

# Similarities and distances

*Gilles San Martin*

*25 September 2018 - 02h22*

## Contents

<b>1</b>	<b>Similarities, dissimilarities and distances</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Which approach should you use ? . . . . .	3
1.3	Data used : Orthoptera dataset . . . . .	4
1.4	Data transformation . . . . .	5
1.4.1	Common general transformation strategies (for all kind of datasets) . . . . .	5
1.4.2	Specific standardizations frequently used in community ecology . . . . .	9
1.4.3	Transformation based distances . . . . .	10
1.5	Overview of the different similarity/disimilarity measures . . . . .	11
1.6	Q mode : distances between observations (rows) . . . . .	14
1.6.1	Asymetric indices for quantitative data (typically : Species abundances) . . . . .	14
1.6.2	Asymetric indices for binary data (typically : Species presence/absence) . . . . .	14
1.6.3	Comparison of the asymetric indices in Q mode . . . . .	15
1.6.4	Symetric indices for quantitative data (typically : Environmental variables) . . . . .	16
1.6.5	Symetric indices for binary data . . . . .	16
1.6.6	Indices for qualitative data and mixed types . . . . .	17
1.7	R mode : dependance between variables (columns) . . . . .	19
1.7.1	Asymetric indices for quantitative data (typically : Species abundances) . . . . .	20
1.7.2	Asymetric indices for binary data (typically : Species presence/absence) . . . . .	21
1.7.3	Symetric indices for quantitative, ordinal and binary data . . . . .	22
1.8	Typical problems when measuring distances . . . . .	23
1.8.1	Illustration of the double 0 problem . . . . .	23
1.8.2	Scale problems with Euclidean and Bray-Curtis distances . . . . .	29
1.8.3	Potential problems with Hellinger and Chord distance . . . . .	32
1.8.4	Potential problems with the Chi Squared distance . . . . .	35

---

# 1 Similarities, dissimilarities and distances

```
source("/home/gilles/stats/mytoolbox.R")
setwd("/home/gilles/stats/Formation_R_stats/Formation_Stats_4_Multivariate/")

# load the packages used for this chapter
library(vegan)
library(ade4)
library(FD)

# load ggplot, change the default theme and change the locale (language = English)
library(ggplot2)

Sys.setlocale("LC_ALL", 'en_GB.UTF-8')
mytheme <- theme_bw(10) + theme(axis.text.x=element_text(size=8),
                                legend.key = element_rect(color = NA))
theme_set(mytheme)
```

## 1.1 Introduction

Most of the multivariate methods - supervised and unsupervised- are based implicitly or explicitly on a measure of similarity either between the observations (rows) or the variables (columns) of the dataset.

The aim of most of these methods is indeed to simplify a complex multidimensional dataset by grouping similar observations or variables or by reorganising them along gradients. To do that we need to define what “similar” means and there are many possibilities...

For each dataset you can imagine a geometric representation where the variables are the axes and the observations are placed in the space created by these axes. You can then measure the distance between the observations in this Euclidean space. This is a measure of how dissimilar these observations are...

One of the most important and frequently used distance is the Euclidean distance which measures the distance between two points in a cartesian space like in the example above:

$$d_{ij} = \sqrt{\sum_{k=1}^p (x_{ik} - x_{jk})^2}$$

There are however many other ways to measure the distances/similarities that can provide very different results.

The words “similarity”, “dissimilarity” and “distance” are often used interchangeably even if they are not exact synonyms. A similarity is 1 when two observations are perfectly similar and 0 when they are completely different. A dissimilarity is just the reverse and a similarity “S” can simply be transformed into a dissimilarity with  $D = 1 - S$ . The dissimilarity is bounded between 0 and 1 and as such when for example 2 sites share no species their maximum dissimilarity can only be 1. A distance is 0 for perfectly identical observations but has no upper bound so the distance between 2 sites sharing no species will depend on the number and abundance of their species.

In R all methods need a distance matrix and so similarities must first be transformed into dissimilarities and will be treated as distances. In practice this makes not much difference and these terms will be used almost as synonyms here.

## 1.2 Which approach should you use ?

Choosing the “right” distance index and the method to visualise them (clustering, ordination) is not always easy because so many different options are available.

Here are some recommendations to help you chose an appropriate distance index :

- follow the guidelines provided by Legendre & Legendre and detailed here after :
  - distances between variables (R mode) or observations (Q mode) ?
  - type of variables : quantitative, ordinal, binary, qualitative, mixed ?
  - should double 0 increase the similarity (symetric distances) or should they be ignored (asymetric distances) ?
  - Should you transform your data before computing the distance ?
- look at the methods/indices typically used in your field (but keep your mind open and critical...)
- understand the distances you want to use by
  - recomputing the distance manually and compare to the distance provided by the R function
  - experimenting on small, fake dataset that will push these indices to extreme conditions
- compute several different “reasonable” type of distances on your real life dataset, look how much their results diverge and try to understand why. Quite frequently, on real life datasets, different indices will provide very similar information. So the choice is less critical in that case.
- keep in mind that there is often not one right index/method but several reasonable methods that will provide different views and different insights on your dataset
- In practice, we generally end up using almost always the same indices and methods that are “reputed” to work well for each particular case without exploring the many different other possibilities for each analysis ...

If you are lost, these are generally easy and relatively safe starting points :

- Hellinger or Bray-Curtis distance (for rows) or correlation distance on hellinger transformed data (for columns) for species abundances
- Jaccard dissimilarity for binary species data (both for the rows and columns)
- For all other cases with quantitative (and to some extend binary) data : euclidean distance on the rows (standrardization of the data is then often necessary or preferable) and correlation distance on the columns
- If you have qualitative data (or a mix), the situation is more complex... You can use one of the several forms of the Gower distance or use specific clustering or ordination methods (SOM, FAMD)
- for clustering : hierarchical clustering with ward.D method will most of the time provide what you want : well seperated, homogenous, interpretable clusters (as much as possible). Do systematically a heatmap with the dendrograms to be able to interpret the results.
- for ordination - a good starting point could be :
  - PCA on hellinger transformed data for species like abundance or presence/absence data
  - PCA on scaled data (PCA on the correlation matrix) for all other cases
  - If you have to frequently analyse species community data or qualitative data, you should probably invest some time to learn Non Metric Multidimentional Scaling (NMDS) (and maybe Corespondance Analysis (CA))

### 1.3 Data used : Orthoptera dataset

Orthoptera relevés in chalk grasslands in Belgium (4 sites, several zones per site, 3 quadrats per zone → 56 quadrats) + quantitative and qualitative environmental variables measured. *Gomphocerippus rufus* (rufu) and *Chorthippus parallelus* (para) are the most abundant species while *Chorthippus albomarginatus* (albo), *Metrioptera roeselii* (roes) and *Chorthippus vagans* (vaga) are the rarest species

```
spe <- read.csv2("data/orthops/orthops_misc/Orthops_Species.csv", row.names = 1)[-52,]
envquant <- read.csv2("data/orthops/orthops_misc/Orthops_EnvQuant.csv", row.names = 1)[-52,]
envqual <- read.csv2("data/orthops/orthops_misc/Orthops_EnvQual.csv", row.names = 1)[-52,]
sites <- read.csv2("data/orthops/orthops_misc/Orthops_Sites.csv")[-52,]

env <- cbind(envquant, envqual[, c("Shadow", "Humidity", "Exp")])
spe <- spe[, !colnames(spe) %in% c("tenu", "subu")]

envqual <- envqual[, 1:4]
envqual$Slope <- factor(envqual$Slope, levels = c("Null", "Weak", "Strong"))
envqual$Shadow <- factor(envqual$Shadow)
envqual$Humidity <- factor(envqual$Humidity)

row.names(sites) <- sites$Quadrat

# summary(env)
# summary(spe)
# summary(envqual)
```

Frequency of the different species :

```
rev(sort(colSums(spe)))
```

##	rufu	para	sylv	falc	bigu	gris	disp	line	rufi	oviri	disc	tviri	brac	bico	undu
##	500	353	184	162	119	100	81	59	58	46	46	31	29	29	24
##	bipu	brun	punc	albo	roes	vaga									
##	21	21	10	3	2	2									

## 1.4 Data transformation

It is often useful or even necessary to transform the data before an analysis and before computing the distances. There are also a few specific transformations that will transform a Euclidean distance into other distances. The `decostand` function from `vegan` is a useful tool for these transformations.

### 1.4.1 Common general transformation strategies (for all kind of datasets)

**scaling** : for each column remove the average and divide by the standard deviation.

The result is a dataset without unit, with the same variance for all variables ( $= 1$ ) and an average of 0 for all variables. This is particularly useful when you have a dataset composed of variables with different units. In that case it is almost always useful to scale the data in order to make the variables comparable and without units (particularly before computing Euclidean distances). Many methods can do that internally (eg correlation coefficient is the covariance of a scaled dataset, PCA often applies scaling internally, ...)

In the iris dataset for example, the units are the same but the total variance is not comparable between the different measures. If you compute a euclidean distance based on these raw data, the variable with the highest standard deviation will always have the highest weight in the distance computation while this variable is maybe not the most important to characterize the samples.

```
# Compute the sd of each numeric column --> they are different
```

```
tmp <- iris
apply(tmp[,1:4], 2, sd)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
##      0.8280661      0.4358663      1.7652982      0.7622377
```

```
# scale
```

```
tmp[,1:4] <- scale(tmp[,1:4])
```

```
# Recompute the sd of each numeric column --> they are now identical and =1
```

```
apply(tmp[,1:4], 2, sd)
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
##           1           1           1           1
```

**limit the influence of high values** :  $\log$ ,  $\log(x+1)$  (`log1p`) and square root (`sqrt`) transformations are useful when the distribution of a variable is strongly asymmetric with few high values. In this situation these few values will overly influence the whole analysis and may mask interesting patterns in lower numbers. The shape of the distribution must be checked with histograms or density plots. The log transformation is stronger than the square root transformation. These transformations are often almost automatically applied on some types of data. For example a square root transformation is almost always beneficial on areas data.

Another typical reason for using these transformations is to linearise the relationship between the variables. This might be desirable in linear methods like PCA or distances based on linear relationships like correlation distances.

```
data(mite, package = "vegan")
```

```
# dev.new(width = 18/2.54, height = 5/2.54)
```

```
par(mfrow = c(1,4), mar = c(3,3,1,0.5), mgp = c(1.8, 0.6, 0), cex = 0.8, las = 1)
```

```
hist(mite[,1], main = "")
```

```
hist(sqrt(mite[,1]), main = "")
```

```
hist(mite[,1]^0.25, main = "") # fourth root = sqrt(sqrt(x))
```

```
hist(log1p(mite[,1]), main = "") # log1p(x) is equivalent to log(x+1)
```

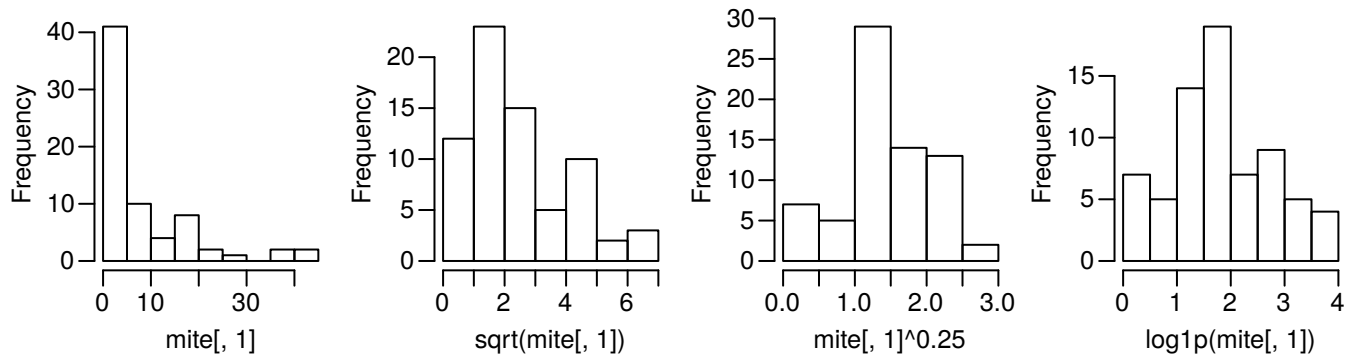


Figure 1:

```
# If you want to apply a log1p transformation to all species :
```

```
mite_sqrt <- sqrt(mite)
```

```
mite_log <- log1p(mite)
```

