

Principal Component Analysis (PCA)

A.B. Dufour

Contents

1	Introduction	2
2	A full example	2
2.1	Preparing the dataset	2
2.2	Visualizing the data in a 3-dimensional space	3
2.3	Centring and reducing the variables	4
2.3.1	Centring the data	5
2.3.2	Reducing the data	6
3	The global shape of the cloud of points	6
3.1	Normed PCA in <code>ade4</code>	8
3.2	Scatter plots in <code>ade4</code>	12
3.2.1	Scatter plot of individuals	12
3.2.2	Scatter plot of variables	14
3.2.3	Simultaneous plot of individuals and variables	15
4	Your turn!	16
5	Conclusion	17
	References	17


1 Introduction

Multivariate Analyses or Exploratory Data Analyses gather all eigenanalyses such as Principal Component Analysis (PCA), Correspondence Analysis (CA), or Canonical Correlation Analysis (CCA). An eigenanalysis is a mathematical operation on a square symmetric matrix, and is therefore central for linear algebra. The output of an eigenanalysis consists of a series of eigenvalues and eigenvectors (each eigenvalue corresponding to an eigenvector). Eigenvalues are ranked from the greatest to the least, and measure the amount of variation along an axis. For instance, in the case of PCA, eigenvalues have a strong meaning: they represent the explained variance.

All eigenanalyses (from one table to a set of tables) are derived from a common mathematical framework, called the **duality diagram**. This framework was proposed in the early 1970's ([3], [2]) and was introduced 10 years after in Ecology [7]. The duality diagram is characterized by a triplet of three matrices:

- ★ **X** containing the dataset with p variables measures on n observations,
- ★ **D** a matrix of weights on the observations, most often diagonal,
- ★ **Q** a matrix of weights on the variables, diagonal.

In classical methods, one doesn't need to specify the two matrices **D** and **Q** (default option for each). They are however tools for more sophisticated analyses.

The  package `ade4` is the most complete software for exploratory data methods displayed in the duality diagram scheme.

<code>dudi.pca</code>	principal component analysis
<code>dudi.coa</code>	correspondence analysis
<code>dudi.acm</code>	multiple correspondence analysis
<code>dudi.fca</code>	fuzzy correspondence analysis
<code>dudi.mix</code>	mixed analysis: numeric and factors
<code>dudi.nsc</code>	non symmetric correspondence analysis

2 A full example

2.1 Preparing the dataset

Let's analyse a famous dataset on Painted Turtles ("tortues"), which can be found in `ade4` [8]. "Long" corresponds to the length of the animal, "larg" to its width, "haut" to its height, "sexe" to its sex.

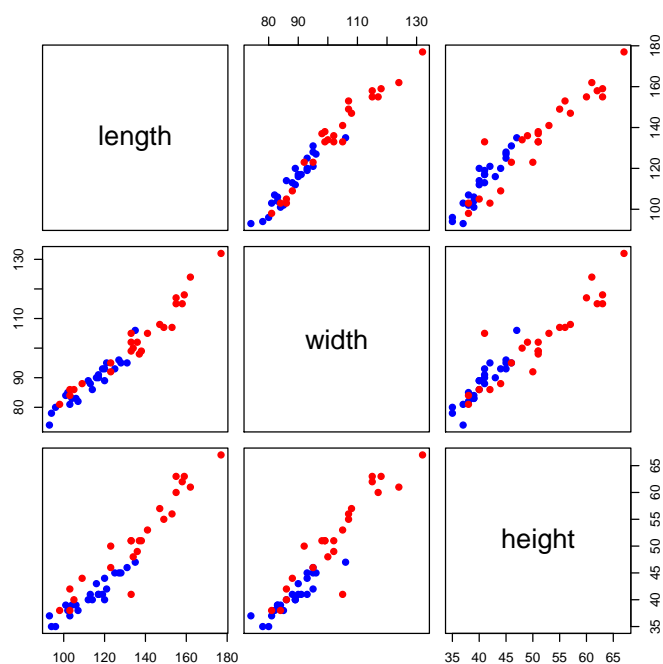
```
library(ade4)
data(tortues)
names(tortues)
[1] "long" "larg" "haut" "sexe"
pturtles <- tortues
names(pturtles) <- c("length", "width", "height", "sex")
```

We create a factor containing each individual's sex and generate a colour vector to distinguish male and female.

```
sex <- pturtles$sex
sexcol <- ifelse(sex == "M", "blue", "red")
```

We extract from the dataset all the continuous variables: the length, the width and the height of the turtle carapace. All variables are expressed in centimeters.

