Statistical Workshop for Entomology Lab Members

Gregoire Noel, Chloe Galland & Clement Martin

18 March 2022

Contents

1 Citation

For the beginning, one little citation before to read the manuscript G.E.P Box: *We have a large reservoir of engineers (and scientists) with a vast background of engineering know how. They need to learn statistical methods that can tap into the knowledge. Statistics used as a catalyst to engineering creation will, I believe, always result in the fastest and most economical progress.*

2 Install R

What you can do if you do not have RStudio sofware on your laptop ?

First of all, you need to download the source code of R on the following link : https://cran.r-project.org/

Please, be carefull to download the software related to your exploitation system (Windows, Linux, Mac)! Then, install it.

Next, you need to download RStudio on the following link : https://www.rstudio.com/products/rstudio/download/#download

Take also the free and adequate version related to your exploitation system (Mac, Linux or Windows). Then, also install it. Launch RStudio and use default parameters.

If you already have Rstudio on your computer and you use Windows as exploitation system, you could type following instructions in the consol part to get the latest version of the sofware (here for the workshop, v.3.6.2.). This is to be sure that everybody use the same version and avoid some version issues:

```r
##########
# installing/loading the package:
if(!require(installr)) {
install.packages("installr"); require(installr)} #load / install+load installr

# using the package:
updateR() # this will start the updating process of your R
#installation.  It will check for newer versions, and if one
#is available, will guide you through the decisions you'd
#need to make.
#########
```

If you have already Rstudio on Mac or Linux system, you could follow this link : https://bioinfo.umassmed.edu/bootstrappers/bootstrappers-courses/courses/rCourse/Additional_Resources/Updating_R.html#updating-on-mac-and-ubuntu

If you have any problem to install and launch R and/or RStudio or update the software, do not hesitate to contact us.

3 Objectives of the workshop

For all entomology lab members, we targeted some objectives that we hope to reach at the end of the workshop.

1. **Understand** R coding environment
2. **Adapt** the code for your own research analysis
3. **Interpret** the major outcomes/informations from your analysis
4. **Plot** figures displaying the targeted informations

Be also conscient that we are not statisticians ! Most of the part is still in progress. So, remain critical about this manuscript, we do not give you the truth for your statistical research but strong basis (and especially the code) to find your trends and be robust to handle a discussion with the sceptics.

If you could give some feedbacks on the manuscript and the lecture, it will be great to improve it. It could be very profitable for us and for the next student cohort that should have this workshop.

4 Before using R

4.1 How to encode your data ?

There are a lot of ways to encode your measured data when you carry out your experiment. Most of the case you encode your data on an Excel file, but what is the most sufficient way to not loose much time on R manipulation and more largely for any database creation ?

The general rule : **your_observations_are_rows** AND **your_variables_are_columns**

You have to always prepare the most decomposed dataset. A measure is a measure ! So it is better to avoid encoding means for example because you will lose vthe information related to the variability of your dataset.

4.2 Working with .csv file

Most of the time all data to implement into R software are stocked in .csv () file. To make a .csv file you just need to save your excel dataset as a .csv file.

5 Introduction of R environment

As rookies in data management, you could ask several questions: why is it interesting to use R software ? What can it bring to me for my personal purposes or my career as researcher or bioengineer ?

Before to practice, we will try to convince you that R code is not useless by some rapid state points:

1. Academic popularity and industrial asset

R is very popular in academia kingdom and therefore, when moving to the industry, control R coding system could be a real asset for your future job. That relies big pool of people able to code in R and make predictions for various research goals such as quality control in agro-chemical company or automated survey for political campaign.

2. Statistical analysis

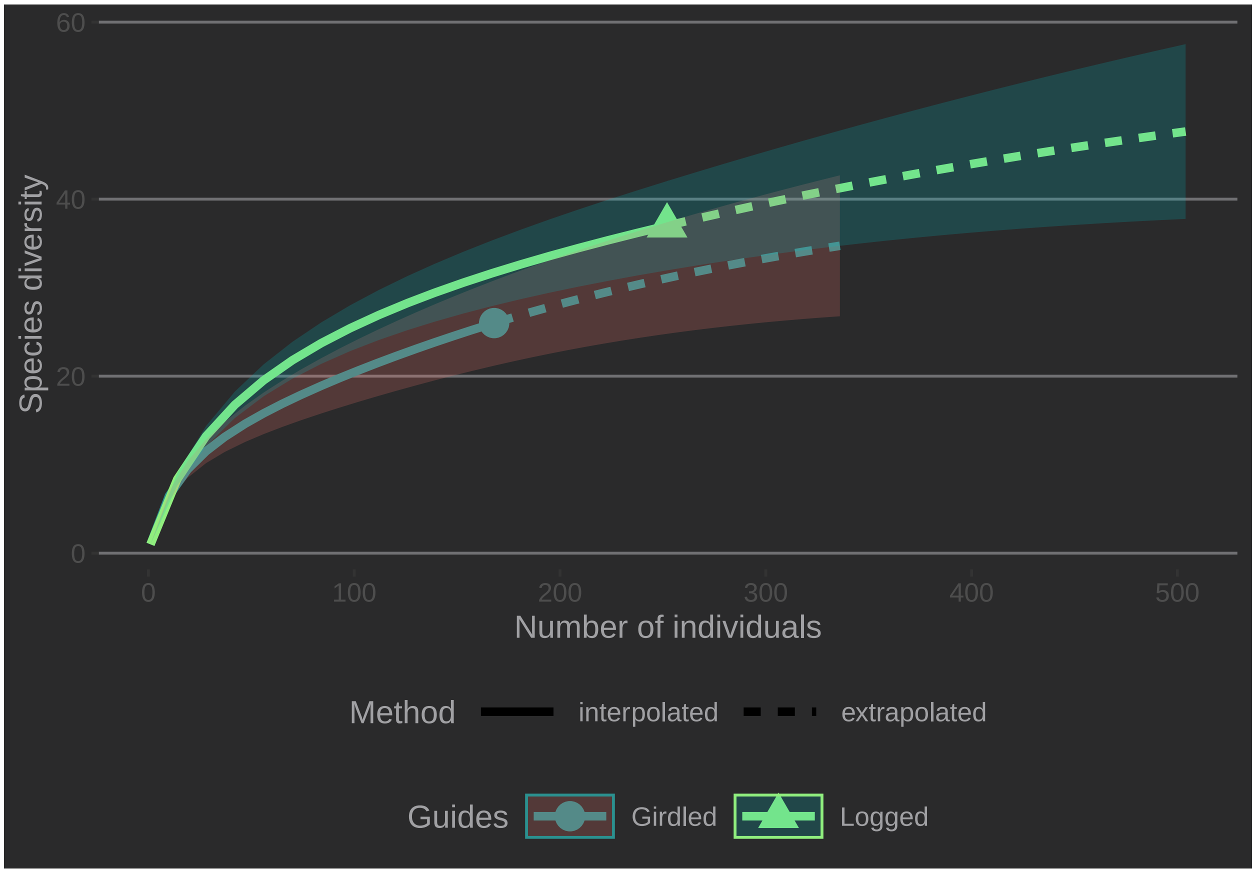
R coding is system is pretty interesting to analyze a dataset because the system proposes a wide number of useful statistical tools, many of them have been painstakingly tested.

3. Data vizualisation.

Most of the based-data graphics are usually built with Excel. Nevertheless, you do not have a lot of possibilities to change your geometrics features. Moreover Excel plots are usually not very aesthetic in our opinion. With R, you have a large panel to fine-tune your plots especially with some latest packages such as ggplot2. Every graphics features settled in command lines with the possibility to export your graphics in vectorial plots and finish the fine-tuning in Inkscape for example.

For example, look at this beauty :

```
## Warning in theme_hc(bgcolor = "darkunica"): ‘bgcolor‘ is deprecated. Use ‘style‘
## instead.
```



Pretty dark isn't it ?

This plots comes from iNEXT package for comparing alpha diversity between two spider communities.

4. Specificity & Transversality

R is cross-platform (Linux, MAC, Windows) and owns a language designed especially for statistical analysis and data reconfiguration. All R libraries focus on one thing - to make data analysis easier, more approachable and detailed. Any new statistical method is first enabled through R libraries. Moreover, members of the R community are very active and supporting and they have a great knowledge of statistics as well as programming. This all gives R a special edge, making it a perfect choice for data science projects.

5. Avalaibility

R is free and open source software. R is distributed under GNU license which allows :

- Download the source code
- Modify the source code to your heart's content
- Distribute the modified source code and even change money for it, but you must distribute the modified source code the original GNU license

This license means that R will always be available, will always be open source, and can grow organically without any constraints.

So convinced ? No maybe, so let's go for your first steps and moves in the R world and take the red pill (Matrix, 1999) !

6 First manipulation on R software

6.1 Some keywords

- **directory** = the actual repertoire
- **console** = window allowing the interaction between the user.
- > = the command invitation and provide the answer via R console for a command
- **R object** = variable containing an information
- **attribute** = information on the variable (e.g. structure, nature)
- **function** = R object defined by R code section compiling operation irrespective to the reste of the program. All functions are always following by *()*
- **operator** = native function on R whose the name is composed by special character
- **workspace** = working environment containing all the current objects into the RAM (.RData) for a working session
- **history** = working background with all the executed commands from the beginning of the working session

- **R session** = set of produced objects and executed commands from the beginning of the working session
- **packages** = depending on the specifi needs for analysis of your data, it is like a supplementary toolbox of your basic R environment
- **R script** = folder "script_name.R" containing your successive commands

6.2 Used Interface : RStudio

There are several intefaces to use R coding system, here in the workshop, we will use RStudio. There are four part displaying by the interface (Fig. 1). On the left top, you have the script part where you write your code and make importation, transformation and exportation of your datasets/results. On left bottom, you have the Console part where your code is compiled and plot the results. On right top, you have the Environment part where all your data, objects, lists, vectors and values are stored in RAM memory of your computer. In this part, you have also History tab where you could refind some functions or command that you could forget. On right bottom, you have several tabs that encompasse your file directory, your computed Plot flow, your packages activation, the Help and the Viewer part.

6.3 R features description

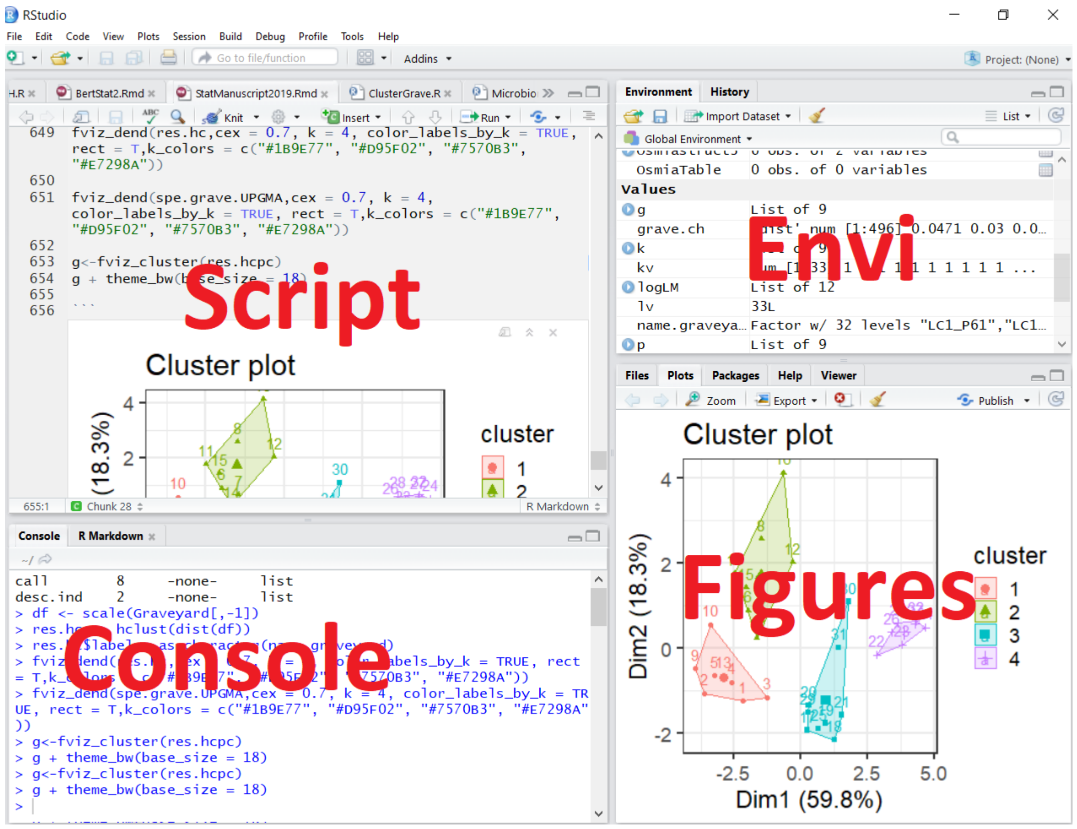
The most important thing to remind is that data (to analyze) constitute an **object** in R and other features such as functions and operators are used to serve to create, manipulate and analyze your data **object**.

Any typed character,numeric or symbols succession is an instruction or a command line. To comment your instruction you could type an "#", but be carefull to the hard return. To process your instructions flow from your script to the Console, you press to ENTER (hard return) touch and CTRL touch at the same time (on Windows and Linux) or CMD touch (on MAC).

Furthermore, R system is designed by the **object**. Everything is stored in the objects : data, function, analysis result… What you can do with an R object is :

1. To describe the object to check format, number and nature of its elements
2. To modify it in order to adapt for an analysis
3. To use it, for example with a statistical test

Be carefull R make distinction between Major and Minor letters (sensitive case) !



```
fviz_dend(res.hc,cex = 0.7, k = 4, color_labels_by_k = TRUE,
rect = T,k_colors = c("#1B9E77", "#D95F02", "#7570B3",
"#E7298A"))

fviz_dend(spe.grave.UPGMA,cex = 0.7, k = 4,
color_labels_by_k = TRUE, rect = T,k_colors = c("#1B9E77",
"#D95F02", "#7570B3", "#E7298A"))

g<-fviz_cluster(res.hcpc)
g + theme_bw(b[illegible]se_size = 18)

```
```

```
OsmiaTable        0 obs. of 0 variables
Values
g                 List of 9
grave.ch          [illegible]ist' num [1:496] 0.0471 0.03 0.0…
k                 [illegible] 9
kv                [illegible]um [1[illegible]33] 1 [illegible] 1[illegible]1 1 1 1 1 1 ...
logLM             List of 12
lv                33L
name.graveya…     Factor w/ 32 levels "LC1_P61","LC1…
p                 List of 9
```

```
call          8      -none-     list
desc.ind      2      -none-     list
> df <- scale(Graveyard[,-1])
> res.h[illegible] hclust(dist(df))
> res.[illegible]$label[illegible]as[illegible](n[illegible]gr[illegible]ve[illegible]d)
> fviz[illegible]dend([illegible]s.[illegible],[illegible]ex[illegible] = [illegible]abels_by_k = TRUE, rect
= T,k_[illegible]lors[illegible] c[illegible]#[illegible]9E[illegible]7",[illegible]#[illegible]5F[illegible]2"[illegible]7570B3", "#E7298A"))
> fviz_dend(spe.grave.UPGMA,cex = 0.7, k = 4, color_labels_by_k = TR
UE, rect = T,k_colors = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A"
))
> g<-fviz_cluster(res.hcpc)
> g + theme_bw(base_size = 18)
> g<-fviz_cluster(res.hcpc)
> g + theme_bw(base_size = 18)
> 
```

Figure 1: RStudio
```

Functions are also R object family. The function is a box that contains no data but code lines (instruction) and allow to act on the data. Function name is followed by "()" and what you input in this are the arguments.

For example:

```
four<-4 #Value 4 is stocked in the object "four"
one<-1 #Value 1 is stocked in the object "one"
sum (four,one) #Function "sum" where object four and one are the arguments
```

```
## [1] 5
```

In this example, the result "5" is preceded by "[1]" which means order of the element displayed.

Function could also compute random number according to the normal distribution (mean=0;sd=1). In the example below, 20 is the first argument which process 20 number of observation.

```
rnorm(20)
```

```
##  [1]  0.02254538 -0.40976803  1.16015058  0.10915097  0.91357841  0.80661910
##  [7] -0.59509529  1.34430700 -1.02094208  0.27607968  1.25514479 -0.50639462
## [13]  0.76019936  0.41229112  0.45420116  0.39412516 -0.89813582 -0.64580589
## [19] -0.29702108 -0.92832378
```

Conclusion, you have 20 results computed by the function *rnorm()* which are showed.

Also, a pretty nice thing on R, when you do not understand all the specificities of a function object, is that you have a well documented HELP system. For example, to understand all the subtilities of your *rnorm()* function, you could type *?rnorm()* and then browse the documentation of the function.

```
?rnorm()
```

The operators are specific attributes in R which permit also to act on data object (Table 1).

Table 1: Non exhaustive list of operator and special attribute

Operator.in.R	R.Translation
Arithmetic operator	
+	Sum
-	Substraction
*	Multiplication
/	Division
Logical operator	
&	AND
|	OR
Comparative operator	
<	Lower
<=	Lower or equal
>	Upper
>=	Upper or equal
==	Equal
!=	Different
Assignation	

Operator.in.R	R.Translation
=	Object assignation
<-	Object assignation (right to left)
Special attribute	
NA	Missing value
Inf	Infinity
NULL	Null object
TRUE	“True” logical argument
FALSE	“False” logical argument

##First step in R

6.3.1 Choosing your working directory

First of all, you can set your working directory for your script by “setwd()”" function. The argument is a string character offering the path. Be aware that path structure is not the same between Windows or Linux or Mac exploitation system. There are differences in / and "" character. Here,we use the structure according to Windows exploitation system.

```r
setwd("C://Users/Abeille/Documents") #In character you could change for your own script
```

We also presente there other usefull functions to manage your working environment

```r
getwd() # return to the absolute path which point to the current directory
```

```
## [1] "C:/Users/Abeille/OneDrive/Assistanat/Statworkshop/2021"
```

```r
dir() # list the content of the current directory
```

```
##  [1] "code.docx"
##  [2] "Intro_FirstStepR_DescriptiveSTAT_26052021.mp4"
##  [3] "Statistical analysis applyied in Entomolgy studies.pptx"
##  [4] "StatManuscript2020 (1) (1).docx"
##  [5] "StatManuscript2021.html"
##  [6] "StatManuscript2021.log"
##  [7] "StatManuscript2021.Rmd"
##  [8] "StatManuscript2021.tex"
##  [9] "StatManuscript2021_files"
## [10] "StatManuscript2021_Final.docx"
## [11] "StatManuscript2021_Final.pdf"
## [12] "StatManuscript2021_Test.docx"
## [13] "StatManuscript2021_Test.html"
## [14] "StatManuscript2021_Test.log"
## [15] "StatManuscript2021_Test.Rmd"
## [16] "StatManuscript2021_Test.tex"
## [17] "survival analysis.Rmd"
## [18] "test.png"
## [19] "Univariate_and_Multivariate_Analysis26052021.mp4"
```

```
ls() # list the current objects into the RAM
```

```
## [1] "four"     "g"        "g7"       "one"      "Operator" "out"      "spider"
```

```
rm() # remove the environment of the listed objects in brackets
```

6.3.2 Load your libraries

After setting up the working directory, next step is to load all your R packages that you would use to analyze your data. The R packages are coding extensions from R library. Some are installed (such as "stat", "graphics"...) by default while most of them need to be loaded for special purposes. To check the available package on your RStudio version, you could load the tab Packages in the bottom right of yourRStudio interface (Figure 1). If you do not have some packages you could install them by the function *install.package("Package_Name").*

When the package is loaded, you have to import it in the program. It allows to mention to R that you want to use this package during the present analysis of your result. To do so, you only have to write library following by the package name between brackets.

library(Package_Name).

All functions of interest are now loaded Nfor your data analysis.

For the workshop we will use the following packages:

```
### Libraries Installation
install.packages("ggplot2")#Creating Graphics
install.packages("dplyr")#Data manipulation
install.packages("ggpubr")#Creating Graphics
install.packages("plyr")#Data manipulation
install.packages("png")#Export Graphics into .png format
install.packages("lme4")# Functions for fitting and analyzing mixed models
install.packages("lmerTest")# Tests in Linear Mixed Effects Models
install.packages("ade4")# Multivariate Analysis
install.packages("vegan")# Multivariate Analysis
install.packages("FactoMineR")# Multivariate Analysis
install.packages("factoextra")#Multivariate Analysis
install.packages("wesanderson")# Color Palette for R
install.packages("multcomp")# Multiple Comparisons Using R
install.packages("corrplot")# Plot Correlations Graphics
install.packages("Rmisc")#access to function summarySE
install.packages("car")# Stat functions
install.packages("dunn.test")# Non parametric test Post-hoc of Dunn
install.packages("gridExtra")# Graphics
install.packages("ggsignif")#Graphics
install.packages("ecodist")
install.packages("BiodiversityR")
install.packages("ellipse")
install.packages("party")
install.packages ("randomForest")
install.packages ("caret") #PLSDA

#Load Libraries
library(ggplot2)
library(dplyr)
library(ggpubr)
library(plyr)
library(png)
library(lme4)
library(lmerTest)
library(ade4)
library(vegan)
library(FactoMineR)
library(factoextra)
library(wesanderson)
library(multcomp)
library(corrplot)
library(Rmisc)
library(car)
library(dunn.test)
library(gridExtra)
library(ggsignif)
library(ecodist)
library(BiodiversityR)
library(ellipse)
library(party)
library(randomForest)
library(caret)
#install.packages("DescTools v0.99.19")
#install.packages("multcomp")
```

6.4 Creating data object

There are different data object classes with their related function to generate it (Table 2). It is important because some function need particular input of classes object.

Table 2: Type of data object

Type	Mode.number.by.object	Function
Simple	One	NA
Vector	One	c(), vector()
Matrix	One	matrix()
Array (Tab to n dimensions)	One	array()
Data frame	Several	data.frame()
List	Several	list()
Factor	One	factor(), gl()
Table	One	table()

You could also directly create some data object in your script part.

```
x<-4 #simple object

height<-c(120,150,160) #Vector with three numerical elements
height
```

```
## [1] 120 150 160
```

```
bee_race<-c("mellifera","carnica","ligustica","adamsonii") #Vector with four
#character elements
bee_race
```

```
## [1] "mellifera" "carnica"   "ligustica" "adamsonii"
```

```
sting_pain<-rep(c("Weak", "Middle", "Strong"),c(1,3,2)) # rep() function replicate
#elements from the first vector number of time from the second vector
sting_pain
```

```
## [1] "Weak"   "Middle" "Middle" "Middle" "Strong" "Strong"
```

```
#You can also check the length of your vector by length() function

length(bee_race)
```

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

```
Number<-c(1:15)
matrice.1<-matrix(Number,ncol=3) # argument ncol indicate number iof columns of the "Number" matrix
```

In the workshop, we will mostly work with dataframe object. Dataframes could be generated from vector or matrices. Columns are systematically variables. If the vector has no names, the column carry “X” or “V” name. In general dataframe correspond to your sampling dataset where each column constitute the measured variable. **Be carefull** : column with character data is considered as a factor in R or qualitative variable. So, if you can not process your script on some column, it is probably because your data are considered as characters.

In the code, the “$” character following the dataframe object mentions the selected column that you want to change.

Here, we will use the adapted dataframe from paper of Hoc et al. (2019). The goal of the analysis was to generate linear regression between different modalities of adult density and to measure the egg’s production. You will have the dataset in your mail box.

```
#Importation and dataframe assignation
Male<-read.csv2("C://Users/Abeille/OneDrive/Bert_Analyse/Joakim_Stat/Male.csv")
str(Male) #Structure of the dataframe
```

```
## 'data.frame':    72 obs. of  6 variables:
##  $ Code       : chr  "26.1" "26.2" "26.3" "26.4" ...
##  $ Population : chr  "A" "A" "A" "A" ...
##  $ Masse      : int  50 100 150 200 250 300 350 400 450 500 ...
##  $ Nbr_male   : int  21 25 22 20 19 19 13 18 20 16 ...
##  $ Nbr_femelle: int  10 7 8 12 13 12 18 13 9 17 ...
##  $ MaleRatio  : num  0.677 0.781 0.733 0.625 0.594 ...
```

```r
#Read structure of dataframe and their different option of changes
Male$Nbr_male<-as.numeric(Male$Nbr_male)
Male$Nbr_femelle<-as.numeric(Male$Nbr_femelle)
Male$Masse<-as.numeric(Male$Masse)
str(Male)
```

```
## 'data.frame':    72 obs. of  6 variables:
##  $ Code       : chr  "26.1" "26.2" "26.3" "26.4" ...
##  $ Population : chr  "A" "A" "A" "A" ...
##  $ Masse      : num  50 100 150 200 250 300 350 400 450 500 ...
##  $ Nbr_male   : num  21 25 22 20 19 19 13 18 20 16 ...
##  $ Nbr_femelle: num  10 7 8 12 13 12 18 13 9 17 ...
##  $ MaleRatio  : num  0.677 0.781 0.733 0.625 0.594 ...
```

6.4.1 Create a function

Create your own function according to current implemented function in R is the self-concept of the software. It allows you to create your own "toolbox" for your own research. So it is pretty interesting !

```r
mynewfunction<-function(arglist) {expr}
#arglist = argument list
#expr = executed expression

bee_distance<-function(x) {
  sum(x)
}

#Creation of a distance vector
distance<-c(3.3,5.0,2.0,5.8)

#Calculation
bee_distance(distance)
```

```
## [1] 16.1
```

Having another script that you called "Thetoolbox.R" is one step further to be a good practitionner on R data analysis. Anytime, you could source your toolbox to do not always write same useful functions.

```r
source("the_path_of_your_toolbox.R")
```

6.4.2 Data importation

Most important part deals with the importation of the dataset of your sampling design. One of the best import function is *read.csv2("your_path")* that import and read your dataset of .CSV format. This function allows to link your working directory to a file on your computer. It means that you will be able to save your R code but also all graphics that you will create immediately in the referenced file. You can also read other data format from your computer like Excel or Text document with their related function *read.table()* or *read.xls()*. However, these functions are not linked to a folder which means that to save your code and your graphics, you will need to find the pathway at each time on your computer.

```
Densi_Egg<- read.csv2("C://Users/Abeille/OneDrive/Bert_Analyse/DensiouefLastGraph.csv",
                      dec=",")
#Densi_Egg is the data frame object that contain all sampled data

class(Densi_Egg) #Check data object type with class() function
```

```
## [1] "data.frame"
```

```
str(Densi_Egg) #check the structure of your data object
```

```
## 'data.frame':    29 obs. of  3 variables:
##  $ Density : int  500 500 500 2500 2500 2500 4500 4500 4500 6500 ...
##  $ SexRatio: chr  "MD" "MD" "MD" "MD" ...
##  $ Mass    : num  0.33 0.281 0.324 1.669 1.661 ...
```

6.4.3 Data manipulation

You can also extract specific elements, lines, columns or cells from your dataframe by using the function: *dataframe_name[line_number,column_number]*.You can also assign a name to each extracted vectors or matrix which could be used during your analyses. The function allowing it is: **Name**<-Dataframe_name*[line_number,column_number]*. To see what you have extracted, you only have to write the name of the extracted items. What can also be interesting is to know the class and the length of the different vector that you extract. To now the class (integer, character, dataframe.) use: **class**(assign-name). To now the length of the vector, which means the number of elements of your extracted vector, you have to use: **length**(assign-name). another interesting function to have the basic information of your datafram is the function **str**(dataframe). If some vector are not well classified (a vector is identified as character but is numeric), you can also change the class of a vector by using : **dataframe$column1<-as.factor(dataframe-vector1)** (factor can be replaced by as.integer, as.character)

```
#Extract column density
Col1<-Densi_Egg[,1] #same as Densi_Egg$Density
Col1
```

```
##  [1]  500  500  500 2500 2500 2500 4500 4500 4500 6500 6500 6500 8500 8500 8500
## [16]  500  500  500 2500 2500 2500 4500 4500 4500 6500 6500 6500 8500 8500
```

```
#Object Class adn dimensions of Col1
class(Col1) #Integer
```

```
## [1] "integer"
```

```
length(Col1) #Vector of 29 elements
```

```
## [1] 29
```

```
#Col1 is now a vector object

#Extract a line
Row2<-Densi_Egg[2,]
Row2
```

```
##   Density SexRatio  Mass
## 2     500       MD 0.281
```

```
class(Row2)
```

```
## [1] "data.frame"
```

```
#dataframe of 1 observation and 3 variable

#Extract a particular cell of Densi_Egg dataframe
Value4_3<-Densi_Egg[4,3] # 4th row and 3rd column
Value4_3
```

```
## [1] 1.669
```

You might be interesting in the generation of random number and to add it to your dataset. To generate a vector of random elements that have the same length of the other vectors of your dataframe use: **rnorm(vector-length)**. If you want to import a complete vector with recorded data use the previously seen function: **c(N1,N2,N3)**. Pay attention if you want to add it to you dataframe you have to assign a name to this new vector! To add it you use: **Dataframe$column__name<-assign-name-New-vector**.

You can also add some new columns or row to your dataset.

```
#Generate a vector with the same column length than Densi_Egg
ColToAdd<-rnorm(29)#cause I want 29 random elements

Densi_Egg$NewCol<-ColToAdd #We add new column with the column name of NewCol

str(Densi_Egg) #Checking ==> Yes we succeed !!
```

```
## 'data.frame':    29 obs. of  4 variables:
##  $ Density : int  500 500 500 2500 2500 2500 4500 4500 4500 6500 ...
##  $ SexRatio: chr  "MD" "MD" "MD" "MD" ...
##  $ Mass    : num  0.33 0.281 0.324 1.669 1.661 ...
##  $ NewCol  : num  -0.49 0.337 0.5 0.502 0.194 ...
```

```
#Change the name of the 4th column
colnames(Densi_Egg)[4]<-"RandomData"

#Add a row
RowToAdd<-c("500","MD",0.02,5.5)

Densi_Egg<-rbind(Densi_Egg,RowToAdd)

#good but problem in the nature of the columns
str(Densi_Egg)
```

```
## 'data.frame':    30 obs. of  4 variables:
##  $ Density   : chr  "500" "500" "500" "2500" ...
##  $ SexRatio  : chr  "MD" "MD" "MD" "MD" ...
##  $ Mass      : chr  "0.33" "0.281" "0.324" "1.669" ...
##  $ RandomData: chr  "-0.489599214823829" "0.337145776675974" "0.500286491098114" "0.502071093609129"
```

```
#So we change the nature of the columns
Densi_Egg$Density<-as.factor(Densi_Egg$Density)
Densi_Egg$Mass<-as.numeric(Densi_Egg$Mass)
Densi_Egg$RandomData<-as.numeric(Densi_Egg$RandomData)
#Checking
str(Densi_Egg)
```

```
## 'data.frame':    30 obs. of  4 variables:
##  $ Density   : Factor w/ 5 levels "2500","4500",..: 3 3 3 1 1 1 2 2 2 4 ...
##  $ SexRatio  : chr  "MD" "MD" "MD" "MD" ...
##  $ Mass      : num  0.33 0.281 0.324 1.669 1.661 ...
##  $ RandomData: num  -0.49 0.337 0.5 0.502 0.194 ...
```

7 Statistical basics

7.1 Notions & Definitions

In statistic, you analyze data recorded on what we call "the statistical population". What is a statistical population ? A statistical population is a sample, a set of individual or object, on which the study is carried on, from the studied population. We do not have the access to all the population. So what we do ? We take a sample of it and we try to infer some features of the total populations based on the sample. **Be careful** to select the correct number of samples! You have to be sure that what you observed if representative to what happen in the total population.