Quick and dirty graphical check of the effect of the log transformation on all species

```
library(reshape2)
```

```
mite_melt <- melt(mite)
```

```
mite_log <- melt(mite_log)
```

```
# dev.new(width = 8/2.54, height = 6/2.54)
```

```
ggplot(mite_melt, aes(x=value, group = variable)) +  
  geom_density()
```

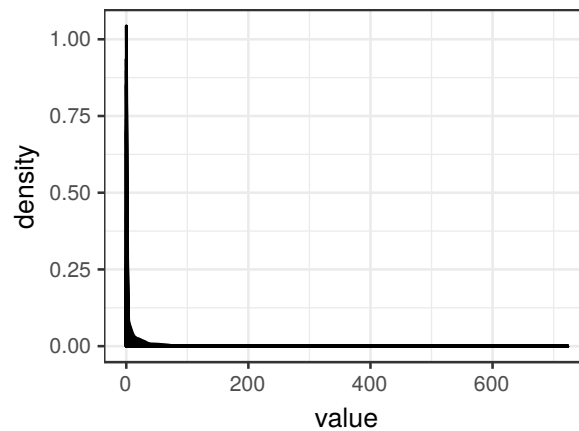


Figure 2:

```
ggplot(mite_log, aes(x=value, group = variable)) +  
  geom_density()
```

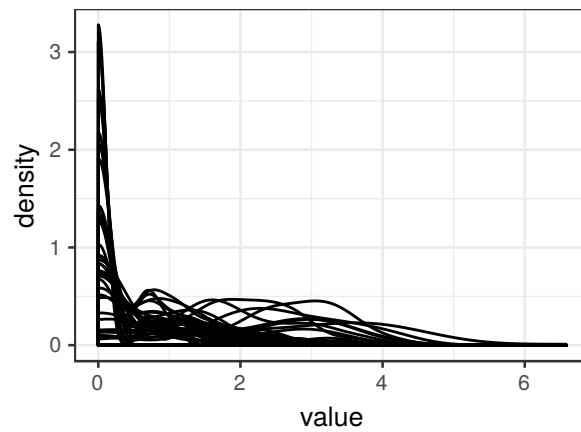


Figure 3:

### Transformation of ordinal data

It is frequent to have ordinal range of data for example with the Braun-Blanquet classes in vegetation science :  $r = 1$  individual ( $\sim 0.1\%$ ),  $+$  = a few individuals ( $\sim 0.2\%$ ),  $1 = [1-5\%]$ ,  $2 = [5-25\%]$ ,  $3 = [25-50\%]$ ,  $4 = [50-75\%]$ ,  $5 = [75-100\%]$ . There are some controversies about how use this kind of data but a pragmatic approach is either to use the classes as number directly or to use the mid point of the class (eg here :  $0.1\%$ ,  $0.2\%$ ,  $2.5\%$ ,  $15\%$ ,  $37.5\%$ ,  $62.5\%$ ,  $87.5\%$ ). With the last option, the resulting variable is often highly asymmetric and a log or square root transformation is often useful.

The dune dataset contains vegetation relevés using the “Van der Maarel” scale (0,1,2,3,4,5,6,7,8,9) which corresponds to an extended version of the Braun-Blanquet scale (0, r, +, 1, 2m, 2a, 2b, 3, 4, 5) with  $2m = 5\%$ ,  $2a = 5-12\%$  and  $2b = 12-25\%$  vegetation cover (see Van der Maarel 1979). The vander Maarel scale is often used “as is”, ie as a numerical scale but if you want to convert it into mid class intervals you can proceed as follows :

```
data(dune, package = "vegan")

# check the unique values (we have indeed a range of integers from 0 to 9:
sort(unique(unlist(dune)))

## [1] 0 1 2 3 4 5 6 7 8 9

tmp <- dune
for(i in 1:ncol(tmp)){
  # Transform each column into a factor with levels = 0,1,2,3,4,5,6,7,8,9
  tmp[,i] <- factor(tmp[,i], levels = 0:9)
  # convert these levels into the mid point of the classes
  levels(tmp[,i]) <- c("0", "0.1", "0.2", "2.5", "5", "8.5", "18.5", "37.5", "62.5", "87.5")
  # convert into numeric value.
  # NB : for factors you have to first convert into characters !!!
  tmp[,i] <- as.numeric(as.character(tmp[,i]))
}

dune[1:5, 1:10]
```

```
## Achimill Agrostol Airaprae Alop geni Anthodor Bellpere Bromhord Chenalbu Cirsarve Comapalu
## 1      1      0      0      0      0      0      0      0      0      0
## 2      3      0      0      2      0      3      4      0      0      0
## 3      0      4      0      7      0      2      0      0      0      0
```

```
## 4      0      8      0      2      0      2      3      0      2      0
## 5      2      0      0      0      4      2      2      0      0      0

tmp[1:5, 1:10]
```

```
##    Achimill Agrostol Airaprae Alopgei Anthodor Bellpere Bromhord Chenalbu Cirsarve Comapalu
## 1      0.1      0.0      0      0.0      0      0.0      0.0      0      0.0      0
## 2      2.5      0.0      0      0.2      0      2.5      5.0      0      0.0      0
## 3      0.0      5.0      0     37.5      0      0.2      0.0      0      0.0      0
## 4      0.0     62.5      0      0.2      0      0.2      2.5      0      0.2      0
## 5      0.2      0.0      0      0.0      5      0.2      0.2      0      0.0      0
```

**Binary transformation** : transform all values  $> 0$  into 1.

Typically usefull when you have many 0, some low abundance data and then a few very high abundance data. In that case log or sqrt are useless. Usefull also when the precision on the estimation of the abundance is very low. Of course you degrade your information but sometimes you obtain much more robust results.

```
# with base R
pa <- mite
pa[pa>0] <- 1

# with vegan
pa <- decostand(mite, method = "pa")

mite[1:5, 1:5]
```

```
##    Brachy PHTH HPAV RARD SSTR
## 1      17      5      5      3      2
## 2       2      7     16      0      6
## 3       4      3      1      1      2
## 4      23      7     10      2      2
## 5       5      8     13      9      0
```

```
pa[1:5, 1:5]
```

```
##    Brachy PHTH HPAV RARD SSTR
## 1       1      1      1      1      1
## 2       1      1      1      0      1
## 3       1      1      1      1      1
## 4       1      1      1      1      1
## 5       1      1      1      1      0
```

### dropping variables with 0 or low variances

Variables with no variability are useless and you can always drop them. But variables with very low variability will also bear almost no information and increase the complexity and noise in the data. A typical example are very rare species present in only one or 2 sites. Of course it is possible that this species are rare because they are indicator of particularly rare or preserved biotopes. But you simply don't have enough replicates to be certain of that and they will not really convey any usefull information. You also often have "tourist" or "vagrant" species that have been observed on a site while they are absolutely not associated with this kind of biotope. This is frequent and they will stringly increase the noise in the data and make the main patterns difficult to spot.

Look for example at the Orthoptera dataset. When you compute the total number of individuals observed for each species, it appears that "subu" and "tenu" (Tetrix subulata and Tetrix tenuicornis) have been counted only once. These are small species, difficult to identify and are generally not adult (and hence impossible to



identify) at the time the samples were done (However “undu” = *Tetrix undulata* is a species from the same group and was observed several time here maybe due to misidentification). Vaga, roes and albo are also very rare but vaga is known to be a good indicator species of very dry, rocky, chalk grasslands...

Note that the knowledge about the dataset and the ecology of the species is of hudge importance to make the right decision about the data transformation and analysis...

```
tmp <- read.csv2("data/orthops/orthops_misc/Orthops_Species.csv", row.names = 1)
colSums(tmp)
```

```
## bigu brun para vaga disp disc rufu punc bico brac roes sylv rufi oviri falc
## 119 21 359 2 81 48 506 10 31 33 2 184 58 46 165
## gris albo line bipu subu tenu undu tviri
## 100 3 59 21 1 1 24 32
```

We might decide in a first step to remove the species with less than 2 individuals observed in total :

```
tmp <- tmp[,colSums(tmp)>=2]
colSums(tmp)
```

```
## bigu brun para vaga disp disc rufu punc bico brac roes sylv rufi oviri falc
## 119 21 359 2 81 48 506 10 31 33 2 184 58 46 165
## gris albo line bipu undu tviri
## 100 3 59 21 24 32
```

#### 1.4.2 Specific standardizations frequently used in community ecology

(ie for sites x species datasets)

**species maximum standardisation** : divide each collumn by the maximum value of that collumn. All species have then a value between 0 and 1. This will give a much higher influence to rare species and will strongly reduce the dominance structure in the community. This might be usefull when rare species are of particular interest (indicator species) but the destruction of the dominance structure is often not desirable and the higher weight given on rare species might be problematic when you have a lot of “noise” species.

```
tmp <- decostand(mite, method = "max") # by default on the columns (MARGIN = 2)
summary(tmp[,1:5])
```

##	Brachy	PHTH	HPAV	RARD	SSTR
## Min.	:0.00000	Min. :0.0000	Min. :0.0000	Min. :0.00000	Min. :0.00000
## 1st Qu.:	0.07143	1st Qu.:0.0000	1st Qu.:0.1081	1st Qu.:0.00000	1st Qu.:0.00000
## Median	:0.10714	Median :0.0000	Median :0.1757	Median :0.00000	Median :0.00000
## Mean	:0.20782	Mean :0.1589	Mean :0.2301	Mean :0.09341	Mean :0.05238
## 3rd Qu.:	0.27976	3rd Qu.:0.2500	3rd Qu.:0.3243	3rd Qu.:0.07692	3rd Qu.:0.00000
## Max.	:1.00000	Max. :1.0000	Max. :1.0000	Max. :1.00000	Max. :1.00000

**Sample total standardization** : divide each line by its total. You are then working with the within site relative abundance. This is particularly usefull whan the sampling effort or the size of the plots is not identical. Some distance indices like the Hellinger and Chord distances use internally this transformation and has other intersting properties (see below)

```
tmp <- decostand(mite, method = "total") # by default on the rows (MARGIN = 1)
rowSums(tmp[1:5,]) # sum of the 5 first rows
```

```
## 1 2 3 4 5
## 1 1 1 1 1
```

**Wisconsin double standardisation** : species maximum standardisation followed by sample total standardisation. This transformation combined with the Bray-Curtis distance has often been shown to be one of the best way to represent unconstrained gradients with NMDS (see ?metaMDS in vegan package that computes this transformation automatically.)

```
tmp <- wisconsin(mite)
tmp2 <- decostand(mite, method = "max")
tmp2 <- decostand(tmp2, method = "total")
```

```
tmp[1:5, 1:5]
```

```
##           Brachy           PHTH           HPAV           RARD           SSTR
## 1 0.069454241 0.10724552 0.023188220 0.03959835 0.05719761
## 2 0.005422741 0.09964287 0.049244352 0.00000000 0.11387756
## 3 0.018417461 0.07251875 0.005226577 0.01487564 0.06446111
## 4 0.067150808 0.10729531 0.033141409 0.01886511 0.04087440
## 5 0.013561183 0.11391394 0.040023816 0.07886350 0.00000000
```

```
tmp2[1:5, 1:5]
```

```
##           Brachy           PHTH           HPAV           RARD           SSTR
## 1 0.069454241 0.10724552 0.023188220 0.03959835 0.05719761
## 2 0.005422741 0.09964287 0.049244352 0.00000000 0.11387756
## 3 0.018417461 0.07251875 0.005226577 0.01487564 0.06446111
## 4 0.067150808 0.10729531 0.033141409 0.01886511 0.04087440
## 5 0.013561183 0.11391394 0.040023816 0.07886350 0.00000000
```

### 1.4.3 Transformation based distances

Legendre & Gallagher(2001) have described a series of data transformations (Hellinger, Chord, chi-squared,...) that have interesting properties when combined with a Euclidean distance.

