Data challenge & SHS: Principal component analysis and clustering in R

Julie Josse, Gaël Varoquaux, and Bénédicte Colnet

February 2021

Abstract

This is the practical class associated with the class 2 on principal component analysis and clustering. In this tutorial, you will learn how to perform a principal component analysis and how to interpret it. You will also learn how to perform a clustering on quantitative data. This notebook makes an intensive use of the package `FactoMineR`. Interpretation of the results remains the most important part of this tutorial.

Contents

Aknowledgments: François Husson class on youtube, the book "R pour la statistique et la science des données".

```
# Load all packages needed to execute the job
# If the packages are not installed, write
# install.packages("<name of package>")

library(ggplot2) # plot
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
# Clear any existing variables
rm(list = ls())

# Set seed for reproducibility
set.seed(123)
```

Principal component analysis

Illustrative example

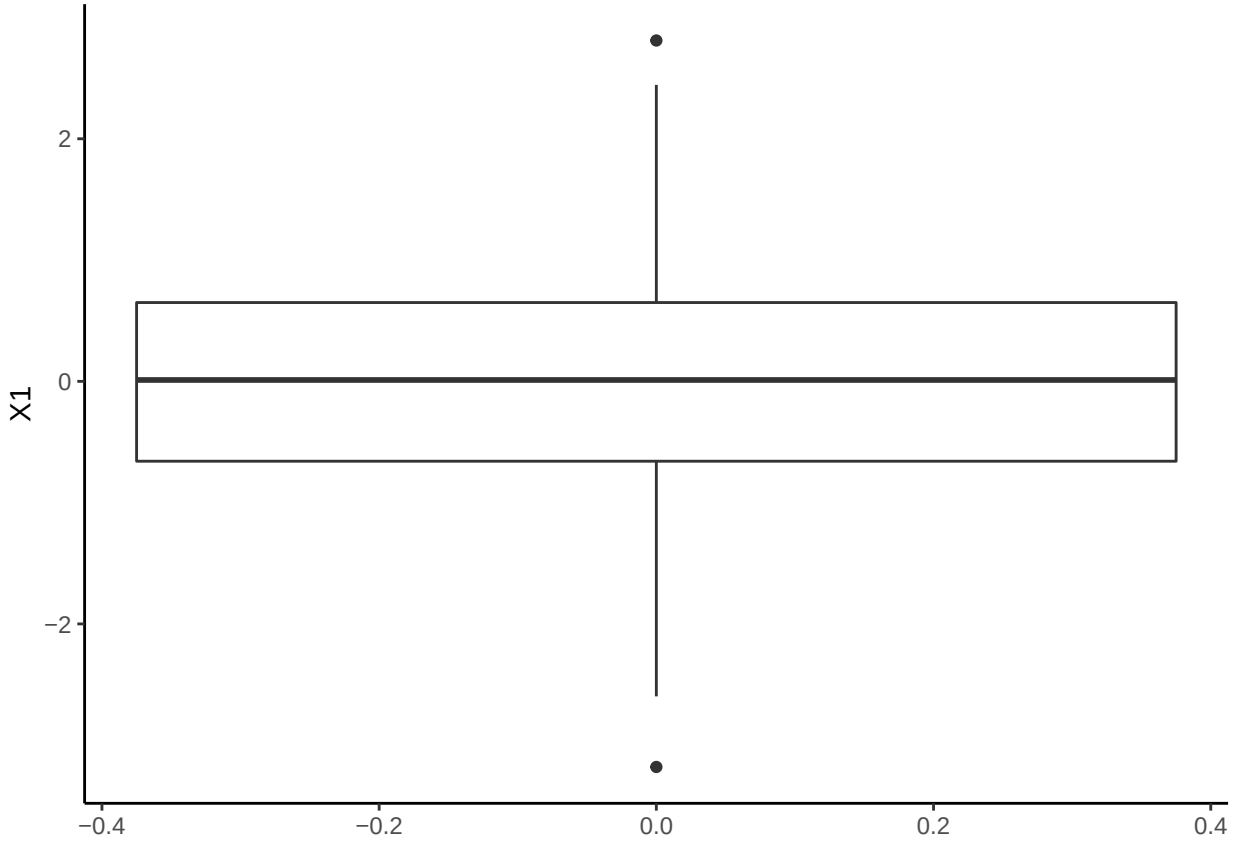
Before going into details, let us look at a funny example. Imagine that I generate two variables X_1 and X_2 from normal distributions. We want these variables to be linked (correlated) and such that $X_j \sim \mathcal{N}(0,1)$. The following chunk performs the simulation. You can take the output data frame and explore the data first with univariate analysis. And then with a bivariate plot.

Remark: An outlier is in the dataset. Can you recover it?

```
library(MASS) # for simulations
Sigma <- matrix(c(1,0.8,1,0.8),2,2)
simulated_data <- mvrnorm(n = 500, mu = c(0,0), Sigma)
output <- data.frame(simulated_data)
names(output) <- c("X1", "X2")
output[501,] <- c("X1" = 2, "X2" = -2) # outlier step
```

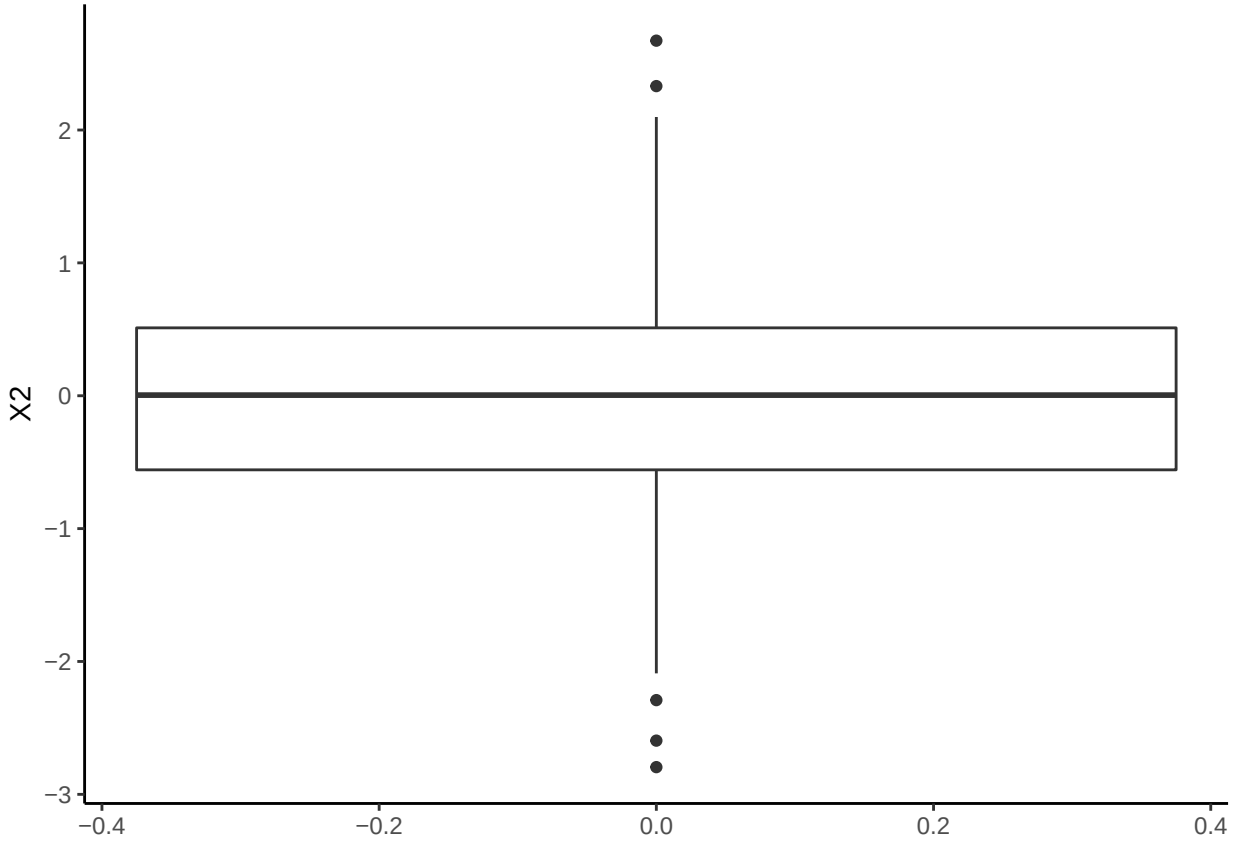
Solution

```
library(ggplot2)
ggplot(output, aes(y = X1)) +
  geom_boxplot() +
  theme_classic()
```

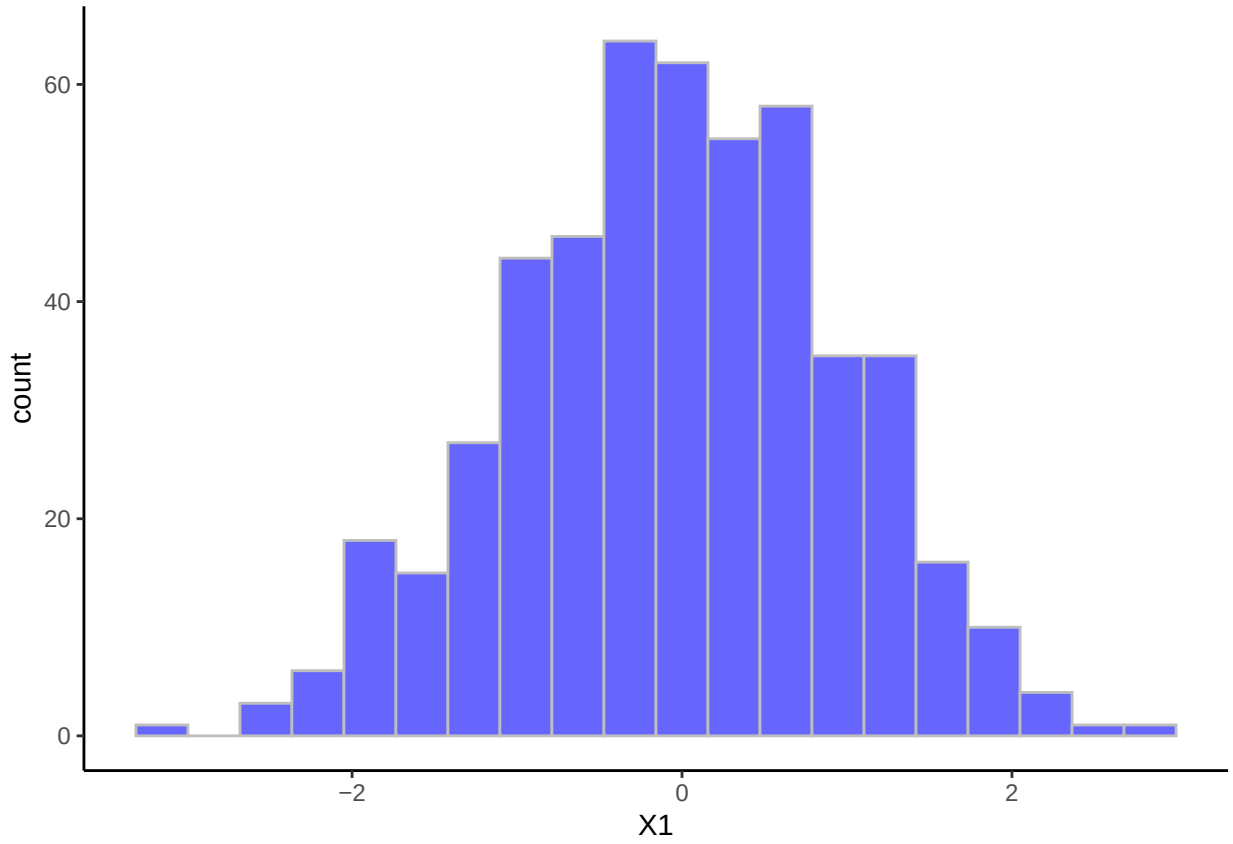



```
ggplot(output, aes(y = X2)) +
  geom_boxplot() +
```

