Syst. Biol. 0(0):1–11, 2014

DOI:10.1093/sysbio/syu051

Bias in Tree Searches and its Consequences for Measuring Group Supports

PABLO A. GOLOBOFF[1,*] AND MARK P. SIMMONS[2]

[1]*CONICET, INSUE, Fundación Miguel Lillo, Miguel Lillo 251, 4000 S.M. de Tucumán, Argentina; and* [2]*Department of Biology, Colorado State University, Fort Collins, CO 80523, USA*

[*]*Correspondence to be sent to: Fundación Miguel Lillo, Miguel Lillo 251, 4000 S.M. de Tucumán, Argentina; E-mail: pablogolo@yahoo.com.ar.*

Received 20 September 2013; reviews returned 21 February 2014; accepted 15 July 2014
Associate Editor: Dan Faith

Abstract.—When doing a bootstrap analysis with a single tree saved per pseudoreplicate, biased search algorithms may influence support values more than actual properties of the data set. Two methods commonly used for finding phylogenetic trees consist of randomizing the input order of species in multiple addition sequences followed by branch swapping, or using random trees as the starting point for branch swapping. The randomness inherent to such methods is assumed to eliminate any consistent preferences for some trees or unsupported groups of taxa, but both methods can be significantly biased. In the case of trees created by sequentially adding taxa, a bias may occur even if every addition sequence is equiprobable, and if one of the equally optimal positions for each terminal to add to the tree is selected equiprobably. In the case of branch swapping, the bias can happen even when branch swapping equiprobably selects any of the trees of better score in the subtree-pruning-regrafting-neighborhood or tree-bisection-reconnection-neighborhood. Consequently, when the data set is ambiguous, both random-addition sequences and branch swapping from random trees may (i) find some of the optimal trees much more frequently than others and (ii) find some groups with a frequency that differs from their frequency among all optimal trees. When the data set defines a single optimal tree, the groups present in that tree may have a different probability of being found by a search, even if supported by equal amounts of evidence. This may happen in both parsimony and maximum-likelihood analyses, and even in small data sets without incongruence. [Bootstrap; group support; jackknife; phylogenetic analysis; resampling; tree searches.]

In phylogenetic analysis, the degree of support for groups is most commonly evaluated by resampling the original data set and recalculating the tree, as in bootstrapping (Felsenstein 1985) and jackknifing (Farris et al. 1996). When a pseudoreplicate supports alternative trees, some authors propose using the strict consensus to summarize those results (Farris et al. 1996; Davis et al. 1998; Goloboff et al. 2003; De Laet et al. 2004). But the most common approach is weighting each group found by its frequency among the optimal trees for the pseudoreplicate, as implemented in PAUP* (Swofford 1998) and Phylip (Felsenstein 2005). Felsenstein (2004, p. 339) defended this approach on the grounds that, as more trees are tied for a pseudoreplicate, the lower the weight each tree has to receive to avoid overemphasizing the results of that pseudoreplicate when the trees are subsequently combined. When using such a treatment, groups that are absent from some but present in the majority of optimal trees will tend to be considered as strongly supported (see Simmons and Goloboff [2013], and references therein). Such groups will also be considered as strongly supported (Goloboff and Pol 2005; Yang 2006) by the standard way of summarizing the results of a Markov chain in a Bayesian analysis (Yang and Rannala 1997), that is, from considering the posterior probability of a group to be the frequency of trees sampled from the posterior having that group (Larget and Simon 1999; Huelsenbeck et al. 2002). In this way, both methods (bootstrap and Bayesian analyses) incorporate the idea that if the majority (but not all) of the optimal trees for a data set display a group, then the group should be considered as supported.

Since resampling requires extensive calculations, a number of approximations have been proposed. The most common simplification—used by RAxML (Stamatakis et al. 2005), GARLI (Zwickl 2006), and PhyML (Guindon and Gascuel 2003)—is saving a single tree during the heuristic search for each pseudoreplicate matrix, even when the (resampled) data would support several equally likely alternatives. The simplification of saving a single tree per pseudoreplicate could be justified on the grounds that it produces the same results (Goloboff and Pol 2005; Simmons and Goloboff 2013) as the frequency-within-replicates approach implemented in PAUP* and Phylip, if the tree search has the same chances of finding any of the equally optimal trees—that is, *if the tree search is unbiased*. However, the absence of bias is an important requirement for this equivalence. If a biased tree search on data with multiple equally optimal trees finds some of the trees with higher probability, then the results of resampling will not be solely determined by the data. Rather, they will also be determined by the bias in the search, and thus artifactual.

In the case of ties in the optimality criterion, using stepwise Wagner trees (Farris 1970; also known as "greedy" trees) with a consistent addition order may produce high frequencies for some unsupported groups (Backeljau et al. 1996; Farris et al. 1996). Given that using the same taxon ordering to build the trees can produce a systematic bias favoring some groups, it has been assumed that the opposite would be true: that randomizing the taxon ordering would eliminate the bias favoring some groups. Although few authors have discussed this idea explicitly (an exception being Felsenstein [2004, p. 168–169]), the notion that randomizing the addition sequences or using random trees as starting points for branch swapping eliminates bias is so intuitively appealing that it has

never been questioned (and is often assumed more or less explicitly; e.g., Goloboff and Farris [2001, p. S27]; Simmons and Goloboff [2013, p. 272]). The absence of bias is also implicitly required for a bootstrap with a single tree saved per pseudoreplicate to produce meaningful values.

This article focuses on the most widely used methods for tree searches—random-addition sequence trees and branch swapping—demonstrating that these methods are significantly biased, not only in the usual computer implementations, but even when all of the possibly deterministic elements of the algorithms are eliminated (i.e., when every addition and insertion sequence is equiprobable, and when branch swapping can equiprobably start from any tree and equiprobably select any of the trees of better score in the subtree-pruning-regrafting [SPR]- or tree-bisection-reconnection [TBR]-neighborhoods). This unexpected finding further calls into question the approach of measuring group support while saving a single tree per pseudoreplicate, and suggests that, if appropriately implemented, measures based on comparing tree scores (such as the likelihood ratio test [Felsenstein 1988] or the Bremer support [Bremer 1994]) may be preferable.

