

Ordination

Gilles San Martin

28 September 2018 - 08h38

Contents

1	Ordination	2
1.1	Overview of the different methods	2
1.2	PCA - Principal Component Analysis	4
1.2.1	Introduction	4
1.2.2	Typical PCA outputs and basic interpretation	6
1.2.2.1	PCA scores	6
1.2.2.2	PCA “eigenvalues”	8
1.2.2.3	PCA “eigenvectors”	9
1.2.3	PCA - in practice	13
1.2.3.1	Is the dataset adapted to PCA ?	13
1.2.3.2	Should we use scaled/standardised variables or not ?	14
1.2.3.3	How informative is the representation and which PC axes are important ?	15
1.2.3.4	How to represent the results ? How to interpret these representations ?	16
1.2.3.5	How to assess the quality of the representation of the variables in the reduced space? What is the contribution of each variable to the reduced space ?	19
1.2.3.6	How to assess the quality of the representation of the observations in the reduced space ?	20
1.2.3.7	The PCA graphs are overcrowded and difficult to read. What can we do ?	21
1.2.4	Example of PCA on the BIO worldclim variables	24
1.3	NMDS : Non Metric Multidimensional Scaling	37
1.3.1	Introduction	37
1.3.2	Computation and graphical representation	37
1.3.3	How many dimensions to choose for NMDS ?	39

```
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(FactoMineR)

# 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 Ordination

1.1 Overview of the different methods

Disclaimer note : the question of which method is the “best” for each use is generally highly controversial and many scientists disagree... We made some quite arbitrary choices here but you might forge your own opinion by testing the different methods on simulated datasets or reading the dedicated literature.

The 3 most important ordination methods used in ecology (and other disciplines) are :

- **PCA : Principal Component Analysis** - Arguably, the most widely used method in all disciplines. Uses the correlation or covariance between quantitative data and represents the Euclidean distance between the observations. But it is generally a very robust method that can be used on many different types of data (ordinal, binary, species abundance,...) particularly when combined with some specific transformations like the Hellinger transformation.
- **MDS : Metric Dimensional Scaling** also called (eg by Legendre et al) **PCoA : Principal Coordinates Analysis**. Can be used to obtain a graphical representation based on any distance matrix (generally of the observations/rows) in a reduced number of dimensions.
- **NMDS : Non Metric Multidimensional Scaling**. As MDS it can work on any distance matrix but instead of preserving the exact distance (in the reduced space) it tries to preserve the relative order between the observations in a few predetermined number of dimensions. If the aim is visualisation in few dimensions NMDS provides often a better solution than MDS.

CA, Correspondance Analysis, has also been (and still is) very popular in community ecology to perform ordination or species abundance or presence/absence datasets (sites x species data). It is an eigenvalue method that tends to preserve the chi-squared distance between the observations (rows) or between the species (columns). Historically the use of PCA on raw species data has been duly criticized, because of the double 0 problem and because PCA modelizes linear relationships (the species abundance can only increase or decrease along the PCA axes) while the abundance of species is often unimodal along ecological gradients (the species abundance increases up to an optimum then decreases). The CA with its chi-squared distance is arguably better adapted to visualize these unimodal distribution along unknown gradients. However the use of simple data transformation like the Hellinger transformation reduces some of these problems. In addition, the use of the chi-squared distance might have some drawbacks (too much weight on rare species) and you can obtain similar results by using a MDS on a chi-squared distance matrix (the results will however not be exactly identical to CA which uses internally different weights).

DCA, Detrended Correspondance Analysis is a modified version of CA that has also been widely used by community ecologists to remove a frequent artifact of ordination of species data with many 0, called “the arch or horseshoe effect”. However its use is very controversial and generally not recommended. The use of NMDS is generally accepted as a better solution to this problem with the added flexibility of the choice of any distance measure (popular choice : Bray-Curtis distance after double wisconsin transformation).

With PCA, MDS and NMDS in your toolbox you can deal with almost any situation you can encounter in a satisfactory manner. The advantage of MDS and NMDS is their great flexibility as they can be used on any distance matrix.

However other scientists approach the problem of ordination in a different, more case specific way. For example you could roughly summarize the approach proposed by the developers of the **FactoMineR** package (university of Renne , France) as follows :

- for quantitative variables : use PCA (based on Euclidean distance)
- for 2 qualitative variables (contingency table) : use CA (based on chi-squared distance)
- for >2 qualitative variables : use MCA - Multiple Correspondance Analysis (based on chi-squared distance)
- for Mixed qualitative and quantitative variables : use FAMD - Factor Analysis of Mixed Data (based on a combination Euclidean and chi-squared distance)

We will not follow this approach here but the approach provided by this package might be usefull in particular if you are dealing with a lot of qualitative data and that you feel limited by the Gower distance (combined with MDS or NMDS). In addition the graphical and statistical outputs of this package are generally well done and its documentation (including online courses) is truly excellent. You can combine it with the **factoExtra** package for even more flexibility in the graphical outputs.

Some methods are widely used in other disciplines but completely ignored in ecology for a reason unknown to me (e.g. Factor Analysis in Human Sciences).

Many algorithms have also been developped in the recent years but seems to remain little used in ecology. Their aim is often either to improve computation speed (because we are dealing with increasingly larger datasets) or to deal with complex non linear structures in the multidimensional space that are difficult to represent with simple projections (like PCA does) : incremental PCA, randomized PCA, kernel PCA, Locally Linear Embedding (LLE), t-Distributed Stochastic Neighbor Embedding (t-SNE),...

1.2 PCA - Principal Component Analysis

1.2.1 Introduction

The aims of PCA are :

- the representation of the (euclidean) distances between the observations (rows) in 2 (or a few) dimensions (ordination *per se*)
- the representation of the correlations or covariances between the columns/original variables
- the reduction of the number of dimensions : computing new uncorrelated axes (Principal Components) that summarize as best as possible the dataset (for example to use them as new explanatory variables in a regression)

This last objective is close to the objective of clustering when you want to create a “typology”. In clustering you simplify the dataset by creating discrete groups of observations i.e. the data are summarized by a qualitative variable. In PCA, the variables are summarized by a series of quantitative variables (the Principal Component Axes) that represent a decreasing proportion of the information contained in the original dataset.

PCA theory in a (very small) nutshell :

Each variable of the data table is seen as a dimension in which you can position the observations (rows). The PCA will create new variables (axes) that are a combination of the original axes (variables/columns). The first of these new axes is built to capture as much variation as possible from the original variables and the next axes are built to be uncorrelated and to capture a decreasing amount of the variation. You can also view PCA as a rotation of the original axes (variables) in order to capture as much variation as possible. . .

In the background, PCA is using either the covariance or the correlation matrix of the variables. It applies a mathematical tool called eigenvalue (or single value) decomposition to this matrix. As a result we obtain typically 3 groups of results : the scores, the eigenvalues and the eigen vectors that will be presented below in a short practical example. Original observations (rows) can be plotted on these new PC axes in a way that preserves as much as possible their original Euclidean distance. It is also frequent to add on the same graph a projection of the original axes to help to the interpretation of the results. The graph is then called a “biplot”.

Different PCA functions

Many different functions in R allow you to compute a PCA : `prcomp`, `princomp`, `vegan::rda`, `FactoMineR::PCA`, ... The `FactoMineR` function produces nice results and graphs but it is not easy to produce a “biplot” with it. The default graphical output of `vegan` are difficult to use but this function provides a lot of helper function to extract the information and produce nice biplots.

Note that different PCA functions in R or other softwares will often provide different results !! There are 3 main reasons to that fact :

1. The sign might be inversed on each PC axis (and also the graphical representation). This does not change anything for the interpretation.
2. There are several options that must be specified and the defaults are not the same in the different softwares : eg. should we work on the correlation or covariance matrix ? What kind of “scaling” do we want (type 1 for distance biplots, type 2 for correlation biplots, ...)? This might considerably change the interpretation !!
3. Even when you specify a desired scaling, there are small differences between the functions in the way they apply this scaling. In addition some functions (like `vegan::rda`) apply certain constants to the results in order to improve the graphical representation of the data. . . See for example here the `vegan` “design” vignette for a discussion about the different implementations This should have minor impact on the interpretation.

The second point is the most important. It is important to chose carefully the options and to describe what you did !!

1.2.2 Typical PCA outputs and basic interpretation

We can show the results of a PCA with a simple example on the iris data and the base R `prcomp` function. We will work here on the correlation matrix (option `scale = TRUE`)

```
pca <- prcomp(iris[,1:4, ], scale = TRUE)
```

Remember that you can explore the content of an object with `str`

```
# str(pca)
```

PCA outputs contain typically 3 important components : scores, eigenvalues and eigen vectors (or loadings)

1.2.2.1 PCA scores

The position of the observations (rows) on the new PC axes. The number of PC (Principal Component) axes is equal to the number of variables (or to the number of observations if it is smaller than the number of variables). We have here 4 PC axis and for each 150 scores because we have 150 observations in the dataset. We show here only the first ones :

```
head(pca$x)
```

```
##           PC1           PC2           PC3           PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

This information can be used to plot the observations in the reduced space. The 2 first axes should provide a better summary of the data than any pair of the original variables.

```
# dev.new(width = 8/2.54, height = 8/2.54)
par(mar = c(3.5,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
plot(pca$x[,1], pca$x[,2],
     pch = as.numeric(iris$Species), col = iris$Species,
     xlab = "PC axis 1", ylab = "PC axis 2")
legend("bottom", pch = 1:3, col = 1:3, legend = levels(iris$Species))
```

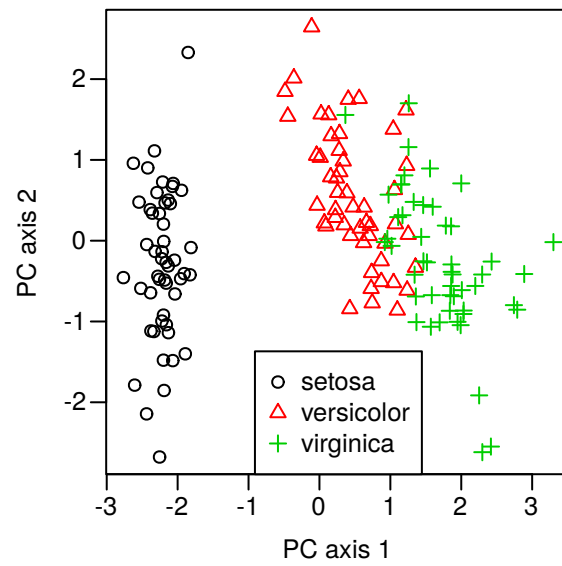


Figure 1:

1.2.2.2 PCA “eigenvalues”

One number of each PC axis (4 here). This strange number is simply the variance of the points (observations, rows) projected on each axis (ie the variance of the PCA “scores”). In `prcomp` they are expressed as standard deviation so they are squared here to obtain the TRUE eigenvalues.

```
eig <- pca$sdev^2
eig
```

```
## [1] 2.91849782 0.91403047 0.14675688 0.02071484
```

If you compute the variance of each column of the PCA scores matrix, you obtain indeed the same result...

```
apply(pca$x, 2, var)
```

```
##          PC1          PC2          PC3          PC4
## 2.91849782 0.91403047 0.14675688 0.02071484
```

The sum of the eigenvalues is simply the sum of the variances of the original axes. Because we chose the option `scale = TRUE`, the variables have been scaled, i.e. divided by their standard deviation. Each variable has then a sd and variance = 1. The sum of these variance is then 4 because we have 4 variables

```
sum(eig)
```

```
## [1] 4
```

We can check that if we use an unscaled dataset, the sum of the eigenvalues is equal to the sum of the variances of the original dataset

```
tmp <- prcomp(iris[,1:4, ], scale = FALSE)
sum(tmp$sdev^2)
```

```
## [1] 4.572957
```

```
sum(apply(iris[,1:4], 2, var))
```

```
## [1] 4.572957
```

The eigenvalues are often represented as a % of the total variance. Their value is decreasing for each new PC axis. Here the first PC axis represent ~72% of the original variance (of the scaled dataset), the second PC axis represent ~22% of the variance, etc. Together the 2 first PC axes summarize ~94% of the variance. Basically the PC axes 3 and 4 are useless here to have a correct representation of the data.

```
eig*100/sum(eig)
```

```
## [1] 72.9624454 22.8507618 3.6689219 0.5178709
```

Typically these values are represented on a barplot called a “screeplot”.

```
# dev.new(width = 6/2.54, height = 5/2.54)
par(mar = c(2,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
barplot(eig*100/sum(eig), names.arg = paste0("PC", 1:4), ylab = "% of the variance")
```

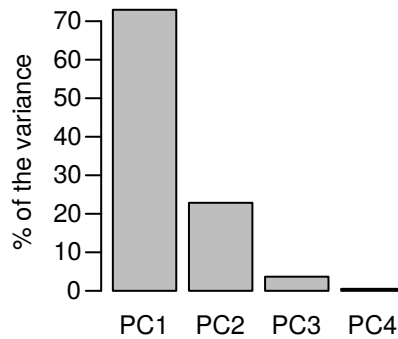



Figure 2:

