A Manual Method for Character Compatibility Analysis
Author(s): Christopher A. Meacham
Source: *Taxon*, Vol. 30, No. 3 (Aug., 1981), pp. 591-600
Published by: International Association for Plant Taxonomy (IAPT)
Stable URL: http://www.jstor.org/stable/1219940
Accessed: 17/06/2014 13:15

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at
http://www.jstor.org/page/info/about/policies/terms.jsp

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms of scholarship. For more information about JSTOR, please contact support@jstor.org.

International Association for Plant Taxonomy (IAPT) is collaborating with JSTOR to digitize, preserve and extend access to *Taxon.*

http://www.jstor.org

TAXON 30(3): 591–600. AUGUST 1981

A MANUAL METHOD FOR CHARACTER COMPATIBILITY ANALYSIS

Christopher A. Meacham[1]

Summary

An exact method of performing character compatibility analysis by hand is presented.

Introduction

Character compatibility analysis is a technique that reveals patterns of agreement and disagreement among characters in a data set. It is based on facts first noticed by Wilson (1965) and Le Quesne (1969, 1972) which were subsequently provided with a mathematical foundation by the work of Estabrook and others (Estabrook 1972; Estabrook, Johnson, and McMorris 1975, 1976a, b; Estabrook and Landrum 1975; McMorris 1977; Estabrook and Meacham 1979). It has been applied to groups of animals (Estabrook, Strauch, and Fiala 1977; Strauch 1978; Nussbaum 1979) and plants (Baum 1978, Estabrook and Anderson 1979, Gardner and La Duke 1979, La Duke and Crawford 1979, Duncan 1980, Meacham 1980). These investigators used computer programs to perform their analyses, but for reasonably small data sets the necessary calculations can easily be performed by hand. The procedure described here is exact and if properly done will produce results identical with those of the computer algorithm. However, even if a computer is used, a knowledge of how to do the manipulations by hand will provide a more fundamental understanding of the technique. The description that follows assumes some familiarity with the terms and concepts of character compatibility. The outline of character compatibility analysis found in Meacham (1980) will provide a sufficient background.

The Method

Performing a character compatibility analysis can be divided into five steps: 1) Define evolutionary units (the groups of organisms under study, abbreviated *EUs*) and characters, 2) Create data matrix and hypothesize character state trees, 3) Determine compatibilities, 4) Find cliques, and 5) Draw trees. Given the group to be studied, the first step in a cladistic analysis is to define the limits of the EUs. The EUs should be given rather narrow definitions for two reasons. Doing so helps the analysis by reducing within EU variability thus making the EUs easier to characterize and also helps by making it somewhat less likely that the EUs contain organisms that are not closely related. The next step is to construct characters. A character describes a basis for comparing the EUs in the study collection with the implication that the alternative forms, called the *character states,* are features of homologous structures. For example, the character 'chromosome number' might have the states 'n = 7' and 'n = 8.' It is helpful if the character states are distinct enough to allow

[1] Division of Biological Sciences, University of Michigan, Ann Arbor, MI 48109, U.S.A. [Paper presented at the Second International Congress of Systematic and Evolutionary Biology, Vancouver, B.C., Canada, July 1980.]

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

each EU to be assigned to a character state with some degree of certainty. Characters that are continuous must be converted to qualitative characters by defining arbitrary break-points. It is usually better to choose break-points that match gaps in the distribution of EUs for that character. Construction of the data matrix consists of deciding to which character state of each character each EU belongs. Figure 1 shows a data matrix for four EUs, labeled *a* through *d*, and three characters, 1–3.