Take for example the Hellinger distance. You can obtain a Hellinger distance matrix by two ways :

1. Raw data → compute Hellinger distance → Hellinger distance matrix
2. Raw data → apply a “Hellinger transformation” → Hellinger transformed data matrix → compute the Euclidean distance → Hellinger distance matrix

The big advantage of this approach is that several widely used multivariate methods use internally the Euclidean distance (PCA, k-means, RDA,...). If you apply these transformations before applying these euclidean based distance the result will not be based on euclidean distance but on the other distances (Hellinger, Chord, Chi-squared) that might have very different properties and are generally more adapted to treat species data.

The chord transformation is simply the “Sample total standardization” discussed above : all abundances are divided by the total abundance of the line. The Hellinger transformation is the square root of the chord distance and gives less influence to the higher values.

```
# Hellinger transformation
# = sqrt of Sample total Standardization = sqrt of Relative abundance for each row
hellinger <- decostand(mite, "hellinger")

hellinger[1:4,1:4]
```

```
##          Brachy          PHTH          HPAV          RARD
## 1 0.34846603 0.1889822 0.18898224 0.14638501
## 2 0.08638684 0.1616150 0.24433889 0.00000000
## 3 0.14664712 0.1270001 0.07332356 0.07332356
## 4 0.28358346 0.1564466 0.18698940 0.08362420
```

```
sqrt(mite/rowSums(mite))[1:4,1:4]
```

```
##          Brachy          PHTH          HPAV          RARD
## 1 0.34846603 0.1889822 0.18898224 0.14638501
## 2 0.08638684 0.1616150 0.24433889 0.00000000
## 3 0.14664712 0.1270001 0.07332356 0.07332356
## 4 0.28358346 0.1564466 0.18698940 0.08362420
```

```
# Chord transformation
```

```
chord <- decostand(mite, "normalize")
```

```
# Chi-Squared transformation
```

```
chisquared <- decostand(mite, "chi.square")
```

Computing a euclidean distance on these transformed datasets will provide a Hellinger, chord or chi squared distance matrix instead of the real Euclidean distance matrix.

## 1.5 Overview of the different similarity/disimilarity measures

Legendre & Legendre (2012) describe ~ 50 different measures of similarity/disimilarity/distance. We present here only a selection of these indices that are the most used in practice.

To add some complexity the same index can have different names in the literature and sometimes the same name designate in fact different indices... To clarify the situation Legendre & Legendre (2012) designate the indices by a number like S1 (similarity indices) or D14 (distance indices) that we are using here too. It remains however useful to have an idea of the most common names of the indices.

Many of these indices are identical when they are applied after some data transformation (for example the Bray-Curtis distance applied on binary data is equivalent to the Sorensen disimilarity). Many of these indices are also very similar and redundant.

The choice of the “right” index will depend on the answers to the following questions :

- Q mode vs R mode : do I want to compare the columns (variables) or the lines (observations) of the dataset ?
  - Q mode analysis = ressemblance between observations/objects/sites
  - R mode analysis = ressemblance between variables/descriptors/species
- Do I want that double 0 increase the similarity (symetric indices) or do I want to ignore double 0 (asymetric indices) (typically for Species data you want to ignore the double 0) ?
- Are the variables quantitative, presence/absence, qualitative or mixed ?

Legendre & Legendre insist that some indices can be applied both on the observations and variables (for example Chisquared and Jaccard) but other indices should be restricted to one use only. For example they strongly argue against the use of the pearson correlation coefficient on the observations (this index should be used only on the variables).

The double 0 problem is detailed below. Typically, for Species data (and similar type of data) you don’t want that the shared absence of species makes two sites more similar. The classical Euclidean distance is typically a symetric index were the shared absence will make two sites more similar and it is there fore not

recommended to use it for Species data along with methods typically based on Euclidean distance like PCA, RDA and k-means clustering. Some data transformation (chord, Hellinger, Chi-squared transform) can make species data compatible with Euclidean distance.

Depending on the replies to these questions the indices can be organized as follows. NB : There are many other indices available. These ones are the most frequently used. The S or D followed by a number correspond to the numerotation of Legendre & Legendre (2012).

- Q mode : distance between rows/sites/observations,...
  - Asymetric distances : double 0 are ignored (typically with Species data)
    - \* Quantitative data (eg Species abundances)
      - Bray-Curtis dissimilarity (D14)
      - Chord distance (D3) = Euclidean distance on normalized data
      - Hellinger distance (D17) = Euclidean distance on sqrt(proportions)
      - Chi Squared distance (D21)
    - \* Presence/absence data
      - Jaccard (S7) : proportion of species present on both sites (relative to the total species present in these two sites)
      - Sorensen (S8) : equivalent to applying a Bray-Curtis distance on binary data
      - Ochiai (S14) : Chord and Hellinger are related to this index
      - Chord distance (D3) = Euclidean distance on normalized data
      - Hellinger distance (D17) = Euclidean distance on sqrt(proportions)
      - Chi Squared distance (D21)
    - \* Mix of qualitative and binary and or quantitative data
      - Asymetric version of the Gower similarity (S19) : very similar to S15 (below) but uses the Jaccard index (S7) for binary data instead of the Simple matching coefficient (S1)
  - Symetric distances : double 0 indiquate similarity
    - \* Quantitative data
      - Euclidean distance (D1) - needs standardisation if the data are not homogenous
    - \* Binary data
      - Simple Matching Coefficient (S1) =  $\text{nbr double 0} + \text{double 1} / \text{nbr data}$
    - \* Qualitative + mixed type data
      - Gower similarity (S15) : As S19 but uses S1 for binary data internally
- R mode : = correlations/covariances - “dependence” matrix between the collumns/species/variables/descriptors
  - Asymetric distances : double 0 are ignored (typically with Species data)
    - \* Quantitative Species data :
      - Chi square distance
      - 1 - Pearson, Spearman or Kendall correlation coefficients on chord/Hellinger transformed data
    - \* Presence/absence data
      - same indices as for the Q mode : Jaccard, Sorensen, Ochiai, etc...
  - Symetric distances : double 0 indiquate similarity
    - \* Quantitative and ordinal data :
      - Covariance for homogenous data
      - 1 - Pearson, Spearman or Kendall correlation coefficients for heterogenous data
    - \* Presence/absence data
      - Pearson correlation , called point correlation coefficient, related to chi squared

NB : Some similarity coefficients should ideally be transformed into distance by  $\sqrt{1-S}$  instead of simply  $1-S$  to avoid negative eigenvalues in MDS and other problems for their representation in 2 dimensions (with MDS). This is done directly by `ade4` but not by `vegan` (ie `vegdist`).

## 1.6 Q mode : distances between observations (rows)

### 1.6.1 Asymetric indices for quantitative data (typically : Species abundances)

Hellinger distance gives less importance to abundant species than Chord distance (because of the square root transformation involved) and Chi squared distance gives less importance to abundant species than the Hellinger distance. Otherwise said the rare species have an increasing relative importance in the similarity like this : Chord < Hellinger < Chi squared.

Bray-Curtis does not work on relative abundances like the others so an increase of 5 individuals will have the same effect if you have 5 or 1000 individuals... Data are therefore often  $\log(x+1)$  transformed first (log1p function) to avoid an over emphasis on the abundant species. Another common transformation before applying the Bray-Curtis distance is the double “Wisconsin” transformation.

Hellinger seems to be a good compromise in many cases and can be used in a wide range of applications...

```
# Bray-Curtis distance
speBC <- vegdist(spe) # default

# Chord distance
# Euclidean distance on a chord transformed dataset = Chord distance
speChord <- dist(decostand(spe, "normalize"))

# Hellinger distance
# Euclidean distance on a Hellinger transformed dataset = Hellinger distance
speHellinger <- dist(decostand(spe, "hellinger"))

# Chi squared distance
# Euclidean distance on a Chi squared transformed dataset = Chi squared distance
speChisq <- dist(decostand(spe, "chi.square"))
```

### 1.6.2 Asymetric indices for binary data (typically : Species presence/absence)

Jaccard and Sorensen similarity are typical indices for such data.

Jaccard ( $S_7$ ) =  $a / a+b+c$  where a is the number of species present in both sites (double 1), b is the number of species present in site 1 but not in site 2 (1,0) and c is the reverse (0,1). This is the proportion of species that are present on both sites.

Be careful that this is the formula of the Similarity index while vegdist is providing a dissimilarity ie 1-S. So the value provided by vegdist is the proportion of species that are present in one of the two sites only...

Sorensen is the Bray-Curtis index applied to a binary dataset and is  $= 2a / 2a+b+c$ .

It can be shown that Chord and Hellinger distances applied on binary data are transformations of the Ochiai index which has been developed for binary data so these distances can also be used for binary data. Chord and Hellinger distances are very highly correlated to the Jaccard index (see below) so they can be interpreted in a similar way.

Chi squared distances have also been used a lot for binary data (ie within CA). However its tendency to give more weight to rare species has some times been seen as a problem (see box 9.2 p 480 L&L).

```
# transform the abundances into binary data
spebin <- spe
```

```

spebin[spebin>1] <- 1

# You can also use the vegan function to obtain exactly the same result
spebin <- decostand(spe, "pa")

# Jaccard distance
spebinJaccard <- vegdist(spebin, "jaccard")

# Sorensen distance
spebinSorensen <- vegdist(spebin) # Default Bray-Curtis on Presence-Absence = Sorensen

# Chord distance
# Euclidean distance on a chord transformed dataset = Chord distance
spebinChord <- dist(decostand(spebin, "normalize"))

# Hellinger distance
# Euclidean distance on a Hellinger transformed dataset = Hellinger distance
spebinHellinger <- dist(decostand(spebin, "hellinger"))

# Chi squared distance
# Euclidean distance on a Chi squared transformed dataset = Chi squared distance
spebinChisq <- dist(decostand(spebin, "chi.square"))

```

We can also compute the Euclidean distance on the Species just to compare with the other coefficients but this is something that you should normally avoid (double 0 problem...)

```
speEucl <- dist(spe)
```

### 1.6.3 Comparison of the asymmetric indices in Q mode

Comparison of the different distances used for Species data (without double 0) in R mode analysis. The Euclidean distance (including double 0) is indeed quite different from the other methods that ignore the double 0. For binary data the Hellinger distance is almost equal to the widely used Jaccard index. The advantage is that with Hellinger transformation you can directly use it in PCA, K-means, ...

```

d <- list(speBC, speChord, speHellinger, speChisq, speEucl,
          spebinJaccard, spebinSorensen, spebinChord, spebinHellinger, spebinChisq)
d <- sapply(d, as.vector)
colnames(d) <- c("speBC", "speChord", "speHellinger", "speChisq", "speEucl",
                "spebinJaccard", "spebinSorensen", "spebinChord", "spebinHellinger",
                "spebinChisq")

# dev.new(18/2.54, 12/2.54)
pairs2(d, pt.cex = 0.7 )

