LUDWIG-MAXIMILIANS-UNIVERSITÄT
TECHNISCHE UNIVERSITÄT MÜNCHEN

**Scientific Computing Group (HITS)/
Lehrstuhl XII: Bioinformatik (TUM)**

Master's Thesis
in Bioinformatik

Advanced Methods for Phylogenetic Post-Analysis

Andre J. Aberer

Aufgabensteller:	Prof. Stefan Kramer
Betreuer:	Dr. Alexandros Stamatakis
Abgabedatum:	14/01/2011

Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

14/01/2011 ____________________________

Andre J. Aberer

Abstract

Bootstrap analysis is a common method in phylogenetic inference for assessing to what degree the underlying genetic data support evolutionary relationships. The result of such an analysis can be flawed by the presence of *rogue taxa* in the dataset. These problematic taxa may assume various ambiguous positions in the phylogenetic tree. We review methods for identification of rogue taxa, propose appropriate variations, and also develop novel approaches. Some of the methods we propose are capable of identifying rogue taxa without using bootstrap trees as input. We define a criterion that specifies the optimality of a bootstrap analysis. Under this criterion, we compare how much the result of a bootstrap analysis can be improved, when rogue taxa as identified by competing methods are removed. We show that, the support in consensus trees can be substantially improved. A simple algorithm, that we propose, outperforms all other methods for identifying rogue taxa under the optimality criterion we define. We examine the question, whether we have to compute a new collection of bootstrap trees after removing identified rogue taxa from the original alignment. We find that, this is not necessary with respect to the optimality criterion. Thus, the direct output of our algorithm (a reduced consensus tree) can be used as a result. Furthermore, we provide an optimized implementation for a resolution-optimizing approach of rogue taxa identification that is two to three orders of magnitude faster than the original implementation.

Die Bootstrap-Analyse ist eine in phylogenetischen Studien häufig eingesetzte Methode, um zu bestimmen, wie stark evolutionäre Beziehungen durch die zu Grunde liegenden genetischen Daten unterstützt werden. Das Ergebnis einer solchen Studie kann erheblich verschlechtert werden, wenn sich im Datensatz sogenannte *rogue taxa* befinden. Diese Art problematischer Taxa kann verschiedene nicht eindeutige Positionen im phylogenetischen Baum einnehmen und somit das gesamte Signal empfindlich stören. In dieser Arbeit werden Methoden diskutiert, um rogue taxa zu identifizieren. Wir schlagen Variationen und neuartige Ansätze vor. Einige der von uns vorgeschlagenen Methoden können rogue taxa identifizieren ohne eine vorher berechnete Sammlung von Bootstrap-Bäumen dafür zu benötigen. Wir definieren zunächst ein Optimalitätskriterium für Bootstrap-Analysen. Anhand dieses Kriteriums evaluieren wir, zu welchem Grad das Ergebnis einer Bootstrap-Analyse (die Auflösung und Unterstützung des entsprechenden Konsensus-Baumes basierend auf den Bootstrap-Replikanten) verbessert werden kann, wenn rogue taxa (die anhand der verschiedenen Methoden identifiziert wurden) entfernt werden. Ein einfacher von uns vorgeschlagener Algorithmus übertrifft bei diesem Vergleich alle anderen Methoden bei der Identifizierung von rogue taxa signifikant in Bezug auf das Optimalitätskriterium. Wir untersuchen die Fragestellung, ob es notwendig ist, erneut eine Sammlung von Bootstrap-Bäumen zu berechnen, nachdem rogue taxa aus dem Orginalalignment entfernt wurden. Wir kommen zu dem Ergebnis, dass bezüglich des Optimalitätskriteriums dies nicht der Fall ist und somit die direkte Ausgabe unseres Algorithmus (ein reduzierter Konsensus-Baum) als Ergebnis verwendet werden kann. Des Weiteren, stellen wir eine optimierte Implementierung für einen Ansatz zur Identifizierung von rogue taxa zur Verfügung, dessen Ziel darin besteht die Auflösung des Konsensus-Baumes einer Bootstrap-Analyse zu optimieren. Diese Implementierung ist zwei bis drei Größenordnungen schneller als die ursprünglich veröffentlichte Version.

Acknowledgements

Many people have contributed to this thesis. Formerly, when reading acknowledgements, I always found them to be prone to clichés. Let us make it so:

- The biggest scientific *efcharisto* of course goes to Alexandros Stamatakis. If I had to limit myself to three things, I would say, that I liked the cheerful atmosphere in our discussions, his optimism and trust in my work in advance and his efficiency and accurateness concerning scientific work.
- I am grateful to Prof. Kramer for making this thesis possible.
- I would like to thank Nicholas D. Pattengale, Krister M. Swenson and Bernard M. E. Moret for giving me the opportunity to contribute to their publication and for all the interesting discussions, although we never met in person (which, I hope, changes some time).
- I am grateful to all biologists who sent us their precious data for our analyses. Bioinformatic analyses always are at most as good as the data they have for experimentation. Having stumbled through piles of old mail, I hope that I did not forget anybody or misspelled their name: Chen Wei Jen, Jonathan Fong, Markus Göker, Jessica Grant, Huang Jianhua, Brendan Hodkinson, Frank Kauff, Daniel Lahr, Annie Lindgren, Lora Lindley, Karen Meusemann, Bernhard Misof, Matthew Philips, Dave Redding, Niklas Wahlberg and Alexandra Wey.
- This thesis would not have been possible without the excellent previous and related work (see the bibliography). As always in science: *nanos gigantium humeris insidentes.*
- Concerning personal matters, I would like to thank my parents, Anita and Jakob Aberer for making my studies (including this thesis) possible and their relentless support throughout my studies.
- The biggest personal thanks of course goes to my beloved, Sabrina Heinzmann. Thank you for your never-ending patience, support and love, mi corazón.

Contents

1. Introduction

1.1. Phylogeny Inference

Phylogenetics is a biological discipline with the ultimate (yet still distant) goal to infer the evolutionary relationships among the immense variety of living and extinct species. One key question is where this huge diversity comes from. The self-assembly of increasingly complex bio-molecules and subsequent evolution at this level [Gilbert, 1986], finally led to the first entities that match our definition of life. In phylogenetics, we assume that all species descend from a hypothetical universal common ancestor. Darwin [1859] describes this speciation process from this common ancestor as a sequence of branching events that is driven by natural selection. The result of this process is the *universal tree of life*, the most comprehensive possible phylogeny, that contains all species on earth and the relationships among them. Due to the sheer number of distinct species, biology is still far away from assembling the tree of life. Biologists usually reconstruct trees with about 100 to 1000 species in their studies.

From a mathematical perspective, phylogenies are represented either as a rooted or an unrooted leaf-labelled binary tree (see Figure 1 for instances of unrooted trees). The leaves in this tree (nodes of degree one) stand for the various species under examination and the inner nodes (with a degree of three) depict the speciation events by means of hypothetical common ancestors. A disadvantage of this simple tree-based model of evolution is that binary trees can not be deployed to model evolutionary processes such as horizontal gene flow or hybrid speciation. To model such complex evolutionary processes, phylogenetic networks have been proposed [reviewed in Moret et al., 2004]. Nevertheless, phylogenetic trees are by far the most common representation. This is not only because they are rather straight-forward to interpret. Trees as a model are either sufficient for most occasions or often they are legitimate simplifications (when vertical gene flow mostly determines evolutionary relationships).

Methods and data used for phylogenetic inference have changed significantly during past decades. Originally, mostly morphological traits were used to assess similarities between species (also called *taxonomic units* or *taxa*). With the discovery of the DNA as the container for genetic information and the constant improvement of DNA sequencing techniques, genomic data of the species under study (mostly used in form of multiple sequence alignments) complements and has now largely replaced morphological information. Other key technological innovations that make current-day large scale phylogenetic analyzes feasible, are the continuous advances in processor technology (and more recently the era of multi-core machines) and parallel *next generation sequencing* (NGS) methods [see Metzker, 2010].

In general terms, phylogenetic inference is the process of reconstructing an evolutionary model (tree) that best fits the data. The models and algorithms required for reconstructing biological meaningful trees extend way beyond a simple similarity clustering of the input sequences. The following three inference methods are currently most widely used:

- **Maximum Parsimony** (MP) is a non-parametric statistical approach. This means that explicit mutation rates are, for instance, not, estimated from the data. Instead when applying MP, we assume that the optimal tree requires the smallest number of mutations for explaining the data. It is the fastest of these three methods outlined here, but usually yields sub-optimal trees, when taxa under study are not closely-related [Holder and Lewis, 2003]. PAUP* [Swofford, 2003] or TNT [Goloboff et al., 2008] are popular implementations of this method.

- **Maximum Likelihood** (ML) strives to maximize the likelihood of an evolutionary tree and model given the input data. ML is known to be more robust than MP with respect to missing data and noise, but comes at a significantly higher computational cost. Thus, popular tools for ML-based phylogenetic inference such as GARLI [Zwickl, 2006], RAxML [Stamatakis, 2006a], TreePuzzle [Schmidt et al., 2002] or PhyML [Guindon and Gascuel, 2003] offer highly optimized and parallelized implementations of the likelihood function and/or respective search algorithms to allow for phylogenetic inference on large input datasets (multiple sequence alignments).

- **Bayesian Inference** methods (implemented in MrBayes [Huelsenbeck and Ronquist, 2001] or PhyloBayes [Lartillot et al., 2009]) assess trees under the maximum a posteriori probability. As this probability includes the computation of the likelihood function, Bayesian and ML methods are similar with respect to the computational challenges they face. An important difference is that while ML strives to maximize the parameters of the likelihood function, Bayesian inference methods integrate over the model space parameters with the goal to "integrate out phylogenetic uncertainty". There exist cases where Bayesian inference outperforms ML and vice versa [Holder and Lewis, 2003]. Thus, the two methods may be regarded more as being complementary, rather than rivaling approaches.

Finding the global optimum for any of the three criteria is extremely unlikely for non-trivial instances because of the combinatorial explosion in the possible number of tree topologies for a set of n taxa. Thus, phylogenetic tree reconstruction under any of these optimality criteria is $\mathcal{NP}$-hard [Chor and Tuller, 2005]. In current programs (such as MrBayes, RAxML, TNT, etc.) various search strategies/heuristics are deployed to decrease the amount of time spent for examining sub-optimal regions of the search/tree space. Since the problem is $\mathcal{NP}$-hard, tree search heuristics mostly yield a locally optimal tree.

1.2. Phylogenetic Post-Analysis

All phylogenetic inferences and post-analyses conducted within the framework of this thesis are based on RAxML. Hence, we limit our focus to ML-based phylogenetic inference.

The search for the optimal, or best-known for that matter, ML tree is usually supplemented by a bootstrap analysis [Felsenstein, 1985]. In this kind of analysis, bootstrap replicate alignments are sampled by drawing columns/sites from the original input alignment at random with replacement. Subsequently, a ML tree is inferred for each of the bootstrap replicate alignments. The resulting bootstrap replicates provide information, as to how well phylogenetic statements (branches in the tree) are supported by the underlying data. If there is no insufficient signal in the data to support the relationships between two subsets of taxa, corresponding BS support values in this region of the tree are expected to be low.

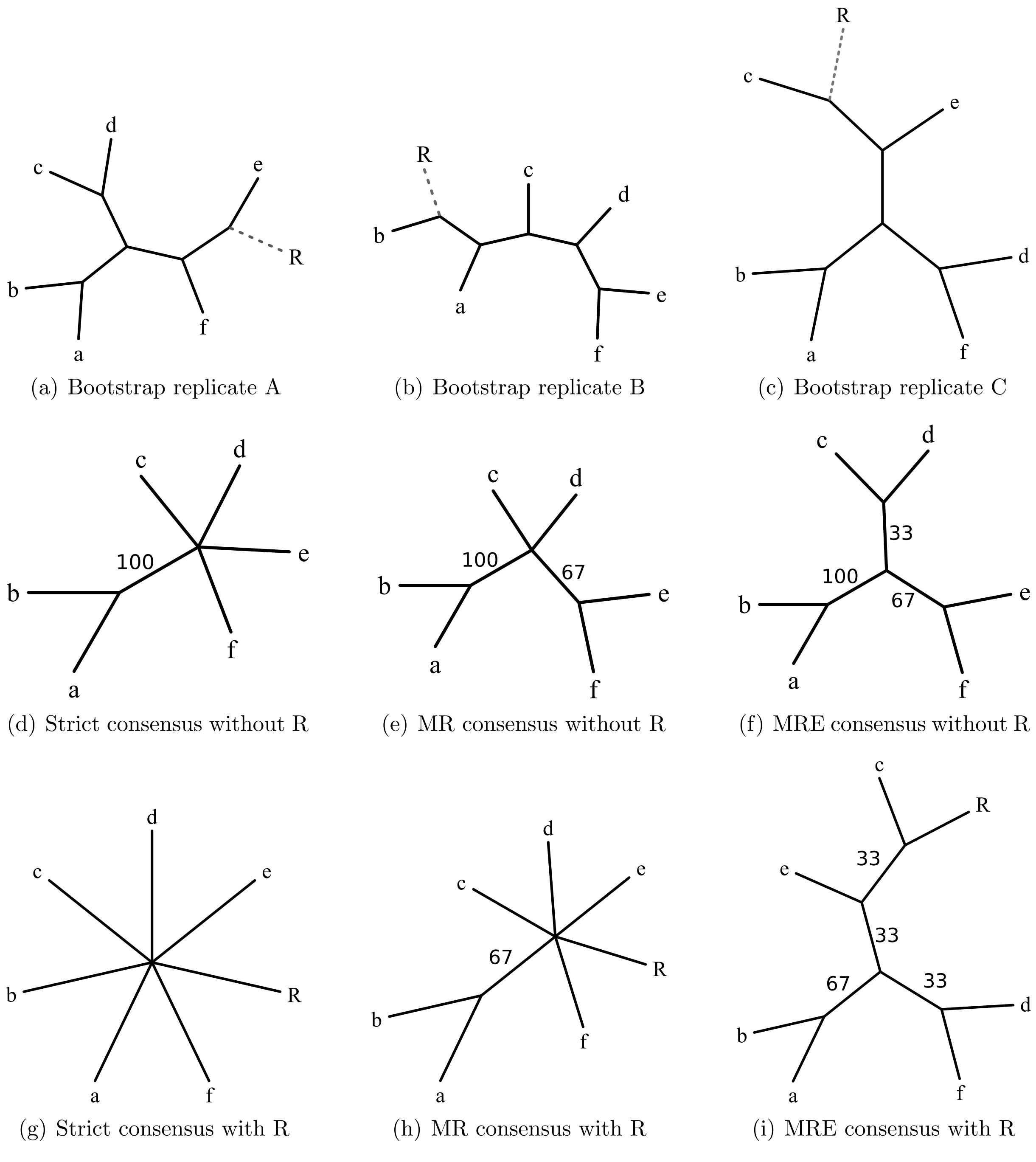
Bootstrap support values can be determined for every branch that connects two inner nodes of the tree, that is, for each inner branch of the tree. Consider the inner branch in tree A (see Figure 1(a)) that connects taxa a and b to the rest of the tree: the implicit phylogenetic hypothesis of this branch is that a and b are more related to each other than to taxa c, d, e or f. Removing this inner branch that leads to a and b, splits the tree into two partitions. Such a split of the tree into two parts is thus termed a *bipartition* or *split* of the taxon set. The bipartition in our example can be written as: $ab|cdef$. Note that, a split leading to a tip, that separates a single leaf from the rest of the nodes/taxa is contained in every tree. Therefore, this type of bipartitions is trivial because they do not carry any information about the structure of the underlying tree topology.

For processing the information contained in a set of bootstrap replicates (a set of trees), the trees are transformed into a *bipartition profile* [as described for instance in Pattengale et al., 2010a]. This profile consists of the set of all non-trivial bipartitions $b_1, b_2, \dots b_k$, where k is the number of bipartitions. Each bipartition also has an attribute $sup(b_i)$ that contains the frequency of occurrence of a bipartition in the set of bootstrap trees. The information provided by a bipartition profile can be used in the following two ways:

- the support values (frequencies) of the bipartitions in the bipartition profile can be drawn onto the best-known ML tree
- a consensus tree can be computed from the bipartition profile.

Several consensus tree building methods have been proposed over the years [reviewed in Bryant, 2003]. However, in most phylogenetic studies the following three consensus methods are predominantly employed:

- The **strict** consensus tree consists of bipartitions that are contained in every bootstrap replicate tree. As depicted in Figure 1(d), the strict consensus tree mostly is not a fully resolved binary tree. Strict consensi usually contain multifurcations. Here, for instance, the strict consensus does not yield any information about the relationship between taxa c and f in the bootstrap trees.
- The bipartitions that are contained in at least half of the bootstrap trees are used to build a **majority-rule** (MR) consensus (see Figure 1(e)). A MR tree therefore contains more information about branching patterns than the – rather conservative – strict consensus. A MR consensus is also relatively cheap and straight-forward to compute.
- The **extended MR** (MRE) consensus is computationally significantly more demanding. The extended MR algorithm intends to include further bipartitions with high support that occur in less than half of the bootstrap replicate trees to the MR consensus tree. However, not every bipartition present in less than 50% of the trees is compatible with the MR consensus. Thus, each candidate bipartition needs to be checked against all previously included bipartitions of the consensus tree. This step is not necessary for the non-extended MR consensus, since bipartitions that occur in more than 50% of the tree set are always compatible [Margush and McMorris, 1981]. Note that, the MRE method is based on greedy heuristics. Computing the maximum set of compatible bipartitions is equivalent to finding the maximum independent set on colored graphs and is therefore $\mathcal{NP}$-hard [Phillips and Warnow, 1996]. Although the example MRE tree in Figure 1(f) is fully resolved, the bipartition $abef|cd$ actually occurs in solely one tree. So the information value of the additional resolution is questionable.



(a) Bootstrap replicate A
(b) Bootstrap replicate B
(c) Bootstrap replicate C
(d) Strict consensus without R
(e) MR consensus without R
(f) MRE consensus without R
(g) Strict consensus with R
(h) MR consensus with R
(i) MRE consensus with R
a
b
c
d
e
f
R
100
67
33

Figure 1: a-c) Three bootstrap replicates A, B and C. **d-f)** The strict (d), majority rule (e) and majority rule extended (f) consensus trees of A, B and C without rogue taxon *R*. **g-i)** The strict (g), majority rule (h) and majority rule extended (i) consensus of A, B and C including the rogue taxon *R*. The inner branches of consensus trees are annotated with support values in [%].

1.3. Rogue Taxa

The taxa c and d in tree A and B (Figure 1) provide an example of how bootstrap replicate trees may disagree with respect to the exact placement of taxa. Moreover, there are cases, where taxa or a single taxon can assume a multitude of positions in distinct bootstrap replicates – or in the worst case, a different position in each replicate tree. Wilkinson [1994, 1995, 1996] first examined these unstable taxa and coined the term *rogue taxa* [Wilkinson, 1996] for such problematic taxa.

Figure 1 g-i) depict the impact of a single rogue taxon R on trees built with the three aforementioned consensus methods. The strict consensus degenerates to a star tree that does not contain any information about the evolutionary relationships. Strict consensus trees are particularly prone to rogue taxon effects: every placement of R, apart the first one leads to the loss of one consensus bipartition. Figure 1(h) illustrates how the MR consensus loses a bipartition because of a rogue taxon. Also, the support of bipartition $ab|cdef$ is decreased. Figure 1(i) provides the respective MRE consensus tree that has one bipartition more than the MRE tree in in Figure 1(f). This is because, with the additional taxon R, the $n-3$, where n is the number of taxa, possible consensus bipartitions are also increased by one. Thus, the strict and MR consensi do not only loose consensus bipartitions and support for the remaining ones, but also fail to attain additional ones.

Sanderson and Shaffer [2002] suggest that the reasons for occurrence of rogue taxa are manifold. For instance, missing sequence data for a taxon in the alignment can yield its placement unstable or a particularly high mutation rate might lead to contradicting signals in the alignment. On the other hand, very low mutation rates may also not provide enough information for an unambiguous inference of the evolutionary relationships. Erroneous sequences that are composed of different sequence parts from distinct species (so-called *chimeras* [Wintzingerode et al., 1997]) are another potential source of rogueness. Furthermore, if evolutionary very distant species are placed in a tree with otherwise short branch lengths (i.e., a tree comprising closely related species), those distantly related species may assume highly variable positions in the set of bootstrap trees. In this case, possible placements just become too similar, that is, indistinguishable from each other with respect to their likelihood scores for instance.

The negative impact of rogue taxa on phylogenetic inference is not limited to bootstrap analyses. These taxa also induce instability when different estimation parameters or methods (such as MP, ML or Bayesian inference) are used [mentioned by Pattengale et al., 2010a]. Problems with rogue taxa were also reported for supertree analyses [Pisani et al., 2002]. Supertrees methods strive to combine smaller phylogenies into a more comprehensive tree. This way, supertrees make statements about the evolutionary relationships between species that were examined in different studies.

An alternative approach to supertrees that tries to harness huge amounts of data for the construction of phylogenies is the super-matrix approach (also called total evidence or phylogenomic inference). By concatenating many single-gene alignments for the species under study into a super-matrix, the probability that contradicting phylogenetic signals in the genes will produce rogue taxa increases [as in Thomson and Shaffer, 2010a]. All of the above indicates that rogue taxa are more likely to occur in huge phylogenies.

While, for small datasets biologists may be able to identify rogue taxa by manual/visual inspection, the general trend for larger phylogenies to contain more rogues calls for automated rogue taxa identification methods. To date, only few phylogenetic studies explicitly mention that a systematic rogue taxon analysis was conducted. Two common measures of instability are used to identify rogues: the **leaf stability index** [Thorley and Wilkinson, 1999] and the **taxonomic instability** implemented in the phylogenetic toolbox Mesquite [Maddison and Maddison, 2010]. Recently, Pattengale et al. [2010a] expanded the theoretical work on rogues and developed new criteria and heuristics for rogue taxon identification. These methods will be discussed in detail in Section 3. Wilkinson's reduced consensus trees are omitted due to the issues pointed out by Pattengale et al. [2010a]: computational complexity (a clique search on graphs of considerable size would be necessary) and a relative difficulty in interpreting results because sets of trees are returned, rather than a list of potential rogue taxa.

All of these methods handle rogues by removing them from bootstrap replicates and/or the alignment for a subsequent rerun of the entire analysis. This does not necessarily reflect the way biologists would like to handle rogues. It would be desirable to still be able to infer a reliable phylogenetic position of the candidate rogue taxa. However, sometimes, taxa will simply be difficult to place in a phylogeny, even if difficulties as those discussed above (e.g., missing data or mutation rates inappropriate for a phylogenetic inference) can be circumvented. In such cases, extensive manual data inspection and hypothesis testing may be necessary to find a meaningful placement [Shafer and Hall, 2010].

Recently, the effect of rogues was examined from a mathematical perspective [Cueto and Matsen, 2010] in the context of balanced minimum evolution (a rather infrequently used distance matrix-based phylogenetic inference approach). Cueto and Matsen show that, the addition of a rogue taxon can lead to a maximally different optimal tree. Furthermore, they simulate and examine the behavior of rogue taxa in detail for small trees with four up to six taxa.

1.4. Own Contribution

In this thesis, we strive to extend previous work (discussed in Section 1.3), most notably that of Pattengale et al. [2010a]. Part of this thesis is published in an extended version of Pattengale's conference publication, which will appear in the journal *IEEE/ACM Transactions on Computational Biology and Bioinformatics* in 2011. The journal contribution constitutes a highly efficient implementation of Pattengale's algorithm (Sections 6.1 and A.3) and a comparison of Pattengale's algorithm to the Maximum Agreement Subtree approach (see Section 6.2). In addition to this, we propose an alternative to Pattengale's optimality criterion, focusing on optimization of the bipartition support in consensus trees.

Moreover, we provide efficient implementations for two stability measures and deploy these measures in two distinct algorithms for optimizing the criterion we define. We propose a simple, yet efficient and accurate, algorithm for the solution of the optimization problem on a given collection of bootstrap replicate trees. Furthermore, we propose a family of methods that approximates a bootstrap analysis and is thus capable of identifying rogues without relying on a bootstrap tree collection as input. We compare the performance of all of the above methods based on our optimality criterion. To the best of our knowledge, this is the first comparative analysis of methods for rogue taxon identification. This analysis is carried out on real-world datasets that are suspicious for containing rogue taxa. We also address the question, whether it is necessary to re-compute a new collection of bootstrap trees using an alignment from which identified rogue taxa have been removed.

1.5. Structure of the Thesis

In Section 2, we discuss various criteria for the optimization of the quality of a consensus tree and propose a novel criterion that is deployed throughout this thesis. We review existing methods for rogue taxa identification and propose new methods or variations to existing ones in Section 3. All methods of this section identify rogue taxa based on a given collection of input (bootstrap) trees. We also propose a novel approach to rogue taxa identification, that does not rely on bootstrap trees as input in Section 4. The datasets that we use for performance analysis of the various rogue taxa identification methods are described in Section 5. Then, in Section 6, we conduct a comparative performance analysis of all methods based on our optimality criterion. We analyze the practical advantage of different features inherent to the alternative approaches for rogue taxa identification. Furthermore, we examine the runtime improvement of our optimized implementation of Pattengale's algorithm. We also discuss the necessity of repeating/re-computing a bootstrap analysis for a pruned set of taxa from which rogue taxa have been removed. We summarize the results of this thesis and present potential directions for future work in Section 7. In the appendix (see Sections A and B), we provide a more detailed description of important implementation aspects of the respective methods for rogue taxa identification.

2. Optimality Criterion

2.1. Rogue Taxa and Optimality of the Consensus

Throughout this thesis, rogue taxa identification is defined as the task of improving the consensus tree quality by means of eliminating (*pruning*) putative rogue taxa from the bootstrap tree collection. The definition of a rogue taxon is problematic: while there usually exists a subset of comparatively stable backbone taxa, all remaining taxa are relatively unstable. This instability may be biological meaningful. If rogue taxa impact the bipartition profile, then there is no taxon that is entirely stable. Even the most stable taxa are unstable in their relationships with the rogue taxa. We may hypothesize that such destabilizing effects can accumulate and induce rogue behavior on de facto well-behaving taxa. If we prune those taxa that give rise to this instabilities, the taxon may regain its stability. Thus, rogue taxa identification does not represent a classic categorization task as known from the field of machine learning. The effect of pruning a taxon depends on which other taxa are also pruned from the trees.

Here, we deploy a loose definition of rogue taxa: a rogue taxon is a taxon that has a negative impact on the support of one or more bipartitions in a consensus tree or whose removal induces the appearance of additional bipartitions in the consensus tree. We define the *impact* of a rogue as the degree of its deleterious effect, that is, the improvement of bipartition support that is induced by removing it. The above definitions also apply accordingly to sets of taxa.

For the optimization we need to decide on a quality criterion. Thus, we intend to identify rogue taxa such as to maximize either the resolution

$$\#bip = |\{b_i|\ b_i \text{ is a consensus bipartition}\}| \tag{1}$$

or the relative sum of the bootstrap support values in the consensus tree

$$sumSup = \sum_{i=\{1..k\}} sup(b_i), \text{ where } k \text{ is the number of consensus bipartitions.} \tag{2}$$

By using support values (in *sumSup*), we ensure that the leaf sets (the set of rogue taxa) to be removed (also termed: *dropsets*) are chosen such that they maximize the overall support of the bipartitions in the consensus tree. Consider using a MR consensus tree. We may be able to choose between dropping a taxon set such that a bipartition with 100% appears in the consensus tree or dropping another taxon set such that a bipartition with 50% support appears in the consensus tree. On the other side, *sumSup* does not distinguish between cases, where the consensus is extended by a single bipartition with 100% support or two bipartitions with 50% support. Biologists may however prefer the latter choice, since trees with increased resolution (less multifurcations) are easier to interpret and visualize.

2.2. Relative Bipartition Information Content

The two quality measures in Equations (1) and (2) are suboptimal in the sense, that their value strongly depends on the size of the tree they are applied to. While a consensus tree with 10 taxa is fully resolved with 7 consensus bipartitions, the same number of bipartitions in a consensus tree of 100 taxa means that there is almost no resolution at all.

Due to the advantages outlined in Section 2.1, we derive an optimality criterion based on Equation (2). We then deploy this optimality criterion for all comparative analyses conducted in this thesis. Simply using the average support of a consensus bipartition does not represent a good choice for an optimality criterion. It implies, that eliminating low-frequency bipartitions from the consensus improves the score. Thus, the resolution of the consensus tree might be reduced, while the score improves.

Alternatively, we can normalize Equation (2) by the maximum possible support in the consensus tree after dropping/removing candidate rogue taxa. If we start our analysis with n taxa, we have n' taxa . Thus, a possible optimality criterion can be:

$$\frac{\sum_{i=\{1..k\}} sup(b_i)}{n'-3}. \tag{3}$$

The problem with score (3) is a lack of conservativeness: Assume that there is one bipartition in the bootstrap tree set with maximum support. If we prune all, except four taxa, that give rise to this fully supported bipartition, we will obtain a maximum score of 1.

When dropping taxa, we rapidly reduce the number of possible distinct bipartitions in a consensus tree. Thus, we should appropriately adjust for this to obtain a more objective quality measure. The number of possible distinct bipartitions as a function of the number of taxa in a tree with n taxa is

$$f(n) = 2^{n-1} - n - 1. \tag{4}$$

See Table 1 for an exhaustive enumeration for 5-taxa case. Because of symmetry, we always choose a reference taxon and a reference partition. In the example taxon A is the reference taxon and the partition containing A is the reference partition. Table 1 shows a bit vector representation of a bipartition: the bit is set for all taxa that are in the same partition as the reference taxon. Neglecting this reference taxon (that always needs to be set), the power set of the remaining $n-1$ taxa yields 2^{n-1} possible bipartitions. If we exclude all trivial bipartitions and also bipartitions with empty partitions on the left or the right, we obtain Equation (4).

A	1	1	1	1	1	1	1	1	1	1
B	0	0	0	1	0	0	1	0	1	1
C	0	0	1	0	0	1	0	1	0	1
D	0	1	0	0	1	0	0	1	1	0
E	1	0	0	0	1	1	1	0	0	0

Table 1: The 10 possible distinct bipartitions for the five taxa A, B, C, D, and E. Bipartitions are denoted with respect to reference taxon A, that is, the first bit in the bipartition bit vector is always set to 1. If a taxon is in the same partition as A, the value in the matrix is 1.

For large numbers of n, the adjustment factor $\frac{f(n)}{f(n-1)}$ for dropping one taxon converges to 2. Thus, adjusting Equation (3) using this factor would require the unadjusted quality score to be increased by a factor of 2 for each taxon that is dropped. This renders such an adjustment unusable, as the adjustment factor is over-restrictive: assume we have 50 bipartitions with full support in a tree of 100 taxa. Dropping a rogue taxon such that the optimality of the consensus tree increases would require the consensus tree to contain approximately at least 100 bipartitions with full support. However, the pruned tree contains only 96 bipartitions.

Thus, the unadjusted optimality score we deploy here has a different desired property: it should depict the information of the bipartition support in the pruned consensus tree in relation to the maximal support possible in the unpruned consensus tree with n taxa. The sum of the full support on the fully resolved consensus tree comprising all n taxa, is $n-3$, thus we can deploy the following optimality criterion

$$\mathrm{RBIC}(\mathcal{C}') = \frac{\sum_{i=\{1..k\}} sup(b_i)}{n-3}. \tag{5}$$

We denote this measure as the *relative bipartition information content* (RBIC) of the pruned consensus $\mathcal{C}'$. The RBIC measure takes into account, that the number of all possible bipartitions decreases with each taxon that is pruned. Thus, if a bipartition vanishes after a taxon is pruned, the score will decrease (which is not necessarily the case for the criterion in Equation (3)). The RBIC attains the maximum score of 1.0 only if all bipartitions in the unpruned consensus tree are fully supported.

We can compute the RBIC for all three consensus tree methods (SC, MR, MRE) discussed in Section 1.2. As already outlined, a MR consensus tree refines the strict consensus tree and a MRE consensus tree is a further resolved MR tree. With respect to the RBIC, this means that $\mathrm{RBIC}(\mathcal{C}_{\mathrm{strict}}) \leq \mathrm{RBIC}(\mathcal{C}_{\mathrm{mr}}) \leq \mathrm{RBIC}(\mathcal{C}_{\mathrm{mre}})$. For our comparative analysis of rogue taxa identification methods, we will mostly use the RBIC of MR consensus trees. The rationale for focusing on MR trees is that, the strict consensus for large phylogenies (i.e., more than 500 taxa) is usually very poorly resolved. A reason against using MRE consensi tree the $\mathcal{NP}$-hardness of the MRE tree building problem. Thus, the resolution of a MRE tree is influenced by the heuristics that are deployed and does thus not represent a reliable measure. In other words, it is unclear, if an improved RBIC obtained from a MRE consensus is the result of the removal of a rogue taxon or a specific bipartition profile on which the MRE heuristics work better by chance alone.

2.3. Relative Information Content

The *relative information content* (RIC) which was introduced by Pattengale et al. [2010a] is conceptually similar to the RBIC. In fact, the RBIC is a simplified version of the RIC.

Using the RIC does not lead to dropping taxa at any cost. The RIC only improves, if the resolution of the consensus tree increases by a number of bipartitions that is greater than the number of taxa that has been dropped to achieve this increase of consensus bipartitions.

