**PER-SIMPER – a new tool for inferring community assembly processes from taxon occurrences**

Corentin Gibert and Gilles Escarguel

DOI :

**SUPPLEMENTARY MATERIAL S2. R Function**

**S2.1 PER-SIMPER**

**Description**

Based on a presence/absence matrix, this function allows the identification of the first-order process of assembly underlying a set of taxonomic assemblages. Its use should therefore be limited to the comparison of **significantly distinct taxonomic sets**. But at the same time **connected enough** to allow the potential dispersal of species between these different sets. The PER-SIMPER method distinguishes the main ecological assembly process (between **species dispersal capacity** and **niche** **richness**) at the origin of the observed taxonomic differences between two (or more) compared sets of assemblages. The PER-SIMPER method is associated with the calculation of the *E* index (the logarithm of the sum of squared deviations between empirical and simulated SIMPER profiles) to assist in distinguishing the result of PER-SIMPER analyses.

**Arguments**

The following R function requires 5 arguments, 2 are necessary -in italics- (*matrixSIMP*, *Groups*, log, leg, count) :

*matrixSIMP* a presence/absence or a abundance (and numerical) **matrix** with **taxa in columns** and **locality in rows**. Columns and rows can have names but it’s not necessary. For pre/abs matrix presence need to be coded by 1 and absence by 0, no other characters (like « ? » or « x ») are supported.

*Groups* a vector allowing to assign the different compared localities (the lines of the matrix) to 2 or more clusters of assemblies. This vector accepts numeric characters such as strings. For example, for a matrix consisting of 10 localities, where 6 are grouped in one cluster and 4 in the other, the vector *Groups* can take this form: "1", "1", "1", "1", "1", "1", "2", "2", "2", "2", or this form : "region1", "region1", "region1", "region1", "region1", "region1", "region2", "region2", "region2", "region2". This argument is similar to the one required for the "**simper**" function of the "**vegan**" package named "**group**".

PerSIMPER function requests exactly the same required arguments (*matrixSIMP*, *Groups*) as the functions used to compute SIMPER method like the *simper* function of the *vegan*package. The following three arguments are optional, but their use is strongly recommended to optimize the use of PER-SIMPER.

log allows to log the y-axis (i.e. the percentage contribution to the overall average dissimilarity –OAD- of each species) of the SIMPER and PER-SIMPER profiles. **By** **default = FALSE**.

leg allows a legend to be displayed on the produced PER-SIMPER profiles. Percentage of overlapping of each null model with empirical SIMPER profile are displayed. **By default = TRUE**.

count allows a screen output of the number of iterations performed. This option is used to indicate if the permutation function is unable to swap the matrix cells. This incapacity is usually the result of a matrix too sparse in data (too many cells at 0). **By default = FALSE**.

dataTYPE allows to choose between presence/absence data or abundance data. By default the algorithm use presence/absence permutation for presence/absence data, if you use abundance dataset, you need to write "count" in the dataTYPE argument. **By default = « prab »**.

**Example**

* 2 **tests files** are included in the DRYAD package, *PerSIMPERmatrix* is a matrix in *.txt* with species names in columns and localities names in lines. *PerSIMPERgroups* is a numeric vector in *.txt* filled with 1 for the first cluster of locality and 2 for the second one.

**Example of PER-SIMPER use :** *R Code*

source("PERSIMPER.R")

ExampleMatrix <- read.table("PerSIMPERmatrix.txt", h = T)

ExampleMatrix <- as.matrix(ExampleMatrix)

ExampleGroups <- read.table("PerSIMPERgroups.txt")

ExampleGroups <- ExampleGroups[,1]

Results <- PerSIMPER(ExampleMatrix, ExampleGroups, log = TRUE, count = TRUE)

**Complete R Code of PER-SIMPER function**

PerSIMPER <- function(matrixSIMP, Groups, log = TRUE, leg = TRUE, count = TRUE, dataTYPE = "prab"){

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

## for every problems or questions, please contact me by mail or ResearchGate: ##

##### at corentingibert@gmail.com | corentin.gibert@univ-poitiers.fr ############

########### https://www.researchgate.net/profile/Corentin\_Gibert ################

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

# The PerSIMPER function requests exactly the same required arguments (matrixSIMP & Groups)

# as the functions used to compute SIMPER method in R, i.e. a presence/absence matrix and

# a vector encoding cluster information.

#Arguments :

#matrixSIMP <- Stores the matrix to use in SIMPER analysis (i.e. the result

#of the presence/absence or the abundance distribution of taxa in at least 2 clusters of assemblies)

# LOCALITIES in LINES

# TAXA in COLUMNS

#Groups <- A vector allowing to assign to the lines of the SIMPER matrix (matrixSIMP) a cluster of

#localities ; for example in a matrix of 10 lines built from 2 sets of localities

#of the same size : 1, 1, 1, 1, 1, 2, 2, 2, 2, 2. Strings are accepted, e.g., RegionA, Region, A, Region B, etc...

