```
%copyright Charles Davi 2022
%BLACK TREE AUTOML
%HUMANITY'S FASTEST DEEP LEARNING SOFTWARE

%=========================================================================
%=========================================================================
%VECTORIZED COMPUTATIONAL GENOMICS - mtDNA Dataset Command Line
%=========================================================================
%=========================================================================

%=========================================================================
%INPUT VARIABLES
%=========================================================================

X = 6; %root class
Y = 75; %comparison class 1
Z = 30; %comparison class 2

best_match = 0; %default value for maximum matching percentage

clc

temp_vector = [X Y Z];

%uses simple modulo to iterate over all combinations
for K = 0 : 2

%cycles through each combination using modular arithmetic
tempA = mod(K,3) + 1;
tempB = mod(K + 1,3) + 1;
tempC = mod(K + 2,3) + 1;

A = temp_vector(tempA);
B = temp_vector(tempB);
C = temp_vector(tempC);

x = find(dataset(:,N+1) == A); %finds all rows in root class
y = find(dataset(:,N+1) == B); %finds all rows in comparison class 1
z = find(dataset(:,N+1) == C); %finds all rows in comparison class 2
```

```
40
41  %the number of genomes per class
42  a = size(x,1);
43  b = size(y,1);
44  c = size(z,1);
45
46  match_count = 0; %the number of successful ancestry tests
47
48  counter = 0;
49
50  for i = 1 : a
51
52    root = dataset(x(i),:);
53
54    for j = 1 : b
55
56      top = dataset(y(j),:);
57
58      for k = 1 : c
59
60        bot = dataset(z(k),:);
61
62        L1 = sum(root(1:N) == top(1:N)); %root match to top
63        L2 = sum(root(1:N) == bot(1:N)); %root match to bot
64        R = sum(top(1:N) == bot(1:N));   %top match to bot
65
66        counter = counter + 1;
67
68        %if true, the root is the ancestor of the top and bot
69        if(L1 > R && L2 > R)
70
71          match_count = match_count + 1;
72
73        endif
74
75      endfor
76
77    endfor
78
79  endfor
80
81    ["Total number of matches assuming " Names{A} " is the common
      ancestor of " Names{B} " and " Names{C}]
82
83  match_count
84
85    ["Percentage of matches assuming " Names{A} " is the common
```

```
    ancestor of " Names{B} " and " Names{C}]
86
87  match_count/counter
88
89  %stores the best root population
90  if(match_count/counter > best_match)
91
92    best_match = match_count/counter;
93    best_root = A;
94
95  endif
96
97  endfor %end of outer loop
98
99  ["The best root population among " Names{A} ", " Names{B} ", and
    " Names{C} " is " Names{best_root}]
```