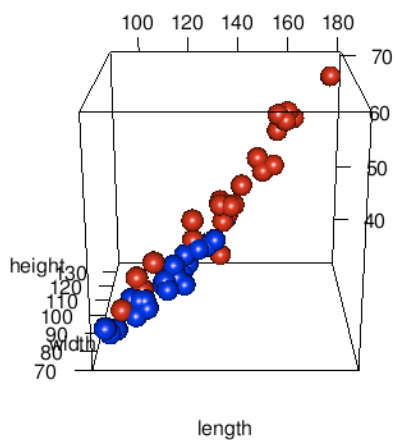
```
measures <- pturtles[, 1:3]
plot(measures, col = sexcol, pch = 19)
```



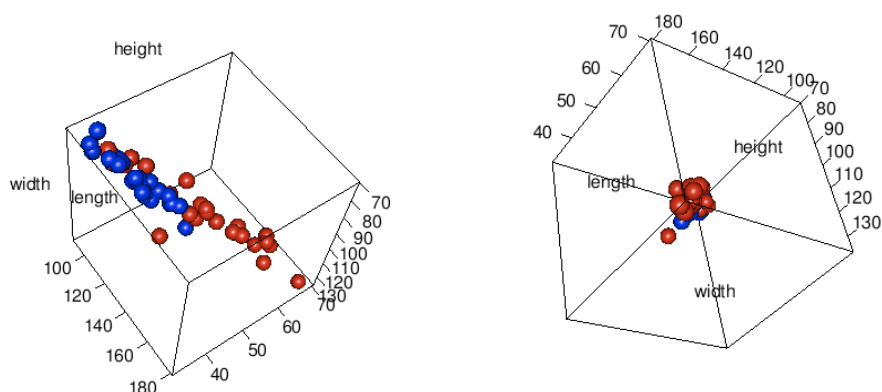
2.2 Visualizing the data in a 3-dimensional space

Each turtle is characterized by three variables i.e. a point in \mathbb{R}^3 . The `plot3d()` function of the `rgl` function [1] allows to see in a 3-dimensional space.

```
library(rgl)
plot3d(measures, type = "s", col = sexcol)
```



Let's examine the representation from every angle, by turning it with the cursor.



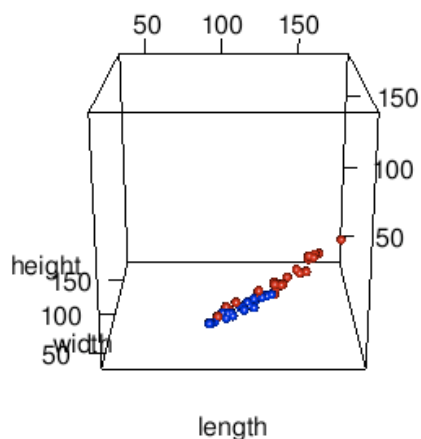
2.3 Centring and reducing the variables

The previous plots are misleading because of the different scales of the variables (e.g., height varying from 30 to 70, length from 100 to 180, width from 70 to 130). At first, we need to define and fix a common scale.

```

lims <- c(min(measures), max(measures))
plot3d(measures, type = "s", col = sexcol, xlim = lims, ylim = lims,
       zlim = lims)

```



In this new plot all the variables are represented using the same scale (e.g., from 50 to 150). Something looks wrong however, and this is because of the different means of variables.

```
sapply(measures, mean)
  length  width  height
124.68750 95.50000 46.14583
```

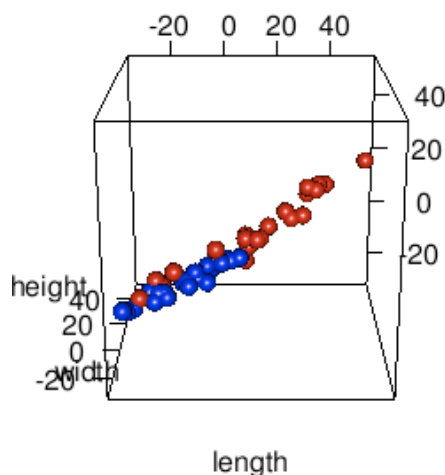
Because of these differences, points are completely located at the bottom of the 3-D graph.

