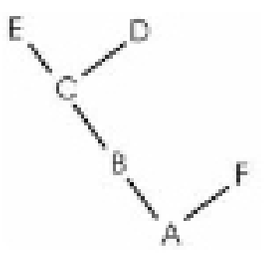
BINARY, ADDITIVE, AND NONADDITIVE CHARACTERS

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:



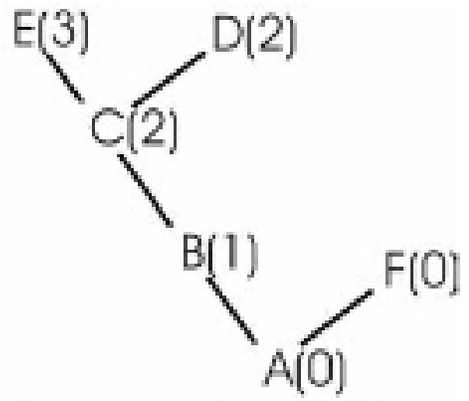
Take the character hierarchy, and make a matrix with the character states down the left side, as if they were taxa. Score each "monophyletic" group of character states with a group-membership character (Farris 1974). As you do this, work away from the root, and score each node and all of its descendants 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
```

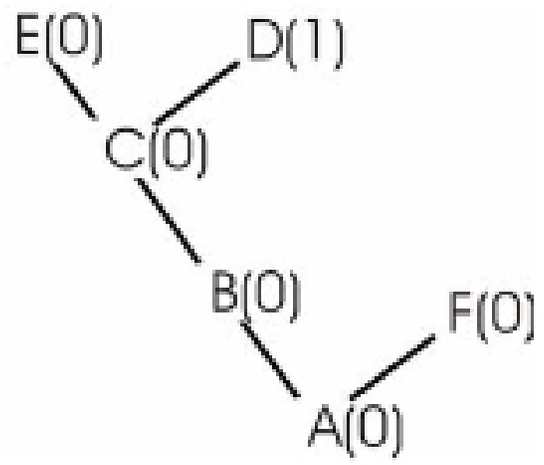
Notice that each character state is represented by a unique combination of "O"s and "1"s as you look at the rows in the binary matrix. Note that the terminal character states must be scored using autapomorphic group-membership characters in order to fully describe the character-state hierarchy.

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.

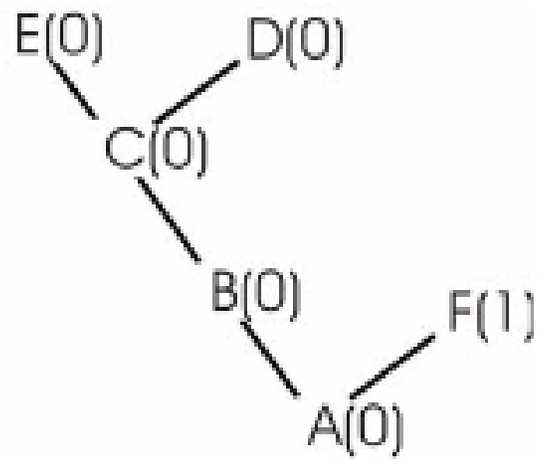
Select a terminal character state. Follow the path from the root to that terminal, and place a state change on each branch along that path that does not already have a state change. Then substitute the ordinal values for the implied state changes. Assign the highest value attained in an "ancestral" state 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:



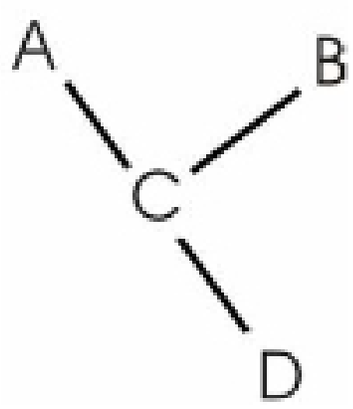
A 0 0 0
B 1 0 0
C 2 0 0
D 2 1 0
E 3 0 0
F 0 0 1

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.

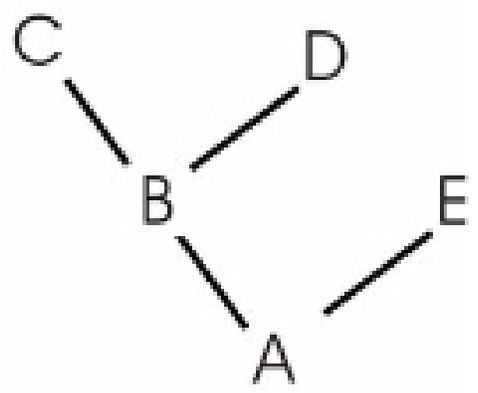
Character 1

A			
B			
C			
D			



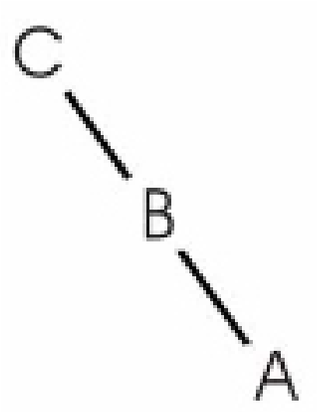
Character 2

A				
B				
C				
D				
E				



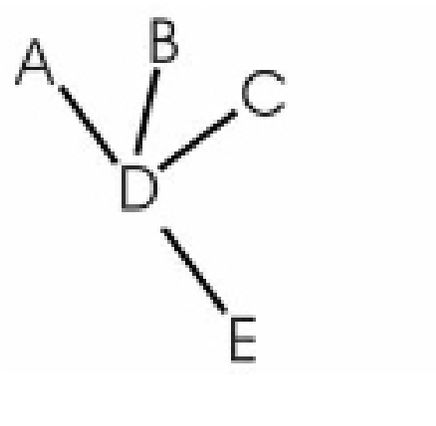
Character 3

A		
B		
C		



Character 4

A				
B				
C				
D				
E				



Character 5

A					
B					
C					
D					
E					
F