```

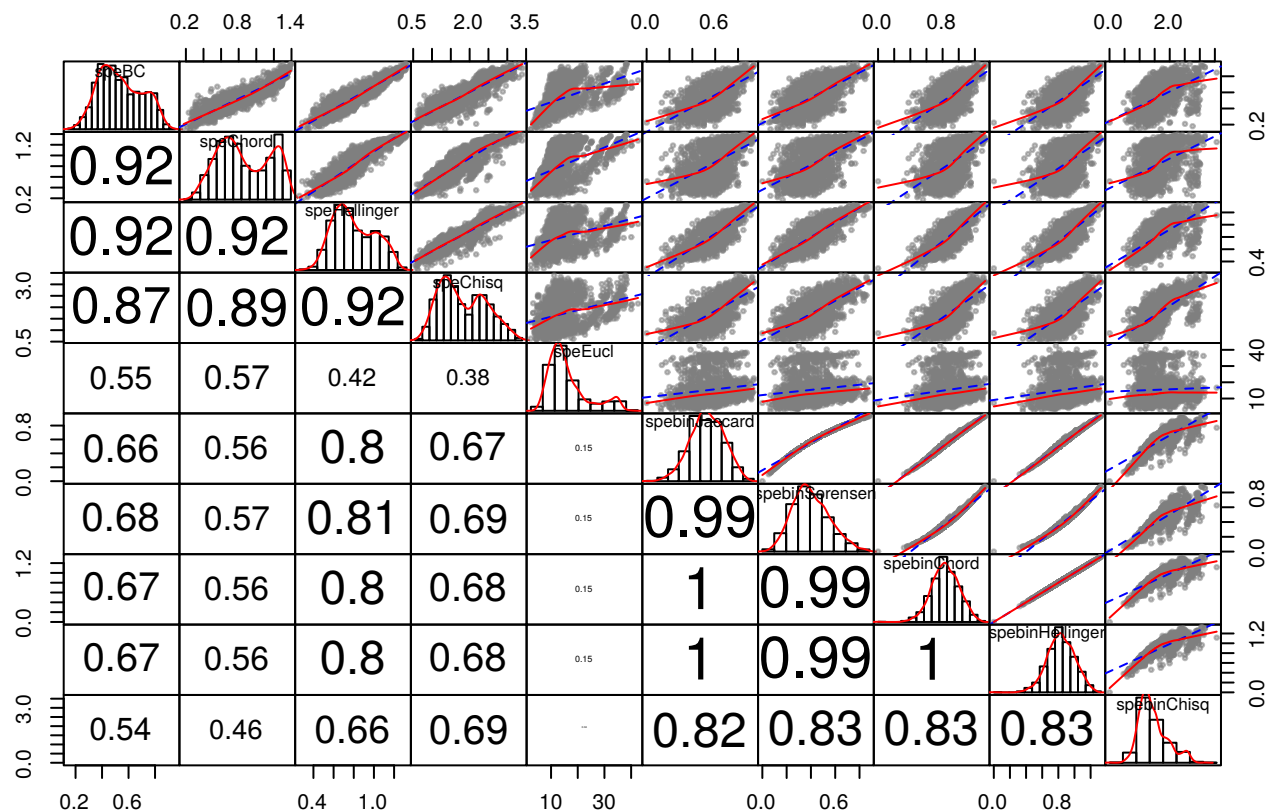


Figure 4:

#### 1.6.4 Symetric indices for quantitative data (typically : Environmental variables)

The Euclidean distance is typically used in this case. It can be applied only on homogenous variables (ie with the same measurement units). If the data are not homogenous, you need first to standardize it (subtract column mean and divide by the standard deviation : function `scale` in R). If the data are homogenous you can choose to not standardise the data. In that case the variables with the highest values will dominate the Euclidean distance calculation. If you standardise on the other hand, each variable will have an equal weight in the global distance measure.

It can also be used on projected geographic coordinates (not on longitude - latitude !!!) to compute the geographic distance between locations.. NB : Do not standardize geographic coordinates !.

```
#Euclidean distance after standardisation of the data
envEucl <- dist(scale(env))
# Euclidean distance without standardisation on the geographic coordinates
geoDist <- dist(sites[,c("X", "Y")])
```

#### 1.6.5 Symetric indices for binary data

Simple Matching Coefficient = Sokal & Michener coefficient

$$S1 = (0,0) + (1,1) / (0,0) + (1,1) + (1,0) + (0,1)$$

$$S1 = (a+d)/(a+b+c+d)$$

NB : ade4 provides `sqrt(1-S1)`

```
# summary(env)
envbin <- env[,c("PctBare", "Old", "Shadow", "Exp")]
```



```
envbin[envbin>1] <- 1
envbin[envbin<1] <- 0

envbinS1 <- ade4::dist.binary(envbin, method = 2)^2
```

### 1.6.6 Indices for qualitative data and mixed types

The Gower index is typically used when you have a mix of qualitative, binary and or quantitative data (see Legendre & Legendre 2012 p 278-280 for a very clear and detailed description). A separate similarity value is computed separately for each variable in a way that depends of the type of the variable. Then the different similarities values computed for each variable are averaged to obtain the global value. It can also handle missing value by simply ignoring the variable when the value is missing for a given observation.

The different type of data are treated as follows :

- Qualitative variables : the Simple matching rule is used - if the two elements share the same level the similarity receives the value 1 and it is 0 if the levels are different.
- Binary variables :
  - S15 uses the simple matching coefficient (symetric index so the double 0 increase similarity)
  - S19 uses the Jaccard index (asymetric index ignoring double 0)
- Quantitative variables : sum of absolute value of the difference (like Manhattan dist) divided by the maximum difference between 2 observations So it is a sort of Manhattan distance standardized to have a maximum distance of 1.
- Ordinal : by default treated as numeric but 2 other possibilities exist (implemented in `FD::gowdiss`)

This gower index has in fact many options ... You have to carefully read the help and evaluate what options are used by the function you use.

Warning !! the `vegdist(x, method = "gower")` works only for binary and quantitative variables not for qualitative ones. Use `cluster::daisy()` or `FD::gowdis` instead. The last one is more flexible : it can also optionally compute the asymmetrical S19 version, handles ordinal variables in 3 different ways, add weights,....

Create a fake dataset with 4 type of variables

```
tmp <- data.frame(fac = factor(c("Yellow", "Red", "Black", "Yellow", "Red", "Black")),
                 ord = factor(c("1", "2", "3", "3", "2", "1"), ordered = TRUE),
                 bin = c(1,0,1,0,1,0),
                 num = 1:6)

tmp
```

```
##      fac ord bin num
## 1 Yellow  1   1   1
## 2   Red  2   0   2
## 3 Black  3   1   3
## 4 Yellow  3   0   4
## 5   Red  2   1   5
## 6 Black  1   0   6
```

To obtain exactly the same result with both functions, you need to specify for `FD::gowdiss` that ordered factors should be treated as numeric values with the argument `ord = "classic"`. This is the symetric version of the gower index "S15".

```
# NB : for cluster::daisy, Gower distance is the default if you don't have only
# numeric variables
cluster::daisy(tmp, metric = "gower")
```

```
## Dissimilarities :
##      1      2      3      4      5
## 2 0.675
## 3 0.600 0.675
## 4 0.650 0.475 0.550
## 5 0.575 0.400 0.475 0.675
## 6 0.750 0.575 0.650 0.600 0.675
##
## Metric : mixed ; Types = N, 0, I, I
## Number of objects : 6
```

```
FD::gowdis(tmp, ord = "classic")
```

```
##      1      2      3      4      5
## 2 0.675
## 3 0.600 0.675
## 4 0.650 0.475 0.550
## 5 0.575 0.400 0.475 0.675
## 6 0.750 0.575 0.650 0.600 0.675
```

Lets compute the dissimilarity between line 1 and 2 manually. We compute the similarity for each variable separately and with a different index for each type of variable

```
fac <- tmp$fac[1] == tmp$fac[2] # here = FALSE = 0 because the value is not identical
bin <- tmp$bin[1] == tmp$bin[2] # simple matching for binary data
num <- 1-(abs(tmp$num[1] - tmp$num[2]) / (max(tmp$num) - min(tmp$num)))
# ordinal value treated as a continuous value
ord <- as.numeric(tmp$ord)
ord <- 1-(abs(ord[1] - ord[2]) / (max(ord) - min(ord)))

# Compute the average and reverse to obtain a dissimilarity
1-((fac + bin + num + ord) / 4)

## [1] 0.675
```

Create a fake purely binary dataset to see how these binary variables are treated.

```
tmp <- as.data.frame(rbind(
  A = c(0,0,1,1,1),
  B = c(0,0,0,0,1),
  C = c(1,1,1,1,1),
  D = c(1,1,0,0,0)))
```

S15 symmetric Gower index : binary variables dissimilarity are identical to the single matching coefficient (S1)

```
cluster::daisy(tmp, "gower")
```

```
## Dissimilarities :
##      A      B      C
## B 0.4
## C 0.4 0.8
```

```
## D 1.0 0.6 0.6
##
## Metric : mixed ; Types = I, I, I, I, I
## Number of objects : 4
```

```
FD::gowdis(tmp)
```

```
##      A      B      C
## B 0.4
## C 0.4 0.8
## D 1.0 0.6 0.6
```

```
ade4::dist.binary(tmp, method = 2)^2 # single matching coefficient S1
```

```
##      A      B      C
## B 0.4
## C 0.4 0.8
## D 1.0 0.6 0.6
```

To obtain the asymmetrical S19 version of the index (equivalent to Jaccard index for binary data) you need 1) that the binary variables are numeric with 1 and 0 values only (for FD:gowdiss) 2) to specify which variables should be treated as asymmetric (ignoring double 0)

```
cluster::daisy(tmp, "gower", type = list(asymm = 1:5))
```

```
## Dissimilarities :
##           A           B           C
## B 0.6666667
## C 0.4000000 0.8000000
## D 1.0000000 1.0000000 0.6000000
##
## Metric : mixed ; Types = A, A, A, A, A
## Number of objects : 4
```

```
FD::gowdis(tmp, asym.bin = 1:5 )
```

```
##           A           B           C
## B 0.6666667
## C 0.4000000 0.8000000
## D 1.0000000 1.0000000 0.6000000
```

```
vegdist(tmp, method = "jaccard") # Jaccard index
```

```
##           A           B           C
## B 0.6666667
## C 0.4000000 0.8000000
## D 1.0000000 1.0000000 0.6000000
```

## 1.7 R mode : dependance between variables (columns)

The name “R” mode comes from the Pearson correlation coefficient “R” which is one of the most widely used for comparison of variables. However in Pearson and equivalent rank correlation coefficients, the double 0 contribute to the increase of the correlation which is not always a desired behavior.

Note that the correlation coefficient (comprised between -1 and 1) is transformed into distance by taking

1-R so that a value of 0 indicates cases with completely negative correlation , value of 1 indicates completely independent data and a value of 1 indicates perfectly correlated data.

The choice among the type of correlation should be made based on the expected relationship between the variables. Pearson R will measure linear relationships while rank based correlation coefficients like Spearman and Kendal will measure monotonic relationships that does not need to be linear. Chi square distance will measure non monotonic association for example for categorical data.

### 1.7.1 Asymmetric indices for quantitative data (typically : Species abundances)

Correlation coefficients are sometimes used on row data but species that are absent from the same sites are going to be considered as similar. So the R in this case can't be interpreted as an index of co-occurrence/co-abundance.

Legendre proposed to use classical correlation coefficients on Hellinger or Chord transformed data to decrease this problem of double 0.

You can also compute the chi squared distance on the transposed matrix as this index has been developed for contingency matrices that are transposable. In our case the result of the chi square distance is very different from the other approaches that provide similar results. The chi-square seems to be of lower interest because it just separates the 3 rarest species (albo, roes, vaga) from the other species... The analysis could be repeated after removing these rare species.

When using correlation coefficients, all species have the same weight in the analysis while for species community data, you typically want to see the structure due to the dominant species. This can be done by using the covariance instead of the correlation.

