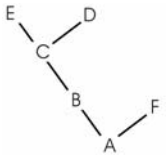
BINARY, ADDITIVE, AND NONADDITIVE CHARACTERS

IB200a exercises*

INTRODUCTION: In this lab we will compare binary, multistate nonadditive, and multistate additive character coding. We will compare their effects on tree length and on optimization of internal nodes of the trees. We will begin by learning how to code complex-character-state hierarchies as additive binary and additive multistate characters.

ADDITIVE BINARY CODING (A.K.A. HIERARCHIC BINARY CODING): Additive binary coding allows any complex hierarchy to be translated into a series of binary characters. It can be utilized to transform any additive multistate character, or any tree structure, into variables that represent the nodes or transformations in the character. Although the easiest way to implement additive binary coding is by determining a "root," it should be noted that the root is actually arbitrary and will not change the results of an analysis using the additive character.

To perform additive binary coding, first draw a character hierarchy, or character cladogram, in which each state is a terminal unit, or an internal node. You can draw this as an unrooted tree, and then root it directly to one of the character state terminals, or you can draw it in such a rooted manner from the start. For example:



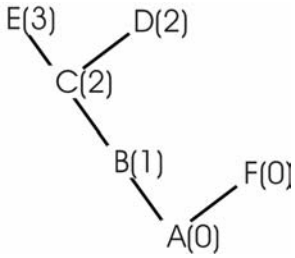
Take the character hierarchy, and make a matrix with the character states down the left side, as if they were taxa. Score each "monophyletic" group of character states with a group-membership character (Farris 1974). As you do this, work away from the root, and score each node and all of its descendants as a unique group (i.e., the descendents are all assigned a "1" and all others not in the group are assigned a "0"). Thus, each node defines a group of states above it as ones and all others as zeros. Also, code a group-membership character for each terminal character state in the tree, for which that terminal is scored "1" and all others "0" (i.e., an autapomorphic character). In the example above, this would result in the groups (B, C, D, E) and (C, D, E), as well as 3 variables for terminals D, E, and F, respectively:

```
  B
  C C
  D D
  E E D E F
A 0 0 0 0 0
B 1 0 0 0 0
C 1 1 0 0 0
D 1 1 1 0 0
E 1 1 0 1 0
F 0 0 0 0 1
```

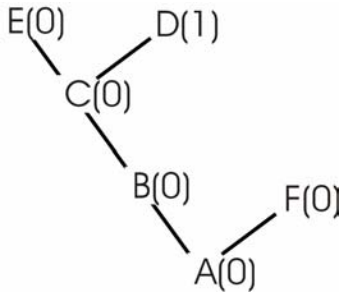
Notice that each character state is represented by a unique combination of "O"s and "1"s as you look at the rows in the binary matrix. Note that the terminal character states must be scored using autapomorphic group-membership characters in order to fully describe the character-state hierarchy.

MULTISTATE HIERARCHIC CODING (A.K.A. LINEAR NONREDUNDANT CODING): In this type of coding, the hierarchy is coded using linear multistate characters. The path from the root to each terminal is described with one multistate character.

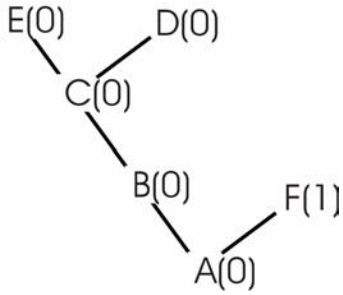
Select a terminal character state. Follow the path from the root to that terminal, and place a state change on each branch along that path that does not already have a state change. Then substitute the ordinal values for the implied state changes. Assign the highest value attained in an "ancestral" state for any nodes not on the path for that terminal (for variable E: F = 0 and D = 2, since they are not on the path to E but are attached to nodes with states 0 and 2, respectively). For example, for terminal E:



Continue this process with each terminal until all terminals are scored. For terminal D, the branches A-B and B-C were already assigned steps, so only the branch C-D has a step. Thus, the D variable has only the states 0 and 1:



And, similarly for F:



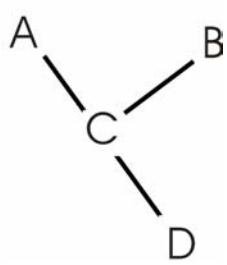
A 0 0 0
B 1 0 0
C 2 0 0
D 2 1 0
E 3 0 0
F 0 0 1

Note that resolving the terminals in a different order will result in a different coding, but all codings require the same number of steps (one step for each branch in the character-state hierarchy).

