sim-mad

Weigang Qiu

5/21/2021

## Part 1. Construct MAD populations with and without crossing-over (recombination)

Generate maximally divergent 20-bits long binary sequences using genetic algorithms:

library(GA)

## Loading required package: foreach

## Loading required package: iterators

## Package 'GA' version 3.2.1  
## Type 'citation("GA")' for citing this R package in publications.

##   
## Attaching package: 'GA'

## The following object is masked from 'package:utils':  
##   
## de

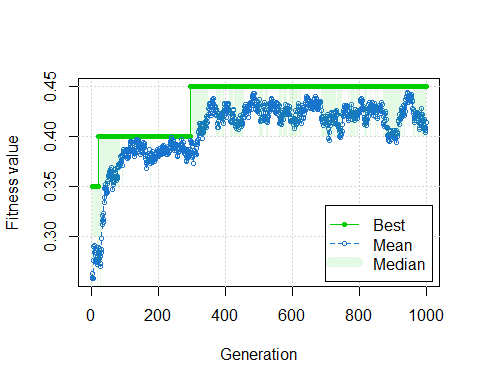
library(stringdist)  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

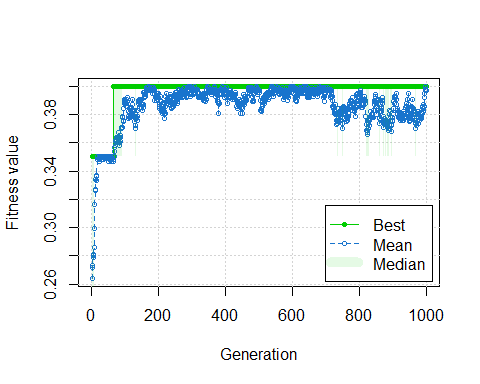
## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x purrr::accumulate() masks foreach::accumulate()  
## x tidyr::extract() masks stringdist::extract()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x purrr::when() masks foreach::when()

library(ape)  
  
len.seq <- 20 # length of sequences  
num.alle <- 10 # number of alleles  
size.pair <- num.alle \* (num.alle-1)/2  
  
# Fitness function, to maximize the minimal Hamming distance:  
epi.fit <- function(x) {   
 x.list <- split(x, rep(1:num.alle, each=len.seq))  
 min(seq\_distmatrix(x.list, method = "hamming"))/len.seq  
}  
  
# Run genetic algorithm with default parameters:  
mad.ga <- ga(type = "binary", fitness = epi.fit, nBits = len.seq \* num.alle, maxiter = 1000)  
  
plot(mad.ga)



# Run genetic algorithm without recombination:  
mad.clonal <- ga(type = "binary", fitness = epi.fit, nBits = len.seq \* num.alle, maxiter = 1000, pcrossover = 0)  
plot(mad.clonal)

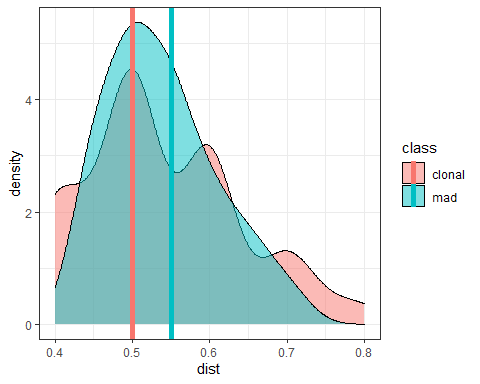


top.pop = mad.ga@solution[1,]  
top.clonal = mad.clonal@solution[1,]  
  
# Split into 20-bit strings & get consensus  
x.list <- split(top.pop, rep(1:num.alle, each=len.seq))  
mad.alleles <- t(as.data.frame(x.list))  
  
x.clonal <- split(top.clonal, rep(1:num.alle, each=len.seq))  
clonal.alleles <- t(as.data.frame(x.clonal))  
  
