Principal Component Analysis

Part 4 - Results interpretation

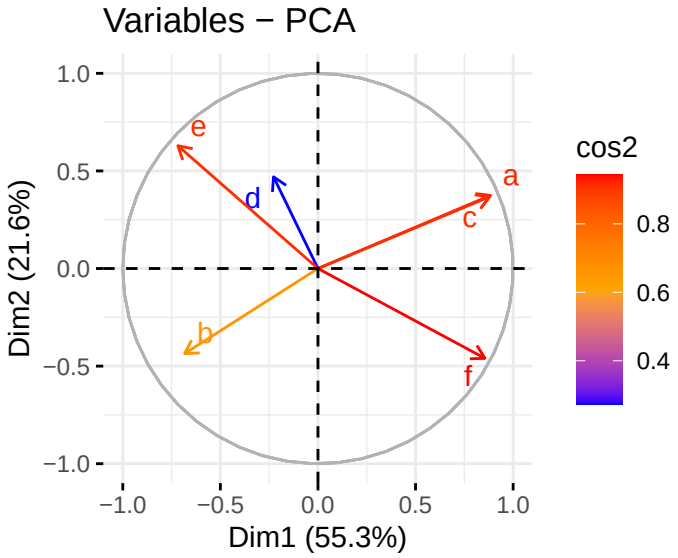
The aim of this course is to give you all the information you need to understand the PCA, as well as give you all tools necessary to put it in place. We will focus on the practical aspects of PCA without ignoring the maths beyond. Each part ends with several exercices to do. Some of them are very easy and other harder. I highly recommend to verify all demonstration presents during the course, for 2 main reasons: verify there is no mistake and help you to understand the objects manipulated. I truly believe that equations manipulation is very helpful in order to have a good intuition of why they are the way they are.

> *"Statistics is about reducing the amount of data."* **R. Fisher**

Projected inertia and other PCA applications Let's consider the following PCA:

```
library(FactoMineR)
results = PCA(data, graph = FALSE)

library(factoextra)
fviz_pca_var(results, axes = c(1,2),
             col.var = "cos2",
             gradient.cols = c("blue", "orange", "red"),
             repel = TRUE)
```



We said in Part 3 of this course that it's important to center-reduce our variables and that R's `PCA()` function do it by default. It means that if we perfectly project our variables (i.e: keep total initial inertia) when they are centered-reduced, their variance is still equal to 1. We then understand that the variance of our projected variables will be lower if our projection is worse. Concretely, it means that the closer a vector-variable to the correlation circle is, the closer to one its variance is and the better projected the variable is.

Let's take the example from above and do an interpretation of the correlation circle:

The first and second principal components represents respectively 55% and almost 22% of the original inertia. This means that the correlation circle plot summarizes more than 75% of the information present in the original data set. The variables a and c are very positively correlated, but negatively correlated to b. However, the latter is not perfectly projected. The vector-variables a, b and c seems orthogonal to d and per consequent uncorrelated. We have to keep in mind that d is badly projected. The last vector-variables e and f are negatively correlated and don't seem to have strong correlation to other variables.

Fortunately there is a way to tell is a projection is *globally* good:
We compute the percentage of projected inertia for the number of dimensions we choose. If we have p initial variables and we project in a subspace of d dimensions, we calculate it this way:

$$\frac{I_{proj}(X)}{I(X)} = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

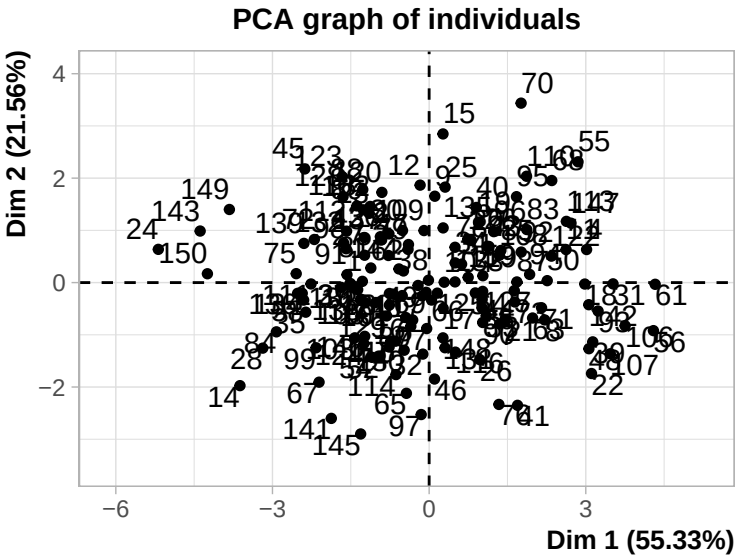
Let's see a concrete case with our last example (project on 2 dimensions instead of 6):

```
first_two_eig = results$eig[1:2]
all_eig = results$eig[1:6]
cat("Percentage of explained inertia:", sum(first_two_eig)/sum(all_eig)*100, "%")
```

```
## Percentage of explained inertia: 76.89336 %
```

Nice! We have the same number as before.

Now, let's project our individuals instead of the variables:

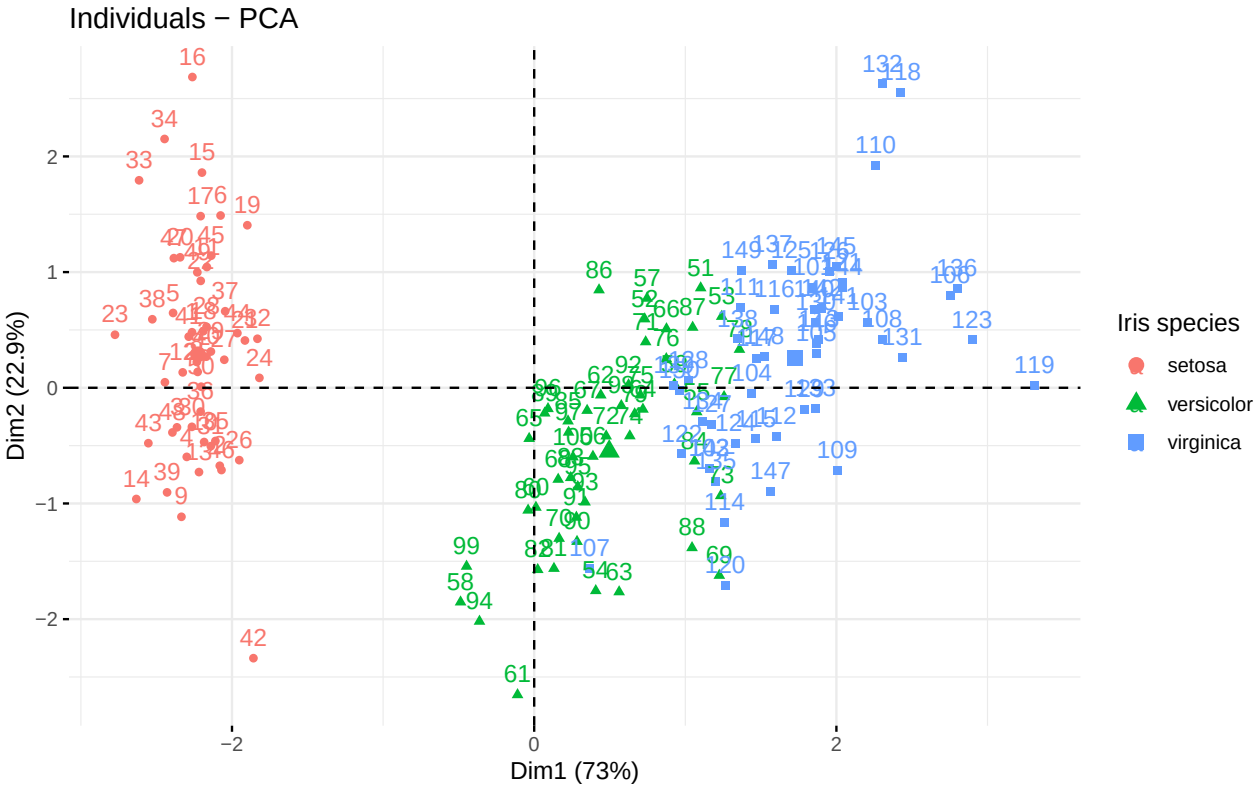


As you can see, it's a bit harder to interpret compared to variables. In my opinion, **interpreting the point-individuals projection is relevant in 2 situations**:
- You add a qualitative variable that allows to distinguish individuals thanks to colors.
- You don't have a high sample size (around 40 or 50 individuals maximum probably? Not sure, maybe more) and you *know* each individual. For example, if you work on Olympic athletes, it might be interesting to compare individuals with each other, compared to *random* unknown individuals.
- You're doing a clustering analysis like k-means. We will discuss an example after.

Let's see example of **projection with a qualitative variable to color individuals**

```
data("iris")
library(FactoMineR)
library(factoextra)

results = PCA(iris[, 1:4], graph = FALSE)
fviz_pca_ind(results, axes = c(1,2),
             col.ind = iris$Species,
             legend.title = "Iris species")
```



It seems that iris from the same species have the same characteristics (i.e: "correlated").

Example of clustering algorithms: k-means

I've wrote an article on the k-prototypes algorithm on my website (*click here*), which is very similar to k-means but a bit more complex. We will focus on k-means here since we're working on quantitative variables.

There is an important PCA application that we still don't have talked about. For the moment, we only mention PCA for correlation analysis, but it might be very useful of decreasing the number of dimension of a data set in itself. Why so?.

Curse of dimensionality:

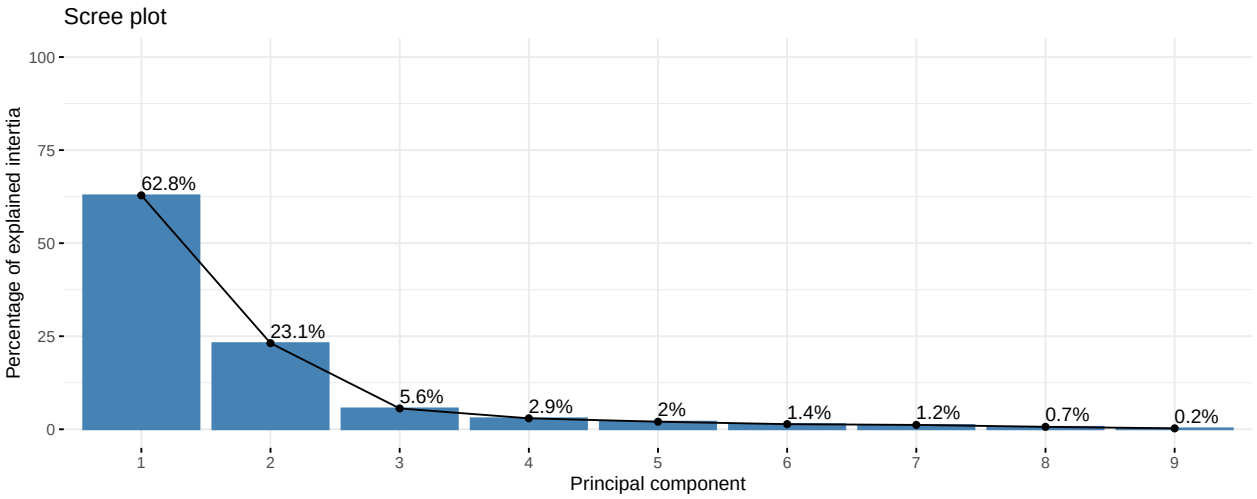
This is a problem that describes the fact that as the number of dimensions increases, the volume of space increases very rapidly and the data become sparse. By reducing the dimension of the data, this is much less a problem.

More generally, dimensionality reduction is useful in **fields where data tends to have lots of observations and/or variables**. For example, you can think of working on 2000 genes and a disease with a sample of 200 individuals. It's pretty hard to investigate relationship for each gene *by hand*. Work on this type of project is not the main goal of this course, but we will try to **solve some common problems encountered** compared to working with low-dimensional data.

What if we don't want to project on a 2 dimensional space?
We want to look at how much explained inertia a principal component can add to our analysis. For this, we generally use the following graph:

```
data("mtcars")
library(FactoMineR)
library(factoextra)
data = mtcars[, c(1:7, 10:11)]
result = PCA(data, graph=FALSE)

fviz_eig(result, addlabels = TRUE, ylim = c(0, 100),
         xlab = "Principal component",
         ylab = "Percentage of explained intertia")
```



We easily see here that most of principal components don't give us lot of information. One famous way to determine how many principal components keep is the **elbow method**. Imagine the last graph as an arm. Try to find the number of the principal component that represents the elbow of this arm. Put another way, that's the break where **adding another principal component is not synonym of parsimony**. In this case, it's not obvious and it might be 2 or 3. Have in mind that we use the **exact same logic** in order to determine the number of clusters to use (i.e: k) when doing, for example, k-means.

You might feel that **this method is rather subjective**, and it's a bit true: 2 different people can easily disagree on this. However, it's still often likely to find a break that satisfies most people. If you prefer a more objective method, you can use the **Kaiser rule, which consists on keeping only principal components where the inertia is above the average inertia**. More formally, we keep the d principal components such as:

$$\lambda_1, \cdots, \lambda_d, \ where \ \lambda_d > \frac{I_{proj}}{p}$$

Quality of representation and contribution We can quantify the quality of representation of the i-th individual to the α-th principal component:

$$QLT_\alpha(i) = cos^2(x_i, P_\alpha x_i) = \frac{(\Psi_i^\alpha)^2}{\|x_i\|^2}, \ \in \ ]0,1[$$

But we can calculate this for multiple principal components: we just have to sum the contribution to each principal component.

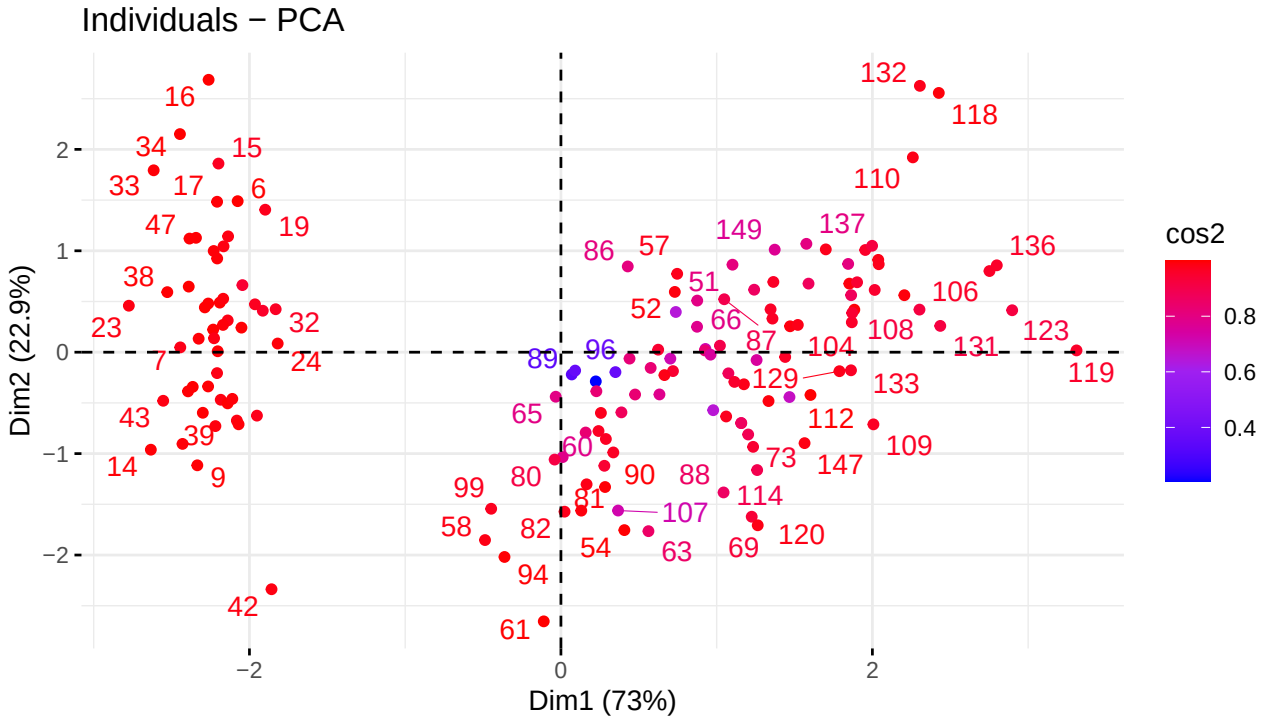
Let's compute an example of the quality of representation of the 1st individual on the correlation circle (i.e: the 2 first principal components):

```
results$ind$cos2[1,1] + results$ind$cos2[1,2]
```

```
## [1] 0.9968578
```

It's difficult to interpret in itself. One way to use this information is to create plots where **the color of the points-individuals is proportional to their quality of representation**. Let's see an example:

```
fviz_pca_ind(results, col.ind="cos2",
             gradient.cols = c("blue", "purple", "red"),
             repel = TRUE)
```



There are just a few flowers that are not very well projected, like 89 and 96.

We can also quantify the **quality of representation** of the j-th variable to the α-th principal component:

$$QLT_\alpha(j) = cor^2(x^j, \Psi_\alpha), \quad \in \ ]0,1[$$

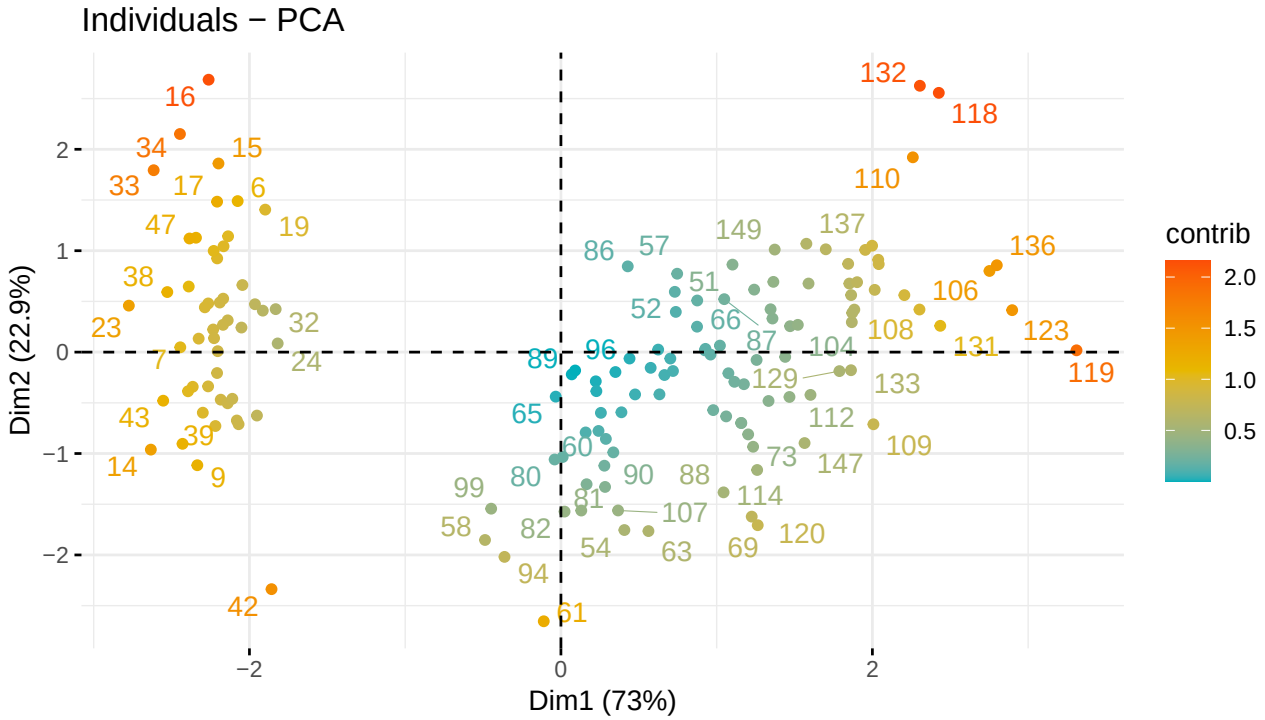
For the moment, we have only worked with very simple data set, without *missing values* or *outliers*. The latter refers to individuals that are **outstanding** compared to each other. For example, if we are working on alcohol consumption and in our sample the average weekly consumption is 10 units of alcohol, an individual who consumes 10 units **per day** can be considered as an outlier. There are multiple ways to detect them (*boxplot, z-score, isolation forest...*) but there is no perfect and consensual way to manage them. When doing PCA, there are a problem because **they make our analysis not robust**. A Robust analysis in statistics refers to the fact that we have very similar results whether or not we have our individuals. If removing an individual changes the conclusions, it means that our analysis is **not as robust as we would like to**.

How to know is an individual have too much impact on our PCA? We can compute **its contribution (in %) to each principal component**! Formally, it's described as follow:

$$CTR_\alpha(i) = \frac{\frac{1}{n}(\Psi_i^\alpha)^2}{Var(\Psi_\alpha)}$$

Here's a way to represent the contribution of our individuals:

```
fviz_pca_ind(results, col.ind="contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```

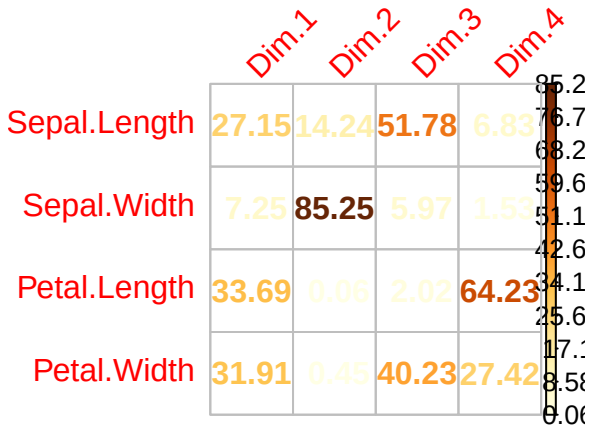


There is a similar equation in order to calculate the **contribution of a variable to a principal component**:

$$CTR_\alpha(j) = \frac{cor^2(x^j, \Psi_\alpha)}{\lambda_\alpha}$$

We can easily compute contribution of each variable to each principal component:

```
contribution = results$var$contrib
library(corrplot)
corrplot(contribution, is.corr=FALSE, tl.srt=45, method="number")
```



Exercises

1. Prove that $\sum_1^n CTR_\alpha(i) = 1$ (knowing that Ψ_α is centered).
2. Make a sentence in order to interpret the following vector:

```
results$ind$cos2[5:10,3] + results$ind$cos2[5:10,4]
```

```
##            5            6            7            8            9           10
## 0.0002508676 0.0001181242 0.0187033350 0.0016797895 0.0032411907 0.0130483736
```

3. What is the inertia of the iris data set when variables are scaled VS when they're not?

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

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##    0.6856935    0.1899794    3.1162779    0.5810063
```

```
apply(scale(iris[, 1:4]), 2, FUN=var)
```

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

4. Is the following PCA done on scaled variables?

```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1 4.20005343             92.4618723                           92.46187
## comp 2 0.24105294              5.3066483                           97.76852
## comp 3 0.07768810              1.7102610                           99.47878
## comp 4 0.02367619              0.5212184                          100.00000
```

5. According to the Kaizer rule and if our PCA is done on the correlation matrix, how can we quickly know which axes to keep?