To perform a character compatibility analysis, it is necessary to determine an evolutionary ordering for the character states, that is, to hypothesize a character state tree for each character. At this stage, it is not necessary to choose an ancestral state; one need only specify the tree in an undirected fashion. Two-state (binary) characters present no difficulty because there is only one possible undirected character state tree. For three-state characters there are three possible orderings depending on which character state is placed between the other two. Deciding how the character states should be ordered can be a difficult problem if many character states are involved. This is a general problem for all cladistic techniques and cannot be effectively dealt with here. If the investigator wishes to perform a directed character compatibility analysis, it is only necessary to add a hypothetical ancestral EU to the study collection. This EU should be given the putative primitive state for each character. Upon completion of the analysis, the node that contains this EU is the root of the tree. If no ancestral EU is added, the resulting trees will be undirected. It may be possible to determine a probable location for the root after the analysis by pulling together evidence for many characters rather than trying to determine the primitive state for each character in isolation. Crisci and Stuessy (1980) and Stevens (1980) discuss criteria for determining polarity of character trends. In Fig. 1, a hypothetical ancestral EU, *x*, has been included which gives the binary directed character state trees shown in Fig. 2.

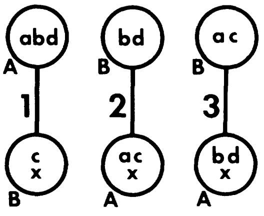
Before beginning the tests for compatibility it is necessary to break the characters that consist of more than two states into their binary factors which are two-state characters that, when put together, produce the original character state tree. This will be explained in a later section when the reader is more familiar with operations on trees. We assume that this has already been done and continue from there. The data matrix now consists of all binary characters. All pairs of characters must be tested for compatibility. If there are *n* characters in a study, there are $n(n - 1)/2$ comparisons that must be made. As Le Quesne has shown (1969), two undirected binary characters are incompatible if and only if all of the four possible combinations of character states are present in the data. To make this checking easier, a triangular matrix set up as in Fig. 3 can be used. For each pair of characters there is a set of four boxes with 'AA,' 'AB,' 'BA,' and 'BB' in them. As one scans down the columns for a pair of characters, an 'X' is placed in a box if that particular combination appears in the data. In Fig. 3, the boxes have been filled in for this small data set. 'BB' is the only combination that does not occur in the data for characters 1 and 2, nor for 2 and 3. Because not all four combinations occur in the data, these two pairs of characters are compatible. However, all four combinations do occur for characters 1 and 3, so these characters are incompatible. The testing of compatibilities is a straightforward procedure that requires only time and patience. After the matrix has been marked, the information can be transferred to a compatibility matrix (Fig. 4), which has exactly the same form as the original matrix, by placing a 'C' in it for pairs that are compatible and an 'I' for pairs that are not. Figure 15 is a blank matrix which may be photocopied for use by readers.

The next step is hunting for cliques among the characters. A clique is a set of characters that are mutually compatible. One way, which is easy if the number of characters is not large, is to place a point on a page for each character and to connect the pairs of points that correspond to pairs of compatible characters. A clique is a set of points that is completely connected, i.e., every point in the set is connected *directly* to every other point in the set. Figure 5 shows this diagram, called the *compatibility graph*, for the three characters in the data set. There are only two

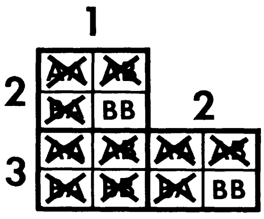
This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline a & A & A & B \\ b & A & B & A \\ c & B & A & B \\ d & A & B & A \\ x & B & A & A \end{array}$$

1



2



3

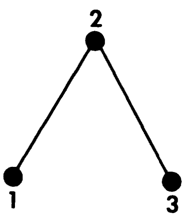
Figs. 1–3. 1: A data matrix for four extant EUs (a, b, c, and d) and a hypothetical ancestor (x), and three characters, 1–3. *2:* The character state trees from data matrix in Fig. 1. *3:* Triangular matrix marked for data matrix in Fig. 1.

cliques of two characters each; 1 and 2, and 2 and 3. This example is too simple to convey much information, so examine Fig. 8 which shows several cliques. Character 1 forms a clique by itself; 2 and 3 form a clique as do 4 and 5; 5, 6, and 7; and 6, 7, 8, and 9. Note that it is possible for a character to belong to more than one clique. This is usually the case. The compatibility graph for characters 10–14 shows a more complicated situation. The cliques here are 10, 11, 12, and 13; and 10, 11, 13, and 14. All five characters cannot belong to the same clique because 12 and 14 are incompatible.