On what your stats are supported ? On **variables**. Variables are attributes that you observe and measure for you study/experiment (depending on your research goals). There are different kind of variables :

- **Quantitatives variables** which corresponds to number(s) and on R by a numerical/integer vector
- **Qualitative variable** or nominal variable wich corresponds to character or factor from your study (e.g. treatment, observation name…)
- **Ordinal variable** is same as qualitative variable but the variable are ordered according to observation rank

There are also constinuous and discrete variable.

##Descriptive statistics

After the inference, we should have a value (simple algebra calculation) which is called an **estimation** and which describe the properties of the population of interest. For example, which is the best line passing in scatter point plot.

Location estimator:

- Mean (cfr below chapter)
- Median (cfr below chapter)
- Mode = most frequent value

Spread estimator:

- Variance
- Standard deviation
- Standard error

- Range
- Quantile = any data proportion (10% . . .)

Association estimator: these estimators measure the linear association (the strength) between 2 variables

- Covariance
- Pearson correlation

The following code show to you a function that we use for calculate some descriptive statistics of the Densi_Egg dataset.

```r
# Following function comes from
# http://www.cookbook-r.com/Manipulating_data/Summarizing_data/


## Gives count, mean, standard deviation, standard error of the mean,
## and confidence interval (default 95%).
##   data: a data frame.
##   measurevar: the name of a column that contains the variable to be summariezed
##   groupvars: a vector containing names of columns that contain grouping variables
##   na.rm: a boolean that indicates whether to ignore NA's
##   conf.interval: the percent range of the confidence interval (default is 95%)


summarySE <- function(data=NULL, measurevar, groupvars=NULL, na.rm=FALSE,
                      conf.interval=.95, .drop=TRUE) {
  library(plyr)

  # New version of length which can handle NA's: if na.rm==T, don't count them
  length2 <- function (x, na.rm=FALSE) {
    if (na.rm) sum(!is.na(x))
    else       length(x)
  }

  # This does the summary. For each group's data frame, return a vector with
  # N, mean, and sd
  datac <- ddply(data, groupvars, .drop=.drop,
                 .fun = function(xx, col) {
                   c(N    = length2(xx[[col]], na.rm=na.rm),
                     mean = mean   (xx[[col]], na.rm=na.rm),
                     sd   = sd     (xx[[col]], na.rm=na.rm)
                   )
                 },
                 measurevar
  )

  # Rename the "mean" column
  datac <- rename(datac, c("mean" = measurevar))

  datac$se <- datac$sd / sqrt(datac$N)  # Calculate standard error of the mean

  # Confidence interval multiplier for standard error
  # Calculate t-statistic for confidence interval:
```

```
  # e.g., if conf.interval is .95, use .975 (above/below), and use df=N-1
  ciMult <- qt(conf.interval/2 + .5, datac$N-1)
  datac$ci <- datac$se * ciMult

  return(datac)
}
```

The function is written, so now we will use the function on the Densi_Egg dataframe. As you can see, we summarize descriptive statistics on quantitative mass variable egg *Mass* according to the combination of Density and SexRatio modality (Table 3).

```
Stat_egg <- summarySE(Densi_Egg, measurevar="Mass", groupvars=c("Density","SexRatio"))
#Use of summar

kable(Stat_egg, caption = "Mean of Mass egg according to the combination of Density and SexRatio modali
```

Table 3: Mean of Mass egg according to the combination of Density and SexRatio modality. N = number of observation; sd = standard deviation; se = standard error; ci = confidence interval

Density	SexRatio	N	Mass	sd	se	ci
2500	FD	3	2.159000	0.3784931	0.2185231	0.9402289
2500	MD	3	1.779000	0.1974943	0.1140234	0.4906030
4500	FD	3	4.589333	0.8626293	0.4980393	2.1428900
4500	MD	3	3.801667	0.2322893	0.1341123	0.5770387
500	FD	3	0.464000	0.0927955	0.0535755	0.2305167
500	MD	4	0.238750	0.1474571	0.0737285	0.2346371
6500	FD	3	7.554000	0.1473465	0.0850706	0.3660291
6500	MD	3	5.164333	0.6083258	0.3512171	1.5111652
8500	FD	2	9.029500	0.3047630	0.2155000	2.7381871
8500	MD	3	6.608000	0.8920824	0.5150440	2.2160555

There are other descriptive statistics functionsn (Table 4).

Table 4: Descriptive statistics functions on R

Parameter	Function
Mean	mean()
Variance	var()
Standard deviation	sd()
Median	median()
Quantiles	quantile()
Minimum	min()
Maximum	max()
Correlation	cor()
Distribution	summary()

Be carefull that variance, standard deviation, standard error and confidence interval are often mingled. Variance and standard deviation are population features which we estimate from sample. That means that we can measure them directly. Although, standard error and confidence interval are features from sample parameter in particular. That means we can not directly measure them. Standard error of the mean is the most famous and the most easy to infer but all the parameters could have a standard error.

#How to plot your graphics ?

There are different plotting systems with different packages such as *ggplot2* or *Lattice* packages. Everything is set up by the command line and all parameters could be fine-tuned. You can also save your plot as vectorial image to modify it later.

For more documentation, you can also find the url links at the end of the manuscript.

In R there are five kinds of graphics functions:

- High level function
- Low level function
- Graphics parameters *par()* function
- Graphics interaction function
- Graphics devices function

7.2 High level function

These functions are used to generate the plot, you only need one object element. There are other function to generate plots (Table 5):

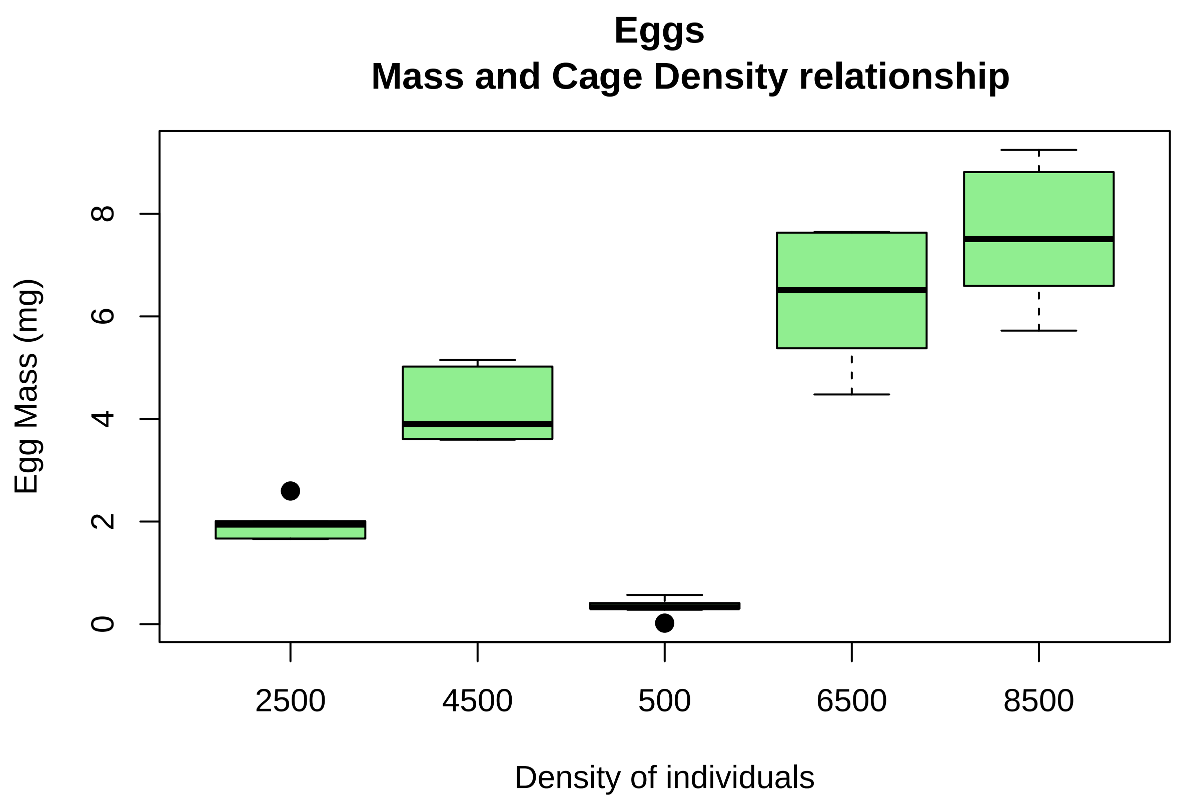
Table 5: Other High Level functions on R

Type.of.Graphics	R.function
Scatterplot	plot(numeric,numeric)
Boxplot	boxplot(factor, numeric)
Barplot	barplot(matrix, beside=FALSE)
Pie chart	pie(numeric)
Dotchart	dotchart(matrix)
Histogram	hist(numeric, breaks =sturges)
Density	plot(density(numeric))
Mosaic plot	mosaicplot(table)
Data series	mathplot()
Conditional graphics	coplot(y ~ x | a * b)
Scatterplot matrix	pairs(data.frame)

For the workshop, we only describe the scatterplot and the boxplot function.

The scatterplot is a basic graph that you can generate in R to see how your data are distributed in 2D. the operator “~” mean “in function of”.

```
#High level function plot()
#Lot of arguments to set up what you want on your graph
plot(Densi_Egg$Mass ~ Densi_Egg$Density, pch=19,cex = 1.2, col = "lightgreen",
     xlab = "Density of individuals", ylab = "Egg Mass (mg)", main = "Eggs
     Mass and Cage Density relationship")
```



7.3 Low level function

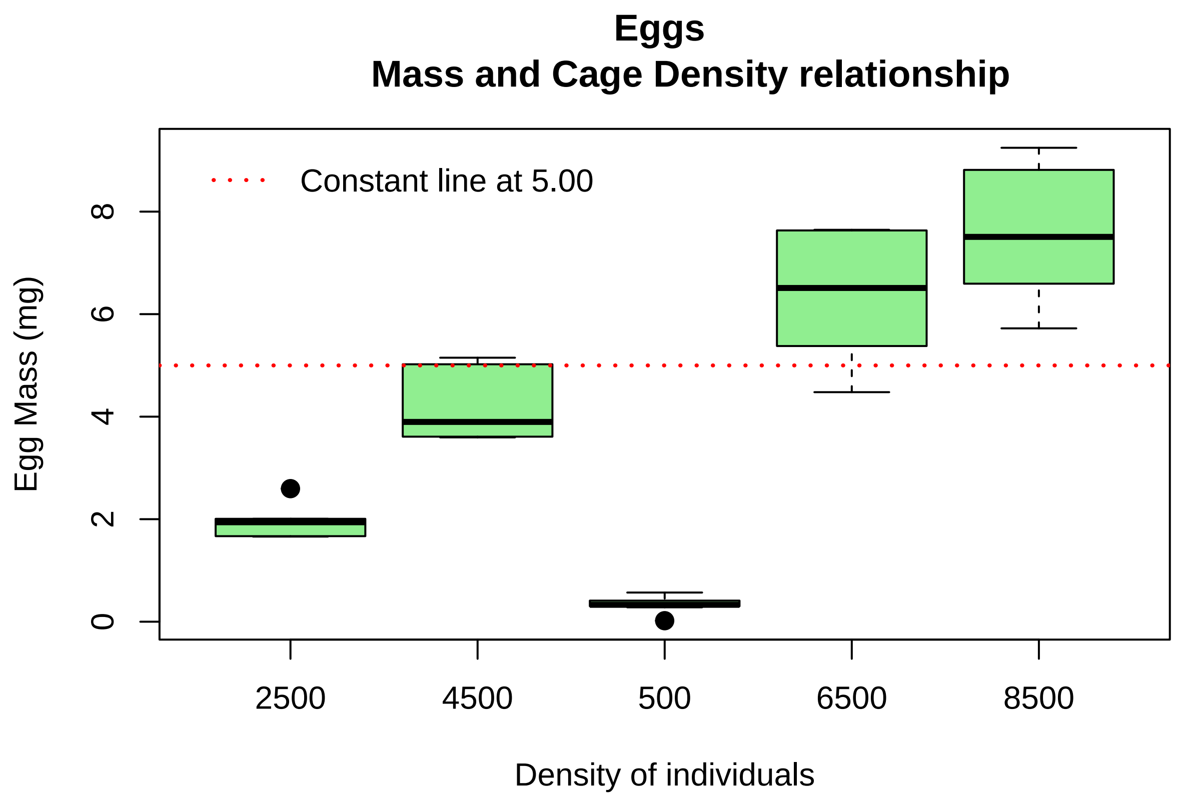
Add some other elements to the high level graphic such as a line, legend . . . (Table 6)

Table 6: Other Low Level functions on R

R.Function	Definition
points(x, y)	Add point (symbol) in xy coordinate
lines(x,y)	Lines (curves) connecting xy point
abline(v=0) ; abline(h=0); abline(a,b) ; abline(lm.object)	Straight line either vertical either horizontal or defined by a slope and an intercept or a regression line (lm object)
segments(x0,y0,x1,y1)	Segments between (x0,y0) and (x1,y1)
arrows(x0,y0,x1,y1,angle=30)	Segment with arrow tip
rect(xl,yb,xr,yt)	Rectangle with corners in (xl, yb) and (xr, yt)
polygon (x,y)	Polygone connecting xy points
text(x,y,labels)	Text in xy coordinates
pointLabel (package maptools)	Labels limiting the overlaps
legend(x,y,legend)	Legend (lot of options)
title()	Add title and legend to the axes
mtext(text, side=3, line=0)	Text in the margins
axis()	Add an axe

This is the example when you add some low level element to your graph.

```
#High level function plot()
#Lot of arguments to set up what you want on your graph
plot(Densi_Egg$Mass ~ Densi_Egg$Density, pch=19,cex = 1.2, col = "lightgreen",
     xlab = "Density of individuals", ylab = "Egg Mass (mg)", main = "Eggs
     Mass and Cage Density relationship")
#Low level function
abline(h=5, lty = 3, col = "red", lwd = 2)
legend("topleft", lty=3, col = "red", lwd = 2,legend="Constant line at 5.00",
bty="n", inset = 0.025)
```



7.4 Boxplot

This function produce a box fitted with whiskers. This plot could be usefull when you have a lot of data and you want to know their distribution.

Its formula is *boxplot(factor,numeric)*.

Example:

```
boxplot(Densi_Egg$Mass ~ Densi_Egg$Density,data=Densi_Egg,xlab = "Density of individuals (ind/m³)", ylal
```

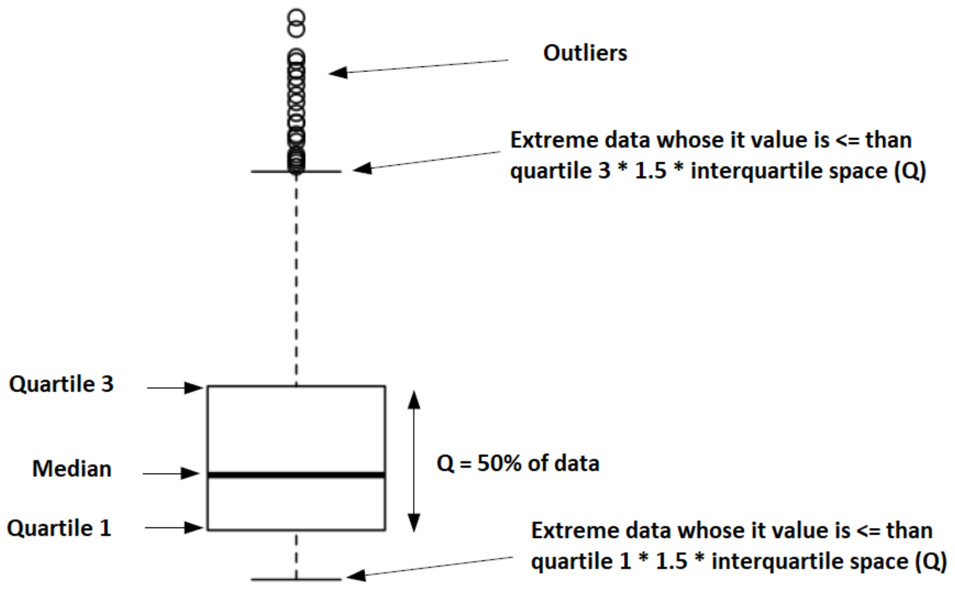

Outliers
Extreme data whose it value is <= than quartile 3 * 1.5 * interquartile space (Q)
Quartile 3
Median
Quartile 1
Q = 50% of data
Extreme data whose it value is <= than quartile 1 * 1.5 * interquartile space (Q)

Figure 2: Boxplot Architecture

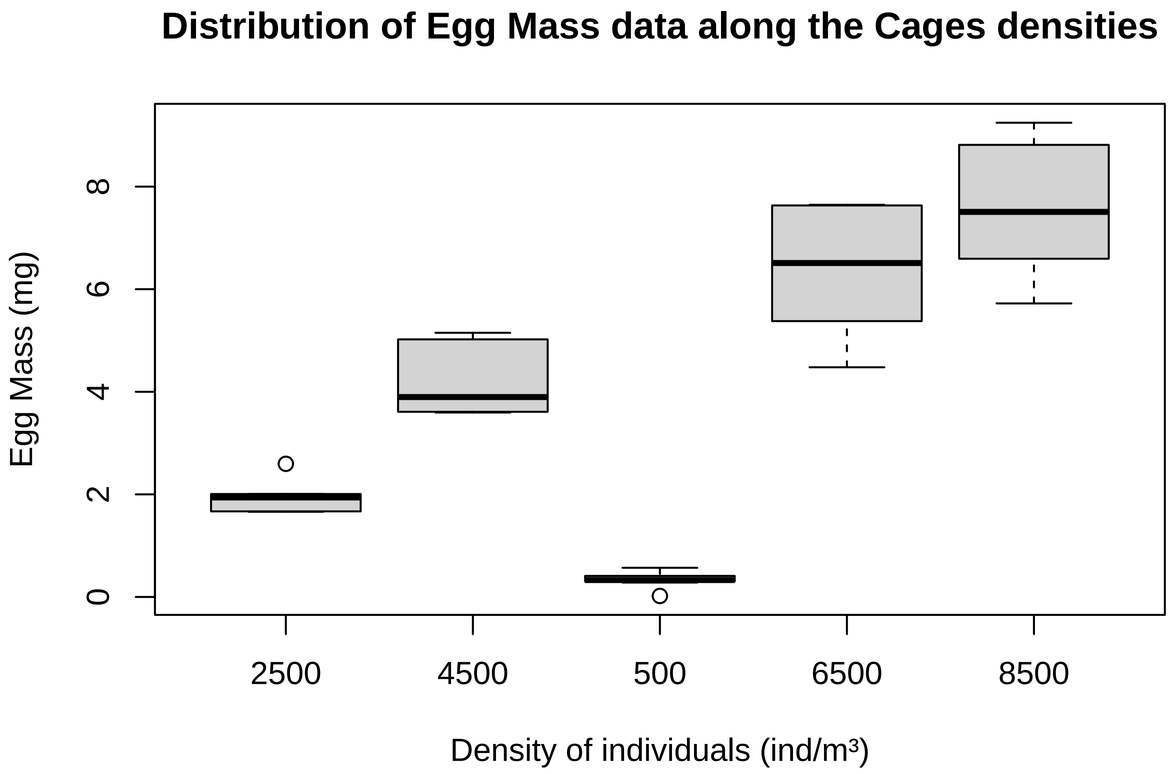


Table 7: High level functions on R for the Graphs

Type.of.Graphics	R.function
Scatterplot	plot(numeric,numeric)
Boxplot	boxplot(factor, numeric)
Barplot	barplot(matrix, beside=FALSE)
Pie chart	pie(numeric)
Dotchart	dotchart(matrix)
Histogram	hist(numeric, breaks =sturges)
Density	plot(density(numeric))
Mosaic plot	mosaicplot(table)
Data series	mathplot()
Conditional graphics	coplot(y ~ x | a * b)
Scatterplot matrix	pairs(data.frame)

7.5 Other graphics functions

To have an idea that there are other graphics functions set up by *par()* function, *identify()* function and all the devices options (exportation, resolution ...).

##ggplot2 package

This is an advanced library dedicated to the generation of various plots by common components and concepts. This package does not limit to the set of predefined graphics.

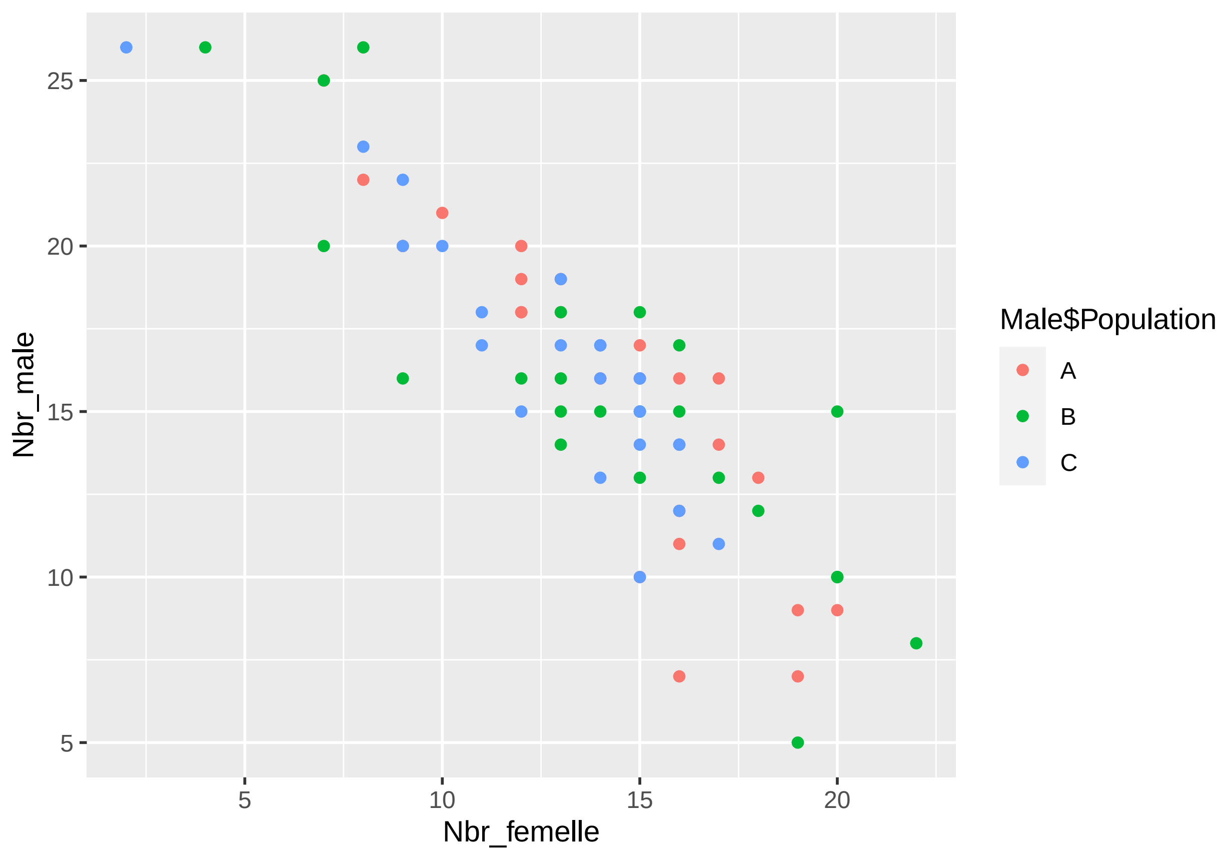
A lot of documentation about this package are set online at this url : https://ggplot2.tidyverse.org/

Principal functions are *ggplot()* and *qplot()* and the input data are in dataframe R format.

All the components in the princpal function are added with the "+" operator.

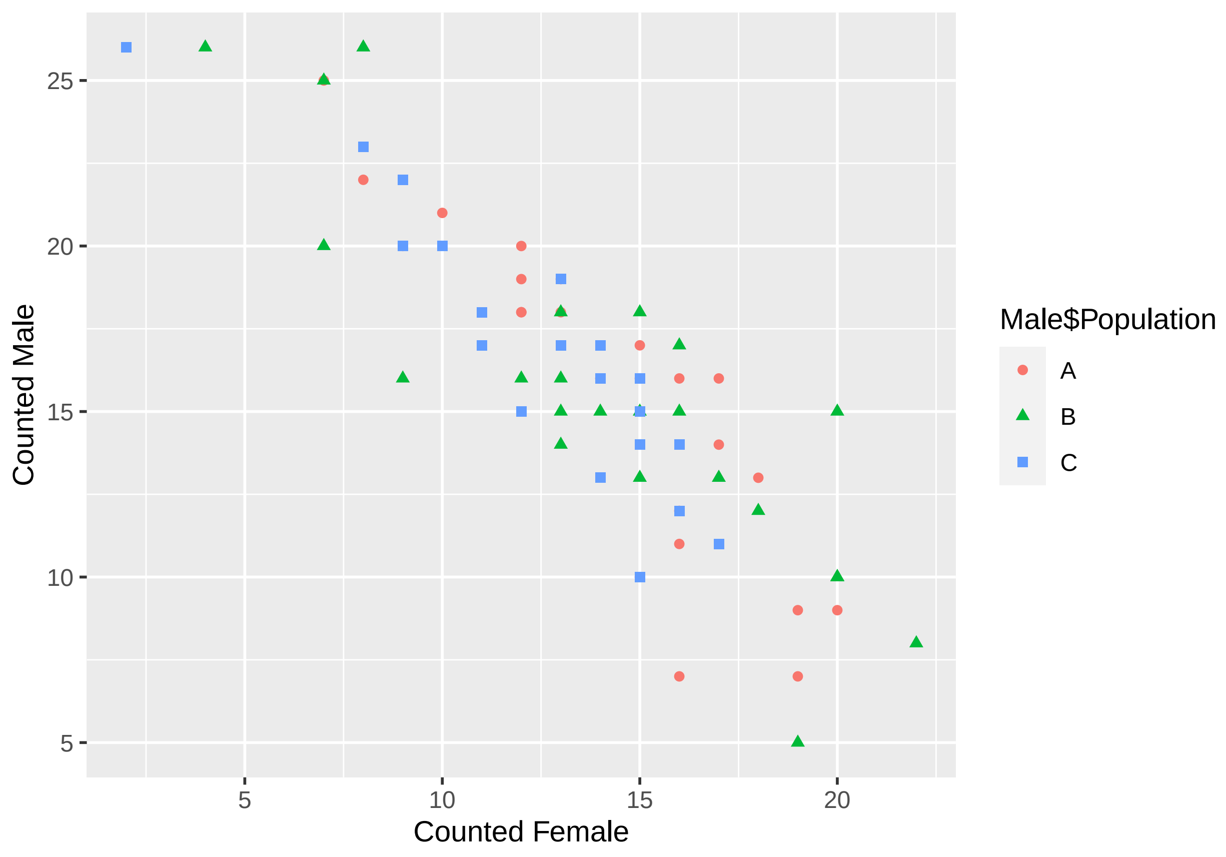
```r
Male<-read.csv2("C://Users/Abeille/OneDrive/Bert_Analyse/Joakim_Stat/Male.csv")
#str(Male)
#Structure Changes
Male$Nbr_male<-as.numeric(Male$Nbr_male)
Male$Nbr_femelle<-as.numeric(Male$Nbr_femelle)
Male$Masse<-as.numeric(Male$Masse)
#Graph ggplot2
#Scatterplot according to different population of male and #female
ggplot(Male, aes(Nbr_femelle,Nbr_male, colour=Male$Population)) +
         geom_point()
```

```
## Warning: Use of `Male$Population` is discouraged. Use `Population` instead.
```



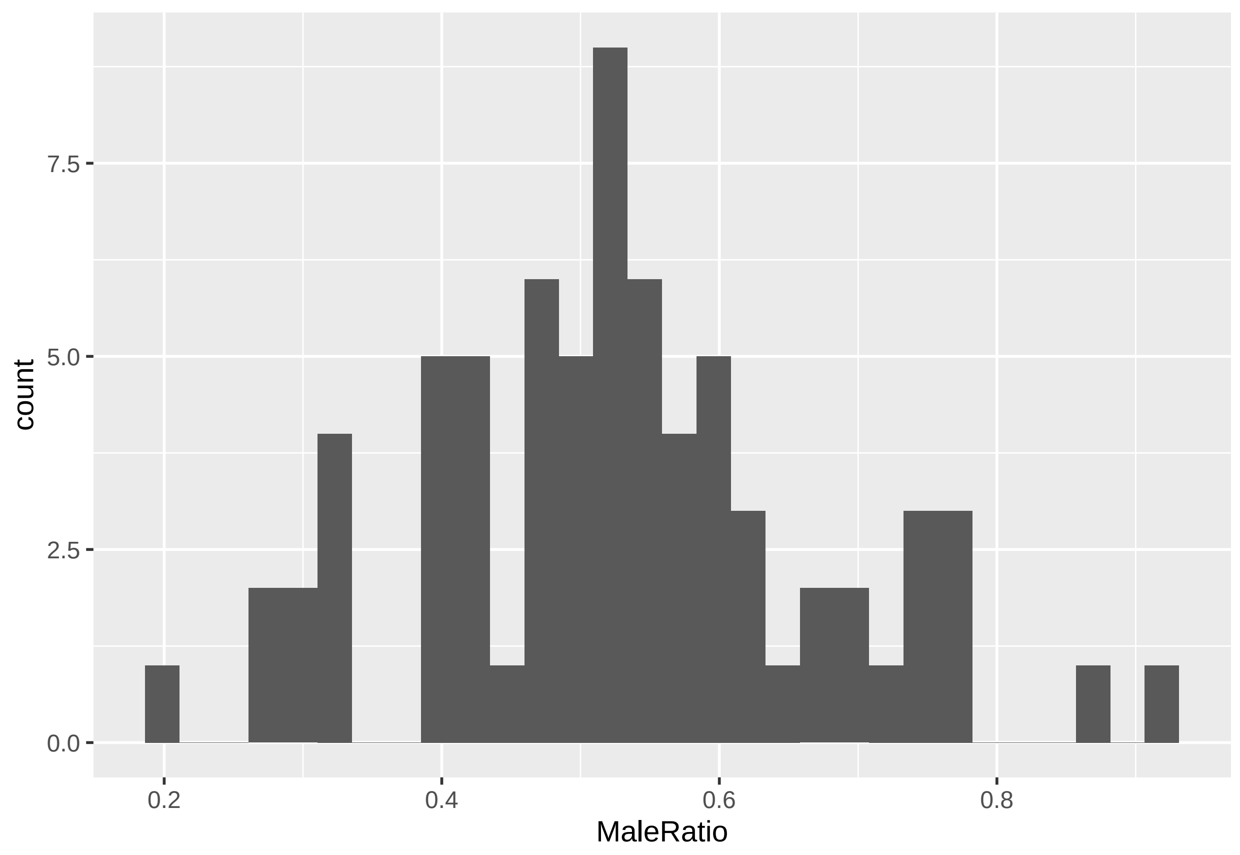
```
#For other aesthetic feature you could read the help
ggplot(Male, aes(Nbr_femelle,Nbr_male, colour=Male$Population))+ geom_point(aes(shape=Male$Population))
```

```
## Warning: Use of `Male$Population` is discouraged. Use `Population` instead.
## Use of `Male$Population` is discouraged. Use `Population` instead.
```