2.3.1 Centring the data

This operation consists in subtracting the mean of each variable to each point. The `scale()` function allows to compute the centring of all variables in one command.

```
measures.c <- scale(measures, center = TRUE, scale = FALSE)

lims <- c(min(measures.c), max(measures.c))
plot3d(measures.c, type = "s", col = sexcol, xlim = lims, ylim = lims,
       zlim = lims)
```



The plot definitely looks better. The cloud of points is now centred around the origin point (0,0,0). But, we can now see that the variability is more important for the length than for the two other variables.

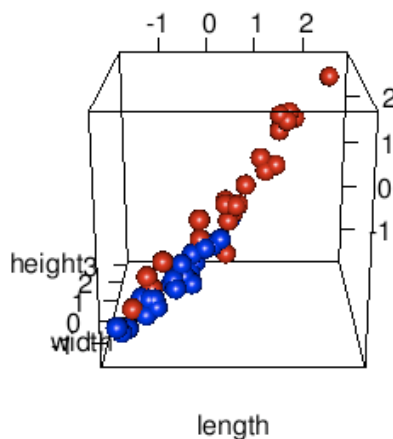
```
sapply(as.data.frame(measures.c), sd)
      length      width      height
20.481602 12.716215  8.384356
```

2.3.2 Reducing the data

Let's now reduce the centred data, by dividing each point by the standard deviation of each variable. The `scale()` function still allows this operation.

```
measures.cr <- scale(measures)

lims <- c(min(measures.cr), max(measures.cr))
plot3d(measures.cr, type = "s", col = sexcol, xlim = lims, ylim = lims,
       zlim = lims)
```



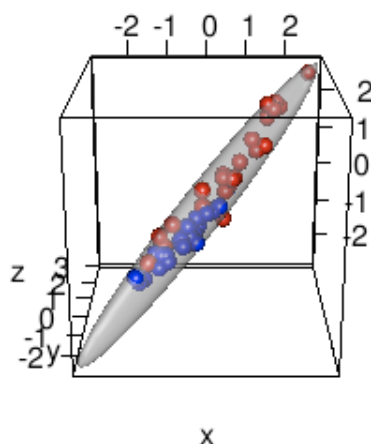
A classical PCA deals with normed variables (i.e., variables which have been centred and reduced).

3 The global shape of the cloud of points

The global shape of the cloud of points looks like an ellipsoid.

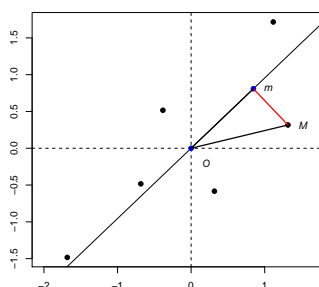
```
plot3d(ellipse3d(cor(measures.cr)), col = "grey", alpha = 0.5)
plot3d(measures.cr, type = "s", col = sexcol, xlim = lims, ylim = lims,
       zlim = lims, add = TRUE)
```





An ellipsoid is defined by three axes. From a statistical point of view, this refers to the three variances (for each variable) and the three covariance between each set of two variables (xy , xz , yz). The covariance matrix is thus enough to draw the global shape.

Let's turn the previous representation to show the cloud of points on the planes defined by the axes (1,2), (1,3) and (2,3). If you had to choose between these three planes, you would probably keep the first one, since the loss of information is not too bad. In other words, the projection of all points are spread out in the plane - one says that we keep "the maximum of the total inertia (i.e. total variance)". By doing so, we performed a handmade PCA.



Axes are created such that the perpendicular distance from each individual M to the ordination axes is minimized (red line). Axes are linear combination of variables. The weights are known as 'coefficients' or 'loadings'.

- The sum of all distances \overrightarrow{OM} represents the total inertia or the total variance.
- The sum of all distances \overrightarrow{Mm} is called the mechanical inertia (to be minimized).
- The sum of all distances \overrightarrow{Om} is called the statistical inertia (to be maximized).