1. Given the character hierarchies for characters 1-5, create a character-state matrix for each character hierarchy using additive binary coding as described on page 1. Throughout use the specified root state.

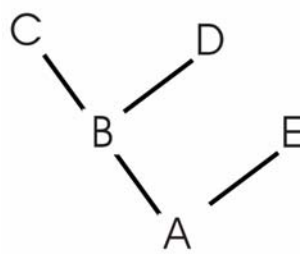
Character 1

A			
B			
C			
D			



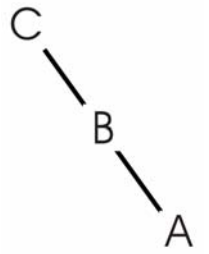
Character 2

A				
B				
C				
D				
E				



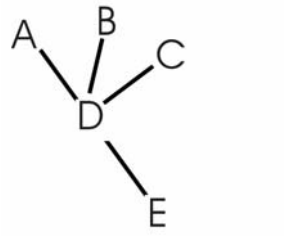
Character 3

A		
B		
C		



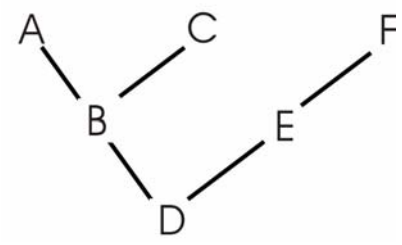
Character 4

A				
B				
C				
D				
E				



Character 5

A					
B					
C					
D					
E					
F					



2. Using the following matrix, substitute your additive binary codings for each character (1-5):

character	1	2	3	4	5
TAXON0	A	C	A	A	A
TAXONl	A	B	A	E	A
TAXON2	B	D	B	B	C
TAXON3	C	A	C	D	B
TAXON4	C	E	C	C	F
TAXONS	C	E	C	C	E
TAXON6	D	A	A	E	D

Create a new matrix for TAXON0 to TAXON6 by substituting the recoded additive binary variables for each of the characters in the matrix above. Enter your recoded matrix below. You should have 7 taxa and 18 characters.

Taxon	1	1	1	2	2	2	2	3	3	4	4	4	4	5	5	5	5	5
Taxon0																		
Taxon1																		
Taxon2																		
Taxon3																		
Taxon4																		
Taxon5																		
Taxon6																		

Input your new data matrix into a new WinClada matrix file ("New matrix (create)" from the "Matrix" menu: type in "7" when asked for number of taxa, and "18" for number of characters). After you have entered the matrix, move taxon6 to the first row of the matrix (double-click on "taxon6"; select "Move selected terms: to Beginning" from the "Terms" menu). NONA will now treat taxon6 as the outgroup. Save the matrix (select "Save As (Nona format)" from the "Matrix" menu; name the file "binary.ss").

Submit your matrix to Nona by using “Heuristics” in the “Analyze” menu. Set (hold)=1000; (Mult*N)=10; (hold/)=10; name = binary (this will name the output and tree files). Run and then view the trees for the following:

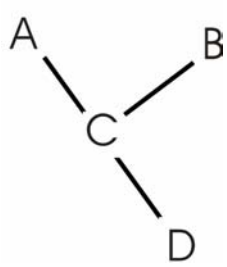
Number of trees= ____; tree length=___

Topologies:

MULTI STATE HIERARCHIC CODING (A.K.A. LINEAR NONREDUNDANT CODING): Throughout this exercise, please code the characters using the specified root.