# Obtain majority-rule consensus sequences  
consense <- apply(mad.alleles,2, function(x) {table(x)} %>% which.max()-1)

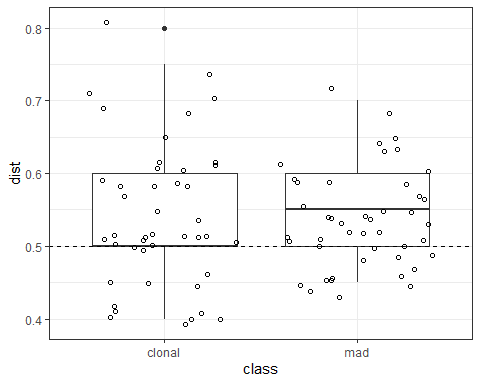
## Part 2. Compare seq separation

Compare among variants with crossing-over, without crossing-over, and randomly permuted (with crossing-over). Recombination creates more optimal separation (higher median with low variance).

dist.optimal <- seq\_distmatrix(x.list, method = "hamming")/len.seq  
dist.clonal <- seq\_distmatrix(x.clonal, method = "hamming")/len.seq  
  
mad.shuffled <- apply(mad.alleles, 2, sample) # permute by column  
xy.list <- split(mad.shuffled, seq(nrow(mad.shuffled)))  
dist.shuffled <- seq\_distmatrix(xy.list, method = "hamming")/len.seq  
  
three.dist <- tibble(dist = c(dist.optimal, dist.shuffled, dist.clonal), class = c(rep("mad", size.pair), rep("permuted",size.pair), c(rep("clonal", size.pair))))  
  
two.dist <- three.dist %>% filter(class != "permuted")  
  
med <- two.dist %>% group\_by(class) %>% summarise(med = median(dist))  
  
two.dist %>% ggplot(aes(x=dist, fill = class)) + geom\_density(alpha=0.5) + theme\_bw() + geom\_vline(data=med, aes(xintercept = med, color=class), size=2)



two.dist %>% ggplot(aes(x = class, y = dist)) + geom\_boxplot() + geom\_jitter(shape=1) + theme\_bw() + geom\_hline(yintercept = 0.5, linetype = 2)



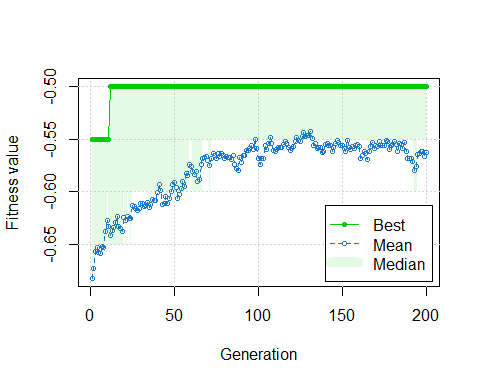
t.test(data = two.dist, dist ~ class)

##   
## Welch Two Sample t-test  
##   
## data: dist by class  
## t = 0.060392, df = 78.097, p-value = 0.952  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -0.03551630 0.03773852  
## sample estimates:  
## mean in group clonal mean in group mad   
## 0.5388889 0.5377778

## Part 3. Obtain centroids

Find broadly cross-reactive centroids, again using genetic algorithm:

# Fitness function, minimize max distance  
center.fit <- function(x) {  
 ref.alleles <- split(top.pop, rep(1:num.alle, each=len.seq));  
 -1 \* max(seq\_dist(x, ref.alleles, method = "hamming")/len.seq);  
}  
center.ga <- ga(type = "binary", fitness = center.fit, nBits = len.seq, popSize = 100, maxiter = 200);  
  
plot(center.ga)



centroid <- center.ga@solution[1,]

## Part 4. Validation with tree

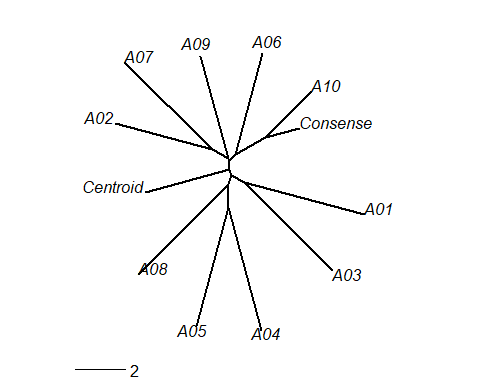
x.list[[11]] <- consense  
x.list[[12]] <- centroid  
  
names(x.list) <- c("A01", "A02", "A03", "A04", "A05", "A06", "A07", "A08", "A09", "A10", "Consense", "Centroid")  
  