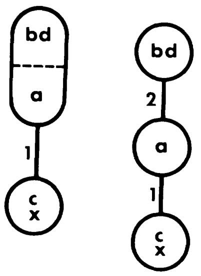
As the number of characters in the study becomes large, the graph of compatibilities for a set of characters becomes increasingly complex. There are two techniques for reducing this complexity. First, count the number of compatibilities for each character. Those characters that are compatible with every other character can be eliminated when drawing the graph because they will be included in every clique. Also when looking for the largest clique, characters with few compatibilities can be ignored. For example, a character with 8 compatibilities cannot be a member of a clique that contains 10 characters because it would have to be compatible with 9 other characters.

$$\begin{array}{c|cc} & 1 & 2 \\ \hline 2 & C & \\ 3 & I & C \end{array}$$

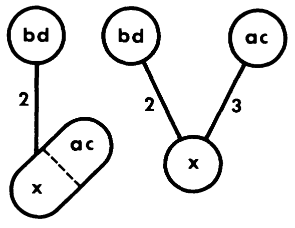
4



5



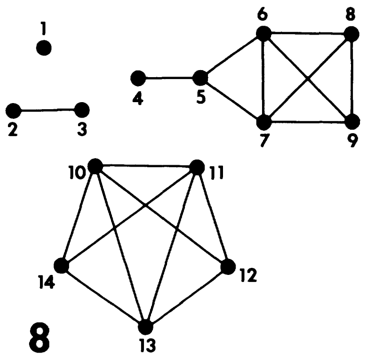
6



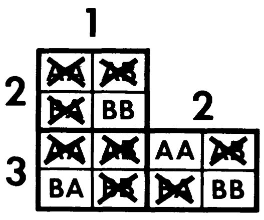
7

Figs. 4–7. 4: Compatibility matrix corresponding to Fig. 3. *5:* Compatibility graph for matrix in Fig. 4. *6:* Popping character 2 from character 1. *7:* Popping character 3 from character 2.

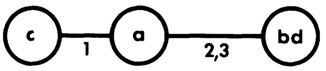
This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions



8



9



10

Figs. 8–10. *8:* Compatibility graph for fourteen characters (no relationship to Figs. 1–7). *9:* Triangular matrix marked for data matrix in Fig. 1 but excluding hypothetical ancestor. *10:* Undirected tree derived from Fig. 9.

After the largest cliques have been identified, the next step is to produce the trees that correspond to each clique. This is a rather simple stepwise procedure sometimes called 'popping' the trees. Begin with any character state tree (Fig. 2). Choose a second from the same clique. There will always be a single place on the first tree where the tree can be split so that the EUs on each side will correspond to the distribution of EUs in the character being added. Figure 6 shows the character state tree for character 1 with a dashed line through one node where a split will divide the tree into two pieces that correspond to the states of character 2. The resulting 'popped' tree shows two internodes. Each internode divides the EUs into two sets in the same way as the corresponding character state tree (compare characters 1 and 2 in Fig. 2 with the final tree in Fig. 6). Figure 7 shows the same procedure with characters 2 and 3. If a character is added to a tree that already contains an identical character, an internode will already exist that splits the tree in the proper way. Although this will add a character state transition to the tree and may thus reinforce the other character, it will not add any new information on possible phylogenetic relationships. The tree will retain the same shape. By starting with any character state tree and adding the other characters in the clique to it, one will obtain the tree the clique supports. It does not matter in what order the characters are added to the tree. One will always obtain the same result if one uses the same characters. If it should ever happen that it is not possible to break the first tree into two sections based on the character being added, then a mistake has been made. Either the characters being put together to form the tree are not really compatible or the characters added to the tree earlier were incorrectly done. Sets of compatible characters will *always* produce a tree by this straightforward technique. The final tree will have exactly as many character state transitions as there were binary characters in the original clique. (Another note: Adding a character to a tree that contains some characters that are incompatible with it always requires popping the tree at more than one point.)