Materials and Methods

The following programs were tested: PhyML version 4 December 2012, RAxML version 7.7.2 (compiled for Windows with CygWin), GARLI version 2.0.1019, PAUP* version 4.0b10, and TNT (Goloboff et al. 2008) version September 2013. For simplicity of treatment, zero-length branches were retained by all programs for all tests. Note that tree searches themselves treat trees as fully binary in most programs, only eliminating zero-length branches at the end, so that any preference for some binary trees will continue affecting the results when zero-length branches are collapsed. In all of the model-based programs but RAxML, a Jukes–Cantor (1969) model (JC69) with a single rate of change was used. In RAxML, the GTRGAMMA model was used (while this may affect the specific values of bias, it does not affect any of our main conclusions). The programs for exact calculations of bias were written in C and compiled with the OpenWatcom compiler; code and binaries can be found at http://www.lillo.org.ar/phylogeny/published/Search_bias.zip, and as Supplementary Material in the Dryad data repository at http://dx.doi.org/10.5061/dryad.tm80k. This article is confined to small cases where the expected values can be calculated exactly, because demonstrating the existence of bias in searches requires certainty that all the optimal trees are found (possible only for small data sets via branch-and-bound or exhaustive searches). Larger cases cannot be analyzed in this manner: there is no way to know whether an unexpectedly high frequency for a group absent from the consensus of optimal trees is caused by the group being indeed present in the majority of optimal trees, or caused by bias in the tree search.

Tree Searches in Actual Programs

If heuristic searches are unbiased, the frequency for each group found by branch-and-bound must match almost exactly the frequency of the group when a heuristic search saving a single tree is repeated with different random seeds. This is not observed in either of the computer programs examined here that are capable of exact searches (TNT and PAUP*); both programs produce very strong deviations for many data sets. The results obtained with the programs that always use heuristic searches to save a single tree per starting point (RAxML, GARLI, and PhyML) resemble those obtained when saving a single tree with TNT or PAUP*.

The bias observed in actual programs does not prove that search algorithms themselves are biased, since there is a number of ways in which a program may fall short of ideal randomization. The first and most obvious way is that any computer program uses pseudorandom numbers (truly random numbers are difficult to generate), but a program may introduce determinism in other ways. In the case of Wagner trees, even when the taxa are added to the growing tree with a randomized sequence, positions for a given taxon may be tied. Most programs simply select the first best position, but which node is tried first depends on the order in which the nodes are visited—that is, on the *insertion* sequence (Simmons and Goloboff 2013, p. 267). The easiest approach for the insertion sequence is using a preorder or postorder traversal of the tree (i.e., an up- or down-pass; e.g., as done by RAxML, and the default option in TNT). But that approach might conceivably introduce some determinism. Avoiding this determinism would require making a list of the tree nodes, randomizing it, and then trying to insert each taxon following this randomized list; this is a Wagner tree where both the insertion and addition sequences are randomized. The same result is obtained by creating a list of all positions tied for best, for each taxon added to the tree, and then randomly choosing one of the positions in the list (the former method has the advantage, from the computational point of view, that the score calculation for a given position can be abandoned as soon as the best score is tied, but both produce the same final result).

In the case of branch swapping, several implementation details may add determinism. The most obvious ones are the sequence in which clades to be moved are clipped from the tree and the sequence of reinsertion positions in the remaining subtree. Actual programs may introduce a number of deviations from randomness in these aspects, for the sake of speeding up calculations.

Exact Calculations of Tree-Search Bias

For small data sets, instead of relying on existing computer implementations, the probability of each possible end result can be calculated analytically, both for random-addition sequences and branch swapping. This

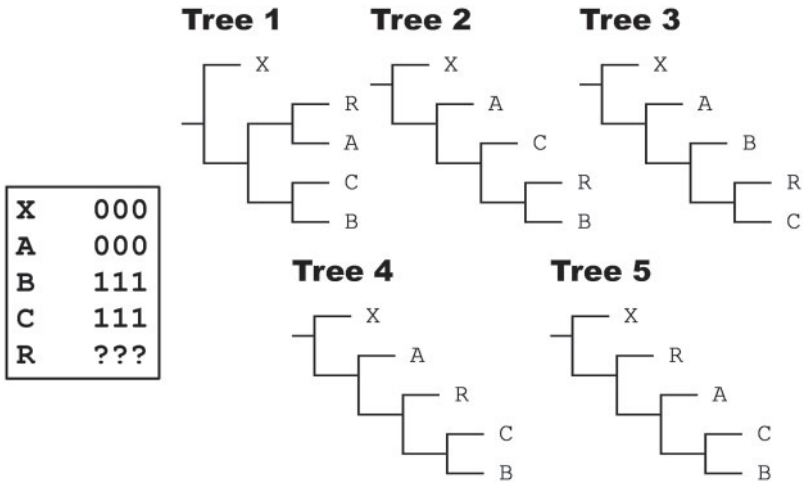

FIGURE 1. Example to illustrate exact calculations of tree-search bias. The data set produces five optimal trees, under both parsimony and maximum likelihood. A search is unbiased when it has the same probability of finding each of the optimal trees (0.200 in this case). Both Wagner trees and branch-swapping starting from random trees are more likely to find some of the five optimal trees for this data set; tree 4 is the least frequently found tree in both cases.

can be done by enumerating all possible outcomes of each stage of a random-addition sequence Wagner tree, or branch swapping from random starting trees.

Figure 1 shows one of the simplest examples where a bias is evident. X is the taxon used to root the tree. We will first consider parsimony as the optimality criterion. Any tree in which B and C are closer to each other than they are to A is most parsimonious; taxon R has a missing entry for all three characters in the data set, and thus could be placed at five alternative positions (trees 1–5 in Fig. 1). For a tree search to be unbiased, each of those five trees must be found equiprobably, with probability 0.200.

Stepwise Addition

"As is".—When the addition sequence is as given in the data set, then a Wagner tree produces exactly the expected result—if the insertion sequence is randomized. Recall that a randomized insertion sequence retaining the first best position found is equivalent to creating for each taxon added to the tree a list of all the best positions, and then choosing one randomly. For the "as is" addition sequence, of the three possible insertion points for taxon C in the subtree X(AB), only the one that produces X(A(BC)) is best. Taxon R can then be added at five possible branches, each of which has the same score. If these five possible insertion points are tried in a random sequence, then each has a probability of 0.200 to be the first. Evidently, only a randomized insertion sequence will produce the expected result of 0.200 for each tree with the "as is" addition sequence. An upward insertion sequence will produce only tree 5, and a downward insertion sequence (assuming the possible preorder traversals are equiprobable) will produce only trees 1, 2, or 3, with probability 0.333 each.

