Evolutionary trajectories in the Long-term Evolution Experiment

William R. Shoemaker, Jay T. Lennon

22 March, 2018

Background

One of the goals of our research is to determine whether independently evolving populations are evolving under similar trajectories. One of the more recent and thorough attempts at addressing this goal is a recent study that examined fine-scale temporal sampling of Richard Lenski's Long-term Evolution Experiment (Good et al., 2017). In Good et al. (2017) the authors propose a measure of gene multiplicity to detect evolutionary parallelism at the gene level among replicate populations. The multiplicity for each gene is

$$m_i = n_i \cdot \frac{\overline{L}}{L_i}$$

where n_i is the number of mutations in gene i across all replicate populations, L_i is the number of non-synonymous sites in gene i, and $\overline{L}$ is the average value of L_i across all genes in the genome. Under the null model that the probability that a gene contains a mutation is simply proportional to the length of the gene ($p_i \propto L_i$), all genes have the same expected multiplicity $\overline{m} = n_{tot}/n_{genes}$, where n_{tot} is the number of mutations among all replicate populations and n_{genes} is the number of genes in the genome.

In Good et al. (2017) the authors determine that in nonmutator LTEE populations approximately half of all mutations occurred in genes with $m_i \geq 2$, twice as many as expected under the null model. The authors concluded that the null model should be replaced with an alternative where mutations are assigned to each gene with probability

$$p_i \propto L_i r_i$$

where r_i is an enrichment factor that is not equal to 1. Under the alternative model the maximum likelihood estimator for the enrichment factor is the ratio of observed and expected multiplicities, $r_i = m_i/\overline{m}$ and the net increase relative to the null model across all genes is

$$\Delta\ell = \sum_i n_i \log\left(\frac{m_i}{\overline{m}}\right)$$

The authors note that the maximum likelihood estimate r_i may overfit the data and propose that a more appropriate alternative model would be one that focuses on a subset I of the genes where $r_i \neq 1$, while the remaining genes have $r_i = 1$. The authors identify this set of genes using a cricial P-value, P^*, for a the False Discovery Rate $\alpha = 0.05$ and modify the enrichment factors as follows

$$r_i = \begin{cases} \frac{m_i}{\overline{m}} \left(\frac{1 - \frac{\sum_{i \epsilon I} {}^{L_i}}{\overline{L} n_{genes}}}{1 - \frac{\sum_{i \epsilon I} {}^{n_i}}{n_{tot}}} \right) & \text{if } i \epsilon I \\ 1 & \text{else.} \end{cases}$$

This is an innovative approach that builds off of statistical distributions used to describe parallel evolutionary outcomes. However, this measure pools the mutation data for all replicate populations for each gene. To allow for the comparison between replicate populations so that we can begin to develop statistics to determine

whether replicate populations have similar evolutionary trajectories from pooled sequencing, the multiplicity statistics presented in Good et al. (2017) need to be deconstructed to the level of individual populations. To accomplish this goal, we propose a multiplicity measure for the ith gene in the jth population

$$m_{i,j} = n_{i,j} \cdot \frac{\overline{L}}{L_i}$$

with the expected multiplicity in population j of $\overline{m_j} = n_{tot,j}/n_{genes}$, giving a log-likelihood compared to the null model (which is now $r_{i,j} = m_{i,j}/\overline{m_j}$)

$$\Delta\ell_j = \sum_i n_{i,j} \log\left(\frac{m_{i,j}}{\overline{m_j}}\right)$$

and the modified enrichment factor

$$r_{i,j} = \begin{cases} \frac{m_{i,j}}{\overline{m_j}} \left(\frac{1 - \frac{\sum_{i \epsilon I} L_i}{\overline{L} n_{genes}}}{1 - \frac{\sum_{i \epsilon I} n_{i,j}}{n_{tot,j}}} \right) & \text{if } i \epsilon I \\ 1 & \text{else.} \end{cases}$$

Using these modified population level gene enrichment scores and the publically available data presented in Good et al. (2017), we calculate the multiplicity score for each gene within each population for all genes within set I, generating a gene-by-population multiplicity matrix. We then used Principal Coordinates Analysis (PCoA) to reduce the dimensionality of the dataset and visualize the evolutionary trajectories of the six nonmutator populations. To determine whether or not the rate that mutations are acquired in each gene decreases, we calculated the Euclidean distance of the first three PCoA axes between timepoints for each population.