Let the initial consensus tree consists of n taxa and k bipartitions. After dropping some potential rogue taxa, the consensus tree will comprise n' taxa and k' bipartitions. Then the RIC of the restricted consensus tree $\mathcal{C}'$ (i.e. the consensus tree after pruning the drop set/rogue taxa) is

$$\mathrm{RIC}(\mathcal{C}') = \frac{n'+k'}{n+n-3}. \tag{6}$$

The goal of the RIC measure is to optimize two parameters: the resolution in the pruned consensus tree and the number of taxa included therein. Using an additional parameter α for weighting n' against k' (and analogous modification in the denominator), biologists can adjust the RIC to their specific needs. Using n' in the enumerator particularly makes sense, as having more taxa in a consensus tree can mean that more information on the tree is available, irrespective of the number of bipartitions. Consider the MR consensus tree in Figure 1(g): we see that f is more distant to a and b than a is to b. If we prune f from the consensus tree, we loose this information on relative evolutionary distances. While a decreasing RIC can accommodate this effect, the RBIC of the pruned consensus does not change with respect to the comprehensive consensus. On the other hand, for an unrestricted consensus tree as outlined in Figure 1(g) (this tree does not contain any evolutionary information), the RIC is greater than zero, while the RBIC of this consensus is zero. Thus, there exist situations where an unpruned star tree has a higher RIC than a considerably pruned-down consensus tree with bipartitions.

2.4. Entropy-Based Measures

The criteria discussed so far lack two desirable attributes:

1. **Independence of consensus method used:** All criteria compute optimality based on a specific consensus method. After a consensus tree has been built, all information about bipartitions that are not contained in the consensus is lost. Therefore, a direct optimization on the entire bipartition profile (and not only on the highly supported bipartitions therein) would be desirable. One would expect that any consensus method that is applied to the improved bipartition profile will yield better consensus trees.

2. **Comparability among consensus trees with different taxon sets:** For our analyses it suffices to start with the maximum number of taxa and prune taxa until no further improvement is achieved. RBIC and RIC are not absolute measures, but measure the information relative to the initial number of taxa n. This is undesirable, if biologists consider to extend their initial set of taxa. Furthermore, it is not possible to make a statement if, for instance, a consensus tree with 100 taxa and 30 consensus bipartitions contains more information than a consensus tree with a set of 80 (different) taxa and 25 bipartitions.

The second issue was solved for strict consensus trees by Wilkinson et al. [2004], who proposed an entropy-based measure for the information content in consensus trees called *cladistic information content* (CIC):

$$\mathrm{CIC}(\mathcal{C}) = -\log \frac{\text{number of trees allowed by } \mathcal{C}}{\text{number of possible trees with the taxon set of } \mathcal{C}} \tag{7}$$

The number of allowed trees in the enumerator refers to the number of possible refinements of the consensus. For instance, the consensus tree in Figure 1(e) (assume it was a strict consensus) can be refined by adding a bipartition $abc|def$ or $abd|cef$ or $abef|cd$. So the consensus tree allows for three trees. Before the information is conveyed (in form of the consensus tree), we have a total of $(2 \cdot 6 - 3)!! = 945$ possible trees [see Felsenstein, 2004] for a tree with 6 taxa. This means that according to the CIC, this consensus tree represents $-\log\left(\frac{3}{945}\right) = 8.3$ bits of information.

For an objective criterion that is independent of the consensus method, we may consider measuring/computing the entropy of a bipartition profile. There is high entropy (and little information) in the bootstrap trees, if the bipartition profile consists of a large number of infrequent bipartitions. A low number of frequent (highly supported) allows the virtual observer to convey a large amount of information.

If we interpret a bipartition profile as a source of information X [Shannon and Weaver, 1949] that emits symbols (here bipartitions), then the entropy of X (with k bipartitions) can be defined as

$$H(X) = -\sum_{i=1}^{k} sup(b_i) \cdot ld(sup(b_i)). \tag{8}$$

Such a definition is problematic for two reasons: We can not assume that bipartitions occur with the same probability. The rationale used by Wilkinson et al. [2004] is that, before the information (in their case in form of the consensus tree) is conveyed, every tree topology has the same probability. The probability of a bipartition being contained in the set of all possible trees is dependent on the sizes of its partitions. For instance the bipartition $abc|def$ is contained in 9 out of the 945 possible 6-taxon trees. However, a bipartition with partition sizes of 2 and 4 (e.g., $ab|cdef$) occurs in 15 trees (12 possibilities of taxon assignments to the topology on the left in Figure 2 and 3 possibilities for the topology on the right).

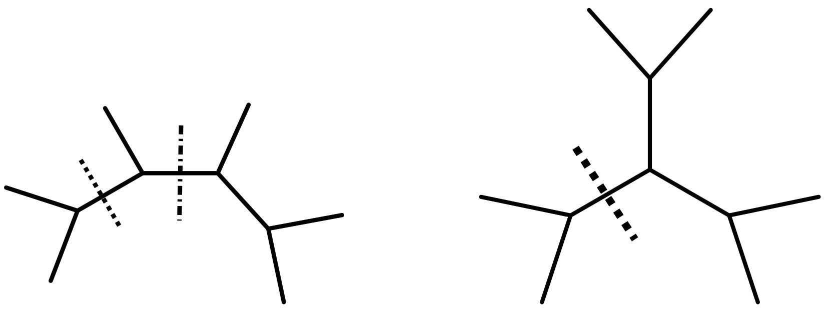


Figure 2: The two different classes of tree topologies for six taxa. A partition size of three is only possible for the left topology. Both topologies contain bipartitions with a partition size of two.

The other problem is that in information theory, we usually assume that the observed symbols are observed independently of each other. If there are dependencies among the symbols (e.g., among syllables in natural languages), we use the conditional probability given the previously observed symbol. This usually is an appropriate simplification. However, as bipartitions do not have a natural order, we would need to compute the conditional probability of a bipartition given all previously observed bipartitions. Furthermore, there is a particularly strong dependency among bipartitions. For example, reconsider the consensus tree in Figure 1(e) for which there exist three bipartitions that refine the tree. When we observe the bipartitions of this consensus tree, this means that only these three possibilities (bipartitions) can be a possible next symbol.

Thus deriving and computing an objective entropy-based optimality score for a bipartition profile is particularly difficult. Therefore, while keeping the limitations of the RBIC in mind, we will mainly base our findings on this measure.

3. A Posteriori Methods For Rogue Taxa Identification

In this section, we discuss various existing rogue identification, adapt them to our optimality criterion and propose appropriate variations. The all fall into the category of *a posteriori* methods, since they strive to identify rogues from bootstrap replicate trees. Thus, they are applied **after** a bootstrap analysis has been conducted.

3.1. Maximum Agreement Subtree Method

The *Maximum Agreement Subtree* (MAST) differs from the other methods discussed in this Section. The goal of MAST is not to identify rogues, but to identify (sub-)tree topologies that are present in all input trees. The original name *common pruned tree* [Finden and Gordon, 1985] describes their function in a more intuitive way: pruning from a specific set of taxa from all input trees will yield identical pruned trees. Any of the pruned tree will be identical to the strict consensus tree of the pruned input trees. In mathematical terms, we are searching for a subtree with the maximum number of leaves that is homeomorphically included in all of the bootstrap trees [Berry and Nicolas, 2006].

As an example, let us consider again the three bootstrap trees A, B and C from Figure 1 (without the rogue taxon R). All three trees contain the subtree in Figure 3(a). As one may observe, there exist various alternative taxon sets that will produce an agreement subtree of maximal size in this specific example. For instance, we can choose between a MAST that contains taxon c or taxon d. There even exists a MAST that contains c and d, but unlike the examples provided in Figure 3(a) this MAST does not contain taxa e nor f. The MASTs of bootstrap tree A and B (of Figure 1) is larger, since it comprises 5 taxa (see Figure 3(b)). Once again, there exists more than one unique MAST. The MASTs of all three bootstrap trees are also agreement subtrees of bootstrap trees A and B. However, as they are not maximal, they are not MASTs.

Based on the example, we may conclude that with an increasing number of taxa and input trees, the size of the MAST is likely to decrease. Furthermore, we see that MAST algorithms may not return a unique tree, but may indeed return a set of MASTs. In fact, the number of MASTs can be exponential in the number of taxa [Kubicka et al., 1992]. While this is inconvenient with respect to result interpretation, it does not represent a problem for scoring the trees under the RBIC and RIC criteria: Each MAST contains the maximum number of bipartitions for the taxon set in the MAST and each bipartition is contained in every input tree, that is, each bipartition has 100% support. Thus, RBIC and RIC, are invariant with respect to alternative MASTs for a tree set.

Nonetheless, the MAST is rarely used for summarizing information in bootstrap tree collections. The method is usually deployed to summarize information of tree collections that have equally good optimality scores (e.g., equally parsimonious Maximum Parsimony trees). Consensus trees may be too poorly resolved for this specific task. Beside that, the usage of MR consensus trees for summarizing equally parsimonious trees has been criticized, since by using consensus methods, some optimal solutions may be completely excluded from the MR consensus and lead to biased results [Sumrall et al., 2001]. Moreover, there are interesting applications for MASTs in the field of supertree construction [Berry and Nicolas, 2007]. Finally, Cranston and Rannala [2007] proposed a method that is similar to the MAST, but less strict. They use agreement subtrees for summarizing trees that are sampled during Bayesian inference.

With respect to computational complexity, the MAST problem is sub-quadratic in the number of taxa for two rooted trees [Cole and Hariharan, 1996]. To the best of our knowledge, the fastest algorithm for more than two unrooted binary trees is described by Berry and Nicolas [2006]. Berry and Nicolas showed that this problem can be solved in

$$\mathcal{O}\left((p+1)\cdot\min\{3^p kn, 2.27^p + kn^3\}\right), \tag{9}$$

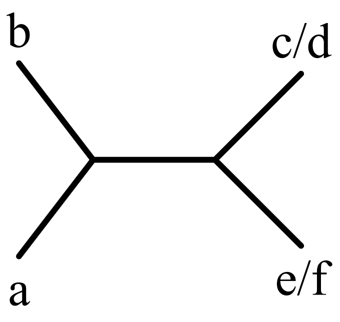
where k is the number of trees, n the number of taxa and p is the number of leaves that are removed from the initial taxon set. As the only MAST reference implementation for more than two trees in PAUP* [Swofford, 2003] is not freely available, we decided to implement a MAST algorithm in RAxML. For the sake of simplicity, we implemented the algorithm by Bryant [1996]. This algorithm computes the MAST for k binary rooted trees with n taxa in $\mathcal{O}(kn^3 + n^3)$. For unrooted trees, we have to apply this algorithm to all of the n possible rootings and thus the complexity increases by a factor of n. While it is remarkable that Berry and Nicolas manage to reduce this additional factor to $p+1$, we would like to point out that for large values of n and k the number of dropped leaves p is often only slightly smaller than n. Furthermore, Bryant's algorithm can be modified to return all possible MASTs. Unfortunately, Berry and Nicolas do not describe, how this can be achieved using their algorithm. While our implementation is definitely not optimal, we believe that it has been sufficiently optimized (also at a low technical level) to handle large datasets in reasonable time.

We will briefly outline the algorithm for rooted trees and provide some additional implementation details in Section A.1.

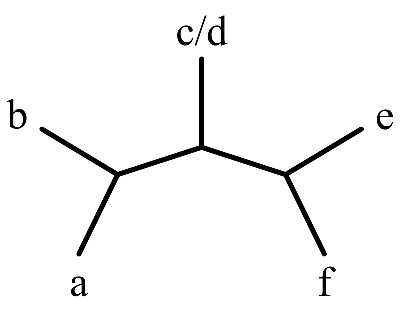
Initially, we extract the set of all *rooted triplets* C that occur in every input tree. In a rooted triplet $(a,(b,c))$, taxa b and c are related more closely to each other than to taxon a. The intersection of the rooted triplets of all input trees constitutes C.

The rooted triplets of C are the smallest instances of agreement subtrees for the given tree set. Given the triplets, we use these smallest units to compose larger agreement subtrees using a function mast(a,b) for taxa a and b. This function yields the taxon set of the MAST, such that the least common ancestor of taxa a and b is the root of this MAST. We define the function as:

$$\text{MAST}(a,b) = \begin{cases} a, & \text{if a = b,} \\ \text{mast}(a,x) \cup \text{mast}(b,y), & \text{else,} \end{cases} \tag{10}$$



(a) possible MASTs that are contained in all three bootstrap replicates of Figure 1.



(b) the MASTs of bootstrap tree A and B of Figure 1.

Figure 3: Two examples of MASTs. The slash signs depict alternative taxa assignments to different positions.

where $x \in \{x : ((a,x),b) \in C \wedge max(|\text{mast}(a,x)|)\}$ and $y \in \{y : ((b,y),a) \in C \wedge max(|\text{mast}(b,y)|)\}$. We can compute this function using a dynamic programming matrix. The largest taxon set returned by $\text{mast}(a,b)$ is the MAST for this specific rooting of the input trees. Finally, the MAST for the unrooted input trees is the largest taxon set returned by $\text{mast}(a,b)$ over all n rootings, where n is the number of taxa. If we want to compute all MASTs, we need to enumerate all possible values for x and y in each recursive step of the $\text{mast}(a,b)$ function.

3.2. A Naïve Pruning Algorithm

When we intend to optimize a consensus tree under criteria like the RIC or RBIC, the most straight-forward approach may be the following: we can prune each taxon (one at a time) from all bootstrap trees and assess the effect of pruning this taxon by computing the RIC/RBIC on the consensus tree. Thereby, we obtain a "rogueness" score for each taxon. We can then sort this list of scores and drop the taxon with the highest score. Because of its inherent simplicity, we denote this method the *naïve algorithm*.

This approach works well, if only a few rogue taxa have a comparatively large deleterious effect in different regions of the bootstrap trees. However, it may be problematic, if two rogues R_1 and R_2 occur in the same region of the tree.

For instance, R_1 and R_2 could wander as a so-called rogue clade through different positions in the bootstrap trees. This means, they assume various positions in the bootstrap trees, while R_1 always is a sister taxon of R_2 in every tree. In this case, dropping only R_1 or only R_2 has no effect on the RBIC. As the algorithm computes the RBIC scores only once during the first step, the RBIC score of R_1 does not improve after R_2 has been pruned (although if recomputed, the method would recommend pruning R_1).

Another potential shortcoming of the naïve algorithm is depicted in Figure 4. In this example, two rogues R and Q change their position in the two trees with an otherwise stable backbone a, b, c and d. If we use the naïve algorithm to optimize the RBIC of the strict consensus tree (a star tree), the algorithm will fail to recover the consensus bipartition $ab|cd$. The only way to recover this bipartition is by testing the effect of a dropset of size two (R, Q) on the RBIC score. Both examples show that the naïve algorithm is a greedy heuristic method that can easily mislead (although it is presently not clear how relevant the discussed examples are for real biological trees). Furthermore, this illustrates that, we can not expect the optimization landscape to have a trivial/smooth shape.

Another critical look at this approach reveals that one significant assumption/simplification is made that does not only apply to this algorithm though. When we prune taxon a from the bootstrap trees, we can compute the RBIC of the restricted consensus tree and obtain a score s_{pruned}. It is unclear, how strongly or weakly the scores s_{pruned} of pruned trees are correlated to the scores of consensus trees that were created without the candidate rogue taxon (s_{real}). As already mentioned, the addition of a taxon to a phylogenetic analysis can, in theory, lead to a maximally different tree [Cueto and Matsen, 2010]. Thus, if we wanted to attain absolute certainty for obtaining a globally maximal RBIC, we would have to execute a phylogenetic analysis for each element of the power set of the taxa under study. Clearly, this is computationally not feasible except for the smallest instances (small numbers of taxa). Nevertheless, the additional bipartitions that we recover using, for instance, the naïve algorithm are legitimate, since the underlying phylogenetic relationships were observed in the pruned and unpruned bootstrap trees.

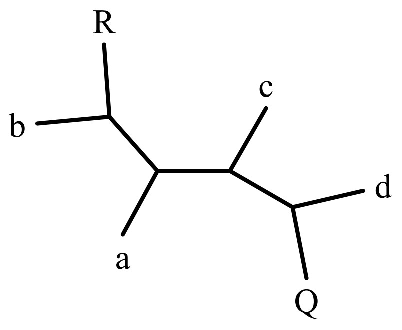
3.3. Maximum-Information Subtree Consensus

Conceptually, the *Maximum-Information Subtree Consensus* (MISC) [Pattengale et al., 2010a] tries to find a compromise between a “classic” consensus tree and a MAST. A consensus tree identifies all evolutionary statements (i.e., bipartitions) between all n taxa that occur in a certain number of trees. On the other side, the MAST only identifies statements that occur in all trees. However, these statements do not necessarily include all n taxa. The MISC represents a trade-off between these two criteria for summarizing tree collections: it is a consensus tree that does not need to contain all n taxa nor contain bipartitions that occur in all bootstrap trees. This property allows for maximizing of an optimality criterion such as the RIC or RBIC. Note that, the MISC depends on the actual definition for quantifying information that is used. Thus, the RIC-MISC may be expected to be different from the RBIC-MISC in most cases.

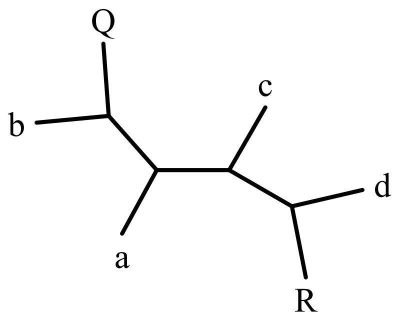
The MISC problem may be $\mathcal{NP}$-hard (see below). Thus, approximations are most likely essential to obtain a computationally feasible solution to this problem. Essentially, the naïve algorithm (see Section 3.2) is a simple approximation of the MISC with respect to the RBIC. As already mentioned, it is relatively easy to construct examples, where the naïve algorithm will fail to identify a interdependent combination of taxa that yield an improvement with respect to the optimality criterion (see Figure 4). Pattengale et al. [2010a] proposed an approximation to the MISC (with respect to the RIC), that strives to overcome the problems of the naïve algorithm.

For solving the instance of Figure 4, it is necessary to simultaneously prune taxa R and Q. An exhaustive algorithm could solve this instance by just generating all possible combinations of taxa (as drop set candidates) and measure the improvement with respect to the optimality criterion. The combinatorial explosion makes it impossible to apply this approach to real-world datasets. However, the number of possible pruning candidate sets can be significantly reduced, if we explicitly use the bipartition profile to generate of dropset candidates.

Dropset generation. In the example of Figure 4, the two bipartitions $\mathbf{A} = abR|cdQ$ and $\mathbf{B} = abQ|cdR$ form part of the bipartition profile. For deriving a general formula for generating dropset candidates from a bipartition set, we need the following definitions. Let us define $\mathbf{A}_1$ as the first partition of bipartition $\mathbf{A}$ and $\mathbf{A}_2$ as its second partition. This means that, $\mathbf{A}_1 = abR$ and $\mathbf{A}_2 = cdQ$. We denote the symmetric difference of two partitions as $\mathbf{A}_1$ and $\mathbf{B}_1$ as $\mathbf{A}_1 \Delta \mathbf{B}_1$ [Pattengale et al., 2010a, see]. Since a partition is a



(a) Bootstrap tree A



(b) Bootstrap tree B

Figure 4: Two bootstrap trees. Here the naïve algorithm fails to optimize the RBIC of a strict consensus tree.

set, for two sets x and y, the symmetric difference is defined as:

$$x \Delta y = (x \cup y) \setminus (x \cap y) \tag{11}$$

In other words, the symmetric difference is the set of taxa that is part of exactly one of the two partitions x and y (i.e., the taxa that are unique to one of the two bipartitions).

In order to recover the bipartition $\mathbf{C} = ab|cd$ for a strict consensus tree, we need to prune the dropset (R, Q) from the bootstrap trees. If we prune (R, Q) from bipartitions **A** and **B**, both bipartitions *merge* into bipartition **C**. As a direct consequence, the support of **C** increases and **C** becomes part of the strict consensus tree. For each pair of bipartitions $\mathbf{X} = \mathbf{X}_1|\mathbf{X}_2$ and $\mathbf{Y} = \mathbf{Y}_1|\mathbf{Y}_2$ there exist two dropsets, that induce merging bipartition **X** with bipartition **Y**. The two dropsets are [Pattengale et al., 2010a]:

$$(\mathbf{X}_1 \Delta \mathbf{Y}_1) \cup (\mathbf{X}_2 \Delta \mathbf{Y}_2) \text{ or } (\mathbf{X}_1 \Delta \mathbf{Y}_2) \cup (\mathbf{X}_2 \Delta \mathbf{Y}_1) \tag{12}$$

In other words, the dropset contains those taxa, that will lead both partitions of one bipartition to merge with both partitions of another bipartition. There exist two dropsets, because we have two alternatives for combining/merging the partitions. If we apply bipartitions **A** and **B** to Formula (13), then we get the “desired” dropset (R, Q) and the alternative dropset (a, b, c, d). For simple examples like the one discussed here, the dropsets with more taxa usually comprise the entire stable subset of the consensus tree. In this example, we can omit considering the alternative dropset (a, b, c, d). However in real-world datasets, there may be (a few) instances, where both possible dropsets represent “good” alternatives. Due to the symmetric nature of bipartitions, we can simplify Formula (13) to

$$(\mathbf{X}_1 \Delta \mathbf{Y}_1) \text{ or } (\mathbf{X}_1 \Delta \mathbf{Y}_2). \tag{13}$$

We can compute the MISC with respect to a consensus method and an optimality criterion, if we measure the impact of each possible dropset on the consensus tree. As we have seen in the example, this dropset approach is more powerful than the naïve algorithm that tries to improve the optimality criterion by pruning one taxon at a time.

Pathological Instances. Based on the prolegomena, it may however not suffice to only compute the dropsets for all pairs of bipartitions. There are cases, where it is necessary to merge three (or even more) bipartitions to recover a consensus tree bipartition. Consider bootstrap tree A (see Figure 5(a)), B (see Figure 5(b)) and C (see Figure 5(c)) without taxon T. Again we have the stable backbone consisting of taxa a, b, c and d. There are three rogue taxa R, S and Q, that assume a different position with respect to the stable backbone tree in each replicate tree. Here, it is not possible to merge any pair of bipartitions such that the resulting bipartition becomes part of a strict consensus tree. However, it is possible to recover the bipartition $ab|cd$, if the input tree set consists of only two of the three trees.

In the last example with three rogue taxa R, S and Q in the bootstrap trees A, B and C, we are still able to recover the bipartition $ab|cd$ by using a MR consensus tree (instead of a strict consensus), if we prune a dropset that was computed from a pair of bipartitions. If we drop such a pair (e.g., (R, Q)), then an additional bipartition will occur in two out of three pruned trees and thus becomes a consensus bipartition. To obtain an an example where $ab|cd$ can not be recovered by a MR consensus tree, we have to add another rogue taxon T and an additional bootstrap tree replicate (see Figure 5). The example is slightly artificial because the number of rogue taxa is identical to as the number of stable taxa (one would not expect a biologist to prune 50% of his taxon sample from a tree). Nevertheless, we use it to illustrate the potential problem. In this example, a consensus tree bipartition needs to occur in three of the four trees. Consider the four different bipartitions at the branches that connect the taxa a, b and a rogue taxon to the rest of the trees: pairs of bipartitions such as $abR|STcdQ$ (tree A) and $abS|TQcdR$ (tree B) lead to dropsets that comprise two rogues (S and R) in this example. However, this dropset increases the support of the pruned bipartitions only in two out of four trees. This is not sufficient for the pruned bipartition to be included in the MR consensus. Based on the examples discussed here, we can derive an algorithm for constructing a difficult bootstrap tree set for rogue taxon identification algorithms that rely on the dropset-based approach:

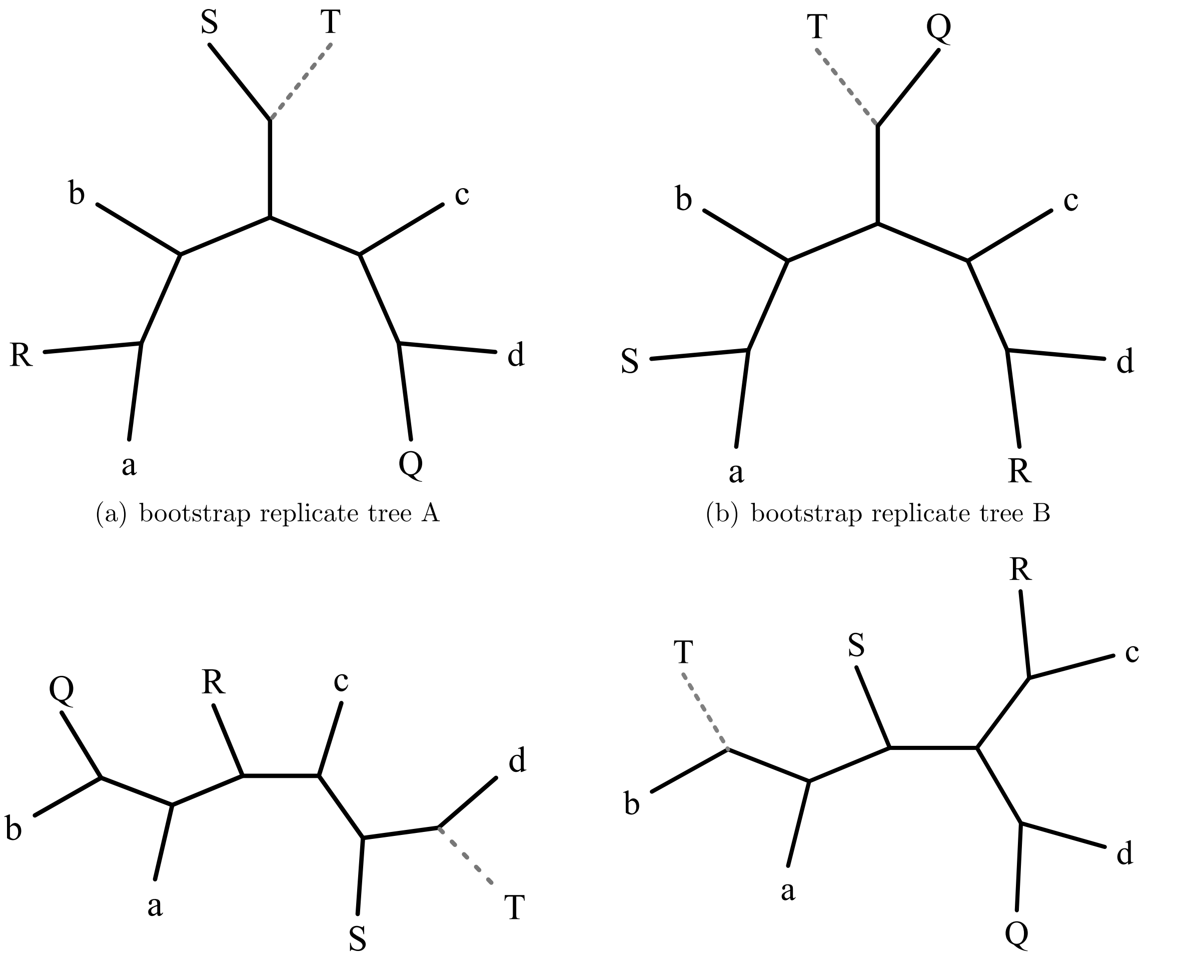
1. Start with a stable backbone tree as a basis for every bootstrap replicate tree.
2. Add a sufficient number of rogue taxa to a sufficient number of bootstrap replicates, such that
   a) each rogue taxon has another sister taxon in each tree and
   b) the bipartitions of the stable backbone are not part of the consensus tree (with respect to the consensus method used).

Note that, these above pathologic examples all contain a common subtree that occurs in every bootstrap tree, that is, such problem instances can be solved by the MAST algorithm.

These difficult cases show that, there are instances, where we need to prune dropsets which can only be found, if we attempt to merge three (or more) bipartitions. With the rotating rogue taxa algorithm, we can create instances, such that merging of an arbitrarily large number of bipartitions is necessary to gain bipartitions in consensi. This means that, a dropset-based exhaustive algorithm for the MISC needs to compute the dropsets for all combinations of arbitrarily large numbers of bipartitions. In other words, if $\mathcal{B}$ is the set of all bipartitions, we need to compute dropsets for each element in the power set of $\mathcal{B}$. Although, the dropset-based approach significantly reduces the number of candidates compared to enumerating all possible dropsets, this number is still exponential. Thus, we think that the MISC problem may be $\mathcal{NP}$-hard.

An Approximate RIC-MISC Algorithm. Pattengale's approximation computes dropsets for pairs of bipartitions and thus is in $\mathcal{O}(k^2)$, where k is the number of bipartitions in the bipartition profile. Given these dropsets, we can compute a consensus tree for each candidate dropset after pruning the respective taxa. Finally, we can assess the optimality of the pruned consensus tree to evaluate the impact of dropping the dropset. Concerning complexity, computing the consensus tree is not problematic. However, if a consensus tree is calculated for each dropset candidate, the implementation becomes too slow for practical purposes. This is the reason, why a RIC-specific approximation is used instead of an exact computation of the consensus tree.

The main goal of the RIC is to increase the resolution of the tree – a better RBIC is just a byproduct of the RIC optimization. Thus, the algorithm generates dropsets for all pairs of bipartitions that are not contained in the consensus tree. Analyzing dropsets generated from bipartitions pairs where at least one of the bipartitions is part of the consensus tree does not make sense for RIC optimization: pruning the dropset would either not change



(a) bootstrap replicate tree A

(b) bootstrap replicate tree B

(c) bootstrap replicate tree C

(d) bootstrap replicate tree D

Figure 5: Four bootstrap tree replicates. Trees A, B and C without taxon T represent a pathological case for a dropset-based algorithm that strives to recover bipartition $ab|cd$ for a strict consensus tree, if dropsets are generated from pairs of bipartitions. Trees A, B, C and D with taxon T make it impossible for the same kind of algorithm to recover an additional bipartition for a MR consensus tree.

the number of consensus bipartitions or would even remove one bipartition (in the case where both merging bipartitions are part of the consensus).

The algorithm maintains a function that maps each bipartitions to the set of trees in which the bipartitions occur. If we merge the bipartitions **A** and **B** by pruning their dropset, the merged bipartition is part of all trees that contained **A** or **B**. If the number of trees is greater than the threshold for the consensus tree method (e.g., 50% for the MR consensus tree) that is used, then, the merged bipartition is a consensus tree bipartition.

For each bipartition pair, the algorithm either stores the smaller dropset or both, if the number of taxa in both dropsets is equal. Finally, the dropsets are ranked according to their *impact* on the bipartition profile. The impact of a dropset is defined as the number of times, the dropset induced the merger of a bipartition pair minus the number of taxa the set contains. Thus, the impact of a dropset is used to approximate the RIC. The main idea of the approximation is the following: The explicit computation of the RIC for a pruned consensus tree is expensive. Instead, the algorithm keeps track of the relevant changes (merging of non-consensus tree bipartitions) that are induced by pruning a dropset.

The algorithm determines the dropset with maximum impact, that is, the dropset where we have to prune the fewest number of taxa in order to achieve the biggest increase of resolution in the pruned consensus tree. Next, the algorithm prunes all taxa in this dropset from the input tree collection, recomputes the bipartition profile and repeats the procedure described above by once again generating candidate dropsets from pairs of bipartitions. The iterative procedure stops, if there is no dropset left that has a positive impact (this means that the RIC can not be improved by dropping further taxa). This iterative approach is necessary, since taxon pruning can significantly influence the impact of other dropsets. We briefly outlined an analogous issue for the naïve algorithm in Section 3.2. Here, a re-assessment (using an iterative version of the algorithm) is more appropriate.

The approximation will yield sub-optimal results for the following configurations:

1. **Dropset generation from at least three bipartitions is required**: As discussed, there are instances that can only be solved optimally, if we generate dropsets from three or more bipartitions.

2. **Vanishing consensus tree bipartitions**: The algorithm does not keep track of merging events between consensus tree bipartitions. Pruning a dropset can reduce a consensus tree bipartition to a trivial bipartition or to a bipartition with an empty partition. All of these events lead to the loss of a consensus bipartition (which is not accounted for in the approximation).

3. **Complicated merging events**: There exist cases, where, for instance, one dropset induces a merger of three bipartitions into a single bipartition. In this case, the RIC approximation incorrectly assumes that the double merging event gives rise to two distinct consensus tree bipartitions. Instead, only one additional bipartition will be included in the consensus tree.

3.4. A RBIC-optimizing MISC

We modified Pattengale's algorithm for the optimization of the RBIC optimality criterion. Because of the high computational cost, we also limited this RBIC-optimizing MISC algorithm to only compute dropsets from pairs of bipartitions.

The merging of additional bipartitions is a comparatively rare event when dropsets are pruned. This is why the RIC of a pruned consensus tree is approximated well by Pattengale's algorithm. The RBIC strives to measure the quality of a consensus tree at a finer level and is thus more prone to inaccurate approximations. This is because, each dropset, when pruned, may slightly affect the support values of the bipartitions. As all changes to the bipartition profile are relevant for an RBIC-optimizing MISC algorithm, we can not apply the sorting optimization (described in Section A.3).

An algorithm that keeps track of all changes to a bipartition profile that are induced by dropset pruning exhibits a high space complexity (quadratic in the number of bipartitions). We would need to compute and store merging events between all pairs of bipartitions and check for vanishing bipartitions for each dropset. Therefore, we explicitly calculate pruned consensus trees for each dropset. This allows for an exact computation of the RBIC. As for the RIC approximation, this algorithm is iterative. We determine the dropset with maximal impact, prune it from the bipartition profile and repeat this process until the RBIC can not be further improved. The impact of a dropset is defined as the RBIC improvement (with respect to the current pruned consensus tree), divided by the number of taxa in the dropset.

In contrast, to the RIC-optimizing approximation, this version has linear memory requirements. However, time complexity increases. Thus, this algorithm can only be applied to small instances (e.g., 1000 trees with 200 taxa), but we may expect it to return the best approximation for a RBIC-MISC on relatively small input tree sets.