Other addition sequences.—Although an "as is" addition with a random insertion sequence produces unbiased results, that is not the case for all addition sequences. Consider the results of using a sequence where XAB are joined first, followed by R and then C. Terminal R can be added at three locations in the subtree X(AB), producing subtrees X(B(AR)), X(A(BR)), and X(R(AB)) with probability 0.333 each. On the first subtree, the last terminal, C, can be added at only one position, as sister of B, producing tree 1 as the final tree. On the second subtree, C can be added at three positions, producing trees 2, 3, or 4 as the final trees. On the third subtree, C can be added at only one position, as sister of B, producing tree 5 as the final tree. Thus, given the addition sequence of XAB followed by R and then C, the probability of obtaining tree 1 or tree 5 is 0.333 (a probability of 1/3 that the first subtree is chosen, and then a single choice leading to tree 1; likewise for the third subtree, leading to tree 5), and the probability of obtaining trees 2, 3, or 4 is 0.111 (a probability 0.333 that the second subtree is chosen, and then three possible choices leading to trees 2, 3, or 4). In other words, for this addition sequence, a completely randomized insertion sequence will produce some of the five trees with higher probability than others.

Multiple random-addition sequences.—When all possible addition sequences are equiprobable, the results are also biased. If X (the root taxon) is always one of the first three terminals placed in the initial subtree, then 12 distinct addition sequences for the remaining four taxa exist (4 taxa have 24 orderings, but only 12 are relevant because the first 2 taxa after the root can be placed in any sequence: XAB is the same as XBA). When all 12 distinct sequences are considered, the probability of selecting tree 1 through the sequence XAB followed by R and then C (discussed in the previous section) is 1/12

$\times$ 1/3 = 0.028, and the probability of selecting tree 2 through that sequence is 1/12 $\times$ 1/3 $\times$ 1/3 = 0.009. The probability of selecting tree 1 or tree 2 through the "as is" addition sequence is 1/12 $\times$ 1/5 = 0.017 in each case. Summing the probabilities of selecting each of the possible trees through each of the possible addition sequences, we obtain different probabilities for the different trees: $P_{(\text{tree 1})} = P_{(\text{tree 5})} = 0.244$, $P_{(\text{tree 2})} = P_{(\text{tree 3})} = 0.189$, and $P_{(\text{tree 4})} = 0.133$.

Cause of the bias.—The only way in which the different tree probabilities could be equal when all possible addition sequences are considered would be if the differences in probability for the five different trees exactly cancel out. Such an expectation of symmetry (however aesthetically pleasing) is simply not fulfilled, and the net result is a clear bias. The tree that places the floating taxon R between the two partitions (tree 4 in Fig. 1) is the least probable tree, and the trees that place R at the first split or as sister of A (trees 1 and 5) are much more probable than if one of the most parsimonious trees was selected at random.

The difference in probabilities also affects the probability of finding a tree with a given group, and the probability of finding the group can be either increased or decreased. Consider the groups BC and RBC: each is present in three of the five possible trees for the data set in Figure 1, so that their expected frequency in an unbiased search is 0.600. But the probability of a Wagner tree having group BC sums up to 0.622, so that the group will appear *more* frequently than expected; the probability of the tree having RBC sums to 0.511, so that the group will appear *less* frequently than expected.

Implementation.—The example of Figure 1 is a small data set, such that the calculations can be performed manually. For cases with character conflict or more taxa, this approach becomes intractable. For these cases, the calculations were implemented in a small program, *wagbias*. This program allows calculation of exact probabilities of both trees and specific groups in Wagner trees (optionally using a randomized, an upward, or a downward insertion sequence), for data sets with up to 10 taxa.

Variations in the stepwise tree.—The results reported above for Figure 1 always included taxon X as the root in the initial three-taxon subtree. Randomizing root taxon X as well (i.e., considering also the addition sequences where the first three-taxon subtree does not include X), the net result continues being biased. Likewise, using deterministic insertion sequences (either upward or downward) produces greater deviations from the expected frequencies.

Branch Swapping (Parsimony)

The previous sections showed that trees created by stepwise taxon addition are irremediably biased, but using fully random trees as starting points for randomized branch swapping still may produce unbiased results. Randomizing the swapping would involve cutting the clades to swap in a random order, and then reinserting (and rerooting, in the case of TBR) at randomly chosen points. As an alternative description of the same process, one may consider that the neighborhood or vicinity of each tree $\boldsymbol{T}$ includes a series of $\boldsymbol{n}$ trees $\{\boldsymbol{V_{T,1}}, \boldsymbol{V_{T,2}} \ldots \boldsymbol{V_{T,n}}\}$ of a better score than $\boldsymbol{T}$. The set of neighbor trees when $\boldsymbol{T}$ is an optimal tree is an empty set. SPR and TBR define different neighborhoods. A fully randomized swapper would pick any one of the $\boldsymbol{n}$ trees in $\{\boldsymbol{V_{T,1}}, \boldsymbol{V_{T,2}} \ldots \boldsymbol{V_{T,n}}\}$, with uniform probability $\boldsymbol{1/n}$. Each of the $\boldsymbol{Q}$ possible (binary) trees would be a starting point, with probability $\boldsymbol{1/Q}$.

An example worked out under SPR.—Consider the case of Figure 1: tree 4 is found with a probability of 0.066 (1/15, as there are 15 possible trees for that data set) by picking a random starting tree, but tree 4 can also be found by using some of the 10 suboptimal trees as starting points for swapping. Figure 2 shows all of the possibilities for SPR branch swapping. For example, given the suboptimal tree X((RC)(AB)) (i.e., the right pointing horizontal arrow), SPR swapping will lead to tree 4 with probability 1/4, since four SPR neighbors of tree X((RC)(AB)) have fewer steps. Given that there is a 1/15 probability that tree X((RC)(AB)) itself is the starting point, then the probability of arriving at tree 4 through X((RC)(AB)) is 1/15 $\times$ 1/4 = 0.167. The sum of probabilities of selecting (via SPR) tree 4 through each of the possible trees totals $P_{(\text{tree 4})} = 0.178$. Repeating similar calculations for the other trees, $P_{(\text{tree 1})} = 0.255$ and $P_{(\text{tree 2})} = P_{(\text{tree 3})} = P_{(\text{tree 5})} = 0.189$. Again, the tree in which R is in the middle of the partition (tree 4 in Fig. 1) is the least likely tree and $P_{(\text{group AB})} = 0.622$ (greater than expected) and $P_{(\text{group RBC})} = 0.555$ (smaller than expected). As in the case of Wagner trees, the expectation of symmetry when all possible starting points are considered is unfounded. For example, tree 4 can be obtained by SPR swapping from 6 of the 10 suboptimal trees, while tree 1 can be obtained by swapping from eight of those (and for two of those eight, with conditional probability 0.5, as only two trees in their SPR neighborhood improve the score). Those differences make the sum of probabilities for the possible ways to obtain tree 1 greater than the sum of probabilities to obtain tree 4.