If one were to perform an undirected analysis by removing the hypothetical ancestor, characters 1 and 3 would become compatible, because there would no longer be a 'BA' for these two characters (Fig. 9). Characters 2 and 3 would now be identical because they divide the EUs in the same way and their directionality is no longer specified. The resulting tree is shown in Fig. 10. Because the hypothetical ancestor is the only EU that possesses the combination 'BA,' the removal of the ancestor removes this combination from the data set. If some extant EU possessed

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

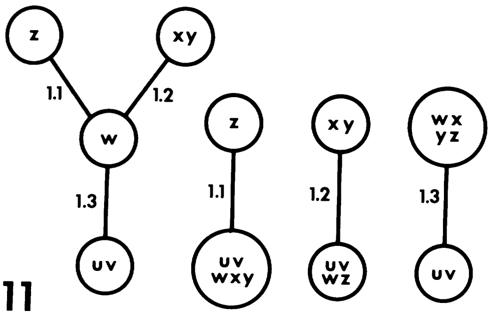


Fig. 11. A four-state character and its three binary factors.

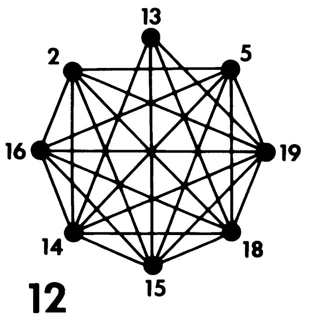
the combination 'BA,' the removal of the hypothetical ancestor would make no difference; the characters would remain incompatible. Thus, the only effect of including an ancestral EU (i.e., specifying directed characters) has on the compatibility matrix is to cause some characters that would otherwise be compatible to become incompatible. It is also clear that if the hypothetical ancestral EU is identical with any EU in the study collection, the resulting compatibility matrix will be the same as the one obtained without adding the ancestral EU.

A general procedure that will make tree popping easier can be outlined as follows. Draw all the individual compatible binary characters. If the binary characters are directed, draw them all so the ancestor is in the bottom state. If the binary characters are undirected, pick any arbitrary EU and draw the trees so that the arbitrarily chosen EU is in the bottom state. Number the binary characters in order of the number of EUs in the bottom state beginning with the tree that has the fewest number of EUs in the bottom state. Now redraw the binary character that was given number 1. Pop character number 2 from number 1. Continue popping characters in numerical order. This procedure causes most of the popping to occur at the tips of branches, but one must be careful to recognize binary characters that produce a new side branch (examine Fig. 7). As stressed before, the order in which characters are popped has no effect on the result; this technique is only recommended because it makes the operations easier to visualize at each step.

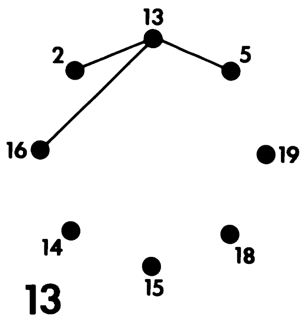
It is now easy to understand how one breaks a multistate tree into its binary factors. Each factor is a binary character that corresponds to one internode on the original character state tree. Figure 11 shows a four-state character and its three binary factors, 1.1–1.3. If one were to pop a tree from these three factors, the result would be the original character state tree. Two multistate characters are compatible if and only if all their binary factors are compatible. Practice in tree-popping can be acquired by making up a tree with six or more internodes and extracting its binary factors. The binary factors, when popped from each other, must recreate the original tree.