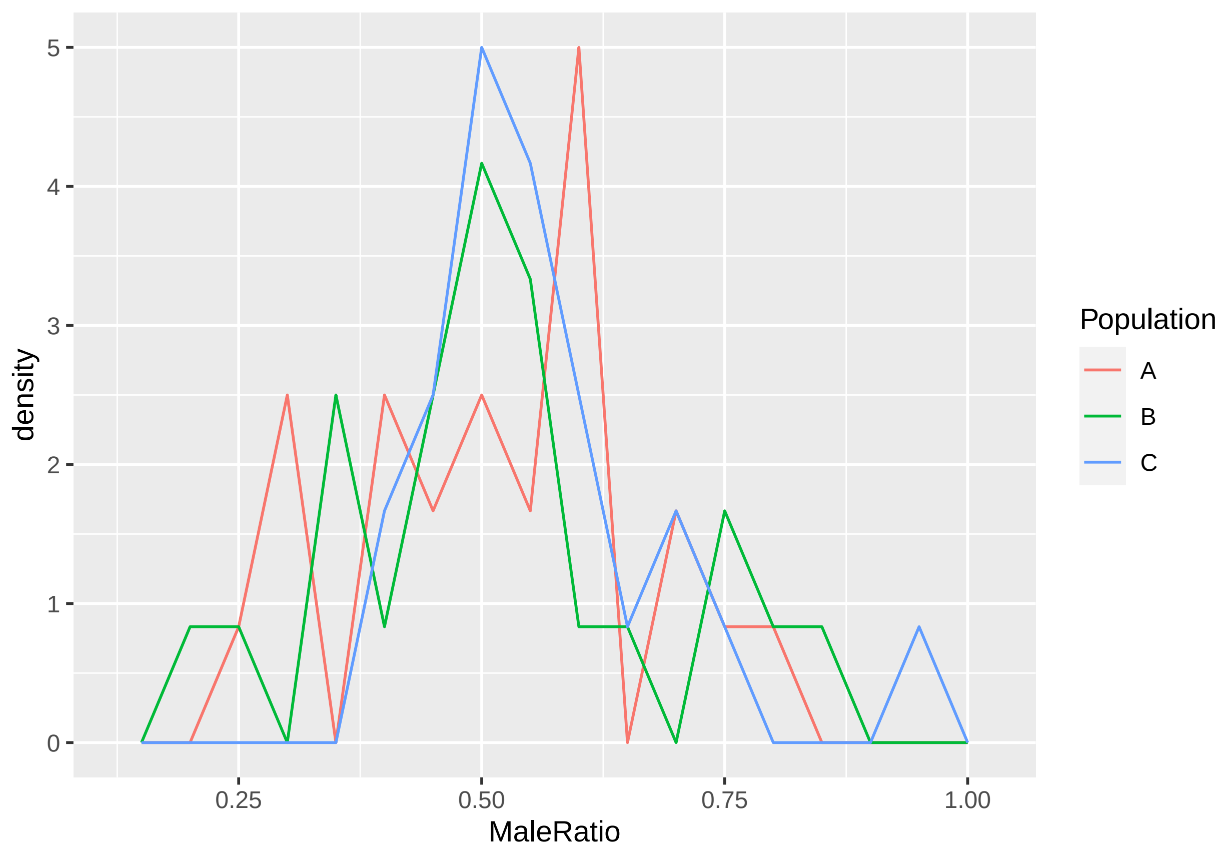


```r
#Histogram
#Visualise the distribution of a single continuous variable #by dividing the x
#axis into bins and counting the number of observations in each bin.
ggplot(Male, aes(MaleRatio))+
  geom_histogram()
```

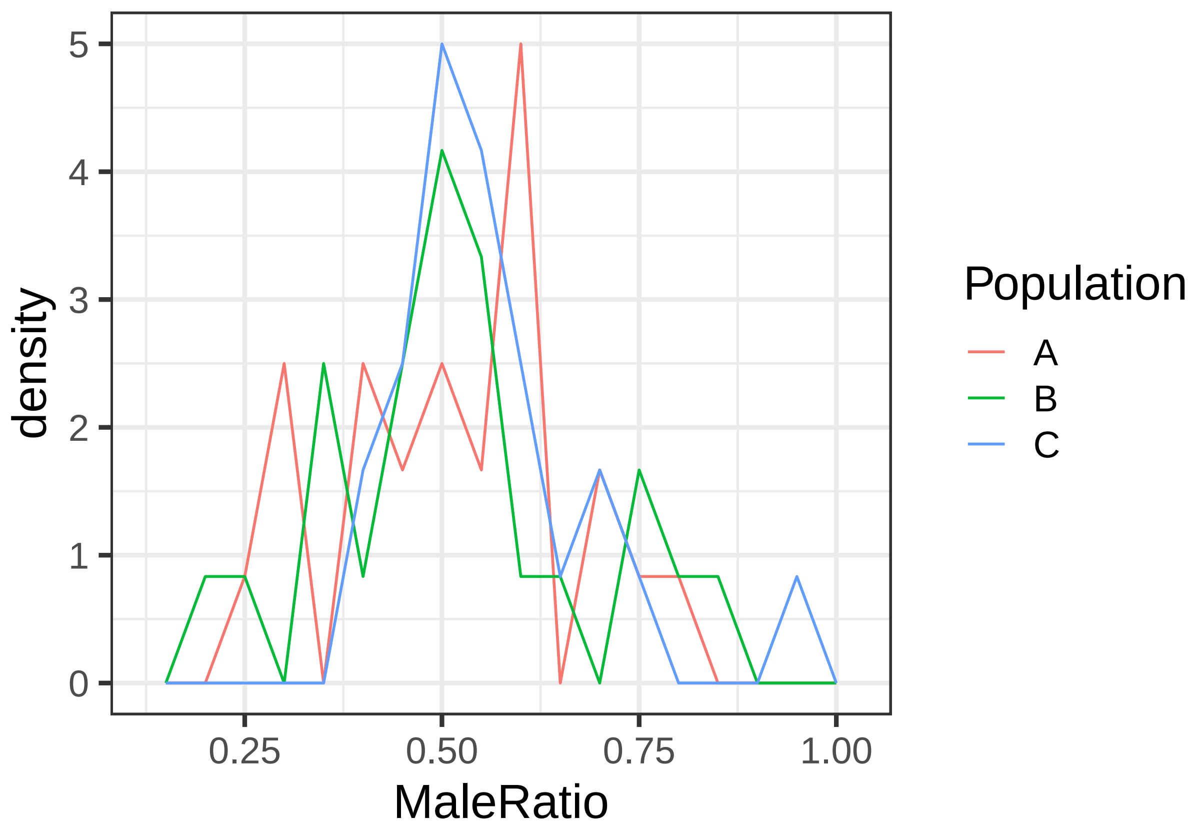
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



```
#Histogram 2.0.
ggplot(Male, aes(MaleRatio,stat(density),colour = Population))+
  geom_freqpoly(binwidth=0.05)
```



```
#Assignation and change the theme
g<-ggplot(Male, aes(MaleRatio,stat(density),colour = Population))+
  geom_freqpoly(binwidth=0.05)
g + theme_bw(base_size=18) #change in graphic theme 18
```



8 Survival Analysis

8.1 Introduction

Survival analyses are the set of static methods to investigate the time it takes for an event to occur. These analyses calculate a probability that the event will occur at a given time based on your data. The stronger your data, the stronger the probabilities will be. These analyses are widely used in medicine and sociology.

In our case, we can use it for :

- the survival time of an insect (life history trait, test for the effect of a pesticides / biopesticides . . .)
- but also for behavior tests (latency time before an insect performs a behavior)

Two main methods exist: Kaplan-Meier (univariate) and Cox porportional-Hazards model (multivariate). Here, we just speak about Cox regression because it is more complete.

With this example, we are interested in monitoring mortality in insects having received a different biopesticide (treatment) for a different time (exposure_time). Some life history traits (sex and age) were also recorded. The main questions in this case is :

- What is the probability of an individual surviving during X days/week/months . . . ?
- Does the treatment and/or exposure time have an effect on insect survival?
- Do life history traits influence survival?

8.2 The survival code

8.2.1 Packages

```r
library("survival")
library("survminer")
```

```
## Warning: le package 'survminer' a été compilé avec la version R 4.1.3
```

```
## Le chargement a nécessité le package : ggpubr
```

```
##
## Attachement du package : 'ggpubr'
```

```
## L'objet suivant est masqué depuis 'package:plyr':
##
##     mutate
```

```
##
## Attachement du package : 'survminer'
```

```
## L'objet suivant est masqué depuis 'package:survival':
##
##     myeloma
```

```r
library("lsmeans")
```

```
## Le chargement a nécessité le package : emmeans
```

```
## The 'lsmeans' package is now basically a front end for 'emmeans'.
## Users are encouraged to switch the rest of the way.
## See help('transition') for more information, including how to
## convert old 'lsmeans' objects and scripts to work with 'emmeans'.
```

```r
library("ggsci")
```

8.2.2 Opening and formating data set

```r
# Clear all objects (even invisible objects)
rm(list = ls())

# Open your data
d <- read.csv("C://Users/Abeille/OneDrive/Assistanat/data.csv", sep=";")

# Check your data
head(d, 25)
```

```
##    id treatment exposure_time sex age status death_day
## 1   3 MUCL 8115          10sec   M   6      1         9
## 2  17 MUCL 8115          10sec   M   6      1         9
## 3  19 MUCL 8115          10sec   M   6      1         7
## 4  25 MUCL 8115          10sec   M   2      1         8
## 5  30 MUCL 8115          10sec   M   6      1         7
## 6  31 MUCL 8115          10sec   M   1      1         1
## 7  32 MUCL 8115          10sec   M   3      1         5
## 8  33 MUCL 8115          10sec   M   1      1         8
## 9  34 MUCL 8115          10sec   M   5      1         4
## 10 36 MUCL 8115          10sec   M   4      1         6
## 11 38 MUCL 8115          10sec   M   1      1         4
## 12 40 MUCL 8115          10sec   M   2      1         6
## 13 41 MUCL 8115          10sec   M   1      1         8
## 14 44 MUCL 8115          10sec   M   1      1         9
## 15 47 MUCL 8115          10sec   M   5      1         3
## 16 25 MUCL 9645          10sec   M   6      1         6
## 17 26 MUCL 9645          10sec   M   6      1         5
## 18 27 MUCL 9645          10sec   M   6      1         1
## 19 29 MUCL 9645          10sec   M   3      1         5
## 20 30 MUCL 9645          10sec   M   2      1         6
## 21 33 MUCL 9645          10sec   M   3      1         7
## 22 36 MUCL 9645          10sec   M   5      1         7
## 23 38 MUCL 9645          10sec   M   4      1         5
## 24 39 MUCL 9645          10sec   M   2      1         5
## 25 41 MUCL 9645          10sec   M   5      1         3
```

```
summary(d)
```

```
##        id         treatment          exposure_time          sex
##  Min.   : 1.00   Length:1606        Length:1606        Length:1606
##  1st Qu.:12.00   Class :character   Class :character   Class :character
##  Median :24.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :24.37
##  3rd Qu.:36.00
##  Max.   :48.00
##       age            status         death_day
##  Min.   :1.000   Min.   :0.0000   Min.   : 1.000
##  1st Qu.:2.000   1st Qu.:1.0000   1st Qu.: 3.000
##  Median :3.000   Median :1.0000   Median : 6.000
##  Mean   :3.474   Mean   :0.8387   Mean   : 5.885
##  3rd Qu.:5.000   3rd Qu.:1.0000   3rd Qu.: 9.000
##  Max.   :6.000   Max.   :1.0000   Max.   :10.000
```

```
# Set as reference variable "control"
d$treatment <- as.factor(d$treatment)
d$treatment <- relevel(d$treatment, ref = "Control")
```

8.2.3 Random variables

We will choose which random variables (sex and age) have an effect on insect survival.

We use the sequential comparison of the nested sub-models (i.e. the models with and without a given covariate) and backward stepwise elimination of non-significant variables and interaction terms.

```r
# age
mod1 <- coxph(Surv(death_day,status) ~ 1, data=d)
mod2 <- coxph(Surv(death_day,status) ~ age, data=d)
anova(mod1,mod2) # p < 5% so age has an influence on survival
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(death_day, status)
##  Model 1: ~ 1
##  Model 2: ~ age
##    loglik  Chisq Df P(>|Chi|)
## 1 -9069.4
## 2 -9037.2 64.376  1 1.028e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# sex
mod3 <- coxph(Surv(death_day,status) ~ sex, data=d)
anova(mod1,mod3) # p < 5% so sex has an influence on survival
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(death_day, status)
##  Model 1: ~ 1
##  Model 2: ~ sex
##    loglik  Chisq Df P(>|Chi|)
## 1 -9069.4
## 2 -9066.4 5.9858  1   0.01442 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two random variables (sex and age) have an effect on insecte survival so these variables should be integrate on final model.

8.3 Predictor variables

```r


# We create a subset data with a long exposure time
d3h <- subset(d,exposure_time=="3heures")

# Treatment
mod1 <- coxph(Surv(death_day,status) ~ age + sex + treatment, data=d3h)
mod2 <- coxph(Surv(death_day,status) ~ age + sex , data=d3h)
anova(mod1,mod2) # p < 5% so treatment has an influence on survival
```

8.3.0.1 Q1: Which treatment is effective with a long time of exposure ?

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(death_day, status)
##  Model 1: ~ age + sex + treatment
##  Model 2: ~ age + sex
##    loglik  Chisq Df P(>|Chi|)
## 1 -1294.4
## 2 -1374.7 160.66  6 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# We select mod1


## Where is these differences ? Which treatment is more effective than another ?

# Post-hoc
# Method 1:
res <- pairwise_survdiff(Surv(death_day, status) ~ treatment, data = d3h)
res
```

```
##
##  Pairwise comparisons using Log-Rank test
##
## data:  d3h and treatment
##
##             Control Insecticide MUCL 15122 MUCL 1555 MUCL 6859 MUCL 8115
## Insecticide 3.6e-15 -           -          -         -         -
## MUCL 15122  0.86187 < 2e-16     -          -         -         -
## MUCL 1555   2.6e-07 1.1e-10     1.2e-06    -         -         -
## MUCL 6859   0.12586 4.6e-14     0.13681    0.00029   -         -
## MUCL 8115   0.12583 1.1e-15     0.14571    5.0e-05   0.86187   -
## MUCL 9645   0.14571 1.3e-11     0.13150    0.00025   0.86187   0.86187
##
## P value adjustment method: BH
```

```
# Method 2:
lsmeans(mod1, pairwise~treatment) # choose the model which you have selected
```

```
## $lsmeans
##  treatment   lsmean    SE  df asymp.LCL asymp.UCL
##  Control      0.658 0.146 Inf    0.3717     0.945
##  Insecticide  3.718 0.299 Inf    3.1326     4.303
##  MUCL 15122   0.624 0.287 Inf    0.0624     1.186
##  MUCL 1555    1.936 0.278 Inf    1.3910     2.480
##  MUCL 6859    1.101 0.276 Inf    0.5614     1.641
##  MUCL 8115    1.068 0.274 Inf    0.5314     1.605
##  MUCL 9645    1.157 0.292 Inf    0.5844     1.730
##
## Results are averaged over the levels of: sex
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast                 estimate    SE  df z.ratio p.value
##  Control - Insecticide     -3.0595 0.258 Inf -11.862  <.0001
##  Control - MUCL 15122       0.0346 0.249 Inf   0.139  1.0000
##  Control - MUCL 1555       -1.2771 0.234 Inf  -5.462  <.0001
##  Control - MUCL 6859       -0.4430 0.234 Inf  -1.891  0.4866
##  Control - MUCL 8115       -0.4096 0.233 Inf  -1.758  0.5769
##  Control - MUCL 9645       -0.4985 0.242 Inf  -2.059  0.3774
##  Insecticide - MUCL 15122   3.0941 0.264 Inf  11.710  <.0001
##  Insecticide - MUCL 1555    1.7824 0.236 Inf   7.545  <.0001
##  Insecticide - MUCL 6859    2.6166 0.245 Inf  10.666  <.0001
##  Insecticide - MUCL 8115    2.6499 0.246 Inf  10.792  <.0001
##  Insecticide - MUCL 9645    2.5610 0.252 Inf  10.146  <.0001
##  MUCL 15122 - MUCL 1555    -1.3117 0.240 Inf  -5.459  <.0001
##  MUCL 15122 - MUCL 6859    -0.4775 0.241 Inf  -1.984  0.4251
##  MUCL 15122 - MUCL 8115    -0.4442 0.240 Inf  -1.854  0.5115
##  MUCL 15122 - MUCL 9645    -0.5330 0.249 Inf  -2.143  0.3270
##  MUCL 1555 - MUCL 6859      0.8342 0.220 Inf   3.785  0.0029
##  MUCL 1555 - MUCL 8115      0.8675 0.219 Inf   3.962  0.0014
##  MUCL 1555 - MUCL 9645      0.7787 0.231 Inf   3.375  0.0131
##  MUCL 6859 - MUCL 8115      0.0334 0.222 Inf   0.150  1.0000
##  MUCL 6859 - MUCL 9645     -0.0555 0.233 Inf  -0.239  1.0000
##  MUCL 8115 - MUCL 9645     -0.0889 0.232 Inf  -0.383  0.9998
##
## Results are averaged over the levels of: sex
## Results are given on the log (not the response) scale.
## P value adjustment: tukey method for comparing a family of 7 estimates
```

The only treatment which is different than control is "MUCL_1555".

```
# We create a subset data with just MUCL_1555 treatment
d1555 <- subset(d,treatment=="MUCL 1555")

# Exposure time
mod1 <- coxph(Surv(death_day,status) ~ age + sex + exposure_time, data=d1555)
mod2 <- coxph(Surv(death_day,status) ~ age + sex , data=d1555)
anova(mod1,mod2) # p < 5% so treatment has an influence on survival
```

8.3.0.2 Q2: Can you have the same effective with a short time of exposure ?

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(death_day, status)
##  Model 1: ~ age + sex + exposure_time
##  Model 2: ~ age + sex
##    loglik  Chisq Df P(>|Chi|)
## 1 -968.93
## 2 -975.20 12.539  4   0.01376 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# We select mod1


## Where is these differences ? Which treatment is more effective than another ?

# Post-hoc
# Method 1:
res <- pairwise_survdiff(Surv(death_day, status) ~ exposure_time, data = d1555)
res
```

```
##
##  Pairwise comparisons using Log-Rank test
##
## data:  d1555 and exposure_time
##
##         10min  10sec  1heure 1min
## 10sec   0.9072 -      -      -
## 1heure  0.1414 0.0767 -      -
## 1min    0.1970 0.1414 0.9072 -
## 3heures 0.0476 0.0097 0.7033 0.5777
##
## P value adjustment method: BH
```

```
# Method 2:
lsmeans(mod1, pairwise~exposure_time) # choose the model which you have selected
```

```
## $lsmeans
##  exposure_time lsmean    SE  df asymp.LCL asymp.UCL
##  10min         0.1594 0.164 Inf   -0.1619     0.481
##  10sec         0.0932 0.272 Inf   -0.4398     0.626
##  1heure        0.5318 0.264 Inf    0.0140     1.050
##  1min          0.6302 0.274 Inf    0.0923     1.168
##  3heures       0.6762 0.276 Inf    0.1345     1.218
##
## Results are averaged over the levels of: sex
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast         estimate    SE  df z.ratio p.value
##  10min - 10sec      0.0662 0.221 Inf   0.299  0.9983
##  10min - 1heure    -0.3724 0.223 Inf  -1.674  0.4503
##  10min - 1min      -0.4709 0.222 Inf  -2.118  0.2122
##  10min - 3heures   -0.5169 0.221 Inf  -2.342  0.1316
##  10sec - 1heure    -0.4386 0.219 Inf  -2.001  0.2655
##  10sec - 1min      -0.5370 0.220 Inf  -2.439  0.1052
##  10sec - 3heures   -0.5831 0.217 Inf  -2.682  0.0567
##  1heure - 1min     -0.0984 0.221 Inf  -0.446  0.9918
##  1heure - 3heures  -0.1444 0.217 Inf  -0.665  0.9638
##  1min - 3heures    -0.0460 0.219 Inf  -0.210  0.9996
##
## Results are averaged over the levels of: sex
## Results are given on the log (not the response) scale.
## P value adjustment: tukey method for comparing a family of 5 estimates
```

At 1 hour and 1 minute, the effective is the same as at 3 hours.

8.3.0.3 To go further You can tested interaction between your variables

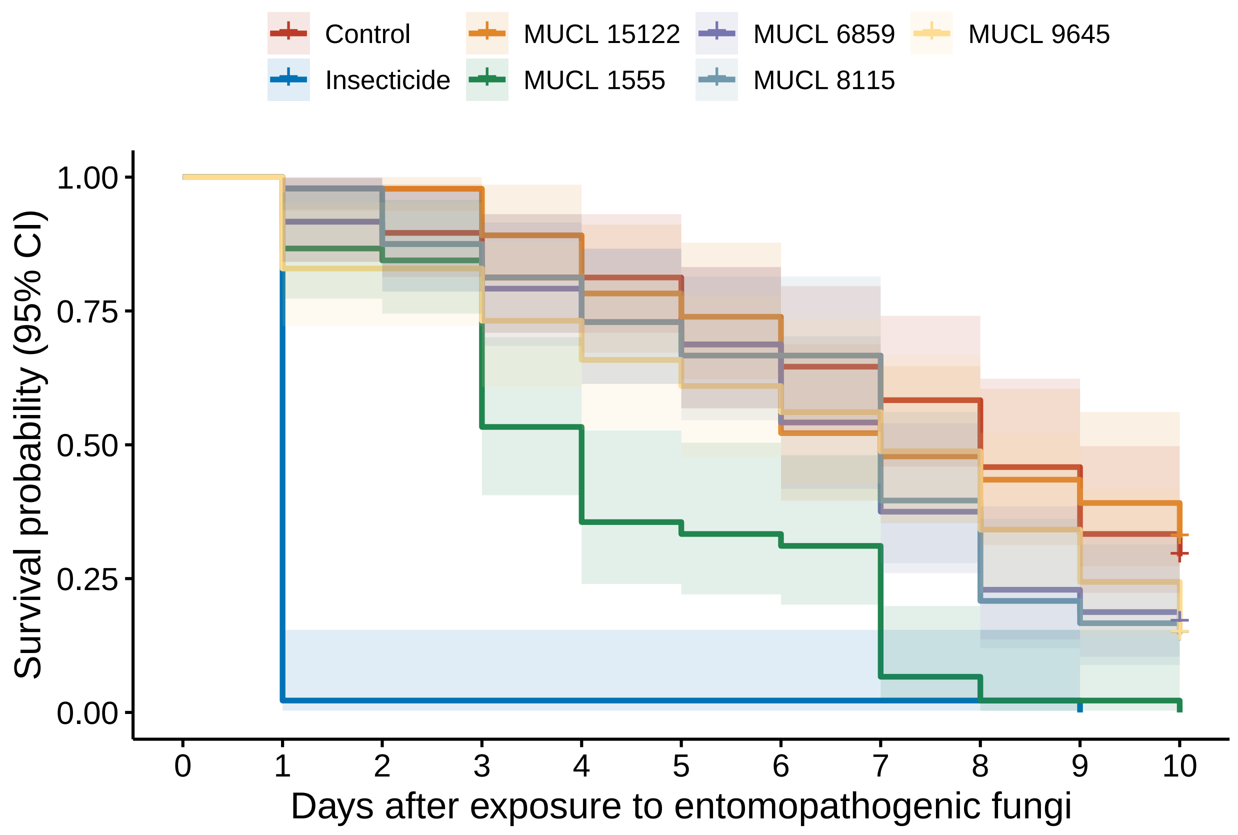
```
# age * sex
mod1 <- coxph(Surv(death_day,status) ~ age * sex, data=d)
mod2 <- coxph(Surv(death_day,status) ~ age + sex, data=d)
anova(mod1,mod2) # p > 5% so interaction between age and sex has an influence on survival
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(death_day, status)
##  Model 1: ~ age * sex
##  Model 2: ~ age + sex
##    loglik  Chisq Df P(>|Chi|)
## 1 -9034.2
## 2 -9034.2 0.0251  1    0.8742
```

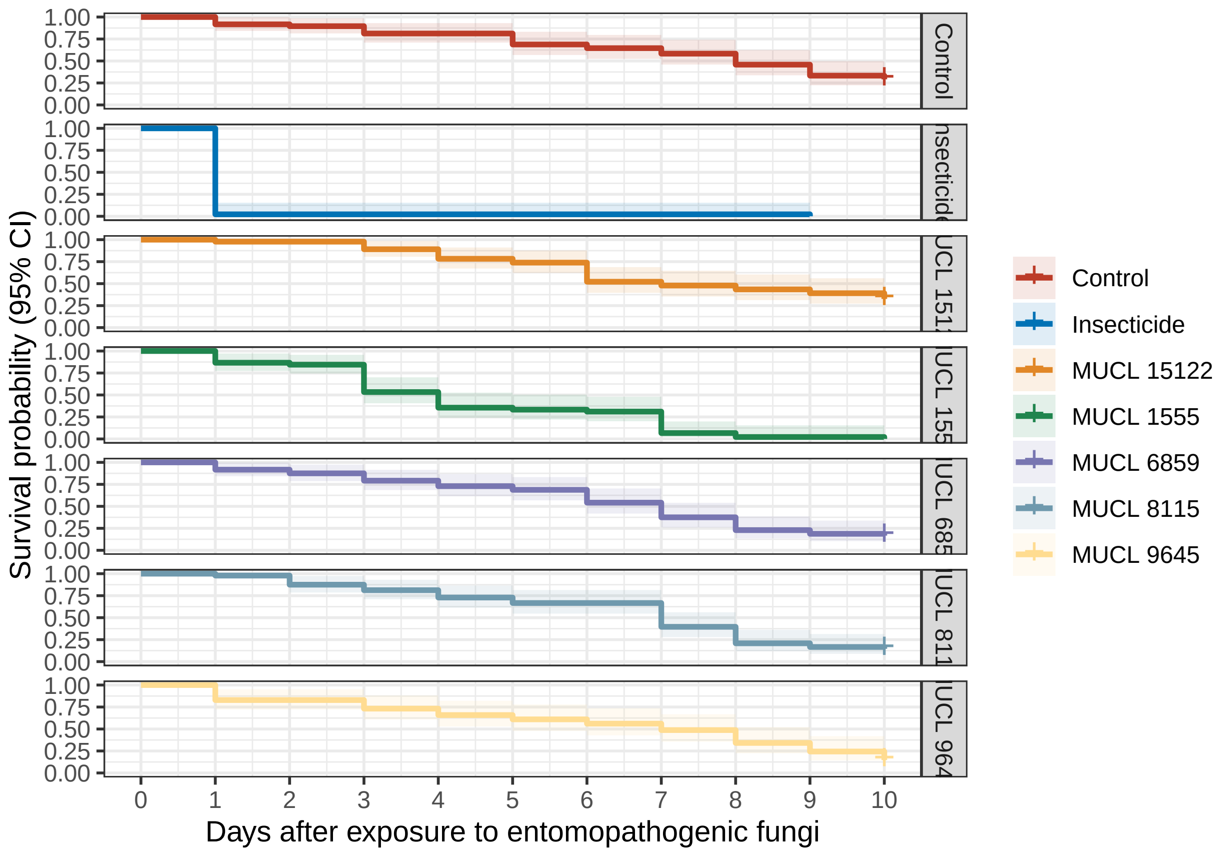
You can combine these analyses with a GLM (see Chapter . . .)

8.3.1 Graphical

```
graphique <- survfit (Surv (death_day,status) ~ treatment ,data = d3h)


plot1 <- ggsurvplot(
        graphique,                   # survfit object with calculated statistics.
        data = d3h,                  # data used to fit survival curves.
        # risk.table = TRUE,         # show risk table.
        # pval = TRUE,               # show p-value of log-rank test.
        conf.int = TRUE,             # show confidence intervals for point estimates of survival curves.
        palette = "nejm",            # change color (see "ggsci" package)
        conf.int.style="ribbon",     # style of confidence intervals
        conf.int.alpha = 0.12,       # intensity of confidence intervals
        xlim = c(0,10),              # present narrower X axis, but not affect survival estimates.
        xlab = "Days after exposure to entomopathogenic fungi",
                                     # customize X axis label.
        ylab="Survival probability (95% CI)",
                                     # customize y axis
        break.time.by = 1,           # break X axis in time intervals by 1.
        # surv.median.line = "hv",   # add the median survival pointer.
        legend.labs =
        c("Control", "Insecticide", "MUCL 15122",
        "MUCL 1555", "MUCL 6859", "MUCL 8115",
        "MUCL 9645"),                # change legend labels. Warning: verify order of your labels !
        legend.title=""
        # ggtheme = theme_minimal()   # customize plot and risk table with a theme.
)

plot1
```



```
plot2 <- plot1$plot +
  theme_bw() +
  facet_grid(treatment~ .)

plot2
```



```
# Save your graph
#ggsave("test.png", width = 11, height = 11)
```

9 Univariate analysis

These analyses are only based on one variable (discrete or continuous). For the workshop, we only work on measured variable that are continuous number not on proportion or binary measure.

9.1 Inference : Null Hypothesis Test

If I measure two variables such as the size of an hoverfly species and its proboscis length and that I observe a correlation of 0.87 (strong correlation), how can I know that there is an actual relationship between these two variables or these result is only due to random process ? That is why we estimate the **pvalue** which is the probability to obtain this result in the hypothesis that there should be no relationship between these both variables (which is called in this case **null hypothesis**).

If this probability is high ($p > 0.05$), we consider that we could have this result randomly without relation between these both variable (be carefull, it could have a relationship between our both variables but our data are insuficient to affirm it).

Therefore, we accept the null hypothesis or H0.

If this probability is low ($p < 0.05$), we consider that it is unlikely to obtain this result randomly and therefore our data are sufficient to support the observed outcome. Therefore, we reject HO (RHO) and we accept the alternative hypothesis H1.

Be careful the threshold of 0.05 is arbitrary. This is one conventionnal rule, particularly in the Entomology Science. In practice (on R), there are two common approach:

- permutation test or randomization : wich is a method by resampling your data so need a computer
- classical test based on probability distribution (normal, Poisson, Gamma . . .)

Here in the workshop, we only see the second one.

It is also possible to define probability of an expected event by the Bayes theorem, but it is also out of the scope for this workshop.

9.1.1 Probability distribution

We used graveyard dataset from Clement Martin study on chemical compound analyzed in different loamy and sandyloamy gravesoil layers. Here, in this example, we want to show that two variables, dry matter (MS) and organic carbon concentration (Corg), are distributed according to normal distribution and their correlation is null (H0). So, we computed a statistics value t=R/sqrt(1-R^2/n-2) where standard error is R and n is the number of observation. Once we got this value, we calculate the probability to get again this value or an extreme value according to Student distribution. That means we calculate the aera under the curve that is hereafter of the t value.

Example in R code:

```r
#Graveyard dataset
graveyard<-read.csv2("C://Users/Abeille/OneDrive/Assistanat/matrice de resultats-tt.csv")

#graveyards name
name.graveyard<-graveyard[,3]

#Vector creation of Corg and MS
MS<-graveyard$MS
Corg<-graveyard$Corg

#the correlation test

cor.test(MS,Corg, method = "pearson") #Pearson method for quantitative variable
```

```
##
##  Pearson's product-moment correlation
##
## data:  MS and Corg
## t = -3.3418, df = 30, p-value = 0.002241
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.7358844 -0.2103431
## sample estimates:
##        cor
## -0.5208361
```

9.2 Application conditions test

Most of the common application condition to execute a test are the normality and the homoscedasticity of the data (>1 factor).

9.2.1 Shapiro-Wilk test

H0 = data are normal ; H1 = data are not normal

```
#limon dataset
limon<-read.csv2("C://Users/Abeille/OneDrive/Assistanat/matrice-resultat-limon.csv")

#Normality test
shapiro.test(limon$P205)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  limon$P205
## W = 0.96393, p-value = 0.7332
```

The result is >0.05 so we do not reject null hypothesis that our data are normal.