3.5. Taxonomic Instability

So far, the a posteriori methods we have presented all try to directly maximize some optimality criterion. If pruning of single taxon or a set of taxa leads to improvements, we assume that the taxon or taxa were unstable and can be considered as rogues. A different approach is to assess the instability of taxa with respect to some stability measure. Based on such a measure, one can then prune a specific set of instable taxa from the input trees. We could assume, that algorithms based on stability measures are able to better solve the aforementioned pathological cases and thus outperform direct optimization algorithms. Furthermore, an advantage of stability measures is that, they are not computed using a consensus tree method. As discussed in Section 2.4 this is a desirable feature.

A popular stability measure that is implement in the Mesquite tool suite [Maddison and Maddison, 2010] is the *taxonomic instability* measure:

$$taxInstab(i) = \sum_{(x,y),j \neq i} \frac{|d_{ijx} - d_{ijy}|}{(d_{ijx} + d_{ijy})^z} \tag{14}$$

In Equation (14) d_{ijk} denotes the unweighted patristic distance between taxa i and j in tree k. The patristic distance between i and j is defined as the sum of the branch lengths along the unique path in the tree between taxa i and j [see e.g. Farris, 1972]. The unweighted patristic distance is the number of nodes that lie on the path between two leaves. For the taxonomic instability measure, the unweighted patristic distance is deployed, as bootstrap trees sometimes do not contain branch lengths.

The parameter z can be used to increase the influence of distant evolutionary relationships (for large values) or close evolutionary relationships (for small values). For Mesquite the default value is $z := 2$. We use the default value for our experiments, since we assume that most rogue taxa wander around in rather small regions of the trees.

Computation of the taxonomic instability measure is expensive. As Formula (14) indicates, we need to iterate over all pairs of taxa (i and j) and all pairs of trees (x and y). Thus, computing the taxonomic instability for all taxa is in $\mathcal{O}(n^2k^2)$, where n is the number of taxa and k the number of bootstrap trees.

Note that, the taxonomic instability measure represents a relative instability measure. Its value depends on the number of trees and the number of taxa therein. The taxonomic instability of just one taxon is meaningless and does not provide information about the rogueness of a taxon.

While the taxonomic instability may be used to assess biological meaningful instability, it has also been used for rogue taxon identification [Thomson and Shaffer, 2010a]. Here, the taxonomic instability of all taxa is computed as a pre-filtering step. Then, Thomson and Shaffer tested, whether pruning taxa (starting with the most unstable one) improved bootstrap support values in the pruned consensus tree. In an attempt to reconstruct a particularly large phylogenetic tree (12,000 species and 210,000 sequences) a rogue taxon analyses based on the taxonomic instability was carried out [Thomson and Shaffer, 2010b]. Two different taxonomic instability cutoffs were applied: 5% and 10% of the least stable taxa were pruned from the dataset.

Here, we explore the utilization of taxonomic instability measure for maximizing our quality criterion (the RBIC). Initially, we determine the taxonomic instability of all taxa. Then, we prune the taxa from the input trees in descending order of their taxonomic instability. We compute the RBIC of the pruned consensus tree after each pruning step.

Initially, we assumed that algorithms using a stability measure may be able to solve instances that are pathological for direct optimization algorithms. For trees A, B and C of Figure 5 (without taxon T), this is indeed the case: The three rogue taxa R, Q and S have a higher taxonomic instability than taxa a, b, c and d. This means that, our algorithm using the taxonomic instability measure is capable of reconstructing the bipartition $ab|cd$ on this simple example. We discuss in Section 6.3, how this algorithm performs on real-world datasets.

3.6. Leaf Stability Index

The *leaf stability index* is a stability measure for rooted trees that was introduced by Thorley and Wilkinson [1999]. Similar to the MAST algorithm (see Section 3.1), it is based on rooted triplet relationships. The MAST algorithm only considers triplets that occur in all trees. Contrary to that, the leaf stability index is computed from the frequency of the triplets in the bootstrap trees. For three taxa a, b and c, there exist three rooted triplets (in Newick notation [Felsenstein et al., 1986]):

$$((a,b),c), \qquad ((a,c),b) \qquad \textit{and} \qquad ((b,c),a)$$

Let $\max_{abc}$,mid_{abc} and $\min_{abc}$ be the relative frequencies of these triplet relationships, such that $\max_{abc} \geq \text{mid}_{abc} \geq \min_{abc}$. The leaf stability index of a triplet is defined as the difference between the most frequent triplet $\max_{abc}$ and the second most frequent triplet mid_{abc}. Finally, we obtain the leaf stability index of a taxon a by computing the average leaf stability index of all triplets that contain a. Let n be the number of taxa, then the standard leaf stability index is:

$$\mathrm{LS}(a) = \mathrm{LS}_{DIF}(a) = \frac{\sum_{b=1}^{n-1}\sum_{c=1}^{n-2} \overbrace{\max_{abc} - \mathrm{mid}_{abc}}^{\text{triplet leaf stability index}}}{(n-1)\cdot(n-2)} \tag{15}$$

If there is a frequently occurring triplet for all triplets that contain a, then $\mathrm{LS}(a)$ converges to 100%. If there is a small difference between all $\max_{abc}$ and mid_{abc}, then $\mathrm{LS}(a)$ is minimal with a value of or close to 0%. This means, that a triplet abc is considered to be very unstable without respect to the lowest relative frequency $\min_{abc}$. For all values of $\max_{abc}$ with $33.\overline{3}\% \leq \max_{abc} \leq 50\%$, we obtain minimal leaf stability indices for the triplet abc (see Figure 6).

There are three other versions of the leaf stability index [Thorley, 2000]. We will discuss only two of them, since the one we omit has an unbounded maximum and can thus not be normalized to values between 0% and 100%. The standard leaf stability index of Equation (15) is referred to as LS_{DIF}, because it employs the discussed difference to assess the leaf stability of a triplet. We call LS_{DIF} the standard leaf stability index, because it is the only one that is explicitly introduced in Thorley's first study of this measure [Thorley and Wilkinson, 1999]. Furthermore, it is the only measure for which an up-to-date implementation is available [Smith and Dunn, 2008].

The first alternative is LS_{MAX} (see Figure 7), for which the leaf stability index of a triplet simply amounts to $\max_{abc}$, that is, the relative frequency of the most frequent triplet. Obviously, this measure emphasizes the importance that most bootstrap replicate trees agree on one of the tree possible triplets. Concerning the influence of the most frequent triplet $\max_{abc}$, LS_{MAX} is superior to LS_{DIF}, in which the term $\max_{abc}$ is only of relative importance. The downside of LS_{MAX} is that the frequency distribution among mid_{abc} and $\min_{abc}$ is completely ignored. For instance, if $\max_{abc} = 50\%$, then LS_{MAX} assumes the same value, irrespective of mid_{abc} which may range between 50% and 25%.

The third alternative is the entropic leaf stability index LS_{ENT}. The entropic leaf stability index for a triplet is defined as the entropy of the three possible triplets $\max_{abc}$, mid_{abc} and $\min_{abc}$. The concept of entropy seems to represent an appropriate choice in the context of phylogenetic stability. For instance, if all three triplets occur with the same relative frequency (33%), then the entropy is maximized and we obtain virtually no information about the evolutionary relationship of the three taxa. If only two alternative triplets are contained in the bootstrap trees (i.e., $\max_{abc} = \mathrm{mid}_{abc} = 50\%$), then LS_{ENT} unlike LS_{DIF} takes into account that this is a more stable configuration than in the scenario where all three triplets occur with the same frequency. As shown in Figure 8, the entropic leaf stability index is a particular smooth function. It assumes low values for all frequency distributions that do not contain at most one or two dominant triplets.

A clear advantage of the leaf stability index over the taxonomic instability measure is that, all three versions can be normalized to values between 100% and 0%. This allows for comparing the measures across different phylogenetic studies. Furthermore, it facilitates the empirical determination of a suitable cutoff threshold value that determines when a taxon should be dropped. For instance, Dunn et al. [2008] removed all taxa from their study with $\mathrm{LS}_{DIF} < 90\%$. Sperling et al. [2009] assessed the leaf stability indices for trees using all three versions of the leaf stability index. Instead of using a cutoff, it was their goal to prune significantly unstable taxa from the dataset. As the leaf stability indices are not normally distributed, taxa were considered to be rogues, when their leaf stability indices were below the first quartile in a box-plot representation.

We limited our focus to LS_{DIF} for the same reasons for which LS_{DIF} is considered as

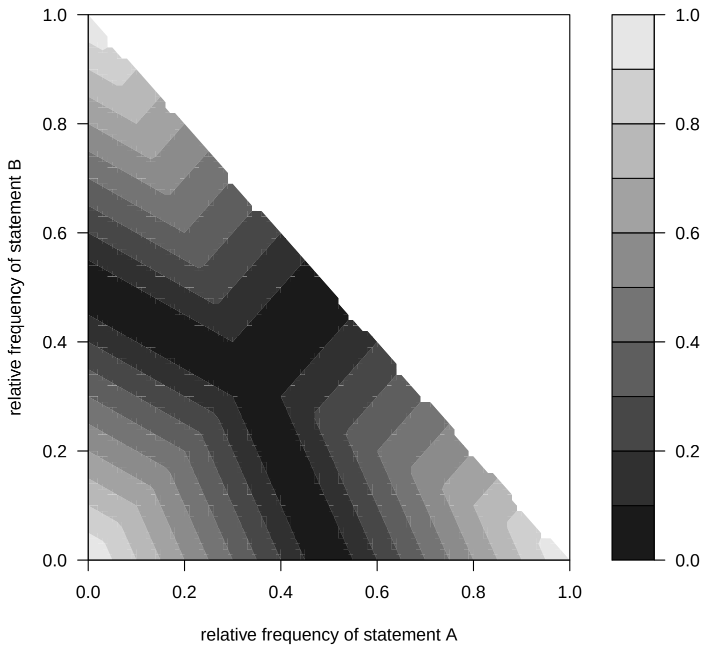


Figure 6: Difference version of the leaf stability index (LS_{DIF}). Values on the x- and y-axis represent the relative frequencies of two distinct triplets (the relative frequency of the third alternative is determined by the other two).

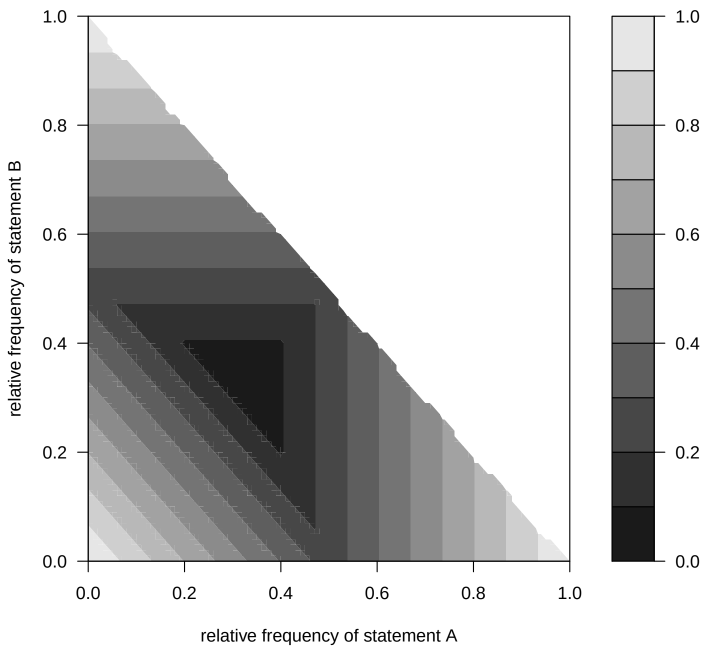


Figure 7: Maximum version of the leaf stability index (LS_{MAX}). Values on the x- and y-axis represent the relative frequencies of two distinct triplets (the relative frequency of the third alternative is determined by the other two).

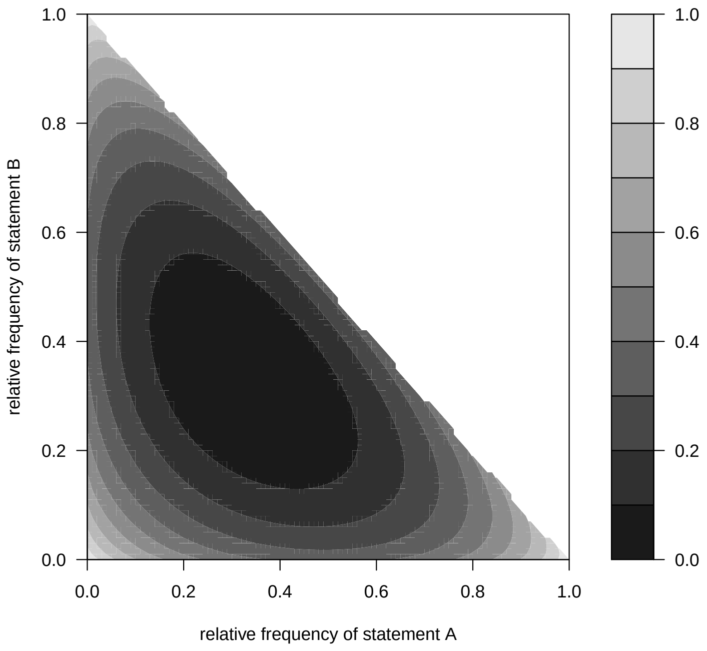


Figure 8: Entropy version of the leaf stability index (LS_{ENT}). Values on the x- and y-axis represent the relative frequencies of two distinct triplets (the relative frequency of the third alternative is determined by the other two).

the standard leaf stability index (see above). It is questionable, if the three versions of the leaf stability index behave differently on real-world datasets. In the supplementary material, Sperling et al. [2009] report that, the results for all three measures were consistent. While LS_{ENT} is appealing from a theoretical point of view, a pathological instance is known, where LS_{ENT} produces misleading results, while the other two versions do not. In this example [Wilkinson, 2006], two taxa W and X wander through a stable backbone tree. They assume various positions, but W is always a sister taxon to X in every bootstrap tree. Wilkinson constructed this pathological example to dismiss the leaf stability index as means for identification of stable reference taxa in the phylogenetic nomenclature (proposed by Lee [2005]). However, this example is also valid for the rogue taxon context, if W and X are interpreted as rogues.

Adapting leaf stability indices for unrooted trees is straight-forward. The analog for triplets in unrooted trees are 4-taxon statements or quartets [see Wilkinson, 2006]. The three possible quartets for taxa a, b, c and d are shown in Figure 9. Instead of deploying one of the aforementioned cutoffs for leaf stability indices, we rank the stability of taxa and prune taxa (one at a time) according to this ranking. All taxa that are not part of a taxon set for which the consensus tree maximizes the RBIC are considered as rogues.

3.7. Iterative Derivatives

A central aspect of Pattengale's RIC-optimizing algorithm is the idea to recompute dropsets in each iteration. Similarly, we can re-assess the taxonomic instability, leaf stability indices and the RBIC (in case of the naïve algorithm) after each pruning step. In other words, we identify the most instable taxon that most improves the RBIC most

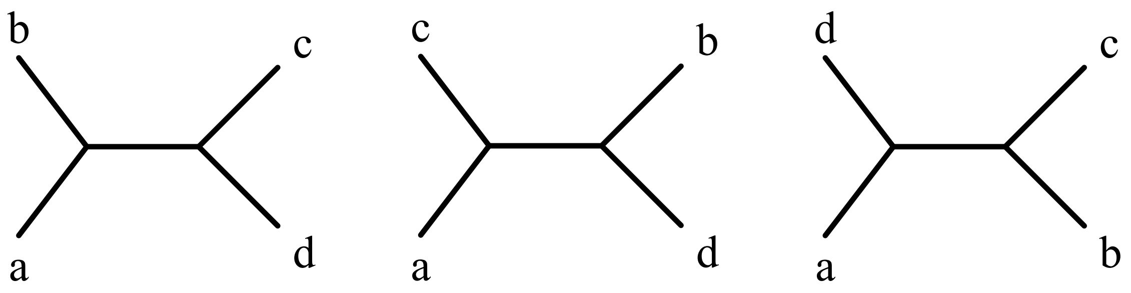


Figure 9: The three possible 4-taxon statements (quartets) for taxa a, b, c and d.

and prune it from the set of bootstrap trees. Then, we recompute the stability measure (RBIC) and determine the next taxon to prune.

We may expect that iterative versions of the taxonomic instability, the leaf stability index and the naïve algorithm always perform at least as well as the monolithic versions that rely on a one-time initial computation of the measures. In the case of leaf stability indices, the iterative version is more accurate: if we decided to prune taxon a in the last iteration, then triplets containing taxon a are not considered any more for computing leaf stability indices in the next iteration. Analogously, distances to already pruned taxa, do not influence the recomputed taxonomic instabilities. For the naïve algorithm, we already noted that a re-assessment may be advantageous (see the first example in Section 3.2). On the downside, we add a run time factor of n (the number of taxa) to each iteration of the algorithms. This renders large datasets (e.g., 1000 trees with 1000 taxa) hard to analyze for the iterative versions of the algorithms.

4. A Priori Methods for Rogue Taxa Identification

For all a posteriori methods described in Section 3, we initially need to compute a sufficient collection of bootstrap replicate trees. While there exist fast bootstrapping implementations [Stamatakis et al., 2008], a bootstrap analysis is still computationally expensive. When we have identified a set of rogue taxa using an a posteriori technique, we then may want to prune the rogue taxa from the alignment, recompute a bootstrap tree collection and repeat the entire analysis procedure again. This means that the initial bootstrap tree collection was solely created for the purpose of rogue taxa identification and is otherwise useless. Methods which determine rogues taxa before the actual post-analysis (bootstrap analysis) may be advantageous, since then we can exclude rogue taxa that influence the post-analysis beforehand. We call those methods *a priori methods*, as they strive to identify rogue taxa prior to executing a bootstrap analysis.

Pre-scanning for rogue taxa solely using information from the multiple sequence is difficult. While potential rogues may be identified by analyzing the alignment (e.g., taxa with many gaps or missing data for several genes). The impact of potentially deleterious effects on the output of a phylogenetic inference method is hard to predict.

Instead, our a priori method is applied to a later stage of phylogenetic inference, where the best-known Maximum Likelihood tree has already been inferred. For this purpose, we modified the *Evolutionary Placement Algorithm* (EPA) [Berger and Stamatakis, 2009] for rogue taxa identification. The EPA is deployed for phylogenetic inference of large sets of short sequence reads that are produced by NGS (next generation sequencing) techniques. For each of these query sequences, the algorithm strives to find a biological meaningful placement on a given best-known ML tree. Using a reference alignment (from which the best-known ML tree was inferred) the EPA computes the likelihood score for each placement of a query sequence into the tree. If we normalize the likelihoods with the sum of the likelihoods of the placements, we obtain *likelihood weights* [see Strimmer and Rambaut, 2002]. The sum of all likelihood weights is 1.0. A likelihood weight (lw) is interpreted as the probability, that a specific placement is correct given the best-known ML tree and the reference alignment.

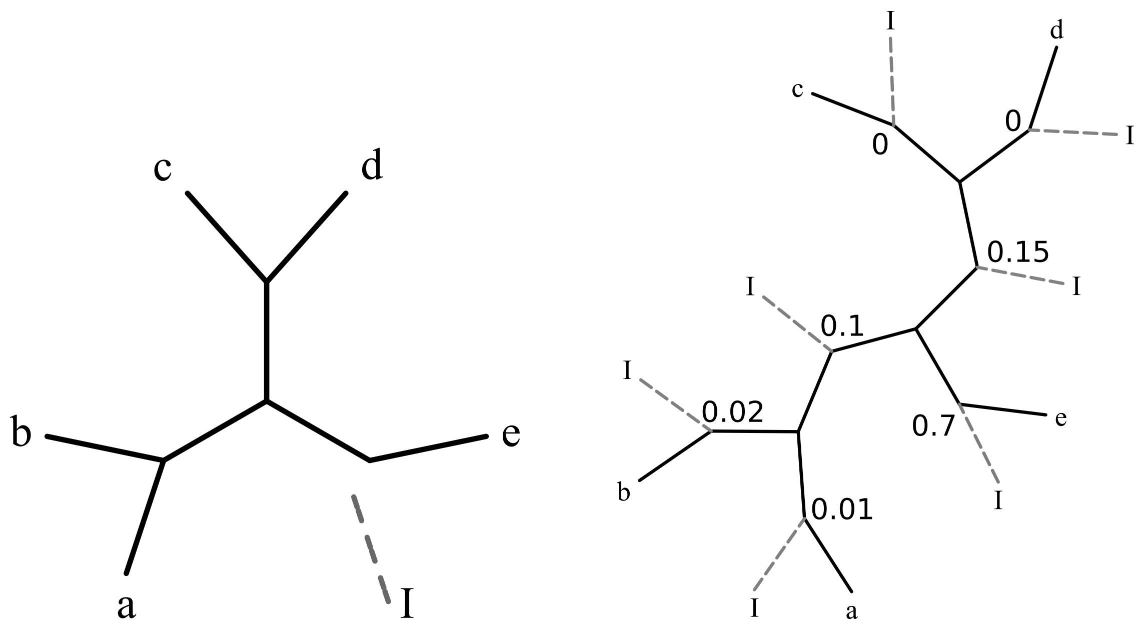
Likelihood weights are the basis of all a priori methods, we discuss in this Section. For obtaining a "rogueness" score of a taxon, we prune taxon I from the best-known ML tree (see 10(a)). Then, we compute the likelihood weights for each placement of that specific taxon in the tree. If our initial taxon set comprises n taxa, then I can be placed as a sister taxon to all remaining $n-1$ taxa. Furthermore, there are $n-1-3$ placements on inner branches to evaluate. Overall, $2n-5$ likelihood weights are computed. Figure 10(b) illustrates all possible placements for a best-known ML tree with 6 taxa. In this example, the placement of taxon I as a sister taxon to taxon e is highly likely and other placements close to e are more likely than distant placements (e.g., near taxa a or d).

4.1. Measures for Likelihood Weight Distribution Among Placements

The following measures deploy the distribution of likelihood weights among placements as a means to identify potential candidate rogue taxa.

4.1.1. Expected Distance between Placement Locations

The *Expected Distance between Placement Locations* (EDPL) is a measure for likelihood weights that was proposed by Matsen et al. [2010]. As the name suggests, this measure



(a) taxon I is pruned (b) likelihood weights for placements

Figure 10: Determination of likelihood weights for placements of taxon I. **Left**: Taxon I is pruned from the best-known Maximum Likelihood tree. **Right**: Various placements for taxon I are assessed. The values that are associated to the placements represent the likelihood weight of a specific placement.

accounts for the distance between placements. Unlike the taxonomic instability measure, the weighted patristic distance (see Section 3.5) is used for the EDPL. The EDPL for taxon a is defined as:

$$\mathrm{EDPL}(a) = \sum_{(i,j)} \frac{lw_i \cdot lw_j \cdot d_{ij}}{L}, \tag{16}$$

where lw_i and lw_j are the likelihood weights of placements i and j, d_{ij} is the distance between placements i and j and L is the total tree length, that is the sum of all branch lengths in the pruned tree.

The EDPL can be used to quantify the placement uncertainty of a taxon in the tree. Rogue taxa have a deleterious effect on bootstrap results as they assume various distinct positions in the bootstrap tree. Thus, a rogue will most likely be placed in various branches of the ML tree. Hence, likelihood weights of a rogue will presumably be distributed across different branches in the ML tree. If this is the case, we will obtain a high EDPL score. Thus, we can deploy the EDPL for predicting the rogueness of a taxon. If there is only one highly likely placement for a taxon (i.e., a single likelihood weight is close to 1.0), then the EDPL score will be almost 0.0 and we may expect the taxon to be phylogenetically stable. Thus, our EDPL-based algorithm for rogue taxa identification is analogous to the algorithms that use stability measures presented in Section 3: initially we compute an EDPL for each taxon, and then prune single taxa ordered by their EDPL scores from the bootstrap tree collection and assess impact on the RBIC. Note that, the bootstrap tree collection is just used to assess the results of the a priori method and deployed to identify rogues.

From an evolutionary perspective, the weighted patristic distance is to be preferred over the unweighted patristic distance as a distance measure for the EDPL. Let us consider a set of taxa whose members are all closely related to each other, that is, they form a subtree/clade T with short branch lengths. If the likelihood weights for the placements of a taxon a are equally distributed among the branches in subtree T, then there is a high placement uncertainty. However, we may conclude with high confidence, that taxon a can be placed into this clade T. In contrast to this, we may have only two placements for taxon a with large likelihood weights that are situated in two evolutionary distant clades (i.e., situated at a large weighted patristic distance from each other). While the unweighted distance between both placements may not necessarily be large, the placement is highly uncertain or ambiguous for that matter from a biological point of view. If we calculate the EDPL using the weighted patristic distance, we will obtain a higher EDPL for the second scenario. An EDPL score based on the unweighted patristic distance is higher in the first case, where many likely placements are situated in a clade with short branch lengths. Thus, for biologists an EDPL with weighted distances better represents placement uncertainty than an EDPL score using unweighted distances.

4.1.2. Number of Likely Placements

An alternative measure to assess the distribution of the likelihood weights among placements, is the number of likely placements. Obviously, it is highly subjective, when a placement can be considered likely. By definition, no likelihood weight has a value of 0. Thus, we decided to omit all placements with a likelihood weight threshold of $< 0.1\%$ as being unlikely and count the remaining placements. We experimentally assessed various cutoff thresholds. The value of 0.1% represents a reasonable choice, since larger cutoff values tend to become in-discriminative because there are many taxa that have the same number of likelihood weights $> 10\%$. On the other side, if the cutoff is chosen significantly smaller than 0.1% then too few placements are below the threshold and no discrimination is achieved.

Thus, we may expect that taxa with an uncertain placement (potential rogue taxa) have a high number of equally likely placements. On the other hand, taxa that are expected to be stable in a bootstrap analysis, should only have one (or a few neighboring) highly likely placement.

4.1.3. Standard Deviation of Likelihood Weights

As a third measure, we deployed the standard deviation of likelihood weights as means for a priori rogue taxa identification. Given the likelihood weights lw_i for the $2n-5$ placements of a specific taxon (with n being the number of taxa in the unpruned ML tree) and their mean μ, this score for taxon a is:

$$\mathrm{sd}_{\mathrm{place}}(a) = \sqrt{\sum_{i=1}^{2n-5} \frac{(lw_i - \mu)^2}{2n-5}} \tag{17}$$

This measure takes into account if likelihood weights are distributed evenly among a number of possible placements. The measure $\mathrm{sd}_{\mathrm{place}}$ has a minimum of 0, if all placements have the same likelihood weight and is maximal, if there is only one highly likely placement with a likelihood weight near 100%.

4.2. Measures for Likelihood Weight Distribution Among Samples

Our hypothesis is that, likelihood weights of placements for taxa in the best-known ML tree are correlated with the “rogue potential” of taxa in a bootstrap analysis. This is the basis for deploying the measures in Section 4.1 to identify rogue taxa without actually conducting a bootstrap analysis. Likelihood weights are computed from the original (non-bootstrapped) reference alignment. However, the bootstrap replicate trees we use in phylogenetic post-analysis are not inferred from the original alignment, but from bootstrapped alignment replicates. To create a bootstrap alignment replicate, we draw columns from the original alignment at random with replacement until our replicate alignment has as many alignment sites as the original alignment.

We can try to adopt this sampling procedure for the placement algorithm used to identify rogues in order to emulate the bootstrap procedure. As discussed in Section 4, likelihood weights are computed from the likelihood score obtained for the placement of a taxon into a specific position/branch of the tree. For computing the likelihood of a tree (with a specific placement of the taxon that is scored) given the alignment, we actually calculate the logarithmic likelihood (*log-likelihood*) for each site (column) of the alignment for numerical reasons. Thus, the overall log-likelihood score of the placement is the sum of the per-site log-likelihood scores. Analogous to bootstrap analyses, we can re-weight the per-site log likelihoods. This procedure allows to rapidly calculate the likelihood weights for placements using a bootstrap-like-procedure. However, this is just an approximation, since we still assess the likelihood weights for placements of a taxon in the best-known ML tree. We have to expect bootstrap trees to be topologically different from the best known ML tree. Moreover, all branch lengths and model parameters normally need to be re-estimated for each bootstrap replicate, which is not done using our re-sampling approach to save time. We just re-sample log-likelihood scores at the insertion position of the specific taxon, keeping the model parameters, tree topology, and branch lengths of the best-known ML tree fixed.

Using this approach to approximate the bootstrap, we obtain a matrix of likelihood weights that can be deployed for a priori rogue taxa identification:

$$
j \text{ samples} \left\{ \overbrace{\begin{array}{ccccc}
lw_{1,1} & lw_{1,2} & \dots & lw_{1,2n-6} & lw_{1,2n-5} \\
lw_{2,1} & lw_{2,2} & \dots & lw_{2,2n-6} & lw_{2,2n-5} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
lw_{j-1,1} & lw_{j-1,2} & \dots & lw_{j-1,2n-6} & lw_{j-1,2n-5} \\
lw_{j,1} & lw_{j,2} & \dots & lw_{j,2n-6} & lw_{j,2n-5}
\end{array}}^{2n-5 \text{ placements}} \right.
$$

Figure 11: Likelihood weight matrix for $2n-5$ placements of a taxa. Likelihood weights have been computed for i re-samplings of the reference alignment.

The aforementioned measures for likelihood weight distribution among placements (see Section 4.1) can easily be modified/adapted to placement score samples. We simply compute the measure for each sample and then take the average over all samples.

4.2.1. Number of Distinct Most Likely Placement Locations

A simple measure that can be derived from a likelihood weight matrix as depicted in Figure 11 with re-sampling is the number of distinct most likely placement locations (branches). For each sample we only consider the placement with the highest log-likelihood score. If the position with highest log-likelihood varies substantially among samples, we may expect that the taxon under consideration assumes many different positions in the bootstrap tree collection. Taxa which just exhibit a few distinct (in the best case 1) placements for different samples, are expected to have stable positions in the bootstrap trees.

4.2.2. Standard Deviation of Likelihood Weights Among Samples

The number of distinct most likely placement locations reduces the information in the likelihood weight matrix (see Figure 11) to just one likelihood weight per sample. A measure that takes all likelihood weights in the matrix into account is the standard deviation of likelihood weights among samples. For this measure, we average over the standard deviations of the likelihood weights for a specific placement among different samples. For the example matrix, this means, that we average over the standard deviations of the columns (unlike for the measure in Section 4.1.3, where we average over the standard deviations of the rows). For a taxon a the measure is defined as:

$$\mathrm{sd}_{\mathrm{sample}}(a) = \frac{1}{2n-5} \sum_{j=1}^{2n-5} \sqrt{\frac{\sum_{i=1}^{m} (lw_{ij} - \mu_i)^2}{k}}, \tag{18}$$

where n is the number of taxa, lw_{ij} is the likelihood weight of placement j in sample i, μ_i is the mean of likelihood weights of placement j and m is the number of score samples.

In this measure the contribution of the columns in the likelihood matrix to the overall score is implicitly weighted. Columns that have small likelihood weights in all rows barely influence the measure. This is usually the case for most columns. On the other side, columns that contain at least one sample with a high likelihood weight (and small likelihood weights otherwise) have a strong impact on the score. For taxa that are placed into the same branch for all samples with high confidence, we expect this score to be small (with a minimum of 0). This measure will produce misleading results for many equally likely placements when the likelihood does not vary between samples.

4.2.3. Distance-Weighted Likelihood Weight Products

The most obvious disadvantage of using the standard deviation of likelihood weights among samples is that, this measure does not directly consider the distribution of likelihood weights among placements. This is why, we can easily construct an example, for which this measure will produce a misleading result (see end of Section 4.2.2). We will derive a measure that extends the EDPL, such that it accounts for the standard deviation of the rows in the likelihood weight matrix on the one hand and for the standard deviation of the columns on the other hand. We call this measure *distance-weighted likelihood weight products* (dwlwp) and define it for taxon a as:

$$\mathrm{dwlwp}(a) = \frac{1}{L} \sum_{i=1}^{2n-5} \sum_{l=i+1}^{2n-5} \sum_{\substack{(j,k) \\ j \neq k}} d_{il} \cdot lw_{ij} \cdot lw_{lk}, \tag{19}$$

where n is the number of taxa, j and k are pairs of different samples, lw_{ij} is the likelihood weight of placement i in sample j and L is the tree length of the pruned tree. The dwlwp measure is similar to the EDPL, since we sum up products of likelihood weights of placements that are weighted by the distance between corresponding placements. While the EDPL considers the likelihood weights of one sample at a time, we only use likelihood weights from different samples for the dwlwp. If a taxon can be placed with confidence close to 100% into the same branch of the tree for each score sample, then the measure converges to 0.0. If the placement branch with a high or highest likelihood weight changes for each sample, then the corresponding summands assume large values and dwlwp indicates that the taxon can not be placed with high confidence for different score samples. Finally, when likelihood weights are uniformly distributed across placements (the pathological case for $\text{sd}_{\text{sample}}$ in Section 4.2.2), then all summands in the dwlwp assume values > 0 and the dwlwp indicates that the placement of this taxon is unstable.

In terms of computational complexity, this is the most expensive of the measures, discussed so far. Neglecting the complexity for calculation the likelihood weight matrix, computing the measure requires $O(n^3k^2)$ time and a distance matrix comprising $\frac{(2n-6)^2}{2}$ distance values for each taxon in the best-known ML tree.

4.3. Iterative A Priori Algorithms

The basic principle for algorithms using any of the measures discussed in Sections 4.1 and 4.2 is as follows. Initially, we compute the respective rogueness measure for each taxon. For assessing the performance of the measures, we can prune the taxa one-by-one from the bootstrap trees by order of their instability as obtained by the respective measure we want to assess. For each pruning step, we compute the RBIC of the pruned bootstrap trees. Thus, we can compare the performance of those measures to each other as well as to the a posteriori methods (see Section 3).