It is also possible to include directed and undirected characters in the same study. This is done by assigning the ancestral EU to a state for directed characters and leaving it unassigned (blank) for undirected characters. The presence of the ancestral EU may cause pairs of directed characters to be incompatible but will never cause undirected characters to be so. Only one complication is produced by using directed and undirected characters in the same study. If an undirected character causes the node that contains the ancestral EU to be split, the ancestral EU may be in either of the two resulting nodes. If another undirected character splits one of these nodes, then the ancestral EU may be in any of three nodes. For a 'semidirected' tree, the root is confined to a region of the tree but not necessarily to one node. If a directed character causes one of the nodes in the region of the ancestral EU to be split, the size of the region may be reduced again, because a directed character specifies on which side of the split the ancestral EU must occur. The use of both directed and

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions



12



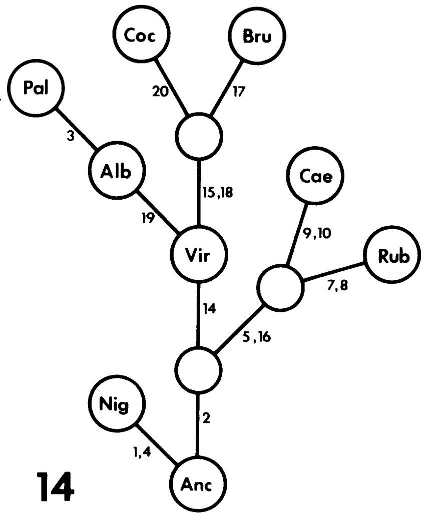
13

Figs. 12–13. *12:* Compatibility graph for characters 2, 5, 13, 14, 15, 16, 18, and 19. *13:* Incompatibility graph corresponding to Fig. 12.

undirected character state trees may solve the dilemma of the investigator who has good reasons for hypothesizing a direction for some characters but has little evidence for the direction of many others.

Another Example

A more involved example may clarify the procedure of finding the largest clique. Table 1 shows a data matrix for eight species of a hypothetical genus developed by Wagner (1980). The compatibility matrix for the 20 characters has 190 entries. Finding the largest clique by inspection would be a formidable task. However, there are several ways in which the job can be made easier. First count the number of compatibilities for each character and then sort the characters in descending order with respect to this number. Table 2 is a list ordered in this way. Note that characters 1, 3, 4, 7, 8, 9, 10, 17, and 20 are compatible with every other character and are thus members of *every* clique, the smallest as well as the largest. Temporarily ignore these characters and find the largest clique among the remaining characters. In order



14

Fig. 14. Tree supported by largest clique from Table 1.

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

Table 1. Data matrix for a hypothetical genus (Wagner, 1980).

	Characters																			
EU's	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Viridis	A	B	A	A	A	A	A	A	A	A	A	A	A	B	A	A	A	A	A	A
Alba	A	B	A	A	A	B	A	A	A	A	B	A	A	B	A	A	A	A	B	A
Pallida	A	B	B	A	A	A	A	A	A	A	A	A	A	B	A	A	A	A	B	A
Coccinea	A	B	A	A	A	A	A	A	A	A	A	B	A	B	B	A	A	B	A	B
Brunnea	A	B	A	A	A	A	A	A	A	A	B	B	A	B	B	A	B	B	A	A
Caerulea	A	B	A	A	B	A	A	A	B	B	A	B	B	A	A	B	A	A	A	A
Rubra	A	B	A	A	B	A	B	B	A	A	A	A	A	A	A	B	A	A	A	A
Nigra	B	A	A	B	A	B	A	A	A	A	A	B	B	A	A	A	A	A	A	A
Ancestor	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