```
#Normality test
shapiro.test(limon$CaO) #not normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  limon$CaO
## W = 0.80864, p-value = 0.003556
```

```
CaO<-limon$CaO
CaOLog<-log(CaO)

shapiro.test(CaOLog)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  CaOLog
## W = 0.87594, p-value = 0.03356
```

If you do not have normality on your data, you can make some mathematics transformation to return your data distribution as normal.

Transformation commonly used where x is numerical vector :

- log(x), log10(x)
- sqrt(x)
- xˆ3
- 1/x
- exp(x), 10ˆx
- xˆ2
- xˆ3, xˆ4
- rank(x)
- arcsin(x)

- rank transformation to normality

For the last transformation, we used the function to transform your data **rntransform** (see below for the code)

```r
##function ztransform

ztransform <- function (formula, data, family = gaussian)
{
  if (missing(data)) {
    if (is(formula, "formula"))
      data <- environment(formula)
    else data <- environment()
  }
  else {
    if (is(data, "gwaa.data")) {
      data <- data@phdata
    }
    else if (!is(data, "data.frame")) {
      stop("data argument should be of gwaa.data or data.frame class")
    }
  }
  if (is.character(family))
    family <- get(family, mode = "function", envir = parent.frame())
  if (is.function(family))
    family <- family()
  if (is.null(family$family)) {
    print(family)
    stop("'family' not recognized")
  }
  if (is(try(formula, silent = TRUE), "try-error")) {
    formula <- data[[as(match.call()[["formula"]], "character")]]
  }
  if (is(formula, "formula")) {
    mf <- model.frame(formula, data, na.action = na.pass,
                      drop.unused.levels = TRUE)
    mids <- complete.cases(mf)
    mf <- mf[mids, ]
    y <- model.response(mf)
    desmat <- model.matrix(formula, mf)
    lmf <- glm.fit(desmat, y, family = family)
    resid <- lmf$resid
  }
  else if (is(formula, "numeric") || is(formula, "integer") ||
           is(formula, "double")) {
    y <- formula
    mids <- (!is.na(y))
    y <- y[mids]
    resid <- y
    if (length(unique(resid)) == 1)
      stop("trait is monomorphic")
    if (length(unique(resid)) == 2)
      stop("trait is binary")
  }
```

```
    else {
      stop("formula argument must be a formula or one of (numeric, integer, double)")
    }
    y <- (resid - mean(resid))/sd(resid)
    tmeas <- as.logical(mids)
    out <- rep(NA, length(mids))
    out[tmeas] <- y
    out
  }

   ##rntransform code

  rntransform <- function (formula, data, family = gaussian)
  {
    if (is(try(formula, silent = TRUE), "try-error")) {
      if (is(data, "gwaa.data"))
        data1 <- phdata(data)
      else if (is(data, "data.frame"))
        data1 <- data
      else stop("'data' must have 'gwaa.data' or 'data.frame' class")
      formula <- data1[[as(match.call()[["formula"]], "character")]]
    }
    var <- ztransform(formula, data, family)
    out <- rank(var) - 0.5
    out[is.na(var)] <- NA
    mP <- 0.5/max(out, na.rm = T)
    out <- out/(max(out, na.rm = T) + 0.5)
    out <- qnorm(out)
    out
  }
```

But here, in the case of CaO concentration we do not have a good transformation to fit our data, that means we need to use other statistical tests which does not follow normal distribution such as non-parametric test.

9.2.2 Bartlett test

This is the test for homoscedasticity !

H0 = all k population variances are equal ; H1 = at least two variance are different

In general, heteroscedasticity has no effect on the estimation of the parameter in ANOVA and linear model tests but in certain extreme case (huge heteroscedasticity) could affect the inferences.

```
#formula bartlett.test(numeric, factor)
bartlett.test(limon$MS, limon$layer)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  limon$MS and limon$layer
## Bartlett's K-squared = 1.7845, df = 4, p-value = 0.7753
```

The same transformation as the ones used in the normality can be used to make the dataset more homoscedastic

9.2.3 Remarks on normality and homoscedasticity test on data

These both tests applied on data are usually useless because if we have a lot of data, they highlight very low deviations according to normality and homoscedasticity. ANOVA and linear model are very robust against these low or middle deviations(see after). If you have small dataset, you could miss some differences (also with normality of residuals).

So better way is to check normality of the residuals.

9.3 Parametric analysis

Follow normal distribution.

9.3.1 Student t-test

For only two populations.

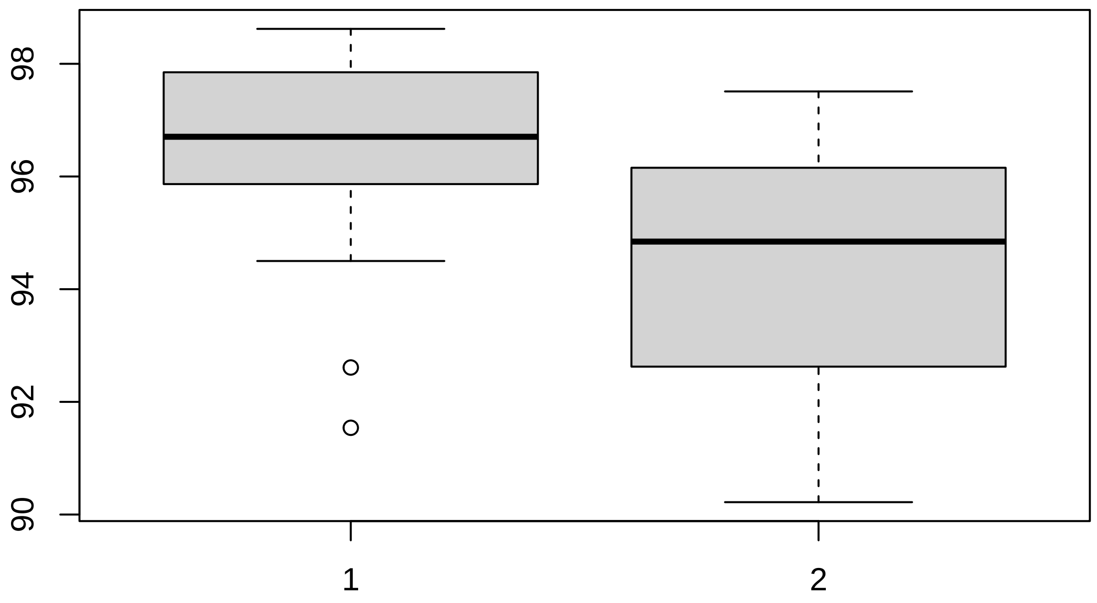
HO = means are the same ; H1 = means are different (in both way)

```r
#Create loamy and sandy dataframe by subset function
loamysoil<-graveyard[1:16,] #Select loamy data line
sandysoil<-graveyard[17:32,] #Select sandy data line

#Create vector to compare based on dry matter
loamyMS<-loamysoil$MS
sandyMS<-sandysoil$MS


t.test(loamyMS,sandyMS)
```

```
##
##  Welch Two Sample t-test
##
## data:  loamyMS and sandyMS
## t = 2.5884, df = 29.476, p-value = 0.01482
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.4182877 3.5579623
## sample estimates:
## mean of x mean of y
##  96.37625  94.38813
```

See the graphics

```r
boxplot(loamyMS,sandyMS)
```



Differences on the test but not on the graph, it is not because the graph show an overlap that this is not different because pvalue depends on sample size, population variability and effect size !

9.3.2 ANOVA (one-way)

On male dataset in Hoc et al. 2019.

For more than two populations.

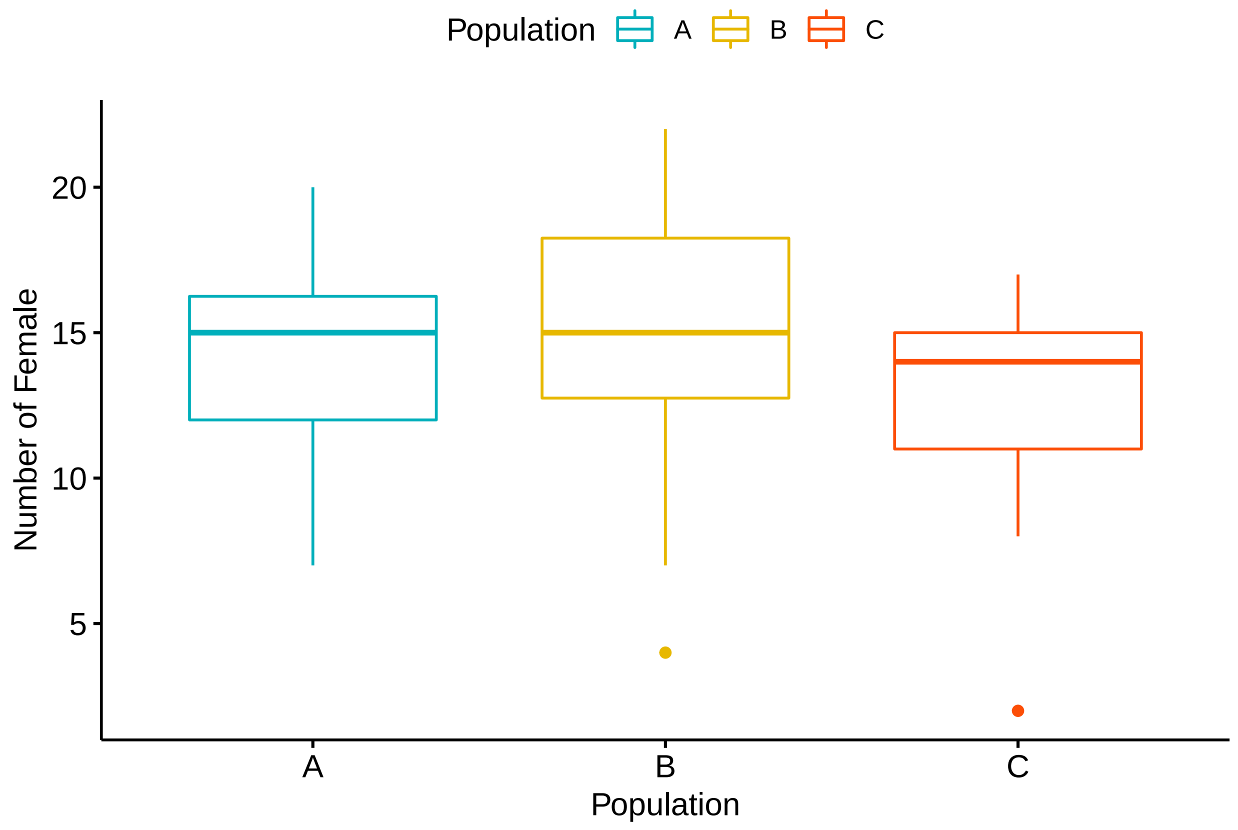
H0 = the means of the different groups are the same; H1 = At least one sample mean is not equal to the others.

```
Male<-read.csv2("C://Users/Abeille/OneDrive/Bert_Analyse/Joakim_Stat/Male.csv")
str(Male)
```

```
## 'data.frame':    72 obs. of  6 variables:
##  $ Code       : chr  "26.1" "26.2" "26.3" "26.4" ...
##  $ Population : chr  "A" "A" "A" "A" ...
##  $ Masse      : int  50 100 150 200 250 300 350 400 450 500 ...
##  $ Nbr_male   : int  21 25 22 20 19 19 13 18 20 16 ...
##  $ Nbr_femelle: int  10 7 8 12 13 12 18 13 9 17 ...
##  $ MaleRatio  : num  0.677 0.781 0.733 0.625 0.594 ...
```

```
#Change of structure
Male$Nbr_male<-as.numeric(Male$Nbr_male)
Male$Nbr_femelle<-as.numeric(Male$Nbr_femelle)
```

```r
Male$Masse<-as.numeric(Male$Masse)


#See the results

library("ggpubr")
ggboxplot(Male, x = "Population", y = "Nbr_femelle",
          color = "Population", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
          order = c("A", "B", "C"),
          ylab = "Number of Female", xlab = "Population")
```



###In the case of ANOVA non significant results

```r
#Study of mean of female of the three populations
res.aov<-aov(Male$Nbr_femelle ~ Male$Population)
summary(res.aov)
```

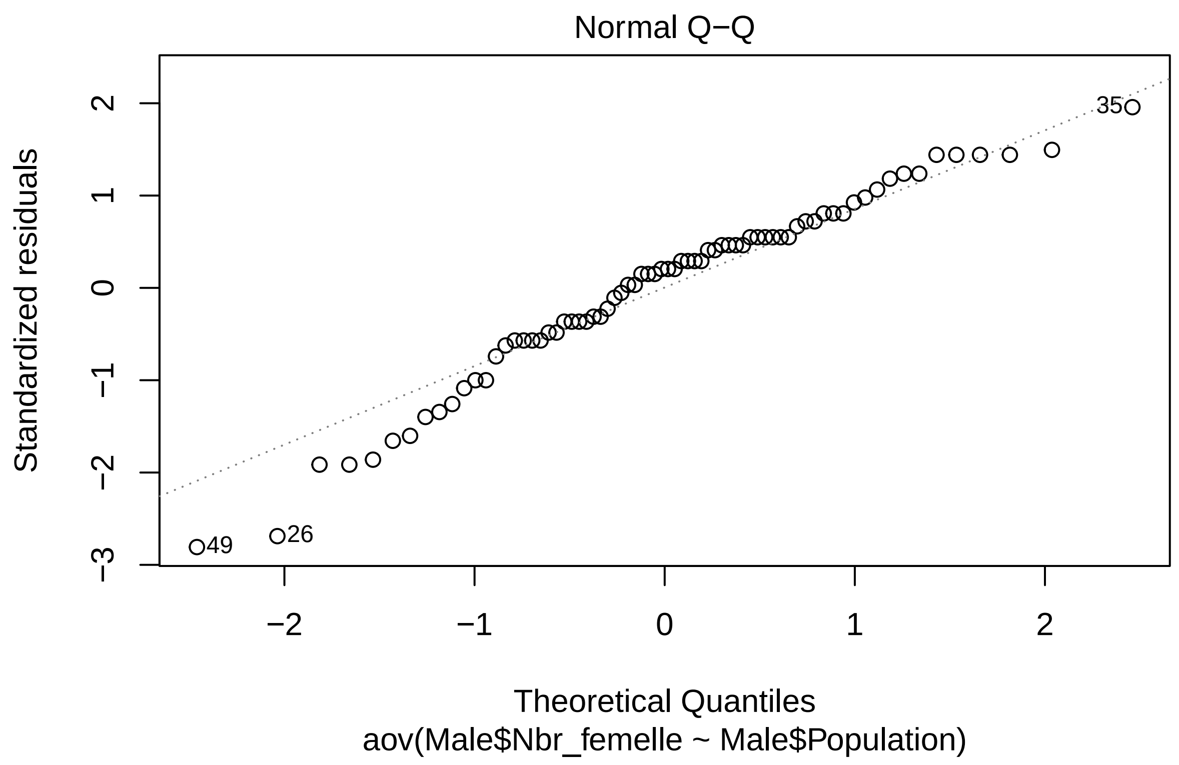
```
##                  Df Sum Sq Mean Sq F value Pr(>F)
## Male$Population   2   33.6   16.79   1.072  0.348
## Residuals        69 1080.4   15.66
```

```r
shapiro.test(res.aov$residuals) #Ok not normal
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  res.aov$residuals
## W = 0.96269, p-value = 0.03137
```

```
plot(res.aov, 2)
```



Ok no differences but the test is not the good one, cause we do not have the normality of our residuals. Alternative is to pass on non-parametric test

###Parametric post-hoc tests In the case that you will have significant results in the ANOVA, you could use post-hoc test to compare pairwise and see differences between pair of results. In parametric tests, the use of HSD Tukey test.

As you can see it below, there are no significant results on the results.

```
TukeyHSD(res.aov, "Male$Population") #Take the column "Male$Population"
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Male$Nbr_femelle ~ Male$Population)
##
## $`Male$Population`
##           diff       lwr      upr     p adj
## B-A  0.2083333 -2.527833 2.944500 0.9818346
## C-A -1.3333333 -4.069500 1.402833 0.4766388
## C-B -1.5416667 -4.277833 1.194500 0.3729845
```

9.4 Non-parametric analysis

You use it when your variables of your dataset do not follow the normal distribution. All data are distributed into abundance ranks.

9.4.1 Wilcoxon Mann-Withney test

For only two populations.

H0 = medians are the same ; H1 = medians are different

```
wilcox.test(loamyMS,sandyMS)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  loamyMS and sandyMS
## W = 196, p-value = 0.009551
## alternative hypothesis: true location shift is not equal to 0
```

9.4.2 Kruskall-Wallis test

For more than two populations.

H0 = the medians of the different groups are the same; H1 = At least one sample median is not equal to the others.

```
kruskal.test(Male$Nbr_femelle ~ Male$Population)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Male$Nbr_femelle by Male$Population
## Kruskal-Wallis chi-squared = 2.2232, df = 2, p-value = 0.329
```

Ok for this univariate data as the number of female, the test was not able to detect any significant differences among the medians of the number of female according to the three populations.

9.4.3 Non parametric post-hoc tests

In the case that you will have significant results, you could use non parametric dunn test to compare pairwise your modalities. Let's try with the code below. As you can see, there is non significant pairwise comparisons cause the KW test was non significant.

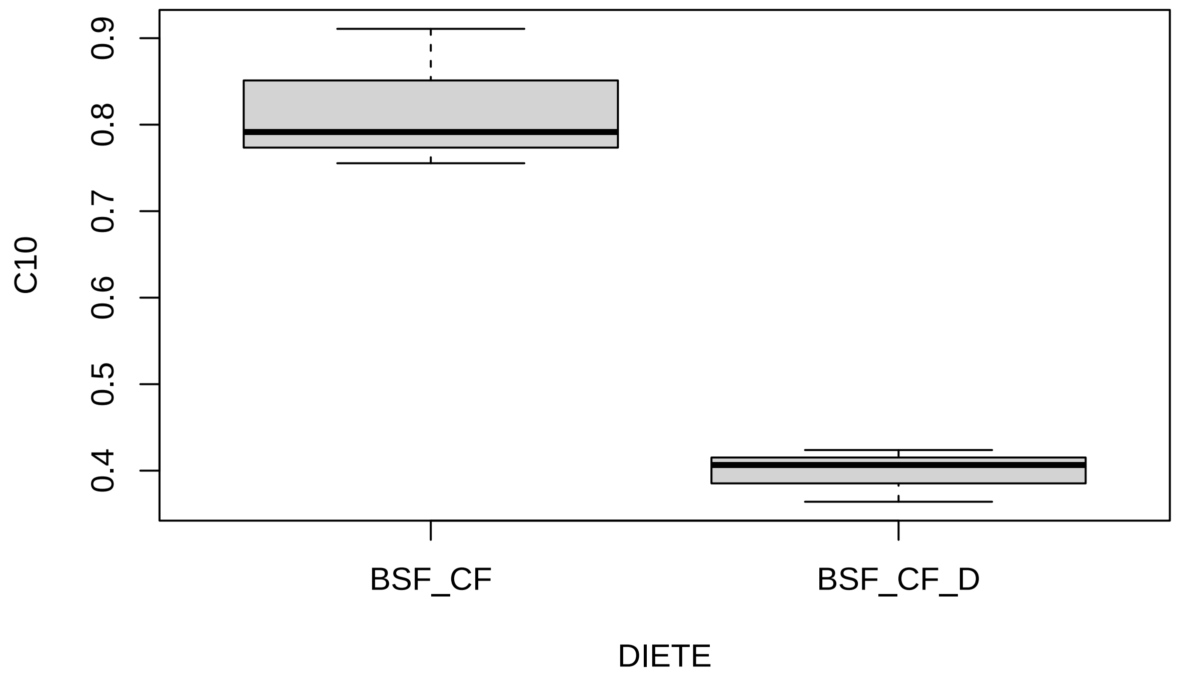
```
library(dunn.test)

dunn.test(Male$Nbr_femelle,Male$Population)
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 2.2232, df = 2, p-value = 0.33
##
##
##                              Comparison of x by group
##                                   (No adjustment)
## Col Mean-|
## Row Mean |          A          B
## ---------+----------------------
##        B |  -0.246052
##          |     0.4028
##          |
##        C |   1.150557   1.396610
##          |     0.1250     0.0813
##
## alpha = 0.05
## Reject Ho if p <= alpha/2
```

```
#Use with Bonferroni correciton, if it is suitable for your analysis
#dunn.test(Male$Nbr_femelle,Male$Population, method = "bonferroni")
```

##Automate statistical tests Sometimes, it will be usefull to automate your analysis if you have a lot of statistical comparisons and directly format them with the right table with all interesting informations. Here, we will show you an example of automated t-test between three Carbon strains from two modalities: the diet control modality and diet D modality.

```
library(stats)
#Import Table dataset
Table_data <- read.table("C://Users/Abeille/OneDrive/Assistanat/CF_CFD.csv", sep=";", head=T)
#Check difference
boxplot(C10~DIETE,data=Table_data )
```



```
#Do just one statistical test
firstresult<-t.test(Table_data$C10~Table_data$DIETE, var.equal=F, alternative = "two.sided")


library(plyr) #Package containing the function
cols_to_test <- colnames(Table_data[-1]) #Keep only Column name except the diet

#Test results generation
results <- ldply(
  cols_to_test,
  function(colname) {
    t_val = t.test(Table_data[[colname]] ~ Table_data$DIETE)$statistic
    p_val = t.test(Table_data[[colname]] ~ Table_data$DIETE)$p.value
    df = t.test(Table_data[[colname]] ~ Table_data$DIETE)$parameter
    return(data.frame(colname=colname, t_value=t_val, p_val=p_val, df = df))
  })

results
```

```
##   colname    t_value        p_val       df
## 1   C100D -18.574656 0.0003150773 3.038576
## 2    C100  16.213463 0.0027509043 2.150555
## 3     C10   8.390161 0.0062789849 2.561075
```

##Workflow example combining descriptive stats, statistical analysis with hypothesis tests and high-level graphs

###Introduction

Here we take an example of analytics workflow from Jessica Caetano (2019) master thesis. She worked on the larval development of *Episyrphus balteatus* (De Geer 1776) (Diptera : Syrphidae) according different temperature modalities which are representative of the climate warming in our temperate regions. We have done measurements on the larval weight and size and also on the aphid consumption on eight days of experiment. When the eggs hatched, we take the measure on the first instar of the larvae.

###Prepare your dataset

```
#Choose your working directory
#setwd("Path for your working directory")

#Import your dataset
Dataset<-read.csv2("C://Users/Abeille/OneDrive/Assistanat/Temperature3.csv")

#Transform dataset vector into the targeted class of object
Dataset$Temperature<-as.factor(Dataset$Temperature)
Dataset$Jour<-as.factor(Dataset$Jour)

#Have a look on your dataset, the six first lines
head(Dataset)
```

```
##   EchantillonLarve Taille Poids Pucerons Temperature Jour
## 1               L1    1.4   0.3       49          20    1
## 2               L2    1.5   0.3       50          20    1
## 3               L3    1.4   0.4       45          20    1
## 4               L4    1.1   0.2       45          20    1
## 5               L5    1.2   0.3       44          20    1
## 6               L6    1.1   0.3       48          20    1
```

###Descriptive statistics

Now you have your dataset in the right structure to work on it, let is go for the following steps and the descritpitve statistics on the dataset.

```
#Descriptive Statistics by temperature and day decomposition
#summarySE function provides the standard deviation, standard error of the mean, and a (default 95%) co
tgc <- summarySE(Dataset, measurevar="Poids", groupvars=c("Temperature","Jour"))
tgc
```

```
##    Temperature Jour  N      Poids        sd         se         ci
## 1           20    1 19  0.4631579 0.2832559 0.06498337  0.1365250
## 2           20    2 19  2.5473684 1.2536030 0.28759625  0.6042173
## 3           20    3 19  6.0526316 2.9777636 0.68314582  1.4352361
## 4           20    4 19 15.6421053 7.7121344 1.76928498  3.7171298
## 5           20    5 19 29.1368421 8.4624800 1.94142605  4.0787848
## 6           20    6 19 38.1315789 7.4048822 1.69879649  3.5690390
## 7           20    7 12 35.3416667 4.9077783 1.41675356  3.1182536
## 8           20    8  4 36.3000000 6.3377178 3.16885889 10.0847233
## 9           23    1 22  0.4909091 0.7217103 0.15386916  0.3199884
## 10          23    2 22  2.5090909 1.9417491 0.41398230  0.8609233
## 11          23    3 22  5.4590909 4.9681518 1.05921353  2.2027551
## 12          23    4 22 12.5500000 8.1929557 1.74674403  3.6325531
## 13          23    5 22 23.1136364 9.0637580 1.93239971  4.0186452
## 14          23    6 20 29.1700000 7.0532635 1.57715767  3.3010289
## 15          23    7 14 32.2000000 7.4341209 1.98685239  4.2923336
## 16          23    8  6 30.9666667 3.5364766 1.44376052  3.7113046
## 17          26    1 17  0.9588235 0.5523346 0.13396082  0.2839843
## 18          26    2 17  5.0117647 2.0907183 0.50707367  1.0749482
## 19          26    3 17 13.3235294 8.2780983 2.00773375  4.2562054
## 20          26    4 17 23.3294118 7.6575751 1.85723475  3.9371618
## 21          26    5 16 28.2937500 6.3006316 1.57515790  3.3573696
## 22          26    6 12 30.8258333 6.9446126 2.00473699  4.4123964
## 23          26    7  3 26.0666667 8.9271123 5.15407067 22.1761762
## 24          26    8  2 22.4500000 8.5559921 6.05000000 76.8725387
```

```
#Following this code you could save the results of the descriptive statistics

#write.csv2(tgc,"Weight_dataset.csv")
```

###Univariate analysis

Now you could be able to make an univariate analysis.

```
#Number of data/contengency table
table(Dataset$Temperature)
```

```
##
##  20  23  26
## 130 150 101
```

```
#ANOVA/KW test on Weight
res.aov<-aov(Dataset$Poids~Dataset$Temperature)
summary(aov(Dataset$Poids~Dataset$Temperature))
```

```
##                      Df Sum Sq Mean Sq F value Pr(>F)
## Dataset$Temperature   2    735   367.5   1.859  0.157
## Residuals           378  74716   197.7
```

```
shapiro.test(res.aov$residuals) #not normal ==> move to a non parametric test
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.aov$residuals
## W = 0.89927, p-value = 3.495e-15
```

```
kruskal.test(Dataset$Poids~Dataset$Temperature) # X2= 4.12 p-value= 0.127
```

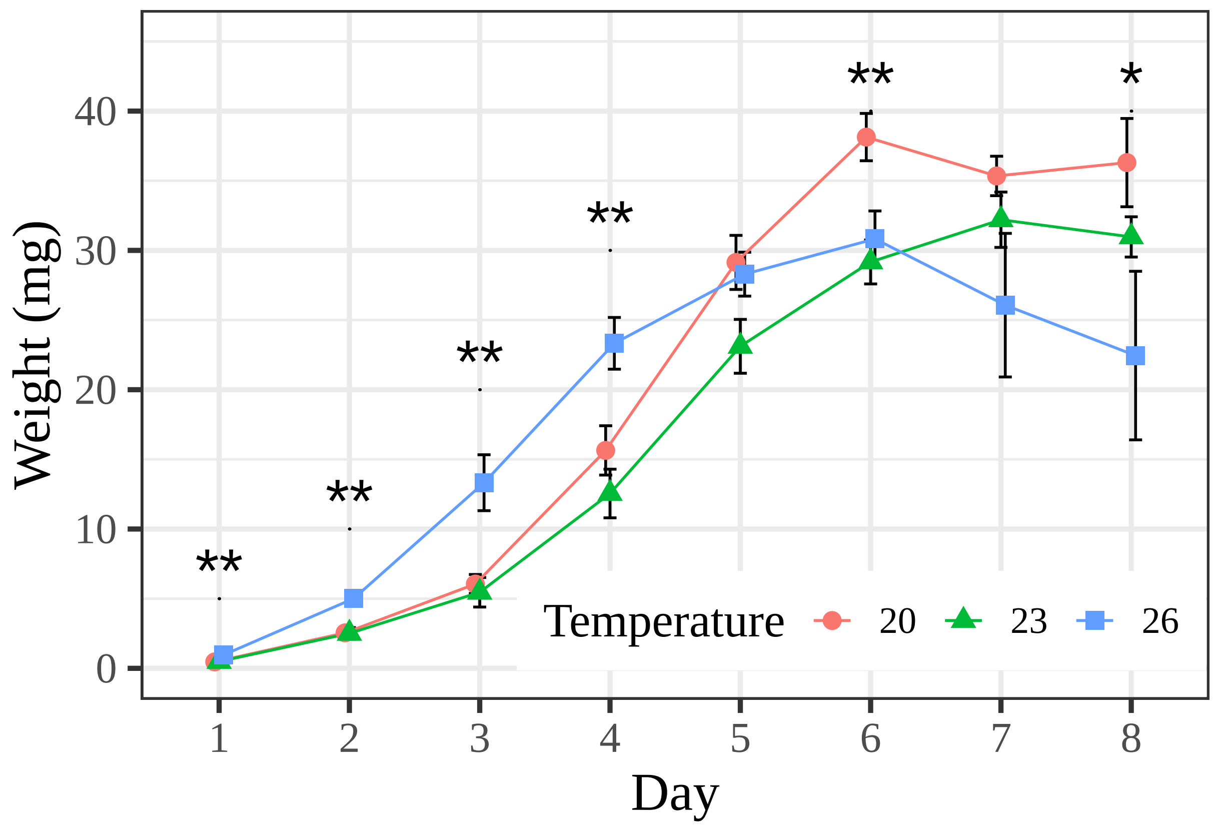
```
##
##  Kruskal-Wallis rank sum test
##
## data:  Dataset$Poids by Dataset$Temperature
## Kruskal-Wallis chi-squared = 4.1224, df = 2, p-value = 0.1273
```

```
#==> in this case a post hoc test is non suitable
```

###Plot your results using ggplot2

Now it is time to plot the result, all the options are commented, it is on the more complex that you could find, so if you want less complec graph, you just need to remove code lines.