I'm not certain how to transform correlations and covariances into a distance matrix...

```
# Chisquared distance on transposed species matrix
RspeChi <- dist(decostand(t(spe), "chi.square"))

# Pearson on untransformed data : not recommended !
# (This is just to compare with the other indices)
RspePearson <- as.dist(1-cor(spe))

# Pearson on Chord transformed data
RspePearsonchord <- as.dist(1-cor(decostand(spe, "norm"))))

# Pearson on Hellinger transformed data
RspePearsonHellinger <- as.dist(1-cor(decostand(spe, "hellinger"))))

# Pearson on Chi squared transformed data
RspePearsonChi <- as.dist(1-cor(decostand(spe, "chi.square"))))

# Covariance on Hellinger transformed data
# The standardisation used here is maybe hazardous and this should probably
# not be used in practice...
RspeCovHellinger <- cov(decostand(spe, "hellinger"))
RspeCovHellinger <- as.dist(abs(RspeCovHellinger-max(RspeCovHellinger)))

# Spearman correlation on Hellinger transformed data
RspeSpearmanHellinger <- as.dist(1-cor(decostand(spe, "hellinger"), method = "spearman"))
```

According to Yari Oksanen, for species clustering, most dissimilarity indices do a bad job when comparing rare and common species, but Raup-Crick makes sense :

```
RspeRaup <- vegdist(t(spe), "raup")
```

Comparison of the different distances for Species data in R mode analysis. The Chi squared distance is completely different from the others. The covariance relationship is curved at the extremes (higher similarities or dissimilarities). Basically the distances based on covariances are much more contrasted between sites, similar sites are considered as more similar and distant sites are considered as more distant than with the correlation.

```
d <- list(RspePearson, RspePearsonchord, RspePearsonHellinger, RspeSpearmanHellinger,
          RspePearsonChi, RspeChi, RspeCovHellinger, RspeRaup)
d <- sapply(d, as.vector)
colnames(d) <- c("Pearson", "Pearson Chord", "Pearson Hellinger", "Spearman Hellinger",
                "Pearson Chi", "Chi sq dist", "Cov Hellinger", "Raup-Crwick")
```

```
# dev.new(18/2.54, 12/2.54)
pairs2(d, pt.cex = 0.7 )
```

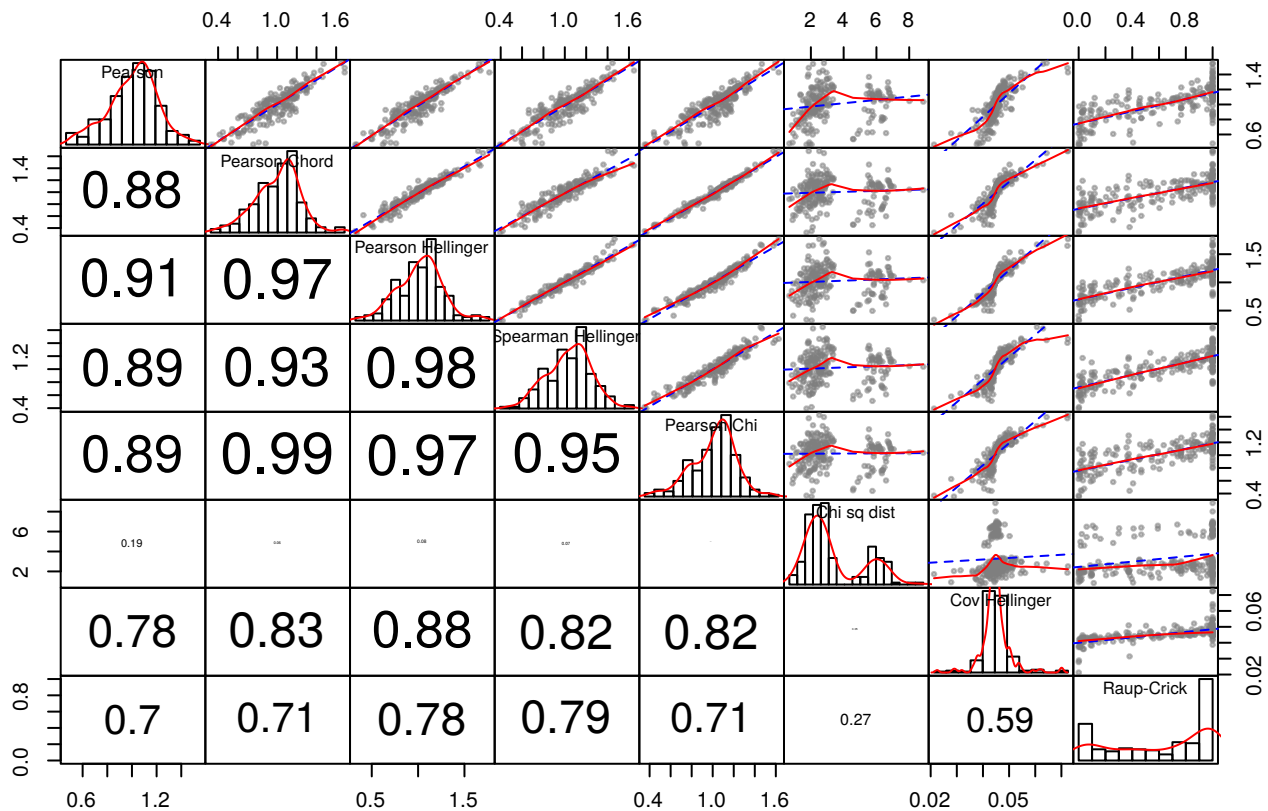


Figure 5:

### 1.7.2 Asymetric indices for binary data (typically : Species presence/absence)

You can use the same indices as for the Q mode. To do that you must first transpose the presence/absence matrix with the function `t()`. We show here only an example for 3 indices.

```
# transform the abundances into binary data
spebin <- spe
spebin[spebin>1] <- 1
```

```

# You can also use the vegan function to obtain exactly the same result
spebin <- decostand(spe, "pa")

# Jaccard distance
spebinJaccard <- vegdist(t(spebin), "jaccard")

# Sorensen distance
spebinSorensen <- vegdist(t(spebin)) # Default Bray-Curtis on Presence-Absence = Sorensen

# Hellinger distance
# Euclidean distance on a Hellinger transformed dataset = Hellinger distance
spebinHellinger <- dist(decostand(t(spebin), "hellinger"))

```

### 1.7.3 Symetric indices for quantitative, ordinal and binary data

The pearson, Spearman and Kendal correlation coefficients can be applied here and then transformed into distances with `as.dist(1-R)`. The resulting distance index is 0 for perfectly positively correlated variables, 1 for uncorrelated variables and 2 for perfectly negatively correlated variables. These correlation coefficients are meaningful even when applied on binary data (point correlation coefficient) if you accept that double 0 increase the similarity. The pearson correlation measures the linear correlation and is sensitive to outliers. Data transformation (log, sqrt, ...) can often improve linearity and should be chosen based on graphical exploration or prior knowledge (eg typically area data are sqrt transformed).

The Spearman and Kendal correlation distance are less sensitive to outliers and measure monotonic (always increasing or decreasing) but not necessarily linear relationships.

The Euclidean distance is not cited by Legendre & Legendre but frequently used (eg in genomics along with the correlation distance for genes). The Euclidean distance based on the scaled variables is in fact strictly equivalent to the square root of the Pearson correlation distance. If the units of the variables are the same and that you don't want to give the same weight to each variable, you might want to use the Euclidean distance on unscaled data.

```

# Pearson correlation distance
envPearson <- as.dist(1-cor(env))

# Spearman Correlation distance
envSpearman <- as.dist(1-cor(env, method = "spearman"))

# Kendal Correlation distance
envKendal <- as.dist(1-cor(env, method = "kendal"))

# Euclidean distance on scaled data
envEucl <- dist(t(scale(env)))^2

```

Comparison of the 4 indices

```

d <- list(envPearson, envSpearman, envKendal, envEucl)
d <- sapply(d, as.vector)
colnames(d) <- c("Pearson", "Spearman", "Kendal", "Euclidean^2")

# dev.new(18/2.54, 12/2.54)
pairs2(d, pt.cex = 0.7 )

```

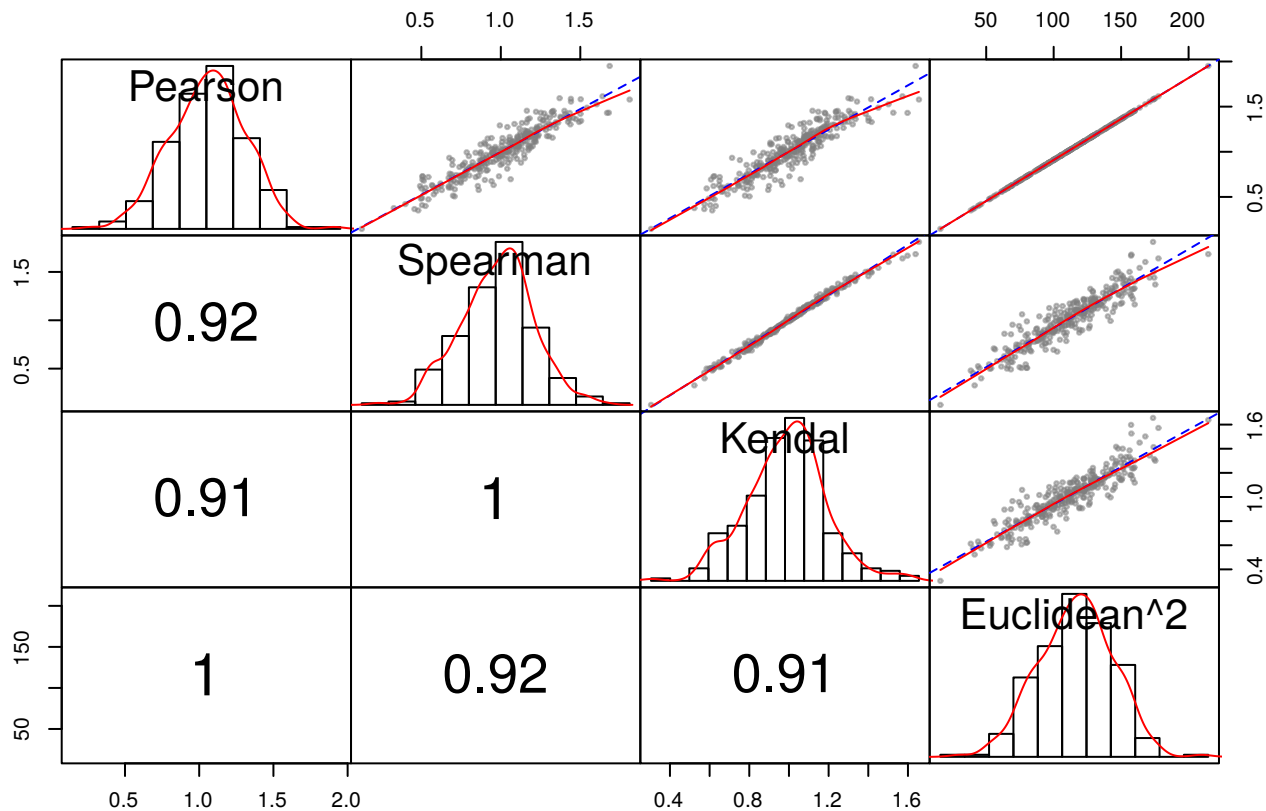


Figure 6:

## 1.8 Typical problems when measuring distances

### 1.8.1 Illustration of the double 0 problem

When you chose a distance index you have to decide if you want that a 0 value in both observations (double 0) will make your two observations more similar (symetric indices like Euclidean distance) or if these double 0 should be ignored or their impact minimized (asymetric indices, like Brat-Curtis, Hellinger, Jaccard,...)