3.1 Normed PCA in ade4

Let's use the `dudi.pca()` function from `ade4` ([4], [6], [5]) to compute a normed PCA.

```
library(ade4)
pca1 <- dudi.pca(measures, scann = FALSE, nf = 3)
names(pca1)
[1] "tab" "cw" "lw" "eig" "rank" "nf" "c1" "l1" "co" "l1" "call"
[12] "cent" "norm"
```

Have a look at the command: we here used two options: `scann = FALSE` and `nf = 3` to keep all the eigenfactors. We can also use the eigenvalues barplot to select the number of axes. Eigenvalues represent the fractions of the global decomposed variability on each successive axis. Let's try the following command:

```
pca2 <- dudi.pca(measures)
```

Let's answer the question "Select the number of axes:".

The object given by the `dudi.pca()` function contains a lot of elements. Let's have a look at them.

1. The dataframe `tab` contains the initial data after centring and/or reduction. One sees some differences with the `scale()` results.

```
tail(pca1$tab)
  length width height
43 1.495648 1.549703 2.031465
44 1.495648 1.708647 1.669869
45 1.643671 1.549703 1.910933
46 1.693012 1.788119 2.031465
47 1.841035 2.264950 1.790401
48 2.581150 2.900726 2.513592

tail(measures.cr)
  length width height
43 1.479987 1.533475 2.010192
44 1.479987 1.690755 1.652383
45 1.626460 1.533475 1.890922
46 1.675284 1.769394 2.010192
47 1.821757 2.241233 1.771653
48 2.554122 2.870351 2.487271
```

This is due to the use of the variance: $\frac{1}{n}$ in `dudi.pca()` vs $\frac{1}{n-1}$ in `scale()`. Using one or the other does not matter when it comes to performing PCA, as long as one knows which one is used.

2. The `cw` vector gives the column weights, i.e. the weight attributed to each variable. By default, each variable has a weight equal to 1.

```
pca1$cw
[1] 1 1 1
```

3. The `lw` vector gives the row weights, i.e. the weight attributed to each row (generally an individual). By default, the distribution is uniform and each individual has a weight equal to $\frac{1}{n}$.

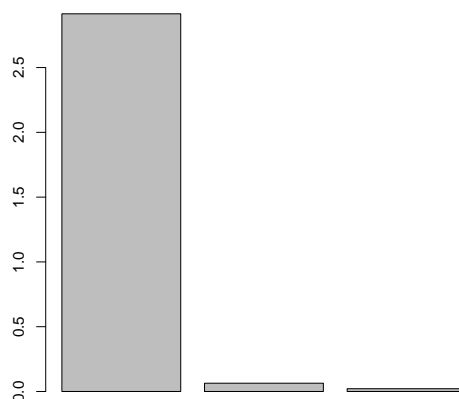
```
head(pca1$lw)
```



```
[1] 0.02083333 0.02083333 0.02083333 0.02083333 0.02083333 0.02083333
head(pca1$lw) * nrow(measures)
[1] 1 1 1 1 1 1
```

4. The `eig` vector gives the eigenvalues computed in the smallest dimension (p or n). We therefore can draw the bar plot of the eigenvalues also called `screeplot`.

```
pca1$eig
[1] 2.91474158 0.06390899 0.02134943
sum(pca1$eig)
[1] 3
barplot(pca1$eig)
```



The eigenvalues inform us on the inertia kept by each axis:

```
(kip <- 100 * pca1$eig/sum(pca1$eig))
[1] 97.1580527 2.1302996 0.7116477
cumsum(kip)
[1] 97.15805 99.28835 100.00000
```

Looking at the outcome of the PCA performed, we see that the first axis extracts nearly all the information (97.16% especially when performed on normed variables). The first two axes kept 99.29% of the total inertia. Viewing the data in a 2-dimensional space instead of a 3-dimensional space allows to see nearly all the contained information.

The sum of the eigenvalues will equal the number of variables for a normed P.C.A. and will equal the sum of variance of all variables for a centred P.C.A..