```
pd <- position_dodge(0.1) # move them .05 to the left and right, it is to avoid the overlapping of the

#Weigth Evolution Graph
Graph_Art1<-ggplot(tgc, aes(x=Jour, y=Poids,colour=Temperature, group=Temperature)) +
  ylim(0,45)+
  geom_errorbar(aes(ymin=Poids-se, ymax=Poids+se), colour="black", width=.3, position=pd) +
  geom_line(position=pd) +
  geom_point(aes(shape = Temperature),position=pd, size=3)+
   xlab("Day") +
  ylab("Weight (mg)") +
  theme_bw(base_size = 20, base_family = "serif") +
  theme(legend.justification=c(0.99,0.05),
        legend.position=c(0.99,0.05),
        legend.direction = "horizontal",
        legend.title = element_text(size = 18),
        legend.text = element_text(size = 14))+
  geom_signif(y_position = 20, xmin = 3, xmax=3, size=0.5, textsize =
              8, annotations = "**",tip_length = 0, col="black")  +
  geom_signif(y_position = 5, xmin = 1, xmax=1, size=0.5, textsize =
              8, annotations = "**",tip_length = 0, col="black")  +
  geom_signif(y_position = 10, xmin = 2, xmax=2, size=0.5, textsize =
              8, annotations = "**",tip_length = 0, col="black")  +
  geom_signif(y_position = 30, xmin = 4, xmax=4, size=0.5, textsize =
              8, annotations = "**",tip_length = 0, col="black")  +
  geom_signif(y_position = 40, xmin = 6, xmax=6, size=0.5, textsize =
              8, annotations = "**",tip_length = 0, col="black")  +
  geom_signif(y_position = 40, xmin = 8, xmax=8, size=0.5, textsize =
              8, annotations = "*",tip_length = 0, col="black")


# Weight Graph
Graph_Art1
```



9.5 Linear regression analysis

9.5.1 Notions

Linear regression establishes a linear link or a relationship between a unique variable to explain or to estimate (Y) and one or several explicative/descriptive variables (X).

On this form :

Y = b + a1X1 + a2X2 + . . . + e

Where b, a1, a2 . . . are parameters to estimate. The set of a are the slopes of X variables and b is the intercept. Where X1, X2 . . . are explicative variables Where Y is the dependent variable Where e is the residual or the error, some parts of the variability that we could not explain with these variables

Be careful, some properties of (General) Linear Model (LM) or Linear Regression are that Y is a continuous variable as well as e and the estimation method is based on sum of squares. For Generalized Linear Model (GLM), Y is a continuous or binary or proportion variable that does not directly follow normal distribution. Residual or e could be follow another distribution family (Poisson, Binomial . . .). And the estimation method is based on the maximum likelihood (ML).

So, linear regression is to find best line passing by scatter plot.

9.5.2 R Code

We used data from Hoc et al. 2019 in the optimization of black soldier fly reproduction experiment.

```r
Male<-read.csv2("C://Users/Abeille/OneDrive/Bert_Analyse/Joakim_Stat/Male.csv")
str(Male)
```

```
## 'data.frame':    72 obs. of  6 variables:
##  $ Code       : chr  "26.1" "26.2" "26.3" "26.4" ...
##  $ Population : chr  "A" "A" "A" "A" ...
##  $ Masse      : int  50 100 150 200 250 300 350 400 450 500 ...
##  $ Nbr_male   : int  21 25 22 20 19 19 13 18 20 16 ...
##  $ Nbr_femelle: int  10 7 8 12 13 12 18 13 9 17 ...
##  $ MaleRatio  : num  0.677 0.781 0.733 0.625 0.594 ...
```

```r
#Change of structure
Male$Nbr_male<-as.numeric(Male$Nbr_male)
Male$Nbr_femelle<-as.numeric(Male$Nbr_femelle)
Male$Masse<-as.numeric(Male$Masse)

#Test of AC
shapiro.test(Male$MaleRatio)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Male$MaleRatio
## W = 0.98692, p-value = 0.6636
```

```r
#Good our Male ratio data are normal pvalue=0.6636.

#Linear regression
#Estimation of the model
LM = lm(Male$MaleRatio~Male$Masse)

#Summary of the model
summary(LM)
```

```
##
## Call:
## lm(formula = Male$MaleRatio ~ Male$Masse)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28561 -0.06861  0.02282  0.06260  0.22667
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.169e-01  2.388e-02  30.025  < 2e-16 ***
## Male$Masse  -2.993e-04  3.342e-05  -8.955  3.2e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09815 on 70 degrees of freedom
## Multiple R-squared:  0.5339, Adjusted R-squared:  0.5273
## F-statistic: 80.19 on 1 and 70 DF,  p-value: 3.2e-13
```

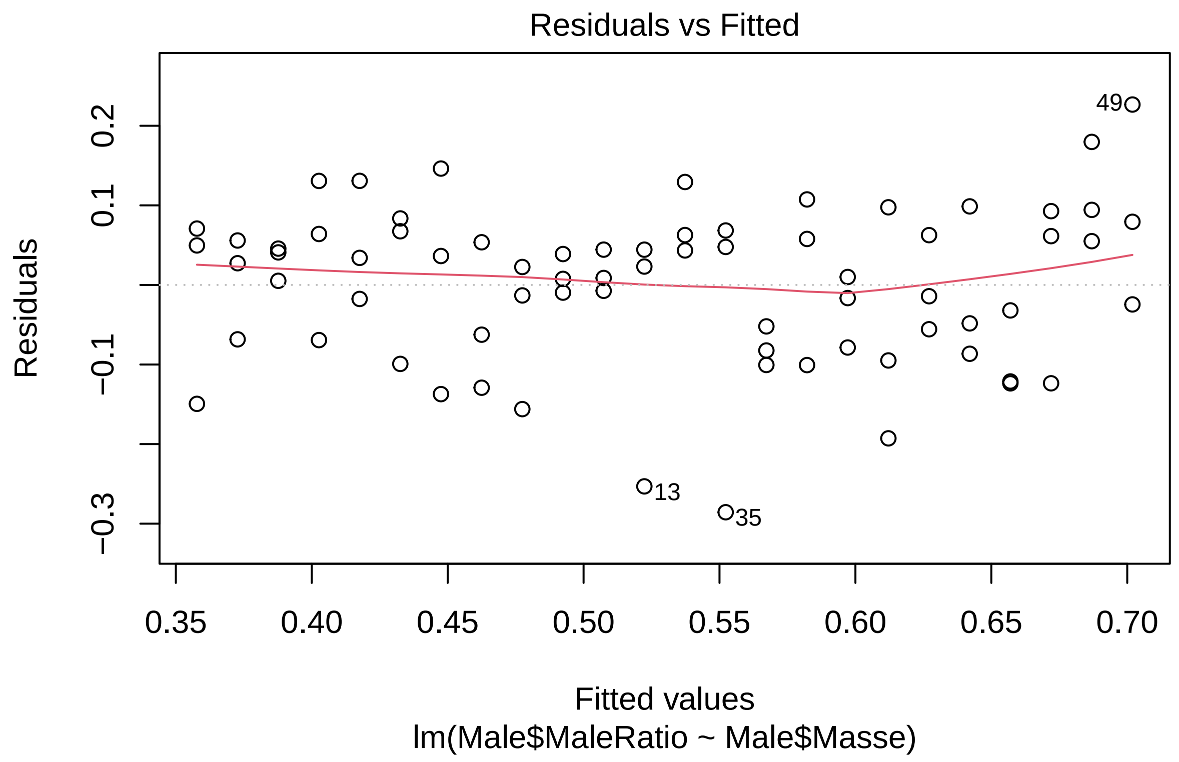
Here we could see some residuals graphics of lm function. R^2 is the determination coeficient which represent the percentage of the expressed variance. If $R^2 = 1$, that means that the model predicts pefectly the data, residuals are nul and all the points are completely aligned on the line (or curve) whose the slope is supposed to be non nul. If $R^2 = 0$, that means that your linear model has no predictive power (so not good model). This is the ratio between predicted vales (or variability explained by the model) and the variance of the observed values (total variability). We also test the normality of our residuals.

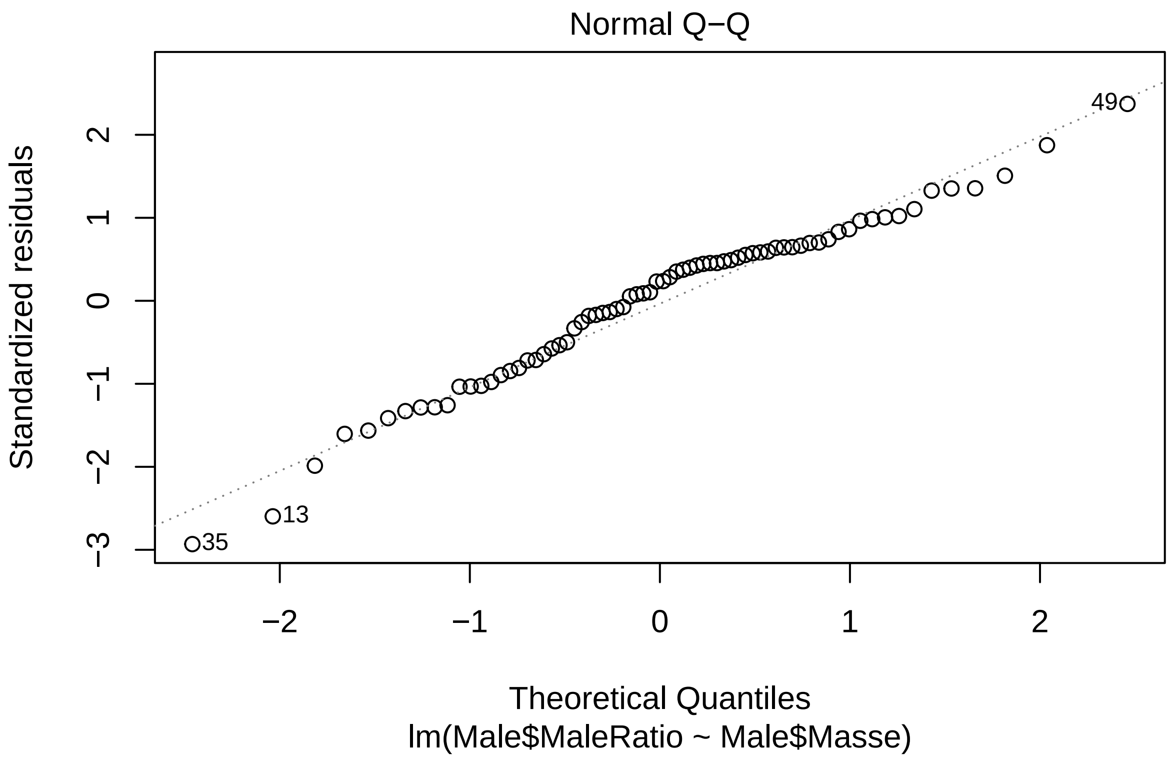
```
shapiro.test(LM$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  LM$residuals
## W = 0.97214, p-value = 0.1103
```

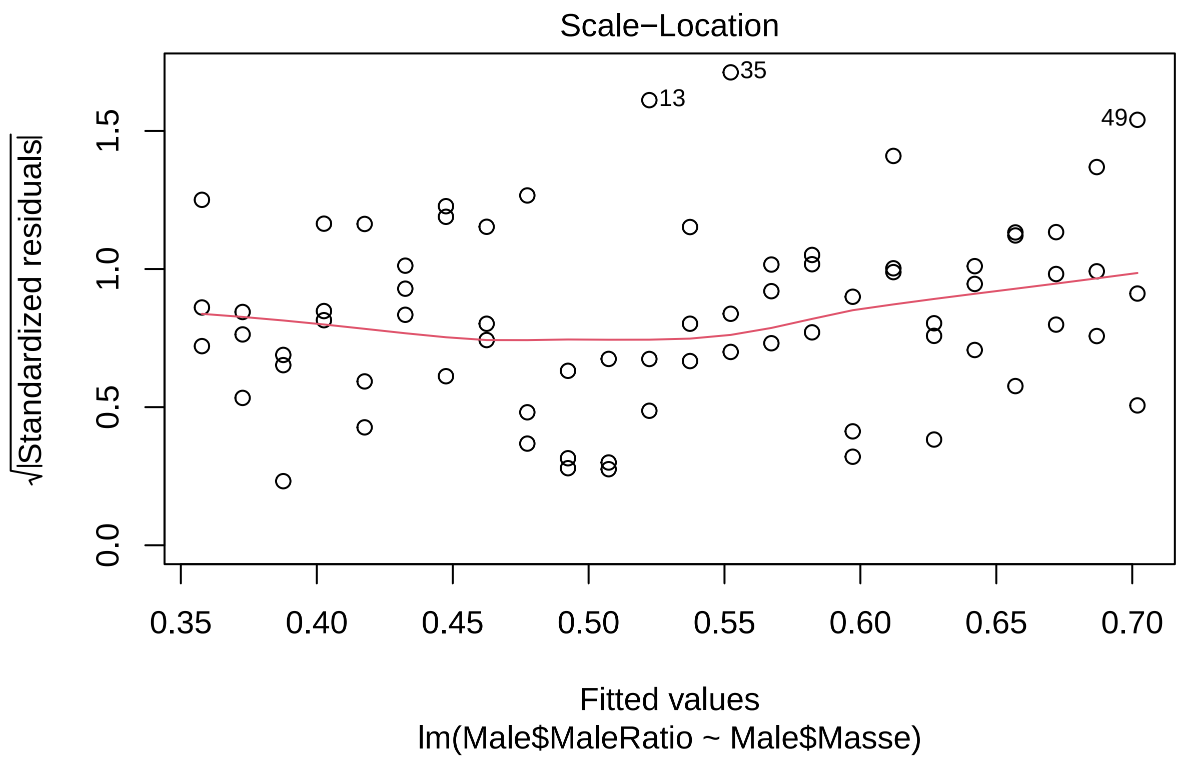
Ok good ! Our residuals follow a normal distribution. But, it is better to see the real graphics.

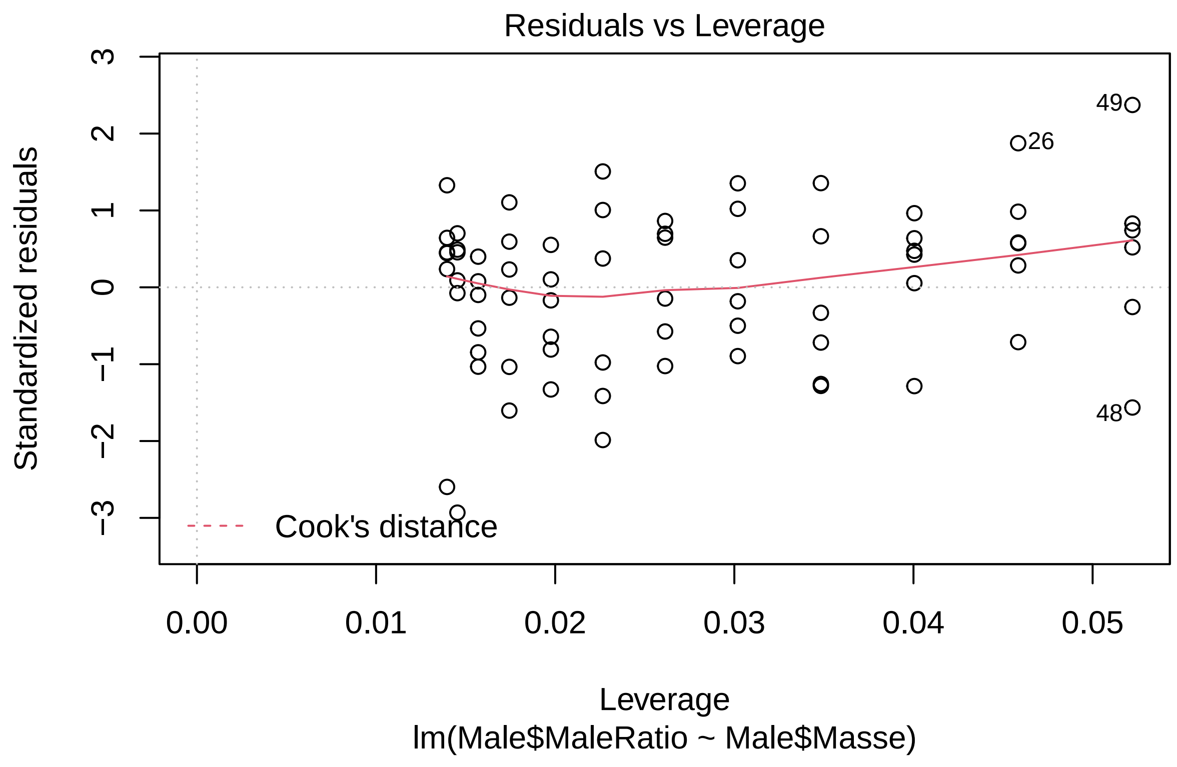
```
plot(LM)
```





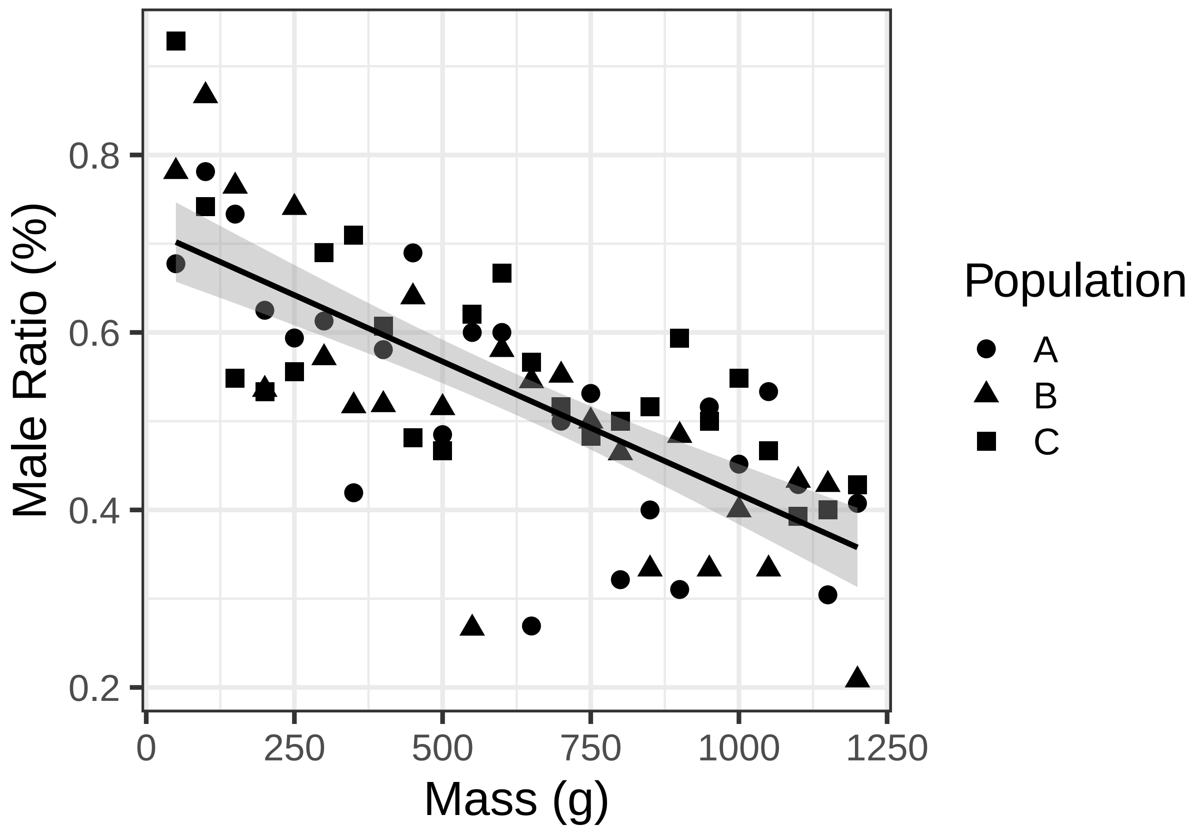
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Male$MaleRatio ~ Male$Masse)
49
13
35
−3
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
Fitted values
lm(Male$MaleRatio ~ Male$Masse)
35
13
49
0.0
0.5
1.0
1.5
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70



```
#Creation of a nice function in ggplot2 to see the linear regression
ggplotRegression <- function (fit,Population, xname, yname) {

  require(ggplot2)

  ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
    geom_point(aes(shape = Population),size=3) +
    stat_smooth(method = "lm", col = "black") + xlab(xname) +ylab(yname)
}

#Function ggplotRegression calling
p<-ggplotRegression(lm(MaleRatio ~ Masse, data = Male), Male$Population,
                    "Mass (g)","Male Ratio (%)")

#Other theme on the graphics
p<- p + theme_bw(base_size=18)
p
```

```
## `geom_smooth()` using formula 'y ~ x'
```



9.5.3 Interpretation of the graphics and the linear model

We have a model that has a middle prediction value : $R^2 = 0.53$ but we have a significant relationship (p-value: 3.2e-13) against the null hypothesis between MaleRatio variable (Y) and only one explicative variable Mass (X). All the parameter are also well estimated « 0.05, here also we reject null hypothesis that our parameter are null.

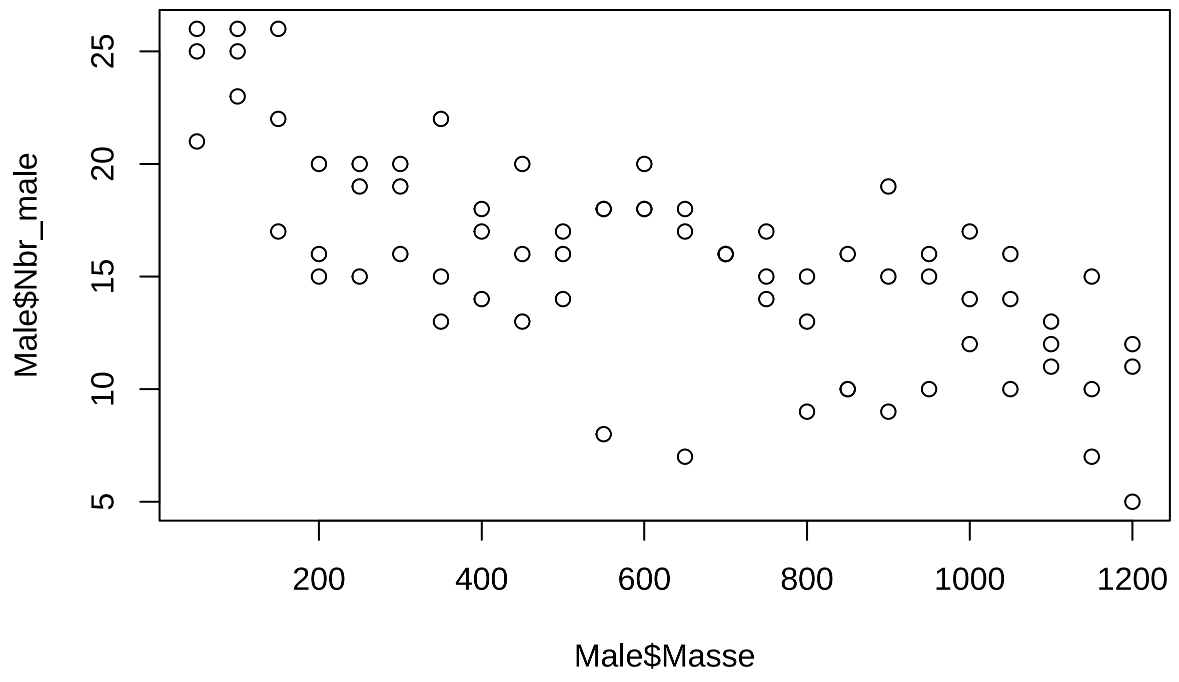
9.6 Generalized Linear Model (GLM)

9.6.1 Notions

The notions of this part change a little. Major change are done on residuals distribution and Y data and distribution.The estimation method change also into maximum likelihood. The normal GLMs give the same results as for LM supposing that the residuals have actually a normal distribution. LMs do not need this hypothesis for the estimation of the parameters but only for the inference. For this reason, it is recommended to use lm() function than glm() function when you want to estimate model with normal distribution.

So here, we used glm() to modelize positif integer such as counting data as male counting in our dataset.

```
plot(Male$Nbr_male ~ Male$Masse)
```

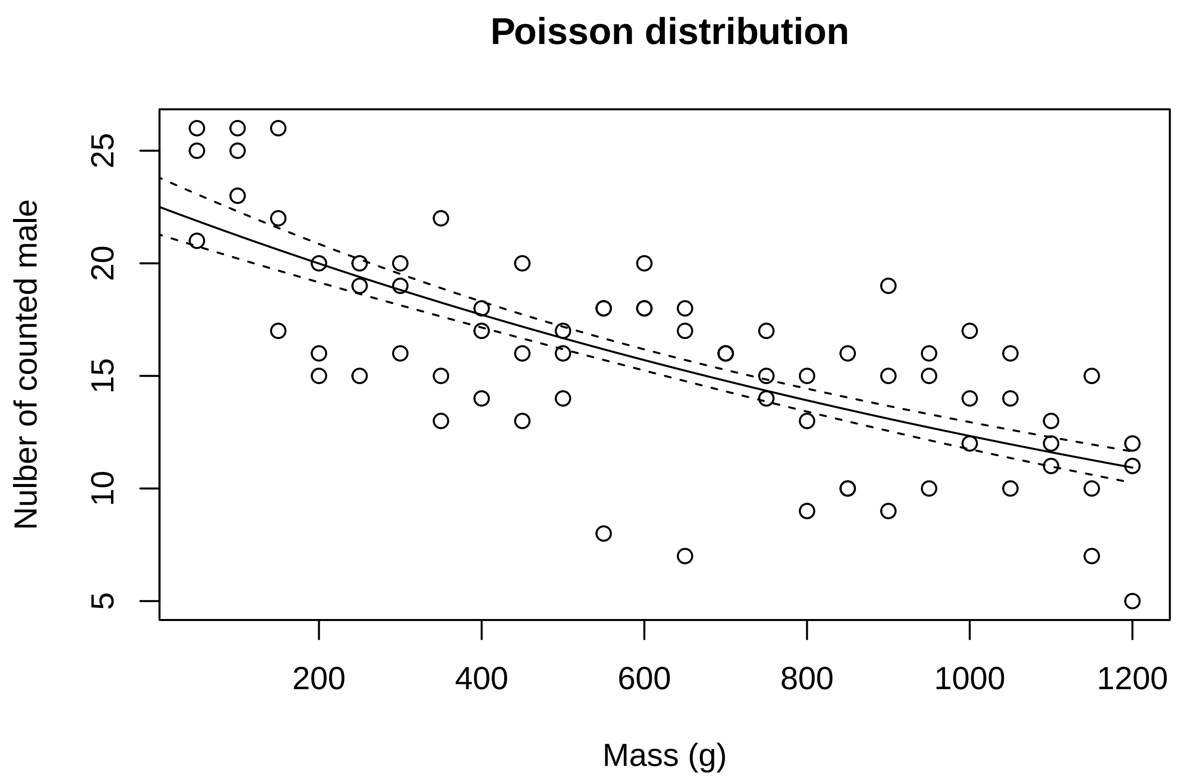


```r
modglm<-glm(Male$Nbr_male ~ Male$Masse, family = poisson)
summary(modglm)
```

```
##
## Call:
## glm(formula = Male$Nbr_male ~ Male$Masse, family = poisson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3625  -0.6690   0.1253   0.5683   1.5264
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.116e+00  5.642e-02  55.224  < 2e-16 ***
## Male$Masse  -6.038e-04  8.677e-05  -6.958 3.44e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 99.804  on 71  degrees of freedom
## Residual deviance: 50.748  on 70  degrees of freedom
## AIC: 383.23
##
## Number of Fisher Scoring iterations: 4
```

Graphics representation and standards error are in log scale. But, we retro-transform the borns and not the standard errors.