We have 3 sites (a, b and c) with 12 abundances of tree species. Site b is a pure spruce stand (*Picea abies*) with no other species, site c is a site with only poplars (*populus* sp.) and site a is a mixed forest including poplars and spruce.

```
d <- rbind(a = rep(100, 12), b = 0, c = 0)
d[1:2,1] <- 100
d[c(1,3),12] <- 100

colnames(d) <- c("Picea", "Pseudotsuga", "Larix", "Pinus",
                "Quercus", "Fagus", "Fraxinus", "Carpinus", "Betula", "Salix", "Sorbus",
                "Populus")

d
```

```
##   Picea Pseudotsuga Larix Pinus Quercus Fagus Fraxinus Carpinus Betula Salix Sorbus Populus
## a   100         100   100   100    100   100    100    100    100   100   100    100
## b   100          0     0     0     0     0     0     0     0     0     0     0
## c     0          0     0     0     0     0     0     0     0     0     0    100
```

In this situation all these forests are quite different but you expect that site b will be considered as more

similar to site a than site c because site b and c have no species in common. However, if you compute a simple euclidean distance, c and b are considered to be closer due to the shared 0.

```
dist(d)
```

```
##           a           b
## b 331.6625
## c 331.6625 141.4214
```

```
# graphical representation of the distances
```

```
mds <- cmdscale(dist(d))
```

```
plot(mds, type = "n", asp = 1, main = "Euclidean dist", xlab = "", ylab = "")
```

```
text(mds, labels = row.names(d))
```

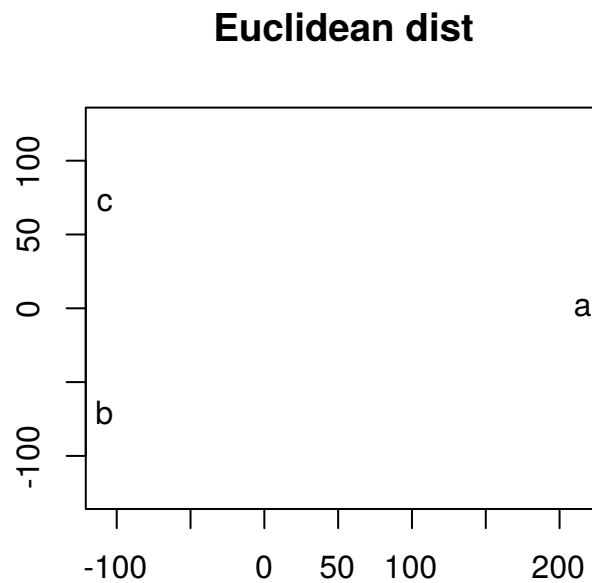


Figure 7:

The fact that double 0 should increase or not the similarity is really dependent on the type of data or questions. In the following example, the data are exactly the same but now the data represent the rainfall for each month of the year in 3 different sites. Here you expect that b is closer to c because the absence of rainfall during 10 months of the year is really a similarity. So the Euclidean distance is adapted here.

```
d <- rbind(a = rep (100, 12), b = 0, c = 0)
```

```
d[1:2,1] <- 100
```

```
d[c(1,3),12] <- 100
```

```
colnames(d) <- paste0("prec", 1:12)
```

```
d
```

```
##  prec1 prec2 prec3 prec4 prec5 prec6 prec7 prec8 prec9 prec10 prec11 prec12
## a   100   100   100   100   100   100   100   100   100   100   100   100
## b   100    0    0    0    0    0    0    0    0    0    0    0
## c    0    0    0    0    0    0    0    0    0    0    0   100
```

```
dist(d)
```

```
##           a           b
## b 331.6625
```



```
## c 331.6625 141.4214
```

```
mds <- cmdscale(dist(d))
plot(mds, type = "n", asp = 1, main = "Euclidean dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))
```

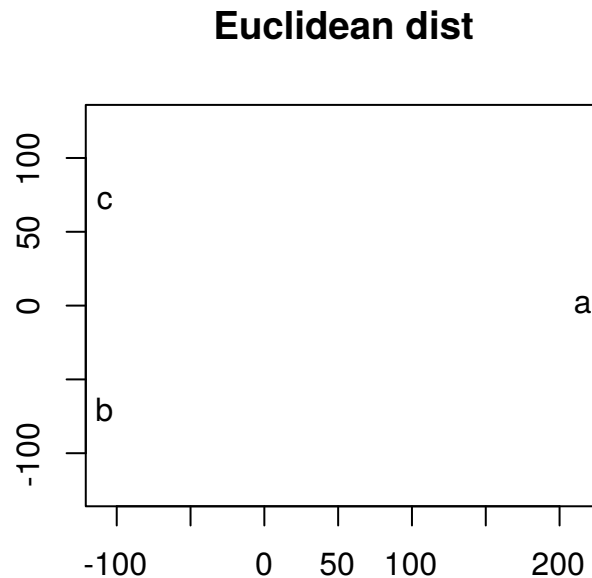


Figure 8:

This kind of problem is more frequent with species composition data with lots of 0 (and particularly when the sum of the rows are very different). Take the following artificial example in which with 3 sites a, b and c and presence/absence (1/0) of 30 species (columns). This dataset could also be seen as 3 observations/replicates a, b and c with 30 peptides, gene expression etc. . .

For species data you expect that site b is more similar to site a than site c because site b and a have 10 species in common while site b has only one species in common with site c.

Next we compute several distances indices between sites/observations and make a graphical representation of these distances (with Metric Dimentional Scaling also called Principal Coordinate Analysis). With a PCA, you will obtain exactly the same plot as the plot done with Euclidean distance (exepted maybe that the positive/negative values might be reversed)

```
d <- rbind(
  a = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
  b = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
  c = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)
)
library(vegan)

# w11(width = 15/2.54, height = 10/2.54)
par(mfrow = c(2,3), mar = c(2,2,2,1))

mds <- cmdscale(dist(d))
plot(mds, type = "n", asp = 1, main = "Euclidean dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(vegdist(d, method="jaccard"))
```

```

plot(mds, type = "n", asp = 1, main = "Jaccard", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(vegdist(d, method="bray"))
plot(mds, type = "n", asp = 1, main = "Sorensen", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "hellinger")))
plot(mds, type = "n", asp = 1, main = "Hellinger dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "normalize")))
plot(mds, type = "n", asp = 1, main = "Chord dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

# mds <- cmdscale(dist(decostand(d, "total")))
# plot(mds, type = "n", asp = 1, main = "Profile dist", xlab = "", ylab = "")
# text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "chi.square")))
plot(mds, type = "n", asp = 1, main = "Chi Square dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

```

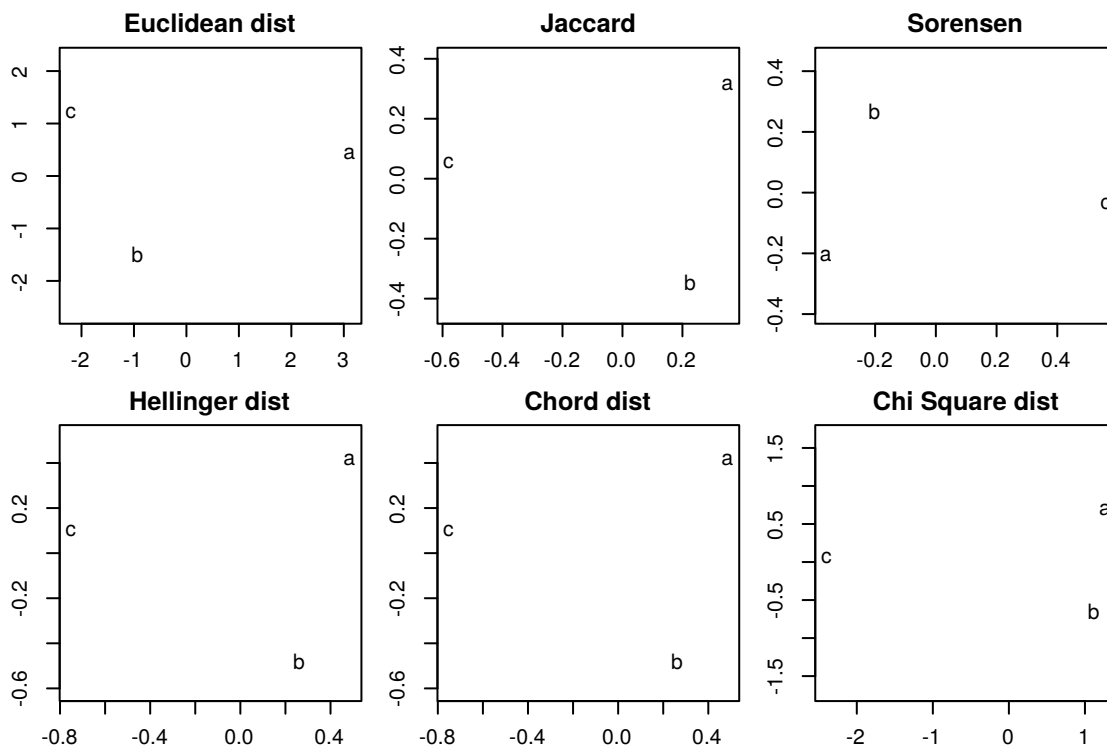


Figure 9:

```

par("usr") + 0.1*par("usr")

```

```
## [1] -2.800762 1.563950 -2.011134 1.980552
```

You can see that with the Euclidean distance b is closer to c than a while it is the reverse for all other

distance measures. This is because for Euclidean distance the absence of the 20 first species (or peptides) is considered as an indication of similarity while these “double zeroes” are ignored with other the distances (or their importance is strongly decreased). The Euclidean distance is a “symetric distance” while the other ones are “asymetric distances”.

The Jaccard distance is simply the proportion of different values after removing the columns with double 0. Jaccard dist =  $1 - (a / (a+b+c))$  where a is the number of species present in both sites (double 1), b is the number of species present in site 1 but not in site 2 (1,0) and c is the reverse (0,1).

```
vegdist(d, method="jaccard")
```

```
##           a           b
## b 0.6666667
## c 0.9666667 0.9000000
```

```
# a vs b
20/30
```

```
## [1] 0.6666667
```

```
# a vs c
29/30
```

```
## [1] 0.9666667
```

```
# b vs c
9/10
```

```
## [1] 0.9
```

The Sorensen distance gives more weight to similarities (1,1 values) than to dissimilarities (0,1 or 1,0 values) : Sorensen dist =  $1 - (2a / (2a+b+c))$  and still ignores the double 0

```
vegdist(d, method="bray") # Bray Curtis distance on binary data is equivalent to Sorensen
```

```
##           a           b
## b 0.5000000
## c 0.9354839 0.8181818
```

```
# a vs b
1-(2*10/(2*10 + 20))
```

```
## [1] 0.5
```

```
# a vs c
1-(2*1/(2*1 + 29))
```

```
## [1] 0.9354839
```

```
# b vs c
1-(2*1/(2*1 + 9))
```

```
## [1] 0.8181818
```

The other distances have the peculiarity that they can be computed either directly or the same values can be obtained by transforming the data and then calculate the euclidean distance on the transformed data. The advantage of the transformation approach is that you can use these asymmetric distances ignoring double zeroes in classical euclidean methods and methods aiming to produce groups with lowest variance like PCA, k-means,...

For example, the profile transformation is just dividing each line by the sum of the line. So we are working on the relative “abundance” (= profile) of the species. The Hellinger distance is just the square root of this transformation, giving less weight to dominant species (). The chi square distance is a little bit more complex. This is the distance used in Correspondance Analysis. The Chord distance is a fourth possibility not represented on the graphs.

```
# Profile
round(dist(decostand(d, "total")),1)
```

```
##      a      b
## b 0.3
## c 1.0 0.9
```

```
round(dist((d/rowSums(d))),1)
```

```
##      a      b
## b 0.3
## c 1.0 0.9
```

```
# Hellinger
round(dist(decostand(d, "hellinger")),1)
```

```
##      a      b
## b 0.9
## c 1.3 1.2
```

```
round(dist(sqrt(d/rowSums(d))),1)
```

```
##      a      b
## b 0.9
## c 1.3 1.2
```

## Conclusion

The Jaccard index is really the most intuitive and probably the best adapted here (for binary data). The Hellinger transformation provides results that are very similar to Jaccard with the advantage of being usable for example in a PCA or if you want to measure correlation coefficients between the species,...