1.2.2.3 PCA “eigenvectors”

Also called “loadings” or “rotation” = the “position” of the original variables on the PC axes.

```
pca$rotation
```

```
##              PC1              PC2              PC3              PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

They are typically used in 3 ways

1 - to interpret the PC axes

The higher the value of the eigenvectors, the higher their contribution to this axis. This why they are sometimes called “loadings”. For example here PC1 has the highest values for Sepal.Length, Petal.Length and Petal.Width. Points on the right along this axis (positive loadings) will be the points with higher values for these variables. PC2 is mainly dominated by Sepal.Width. Points on the left on this axis (negative loading) will tend to have high values of Sepal.Width.

Remeber that in different PCA functions the signs of the PC axes might be reversed and the graphs can then be inversed but this will not change the interpretation.

2 - to compute the scores from the original data

(you never do that in practice but it is helpfull to understand the meaning of these numbers).

These eigenvectors provide the information you need to project the original observations (rows) on the new PC axes :

```
# scores of the first observation = position of the first obesrvation of the first axes
pca$x[1,]
```

```
##              PC1              PC2              PC3              PC4
## -2.25714118 -0.47842383  0.12727962  0.02408751
```

```
# original values of the first observation (after scaling)
val <- scale(iris[,1:4])[1,]
val
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width
##   -0.8976739    1.0156020   -1.3357516   -1.3110521
```

```
# recompute these scores with the eigenvectors/loadings and the original (scaled) dataset
sum(pca$rotation[, "PC1"] * val)
```

```
## [1] -2.257141
```

```
sum(pca$rotation[, "PC2"] * val)
```

```
## [1] -0.4784238
```

```
# you can easily do that for all values with a matricial product
head(scale(iris[,1:4]) %*% pca$rotation )
```

```
##           PC1          PC2          PC3          PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

3 - to produce a biplot

To make a graphical representation of the original variables along with the observations in a graph called a “biplot”. The projection of the tips of the arrows on the axes helps to interpret these axes and the PCA plot.

```
# dev.new(width = 10/2.54, height = 10/2.54)
par(mar = c(3.5,3.5,2,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
plot(pca$x[,1], pca$x[,2],
     pch = as.numeric(iris$Species), col = iris$Species,
     xlab = "PC axis 1", ylab = "PC axis 2", main = "Distance PCA biplot")
abline(v=0, h=0, lty=2, col = "gray80")
cst <- 2.5 # arbitrary multiplicative constant to improve the graphical representation
arrows(x0 = 0, y0= 0, x1 = pca$rotation[, "PC1"]*cst, y1 = pca$rotation[, "PC2"]*cst,
       col = "gray50", length = 0.1, angle = 20 )
text(x = pca$rotation[, "PC1"]*cst, y = pca$rotation[, "PC2"]*cst,
     labels = row.names(pca$rotation),
     col = "gray50", adj = 0)
legend("topright", pch = 1:3, col = 1:3, legend = levels(iris$Species))
```

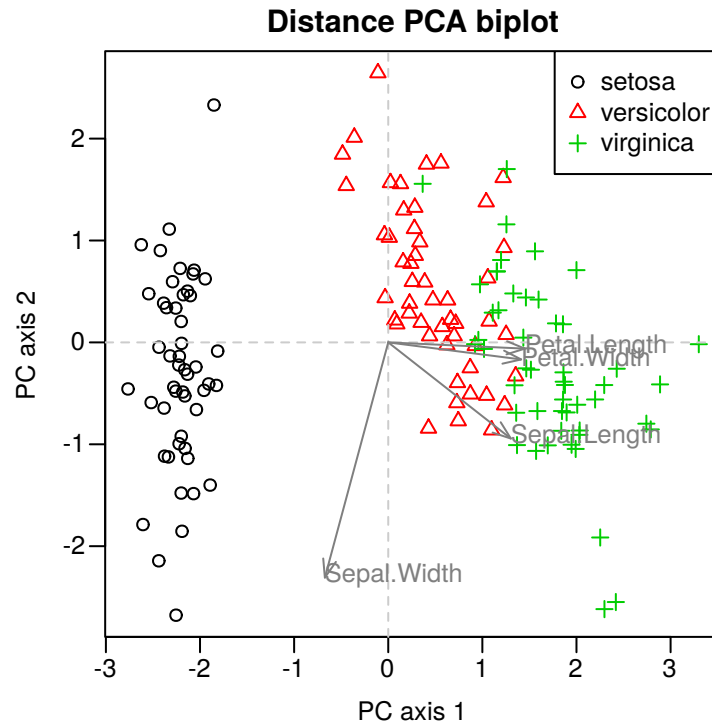


Figure 3:

There are in fact several ways to present both the observations and variables on the same graph in a biplot that will change slightly the interpretation. The plot presented above is the most frequent representation with a scaling of type 1 (in the terminology of Legendre & Legendre 2012).

An other scaling called scaling of type 2 is obtained by dividing the scores and the eigenvector values of each PC axis by the square root of the eigenvalue of this axis. The result is then a “correlation biplot” with a slightly different interpretation.

NB : `vegan::rda` and its `vegan::scores` function have a slightly different way to compute these scalings but the interpretation remains the same.

This can be computed like this :

```
vect <- pca$rotation %*% diag(pca$sdev) # eigenvectors scaling = 2
scores <- pca$x %*% diag(pca$sdev) # points scaling = 2
```

```
vect
```

```
##           [,1]      [,2]      [,3]      [,4]
## Sepal.Length 0.8901688 -0.36082989 0.27565767 0.03760602
## Sepal.Width -0.4601427 -0.88271627 -0.09361987 -0.01777631
## Petal.Length 0.9915552 -0.02341519 -0.05444699 -0.11534978
## Petal.Width 0.9649790 -0.06399985 -0.24298265 0.07535950
```

```
head(scores)
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -3.856012 -0.4573968 0.048759373 0.0034668307
## [2,] -3.543163 0.6423530 0.089575890 0.0147759036
## [3,] -4.025471 0.3257895 -0.016876547 0.0040705731
## [4,] -3.915063 0.5692317 -0.034855430 -0.0094610572
```

```
## [5,] -4.069082 -0.6163418 -0.006008993 -0.0051529817
## [6,] -3.534088 -1.4189736 -0.010296751 0.0009479166
```

The resulting graph (“correlation biplot” is in this case almost identical)

```
# dev.new(width = 10/2.54, height = 10/2.54)
par(mar = c(3.5,3.5,2,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
plot(scores[,1], scores[,2],
     pch = as.numeric(iris$Species), col = iris$Species,
     xlab = "PC axis 1", ylab = "PC axis 2", main = "Correlation PCA biplot")
abline(v=0, h=0, lty=2, col = "gray80")
cst <- 2.5 # arbitrary multiplicative constant to improve the graphical representation
arrows(x0 = 0, y0 = 0, x1 = vect[,1]*cst, y1 = vect[,2]*cst,
      col = "gray50", length = 0.1, angle = 20 )
text(x = vect[,1]*cst, y = vect[,2]*cst,
     labels = row.names(vect),
     col = "gray50", adj = 0)
legend("topright", pch = 1:3, col = 1:3, legend = levels(iris$Species))
```

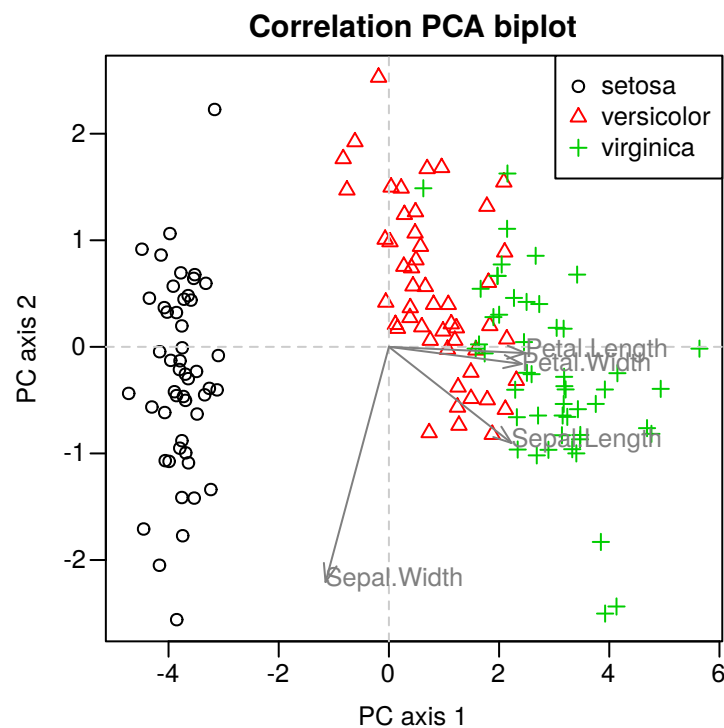


Figure 4:

1.2.3 PCA - in practice

Step by step approach of PCA and frequent questions (heavily inspired by Legendre & Legendre 2012)

1.2.3.1 Is the dataset adapted to PCA ?

You need a table with observations on the rows and variables on the columns. You mustn't transpose this matrix. You cannot use a contingency table (use Correspondance analysis for this kind of table).

- Ideally gaussian quantitative variables
- Ordinal and binary variables are OK if transformed into numeric (0,1 for binary variables)
- If quantitative variables too much skewed : transform (typically log, sqrt) to avoid an exaggerated effect of the high values
- any transformation that improves the linearity between the variables (ie that improves the correlation coefficient between the variables) will improve the PCA (it will be able to summarize more variance in fewer axes)

Typically there are two types of problems :

a) qualitative variables are not really adapted to PCA

Potential solutions when you have qualitative variables :

- use adapted distances measures (eg Gower dissimilarity) and MDS or NMDS
- use specific ordination methods (like MCA if you have only qualitative variables or FAMD if you have a mix of quantitative and qualitative variables.
- If you have a few qualitative variables mixed with quantitative variable an easy but unorthodox (and generally not recommended) solution is to transform the qualitative variables into dummy variables (one binary variable for each level) and then proceed with PCA.

b) if many 0 : double 0 will be considered as similarities

- this is not a problem with some type of measures (ex : precipitations and environmental variables)
- this is not a problem with many binary variables (eg Sex with males coded as 0 and females coded as 1 -> double 0 are real similarities !)
- this is a problem with other type of measures (typically species abundances or presence absence data)

Potential solutions when double 0 are a problem in particular for Species abundance or presence/absence data :

- apply Hellinger transformation (or chord, or chi-squared) and then PCA. You work then on the relative abundance of each species on each site (so you must accept to lose the information of the absolute abundance). But the distance conserved between the points is now the Hellinger distance which ignores the double 0. If you working with other data that species data ask yourself if it makes sense to divide the values by the sum of the rows...
- use other adapted distances measures (Sorensen, Bray-Curtis) and then MDS or NMDS

1.2.3.2 Should we use scaled/standardised variables or not ?

This is equivalent to the question : Should we use the correlation or the covariance matrix ?

In practice you don't need to effectively scale the data. Almost all PCA functions provide an option to scale or not the data internally. Scaling the data is equivalent to performing the PCA on the correlation matrix. Unscaling the data is equivalent to performing the PCA on the covariance matrix. Note that the default might be different from a function to another and as a consequence, PCA results will not be the same...

- if the units are not the same between descriptors → you must use scaled variables !
- if the dimensions are homogenous you can choose :
 - if you want that all variables contribute equally : use scaled variables
 - if you want that the variables that have higher variance contribute more : use unscaled variables.

With species data we often use unstandardized data as we want that dominant species have indeed a stronger effect on the ordination than the very rare ones. You can use log or sqrt transformation to decrease a little the importance of the dominant species. (note that the Hellinger transformation already uses a sqrt transformation internally).

Correspondance analysis (and chi squared distance) on the contrary place more weight on the rare species

Example : iris dataset

In the iris dataset the units are the same for the 4 variables. However we use `scale = TRUE` because we don't want that the morphological variable with the highest variance have a higher weight in the ordination.

1.2.3.3 How informative is the representation and which PC axes are important ?

Use a screeplot (`eigenplot` function) and look at the % of variance (eigenvalues) represented on each axis. Ideally the % of variance explained on the first axes should be $\gg 100/\text{number of variables}$. If on a 4 variables dataset the first axis summarises 25% of the variance, your PCA is not a good summary of your data !

As rough rule of thumb you might say the the “important” PC axes are :

- Axes with eigenvalue $>$ average eigenvalue (`eigenplot` function)
- Axes with eigenvalue $>$ broken stick value = important ones. (`vegan::screeplot`)

For some applications like regression on PC axes you may want to use the PC axes that represent a total of 95% of the variance.

Example : iris dataset

The screeplots show that the PCA is a very good summary of the data. The first é PC axis represent 96% of the data. Both the average and broken stick rules tend to say that the first PC is enough to summarize the data. However the PC2 is close to the threshold and might probably be usefull here (don't follow too strictly these rules - that will sometimes disagree...)

```
pca <- vegan::rda(iris[,1:4], scale = TRUE)

# dev.new(width = 10/2.54, height = 10/2.54)
par(mfrow = c(2,1), mar = c(3.5,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
eigenplot(pca)
screeplot(pca, bstick = TRUE, main = "")
```

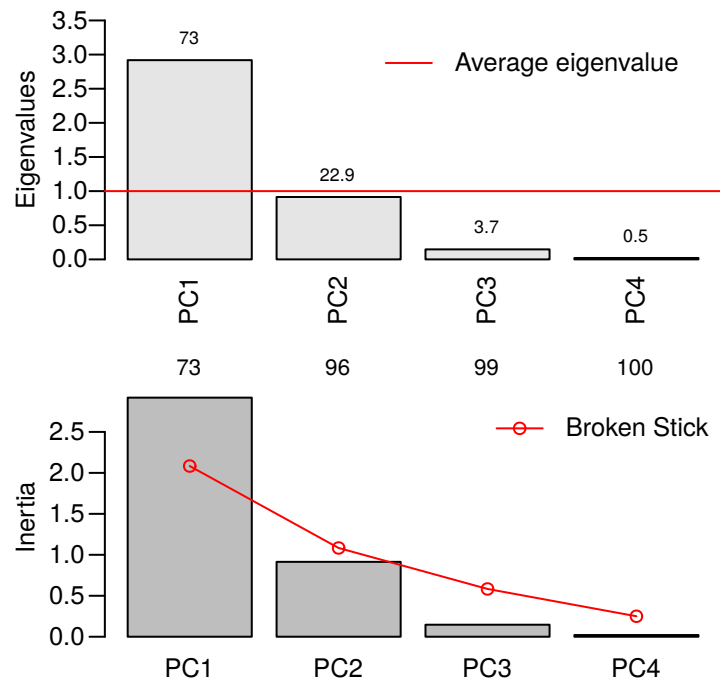


Figure 5:

1.2.3.4 How to represent the results ? How to interpret these representations ?

You will represent on a graph 2 PC axes, generally among the first ones.