What else? Betadisper? Other appropriate (and not too difficult) analyses? I'd like to get feedback on this approach before carrying out much more work

Setup work environment

```
rm(list=ls())
getwd()
#setwd("~/GitHub/ParEvol")
knitr::opts_knit$set(root.dir = '~/GitHub/ParEvol/')

# Load dependencies
# Load dependencies
require("vegan")
require("png")
require("grid")
library("scales")
library("data.table")
```

Load and clean data

```
pop_by_gene <- c("data/ltee/gene_by_pop_m_I.txt")
df <- read.table(paste(pop_by_gene, collapse = ''), sep = "\t",
```

```
                header = TRUE, row.names = 1)
# only look at nonmutators (for now)
complete_nonmutator_lines <- c('m5','m6','p1','p2','p4','p5')
complete_mutator_lines <- c('m1','m4','p3')
to_keep <- rownames(df) %like% "m5" + rownames(df) %like% "m6" +
  rownames(df) %like% "p1" + rownames(df) %like% "p2" +
  rownames(df) %like% "p4" + rownames(df) %like% "p5"
df.noMut <- df[as.logical(to_keep),]
df.no0 <- df.noMut[apply(df.noMut[,-1], 1, function(x) !all(x==0)),]
```

Make and visualize ordination

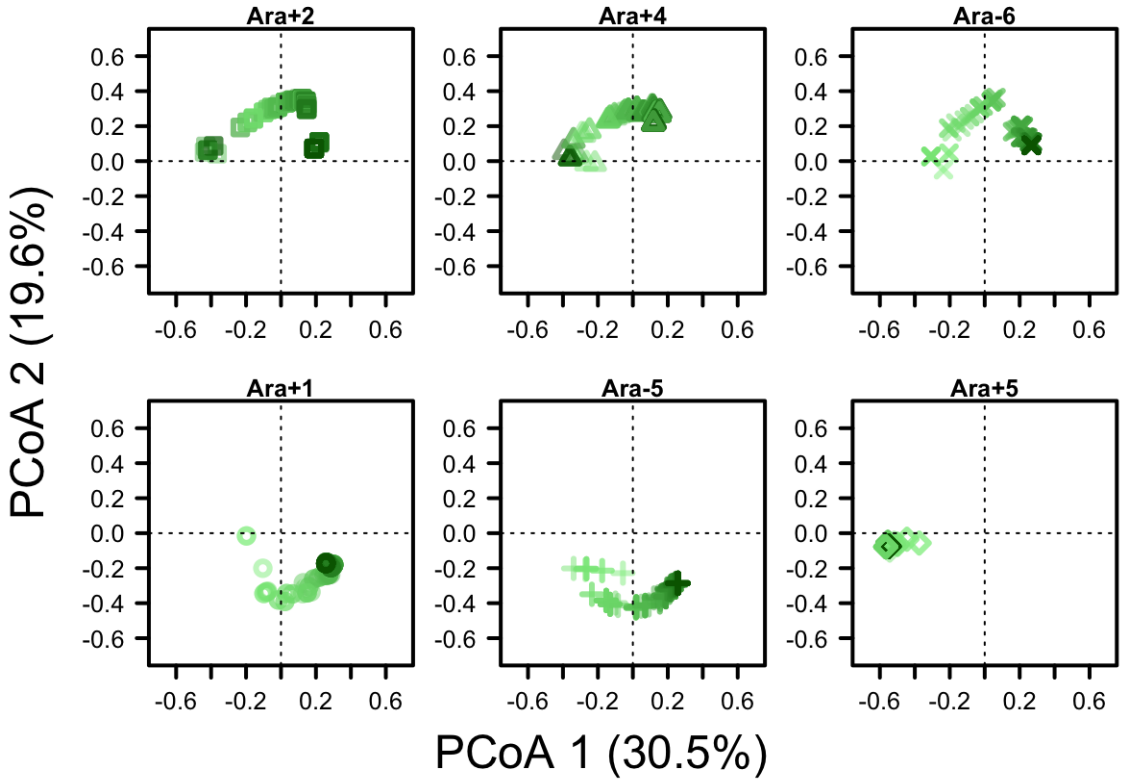
```
df.no0.db <- vegdist(df.no0, method = "bray", upper = TRUE, diag = TRUE)
df.pcoa <- cmdscale(df.no0.db, eig = TRUE, k = 3)
explainvar1 <- round(df.pcoa$eig[1] / sum(df.pcoa$eig), 3) * 100
explainvar2 <- round(df.pcoa$eig[2] / sum(df.pcoa$eig), 3) * 100

times <- c()
for (x in rownames(df.pcoa$points)){
  time <- strsplit(x, '_')[[1]][2]
  times <- c(times, time)
}
# function to return color gradient for time points
get.times.cols <- function(pcoa.points, times){
  # get colors for times
  times.sorted <- as.character(sort(as.numeric(unique(times))))
  number.times <- length(times.sorted)
  colfunc.nonMut <- colorRampPalette(c("lightgreen", "darkgreen"))
  time.cols.nonMut <- colfunc.nonMut(number.times)
  times.cols.pcoa <- c()
  for (x in rownames(pcoa.points)){
    time <- strsplit(x, '_')[[1]][2]
    pop <- toString(strsplit(x, '_')[[1]][1])
    time.position <- match(time, times.sorted)
    if (pop %in% complete_nonmutator_lines){