for a clique of size n to exist in a data set, there must be at least n characters with at least $n - 1$ compatibilities. This fact can be used to advantage. Simply scan the list comparing the number of compatibilities for a character with its ordinal number. If the ordinal number n is more than one plus the number of compatibilities for the character, then the data set cannot contain a clique of size n because it does not include n characters with $n - 1$ compatibilities. In this list (Table 2), there *are* at least 16 characters with 15 or more compatibilities (actually there are 17), but there are *not* 17 characters with 16 or more compatibilities (there are only 16). The largest clique in this data set cannot contain more than 16 characters. This means characters 11, 6, and 12 can be ignored while looking for a clique of size 16 because these characters have fewer than 15 compatibilities each. It is only necessary to draw the compatibility graph for characters 15, 18, 5, 14, 16, 19, 2, and 13. Unfortunately, this graph (Fig. 12) is itself rather imposing. Although it is clear that all eight characters do not form a clique, it is not yet evident what the largest clique is. In this situation it is best to examine the *incompatibility graph* (Fig. 13). The incompatibility graph is made by connecting two points if their corresponding characters are incompatible. It thus contains all the possible lines that the compatibility graph does not. In mathematical terms it is called the 'graph complement' of the compatibility graph. A set of characters that forms a clique in the compatibility graph is completely unconnected in the incompatibility graph. Characters that are directly connected by lines in the incompatibility graph cannot be members of the same clique. For example if we include character 2 in a clique (Fig. 13), we cannot include character 13. Examination of Fig. 13 shows us that the largest clique will be obtained if character 13 is excluded. The clique obtained will contain characters 2, 5, 14, 15, 16, 18, and 19 plus the nine characters that are members of every clique: 1, 3, 4, 7, 8, 9, 10, 17, and 20, a total of 16 characters. Table 2 has shown that this is the largest clique that can be obtained from this data set and Fig. 13 shows that this is the only clique of this size. The tree supported by these 16 characters is illustrated in Fig. 14. The numbers indicate where the character transitions occur on the tree. The reader may wish to practice by popping this tree from the 16 characters that support it.

The problem of finding the largest clique can be the most difficult part of doing the analysis by hand. If the data set is fairly large and there are many cliques of the same size present, the task may be truly formidable. The main advantage of using

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

Table 2. Characters in Table 1 listed in descending order of number of compatibilities.

Character Number	Number of Compat's	Order	Character Number	Number of Compat's	Order
1	19	1	18	18	11
3	19	2	5	17	12
4	19	3	14	17	13
7	19	4	16	17	14
8	19	5	19	17	15
9	19	6	2	16	16
10	19	7	13	15	17
17	19	8	11	14	18
20	19	9	6	13	19
15	18	10	12	13	20

a computer program (other than a general saving of drudgery) is that the computer algorithm can list *all* the cliques in the data set and performs these calculations accurately.

Another point worth making concerns the significance of a tree diagram in the context of character compatibility. A character or a clique supports a *set* of trees. Any tree the clique supports can be obtained by further popping of the original tree. A set of characters, then, supports all the trees that are refinements of the original tree (Estabrook and McMorris, 1980). Viewed in this way, a tree, even a binary tree, is a symbol for an entire family of trees. As the number of different binary factors on the tree increases the family of trees becomes smaller until the tree is completely resolved. If there are n EUs, $2n - 2$ different binary factors will fully resolve the tree and the family of trees supported will then consist of only one tree. Knowing how to pop trees will allow the investigator to visualize the sets of trees supported by competing cliques.

Conclusion

Character compatibility analysis is more than a 'black box' into which data are dumped and from which trees emerge. It supplies a precise, mathematical framework for the problems of phylogenetic inference. Character compatibility analysis displays for the investigator the patterns of agreement and disagreement among the characters in the data set. The information provided about relationships of EUs and characters may suggest a clear-cut estimate of phylogeny or may reveal just how conflicting the data are. We can, of course, never know the true evolutionary history for any group in an absolute sense, but the logic behind character compatibility analysis can provide information about what the most reasonable alternatives are.

