Principal Component Analysis

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 why they are the way they are.

At the end, you will be able to:

- Understand the relevance (or not) of a PCA
- Interpret results from other people work
- Implement PCA for your own projects

Applications of PCA:

- Data visualization (i.e. all possible and unimaginable fields)
- Psychometrics, econometrics, epidemiology, bio-stats... (latent variables, simple/multiple causality determination, structural equations, and more generally: factorial analysis)
- Over-fitting of high dimensional data / Curse of dimensionality
- Machine learning algorithms

Pre-requisites:

- *Linear algebra*: not a high level is necessary, but it is welcomed in order to fully understand
- *Basic descriptive statistics*: know what is a mean, standard deviation, variance, sample distribution, (ideally) correlation analysis...
- *R programming*: since it will be the software used in this course
- *A good intuition*: always nice to have

Even if there are some pre-requisites, this course will take one thing at a time and you might understand most of it without all the latter.

Table of content

This course is **divided in 4 parts**. The 1st and 2nd part are the shortest and easiest to understand but are very important to give you the idea of **why** do we do PCA, as well as help you understand next parts. The 3rd and 4th part are the hardest to understand, mainly because of the maths. Once again, **it will not be an informal academic course**: it will has lots of example and favoring good intuition over exactitude. Also, it is not exhaustive about PCA: you can see it as a good introduction to correlation analysis with PCA. **The R software will be used all along this course** and you're highly encourage to verify all calculations did with your own computer (and ideally on other data than mines).

Part 1 - Descriptive statistics and correlation analysis

- Reminder of the basic of statistics
- What is correlation?
- Why are we interested in correlation?

- Exercises

Part 2 - Relevance and intuition beyond PCA

- Why variance matters so much?
- What is PCA useful for?
- Concretely, what is going on when doing a PCA on a software?
- Exercises

Part 3 - How PCA works?

- Premise on the mathematical tools used
- How our variables are projected?
- Importance of normalization
- What is a principal component?
- Exercises

Part 4 - Results interpretation

- How to know if a PCA has gone well?
- What are the criteria that allow us to interpret the results?
- Quality of representation and contribution of individuals
- Exercises

Part 5 - Project

- A PCA project to put everything in practice
- You choose the data you want and have to answer different questions about them

Part 1 - Descriptive statistics and correlation analysis

Reminder of basic statistics For a sample size of n, the mean is defined as the average value of each individual:

$$\overline{x} = \frac{\sum x_i}{n}$$

But you can easily find examples where 2 different samples have the same mean but don't have the same distribution. One way to illustrate this, is to compute other statistic, like standard deviation, defined as:

$$\sigma_x = \sqrt{\frac{\sum(x_i - \overline{x})^2}{n}}$$

You can interpret it as (*more or less*), the average difference between an individual and the mean of the sample he comes from.

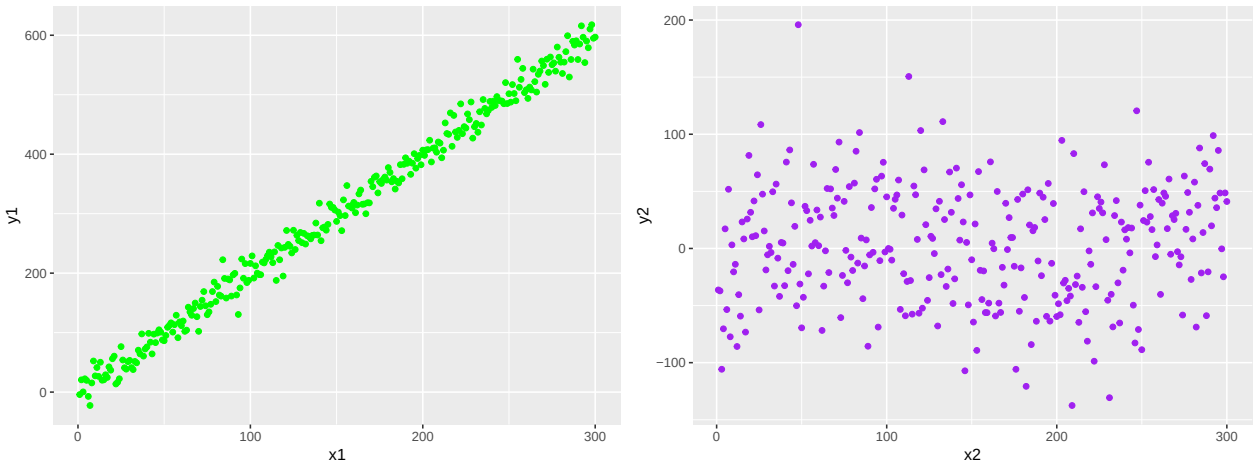
It's pretty common to also compute, the variance, the squared standard deviation:

$$Var(x) = \frac{\sum(x_i - \overline{x})^2}{n}$$

These are the simplest univariate descriptive statistics you can find. You should be very comfortable working with them and interpret their value.

What is correlation? We want to know if 2 variables are correlated. But, if we ask ourselves what this formally means, it's not that easy to define. Because we are statisticians, we will use the following definition: 2 variables are correlated if they tends to not be dependent.

```
#sample1
n = 300
x1 = seq(1, n)
y1 = 2*x1 + rnorm(n=n, sd=18)
data1 = data.frame(cbind(x1, y1))

#sample2
n = 300
x2 = seq(1, n)
y2 = rnorm(n=n, sd=50)
data2 = data.frame(cbind(x2, y2))

#plot both relationship in order to make it more intuitive
library(ggplot2)
plot1 = ggplot(data1, aes(x=x1, y=y1)) + geom_point(col="green")
plot2 = ggplot(data2, aes(x=x2, y=y2)) + geom_point(col="purple")

library(ggpubr)
ggarrange(plot1, plot2)
```



With plots, it's pretty obvious to detect when variables are correlated. So the question we want to ask is: **how can we objectively qualify correlation** (i.e: without using plots)?
For this, we use **Pearson correlation coefficient**:

$$r = \frac{cov(x,y)}{\sigma_x \sigma_y}$$

The function $cov(x,y)$ is the covariance between x and y. The latter tends to tell us if, in a finite sample, when x_i is above (or below) the mean of its sample, y_i tends to also be above (or below) the mean of its sample. **The higher the covariance is, the more when x_i is above its mean, y_i also is**. To the contrary, the lower the covariance is, the more when x_i is above its mean, y_i is below. More formally, we can describe it this way:

$$cov(x,y) = \frac{1}{n}\sum(x_i - \overline{x})(y_i - \overline{y})$$

Covariance actually measures correlation between 2 variables. But **Pearson correlation coefficient is better because it normalizes the covariance between -1 and 1**. In fact, covariance unit is the product of the units of x and y: doesn't make lot of sense for us. That's why we divide it by the product of standard deviation.

Pearson correlation coefficient can be interpret as follow: the closer it is to 1 (resp. -1), the more there is a positive (resp. negative) correlation. If it's near 0, it seems that there is no correlation between variables (that is not completely true, but we will do like it is). If we take back our 2 last sample back, we can compare their correlation coefficient.
Ask yourself before checking the results: which one will be the highest? Why?

```
#sample1
n = 100
x1 = seq(1, n)
y1 = -2*x1 + rnorm(n=n, sd=18)
cat("Correlation between variables from our first sample: r =", cor(x1,y1))
```

```
## Correlation between variables from our first sample: r = -0.9512587
```

```
#sample2
n = 100
x2 = seq(1, n)
y2 = rnorm(n=n, sd=5)
cat("Correlation between variables from second first sample: r =", cor(x2,y2))
```

```
## Correlation between variables from second first sample: r = -0.09381702
```

The first 2 variables are very (negatively) correlated (r is near -1).
The other 2 variables are much less correlated (r is near 0).

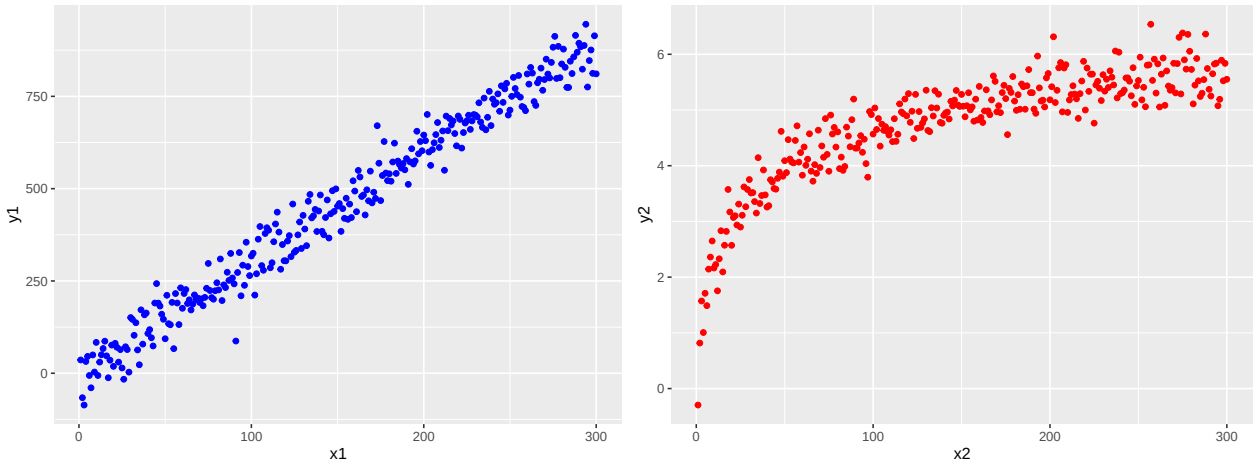
Interlude: non-linear correlation

Since the Pearson coefficient correlation only measures linear correlation, **it will miss correlation that are not linear** (e.g. exponential or logarithmic). In order to solve this issue, statisticians have invented other measure of correlation. We will discuss the main one: Spearman coefficient correlation.
It's defined as follow:

$$s = \frac{\frac{1}{n}\sum(rx_i - \overline{rx}_n)(ry_i - \overline{ry}_n)}{r_\sigma(x)r_\sigma(y)}$$

Where r design the rank of the individual compared to other individuals. Using this statistic compared to the Pearson's one allow us to measure non-linear relationship. Spearman's coefficient is considered as non-parametric because it doesn't use the value of x_i but its rank. Individual will be ranked. Let's see an example of when this might be useful.

```
#sample with linear relationship between x and y
n = 300
x1 = seq(1, n)
y1 = 3*x1 + rnorm(n=n, sd=45)
data1 = data.frame(x1,y1)

#sample with logarithmic relationship between x and y
n = 300
x2 = seq(1, n)
y2 = log(x2) + rnorm(n=n, sd=0.3)
data2 = data.frame(x2,y2)

#plot both relationship in order to make it more intuitive
library(ggplot2)
plot1 = ggplot(data1, aes(x=x1, y=y1)) + geom_point(col="blue")
plot2 = ggplot(data2, aes(x=x2, y=y2)) + geom_point(col="red")

library(ggpubr)
ggarrange(plot1, plot2)
```



Compute both Pearson and Spearman coefficients correlation

```
## Pearson correlation between variables from our first sample: r = 0.985991
## Spearman correlation between variables from our first sample: r = 0.9870705

## Pearson correlation between variables from our second sample: r = 0.83647
## Spearman correlation between variables from our second sample: r = 0.8982242
```

Spearman coefficient in higher for the sample with a logarithmic (non-linear) relationship. This tool is useful to illustrate that 2 variables can be considered as correlated but their relationship is more complex that a straight line.

Why are we interested in correlation? Depending of what you're interested in (descriptive or inferential statistics), you will not use correlation for the same reason and the same way. **Correlation is an important statistic tool used in a daily basis in lots of field**. For example, if we observe a strong correlation between a genome and breast cancer, we might do some prevention for people with this genome and easily attenuate the cancer impact, especially if it's not already present.
However, **you should always have in mind that correlation has not so much to do with causality**. The latter is way more complex and will not be discussed here.

Exercises

1. Prove that $cov(x, x) = var(x)$
2. Prove that $cor(x, x) = 1$
3. In a finite sample, what is the difference between: the average difference between an individual and the mean VS the standard deviation (yes it's not the same)?
4. With R, create two samples: one where the Pearson coefficient correlation is equal to 1 (but with 2 distinct variables) and one where it's between [-0.1 ; 0.1]. Use a sample size >100.
5. What is the condition on the type of the variables used in order to compute the Pearson (or Spearman) coefficient correlation (it's implicitly said in the course)?
6. *With your own words*, make a short description of what correlation is.

Part 2 - Relevance and intuition beyond PCA

Why variance matters so much? In statistics, it's very common that we want to explain the more variance possible. It might sound not very clear at first. In order to give the intuition of what it means, we will see concrete examples. The variance is a measure of variability of a variable. The higher (resp. lower) it is, the more (resp. less) volatile our variable is.

For example, if we want to make a linear regression between annual income, age, level of education and number of children, we want to explain the more variance possible. Intuitively, if our model (i.e: a simplification of reality) explains most (>90%) of the variance from our initial data, we can be confident with our model predictions. It's because individual are all different that statistics is so relevant.

What is PCA useful for? For the moment, we only have computed correlation between two variables at a time. But in real life data, you will have lot of different variables and calculate correlation for each pair has several problems. First, it's hard to have a global idea of which variable is correlated to which one. In fact, if you have p variables, you have $C(p) = \frac{p!}{2!(p-2)!} = \frac{p(p-1)}{2} = \frac{p^2-p}{2}$ Pearson coefficients correlation to interpret (trust me, not very fun and/or relevant). You can see that C increases quadratically with p.

But, there is a way to make it easy to compute: matrix correlation. It's a sheet that contains Pearson coefficients correlation for all numeric variables. It's an easy way to summarize lot of information. In R, it's

pretty easy to compute:

```
library(corrplot)
data("iris")
data = iris[, 1:4] #remove the qualitative variable
summary(data)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

```
correlation = cor(data)
corrplot(correlation, tl.srt=45, method="number")
```

In this case we have only 4 variables. And, as said before, it's very common to have way more than that. I let you imagine **how hard it is to interpret a correlation matrix for 10, 20 or more variables**. And that's where PCA appears: it proposes a way to show correlation between multiple variables in a very simple way. It projects our initial variables into a smaller space and plots vector of variables. If 2 vectors seems to go in the same direction and our projection had work well (we will see what it means), **that imply that these 2 variables are correlated**.

The main motivation for PCA is that in order to represent/visualize data, **one must restrict oneself to a**

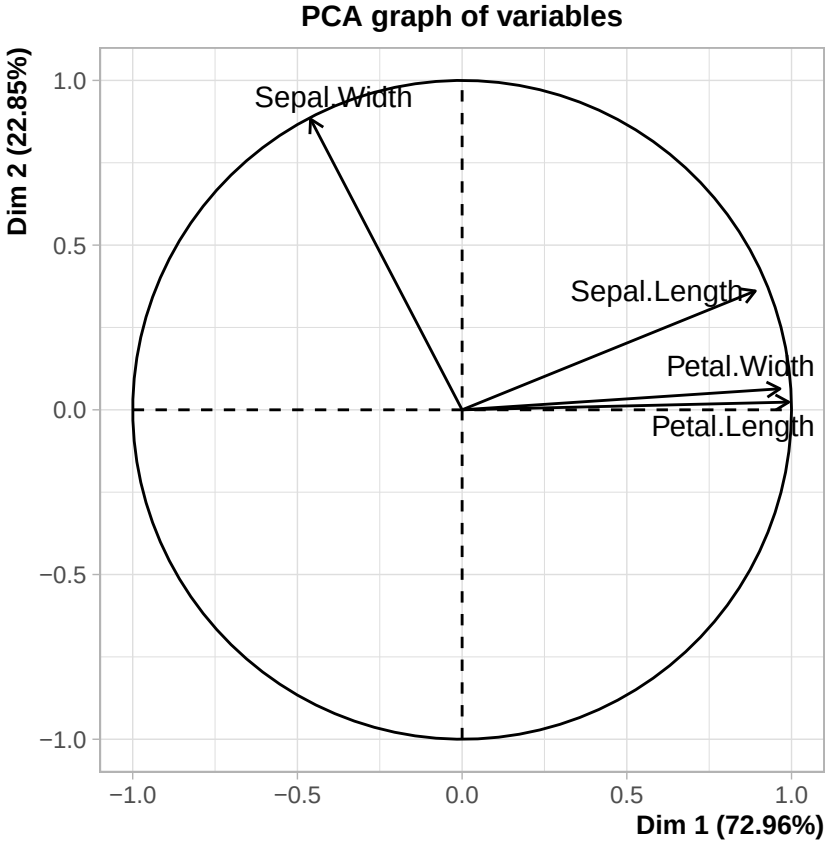
small number of dimensions. Indeed, a correlation circle gives, more or less, the same information as a correlation matrix but in a much more intuitive and accessible way.

From a data set of more or less correlated quantitative variables, we wish to create new variables, de-correlated, representing **the principal components**. The objective is to keep the maximum amount of information present in our initial data set while reducing the number of dimensions necessary to express it (generally 2 or even 3).

Initially, in a data set with p variables, p dimensions are needed to express all the information contained (one dimension per variable). **PCA seeks to "create" the dimensions with the maximum amount of information**. Expressed more formally, we want to create the principal components that express the most variance (or inertia, a geometric formalization of the variance) possible.
Important: unless our initial variables are perfectly correlated (we can express each variable as a linear combination of one or more others), **it is impossible to keep all the information in our principal components**. However, we quickly realize that there is no point in keeping co-linear variables from the point of view of data analysis (e.g. having data in two different currencies at a fixed date).

Concretely, what is going on when doing a PCA on a software? Let's do a PCA on the iris data set used before!

```
library(FactoMineR)
pca_results = PCA(data, graph = FALSE)
plot.PCA(pca_results, choix = "var")
```



For the moment, we will only be interested in the **correlation circle plot** (graph of variables). If you remember what's said before, you should have a first idea of which variable is correlated to which one. If you

don't, the thing important to know here is the fact that **each variable is associated to a vector**, and if 2 vectors are going in the same direction, **it means that they are positively correlated**.
Based on this, we can intuitively think of a relationship between correlation and cosinus angle of 2 variables from our correlation circle. In fact, there is, and it's pretty intuitive if you understood the correlation circle:

$$cor(x, y) = cos\theta(x, y)$$

For the moment, it doesn't really matter if you don't understand what the axes are and how to interpret the percentage associated. The only thing that really matter here is the fact we are able to summarize a matrix correlation in a very simple way. If you're not convince, **I highly encourage you** to compare the Pearson coefficients correlation obtained before and the direction of each vector-variable.

PCA is very powerful because it's allow us to **show correlation between a lot of variables** in a 2 dimensional graph although it should require as much dimensions as the numbers of variables to represent them. You might acquire with the fact that it's **not very intuitive to create a scatter plot with more than 3 dimensions**.

Exercises

1. With R, create a variable which variance is equal to 0 with a sample size of 100.
2. If 2 vector-variables x and y are going in the striclty opposite direction, to what is equal $cos\theta(x, y)$?
3. Using internet (or other), find a statistic that measures the amount of variance explained (there are multiple ones).
4. Why did I remove the qualitative variable of the iris data set before doing the PCA?
5. Choose and explain a concrete case where PCA might be relevant? irelevant?

Part 3 - How PCA works?

Premise on the mathematical tools used You don't need to memorize all of the following, but you can refer yourself to it when you don't fully understand a calculation.

Definitions
X = is the matrix of our initial variables. In a sample size of n and p variables, $dim(X) = (n \times p)$.
Y = is the centered ($y_i^j = x_i - \overline{x}^j$) matrix of our initial variables.
Z = is centered reduced ($z_i^j = \frac{x_i - \overline{x}^j}{\sigma_j}$) matrix of our initial variables.
$N = diag(\frac{1}{n})$
$M = I_d$

$V = Y^T NY$ = a covariance matrix
$R = Z^T NZ$ = a correlation matrix

Some properties
$Proj(a) = Pa = uu^T a = b$, where $Proj(a)$ is the projection of a on the subspace generated by u
$\left\|y^j\right\|_N = \sqrt{< y^j, y^j >_N} = \sqrt{(y^j)^T N y^j} = \sigma_j$, where $mean(y^j) = \overline{y}^j = 0$
$cor(x, y) = cos\theta(x, y) = \frac{<x,y>}{\|x\|\|y\|}$

Inertia
The total inertia of a point cloud is measured as the sum of the squares of the distances of the points from the center of gravity. Assuming all variables and individuals have the same weight compared to each other

$(N = diag(n_i) = diag(\frac{1}{n})$ and $M = I_d)$, we can formally describe inertia as follow:

$$I_g = \sum n_i \|x_i - g\|_M^2$$

$$= \sum n_i (x_i - g)^T M (x_i - g)$$

$$= \sum n_i < x_i - g, x_i - g >_M$$

$$= \sum n_i < y_i, y_i >_M$$

$$= \sum n_i y_i^T M y_i$$

Assuming x is centered $(mean(x) = 0)$, we have:

$$I_g = \sum n_i x_i^T M x_i = \frac{1}{n} \sum_1^n \sum_1^p (x_i^j)^2 = \sum_1^p Var(x^j)$$

We will try to represent this living cloud in $\mathbb{R}^p$ by projecting it on a subspace of dimension $d < p$ such that the cloud of projected points distorts as little as possible the distances compared to the initial distances. And we will see that we will necessarily lose inertia. Put more formally, we want to project this point cloud is a sub-space that maximizes the projected inertia.

How our variables are projected? The goal of PCA is to reduce dimension of our data set: we want to summarize the more information possible by projecting our variables into the first few principal components. These latters are defined as:
"*The principal components of a collection of points in a real coordinate space are a sequence of p unit vectors, where the i-th vector is the direction of a line that best fits the data while being orthogonal to the first* $i - 1$ *vectors"* (Wikipedia).

We want to project our variables in a specific way. Based on your intuition, which one of the following projections (green points) is the best?

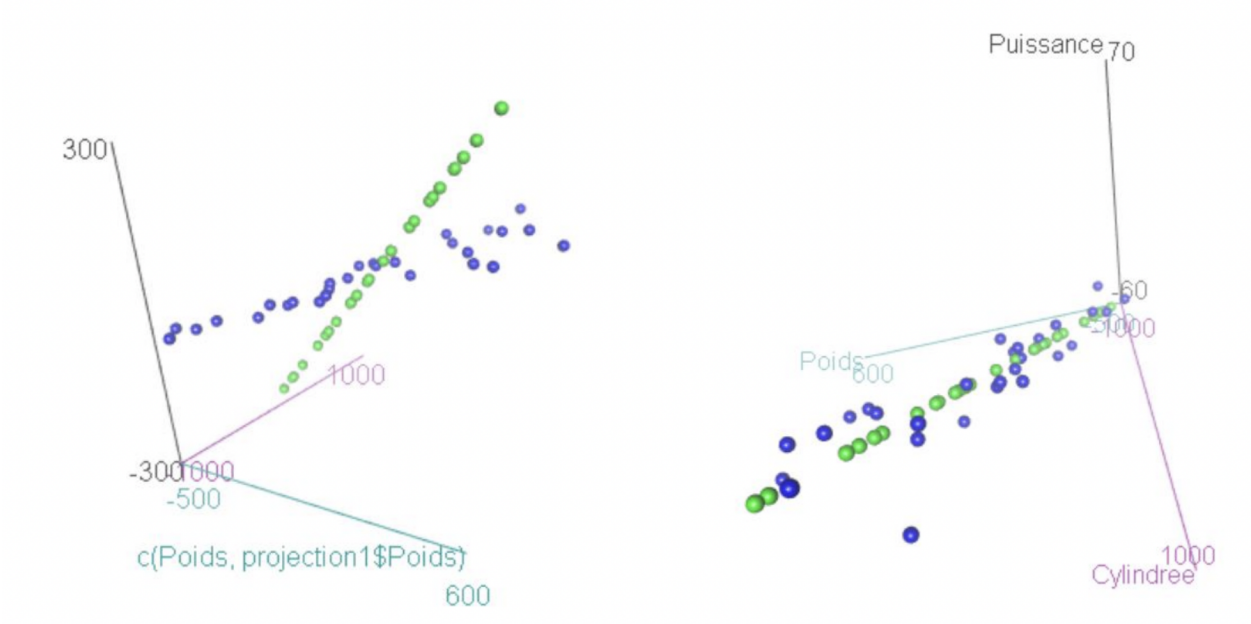


Figure 1: ***2 differents projections of the same points cloud***

Projection in a sub-space of dimension 1
As said before, we want to project a point cloud is a sub-space that maximizes the projected inertia. This essentially means that doing a PCA is equivalent as solving an optimization problem, formally describe as follow:

$$u = arg\ max\ I_{projected} = arg\ max \sum_{1}^{n} \|Py_i\|_M^2$$

$$subject\ to:\ u\ \in\ \mathbb{R}^p,\ \|u\|_M = 1$$

With P an M-orthogonal projector on a subspace of dimension 1, generated by a vector u, M-orthonormal: $u^T M u = 1$. The projector is written so: $P = uu^T M$.

Solving this problem is obtained for the largest eigenvector of the matrix VM: it suffices to take an M-orthonormal vector of VM associated to the largest eigenvalue λ_1 of VM.

Projection in a sub-space of dimension d
Generalization of the last problem is equivalent to determines the d largest eigenvalues $\lambda_1, \cdots, \lambda_d$ of VM. This is the same as diagonalizing the square matrix VM:

$$VM = S^T \Lambda S = \sum_{k=1}^{p} \lambda_k u_k u_k^T M$$

With $S = [u_1, \cdots, u_p]$. We choose the d largest in order to defines the optimal projector.

Principal components
Wikipedia description of principal components is great:
"In data analysis, the first principal component of a set of p variables, presumed to be jointly normally distributed, is the derived variable formed as a linear combination of the original variables that explains the most [inertia]. The second principal component explains the most [inertia] in what is left once the effect of the first component is removed, and we may proceed through p iterations until all the [inertia] is explained. PCA is most commonly used when many of the variables are highly correlated with each other and it is desirable to reduce their number to an independent set.".

In this course, we define the **first principal component** as follow:

$$\Psi_1 = YMu_1$$

With

$$\begin{aligned} Var(\Psi_1) &= \|\Psi_1\|_N^2 \\ &=< \Psi_1, \Psi_1 >_N = \Psi_1^T N \Psi_1 \\ &= (YMu_1)^T NYMu_1 \\ &= u_1^T MY^T NYMu_1 \\ &= u_1^T M(VMu_1) = u_1^T M(\lambda_1 u_1) \\ &= \lambda_1 (u_1^T Mu_1) = \lambda_1 \end{aligned}$$

One last thing important for the next part: we can calculate the correlation between a variable and a principal component. If the correlation is strong, it means that the principal component mainly *represents* the variable in question. More formally, we can describe it this way:

$$cor(x^j, \Psi_d) = cos\theta(x, \Psi_d)$$

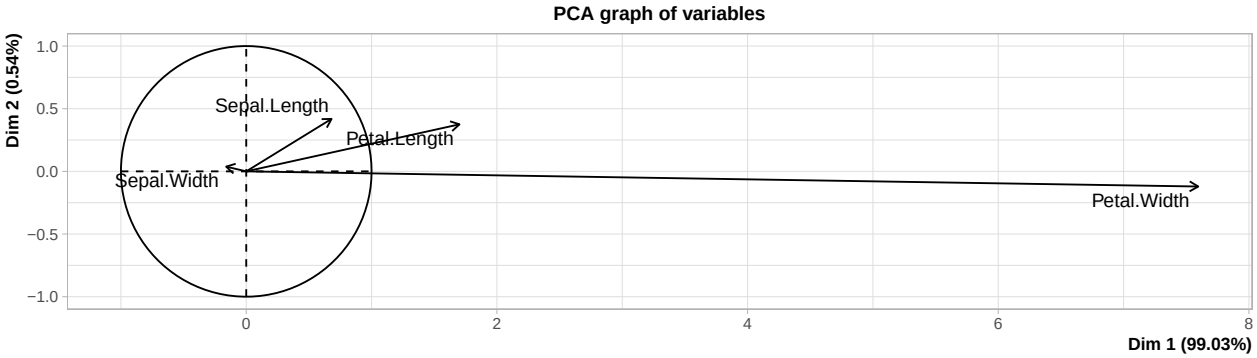
Importance of normalization In this course, we have talked multiple times of centered reduced variables without explanation. If you have a really good intuition (or already know about PCA), you might have an idea of why this is important. We said that PCA consists in solving an optimization problem where we want to maximize projected inertia. The other important fact is that inertia is the sum of all the variances of our initial data set (for centered variables). You get it?

Let's take a concrete example with the iris data set! In the latter, all variables are in cm, **but I will change one of them in tens of mm**.

```
data("iris")
data = iris[, 1:4] #remove the qualitative variable
data$Petal.Width = data$Petal.Width*10 #Petal.Width is now in tens of cm
summary(data)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   : 1.00
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.: 3.00
##  Median :5.800   Median :3.000   Median :4.350   Median :13.00
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :11.99
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:18.00
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :25.00
```

```
library(FactoMineR)
#explicit that we don't want to normalize our data (i.e: center-reduce)
pca_results = PCA(data, graph = FALSE, scale.unit = FALSE)
plot.PCA(pca_results, choix = "var")
```



This seems like petal width is different than other variables, but where does it come from? When R is computing the eigenvalue associated to the first principal component (= maximizing projected inertia), **it uses the covariance matrix**. But because variables are not on the same scale, each variable will a have a weight equivalent to the scale of its variance. Because petal width variance is in a higher scale, the **first principal component and itself are very correlated**. The weight of petal width is very important compared to the other variables. Does this means that we can only do PCA on variables that are on the same scale? Yes (not exactly right, but doesn't really matters here). But it's very easy to transform our initial variables into new ones that are on the same scale: center-reduction. The latter has some very useful properties.

```
data("iris")
iris$Sepal.Length2 = scale(iris$Sepal.Length)

cat("Mean (SD) of sepal lenght:",
    mean(iris$Sepal.Length),
    "(",sd(iris$Sepal.Length),")",
    "\nMean (SD) of sepal lenght centered-reduced:",
```

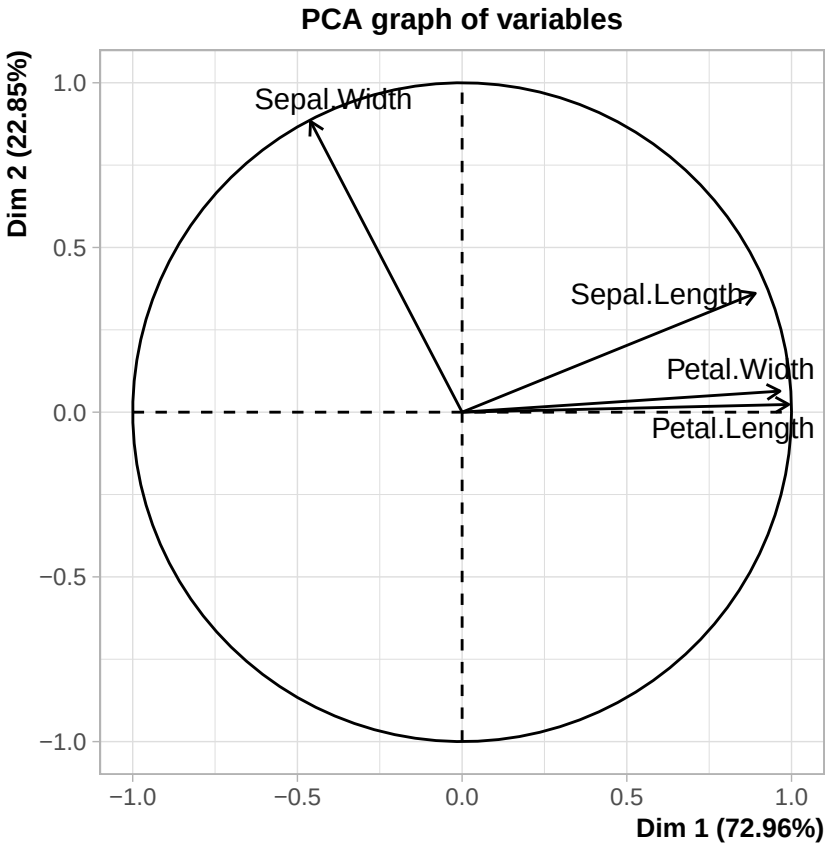
```
    round(mean(iris$Sepal.Length2)),
    "(",sd(iris$Sepal.Length2),")")
```

```
## Mean (SD) of sepal lenght: 5.843333 ( 0.8280661 )
## Mean (SD) of sepal lenght centered-reduced: 0 ( 1 )
```

Centered-reduced variables always have, by definition, a **mean of** 0 **and standard deviation of** 1. By doing this transformation on our initial data set, we ensure that none of the variables will have a different weight than others when solving the optimization problem. By default, `PCA()` function from R normalize all the variables, but it's important than you understand why we do this. You will see that this also has a **particular importance during results interpretation**.

To summary: we compute **eigenvalues of the correlation matrix and not the covariance one** in order to put all variables on the same scale, which gives them the same weight. If we don't do it, the variables that are on a higher scale will be the only variables that are well projected. In fact, **maximizing projected inertia will essentially maximizes inertia of these variables**.

```
data("iris")
data = iris[, 1:4] #remove the qualitative variable

library(FactoMineR)
#explicit that we want to normalize our data
pca_results = PCA(data, graph = FALSE, scale.unit = TRUE)
plot.PCA(pca_results, choix = "var")
```



Exercises

1. In R, what is the difference between `cor(X)`, `cor(Y)` and `cor(Z)`? Why so?

2. With R, on the iris data set, check if there is any difference between `cor(Z)` (Z is the matrix of the centered reduced inital data) and $Z^T NZ$ (the symbol for matrix multiplication in R is `%*%`). What's the cause of that?
3. What is the total inertia of a cloud of point-individuals when the variables are centered reduced? Prove it from: $I_g(X) = \frac{1}{n} \sum \|y_i\|^2$
4. What can we say about orthonality and correlation? You can use the example of Sepal.Width-Sepal.Length and Petal.Width-Petal.Length from above to illustrate. Is is consistent with computing `cor(iris$Sepal.Width, iris$Sepal.Length)` and `cor(iris$Petal.Width, iris$Petal.Length)`?
5. Knowing that inertia is the sum of the variances of a data set, describe it as the trace of a matrix.
6. *With your own words*, make a short description of what PCA is and how it works.

Part 4 - Results interpretation

How to know if a PCA has gone well? As said multiple times, PCA mainly consists in solving the optimization problem of maximizing projected inertia. Fortunately, we have the possibility to know how much of the total inertia has been conserved. A good first measure of the projection quality is the percentage of inertia projected on total inertia. In we project on a subspace of d dimensions of our initial p variables, we can describe it formally as:

$$Q_p = \frac{I_{proj}(X)}{I(X)} = \frac{\sum_1^d \lambda_i}{\sum_1^p \lambda_i}$$

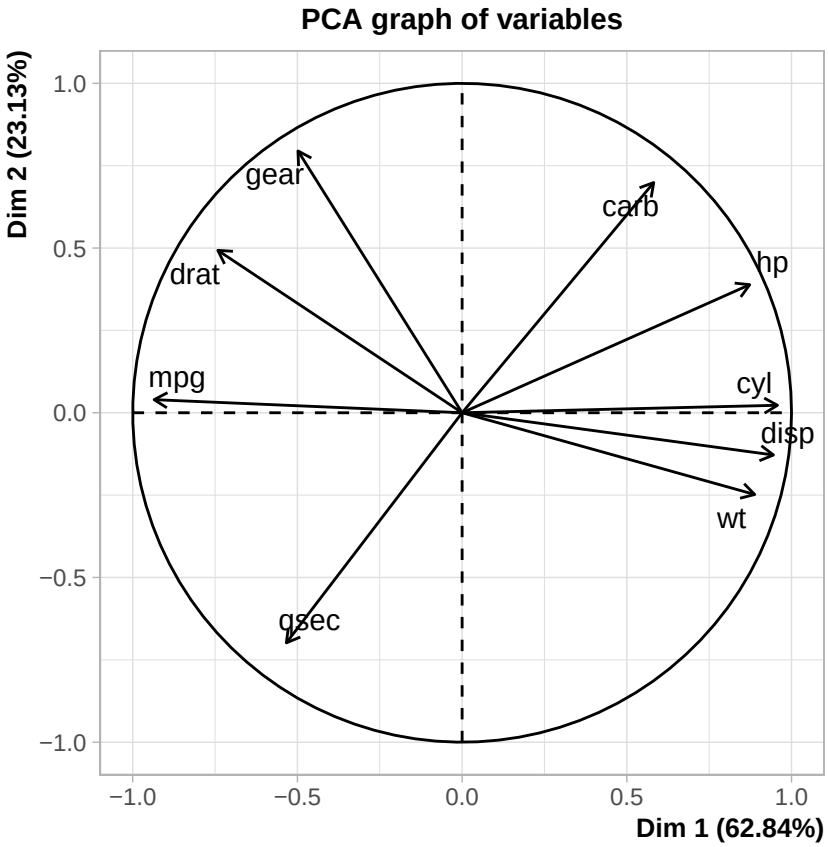
Because

$$I_{proj}(X) = Trace(VMP) = \sum_1^d \lambda_i$$

The percentage on each axis corresponds to the percentage of inertia conserved in the principal component associated. In the following example, the first principal component (x axis) corresponds to 62.84% of the total inertia. The second principal component (y axis) corresponds to 23.13% of the total inertia. It tells us that the 2 first principal components corresponds to 62.84 + 23.13 = 85.97 % of the total inertia. Put another way, we have been able to conserved more than 85% of the initial information contained in 9 dimensions into only 2. Impressive right?

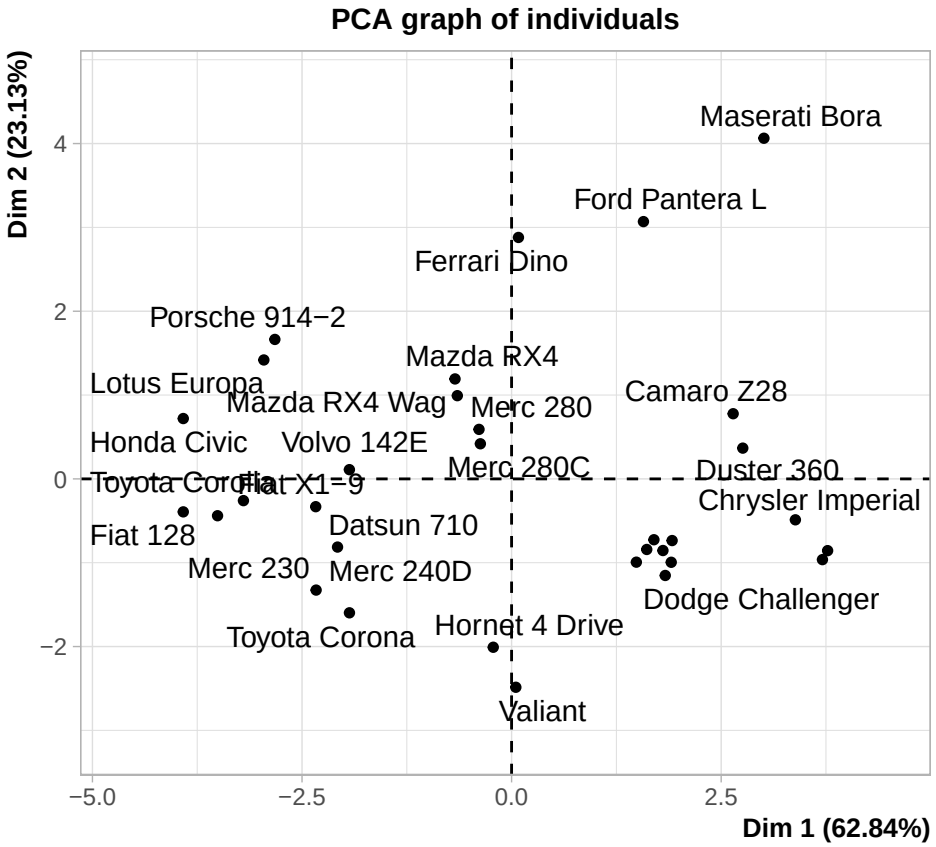
Also, there is a graphic way to determine if variables have been correctly projected. We know that a vector norm is equal to its standard deviation. It means that, when looking at the correlation circle, the closer a vector-variable is to the circle (standard deviation close to 1), the better it is projected. In the example below, all variables are near the circle which means that the projection is pretty good (we already knew it from the 85% before). However, we can say that the variables *qsec* and *drat* seems a bit less well projected.

```
data("mtcars")
data = mtcars[, c(1:7, 10:11)]

library(FactoMineR)
pca_results = PCA(data, graph = FALSE)
plot.PCA(pca_results, choix = "var")
```



Because projection is pretty good here, we can legitimately interpret correlation between variables. However, for the moment, we have only talked about correlation between variables. What about individuals? PCA allows us to projected individuals into that new space and so, naturally interpret correlation between them. We can consider that 2 individuals are *"correlated"* (not a real correlation measure, but more a similarity) if they tends to have the same characteristics: they're close in the space, according to a metric like euclidean distance for example. Let's see what it looks like.

```
data("mtcars")
data = mtcars[, c(1:7, 10:11)]

library(FactoMineR)
pca_results = PCA(data, graph = FALSE)
plot.PCA(pca_results, choix = "ind")
```



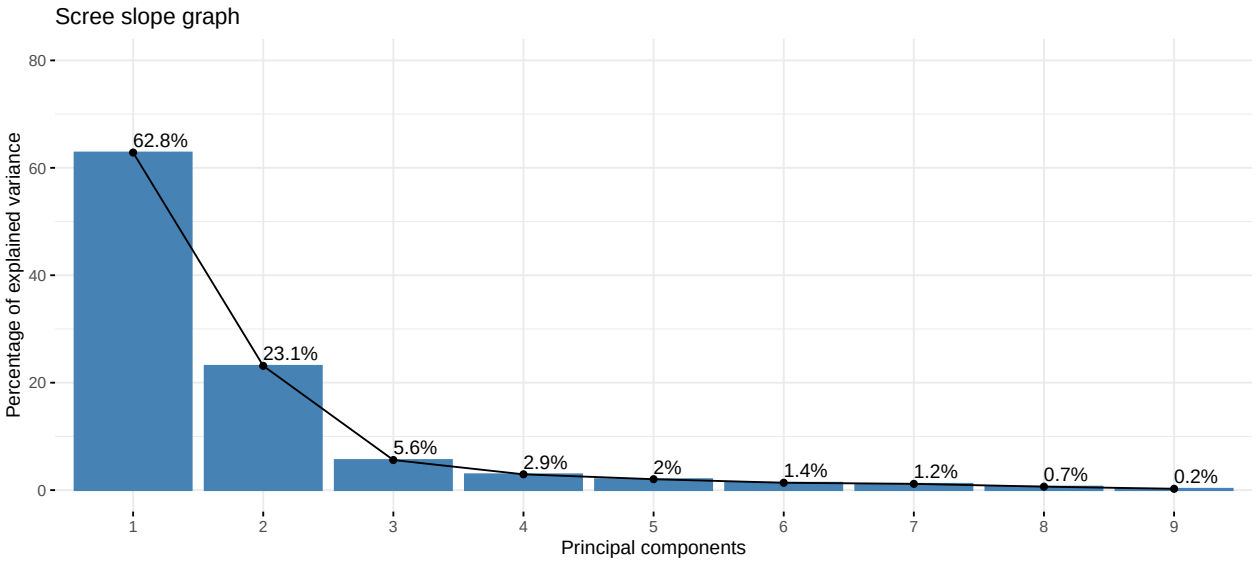
This type of analysis is relevant when you don't have a sample size too high. You can see here that results are a bit harder to understand because of the quantity of information. It might be relevant to combine this type of analysis with a clustering algorithm like k-means (for quantitative variables). I've wrote an article about the k-prototypes (for mixed data set) algorithm with R on my website.

What are the criteria that allow us to interpret the results?

Percentage of variance explained by each component

It is common to say that at least we want to project 50% of the total inertia. Beyond, it is considered as too low for a relevant interpretation. In our case, we project on a 2 dimensions subspace and we already know that the first principal component contains more than 50% of the total inertia. Let's take a graph that shows it.

```
data("mtcars")
data = mtcars[, c(1:7, 10:11)]

library(FactoMineR)
pca_results = PCA(data, graph = FALSE)

library(factoextra)
fviz_eig(pca_results, addlabels = TRUE, ylim = c(0, 80),
         main = "Scree slope graph",
         xlab = "Principal components",
         ylab = "Percentage of explained variance")
```



There are several methods for determines the number of principal components wanted. However, there is no perfect answer for this and **the best answer is the one(s) you understand**.

Elbow method
The elbow method consists as, from a curve of the variance explained function, the number d which represents the "elbow" (thus the break between the arm and the forearm). In the example from above, it seems that this number is 2 or 3. However, because we don't work with high dimensional data, we prefer to use $d = 2$ because it allows us to make more intuitive graphs.

Kaiser's rule
Only the axes whose inertia is greater than the average inertia $\frac{Ig}{p}$ are retained. In a normalized PCA, this means selecting the axes whose eigenvalues are greater than 1 (since $Ig = p$).

Contribution of individuals/variables to principal components R allows us to have lot of information about our projection. Let's see it in more details!

```
data("mtcars")
data = mtcars[, c(1:7, 10:11)]

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

```
#contribution of each variable to each component (in %)
head(pca_results$var$contrib)
```

```
##          Dim.1       Dim.2      Dim.3        Dim.4     Dim.5
## mpg  15.456509  0.07583750  4.8926384  0.003753251 10.288826
## cyl  16.204947  0.02467964  6.3663441  0.165651047  1.372170
## disp 15.788925  0.79004878  0.6123280 11.525599379 23.695953
## hp   13.474879  7.25837471  0.0296239  0.466502562  8.686679
## drat  9.722953 11.67265515  2.2486522 71.513827382  2.622001
## wt   13.948515  2.95644165 20.5874707  3.658039870  3.514959
```

Contribution of the $i-th$ individual to the $\alpha-th$ principal component

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

Contribution of the $i-th$ individual to the plan α, α'

$$CTR_{\alpha,\alpha'}(i) = \frac{\frac{1}{n}[(\Psi_i^{\alpha})^2 + (\Psi_i^{\alpha'})^2]}{\lambda_\alpha + \lambda_{\alpha'}}$$

```
#contribution of the first 6 individual to each component (in %)
head(pca_results$ind$contrib)
```

```
##                          Dim.1     Dim.2      Dim.3      Dim.4     Dim.5
## Mazda RX4          0.251629783 2.1333594 0.26707467 0.19328988 9.9599151
## Mazda RX4 Wag      0.231570089 1.4786886 0.07851057 0.08934633 7.5950275
## Datsun 710         3.016404099 0.1652496 0.28253979 0.14361798 0.1023261
## Hornet 4 Drive     0.026436765 6.0543366 0.69446402 1.15511248 1.0478477
## Hornet Sportabout 1.436392738 1.0640480 6.82679829 0.26439244 0.8735085
## Valiant            0.001403417 9.2745395 0.07993433 9.24551438 0.2784956
```

Exerices

1. Knowing that $mean(\Psi_\alpha) = 0$ and that the individual contribution of an individual to a principal component is $CTR_\alpha(i) = \frac{\frac{1}{n}(\Psi_i^{\alpha})^2}{Var(\Psi_\alpha)}$, prove that $\sum_1^n CTR_\alpha(i) = 1$.
2. Prove $\sum_1^n CTR_\alpha(i) = 1$ empirically with the iris data set for the first principal component.

Part 5 - Project

Based on the data set of your choice, do a principal component analysis. Your data set should have between 5 and 20 of quantitative variables. You have to propose a complete analysis: description of the data, some descriptive statistics and PCA. It also should have multiple (relevant) graphs. The output of the project should be a html or pdf document created with R markdown. All important results should be interpreted (a non-positive result is still a result). By looking at your document, someone who knows PCA should be able to determine if you understand the core of this course.

Questions

What is the total inertia of a cloud of point-individuals when the variables are centered reduced? Prove it from: $I_g(X) = \frac{1}{n}\sum \|y_i\|^2$

$$\begin{aligned} I_g(X) &= \frac{1}{n}\sum_1^n \|y_i\|^2 \\ &= \frac{1}{n}\sum_1^n \sum_1^p y_i^2 \\ &= \frac{1}{n}\sum_1^n \sum_1^p (y_i - \overline{y})^2 \\ &= \frac{1}{n}\sum_1^n p = \frac{np}{n} = p \end{aligned}$$

Prove that $\sum_1^n CTR_\alpha(i) = 1$

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

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

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

$$= \frac{1}{Var(\Psi_{\alpha})} \times \frac{1}{n} \sum_{1}^{n} (\Psi_i^{\alpha} - \overline{\Psi}_{\alpha})^2 = \frac{Var(\Psi_{\alpha})}{Var(\Psi_{\alpha})} = 1$$