Implementation.—The approach illustrated in Figure 2 has been implemented in a small computer program, *tbrias*. The program can handle up to 10 taxa; it operates by generating all possible trees and calculating their parsimony scores, then finding all of the TBR (or SPR or NNI) neighborhoods for each tree, finally performing the probability calculations. For faster calculation of the probabilities of arriving at each final tree, the program takes advantage of the fact that the neighborhoods are ordered (including only trees of better score), so that the

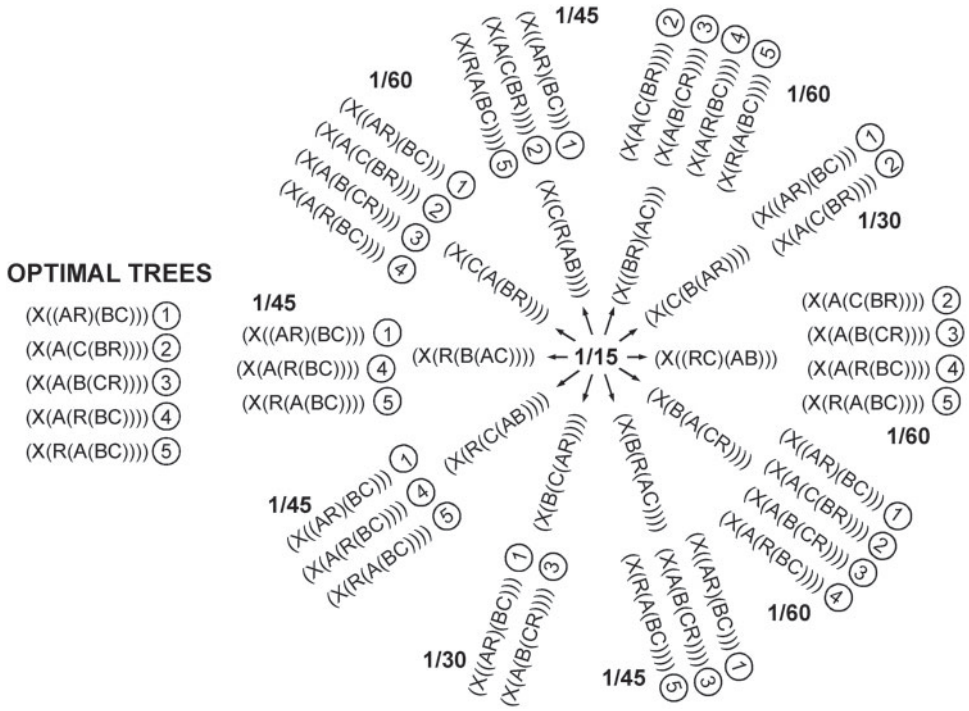

FIGURE 2. Calculation of the exact probabilities of each of the five trees in Figure 1 under SPR starting from random trees. The optimal trees (1–5, numbered as in Fig. 1) can be found either by direct generation (with probability 1/15, since there are 15 possible trees for this data set), or they can be found by swapping from 1 of the 10 suboptimal trees (each of which is the starting point with $P = 1/15$). Some of the optimal trees (e.g., tree 1) can be found by swapping from eight suboptimal trees, whereas others (e.g., tree 4) can be found only from six suboptimal trees. Therefore, the probability of finding tree 1 is higher than the probability of finding tree 4.

partial probabilities can be calculated first for the longest trees, then for successively shorter trees. Note that the SPR implementation in *tbrias* never moves the root taxon (which it could do by moving the root taxon to other parts of the tree and then rerooting—those moves are considered in *tbrias* as only part of the repertoire of TBR moves).

TBR swapping.—Unlike SPR, TBR is completely unbiased for the example of Figure 2—all five trees have exactly the same probability of $P = 0.200$. However, a matrix with a single taxon floating in a subtree of more than four taxa (as well as other examples shown in the next section) does have a significant bias in TBR. TBR has a slight (but not universal) tendency to be less biased than SPR. Although in the case of SPR swapping the root taxon might make a difference, there cannot be any such hope in the case of TBR—the moves performed by TBR include all rootings of the clipped clade, so that the taxon chosen as root is not relevant for the calculation of probabilities under TBR.

Variations in swapping algorithm.—Alternative ways to perform branch swapping are biased as well. NNI, known as a superficial and ineffective swapping algorithm, is not only more strongly biased than SPR and TBR; it can also fail to find the optimal tree even in the simplest cases (e.g., for Fig. 1, NNI from a random tree fails to find the optimal tree with $P = 0.133$).

The approach described in the previous sections avoids redundant swaps (i.e., alternative swaps that result in the same topology)—given that some better trees exist in the neighborhood of a tree, each of those is chosen equiprobably. However, one could think of the algorithms as defined in terms of moves instead of neighborhoods, in which case some of the moves are redundant (implying that some of the transitions between trees may be more likely than others). When redundant moves are allowed, the bias in branch swapping is generally intensified.

An alternative form of swapping (to our knowledge implemented only in PAUP* and POY version 4; Varón et al. 2010) is the "steepest descent" option. Given that steepest descent is more time-consuming than the standard algorithms, it is rarely used in phylogenetic analyses. The exact calculation of possible outcomes of steepest descent SPR and TBR (done by simply selecting all of the best trees in $\{\boldsymbol{V_{T,1}}, \boldsymbol{V_{T,2}} \ldots \boldsymbol{V_{T,n}}\}$ and then picking one of those at random) shows that in all examples given in this article swapping remains biased. A closer examination of Figure 2 shows why: each of the suboptimal trees may lead to an optimal tree by just one SPR rearrangement—that is, without passing through trees of intermediate length. Thus, steepest descent makes no difference for the data set of Figure 1.

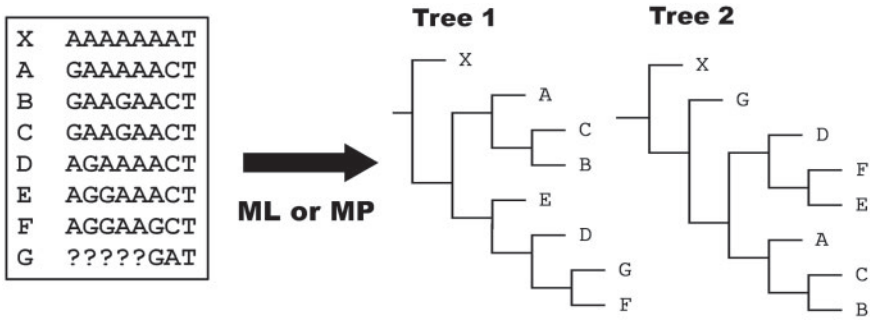

FIGURE 3. A case where maximum likelihood and parsimony produce the same two optimal trees, but because of differences in the ordering of suboptimal trees, the two criteria have a SPR or TBR bias preferring a different optimal tree. Tree 1 and group FG are found with $P = 0.475$ under parsimony TBR, but with $P = 0.545$ under maximum likelihood. The probability of tree 2 and group EF is complementary ($P = 0.524$ under parsimony and $P = 0.454$ under likelihood).

We have demonstrated that all possible versions of NNI, SPR, and TBR are biased: even when swapping starts from fully random trees the final results are more likely to contain some of the optimal trees than others. This bias is introduced by intrinsic properties of the swapping algorithms, and cannot be justified as a property of the data.

Branch Swapping (Maximum Likelihood)

Parsimony trees are usually considered as acceptable starting points for subsequent branch swapping under maximum likelihood (as done in, e.g., RAxML and PhyML). These starting points may introduce a bias in the subsequent likelihood searches. An alternative would be to use random trees as starting points for swapping under likelihood. The same type of calculations as in the preceding section can be done for maximum likelihood. This has been implemented in another small program, *likbias*, which calls PAUP* as a daughter process to generate all trees and calculate their likelihoods. Then, *likbias* reads the trees and their likelihoods from PAUP* output files, and proceeds to make calculations in exactly the same way as illustrated in Figure 2. Given that calculation of the likelihood of a tree is not exact (see Swofford et al. [1996] for the basic principles), the values obtained with *likbias* are thus "exact" only to the extent that the likelihood scores calculated by PAUP* are accepted as correct. To decrease the effect of this imprecision, *likbias* treats trees differing by 10^{-5} or less units of log likelihood as having "the same" likelihood (in small data sets, differences in likelihood scores for different trees are normally much larger than this).

For the example of Figure 1, maximum likelihood will produce exactly the same results as parsimony: the five optimal parsimony trees have the same likelihood, and the probabilities of obtaining each of the possible trees or groups, by swapping from fully random trees, are exactly the same as before. But the bias in parsimony and likelihood can be different for some data sets, even in cases when the two criteria produce the same optimal trees. This difference is because the bias depends on how the suboptimal trees are arranged and interconnected through successively better trees, and this ordering may differ between the two optimality criteria. Figure 3 shows an example, where G can float between two positions (producing only two possible optimal trees, both under parsimony or likelihood). In this case, an unbiased search should display the groups EF and FG with a frequency of exactly 0.500, but the bias toward one of the optimal trees causes some group (EF for parsimony and FG for likelihood) to appear with a higher frequency. Tree 1 and group FG are found with $P = 0.475$ under parsimony TBR, but with $P = 0.545$ under likelihood TBR. The probabilities for tree 2 and group EF are complementary, $P = 0.524$ under parsimony TBR, and $P = 0.454$ under likelihood TBR. This is one of the situations where tree-search bias typically occurs: when a terminal can float between positions at different levels of nesting (i.e., when the number of tree nodes between the floating terminal and the root differs).

EFFECT OF SEARCH BIAS ON BOOTSTRAPPING

When each bootstrap pseudoreplicate saves a single tree, the bias in tree searches may increase the frequency of some groups, to the extent of making otherwise unsupported groups seem supported. This is not a problem with bootstrapping per se, or with the optimality criterion used; the increase in frequency of some groups is simply a distortion caused by the bias in tree searches. While unbiased searches using Wagner trees and/or branch swapping seem impossible without saving multiple trees, actual implementations may (as already discussed) add a number of deterministic factors that increase the bias. This section illustrates the effect of this bias on actual programs.

Consider the case of Figure 4, a data set with 40% missing entries, non-randomly distributed in three blocks of data, which is typical of supermatrices formed by concatenation of genes. The subtrees for each of the blocks are compatible, but since those subtrees are incomplete they can be combined in different ways, and multiple equally optimal trees belong to the same "terrace" of trees (Sanderson et al. 2011). As noted

X	AAGCTAGCTAG	AAGCTAGCTAG	AGCTA
A	AAGCTAGCTAG	??????????	AGCTA
B	GAGCTAGCTAG	??????????	AGCTA
C	GCTAGCGCTAG	??????????	AGCTA
D	GCTAGCGCTAG	??????????	?????
I	??????????	AAGCTAGCTAG	AGCTA
J	??????????	GAGCTAGCTAG	AGCTA
K	??????????	GCTAGCGCTAG	AGCTA
L	??????????	GCTAGCGCTAG	?????

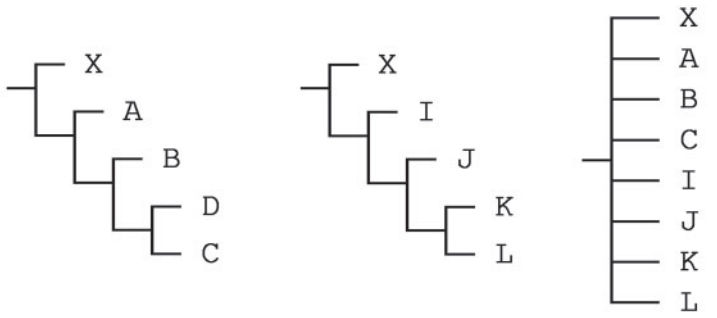

FIGURE 4. A data set for which the branch-swapping bias will lead to the conclusion, under maximum likelihood, that some groups are moderately well supported (see Table 2), even if their frequency in an unbiased search would be well below 0.5. The three trees shown below the data set correspond to the (consensus of) optimal tree(s) for each of the subsets of data.

by (Sanderson et al., 2011), tree terraces (or some approximation of them, when non-partitioned models are used) are to be expected often when combining data for different genes with uneven taxon representation.