5. `rank` is an integer giving the rank of the analysed matrix i.e. in that case, the number of independent variables. Let's suppose that one new variable ("newvar") is already a combination of some other, known variables.

```
pca1$rank
[1] 3

newvar <- 2 * measures[, 1] - 0.5 * measures[, 3]
newmeasures <- cbind(measures, newvar)
head(newmeasures)

  length width height newvar
1     93   74    37  167.5
2     94   78    35  170.5
3     96   80    35  174.5
4    101   84    39  182.5
5    102   85    38  185.0
6    103   81    37  187.5

dudi.pca(newmeasures, scann = F, n = 3)$rank
[1] 3
```

6. `nf` is an integer giving the number of axes kept.

```
pca1$nf
[1] 3
```

7. `c1` gives the variables' coordinates, normed to 1. It is also called the coefficients of the combination or the loadings of variables.

```
pca1$c1

      CS1      CS2      CS3
length -0.5806536 -0.2706983  0.7678306
width  -0.5780575 -0.5270479 -0.6229526
height -0.5733158  0.8055699 -0.1495531

sum(pca1$cw * pca1$c1$CS1^2)
[1] 1
```

8. `l1` gives the individuals' coordinates, normed to 1. It is also called the loadings of individuals.

```
head(pca1$l1)

      RS1      RS2      RS3
1  1.4804683  1.72364615  0.1969053
2  1.4370059  0.23990810 -0.6523719
3  1.3496269 -0.19712944 -0.8114491
4  0.9961844  0.41229065 -1.3637908
5  0.9929707 -0.19031038 -1.3199609
6  1.1242981  0.03551319  0.4179949

sum(pca1$lw * pca1$l1$RS1^2)
[1] 1
```

9. `co` gives the variables' coordinates, normed to the square root of the eigenvalues.

```
pca1$co
```

```

      Comp1      Comp2      Comp3
length -0.9913274 -0.06843314  0.11219115
width  -0.9868953 -0.13323892 -0.09102238
height -0.9787999  0.20364991 -0.02185187

```

```
sum(pca1$cw * pca1$co$Comp1^2)
```

```
[1] 2.914742
```

The link between `c1` and `co` is defined by:

```
pca1$c1$CS1 * sqrt(pca1$eig[1])
```

```
[1] -0.9913274 -0.9868953 -0.9787999
```

```
t(t(pca1$c1) * sqrt(pca1$eig))
```

```

      CS1      CS2      CS3
length -0.9913274 -0.06843314  0.11219115
width  -0.9868953 -0.13323892 -0.09102238
height -0.9787999  0.20364991 -0.02185187

```

10. `li` gives the individuals' coordinates, normed to the square root of the eigenvalues. It's also called the individuals' scores.

```
head(pca1$li)
```

```

      Axis1      Axis2      Axis3
1  2.527546  0.435741657  0.02877071
2  2.453345  0.060649312 -0.09532096
3  2.304166 -0.049834771 -0.11856443
4  1.700747  0.104228011 -0.19926954
5  1.695261 -0.048110896 -0.19286536
6  1.919471  0.008977814  0.06107509

```

```
sum(pca1$lw * pca1$li$Axis1^2)
```

```
[1] 2.914742
```

```
head(pca1$l1$RS1 * sqrt(pca1$eig[1]))
```

```
[1] 2.527546 2.453345 2.304166 1.700747 1.695261 1.919471
```

```
head(t(t(pca1$l1) * sqrt(pca1$eig)))
```

```

      RS1      RS2      RS3
1  2.527546  0.435741657  0.02877071
2  2.453345  0.060649312 -0.09532096
3  2.304166 -0.049834771 -0.11856443
4  1.700747  0.104228011 -0.19926954
5  1.695261 -0.048110896 -0.19286536
6  1.919471  0.008977814  0.06107509

```

11. `call` keeps a record of your principal component analysis.

```
pca1$call
```

```
dudi.pca(df = measures, scannf = FALSE, nf = 3)
```

The function `eval()` allows to compute again the same calculation.

```
eval(pca1$call)
```

```

Duality diagramm
class: pca dudi
$call: dudi.pca(df = measures, scannf = FALSE, nf = 3)
$nf: 3 axis-components saved
$rank: 3
eigen values: 2.915 0.06391 0.02135
vector length mode content
1 $cw 3 numeric column weights
2 $lw 48 numeric row weights
3 $eig 3 numeric eigen values

data.frame nrow ncol content
1 $tab 48 3 modified array
2 $li 48 3 row coordinates
3 $li 48 3 row normed scores
4 $co 3 3 column coordinates
5 $c1 3 3 column normed scores
other elements: cent norm

identical(eval(pca1$call), pca1)

[1] TRUE

```

12. **cent.** This vector provides the means (cent for centring) of the analysed data:

```

pca1$cent

length width height
124.68750 95.50000 46.14583

colMeans(measures)

length width height
124.68750 95.50000 46.14583