Acknowledgments

I thank Leslie R. Landrum and David L. Schultz for helpful conversations and William R. Anderson for his careful reading of the manuscript. This paper is a portion of a doctoral dissertation to be submitted to the Horace H. Rackham School of Graduate Studies of the University of Michigan.

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

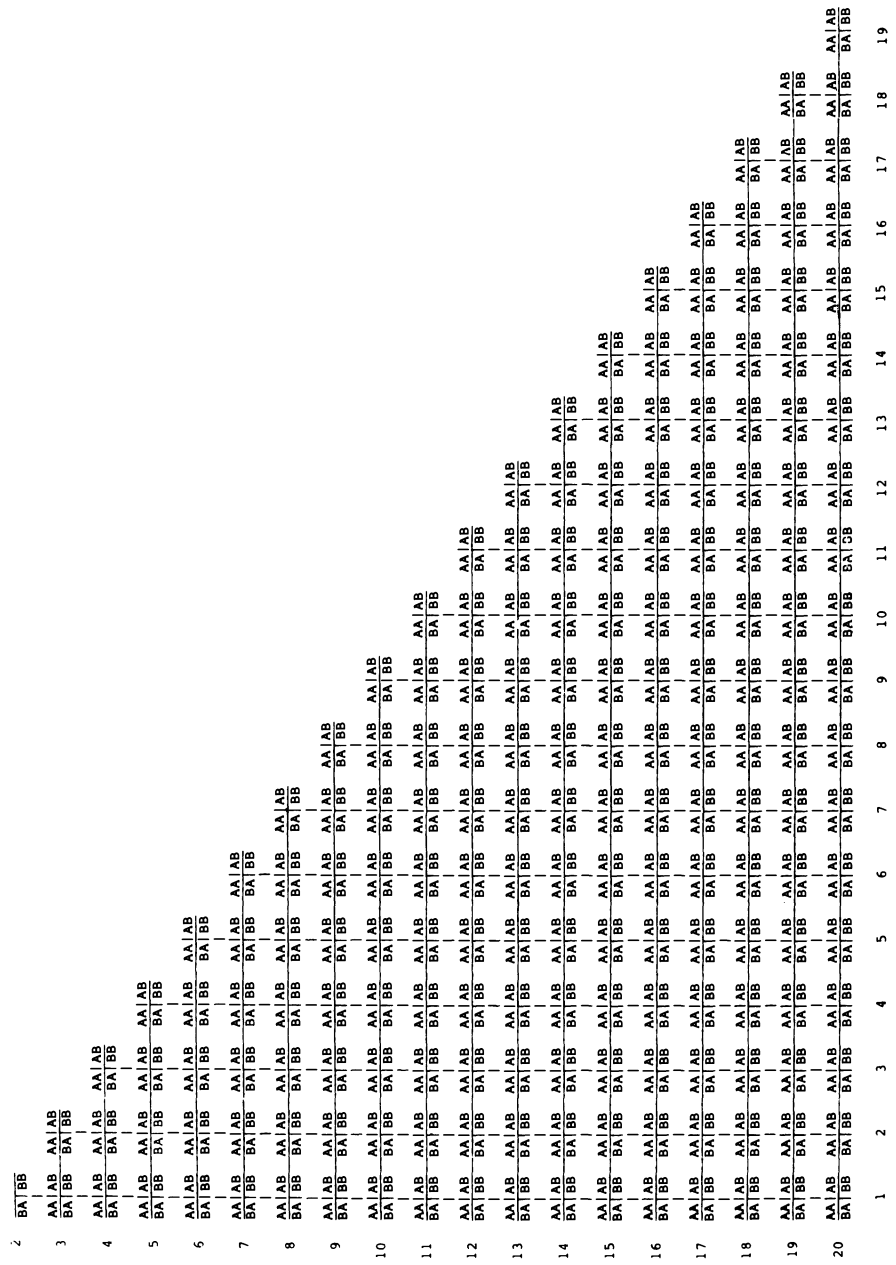


Fig. 15. Blank matrix for photocopying by reader.