# distance matrix  
x.dist <- seq\_distmatrix(x.list, method = "hamming")  
x.dist

## A01 A02 A03 A04 A05 A06 A07 A08 A09 A10 Consense  
## A02 11   
## A03 10 11   
## A04 11 10 11   
## A05 13 12 13 10   
## A06 11 10 11 12 10   
## A07 14 9 12 13 11 9   
## A08 11 10 11 10 12 12 11   
## A09 10 9 12 13 9 9 10 9   
## A10 9 10 9 14 10 10 11 10 9   
## Consense 9 6 9 12 12 6 11 10 7 4   
## Centroid 9 10 9 10 10 8 9 10 9 8 6

# show varints  
t(as.data.frame(x.list))

## x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20  
## A01 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0  
## A02 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1  
## A03 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0  
## A04 1 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1  
## A05 1 0 0 1 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 0  
## A06 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 0 1 0 0 0  
## A07 0 1 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1  
## A08 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 1 0 1 1  
## A09 0 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1  
## A10 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1  
## Consense 0 1 0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 1  
## Centroid 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 0 1

# tree  
plot(bionj(x.dist), type = "u", no.margin = T, edge.width = 2)  
add.scale.bar()



## Part 5. Show session info

sessionInfo()

## R version 4.0.5 (2021-03-31)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 10 x64 (build 19041)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=English\_United States.1252   
## [2] LC\_CTYPE=English\_United States.1252   
## [3] LC\_MONETARY=English\_United States.1252  
## [4] LC\_NUMERIC=C   
## [5] LC\_TIME=English\_United States.1252   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] ape\_5.4-1 forcats\_0.5.1 stringr\_1.4.0 dplyr\_1.0.5   
## [5] purrr\_0.3.4 readr\_1.4.0 tidyr\_1.1.3 tibble\_3.1.0   
## [9] ggplot2\_3.3.3 tidyverse\_1.3.1 stringdist\_0.9.6.3 GA\_3.2.1   
## [13] iterators\_1.0.13 foreach\_1.5.1   
##   
## loaded via a namespace (and not attached):  
## [1] Rcpp\_1.0.6 lubridate\_1.7.10 lattice\_0.20-41 assertthat\_0.2.1   
## [5] digest\_0.6.27 utf8\_1.2.1 R6\_2.5.0 cellranger\_1.1.0   
## [9] backports\_1.2.1 reprex\_2.0.0 evaluate\_0.14 highr\_0.8   
## [13] httr\_1.4.2 pillar\_1.6.0 rlang\_0.4.10 readxl\_1.3.1   
## [17] rstudioapi\_0.13 rmarkdown\_2.7 labeling\_0.4.2 munsell\_0.5.0   
## [21] broom\_0.7.6 compiler\_4.0.5 modelr\_0.1.8 xfun\_0.22   
## [25] pkgconfig\_2.0.3 htmltools\_0.5.1.1 tidyselect\_1.1.0 codetools\_0.2-18   
## [29] fansi\_0.4.2 crayon\_1.4.1 dbplyr\_2.1.1 withr\_2.4.1   
## [33] grid\_4.0.5 nlme\_3.1-152 jsonlite\_1.7.2 gtable\_0.3.0   
## [37] lifecycle\_1.0.0 DBI\_1.1.1 magrittr\_2.0.1 scales\_1.1.1   
## [41] cli\_2.4.0 stringi\_1.5.3 farver\_2.1.0 fs\_1.5.0   
## [45] xml2\_1.3.2 ellipsis\_0.3.1 generics\_0.1.0 vctrs\_0.3.7   
## [49] tools\_4.0.5 glue\_1.4.2 hms\_1.0.0 parallel\_4.0.5   
## [53] yaml\_2.2.1 colorspace\_2.0-0 rvest\_1.0.0 knitr\_1.32   
## [57] haven\_2.3.1