```

13. **norm.** This vector provides the standard deviations (on \sqrt{n}) of the analysed data:

```

pca1$norm

length width height
20.26713 12.58306 8.29656

sd.n <- function(x) sqrt(var(x) * (length(x) - 1)/length(x))
apply(measures, 2, sd.n)

length width height
20.26713 12.58306 8.29656

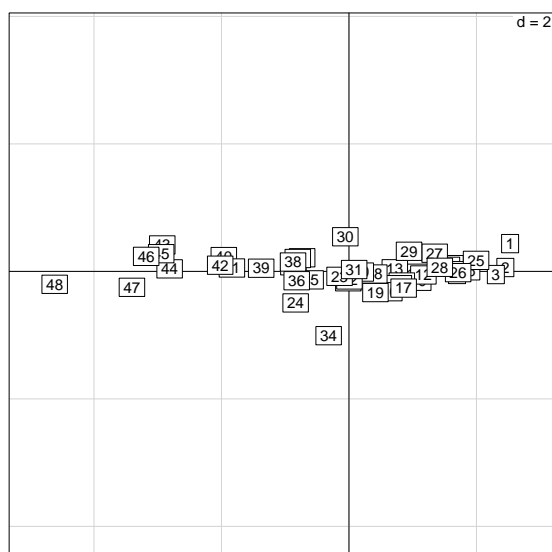
```

3.2 Scatter plots in ade4

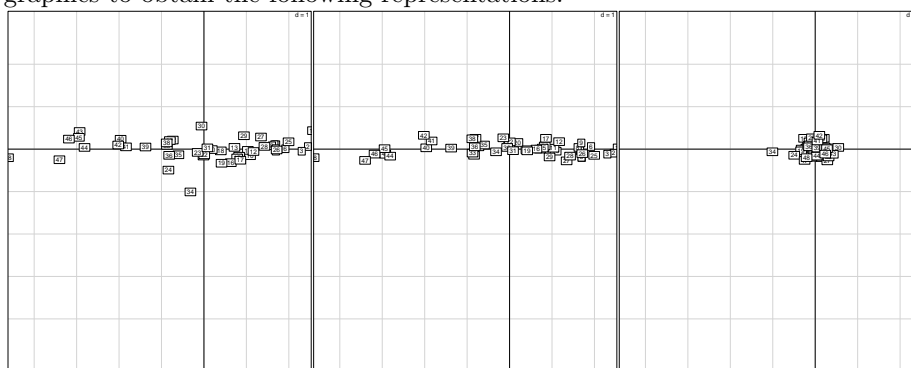
3.2.1 Scatter plot of individuals

The `s.label()` function allows to represent individuals (i.e. rows) on the different planes, such as the first plane (axes 1 and 2):

```
s.label(pca1$li, xax = 1, yax = 2)
```

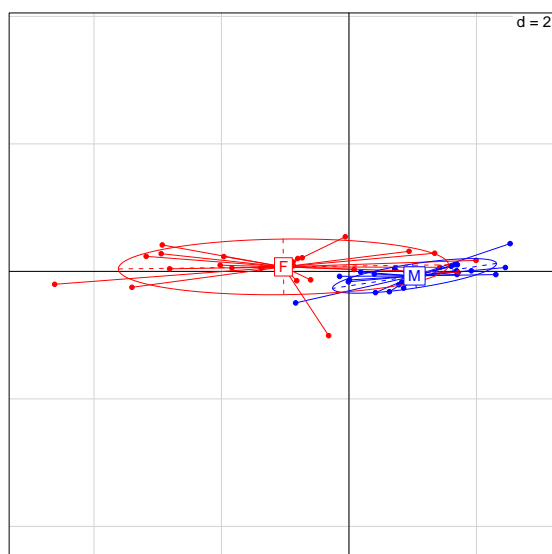


Exercise. Let's represent the 1-2, 1-3 and 2-3 planes using the same scale for all graphics to obtain the following representations.