For the data set of Figure 4, exact calculation with *likbias* and PAUP* (under JC69) produces 135 trees of the same likelihood (when differences of 10^{-5} or less in the log likelihood are ignored). The groups CD and KL occur in 0.200 of those trees, but a completely randomized TBR search starting from a random tree finds each of those groups with probability $P = 0.422$ (over two times more often than expected). Of course, an actual heuristic search with PAUP* (stepwise tree plus TBR) is not ideally randomized; repeating such a search many times for the original data set (with different random seeds), those groups are found with more strongly deviating frequencies ($P_{CD} = 0.720$ and $P_{KL} = 0.731$). That bias affects bootstrapping, artificially increasing the frequency for these groups; the correct bootstrap support when using frequency within pseudoreplicates can be calculated with 1000 bootstraps and a branch-and-bound search (with PAUP*) saving all optimal trees. This finds groups CD and KL with $P_{CD} = 0.390$ and $P_{KL} = 0.311$ (the reason for the difference between otherwise similar groups is unclear). However, when the search uses a random-addition sequence stepwise tree followed by branch swapping without saving multiple trees, bootstrapping produces $P_{CD} = 0.690$ and $P_{KL} = 0.670$, strongly deviating from the values obtained with the exact search, and producing the impression that unsupported groups CD and KL have moderate support. If the search uses a random starting tree (with the "randomize = tree" option of PAUP*), the apparent support is somewhat lower, $P_{CD} = 0.524$ and $P_{KL} = 0.539$, but still much higher than the correct value (0.390 and 0.311, respectively), and above the usual cutoff to ignore groups, 0.500.

The results produced by other maximum-likelihood programs also deviate strongly from the correct values for groups CD and KL. These values are shown in Table 1. Of course, while extreme deviations (such as those in RAxML and GARLI) could perhaps be corrected by using more randomized searches, the deviation from the correct values of support is unavoidable when a single tree per pseudoreplicate is saved. The least biased program for this data set is PhyML, but the low frequency for group CD probably means that the program is biased *against* the group (and that in some other cases group frequencies will be artificially high). Both RAxML and GARLI produce very high frequencies (0.905–0.989) for the groups that, if using a branch-and-bound analysis, should occur with frequency 0.390 or less.

BIAS FROM FAILURE TO FIND OPTIMAL TREES

In the examples of Figures 1–4, a Wagner tree or TBR search never fails to find an optimal tree; the tree-search bias resulted from the search algorithm preferring some of the optimal trees over others. This section shows that,

TABLE 1. Frequencies of different groups under bootstrapping in different maximum-likelihood implementations for the data set of Figure 4

	Expected (PAUP*, exact search)	PhyML	RAxML	PAUP* (stepwise and TBR)	GARLI Stepwise	GARLI Random
CD	0.390	0.315	0.905	0.690	0.977	0.970
KL	0.311	0.325	0.989	0.670	0.950	0.987

Notes: The options used to run each program are: PhyML, parsimony trees for starts, 1000 bootstraps, SPR swapping; RAxML, 1000 rapid bootstraps, GTRGAMMA; PAUP*, 1000 bootstraps, starting from stepwise trees, TBR swapping, nomulpars; GARLI, stepwise ML and random starting trees, 300 bootstraps, genthreshfortopoterm 10000. The expected probability is derived from 1000 bootstraps with a branch-and-bound search saving all optimal trees, with PAUP*. See text for further discussion.

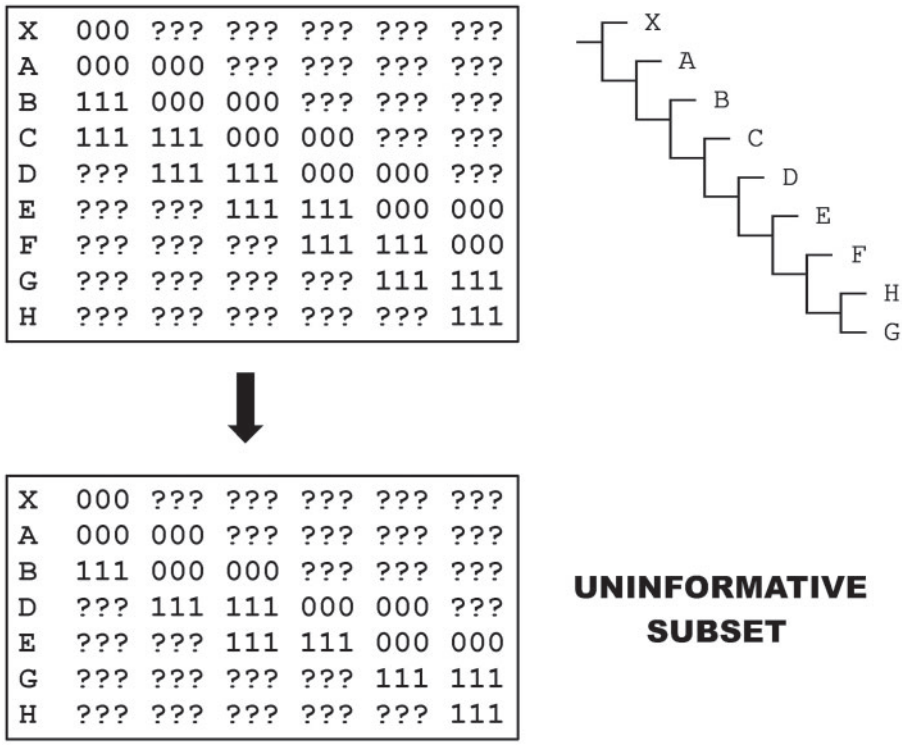

FIGURE 5. A data set for which parsimony search algorithms have a low probability of finding the single most parsimonious tree. Even if each of the quartets of taxa is supported by the same number of characters, a search is more likely to find a tree with some of the quartets (e.g., EF | GH) than with others (e.g., CD | EF). See text for additional discussion and Table 2 for the actual values.

even when a single optimal tree is supported by the data, some of the equally supported groups may be retrieved more frequently than others because of the failure of searches to find the optimal trees.

Consider the case of Figure 5, with six sets of characters that define mutually compatible subtrees that jointly determine a single pectinate most parsimonious tree. When forming the Wagner tree, most of the taxon subsets will be uninformative. Consider the subset shown below the original data set, which would be the subset of taxa corresponding to forming a Wagner tree by first adding XABDEG (in any sequence) and then inserting H at the best available location of the best subtree obtained by the sequential addition of the previously placed taxa. For those seven taxa (only C and F are missing from the tree), all six sets of characters are uninformative, requiring the same number of steps on any possible tree. Of course, other addition sequences are more favorable (such as "as is," which produces the optimal tree as the only possible outcome), but when all possible addition sequences are considered, only a small fraction actually produces optimal trees, and the actual probability of finding an optimal tree with a random-addition sequence Wagner tree for this data set is 0.005. A Wagner tree fails to detect the structure in this data set to such a degree that there are 36,140 suboptimal trees (of lengths 21–27) as possible outcomes of random-addition sequence Wagner trees.

