Cladistics (1996) 12:99–124

PARSIMONY JACKKNIFING OUTPERFORMS NEIGHBOR-JOINING

James S. Farris[1], Victor A. Albert[2], Mari Källersjö[1], Diana Lipscomb[3] and Arnold G. Kluge[4]

[1]*Molekylärsystematiska laboratoriet, Naturhistoriska riksmuseet, Box 50007, S 104 05 Stockholm, Sweden,* [2]*New York Botanical Garden, Bronx, New York 10458-5126 U.S.A.,* [3]*Department of Biology, George Washington University, Washington, D.C. 20052, U.S.A.,* [4]*Division of Reptiles and Amphibians, Museum of Zoology, The University of Michigan, Ann Arbor, Michigan 48109-1079, U.S.A.*

Received for publication 12 May 1995; accepted 18 September 1995

Abstract — Because they are designed to produced just one tree, neighbor-joining programs can obscure ambiguities in data. Ambiguities can be uncovered by resampling, but existing neighbor-joining programs may give misleading bootstrap frequencies because they do not suppress zero-length branches and/or are sensitive to the order of terminals in the data. A new procedure, parsimony jackknifing, overcomes these problems while running hundreds of times faster than existing programs for neighbor-joining bootstrapping. For analysis of large matrices, parsimony jackknifing is hundreds of thousands of times faster than extensive branch-swapping, yet is better able to screen out poorly-supported groups.

Introduction

A large, ambiguous matrix may have many most parsimonious trees, and finding them all by branch-swapping may take considerable time. Stoneking et al. (1992: 386) advanced this as a reason to use neighbor-joining instead.

> "There is thus an inherent and, at this time, insoluble problem with applying parsimony analysis to these data . . .. By contrast . . .. The NJ [neighbor-joining] algorithm (Saitou and Nei, 1987) is computationally much more efficient than parsimony algorithms, so an NJ tree can be constructed even for a very large data set."

Neighbor-joining programs seem fast primarily because they are designed to produce just one tree; Saitou and Nei (1987: 406) introduced their method as yielding a "unique final tree". But in fact that method may admit more than one tree, so that, if used alone, a neighbor-joining program may conceal ambiguities in data. Ambiguities can be uncovered by using resampling methods, and this takes less effort than branch-swapping. But resampling can be employed with parsimony too, and far more efficiently than with neighbor-joining.

Neighbor-joining

A neighbor-joining program constructs just one tree for any one input matrix, but may achieve this by making arbitrary choices. Matrices Two and Three from Källersjö et al. (1992) will provide examples.

0748-3007/96/020099+26/$18.00/0

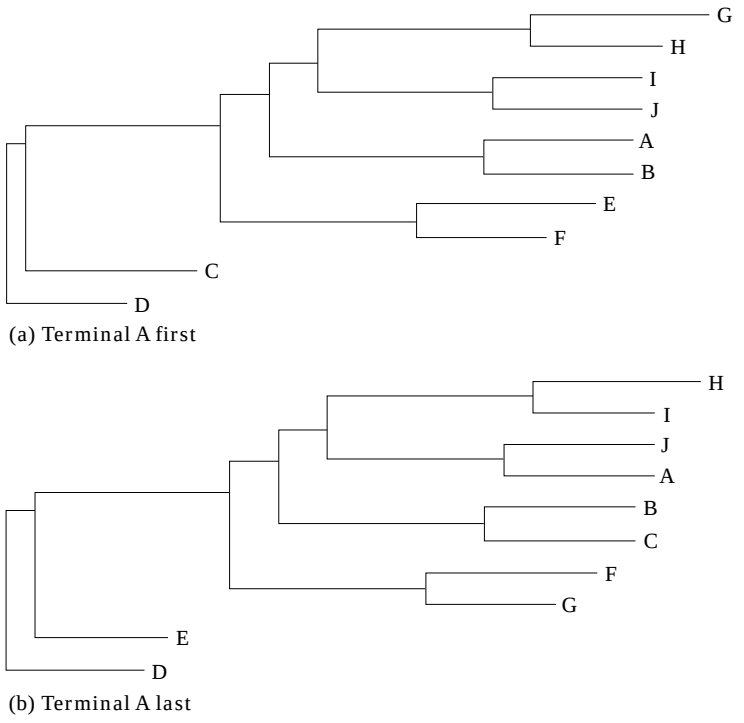


Fig. 1. MEGA neighbor-joining trees for two orderings of matrix Two.

	Two	Three
A	attttttta	aatttttttt
B	attttatttt	aaaatttttt
C	tatttatttt	aaaaaatttt
D	tattttattt	aaaaaaaatt
E	ttatttattt	aaaaaaaaaa
F	ttattttatt	ttaaaaaaaa
G	tttatttatt	ttttaaaaaa
H	tttattttat	ttttttaaaa
I	ttttatttat	tttttttaa
J	ttttatttta	ttttttttt

The trees in Figs 1 and 2 were obtained using Kumar et al.'s (1993) MEGA program. Terminal D was employed as an outgroup to simplify comparison with later figures. The Kimura two-parameter distance was used for matrix Two. For matrix Three, MEGA gave a "Failure in estimating distances" diagnostic with the Kimura distance, and we used number of sequence differences instead.

Analysed by the neighbor-joining procedure in MEGA, matrix Two gives the tree of Fig. 1a. But if terminal A is moved so as to become the last row of the matrix, the program gives the tree of Fig. 1b instead. The two trees have no informative groups in common.

For matrix Three, MEGA's neighbor-joining produces the tree of Fig. 2a. If the

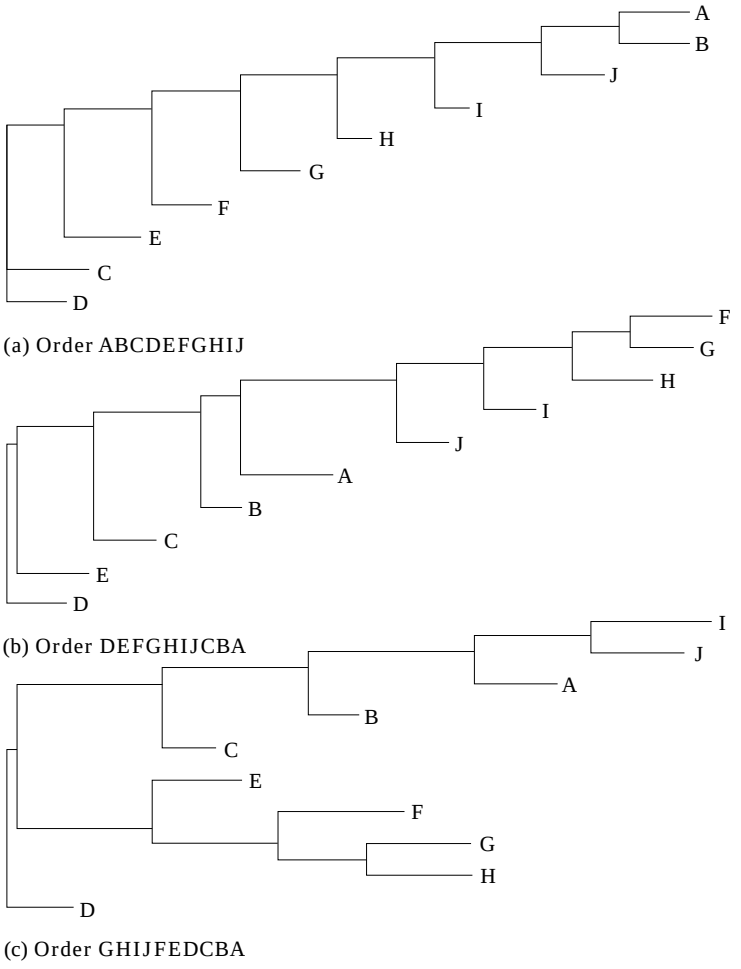


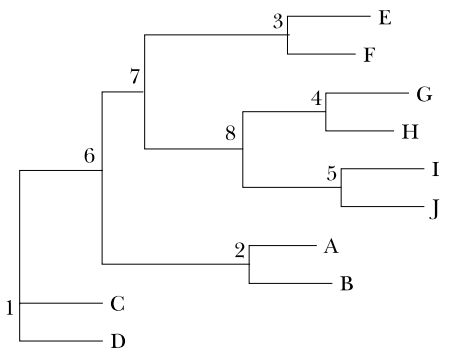
Fig. 2. MEGA neighbor-joining trees for three orderings of matrix Three.

terminals are rearranged into the order DEFGHIJCBA, the tree of Fig. 2b is obtained. Order GHIJFEDCBA gives the tree of Fig. 2c. There are still other possibilities, but the strict consensus of just these three is entirely unresolved.

The reason for such results is readily found from Saitou and Nei's (1987) description of their method. A group of two terminals is first selected to minimize the length of an otherwise-unresolved tree, the branch lengths being least-squares fits to the distance matrix. The two terminals are united and their columns in the distance matrix are pooled; they are treated as a single unit for purposes of further grouping. These steps are repeated until a complete tree has been formed.