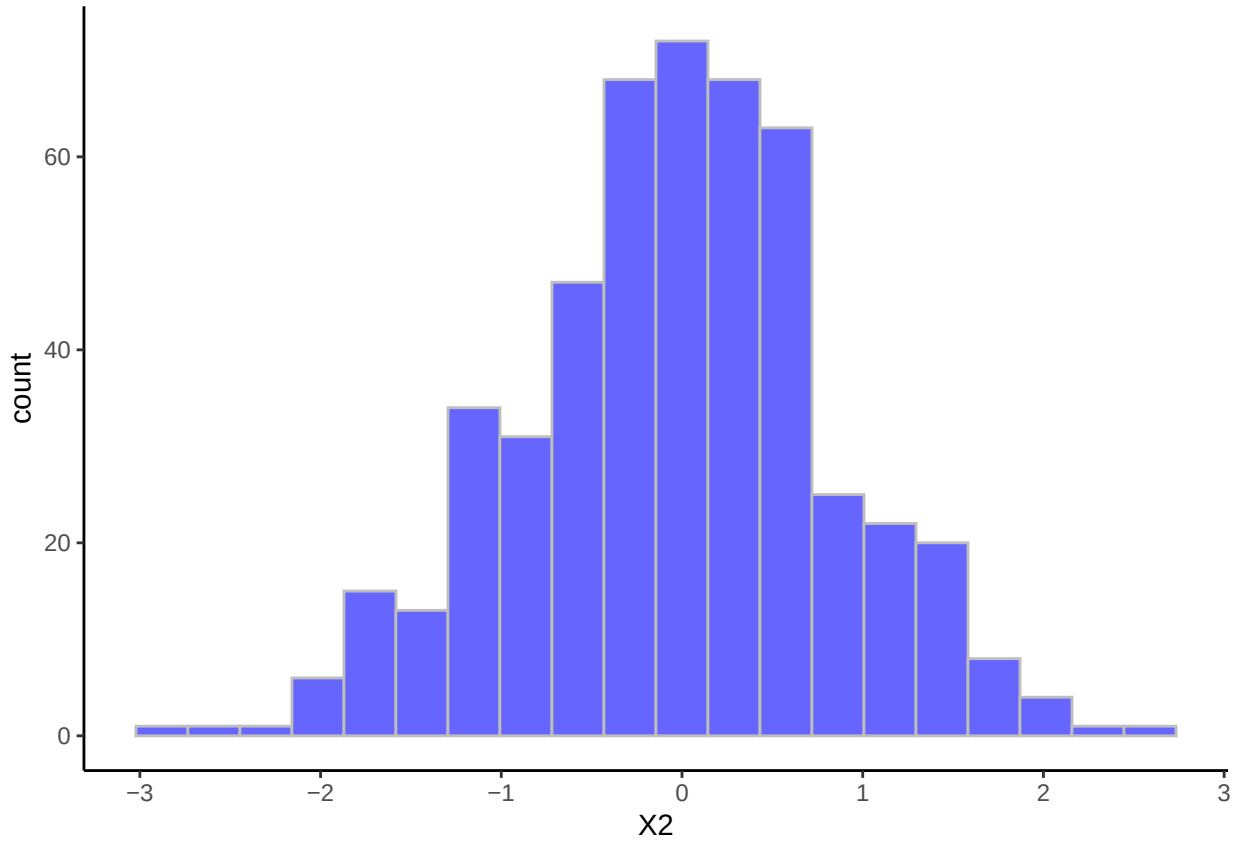
```
  theme_classic()
```



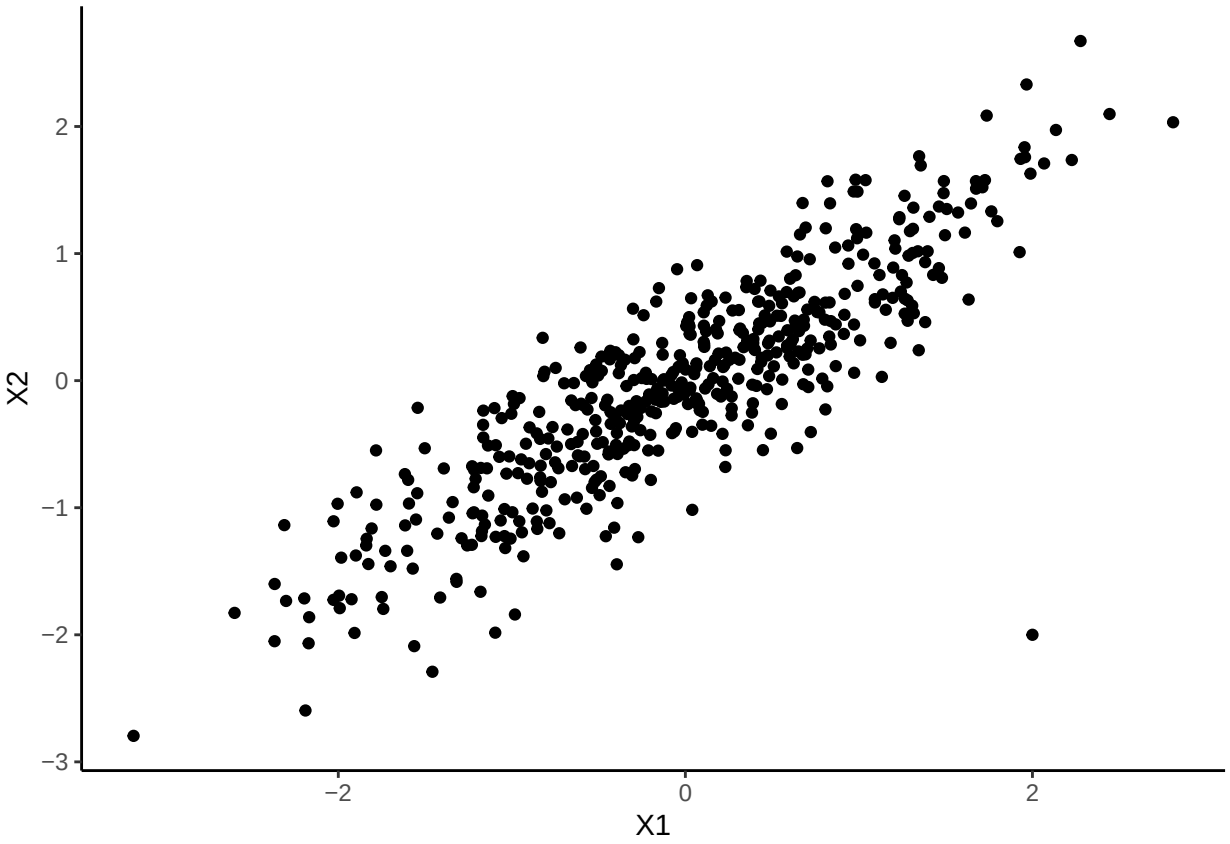

```
ggplot(output, aes(x = X1)) +
  geom_histogram(bins = 20, fill = "blue", alpha = 0.6, color = "grey") +
  theme_classic()
```




```
ggplot(output, aes(x = X2)) +
  geom_histogram(bins = 20, fill = "blue", alpha = 0.6, color = "grey") +
  theme_classic()
```




```
ggplot(output, aes(x = X1, y = X2)) +
  geom_point() +
  theme_classic()
```



The outlier is clearly identifiable on this scatter plot, but not using only the boxplot or any univariate tool. This is to highlight that PCA will allow us to see high dimensional outliers. In other words, PCA will allow you to observe multidimensional outliers.

End of solution

General introduction

Context

Principal Component Analysis (usually the shortname is PCA but you can also find ACP in French) focuses on typical data you can find in several domains: *observations* (or individus) in rows, and *variables* in column. Note that the PCA focuses on *quantitative* variables (for example age, or price, but not color or sex). For example we can study the average temperature depending on cities. In that case cities are rows, and in column the average temperature per month.

Typical question an ACP answers

A typical question you may ask on your data is: how much the different observations are close to one another considering the variables? (remember that everything you will conclude depends on these variables that you added in your initial model) You can also see PCA as a way to find a low-dimensional representation that captures the "essence" of high-dimensional data

What can you interpret from data?

The PCA will group similar individuals together. Information are also learned on variables, with the correlated variables (meaning that you have a linear link between two variables), and also which variables synthetize the most the observations, or which variables bring different informations.

Package

In this notebook we propose to use the package `FactoMineR` and the function `PCA`.

An example: the decathlon data set

The data set is based on the decathlon results during the Athene's olympic games and the Décastar (another competition). For each athletes the data set contains the results in the 10 tests, with the total number of points and ranking. The competition in which the athlete participated is also mentioned.

For both competitions, the following information is available for each athlete: performance for each of the 10 events, total number of points (for each event, an athlete earns points based on performance; here the sum of points scored), and final ranking. The events take place in the following order: 100 meters, long jump, shot put, high jump, 400 meters (first day) and 110 meter hurdles, discus, pole vault, javelin, 1500 meters (second day).

The overall objective of this exercice is to characterize the athletes and their differences, and to observe it tests evaluate similar skills or different ones. The aim of conducting PCA on this dataset is to determine profiles for similar performances: are there any athletes who are better at endurance events or those requiring short bursts of energy, etc? And are some of the events similar? If an athlete performs well in one event, will he necessarily perform well in another?

Question 1

First, load the data and inspect the data (for example which variables are quantitative or qualitative?).

Remark: This step is the first step you should do before any data analysis, and not only PCA.

Solution question 1

```
# Load data
decathlon <- read.csv(file = "decathlon.csv", row.names=1)
dim(decathlon)
```

```
## [1] 41 13
```

```
summary(decathlon)
```

```
##      X100m         Long.jump       Shot.put       High.jump         X400m
##  Min.   :10.44   Min.   :6.61   Min.   :12.68   Min.   :1.850   Min.   :46.81
##  1st Qu.:10.85   1st Qu.:7.03   1st Qu.:13.88   1st Qu.:1.920   1st Qu.:48.93
##  Median :10.98   Median :7.30   Median :14.57   Median :1.950   Median :49.40
##  Mean   :11.00   Mean   :7.26   Mean   :14.48   Mean   :1.977   Mean   :49.62
##  3rd Qu.:11.14   3rd Qu.:7.48   3rd Qu.:14.97   3rd Qu.:2.040   3rd Qu.:50.30
##  Max.   :11.64   Max.   :7.96   Max.   :16.36   Max.   :2.150   Max.   :53.20
##   X110m.hurdle       Discus        Pole.vault       Javeline
##  Min.   :13.97   Min.   :37.92   Min.   :4.200   Min.   :50.31
##  1st Qu.:14.21   1st Qu.:41.90   1st Qu.:4.500   1st Qu.:55.27
##  Median :14.48   Median :44.41   Median :4.800   Median :58.36
##  Mean   :14.61   Mean   :44.33   Mean   :4.762   Mean   :58.32
##  3rd Qu.:14.98   3rd Qu.:46.07   3rd Qu.:4.920   3rd Qu.:60.89
##  Max.   :15.67   Max.   :51.65   Max.   :5.400   Max.   :70.52
##      X1500m           Rank           Points       Competition
##  Min.   :262.1   Min.   : 1.00   Min.   :7313   Decastar:13
##  1st Qu.:271.0   1st Qu.: 6.00   1st Qu.:7802   OlympicG:28
##  Median :278.1   Median :11.00   Median :8021
##  Mean   :279.0   Mean   :12.12   Mean   :8005
##  3rd Qu.:285.1   3rd Qu.:18.00   3rd Qu.:8122
##  Max.   :317.0   Max.   :28.00   Max.   :8893
```

All the variables are quantitative except the competition type. Not that if you don't add the item `row.names=1`*, then you will have an additional variable being the name of the participant. It is not a problem, but don't forget to remove it when doing the PCA or else. Probably the simplest solution is to have it as row names. The commant* `dim` *also inform us on the number or observations. here we have 41 different observations.*

End of solution question 1

Question 2

Apply a PCA on the data using the function from FactoMineR, and interpret it.

Tips: - First install the package.

- The appropriate function is called PCA.
- You can check if this function does or not the normalization step going in the documentation (`?PCA`).
- Why are normalization and reduction an important step?
- Explain your choices for the active and illustrative variables/individuals? (because you don't have to use all the variables to perform the PCA, you can only run it on a subset of variables that makes more sens.)
- When you interpret the data, you can also do a bar plot of the eigenvectors found by the PCA. For this purpose you can use the result object of the PCA analysis, and look at the `eig` component of this object. You can plot this using the `barplot` function or ggplot2 (which is a little bit more challenging, but a good exercice)

Solution question 2

First, check that the package is installed and don't forget to call the library. Usually in a notebook all the librairy calls are at the beginning of the notebook for more clarity. Also, it can be useful to recall why you call a library, so that you do not end up with a long list of packages where you don't remember the purpose.

```
# Install the package
#install.packages("FactoMineR", dependencies = TRUE)

library(FactoMineR) # package for PCA
```

```
## Warning: package 'FactoMineR' was built under R version 3.6.2
```

```
?PCA
```

Discussion on normalization

The normalization is an important step as it allows to compare all the variables with the same importance. For example imagine we have a data set with two variables A and B. The variable A is in kg and the other B in g, then the variable in g will count more in the distance. We recall that the distance between two observations is given with $d = (a_i - a_j)^2 - (b_i - b_j)^2$

Therefore if an identical difference in weights will be counted differently 0.2kg squarred or 200 squarred. Because in this data set the data have different unis we have no choice but to center the data and normalize it. Note that this is automatically done with the `scale.unit = TRUE` *command.*

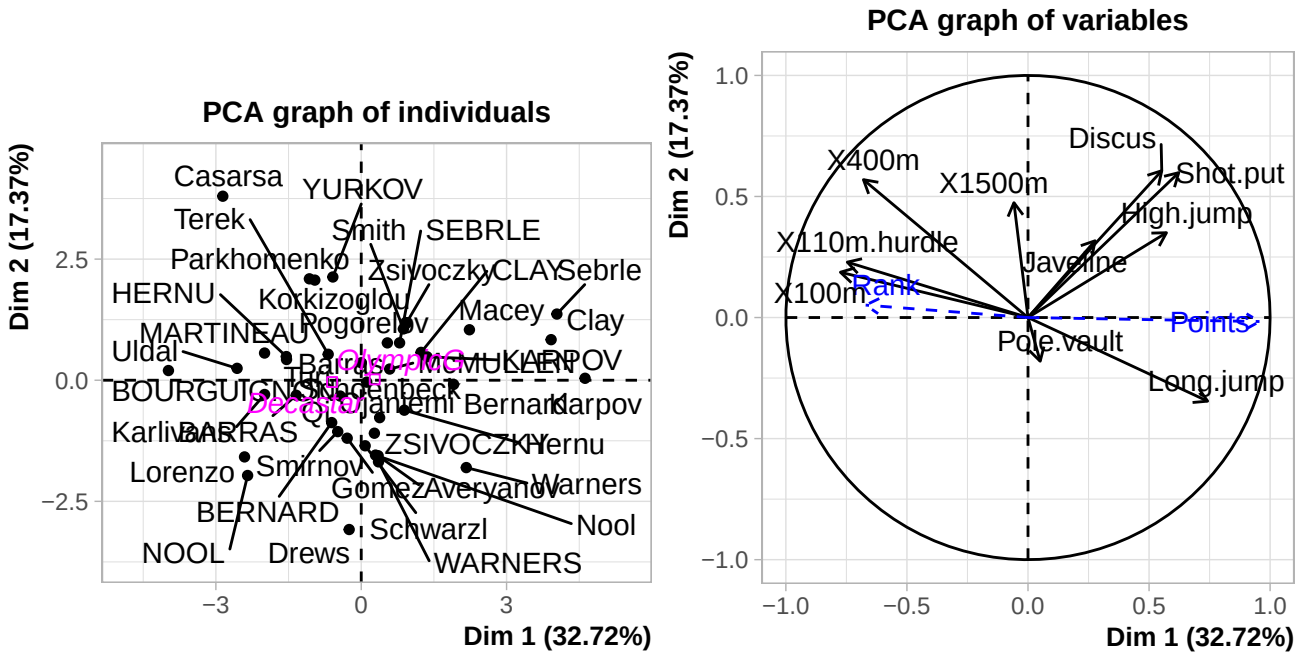
In our specific example when the data is standardized, it is possible to compare two variables with different units and to say sentences such as "Paul is more remarkable by his performance on 100m than John is by is X400m". With a value above 2, it means that the performance is way beyond average for example.

Which variables matter?

Only the result at each test matters. In fact, to obtain a typology of the athletes based on their performances for the 10 decathlon events, such as "two athletes are close since they have similar performance profiles", the distances between two athletes are defined on the basis of their performances in the 10 events. Thus, only the

performance variables are considered active; the other variables (number of points, rank, and competition) are supplementary. Here, the athletes are all considered as active individuals.

```
res.PCA <- PCA(decathlon, # data used
               scale.unit = TRUE, # scale the data, true by default
               graph = TRUE, # plot the graph, true by default
               quanti.sup = c(11:12), # additional quantitative variables
               quali.sup = 13) #additional qualitative variables
```



Outputs can bw summarized with the function summary, and for example with the first 2 dimensions

```
summary(res.PCA)
```

```
##
## Call:
## PCA(X = decathlon, scale.unit = TRUE, quanti.sup = c(11:12),
##      quali.sup = 13, graph = TRUE)
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
## Variance               3.272   1.737   1.405   1.057   0.685   0.599   0.451
## % of var.             32.719  17.371  14.049  10.569   6.848   5.993   4.512
## Cumulative % of var.  32.719  50.090  64.140  74.708  81.556  87.548  92.061
##                        Dim.8   Dim.9  Dim.10
## Variance               0.397   0.215   0.182
## % of var.              3.969   2.148   1.822
## Cumulative % of var.  96.030  98.178 100.000
##
## Individuals (the 10 first)
##                  Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## SEBRLE       |  2.369 |  0.792  0.467  0.112 |  0.772  0.836  0.106 |  0.827
## CLAY         |  3.507 |  1.235  1.137  0.124 |  0.575  0.464  0.027 |  2.141
## KARPOV       |  3.396 |  1.358  1.375  0.160 |  0.484  0.329  0.020 |  1.956
```

```
## BERNARD      |  2.763 | -0.610  0.277  0.049 | -0.875  1.074  0.100 |  0.890
## YURKOV       |  3.018 | -0.586  0.256  0.038 |  2.131  6.376  0.499 | -1.225
## WARNERS      |  2.428 |  0.357  0.095  0.022 | -1.685  3.986  0.482 |  0.767
## ZSIVOCZKY    |  2.563 |  0.272  0.055  0.011 | -1.094  1.680  0.182 | -1.283
## McMULLEN     |  2.561 |  0.588  0.257  0.053 |  0.231  0.075  0.008 | -0.418
## MARTINEAU    |  3.742 | -1.995  2.968  0.284 |  0.561  0.442  0.022 | -0.730
## HERNU        |  2.794 | -1.546  1.782  0.306 |  0.488  0.335  0.031 |  0.841
##                 ctr   cos2
## SEBRLE        1.187  0.122 |
## CLAY          7.960  0.373 |
## KARPOV        6.644  0.332 |
## BERNARD       1.375  0.104 |
## YURKOV        2.606  0.165 |
## WARNERS       1.020  0.100 |
## ZSIVOCZKY     2.857  0.250 |
## McMULLEN      0.303  0.027 |
## MARTINEAU     0.925  0.038 |
## HERNU         1.227  0.091 |
##
## Variables
##                  Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
## X100m        | -0.775 18.344  0.600 |  0.187  2.016  0.035 | -0.184  2.420
## Long.jump    |  0.742 16.822  0.550 | -0.345  6.869  0.119 |  0.182  2.363
## Shot.put     |  0.623 11.844  0.388 |  0.598 20.607  0.358 | -0.023  0.039
## High.jump    |  0.572  9.998  0.327 |  0.350  7.064  0.123 | -0.260  4.794
## X400m        | -0.680 14.116  0.462 |  0.569 18.666  0.324 |  0.131  1.230
## X110m.hurdle | -0.746 17.020  0.557 |  0.229  3.013  0.052 | -0.093  0.611
## Discus       |  0.552  9.328  0.305 |  0.606 21.162  0.368 |  0.043  0.131
## Pole.vault   |  0.050  0.077  0.003 | -0.180  1.873  0.033 |  0.692 34.061
## Javeline     |  0.277  2.347  0.077 |  0.317  5.784  0.100 | -0.390 10.807
## X1500m       | -0.058  0.103  0.003 |  0.474 12.946  0.225 |  0.782 43.543
##                 cos2
## X100m         0.034 |
## Long.jump     0.033 |
## Shot.put      0.001 |
## High.jump     0.067 |
## X400m         0.017 |
## X110m.hurdle  0.009 |
## Discus        0.002 |
## Pole.vault    0.479 |
## Javeline      0.152 |
## X1500m        0.612 |
##
## Supplementary continuous variables
##                  Dim.1   cos2    Dim.2   cos2    Dim.3   cos2
## Rank         | -0.671  0.450 |  0.051  0.003 | -0.058  0.003 |
## Points       |  0.956  0.914 | -0.017  0.000 | -0.066  0.004 |
##
## Supplementary categories
##                   Dist    Dim.1   cos2 v.test    Dim.2   cos2 v.test    Dim.3
## Decastar     |  0.946 | -0.600  0.403 -1.430 | -0.038  0.002 -0.123 |  0.289
## OlympicG     |  0.439 |  0.279  0.403  1.430 |  0.017  0.002  0.123 | -0.134
##                cos2 v.test
## Decastar      0.093  1.050 |
```

```
## OlympicG      0.093 -1.050 |
```

We can observe the eigenvalues of the result are in the first column of the result obtained from PCA, and we observe that the percentage is the second variable. We plot this using ggplot2

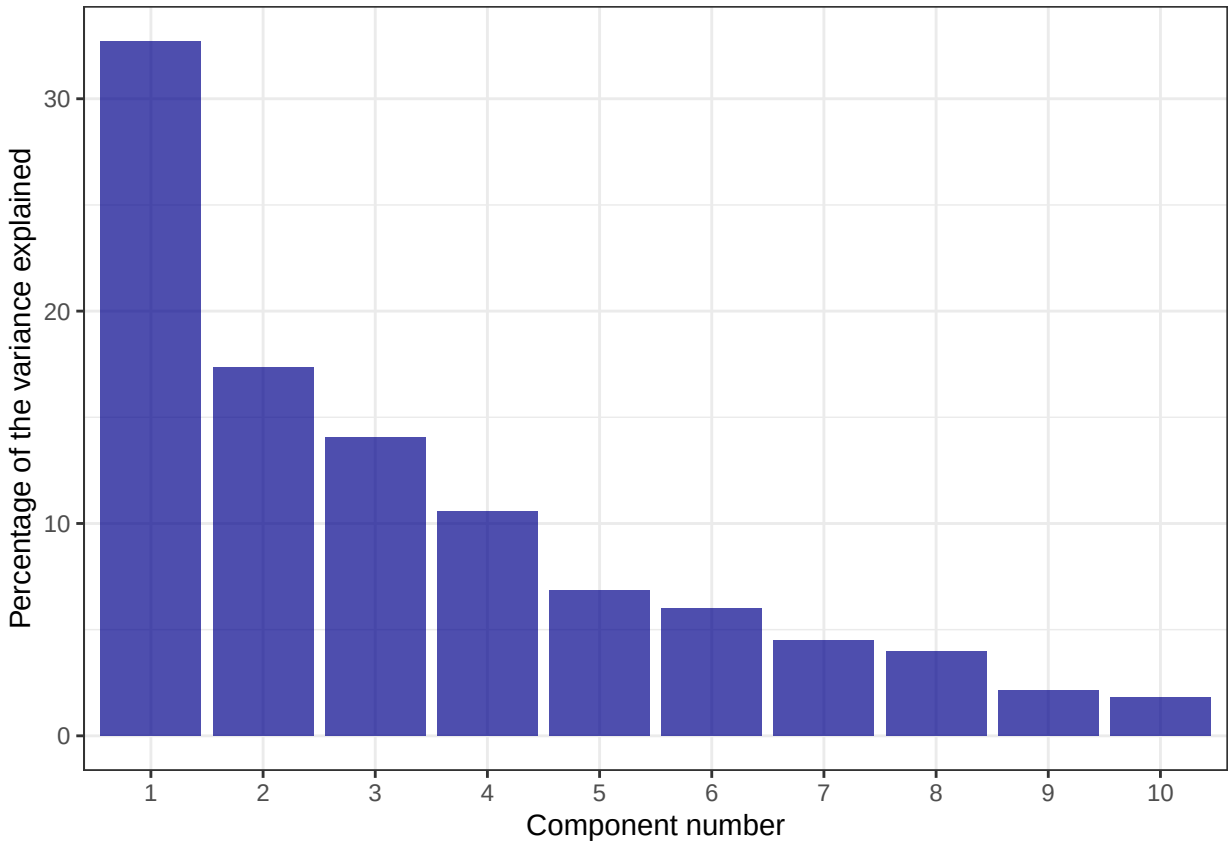
```
res.PCA$eig
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   3.2719055              32.719055                           32.71906
## comp 2   1.7371310              17.371310                           50.09037
## comp 3   1.4049167              14.049167                           64.13953
## comp 4   1.0568504              10.568504                           74.70804
## comp 5   0.6847735               6.847735                           81.55577
## comp 6   0.5992687               5.992687                           87.54846
## comp 7   0.4512353               4.512353                           92.06081
## comp 8   0.3968766               3.968766                           96.02958
## comp 9   0.2148149               2.148149                           98.17773
## comp 10  0.1822275               1.822275                          100.00000
```

```
library(tibble) # Allows to have small data frame
```

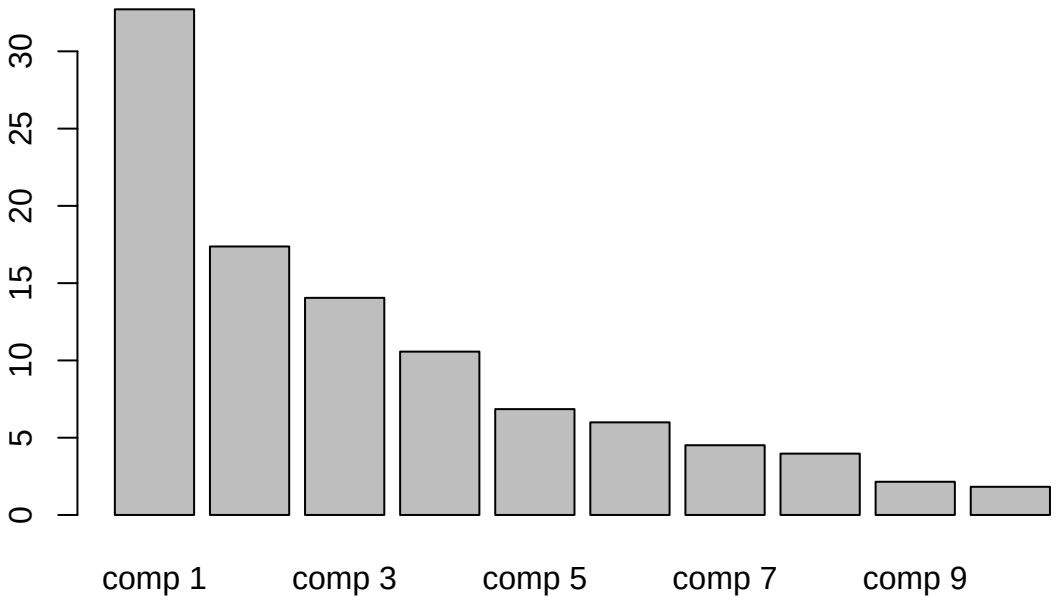
```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
plot_percentage <- tibble("comp" = as.factor(seq(1:length(res.PCA$eig[,2]))),
               "percentage" = res.PCA$eig[,2])

ggplot(plot_percentage, aes(x = comp, y = percentage)) +
  geom_bar(stat="identity", fill = "darkblue", alpha = 0.7) +
  xlab("Component number") + # don't forget to explicit your axis
  ylab("Percentage of the variance explained") +
  theme_bw()
```



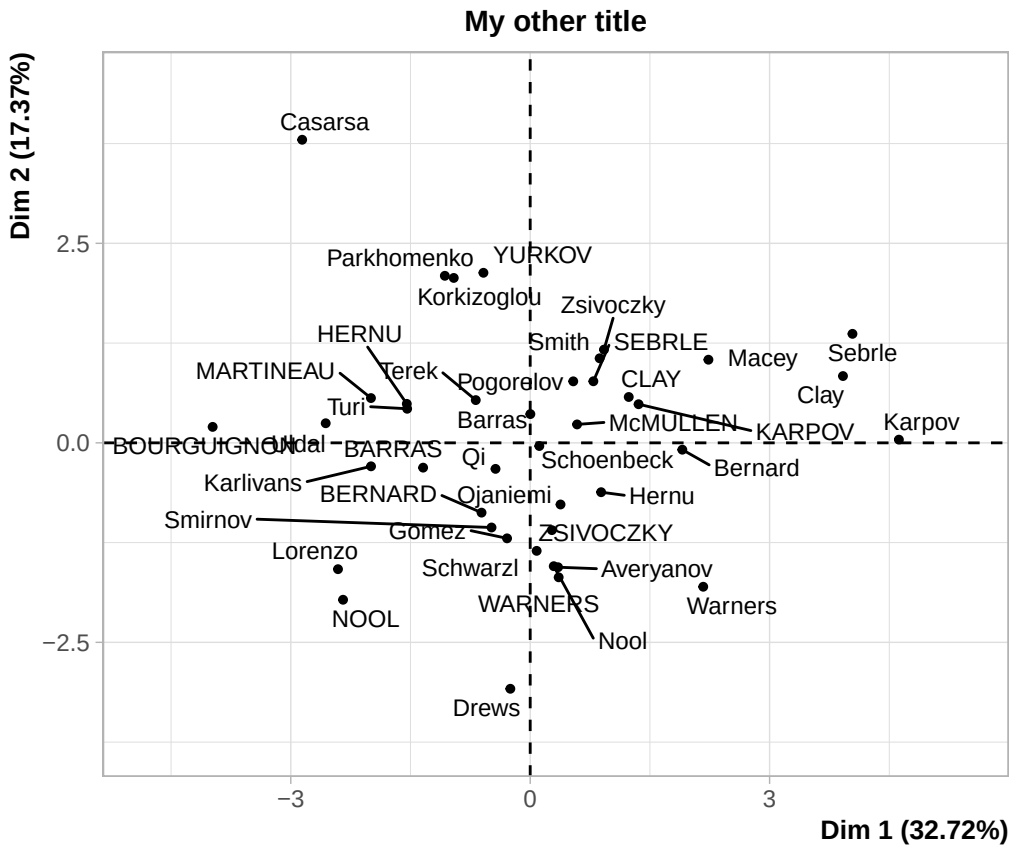
Note that you can go faster without ggplot, the drawback is that you have less liberty for further analysis or for customizing your plot

```
barplot(res.PCA$eig[,2])
```



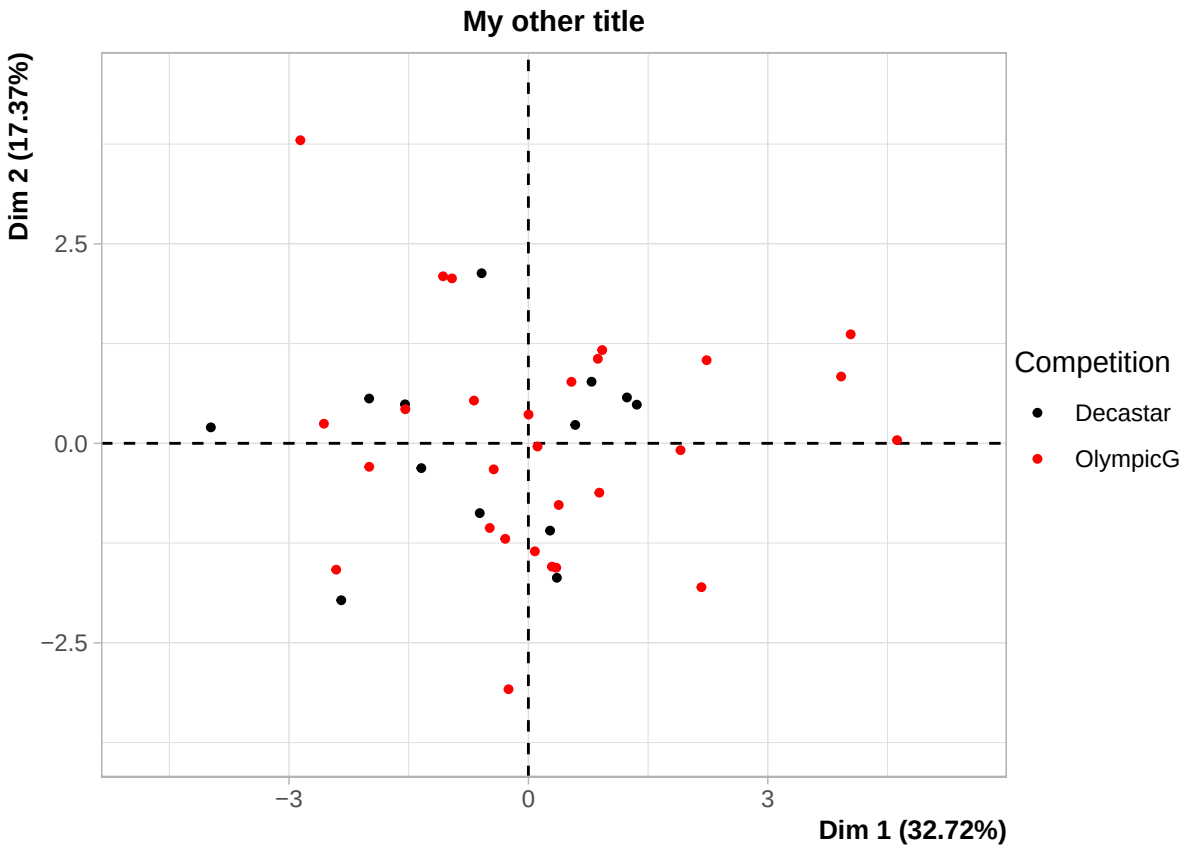
Note that you may want to plot your graph in a different way. For example with smaller font, and without the qualitative variables, and another title.

```
plot(res.PCA, cex=0.8, invisible="quali", title="My other title")
```



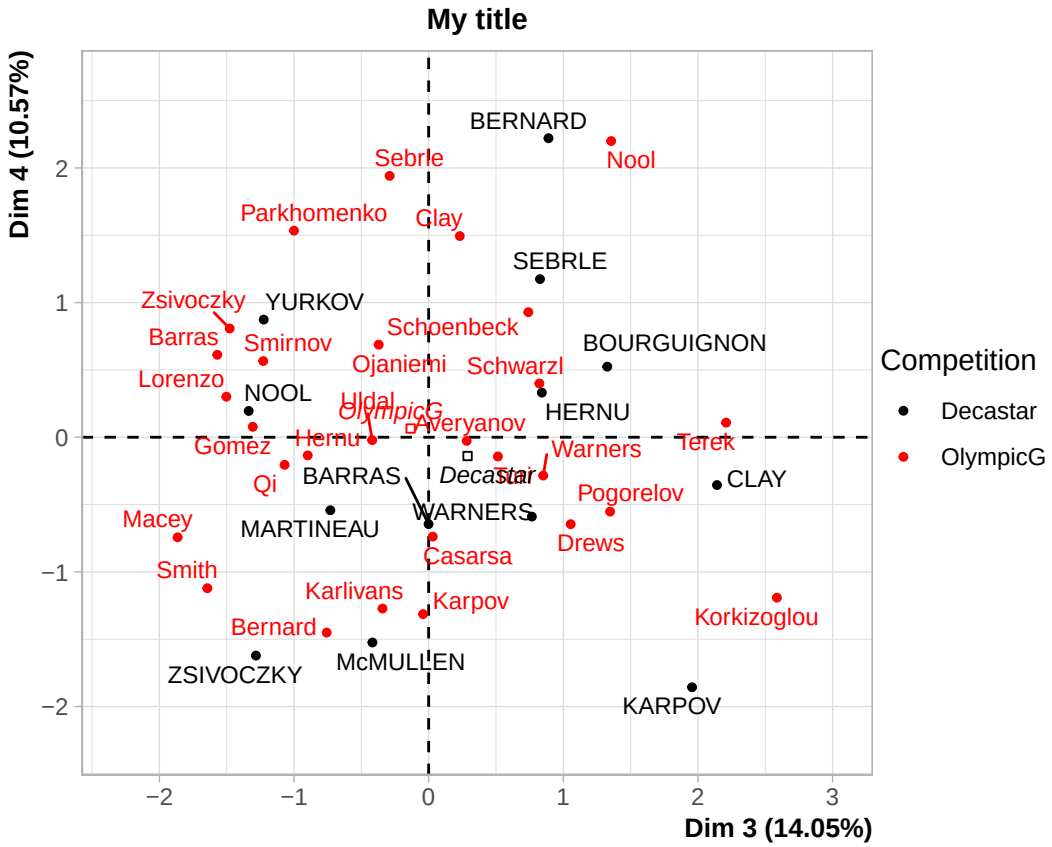
You can also put no label, put the color of a qualitative variable (here you have to call it habillage). Note that you can also write something like: `plot(res, cex=0.8, habillage=13)`.

```
plot(res.PCA, cex=0.8, invisible="quali", label = "none", title="My other title", habillage="Competitio
```



You can also represent your individue on other axes:

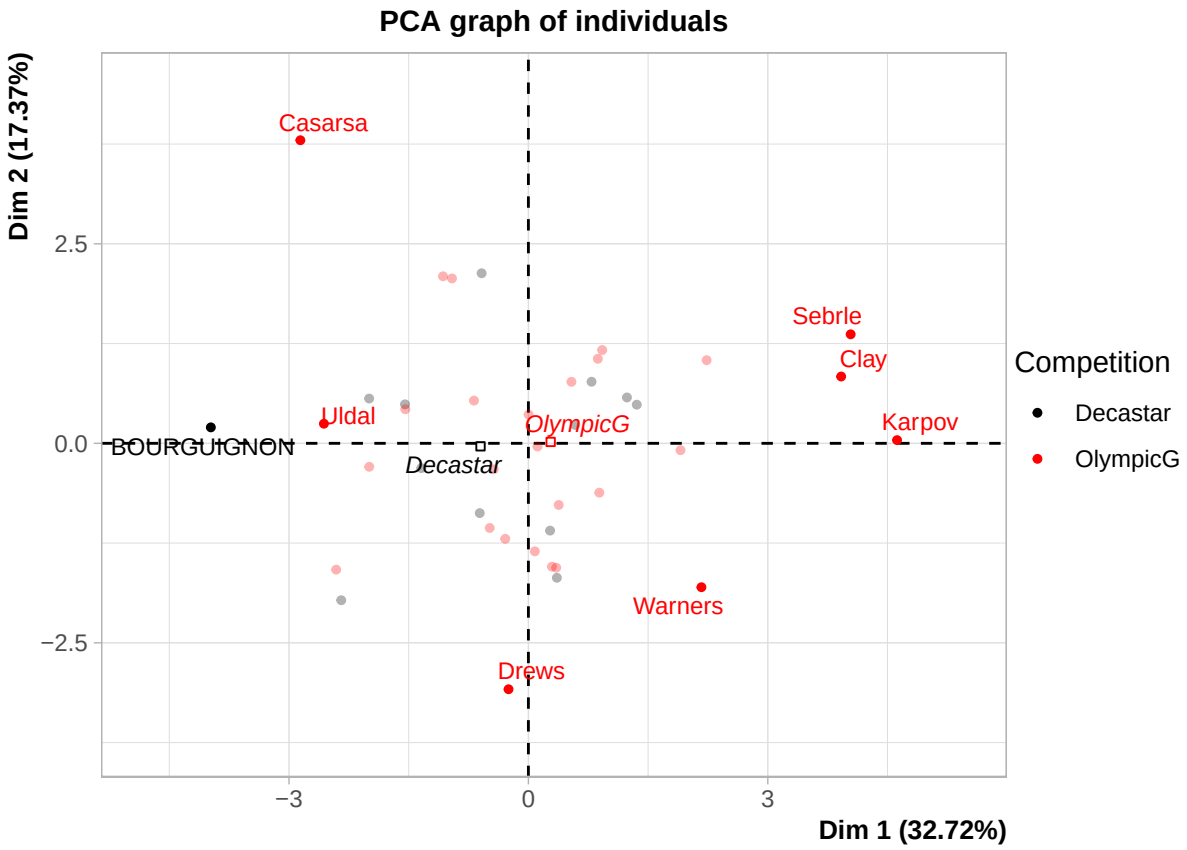
```
plot(res.PCA, choix="ind", cex=0.8, habillage=13, title="My title", axes=3:4)
```



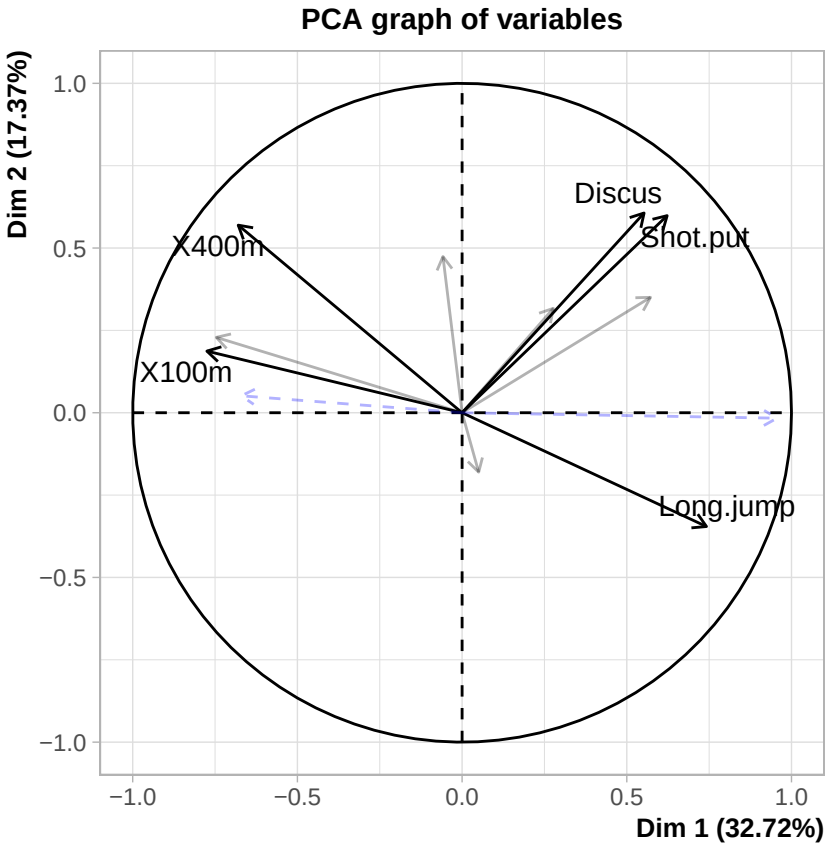
And you can also select individuals.

`select="cos2 0.7"` : select the individuals that have a quality of representation on the map greater than 0.7 `select="cos2 5"` : select the 5 individuals that have the best quality of representation on the map `select="contrib 5"` : select the 5 individuals that contribute the most to the construction of the map `select=c("nom1","nom2")` : select the individuals by their name

```
plot(res.PCA, cex=0.8, habillage=13, select="cos2 0.7")
```




```
plot(res.PCA, choix="var", select="contrib 5")
```



End of solution question 2

Question 3

3. What can you say on the variables related to speed (100m and 400m) versus the long jump?

Tips:

- You can first give a general comment looking at the correlation circle
- You can also access to the details of this graph looking at what hides in the variable results `res.PCA`*var*`else`.

Solution question 3

Variables related to speed are negatively correlated with the first principal component, while the variables shot put and long jump are positively correlated with this component.

You can be surprised to see that long jump is negatively correlated with X100m. Does this mean that people that are fast on the 100m are bad at jumping? This is the reverse! A small value for running (X.100m for example) corresponds to a high score!

We can also observe that the variables High.jump. Shot.put and Discus are not well correlated with the variables related to speed and long jump. Apparently strength and speed are two different things.

```
res.PCA$var$coord[,1:2] # first two dimension, with the precise coordinate
```

```
##                   Dim.1      Dim.2
## X100m        -0.77471983  0.1871420
## Long.jump     0.74189974 -0.3454213
## Shot.put      0.62250255  0.5983033
```

```
## High.jump     0.57194530  0.3502936
## X400m        -0.67960994  0.5694378
## X110m.hurdle -0.74624532  0.2287933
## Discus        0.55246652  0.6063134
## Pole.vault    0.05034151 -0.1803569
## Javeline      0.27711085  0.3169891
## X1500m       -0.05807706  0.4742238
```

```
res.PCA$var$cos2[,1:2] # gives the representation quality of the coordinate
```

```
##                    Dim.1      Dim.2
## X100m        0.600190812 0.03502213
## Long.jump    0.550415232 0.11931587
## Shot.put     0.387509426 0.35796686
## High.jump    0.327121422 0.12270561
## X400m        0.461869674 0.32425938
## X110m.hurdle 0.556882084 0.05234639
## Discus       0.305219255 0.36761593
## Pole.vault   0.002534268 0.03252860
## Javeline     0.076790421 0.10048206
## X1500m       0.003372945 0.22488818
```

```
res.PCA$var$contrib[,1:2] # gives the contribution to the construction of the components
```

```
##                    Dim.1     Dim.2
## X100m        18.34376957  2.016090
## Long.jump    16.82246707  6.868559
## Shot.put     11.84353954 20.606785
## High.jump     9.99788710  7.063694
## X400m        14.11622887 18.666374
## X110m.hurdle 17.02011495  3.013382
## Discus        9.32848615 21.162245
## Pole.vault    0.07745541  1.872547
## Javeline      2.34696326  5.784369
## X1500m        0.10308808 12.945954
```

End of solution question 3

Question 4

4. What can you say on Carsara athlete, Sebrle and Clay, and also Schoenbeck and Barras?

Solution question 4

First, Carsara. Casarsa is located on the top left corner. The first dimension is highly correlated with the number of points: this indicates that he does not have a large number of points. The second dimension is correlated with the Shot.put, High.jump and Discus. This indicates that Casarsa had good results in these three sports. Remember that the second dimension is calculated orthogonally to the first. So Casarsa has good results in these three sports compared to other "bad" athletes.

Sebrle and Clay are close to one another and both far from the center of gravity of the cloud of points. The quality of their projection is therefore good, and we can be certain that they are indeed close in the original space. This means that they have similar profiles in their results across all sports events.

Schoenbeck and Barras are close to one another but they are also close to the center of gravity of the cloud of points. When looking at their cos2 they are not well projected, We cannot interpret their distance based on this plot only.

```
res.PCA$ind$cos2[c("SEBRLE", "CLAY", "Schoenbeck", "Barras"),1:2]
```

```
##                   Dim.1        Dim.2
## SEBRLE     1.116789e-01 0.1061026225
## CLAY       1.240094e-01 0.0268426547
## Schoenbeck 4.055023e-03 0.0004917858
## Barras     9.304287e-07 0.0262522035
```

End of solution question 4

Question 5

5. Which variable predict the best the final score?

Solution question 5 *The supplementary variable "number of points" is almost collinear to the first principal component. Therefore, the athletes with a high number of points are particularly good in the trials correlated with the first principal component. The most correlated variables with this component are 100m, X110m.hurdle and long jump. You can see this on the correlation circle, but you can also look at the contributions.*

```
res.PCA$var$contrib[,1:2]
```

```
##                    Dim.1     Dim.2
## X100m         18.34376957  2.016090
## Long.jump     16.82246707  6.868559
## Shot.put      11.84353954 20.606785
## High.jump      9.99788710  7.063694
## X400m         14.11622887 18.666374
## X110m.hurdle  17.02011495  3.013382
## Discus         9.32848615 21.162245
## Pole.vault     0.07745541  1.872547
## Javeline       2.34696326  5.784369
## X1500m         0.10308808 12.945954
```

Don't forget that the quantity in X100m is in second, so the higher the value, the lower the performance. Therefore it is "normal" to see a negative correlation.

A general conclusion you can make is the fact that these three sports govern the decathlon final score.

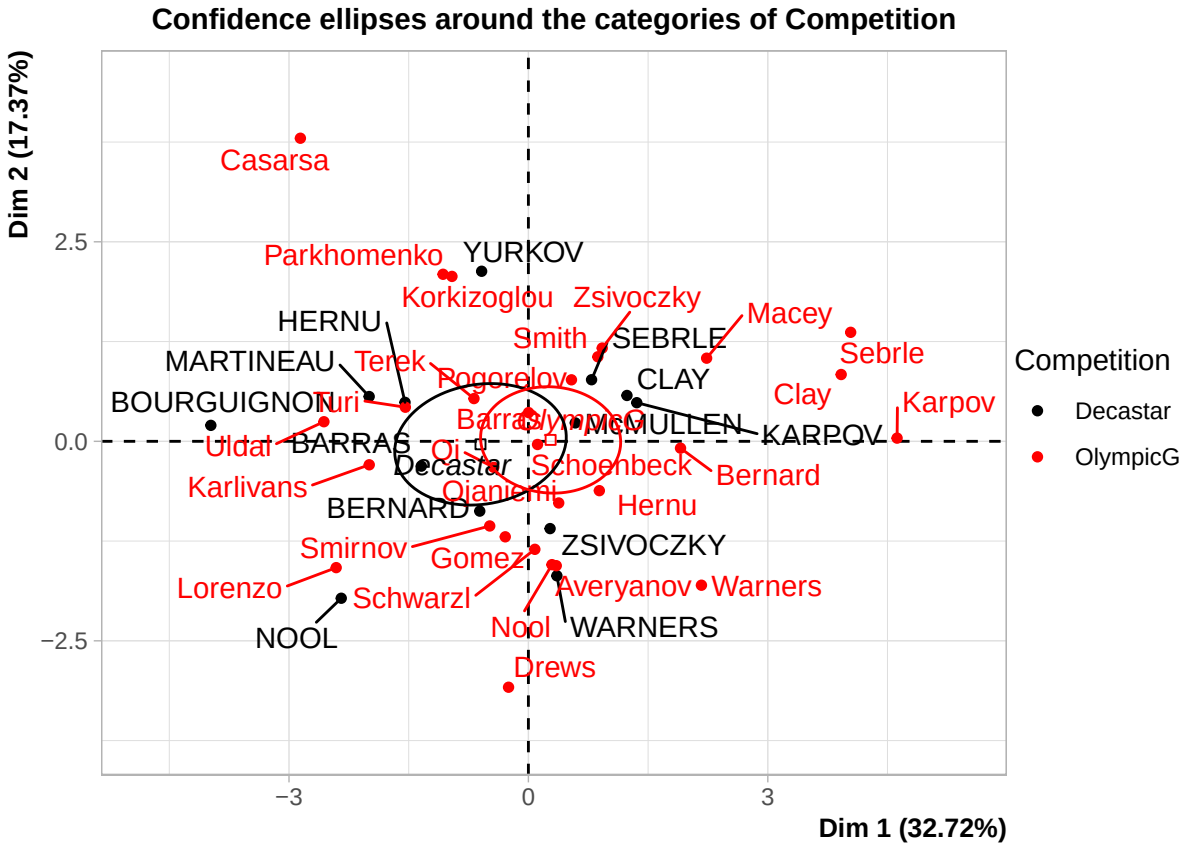
End of solution question 5

Question 6

Try the command `plotellipses()` on the PCA results. What can you say?

Solution question 6

```
plotellipses(res.PCA)
```



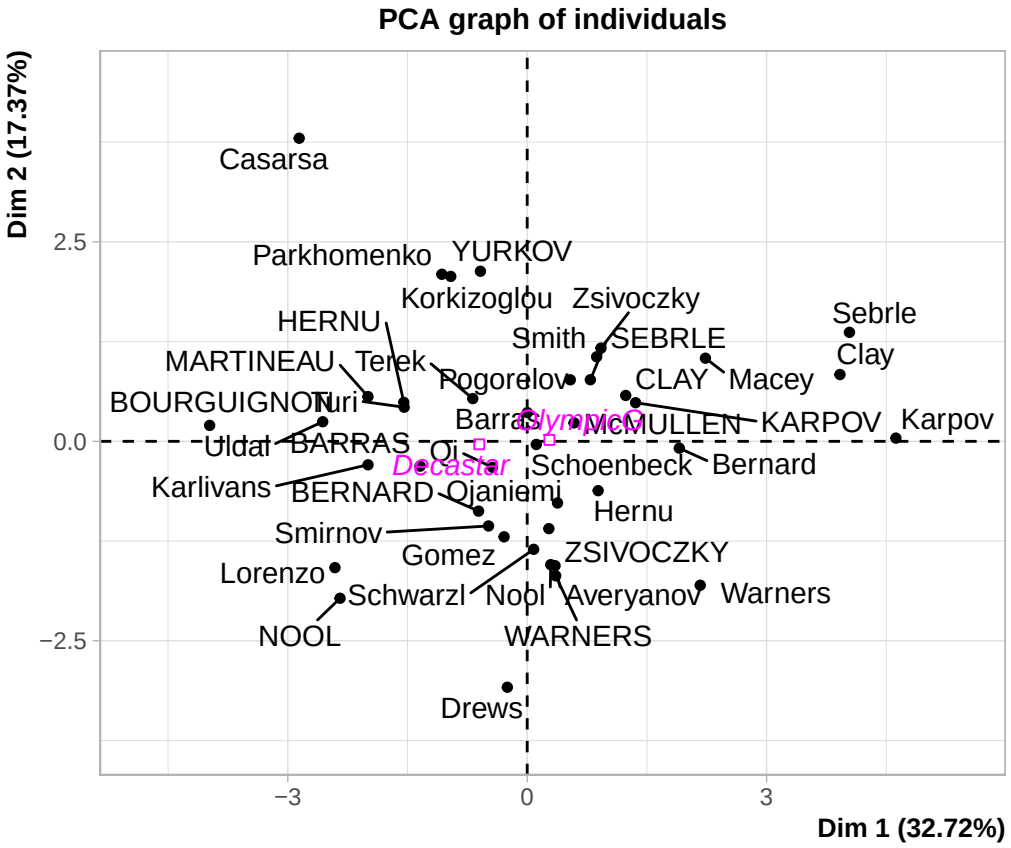
If several qualitative variables are available, there will be as many graphs as qualitative variables. And on each graph the confidence ellipses around the categories of a categorical variable.

We observe that the barycenters of the two competitions (Decastar and Olympic) are different, but that this is not significant.

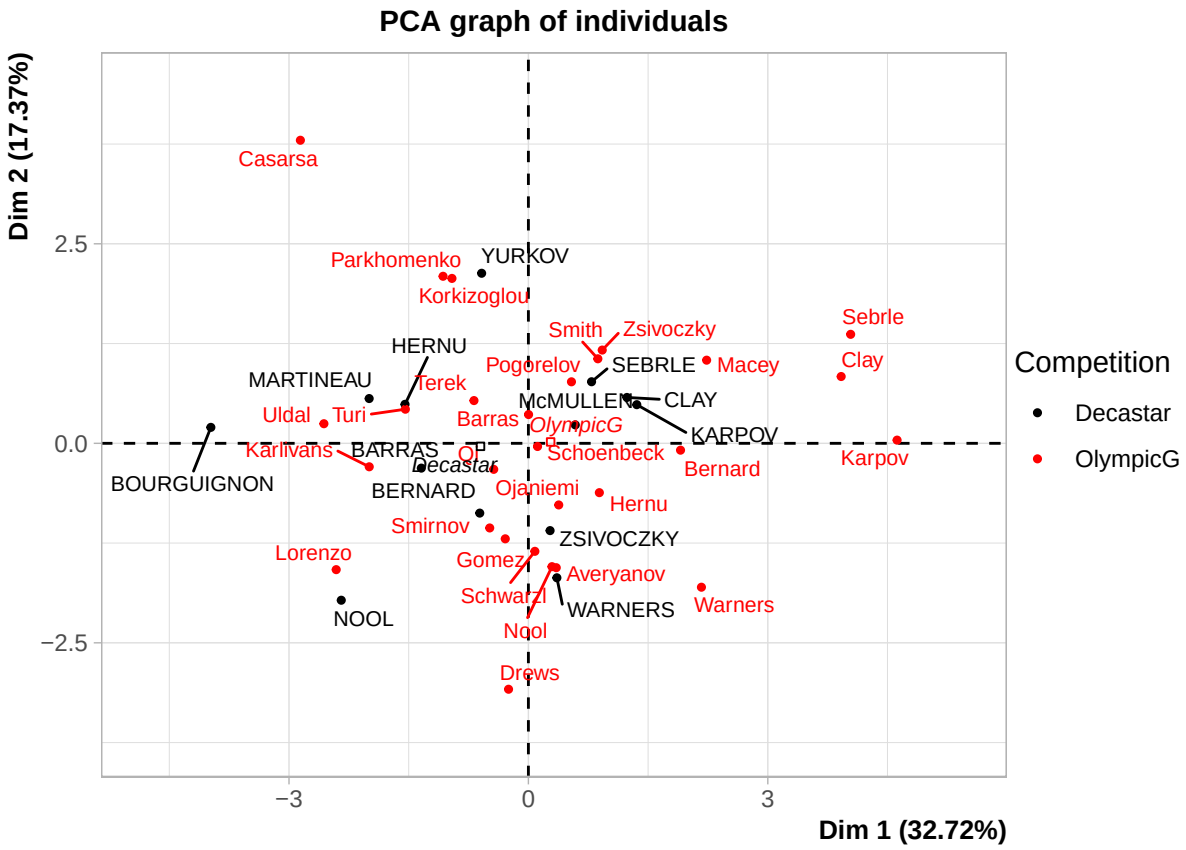
End of solution question 6

Bonus: in report you may want to do beautiful plots! Here are some commands you can use.

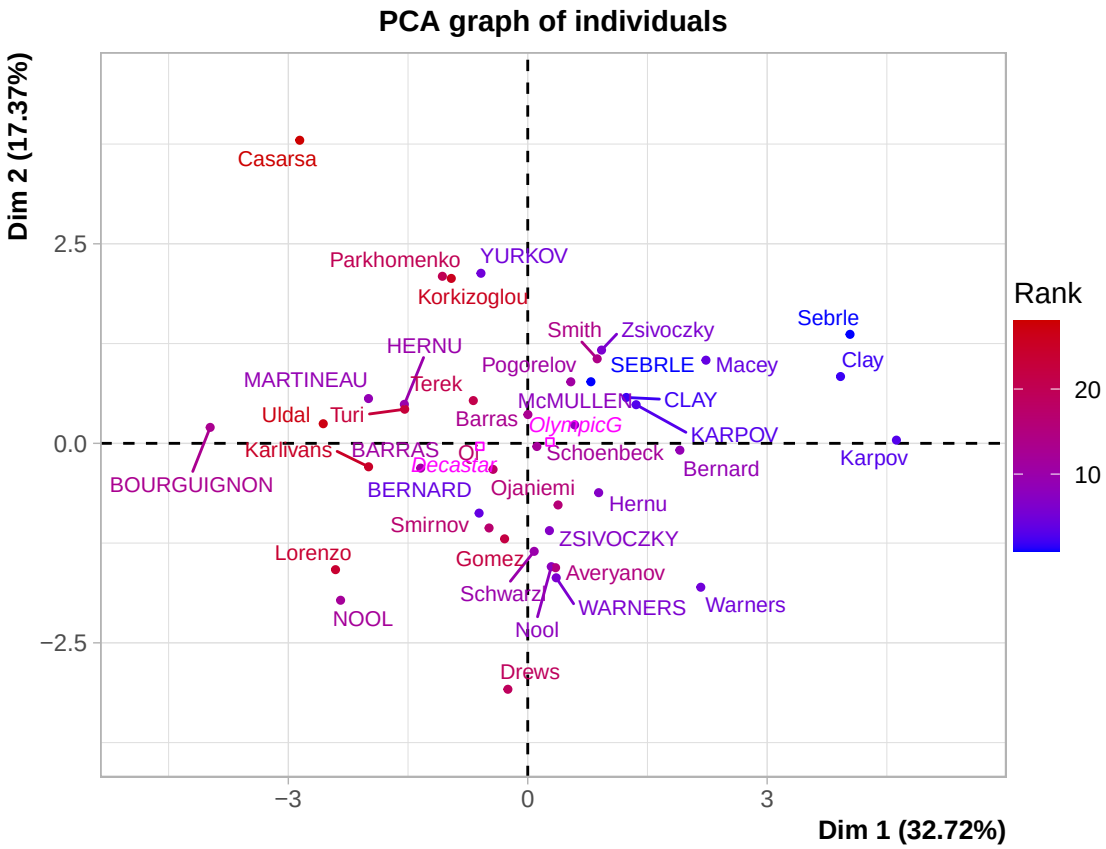
```
plot.PCA(res.PCA, choix = "ind") # choix: meaning you only want to represent the "ind" plot, you can al
```



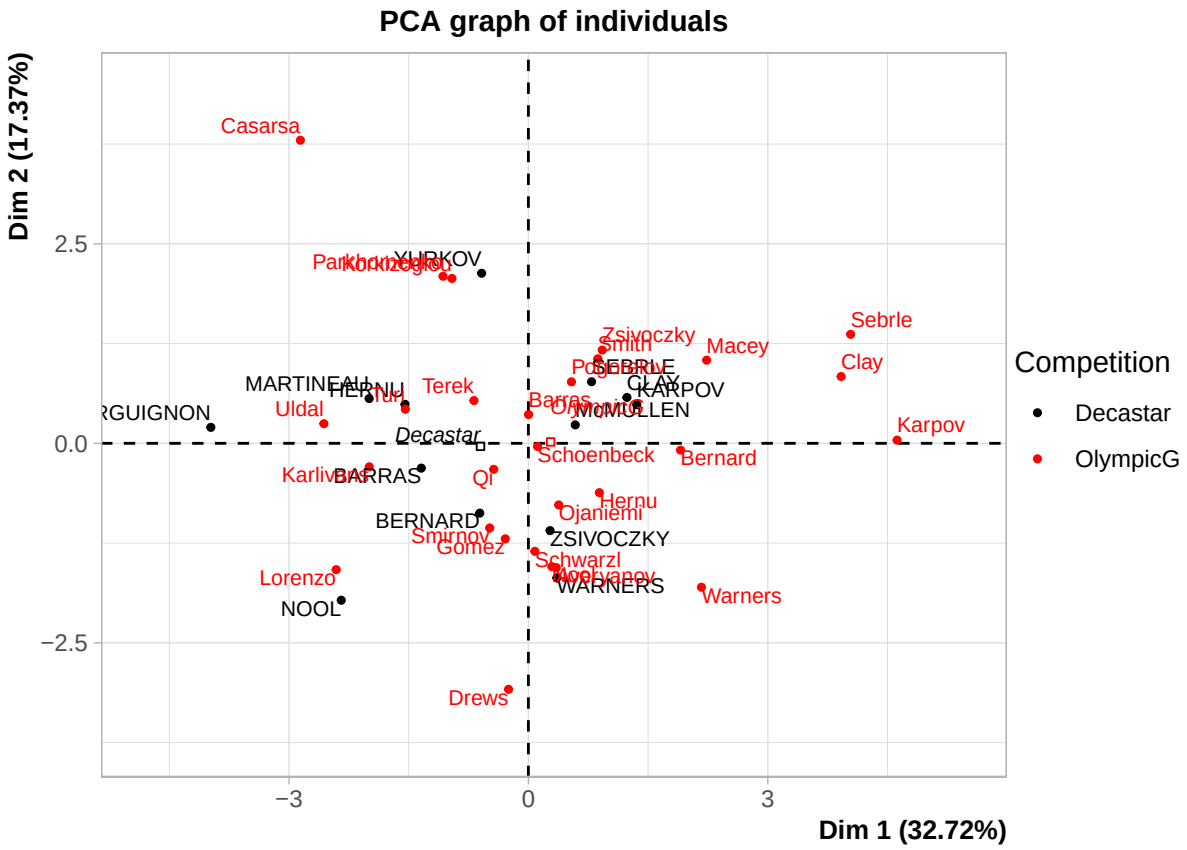

```
plot.PCA(res.PCA, choix = "ind", habillage = "Competition",  cex = 0.7) # choose the variables that wil
```



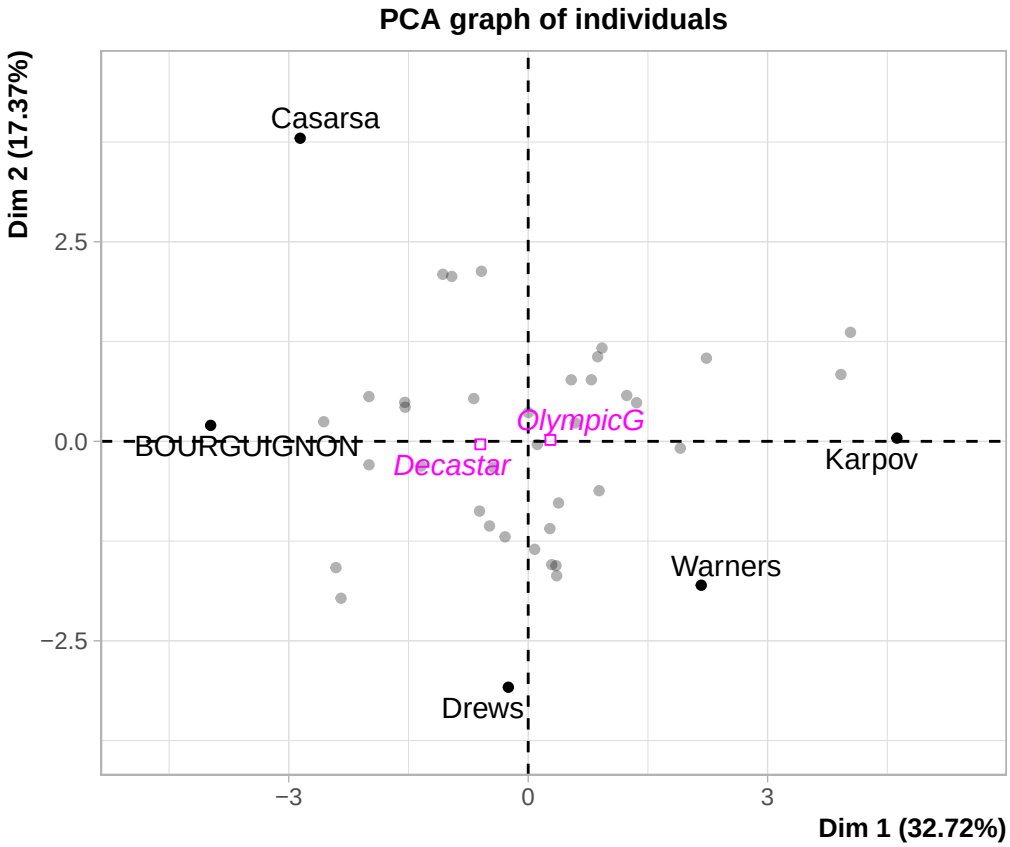

```
plot.PCA(res.PCA, choix = "ind",habillage = "Rank",  cex = 0.7) # choose the variables that will do the
```



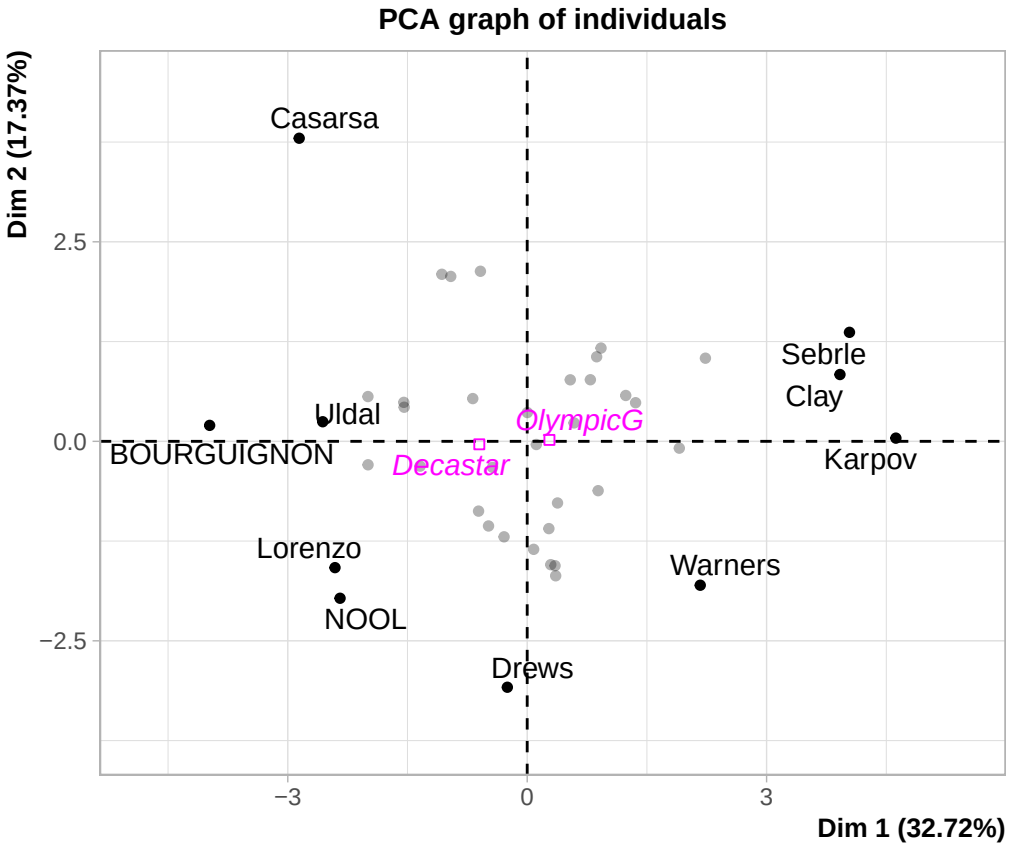

```
plot.PCA(res.PCA, choix = "ind", habillage = ncol(decathlon), cex = 0.7, autoLab = "no") # Different la
```



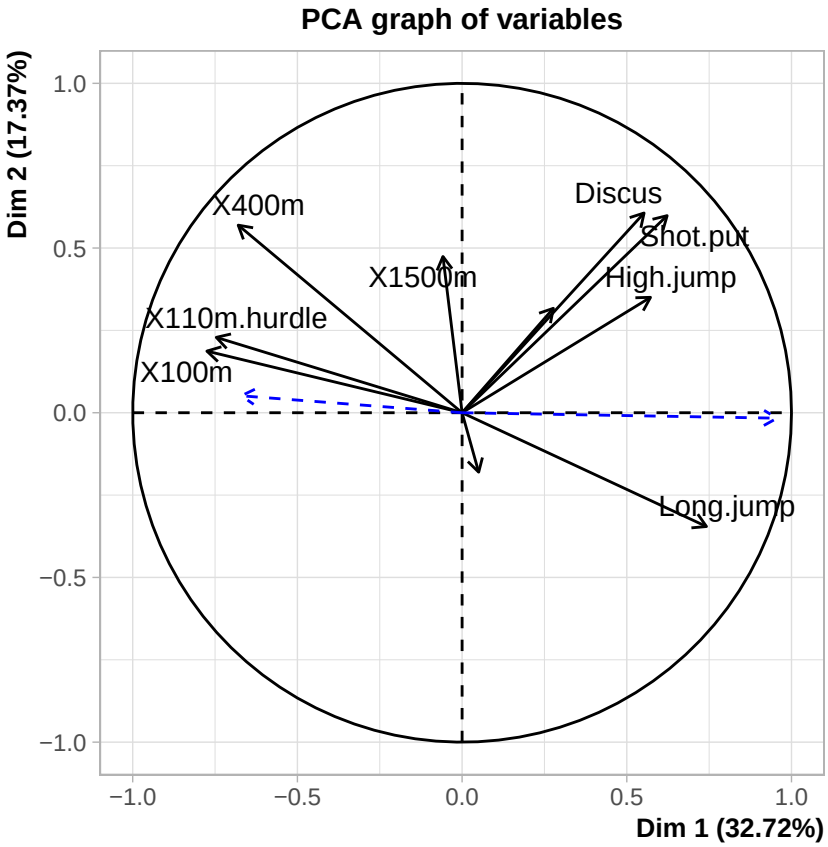

```
plot.PCA(res.PCA, select = "cos2 0.8") # put a threshold on the ind that have a high cos2
```



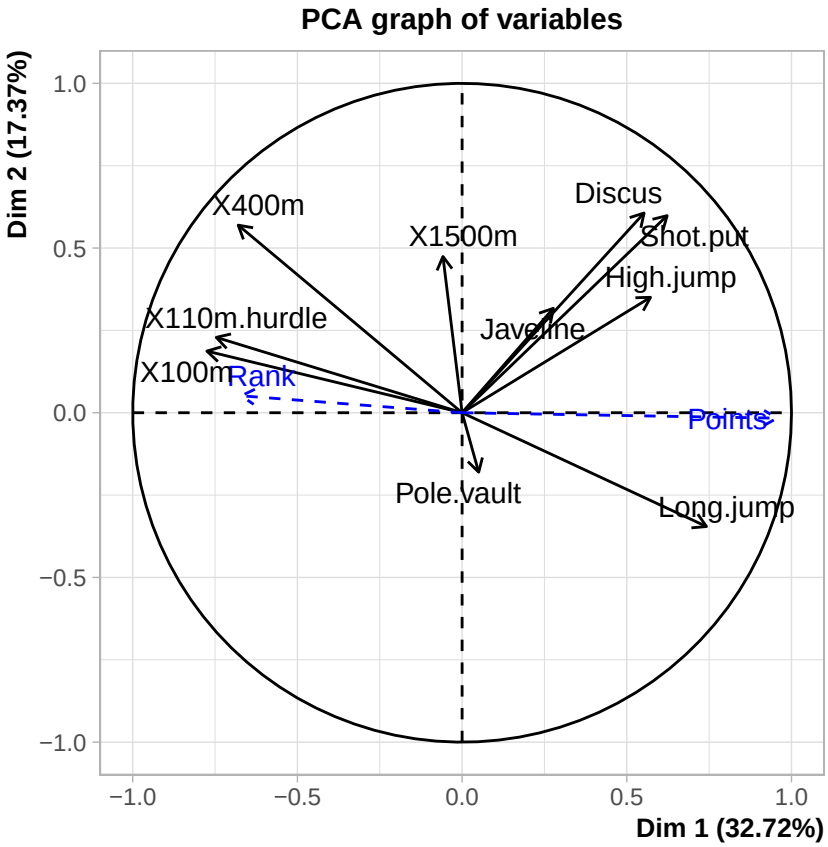

```
plot.PCA(res.PCA, select = "contrib 10")
```




```
plot.PCA(res.PCA, choix = "var", select = "contrib 8", unselect = 0)
```




```
plot.PCA(res.PCA, choix = "var", select = c("400m", "1500m"))
```



Note that you can also use the command `Factoinvestigate()` that does the

```
library(FactoInvestigate)
```

```
## Warning: package 'FactoInvestigate' was built under R version 3.6.2
```

```
Investigate(res.PCA)
```

```
## -- creation of the .Rmd file (time spent : 0s) --
##
## -- detection of outliers (time spent : 0s) --
## 0 outlier(s) terminated
##
## -- analysis of the inertia (time spent : 0.01s) --
## 3 component(s) carrying information : total inertia of 64.1%
##
## -- components description (time spent : 2.98s) --
## plane 1:2
## dim. 3
##
## -- classification (time spent : 3.18s) --
## 4 clusters
##
## -- annexes writing (time spent : 3.25s) --
##
## -- saving data (time spent : 3.32s) --
##
## -- outputs compilation (time spent : 3.32s) --
```

```
## -- task completed (time spent : 9.93s) --
## This interpretation of the results was carried out automatically,
## it cannot match the quality of a personal interpretation
```

FactoShiny

`FactoShiny` is a graphical interface to the `FactoMineR` package to plot interactive plots. Therefore the underlying tools are the same as we saw previously. But this graphical interface can help you while working on data, and also to present in a funny way your data to a team. In this part we keep the same decathlon data.

To test it on your own, you can load the `Factoshiny` library and use the command `PCAshiny`.

Tip:

- If you use Mac you don't have a working Tcl/Tk by default, while it is needed for this package. So don't worry if you see an error while installing it! Go in the console and type `brew install tcl-tk` (if you use brew, what we recommend for Mac). The error can be quite complex as explained here[1].

Clustering

Hierarchical Cluster Analysis (HCA)

or Classification Ascendante Hiérarchique (CAH) in French!

Question 1

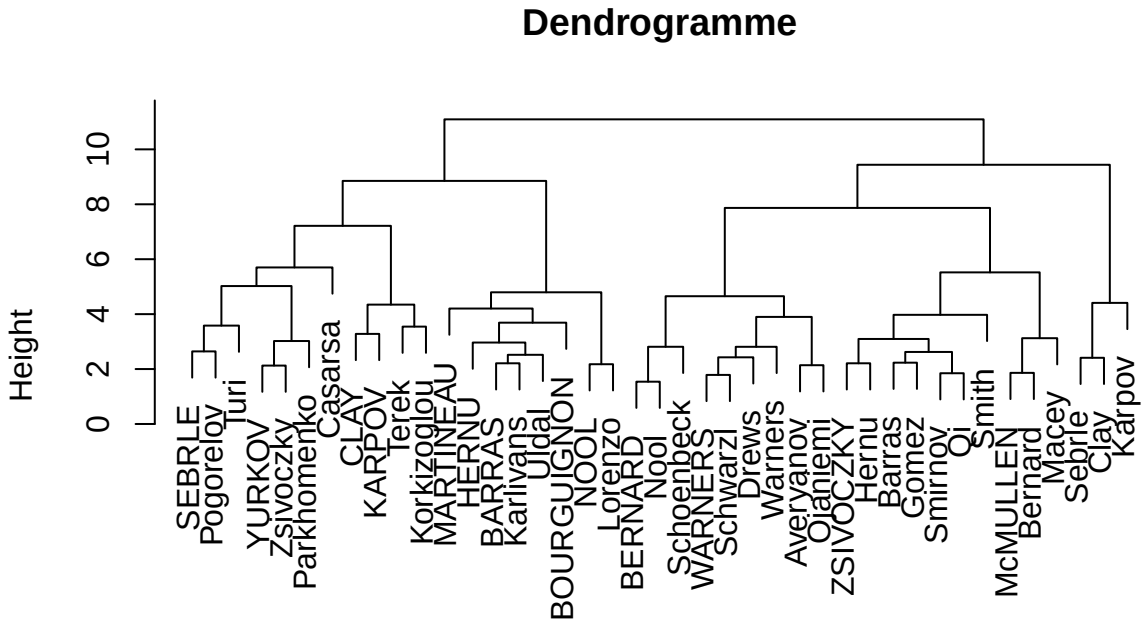
1. Explain the general principle. Do you need to scale your data or not? (explain why) Launch a HCA on the decathlon data using the package `cluster` and the function `agnes()`. You can visualize your results using the function `plot()`. Be careful to use the ward distance.

Solution question 1 Cf. class 2. It is very important to normalize because units are different in this data set. And the clustering is grounded on comparing distance between units. It is the same reasoning as the PCA question. Remember that you can use the command `scale()` to scale your data in R.

```
library(cluster)
classification <- agnes(scale(decathlon[,1:10]), method = "ward")
```

```
plot(classification ,xlab="Individu", which.plot=2, main="Dendrogramme")
```

[1] https://swvanderlaan.github.io/post/getting-r-with-tcl-tk-on-my-mac/



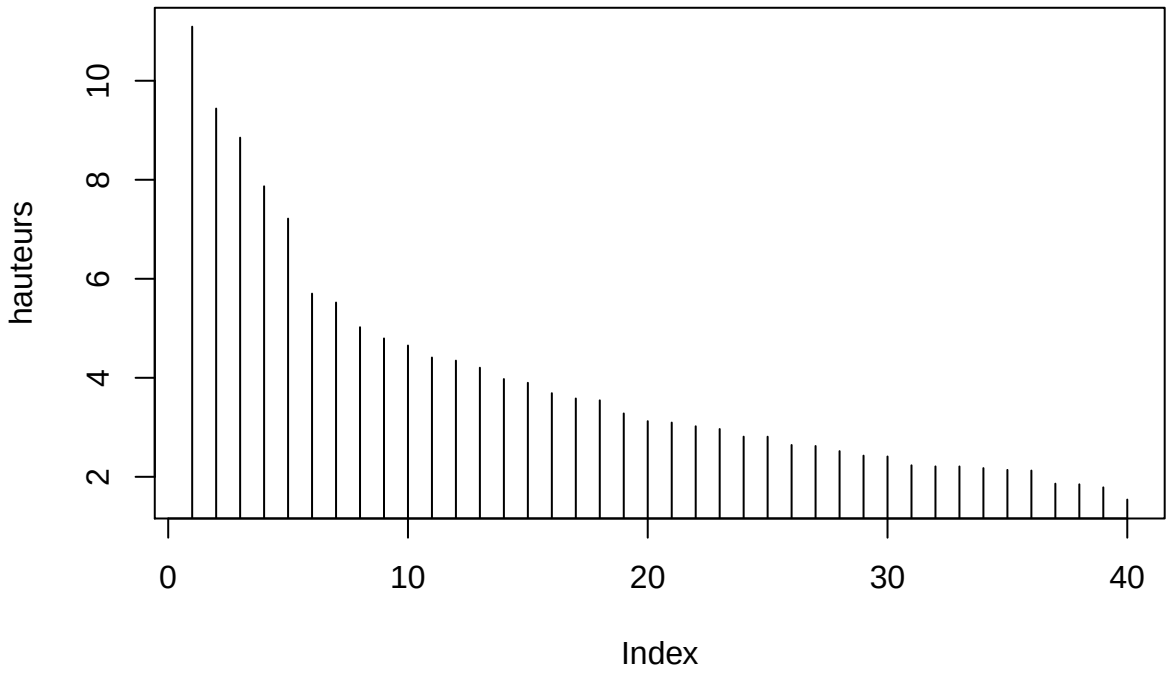
End of solution question 1

Question 2

As seen in class, a challenge is to cut the tree (corresponds to choosing the number of classes). Using the result you had and the function `as.hclust` you can observe the height where two classes are merged. How can you use this result? Once you know where you want to cut your tree, you can use the function `cutree` with the number of classes you want to keep.

Solution question 2

```
height <- as.hclust(classification)
plot(rev(height$height), type = "h", ylab = "hauteurs")
```



The most obvious gap is between 5 to 6 classes. So a good choix could be to choose this one. You can also choose the gap between 3 to 4 because it is still important, and will give use still an important number of individus in each class.

```
classe <- cutree(classification, k = 4)
```

End of solution question 2

Question 3

Once you decided a certain number of class, how can you describe the class? For example which variables characterize the classes you have?

Tips: you can use the function `catdes()`

You can also use the paragons of the classes to describe the classes.

Solution question 3

We first try to understand which variables are linked with classes.

First, you need to bind your classification result to the initial data frame:

```
# build a new data frame with the labels
decathlon.with.classif <- cbind(decathlon, classe = as.factor(classe))
```

```
catdes(decathlon.with.classif, num.var = 14)
```

```
##
## Link between the cluster variable and the quantitative variables
## ================================================================
##                   Eta2      P-value
## Points       0.7181466 2.834869e-10
## X100m        0.5794940 4.206021e-07
## Shot.put     0.4746713 2.358915e-05
## Long.jump    0.4405023 7.319114e-05
## X110m.hurdle 0.4254154 1.178937e-04
```

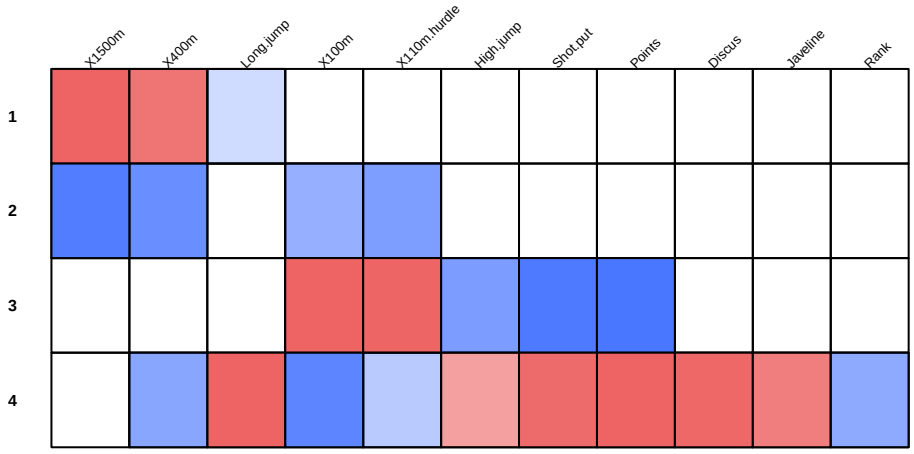
```
## X400m        0.4124012 1.759714e-04
## X1500m       0.4078248 2.021278e-04
## Discus       0.3593367 8.199096e-04
## High.jump    0.2864826 5.454320e-03
## Javeline     0.1999118 3.912122e-02
## Rank         0.1916166 4.655439e-02
##
## Description of each cluster by quantitative variables
## ======================================================
## $`1`
##              v.test Mean in category Overall mean sd in category Overall sd
## X1500m     3.899044       290.763636    279.02488     12.6274652 11.5300118
## X400m      2.753420        50.435455     49.61634      1.2725877  1.1392975
## Long.jump -2.038672         7.093636      7.26000      0.2836218  0.3125193
##                p.value
## X1500m    9.657328e-05
## X400m     5.897622e-03
## Long.jump 4.148279e-02
##
## $`2`
##                 v.test Mean in category Overall mean sd in category Overall sd
## X100m        -2.265855         10.89789     10.99805      0.1701572  0.2597956
## X110m.hurdle -2.397231         14.41579     14.60585      0.3097931  0.4660000
## X400m        -2.579590         49.11632     49.61634      0.5562394  1.1392975
## X1500m       -2.975997        273.18684    279.02488      5.6838942 11.5300118
##                  p.value
## X100m        0.023460250
## X110m.hurdle 0.016519515
## X400m        0.009891780
## X1500m       0.002920378
##
## $`3`
##                 v.test Mean in category Overall mean sd in category
## X100m         4.053791         11.33625    10.998049     0.14194519
## X110m.hurdle  3.368905         15.11000    14.605854     0.32939338
## High.jump    -2.412939          1.90875     1.976829     0.05464373
## Shot.put     -3.134116         13.65750    14.477073     0.60373318
## Points       -3.643979       7609.62500  8005.365854   143.94611622
##                Overall sd      p.value
## X100m          0.25979560 5.039416e-05
## X110m.hurdle   0.46599998 7.546748e-04
## High.jump      0.08785906 1.582447e-02
## Shot.put       0.81431175 1.723728e-03
## Points       338.18394159 2.684548e-04
##
## $`4`
##                 v.test Mean in category Overall mean sd in category
## Points        4.242103       8812.66667  8005.365854    68.78145745
## Long.jump     3.468581          7.87000     7.260000     0.06480741
## Discus        3.107539         50.16000    44.325610     1.19668988
## Shot.put      2.974272         15.84000    14.477073     0.46568945
## Javeline      2.586808         65.25667    58.316585     6.87867397
## High.jump     2.289003          2.09000     1.976829     0.02449490
## X110m.hurdle -2.119695         14.05000    14.605854     0.06531973
```

```
## Rank          -2.299627          2.00000    12.121951     0.81649658
## X400m         -2.333955         48.12000    49.616341     0.98634004
## X100m         -2.745523         10.59667    10.998049     0.18080069
##                 Overall sd      p.value
## Points       338.18394159 2.214348e-05
## Long.jump      0.31251927 5.232144e-04
## Discus         3.33639725 1.886523e-03
## Shot.put       0.81431175 2.936847e-03
## Javeline       4.76759315 9.686955e-03
## High.jump      0.08785906 2.207917e-02
## X110m.hurdle   0.46599998 3.403177e-02
## Rank           7.82178048 2.146935e-02
## X400m          1.13929751 1.959810e-02
## X100m          0.25979560 6.041458e-03
```

This function catdes gives you first the variables that are the most related to the class variable. Here the variable Points is the most interesting one to explain classes.

For example in class 2, individuals are faster on 1500m than the global group of people. 3 other variables characterize well this group (400m, 110m, and 100m). These athletes are also faster in this discipline than others. Remember that a v-test higher than 2 (in absolute value) means that the mean of the class is statistically significantly different from the global mean.

```
plot(catdes(decathlon.with.classif, num.var = 14), show = "all", level = 0.05)
```



After this, it is also possible to use the parangon of each class to interpret the class.

End of solution question 3

Hierarchical Clustering on Principal Components (HCPC)

or Classification Hiérarchique sur Composantes Principales in French!

Open question (on your own)

It is also possible to perform a clustering on the variables obtained after a PCA analysis. For this you can use the `HCPC()` function. On your own, try to do this analysis with plot and quantitative analysis.

Try to have the plot(s) you prefer to present your results. You can find a documentation here: http://www.imsbio.co.jp/RGM/R_rdfile?f=FactoMineR/man/plot.HCPC.Rd&d=R_CC

Solution question

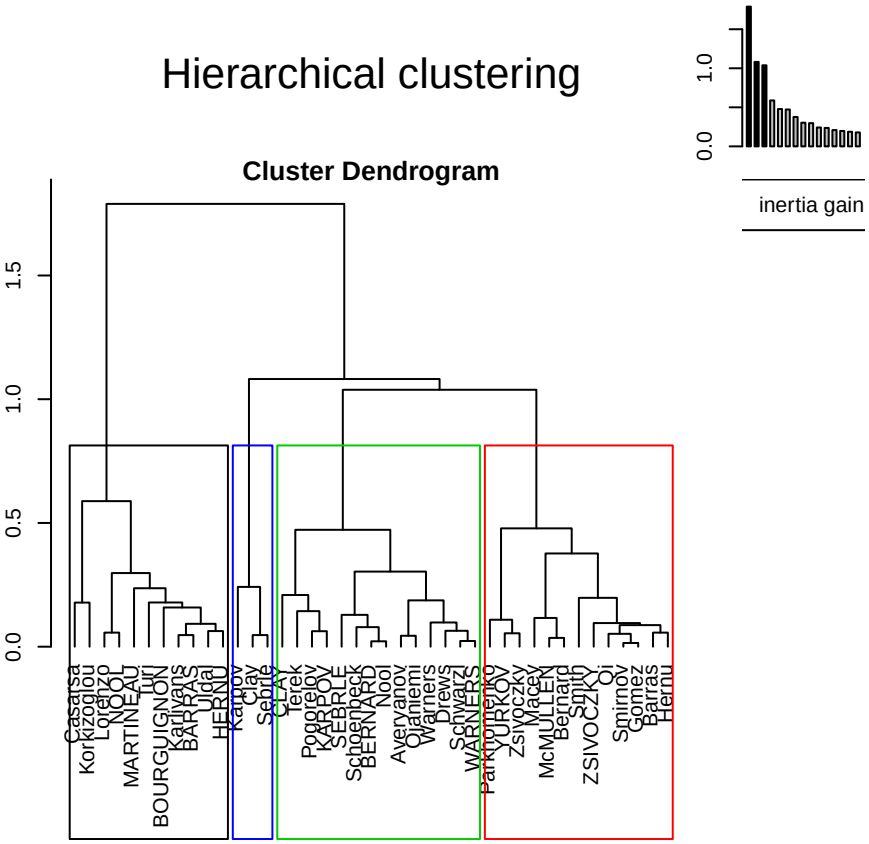
```
# you can use your previous result, or launch again a PCA step.
res.pca <- PCA(decathlon, quanti.sup=11:12, quali.sup=13, graph=F)

res.pca$eig
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   3.2719055              32.719055                           32.71906
## comp 2   1.7371310              17.371310                           50.09037
## comp 3   1.4049167              14.049167                           64.13953
## comp 4   1.0568504              10.568504                           74.70804
## comp 5   0.6847735               6.847735                           81.55577
## comp 6   0.5992687               5.992687                           87.54846
## comp 7   0.4512353               4.512353                           92.06081
## comp 8   0.3968766               3.968766                           96.02958
## comp 9   0.2148149               2.148149                           98.17773
## comp 10  0.1822275               1.822275                          100.00000
```

From that you observe that you need to have the 8 first components to have 95% of the variance explained.

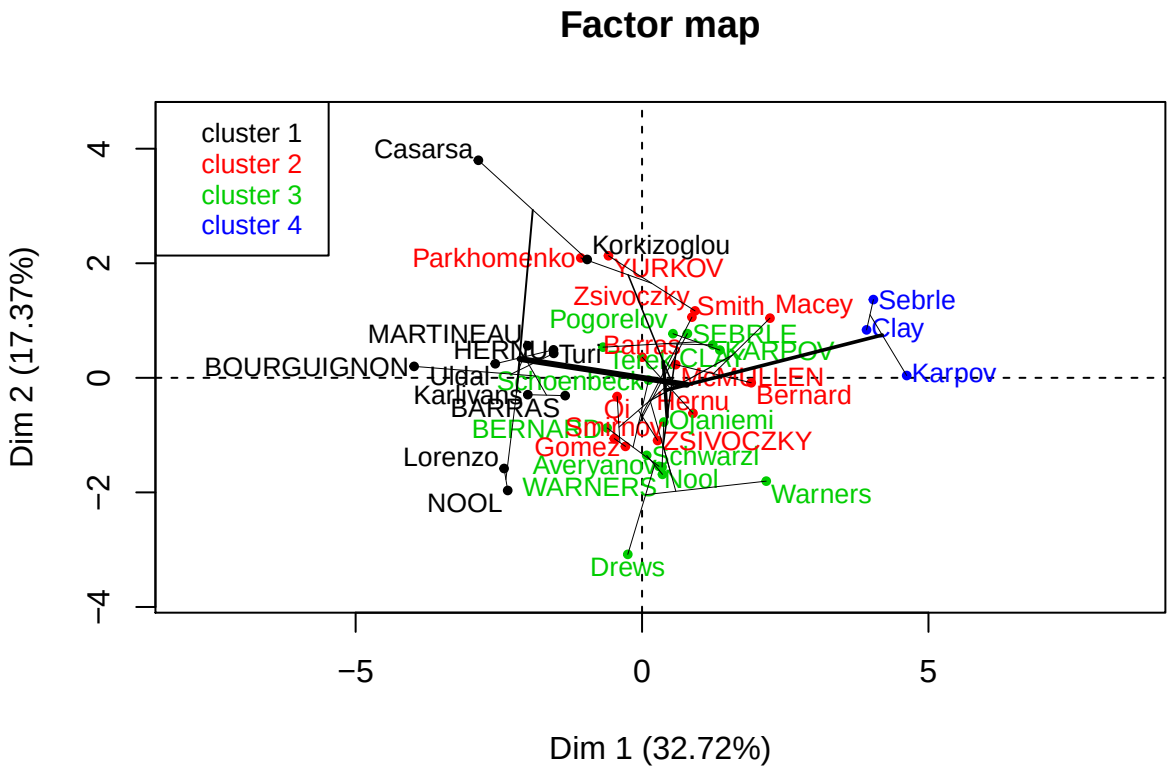
```
# be careful  to the new option
res.pca <- PCA(decathlon, quanti.sup = 11:12, ncp = 8, quali.sup = 13, graph=F)

res.hcpc <- HCPC(res.pca, consol = FALSE,  graph=FALSE) # if consol is TRUE, it means that you allow a
```

```
plot(res.hcpc, choice = "tree") ## tree
```



You can observe that a cut is proposed by the tool when launching with `graph = TRUE`. It is tree. You can also see it when plotting the tree.

```
plot(res.hcpc, choice = "map") ## map
```



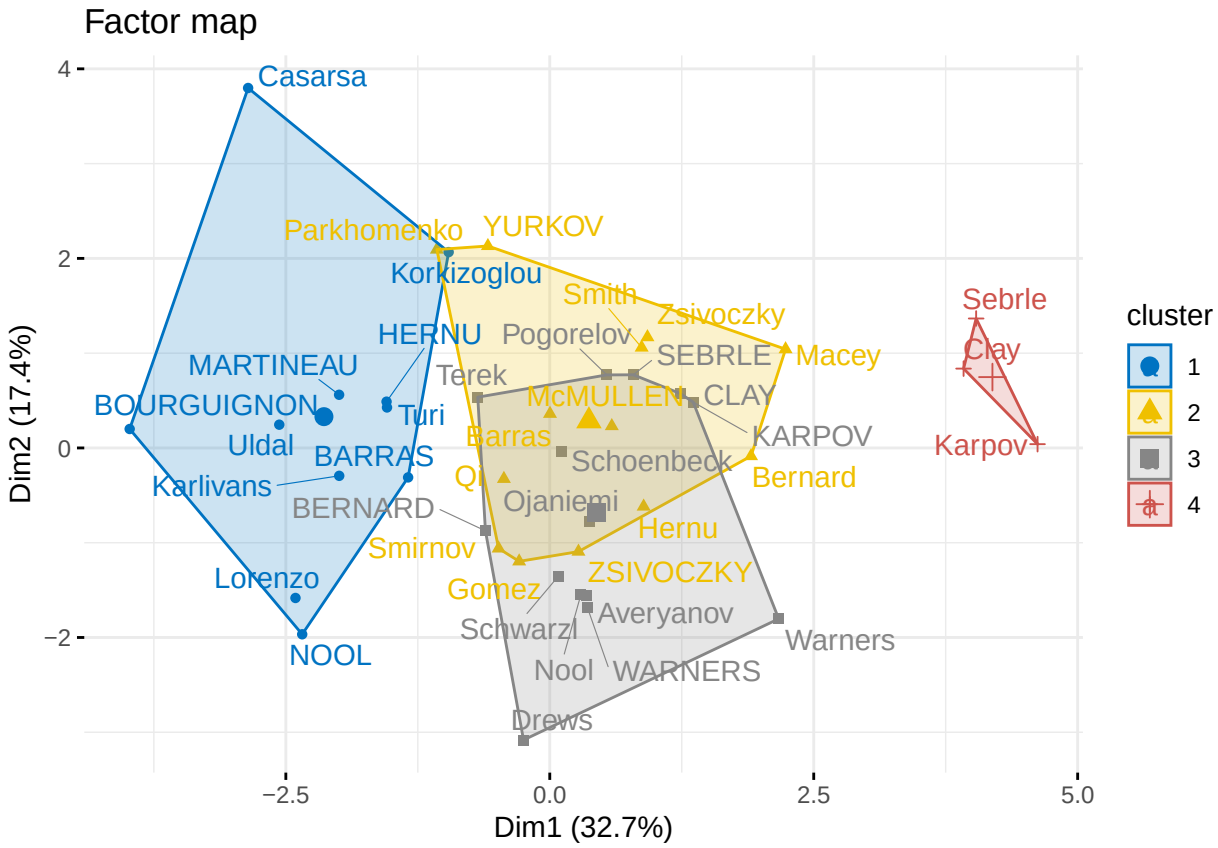
Note: another library exist to plot your result, this library is called `factoextra`.

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.6.2
```

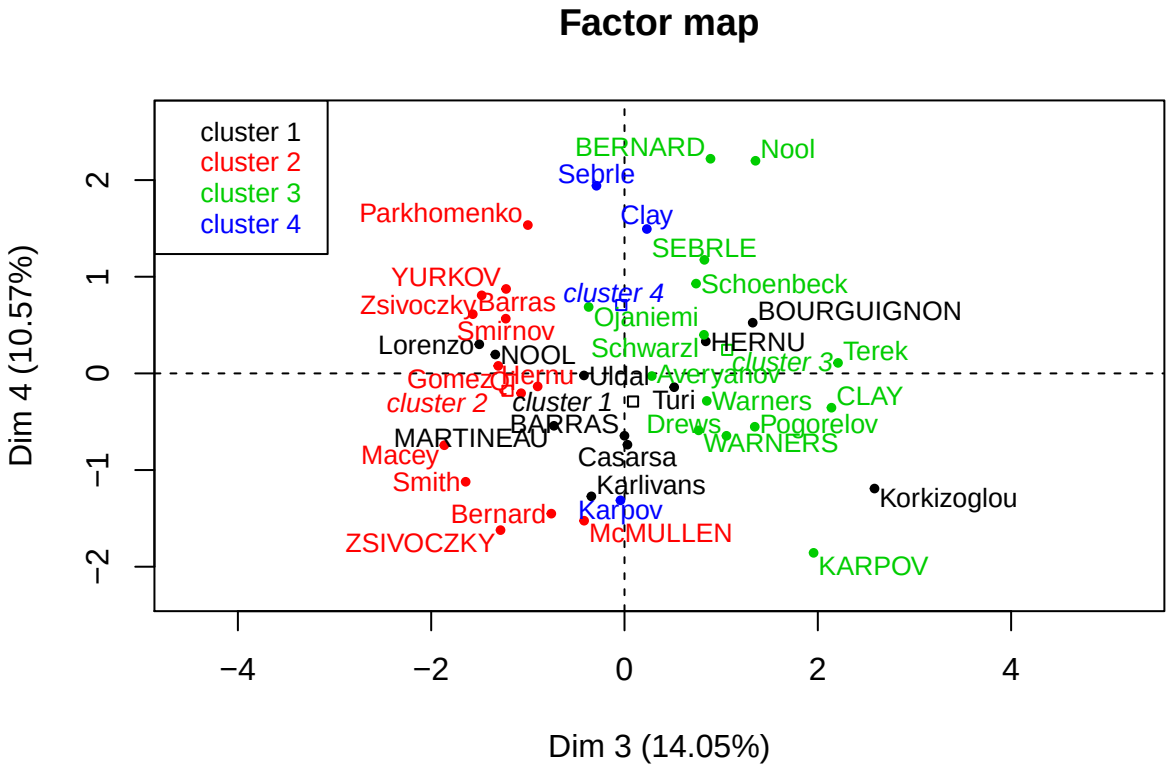
```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_cluster(res.hcpc,
             repel = TRUE,            # Avoid label overlapping
             show.clust.cent = TRUE,  # Show cluster centers
             palette = "jco",         # Color palette see ?ggpubr::ggpar
             ggtheme = theme_minimal(),
             main = "Factor map"
             )
```



Here we observe that classes 2 and 3 are not so different on this view. We can construct this plot on dimension 3 and 4 to see the difference. Here, the clustering algorithm looks at all the dimension to create clusters.

```
plot(res.hcpc, choice = "map", axes = c(3,4), ind.names = TRUE, draw.tree = FALSE, centers.plot = TRUE)
```



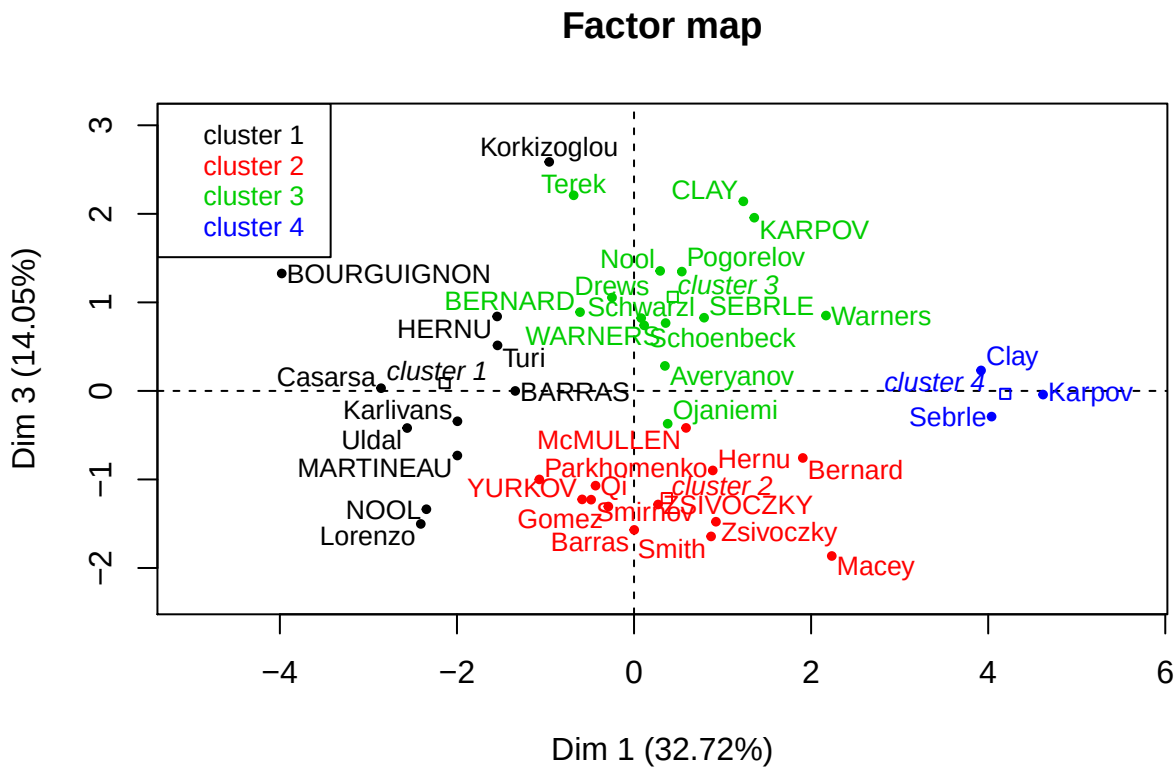

```
res.hcpc$desc.axes
```

```
##
## Link between the cluster variable and the quantitative variables
## ================================================================
##            Eta2      P-value
## Dim.1 0.8016073 4.504293e-13
## Dim.3 0.6067078 1.245846e-07
##
## Description of each cluster by quantitative variables
## ======================================================
## $`1`
##          v.test Mean in category  Overall mean sd in category Overall sd
## Dim.1 -4.528722        -2.138998 -1.273016e-14      0.7963714   1.808841
##           p.value
## Dim.1 5.934161e-06
##
## $`2`
##          v.test Mean in category Overall mean sd in category Overall sd
## Dim.3 -4.402151        -1.210787 -5.14675e-15       0.374769   1.185292
##           p.value
## Dim.3 1.071827e-05
##
## $`3`
##          v.test Mean in category  Overall mean sd in category Overall sd
## Dim.3  4.081905        1.0623706 -5.146750e-15      0.6806713   1.185292
## Dim.2 -2.363951       -0.6841363  3.343362e-15      1.1676867   1.318003
##           p.value
## Dim.3 4.466804e-05
## Dim.2 1.808120e-02
```

```
##
## $`4`
##         v.test Mean in category  Overall mean sd in category Overall sd
## Dim.1 4.118906           4.1926 -1.273016e-14      0.3060936   1.808841
##            p.value
## Dim.1 3.806749e-05
```

From this, we obtain the classes description by axes. It allows to see that class 1 present significatively lower values of dimension 1. Class 2 presents significatively lower values of dimension 3. Classes 3 presents higher values of dimension 3 and lower values for dimension 2. And class 4 has higher values on dimension 1. From this, it seems that a interesting representation can be obtained when plotting the results on axes 1 and 3 to have a representation of the cluster.

```
plot(res.hcpc, choice = "map", axes = c(1,3), ind.names = TRUE, draw.tree = FALSE, centers.plot = TRUE)
```



Finally, we can highlight the parangon of each class. It is a good way to illustrate what is a "typical" member of the class.

```
# command to vizualize parangon
res.hcpc$desc.ind$para
```

```
## Cluster: 1
##       Uldal      BARRAS   Karlivans       HERNU BOURGUIGNON
##    1.435595    1.538945    1.548147    1.757527    2.451654
## ------------------------------------------------------------
## Cluster: 2
##     Hernu    Barras        Qi Zsivoczky   Smirnov
##  1.491839  1.605181  1.692041  1.828199  1.840582
## ------------------------------------------------------------
## Cluster: 3
##   Schwarzl Schoenbeck    WARNERS  Averyanov  Pogorelov
```

```
##   1.130703   1.541205   1.681858   2.017040   2.129745
## ------------------------------------------------------------
## Cluster: 4
##     Clay   Sebrle   Karpov
## 1.535603 1.692349 2.569576
```

For the class 1, the parangon is Uldal, who is at a distance 1.43 from the barycenter of the class. Then you have Barras, Karlivans, and so on. For the second class it is Hernu. And so on.

One can also use the specificity, which are individuals that are the further possibles from other classes (more precisely the further away from other barycenter). For this the command is also simple:

```
res.hcpc$desc.ind$dist
```

```
## Cluster: 1
##     Casarsa BOURGUIGNON Korkizoglou        NOOL   MARTINEAU
##    5.047412    4.567775    4.040819    3.937180    3.827594
## ------------------------------------------------------------
## Cluster: 2
##       Smith Parkhomenko   Zsivoczky       Macey      YURKOV
##    4.343084    3.607607    3.386866    3.365448    3.237275
## ------------------------------------------------------------
## Cluster: 3
##     CLAY    Drews     Nool   KARPOV    Terek
## 4.192872 4.153718 4.006100 3.908004 3.545632
## ------------------------------------------------------------
## Cluster: 4
##   Karpov   Sebrle     Clay
## 4.799009 4.608231 4.465839
```

Here you can read that Casarsa is the individual from class 1 that is the further away from the other barycenters. In particular the closest barycenter is at a distance 5.08. You can interpret it as "Casarsa is really specific to class 1, it could not be classified in another class." For class 2 the equivalent is Smith.

End of solution question