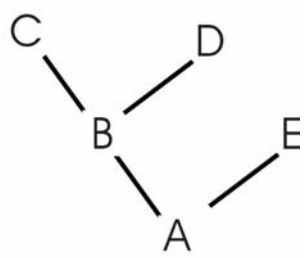
Character 1

A		
B		
C		
D		



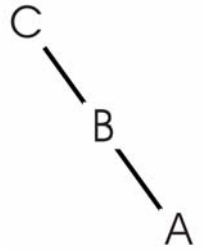
Character 2

A			
B			
C			
D			
E			



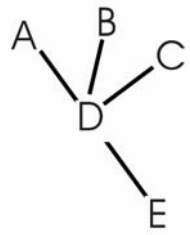
Character 3

A	
B	
C	



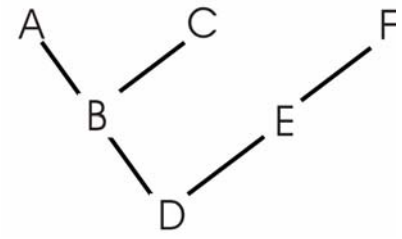
Character 4

A			
B			
C			
D			
E			



Character 5

A			
B			
C			
D			
E			
F			



4. Given the character hierarchies for characters 1-5 (the same as those on page 3), create a character-state matrix for each character hierarchy using multistate hierarchic coding, as outlined on page 2. Repeat the process of entering data and finding trees as above (page 4). Your new matrix should have 7 taxa and 12 characters. Enter the matrix below and then create a new matrix in WinClada. Make sure that your characters are all **additive**. Again, move Taxon6 to the first row. Save the matrix as "mulhier.ss" and analyze as above set name = "mulhier". View the trees and compare to trees obtained above.

Taxon	1	1	2	2	2	3	4	4	4	5	5	5
Taxon0												
Taxon1												
Taxon2												
Taxon3												
Taxon4												
Taxon5												
Taxon6												

Record the number of trees and their length: Draw the rooted trees by hand.

number of trees=___; tree length=___;

topologies:

QUESTIONS
*Do you think these are equivalent methods? What evidence do you have of this?

*Why do the characters have to be coded as additive for multistate hierarchic coding?

COMPARING ADDITIVE AND NONADDITIVE CODING:

5. Convert all characters in mulhier.ss to nonadditive (choose "Select all chars" from the "Chars" menu; choose "Make sel chars NONADDITIVE (fitch)" from the "Chars" menu). Save this matrix (NONA format) as "nonadd.ss".

6. Open the tree file "mulhier.tre" ("Open Tree file" from the "File" menu). Record the length of each of the most parsimonious trees from mulhier.ss when the characters are coded as nonadditive (use "Next tree" from the "Trees" menu to move through the trees). Compare to the lengths recorded from part 4.

Tree 1 length=____ ;Tree 2 length=____;Tree 3 length=____

7. To understand what happens when multistate characters are switched from additive to nonadditive, look at the tree from the mulhier.tre file with the topology below (page 8) and fill out the table for data matrix "nonadd.ss" optimized on this tree using "unambiguous" optimization (select "Diagnoser TOGGLE" and "Show All states at node" from the "Diagnoser" menu).Then optimize the data matrix "mulhier.ss" on this tree and fill out the second table.

QUESTIONS

*Why is the tree in part 7 shorter when the characters are nonadditive than when the characters are additive?

*What difference do you observe when you treat a binary character as additive relative to treating it as a nonadditive?

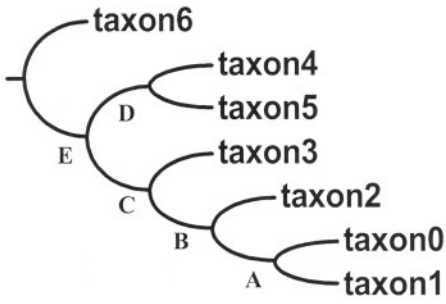
*What is the relationship between character additivity and the number of equally optimal states optimized at an internal node? Why?

*Can you make any generalizations about the length of the same character coded as additive or nonadditive?

LITERATURE CITED

Farris, J. S. 1974. Formal definitions of paraphyly and polyphyly. Systematic Zoology 23: 548-554.

*Handout based on original lab handout provided to K.Will by K. Nixon



Data matrix nonadd.ss optimized on this tree.

Char	#states	length	all possible states at node, i.e. "unambiguous"optimization				
			A	B	C	D	E
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

Data matrix mulhier.ss optimized on this tree.

Char	#states	length	all possible states at node, i.e. "unambiguous"optimization				
			A	B	C	D	E
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							