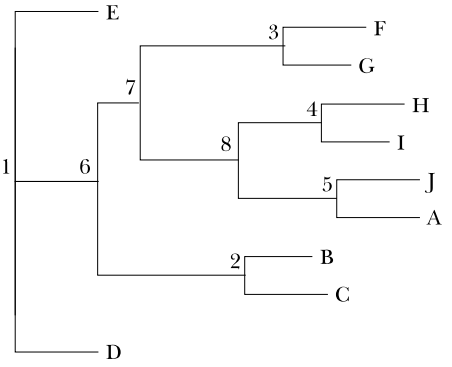
The criterion for choosing pairs is distinctive, but the algorithm is otherwise quite like much older clustering techniques such as UPGMA (reviewed by Farris, 1977). As in earlier methods, the sensitivity to the order of terminals in the input arises in selecting the next pair to be united. If any of several pairs might equally well be picked, the program simply uses the first such pair identified in its search.

(a) Terminal A first.



Between	And	Length
1	6	0.15694
6	7	0.03923
7	3	0.17002
3	E	0.13709
3	F	0.10221
7	8	0.03923
8	4	0.14386
4	G	0.14058
4	H	0.09873
8	5	0.17002
5	I	0.11965
5	J	0.11965
6	2	0.15694
2	A	0.10470
2	B	0.13460
1	C	0.11965
1	D	0.11965

(b) Terminal A last.



Between	And	Length
1	E	0.11965
1	6	0.15694
6	7	0.03923
7	3	0.17002
3	F	0.13709
3	G	0.10221
7	8	0.03923
8	4	0.14386
4	H	0.14058
4	I	0.09873
8	5	0.17002
5	J	0.11965
5	A	0.11965
6	2	0.15694
2	B	0.10470
2	C	0.13460
1	D	0.11965

Fig. 3. NEIGHBOR neighbor-joining trees for two orderings of matrix Two.

This does not apply only to MEGA; Felsenstein's (1993) NEIGHBOR program for neighbor-joining shows similar behavior. (For all PHYLIP runs, we used the extended DOS executables distributed by Felsenstein.) Using MEGA's Kimura distances for matrix Two, NEIGHBOR yields the tree of Fig. 3a. If terminal A is placed last, NEIGHBOR instead gives the tree of Fig. 3b, which shares no informative groups with the first.

The effects of search order are not limited to the input order of terminals. Available neighbor-joining programs vary in detail and may consequently produce different trees from the same input. Group GHIJ is placed with AB by MEGA in Fig. 1a, but with EF by NEIGHBOR in Fig. 3a. MEGA groups AJHI with BC in Fig. 1b, while NEIGHBOR puts AJHI with FG in Fig. 3b.

Rzhetsky and Nei's (1994) METREE program provides another illustration. Using Kimura distances for matrix Two, METREE's neighbor-joining yields the tree of Fig. 4a when terminal A is placed first, but the tree of Fig. 4b when A is placed last. Again, those two trees have no groups in common. But further,

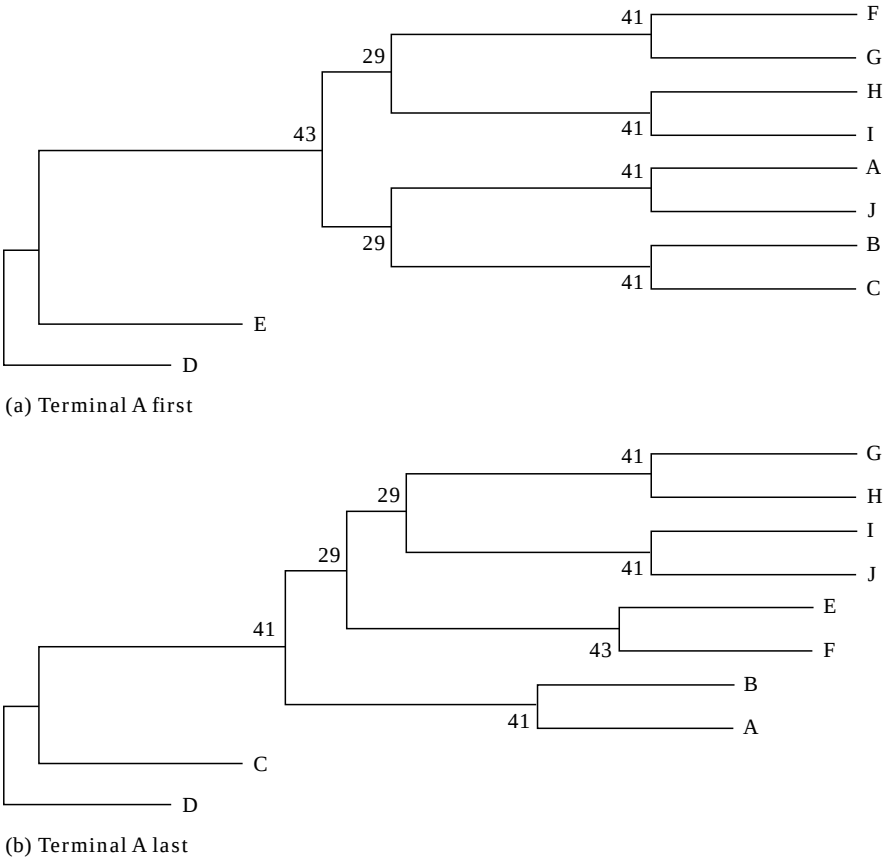


Fig. 4. METREE neighbor-joining trees for two orderings of matrix Two.

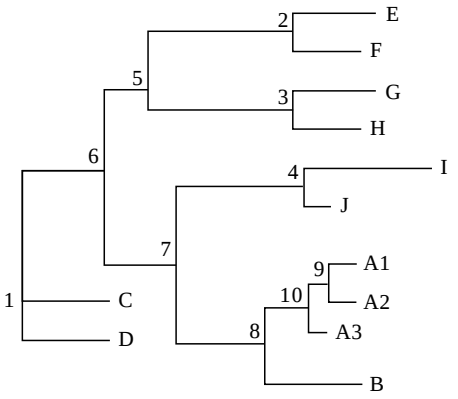
METREE's tree of Fig. 4a has no groups in common with MEGA's tree of Fig. 1a, although these are obtained from the same matrix with the same order of terminals. Likewise for Figs 4b and 1b.

For a last example, replace terminal A of matrix Two with three copies A1–A3, the analyses being performed as before. Making the As a group rather than a terminal causes changes in other parts of the tree. NEIGHBOR (Fig. 5) now puts IJ with AB, not with GH as in Fig. 3a. MEGA's new result (Fig. 6) matches neither tree from Fig. 1, and the two programs now agree only in uniting the As.

It is certainly most convenient to represent conclusions as a single tree, but this hardly means that the tree can be picked arbitrarily. If one program is run with one order of terminals, neighbour-joining seems to provide a single, definite solution, but this is highly misleading when the data are ambiguous. The only single tree that could reasonably be concluded for matrix Two or Three would be an unresolved consensus.

Documentation

While it has been considered before (for example by Sourdis and Nei, 1988), the possibility of multiple trees with neighbor-joining has seldom been emphas-



Between	And	Length
1	6	0.16740
6	5	0.04185
5	2	0.15694
2	E	0.13127
2	F	0.10803
5	3	0.15694
3	G	0.13273
3	H	0.10657
6	7	0.07847
7	4	0.13078
4	I	0.19438
4	J	0.04492
7	8	0.14386
8	10	0.09349
10	9	0.00000
9	A1	0.00000
9	A2	0.00000
10	A3	0.00000
8	B	0.14581
1	C	0.11965
1	D	0.11965

Fig. 5. NEIGHBOR neighbor-joining tree for matrix Two with terminal A replicated.

ized[1]. Kumar et al. (1993) did not mention the subject at all, and Felsenstein (1993) even maintained that multiple trees should not be calculated. It will be illuminating to consider his reasons.

According to Felsenstein (1993, in MAIN.DOC):

> "The '. . . PENNY' programs and CLIQUE are not sensitive to the input order of species, and NEIGHBOR is only slightly sensitive to it, so that multiple Jumbling is not possible with these programs."

Multiple jumbles—random reorderings of the terminals—might well help direct ambiguities, but Felsenstein apparently did not consider this desirable. He never

[1]A noteworthy exception is Backeljau et al. (1996), which appeared after the present paper had been accepted.

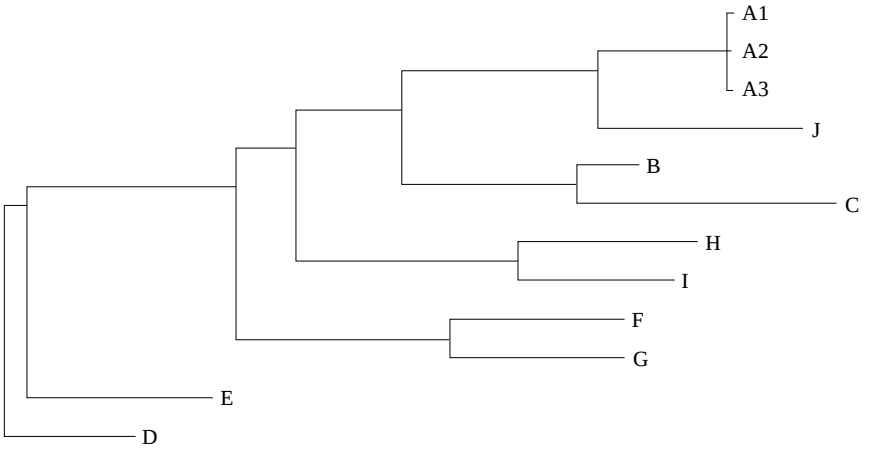


Fig. 6. MEGA neighbor-joining tree for matrix Two with terminal A replicated.

disclosed the evidence for his theory on the order-insensitivity of NEIGHBOR, but in NEIGHBOR.DOC he added further grounds for leaving out multiple orders.