      time.color <- time.cols.nonMut[time.position]
    }
    times.cols.pcoa <- c(times.cols.pcoa, time.color)
  }
  return(times.cols.pcoa)
}

pcoa.m5 <- df.pcoa$points[rownames(df.pcoa$points) %like% "m5", ]
pcoa.m6 <- df.pcoa$points[rownames(df.pcoa$points) %like% "m6", ]
pcoa.p1 <- df.pcoa$points[rownames(df.pcoa$points) %like% "p1", ]
pcoa.p2 <- df.pcoa$points[rownames(df.pcoa$points) %like% "p2", ]
pcoa.p4 <- df.pcoa$points[rownames(df.pcoa$points) %like% "p4", ]
pcoa.p5 <- df.pcoa$points[rownames(df.pcoa$points) %like% "p5", ]

m5.cols <- get.times.cols(pcoa.m5, times)
m6.cols <- get.times.cols(pcoa.m6, times)
p1.cols <- get.times.cols(pcoa.p1, times)
```

```r
p2.cols <- get.times.cols(pcoa.p2, times)
p4.cols <- get.times.cols(pcoa.p4, times)
p5.cols <- get.times.cols(pcoa.p5, times)

# make plot
png(filename = paste(c("figs/pcoa.png"), collapse = ''),
    width = 1200, height = 900, res = 96*2)
#layout(matrix(1:6, 2, 3))
#par(mar = c(0.5, 2.5, 0.5, 0.5), oma = c(4, 1, 1, 1))
#par(mar =c(0.5, 3.1, 1.1, 0.5), oma = c(2, 1, 0, 0 ), pty="s")#, mai = c(0.3, 0.2, 0.2, 0.2))
#par(mfrow = c(2,3), mar =c(0, 3.1, 1.1, 0.5), oma = c(5, 4, 0, 0 ), pty="s")
par(mfrow = c(2, 3),
    oma = c(5, 4, 0, 0), # two rows of text at the outer left and bottom margin
    mar = c(1.5, 3.1, 1.1, 0.5), # space for one row of text at ticks and to separate plots
    pty="s") # make the plots square

# plots for each population
pcoa.p2.plot <- plot(pcoa.p2[ ,1], pcoa.p2[ ,2],
                     xlim = c(-0.7, 0.7), ylim = c(-0.7, 0.7),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara+2")
points(pcoa.p2[ ,1], pcoa.p2[ ,2],
       pch = 0, cex = 1.5, bg = "gray", col = alpha(m6.cols, 0.5), lwd  = 3)
# Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

pcoa.p4.plot <- plot(pcoa.p4[ ,1], pcoa.p4[ ,2],
                     xlim = c(-0.7, 0.7), ylim = c(-0.7, 0.7),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara+4")
points(pcoa.p4[ ,1], pcoa.p4[ ,2],
       pch = 2, cex = 1.5, bg = "gray", col = alpha(m6.cols, 0.5), lwd  = 3)
# Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

pcoa.m6.plot <- plot(pcoa.m6[ ,1], pcoa.m6[ ,2],
                     xlim = c(-0.7, 0.7), ylim = c(-0.7, 0.7),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara-6")
points(pcoa.m6[ ,1], pcoa.m6[ ,2],
       pch = 4, cex = 1.5, bg = "gray", col = alpha(m6.cols, 0.5), lwd  = 3)
# Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)
```

```
pcoa.p1.plot <- plot(pcoa.p1[ ,1], pcoa.p1[ ,2],
                    xlim = c(-0.7, 0.7), ylim = c(-0.7, 0.7),
                    pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                    axes = FALSE, xlab = '', ylab ='', main = "Ara+1")
points(pcoa.p1[ ,1], pcoa.p1[ ,2],
       pch = 1, cex = 1.5, bg = "gray", col = alpha(m6.cols, 0.5), lwd  = 3)
# Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

pcoa.m5.plot <- plot(pcoa.m5[ ,1], pcoa.m5[ ,2],
                    xlim = c(-0.7, 0.7), ylim = c(-0.7, 0.7),
                    pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                    axes = FALSE, xlab = '', ylab ='', main = "Ara-5")
points(pcoa.m5[ ,1], pcoa.m5[ ,2],
       pch = 3, cex = 1.5, bg = "gray", col = alpha(m5.cols, 0.5), lwd  = 3)
# Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

pcoa.p5.plot <- plot(pcoa.p5[ ,1], pcoa.p5[ ,2],
                    xlim = c(-0.7, 0.7), ylim = c(-0.7, 0.7),
                    pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                    axes = FALSE, xlab = '', ylab ='', main = "Ara+5")
points(pcoa.p5[ ,1], pcoa.p5[ ,2],
       pch = 5, cex = 1.5, bg = "gray", col = alpha(m6.cols, 0.5), lwd  = 3)
# Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

mtext(paste("PCoA 1 (", explainvar1, "%)", sep = ""), side=1, line=1,
      cex=1.5, col="black", outer=TRUE)
mtext(paste("PCoA 2 (", explainvar2, "%)", sep = ""), side=2, line=1,
      cex=1.5, col="black", outer=TRUE)

dev.off()
```

```
## pdf
##   2
```

```
# Show Plot
img <- readPNG(paste(c("figs/pcoa.png"), collapse = ''))
grid.raster(img)
```



Euclidean distance

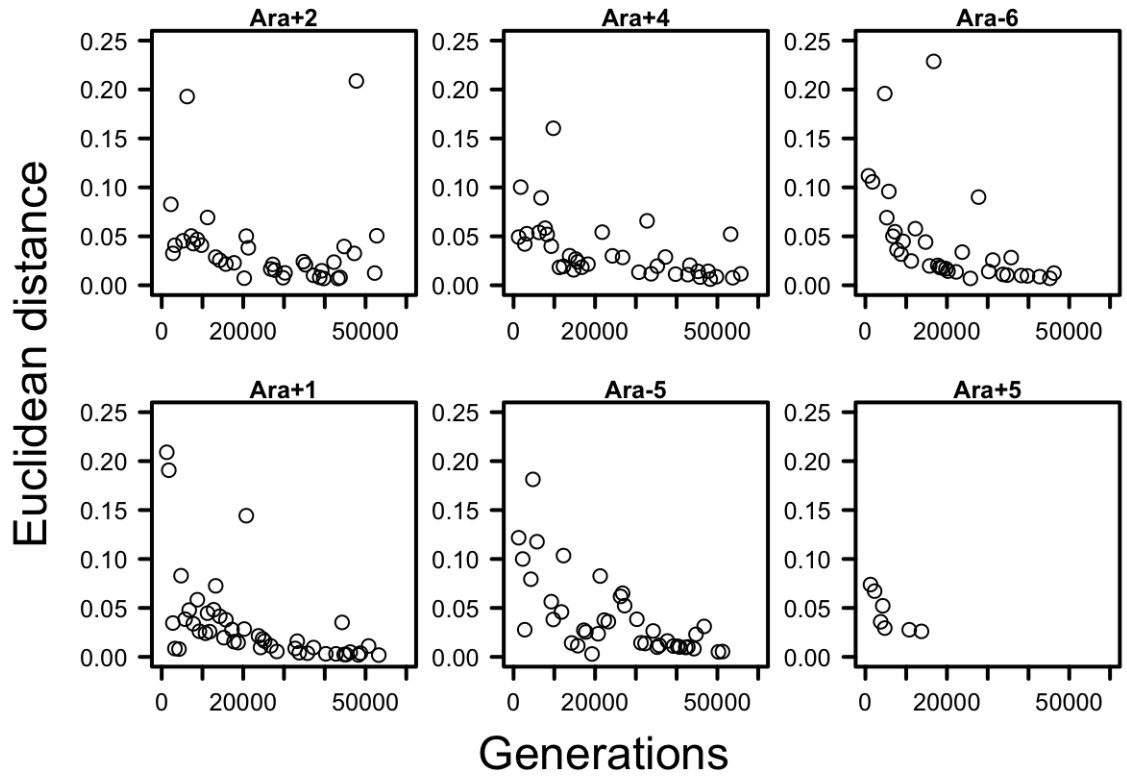
```
times.sorted <- as.character(sort(as.numeric(unique(times))))
time_steps <- c()
for (index in seq(1, length(head(times.sorted, -1)))) {
  time_step <- paste(times.sorted[index], times.sorted[index+1], sep = "_")
  time_steps <- c(time_steps, time_step)
}

euc.df <- as.data.frame(matrix(data=NA,nrow=length(complete_nonmutator_lines),
                               ncol=length(tail(times.sorted, -1))))
rownames(euc.df) <- complete_nonmutator_lines
colnames(euc.df) <- tail(times.sorted, -1)

for (pop in complete_nonmutator_lines) {
  for (time_step in time_steps){
    time1 <- strsplit(time_step, '_')[[1]][1]
    time2 <- strsplit(time_step, '_')[[1]][2]
    pop.time1 <- paste(pop, time1, sep = "_")
    pop.time2 <- paste(pop, time2, sep = "_")
    if ( (pop.time1 %in% rownames(df.pcoa$points))
         & (pop.time2 %in% rownames(df.pcoa$points)) ) {
      euc.dist <- dist(rbind(df.pcoa$points[pop.time1, ], df.pcoa$points[pop.time2, ]))[1]
      euc.df[pop, time2] <- euc.dist
```

```
    }
  }
}

euc.df.p2 <- euc.df['p2',]
euc.df.p2.clean <- euc.df.p2[,!is.infinite(colSums(euc.df.p2))
                                        &  !is.na(colSums(euc.df.p2))
                                    & (log10(colSums(euc.df.p2)) > -10) ]
euc.df.p4 <- euc.df['p4',]
euc.df.p4.clean <- euc.df.p4[,!is.infinite(colSums(euc.df.p4))
                                        &  !is.na(colSums(euc.df.p4))
                                    & (log10(colSums(euc.df.p4)) > -10) ]
euc.df.m6 <- euc.df['m6',]
euc.df.m6.clean <- euc.df.m6[,!is.infinite(colSums(euc.df.m6))
                                        &  !is.na(colSums(euc.df.m6))
                                    & (log10(colSums(euc.df.m6)) > -10) ]
euc.df.p1 <- euc.df['p1',]
euc.df.p1.clean <- euc.df.p1[,!is.infinite(colSums(euc.df.p1))
                                        &  !is.na(colSums(euc.df.p1))
                                    & (log10(colSums(euc.df.p1)) > -10) ]
euc.df.m5 <- euc.df['m5',]
euc.df.m5.clean <- euc.df.m5[,!is.infinite(colSums(euc.df.m5))
                                        & !is.na(colSums(euc.df.m5))
                                & (log10(colSums(euc.df.m5)) > -10)]
euc.df.p5 <- euc.df['p5',]
euc.df.p5.clean <- euc.df.p5[,!is.infinite(colSums(euc.df.p5))
                                        &  !is.na(colSums(euc.df.p5))
                                & (log10(colSums(euc.df.p5)) > -10) ]

png(filename = paste(c("figs/euc_pcoa.png"), collapse = ''),
    width = 1200, height = 900, res = 96*2)
par(mfrow = c(2, 3),
    oma = c(5, 4, 0, 0), # two rows of text at the outer left and bottom margin
    mar = c(1.5, 3.1, 1.1, 0.5), # space for one row of text at ticks and to separate plots
    pty="s") # make the plots square

# plots for each population
euc.p2.plot <- plot(colnames(euc.df.p2.clean), euc.df.p2.clean,
                    xlim = c(0, 60000), ylim = c(0, 0.25),
                    pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                    axes = FALSE, xlab = '', ylab ='', main = "Ara+2")
points(colnames(euc.df.p2.clean), euc.df.p2.clean,
       pch = 1, cex = 1.5, bg = "gray",  lwd  = 1)
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
box(lwd = 2)

pcoa.p4.plot <- plot(colnames(euc.df.p4.clean), euc.df.p4.clean,
                     xlim = c(0, 60000), ylim = c(0, 0.25),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara+4")
points(colnames(euc.df.p4.clean), euc.df.p4.clean,
       pch = 1, cex = 1.5, bg = "gray",  lwd  = 1)
```

```
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
box(lwd = 2)

pcoa.m6.plot <- plot(colnames(euc.df.m6.clean), euc.df.m6.clean,
                     xlim = c(0, 60000), ylim = c(0, 0.25),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara-6")
points(colnames(euc.df.m6.clean), euc.df.m6.clean,
       pch = 1, cex = 1.5, bg = "gray",  lwd  = 1)
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
box(lwd = 2)

pcoa.p1.plot <- plot(colnames(euc.df.p1.clean), euc.df.p1.clean,
                     xlim = c(0, 60000), ylim = c(0, 0.25),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara+1")
points(colnames(euc.df.p1.clean), euc.df.p1.clean,
       pch = 1, cex = 1.5, bg = "gray",  lwd  = 1)
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
box(lwd = 2)

pcoa.m5.plot <- plot(colnames(euc.df.m5.clean), euc.df.m5.clean,
                     xlim = c(0, 60000), ylim = c(0, 0.25),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara-5")
points(colnames(euc.df.m5.clean), euc.df.m5.clean,
       pch = 1, cex = 1.5, bg = "gray",  lwd  = 1)
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
box(lwd = 2)

pcoa.p5.plot <- plot(colnames(euc.df.p5.clean), euc.df.p5.clean,
                     xlim = c(0, 60000), ylim = c(0, 0.25),
                     pch = 2, cex = 2.0, type = "n", cex.lab = 1.5, cex.axis = 1.2,
                     axes = FALSE, xlab = '', ylab ='', main = "Ara+5")
points(colnames(euc.df.p5.clean), euc.df.p5.clean,
       pch = 1, cex = 1.5, bg = "gray",  lwd  = 1)
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
box(lwd = 2)

mtext('Generations', side=1, line=1,
      cex=1.5, col="black", outer=TRUE)
mtext('Euclidean distance', side=2, line=1,
      cex=1.5, col="black", outer=TRUE)

dev.off()
```

```
## pdf
##   2
```

```
# Show Plot
img <- readPNG(paste(c("figs/euc_pcoa.png"), collapse = ''))
grid.raster(img)
```



Beta disper

References

Good, B. H., M. J. McDonald, J. E. Barrick, R. E. Lenski, and M. M. Desai. 2017. The dynamics of molecular evolution over 60,000 generations. Nature 551:45–50