If we want to use the measures presented here as true a priori methods for rogue taxa identification (without using a set of bootstrap trees), we need to come up with a criterion, to determine when a taxon is considered as a rogue taxon and when not based on its rogueness score. This stopping criterion can be based upon the measure itself by using a cutoff threshold. Another option is to deploy SH-like support values derived from *(approximate) Likelihood Ratio Tests* [as discussed in Anisimova and Gascuel, 2006]. SH-like supports are a fast alternative to computing bootstrap support values that are based on statistical likelihood tests and on a given, best-known ML tree with a corresponding original multiple sequence alignment. We tested, if these support values can be used as alternative to the support values we obtain from a pruned collection of bootstrap trees.

Furthermore, we can evaluate the topological changes the best-known ML trees undergoes, when rogue taxa are pruned, using the Robinson-Foulds (RF) metric [Robinson and Foulds, 1981]. The RF metric is a distance measure for phylogenetic trees. The RF metric of two trees A and B is defined as the number of elements in the symmetric difference (see Section 3.3) between the bipartitions contained in A and the bipartitions contained in B.

Both stopping criteria only work in the context of iterative a priori algorithms. In each iteration, we compute one of the placement uncertainty measures for each taxon. Then, we prune the taxon with the most unfavorable score according to our measure. The taxon is pruned from the best-known ML tree (yielding T_{pruned}) and the reference alignment. Then, we re-compute the best-known ML tree T'_{opt} from the pruned alignment.

The pruned alignment and the new (pruned) best-known ML tree are used in the next iteration for re-computation of the placement uncertainty measure.

We can compute the RF distance between the pruned ML tree T_{pruned} and the recomputed ML tree T'_{opt} for assessing the impact of pruning the candidate rogue taxon from the tree topology and alignment. Our hypothesis is that, pruning highly deleterious rogue taxa will have a high impact on respective tree topologies. In contrast to this, when we prune stable taxa, the tree topology should not change anymore. Thus, the RF metric may be deployed as a stopping criterion to decide when to stop pruning taxa from the tree.

After each pruning step, we can compute SH-like support values from the pruned alignment and the re-computed best-known likelihood tree T'_{opt}. We expect SH-like support values to behave in a similar way as bootstrap support values. Thus, we can compute the RBIC of SH-like support values and expect this SH-like support value-based RBIC to increase until all rogue taxa have been pruned. When the RBIC of the SH-like supports decreases, the algorithm stops.

We expect iterative *a priori* algorithms to perform at least as well as their monolithic counterparts that are just applied once to the full tree and alignment to compute the respective measures. As for the iterative versions of the *a posteriori* methods, re-computing the best-known ML tree and then the respective measure after each pruning step, ensures that the removed rogue will not influence/bias the measure for other taxa. However, this potential improvement comes at a high computational cost. Computing a new ML tree each round (after each pruning step) is time-consuming. Furthermore, by recomputing the measure for all taxa at each iteration, we add a factor of n (number of taxa) to the complexity of the measure calculations.

The performance of the iterative algorithm and the stopping criteria are discussed in Section 6.4.2.

5. Datasets

In Sections 3 and 4 we discussed the performance of rogue identification methods on small instances and provided some pathological examples that are hard to solve for specific methods. We complement our study with a detailed performance assessment of the proposed methods using real-world datasets.

For all input alignments, we initially compute the best-known ML tree and 1000 bootstrap replicate trees, under the GTR+CAT approximation of rate heterogeneity [see Stamatakis, 2006b] for DNA alignments. We use the Dayhoff substitution matrix for protein alignments. If the number of bootstrap trees is too small, we risk that our post-analysis is not representative. A collection of 1000 bootstrap trees can be considered as being sufficient for the number of taxa in the datasets under study [Pattengale et al., 2010b]. Thus, we can safely assume that a taxon does not exhibit instable behavior due to random effects that are induced by insufficient sampling, but that this taxon is in fact a rogue taxon.

We used datasets with 125 up to 7764 taxa that were previously examined in other contexts (such as Pattengale et al. [2010b]). The datasets are available at `http://lcbb.epfl.ch/BS.tar.bz2`. We used 1000 bootstrap trees from the large bootstrap tree collections and computed 1000 bootstrap trees for the dataset with 6718 and 7764 taxa (the largest datasets in terms of number of taxa examined in this thesis). Those two datasets are representative samples of large single-gene DNA alignments.

For these 19 no information about the presence of potential rogue taxa was available. Thus, we asked the RAxML user community to provide datasets that are suspicious for rogues via the RAxML mailing list. We received 6 single-gene and 11 multi-gene alignments (see Figure 12). For the multi-gene datasets, we executed partitioned analyses, that is, we used RAxML to estimate a distinct set of evolutionary model parameters for each partition/gene as provided by the biologists.

Four of the datasets (with 53, 88, 117, and 451 taxa) are amino acid alignments and one dataset comprising 143 taxa consists of both amino acid and DNA sequences. The remaining datasets are alignments of DNA sequences.

Figure 12 shows statistics about the number of partitions for partitioned datasets and the number of taxa per partition for which no sequence data is available. It is likely that, datasets with a high number of partitions (e.g., the datasets with 117, 53, or 128 taxa in Figure 12) are prone to rogue taxa, since the different partitions (genes) may contain ambiguous or contradictory phylogenetic signals (gene tree species tree problem).

We already mentioned missing data (unavailability of sequence data for a specific taxon and gene in multi-gene alignments) as another potential reason for rogue taxa. Figure 13 shows the proportion of gaps and missing data per alignment. With more than 70% undetermined/missing characters in the datasets with 424, 404, 994 and 141 taxa, we may expect more rogue taxa to occur in these datasets. Also, a high number of missing taxa per partition (gene) may be indicative for rogues in the partitioned multi-gene alignments (see Figure 12).

Currently, some of the datasets examined here are used in ongoing unpublished research projects and are hence confidential. For the published datasets, we briefly provide some details about their provenance and species they contain.

In the dataset with 24 taxa mitochondrial genome sequences of ratites (a group of large flightless birds such as ostrich and emu) were examined [Phillips et al., 2009]. The dataset with 52 taxa was created to reassessment the phylogenetic position of a downy

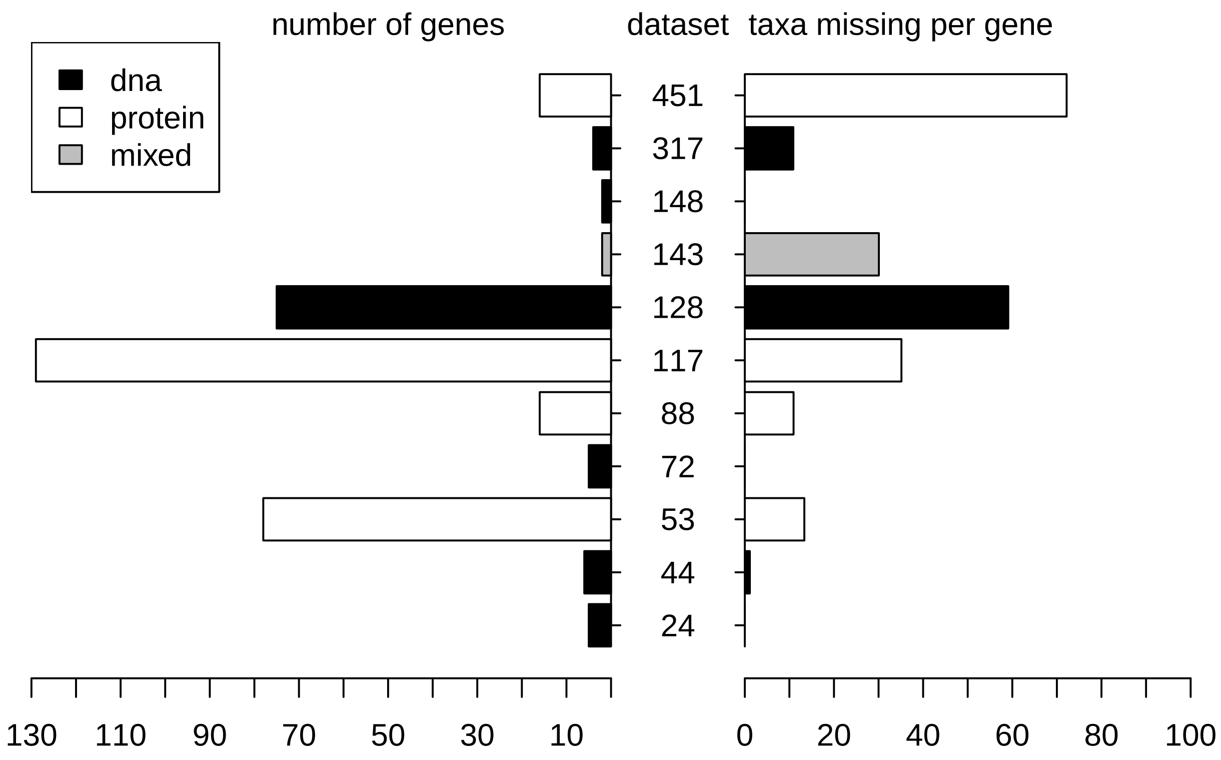


Figure 12: Left: Number of distinct genes/sequences used in partitioned datasets. **Right**: Number of taxa missing per gene/sequence. Colors of bars indicate the type of the sequences: DNA (black), protein (white) or both (grey).

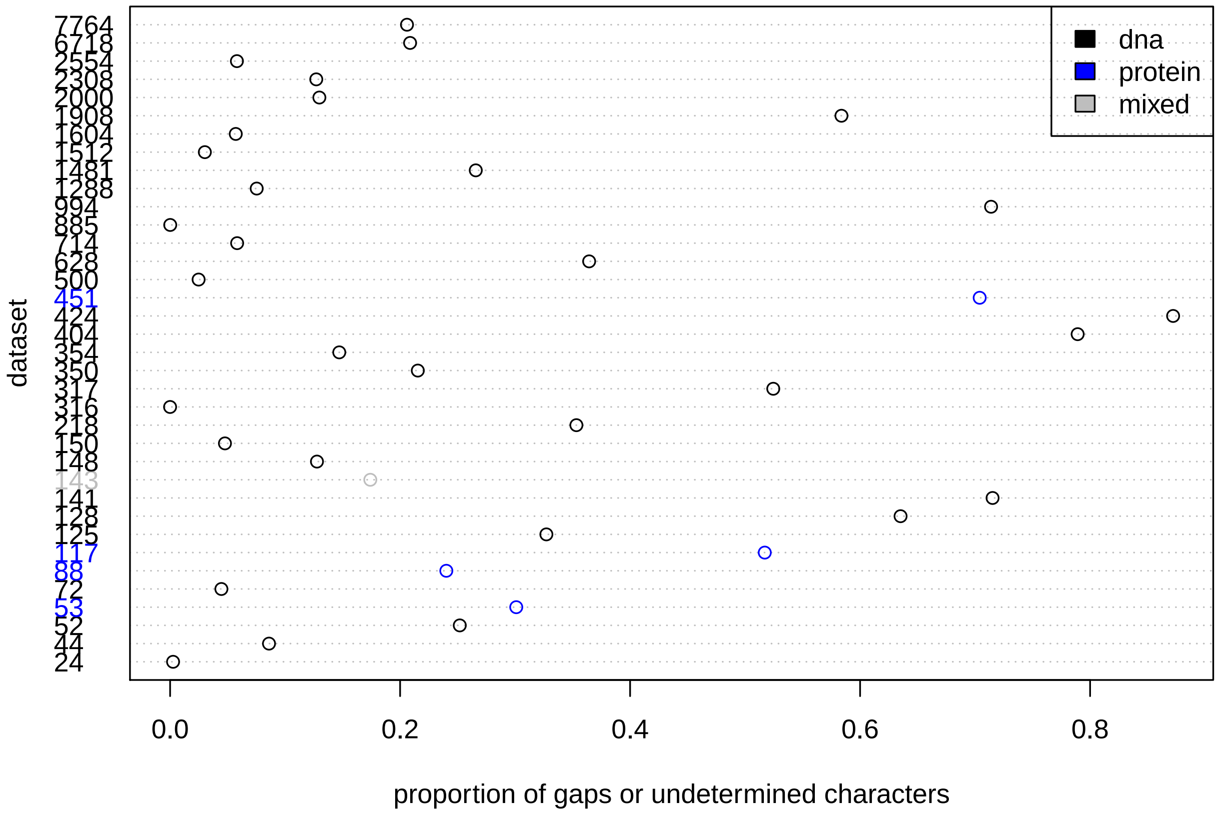


Figure 13: Proportion of gaps or undetermined characters for all datasets under study. Colors indicate the three alignment types: DNA (black) or protein (blue) sequences or sequences of both types (grey).

mildew pathogen (*Bremia graminicola*). Bootstrap support values could be improved in this study after pruning a rogue taxon [Thines et al., 2006]. The dataset with 72 taxa is from a study which combines mitochondrial, nuclear ribosomal and nuclear protein coding DNA sequences of Teleostei [Chen et al., 2003]. Sequences for the datasets comprising 88 and 451 taxa were sampled from diverse eucaryotic organisms [Parfrey et al., 2010]. A well-resolved tree could be established without removing rapidly evolving genes or taxa (i.e., potential rogue taxa). The dataset with 117 taxa compiled by [Meusemann et al., 2010], comprises phylogenomic sequences of Arthropods. In the dataset with 141 taxa the species under study are Amoebae [Shadwick et al., 2009]. The dataset with 148 taxa also examines phylogenetic relationships among Arthropods [Von Reumont et al., 2009]. 350 taxa of moths and butterflies were analyzed by [Mutanen et al., 2010]. A study by [Hodkinson and Lutzoni, 2009] is based on 885 sequences of lichen-associated bacteria.

6. Results And Discussion

In this Section, we assess and discuss various aspects concerning the performance and behavior of the a posteriori and a priori methods we have introduced.

6.1. Optimization of the RIC-MISC Implementation

As discussed in Section 3.3 and in Section A.3, we implemented the RIC-MISC approximation [Pattengale et al., 2010a] in RAxML and optimized the implementation for runtime and memory efficiency. We tested the scalability of the implementation on a Sun x4440 server (24 AMD cores, 128 GB RAM) and on a Sun x4600 server (32 AMD cores, 64 GB RAM).

For the RIC-MISC approximation, we need to store a number of dropset candidates that is quadratic in the number of bipartitions. After generation from a pair of bipartitions, a dropset is internally represented as a bit vector. Then, the bit vector is converted to an index list. This memory efficient representation of dropsets allows for handling bootstrap tree collections with up to 10,000 trees and 2,554 taxa. When using a strict consensus threshold, our implementation scales up to 672 trees with 37,831 taxa. Datasets of this size may be specifically prone to rogue taxa: for 37,814 taxa the algorithm recovers 208 additional strict consensus bipartitions by dropping just 13 taxa.

We also tried to compute the RIC-MISC of this large dataset with respect to the majority rule consensus. However, we experienced that main memory becomes limiting factor: for the majority rule consensus threshold, the memory requirements for storing the dropsets exceeded the 128 GB of main memory available on the testing system. Thus, not a single iteration of the algorithm could be completed.

Furthermore, we applied this implementation successfully for approximating the RIC-MISC for datasets with 1,000 trees and up to 7,764 taxa. Hence, we think that our implementation suffices for most current-day analyses. Note that, the test dataset with more than 37,000 taxa can be considered as being an exceptionally large dataset.

Concerning runtime performance, our implementation benefits from the low-level technical optimizations for handling bipartitions in RAxML . Furthermore, a shortcut that is based on sorting the bipartitions (described in detail in Section A.3) allows us to omit the evaluation of a large number of dropset candidates. Table 2 shows execution times for datasets with 10,000 trees that comprise between 150 and 2,554 taxa. We measured the overall execution times for the complete algorithm and the dropset generation phase separately, with and without the aforementioned optimization enabled. In the unoptimized case, the dropset generation phase typically accounts for over 90% of total execution time. With sorting optimization enabled, the dropset generation phase of the algorithm is faster by two to three orders of magnitude. For the dataset with 354 taxa, the dropset generation phase can be reduced from 1,902.7 sec to just 0.3 sec. In general, the overall execution time of the algorithm is improved by one to two orders of magnitude. Compared to the Python prototype, that was used for the publication by Pattengale et al. [2010a], our RAxML implementation is faster by two to three orders of magnitude.

As depicted in Table 2, execution times are highly dataset-specific. For instance, the unoptimized version requires 23 sec for the dataset with 218 taxa, but over 1,900 sec for the 354 taxon dataset that contains a significant larger total number of distinct bipartitions. Also, performance gains vary considerably among datasets. The reason for these variations is that, the computational complexity at the bit-level of the dropset generation phase is

# taxa	150	218	354	404	500	628
runtime	2.9	23.1	1,908.9	837.5	106.2	267.4
dropset time	2.1	21.3	1,902.7	830.9	101.1	260.9
opt. runtime	0.8	1.9	5.7	9.1	6.1	8.1
opt. dropset time	0.1	0.2	0.3	2.1	1.1	1.4
# bipartitions	11,336	23,276	170,397	73,434	42,630	75,156
# taxa	714	994	1,288	1,481	1,512	1,604
runtime	413.9	254.9	6,673.9	634,188.4	225,587.1	141,573.5
dropset time	403.6	243.9	6,640.9	634,025.2	225,451.7	141,406.6
opt. runtime	14.3	14.2	49.9	238.6	166.5	223.7
opt. dropset time	5.3	3.1	18.7	75.4	54.2	106.2
# bipartitions	61,280	64,497	178,242	718,719	589,985	352,229
# taxa	1,908	2,000	2,308	2,554		
runtime	61,942.2	1,704,656.0	5,759.9	423,813.2		
dropset time	61,854.0	1,704,244.0	5,693.1	423,522.0		
opt. runtime	117.5	727.4	114.6	743.8		
opt. dropset time	43.0	315.4	42.5	452.6		
# bipartitions	439,279	761,654	156,812	470,434		

Table 2: Execution times (dropset time) of the dropset generation phase and total execution time (runtime) in seconds for the basic and the optimized (opt.) version of the RIC-MISC approximation. For these measurements, the RIC-MISC is approximated with respect to the majority rule consensus. Each dataset comprises 10,000 bootstrap replicate trees.

determined by three factors:

1. **Number of bipartitions**: The number of bipartitions grows with the number of taxa and grows (until saturation) with the number of trees.
2. **Number of iterations**: The higher the number of rogue taxa that have deleterious effects on the bootstrap trees, the more iterations of the algorithm will be necessary.
3. **Length of bit vectors**: With a higher number of trees and/or taxa, the respective bit vectors become longer. Thus, more operations for comparing or computing the intersection of two bit vectors are required.

Factor 1 explains the observed difference in runtimes between the dataset with 218 and 354 taxa: while the dataset with 218 taxa comprises a total 23,276 bipartitions, the dataset with 354 taxa has more than 170,397 bipartitions (see Table 2). The dataset with 2,000 taxa exhibits the longest overall unoptimized runtime. This is because this dataset requires more iterations (Factor 2) than any other dataset until no further dropset improves the RIC of the consensus tree. Only 8 iterations are necessary for the dataset with 2,308 taxa – the runtime is proportionally shorter.

The runtime improvement achieved with the sorting optimization depends on the frequency distribution in the bipartition profile. If for most bipartitions that are not part of the consensus tree, the frequency is only slightly below the consensus threshold, then the runtime improvement in the optimized version may decrease to 0. Let us consider a datasets that has a high number of bipartitions that are (in consequence) poorly supported. In this dataset, our optimization allows to omit the generation of a number of dropset candidates that is quadratic in the number of bipartitions for which the pair-wise cumulative frequency is below the consensus threshold. This, for instance, is true for the 354-taxon dataset. The dataset has a high number of bipartitions and the RBIC of the unpruned MR consensus tree is very low (only 0.328). Therefore, the optimization yields particularly high runtime improvements for this dataset.

6.2. Comparison between RIC-MISC and MAST

We presented the MAST method in Section 3.1 and an RIC-optimizing approximation of the MISC in Section 3.3. As explained, from a theoretical point of view, the MISC approach is situated between the (strict) consensus tree and the MAST. Table 3 shows the optimality score with respect to the RBIC of the unpruned consensus trees, the MISC (both for strict and MR consensus trees) and MAST for all datasets. Note that, the RIC-MISC approximation is the only method that could be applied to the largest dataset (1000 trees with 7764 taxa) since other methods are computationally significantly more expensive. In case of the MAST, one week of CPU-time was necessary to compute the result of the dataset with 1000 trees and 994 taxa.

The strict consensus trees of virtually all datasets are generally poorly resolved. Few strict consensus trees of datasets with more than 150 taxa have a RBIC that is higher than 0.1. For unpruned consensus trees this RBIC values means that the datasets have only 10% of the support that is possible. On the other side, the datasets with 24, 53, 117 and 125 taxa have comparably high RBIC values between 0.6 and 0.7 for their strict consensus trees. This indicates a high number of consensus bipartitions and the presence of not many rogue taxa. Nevertheless, the RIC (and hence the RBIC as well) can be improved by the RIC-MISC approximation for two of the datasets (the 24- and 88-taxon datasets). In general, for the RIC-MISC approximation of the strict consensus tree, only few rogue taxa are recognized. In many cases, the number of taxa in the strict consensus is the same as in the MISC-based approximation of the strict consensus. A comparatively large number of leaves are dropped from the two largest datasets: for the dataset with 6,718 taxa, 23 rogue taxa are removed and for the 7,764-taxon dataset 22 rogues are dropped. The absolute RBIC gain in these two cases appears to be moderate, but the relative RBIC improvement amounts to 20.5% (6718 taxa) and 23.5% (7764 taxa), respectively.

This rather conservative behavior of the strict consensus trees was to be expected. While the RIC-MISC uncovers some consensus bipartitions, it is not surprising that pruning the identified rogue taxa does not drastically change the RBIC of the strict consensus.

Except for the aforementioned cases with a well resolved strict consensus tree, all MR consensus trees have a significantly higher RBIC (by factors varying between 2 and 10). The RBIC values for most datasets vary between 0.3 and 0.6. A notable exception is the dataset with 885 taxa for which both the strict (RBIC: 0.001) and the MR consensus tree (RBIC: 0.162) are particularly poorly resolved. One reason may be that originally, the dataset was intended for usage in combination with a backbone/restraint tree. This means that, the user restrains the topology of the result tree via empirical biological knowledge. For comparability of the analysis, we did not run the analysis with a constraint tree.

In most cases, the RBIC of the MR consensus can be increased by using the RIC-MISC approximation. A specifically high number of consensus bipartitions is recovered for the dataset with 128 taxa, where the RBIC can be increased from 0.525 to 0.586 by just dropping 4 leaves. Furthermore, for the dataset with 404 taxa, the RBIC of the MR consensus can be improved from 0.5 to 0.548 by removing 11 rogue taxa. As a last

# taxa	24	44	52	53	72	88	117	125	128	141	143	148	150	218
#taxa in MAST	14	13	9	23	7	28	63	88	44	13	25	21	19	19
MAST-RBIC	0.524	0.244	0.122	0.400	0.058	0.294	0.526	0.697	0.328	0.072	0.157	0.124	0.109	0.074
$\text{CONS}_{\text{strict}}$-RBIC	0.619	0.171	0.163	0.660	0.072	0.376	0.702	0.787	0.088	0.225	0.179	0.317	0.136	0.093
#taxa in $\text{MISC}_{\text{strict}}$	22	44	52	53	72	87	116	125	121	141	139	148	150	218
$\text{MISC}_{\text{strict}}$-RBIC	0.714	0.171	0.163	0.660	0.072	0.412	0.737	0.787	0.272	0.225	0.257	0.317	0.136	0.093
CONS_{MR}-RBIC	0.829	0.755	0.543	0.886	0.364	0.783	0.960	0.958	0.525	0.669	0.610	0.611	0.570	0.471
#taxa in MISC_{MR}	24	43	51	53	70	87	117	125	124	139	141	146	149	214
MISC_{MR}-RBIC	0.829	0.786	0.582	0.886	0.409	0.810	0.960	0.958	0.586	0.690	0.628	0.647	0.579	0.485

# taxa	316	317	350	354	404	424	451	500	628	714	885	994	1288	1481
#taxa in MAST	10	29	23	9	52	26	27	32	28	45	8	46		
MAST-RBIC	0.022	0.083	0.058	0.017	0.122	0.055	0.054	0.058	0.040	0.059	0.009	0.043		
$\text{CONS}_{\text{strict}}$-RBIC	0.048	0.096	0.098	0.031	0.045	0.064	0.105	0.107	0.091	0.120	0.001	0.178	0.107	0.108
#taxa in $\text{MISC}_{\text{strict}}$	314	314	347	354	403	420	447	500	625	714	885	989	1282	1478
$\text{MISC}_{\text{strict}}$-RBIC	0.061	0.108	0.115	0.031	0.052	0.086	0.123	0.107	0.106	0.120	0.001	0.193	0.118	0.111
CONS_{MR}-RBIC	0.365	0.541	0.495	0.328	0.500	0.501	0.492	0.589	0.515	0.569	0.162	0.679	0.584	0.438
#taxa in MISC_{MR}	314	314	346	352	393	413	439	496	624	708	877	991	1281	1462
MISC_{MR}-RBIC	0.373	0.551	0.508	0.334	0.548	0.537	0.529	0.597	0.525	0.579	0.172	0.686	0.591	0.454

# taxa	1512	1604	1748	1908	2000	2308	2554	6718	7764
#taxa in MAST									
MAST-RBIC									
$\text{CONS}_{\text{strict}}$-RBIC	0.109	0.090	0.015	0.090	0.050	0.213	0.088	0.039	0.034
#taxa in $\text{MISC}_{\text{strict}}$	1507	1604	1747	1905	1984	2299	2551	6695	7742
$\text{MISC}_{\text{strict}}$-RBIC	0.125	0.090	0.017	0.091	0.081	0.223	0.092	0.047	0.042
CONS_{MR}-RBIC	0.518	0.487	0.248	0.553	0.449	0.713	0.541	0.421	0.360
#taxa in MISC_{MR}	1501	1585	1731	1897	1960	2299	2517	6611	7655
MISC_{MR}-RBIC	0.530	0.502	0.262	0.557	0.479	0.718	0.555	0.445	0.380

Table 3: Optimality (with respect to the RBIC) of the MAST method and the MISC approximation for all datasets. Some values are missing for the MAST because of computational complexity and excessive resource requirements. RBIC values are shown for strict ($\text{CONS}_{\text{strict}}$-RBIC) and MR consensus trees (CONS_{MR}-RBIC) and the approximation for the RIC-MISC with respect to the strict ($\text{MISC}_{\text{strict}}$-RBIC) and MR consensus trees (MISC_{MR}-RBIC).

particularly successful example, consider the dataset with 2000 taxa, where the RBIC is substantially increased from 0.449 to 0.479 by pruning 40 taxa.

Bearing the different algorithmic approaches in mind, for the RIC-optimizing approximation the question we pose is: “how many leaves are dropped?”, while for the MAST, we are interested in the question “how many leaves remain in the result tree?”. Hence, for the real-world datasets, (see Table 3), the MAST often consists of a very small part of the initial taxon-set, while the MISC only has slightly less taxa than the consensus tree. For the dataset with 52 taxa, the MAST comprises 9 taxa and for the 72-taxon dataset only 7 leaves are in the resulting MAST. We notice a strong correlation between the RBIC optimality of the consensus tree and the number of taxa in the MAST. For instance, the 354-taxon dataset has particularly poorly resolved strict and MR consensus trees and its MAST only consists of 9 taxa. The MAST is even smaller for the dataset with 885 taxa: it just comprises 8 taxa. In contrast to this, the dataset with 125 taxa has well-resolved consensus trees, contains no rogue taxa (according to the RIC-MISC) and the MAST consists of 88 taxa. For such well-behaved datasets, MASTs of this comparatively large size represent a useful alternative for summarizing bootstrap trees. However, this means that the MAST method will barely produce usable results, if we have a poorly resolved consensus tree because of the presence of rogue taxa or other associated factors.

The RBIC of the MASTs is – in most cases – comparable to the RBIC of the respective strict consensus tree (see Table 3). Thus, we conclude, that both the strict consensus tree and the MAST are too conservative and are not very informative (with respect to the RBIC) for most of the datasets we have studied. All MR consensus trees yield higher RBIC values. Also the potential for improvement via the RIC-MISC approximation is higher if the MR consensus trees are deployed.

6.3. Performance of RBIC-Optimizing A Posteriori Methods

In this Section, we assess the performance of the RBIC-optimizing methods on our collection of real-world datasets: the leaf stability index, the taxonomic instability measure, the naïve algorithm as well as the iterative versions of these algorithms were tested. Furthermore, we analyze, how relevant the theoretical advantage of the dropset-based approaches is in practice.

6.3.1. Monolithic A Posteriori Methods

We used the naïve algorithm, the taxonomic instability measure and the leaf stability index for determining a set of taxa S, such that pruning S from the bootstrap trees maximizes the RBIC of the MR consensus tree. The maximum RBIC values that can be achieved using the respective methods for the various datasets as well as the number of taxa we need to prune for achieving this, are depicted in Table 4. Figures 14, 15, 16, 17 and 18 show the behavior of the RBIC score as more and more taxa are pruned from datasets.

The RBIC of datasets with a well-resolved and highly supported consensus tree (such as the 53- and 117-taxon datasets) can not be improved by any of the three methods. However, for the 125-taxon dataset, the naïve algorithm improves the RBIC by pruning a single taxon. The RIC-optimizing approximation does not find any rogue taxa for this dataset.

On small datasets with up to 128 taxa, the leaf stability index and the taxonomic instability measure outperform the naïve algorithm for almost every dataset. In contrast, on larger datasets, the naïve algorithm yields significantly higher RBIC values. For every dataset except the datasets with 24 and 628 taxa, the taxonomic instability measure determines a pruning a set of taxa that is better under the RBIC than that of the leaf stability index. For many of the larger datasets (316, 317 and 350 taxa), the leaf stability index does not find a pruning that can improve the RBIC. In the final analysis, we observe that for datasets of up to 128 taxa, the taxonomic instability measure yields the best prunings, while on larger datasets the naïve algorithm performs best.

We conclude that taxa which are unstable with respect to the two stability measures, do not necessarily lead to a decrease of support in the consensus tree. We think that, the occurrence of most rogue taxa is a special case, where local instability in (small) sub-trees of the consensus tree leads to the loss of consensus bipartitions. On the other hand, there can exist a non rogue-induced global instability in the tree which can hinder the reconstruction of a well-resolved consensus tree. This global instability can not be alleviated by pruning taxa and may further increase with the larger (mainly regarding the number of taxa) and more complex datasets. The leaf stability index is even more influenced by global instability than the taxonomic instability measure (where close evolutionary relationships are favored for our choice of parameters). Genuine rogue taxa may exhibit above-average instability according to those stability measures. However, global instability may cause generally decreased stability values on taxa that do not exhibit a genuine deleterious effect on the consensus tree. Thus, global instability may be one reason, why the algorithms based on stability measures fail to improve the support of the consensus tree for larger datasets. This hypothesis puts the use of stability measures for rogue taxa identification into question.

Figures 14, 15, 16, 17 and 18 further corroborate this finding: For instance, on the datasets with 218 taxa (see Figure 16) and 404 taxa (see Figure 17) the naïve algorithm improves the RBIC and the taxonomic instability measure slightly improves the RBIC, while the leaf stability index suggests prunings that lead to a decrease in the RBIC value. Note that, in many cases (e.g., the 148-taxon dataset in Figure 16 or the 404-taxon dataset in Figure 17) all measures agree on the first few most deleterious rogue taxa to be pruned.

For most (especially larger) datasets, the RBIC curve exhibits a high initial gradient. The RBIC gradient decreases until the maximum or a plateau is reached. If further taxa are pruned, bipartitions of the consensus tree start to merge or vanish and the RBIC decreases. The grey line in the Figures depicts the maximum possible RBIC for the given number of taxa after pruning. For instance, in case of the 128-taxon dataset in Figure 15, the curve for the naïve algorithm approaches this line, when more than 40 taxa are pruned.

For comparison, we included curves for the RIC-optimizing approximation. This algorithm only prunes taxa, if the number of bipartitions, that can be added to the consensus tree, is higher than the number of taxa we need to prune for achieving this improvement in resolution. Since additional bipartitions lead to large RBIC improvements, the naïve algorithm usually starts by pruning the same taxa as the RIC-optimizing approximation. This overlap of prunings is surprising. As we discussed in Section 3.3, there exist instances that can only be solved by a dropset-based approach. Therefore, we may expect the RIC-optimizing approximation to outperform the naïve algorithm. On the other hand, when alternative dropsets yield the same RIC improvement, the RIC-optimizing approximation will not necessarily choose the dropset that is optimal under the RBIC. Thus, the naïve algorithm may occasionally outperform the RIC-optimizing approximation. In fact, the differences are marginal (e.g., for the 128-taxon dataset, the RIC-optimizing approximation achieves an RBIC that is better by 0.001 when 4 taxa are pruned).

For many datasets the RBIC-optimizing algorithms are able to recover twice as much support as the RIC-optimizing approximation. Notable examples are the 218-taxon dataset (see Figure 16) or the dataset with 72-taxa (see Figure 14). The 7,764-taxon dataset is particularly interesting (see Figure 18). Here, the naïve algorithm improves the RBIC from 0.36 to 0.405, while the RIC-optimizing approximation yields a result with an RBIC of only 0.38. However, one should bear in mind, that an immense number of taxa must be pruned to achieve these improvements. In case of the 7,764-taxon dataset, the naïve algorithm identifies 953 taxa (i.e., 12.3% of all taxa) as rogue taxa. For this large dataset with many taxa, pruning 12% of all taxa may be acceptable for biologists. However, for both datasets (with 404 and 2000 taxa) in Figure 17, a high number of taxa is pruned until the RBIC maximum is achieved and the overall RBIC gain compared to the result of the RIC-optimizing algorithm is moderate.

