

```

1  %copyright Charles Davi 2022
2  %BLACK TREE AUTOML
3  %HUMANITY'S FASTEST DEEP LEARNING SOFTWARE
4
5  %=====
6  %=====
7  %VECTORIZED COMPUTATIONAL GENOMICS - mtDNA Dataset Command Line
8  %=====
9  %=====
10
11 %=====
12 %INPUT VARIABLES
13 %=====
14
15 Q = 7; %root class
16 W = 11; %comparison class 1
17 E = 21; %comparison class 2
18
19 best_match = 0; %default value for maximum matching percentage
20
21 clc
22
23 temp_vector = [Q W E];
24
25 %uses simple modulo to iterate over all combinations
26 for K = 0 : 2
27
28 %cycles through each combination using modular arithmetic
29 tempA = mod(K,3) + 1;
30 tempB = mod(K + 1,3) + 1;
31 tempC = mod(K + 2,3) + 1;
32
33 A = temp_vector(tempA);
34 B = temp_vector(tempB);
35 C = temp_vector(tempC);
36
37 x = find(dataset(:,N+1) == A); %finds all rows in root class
38 y = find(dataset(:,N+1) == B); %finds all rows in comparison
   class 1
39 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 percentage = match_count/counter;
88
89 100*percentage
90
91 %stores the best root population
92 if(percentge > best_match)
93
94     best_match = match_count/counter;
95     best_root = A;
96
97 endif
98
99 flow(A) = percentage;
100
101 endfor %end of outer loop
102
103 ["The best root population among " Names{Q} ", " Names{W} ", and ↵
    " Names{E} " is " Names{best_root}]
104
105 ["The net flow from " Names{Q} " to " Names{W} " is "]
106
107 100*(flow(Q) - flow(W))
108
109 ["The net flow from " Names{Q} " to " Names{E} " is "]
110
111 100*(flow(Q) - flow(E))
112
113 ["The net flow from " Names{W} " to " Names{E} " is "]
114
115 100*(flow(W) - flow(E))
116
117
118

```