There are 2 main approaches for each chosen pair of PC axes :

- 2 separated plots : a distance plot to represent the observations and a correlation circle plot to represent the variables
- 1 biplot to represent both the observations and the variables

2 separated plots

The first plot represents the observations. The distance between the points is an approximation their Euclidean distance in the original dataset. The second plot represents the correlation between the original variables and the PC axes. Here all the arrows are almost touching the circle : their representation in these 2 first PC axes is almost perfect. The Petal variables are highly positively correlated with PC axis 1 and the sepal width is highly negatively correlated with axis 2.

You can build these graphs manually as shown before. We are going to use here two functions from `mytoolbox.R` : `biplot2` and `corplot` (beware of the confusion with `corplot::corrplot`).

```
# dev.new(width = 18/2.54, height = 9/2.54)
par(mfrow = c(1,2), mar = c(3.5,3.5,2,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
biplot2(pca,
  vars = FALSE, # plot only the points - the observations
  choices = c(1,2), # plot the PC axes 1 and 2 (default)
  scaling = 1, # scaling = 1 for a distance plot (default)
  obs.col = iris$Species, # colors
  obs.pch = as.numeric(iris$Species), # shape of the points
  title = "PCA : Projection of the Observations")
legend("topright", pch = 1:3, col = 1:3, legend = levels(iris$Species))

corplot(pca, choices = c(1,2))
```

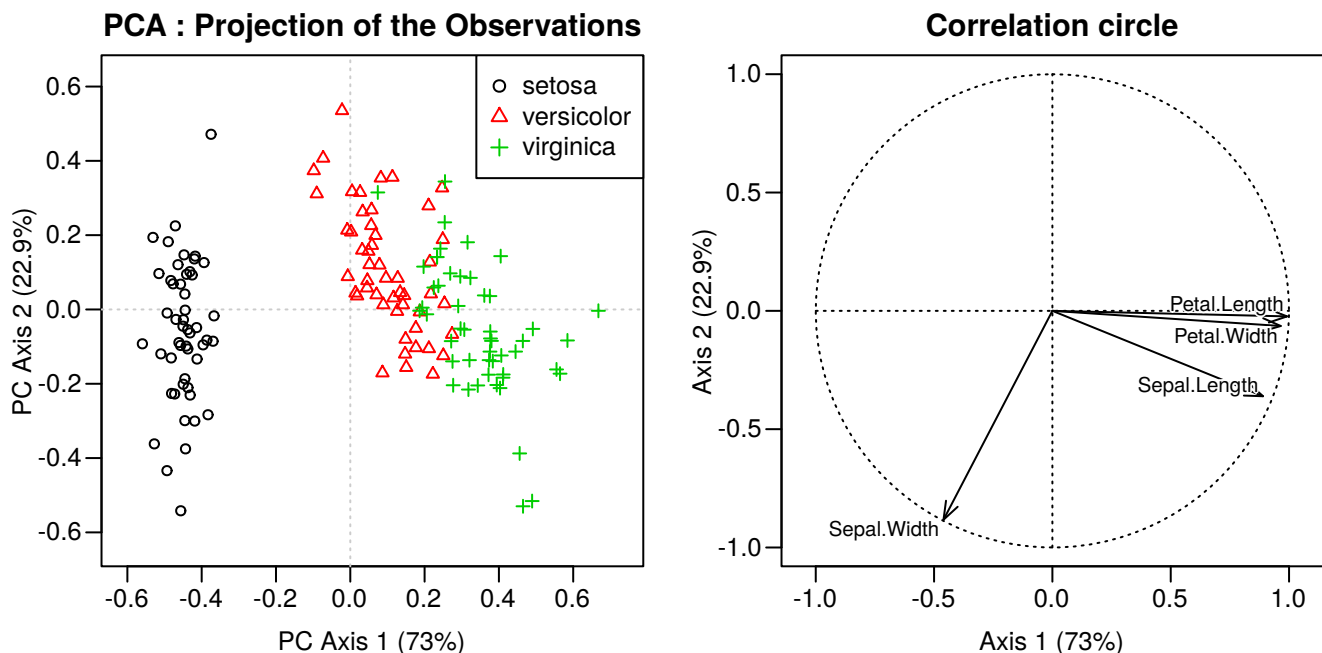


Figure 6:

Biplots

In biplots the observations are generally represented by points and the variables by arrows on the same graph. You have to choose between different scalings.

If you want to compare the distances between the rows (sites) : use distance biplots (scaling = 1). The distance between the points are an approximation of their Euclidean distance in the original dataset. The angles between the arrows are meaningless in theory but long arrows with similar loadings on the PC axes are correlated so basically long arrows pointing in the same direction are correlated. . .

If you want to visualize the correlations/covariances between the variables : use correlation biplots (scaling = 2).

The angles between the arrows represent the correlations/covariances but the distances between the sites are no more related to their euclidean distance.

Longer arrows are better represented in the reduced space. Shorted arrows point outside the plan of the graph in the multidimensional space. You can project the tips of the arrows on the PC axis : this gives you their “loading” ie their contribution to this axis.

In both cases you can project at a right angle the observations/sites/rows on the arrows that represent the original variables. This gives you an approximation of the value of this variable for the given point in the original dataset.

On the distance biplot you might add a **circle of equilibrium**. Beware that this circle has a completely different interpretation than the circle in the correlation circle plot. This “equilibrium circle” represent the length that would have a variable that would be equally well represented on all PC Axes. Arrows that are longer than the radius of this circle are particularly well represented in these 2 dimensions. This circle is not drawn on the correlation biplot because on the correlation biplot the length of the arrows in the multidimensional space is equal to their standard deviation. So the circle should be of a different size for each variable. In the distance biplot the length of the arrows in the multidimensional space is 1 (sometimes multiplied by a constant).

2 common mistakes

1. do not interpret the distance between the points and the tip of the arrows. These arrows are axes and should be interpreted as axes. They can also be continued further than the centre in their negative zone.
2. don't forget the other dimensions. When two points are close to each other it might be because they are similar but they might also be separated in the other dimensions if they are not well represented in these 2 dimensions.

Points that are far away from each other can be safely interpreted as different. Points that are far from the centre are generally well represented in these 2 dimensions and then points that are close to each other in these areas are probably similar. points near the centre can be there either because they are not well represented in these 2 dimensions or they might be very well represented but they have average values of the variables with the highest loadings for these axes.

Always represent - as much as possible - the variables with arrows to decrease the risk of misinterpretation (ie common mistake nr 1). This is however sometimes impossible when the number of variables is too high. . .

The variables (eg species) could also be represented by points as weighted averages of the sites (function `vegan::wascores`) in some circumstances but this is unusual with PCA (while it is a frequent representation in CA, MDS and NMDS).

Example with the iris dataset

NB the difference between the 2 scalings is very low here. . . This happens quite often in practice. . .

```
# dev.new(width = 18/2.54, height = 9/2.54)
par(mfrow = c(1,2), mar = c(3.5,3.5,2,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
set.seed(1)
biplot2(pca,
  sc = 0.2, # arbitrary constant used to spread the points and have a better graph
  scaling = 1, # scaling = 1 for a distance plot (default)
  obs.col = iris$Species, # colors
  obs.pch = as.numeric(iris$Species)# shape of the points
)
legend("bottomright", pch = 1:3, col = 1:3, legend = levels(iris$Species))
biplot2(pca,
  sc = 0.5, # arbitrary constant used to spread the points and have a better graph
  scaling = 2, # scaling = 1 for a distance plot (default)
  obs.col = iris$Species, # colors
  obs.pch = as.numeric(iris$Species)# shape of the points
)
legend("topright", pch = 1:3, col = 1:3, legend = levels(iris$Species))
```

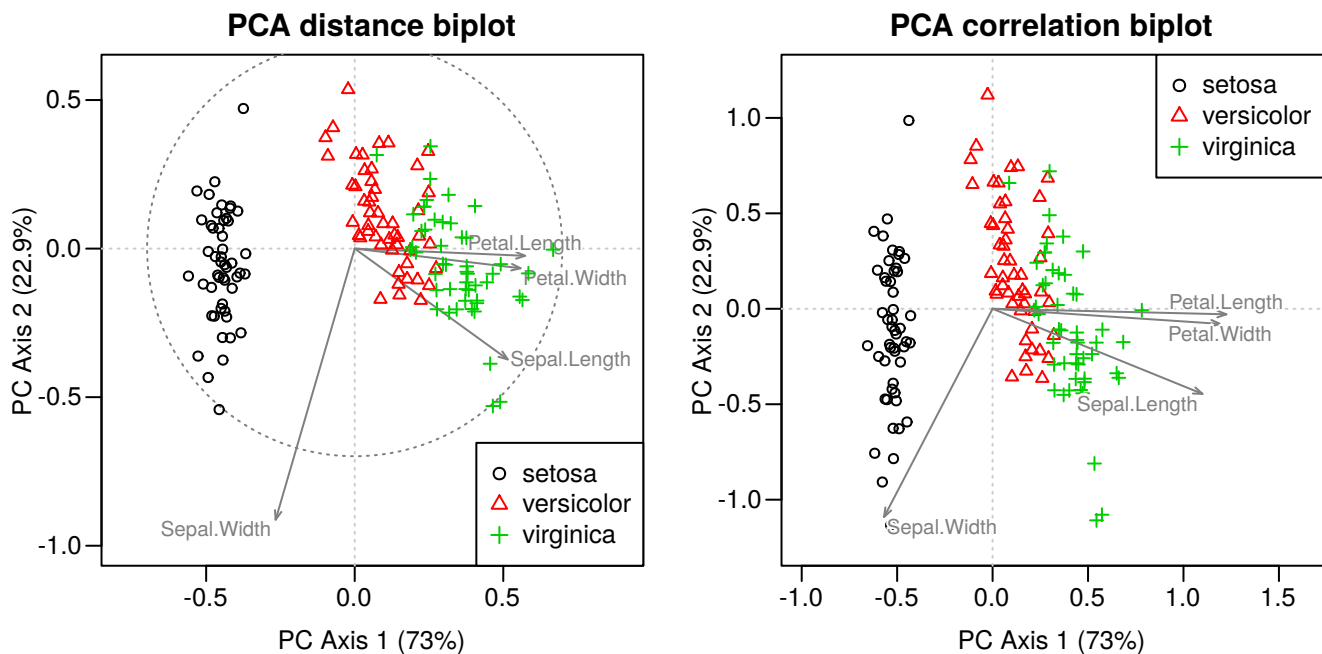


Figure 7:

If you need to build manually this kind of plots you can extract the coordinates of the observations (“sites” in `vegan`) and variables (“species” in `vegan`) with the function `vegan::scores` (not run here)

```
scores(pca, scaling = 1, choices = c(1:2), display = "species")
scores(pca, scaling = 2, choices = c(1:4), display = "species")
scores(pca, scaling = 1, choices = c(1:2), display = "sites")[1:5,]
scores(pca, scaling = 2, choices = c(1:4), display = "sites")[1:5,]
```

1.2.3.5 How to assess the quality of the representation of the variables in the reduced space? What is the contribution of each variable to the reduced space ?

- draw a correlation circle plot : `corplot`. The circle represent a value of 1. Arrows approaching the circle are well represented in these 2 dimensions.
- add a circle of equilibrium contribution on distance biplots: `biplot2`. The arrows that go further than the circle are better represented on average on these 2 dimensions than expected if they were equally well represented in all dimensions
- compute the \cos^2 for the variables (% of their variance represented in the reduced space) - visualize this with a heatmap : `cos2`, `cos2vars` and `cos2heatmap`

Example of \cos^2 computation vor the variables of the iris dataset. These numbers represent the proportion of the variance of the variable that is represented on each PC axis. The sum of the lines is 1. 79.24 % of the variability of Sepal.Length is represented on the first PC axis and 13.02 % on the second. When you make a plot with the first and second PC axis 92.26% of the variance of this variable is represented in the reduced space.

```
cos <- cos2(pca)[[1]]
```

It is generally particularly instructive to make a heatmap of these \cos^2 values. This is particularly usefull whan you have many variables because it shows you globally which varaible is well represented in which PC axis. With the biplots and correlation circle plots you can visualize only 2 variables at the same time. Here you can have an overview of all variables.

```
# dev.new(width = 10/2.54, height = 6/2.54)
cos2heatmap(cos)
```

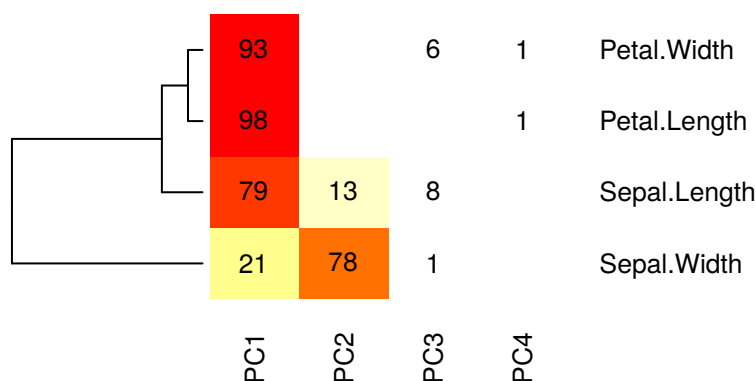


Figure 8:

1.2.3.6 How to assess the quality of the representation of the observations in the reduced space ?

- Compute the `cos2` for the observations (use heatmap for visualization) : `cos2`, `cos2obs` and `cos2heatmap`
- Draw a Shepard diagram to compare the distance in the reduced space (= “cophenetic distance”) with the original distances.