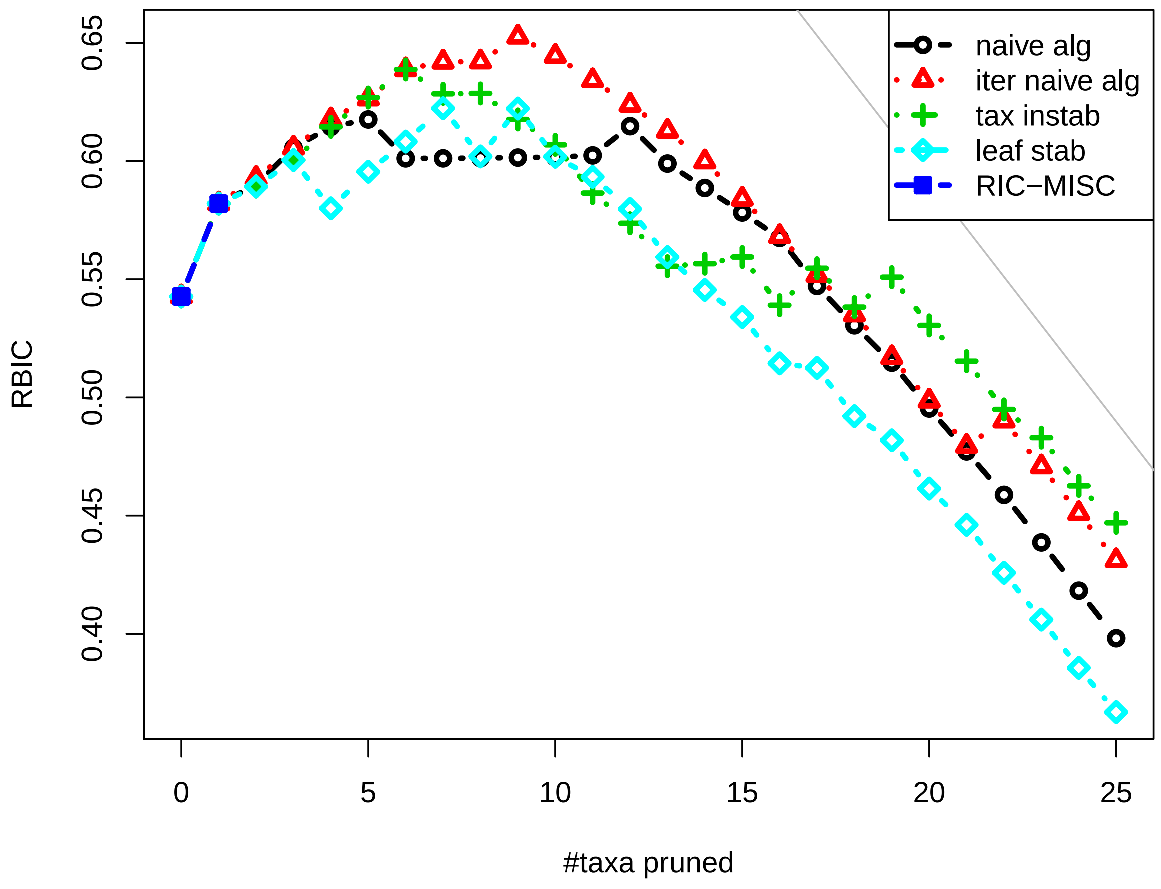
6.3.2. Iterative A Posteriori Methods

In Section 3.7 we explained that, we can easily transform the naïve algorithm and both stability measures into iterative methods. We assumed that, iterative versions perform at least as well as their monolithic counterparts.

Our hypothesis is corroborated for the iterative naïve algorithm which significantly outperforms the monolithic version of the algorithm. Performance plots for the iterative naïve algorithm are provided in Figures 14, 15, 16 and 14. We observe that for every pruning step, the iterative version yields a RBIC that is higher than or equal to the RBIC obtained by the monolithic version. In fact, there does not exist a single pruning step, where any alternative method performs better than the iterative naïve algorithm. Thus, the iterative naïve algorithm achieves the best RBIC-MISC approximation for every dataset (see Table 4). Nevertheless, in some cases, alternative methods yield equally good results with an identical RBIC score.

In particular, the iterative version of the naïve algorithm alleviates the shortcomings of the monolithic version compared to the stability measures for some datasets, such as the datasets with 52, 72 and 128 taxa (see Figures 14 and 15). On these datasets the stability measures yield prunings sets with a higher RBIC than the monolithic, naïve algorithm. The improved performance of the iterative algorithm indicates that, the moderate performance of the monolithic algorithm is not related to problematic instances that can not be solved *per se* by the naïve algorithm. We discuss such examples in Sections 3.3 and 3.5 and find that these instances are problematic for the naïve algorithm, but feasible for dropset-based methods or the taxonomic instability measure. When datasets contain such pathologic rogue taxa constellations, the iterative naïve algorithm will perform similarly to the monolithic version. Since this is not the case, we can conclude that the reason for the sub-optimal performance is a lack of predictive power in the rogue candidate assessment with respect to subsequent, later pruning steps. In the monolithic version of the naïve algorithm, we assess the "rogueness potential" of each taxon only once as the RBIC change in the consensus tree that is induced by pruning the candidate taxon. As we prune more and more taxa, this initial assessment becomes increasingly inaccurate. As a consequence, the monolithic and iterative version often agree on the first pruning steps. While the RBIC curve for the iterative algorithm moves towards a single maximum, the RBIC curve for the monolithic algorithm often produces a plateau with multiple quantitatively similar maxima (see, e.g., the datasets in Figure 14).

The results obtained by the iterative naïve algorithm represent the best approximation for RBIC-MISC developed in the framework of this thesis. The single global maximum



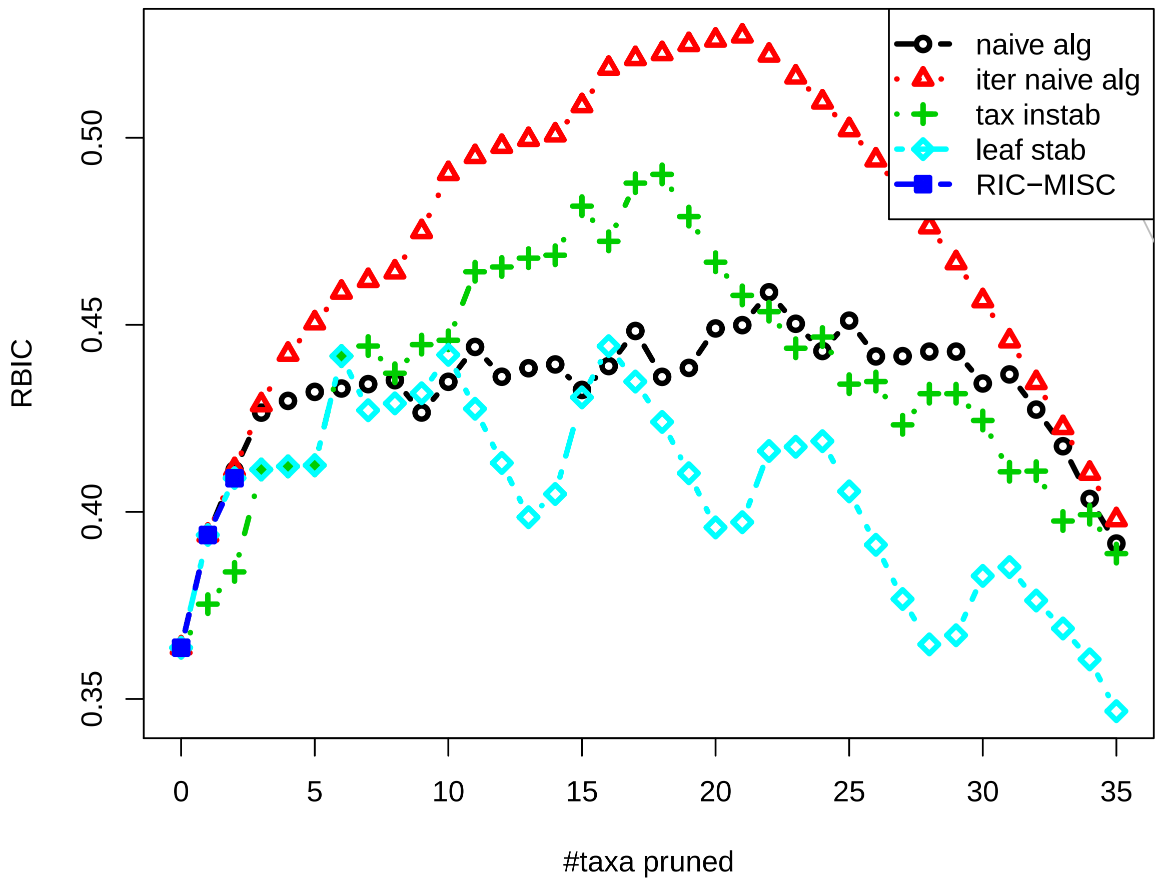
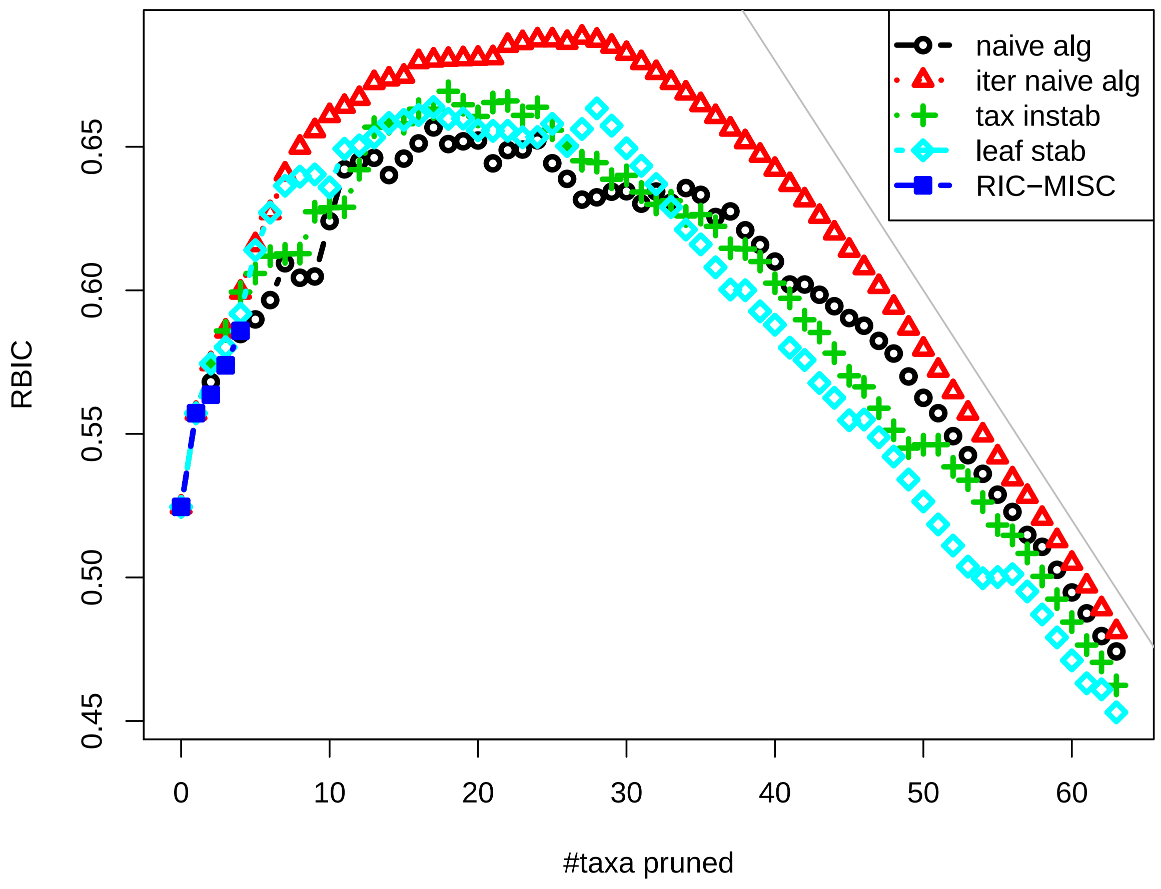


Figure 14: RBIC-optimality of incremental pruning of taxa that are most unfavorable according to the method under study for the **52**-taxon dataset (top) and the **72**-taxon dataset (bottom).



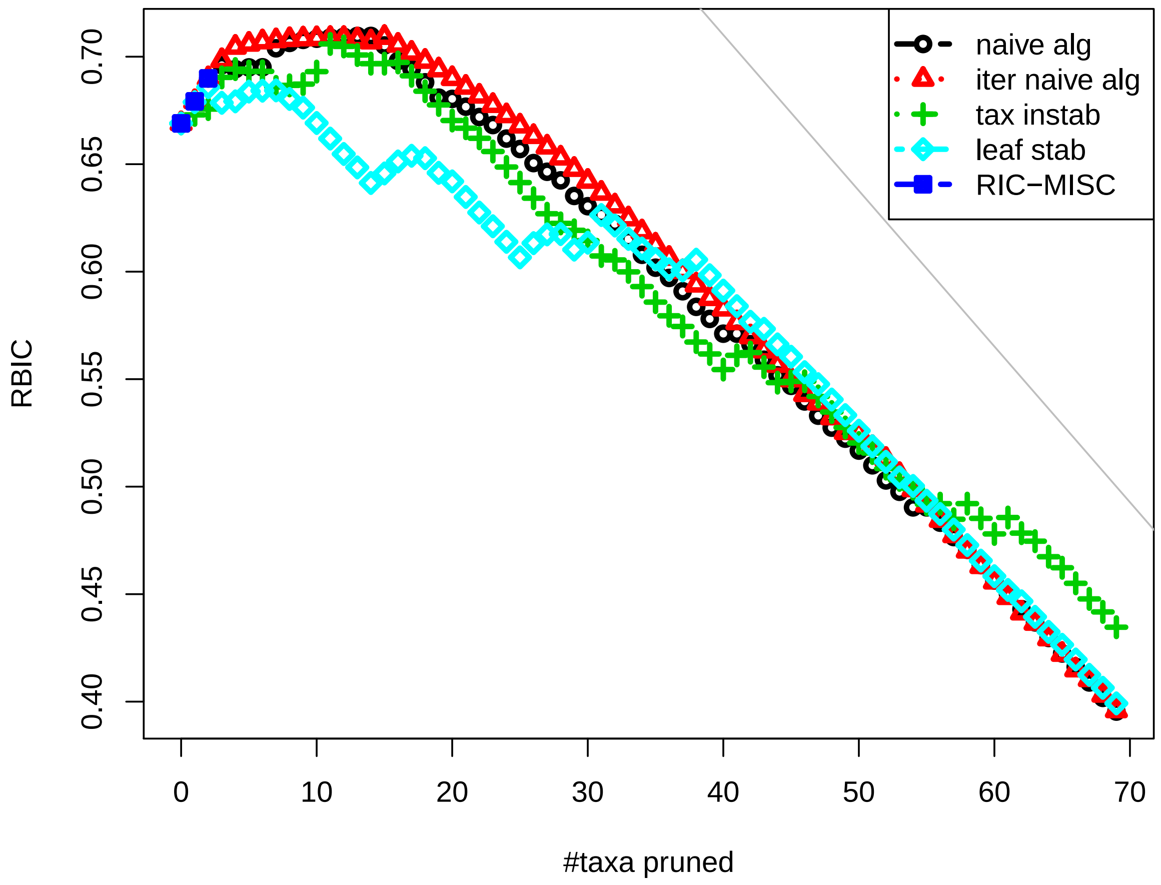
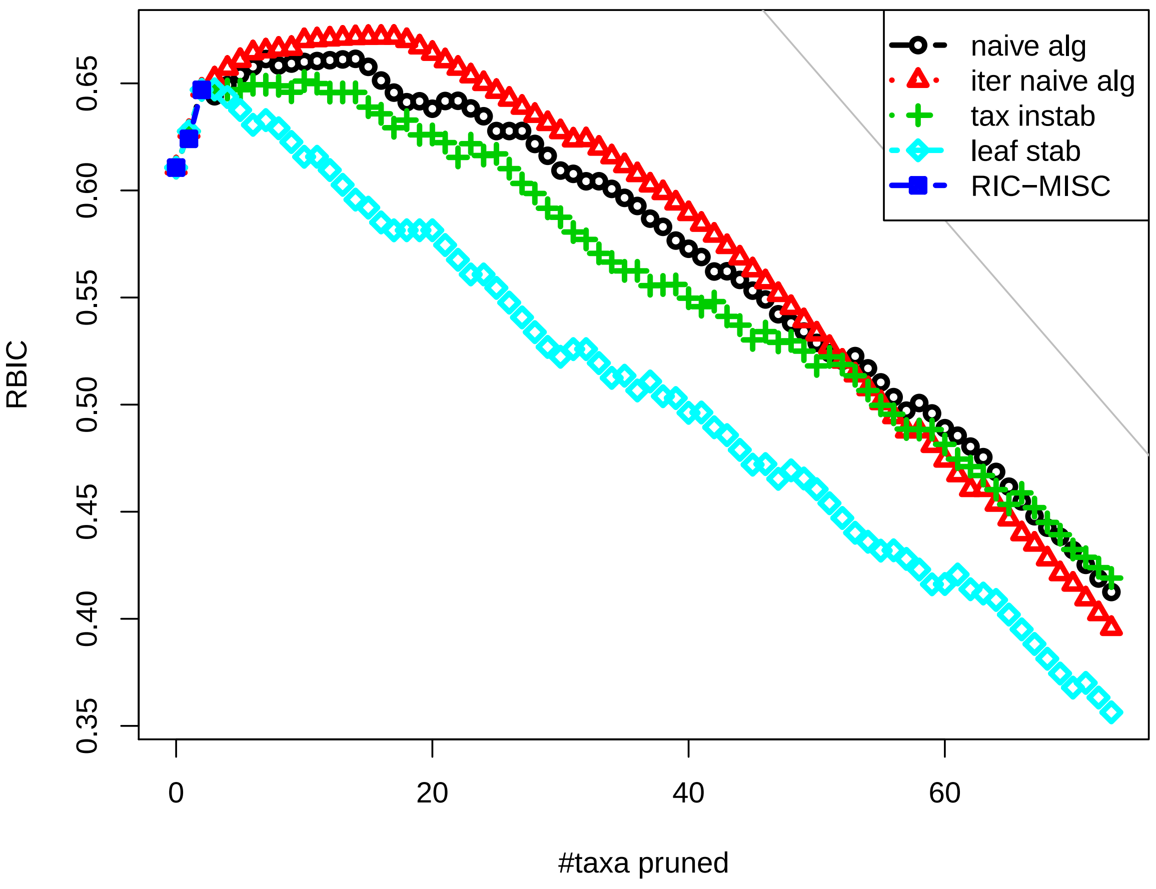


Figure 15: RBIC-optimality of incremental pruning of taxa that are most unfavorable according to the method under study for the **128**-taxon dataset (top) and the **141**-taxon dataset (bottom).



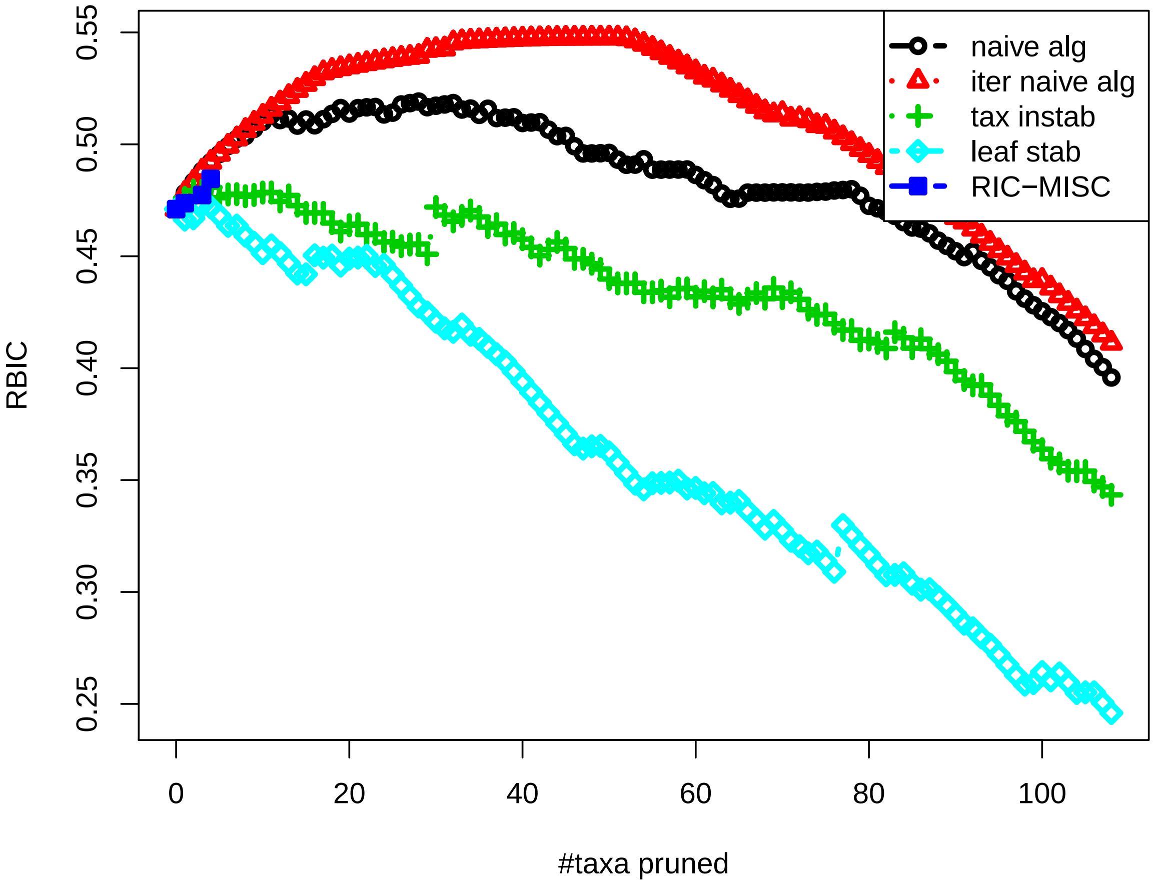


Figure 16: RBIC-optimality of incremental pruning of taxa that are most unfavorable according to the method under study for the **148**-taxon dataset (top) and the **218**-taxon dataset (bottom).

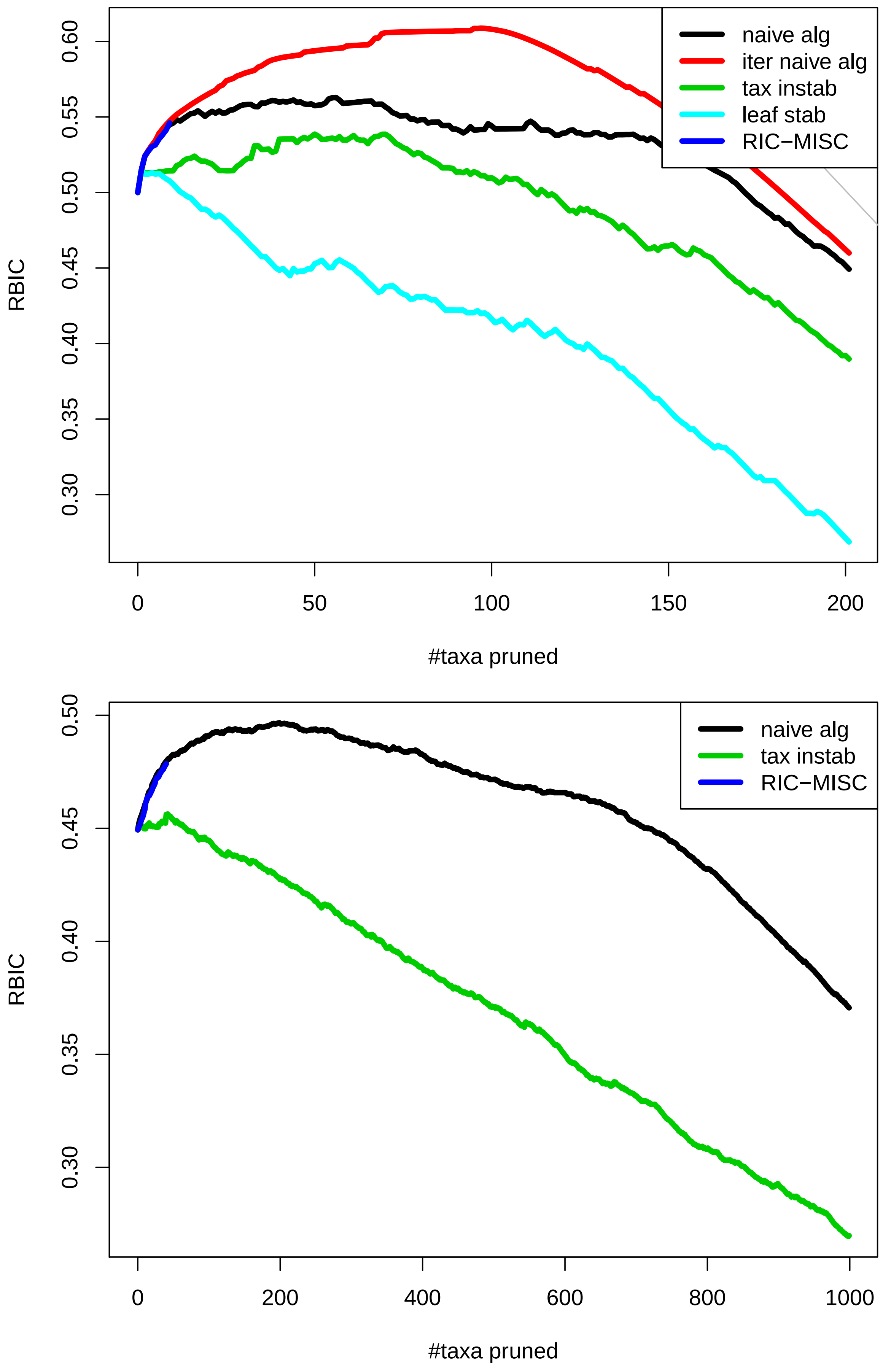


Figure 17: RBIC-optimality of incremental pruning of taxa that are most unfavorable according to the method under study for the **404**-taxon dataset (top) and the **2000**-taxon dataset (bottom).

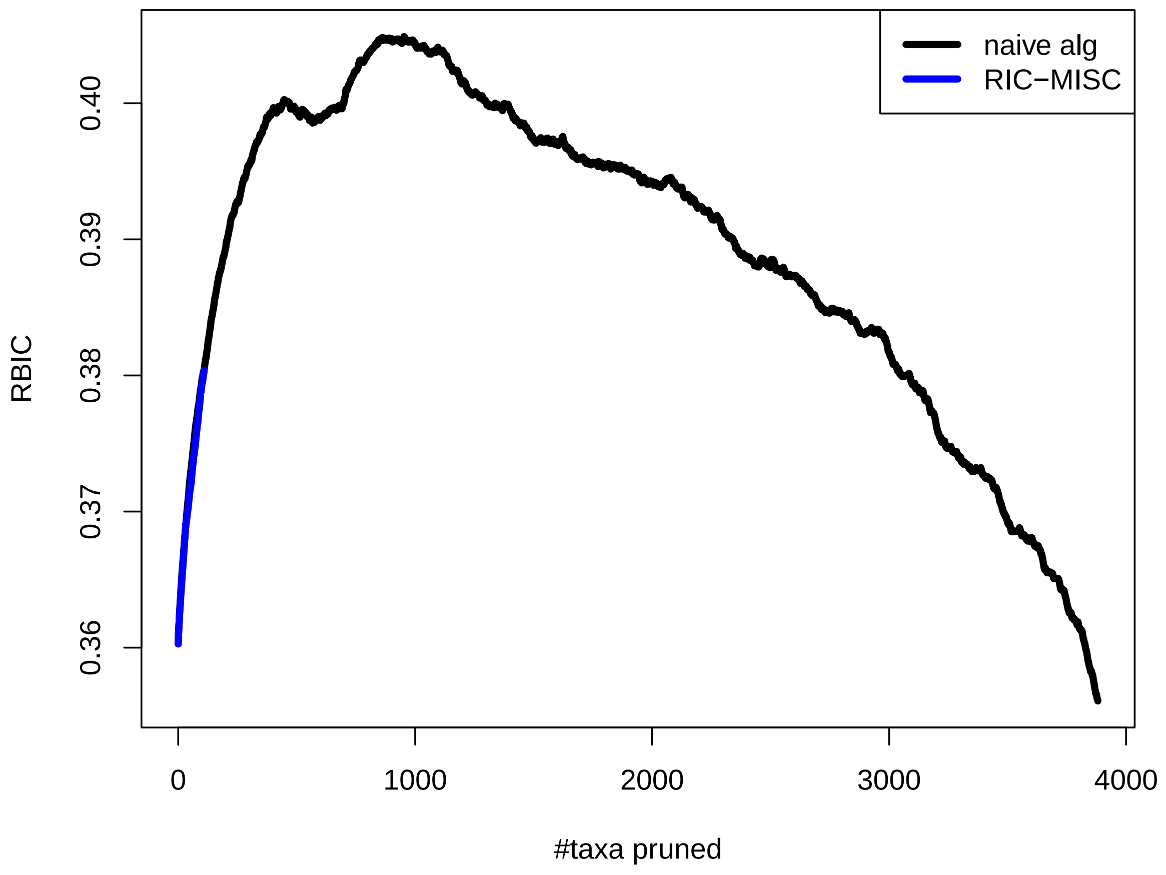


Figure 18: RBIC-optimality of incremental pruning of taxa that are most unfavorable according to the method under study for the **7,764**-taxon dataset.

that one can observe in all RBIC curves for the iterative naïve algorithm, also serves as an efficient and accurate stopping condition. For alternative methods, it is unclear whether a decreasing RBIC is just a random fluctuation, or if the decrease signifies that the RBIC maximum of this curve has been passed.

If the iterative naïve algorithm is stopped after pruning the same number of taxa as the RIC-optimizing approximation, then both methods yield a result that is equally optimal under the RBIC criterion. Thus, even the theoretically more powerful dropset-based approach does not perform better on real-world datasets. While the RIC-optimizing approximation does not compute an accurate MISC either, our hope is, that the iterative naïve algorithm produces results that are close to the global MISC optimum. If this is true, then the slopes of the pruning steps (until reaching the maximum) in the curves of Figures 14, 15, 16 and 17 point towards a specific distribution of the "rogue taxon effect": There exist a few rogue taxa with a highly deleterious effect on the consensus tree (steep slope) and in most cases many rogue taxa that only induce small deleterious effects onto support values (weak rogue taxa). These *weak* rogue taxa may be stable taxa for which a rogue taxon effect was generated by a genuine rogue taxon during computation of the bootstrap trees.

Figure 19 provides performance data for the iterative versions of the leaf stability index and the taxonomic instability measure. For the dataset with 72 taxa, the RBIC curve for the iterative leaf stability index fluctuates to a lesser extent than the analogous curve for the monolithic version. The reason for this reduction in fluctuation lies in the repeated reassessment of the leaf stability that is carried out in the iterative version. For both datasets in Figure 19, the iterative taxonomic instability measure performs slightly better than its monolithic counterpart. According to Table 4 these findings are consistent: the iterative version of the leaf stability index does not yield a better result than the monolithic version for any dataset. In case of the taxonomic instability measure, there is no common trend concerning the best RBIC results and differences between the monolithic and iterative versions are significantly less pronounced than for the naïve algorithm.

In Section 6.3.1, we speculated that, global instability may lead to generally higher instability values (with respect to the stability measures) than induced by genuine rogue taxa. If this hypothesis is true, it would also explain, why the iterative versions of the stability measures (in particular the leaf stability index) do not yield any improvements.

6.3.3. The Dropset-Approach compared to Single-Taxon Droppings

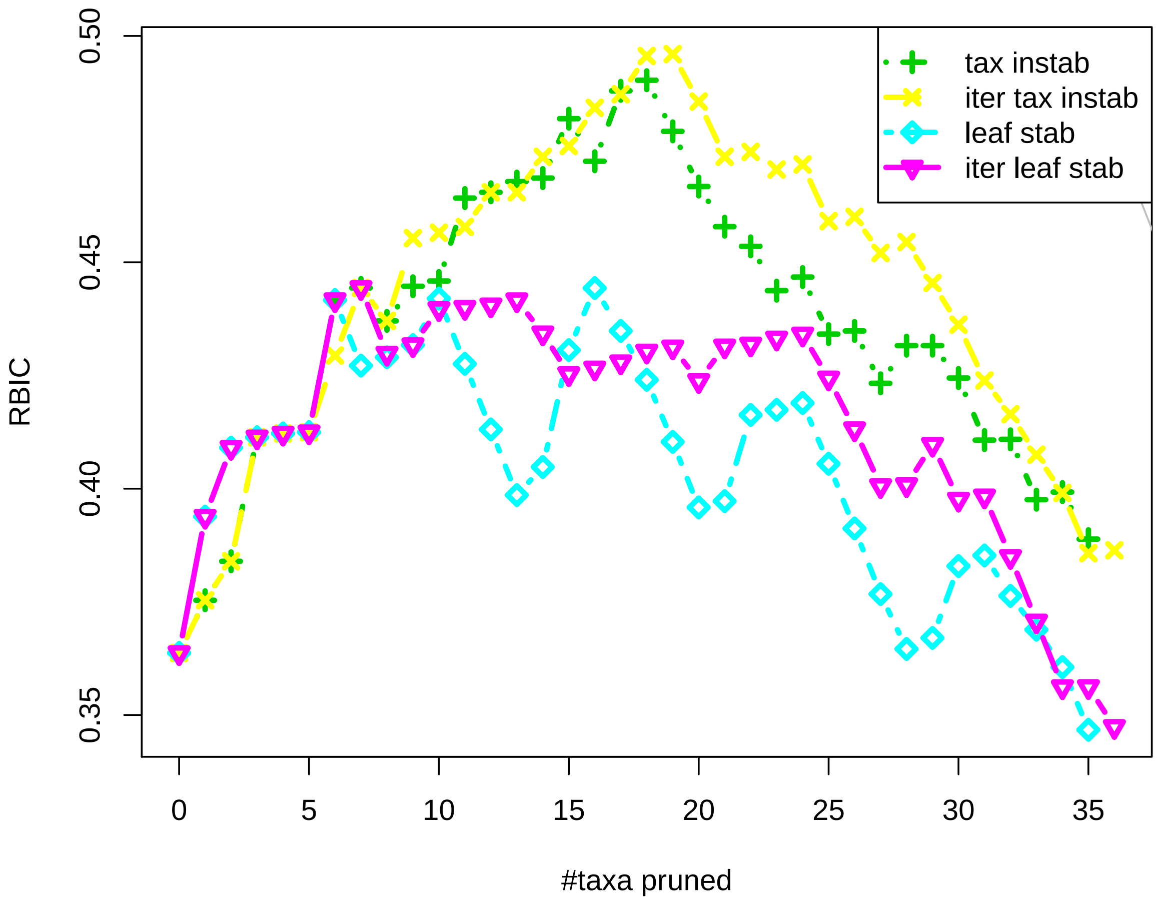
In Section 3.3, we discussed the advantage of methods that generate dropsets by merging two or more bipartitions. We observed, that the iterative naïve algorithm and the RIC-optimizing approximation largely choose taxa for pruning that lead to a result of equal RBIC optimality (see 6.3.2). This finding is not necessarily representative for dropset-based methods, since it is not the task of the RIC-optimizing approximation to optimize the RBIC. This is why we proposed a dropset-based RBIC-optimizing approximation in Section 3.4.

Because we have to compute a consensus tree explicitly for each dropset, the algorithm could only be executed for the datasets with 24, 44, 52, 53, 72, 88, 117, 125, 128 and 141 taxa. For all datasets except the 128- and 141-taxon datasets, the dropset-based algorithm yields the same taxa for pruning as the naïve iterative algorithm and therefore identical results. For the dataset with 52 and 72 taxa, the algorithm repeatedly prunes dropsets with 2, 3 and, 4 taxa. However, these are the same taxa as identified by the naïve iterative algorithm. Apparently, there are no pathological rogue taxon constellations (as described in Section 3.3) in these datasets that would require the usage of a dropset-based method (or we would need to merge more than two bipartitions for resolving these constellations). This is different for the 128-taxon dataset. Here the dropset-based approach prunes two dropsets (one with 2 and 4 taxa) that yield a higher RBIC improvement than the taxa that are pruned one at a time by the iterative naïve algorithm. While the iterative naïve algorithm yields a pruned consensus tree with an RBIC of 0.687, the dropset-based algorithm achieves a notably better result with an RBIC of 0.693. For the dataset with 141 taxa, a dropset with two taxa makes the difference between the result of the iterative naïve algorithm (RBIC: 0.704) and the result of the dropset-based method, which is marginally better (RBIC: 0.705).

If all dropsets that are pruned by the dropset-based approach consist of just one taxon, then the dropset-based approach produces the same result as the iterative naïve algorithm. In fact, for the 10 aforementioned datasets, most dropsets that are pruned contain only one taxon. Apart from dropset size, another important dropset attribute consists in the number of bipartitions that are added to (gained by) the consensus tree, when the dropset is actually pruned. Figure 20 depicts the frequency distributions of both attributes for all dropsets that were identified by the RIC-optimizing approximation for all real-world datasets in our study, once for a strict consensus tree and once for MR consensus tree. While more dropsets are pruned for the optimization of the MR consensus tree, both distributions exhibit a common pattern: most dropsets consist of one taxon (92.5% for the dropsets of the MR consensus RIC-optimization and 89.7% for the RIC-optimization with respect to the strict consensus threshold) and most of these dropsets add two bipartitions

#taxa	naive algorithm				taxonomic instab. measure				leaf stability index			
	monolithic		iterative		monolithic		iterative		monolithic		iterative	
	#p	RBIC	#p	RBIC	#p	RBIC	#p	RBIC	#p	RBIC	#p	RBIC
24	1	0.853	1	0.853	3	0.837	3	0.837	1	0.853	1	0.853
44	2	0.789	2	0.789	2	0.786	2	0.786	1	0.786	1	0.786
52	5	0.618	9	0.653	6	0.639	6	0.627	7	0.622	7	0.622
53	0	0.886	0	0.886	0	0.886	0	0.886	0	0.886	0	0.886
72	22	0.459	21	0.527	18	0.490	19	0.496	16	0.444	7	0.444
88	4	0.832	5	0.832	4	0.832	4	0.832	5	0.813	2	0.813
117	0	0.960	0	0.960	0	0.960	0	0.960	0	0.960	0	0.960
125	1	0.959	1	0.959	0	0.958	0	0.958	0	0.958	0	0.958
128	17	0.657	27	0.688	18	0.669	21	0.667	17	0.664	17	0.657
141	13	0.710	15	0.709	11	0.706	11	0.706	2	0.686	2	0.686
143	7	0.648	12	0.651	3	0.628	20	0.632	7	0.619	7	0.619
148	14	0.661	17	0.672	10	0.651	7	0.655	3	0.647	3	0.647
150	17	0.600	28	0.614	14	0.581	17	0.589	6	0.571	5	0.571
218	28	0.519	51	0.548	4	0.481	4	0.481	3	0.473	3	0.473
316	31	0.392	113	0.438	11	0.371	11	0.371	0	0.365	0	0.365
317	13	0.570	75	0.610	6	0.552	6	0.549	0	0.541	0	0.541
350	24	0.535	62	0.555	9	0.497	5	0.497	0	0.495	0	0.495
354	31	0.349	146	0.386	3	0.328	3	0.328	2	0.328	0	0.328
404	56	0.563	97	0.609	50	0.539	63	0.549	1	0.515		
424	54	0.564	80	0.607	55	0.540	46	0.541	13	0.508		
451	83	0.547	100	0.601	36	0.538	35	0.541	30	0.508		
500	32	0.611	92	0.634	28	0.598	22	0.597	7	0.594		
628	46	0.537	133	0.564	4	0.516	13	0.515	3	0.520		
714	48	0.600	125	0.620	6	0.570	6	0.570	2	0.569		
885	53	0.183	441	0.216	18	0.167	36	0.167	6	0.165		
994	38	0.695	95	0.704	2	0.681	26	0.682	1	0.681		
1288	128	0.611	217	0.634	13	0.587						
1481	113	0.474	428	0.514	3	0.440						
1512	105	0.546			6	0.524						
1604	149	0.515	396	0.548	14	0.490						
1908	115	0.569			0	0.553						
2000	199	0.497			42	0.456						
2308	76	0.728	196	0.734	9	0.716						
2554	248	0.574			51	0.543						
6718	720	0.467										
7764	953	0.405										

Table 4: Performance of the monolithic and iterative version of the naïve algorithm, the taxonomic instability measure and the leaf stability index. For each of the six methods, the maximum RBIC of the pruned consensus tree and the number of taxa (#p) that was pruned for achieving this maximum is shown. For corresponding RBIC values of unpruned MR consensus trees, see Table 3. Some data points are missing in the table because of high computational complexity of the respective algorithms.



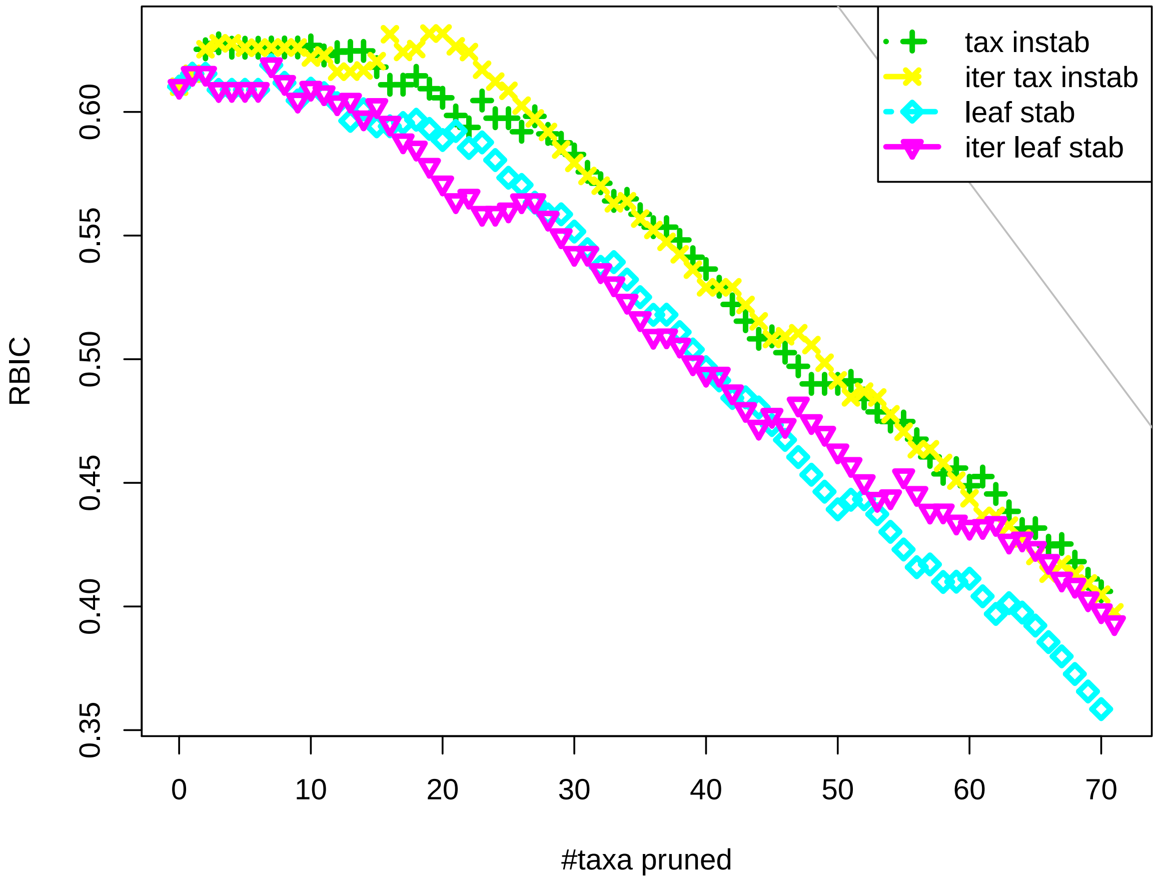


Figure 19: RBIC-optimality of incremental pruning of taxa for the monolithic and iterative version of the leaf stability index and the taxonomic instability measure for the **72**-taxon dataset (top) and the **143**-taxon dataset (bottom).

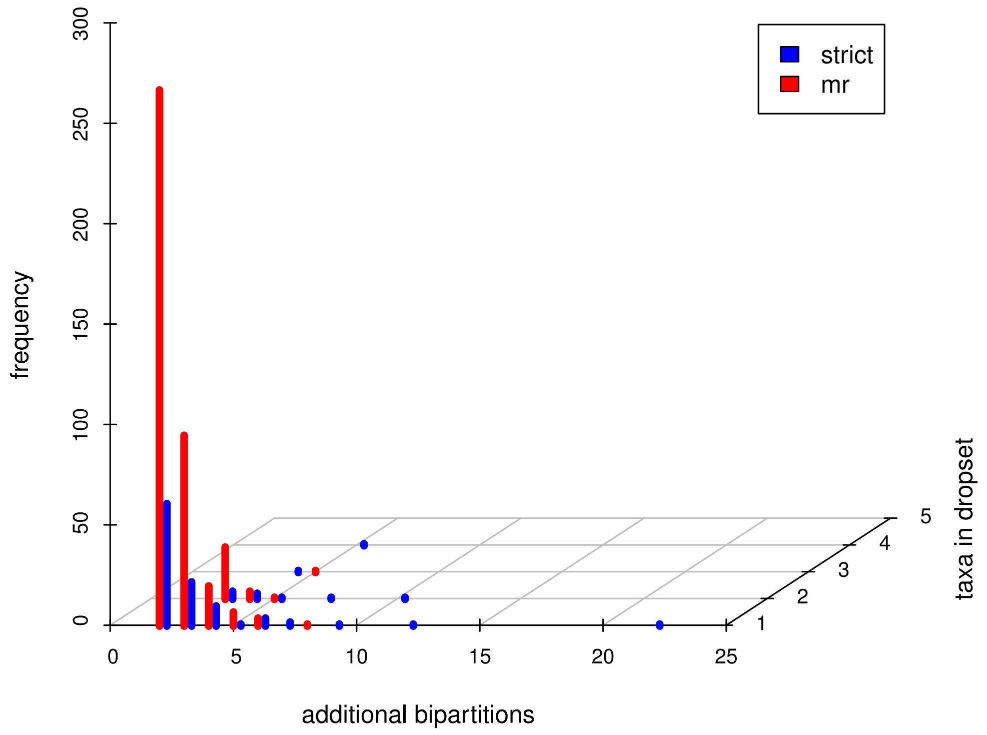


Figure 20: Statistics for all dropsets that are pruned by the RIC-optimizing approximation when applied to the real-world datasets (once with a strict consensus threshold and a MR consensus threshold). On the y-axis, absolute frequencies are depicted for dropsets that add x bipartitions to the consensus tree while pruning z taxa.

to the consensus tree. Furthermore, as discussed in Section 6.3.2, the occurrence of multi-taxon dropsets does not necessarily mean that an algorithm that prunes taxa one by one (e.g., the iterative naïve algorithm) will choose different taxa for pruning. These findings suggest that the ability of dropset-based methods to determine the optimal solution for pathologic rogue taxon instances is of limited importance with respect to result optimality in real-world datasets. Given that, the dropset-based algorithms still are approximative, the iterative naïve algorithm represents as an efficient and – in most cases – sufficient method for RBIC optimization.

There are some notable outliers in Figure 20 for the optimization with respect to the strict consensus. For a single-taxon dropset, the resolution of the strict consensus tree can be increased by 22 bipartitions. In another case, a dropset of size 4 needs to be pruned, such that 5 additional consensus tree bipartitions become part of the consensus tree. Both dropsets are pruned from the tree collections with 2000 taxa. Hence, dropsets with such extreme features may be more likely to occur in large datasets.

6.4. Performance of A Priori Methods

In this Section, we assess the performance of the a priori methods (see Section 4). Furthermore, we examine, if two potential stopping criteria (proposed in Section 4.3) can be used for designing of an iterative a priori algorithm, that identifies rogue taxa without using a set of bootstrap trees as input.

6.4.1. Monolithic A Priori Methods

A priori methods generally show inconsistent performance on the real-world datasets. We discuss various tendencies based on the 72- and 128-taxon datasets (see Figure 21) and the 143- and 354-taxon datasets (see Figure 22). All figures show RBIC-based performance for the six proposed likelihood weight-derived measures for quantifying placement uncertainty. Since the iterative naïve algorithm performed best among the a posteriori algorithms, we included its RBIC curve for comparison. For small datasets with many rogue taxa, there exists the possibility that a method will prune a combination of taxa that highly improves the RBIC by chance alone. To account for this, we created 1000 random dropsets the adequate size for each pruning step and computed the RBIC after pruning the dropset. Figures 21 and 22 include curves for the average and the maximum RBIC of these 1000 random dropsets. Thus, if a method $\mathcal{M}$ outperforms the best of the 1000 random dropsets, we can reject the Null hypothesis *"method $\mathcal{M}$ does not perform better than pruning random taxa"* with a significance $< 0.1\%$.

In general, the a priori methods do not achieve better performance than the iterative naïve algorithm on any dataset. However, since a priori methods do not identify rogue taxa based on the concrete collection of bootstrap trees that is used for evaluation, we can consider equally good performance as a satisfying result. For instance, performance is identical for the 24-taxon dataset, that contains exactly one rogue taxon according to the iterative naïve algorithm. All 6 a priori methods also identify this taxon as a rogue taxon. For the 143-taxon dataset, the $\mathrm{sd}_{\mathrm{sample}}$ measure and number of distinct likely placements (abbreviated as $\mathrm{num}_{\mathrm{best}}$, see top of Figure 22) yield a result that is only marginally worse than the result of the naïve algorithm. For the 72-taxon dataset (see Figure 21), $\mathrm{sd}_{\mathrm{sample}}$ also performs best among the a priori methods. For the dataset with 128 taxa (see Figure 21), $\mathrm{sd}_{\mathrm{sample}}$ performs almost as well as the naïve iterative algorithm for the first pruning steps. However, for this datasets the RBIC curves for $\mathrm{num}_{\mathrm{best}}$ and the number of likely placements (abbreviated as $\mathrm{num}_{\mathrm{place}}$) achieve the highest maximum among the a priori methods.

For small datasets with less than 200 taxa, a priori method performance is satisfactory. Although no a priori method clearly outperforms the alternative a priori methods, $\mathrm{sd}_{\mathrm{sample}}$ and $\mathrm{num}_{\mathrm{best}}$ (sometimes $\mathrm{num}_{\mathrm{place}}$ as well) usually achieve the highest RBIC-maximum in the corresponding curves. This finding suggests that measures for the likelihood weight distribution among samples may be best suited for a priori rogue taxa identification. It is surprising that, $\mathrm{num}_{\mathrm{sample}}$ often is among the best performing methods as well, since this measure is dominated by the likelihood weight distribution among placements instead the likelihood weight distribution among samples.

For datasets with more than 200 taxa, the performance of the a priori methods is significantly worse than the performance of the iterative naïve algorithm. Sometimes, the a priori methods do not yield an RBIC improvement at all. A drastic example is the dataset with 354 taxa (see Figure 22). No a priori method yields a significant RBIC improvement. For this dataset, there exist random dropsets that perform better than the $\mathrm{sd}_{\mathrm{sample}}$ measure. The EDPL and the dwlwp measures yield results that are as sub-optimal as those obtained by an average random dropset. The common feature of these two measures is that they rely on weighted patristic distances. As we discussed in Section 4.1.1, patristic distances are interesting from a biological point of view. However, their performance on our real-world test datasets suggests that, using patristic distances may be misleading for a priori rogue taxa identification.

Note that, the performance of the a priori methods is significantly influenced by the best-known ML reference tree that is used. We also experimented with MP (Maximum Parsimony) trees as an alternative to ML trees, since they are faster to compute. The a priori methods performed worse on MP trees than with ML trees.

It is possible to identify rogue taxa using measures that are based on the likelihood weight distribution among samples as well as using measures that are based on the likelihood weight distribution among placements. This indicates, that rogue taxa influence both dimensions of the likelihood weight matrix. However, our hybrid measure, the dwlwp (see Section 4.2.3) fails to merge information from both dimension into a superior, that is, quantitatively better, measure.

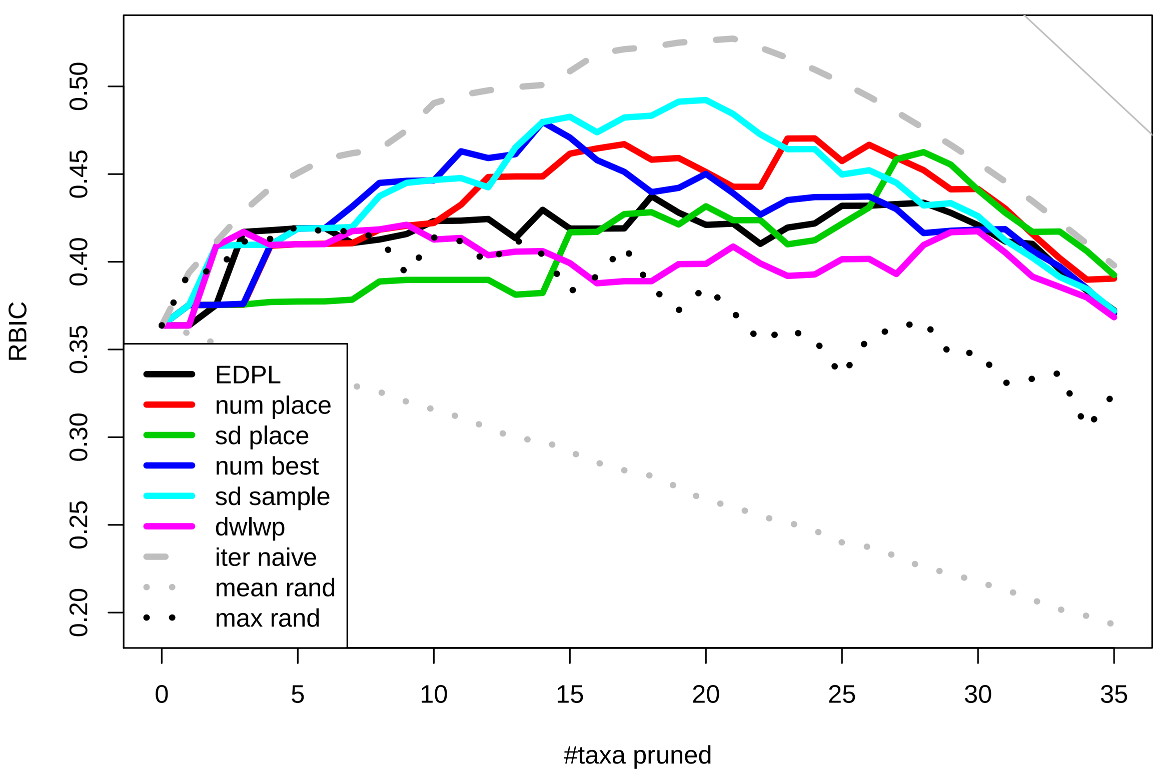
The inconsistent results concerning the performance of the a priori methods are hard to interpret. First, we want to emphasize that, the result of the iterative naïve algorithm is partially very specific to the concrete set of bootstrap trees and may slightly vary for a distinct set of 1000 bootstrap trees. Of course, bootstrapping trees are considered to be representative and reproducible. However, if a bipartition occurs in exactly 50% of the trees (for a MR consensus threshold), then one bootstrap replicate can make the difference with respect to whether the specific bipartition will occur in the consensus tree or not. Thus, a different sampling of alignment columns may lead to slightly different results that can not be predicted exactly by a priori methods. Thus, results of these methods are expected to be sub-optimal with respect to the results of the iterative naïve algorithm that are based on a specific collection of bootstrap trees. Given these considerations, some RBIC curves (especially those of $\text{sd}_{\text{sample}}$ and num_{best}) are successful at identifying rogue taxa. The otherwise sub-optimal RBIC curves of the a priori methods are most likely a result of the fact that the placement uncertainty of a taxon on the best-known ML tree will not accurately predict the instability of the specific taxon in bootstrap trees. If a priori methods correctly predict global instability of taxa, then the problem may be, that globally instable taxa are not necessarily rogue taxa.

On the other hand, we observed, that particularly stable taxa (where pruning yields strong RBIC decrease) very often achieve the maximum (most stable) score according to one of the measures (e.g., the same most likely placement in all score samples for num_{best}). Thus, we believe that, a priori measures can serve as good stability predictors. This finding is supported by the fact that $\text{sd}_{\text{sample}}$, num_{best} and $\text{num}_{\text{place}}$ perform better than random on almost every dataset.

6.4.2. Iterative A Priori Method With Stopping Criterion

We used the standard deviation of the likelihood weights among samples $\text{sd}_{\text{sample}}$ as measure for an iterative version of the a priori method (described in Section 4.3). While there are datasets, where this measure does not identify rogue taxa well (such as the 354-taxon dataset in Figure 22), its overall performance is one of the best among the a priori methods.

We computed the iterative version of the $\text{sd}_{\text{sample}}$ on a subset of smaller datasets since the repeated re-computation of the best-known ML tree is particularly time consuming. The monolithic and iterative versions of $\text{sd}_{\text{sample}}$ often produce slightly different results. However, the iterative version does not perform significantly better than the monolithic $\text{sd}_{\text{sample}}$ method. We attribute the differences in RBIC curves to the fact that distinct best-known ML reference trees are used in each step of the iterative version. Figure 26 depicts the RBIC curves for the monolithic and iterative version of the method on the datasets with 52 and 128 taxa (where $\text{sd}_{\text{sample}}$ performs particularly well). For the 128-taxon dataset, the performance of both methods is very similar, while for the 52-taxon



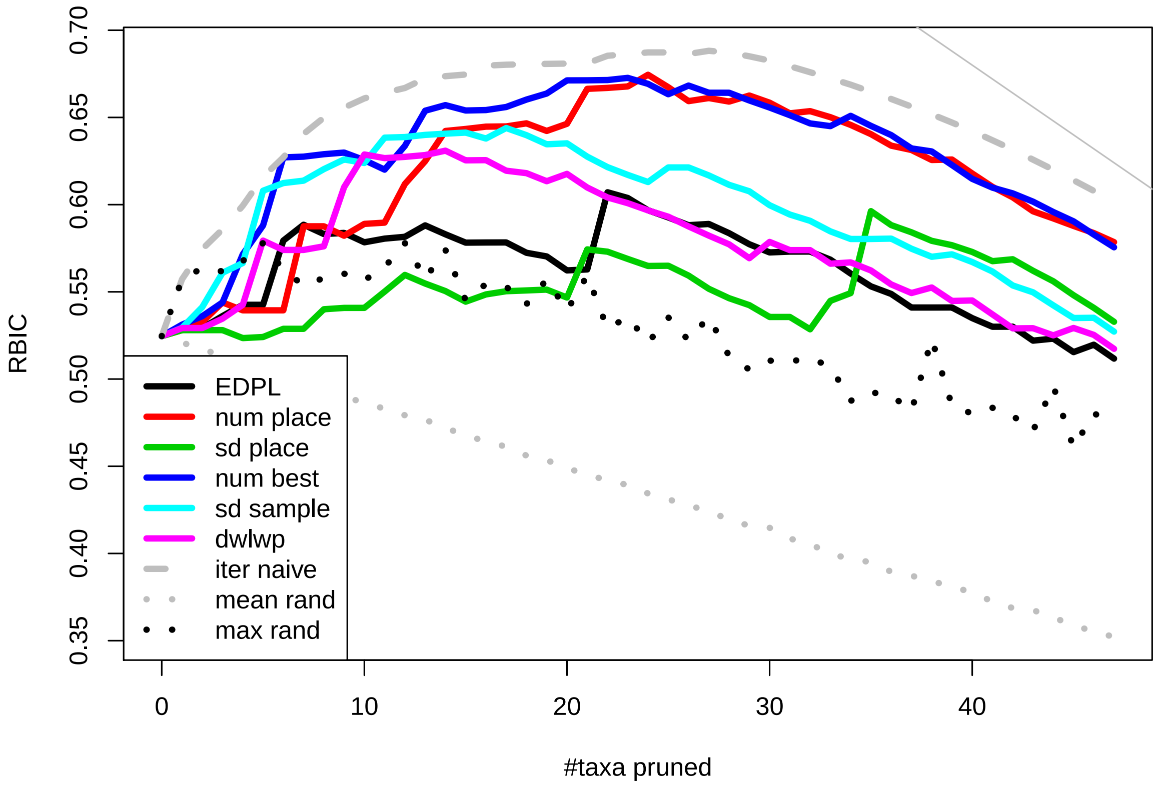
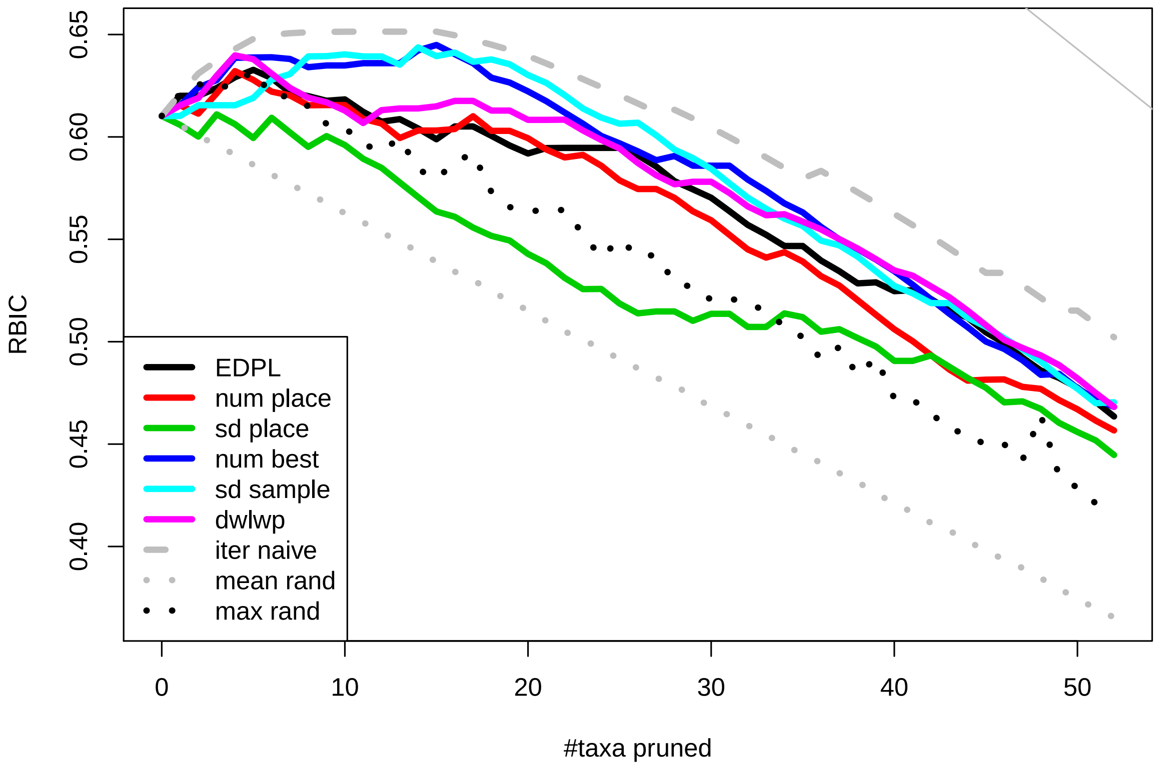


Figure 21: Performance of the a priori methods for the datasets with **72** (top) and **128** (bottom) taxa. RBIC curves for the iterative naïve algorithm (*iter naive*) and the average and maximum performance of random dropsets (*mean rand* and *max rand*) are included for comparison.



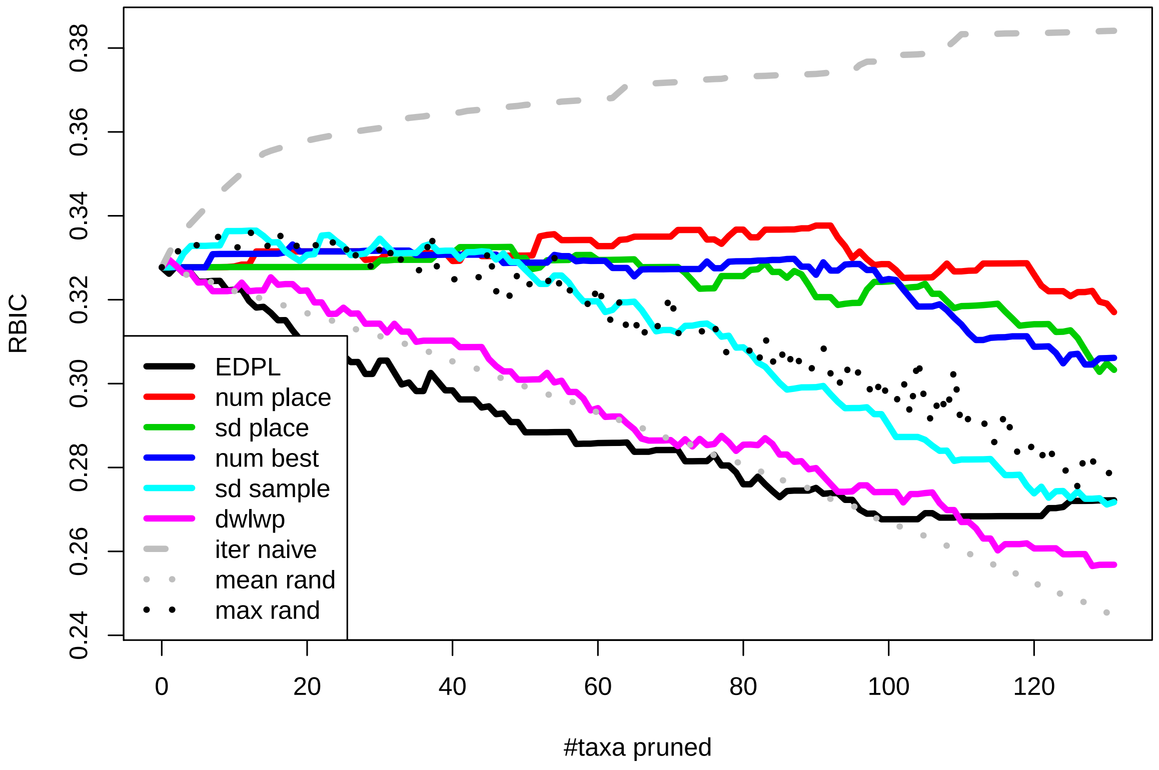


Figure 22: Performance of the a priori methods for the datasets with **143** (top) and **354** (bottom) taxa. RBIC curves for the iterative naïve algorithm (*iter naive*) and the average and maximum performance of random dropsets (*mean rand* and *max rand*) are included for comparison.

dataset, the monolithic $\text{sd}_{\text{sample}}$ performs better than the iterative version.

The second hypothesis we examined for iterative a priori methods is, whether we can come up with a stopping criterion for the pruning process. For a posteriori methods, we stop pruning taxa, when the RBIC maximum in the consensus tree of the bootstrap trees is reached. However, an ideal a priori method should be able to identify a number of rogue taxa and then stop without relying on bootstrap trees as input. A stopping criterion, based on the score of the a priori methods (such as the $\text{sd}_{\text{sample}}$ value), proved to be hard to develop. Usually, between one third and half of the taxa for each dataset assume the smallest possible value for a measure (e.g., $\text{sd}_{\text{sample}} = 0.0$). We consider these taxa as the *stable core* of a dataset. We noticed that, the size of the stable core is inversely proportional to the size of the dataset. As a consequence, scores of taxa are not normally distributed and thus z-value statistics can not be applied.

Furthermore, we could not associate the maximum in the RBIC curves with a specific score threshold. The reason is that the a priori scores are not comparable among datasets with a different number of taxa. In case of $\text{sd}_{\text{sample}}$, there is an inversely proportional relationship between the size of the dataset and the size of the highest scores (indicating high rogue taxon potential). For instance, in the 24-taxon dataset the maximum $\text{sd}_{\text{sample}}$ value is 0.03 and in the dataset with 885 taxa the taxon with the highest $\text{sd}_{\text{sample}}$ has a value of 0.002.

For iterative a priori methods, we also examined, whether SH-like support values can be deployed as a stopping criterion. Alternatively, we analyzed if three exists a relationship between the RBIC-maximum in the performance curve and the RF-metric between the pruned ML-tree and a re-computed ML-tree (see Section 3.7).

Our hypothesis regarding SH-like support values was, that they may behave in similar way when taxa are pruned as the bootstrap support values in the consensus tree. On our datasets, we observed that, the two types of support values are mostly uncorrelated (see Figure 26). For all datasets, the RBIC curves of SH-like support values attain their maximum on the starting tree when no taxa have been pruned. This maximum is significantly higher than the RBIC values of any consensus tree in pruning steps for all datasets. As taxa are pruned, the SH-like support decreases steadily with slight fluctuations. One explanation may be that, the aLRT method, by definition always outputs a fully resolved (NNI-optimized) bifurcating tree. Recovering additional bipartitions usually substantially improves the RBIC of a consensus tree. This is, by definition, not possible for the SH-like support values. However, it is a surprising result that apparently, the SH-like support values are not influenced by the presence of rogue taxa in the best-known ML tree.

As alternative option for a stopping criterion, we considered the RF-metric between the pruned best-known ML tree and the re-computed pruned best-known ML tree (i.e., the trees have the same taxon set). The underlying idea is that, the best-known ML tree will become topologically stable (note change any more), once we have pruned all rogue taxa and only the stable core (backbone) is left. In fact, for one of the smaller datasets (see Figure 26 at top), the initial pruning of rogue taxa does not induce topological changes in the best-known ML tree. Interestingly, a particular high RF-metric is observed (when 13 taxa are pruned), after the RBIC curve for the iterative version of $\text{sd}_{\text{sample}}$ starts decreasing after its maximum. For the 128-taxon dataset (see bottom of Figure 26), we observe an RF-metric $= 0.0$ for a pruning that is slightly below the RBIC maximum. Here, the RF-metric would work as a stopping criterion. However, this is an outlier, since this potential stopping criterion does not work for any of the other datasets (e.g., the 52-taxon dataset). Nevertheless, it is remarkable, that the best-known ML tree undergoes heavy topological changes (in 5 cases the RF-metric is > 0.1, that is, it changes by more than 10%) for the first few rogue taxa that are pruned. Thus, we conclude that, the RF-metric can not be used as a stopping criterion for a priori methods either.

6.5. Verification: Recomputing Bootstrap Trees

6.5.1. Case Study: Recomputing Trees for Single-Taxon Prunings

A bootstrap analysis (comprising enough replicates) is considered to be reproducible. However, we can not assume that pruning a taxon t from a collection of bootstrap trees will yield the same consensus tree and bootstrap tree collection for that matter as the consensus tree we would obtain from a set of re-computed bootstrap trees on a pruned alignment without taxon t. A consensus tree from a pruned set of bootstrap trees is by itself a legitimate result since the underlying evolutionary relationships are indeed contained in the bootstrap trees. We examine, if this result really is representative for a bootstrap analysis that has been conducted without the presence of rogue taxa, that is, we strive to answer the question: "do we need to re-compute bootstrap trees after pruning rogues or not?"

For this purpose, we initially explore the effect of removing a single taxon from the alignment on the consensus tree (after performing a bootstrap analysis). For three datasets (with 52, 218 and 316 taxa), we remove each taxon at a time from the alignment and compute the RBIC of the consensus tree (denoted as RBIC_{rec}). The choice of datasets was motivated by the relatively small alignment size and the expected runtimes of the analyses. We compared the RBIC_{rec} to the RBIC values we obtain when we prune a single taxon from our bootstrap tree collection that comprises all taxa (denoted as RBIC_{pr}). Figure 24 depicts the changes in the RBIC (i.e., the RBIC of the unpruned consensus tree is subtracted from these values) for both kinds of RBIC measurements. While the values for the 52-taxon dataset show a strong Pearson correlation coefficient ($\rho = 0.73$), the correlation is much weaker for the other two datasets (see Figure 24). The correlation appears to depend on the number of taxa in the dataset. The reason for this is, that if pruning a taxon adds the same number of bipartitions to the consensus tree, the RBIC change is smaller in larger datasets because of the larger normalization factor. On the two larger datasets, the smaller RBIC change is harder to distinguish from the slight variations (i.e., noise) in the various bootstrap analyses. Furthermore, we notice that the range of RBIC change variation (along the y-axis of Figure 24) decreases on larger datasets. On larger datasets, there are more bipartitions in the consensus tree. The law of large numbers seems to prevent the occurrence of large RBIC changes.

6.5.2. Verification: Results of Iterative Naïve Algorithm

We conclude that, in larger datasets the signal we obtain from pruning just one taxon may be too weak compared to noise. Beside that, it is more interesting for practitioners, if bootstrap trees where *all* rogue taxa have been pruned correspond to bootstrap trees that have been re-computed from a pruned alignment (without the identified rogue taxa). Thus, we removed all k taxa that were identified as rogue taxa by the iterative naïve algorithm from the alignments of all datasets with up to 885 taxa and re-computed the bootstrap trees on the appropriately pruned alignments. For comparison, we also performed the same experiment removing k taxa that exhibit the highest taxonomic instability for each dataset. In other words, with respect to Figures 14, 15, 16 and 17, we

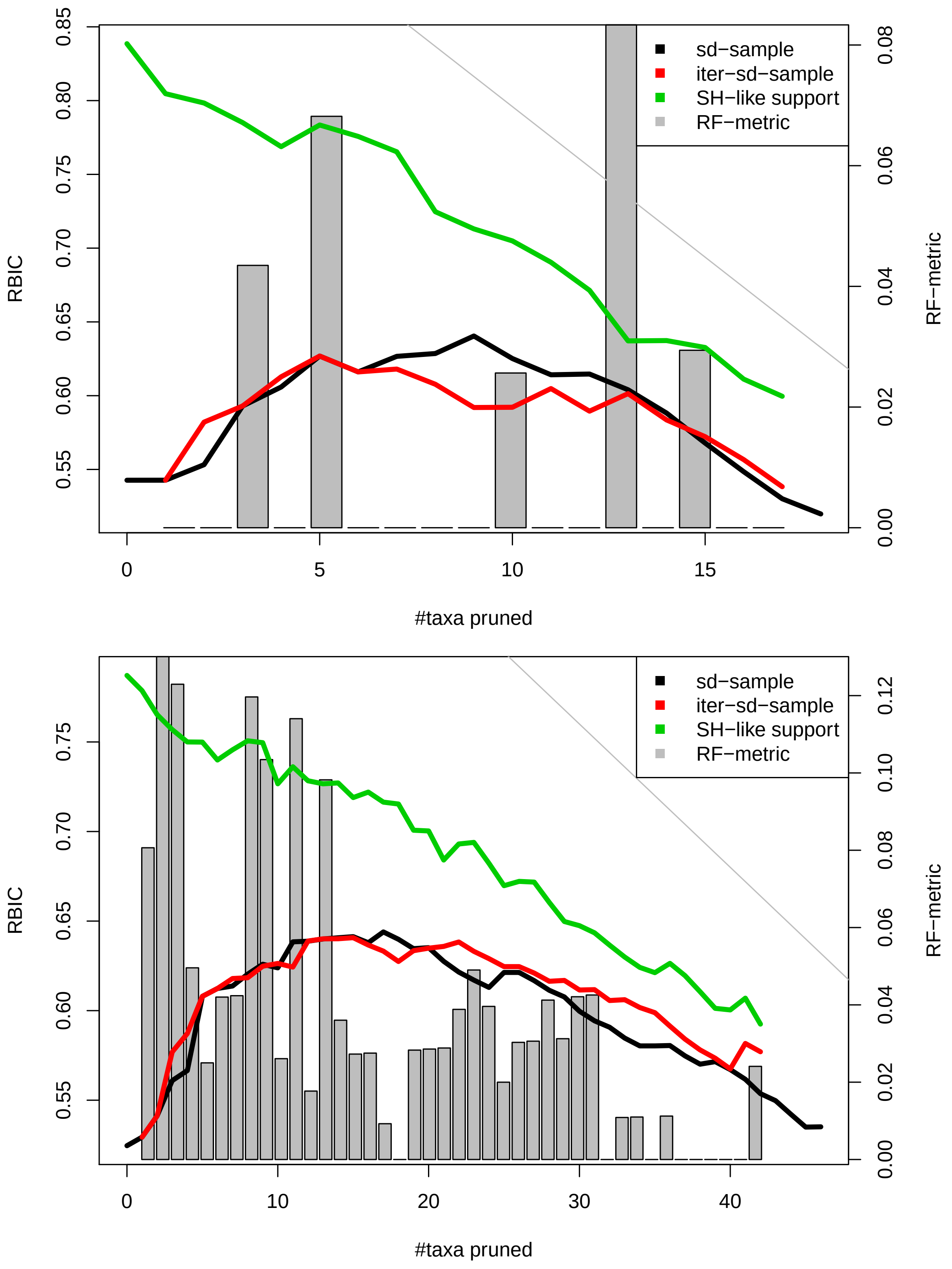


Figure 23: Performance of the monolithic and iterative version of the $\text{sd}_{\text{sample}}$ measure for datasets with **52** (top) and **128** (bottom) taxa. The cumulative SH-like support on the best-known ML trees (after optimization) and the RF-metric between the pruned ML tree and optimized ML tree are depicted.

pruned all taxa that are identified as rogue taxa by the monolithic taxonomic instability measure at the pruning step where the curve for the iterative naïve algorithm has its maximum.

RBIC values of pruned consensus trees are almost perfectly correlated with the RBIC values of re-computed consensus trees ($\rho = 0.995$). The standard deviation of the differences between RBIC_{pr} and RBIC_{rec} is 0.017. This value is small enough to use the RBIC_{pr} as a reliable indicator for RBIC_{rec}. Furthermore, this standard deviation explains the weak correlation between RBIC_{pr} and RBIC_{rec} when a single taxon is pruned on the 218- and 316-taxon datasets. In the most extreme cases, RBIC_{pr} was smaller by 0.0583 than RBIC_{rec} and in the opposite direction for another dataset RBIC_{rec} was smaller by 0.0291 than RBIC_{pr}. In general, the values for RBIC_{rec} were higher than RBIC_{pr}. However, a one-sided Wilcoxon signed rank test did not confirm that RBIC_{rec} values are significantly higher than RBIC_{pr} ($p = 0.144$).

The fact that RBIC_{pr} and RBIC_{rec} are only slightly different for taxa with the highest taxonomic instability index, corroborates our result from Sections 6.3.1 and 6.3.2: Unstable taxa (with respect to a stability measure such as the taxonomic instability) are not necessarily rogue taxa.

For two datasets, the dropset-based RBIC optimizing algorithm achieved a higher RBIC maximum than the iterative naïve algorithm (see Section 6.3.3). We also computed RBIC_{rec} values after removing the rogue taxa identified by the dropset-based RBIC optimizing algorithm. However, the improvements obtained by the dropset-based algorithm are significantly smaller than the standard deviation between RBIC_{pr} and RBIC_{rec}. In fact, on the re-computed bootstrap trees of the pruned alignments, the rogue taxa identified by the dropset-based algorithm performed marginally worse than the rogue taxa identified by the iterative naïve algorithm. For the 141-taxon dataset $\text{RBIC}_{\text{rec}} = 0.7013$ for the iterative naïve algorithm and $\text{RBIC}_{\text{rec}} = 0.7011$ for the dropset-based algorithm. In the case of the 128-taxon dataset, the dropset-based algorithm achieved $\text{RBIC}_{\text{rec}} = 0.673$, while the iterative naïve algorithm reaches $\text{RBIC}_{\text{rec}} = 0.678$. Our data basis is too weak for definite conclusions about dropset-based algorithms. However, our findings indicate that the iterative naïve algorithm represents a good approximation of RBIC-MISC given the fact that dropset-based algorithms are computationally significantly more expensive for approximating the RBIC.

6.5.3. Identifying Rogue Taxa in Recomputed Bootstrap Trees

Ideally, we would expect, that the recomputed bootstrap trees resulting from the pruned alignments do not contain rogue taxa. For verification, we executed the iterative naïve algorithm on the bootstrap trees that had been recomputed from the reduced/pruned alignments (i.e., alignments without rogue taxa identified by the iterative naïve algorithm in the preceding step and alignments pruned from the same number of taxa of most unstable taxa w.r.t. taxonomic instability). We refer to this second run of the iterative naïve algorithm as the second round of rogue taxa identification. Figure 25 depicts the improvements that are possible if rogue taxa are pruned from the recomputed bootstrap trees in successive iterations. Furthermore, the number of taxa we need to prune for achieving the respective RBIC improvement is shown for each dataset containing up to 885 taxa.

In three cases, no further rogue taxa could be found (denoted by circles in the graphic). In one case, the monolithic taxonomic instability measure succeeded in removing all rogue taxa. The trees that have been recomputed without the instable taxa (w.r.t. the taxonomic instability measure) still contain rogue taxa that would have been identified and pruned by the iterative naïve algorithm. Thus, high RBIC gains are possible and a considerable number of taxa needs to be pruned.

Note that, in the first round of rogue taxon identification, we restrained the iterative naïve algorithm to not prune more than half of taxa in the dataset. This is why there is still a large number of rogue taxa in the reduced 885-taxon dataset (with 444 taxa in Figure 25). Thus, the two arrows that originate at the 444 taxa data point are not representative. If the iterative naïve algorithm is deployed in both rounds then we achieve the RBIC maximum of the 885-taxon dataset when it is pruned down to 344 taxa.

While this result for globally unstable taxa is not surprising, it is unexpected, that the recomputed bootstrap trees from which all rogue taxa as identified by the iterative naïve algorithm have been pruned, still contain rogue taxa. However, the number of rogue taxa is usually very small (i.e., short arrows in Figure 25) and only yield marginal improvements (i.e., arrows exhibit a gentle slope in Figure 25). There are two potential reasons, why rogue taxa can occur again in recomputed bootstrap trees of which supposedly all rogue taxa have been removed from the alignments in the first round of rogue taxon detection.

The first reason is that there is a number of rogue taxa with a weak deleterious effect. Given the slight variations in bootstrap analyses (as observed in Section 6.5.2), they only behave as rogue taxa with a specific probability that depends on the strength of their rogue taxon effect. If we repeat the bootstrapping analysis, rogue taxa with a particularly weak effect, will only behave as rogues in few analyses. Another reason may be, that taxa with a very weak deleterious effect (w.r.t. the RBIC) should not be considered as rogue taxa at all. Weak rogue taxon effects just appear to occur at random.

In both cases, it may be better to keep taxa with weak rogue taxon behavior in the analysis, if it is planned to repeat post-analyses with a reduced alignment after pruning identified rogue taxa. Note that, the RIC-optimizing approximation does not suffer from this problem, since this method does not prune rogue taxa with weak deleterious effects.

6.6. Other Criteria For Optimization

As discussed in Section 2.2, our optimality criterion is the RBIC of the MR consensus tree. In this Section, we outline, how results change, if we deploy the monolithic and iterative naïve algorithm for optimizing alternative criteria (the support in an MRE consensus tree and the bootstrap support drawn onto the best-known ML tree). Results for two representative datasets (with 128 and 1,604 taxa) are depicted in Figure 26.

As the first alternative, we consider the RBIC of the MRE consensus tree as an alternative optimization criterion. Since the bipartitions in the MR consensus tree are a subset of the bipartitions in the MRE consensus tree, the initial RBIC of the MRE consensus tree is significantly higher. However, this also means, that possibilities for RBIC improvement are more limited than in the case of the MR consensus tree.

Biologists also use bootstrap trees to asses the support of the bipartitions in the best-known ML tree. We can measure the RBIC of a ML tree with bootstrap support and deploy the two naïve algorithms for optimizing this version of the RBIC. For the datasets with 128 and 1,604 taxa (see Figure 26), the RBIC curves for the ML tree with bipartition support assume higher values for both datasets than for the MR consensus tree. However, they are lower than in the MRE consensus tree. This is, because it is probable that most bipartitions in the ML tree occur in the bootstrap trees. However, if the ML tree contains bipartitions with lower support, its RBIC will always be smaller than that of the MRE

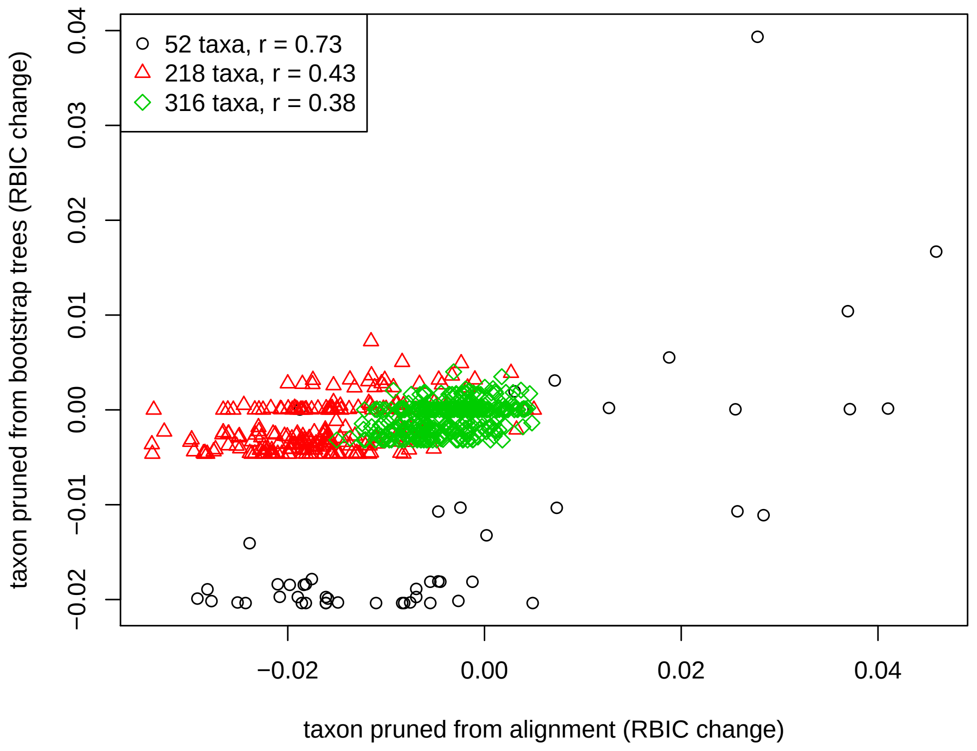


Figure 24: RBIC changes after pruning a taxon compared to RBIC of a collection of bootstrap trees computed without the respective taxon.

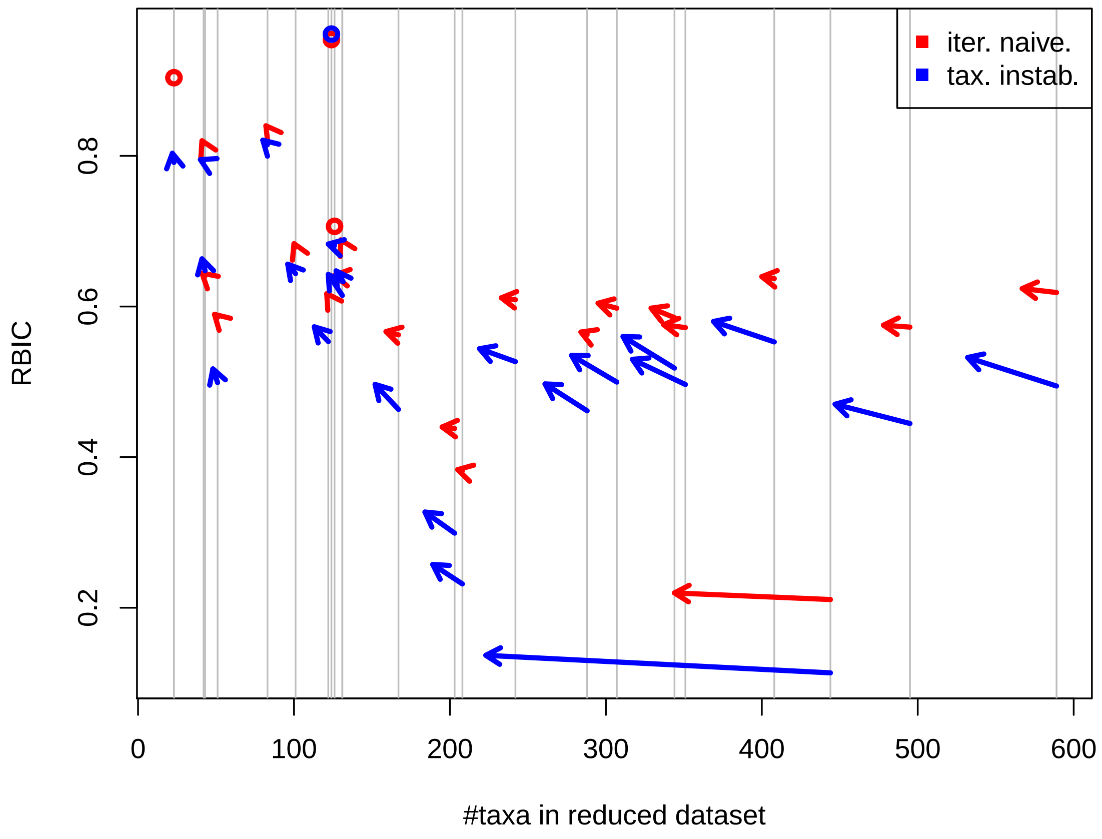


Figure 25: Possible RBIC improvements for all re-computed bootstrap trees (with rogue taxa identified by the iterative naïve algorithm and the taxonomic instability measure) that initially consisted of at most 885 taxa. Circles indicate that no rogue taxa were found. Number of taxa in the dataset are depicted on the x-axis.

consensus tree (where the greedy heuristic may decide for more frequent bipartitions) since MRE strives to reconstruct a binary consensus tree from the bootstrap replicates with maximum support.

For the 128-taxon dataset, we notice that considerable improvements with respect to the RBIC are possible for the ML tree with bipartition support and the MRE consensus tree. In case of the 1,604-taxon dataset, the naïve algorithms optimize both criteria (MRE and ML) only marginally, while the RBIC of the MR consensus tree is significantly increased.

6.7. Empirical Assessment

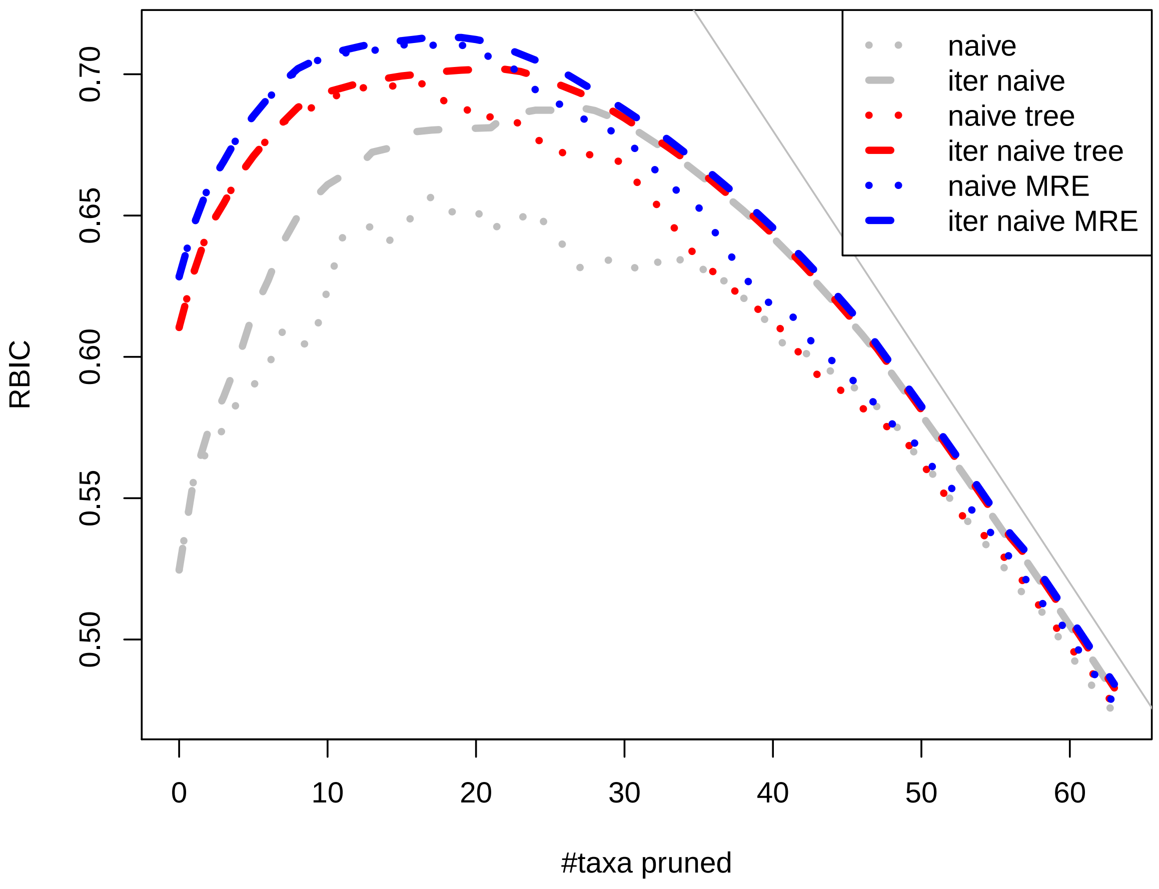
We asked biologists to evaluate the set of rogue taxa that were identified by the iterative naïve algorithm on the dataset they provided to us for our analysis. Furthermore, we were interested in their assessment of the initial unpruned consensus tree and the consensus tree without rogue taxa as identified by our algorithm.

For the 24-taxon dataset, we identified one rogue taxon (*ostrich*). Almost every method in this study, identifies this taxon as a rogue taxon. For instance, it has lowest leaf stability index. However, according to the taxonomic instability measure it is merely the third most unstable taxon. According to Matthew Phillips (who provided this dataset), it is not surprising that ostrich behaves as a rogue taxon, since it is the only taxon in the analysis without a close sister taxon. Phillips points out, that the re-computed consensus tree is better resolved. However, in the re-computed consensus tree the two outgroup taxa (*alligator* and *caiman*) and the *palaeognathae* taxa are closely related. Outgroup taxa are taxa that are distantly related to the taxa under study and are used for rooting a phylogenetic tree. The relationship between the outgroup taxa and the palaeognathae taxa contradicts mitochondrial, nuclear and morphological analysis and most probably is a consequence of a *long branch attraction* effect [reviewed in Bergsten, 2005].

For the 44-taxon dataset, most methods identify one highly deleterious rogue taxon. The iterative naïve algorithm identifies another rogue taxon that only slightly decreases support. Frank Kauff, who provided this dataset, acknowledges a noticeable increase in branch support after removing the rogue taxa. He notes that the highly deleterious rogue taxon is the only outgroup taxon in the study. In their research using this dataset, Kauff et al. focused on conflicts between single gene trees of the 5 partitions and phylogenomic trees inferred on a joint phylogenetic estimate using all partitions/genes simultaneously. Their goal was to remove sequences that caused these conflicts (topological incongruence). Interestingly, rogue taxa identified by our algorithm are not among the sequences that Kauff et al. found to be problematic.

Alexandra Wey provided the dataset with 53 taxa for which branches in the consensus tree are well supported and for which no algorithm was able to identify rogue taxa. Wey notes two probable instances of long branch attraction in the consensus tree and concludes that the respective subtrees do not make sense from a biological point of view. For comparability of the dataset, we used the Dayhoff matrix of protein substitution for all protein dataset. Wey points out that this is a sub-optimal choice for the 53-taxon dataset and suggests that different substitution models might be worthwhile investigating in the context of rogue taxa.

Karen Meusemann provided the datasets with 117 and 148 taxa. For the 148-taxon dataset, our algorithm identifies 17 rogue taxa. According to Meusemann, this number is smaller than expected, given the poorly resolved initial consensus tree. Meusemann notes that, many (but not all) of the rogue taxa identified by our algorithm were also



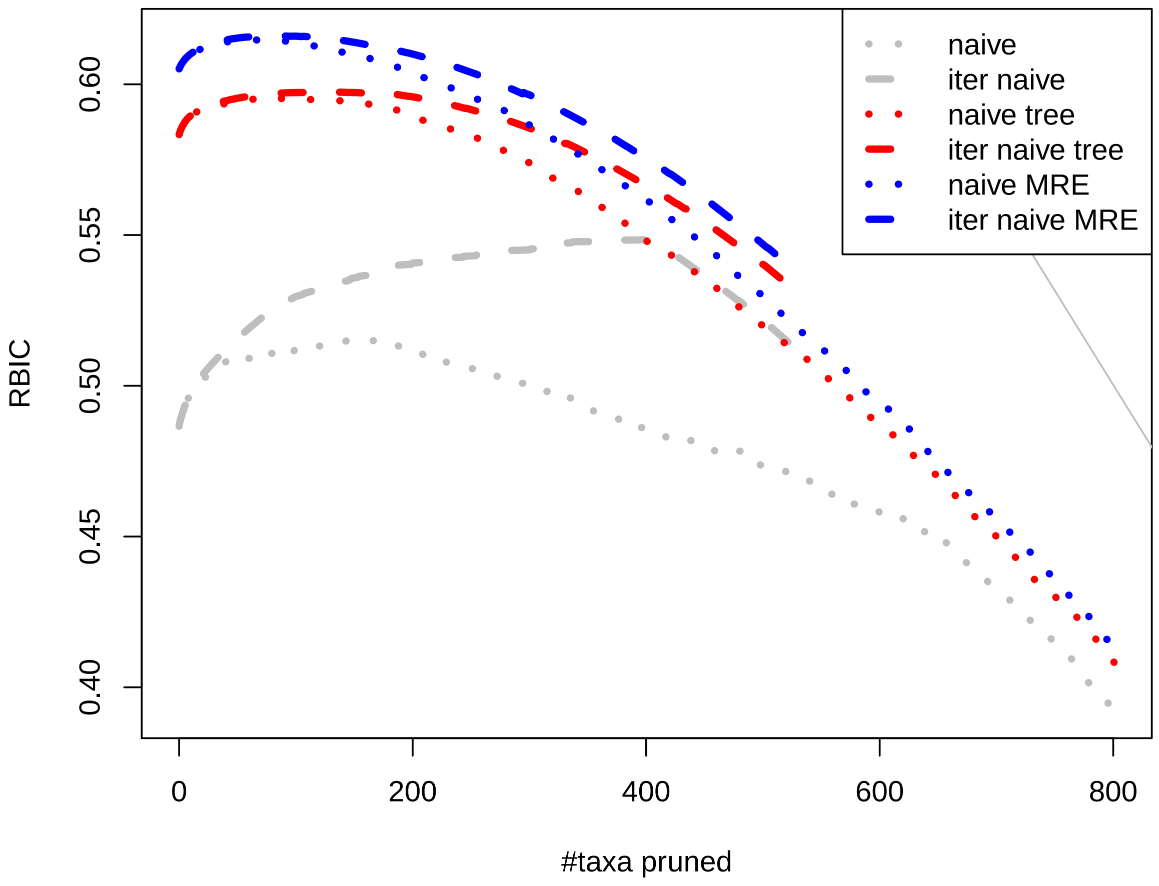


Figure 26: Deploying the monolithic and iterative naïve algorithm for optimization of the bipartition support in the ML tree (*naive tree* and *iter naive tree*) and the support in the MRE consensus tree (*naive MRE* and *iter naive MRE*) for datasets with **128** (top) and **1,604** (bottom) taxa.

only weakly supported in their Bayesian analysis [Von Reumont et al., 2009]. Among the taxa Meusemann suspected to be rogue taxa but that were not identified by our algorithm is a taxon with a particularly long branch. Also Meusemann notes, that there is no tendency for long branches in the rogue taxa as identified by our algorithm. This may be interesting, since biologists preferably remove taxa with long branches from their datasets to keep the overall tree size (evolutionary length of the tree) small.

For the 117-taxon dataset, our algorithm did not identify rogue taxa. Meusemann expected the dataset to contain rogue taxa, as some taxa were known to assume different positions in previous analyses [see Meusemann et al., 2010]. Meusemann deployed the leaf stability index for identification of rogue taxa on this dataset and pruned 5 taxa with a leaf stability index < 0.95. According to Meusemann, pruning these most unstable taxa induced some important changes in the tree topology. While the consensus tree of the 117-taxon dataset is well-resolved, Meusemann notices some questionable evolutionary relationships.