#log <- the "log" argument allows to log the y-axis (i.e. the percentage contribution to the OAD of the species)

#leg <- the "leg" argument allows a legend to be displayed on the Per-SIMPER profile

#count <- the "count" argument allows a Screen output of the number of iterations performed.

#This option is used to indicate if the permutation function is unable to swap the matrix cells.

#This incapacity is usually the result of a matrix too sparse in data (too many cells at 0).

#dataTYPE <- the "dataTYPE" argument allows to choose between presence/absence data or abundance data. By default

#the algorithm use presence/absence permutation for presence/absence data, if you use abundance dataset, you need

#to write "count" in the dataTYPE argument.

#e.g. dataTYPE = "count"

library(vegan)

library(ggplot2)

AnaSimp <- simper(matrixSIMP, Groups)

#Classical SIMPER analysis computed on the compared groups

summary(AnaSimp)

Contribution <- sort(AnaSimp[[1]]$average, decreasing = TRUE)

#Replication in a vector (named 'Contribution') of the sorting

#of species by their contribution to overall dissimilarity (OAD)

Pourcent\_Contribution <- ((Contribution)/sum(Contribution))\*100

#Conversion as a percentage of each species' contribution to the OAD

if(log==TRUE)plot(Pourcent\_Contribution, col = "brown2", log="y", type="p",lwd = 1.5, ylab ="% contribution to dissimilarity", xlab="Species")

if(log==FALSE)plot(Pourcent\_Contribution, col = "brown2", type="p",lwd = 1, ylab ="% contribution to dissimilarity", xlab="Species")

#Ploting SIMPER results in percentage (in Log or not)

dp2 <- permatfull(matrixSIMP, fixedmar = "both", mtype = dataTYPE, times = 1000)

dp3 <- permatfull(matrixSIMP, fixedmar ="rows" , mtype = dataTYPE, times = 1000)

#Randomization of the matrixSIMP matrix ; permatfull need to be used in order to swap cells under

#various conditions. permatswap only allow permutation with both fixed rows and fixed columns count.

#mtype = "prab" is used for presence/absence data, this setting must be changed with "count" if abundance

#analysis are performed.

df2 <- matrix(nrow = 1000, ncol = length(matrixSIMP[2,]))

df3 <- matrix(nrow = 1000, ncol = length(matrixSIMP[2,]))

df4 <- matrix(nrow = 1000, ncol = length(matrixSIMP[2,]))

#Generating matrices that will store the results (the ranked contribution of species to the OAD)

#of the 1000 permutations of the original matrix

for(i in 1:1000)

{

if(count == TRUE && i < 50 || count == TRUE && i > 950 ){print(i)}

#Screen output of the number of iterations performed.

#This option is used to indicate if the permutation function is unable to swap the matrix cells.

#This incapacity is usually the result of a matrix too sparse in data (too many cells at 0).

repeat{

v <- T

dp4 <- permatfull(matrixSIMP, fixedmar = "columns", mtype = dataTYPE, times = 1)

for(j in 1:length(dp4$perm[[1]][,2]))

{

if(sum(dp4$perm[[1]][j,]) == 0){v <- FALSE}

}

if(v == TRUE)break}

simp2 <- simper(dp2$perm[[i]], Groups)

simp3 <- simper(dp3$perm[[i]], Groups)

simp4 <- simper(dp4$perm[[1]], Groups)

#SIMPER analysis performed on each permutated matrix

df2[i,] <- sort(simp2[[1]]$average, decreasing = TRUE)

df3[i,] <- sort(simp3[[1]]$average, decreasing = TRUE)

df4[i,] <- sort(simp4[[1]]$average, decreasing = TRUE)

#Storage of SIMPER results (ranked contribution to OAD)

df2[i,] <- (df2[i,]/sum(df2[i,]))\*100

df3[i,] <- (df3[i,]/sum(df3[i,]))\*100

df4[i,] <- (df4[i,]/sum(df4[i,]))\*100

#Conversion to percentage of SIMPER results

}

dn2 <- apply(df2, 2, sort)

dn3 <- apply(df3, 2, sort)

dn4 <- apply(df4, 2, sort)

lines(dn2[975,], lty="dotted", lwd=2, col="dodgerblue") #sans type="o", avec lwd=2

lines(dn2[25,], lty="dotted", lwd=2, col="dodgerblue")

lines(dn3[975,], lty="dotted", lwd=2, col="chartreuse3")

lines(dn3[25,], lty="dotted", lwd=2, col="chartreuse3")

lines(dn4[975,], lty="dotted", lwd=2, col="orange2")

lines(dn4[25,], lty="dotted", lwd=2, col="orange2")

#Plot of the upper and lower limit of the confidence intervals for fixed rows, fixed columns and fixed rows and columns

title("SIMPER (in red) and PER-SIMPER profiles")

legend(x="topright", bty="n",legend=c("SIMPER profil", "Col+Row fixed", "Rows fixed", "Col fixed"), col=c("brown2", "dodgerblue","chartreuse3","orange2"), pch=c(15,15,15,15))

vecO <- c()

vecGreen <- c()

vecRed <- c(0,0,0,0)

for(i in 1:length(Pourcent\_Contribution))

{

if(dn3[975,i] >= dn2[975,i] && dn3[25,i] <= dn2[25,i]){vecGreen <- c(vecGreen, 100)}

if(dn3[975,i] > dn2[975,i] && dn3[25,i] >= dn2[25,i] && dn3[25,i] < dn2[975,i]){vecGreen <- c(vecGreen, (((dn2[975,i] - dn3[25,i])/(dn2[975,i] - dn2[25,i])) \* 100))}

if(dn3[975,i] <= dn2[975,i] && dn3[25,i] < dn2[25,i] && dn3[975,i] > dn2[25,i]){vecGreen <- c(vecGreen, (((dn3[975,i] - dn2[25,i])/(dn2[975,i] - dn2[25,i])) \* 100))}

if(dn2[975,i] > dn3[975,i] && dn2[25,i] < dn3[25,i]){vecGreen <- c(vecGreen, (((dn3[975,i] - dn3[25,i])/(dn2[975,i] - dn2[25,i])) \* 100))}

if(dn3[975,i] <= dn2[25,i]){vecGreen <- c(vecGreen, 0)}

if(dn3[25,i] >= dn2[975,i]){vecGreen <- c(vecGreen, 0)}

if(Pourcent\_Contribution[i] >= dn3[25,i] && Pourcent\_Contribution[i] <= dn3[975,i]){vecRed[1] <- vecRed[1] + 1}

#Computation of % of empirical SIMPER data included inside the Green simulated confidence interval (under fixed rows condition)

if(dn4[975,i] >= dn2[975,i] && dn4[25,i] <= dn2[25,i]){vecO <- c(vecO, 100)}

if(dn4[975,i] > dn2[975,i] && dn4[25,i] >= dn2[25,i] && dn4[25,i] < dn2[975,i]){vecO <- c(vecO, (((dn2[975,i] - dn4[25,i])/(dn2[975,i] - dn2[25,i])) \* 100))}

if(dn4[975,i] <= dn2[975,i] && dn4[25,i] < dn2[25,i] && dn4[975,i] > dn2[25,i]){vecO <- c(vecO, (((dn4[975,i] - dn2[25,i])/(dn2[975,i] - dn2[25,i])) \* 100))}

if(dn2[975,i] > dn4[975,i] && dn2[25,i] < dn4[25,i]){vecO <- c(vecO, (((dn4[975,i] - dn4[25,i])/(dn2[975,i] - dn2[25,i])) \* 100))}

if(dn4[975,i] <= dn2[25,i]){vecO <- c(vecO, 0)}

if(dn4[25,i] >= dn2[975,i]){vecO <- c(vecO, 0)}

if(Pourcent\_Contribution[i] >= dn4[25,i] && Pourcent\_Contribution[i] <= dn4[975,i]){vecRed[2] <- vecRed[2] + 1}

#Computation of % of empirical SIMPER data included inside the Orange simulated confidence interval (under fixed columns condition)

if(Pourcent\_Contribution[i] <= dn2[975,i] && Pourcent\_Contribution[i] >= dn2[25,i]){vecRed[4] <- vecRed[4] + 1}

#Computation of % of empirical SIMPER data included inside the Blue simulated confidence interval (under fixed rows and fixed columns conditions)

}

vecRed <- (vecRed/length(Pourcent\_Contribution))\*100

a <- min(Pourcent\_Contribution)

a

if(leg == TRUE)

{

min(Pourcent\_Contribution)

text(x=0.8\*(length(Pourcent\_Contribution)), y=(mean(Pourcent\_Contribution)), labels=paste("Red include in blue : ", ceiling(mean(vecRed[4])), " %", "\n",

"Green in Blue : ", ceiling(mean(vecGreen)), " %. Red incl. : ", ceiling(vecRed[1]), " %","\n",

"Orange in Blue : ", ceiling(mean(vecO)), "%. Red incl. : ", ceiling(vecRed[2]), " %","\n" ))

}

#### The following is the calculation and the illustration of E index ####

#### E = Log of the sum of square deviations with empirical profile ####

library(ggplot2)

obs <- Pourcent\_Contribution

Orange <- dn4

Blue <- dn2

Green <- dn3

# Ranked % of contribution to OAD of empirical and simulated profiles

VectorEcartCarreOrangeLog <- vector(mode = "numeric", 1000)

VectorEcartCarreGreenLog <- vector(mode = "numeric", 1000)

VectorEcartCarreBlueLog <- vector(mode = "numeric", 1000)

for(i in 1:1000)

{

SommeEcartCarreOrange <- vector(mode = "numeric", length = length(Orange[1,]))

SommeEcartCarreGreen <- vector(mode = "numeric", length = length(Green[1,]))

SommeEcartCarreBlue <- vector(mode = "numeric", length = length(Blue[1,]))

for(j in 1:length(obs))

{

SommeEcartCarreOrange[j] <- (Orange[i,j] - obs[j])^2

SommeEcartCarreGreen[j] <- (Green[i,j] - obs[j])^2

SommeEcartCarreBlue[j] <- (Blue[i,j] - obs[j])^2

}

# Computation of square deviations with empirical profile (obs)

#Mise en log des carre des ecarts pour symetriser la distribution

VectorEcartCarreOrangeLog[i] <- log10(sum(SommeEcartCarreOrange))

VectorEcartCarreGreenLog[i] <- log10(sum(SommeEcartCarreGreen))

VectorEcartCarreBlueLog[i] <- log10(sum(SommeEcartCarreBlue))

}

# Log conversion of the sum of square deviations

meanCarreOrangeLog <- mean(VectorEcartCarreOrangeLog)

meanCarreGreenLog <- mean(VectorEcartCarreGreenLog)

meanCarreBlueLog <- mean(VectorEcartCarreBlueLog)

DataMeanCarreLog <- data.frame(Orange=VectorEcartCarreOrangeLog, Blue=VectorEcartCarreBlueLog, Green=VectorEcartCarreGreenLog)

# Computation of the mean of the logged sums

## BOXPLOT with 95 % intervals

Ax <- c("Fixed columns", "Both fixed", "Fixed rows")

y <- DataMeanCarreLog

df <- data.frame(

Permutation\_model = Ax,

y0 = quantile(y$Orange, 0.025),

y25 = quantile(y$Orange, 0.25),

y50 = median(y$Orange),

y75 = quantile(y$Orange, 0.75),

y100 = quantile(y$Orange, 0.975)

)

df[2,2] = quantile(y$Blue, 0.025)

df[2,3] = quantile(y$Blue, 0.25)

df[2,4] = median(y$Blue)

df[2,5] = quantile(y$Blue, 0.75)

df[2,6] = quantile(y$Blue, 0.975)

df[3,2] = quantile(y$Green, 0.025)

df[3,3] = quantile(y$Green, 0.25)

df[3,4] = median(y$Green)

df[3,5] = quantile(y$Green, 0.75)

df[3,6] = quantile(y$Green, 0.975)

# Extraction of quantiles of interest

print(df)

E <- ggplot(df, aes(Permutation\_model, fill = Permutation\_model)) +

geom\_boxplot(

aes(ymin = y0, lower = y25, middle = y50, upper = y75, ymax = y100),

stat = "identity") + scale\_fill\_manual(values=c("#0099FF", "#FF6600", "#00FF33")) +

ggtitle("The lower E, the closer the simulated profile to empirical SIMPER profile") +

theme(plot.title = element\_text(lineheight= 2)) +

scale\_y\_continuous(name=" E (Log of the sum of square deviations with empirical profile)")+

labs(fill="Permutation models")+

scale\_x\_discrete(name = "Permutation models")

print(E)

#Computation of ggplots2 boxplot for E index

ListResults <- list(EcartCarreLog = DataMeanCarreLog, Eplot = E, Red=(vecRed), Green=mean(vecGreen), Orange=mean(vecO), mat = matrixSIMP, ContriPercentage = Pourcent\_Contribution, UpOrange = dn4[975,], DownOrange = dn4[25,], MedOrange = dn4[500,], UpBlue = dn2[975,], DownBlue = dn2[25,], MedBlue = dn2[500,], UpGreen = dn3[25,], DownGreen = dn3[975,], MedGreen = dn3[500,], dnOrange = dn4, dnBlue = dn2, dnGreen = dn3)

return(ListResults)

}