Generally the points that are far from the centre are well represented in the reduced space.

The signification of these “variables `cos2`” is not really a % of variance represented (unlike the `cos2` for the variables). They might be seen as an index of the quality of the representation comprised between 0 (very bad) and 1 (very good). Here most of the points are well represented on the first 2 PC axes.

```
cos <- cos2(pca)[[2]]  
cos[1:5,]
```

```
##          PC1    PC2    PC3    PC4  
## sit1 0.9540 0.0429 0.0030 0.0001  
## sit2 0.8928 0.0937 0.0113 0.0022  
## sit3 0.9790 0.0205 0.0003 0.0001  
## sit4 0.9347 0.0631 0.0015 0.0008  
## sit5 0.9315 0.0682 0.0000 0.0002
```

```
# dev.new(width = 6/2.54, height = 12/2.54)  
cos2heatmap(cos, notecex = 0.001, cexRow = 0.1)
```

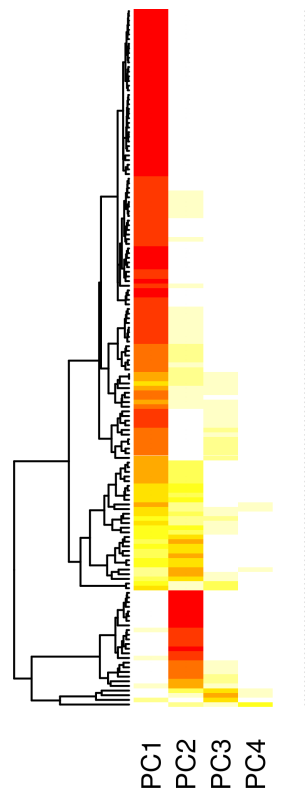


Figure 9:

1.2.3.7 The PCA graphs are overcrowded and difficult to read. What can we do ?

This is frequent particularly when you have large datasets.

Different strategies might help :

- on a biplot it might happen that for example the size of the arrows are much larger than the distances between the points. As a results the points are highly concentrated on the center and make the graph difficult to read. You can simply multiply the values of the points (or the arrows) by an arbitrary coefficient to spread more the points on the graph. This is done internally by `vegan::rda` but it might need to be adjusted. NB : do that only for the graphical representation. With thye `biplot2` function, you can do that with the `sc` argument.
- use to \cos^2 values to draw only the points (observations) and arrows (variables) that are well represented on the graph (eg only the values with a $\cos^2 > 0.7$). You can also draw all the points and/or arrows and label only the ones that are well represented
- instead of one biplot, make two separated plots : one plot with the observations (rows) and separately a correlation circle plot to visualise the contribution of each variable to the different axes
- if you cannot do otherwise draw the variables as points instead of arrows but beware of the misinterpretation : you should not interpret the distances between these points !
- for publication quality plot, remember that you can always save your plot on the disk in a vectorial format (pdf, ps, svg) and edit it in your favorite vector graphic editing software (like the free and open source “inkscape”). A recommended graphical device is `cairo_pdf` that will save your image as a pdf and include in the file the fonts used. If you don't do that the symbol used might change from one computer to another. the functions `savepdf` in `mytoolbox.R` allows you to do that in an interactive way (it will save the plot at the dimension of the graphic window).

It is difficult to demonstrate the use of `cos2` as filter on the iris dataset because all observations are well represented in the first 2 axis. As an example (you would probably not do that in real life) we might look at the points that are well represented in the PC axes 3 and 4 (argument `choices = c(3,4)`). We first add the lables only for the points with a \cos^2 sum > 0.5 in this two dimensions. The graph is still overcrowded so in the next graph we also remove the points to keep the only 4 points with a sum $\cos^2 > 0.5$ in these PC3 and PC4 dimensions.

```
# dev.new(width = 18/2.54, height = 9/2.54)
par(mfrow = c(1,2), mar = c(3.5,3.5,2,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
set.seed(1)
biplot2(pca,
        choices = c(3,4),
        sc = 0.05, # arbitrary constant used to spread the points and have a better graph
        scaling = 1, # scaling = 1 for a distance plot (default)
        obs.col = palette()[iris$Species], # colors
        obs.pch = as.numeric(iris$Species), # shape of the points
        obs.cos2.lim = 0.5,
        labels = TRUE
)
legend("bottomright", pch = 1:3, col = 1:3, legend = levels(iris$Species))
biplot2(pca,
        choices = c(3,4),
        sc = 0.05, # arbitrary constant used to spread the points and have a better graph
        scaling = 1, # scaling = 1 for a distance plot (default)
        obs.col = palette()[iris$Species], # colors
        obs.pch = as.numeric(iris$Species), # shape of the points
        obs.cos2.lim = 0.5,
```

```

obs.cos2.lim.points = 0.5,
labels = TRUE
)
legend("bottomright", pch = 1:3, col = 1:3, legend = levels(iris$Species))

```

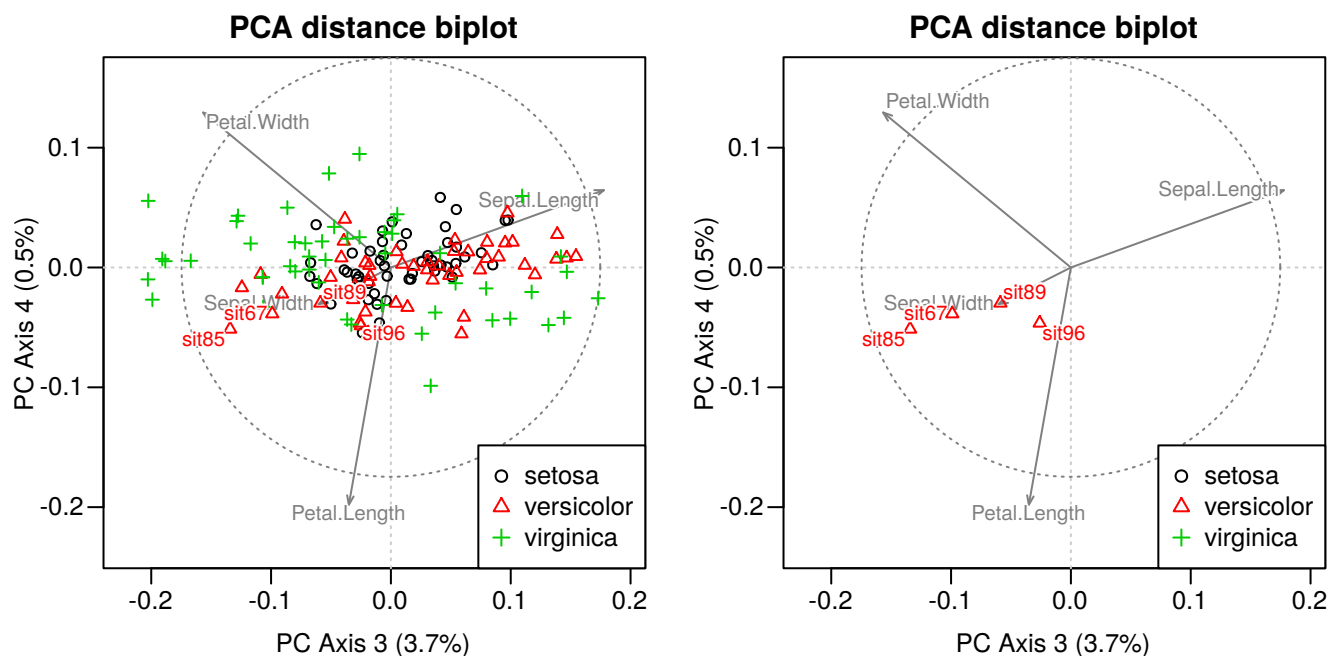


Figure 10:

Note that on the previous graph the labels are places with an algorithm that limits as much as possible the collisions between the labels for an improved readability. This algorithm is quite slow and will be disabled for more than 100 observations (or variables). You might also chose to plot only the label at the coordinates of the points. this might be done with the combination of options `labels = TRUE` and `points = FALSE`.

See the code of the function for more help on the options of this function

```

# dev.new(width = 9/2.54, height = 9/2.54)
par(mar = c(3.5,3.5,2,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
set.seed(1)

biplot2(pca,
  choices = c(3,4),
  sc = 0.05, # arbitrary constant used to spread the points and have a better graph
  scaling = 1, # scaling = 1 for a distance plot (default)
  obs.col = palette()[iris$Species], # colors
  obs.pch = as.numeric(iris$Species), # shape of the points
  obs.cex = 1, # size of the labels and points of the observations
  obs.cos2.lim = 0.5, # remove the points labels with cos2 < 0.5
  obs.cos2.lim.points = 0.5, # remove the points with cos2 < 0.5
  var.cos2.lim = 0.01, # remove arrows labels for vars with cos2 < 0.001
  points = FALSE, # do not plot the points
  labels = TRUE, # plot the labels instead of the points
  circle = FALSE # no equilibrium circle plot
)

```

```
legend("bottomright", pch = 1:3, col = 1:3, legend = levels(iris$Species))
```

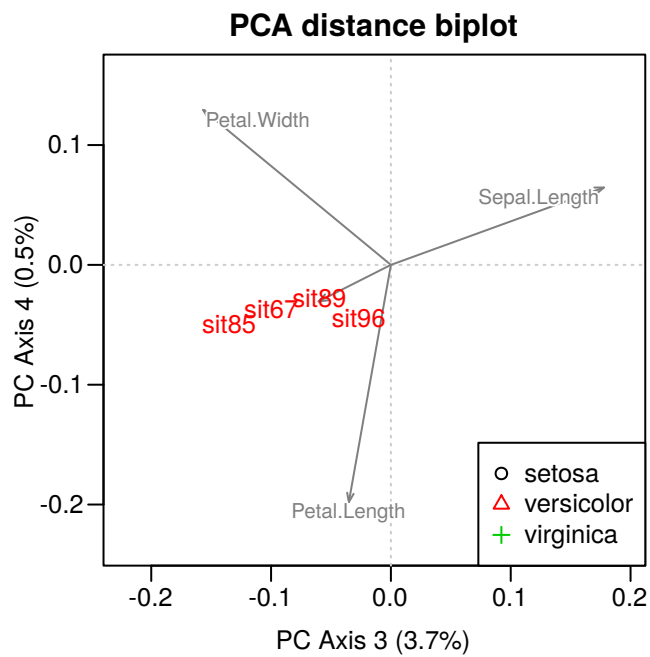


Figure 11:

1.2.4 Example of PCA on the BIO worldclim variables

19 bioclimatic variables in 1375 5x5 km² UTM grid squares in Belgium. See the file `data/UTM5/UTM5data_README.pdf` for a description of the dataset.