```
X<- cbind(1, seq(0, 1200,8))
pred<- X %*% coef(modglm)
se<-sqrt(diag(X %*% vcov(modglm) %*% t(X)))

lwr <- exp(pred - se)
upr <- exp(pred +se)
pred <-exp(pred)

plot(Male$Nbr_male ~ Male$Masse, ylab="Nulber of counted male"
     , xlab = "Mass (g)", main=" Poisson distribution")
lines(pred, x = seq(0, 1200,8))
lines(lwr, x = seq(0, 1200,8), lty=2)
lines(upr, x = seq(0, 1200,8), lty=2)
```



```
#plot(modglm) test this function to see the residuals
```

9.6.2 Interpretation

{still in progress}

10 Behavioral test in entomological studies

10.1 Ask the right research question !

Before setting up an experiment, it is important to ask what you want to highlight. This is the case in all types of scientific experiments but particularly in behavioral analysis. If the question is not clearly defined, it is likely that the behavioral tests performed will not be appropriate. Nevertheless, it is not very complicated ! In behavioral analyzes, three scenarios are most often presented: (i) I want to evaluate the behavior induced by an odour source, (ii) I want to compare the insect behavior facing two odors sources and (iii) I want to evaluate the impact of a parameter on the behavior induced by an odor source. The first thing to do is to be among these three general cases!

10.2 Behavior assessment induced by an odor source

In the case where the investigator wants to highlight an insect behavior in front of a odors source, he wants to show if the odor source has or has not an effect on the insect behavior. If the odor source has an effect, he wants to check if the source is attractive or repulsive for the insect. For these two types of behavioral analysis, the type of test is the same. This is a non-choice test. During these tests, the odor source is presented alone in front of a control (same device less the odor source to be tested). It is necessary to present the control at the same time as the odor source to be tested. If the odor source is presented independently of the control, it is important to check if the insect behaves in exactly the same way when the control is presented in the same environment as the odor source and when the tests on the odor source are made independently of the control. To do this tests in both conditions (control + source of odors in the same experimental unit and control in another experimental unit) must be realized to check that there is no differences and therefore that the insects have the same effect in both cases. Nevertheless, more easily, it is advisable to always present the control and the odor source to the insect at the same time in the same experimental unit. This makes it possible to obtain more robust tests and to avoid environmental biases between the control and the odor source to be tested.

Once someone wants to test the attractiveness of a molecule, it is necessary to check if the group of insects goes more towards the odor source than toward the control/blank one. For example, if an scientific inverstigator wishes to check the attractiveness of molecule A against fly insect. In behavioral cage, it will place a diffuser impregnated of molecule A (generally diluted in a solvent) and a diffuser without molecule A (if dilution is added the solvent alone). If the molecule is attractive, a greater proportion of insects will go to the diffuser containing the molecule. To have a good representation, it is advisable to test a relatively large number of insects (min 30 to 40 insects). The investigator will not always test the response of one insect at a time to the source of the odor to be tested. Indeed, if the behavior of an insect is not affected by the behavior of another insect, several insects can be introduced at the same time in the same experimental device. For example, if one wishes to identify an attractive ladybug molecule, the ladybug will always be introduced alone and not in a group. This is due to the fact that ladybugs deposit compounds during her movements that create a trail that is often followed by other ladybugs. In this case, the behavior of a ladybug influences the behavior of other ladybugs inducing a bias during the experiment.

The setting up of an experiment to highlight if a source of smell is repulsive could be done in the same way. Nevertheless, in this case, the goal is to show that the majority of insects do not go to the odor source tested. The instructions in the following paragraph (number of insects, group of insects or not, presence of the control) are applicable in the case of this test.

10.3 Comparison of induced behavior by several odor sources

In some cases, the investigator wishes to compare the behavioral effect induced by two different odor sources. The questions asked are usually of the type: is the molecule A more attractive than the molecule B. For example, this kind of behavioral analysis can be highlighted when one wants to check if a necrophagous insect prefers one stage of decomposition with respect to another. It is also possible to check if one fruit is more attractive than another for a fruit fly or if a rotten fruit is more attractive than a fresh fruit. In this case, we no longer check the behavior of the insect induced by a odor source but we observe the behavioral response of the insect facing several sources of odors. It is therefore obvious that the two sources of odors are placed together in the same experimental unit.

10.4 One factor effect on the insect behavior

The influence of one factor on the behavior of an insect can be highlighted in front of a single odor source or face several sources. In the case of a single source of odor, it is important to always place the indicator during the test. In this type of behavioral test, the scientific investigator varies a factor that he assumes to affect the source of odors. Then, it checks whether the behavioral response of the insect to the original odor source is changed once the change is applied to the source. During this type of test, one gets closer to a test of choice. Below you will find some examples of such experiments which will allow you to better understand the principle of these tests.

Experiment 1: Evaluation of the impact of silicon enrichment on the attractiveness of a maize crop pest - As first step, maize plants are raised under two different conditions: in a non-enriched substrate silicon and in a substrate enriched in silicon. Then, the plants grown in both types of condition are placed in flight cage. The insect is then introduced into the cage to check if the insect is more attracted to a plant grown with silicon amendments.

Experiment 2: Impact of an increase in CO2 concentration on the attractiveness of bean plants to aphids - Bean plans are grown under high CO2 conditions under normal conditions. Then, by proposing the two types of shots to the aphids, one can check which one is preferred by the aphid.

10.5 Kind of insect behavior

In behavioral experiment, the attractiveness of one or more sources of odor is not always observed. Indeed, other behaviors can be observed and are obviously dependent on the insect tested. For example, in the case of **Episyrphus balteatus** (De Geer 1776), an hoverfly species, we can observe if this specimen hoverflights around the source of odor. For other insects, it is possible to look at whether the odor source induces a laying behavior or even if a source of odors A is preferred for eggs laying compared to a odor source B simply by counting the number of eggs present on the source. It can also be observed whether a chemical compound induces gregarious behavior. In this case we check if a group of insects is around the source of odors. These types of experience must be considered on a case-by-case basis and general rules are more complicated to establish. Nonetheless, the statistical tests used in conventional attractiveness tests can easily be transposed to these more specific experiments

11 Assessment of the attractivity of chemical compound

11.1 Generalities

In order to be as clear as possible to detail the statistical tests to carry on, we will consider, throughout this part that we set up test to check if the molecule A is attractive for an insect. At first, the different tests, that can be performed, will be exposed by specifying for each test in which conditions they can be applied. Then, the statistical tests to be performed on these behavioral tests will be detailed. Finally, the R codes of each of these tests will be explained.

11.2 Behavior assays

Two tests are usually used in behavioral analysis laboratory when studying the behavior of the insect against odor source: (i) dependent non-choice test; (ii) independent non-choice test. The non-choice tests are intended to highlight the sources of odors that are attractive or repellent to the insect studied. The independent non-choice test involves presenting the odor source alone to an insect or group of insects. In this case, the control is presented independently of the odor source studied. The dependent non-choice test consists of presenting the control with an insect at the same time as the source of the odors studied. These non-choice tests each have their advantages and disadvantages. The independent non-choice test has the advantage that the insect populations tested are independent, that is, the attractiveness of the odor source is not related to the attractiveness of the control. This makes the statistical test performed more robust, but only if the control is carried out under the same conditions as the odor source studied (in parallel). Indeed, the behavior of insects can be affected by a large amount of factors such as brightness, the time of day, hunger, temperature, mating ...

It is therefore imperative to carry out the control test in parallel with the tests of the odor source. Nevertheless, this test can only be performed if the insect expresses the same behavior when the control is isolated from the source of odors than when it is proposed at the same time as the odor source. If this is not the case, it affects the behavior of the insect and so has to always be presented at the same time as the source. Indeed, if the control is attractive to the insect it could be that the experimenter concludes that the source of smell is not attractive although it could be. For example, we study the attractiveness of rotten fruit on a hungry fruit fly. The fruit is placed into a Petri dish washed with ethanol. The control is therefore a petri dish washed with ethanol without the fruit. The starving fly will be attracted by the Petri dish that emits alcohol (fruit fermentation product). In this example if the control is presented alone, we can imagine that 50 out of 60 flies will be attracted by the petri dish. The rotten fruit being a food source appreciated by the fly, we also find 50 out of 60 flies that are attracted by the fruit.

Conclusion, the fruit is not attractive since the control alone attracts the same proportion. Except this is not true because if the control had been presented at the same time as the fruit, we can be certain that the majority of the flies would have been towards the rotten fruit since this also releases other compounds which will give of advantage envy to the insect that ethanol alone. It is therefore preferable to always place the control at the same time as the source of odors studied even if the number of insects to be tested must be slightly higher to reach a robustness equal to the independent non-choice test. Once the choice of the test has been made, it is necessary to identify the different fixed and random factors that are present in the test. This will determine the choice of the statistical test to be applied.

11.3 Statistical test

When setting up a behavioral test it is important to highlight the different factors that can affect the behavior of insects. This part is important because it will condition the choice of the statistical test. Two types of factors can be identified: fixed factors and random factors. Fixed factors are the factors that are mastered by the experimenter. These factors are generally unchanged throughout the experiment (e.g., sex of the insect, whether mated or not ...). Random factors are the factors identified but which can not be mastered. These factors are often hard to identify. When an experiment is spread over several days, the temporal aspect must be taken into account. In order to take into account this temporal dimension, the experiments are split into blocks. The block is an experimental unit group realized on a short time. The block is therefore a factor that can not be controlled. It is therefore part of the random factors. It is important to make blocks when it makes sense. Indeed, when placing blocks in the experiments, we can evaluate the impact of these blocks on the behavior. When there is an impact of the block the experience must be redesigned (enlarge the sample, select specific moments of the day to carry out the experiments ...). However, if they have no effect, the scientific result is more robust. Indeed, the wood is part of the residue. The residue is compared to the values of the treatment. When the block has no significant effect, the residue is decreased by the value of the block and is therefore smaller. The presence or absence of significance is therefore more marked. Other random factors can be identified. For example, if the odor source to be tested is biological, this source is a random factor due to the intrinsic variability at the source. For example, the odor of two maize plants raised under the same conditions may be different due to a genetic character not mastered by the scientific investigator. The corn plant is therefore a random factor. The odor source can be a random factor that must imperatively seek to control. In an experiment it is important that the random factors have no effect on the experiment.

A second point is important to evaluate when choosing statistical tests. Indeed, we have to look whether the factors are crossed or hierarchical. Two factors are crossed when all treatments undergo all levels of each factor. For example, if we want to see the attractiveness of the decomposition stages of a body on *Dermestes* sp. males and *Dermestes* sp. females (Coleoptera). Two factors are identified: the stage of decomposition and the sex of the insect. When attractiveness of all stages of decomposition are tested on males and females. In this case each level of the stage factor is applied to each level of the sex factor. The factors are crossed. Two factors are hierarchical when all levels of factor A are not applied to factor B. For example, in the case where an scientific investigator (could be a master student ^^) performs behavioral tests to evaluate the attractiveness of the first two stages of decomposition on these necrophagous beetles and that other scientific investigator (could also be a PhD student ^^) carries out the same tests but on the last two stages of decomposition, two factors are identified. The stage factor and the investigator factor (since the same investigator did not do all the tests). As all stages have not been tested by both the master student and the doctoral student, the stage factor is hierarchized to the investigator.

Therefore several statistical tests have been identified for the behavioral analysis:

- *Chi-squared goodness of fit* is a non-parametric test used to determine how the observed value of a given phenomenon is significantly different from the expected value. The term “goodness of fit” is used to compare the distribution of the observed sample with the expected probability distribution. It determines to what extent the theoretical distribution (such as the normal, binomial or Poisson distribution) corresponds to the empirical distribution. For example when looking at whether a substance is attractive to an insect, the insects tested will be divided between the control and the odor source tested. We will have a number of insects on the control and a number of insects on the source. If the source is not attractive, the distribution of insects between the control and the source is random. We must therefore observe a 50/50 distribution at the level of the sources of odors. In the same way that if you throw a coin in the air several times, the chance of falling on pile or face is 50% if the number of essays is high enough. The chi-square goodness of fit test will therefore check if a distribution is significantly different from a random theoretical distribution of 50/50. This test can only be used if the insects are tested one by one and in the absence of random factors.
- *Independence Chi-squared* This statistical test will be applicable for preference tests with several odor sources (more than two). We also will be applied to compare a distribution observed in the laboratory to another distribution. For example, the proportion of neutered dogs in the world is 40% of neutered and 60% of unnetered dog. After a survey dedicated to detection dogs, we observe that 70% of the detection dogs are neutered. If we want to know if there are more neutered dogs in the police then in general, we compare the police dogs proportion (70%/30%) to the world mean proportion (40%/60%).
- *ANOVA* or ANalysis Of VAriance is a test based on equality of variances on continuous type data in the absence of random factors. It compares the mean of the selected variable. Continuous data is data that can take any positive values. We no longer talk about counting but measurements, for example a measurement of the dose of molecules or a weight, a size ... This test will therefore be less used in behavior or for specific measures such as the time during which the insect expresses in behavior.
- *GLM* or Generalized Linear Model is a test based on equality of the variances. These tests are a generalization of Linear Model (LM), the explicative variables (X) does not change but here the explained variable (Y) could be dependent on other theoric distribution families (such as Poisson distribution) and the estimation method is based on mawimum likelihood (ML). It could be performed on counting data and thus by definition discontinuous. For example, this test is applied when several insects are tested at the same time and the number of insects “attracted” by the source tested is counted.
- *GLMM* or Mixed Generalized Linear Model is a test of equality of variances on continuous and discontinuous data in the presence of random factors. This test therefore makes it possible to check the

impact of the random factor.

Before GLM, ANOVA and GLMM tests, it is important to check both important application conditions: homoscedasticity and normality (see chapter X). In case these conditions are not respected data transformations can be applied. For an ANOVA normality is imperative. On the other hand for a GLM and GLMM, if the normality is not respected, one can change the code to follow a binomial or Poisson distribution or to make a transformation of data (to see on a case by case basis).

Moreover, for these three tests based on equality of the variance, it is necessary to check if the factors are crossed or hierarchical. This is important for coding in R.

11.4 R Code

11.4.1 Chi-squared goodness of fit

```
### result vector creation
result <- c(40,60)
### Chi-Square test
res <- chisq.test(result, p = c(1/2, 1/2))
res
```

```
##
##  Chi-squared test for given probabilities
##
## data:  result
## X-squared = 4, df = 1, p-value = 0.0455
```

result is the vector that contains the results. We give the number of insects attracted by the blank and the number of insects attracted by the source of odors.

p is the vector containing the expected proportions. In this vector, we always indicate proportions and not numbers. As we want to verify that the insect is more attracted by the source of odors studied, we want to verify that the distribution is different from a 50/50 random distribution as we study the distribution between the control and the source.

res is a list object that gives the chi-square value and the p-value.

You could see all the results in typing the object result follow by "$":

```
res$p.value
```

```
## [1] 0.04550026
```

11.4.2 Chi-squared goodness of fit (more than 2 sources)

```
### result vector creation

result <- c(40,60,20,30)

### Chi-square

res <- chisq.test(result, p = c(1/4, 1/4, 1/4, 1/4))
res
```

```
##
##  Chi-squared test for given probabilities
##
## data:  result
## X-squared = 23.333, df = 3, p-value = 3.441e-05
```

result is the vector that contains the results. We give the number of insects attracted by the different sources of odor.

p is the vector containing the expected proportions. In this vector we always indicate proportions and not numbers. Since we want to check that the insect is more attracted by one source of odors than another, we want to check that the distribution is different from a random distribution of 1 / (nbr of sources) in the presented code 1 / 4.

res gives the chi-square value and the p-value.

11.4.3 Independence Chi-squared

Cfr attractiveness part of several odorous sources.

11.4.4 ANOVA

We will make the ANOVA on the graveyard dataset.

```r
###implementation de la matrice de resultat en format csv

Aov_data<-graveyard
Aov_data ## Visualisation du tableau de donnee

### Normalite

shapiro.test(residuals(lm(Aov_data$A~ Aov_data$y)))

### Homoscedasticite

    ## si un seul facteur AV1

bartlett.test(Aov_data$A, Aov_data$y)

    ## si plusieurs facteurs AV2

Data$combinaison <- paste(Aov_data$A1, Aov_data$Type_A2, sep="_")
bartlett.test(Data$combinaison, Aov_data$y)


### Test ANOVA

fit <- aov(y ~ A, data= Aov_data) # y est la variable numerique. A indique les groupes de facteurs
summary(fit)
```

11.4.5 GLM

```r
library(lme4)
###loaing of the matrix of result

glm_data<- read.csv2(file.choose())
glm_data ## Data vizualisation

###GLM

Glm <- glm(y ~ A, data= glm_data, family = gaussian)


### Normality Test

shapiro.test(resid(Glm))

### Homoscedasticity test

overdisp_fun <- function(model) {
  ## number of variance parameters in
  ##   an n-by-n variance-covariance matrix
  vpars <- function(m) {
    nrow(m)*(nrow(m)+1)/2
  }
  model.df <- sum(sapply(VarCorr(model),vpars))+length(fixef(model))
  rdf <- nrow(model.frame(model))-model.df
  rp <- residuals(model,type="pearson")
  Pearson.chisq <- sum(rp^2)
  prat <- Pearson.chisq/rdf
  pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
  c(chisq=Pearson.chisq,ratio=prat,rdf=rdf,p=pval)
}
overdisp_fun(Glm)

###Result

summary(Glm)
```

11.4.6 GLMM

```r
library(lmerTest)

###Matrix implementation csv

GLMM_data<- read.csv2(file.choose())
GLMM_data ## Data visualisation


### Model
```

```
GLMM <- lmer(y~ A, GLMM_data)

### Conditions of application

##normality

shapiro.test(resid(GLMM))

## Homoscedasticity

overdisp_fun <- function(model) {
  ## number of variance parameters in
  ##   an n-by-n variance-covariance matrix
  vpars <- function(m) {
    nrow(m)*(nrow(m)+1)/2
  }
  model.df <- sum(sapply(VarCorr(model),vpars))+length(fixef(model))
  rdf <- nrow(model.frame(model))-model.df
  rp <- residuals(model,type="pearson")
  Pearson.chisq <- sum(rp^2)
  prat <- Pearson.chisq/rdf
  pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
  c(chisq=Pearson.chisq,ratio=prat,rdf=rdf,p=pval)
}
overdisp_fun(GLMM)

### Test

posthocGLMM <- step(GLMM, reduce.random=FALSE, reduce.fixed=FALSE)
posthocGLMM


###FOnction rntransform
function (formula, data, family = gaussian)
{
    if (is(try(formula, silent = TRUE), "try-error")) {
        if (is(data, "gwaa.data"))
            data1 <- phdata(data)
        else if (is(data, "data.frame"))
            data1 <- data
        else stop("'data' must have 'gwaa.data' or 'data.frame' class")
        formula <- data1[[as(match.call()[["formula"]], "character")]]
    }
    var <- ztransform(formula, data, family)
    out <- rank(var) - 0.5
    out[is.na(var)] <- NA
    mP <- 0.5/max(out, na.rm = T)
    out <- out/(max(out, na.rm = T) + 0.5)
    out <- qnorm(out)
    out
}
<bytecode: 0x000000003176d848>
<environment: namespace:GenABEL>
```

12 Multivariate Analysis

Multivariate analysis are used in **Volatolomic** to compare odor profiles.

12.1 Generalities

During a volatolomic analysis, the result obtained is derived from the analyzed chromatograms. Several compounds make up this odorous profile. It is very rare for a single compound to constitute the mixture. In this case, a GLM or GLMM analysis is required. The same type of analysis must be performed if the goal is to analyze the emission changes of a single compound constituting a profile under different parameters. In the case of whole odorous profile analysis, it is necessary to switch to multivariate analysis. In multivariate analysis, it is generally desired to perform a Principal Component Analysis (PCA) followed by PermManova.

12.2 Statistical Test

PCA allows to transform linked variables to new variables uncorrelated from each other. It is a geometric method because the data are represented in a plane defined by two variables that best explain the data. The goal is therefore to determine principal components that are combinations of existing variables. The linked variables are then grouped into one component. Then, the components are placed in pairs and the data set is then distributed in the plane formed by these axes. Under each axis percentages of representativeness of the variability (variance) are expressed. The goal is to represent the data set using the axes (or components) representing the best variability.

At first, a graph of the eigenvalues is realized. This graph represents in abscissa the principal components and in ordinate the percentage of representativity of the various components. From this graph, the components are selected. The goal is to select the variables in order to reach a total of about 60 to 70% representativeness. The higher the representativeness, the better the results will be. Then, the selected components will each be coupled to another component in order to achieve the correlation circles. These circles represent the variables studied in the form of vectors. The axes express the components. The closer a vector is to an axis, the more the component expresses that variable. For example, a vector coinciding with the axis is totally expressed by the component constituting this axis. A graph of contributions of the variables to the axes can also be generated. This makes it possible to see the percentage of contributions of each of the variables for each of the components (axes or also called dimensions). Then, the PCA is represented in a plane defined by the selected components. The goal is to highlight groups of samples that are separated from each other. If they are separated, it means that there is a difference between these groups. Then, we must identify which axis separates these groups of variables. Indeed, if two groups of data are separated by the abscissa axis, the difference is due to the variables composing this axis. Thus, returning to the correlation circle, it is easy to say which variable contributes to the differentiation of this group of data.

PermManova is a statistical test of the variance which is carried out on several variables, from where the interest to use it during the analysis of odorous profile. Indeed, in this case the different compounds constituting the profile will define each of the variables. The data file is thus a matrix whose first columns constitute the factors (parameters to be tested and random parameters) and the other columns represent the air under the peak of the compounds constituting the odorant profile (a column = a compound). The idea of the PermManova is that it will realize several value permutations (peak air) within the matrix. Each permutation is performed randomly and makes it possible to obtain a new data matrix. Then the PermManova will perform a test to see if a difference exists between the new matrix generated and the actual matrix (encoded in R). If a difference exists, it means that an effect of the tested parameters is measured. Indeed, the matrices generated being due to chance, if a difference is observed between the matrices generated by R and the original matrix then the original matrix is not due to chance. This makes it possible to show if a parameter affects the variables presented and therefore if a parameter studied has an impact on the odorous profile obtained. If a difference is observed within a parameter vector, to find out at which parameter change the difference is present, new matrices simpler than the original one must be realized. The advantage of the PermManova compared to the classical anova is that it is robust to the violation of the normality and homoscedasticity conditions. If a slight violation is observed, the test can be applied without any problem.

One specification is that if PCA are only use when Euclidean distribution is used in the Permmanova. If the conditions of applications are not respected in Euclidean distribution, the scientist will try with a Bray Curtis distribution. If the conditions of applications are respect with the news distribution, you use an NMDS instead of the PCA. It is the same graphic but by respecting the Bray Curtis distribution. The goal of NMDS is to collapse information from multiple dimensions (e.g, from multiple communities, sites, etc.) into just a few, so that they can be visualized and interpreted. Unlike other ordination techniques that rely on (primarily Euclidean) distances, such as Principal Coordinates Analysis, NMDS uses rank orders, and thus is an extremely flexible technique that can accommodate a variety of different kinds of data. [Source https://jonlefcheck.net/2012/10/24/nmds-tutorial-in-r/]

12.3 R code

12.3.1 PCA

```r
# Data implementation

sol.data <- read.csv2("C://Users/Abeille/OneDrive/Assistanat/matrice-resultat-limon.csv")
sol.data
```

```
##      sol rep layer    MS Corg    Nk  P2O5   CaO   MgO   K2O  Na2O  Ph Cond
## 1  Limon PR     WR 97.04 2.20 0.137 0.253 0.748 0.488 0.234 0.040 7.0  377
## 2  Limon PR     WR 96.50 2.36 0.129 0.264 0.864 0.675 0.264 0.043 7.0  378
## 3  Limon PR     WR 95.18 2.32 0.126 0.278 0.792 0.544 0.250 0.044 7.1  353
## 4  Limon PR     WR 94.39 2.17 0.092 0.249 0.776 0.497 0.242 0.042 7.1  534
## 5  Limon   r    C1 97.15 2.65 0.147 0.282 0.943 0.551 0.264 0.045 7.0  495
## 6  Limon   r    C2 95.00 2.73 0.186 0.287 0.837 0.528 0.266 0.044 7.6  941
## 7  Limon   r    C3 93.41 2.89 0.228 0.282 0.917 0.516 0.264 0.050 7.3  925
## 8  Limon   r    C4 92.32 3.09 0.227 0.313 0.798 0.514 0.295 0.054 7.9 1034
## 9  Limon   r    C1 95.81 2.77 0.113 0.275 0.811 0.511 0.246 0.043 7.1  304
## 10 Limon   r    C2 91.96 2.84 0.192 0.304 0.891 0.537 0.279 0.049 7.9  646
## 11 Limon   r    C3 90.40 2.78 0.208 0.282 0.863 0.598 0.246 0.058 7.7  895
## 12 Limon   r    C4 90.22 2.99 0.239 0.286 0.716 0.467 0.270 0.037 7.8  873
## 13 Limon   r    C1 97.51 2.48 0.128 0.226 0.706 0.456 0.225 0.035 7.0  832
## 14 Limon   r    C2 95.70 3.00 0.189 0.253 1.217 0.425 0.242 0.035 7.5  733
## 15 Limon   r    C3 94.69 2.78 0.192 0.243 0.784 0.447 0.219 0.034 7.7  639
## 16 Limon   r    C4 92.93 2.80 0.259 0.303 0.811 0.507 0.265 0.050 7.9  903
```

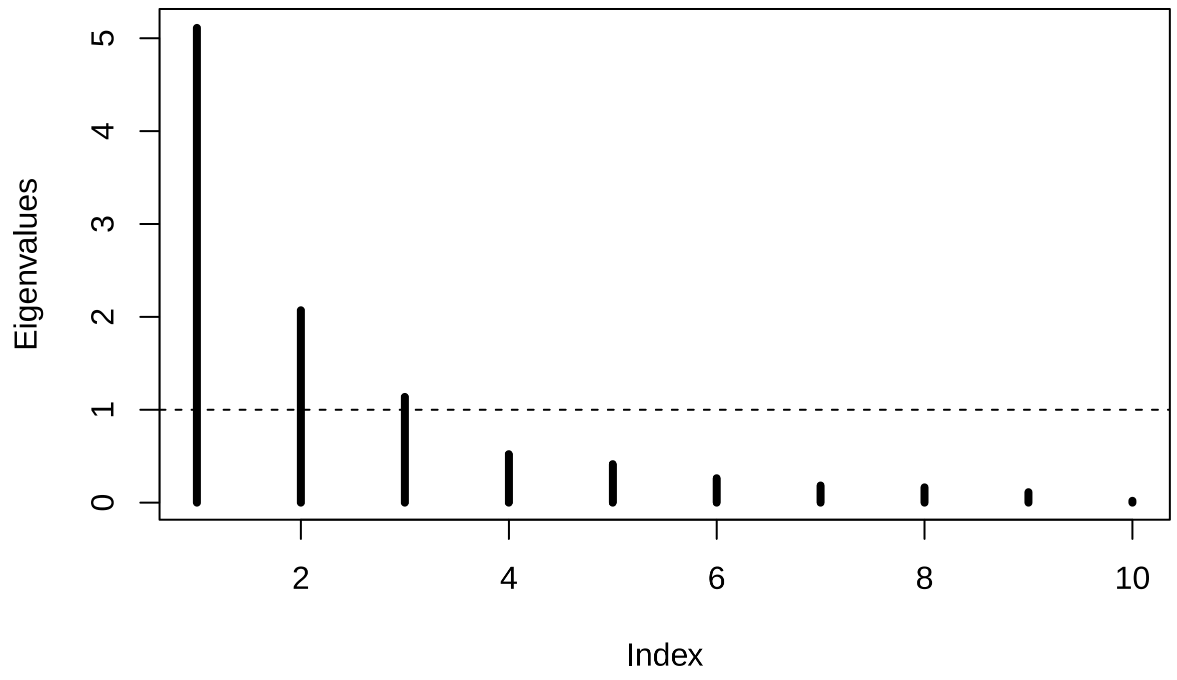
```r
# load packages

library(FactoMineR)
library(factoextra)
```

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

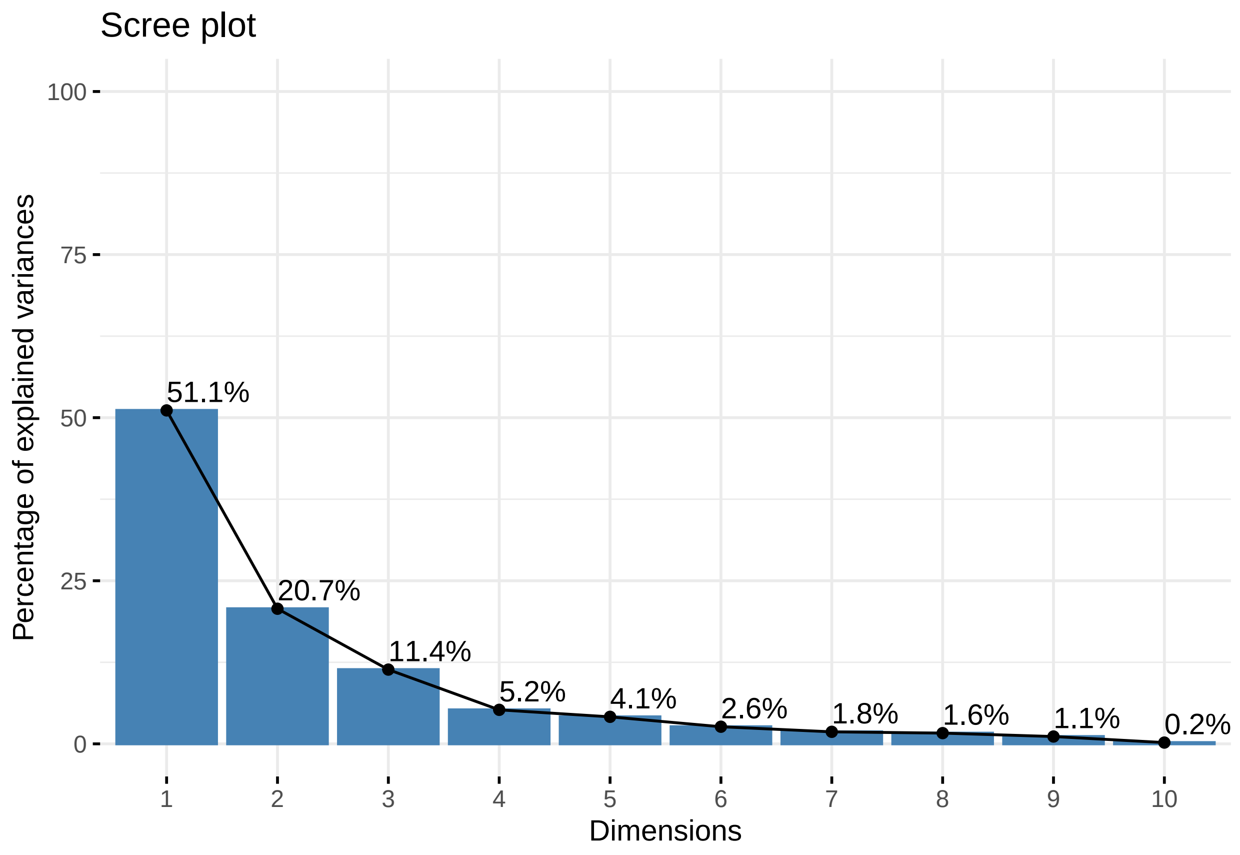
```r
#library(wesanderson)
library(multcomp)
```