The data set of Figure 5 also poses problems to branch-swapping searches that save a single tree, because the parsimony landscape for this data set is very flat—many suboptimal trees can be connected to the optimal tree only through rearrangements producing trees of the same length, and are thus never accepted when only saving a single tree.

In the example of Figure 5 three characters support each quartet (i.e., each four-taxon subtree), so that each quartet is supported equally. The failure to find optimal trees comes from the difficulty in assembling

TABLE 2. Exact calculations of the probabilities that a given tree search finds different quartets and groups of the data set in Figure 5, under parsimony, using the programs *wagbias* and *tbrias*

	Wagner	SPR	TBR
XA | BC	0.829	0.857	1.000
AB | CD	0.533	0.644	0.704
BC | DE	0.723	0.767	0.731
CD | EF	0.705	0.743	0.731
DE | FG	0.598	0.685	0.704
EF | GH	0.865	1.000	1.000
GH	0.294	0.346	0.342
FGH	0.111	0.167	0.174
EFGH	0.055	0.085	0.100
DEFGH	0.042	0.066	0.100
CDEFGH	0.059	0.088	0.174
BCDEFGH	0.199	0.218	0.342

Note: The expected frequency for all quartets and groups is 1.0 because only a single tree is optimal for this data set (see Fig. 5).

the individual subtrees into a complete tree—much as in supertrees—and this will be the case regardless of how strongly supported each quartet is.

Even when all quartets are supported by the same amount of evidence, different groups and quartets in the most parsimonious tree are recovered with a different probability by the searches (see Table 2). For example, the partition AB | CD is found by stepwise addition with probability 0.533, but the probability of finding EF | GH is much larger, 0.865. The same applies to different groups in the optimal tree. This difference is introduced solely by the bias in heuristic tree searches, and when only a single tree per pseudoreplicate is found, it affects bootstrap values because some groups or quartets appear more often than others.

DISCUSSION

The analysis presented here is based on small artificial data sets, since the expected result can be analytically identified only in those cases. For larger empirical data sets, the impossibility of identifying problems like those presented in this article does not mean that they do not exist. The bias inherent to tree searches, which makes finding some optimal trees more likely than others, or finding some groups more frequently than they occur among all optimal trees, has two important implications for phylogenetic analyses.

The first implication is that the trees produced by a search saving only one or a few trees cannot be considered, in any sense, to represent an unbiased sample of multiple equally optimal trees for a given data set. The easiest way to obtain a truly unbiased sample of the optimal trees for a data set seems to be in two steps: first find all optimal trees (either with an exact solution, or with reliable heuristic searches), and then select n trees from those at random. Any attempt to obtain a representative sample of trees by means of branch swapping without saving all possible trees will fail. This probably includes approaches such as the *–fitchtrees* option of POY version 3 (Wheeler et al. 2005), which implements an original idea of Walter Fitch (as a personal communication to W. Wheeler, hence the name of the option; this option is a type of reservoir sampling [Vitter 1986]). The *–fitchtrees* option continuously swaps trees accepting rearrangements of equal score, and stores them using a probability that depends on the size of the sample to be stored. In some cases, this option will provide an unbiased sample of the trees—for example, swapping from any of the five optimal trees for the data set of Figure 2 any one of the other four trees can be found equiprobably if redundant moves are avoided. In more complex cases, however, this option cannot be guaranteed to represent an unbiased sample of the possible optimal trees (even if it still can be useful to gather a number of distinct trees that represent different hypotheses of evolution, with no special statistical meaning). Likewise, exact solutions are not randomized in any way; since they generate possible trees in an ordered sequence, finding all K trees and then selecting n is different from performing an exact search saving up to n trees. The latter will simply save the first n trees it finds, producing even more biased results than random-addition sequences or branch swapping. An alternative (recently implemented in TNT, as the "&" option of the "ienum" command) is to use reservoir sampling during the exact search. In this way, to produce a random sample of n trees, as tree number k is found after having already stored n trees, the probability of storing it is n/k. If the tree is stored, each of the n previously stored trees has the same probability of being replaced. Under such a scheme each of the K trees found by the exact search has a probability n/K of being in the sample, even when we do not know in advance how large K is (Vitter 1986). This allows having a random sample from a large population of trees, without the memory requirements needed to store them all; the results of this method depend quite strongly, however, on the quality of the pseudorandom number generator used.

The most significant consequence of the bias in tree searches, however, is for measuring group supports with a single tree saved per pseudoreplicate. Farris et al. (1996) showed that using the same input order for the initial trees often leads to unsupported groups displaying a high frequency under resampling. They proposed using a different randomization of the taxon sequence for each pseudoreplicate, and doing so has been assumed to eliminate bias in stepwise algorithms. However, this article demonstrates that randomization does not eliminate bias in tree searches, and that this bias may determine the frequencies obtained under bootstrapping more than the actual properties of the data. Although some computer programs are more biased than others, no method based on addition sequences and branch swapping can be free of bias. The bias may occur even in simple data sets and, in the case of data sets where the optimal tree is hard to find by standard search methods, there may be differences in the frequency with which groups or partitions (present on the optimal tree and supported by equal amounts of evidence) are recovered

by a search. The latter situation implies that the bias can be detrimental not only for cases of ambiguity caused by absence of data, but also for ties in the optimality criterion caused by conflict between characters (exact ties become less likely in the resampled matrices in that case, but if the optimal tree is hard to find, group frequencies can be systematically distorted). All of this can happen in both parsimony and likelihood analyses—even for matrices with fully congruent characters.

For measuring group supports, a possible solution (which, if not eliminating the problem completely, will at least mitigate it) is saving multiple trees per pseudoreplicate. It has been shown (Goloboff et al. 2003; Goloboff and Pol 2005; Simmons and Freudenstein 2011; Simmons and Goloboff 2013) that saving a single tree per pseudoreplicate may produce (even in the absence of tree-search bias) illogical values of support because of the uneven frequency of different unsupported groups among optimal trees (the same point was made by Goloboff and Farris [2001] for their consensus-estimation methods). Such a problem is avoided by using the strict consensus for each pseudoreplicate (which also avoids Felsenstein's [2004] perceived problem of overemphasizing the results of those bootstrap pseudoreplicates that produce numerous trees). The discovery that bias is unavoidable in tree searches provides another forceful argument against estimating supports by saving a single fully resolved tree per pseudoreplicate, as done by GARLI, PhyML, and RAxML. If saving numerous trees proves computationally unfeasible, then methods to measure group support based on differences in likelihood or parsimony (e.g., Templeton 1983; Bremer 1994; Shimodaira and Hasegawa 1999; Anisimova and Gascuel 2006) may be more appropriate (Wheeler [2010] and Anisimova et al. [2011]).