These 19 bioclimatic variables are defined as :

- BIO1 = Annual Mean Temperature
- BIO2 = Mean Diurnal Range (Mean of monthly (max temp - min temp))
- BIO3 = Isothermality (BIO2/BIO7) (* 100)
- BIO4 = Temperature Seasonality (standard deviation *100)
- BIO5 = Max Temperature of Warmest Month
- BIO6 = Min Temperature of Coldest Month
- BIO7 = Temperature Annual Range (BIO5-BIO6)
- BIO8 = Mean Temperature of Wettest Quarter
- BIO9 = Mean Temperature of Driest Quarter
- BIO10 = Mean Temperature of Warmest Quarter
- BIO11 = Mean Temperature of Coldest Quarter
- BIO12 = Annual Precipitation
- BIO13 = Precipitation of Wettest Month
- BIO14 = Precipitation of Driest Month
- BIO15 = Precipitation Seasonality (Coefficient of Variation)
- BIO16 = Precipitation of Wettest Quarter
- BIO17 = Precipitation of Driest Quarter
- BIO18 = Precipitation of Warmest Quarter
- BIO19 = Precipitation of Coldest Quarter

Read the data , extract the “bio” variables, add rownames (= UTM grid codes)

```
d <- read.csv2("data/UTM5/UTM5data.csv")
d2 <- d[, which(colnames(d) == "bio01") : which(colnames(d) == "bio19")]
rownames(d2) <- d$MGRS
# summary(d)
# colnames(d)
# pairs2(d2[sample(1:nrow(d2), 500),], reorder = TRUE, pt.cex = 0.7)
```

PCA with vegan

```
res <- rda(d2, scale = TRUE)
```


Screeplot : the 3 first PC axes show 94% of the variance.

```
# dev.new(width = 16/2.54, height = 6/2.54)
par(mar = c(4,3, 1, 1) , cex = 0.8)
eigenplot(res)
```

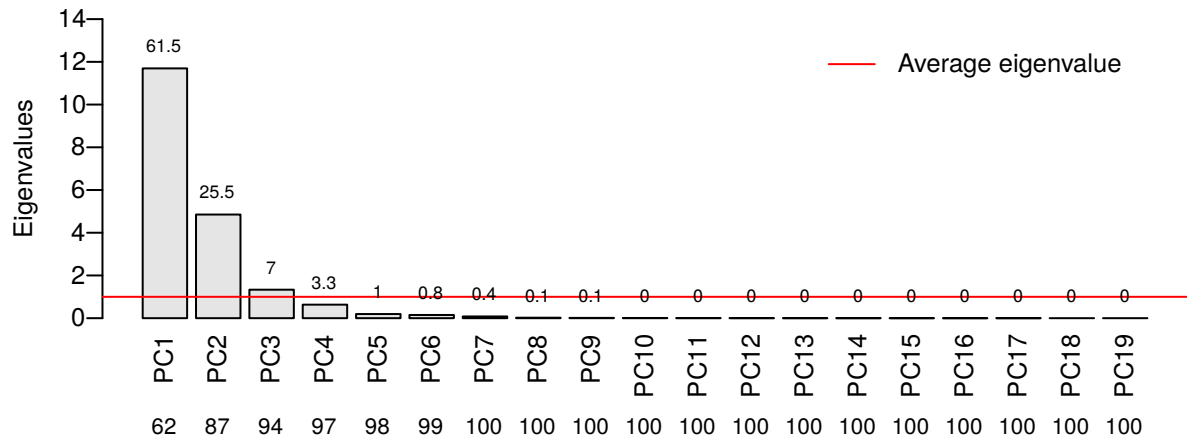


Figure 12:

Vegan provides two functions to visualize the outputs of ordinations but their raw results are not really useable as is ...

```
# dev.new(width = 14/2.54, height = 7/2.54)
par(mfrow = c(1,2), mar = c(3.5,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.7)
biplot(res, scaling = 1)
ordiplot(res, scaling = 1)
```

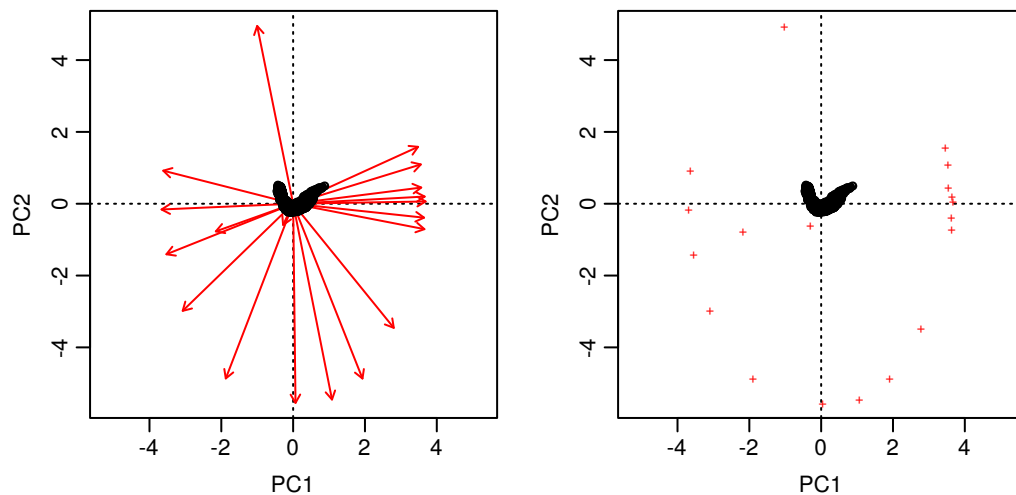


Figure 13:

Correlations circle plot : most variables are very well represented in these 2 first axes excepted bio09 (Mean Temperature of Driest Quarter)

```
# dev.new(width = 12/2.54, height = 12/2.54)
par(mar = c(3,3, 2, 1), mgp = c(1.8, 0.5, 0), cex = 1)
corplot(res)
```

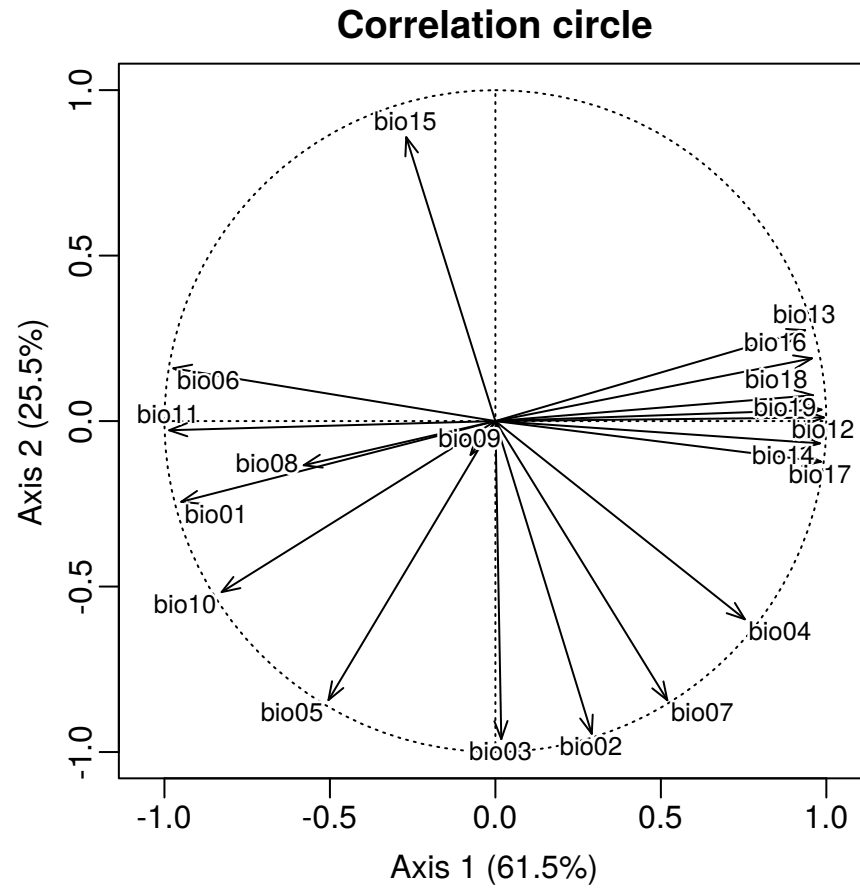


Figure 14:

Distance and correlation biplots (very similar here). NB the labels are automatically disabled because there are more than 100 labels to plot.

```
# dev.new(width = 20/2.54, height = 10/2.54)
par(mfrow = c(1,2), mar = c(3,3, 2, 1), mgp = c(1.8, 0.5, 0), cex = 0.8)
biplot2(res, sc = 0.2, scaling = 1)
biplot2(res, sc = 0.5, scaling = 2, var.col = "orangered")
```

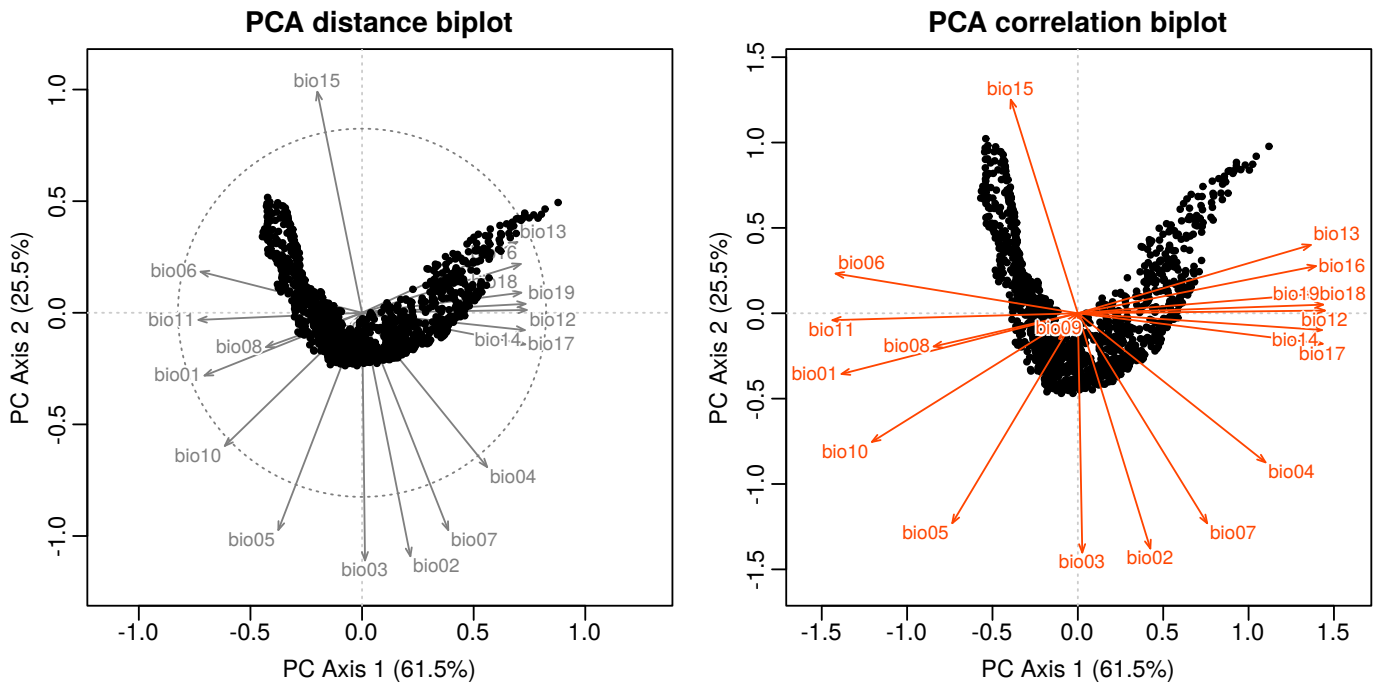


Figure 15:

Add colors for the different natural regions on the distance biplot to help the interpretation

```
# dev.new(width = 16/2.54, height = 12/2.54)
mycols <- c("darkgreen", "gold", "lightgreen", "yellow", "khaki",
           "chocolate4", "pink", "blue", "black", "orangered")
par(mar = c(3,3, 2, 1), mgp = c(1.8, 0.5, 0), cex = 0.8)
biplot2(res, choices = c(1,2), sc = 0.1, scaling = 1, labels = FALSE,
        obs.col = mycols[d$naturalRegion], obs.cex = 1.5)
legend("bottomright", bty = "n", pch = 20, col = mycols,
      legend = levels(d$naturalRegion), pt.cex = 2)
```

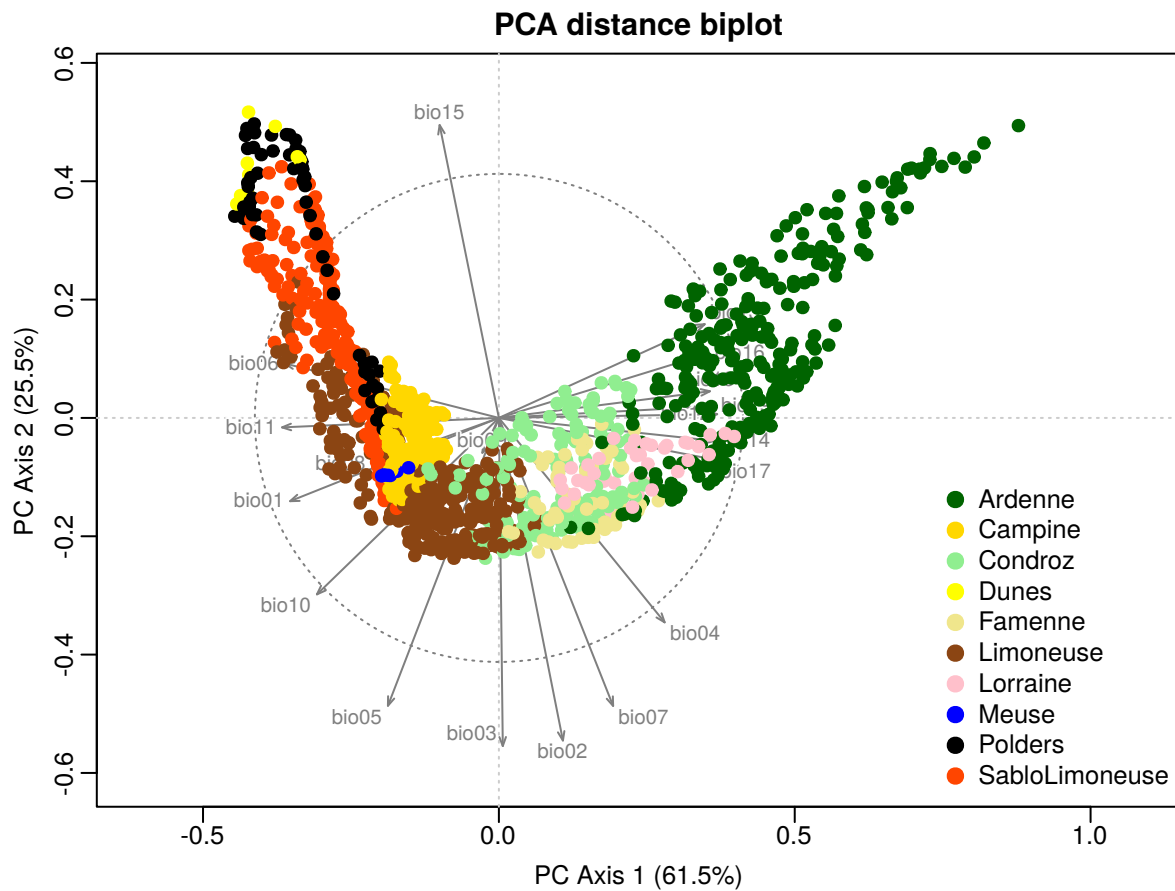


Figure 16:

Cos2 of the variables

```
# dev.new(width = 16/2.54, height = 12/2.54)
cos2heatmap(cos2(res)[[1]])
```

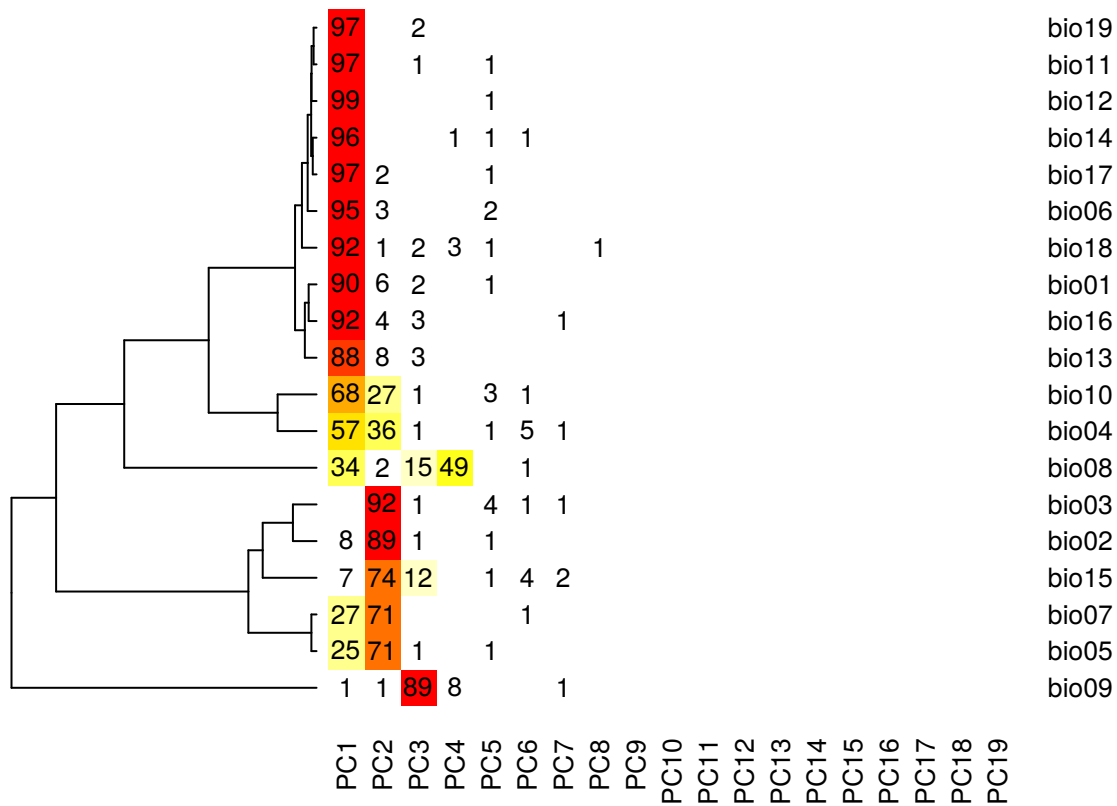


Figure 17:

Cos2 of the observations. A few observations seem to be not so well represented in the 2 first PC axes.

```
# dev.new(width = 12/2.54, height = 15/2.54)
cos2heatmap(cos2(res)[[2]][,1:7], notecex = 0.001, cexRow = 0.1)
```

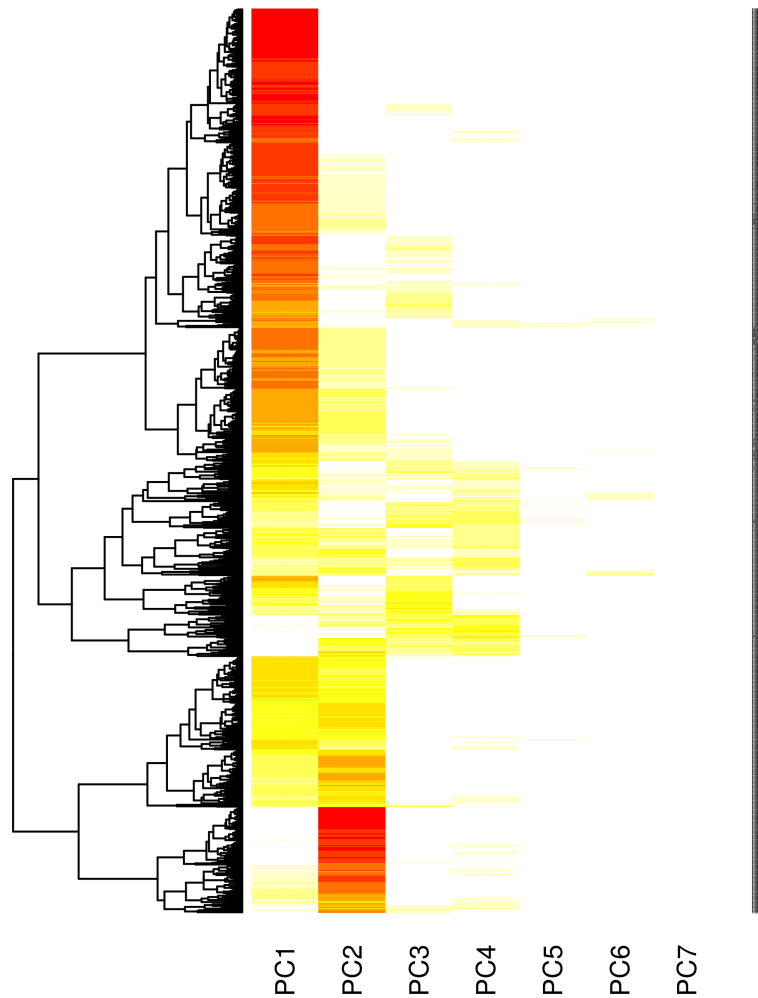


Figure 18:

```
# When the dataset is too large you can also take a random sample of the lines
# to simplify the output. (NB the results are not shown here)
#
```

```
randomsample <- sample(1:nrow(d2), size = 200)
cos2heatmap(cos2(res)[[2]][randomsample,1:7], notecex = 0.001, cexRow = 0.5)
```

Which UTM grid squares are not well represented on the first 2 axes ?

```
# Extract the cos2 and compute their sum for the 2 first axes
# and keep only the cases where this value is < 0.3
tmp <- cos2(res)$obs
tmp <- tmp[(tmp[, "PC1"] + tmp[, "PC2"] < 0.3), ]
# extract the lines corresponding to these low cos2 values
tmp <- d[d$MGRS %in% rownames(tmp), c("MGRS", "x", "y", "naturalRegion")]
```

Lets plot them on a map !

Read a geospatial dataset (shapefile) containing the natural regions of Belgium used here and plot the points that are not well represented.

The points that are not well represented in the 2 first axes are situated in the far east part of the Condroz (e.g. “Pays de Herve”) and the “Sablo-Limoneuse” (Sandy-Loam) region.

```
library(sf)
regnat <- st_read("data/spatial/Regions_Naturelles.shp", crs = 31370)

## Reading layer `Regions_Naturelles' from data source `/home/gilles/stats/Formation_R_stats/Forma
## Simple feature collection with 20 features and 4 fields
## geometry type: POLYGON
## dimension: XY
## bbox: xmin: 22430.47 ymin: 21136.83 xmax: 295227.9 ymax: 243867
## epsg (SRID): 31370
## proj4string: +proj=lcc +lat_1=51.16666723333333 +lat_2=49.83333339 +lat_0=90 +lon_0=4.3674866

# dev.new(width = 10/2.54, height = 10/2.54)
par(mar = c(0,0,0,0))
plot(st_geometry(regnat, NULL))
points(tmp[, c("x", "y")], pch = 20, col = "orangered")
```

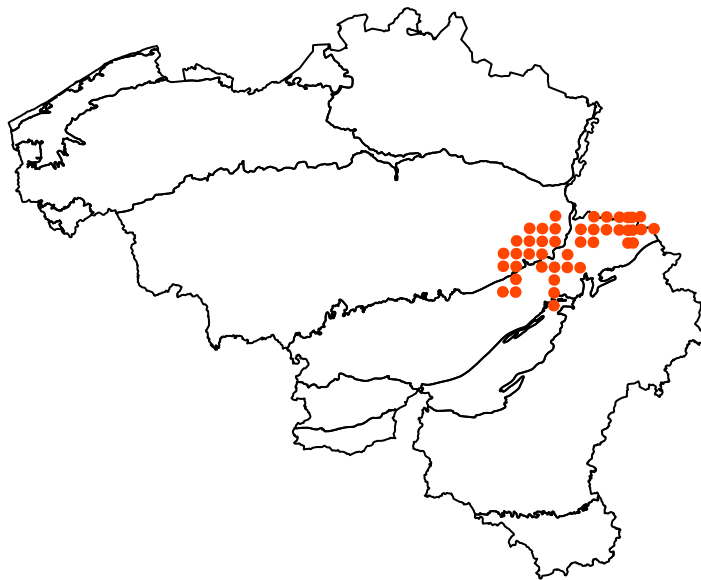


Figure 19:

Biplot for the 3rd and 4th Principal Components axes. The PC axis 3 is mainly related to bio9 and PC axis 4 is mainly related to bio8

```
# dev.new(width = 16/2.54, height = 12/2.54)
par(mar = c(3,3, 2, 1), mgp = c(1.8, 0.5, 0), cex = 0.8)
biplot2(res, choices = c(3,4), sc = 0.03, scaling = 1, labels = FALSE,
        obs.col = mycols[d$naturalRegion], obs.cex = 0.6,
        obs.cos2.lim = 0.6)
legend("bottomleft", bty = "n", pch = 20, col = mycols,
       legend = levels(d$naturalRegion), pt.cex = 2)
```

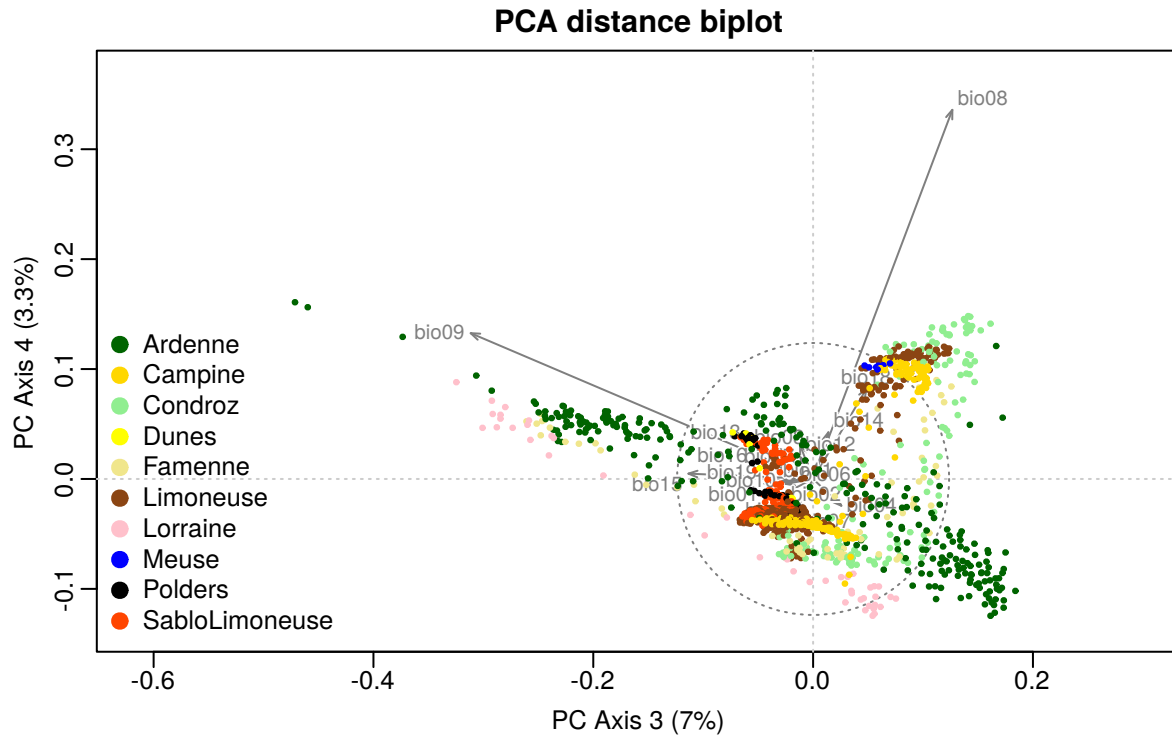


Figure 20:

Combine hierarchical clustering with PCA...

```
# dev.new(width = 18/2.54, height = 12/2.54)
cl <- hclust(dist(scale(d2)), method = "ward.D2")
# reorder the dendrogram according to the first principal component scores
cl <- reorder(cl, order(scores(res, scaling = 1)$sites[, "PC1"]))

# more simple way to reorder (not done here)
# cl <- reorder(cl, rowMeans((d2)))
k = 3
groups <- cutree(cl, k = k)

# Color band top represent the natural region
grcol <- mycols[d$naturalRegion]

# compute cophenetic correlation
coph <- coph_average <- cophenetic(cl)
Rcoph <- round(cor(dist(d2), coph), 3)

par( mar = c(1,1,2,0), cex = 0.75)
plot(cl, main = "", hang = -1, axes = FALSE, xlab = "", ylab = "", labels = FALSE)
title("ward.D2 on Euclidean dist", line = 1)
mtext(paste0("Cophenetic Cor. = ", Rcoph), cex = 0.7, line = -0.25)
symbols(x = 1:nrow(d2), y = rep(-10, nrow(d2)), rectangles= cbind(rep(1, nrow(d2)), 20),
        bg= grcol[cl$order], fg = NA, add=TRUE, inches=FALSE)

groups_col <- c("black", "gold", "darkblue", "orangered", "skyblue")
rect.hclust(cl, k = k, border = groups_col[1:k])

legend("topleft", bty = "n", pch = 20, col = mycols, xpd = NA, inset = -0.01,
       legend = levels(d$naturalRegion), pt.cex = 2)
legend("topright", bty = "n", lty = 1, col = groups_col[1:k],
       legend = paste("Group", 1:k), pt.cex = 2)
```

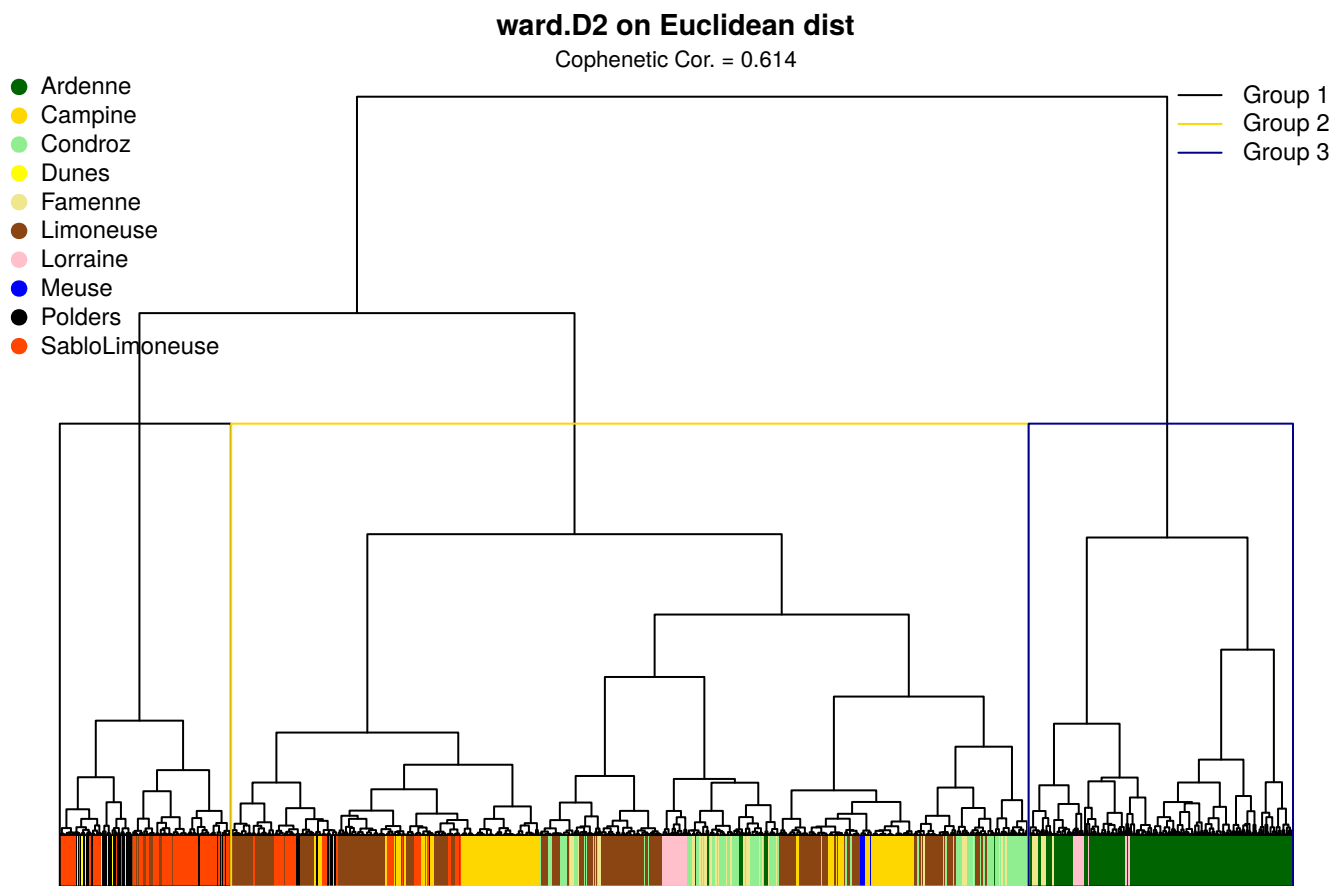


Figure 21:

Add colors for the different groups defined by the dendrogram on a PCA

```
d2 <- d[, which(colnames(d) == "bio01") : which(colnames(d) == "bio19")]
d2 <- scale(d2) # scaling the matrix
res <- rda(d2, scale = TRUE)
```

```
cl <- hclust(dist(d2), method = "ward.D2")
groups <- cutree(cl, k = k)
```

```
# dev.new(width = 16/2.54, height = 12/2.54)
par(mar = c(3,3, 2, 1), mgp = c(1.8, 0.5, 0), cex = 0.8)
biplot2(res, choices = c(1,2), sc = 0.1, scaling = 1, labels = FALSE,
        obs.col = groups_col[groups], obs.cex = 0.8)
legend("bottomright", bty = "n", pch = 20, lty = 1, col = groups_col,
      legend = paste("Group", 1:k), pt.cex = 2)
```

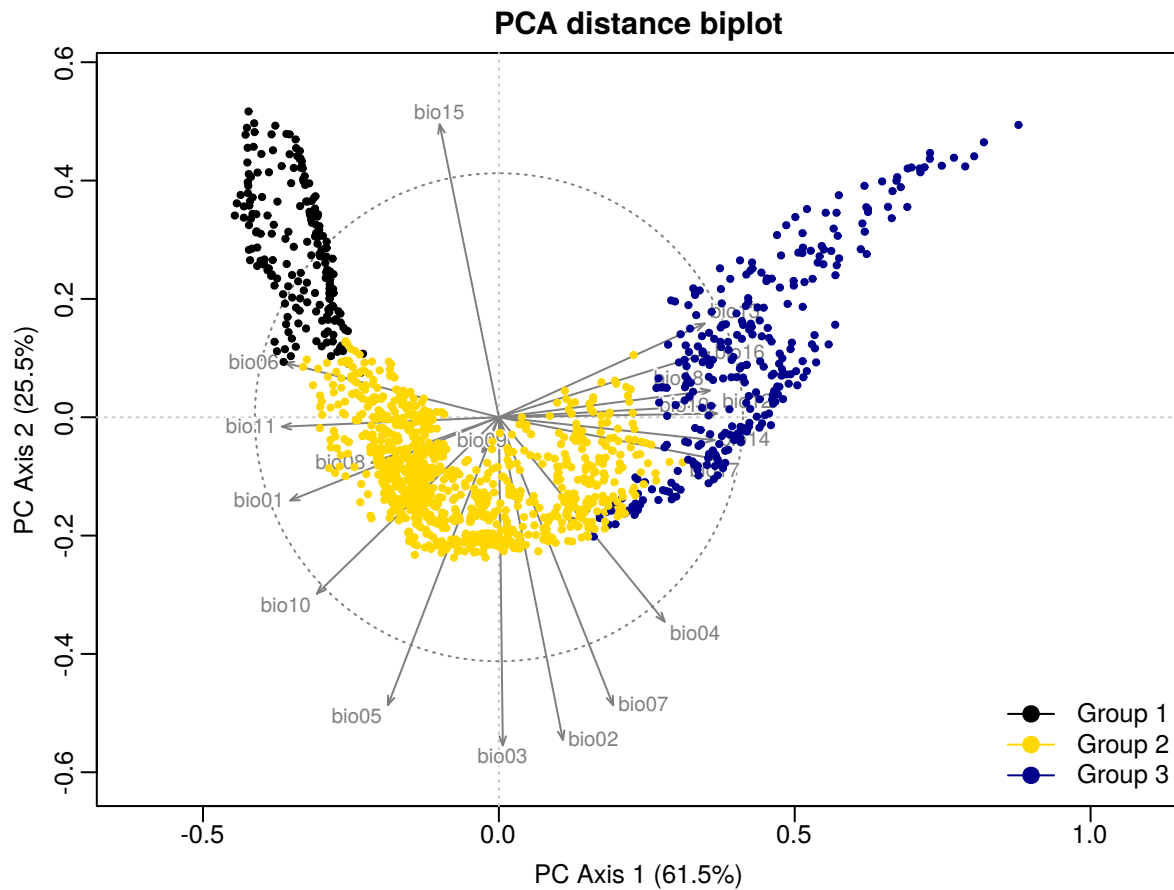


Figure 22:

You can also add the convex hulls defining the groups with the colors of the natural regions :

```
# dev.new(width = 16/2.54, height = 12/2.54)
par(mar = c(3,3, 2, 1), mgp = c(1.8, 0.5, 0), cex = 0.8)
biplot2(res, choices = c(1,2), sc = 0.1, scaling = 1, labels = FALSE,
        obs.col = mycols[d$naturalRegion], obs.cex = 1.5)
vegan::ordihull(res, groups, scaling = 1, col = groups_col[1:k], choices = c(1,2))
legend("bottomright", bty = "n", pch = 20, col = mycols,
      legend = levels(d$naturalRegion), pt.cex = 2)
legend("bottomleft", bty = "n", lty = 1, col = groups_col,
      legend = paste("Group", 1:k), pt.cex = 2)
```

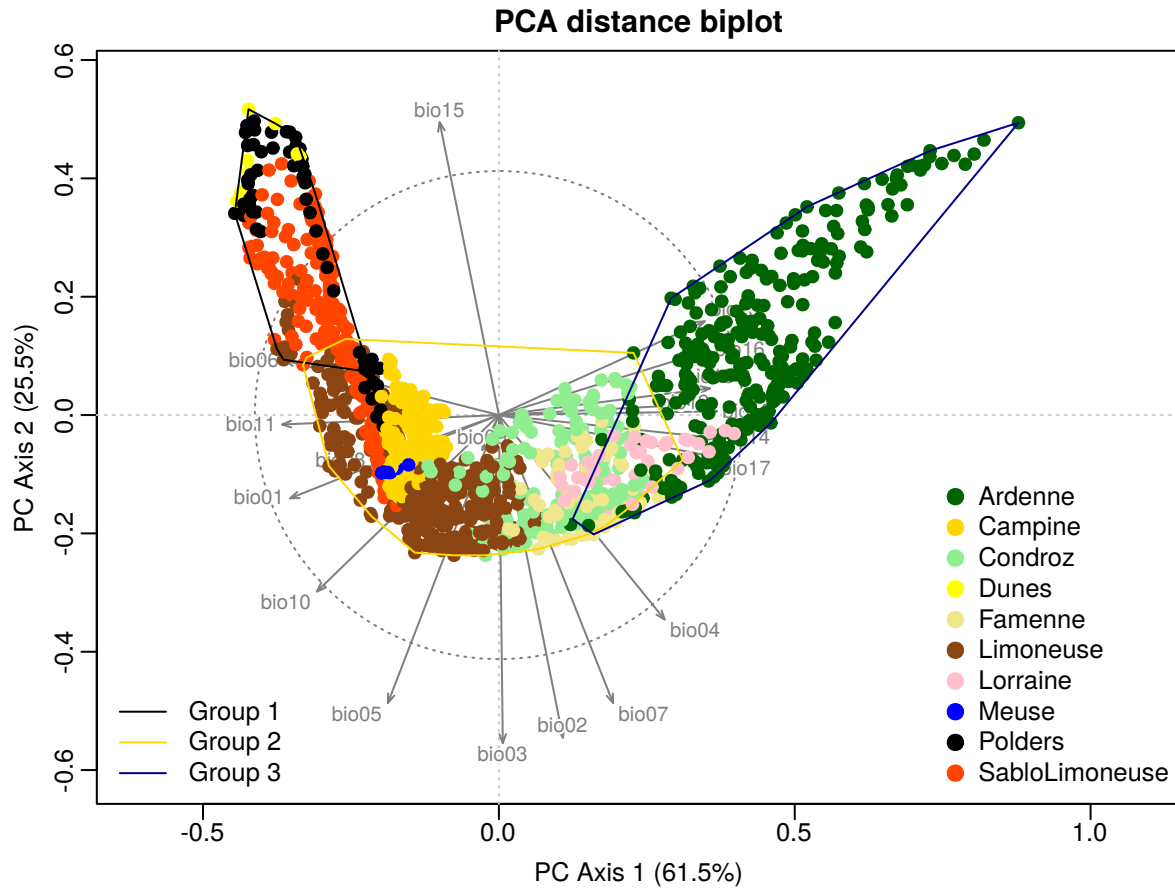


Figure 23:

1.3 NMDS : Non Metric Multidimensional Scaling

1.3.1 Introduction

As MDS, NMDS can be used with any distance matrix.

However, the aim of NMDS is to preserve and represent the ordering of the observations in few dimensions while the objective of other methods like PCA or MDS is more to preserve the exact distance between observations, and to find the combination of original axes that maximise the explained variance.