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions

References

Baum, B. R. 1978. Assessment of cladograms obtained for fourteen species of *Avena* by two methods of numerical analysis. *Syst. Bot.* 2: 141–150.

Crisci, J. V. and T. F. Stuessy. 1980. Determining primitive character states for phylogenetic reconstruction. *Syst. Bot.* 5: 112–135.

Duncan, T. 1980. A cladistic analysis of the *Ranunculus hispidus* complex. *Taxon* 29: 441–454.

Estabrook, G. F. 1972. Cladistic methodology: a discussion of the theoretical basis for the induction of evolutionary history. *Ann. Rev. Ecol. Syst.* 3: 427–456.

——— and W. R. Anderson. 1979. An estimate of phylogenetic relationships within the genus *Crusea* (Rubiaceae) using character compatibility analysis. *Syst. Bot.* 3: 179–196.

———, C. S. Johnson, Jr. and F. R. McMorris. 1975. An idealized concept of the true cladistic character. *Math. Biosci.* 23: 263–272.

———, ——— and ———. 1976a. An algebraic analysis of cladistic characters. *Discrete Math.* 16: 141–147.

———, ——— and ———. 1976b. A mathematical foundation for the analysis of cladistic character compatibility. *Math. Biosci.* 29: 181–187.

——— and L. R. Landrum. 1975. A simple test for the possible simultaneous evolutionary divergence of two amino acid positions. *Taxon* 24: 609–613.

——— and F. R. McMorris. 1980. When is one estimate of evolutionary relationships a refinement of another? *J. Math. Biol.* 10: 367–373.

——— and C. A. Meacham. 1979. How to determine the compatibility of undirected character state trees. *Math. Biosci.* 46: 251–256.

———, J. G. Strauch, Jr. and K. L. Fiala. 1977. An application of compatibility analysis to the Blackiths' data on orthopteroid insects. *Syst. Zool.* 26: 269–276.

Gardner, R. C. and J. C. La Duke. 1979. Phyletic and cladistic relationships in *Lipochaeta* (Compositae). *Syst. Bot.* 3: 197–207.

La Duke, J. C. and D. J. Crawford. 1979. Character compatibility and phyletic relationships in several closely related species of *Chenopodium* of the western United States. *Taxon* 28: 307–314.

Le Quesne, W. J. 1969. A method of selection of characters in numerical taxonomy. *Syst. Zool.* 18: 201–205.

———. 1972. Further studies based on the uniquely derived character concept. *Syst. Zool.* 21: 281–288.

McMorris, F. R. 1977. On the compatibility of binary qualitative taxonomic characters. *Bull. Math. Biol.* 39: 133–138.

Meacham, C. A. 1980. Phylogeny of the Berberidaceae with an evaluation of classifications. *Syst. Bot.* 5: 149–172.

Nussbaum, R. A. 1979. The taxonomic status of the caecilian genus *Uraeotyphlus* Peters. *Occ. Pap. Mus. Zool. Univ. Michigan* 687: 1–20.

Stevens, P. F. 1980. Evolutionary polarity of character states. *Ann. Rev. Ecol. Syst.* 11: 333–358.

Strauch, J. G., Jr. 1978. The phylogeny of the Charadriiformes (Aves): a new estimate using the method of character compatibility analysis. *Trans. Zool. Soc. London* 34: 263–345.

Wagner, W. H., Jr. 1980. Origin and philosophy of the groundplan-divergence method of cladistics. *Syst. Bot.* 5: 173–193.

Wilson, E. O. 1965. A consistency test for phylogenies based on contemporaneous species. *Syst. Zool.* 14: 214–220.

This content downloaded from 195.34.79.176 on Tue, 17 Jun 2014 13:15:20 PM
All use subject to JSTOR Terms and Conditions