```
## Le chargement a nécessité le package : mvtnorm

## Le chargement a nécessité le package : TH.data

## Le chargement a nécessité le package : MASS

##
## Attachement du package : 'TH.data'

## L'objet suivant est masqué depuis 'package:MASS':
##
##     geyser
```

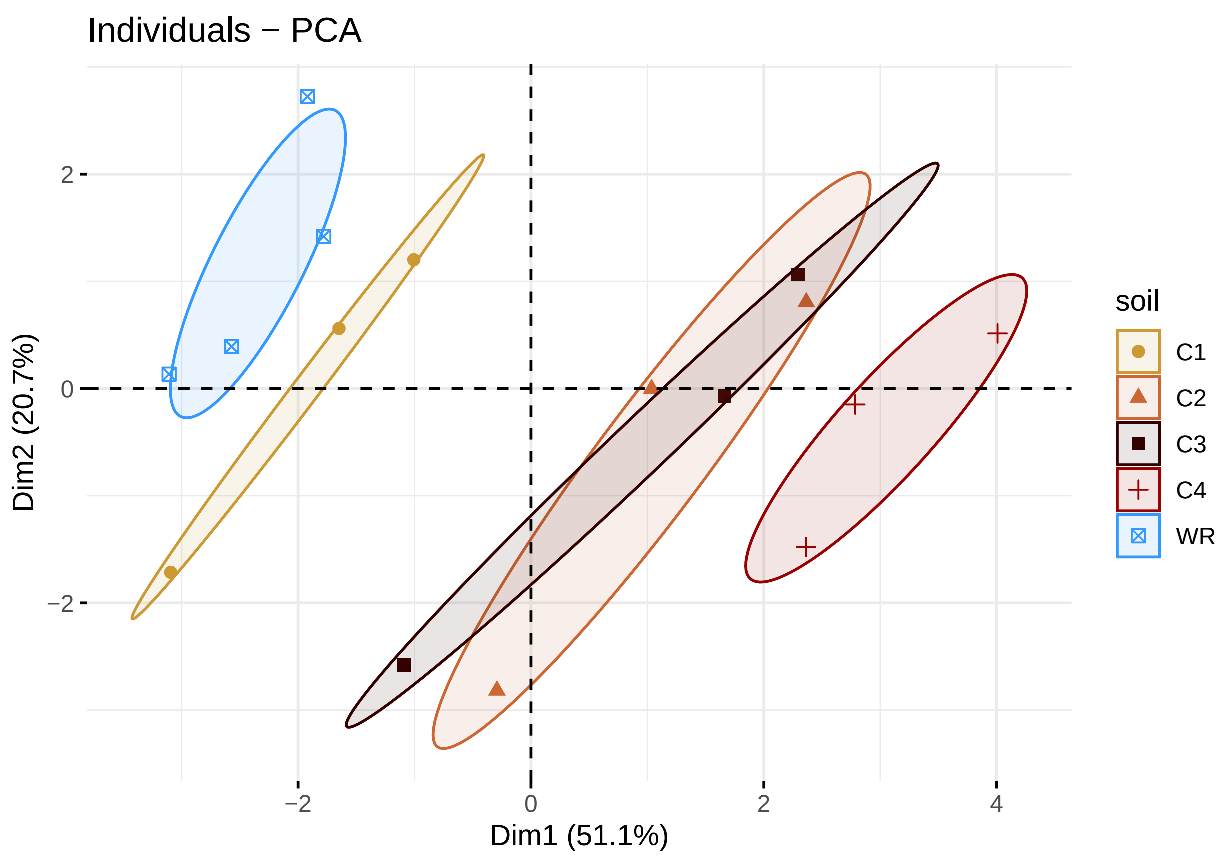
```
############ PCA ############
sol.pca <- PCA(sol.data[-(1:3)],graph = FALSE)

# eigenvalues
#################

plot(sol.pca$eig[,1], type="h", lwd=4, ylab="Eigenvalues")
abline(h=1, lty="dashed")
```



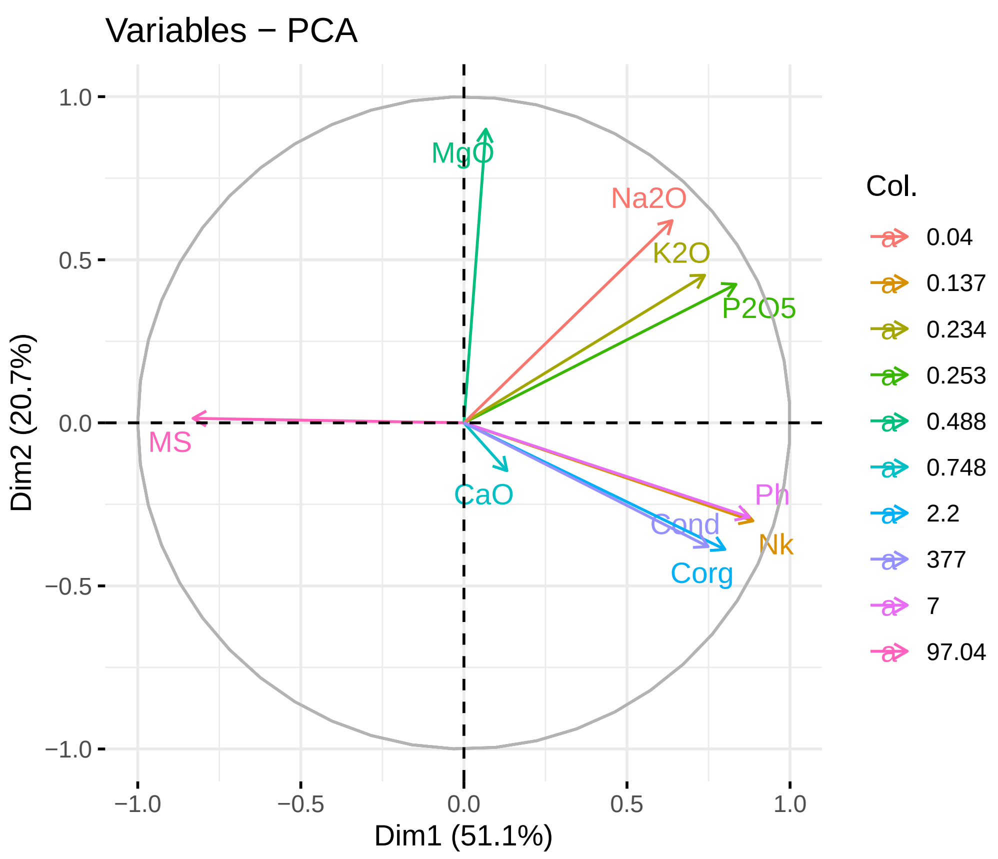
```
#Other figure
fviz_eig(sol.pca, addlabels = TRUE, ylim = c(0, 100))
```



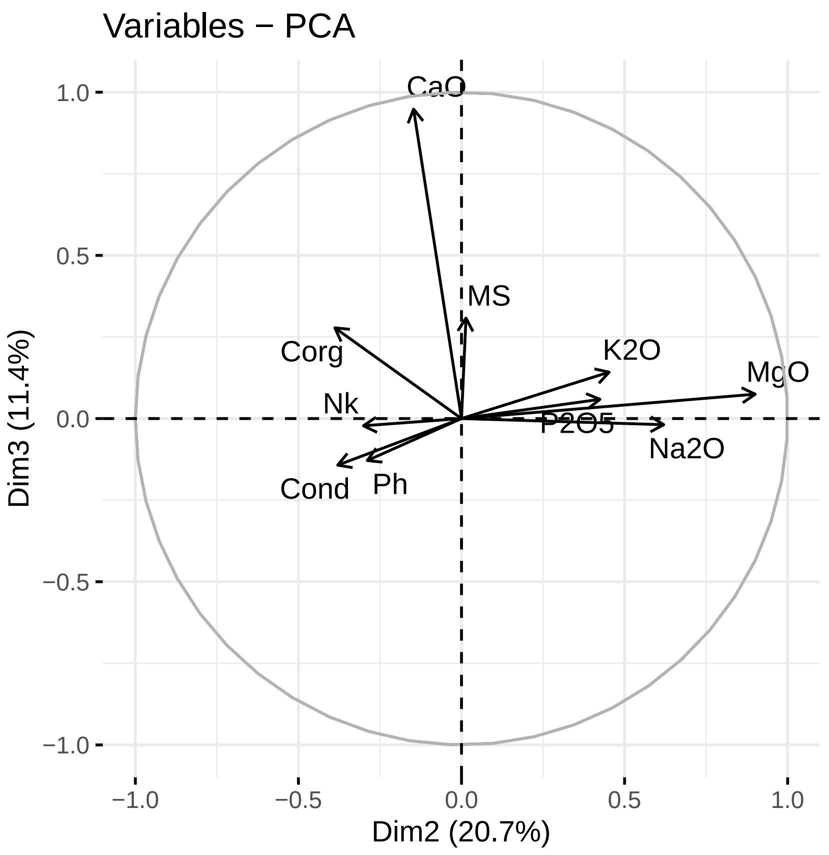
```
############Individual############

#color by group (factor)

fviz_pca_ind(sol.pca, geom.ind = "point",axes=c(1,2), col.ind = (sol.data$layer),
             addEllipses = TRUE, ellipse.type = "confidence", # Ellipses de concentration
             mean.point = FALSE, # retire le centre de gravite des groupes
             legend.title = "soil",
             palette = c("#CC9933", "#CC6633", "#330000", "#990000","#3399FF"),
             pointsize= 2)
```



```
##############Variable###########

# Color according to group !

group <-as.data.frame(t(sol.data[1,]))
colnames(group)<-"col"
group1<-group$col
group1<-group1[-c(1:3)]

fviz_pca_var(sol.pca,axes=c(1,2), col.var = group1,repel = TRUE)
```



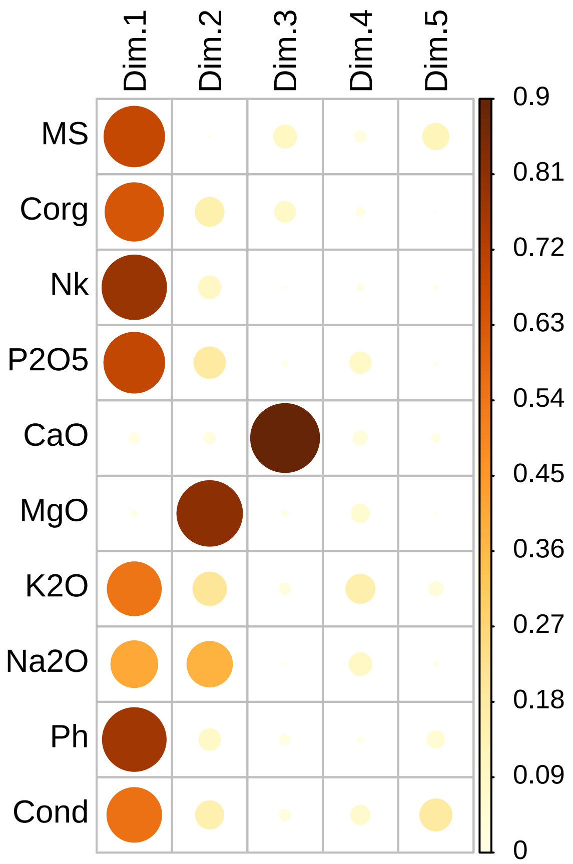
```
fviz_pca_var(sol.pca,axes=c(2,3),repel = TRUE)
```



```
#Vizualize display quality of the variables along all the dimensions

library("corrplot")
```

```
## corrplot 0.92 loaded
```

```
var <- get_pca_var(sol.pca)
corrplot(var$cos2,tl.col = "black",tl.offset =0.2,tl.cex = 1,cl.align.text="l", is.corr=FALSE)
```



```
sol.pca <- PCA(sol.data[-(1:3)],ncp = 3, graph = FALSE) #PCA(X, ncp = 5)
```

12.3.2 PermManova

```
###load the matrix

sterol <- read.table(file.choose(), header=TRUE, sep="\t", na.strings="NA", dec=",", strip.white=TRUE)

 ##remove columns of factor
comp <- sterol[,-c(1:2)]
 ## Define factors
sp <-factor(sterol[,1])
sp2 <- factor(sterol[,1])

### Equal matrices variances-covariances

sterol.bray<-vegdist(comp, method="euclidean")
###==> if conditions are not respected use method=bray (but you have to use NMDS instead of PCA)
sterol.bray.MHV <- betadisper(sterol.bray, sp)
anova(sterol.bray.MHV)
permutest(sterol.bray.MHV)

### permutation multivariate analysis of variance (PerMANOVA) non-parametric gold MANOVA (npMANOVA)
```

```r
library(vegan)
adonis(comp~sp, permutations=999)

### Pairwise for permMANOVA
####If you use bray instead of Euclidean in permMANOVA do the same here in the part sim.method=
pairwise.adonis <- function(x,factors, sim.function = 'vegdist', sim.method = 'euclidian',p.adjust.m ='
{
  library(vegan)
  co = combn(unique(as.character(factors)),2)
  pairs = c()
  F.Model =c()
  R2 = c()
  p.value = c()
  for(elem in 1:ncol(co)){
    if(sim.function == 'daisy'){
      library(cluster); x1 = daisy(x[factors %in% c(co[1,elem],co[2,elem]),],metric=sim.method)
    } else{x1 = vegdist(x[factors %in% c(co[1,elem],co[2,elem]),],method=sim.method)}

    ad = adonis(x1 ~ factors[factors %in% c(co[1,elem],co[2,elem])] );
    pairs = c(pairs,paste(co[1,elem],'vs',co[2,elem]));
    F.Model =c(F.Model,ad$aov.tab[1,4]);
    R2 = c(R2,ad$aov.tab[1,5]);
    p.value = c(p.value,ad$aov.tab[1,6])
  }
  p.adjusted = p.adjust(p.value,method=p.adjust.m)
  sig = c(rep('',length(p.adjusted)))
  sig[p.adjusted <= 0.05] <-'.'
  sig[p.adjusted <= 0.01] <-'*'
  sig[p.adjusted <= 0.001] <-'**'
  sig[p.adjusted <= 0.0001] <-'***'

  pairw.res = data.frame(pairs,F.Model,R2,p.value,p.adjusted,sig)
  print("Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1")
  return(pairw.res)

}
pairwise.adonis(comp,sp)
```

###NMDS For more explanation go to https://jonlefcheck.net/2012/10/24/nmds-tutorial-in-r/

```r
###############
# NMDS graphe #
###############
#installation des packages
#install.packages("metaMDS")
#install.packages("scatterplot3d")
library(scatterplot3d)
#library(metaMDS)
library(vegan)
```

```
## Le chargement a nécessité le package : permute

## Le chargement a nécessité le package : lattice
## This is vegan 2.5-7
```

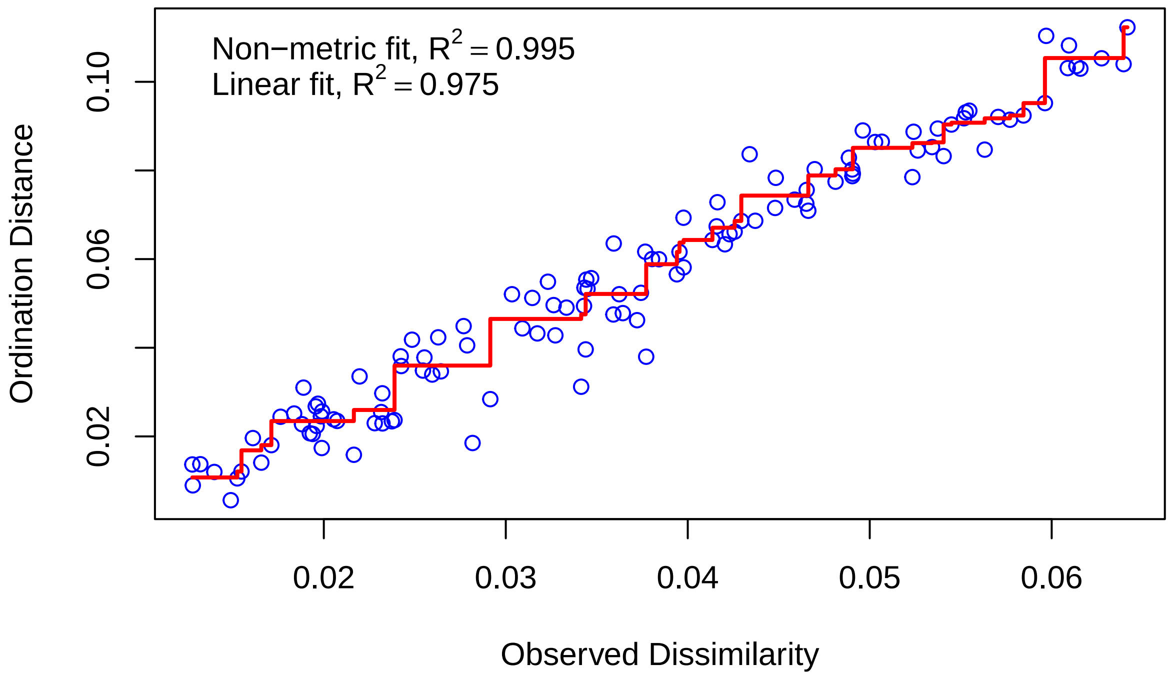
```
# Generally, stress < 0.05 provides an excellent represention in reduced
# dimensions, < 0.1 is great, < 0.2 is good, and stress > 0.3 provides a  poor representation
example_NMDS=metaMDS(sol.data[,4:13],k=2)
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.07303305
## Run 1 stress 0.08620767
## Run 2 stress 0.07836824
## Run 3 stress 0.07303305
## ... New best solution
## ... Procrustes: rmse 4.023848e-06  max resid 8.874851e-06
## ... Similar to previous best
## Run 4 stress 0.07303305
## ... Procrustes: rmse 3.110586e-06  max resid 6.589246e-06
## ... Similar to previous best
## Run 5 stress 0.1173272
## Run 6 stress 0.1096424
## Run 7 stress 0.07985861
## Run 8 stress 0.3404703
## Run 9 stress 0.1176266
## Run 10 stress 0.09005475
## Run 11 stress 0.07794391
## Run 12 stress 0.07985861
## Run 13 stress 0.07794392
## Run 14 stress 0.1086606
## Run 15 stress 0.1087585
## Run 16 stress 0.1173272
## Run 17 stress 0.0859682
## Run 18 stress 0.07379804
## Run 19 stress 0.08030324
## Run 20 stress 0.1123478
## *** Solution reached
```

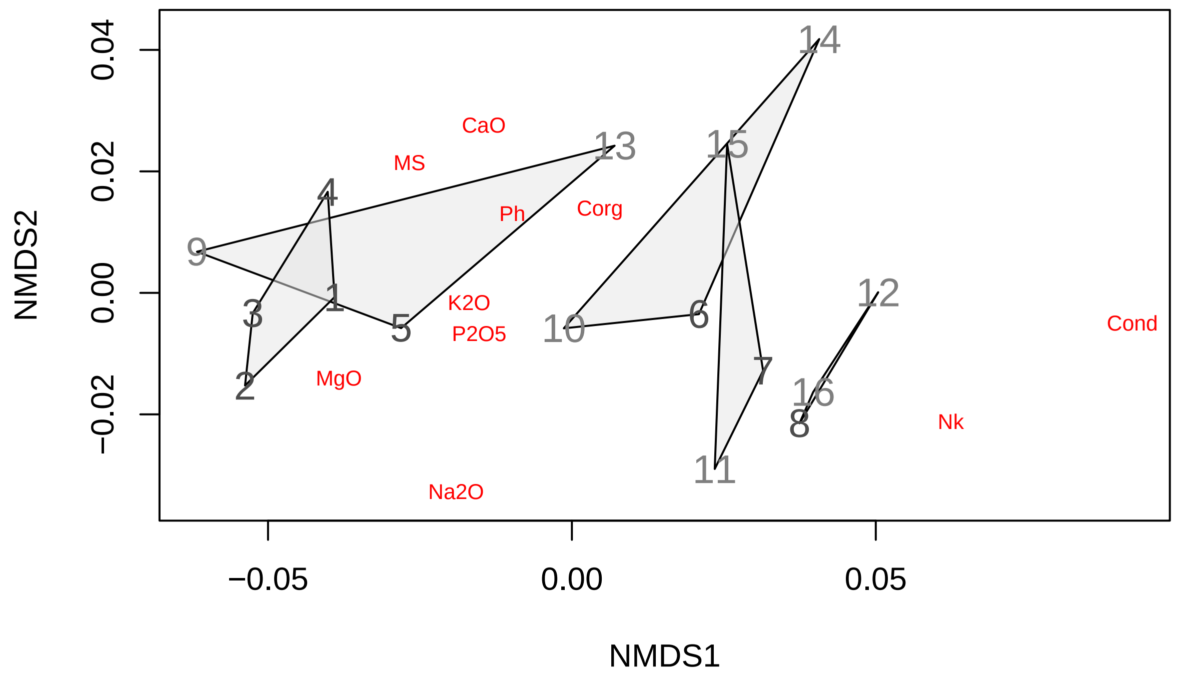
```
###if stress test rymax=100
# And we can look at the NMDS object
example_NMDS
```

```
##
## Call:
## metaMDS(comm = sol.data[, 4:13], k = 2)
##
## global Multidimensional Scaling using monoMDS
##
## Data:     wisconsin(sqrt(sol.data[, 4:13]))
## Distance: bray
##
## Dimensions: 2
## Stress:     0.07303305
## Stress type 1, weak ties
## Two convergent solutions found after 20 tries
## Scaling: centring, PC rotation, halfchange scaling
## Species: expanded scores based on 'wisconsin(sqrt(sol.data[, 4:13]))'
```

```
# Let's examine a Shepard plot,
stressplot(example_NMDS)
```



```
# Large scatter around the line suggests that original dissimilarities are not well preserved in the re
# First, let's create a vector of treatment values:
#treat=layer
ordiplot(example_NMDS,type="n")
ordihull(example_NMDS,groups=sol.data$layer,draw="polygon",col="grey90",
         label=FALSE)
orditorp(example_NMDS,display="species",col="red",air=0.01)
orditorp(example_NMDS,display="sites",col=c(rep("grey30",8),rep("grey50",8),rep("grey70",8),rep("black"
         air=0.01,cex=1.25)
```



###Indicator compounds You can use it if you want to identify the variable that explain the best the variation of the modalities. For example, you want to highlight which compounds change between the volatilome of maize plant and the volatilome of maize plant richer in silicium.

```
####predict random forest
# required packages.
#install.packages("party")
library(party)
```

```
## Warning: le package 'party' a été compilé avec la version R 4.1.3

## Le chargement a nécessité le package : modeltools

## Le chargement a nécessité le package : stats4

##
## Attachement du package : 'modeltools'

## L'objet suivant est masqué depuis 'package:plyr':
##
##     empty

## Le chargement a nécessité le package : strucchange

## Warning: le package 'strucchange' a été compilé avec la version R 4.1.3
## Le chargement a nécessité le package : zoo

##
## Attachement du package : 'zoo'

## Les objets suivants sont masqués depuis 'package:base':
##
##     as.Date, as.Date.numeric

## Le chargement a nécessité le package : sandwich
```

```
library(randomForest)
```

```
## Warning: le package 'randomForest' a été compilé avec la version R 4.1.3

## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attachement du package : 'randomForest'

## L'objet suivant est masqué depuis 'package:gridExtra':
##
##     combine

## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     margin
```

```
sol.data <- read.csv2("C://Users/Abeille/OneDrive/Assistanat/matrice-resultat-limon.csv")

# Create the forest.
#output.forest <- randomForest(rep ~ ., data = sol.data)
# View the forest results.
#output.forest
# Importance of each predictor
#importance(output.forest,type = 2)
```

12.4 PLSDA:Partial Least Squares Discriminant Analyses

The partial least squares regression works as a PCA and a multiple linear regression. It allows to highlight differences between groups. The partial least squares discriminant analyses (PLSDA) allow to classify the different studied categories. To make a long story short, it highlights predictive variable that can classify your factors. It is used in many predictive studies. It was used for example in financial studies to predict economic crisis.

Stay critical to the model generated by PLSDA. As George Edward Pelham said :"all models are wrong, but some are useful" ; "Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful".

```r
# reading
fichier<-read.table(file="C://Users/Abeille/OneDrive/Assistanat/data2.csv",sep=";",header=FALSE)
str(fichier)
```

```
## 'data.frame':    85 obs. of  222 variables:
##  $ V1  : chr  "amphet" "amphet" "amphet" "amphet" ...
##  $ V2  : chr  "Amph" "Amph" "Amph" "Amph" ...
##  $ V3  : chr  "Amph" "Amph" "Amph" "Amph" ...
##  $ V4  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V5  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V6  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V7  : chr  "0" "0" "0" "0" ...
##  $ V8  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V9  : int  339794 494846 329137 507844 107211 0 0 0 0 0 ...
##  $ V10 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V11 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V12 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V13 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V14 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V15 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V16 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V17 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V18 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V19 : chr  "0" "0" "0" "0" ...
##  $ V20 : int  1178834 1159560 961391 1200490 164494 0 0 0 0 0 ...
##  $ V21 : chr  "0" "0" "0" "0" ...
##  $ V22 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V23 : chr  "352741" "344483" "304305" "348431" ...
##  $ V24 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V25 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V26 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V27 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V28 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V29 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V30 : int  2744054 2921861 2375884 2844284 767256 178853 324369 297636 204503 187004 ...
##  $ V31 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V32 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V33 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V34 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V35 : chr  "29278" "0" "33493" "35571" ...
##  $ V36 : chr  "93283" "71355" "82629" "0" ...
##  $ V37 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V38 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V39 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V40 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V41 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V42 : int  174601 185640 145470 181191 0 0 0 0 0 0 ...
##  $ V43 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V44 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V45 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V46 : chr  "83044" "105418" "78229" "109522" ...
##  $ V47 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V48 : int  566306 620689 491764 587988 184052 233462 690454 1104707 365691 809755 ...
##  $ V49 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V50 : int  1087594 1158281 919884 1129344 258425 0 96506 119787 96174 0 ...
##  $ V51 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V52 : chr  "162409" "157719" "166003" "179446" ...
##  $ V53 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V54 : chr  "78935" "93375" "68744" "103974" ...
##  $ V55 : int  146404 167516 129960 156235 60549 0 0 0 0 0 ...
##  $ V56 : chr  "0" "0" "0" "0" ...
##  $ V57 : int  174720 256674 158410 241783 177444 0 2513482 259691 0 423372 ...
##  $ V58 : int  120257 136312 107812 130593 49494 0 0 0 0 0 ...
##  $ V59 : int  0 0 0 0 0 131577 0 165879 92181 102299 ...
##  $ V60 : chr  "0" "0" "0" "0" ...
##  $ V61 : chr  "0" "0" "0" "0" ...
##  $ V62 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V63 : chr  "0" "0" "0" "0" ...
##  $ V64 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V65 : int  1053862 1173161 938160 1083885 429380 159176 650582 721822 263535 612047 ...
##  $ V66 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V67 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V68 : chr  "0" "0" "0" "0" ...
##  $ V69 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V70 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V71 : int  35978 37505 0 36007 0 0 0 0 0 0 ...
##  $ V72 : chr  "0" "0" "0" "0" ...
##  $ V73 : int  22134 25744 0 23786 0 0 0 0 0 0 ...
##  $ V74 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V75 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V76 : chr  "0" "0" "0" "0" ...
##  $ V77 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V78 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V79 : int  864035 980902 759861 910014 334413 0 0 0 0 0 ...
##  $ V80 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V81 : chr  "0" "0" "0" "0" ...
##  $ V82 : chr  "0" "0" "0" "0" ...
##  $ V83 : chr  "0" "0" "0" "0" ...
##  $ V84 : int  10438278 11887421 9502271 10777815 5655145 1027823 2351745 2411138 1496420 0 ...
##  $ V85 : chr  "126898" "0" "0" "118036" ...
##  $ V86 : chr  "0" "0" "0" "0" ...
##  $ V87 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V88 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V89 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V90 : int  87534 146217 102143 110974 137881 0 0 0 0 0 ...
##  $ V91 : int  96118 117422 88931 100866 62270 0 0 0 0 0 ...
##  $ V92 : chr  "0" "0" "0" "0" ...
##  $ V93 : int  0 0 0 0 0 435099 615595 887380 625985 0 ...
##  $ V94 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V95 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V96 : int  304759 428293 309475 347864 352004 0 0 0 0 0 ...
##  $ V97 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ V98 : int  0 0 0 0 0 927485 1318635 0 1300335 924198 ...
##  $ V99 : int  0 0 0 0 0 0 0 0 0 0 ...
##   [list output truncated]
```

```
set.seed(1001) # to fix the algorithm whn chosing randomly the samples
```

```
# prediction matrix
y<-as.factor(fichier$V2)
dim(fichier)
X<-fichier[,4:222]

# predictive model
###install.packages('caret')
library(caret)
set.seed(1001)
ctrl<-trainControl(method="repeatedcv")

set.seed(1001)
pls<-train(x=X, # spectral data
                 y=y, # factor vector
                 method="pls", # pls-da algorithm
                 tuneLength=20, # number of components: maximum 20
                 trControl=ctrl, # ctrl contained cross-validation option
                 preProc=c("center","scale")) # metric is ROC for 2 classes. Accuracy is used for multi

pls
plot(pls)

ypred1<-predict(pls,newdata=X)
confusionMatrix(data=ypred1,reference=y)


#contribution
coefficientsAmph<-pls$finalModel$coefficients[,1,12]/pls$preProcess$std #12 because ncomp=12

coefficientscoc<-pls$finalModel$coefficients[,2,12]/pls$preProcess$std #12 because ncomp=12
coefficientsEcsta<-pls$finalModel$coefficients[,3,12]/pls$preProcess$std #12 because ncomp=12
coefficientshash<-pls$finalModel$coefficients[,4,12]/pls$preProcess$std #12 because ncomp=12
coefficientshero<-pls$finalModel$coefficients[,5,12]/pls$preProcess$std #12 because ncomp=12
coefficientsmary<-pls$finalModel$coefficients[,6,12]/pls$preProcess$std #12 because ncomp=12
coefficientsMDMA<-pls$finalModel$coefficients[,7,12]/pls$preProcess$std #12 because ncomp=12
coefficients<-data.frame(coefficientsAmph,coefficientscoc,coefficientsEcsta,coefficientshash,coefficien
write.table(coefficients,file="coefficients_models.csv",sep=';')

plot(coefficientsAmph,type="p",col="red")
plot(coefficientscoc,type="p",col="blue")
lines(coefficientsEcsta,type="p",col="green")
lines(coefficientshash,type="p",col="black")
lines(coefficientshero,type="p",col="orange")
lines(coefficientsmary,type="p",col="yellow")
lines(coefficientsMDMA,type="p",col="pink")

vipAmph<-order(abs(coefficientsAmph),decreasing = TRUE)
coefficientsfinalAmph<-coefficientsAmph[vipAmph]#pour ne pas avoir les numero de colonne mais de compos
vipcoc<-order(abs(coefficientscoc),decreasing = TRUE)
vipEcsta<-order(abs(coefficientsEcsta),decreasing = TRUE)
viphash<-order(abs(coefficientshash),decreasing = TRUE)
```

```
vipphero<-order(abs(coefficientshero),decreasing = TRUE)
vipmary<-order(abs(coefficientsmary),decreasing = TRUE)
vipMDMA<-order(abs(coefficientsMDMA),decreasing = TRUE)

coefficientsAmph[70]
coefficientsAmph[84]
vipAmph
```

12.5 Coinertia Analysis

The COIA (coinertia analyses) allow to compare to PCA that aim to evaluate the same event with different variables. For example, if you want to determine the link between environments, plants and insect diversity, COIA could be helpful to establish which insects is dependent of which plant in each environment. To do so, first, you measure in each environment the diversity and amount of plant and insect. Then you create a matrix which record for the different environment the plant that you find in it. A second matrix have to be done for the insect. The next step is the creation of an PCA for each matrix. You therefore obtain a PCA which draw the different groups of insects found depending on the environment and the same PCA for the plants recorded in each environment. You can use the previously presented code for PCA. These PCAs allow you to determine if there is a different of plant or insect diversity among the different environments that you have to study. The final step is to identify if there are plants that drive the presence of some insects species. To answer this question, you have to do a COIA on your to PCA. The COIA is like a new PCA based on the two previous PCAs. The COIA will allow to determine which axes/dimensions of your first PCAs is correlated with which axes/dimension of your second PCA. If a correlation is suspected between two axes/dimensions, you have to perform a correlation test between the suspected correlated axes/dimensions with a classic correlation test. Finally, If the axes are well correlated (p-value < 0.05), it means that some insect's variables are correlated with some plant'variables. To be sure you test each plant variables with each insects' variables from the correlated axes to identified the correlated variables.

```
#note
PCAnote_COIA<-na.omit(PCAnote_COIA)
View(PCAnote_COIA)
View(PCAexpe_session3)

PCAnote_COIA<-read.table(file="EV_MC_S2_sansVICTOR.csv",sep=";",dec = ".",header=TRUE)
rownames(PCAnote_COIA) = PCAnote_COIA[,1]
View(PCAnote_COIA)

dudinote <- dudi.pca(PCAnote_COIA[-(1:1)], scale = TRUE, scan = FALSE, nf = 2)

fviz_pca_ind(dudinote, axes = c(1, 2), geom = c("point", "text"),
             label = "all", invisible = "none", labelsize = 4,
             pointsize = 2, habillage = "none",
             addEllipses = FALSE, ellipse.level = 0.95,
             col.ind = "black", title="NotePCA")
fviz_pca_var(dudinote,axes = c(1, 2),
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, title="Notecorrelation"     # Avoid text overlapping
)
#variable contribution
## PC1
fviz_contrib(dudinote, choice = "var", axes = 1, top = 20)
```

```r
## PC2
fviz_contrib(dudinote, choice = "var", axes = 2, top = 20)

#data
PCAexpe_session<-read.table(file="Data_S2_sansVICTOR.csv",sep=";",dec = ".",header=TRUE)

rownames(PCAexpe_session) = PCAexpe_session[,1]
View(PCAexpe_session)

dudiexpe <- dudi.pca(PCAexpe_session[-(1:1)], scale = TRUE, scan = FALSE, nf = 2)

fviz_pca_ind(dudiexpe, axes = c(1, 2), geom = c("point", "text"),
             label = "all", invisible = "none", labelsize = 4,
             pointsize = 2, habillage = "none",
             addEllipses = FALSE, ellipse.level = 0.95,
             col.ind = "black", title="expePCA")
fviz_pca_var(dudiexpe,axes = c(1, 2),
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, title="expecorrelation"     # Avoid text overlapping
)
#variable contribution
fviz_contrib(dudiexpe, choice = "var", axes = 1, top = 20)
fviz_contrib(dudiexpe, choice = "var", axes = 2, top = 20)

#coinertie
coin3 <- coinertia(dudinote,dudiexpe, scan = FALSE, nf = 2)
coin3
summary(coin3)
plot(coin3)
```

12.6 Hierarchical clustering analysis

Multivariate statistics occur when you have more than one measured variable for one observation. Multivariate analysis are useful to observe a particular pattern in your dataset especially when you have a lot of measured variables and many observations. One family of these multivariate analysis is the ordination method where the data are displayed in projecting them in new coordinates systems in conserving at best the original structure. Mainly, it permits to observe the differences gradients and the objects that are the most different. It could be also revealed the particular correlations among original variables.

{this part is still in progress}

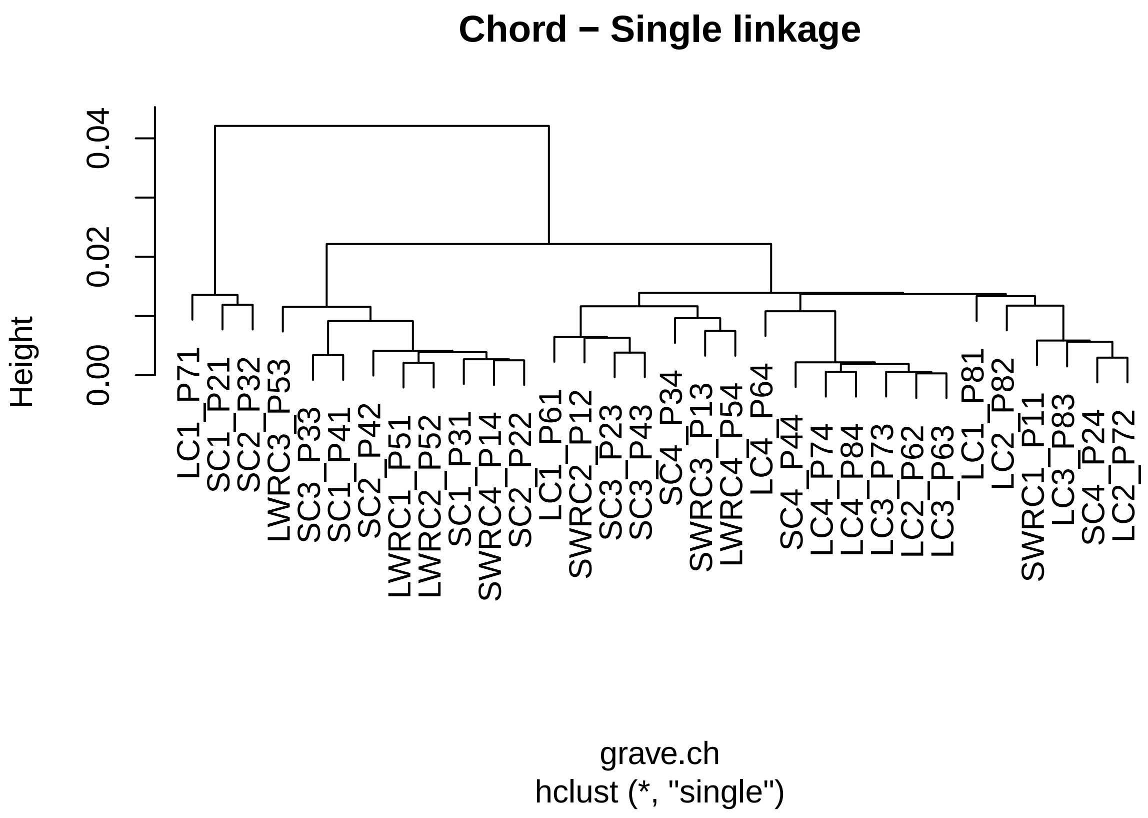
12.6.1 Dataset description & R code

We used a dataset called *graveyard* which correpsond to chemical analysis from Clement Martin research on dead body of rats in different kind of soil : loamy and sandy soils.

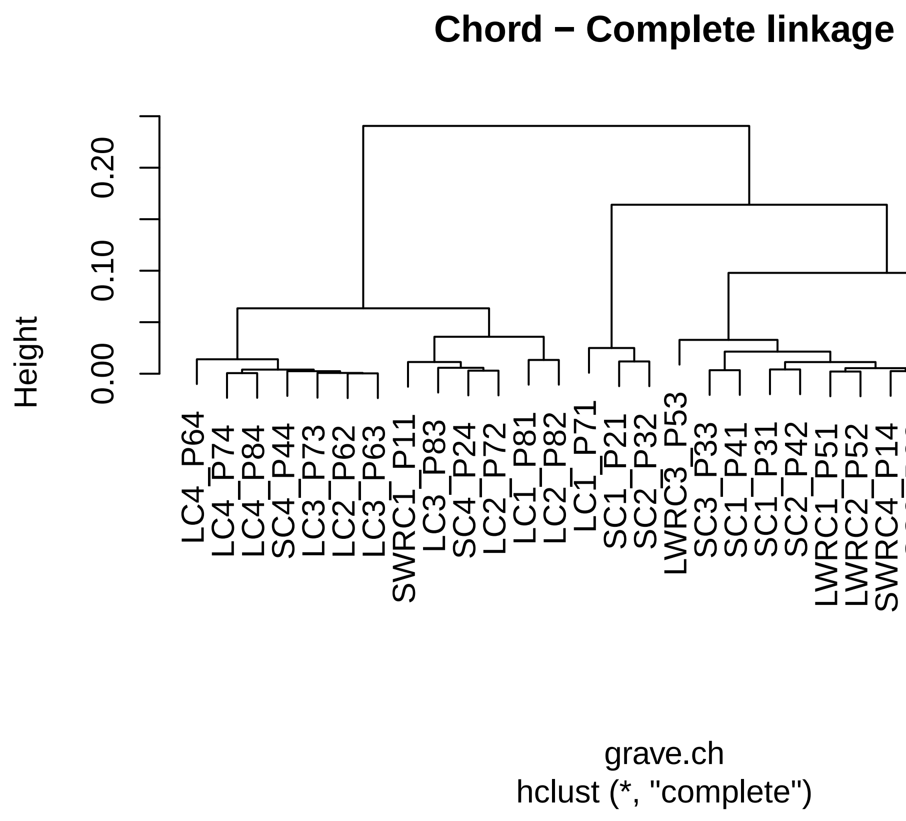
```r
#Ward's method, Euclidean distance for chemical concentration (Michela ROGORA 2008)

#For multivariate analysis

library(vegan)
library(ade4)
```

```
##
## Attachement du package : 'ade4'

## L'objet suivant est masqué depuis 'package:FactoMineR':
##
##     reconst
```

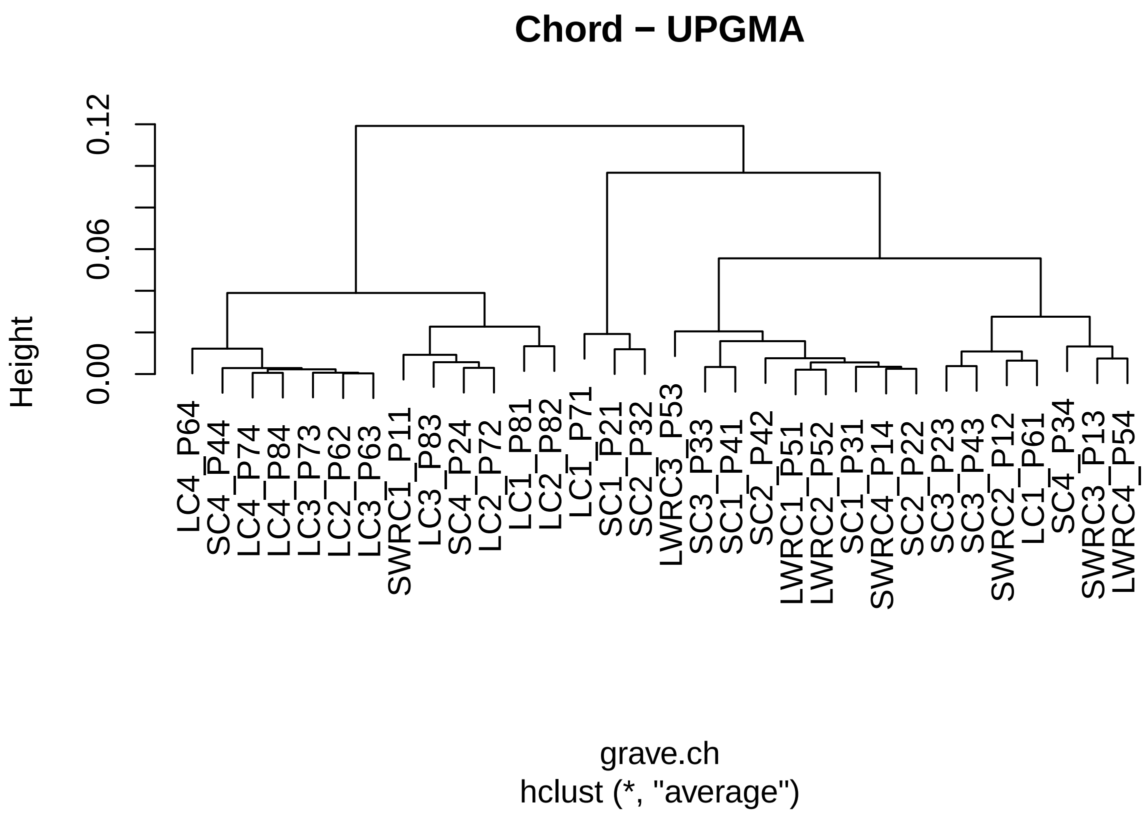
```
graveyard<-read.csv2("C://Users/Abeille/OneDrive/Assistanat/matrice de resultats-tt.csv")

#graveyards name
name.graveyard<-graveyard[,3]

Graveyard<-graveyard[,-(1:2)]

# Compute matrix of chord distance among sites
grave.norm <- decostand(Graveyard[,-1], "normalize")
grave.ch <- vegdist(grave.norm, "euc")
# Attach site names to object of class 'dist'
attr(grave.ch, "labels") <- name.graveyard
# Compute single linkage agglomerative clustering
spe.grave.single <- hclust(grave.ch, method = "single")
# Plot a dendrogram using the default options
plot(spe.grave.single,
     labels = name.graveyard,
     main = "Chord - Single linkage")
```



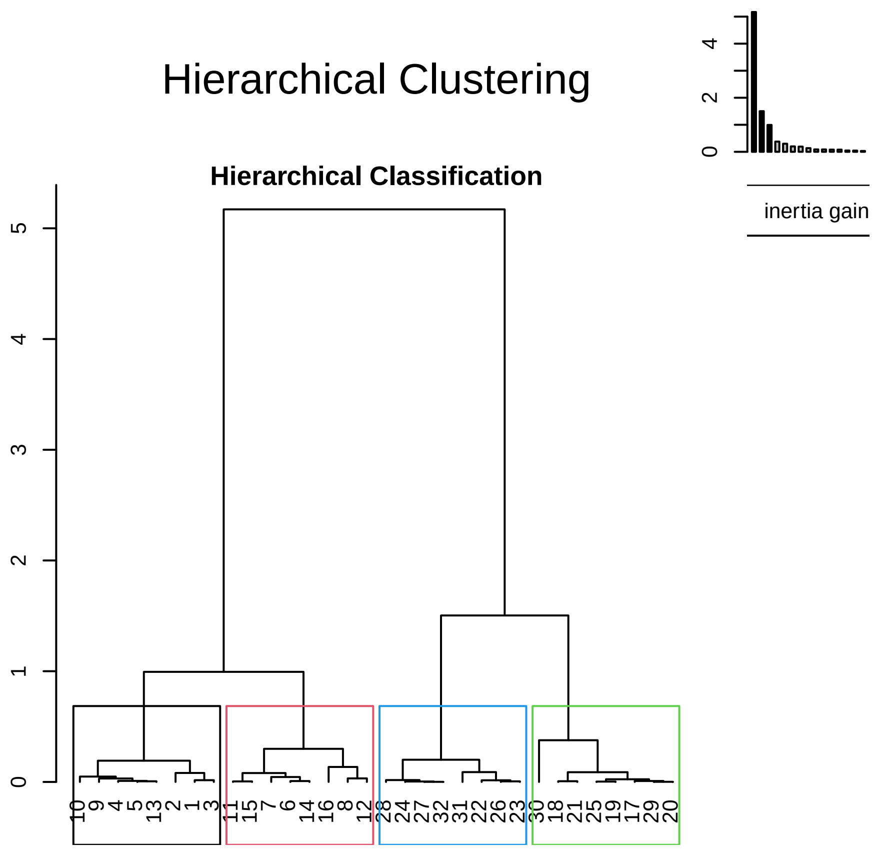
```r
# Compute complete-linkage agglomerative clustering
spe.grave.complete <- hclust(grave.ch, method = "complete")
plot(spe.grave.complete,
     labels = name.graveyard,
     main = "Chord - Complete linkage")
```



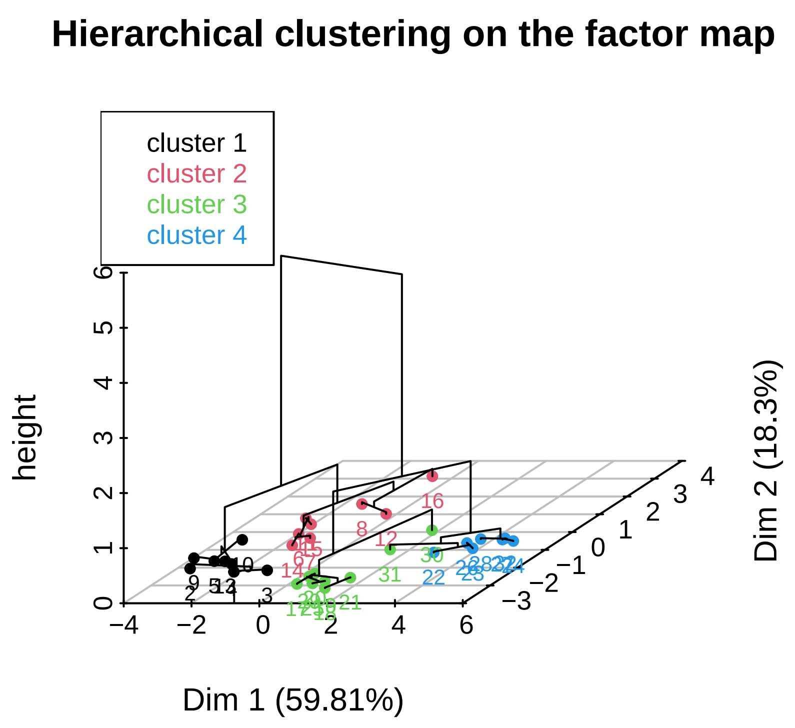
```
# Compute UPGMA agglomerative clustering
spe.grave.UPGMA <- hclust(grave.ch, method = "average")
plot(spe.grave.UPGMA,
     labels = name.graveyard,
     main = "Chord - UPGMA")
```

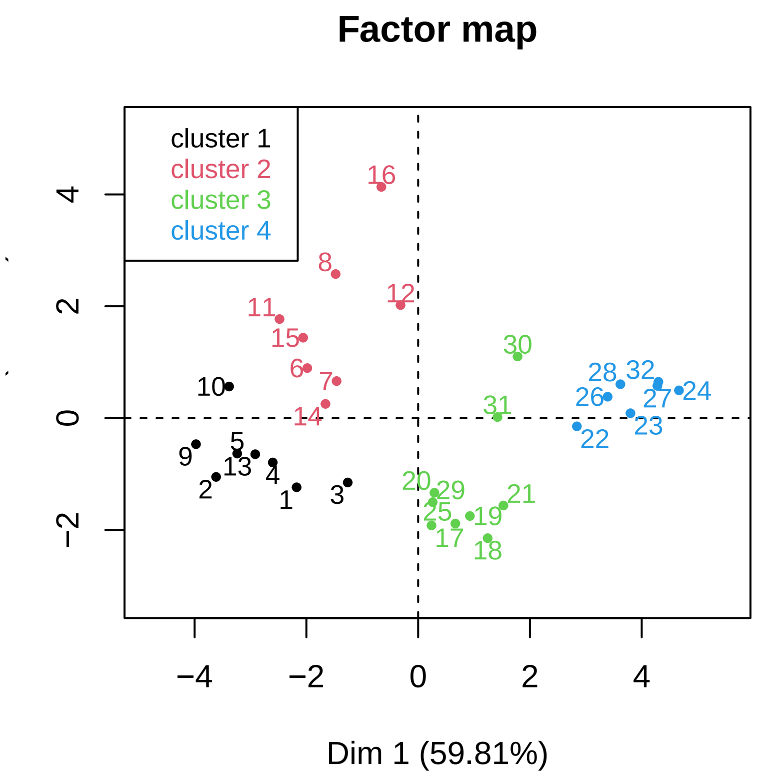


```
##Other test

library(FactoMineR)
library(factoextra)
library(ggplot2)


res.PCA<-PCA(Graveyard[,-1],ncp=3,graph=F)
res.hcpc<-HCPC(res.PCA,nb.clust=-1,graph = T)
```



Hierarchical Clustering
Hierarchical Classification
inertia gain
0
1
2
3
4
5
10 9 4 5 13 2 1 3 11 15 7 6 14 16 8 12 28 24 27 32 31 22 26 23 30 18 21 25 19 17 29 20


Hierarchical clustering on the factor map
cluster 1
cluster 2
cluster 3
cluster 4
height
Dim 1 (59.81%)
Dim 2 (18.3%)



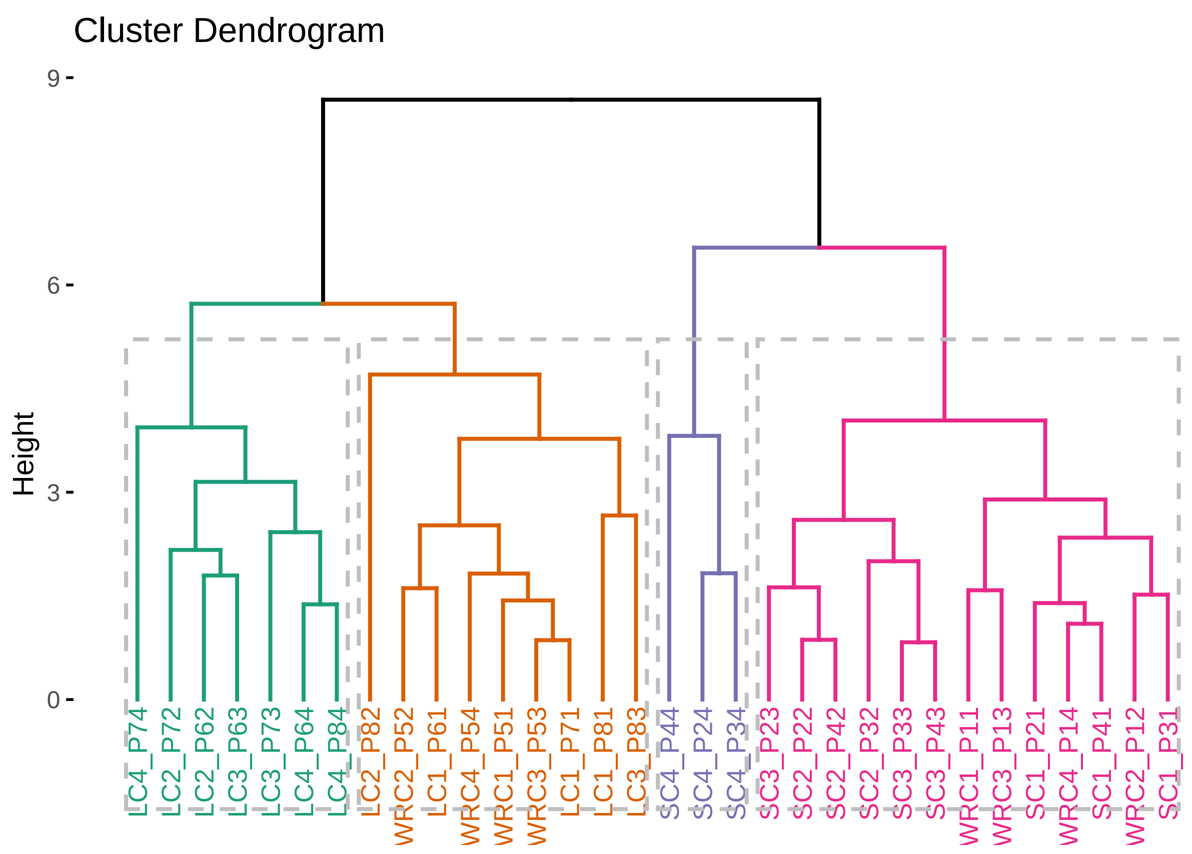
```
summary(res.hcpc)
```

```
##            Length Class      Mode
## data.clust 12     data.frame list
## desc.var    3     catdes     list
## desc.axes   3     catdes     list
## desc.ind    2     -none-     list
## call        8     -none-     list
```

```
df <- scale(Graveyard[,-1])

# Hierarchical clustering
res.hc <- hclust(dist(df))
res.hc$labels<-as.character(name.graveyard)

fviz_dend(res.hc,cex = 0.7, k = 4, color_labels_by_k = TRUE, rect = T,k_colors = c("#1B9E77", "#D95F02"
```

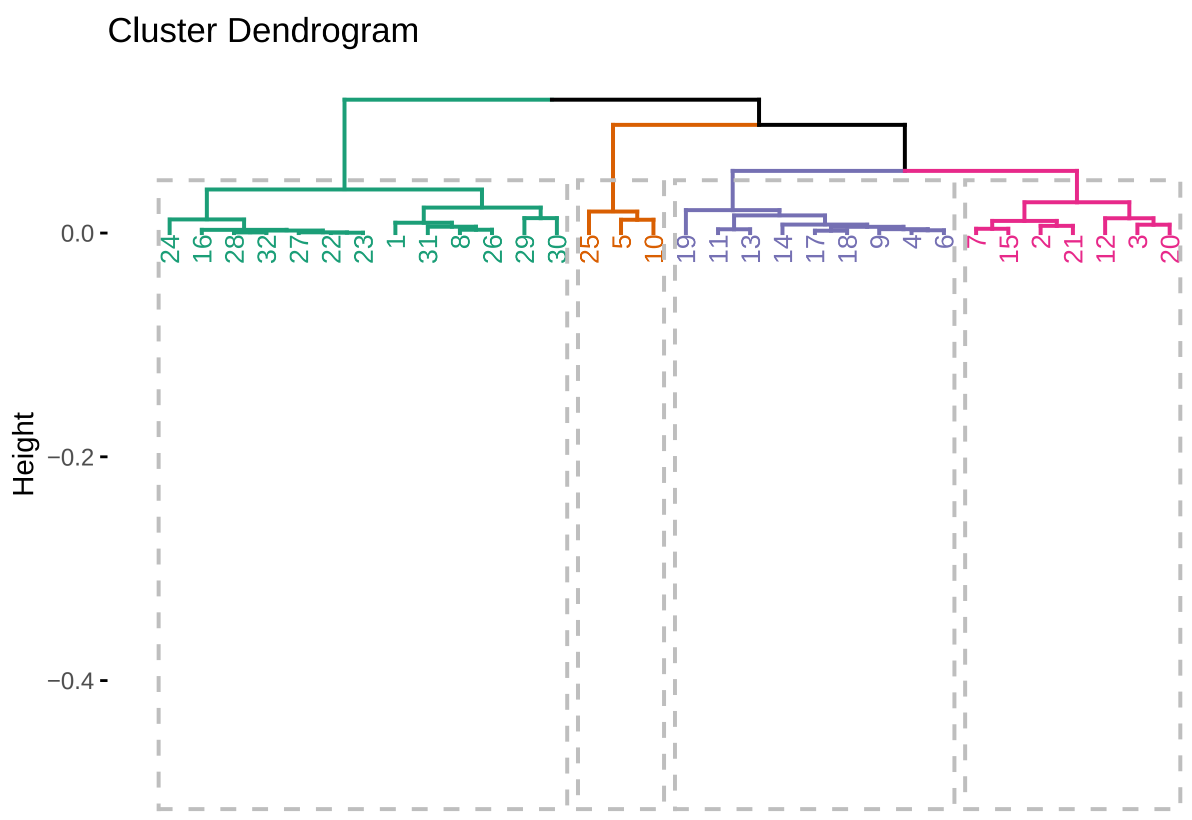
```
## Registered S3 method overwritten by 'dendextend':
##   method     from
##   rev.hclust vegan
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

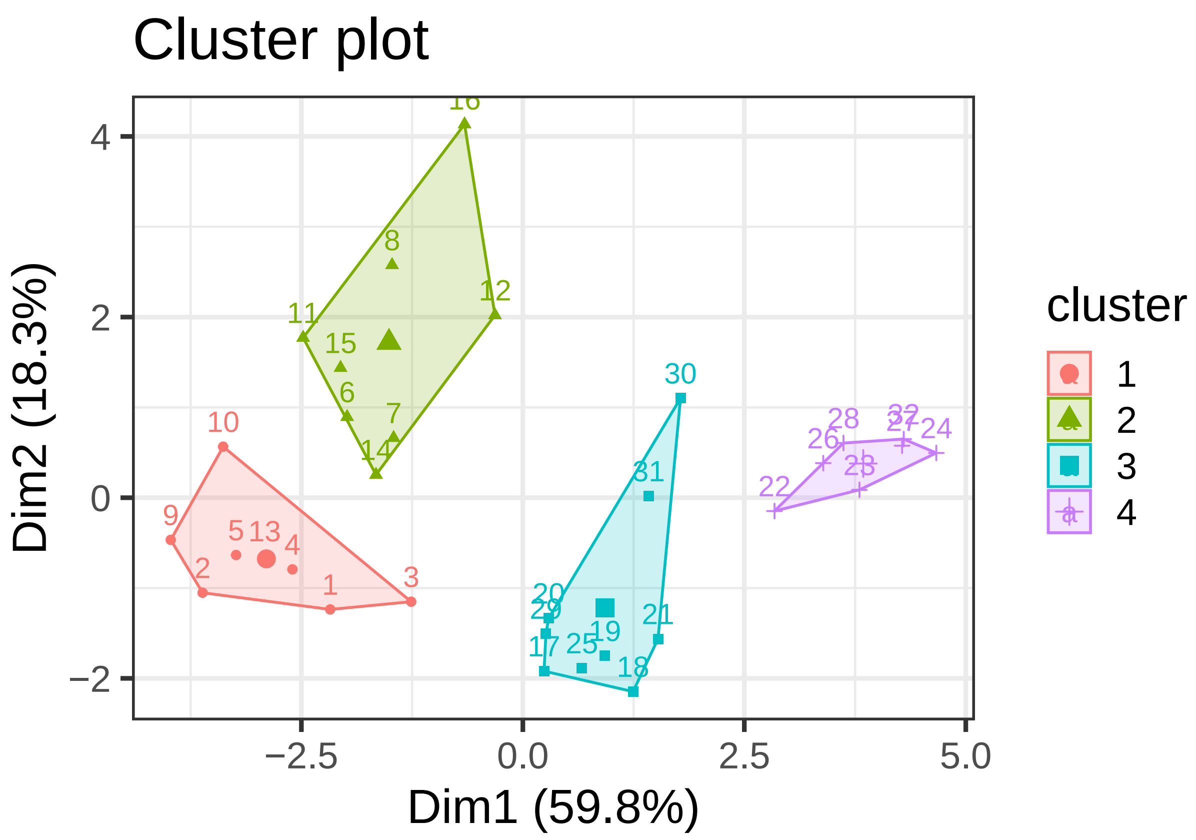


```
fviz_dend(spe.grave.UPGMA,cex = 0.7, k = 4, color_labels_by_k = TRUE, rect = T,k_colors = c("#1B9E77",
```

```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```



```
g<-fviz_cluster(res.hcpc)
g + theme_bw(base_size = 18)
```



#More documentation This workshop is based on our own data but also on theory from the book

For data manipulation and basics statistical analysis, you can also find some supplementary information on these following websites :

http://www.cookbook-r.com/

http://www.sthda.com/english/

For Graphics documentation:

http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf

http://pbil.univ-lyon1.fr/R/pdf/tdr75.pdf

https://www.r-graph-gallery.com/