The `s.class()` function allows adding a supplementary variable such as a factor (groups of individuals).

```
gcol = c("blue", "red")
s.class(dfxy = pca1$li, fac = sex, col = gcol, xax = 1, yax = 2)
```

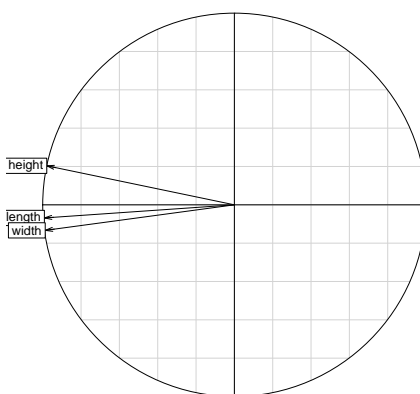


Exercise. Let's represent the groups on all possible planes.

3.2.2 Scatter plot of variables

The `s.corcircle()` function allows representing the variables in the new subspace, i.e. it allows to visualize correlations. It is especially useful in the normed P.C.A. This scatter plot is called the correlation circle.

```
s.corcircle(pca1$co, xax = 1, yax = 2)
```

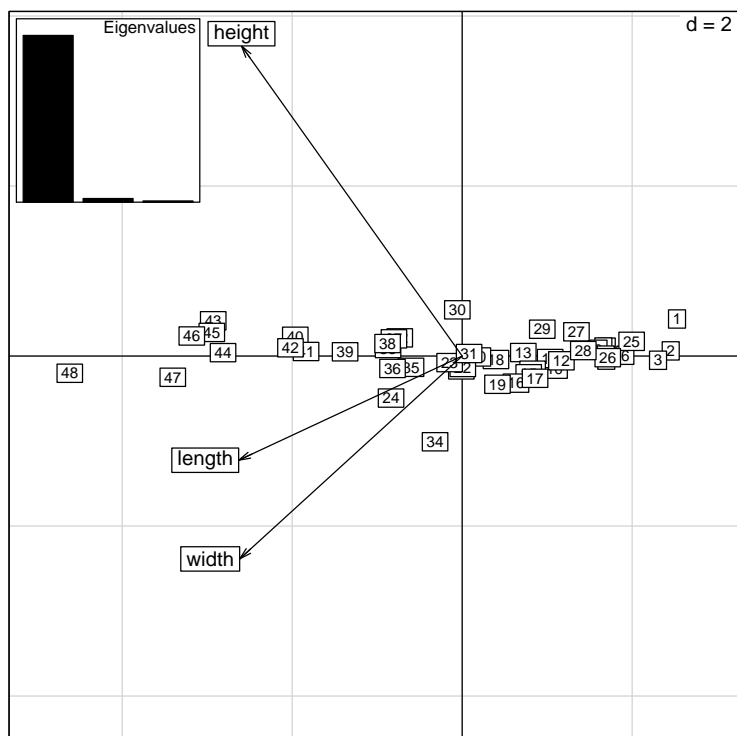


```
par(mfrow = c(1, 3))
for (i in 1:3) {
  plot(x = pca1$li[, 1], y = measures[, i], pch = 19, col = gcol,
       xlab = "First axis of the PCA", las = 1, ylab = colnames(measures)[i])
}
```

3.2.3 Simultaneous plot of individuals and variables

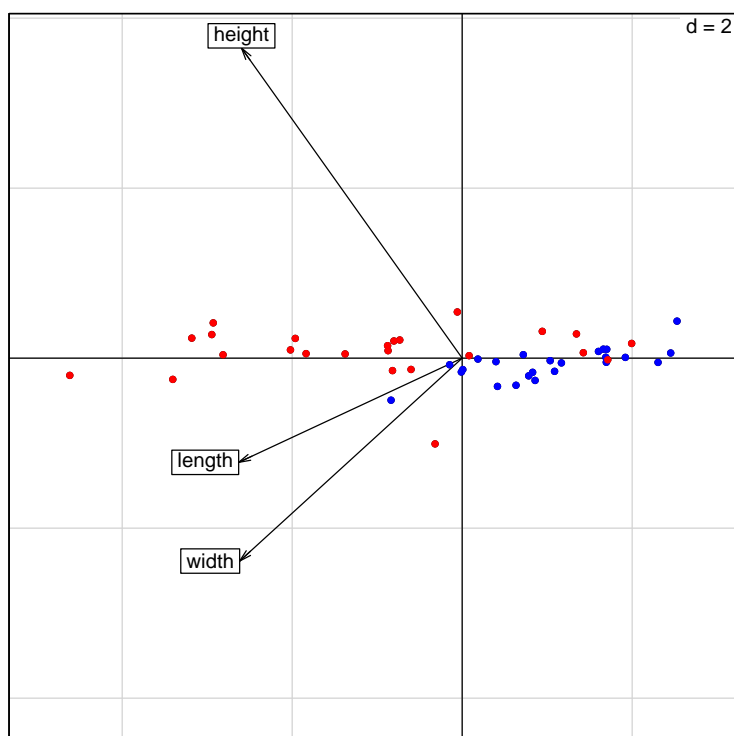
The `scatter()` function allows representing both variables and individuals. It is a generic function of the class object `dudi`.

```
scatter(pca1)
```



The plot can be improved by using the sex information.

```
scatter(pca1, clab.row = 0, posieig = "none")
NULL
s.class(pca1$li, sex, col = gcol, add.plot = TRUE, cstar = 0, clabel = 0,
        cellipse = 0)
```



4 Your turn!

The dataset "meaudret" can be found in `ade4`. It is a classical set of ecological information: three data frames containing environmental variables, Trichopters' abundance collected on 5 sites during 4 seasons along the Meaudret river [experimental design: sites and seasons].

```
data(meaudret)
names(meaudret)
[1] "mil" "plan" "fau"
names(meaudret$mil)
[1] "Temp" "Debit" "pH" "Condu" "Dbo5" "Oxyd" "Ammo" "Nitra" "Phos"
names(meaudret$fau)
[1] "Eda" "Bsp" "Brh" "Bni" "Bpu" "Cen" "Ecd" "Rhi" "Hla" "Hab" "Par" "Cae" "Eig"
names(meaudret$plan)
[1] "dat" "sta"
```

Let's note that the site 6 is not located on the river and was removed. Water samples were taken to study the following physico-chemical variables:

1. Temp Water temperature (Celcius degree)
2. Debit Flow (l/s)
3. pH pH

4. Condu Conductivity (μ S/cm)
5. Dbo5 Biological Oxygen Demand (mg/l)
6. Oxyd Oxygen (mg/l of oxgen)
7. Ammo Ammonium hydroxyde (en mg/l NH_4^+)
8. Nitra Nitrate (en mg/l NO_3^-)
9. Phos Phosphate (en mg/l PO_4^{--})

5 Conclusion

The principal component analysis is the most famous exploratory method. There is several ways to compute it on \mathbb{R} . If `fich` represents the analysed dataset and if `res` represents the PCA results, one obtains the following relationships:

Function	loadings	scores	plot
<code>dudi.pca(fich, center=T, scale=T)</code>	<code>res\$c1</code>	<code>res\$li</code>	<code>scatter(res)</code>
<code>princomp(fich, cor=T)</code>	<code>res\$loadings</code>	<code>res\$scores</code>	<code>biplot(res)</code>
<code>prcomp(fich, center=T, scale=T)</code>	<code>res\$rotation</code>	<code>res\$x</code>	<code>biplot(res)</code>

`dudi.pca` deals with the variables and/or the individuals whereas `princomp` and `prcomp` deal with the individuals only. The first method allows introducing new information (through **D** or **Q**), such as for example distances between neighbouring sites.

References

- [1] D. Adler and D. Murdoch. *rgl: 3D visualization device system (OpenGL)*, 2009. R package version 0.84.
- [2] F. Cailliez and J.P. Pagès. *Introduction à l'analyse des données*. S.M.A.S.H., Paris, 1976.
- [3] P. Cazes. *Application de l'analyse des données au traitement de problèmes géologiques*. Thèse de 3ème cycle, Faculté des Sciences de Paris, 1970.
- [4] D. Chessel, A.B. Dufour, and J. Thioulouse. The ade4 package - i: One-table methods. *R-News*, 4(1):5–10, 2004.
- [5] S. Dray and A.B. Dufour. The ade4 package: implementing the duality diagram for ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007.
- [6] S. Dray, A.B. Dufour, and D. Chessel. The ade4 package - ii: Two-table and k-table methods. *R News*, 7:47–52, 2007.
- [7] Y. Escoufier. The duality diagram : a means for better practical applications. In Legendre P. & Legendre L., editor, *Development in numerical ecology*. Springer-Verlag, Berlin, 1987.
- [8] P. Jolicoeur and J.E. Mosimann. Size and shape variation in the painted turtle. a principal component analysis. *Growth*, 24:339–354, 1960.