> "The Jumble option (J) does not allow multiple jumbles (as most of the other programs that have it do), as there is no objective way of choosing which of the multiple results is best, there being no explicit criterion for optimality of the tree."

It is a curious explanation. If, as Felsenstein first suggested, order of terminals made little difference, why would multiple jumbling now lead to multiple results? Worse, if there were truly no way to choose among trees, this would scarcely provide a reason for settling on NEIGHBOR's tree to the exclusion of others.

In fact, however, neighbor-joining is based on an optimality criterion, as is clear from Kumar et al.'s (1993: 35) discussion.

> "[Neighbor-joining] is a simplified version of the minimum evolution (ME) method (Saitou and Imanishi, 1989, Rzhetsky and Nei, 1992). In the ME method . . . a topology showing the smallest sum (*S*) of all branches [i.e., branch lengths] . . . is chosen as an estimate of the correct tree . . .. In the case of the NJ method, the *S* value is not computed for all or many topologies, but the examination of different topologies is imbedded in the algorithm, so that only one final tree is produced."

For any one search order, at least. Inasmuch as neighbor-joining is meant to minimize *S* the lack of an optimality criterion cannot very well be the reason for calculating only one tree. Felsenstein (1993, in NEIGHBOR.DOC) had one more reason, though.

> "There is no feature saving multiply [*sic*] trees tied for best, partly because we do not expect exact ties except in cases where the branch lengths make the nature of the tie obvious, as when a branch is of zero length."

Exact ties in the optimality criterion, that is, despite "there being no explicit criterion for optimality". Far from justifying neglect of multiple trees, Felsenstein's view of zero-length branches instead encourages misinterpretation of results.

NEIGHBOR produces zero-length branches in cases like Fig. 5, where the zero length between nodes 9 and 10 means that any other bifurcating resolution of the identical As would lead to the same value of *S*. Of course all of those schemes are really just one trifurcation, as indeed MEGA displays it in Fig. 6. The 9–10 branch indicates a "tie" between "multiple trees" only because NEIGHBOR (like most of PHYLIP) always generates a bifurcating tree, even in the absence of relevant data.

But Felsenstein did not restrict his wording to multifurcations. A NEIGHBOR user might then take the result of Fig. 3a to indicate that there is just one optimal tree, since none of the branch lengths is zero. That inference would be incorrect, for the tree of Fig. 3b has the same value of *S*, as do both trees of Fig. 1, both trees of Fig. 4, and 4 others as well[2].

One might think that, if not literally zero-length, at least the short branches would be the suspect ones. Long branches would still set off well-supported groups. That is not safe either. The longest branches in Fig. 3a delimit pair-groups EF and IJ. Neither of those groups occurs in Fig. 3b, which has the same *S*. In the sense of Bremer's (1988) length difference[3], both groups have zero support. The data provide no grounds at all for concluding either.

Zero lengths as such aside, Felsenstein's last reason for not saving multiple trees is that "we do not expect exact ties". This implies that near-optimal trees should be discarded, but that is not a safe procedure. An example can be obtained by modifying matrix Two.

Notice that the first five sites of matrix Two favor the pair-groups of Fig. 1a, while the last five favor those of Fig. 1b. Produce new matrix TwoX by including 21 copies of each of the first five sites and 20 copies of each of the last five, so that the total number of sites is now 205.

With matrix TwoX, the data now favor Fig. 1a with S=2.55. The grounds for preference are obviously weak, however, and Fig. 1b has near-optimal S=2.59. The groups of Fig. 1a, then, cannot be considered strongly supported. But if only exactly optimal trees were saved, Fig. 1b would never be considered and the weakness of the support would not be recognized.

Improvements

Artificial matrices have been used here to provide easily-understood cases, but similar difficulties can occur with real data matrices. All but the cleanest data allow near-optimal alternatives, and even exact ties are sometimes found (see Backeljau et al., 1996). If conclusions are to be drawn with any confidence, then, a method must be used that can uncover ambiguities. That is easy with small matrices; the problem is to detect ambiguities efficiently enough that large matrices may readily be analysed.

One possibility is Rzhetsky and Nei's (1992) confidence probability (CP) technique, which both MEGA and METREE provide for neighbor-joining trees. A CP value is the complement of the tail probability from a Student's *t*-test, the null

[2]Least-squares branch lengths for all 10 trees can be found with PHYLIP's FITCH program.

[3]Now called Bremer support (cf. Källersjö et al., 1992). Bremer had parsimony in mind, but his idea applies to *S* just as well.

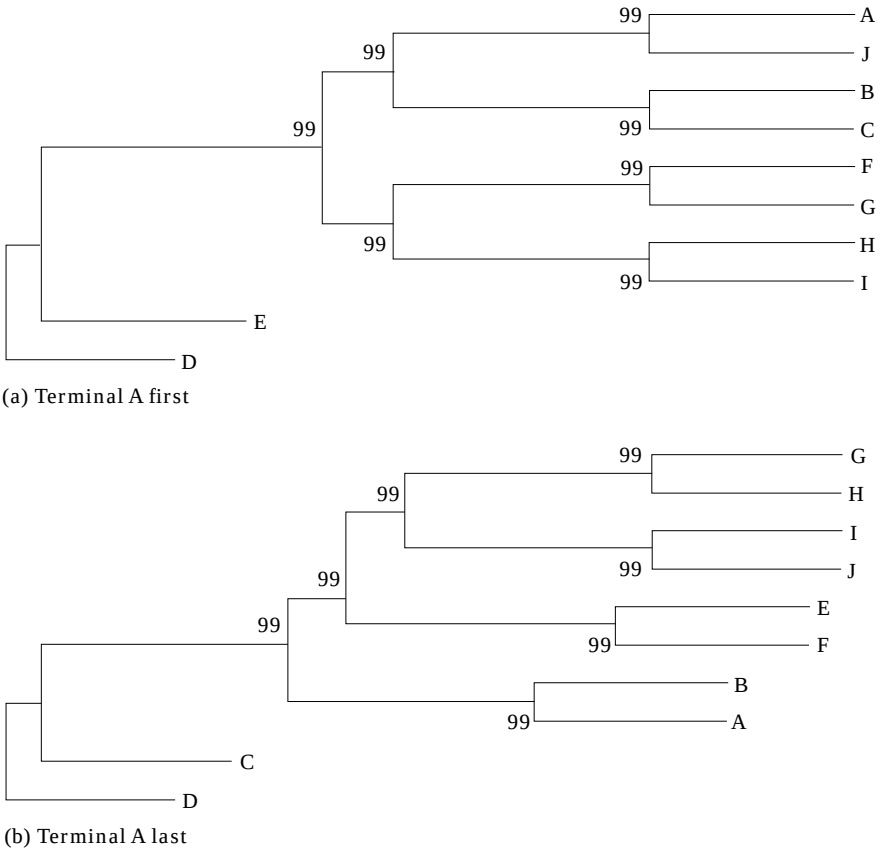


Fig. 7. METREE neighbor-joining trees for two orderings of matrix Two, expanded to 500 sites.

hypothesis being that the length of a branch has zero expectation. CP is intended as the statistical confidence of the conclusion that the branch is real, that is, sets off a true monophyletic group.

This approach seems to work well enough in some cases. Applied to matrix Two (Fig. 4), it assigns low confidence to every branch, correctly indicating that these data offer no substantial grounds for grouping. Nonetheless, CP rests on a serious oversimplification. Rzhetsky and Nei's calculation takes the tree as given, but a realistic assessment of confidence would have to consider the strength of evidence favoring that tree over others.

In particular, CP increases with the number of sites, while support for a tree need not. Again obtain an example by modifying matrix Two. This time produce matrix TwoY by including 50 copies of each site, so that the total number of sites is now 500.

With matrix TwoY analysed as before, the two orderings of terminals still give two trees with no groups in common (Fig. 7). As before, each tree has optimal *S*, so that there are still no grounds for concluding any grouping. But now every branch of each tree is assigned a "confidence probability" of 99%. With this method,

either of the runs by itself would give an entirely misleading impression, so that multiple runs would still be needed.

Rohlf's (1993) NTSYS-pc package offers a different solution to the problem of ambiguity. The neighbor-joining method is extended to search for multiple trees. Unfortunately, promising though this sounds, it does not seem to work well in practice. When instructed to find multiple neighbor-joining trees for small matrix Two, NTSYS-pc ran out of memory and crashed the computer.

Another disadvantage of Rohlf's program is that its execution time increases rapidly with the number of terminals, even when the multiple tree option is not used. For the first 25 terminals from Chase et al.'s (1993) 500-terminal 'Search II' data, NTSYS-pc's neighbor-joining took just 10 seconds on a 66 MHz 80486DX2. But for the first 50 terminals, 261 seconds were required; doubling the number of terminals increases the execution time by a factor of 26. At that rate, calculating a neighbor-joining tree for the whole matrix would take over five months.

METREE provides an extended neighbor-joining method based on a local branch-swapper. Data permitting, this can reduce *S* by rearranging a tree found by simple neighbor-joining, and it can also retain multiple optimal and near-optimal trees.

Such improvements would in principle avoid the arbitrary selections made by the original neighbor-joining algorithm, but METREE's branch-swapper has some drawbacks. The trees of Fig. 1 each have optimal *S* for matrix Two, but METREE could not find both of them in any one of our runs. Much like MEGA's neighbor-joining (and unlike METREE's own neighbor-joining), the branch-swapper found the tree of Fig. 1a if given the original matrix, one similar to Fig. 1b if given the reordered matrix.

Multiple reordering could be helpful in such situations, but the present program makes no provision for it. Further, processing several orders would increase total execution time, and METREE can already take a long time to analyse a poorly-structured matrix. This is seen even with a matrix as small as Three, for which METREE ran for more than 70 minutes on the 80486 mentioned above. When finally interrupted, the program had not yet completed its local branch-swapping.

To put this in perspective: Hennig86's *mh*bb** command sequence completed its global branch-swapping of matrix Three in 0.1 second on the same computer[4]. Whereas Stoneking et al. (1992) considered parsimony branch-swapping too slow to be practical for large matrices, METREE's branch-swapping can be far slower.

Bootstrapping

If the matrix is large, bootstrapping with a quick clustering method like neighbor-joining can go faster than a branch-swapping analysis of just the original data. In practice this offers a substitute for finding multiple optimal and near-optimal trees. The several trees would end up being reduced to a consensus anyway, so that only better-supported groups would survive. Much the same effect can be achieved by eliminating groups with low bootstrap frequencies.

[4]Hennig86 is available from AGK or DL.

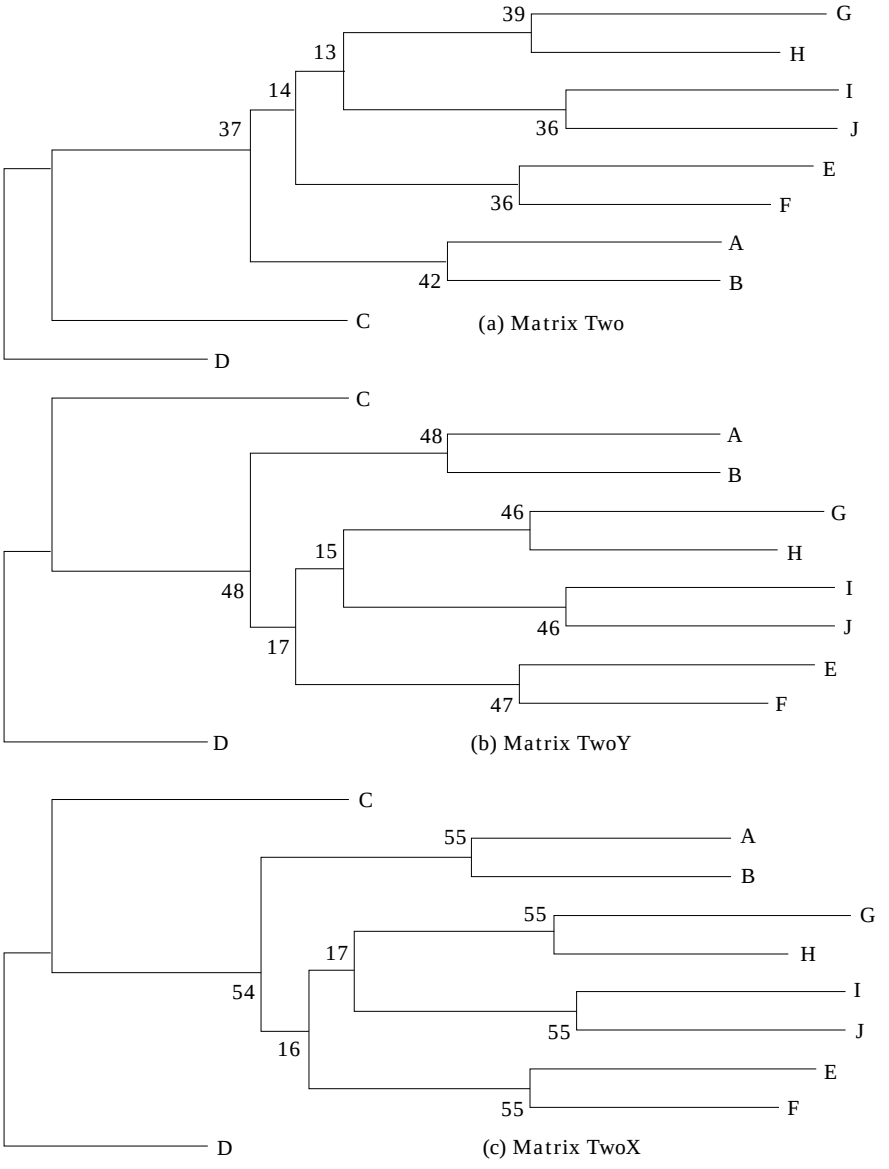
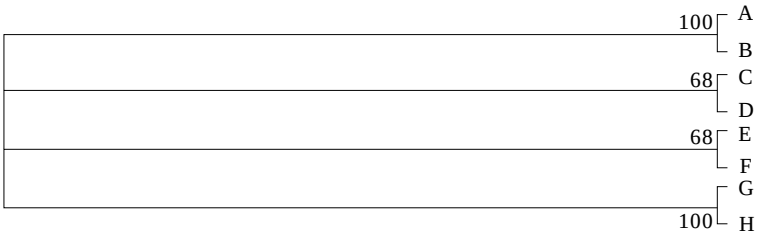


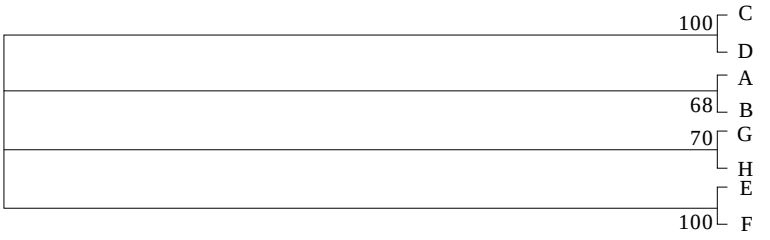
Fig. 8. MEGA neighbor-joining bootstrap trees for (a) matrix Two, (b) matrix TwoY and (C) matrix TwoX.

In Felsenstein's (1985) original proposal, the bootstrap frequency of a group would represent the statistical confidence level of a conclusion of monophyly. That view has limitations, some of which Felsenstein himself pointed out, but these do not matter for present purposes. We will not rely on that interpretation here, but consider the method simply as a way of discovering ambiguities in data.

How this can work is illustrated by bootstrapping matrix Two (Fig. 8a). Recall



(a) Order ABCDEFGH



(b) Order CDABGHEF

Fig. 9. MEGA neighbor-joining bootstrap trees for two orderings of matrix Four.

that the matrix is ambiguous because equal numbers of sites favor each of two alternative suites of pair-groups, shown by the two trees of Fig. 1.

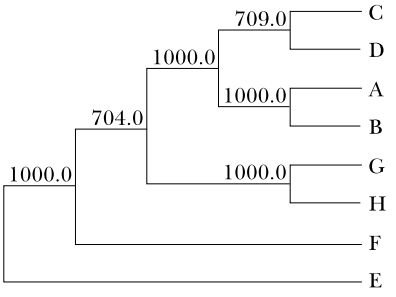
Ties are unlikely to survive random resampling; the relative frequencies of the original sites will typically change in each replicate. The sites favoring a group might at best predominate half the time, and the group will actually be formed less often still, since resampling may omit its sites entirely. All the groups of Fig. 8a thus have frequencies below 50%, and such groups should be discarded from bootstrapping results, as discussed later. Only an unresolved tree remains, evidently the right conclusion for matrix Two.

Bootstrapping largely lacks the drawback of CP illustrated before. In the bootstrap of 500-site matrix TwoY (Fig. 8b), the groups still have frequencies below 50%, so that the right conclusion would be reached. The method also effectively takes near-optimal alternatives into account. Bootstrapping 205-site matrix TwoX (Fig. 8c) gives a frequency of about 55% for each pair-group, correctly indicating positive but weak support.

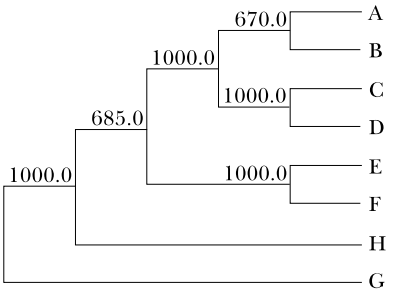
Some precautions, however, must be taken in applying this approach. Bootstrap frequency would be used to indicate group support, but MEGA and NEIGHBOR are not quite ideal for that purpose. Matrix Four will provide an illustration.

Terminal	A	B	C	D	E	F	G	H
	a	a	t	t	t	t	t	t
	t	t	a	a	t	t	t	t
	t	t	t	t	a	a	t	t
	t	t	t	t	t	t	a	a

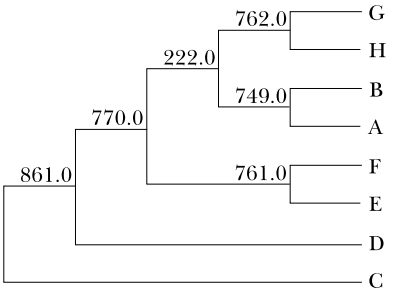
The trees of Fig. 9 were produced by bootstrapping this matrix with 1000 repli-



(a) Order ABCDEFGH



(b) Order CDABGHEF



(c) Jumbled

Fig. 10. PHYLIP neighbor-joining bootstrap trees for three ways of ordering matrix Four.

cates, using MEGA's neighbor-joining. They are rooted by the midpoint method[5], the default in MEGA. MEGA complained of "Failure in estimating distances" when asked to bootstrap matrix Four with the Kimura distance, so we used number of sequence differences. The same distance measure was provided to NEIGHBOR for similar bootstrapping, yielding the unrooted trees of Fig. 10.

For this matrix, MEGA's bootstrap frequencies show no consistent relationship to strength of support. In Fig. 9a, MEGA reports 68% for group CD, but 100% for

[5]Of Farris (1972), although MEGA's documentation does not mention this.

AB. With 1000 replicates, that difference is very highly significant ($P \ll 0.0001$). But it is plain from matrix Four that those two groups are supported by precisely equal amounts of evidence. This is another manifestation of order-sensitivity. In Fig. 9b, obtained by changing the order of terminals to CDABGHEF, the frequencies are reversed.

68% is the right value. Just one of the four sites sets off group AB, and that site will be included once or more in any one bootstrap replicate with probability $1-(3/4)^4$, or 0.6836. (The general formula is given later.) Likewise for the other pair-groups CD, EF and GH. The 100%, if it were real, would imply a substantially larger number of supporting characters. If the data comprised four distinguishing sites for each pair-group, the correct frequency for each group would still be only $1-0.75^{16}$, just under 99%.

In Figs 10a and 10b, NEIGHBOR also shows the AB/CD frequency switch, but now combined with a further peculiarity. Group ABCD has 100% frequency, although it is supported by no evidence whatever. This problem has two causes. The particular choice of terminals is order-sensitivity again. But the branch setting off that group is "resolved" in every one of 1000 replicates because NEIGHBOR always produces a bifurcating tree, even in the absence of evidence. MEGA can avoid that difficulty in Fig. 9 because it makes an effort to suppress zero-length branches.

NEIGHBOR's jumble option is beneficial here. If used in bootstrapping, it causes a different, randomly-chosen order of terminals to be applied for each replicate[6]. As seen in Fig. 10c, this does away with order-sensitivity. The variation among the frequencies of the four pair-groups is now well within sampling error.

The reordering reduces the influence of zero-length branches, but does not eliminate it. Group ABGH of Fig. 10c has only 22% frequency, not the 100% seen before for ABCD (Fig. 10a). But it is still formed, although matrix Four includes no evidence for it. More subtly, the pair-group frequencies are all too high. Averaging nearly 77%, they exceed the correct 68% value by almost 9%, a very highly significant difference ($P \ll 0.0001$). The ideal program would have to include zero-length branch elimination as well as reordering.

Finally, there is the matter of groups with low bootstrap frequencies. Felsenstein (1993, in CONSENSE.DOC) stressed that such groups should not be retained, but his program may nonetheless include them in the tree (Fig. 10c)—an anomaly on which he did not comment. Deleting them is left to the user.

> "You have to decide on the percentage level, figure out for yourself what number of occurrences that would be (e.g. 15 in the above case for 100%), and resolutely ignore any group below that number. Do not use numbers at or below 50%, because some groups occurring (say) 35% of the time will not be shown on the tree. The collection of all groups that occur 35% or more of the time may include two groups that are mutually self contradictory and cannot appear in the same tree."

So that accepting a low-frequency cluster from the diagram amounts to grouping arbitrarily. Unfortunately, not all users understand this. Wainwright et al. (1993), for example, based their main conclusions on groups with PHYLIP bootstrap frequencies below 50% (cf. Rodrigo et al., 1994). It would be safer if the program discarded those groups before printing the tree.

[6]We thank N. Wikström for calling this to our attention. It is not pointed out in NEIGHBOR.DOC.

Misinterpretations such as Wainwright et al.'s probably result from mistaking the effective starting-point of the scale relating support to bootstrap frequency. If zero support meant zero frequency, then any positive frequency would suggest at least some support. But in fact the starting-point is higher. The 22% frequency for group ABGH in Fig. 10c, again, corresponds to no evidence whatever.

Of course that group is only an artifact of NEIGHBOR's treatment of zero-length branches, but a more substantial effect is seen on bootstrapping matrix Two (Fig. 8a). The pair-groups there are hardly artifacts; they correspond to sites in the data. But other sites dispute them, and matrix Two collectively provides no grounds for preferring group AB, for example, over alternative AJ. The shortest trees with AB have the same *S* as those with AJ. AB has zero support, but its bootstrap frequency is over 40%.

That is not the worst case. Matrix Five has four terminals and 1000 informative binary characters. Half the characters split the terminals AB/CD, the other half, AC/BD. The support for either arrangement is again zero. But, idiosyncrasies of programs aside, each of the groups must have a bootstrap frequency of 50%. No lower frequency, then, can sensibly be interpreted as indicating favorable evidence.

Jackknifing

Neither MEGA nor NEIGHBOR eliminates both order-sensitivity and zero-length branches. But a suitably-designed parsimony program can do so, and developing one provides the opportunity to improve the resampling method as well. As will be seen presently, the advantage of the approach taken here is that it provides a great reduction in the time needed for analysis of large matrices.

Kumar et al. (1993: 46) thought of a way to ensure that parsimony analysis would take longer than neighbor-joining.

> "If the number of [terminal taxa] is so large that we have to use the [approximate] search, the bootstrap test is not very meaningful, because we are not sure whether the tree obtained is the [most-parsimonious] tree."

By that reasoning, they should have used exact minimization of *S*—not just simple neighbor-joining—in their own bootstrapping, but of course they did no such thing. The practical benefit of resampling is that it effectively replaces extensive analysis of the original matrix. For the parsimony calculations in our method we use a fast approximate procedure, similar to the *hennig* command of Hennig86.

The *hennig* algorithm is sensitive to the order in which terminals are added to the tree (cf. Farris, 1970), but randomly-selected orders can be used, as in PHYSYS' PIMENTEL[7] command. Our procedure randomly generates a new order for each replicate.

Hennig86 deletes zero-length branches in a sense, but its method is not the best one for present purposes. It discards only branches that would show no change under any parsimonious reconstruction of the states of stem species on the considered tree[8]. It may thus retain groups that are not actually supported by any evi-

[7]For R. A. Pimentel, who first used this method (cf. Mickevich and Farris, 1981: 353).

[8]This criterion was incorporated into Hennig86 at the suggestion of N. I. Platnick—who later reconsidered (cf. Platnick et al., 1991).

dence, and here the aim is to eliminate such groups. In our method, a branch is retained only if it must show a change under every parsimonious reconstruction[9]. This rule is applied separately to the tree for each replicate.

For resampling, we use an independent-removal jackknife, rather than bootstrapping. A resampled matrix (replicate) is formed by deleting characters randomly and independently from the original matrix, each original character having the same fixed probability p of being absent from any one replicate. The advantage of this method is that it simplifies the relationship between group frequency and support. Provided the data have no missing entries, the expected jackknife frequency of a group G set off by r uncontradicted characters is just $1–p^r$.

The simplification can be appreciated from Harshman's (1994) discussion of a peculiarity of bootstrapping. If the matrix has n characters in all, the expected bootstrap frequency of G is $1–(1–r/n)^n$. With r fixed, this decreases as n increases, leading to the counterintuitive result that adding characters logically irrelevant to G—invariable ones, autapomorphies, even characters of separate groups—can lower the bootstrap frequency of G.

In the case of autapomorphies and invariable characters, this problem might be circumvented simply by deleting those characters. But as Harshman pointed out, this can hardly be done for characters of separate groups.

Not wishing to abandon bootstrapping, Harshman maintained that this effect is "small over the range of n likely to be encountered in real data sets" (p. 420). That claim does not seem to hold up, since Carpenter (1996) has recently found larger effects in several published matrices. But Harshman's (p. 422f) concluding suggestion might still be employed.

> "Anyone concerned about the variation in bootstrap values due to irrelevant characters can equalize the effect of irrelevant characters on all nodes by adding a large number of invariant characters (1,000 seems more than adequate) to the data matrix."

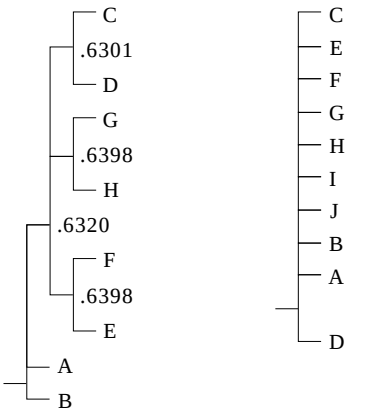
This works because $1–(1–r/n)^n$ settles down to a limit $1–e^{–r}$ if n is made large enough. Of course, that strategy would generally change the frequency of G; Harshman evidently considered any difference between the original $1–(1–r/n)^n$ and the limit $1–e^{–r}$ inconsequential. But in that case, one might just as well use our method, with the removal probability p set to $e^{–1}$ (about 0.3679). That also makes the expected frequency of G simply $1–e^{–r}$, and then there is no need to be concerned with either irrelevant or dummy characters.

Felsenstein (1985: 787) preferred bootstrapping to jackknifing, but his reasons pertained to the classical jackknife: dropping just one character at a time[10]. Trees from different replicates would not usually vary much, and he observed:

> "To make the variance among the jackknife estimates as large as that among the bootstrap estimates, one would have to engage in an extrapolation to make their variances larger. The difficulty in envisaging a procedure like this is that the space of possible phylogenies does not lend itself readily to extrapolation."
>
> That difficulty does not apply to all jackknifing. (Felsenstein, 1985: 787)

[9]This method is also used in P. Goloboff's efficient branch-swapping program NONA. Concerning which, contact Dr J. Carpenter, Dept. Entomology, American Museum of Natural History, Central Park West at 79th St, New York, New York 10024 USA.

[10]For a discussion of resampling methods, see Efron and Gong (1983). Earlier uses of resampling techniques in systematics were reviewed by Farris (1971).



(a) Matrix Four (b) Matrix Two

Fig. 11. Jac parsimony jackknife trees for matrices (a) Four and (b) Two or Three.

> "One way to make the jackknife vary as much as the bootstrap would be to drop not one observation, but half the observations chosen at random. This possibility is worth exploring."

He did not propose any other ways. PHYLIP now includes an option for a delete-half jackknife and, according to Felsenstein (1993, in SEQBOOT.DOC):

> "The random variation from doing this should be very similar to that obtained from the bootstrap."

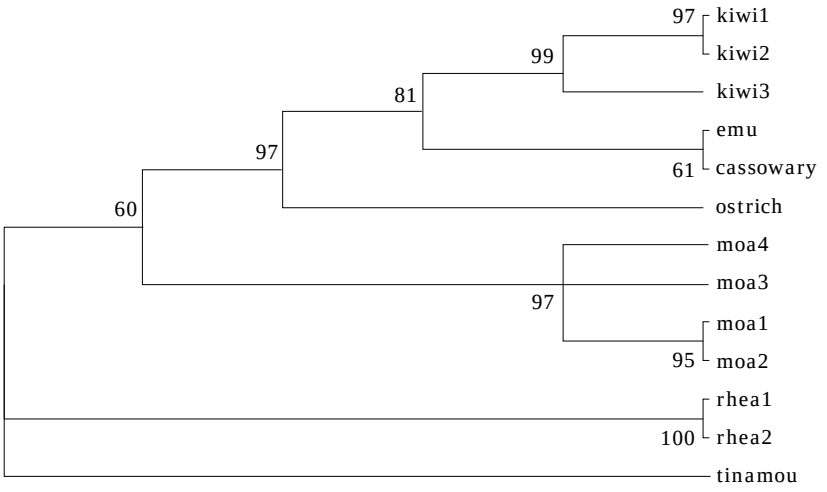
So Felsenstein offered no objection to jackknifing in general, but his last proposal raises another issue. If group jackknife frequencies were to agree with bootstrap frequencies in the limit, the fraction of characters deleted when jackknifing would have to be e^{-1}, not 50%.

Half the characters is too much to delete. With that method, the expected frequency of a group set off by one uncontradicted character would be 50%. But a group with no support can also attain that group frequency, as matrix Five illustrates. p should be kept substantially smaller than 50%, then, in order to maintain a useful relationship between group frequency and support. In our method we use $p=e^{-1}$.

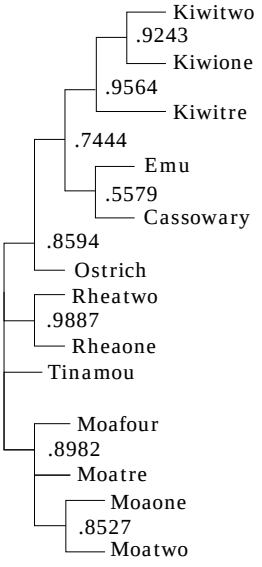
We have incorporated these ideas into a program, Jac[11], that reads a matrix of nucleotide-sequence data, performs resampling internally, and produces a tree with group frequencies. The features of our program seem successful in overcoming the various difficulties seen in MEGA and NEIGHBOR. The tree obtained by jackknifing matrix Four 10 000 times (Fig. 11a) shows neither order-sensitivity nor inclusion of zero-length branches. The group frequencies differ only by sampling error from their common expectation $1–e^{-1}$. Similar analysis of matrix Two (or Three) yields an unresolved tree, as one would like (Fig. 11b). The tree-printer automatically eliminates groups with frequencies below 50%.

An illustration of the use of our program is provided by the ratite 12S rRNA data of Cooper et al. (1992; cf. Albert and Bremer, 1993), which comprise 370 sites for 13 terminals. MEGA's neighbor-joining bootstrapping tree (Fig. 12a) is presented

[11]A menu-driven extended DOS version is available from DL or AGK.



(a) MEGA



(b) Parsimony jackknife

Fig. 12. Trees for Cooper et al.'s (1992) ratite 12S rRNA sequence data. (a) MEGA neighbor-joining bootstrap. (b) Jac parsimony jackknife.

with the result from parsimony jackknifing (Fig. 12b). The tinamou is the outgroup. For Fig. 12a, MEGA's optional deletion of low-frequency groups was employed. The two trees differ only in a group that is at best marginal (60% in Fig. 12a), but it is worthwhile to extend the comparison.

Even with $p=e^{-1}$, jackknife frequencies need hardly agree perfectly with bootstrap values. That aside, the group frequencies from Jac are generally lower than those

from MEGA because Jac's elimination of zero-length branches is more thorough. The thoroughness is desirable. The whole point of using resampling methods, after all, is to avoid drawing poorly-supported conclusions.

These data provide strong support for some groups, as is evident from the jackknife frequencies of Fig. 12b . One might think that MEGA's tree demonstrates the same, but such is not the case. MEGA can—and for these data apparently does—assign high frequencies to well-supported groups. But because of its order-sensitivity, it can also attribute high frequency to clusters with little support. Group frequencies can be interpreted safely only if obtained by a method free of such weaknesses.

Large Matrices

Another difference between Jac and MEGA is speed. For the ratite data, MEGA's neighbor-joining with 1000 bootstrap replicates required 146 seconds on a 66 MHz 80486DX2. On the same computer, Jac took 10.8 seconds for 10 000 replicates. For this matrix, MEGA's execution time per replicate is 135 times longer than Jac's.

The speed advantage of parsimony jackknifing is greater for large matrices, but illustrating this involves some complications. We first simply tried to compare timings for Chase et al.'s (1993) 'Search I' *rbc*L sequences, with 1428 sites and 479 terminals. This did not work. MEGA read[12] the matrix, but crashed the computer when asked to calculate a tree. PHYLIP's DNADIST announced "Error allocating memory"—on a computer with 16 megabytes of RAM. We then decided to extrapolate from timings for subsets of the terminals.

Extrapolating for PHYLIP requires considering the multistage process used for bootstrapping in that package. Working from the original data, SEQBOOT first writes bootstrap replicate matrices to a file. DNADIST reads that file and produces one containing a distance matrix for each replicate. NEIGHBOR then reads the distances and creates still another file, containing a tree for each replicate. Finally, CONSENSE reads those trees and combines them into a bootstrap tree.

Each stage has its own relationship to the number *t* of terminals, as is seen from runs with the first 100 and first 200 terminals from the 'Search I' matrix. All 1428 sites were included in both cases. Using the same 486, the times in seconds for 3 replicates were:

t	SEQBOOT	DNADIST	NEIGHBOR	total	Jac (300)
100	6.6	310.8	51.6	369.0	98.1
200	16.5	1526.8	1349.4	2892.7	607.2

CONSENSE takes relatively little time and will be ignored. The total is that for the three PHYLIP programs. Jac times for 300 replicates are given for comparison. With 200 terminals, PHYLIP's total execution time per replicate is 476 times longer than Jac's.

The extrapolation will be based on relating execution time to a suitable power

[12]After some editing. Question-marks had to be substituted for IUPAC codes K, S, and Y, which MEGA cannot process (cf. Farris and Källersjö, 1994).

of *t*. SEQBOOT's execution time increases linearly with *t* (i.e. as t^1), while DNADIST's increases as t^2, and NEIGHBOR's as at least t^4. All three exponents are conservative, which will favor PHYLIP in comparisons.

The exponents for SEQBOOT and DNADIST are uncontroversial, but Felsenstein (1993, in NEIGHBOR.DOC) held a different view on the exponent for NEIGHBOR.

> "The major advantage of NEIGHBOR is its speed: it requires a time only proportional to the square of the number of species . . .. Thus NEIGHBOR is well-suited to bootstrapping studies and to analysis of very large trees."

His theory is thoroughly at odds with the data. t^2 for NEIGHBOR would predict only a fourfold difference between the times for 100 and for 200 terminals, while the actual ratio is over 26[13].

MEGA performs its resampling internally, and the various calculations cannot be timed separately. Fortunately, the high-exponent part should predominate if *t* is large enough. For 5 replicates with 80 and 90 terminals (MEGA would not run with 100), we obtained timings of 103 and 149 seconds, respectively. MEGA's neighbor-joining execution time then appears to increase as t^3. This is as it should be, if (as seems likely) the program employs Studier and Kepler's (1988) improvements to the Saitou/Nei algorithm.

To extrapolate to an analysis of the whole matrix, the 149 second timing for MEGA is multiplied by $(479/90)^3$ to account for the number of terminals, and again by (1000/5) for 1000 replicates. This gives a predicted time of 52 days. For PHYLIP, each stage is extrapolated separately from the 200-terminal case. The factor for replicates is (1000/3), and that for terminals is the appropriate power of (479/200). The total comes to 205 days—recall that this figure is conservative. No extrapolation is needed for Jac, which completed 1000 replicates of the full matrix in 6.8 hours on the same 486.

PHYLIP's bootstrapping has the further drawback that it may create very large files. DNADIST's input file of sequence-matrix replicates and its output file of distance matrices are open simultaneously. One replicate of the 'Search I' data takes over 690 K of disk space, and one triangular distance matrix (if DNADIST would calculate it) would take over 929 K. For 1000 replicates, the total disk space required would exceed 1.5 gigabytes, far more than a desktop computer is likely to have available. No such difficulty arises with the internal resampling used by Jac and MEGA.

Jac can also be compared to PAUP's parsimony bootstrapping. PAUP's way of removing zero-length branches is like Hennig86's and so would not generally be satisfactory for this application, but this does not matter for comparing execution times. For all PAUP runs we used version 3.1.1 with the fastest options: one random addition sequence per bootstrap replicate, and no branch-swapping.

With the fastest options, PAUP took 3.5 hours to complete ten replicates of the full, 479-terminal matrix on a 33 MHz 68040 (Macintosh Quadra 650). 1000 replicates would then take 14.6 days, better than MEGA's 52 days but still considerably longer than Jac's 6.8 hours.

But then that 6.8 hour timing was obtained on the 486, not the Quadra. To

[13]NTSYS-pc's neighbor joining has a similar ratio, but is even slower, taking nearly 400 times as long as NEIGHBOR.

check the performance of different computers, we recompiled Jac for other chips, then used each version to run 100 replicates of Albert et al.'s (1992) 100-terminal matrix of *rbc*L sequences. The times in seconds were:

PAUP-68040	Jac-68040	Jac-PowerPC	Jac-Pentium
2 354.4	75.3	26.6	28.3

Run on the same computer, Jac is 31 times faster for the 100-terminal matrix. Jac's advantage increases with the number of terminals, as is seen from the times for 10 replicates of the 'Search I' matrix.

PAUP-68040	Jac-68040	Jac-PowerPC	Jac-Pentium
12 615.7	324.0	135.0	142.2

With 479 terminals, the 68040 version of Jac is 39 times faster than PAUP's bootstrapping.

That the newer chips are faster is unsurprising, but comparing Jac times between them proves informative. The 60 MHz Pentium takes 5–6% longer than the 66 MHz PowerPC 601, but this is less than the 10% difference in clock frequency. In contrast to what might be expected from claims made by some manufacturers, the Pentium seems to be at least as efficient as the PowerPac at this type of calculation.

Branch-Swapping

Applications of parsimony in the recent literature have increasingly employed extensive branch-swapping in the effort to find all most-parsimonious trees. Far too slow to be practical, this approach is also unlikely to be adequate as a way of discovering ambiguities in data. Parsimony jackknifing provides improvements in both respects.

The need for a faster method is apparent from some recent studies of *rbc*L sequences. For their 500-terminal 'Search II' matrix, Chase et al. (1993) spent four weeks of calculation on a Quadra. Albert et al. (1992) used more extensive branch-swapping for their 100-terminal matrix, and this took three weeks on a Macintosh IIfx. Olmstead et al. (1993) worked harder still, devoting over 20 weeks of IIfx time to analyzing their 105-terminal asterid data.

Jac took four minutes to complete 1000 replicates of that 105-terminal matrix on a 66 MHz PowerPC 601 (23 minutes on a IIfx). Similar analysis of the 100-terminal matrix, which has a greater number of informative sites, took 4.3 minutes. Even for the 500-terminal 'Search II' data, 1000 replicates took only 3.5 hours.

Olmstead et al. (1993: 701) stressed the importance of thoroughness.

> "An inadequate analysis, in which a search for multiple islands is not carried out, may produce exaggerated resolution, lead to misleading conclusions, and deter potential lines of further research."

Exaggerated resolution of the consensus tree, that is, and they certainly tried to avoid that pitfall. Their consensus (their Fig. 3) was obtained from 8454 most-parsimonious trees. But while they experimented with various combinations of PAUP

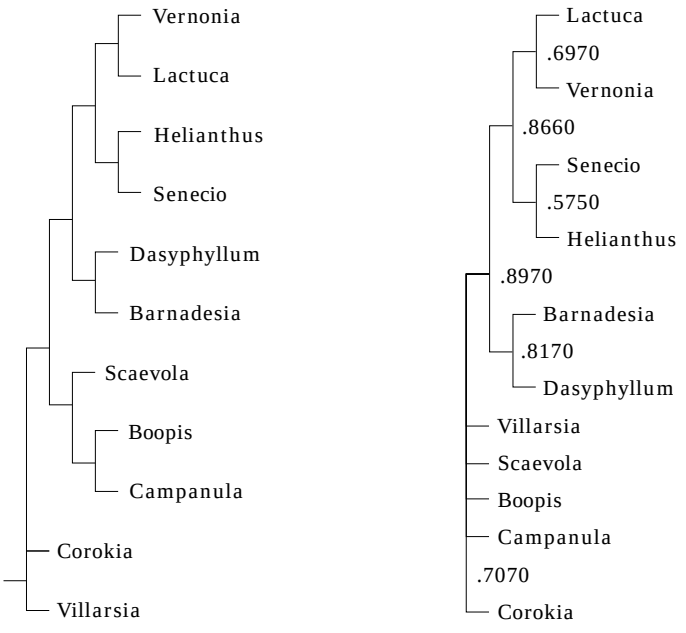


Fig. 13. Trees for Asteraceae and relatives, obtained from analyses of Olmstead et al.'s (1993) 105-terminal matrix. (a) After their Fig. 3. (b) Parsimony jackknife.

options in their searches, there is nothing to insure that no other combination[14] could find further trees. And as they pointed out, even if all most-parsimonious trees had been found, a group present on the consensus need hardly be strongly supported. A one-step difference in length—out of 3597 steps in this case—would suffice to retain it.

The same applies to the other studies, and Jac's results for the three matrices show a consistent pattern. While in no case does a group with high jackknife frequency conflict with groups on the published consensus, several of the latter are absent from the jackknife tree. The published consensus trees, then, are more resolved than the data justify[15]. The branch-swapping analyses may have missed some most-parsimonious trees. At the best, the published consensus included poorly-supported groups, groups that would have been lost, had slightly longer trees been considered.

An example of the differences between our results and those of Olmstead et al. is illustrated in Fig. 13, which shows parts of larger trees. In their consensus tree (Fig. 13a), *Campanula* is grouped with *Boopis*. In the jackknife analysis (Fig. 13b), that group is collapsed, as are two enclosing ones.

Why *Campanula+Boopis* is weakly supported is seen from the corresponding part

[14]Or program. What are several separate islands to PAUP may be just one stand to NONA's more sophisticated branch-swapper.

[15]But Albert et al.'s (1992) conclusions, on convergent evolution of insectivory, are still well supported. These depend only on four main groups (cf. their Fig. 1), all of which have jackknife frequencies of 89% or greater.

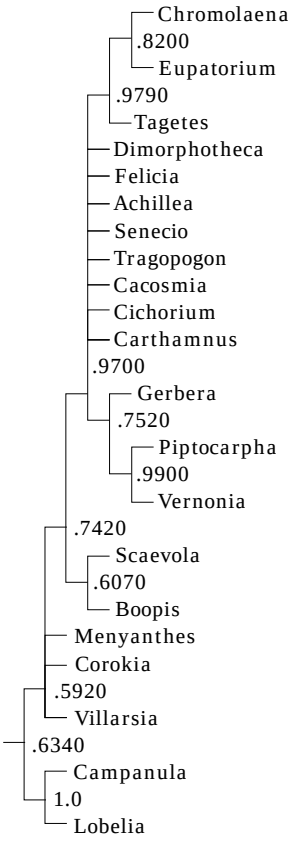


Fig. 14. Tree for Asteraceae and relatives, obtained from a parsimony jackknife analysis of Chase et al.'s (1993) 500-terminal 'Search II' matrix.

of the jackknife tree for the 500-terminal 'Search II' matrix (Fig. 14). There *Campanula* is instead grouped with *Lobelia*. *Boopis+Scaevola* is sister to the Asteraceae, in agreement with the findings of Gustafsson and Bremer (1995).

Fig. 15 shows the same part of the jackknife tree for an 1120-terminal *rbc*L matrix extracted from Genbank by MK. The relationship of *Boopis+Scaevola* to the Asteraceae is now still better supported. With the more extensive sample of terminals, the strongly supported and quite separate group containing *Campanula* and *Lobelia* is now readily recognized as the Campanulaceae.

Campanula could not be placed with other Campanulaceae in Olmstead et al.'s analysis simply because none had been included. In this case the restricted selection of terminals led to just the kind of misleading result that they had hoped to avoid. Ironically, they had used a small suite of terminals precisely in order to make extensive branch-swapping feasible. The speed of parsimony jackknifing makes it easy to avoid such difficulties by including all relevant information.

Parsimony jackknifing can eliminate unjustified resolution more efficiently than can branch-swapping because much of what branch-swapping does is unnecessary

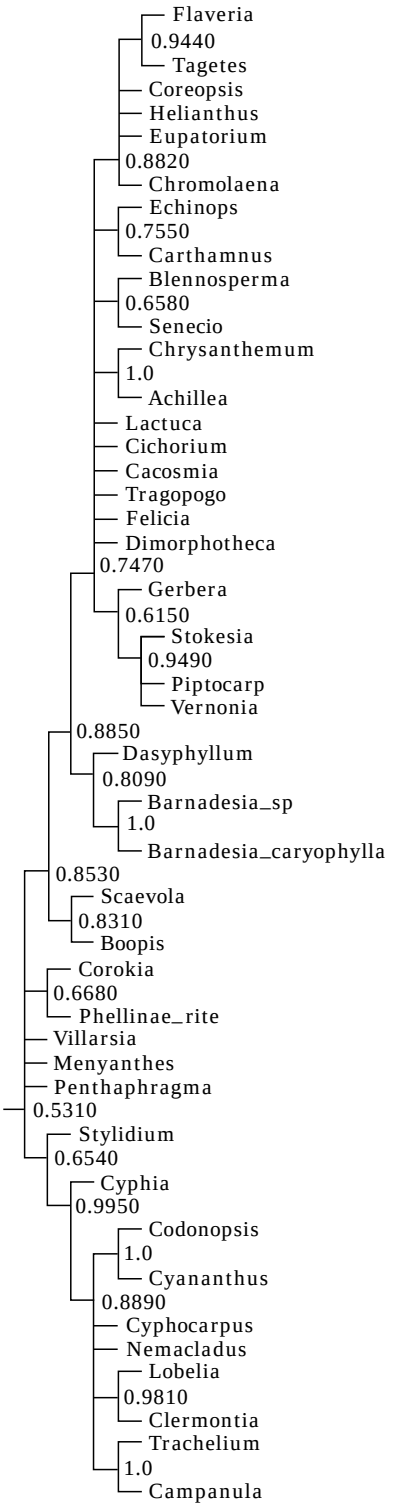


Fig. 15. Tree for Asteraceae and relatives, obtained from a parsimony jackknife analysis of 1120 *rbc*L sequences.

for this purpose. Olmstead et al.'s branch-swapping produced 8454 distinct most-parsimonious trees, to bring out the obvious, because those trees differ in several groups. But such groups have no support and will necessarily be lost in the consensus. The effort of detailing them and the many trees that they distinguish is wasted.

Branch-swapping has been used because finding most-parsimonious trees is known to be a difficult problem, one that requires laborious methods. But the difficulty lies in finding most-parsimonious trees exactly, and that effort is again unnecessary. Forming strongly-supported groups is much simpler, precisely because they are strongly-supported; even an approximate method such as *hennig* is unlikely to miss them. Only a means of eliminating poorly-supported groups from *hennig* trees is then needed, and jackknifing accomplishes this automatically. Those groups cannot survive resampling.

Acknowledgments

This work was supported by NFR grant 10204-300 to J. S. Farris. We thank Drs K. Bremer and J. Carpenter for their excellent advice and strong encouragement. We particularly thank Dr W. Wheeler for sharing his highly entertaining views on compatibility methods.

REFERENCES

ALBERT, V. A. AND K. BREMER. 1993. Flying kiwis and pattern information in biogeography. Cur. Biol. 3: 324–325.

ALBERT, V. A., S. E. WILLIAMS, AND M. W. CHASE. 1992. Carnivorous plants: Phylogeny and structural evolution. Science 257: 1491–1495.

BACKELJAU, T., L. DE BRUYN, H. DE WOLF, K., JORDAENS, S., VAN DONGEN, AND B. WINNEPENNINCKX. 1996. Multiple UPGMA and neighbor-joining trees and the performance of some computer packages. Mol. Biol. Evol. 13: 309–313.

BREMER, K. 1988. The limits of amino-acid sequence data in angiosperm phylogenetic reconstruction. Evolution 42: 795–803.

CARPENTER, J. M. 1996. Uninformative bootstrapping. Cladistics 12 (in press).

CHASE, M., D. SOLTIS, R. OLMSTEAD, D. MORGAN, D. LES, B. MISHLER, M. DUVALL, R. PRICE, H. HILLS, Y. QUI, K. KRON, J. RETTIG, E. CONTI, J. PALMER, J. MANHERT, K. SYTSMA, H. MICHAELS, W. KRESS, K. KAROL, W. CLARK, M. HEDREN, B. GAUT, R. JANSEN, K. KIM, C. WIMPEE, J. SMITH, G. FURNIER, S. STRAUSS, Q. XIANG, G. PLUNKETT, P. SOLTIS, S. SWENCH, S. WILLIAMS, P. GADEK, C. QUINN, L. EGUIARTE, E. GOLENBERG, G. LEARN, S. GRAHAM, S. BARETT, S. DAYANANDAN, AND V. A. ALBERT. 1993. Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene *rbc*L. Ann. Missouri Bot. Gard. 80: 528–580.

COOPER, A., C. MAOURER-CHAUVIRÉ, G. K. CHAMBERS, A. VON HAESELER, A. C. WILSON, AND S. PÄÄBO. 1992. Independent origins of New Zealand moas and kiwis. Proc. Nat. Acad. Sci. U.S.A. 89: 8741–8744.

EFRON, B. AND G. GONG. 1983. A leisurely look at the bootstrap, the jackknife, and cross-validation. Amer. Statist. 37: 36–48.

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

FARRIS, J. S. 1971. The hypothesis of nonspecificity and taxonomic congruence. Ann. Rev. Ecol. Syst. 2: 277–302.

FARRIS, J. S. 1972. Estimating phylogenetic trees from distance matrices. Am. Nat. 106: 645–668.

FARRIS, J. S. 1977. On the phenetic approach to vertebrate classification. In: M. P. Hecht, P.

Goody, and B. M. Hecht (eds.) Major patterns in vertebrate evolution. Plenum, New York, pp. 823–850.

FARRIS, J. S., AND M. KÄLLERSJÖ. 1994. Pro bono publico. Cladistics 10: 85–88.

FELSENSTEIN, J. 1985. Confidence limits on phylogenies: An approach using the bootstrap. Evolution 39: 783–791.

FELSENSTEIN, J. 1993. PHYLIP, ver. 3.5c. Department of Genetics, University of Washington, Seattle.

GUSTAFSSON, M. H. G. AND K. BREMER. 1995. Morphology and phylogenetic relationships of the Astreraceae, Calyceraceae, Campanulaceae, Goodeniaceae, and related families (Asterales). Amer. Jour. Bot. 82: 250–265.

HARSHMAN, J. 1994. The effect of irrelevant characters on bootstrap values. Syst. Biol. 43: 419–424.

KUMAR, S., K. TAMURA, AND M. NEI. 1993. MEGA: Molecular evolutionary genetics analysis, ver. 1.01. Pennsylvania State University, University Park.

KÄLLERSJÖ, M., J. S. FARRIS, A. G. KLUGE AND C. BULT. 1992. Skewness and permutation. Cladistics 8: 275–287.

MICKEVICH, M. F. AND J. S. FARRIS. 1981. The implications of congruence in *Menidia*. Syst. Zool. 30: 351–370.

OLMSTEAD, R. G., B. BREMER, K. M. SCOTT, AND J. D. PALMER. 1993. A parsimony analysis of the Asteridae sensu lato based on *rbc*L sequences. Ann. Missouri Bot. Gard. 80: 700–722.

PLATNICK, N. I., C. E. GRISWOLD, AND J. A. CODDINGTON. 1991. On missing entries in cladistic analysis. Cladistics 7: 337–344.

ROHLF, F. J. 1993. NTSYS-pc, version 1.8. Applied Biostatistics Inc., Setauket, New York.

RODRIGO, A. G., P. R. BERGQUIST, AND P. L. BERGQUIST. 1994. Inadequate support for an evolutionary link between the Metazoa and the Fungi. Syst. Biol. 43: 578–584.

RZHETSKY, A. AND M. NEI. 1992. A simple method for estimating and testing minimum-evolution trees. Mol. Biol. Evol. 9: 945–967.

RZHETSKY, A. AND M. NEI. 1994. METREE: a program for inferring and testing minimum-evolution trees. CABIOS 10: 409–412.

SAITOU, N. AND M. NEI. 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 6: 514–525.

SOURDIS, J. AND M. NEI. 1988. Relative efficiencies of maximum parsimony and distance-matrix methods in obtaining the correct phylogenetic tree. Mol. Biol. Evol. 5: 298–311.

STONEKING, M., S. SHERRY, AND L. VIGILANT. 1992. Geographic origin of human mitochondrial DNA revisited. Syst. Biol. 41: 384–391.

STUDIER, J. A. AND K. L. KEPLER. 1988. A note on the neighbor-joining algorithm of Saitou and Nei. Mol. Biol. Evol. 5: 729–731.

WAINWRIGHT, P., G. HINKLE, M. SOGIN, AND S. STICKEL. 1993. Monophyletic origins of the Metazoa: an evolutionary link with the Fungi. Science 260: 340–342.