For the 128-taxon dataset (provided by Jonathan Fong), our approach identifies 27 rogue taxa. For obtaining improved bootstrap support values, Fong measured taxonomic instability and computed rogue taxa with the RIC-optimizing approximation. Furthermore, Fong removed taxa that were placed (partially with high support) in biologically unexpected regions of the tree. Fong recomputed bootstrap tree replicates after pruning rogue taxa and then repeated his procedure for rogue taxa identification. According to Fong, pruning rogue taxa that are identified in the second iteration does not improve bootstrap support values as much as the rogue taxa from the first iteration. This is in line with our observations (see Section 6.5.2). Fong notes, that there is a high degree of agreement between his methods and our approach for the first 10 taxa that are pruned/identified by our algorithm. However, among pruning steps briefly before the RBIC maximum is achieved, our approach prunes taxa that have a low proportion of missing characters and never appeared as rogue taxa in his analyses. The rogue taxa obtained from our algorithm comprise 7 of the 10 taxa, that Fong removes with regard to his empirical biological knowledge.

The alignment of the 424-taxon dataset (provided by Annie Lindgren) is a phylogenomic alignment composed of 21 genes. With 80 taxa that are identified as rogue taxa by the iterative naïve algorithm, this dataset contains a particularly high number of rogue taxa. This is interesting, because of the high number of missing characters (missing per-gene sequences) in the dataset (almost 80%, see Figure 13). In our analysis, we do not come to the conclusion, that a high proportion of gaps or undetermined characters necessarily yields a high number of rogue taxa (e.g., the datasets with 72, 2,000, and 7,764 taxa contain a high proportion of rogue taxa, but have at most 20% missing characters). Lindgren points out, that the gene coverage in the 424-taxon super-matrix is particularly low. A constraint for the taxa in the dataset is that, at least data for 3 per taxon must be present. A stricter cutoff for the number of available gene data would further reduce the taxon sampling in this dataset. According to Lindgren this low gene coverage may be the main reason why many taxa behave as rogue taxa in this dataset.

7. Conclusion and Outlook

In this thesis we discussed existing a posteriori methods for rogue taxa identification and analyzed small-scale pathological examples. Furthermore, we presented an a priori method for rogue taxa identification that does not require bootstrap trees as input.

We proposed an optimality criterion that is based on cumulative branch support rather than on the resolution of the consensus tree. We observed that, (before RIC-based optimization stops) the differences between the RIC-based and RBIC-based approaches are marginal.

This thesis constitutes the first comparative analysis of rogue taxa identification methods based on a common optimality criterion. We find that, the iterative naïve algorithm outperforms all alternative RBIC-based optimization methods. Consequently, we put the usage of the leaf stability index and the taxonomic instability measure for rogue taxa identification into question. We verified this result for the taxonomic instability measure by recomputing bootstrap trees after pruning the same number of unstable taxa (w.r.t. taxonomic instability) as pruned by the iterative naïve algorithm.

Global optimization algorithms (particularly genetic algorithms) may be well-suited for the RBIC or RIC optimization problem. However, we think that such an approach is too heavy-weight for determining a MISC: the results obtained for the iterative naïve algorithm (especially when compared to the dropset-based RBIC approximation and with respect to the single maximum observed in the RBIC curves) indicate that this simple method yields a good approximation for RBIC-MISC.

We found that, the iterative naïve algorithm performs significantly better than its monolithic counterpart. This does not hold for the iterative versions of the leaf stability index and the taxonomic instability measure. Thus, we can exclude that, the reason for the moderate performance of the stability measures is the same as for the monolithic naïve algorithm. This supports our finding that rogue taxa may be among the most unstable taxa, but do not necessarily have to be.

We also tested a compute-intensive dropset-based RBIC-approximation and observed that, there are only two instances where this algorithm outperforms the iterative naïve algorithm. Although this dropset-based algorithm is implemented efficiently, its high computational requirements do not justify the anecdotal RBIC improvement. We conclude that, the dropset-based algorithms are more powerful than the naïve algorithms from a theoretic point of view. However, we find that, for real-world datasets, a large number of dropsets identified just contain a single taxon. This means that, the naïve algorithm is capable to identify such trivial dropsets as well. For dropsets containing more than one taxon, we can not be sure whether they are just an accumulation of taxa that are dropped by the naïve algorithms in single-taxon steps or if they indeed provide a better solution that could not be achieved using the naïve algorithms (e.g., because of pathological constellations of rogue taxa). Thus, we think that, the iterative naïve algorithm suffices for optimizing the RBIC and represents a good trade-off between speed and accuracy. On the other hand, because of Pattengale's fast approximation, the dropset-based approach is an excellent choice for optimizing the RIC.

On average, we improved the runtime of Pattengale's algorithm by two to three orders of magnitude and also optimized our implementation with respect to memory requirements. Thereby, the algorithm can be applied to datasets that are nowadays considered to be relatively large. We achieved this improvement by reducing the number of candidate dropsets that are examined by the algorithm without losing accuracy. A more detailed study of dropset features might reveal further optimization possibilities. If the pool of dropset candidates with a high potential for improving optimality could be narrowed down further, approximative examination of such a pool of dropsets could be implemented as an extension to the naïve algorithm. In summary, our implementation of the RIC-optimizing approximation and the iterative naïve algorithm offers two efficient and powerful options for rogue taxa identification. Furthermore, with an implementation of the MAST and appropriate adaptations of the taxonomic instability measure and the leaf stability index for unrooted trees, the programs implemented in the scope of this thesis also represent a versatile and efficient toolbox summarizing collections of trees and computing node stability measures.

We think that, the runtime of the iterative naïve algorithm can potentially be further improved. We demonstrated that, the iterative approach is important with respect to accuracy. However, for many taxa, the RBIC change they induce (if pruned), does not change between iterations, if other taxa are pruned. Re-scoring the RBIC improvement of just some suspicious rogue taxa could help to improve the runtime of the algorithm. Furthermore, the current implementation is well-suited for parallelization. Because of the coarse-grain dependencies in the algorithm, a parallelized version of the iterative naïve algorithm will scale up to the largest datasets that were examined in this thesis.

Our a priori methods partially performed well. However, the lack of an a priori stopping criterion coupled with highly variable performance (accuracy) of these methods on many datasets indicates that placement uncertainty may be related to rogue taxa, but does not necessarily predict rogues when the RBIC is used as a criterion.

We pruned rogue taxa identified by the taxonomic instability measure and by the iterative naïve algorithm from the alignment and recomputed bootstrap trees. Note that, we did not realign the sequences in the pruned alignment. Realigning as well as rogue taxa identification during the process of aligning may be highly powerful means for avoiding rogue taxon effects that has not been addressed in thesis. We do not observe that the support (w.r.t. the RBIC) in the recomputed bootstrap trees deviates significantly from the support in the pruned initial bootstrap trees. Nevertheless, based on our experiments, we do recommend to recompute bootstrap trees from a pruned (and realigned) alignment, since the topology of the consensus tree may change (as observed, e.g., by Meusemann).

The overall feedback from biologists regarding the results obtained with the iterative naïve algorithm was positive. However, we conclude from Fong's assessment that, taxa that are pruned some iterations before the RBIC maximum is reached, may not be genuine rogue taxa. In this context, it is interesting, that weak rogue taxon effects could be detected in the recomputed, pruned bootstrap trees. This indicates that the iterative naïve algorithm may tend to "overprune".

Interestingly, in some cases, outgroup taxa were identified as rogue taxa. Furthermore, many biologists raised concerns about biologically questionable relationships in our results (consensus trees) because of potential long branch attraction (LBA) effects. Beside rogue taxa, LBA is another important problem in phylogenetic studies. It may be worthwhile to investigate the relationship between rogue taxa and LBA effects and to re-visit a priori methods (resp. the EPA algorithm) in the context of automated LBA detection.

A. Implementation: A Posteriori Methods

A.1. Maximum Agreement Subtree

The set of triplets C that occur in every bootstrap tree is implemented as an array of bit vectors (`unsigned int***`). The bth bit at position $[a][x]$ of the array represents a triplet $((a, x), b)$. This representation allows for quickly computing the intersection of triplets in the input trees via a bit-wise *and*-operation. We can represent a rooting r of the input trees by starting the triplet extraction at node r. Starting from r, we perform a depth first search (DFS). For each inner node we visit, we keep track of the left and right children of the node using two separate index lists (one for the left subtree and one for the right subtree) as well as bit vectors containing the same information. Each combination of two taxa from the left subtree and one taxon from the right subtree of an inner node yields a triplet. Analogously, selecting two taxa from the right subtree and one from the left subtree also generates a triplet. On the one hand, the dual representation as bit vector and index list allows for rapid insertion of rooted triplets into C via a bit-wise *or*. On the other hand, we obtain direct access to the index set in the bit vector without having to iterate through the entire bit vector.

Once we have computed C, we can start filling the dynamic programming matrix A. The order by which the elements of A are computed is determined by another DFS tree traversal starting at node r. We can use any tree for this DFS, as the rooted triplets of C which are examined in this step are, by definition, contained in any possible tree. If the program is called with the option to generate all possible MASTs, we need to store all possible choices for x and y (see Section 3.1) in separate index lists for each matrix element. Finally, the leaf sets of the MAST are produced via a backtracking operation that starts at the matrix element with the highest $\text{mast}(a, b)$ value. During this backtracking procedure, all possible combinations of x and y are explicitly represented as bit vectors (if the user opted for generating all possible MASTs). To reduce the impact of the combinatorial explosion, we perform a filtering operation after each move in the matrix in order to identify and filter duplicate agreement trees (that exist because of redundant paths through the matrix). When computing all MASTs, we have to carry out this backtracking procedure for each matrix element with maximum leaf set size in each rooting. A final filtering is performed to eliminate duplicates that occurred for different backtracking processes.

A.2. Naïve Pruning Algorithm

Bipartitions are implemented as bit vectors in RAxML. A bit that is set indicates that the corresponding taxon is in the same partition as a dedicated reference taxon [Pattengale et al., 2010b]. A bipartition profile is represented as a hash table of bit vectors. For hashing, we initially generate a random number for each taxon. As hash function for bipartitions, we use the bit-wise *exclusive or* (computed recursively in the tree) of the random numbers of the taxa that are in the same partition as the reference taxon.

We implemented two versions of the RBIC measure. Besides the naïve algorithm, we also use RBIC for assessing the optimality of all other methods considered in this thesis. Furthermore, this RBIC measure forms the basis the dropset-based RBIC optimization algorithm (see Section 3.4).

The first implementation uses the optimized [Aberer et al., 2010] consensus tree building routines in RAxML. In order to test the effect of pruning a taxon (or a set of taxa), all bootstrap trees have to be re-read and parsed again from the input file and are pruned accordingly. Subsequently, bipartitions are extracted by traversing all trees and inserted into the hash table accordingly. We then determine the consensus tree bipartitions and compute the RBIC.

The second implemented avoids the unnecessary file I/O, tree parsing, and tree traversal steps. Instead of pruning the input trees, we construct a bipartition profile (hash table) of the unpruned trees in the very beginning. For each taxon or set of taxa to be scored under RBIC, we then just make a copy of the bipartition profile in memory, prune it accordingly and score the result. The pruning step requires rehashing the pruned bipartitions in the hash table. Thus, the complexity this more efficient implementation is linear in the number of bipartitions. This number is always smaller than $\mathcal{O}(nk)$ (the complexity of the first implementation with n taxa and k trees). For some of our test datasets, we observed that, the second implementation was faster by up to two orders of magnitude. Furthermore, the bipartition profile based implementation also allows for efficient parallelization. Parallel scoring of many different prunings would be problematic using the initial implementation due to the I/O step.

A.3. Maximum-Information Subtree Consensus

As for the naïve algorithm (see Section A.2), we represent the bipartition profile as a hash table. For the extraction of the bipartition profile, the methods implemented for the consensus tree algorithm are used. For each hash element (i.e., each bipartition), we use in a separate bit vector to store the set of trees in which the bipartition occurs as well as the absolute frequency of the bipartition.

Dropsets are stored in a separate hash table. Again, a hash function that is based on random numbers is used. Each hash element contains the dropset itself and an integer counting the (approximated) number of bipartitions that will be included in the consensus tree due to a merging event, if the dropset is pruned. Because the bipartitions are represented as bit vectors, we can use the bit-wise exclusive-or of two bipartitions to compute their symmetric difference (which is required for dropset generation). However, bit vectors only represent one of the two partitions of a bipartition – the other partition is implicitly represented as all taxa bits that are not set. Thus, for computing the alternative dropset of a pair of bipartitions (see Section 3.3), we need to compute the complement of one of the two bit vector representations. For computing the complement partition, we can use the bit-wise complement, but have to account for so-called *padding bits*: these are "junk" bits that do not represent taxa. They are part of the bit vector since we have to allocate a bit vector in units of 32-bit integers.

It is straight-forward to also store the dropsets as bit-vectors. However, this becomes problematic because the number of dropset candidates increases quadratically with the number of bipartitions. Thus, on large datasets (such as 1,000 trees with 2,000 taxa) the memory requirements can become prohibitive. We found that, the dropsets are usually rather sparse, that is they consist only of few taxa. Thus, we chose to convert the bit-vector representation of a dropset (directly obtained from the exclusive-or operation) into an index list. The bit-vector representation is more memory efficient when a dropset contains of at least n/x taxa, where n is the size of the original taxon set and x the number of bits per integer (in our case $x = 32$).

With respect to computational complexity, the dropset generation for all pairs of bipartitions is a dominating factor. We found a way to omit the computation of a large part of irrelevant dropsets. One can initially sort the bipartitions in descending order according to their frequency of their occurrence in the bootstrap trees. Then, we iterate over this list and search for bipartition pairs $(\mathbf{A}_i, \mathbf{B}_j)$, such that $sup(\mathbf{A}_i) \geq sup(\mathbf{B}_j)$ and $sup(\mathbf{B}_{j-1}) \geq sup(\mathbf{B}_j)$. If the cumulative support of $\mathbf{A}_i$ and $\mathbf{B}_j$ does not exceed the frequency threshold of the consensus method, we know that we do not need to check bipartition $\mathbf{A}_i$ against other element in the list because all combinations of $\mathbf{A}_i$ with any other element in the list will not generate a merged bipartition that exceeds the consensus frequency threshold.

We also attempted to parallelize the algorithm by distributing the generation of dropsets to worker threads. However, we found that, the parallel region of the code only accounts for a minor part of overall runtime. Thus, no significant speed-up could be achieved. Furthermore, in the parallel part read-write hash-table accesses consume the largest part of runtime. We have already observed that, parallel operations on hash-tables might perform poorly because of bandwidth limitations and irregular memory access patterns [Aberer et al., 2010].

In each iteration of the algorithm, we need to recompute the bipartition profile without the taxa that already have been identified as rogues. As for the naïve algorithm (see Section A.2), the bipartitions are not explicitly extracted from the input trees again. Instead, the bipartition profile (hash table) is restricted (i.e., the dropset is pruned from the bipartitions by setting respective bits to zero). The bipartitions are then simply rehashed for obtaining the desired merging events.

A.4. Taxonomic Instability

We store the unweighted patristic distances between all taxa in all trees in a $n \times n \times k$ matrix, where n is the number of taxa and k the number of trees. As the memory requirements for this matrix increase by $\mathcal{O}(n^2k)$, we allocate the matrix as `unsigned short int***`. This allows us to keep the matrix size relatively small. However, it limits the scalability of the implementation to input datasets with slightly more than 65,000 taxa. In the worst case scenario, depending on the tree shape, longer paths through the trees can occur than we can represent by short integers (16 bits).

Once the unweighted patristic distances are computed and stored in the matrix, the computation of the taxonomic instability is straight-forward and can easily be parallelized. Computing the distances in advance is more efficient than using ad hoc on-demand approach during the calculation of the actual measure. Firstly, this avoids computing each distance twice, because each summand of Equation (14) appears in the taxonomic instability of two taxa. Secondly, we can efficiently extract patristic distances using a single DFS traversal per tree. We choose an arbitrary leaf as a virtual root for the traversal. If we encounter a leaf, we create a list with the leaf as its only element and an attribute indicating the distance of this leaf to the current node (which equals zero initially). When traversing inner nodes, we increment this distance attribute accordingly. Let x be a node in the left subtree of inner node i and y be a node from the right subtree. The distance attribute d_{xi} respectively d_{yi} indicates the distance from leave x respectively y to the inner node i. Thus, the distance between x and y is: $d_{xy} = d_{xi} + d_{yi}$. Finally, we join both lists and return the result (with incremented distance attributes) to the next inner node in the traversal.

A.5. Leaf Stability Index

For computing of the leaf stability index of a, we need to store the frequency of all quartets of taxa that contain a. The extraction of quartets that contain a from a tree is equivalent to the extraction of all rooted triplets from a tree rooted at a. Thus, the implementation of this step is analogous to the extraction of rooted triplets described in Section A.1.

The pre-computation of all quartets has memory requirements that are in $\mathcal{O}(n^4)$, where n is the number of taxa. Thus, space becomes prohibitive for large datasets (e.g., 1,000 taxa). Thus, we opted for a more compute-intensive implementation: when computing the leaf instability index for taxon a, we extract only those quartets that contain a. This means, that we have to actually re-compute the quartet frequencies for each taxon. However, this limits space requirements to $\mathcal{O}(n^3)$. As for the taxonomic instability (see Section A.4), we allocate the quartet frequency matrix as `unsigned short int***`. Thus, the implementation can not handle more than 65,535 trees. This is sufficient, as most current phylogenetic studies use at most 1,000 to 10,000 trees.

B. Implementation: A-Priori Methods

B.1. Expected Distance between Placement Locations

The computation of weighted patristic distances follows the same pattern as the computation of unweighted patristic distances for the taxonomic instability measure described in Section A.4. Unlike for the taxonomic instability measure, we have to keep track of the distances of leaves to inner nodes and inner nodes among each other.

For the sake of accuracy, we re-compute the distances for each pruning of the best-known ML tree. As the branch lengths may change after pruning the taxon for which the EDPL is determined, the lengths of the neighboring branches are optimized before distance computation. For the distance between two placements, we do not keep track of the exact placement of a taxon on a branch, but we use half of the branch length as an approximation.

B.2. Iterative A Priori Algorithms

We implemented the a priori algorithm in a shell script that executes the RAxML operations that are necessary in each iteration. The RF metric and SH-like support values are efficiently implemented in RAxML. For the RF metric we used the "`-f r`" and for the SH-like support values the "`-f J`" option of RAxML.

References

A. J. Aberer, N. D. Pattengale, and A. Stamatakis. Parallelized phylogenetic post-analysis on multi-core architectures. *Journal of Computational Science*, 1(2):107 – 114, 2010. ISSN 1877-7503. doi: 10.1016/j.jocs.2010.03.006. URL http://www.sciencedirect.com/science/article/B9HC1-4YTN466-1/2/cb6652a60c8bb7f35a500fb5fe12e3a7.

M. Anisimova and O. Gascuel. Approximate likelihood-ratio test for branches: A fast, accurate, and powerful alternative. *Syst Biol*, 55(4):539–552, Aug 2006. doi: 10.1080/10635150600755453. URL http://dx.doi.org/10.1080/10635150600755453.

S. A. Berger and A. Stamatakis. Evolutionary Placement of Short Sequence Reads. *ArXiv e-prints*, Nov. 2009.

J. Bergsten. A review of long-branch attraction. *Cladistics*, 21(2):163–193, 2005. ISSN 1096-0031.

V. Berry and F. Nicolas. Improved Parameterized Complexity of the Maximum Agreement Subtree and Maximum Compatible Tree Problems. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(3):289–302, 2006. ISSN 1545-5963. doi: http://dx.doi.org/10.1109/TCBB.2006.39.

V. Berry and F. Nicolas. Maximum agreement and compatible supertrees. *Journal of Discrete Algorithms*, 5(3):564–591, 2007. ISSN 1570-8667.

D. Bryant. Hunting for trees in binary character sets: efficient algorithms for extraction, enumeration, and optimization. *J Comput Biol*, 3(2):275–288, 1996.

D. Bryant. A classification of consensus methods for phylogenetics. In *Bioconsensus: DIMACS Working Group Meetings on Bioconsensus: October 25-26, 2000 and October 2-5, 2001, DIMACS Center*, page 163. Amer Mathematical Society, 2003. ISBN 0821831976.

W. Chen, C. Bonillo, and G. Lecointre. Repeatability of clades as a criterion of reliability: a case study for molecular phylogeny of Acanthomorpha (Teleostei) with larger number of taxa. *Molecular Phylogenetics and Evolution*, 26(2):262–288, 2003. ISSN 1055-7903.

B. Chor and T. Tuller. Maximum likelihood of evolutionary trees: hardness and approximation. *Bioinformatics*, 21 Suppl 1:i97–106, Jun 2005. doi: 10.1093/bioinformatics/bti1027. URL http://dx.doi.org/10.1093/bioinformatics/bti1027.

R. Cole and R. Hariharan. An O (n log n) algorithm for the maximum agreement subtree problem for binary trees. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, page 332. Society for Industrial and Applied Mathematics, 1996. ISBN 0898713668.

K. A. Cranston and B. Rannala. Summarizing a posterior distribution of trees using agreement subtrees. *Syst Biol*, 56(4):578–590, Aug 2007. doi: 10.1080/10635150701485091. URL http://dx.doi.org/10.1080/10635150701485091.

M. Cueto and F. Matsen. Polyhedral Geometry of Phylogenetic Rogue Taxa. *Bulletin of Mathematical Biology*, pages 1–25, 2010. ISSN 0092-8240. URL http://dx.doi.org/10.1007/s11538-010-9556-x. 10.1007/s11538-010-9556-x.

C. R. Darwin. *The origin of species*. John Murray, 1859.

C. W. Dunn, A. Hejnol, D. Q. Matus, K. Pang, W. E. Browne, S. A. Smith, E. Seaver, G. W. Rouse, M. Obst, G. D. Edgecombe, M. V. Sørensen, S. H. D. Haddock, A. Schmidt-Rhaesa, A. Okusu, R. M. Kristensen, W. C. Wheeler, M. Q. Martindale, and G. Giribet. Broad phylogenomic sampling improves resolution of the animal tree of life. *Nature*, 452(7188):745–749, Apr 2008. doi: 10.1038/nature06614. URL http://dx.doi.org/10.1038/nature06614.

J. Farris. Estimating phylogenetic trees from distance matrices. *American Naturalist*, 106 (951):645–668, 1972. ISSN 0003-0147.

J. Felsenstein. Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution*, 39(4):pp. 783–791, 1985. ISSN 00143820. URL http://www.jstor.org/stable/2408678.

J. Felsenstein. Inferring phytogenies. *Sunderland, Massachusetts: Sinauer Associates*, 2004.

J. Felsenstein, J. Archie, W. Day, W. Maddinson, C. Meacham, F. Rohlf, and D. Swofford. The Newick tree format, 1986.

C. Finden and A. Gordon. Obtaining common pruned trees. *Journal of Classification*, 2 (1):255–276, 1985. ISSN 0176-4268.

W. Gilbert. Origin of life: The RNA world. *Nature*, 319(6055):618–618, Feb. 1986. URL http://dx.doi.org/10.1038/319618a0.

P. Goloboff, J. Farris, and K. Nixon. TNT, a free program for phylogenetic analysis. *Cladistics*, 24(5):774–786, 2008. ISSN 1096-0031.

S. Guindon and O. Gascuel. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic biology*, 52(5):696, 2003. ISSN 1063-5157.

B. Hodkinson and F. Lutzoni. A microbiotic survey of lichen-associated bacteria reveals a new lineage from the Rhizobiales. *Symbiosis*, 49(3):163–180, 2009. ISSN 0334-5114.

M. Holder and P. O. Lewis. Phylogeny estimation: traditional and Bayesian approaches. *Nat Rev Genet*, 4(4):275–284, Apr 2003. doi: 10.1038/nrg1044. URL http://dx.doi.org/10.1038/nrg1044.

J. P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, Aug 2001.

E. Kubicka, G. Kubicki, and F. McMorris. On agreement subtrees of two binary trees. In *Combinatorics, graph theory and computing: 23rd Southeastern international conference: Selected papers*, page 217. Utilitas Mathematica, 1992. ISBN 0919628885.

N. Lartillot, T. Lepage, and S. Blanquart. PhyloBayes 3: a Bayesian software package for phylogenetic reconstruction and molecular dating. *Bioinformatics*, 25(17):2286, 2009. ISSN 1367-4803.

M. S. Y. Lee. Choosing reference taxa in phylogenetic nomenclature. 2005. URL http://hdl.handle.net/2440/32053.

W. P. Maddison and D. Maddison. Mesquite: a modular system for evolutionary analysis, 2010. URL http://mesquiteproject.org. Version 2.73.

T. Margush and F. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43 (2):239–244, 1981. ISSN 0092-8240.

F. A. Matsen, R. B. Kodner, and E. V. Armbrust. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. Mar. 2010.

M. L. Metzker. Sequencing technologies - the next generation. *Nat Rev Genet*, 11(1): 31–46, Jan 2010. doi: 10.1038/nrg2626. URL http://dx.doi.org/10.1038/nrg2626.

K. Meusemann, B. M. von Reumont, S. Simon, F. Roeding, S. Strauss, P. Kück, I. Ebersberger, M. Walzl, G. Pass, S. Breuers, V. Achter, A. von Haeseler, T. Burmester, H. Hadrys, J. W. Wägele, and B. Misof. A phylogenomic approach to resolve the arthropod tree of life. *Mol Biol Evol*, 27(11):2451–2464, Nov 2010. doi: 10.1093/molbev/msq130. URL http://dx.doi.org/10.1093/molbev/msq130.

B. M. Moret, L. Nakhleh, T. Warnow, C. R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic Networks: Modeling, Reconstructibility, and Accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1:13–23, 2004. ISSN 1545-5963. doi: http://doi.ieeecomputersociety.org/10.1109/TCBB.2004.10.

M. Mutanen, N. Wahlberg, and L. Kaila. Comprehensive gene and taxon coverage elucidates radiation patterns in moths and butterflies. *Proc Biol Sci*, 277(1695):2839–2848, Sep 2010. doi: 10.1098/rspb.2010.0392. URL http://dx.doi.org/10.1098/rspb.2010.0392.

L. Parfrey, J. Grant, Y. Tekle, E. Lasek-Nesselquist, H. Morrison, M. Sogin, D. Patterson, and L. Katz. Broadly sampled multigene analyses yield a well-resolved eukaryotic tree of life. *Systematic Biology*, 2010. ISSN 1063-5157.

N. Pattengale, K. Swenson, and B. Moret. Uncovering Hidden Phylogenetic Consensus. In M. Borodovsky, J. Gogarten, T. Przytycka, and S. Rajasekaran, editors, *Bioinformatics Research and Applications*, volume 6053 of *Lecture Notes in Computer Science*, pages 128–139. Springer Berlin / Heidelberg, 2010a.

N. D. Pattengale, M. Alipour, O. R. P. Bininda-Emonds, B. M. E. Moret, and A. Stamatakis. How many bootstrap replicates are necessary? *J Comput Biol*, 17(3):337–354, Mar 2010b. doi: 10.1089/cmb.2009.0179. URL http://dx.doi.org/10.1089/cmb.2009.0179.

C. Phillips and T. Warnow. The asymmetric median tree–A new model for building consensus trees. *Discrete Applied Mathematics*, 71(1-3):311–335, 1996. ISSN 0166-218X.

M. Phillips, G. Gibb, E. Crimp, and D. Penny. Tinamous and moa flock together: mitochondrial genome sequence analysis reveals independent losses of flight among ratites. *Systematic biology*, 2009. ISSN 0039-7989.

D. Pisani, A. Yates, M. Langer, and M. Benton. A genus-level supertree of the Dinosauria. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 269(1494): 915, 2002. ISSN 0962-8452.

D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Math. Biosci.*, 53:131–147, 1981. ISSN 0025-5564.

M. J. Sanderson and H. B. Shaffer. Troubleshooting molecular phylogenetic analyses. *Annual Review of Ecology and Systematics*, 33(1):49–72, 2002. doi: 10.1146/annurev.ecolsys.33.010802.150509. URL http://www.annualreviews.org/doi/abs/10.1146/annurev.ecolsys.33.010802.150509.

H. A. Schmidt, K. Strimmer, M. Vingron, and A. von Haeseler. TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3):502–504, Mar 2002.

L. L. Shadwick, F. W. Spiegel, J. D. L. Shadwick, M. W. Brown, and J. D. Silberman. Eumycetozoa = Amoebozoa?: SSUrDNA phylogeny of protosteloid slime molds and its significance for the amoebozoan supergroup. *PLoS One*, 4(8):e6754, 2009. doi: 10.1371/journal.pone.0006754. URL http://dx.doi.org/10.1371/journal.pone.0006754.

A. B. A. Shafer and J. C. Hall. Placing the mountain goat: a total evidence approach to testing alternative hypotheses. *Mol Phylogenet Evol*, 55(1):18–25, Apr 2010. doi: 10.1016/j.ympev.2010.01.015. URL http://dx.doi.org/10.1016/j.ympev.2010.01.015.

C. Shannon and W. Weaver. The mathematical theory of communication (Urbana, IL, 1949.

S. A. Smith and C. W. Dunn. Phyutility: a phyloinformatics tool for trees, alignments and molecular data. *Bioinformatics*, 24(5):715–716, Mar 2008. doi: 10.1093/bioinformatics/btm619. URL http://dx.doi.org/10.1093/bioinformatics/btm619.

E. A. Sperling, K. J. Peterson, and D. Pisani. Phylogenetic-signal dissection of nuclear housekeeping genes supports the paraphyly of sponges and the monophyly of Eumetazoa. *Mol Biol Evol*, 26(10):2261–2274, Oct 2009. doi: 10.1093/molbev/msp148. URL http://dx.doi.org/10.1093/molbev/msp148.

A. Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, Nov 2006a. doi: 10.1093/bioinformatics/btl446. URL http://dx.doi.org/10.1093/bioinformatics/btl446.

A. Stamatakis. Phylogenetic models of rate heterogeneity: a high performance computing perspective. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, page 278. IEEE, 2006b.

A. Stamatakis, P. Hoover, and J. Rougemont. A rapid bootstrap algorithm for the RAxML web servers. *Systematic Biology*, 57(5):758, 2008. ISSN 1063-5157.

K. Strimmer and A. Rambaut. Inferring confidence sets of possibly misspecified gene trees. *Proc Biol Sci*, 269(1487):137–142, Jan 2002. doi: 10.1098/rspb.2001.1862. URL http://dx.doi.org/10.1098/rspb.2001.1862.

C. Sumrall, C. Brochu, and J. Merck. Global lability, regional resolution, and majority-rule consensus bias. *PALEOBIOLOGY*, 27(2):254–261, SPR 2001. ISSN 0094-8373.

D. L. Swofford. PAUP*: phylogenetic analysis using parsimony, version 4.0b10, 2003.

M. Thines, M. Göker, O. Spring, and F. Oberwinkler. A revision of Bremia graminicola. *Mycol Res*, 110(Pt 6):646–656, Jun 2006. doi: 10.1016/j.mycres.2006.04.001. URL http://dx.doi.org/10.1016/j.mycres.2006.04.001.

R. C. Thomson and H. B. Shaffer. Sparse supermatrices for phylogenetic inference: taxonomy, alignment, rogue taxa, and the phylogeny of living turtles. *Syst Biol*, 59(1): 42–58, Jan 2010a. doi: 10.1093/sysbio/syp075. URL http://dx.doi.org/10.1093/sysbio/syp075.

R. C. Thomson and H. B. Shaffer. Rapid progress on the vertebrate tree of life. *BMC Biol*, 8:19, 2010b. doi: 10.1186/1741-7007-8-19. URL http://dx.doi.org/10.1186/1741-7007-8-19.

Thorley and Wilkinson. Testing the phylogenetic stability of early tetrapods. *J Theor Biol*, 200(3):343–344, Oct 1999. doi: 10.1006/jtbi.1999.0999. URL http://dx.doi.org/10.1006/jtbi.1999.0999.

J. Thorley. Cladistic information, leaf stability and supertree construction. 2000.

B. Von Reumont, K. Meusemann, N. Szucsich, E. Dell'Ampio, V. Gowri-Shankar, D. Bartel, S. Simon, H. Letsch, R. Stocsits, Y. Luan, et al. Can comprehensive background knowledge be incorporated into substitution models to improve phylogenetic analyses? A case study on major arthropod relationships. *BMC Evolutionary Biology*, 9(1):119, 2009. ISSN 1471-2148.

M. Wilkinson. Common Cladistic Information and its Consensus Representation: Reduced Adams and Reduced Cladistic Consensus Trees and Profiles. *Systematic Biology*, 43(3):343–368, 1994. doi: 10.1093/sysbio/43.3.343. URL http://sysbio.oxfordjournals.org/content/43/3/343.abstract.

M. Wilkinson. More on Reduced Consensus Methods. *Systematic Biology*, 44(3):pp. 435–439, 1995. ISSN 10635157. URL http://www.jstor.org/stable/2413604.

M. Wilkinson. Majority-rule reduced consensus trees and their use in bootstrapping. *Mol Biol Evol*, 13(3):437–444, Mar 1996.

M. Wilkinson. Identifying stable reference taxa for phylogenetic nomenclature. *Zoologica Scripta*, 35(1):109–112, January 2006. ISSN 0300-3256. doi: 10.1111/j.1463-6409.2005.00213.x. URL http://dx.doi.org/10.1111/j.1463-6409.2005.00213.x.

M. Wilkinson, J. Cotton, and J. Thorley. The information content of trees and their matrix representations. *Syst Biol*, 53(6):989–1001, Dec 2004. doi: 10.1080/10635150490522737. URL http://dx.doi.org/10.1080/10635150490522737.

F. Wintzingerode, U. GoŰbel, and E. Stackebrandt. Determination of microbial diversity in environmental samples: pitfalls of PCR-based rRNA analysis. *FEMS Microbiology Reviews*, 21(3):213–229, 1997. ISSN 1574-6976.

D. Zwickl. *Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion.* PhD thesis, The University of Texas at Austin, 2006.