Supplementary Material

Supplementary material (code and binary files for *wagbias*, *tbrias*, and *likbias*) can be found in the Dryad data repository at http://dx.doi.org/10.5061/dryad.tm80k.

Funding

P.A.G. thanks financial support from the Consejo Nacional de Investigaciones Científicas y Técnicas (PIP 0687), and a collaborative grant (Dimensions US-Biota-São Paulo 'Assembly and evolution of the Amazon biota and its environment: an integrated approach'), supported by the U.S. National Science Foundation, National Aeronautics and Space Administration, and the Fundacão de Amparo a Pesquisa do Estado de São Paulo to Joel Cracraft and Lucía Lohman (with participation of P.A.G.).

Acknowledgements

Critical comments from S. Arias, S. Catalano, C. Szumik, G. Wilson, W. Wheeler, and especially J. Felsenstein are deeply appreciated. None of them is responsible for any errors that remain, but they are thanked for some of the errors that don't. Editors Dan Faith and Frank Anderson also provided useful comments and assistance during the review process.

References

Anisimova M., Gascuel O. 2006. Approximate likelihood-ratio test for branches: a fast, accurate, and powerful alternative. Syst. Biol. 55:539–552.

Anisimova M., Gil M., Dufayard J.-F., Dessimoz C., Gascuel O. 2011. Survey of branch support methods demonstrates accuracy, power, and robustness of fast likelihood-based approximation schemes. Syst. Biol. 60:685–699.

Backeljau T., De Bruyn L., De Wolf H., Jordaens K., Van Dongen S., Winnepenninckx B. 1996. Multiple UPGMA and neighbor-joining trees and the performance of some computer packages. Mol. Biol. Evol. 12:309–313.

Bremer K. 1994. Branch support and tree stability. Cladistics 10:295–304.

Davis J.I., Simmons M.P., Stevenson D.W., Wendel J.F. 1998. Data decisiveness, data quality, and incongruence in phylogenetic analysis: an example from the monocotyledons using mitochondrial *atp*A sequences. Syst. Biol. 47:282–310.

De Laet J., Farris S., Goloboff P. 2004. Treatment of multiple trees in resampling analyses. Cladistics 20:590.

Farris J.S. 1970. Methods for computing Wagner trees. Syst. Zool. 19:83–92.

Farris J.S., Albert V.A., Källersö M., Lipscomb D., Kluge A.G. 1996. Parsimony jackknifing outperforms neighbor-joining. Cladistics 12:99–124.

Felsenstein J. 1985. Confidence limits on phylogenies: an approach using the bootstrap. Evolution 39:783–791.

Felsenstein J. 1988. Phylogenies from molecular sequences: inference and reliability. Annu. Rev. Genet. 22:521–565.

Felsenstein J. 2004. Inferring phylogenies. Sunderland (MA): Sinauer Associates.

Felsenstein J. 2005. Phylip (Phylogeny Inference Package), version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle. Available from URL: http://evolution.genetics.washington.edu/phylip.html.

Goloboff P.A., Farris J.S. 2001. Methods for quick consensus estimation. Cladistics 17:S26–S34.

Goloboff P.A., Pol D. 2005. Parsimony and Bayesian phylogenetics. In: Albert V.A., editor. Parsimony, phylogeny, and genomics. Oxford: Oxford University Press. p. 148–159.

Goloboff P.A., Farris J.S., Källersjö M., Oxelmann B., Ramírez M.J., Szumik C.A. 2003. Improvements to resampling measures of group support. Cladistics 19:324–332.

Goloboff P.A., Farris J.S., Nixon K.C. 2008. TNT, a free program for phylogenetic analysis. Cladistics 24:774–786.

Guindon S., Gascuel O. 2003. A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. 52:696–704.

Huelsenbeck J., Larget B., Miller R., Ronquist F. 2002. Potential applications and pitfalls of Bayesian inference of phylogeny. Syst. Biol. 51:673–688.

Jukes T.H., Cantor C.R. 1969. Evolution of protein molecules. In: Munro H.N., editor. Mammalian protein metabolism. Vol. 3. New York: Academic Press. p. 21–132.

Larget B., Simon D.L. 1999. Markov Chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. Mol. Biol. Evol. 16:750–759.

Sanderson M.J., McMahon M.M., Steel M. 2011. Terraces in phylogenetic tree space. Science 333:448–450.

Shimodaira H., Hasegawa M. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. Mol. Biol. Evol. 16:1114–1116.

Simmons M.P., Freudenstein J.V. 2011. Spurious 99% bootstrap and jackknife support for unsupported clades. Mol. Phylogenet. Evol. 61:177–191.

Simmons M.S., Goloboff P.A. 2013. An artifact caused by undersampling optimal trees in supermatrix analyses of locally sampled characters. Mol. Phylogenet. Evol. 69:265–275.
Stamatakis A., Ludwig T., Meier M. 2005. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics 21:456–463.
Swofford D.L. 1998. PAUP*: Phylogenetic analysis using parsimony (* and other methods). Version 4. Sunderland (MA): Sinauer Associates.
Swofford D.L., Olsen G.J., Waddell P.J., Hillis D.M. 1996. Phylogenetic inference. In: Hillis D.M., Moritz C., Mable B.K., editors. Molecular systematics. Sunderland (MA): Sinauer Associates. p. 407–514.
Templeton A.R. 1983. Phylogenetic inference from restriction endonuclease cleavage site maps with particular reference to the humans and apes. Evolution 37:221–244.
Varón A., Vinh L.S., Wheeler W.C. 2010. POY version 4: phylogenetic analysis using dynamic homologies. Cladistics 26:72–85.
Vitter J. 1986. Random sampling with a reservoir. ACM Trans. Math. Softw. 11:37–57.
Wheeler W.C. 2010. Distinctions between optimal and expected support. Cladistics 26:657–663.
Wheeler W., Gladstein D., De Laet J. 2005. POY version 3.0. Program and documentation. Available from: URL http://research.amnh.org/scicomp/projects/poy.php.
Yang Z. 2006. Computational molecular evolution. Oxford: Oxford University Press.
Yang Z., Rannala B. 1997. Bayesian phylogenetic inference using DNA sequences: a Markov Chain Monte Carlo method. Mol. Biol. Evol. 14:717–724.
Zwickl D.J. 2006. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion [Ph.D. dissertation]. Austin: The University of Texas. Available from: URL http://garli.googlecode.com.