Other differences with MDS (and PCA) :

- you must specify how many dimensions you want (see below)
- NMDS is not an eigenvalue method it does not try to maximise the variance associated with each axis. It tries to minimise the difference between the distance between the points in for example 2 NMDS dimensions and the true distance
- NMDS is an iterative procedure. It might not converge on complex cases. The computation time is also much higher.
- Like MDS but unlike PCA, the NMDS solution does not provide a projection of the original axes in the reduced space. The typical way to present the original axes (like species) is by computing the average of point coordinates weighed by the value of the original variable (`vegan::wascores`). The interpretation is completely different than the PCA eigenvectors. The points that are close to variables labels are susceptible to have high values for this variable. Variables labels that are close to each other are probably similar.

1.3.2 Computation and graphical representation

For this section, we will work again on the pollen pellets collected by honey bees have been collected in several sites in July, August, September and October. ~ 1000 pollen grains have been extracted and identified up to the family, genus or species level.

```
d <- read.csv2("data/pollen/full_dataset.csv")
# d <- d[d$Buffer == 1500, ]
d <- data.frame(d[, c(2:5)], d[, 14:47])
d <- unique(d)
d <- na.omit(d)
d <- d[,c(1:4, which(colSums(d[, -c(1:4)]) > 50) + 4)]
# select the columns to use
Y <- d[, -c(1:4)]
```

Compute the distances and some clustering...

```
#
hell_trans <- decostand(Y, "hellinger") # hellinger transformation
hell_dist <- dist(hell_trans) # hellinger distance between the rows (samples)
rowClust <- hclust(hell_dist, method = "ward.D2") # ward clustering on the rows
k <- 5
groups <- cutree(rowClust, k)
```

Compute NMDS for 3 dimensions

```
NMDS <- MASS::isoMDS(hell_dist, k = 3)
```

```
## initial value 24.199078
```

```
## iter 5 value 14.439741
## iter 10 value 13.526650
## final value 13.488399
## converged
```

Plots

```
# dev.new(width = 18/2.54, height = 9/2.54)
n <- 15 # how many species to display ?
par(mfrow = c(1,2), mar = c(3.5,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 1)
choices = c(1,2)
plot(NMDS$points[,choices], asp = 1,
     pch = as.character(as.numeric(groups)), col = as.numeric(groups),
     xlab = paste0("NMDS dim ", choices[1]),
     ylab = paste0("NMDS dim ", choices[2]))
abline(v = 0, h = 0, col = "gray80", lty = 2)
spe <- wascores(NMDS$points[,choices], hell_trans)
ordilabel(spe[1:n,], labels = row.names(spe)[1:n], priority = n:1)

# same graph for dimensions 1 and 3
choices = c(1,3)
plot(NMDS$points[,choices], asp = 1,
     pch = as.character(as.numeric(groups)), col = as.numeric(groups),
     xlab = paste0("NMDS dim ", choices[1]),
     ylab = paste0("NMDS dim ", choices[2]))
abline(v = 0, h = 0, col = "gray80", lty = 2)
spe <- wascores(NMDS$points[,choices], hell_trans)
ordilabel(spe[1:n,], labels = row.names(spe)[1:n], priority = n:1)
```

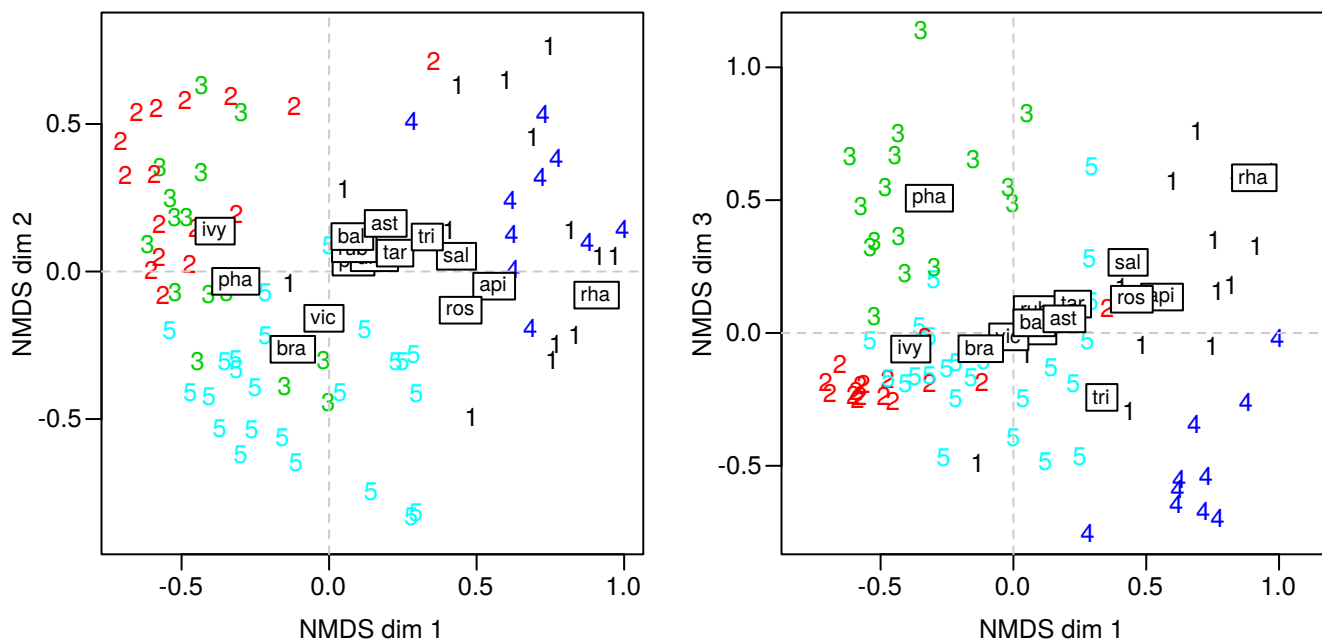


Figure 24:

1.3.3 How many dimensions to choose for NMDS ?

According to Yari Oksanen :

“Some people are very disturbed: how do they know the correct number of dimensions ? Answer is easy: there is no correct number, although some numbers may be worse than others...”

There is no % of variance associated with each axis in nMDS in contrast with other Principal Component Methods like PCA, CA, PCoA (= MDS).

I quote Legendre & Legendre 2012 :

Contrary to PCA, PCoA, or CA, which are eigenvector-based methods, nMDS calculations do not maximize the variability associated with individual axes of the ordination

You can check and visualize the quality of the representation in 2 dimensions with a “Shepard” diagram that represent the distance in the 2 dimensions of an ordination with the original distance in the k dimensional space. Here is an example to compare the quality of the representation in 2 dimensions of a nMDS and a MDS (PCoA) based on the Bray-Curtis distance.

Shepard diagrams : comparison of the distance between points in the full dimensions of the NMDS (= cophenetic distance) with the true Hellinger distances in full space. We also compute a cophenetic correlation coefficient based on the Spearman correlation coefficient because the NMDS method preserves the ranks order of the points and not their true distance.

```
# dev.new(width = 18/2.54, height = 12/2.54)

par(mfrow = c(2,3), mar = c(3.5,3.5,2.5,1), mgp = c(2, 0.6, 0), cex = 0.7, las = 1)
for(k in 1:6){
  NMDS <- MASS::isoMDS(hell_dist, k = k)
  corr <- cor(dist(NMDS$points), hell_dist, method = "spearman")
  plot(x = dist(NMDS$points), y = hell_dist, cex = 0.5,
       ylab = "distance in NMDS dimensions", xlab = "True Hellinger distance")
  mtext(text = paste0("Number of NMDS dimensions : ", k, "\nCophenetic Corr. = ",
                      round(corr,2)),
        side = 3, cex = 0.8)
}
```

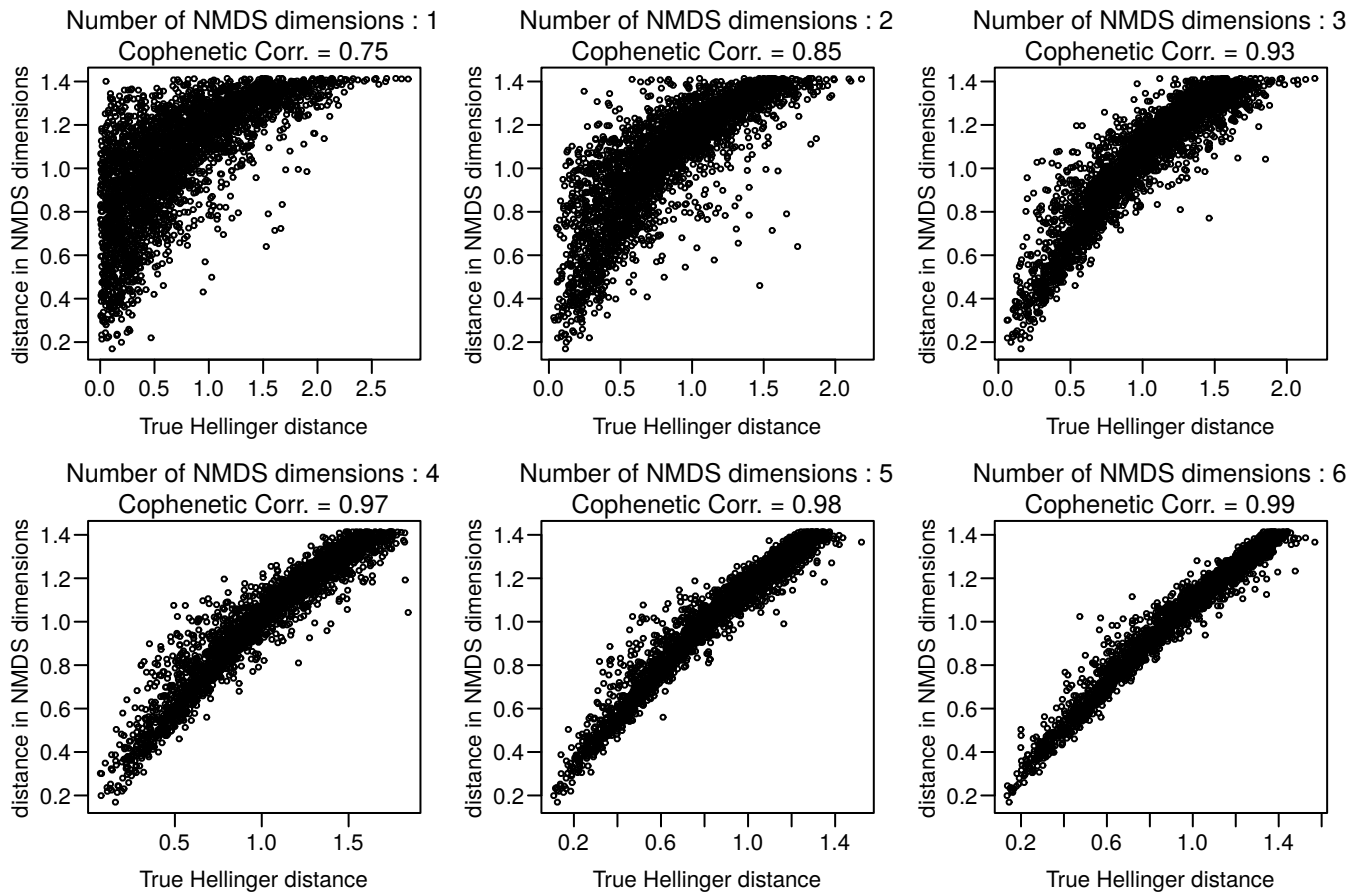


Figure 25:

To decide how much dimensions you need you might also plot the stress as a function of the number of dimensions. Note that, in contrast with a traditional scree plot, each bar does not represent the variance associated with each axis but the total stress (a function of the squared difference between the cophenetic distance in the number of dimensions of the NMDS and \hat{d} is the real distance in the full space of the original data) for all the dimensions. For example, the “3Dim” bar represent the stress of a solution in 3 dimensions, not the stress associated with the 3rd axis...

Here there are 2 stronger drops after 2 and 3 dimensions

```
n = 6
stress <- vector(length = n)
for (i in 1:n) {
  stress[i] <- MASS::isoMDS(hell_dist, k = i)$stress
}
names(stress) <- paste0(1:n, "Dim")

# dev.new(width = 10/2.54, height = 7/2.54)
par(mar = c(3.5,3.5,1,1), mgp = c(2, 0.6, 0), cex = 0.8, las = 2)
barplot(stress, ylab = "stress")
```

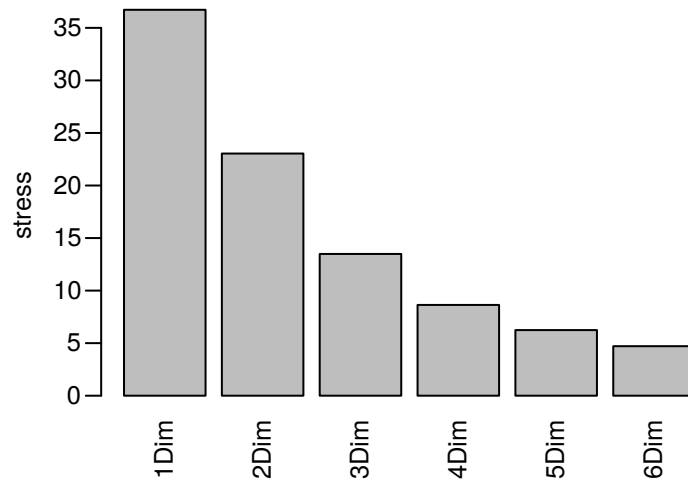



Figure 26: