogenetic trees”. en. In: *BMC Genomics* 19.Suppl 5 (May 2018), p. 272.
- [2] G Chartrand and Ping Zhang. *Introduction to graph theory*. en. The Walter Rudin student series in advanced mathematics. Maidenhead, England: McGraw Hill Higher Education, June 2004. URL: <https://scholarworks.wmich.edu/books/736>.
- [3] D Baum. “Reading a Phylogenetic Tree: The Meaning of Monophyletic Groups”. In: *Nature Education* 1.1 (2008). URL: <https://www.nature.com/scitable/topicpage/reading-a-phylogenetic-tree-the-meaning-of-41956/>.
- [4] Gabriel Cardona, Francesc Rosselló, and Gabriel Valiente. “Extended Newick: it is time for a standard representation of phylogenetic networks”. In: *BMC Bioinformatics* 9.1 (Dec. 2008), p. 532. ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-532. URL: <https://doi.org/10.1186/1471-2105-9-532>.
- [5] Eszter Ari. *IQR Tree Pruner*. 2023. URL: https://github.com/barizona/iqr_tree_pruner.
- [6] F.M. Dekking. *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer Texts in Statistics. Springer, 2005. ISBN: 9781852338961. URL: <https://doi.org/10.1007/1-84628-168-7>.
- [7] Jaime Huerta-Cepas, François Serra, and Peer Bork. “ETE 3: Reconstruction, Analysis, and Visualization of Phylogenomic Data”. en. In: *Mol Biol Evol* 33.6 (Feb. 2016), pp. 1635–1638.

- [8] Eszter Ari. “How do ‘superbugs’ emerge? - A systematic study of the mobility of resistance and virulence genes in human microbiota and pathogenic bacteria”. In: *Hungarian National Research Fund Grant – Postdoctoral Excellence Programme, Final report* (2023). URL: <https://www.otka-palyazat.hu/download.php?type=zarobeszamolo&projektid=131839>.
- [9] Mette V Larsen et al. “Multilocus sequence typing of total-genome-sequenced bacteria”. en. In: *J Clin Microbiol* 50.4 (Jan. 2012), pp. 1355–1361.
- [10] Xavier Didelot et al. “Bayesian inference of ancestral dates on bacterial phylogenetic trees”. en. In: *Nucleic Acids Res* 46.22 (Dec. 2018), e134.