## 1.8.2 Scale problems with Euclidean and Bray-Curtis distances

In the following toy dataset, we have 3 sites and 3 species. A priori it seems obvious that site b is closer to site a than to site c. Indeed there are no species in common between b and c and b shares the exact same species than site a. The only difference is their abundance that is much more important in site a, maybe because site a is bigger for example or has higher densities.

```
d <- rbind(
  a = c(0,20,23),
  b = c(0, 2, 2),
  c = c(1,0,0))
pander(d)
```

<b>a</b>	0	20	23
<b>b</b>	0	2	2
<b>c</b>	1	0	0

```
# x11(width = 15/2.54, height = 10/2.54)
par(mfrow = c(2,3), mar = c(2,2,2,1))

mds <- cmdscale(dist(d))
plot(mds, type = "n", asp = 1, main = "Euclidean dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "hellinger")))
plot(mds, type = "n", asp = 1, main = "Hellinger dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "normalize")))
plot(mds, type = "n", asp = 1, main = "Chord dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(vegdist(d, method="bray"))
plot(mds, type = "n", asp = 1, main = "Bray-Curtis", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "total")))
plot(mds, type = "n", asp = 1, main = "Profile dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "chi.square")))
plot(mds, type = "n", asp = 1, main = "Chi Square dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))
```

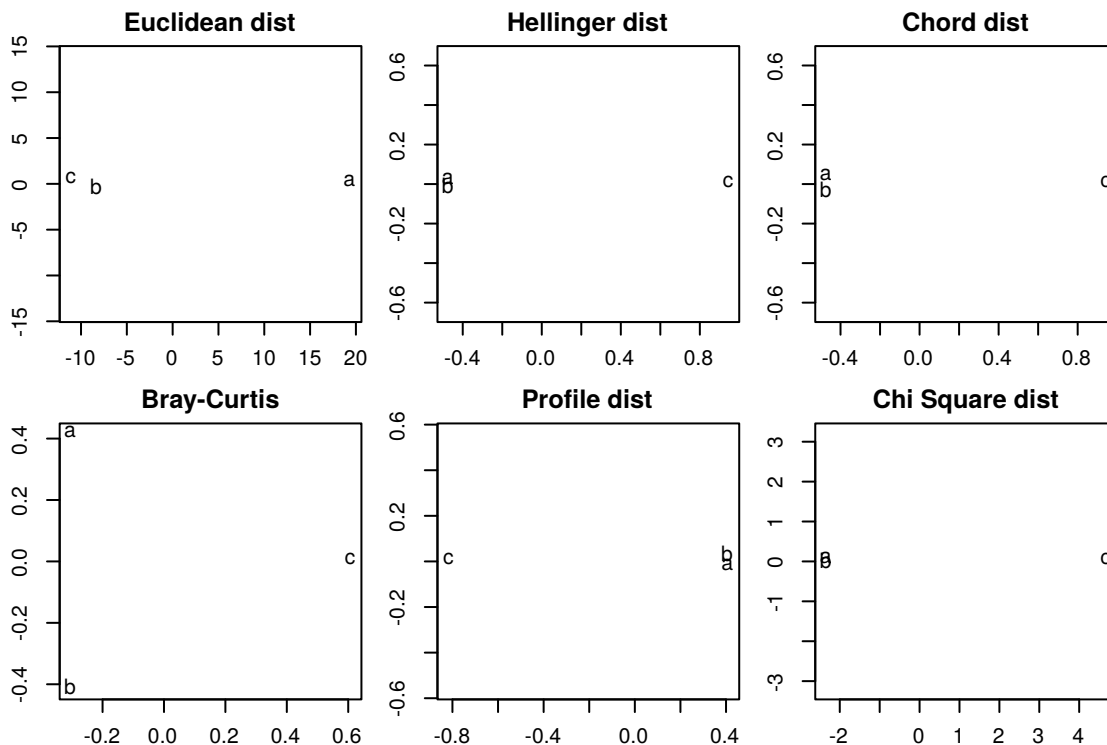


Figure 10:

However the Euclidean distance will consider that b is much closer to c than to a because the difference between site b and c is only 1,2,2 while the difference between b and a is 0, 18, 21...

The Bray-Curtis distance that is quite widely used for species abundance data is also scale independent (ie it will consider that a difference between 0 and 5 individuals is the same as the difference between 1000 and 1005 individuals). In our case, it considers that the 3 sites are equidistant. (NB : Bray-Curtis distance ignores the double 0).

The four other asymmetric indices provide the expected behavior. Chord, Hellinger and profile distances are simply working on relative abundances. This has nice properties but you lose some information...

In PCA you can work on scaled data (correlation matrix). Does this solve the problem? The pattern is less misleading than before but it is still present. NB : Bray-Curtis distance on scaled data makes no sense. Scaling almost does not change the other measures here. For the 4 transformation based distances we first do the transformation then we scale then we calculate the distance. This is the natural workflow in a PCA for example.

```
# x11(width = 15/2.54, height = 10/2.54)
par(mfrow = c(2,3), mar = c(2,2,2,1))

mds <- cmdscale(dist(scale(d)))
plot(mds, type = "n", asp = 1, main = "Euclidean dist on scaled data", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(scale(decostand(d, "hellinger"))))
plot(mds, type = "n", asp = 1, main = "Hellinger dist with scaling", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(scale(decostand(d, "normalize"))))
plot(mds, type = "n", asp = 1, main = "Chord dist with scaling", xlab = "", ylab = "")
```

```

text(mds, labels = row.names(d))

mds <- cmdscale(vegdist(scale(d), method="bray"))
plot(mds, type = "n", asp = 1, main = "Bray-Curtis on scaled data", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(scale(decostand(d, "total"))))
plot(mds, type = "n", asp = 1, main = "Profile dist with scaling", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(scale(decostand(d, "chi.square"))))
plot(mds, type = "n", asp = 1, main = "Chi Square dist with scaling", xlab = "", ylab = "")
text(mds, labels = row.names(d))

```

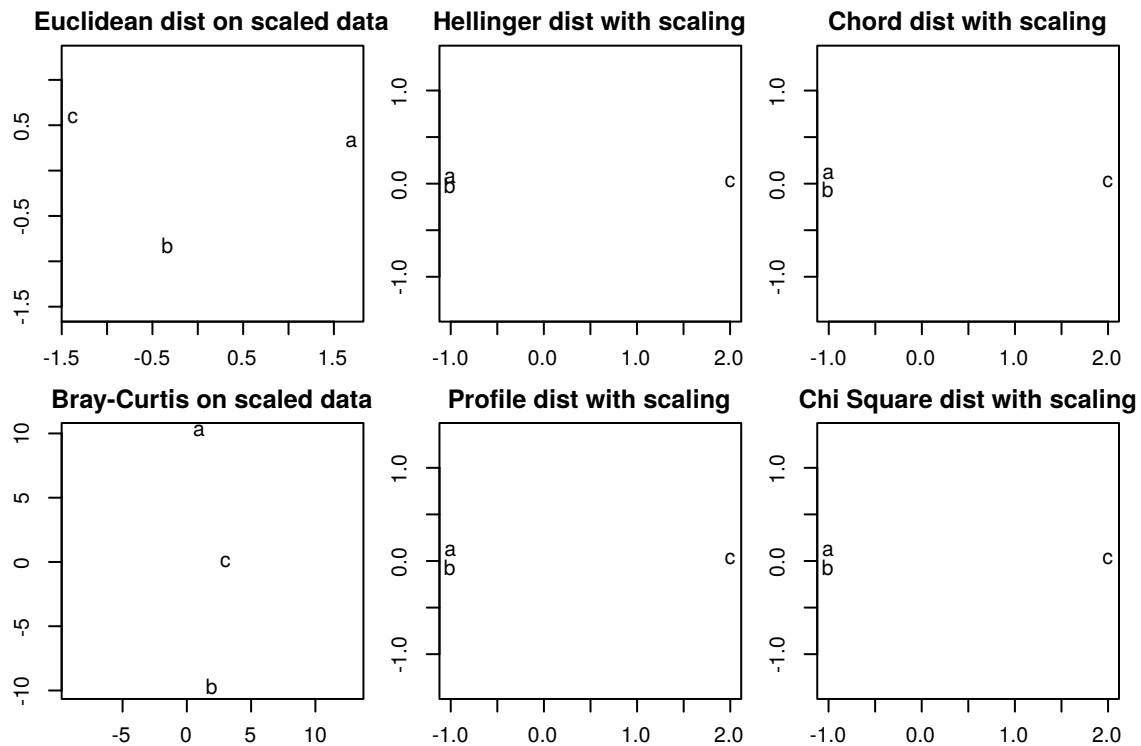


Figure 11:

### 1.8.3 Potential problems with Hellinger and Chord distance

As you work on relative abundances with Hellinger, you lose some information. For example if you have the following configuration, with Hellinger all sites will be identical (distance = 0) while with Bray-Curtis, b will be closer to a than to c. With Hellinger you are only looking at the relative species composition. Whether this is a problem or not depends on the question and the type of data...

Distances based on relative abundances might also be problematic when you have many sites with only one species. In that case the differences of abundances will be ignored and all the sites with the same species will be considered as perfectly identical even if the abundance is different. This is frequent when the “species” measured are in fact chemical compounds doses (eg pesticides).

```
d2 <- rbind(
  a = c(100,200,300),
  b = c(105, 205, 305),
  c = c(1,2,3))
pander(d2)
```

<b>a</b>	100	200	300
<b>b</b>	105	205	305
<b>c</b>	1	2	3

```
round(dist(decostand(d2, "hellinger")),1)
```

```
##    a b
## b 0
## c 0 0
```

```
vegdist(d2, method="bray")
```

```
##           a           b
## b 0.01234568
## c 0.98019802 0.98067633
```

Another property of hellinger transformed data that does not look very good - a priori - is that the abundance of a species in the transformed dataset will be dependent on the values of other species.

For example consider the following case where the abundance of the 3 first species is exactly the same on the 3 sites while the abundance of the fourth species is changing.

```
d2 <- rbind(
  a = c(10,20,30,1000),
  b = c(10,20,30,100),
  c = c(10,20,30,10))
pander(d2)
```

<b>a</b>	10	20	30	1000
<b>b</b>	10	20	30	100
<b>c</b>	10	20	30	10

When you do the Hellinger transformation the abundances of the 3 first species are no more identical. This might seem to be not very good



```
pander(decostand(d2, "hellinger"))
```

<b>a</b>	0.09713	0.1374	0.1682	0.9713
<b>b</b>	0.25	0.3536	0.433	0.7906
<b>c</b>	0.378	0.5345	0.6547	0.378

However when you calculate the distances, you obtain something closer to Bray-Curtis than to Euclidean distance...

```
round(dist(decostand(d2, "hellinger")),1)
```

```
##      a      b
## b 0.4
## c 0.9 0.5
```

```
round(vegdist(d2, method="bray"),1)
```

```
##      a      b
## b 0.7
## c 0.9 0.4
```

```
# x11(width = 20/2.54, height = 5/2.54)
par(mfrow = c(1,4), mar = c(2,2,2,1))
```

```
mds <- cmdscale(dist((d2)))
plot(mds, type = "n", asp = 1, main = "Euclidean dist", xlab = "", ylab = "")
text(mds, labels = row.names(d2))
```

```
mds <- cmdscale(vegdist((d2), method="bray"))
plot(mds, type = "n", asp = 1, main = "Bray-Curtis dist", xlab = "", ylab = "")
text(mds, labels = row.names(d2))
```

```
mds <- cmdscale(dist((decostand(d2, "hellinger"))))
plot(mds, type = "n", asp = 1, main = "Hellinger dist", xlab = "", ylab = "")
text(mds, labels = row.names(d2))
```

```
mds <- cmdscale(dist((decostand(d2, "total"))))
plot(mds, type = "n", asp = 1, main = "Profile dist", xlab = "", ylab = "")
text(mds, labels = row.names(d2))
```

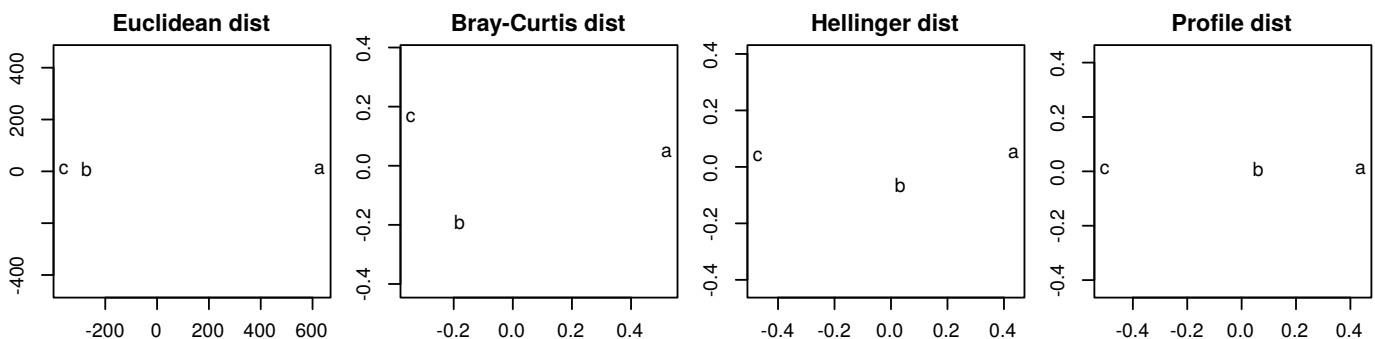


Figure 12:

## **Conclusions**

Depending on the type of data and which kind of information you want to study, the Euclidean and Bray-Curtis distance can be very misleading or show exactly what you want...

