**Scanning genomes for single-cell Probabilistic Trait Loci**

Florent Chuffart, Magali Richard, Daniel Jost, Claire Burny, Hélène Duplus-Bottin, Yoshikazu Ohya and Gaël Yvert

Supplementary Methods

Description of key steps of scPTL mapping

(R implementation)

Note: The implementation of the methods is fully available as an open source R package called *ptlmapper*: (https://github.com/fchuffar/ptlmapper).

**Calculation of the Kantorovich distance**.

Let x1 and x2 be the vectors of single-cell trait values for individuals 1 and 2, respectively. We built two histograms *h1* and *h2* of single cell trait values with a common *breaks* attribute:

breaks = seq(min, max, by = nb\_bin-1)

h1 = hist(x1, plot = FALSE, breaks = breaks)

h2 = hist(x2, plot = FALSE, breaks = breaks)

with *min* (resp. *max)* being the global minimal (resp. the maximal) value of the single cell trait over all individuals. The number of bins (*nb\_bin*) was set to 100 (similar results were obtained at various binning precisions). The distance was then computed using the code:

bin\_size = h1$breaks[2] - h1$breaks[1];

KD = bin\_size^2\*sum(abs(cumsum(h1$density - h2$density))).

**Multi-dimensional Scaling**

Multidimensional scaling was done using function cmdscale() from the stats package. The number of informative dimensions to keep was determined by calling cmdscale a first time with an arbitrary value for the parameter k (here k = 2) and then interrogating how many of the resulting eigenvalues exceeded expectation from a uniform distribution (i.e. if the data has no structure, Kaiser criterion). This was done as follows:

# Compute the number of significant eigen values of multi-dimensional scaling

get\_nb\_eig\_sign = function(pheno\_KD) {

 mds = cmdscale(pheno\_KD, k = 2, eig = TRUE)

 mds\_eig = mds$eig[mds$eig > 0]

 nb\_eig\_sign = sum(mds\_eig/sum(mds\_eig) >= 1/length(mds$eig))

 return(nb\_eig\_sign)

 }

The output of this function was the resulting number of dimensions nb\_dim. If it was lower than 2, we set it to 2. After this, cmdscale was called a second time with k = nb\_dim, the desired number of dimensions. This was done as follows:

nb\_dim = max(2, get\_nb\_eig\_sign(pheno\_KD))

mds = cmdscale(pheno\_KD, k = nb\_dim, eig = TRUE)

data = data.frame(mds$points)

**Canonical Analysis**

Linear discriminant analysis was done using a custom function seqWilk() that we derived from the candisc() function of the candisc package. The interest of seqWilk() is to extract the W score that we use as test statistics.

############################################################

# A function that returns useful values that are initially #

# embedded in the output of the candisc() function. #

############################################################

seqWilks = function (eig, p, df.h, df.e) {

 p.full = length(eig)

 result = matrix(0, p.full, 4)

 m = df.e + df.h - ( p.full + df.h + 1)/2

 for (i in seq(p.full)) {

 test = prod(1/(1 + eig[i:p.full])) # this is the Lambda score that we

 # use as the test statistics

 p = p.full + 1 - i

 q = df.h + 1 - i

 s = p^2 + q^2 - 5

 if (s > 0) {

 s = sqrt(((p \* q)^2 - 4)/s)

 } else {

 s = 1

 }

 df1 = p \* q

 df2 = m \* s - (p \* q)/2 + 1

 result[i,] = c(test, ((test^(-1/s) - 1) \* (df2/df1)),

 df1, df2)

 }

 result = cbind(result, pf(result[,2], result[,3],

 result[,4], lower.tail = FALSE))

 colnames(result) = c("LR test stat", "approx F",

 "num Df", "den Df", "Pr(> F)")

 rownames(result) = 1:p.full

 return(result)

 }

#############################

# Then the actual analysis #

#############################

# Some formatting of the data...

data$allele = as.factor(all)

data = data[!is.na(data$allele),]

# Building the statistical model according to colnames of object data

model\_formula = paste("cbind(", paste(names(data)[-length(names(data))],

 collapse=", "), ") ~ allele")

# ...followed by the analysis itself

mod = lm(model\_formula, data=data)

can = candisc(mod, data=data)

p = can$rank

eig = can$eigenvalues[1:p]

df.h = can$dfh

df.e = can$dfe

tests = seqWilks(eig, p, df.h, df.e)

return(list(W=tests[1,5], can=can, tests=tests))

**Trait heritability in the multi-dimensional phenotypic space**:

Let points be the data matrix giving the coordinates of each sample in the phenotypic space, with the first 3 rows being the replicate samples from one parental strain, rows 4 to 6 the replicate samples from the other parental strain, and rows greater than 7 the samples of the segregant strains (one sample per segregant). Genetic heritability H was computed as follows:

# environmental variance

var\_e\_p1 = sum(diag(cov(points[1:3,]))) # from first parent

var\_e\_p2 = sum(diag(cov(points[4:6,]))) # from second parent

var\_e = mean(c(var\_e\_p1, var\_e\_p2))

# total variance

var\_t = sum(diag(cov(points[7:nrow(points),])))

# heritability

H = (var\_t - var\_e) / var\_t