### 1.8.4 Potential problems with the Chi Squared distance

The problem with Chi Squared distance is that it will tend to show the very particular cases and give a strong weight to rare species that are present only at a few sites (and might for example have been observed by chance). (NB the chi squares distance is the distance used in correspondance analysis).

In the next exemple we have clearly 3 groups of sites : 1) a+b, 2) c+d, 3) e+f. Sites e and f have only one very rare species that is absent elsewhere. Sites a and b have mainly species 3 then less of species 2 and then even less of the species 1. For sites c and d the pattern is reversed.

All measure distances tend to clearly separate the 3 groups excepted the chi square distance that tend to form 2 groups and mask the differences between sites ab and cd.

NB : a,b and cd might be separated in other dimensions but it is not the case here.

Again depending on the question this might be a problem or an advantage...

```
d <- rbind(
  a = c(10,100,1000, 0),
  b = c(15, 113, 942, 0),
  c = c(1000,100,10, 0),
  d = c(1012,115,15, 0),
  e = c(0,0,0,2),
  f = c(0,0,0, 6))
pander(d)
```

<b>a</b>	10	100	1000	0
<b>b</b>	15	113	942	0
<b>c</b>	1000	100	10	0
<b>d</b>	1012	115	15	0
<b>e</b>	0	0	0	2
<b>f</b>	0	0	0	6

```
# x11(width = 15/2.54, height = 10/2.54)
par(mfrow = c(2,3), mar = c(2,2,2,1))
```

```
mds <- cmdscale(dist(d))
plot(mds, type = "n", asp = 1, main = "Euclidean dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))
```

```
mds <- cmdscale(dist(decostand(d, "hellinger")))
plot(mds, type = "n", asp = 1, main = "Hellinger dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))
```

```
mds <- cmdscale(dist(decostand(d, "normalize")))
plot(mds, type = "n", asp = 1, main = "Chord dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))
```

```
mds <- cmdscale(vegdist(d, method="bray"))
plot(mds, type = "n", asp = 1, main = "Bray-Curtis", xlab = "", ylab = "")
text(mds, labels = row.names(d))
```

```
mds <- cmdscale(dist(decostand(d, "total")))
```

```

plot(mds, type = "n", asp = 1, main = "Profile dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

mds <- cmdscale(dist(decostand(d, "chi.square")))
plot(mds, type = "n", asp = 1, main = "Chi Square dist", xlab = "", ylab = "")
text(mds, labels = row.names(d))

```

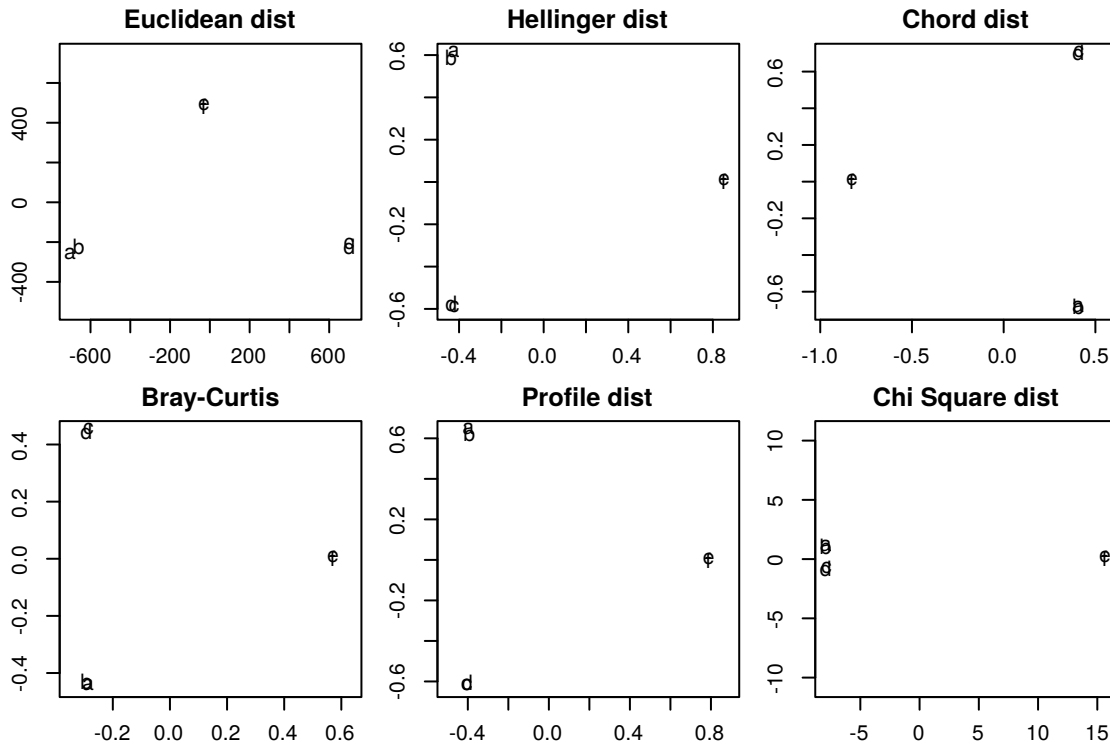


Figure 13:

We can also visualize this over influence of rare species on the Orthoptera dataset. We compute the distance between species based on the chi squared distance and the correlation distance after hellinger transformation.

```

# Chisquared distance on transposed species matrix
RspeChi <- dist(decostand(t(spe), "chi.square"))
# Pearson on Hellinger transformed data
RspePearsonHellinger <- as.dist(1-cor(decostand(spe, "hellinger")))

```

Visualisation of the Chi square distance. You can see that it mainly separates the 3 rarest species.

```

# dev.new(16/2.54, 8/2.54)
res <- cmdscale(RspeChi, eig=TRUE)
par(mfrow=c(1,2))
par(mar = c(3,3,2,1), mgp = c(2,0.8,0), cex = 0.75)
plot(res$points[,1], res$points[,2], asp=1, type = "n",
      xlab="PCo Axis 1", ylab= "PCo Axis 2", main="Chi squared distance")
abline(v=0, h=0, lty = 3)
text(res$points[,1], res$points[,2], labels = rownames(res$points), xpd = NA)

# Cluster
plot(hclust(RspeChi, method = "ward.D"))

```

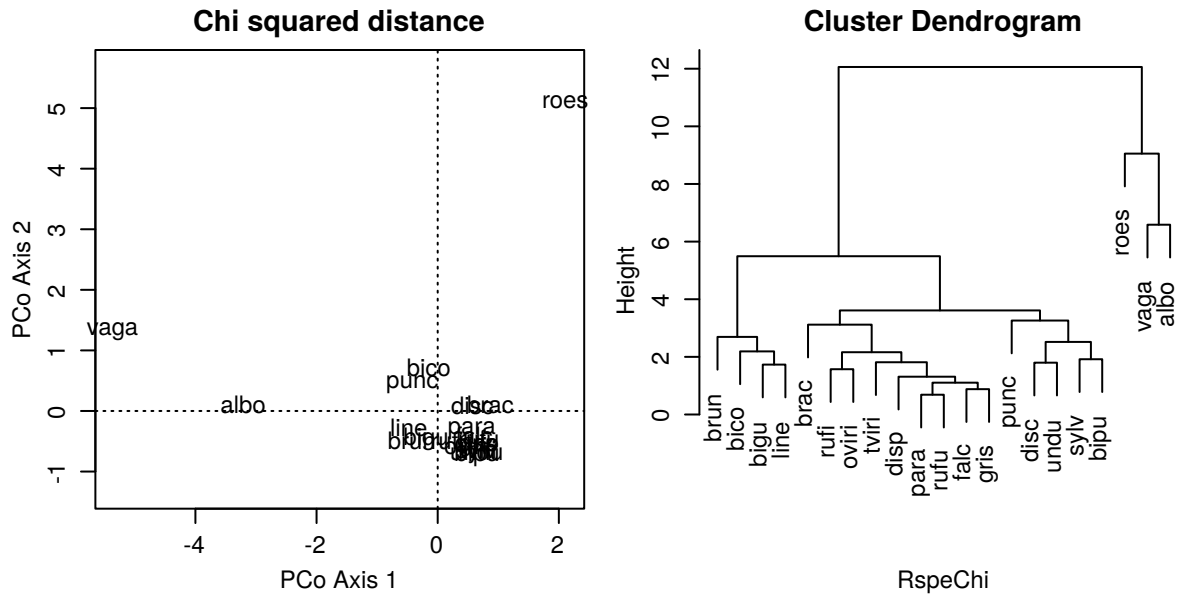


Figure 14:

Visualisation of the distance based on the correlation on hellinger transformed data : the 3 rare species do not dominate the graphs and this is often (but not always) what we want...

```
# dev.new(16/2.54, 8/2.54)
res <- cmdscale(RspePearsonHellinger, eig=TRUE)
par(mfrow=c(1,2))
par(mar = c(3,3,2,1), mgp = c(2,0.8,0), cex = 0.75)
plot(res$points[,1], res$points[,2], asp=1, type = "n",
      xlab="PCo Axis 1", ylab= "PCo Axis 2", main="Correlation on Hellinger transformation")
abline(v=0, h=0, lty = 3)
text(res$points[,1], res$points[,2], labels = rownames(res$points), xpd = NA)

# Cluster
plot(hclust(RspePearsonHellinger, method = "ward.D"))
```

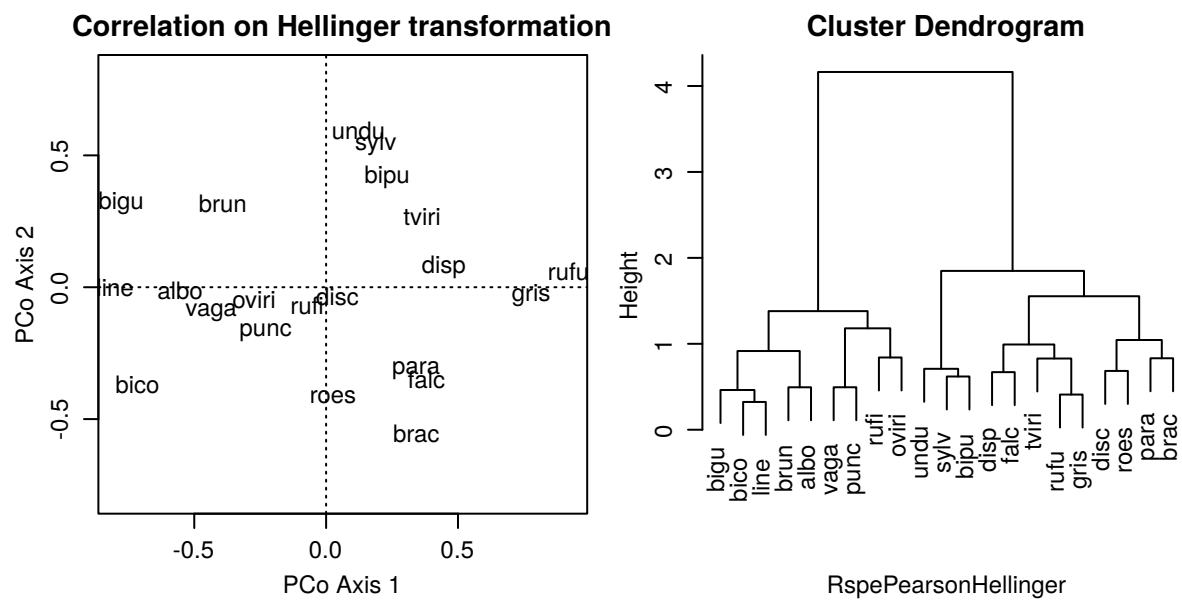


Figure 15: