R Stats Tuto

Pierre-Alexandre

2023-10-11

## *R Markdown*

*This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see* [*http://rmarkdown.rstudio.com*](http://rmarkdown.rstudio.com) *or the YouTube tutorial (in french)* [*https://www.youtube.com/watch?v=lFdB4fIAcLM*](https://www.youtube.com/watch?v=lFdB4fIAcLM)

*A cheat-sheet can be find here:* [*https://rstudio.github.io/cheatsheets/html/rmarkdown.html?\_gl=1\*a65iq2\*\_ga\*MTk3MDA5NzQ0MS4xNjk3MDYyMzE3\*\_ga\_2C0WZ1JHG0\*MTY5NzA2MjMxNi4xLjEuMTY5NzA2MzAzNy4wLjAuMA..*](https://rstudio.github.io/cheatsheets/html/rmarkdown.html?_gl=1*a65iq2*_ga*MTk3MDA5NzQ0MS4xNjk3MDYyMzE3*_ga_2C0WZ1JHG0*MTY5NzA2MjMxNi4xLjEuMTY5NzA2MzAzNy4wLjAuMA..)

## *Preparation of the work space*

### *R & RStudio install*

*R & the IDE RStudio can be install from this web page :* [*https://posit.co/download/rstudio-desktop/*](https://posit.co/download/rstudio-desktop/)

1. *Download then Install R*
2. *Download then Install RStudio*

*Cheat-sheet of RStudio :* [*https://rstudio.github.io/cheatsheets/html/rstudio-ide.html?\_gl=1\*1tpycou\*\_ga\*MTk3MDA5NzQ0MS4xNjk3MDYyMzE3\*\_ga\_2C0WZ1JHG0\*MTY5NzA2MjMxNi4xLjEuMTY5NzA2MzAzNy4wLjAuMA..*](https://rstudio.github.io/cheatsheets/html/rstudio-ide.html?_gl=1*1tpycou*_ga*MTk3MDA5NzQ0MS4xNjk3MDYyMzE3*_ga_2C0WZ1JHG0*MTY5NzA2MjMxNi4xLjEuMTY5NzA2MzAzNy4wLjAuMA..)

### *Install package:*

*Packages can be install with the instruction install.packages() (Don’t forget the" "):*  
*Example for the ggplot2 package:*

# Install from CRAN  
install.packages("ggplot2")

*To see all the packages which are installed use the code below:*

# Installed packages  
installed.packages()

**Use a package**

After the installation of the desired package, it is neceesary to upload it in R thanks to library command:

library("ggplot2", "tibble")

An other way consist to put all the library desired in a variable thank to a vector (see below), then upload it with the function lapply as it is showed below:

x <- c("readr","tibble","tidyr","ggplot2","dplyr","gridExtra", "pracma", "factoextra", "FactoMineR", "Cairo", "Rtsne",   
 "colorspace","dendextend", "RColorBrewer", "ggthemes","ggpubr", "readxl", "pspline", "randomForest", "lattice",  
 "caret", "LiblineaR")  
lapply(x,require,character.only = T)

# The basic

Good resources: <https://bookdown.org/ael/rexplor/>

## Assignation of a variable:

In R, the variable can be assigned with two symbol <- OR =. But the first one is the most used.  
To compile a “list” of variable, it is needed to create a vector thanks to c() with , for the separation.  
Then thanks to the funtion class you can see the class of the variable.

x <- 20  
l <- c(10, 44, 89)  
s <- "I am a character"  
bo <- TRUE # a booleen  
  
paste("Variable x:",x)

## [1] "Variable x: 20"

class(x)

## [1] "numeric"

paste("Variable l:",l)

## [1] "Variable l: 10" "Variable l: 44" "Variable l: 89"

class(l)

## [1] "numeric"

paste("Variable s:", s)

## [1] "Variable s: I am a character"

class(s)

## [1] "character"

paste("Variable s:", bo)

## [1] "Variable s: TRUE"

class(bo)

## [1] "logical"

*Remarks*  
The utilization of the vector c() do not create a real list but a **vector**.  
For a list, it is needed to call the function list:

li <- list(10, 44, 89)  
li

## [[1]]  
## [1] 10  
##   
## [[2]]  
## [1] 44  
##   
## [[3]]  
## [1] 89

class(li)

## [1] "list"

The vector can not create missed value because all that is useless is not created. Instead of, the list permit the creation with null value:

vector <- c(1, 5, NULL)  
liste <- list(1, 5, "u", NULL)  
vector

## [1] 1 5

liste

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] 5  
##   
## [[3]]  
## [1] "u"  
##   
## [[4]]  
## NULL

The command paste() is used to concatenate string and variable. A space is automatically include as separator. To change it, add the argument , sep="" with no space or other symbol. It is possible to used also paste0.

For any help, use the command help()or the symbol ?. Example:

help(class)  
?ggplot2

To remove an object: rm(). At the begining of a session/project, it can be usefull to clean all the data which was kept by RSudio in its memory:

rm(list = ls())

## Dataset Manipulation

### Importation of csv file

csv files are directly import as a data.frame. A data set can be displayed thanks to the command View.

smp <- read.csv2("~/code/R/FUN/fichier 1/smp1.csv")  
View(smp) # With upper V

### Some basic manipulations

To see the 6 first observations for the all variables, use the command head(). Use $ to access to a variable of a data set:

# Utilisation of the Iris data set, directly included in R  
class(iris)

## [1] "data.frame"

head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

head(iris$Sepal.Length)

## [1] 5.1 4.9 4.7 4.6 5.0 5.4

The command whichgive the observation who respect the condition:

* == for an equivalence
* !> for a difference
* <= / >= less/greater than or equal to
* </ > strictly less/greater than
* & / | AND / OR
* isTRUE(x) test if X is TRUE
* na.rm=TRUE to remove the empty values

which(iris$Species != "setosa")

## [1] 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68  
## [19] 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86  
## [37] 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104  
## [55] 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122  
## [73] 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140  
## [91] 141 142 143 144 145 146 147 148 149 150

The command table group and count each observation for a given variable. The command subset(data.frame, condition, c(vector of variable)), take the observation who respect the condition and keep only the variables asked.

table(iris$Species)

##   
## setosa versicolor virginica   
## 50 50 50

table(iris$Species != "setosa")

##   
## FALSE TRUE   
## 50 100

table(subset(iris, Species == "setosa")$Petal.Length > 1.4)

##   
## FALSE TRUE   
## 24 26

head(subset(iris, Species == "setosa", c(Sepal.Length, Petal.Length)))

## Sepal.Length Petal.Length  
## 1 5.1 1.4  
## 2 4.9 1.4  
## 3 4.7 1.3  
## 4 4.6 1.5  
## 5 5.0 1.4  
## 6 5.4 1.7

### Make a *tibble* instead of *data.frame*

Upload the library tibble. This one allowed the creation of a *tibble* (table) that it has the same purpose than a *data.frame* but with more restriction. The *tibble* must to have the same size for each column.

library(tibble)  
# creation of the dataset  
# m1 a list with Null values  
m1<-list(1.311,1.287,1.293,1.308,1.291,1.300,1.274,1.287)  
m1 <- append(m1, vector("list",5)) # append 5 element NULL contain in a list  
m2<-c(1.298,1.309,1.293,1.251,1.338,1.302,1.270,1.339,1.346,1.292,1.291,1.321,1.285)  
Si<-tibble(method1=m1,method2=m2)

**Use the functional programming**

The functional programming is a paradigm of building computer program with a declarative type. In this case, the programs are constructed by applying and composing functions. In R the tibble library (and other *tixxx*) allow this kind programming with the following construction variable %>% function(). The functional is a better way to write a comprehensive workflow:

# recover a list containing the values of method 1 padded with NULL values then converted to a vector, without NULL values  
m1<-Si$method1 %>% unlist()  
m1

## [1] 1.311 1.287 1.293 1.308 1.291 1.300 1.274 1.287

class(m1)

## [1] "numeric"

### Manipulation

The libraries tidyr & dplyr are useful to manipulate a data set.

For example, in the first table, it is supposed that each samples have been measured with the both two methods:

* sample 1 : value m1 / value m2
* sample 2 : value m1 / value m2
* …

In this case, measurements have been performed independently of the samples. So the first table does not work. Here it will be better that each value receive tag corresponding to its origin (method1 or method2). The code below propose this kind of table:

library(dplyr, tidyr)

##   
## Attachement du package : 'dplyr'

## Les objets suivants sont masqués depuis 'package:stats':  
##   
## filter, lag

## Les objets suivants sont masqués depuis 'package:base':  
##   
## intersect, setdiff, setequal, union

Si2<-tibble(method=c(rep("method1",8),rep("method2",13)),values=c(m1,m2))  
  
# Let's simulate a randomized acquisition of the dataset  
Si2<-Si2[sample(nrow(Si2)),] # Indexes are generated by random pick, the tibble is shuffled  
head(Si2)

## # A tibble: 6 × 2  
## method values  
## <chr> <dbl>  
## 1 method1 1.29  
## 2 method2 1.32  
## 3 method2 1.29  
## 4 method2 1.30  
## 5 method2 1.29  
## 6 method1 1.29

**Play with tidyr & dplyr libraries and functional programming**

m1b<-Si2 %>% filter(method == "method1") %>% select(values) %>% unlist()  
m1

## [1] 1.311 1.287 1.293 1.308 1.291 1.300 1.274 1.287

m1b

## values1 values2 values3 values4 values5 values6 values7 values8   
## 1.291 1.287 1.287 1.293 1.308 1.300 1.311 1.274

Cheat-sheet for more information on those libraries:

<https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf>

## Statistics operations

A rapid view of the global statistic descriptor with the function summary:

summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

*Note: iris is a dataset which is already include in R*

### *Basic operations:*

sum(iris$Sepal.Length) # Addition of all the observation in the variable

## [1] 876.5

mean(iris$Sepal.Length) # Average

## [1] 5.843333

sd(iris$Sepal.Length) # Standard deviation

## [1] 0.8280661

q <- quantile(iris$Sepal.Length) # Catch all the quantiles   
q

## 0% 25% 50% 75% 100%   
## 4.3 5.1 5.8 6.4 7.9

q[1]

## 0%   
## 4.3

length(iris$Sepal.Length) # Number of observation in the variable

## [1] 150

### *Confidence interval*

The confidence interval at 95% : with:

* the standard error ==> the standard deviation of the mean
* with the number of observations
* The correspond to a quantile of a normal student law for a confidence of 95% given by
* R can calculate with: qnorm(y) & qnorm(1-y)

y <- (1-0.95)/2  
y

## [1] 0.025

signif(qnorm(y),3)

## [1] -1.96

signif(qnorm(1-y),3)

## [1] 1.96

*The standard error is the standard deviation of the mean of distributions and the standard deviation is for the individual distribution of a normal law.*

x <- iris$Sepal.Length  
x\_N<-length(x)  
x\_sd<-sd(x)  
x\_mu<-mean(x)  
x\_se<-x\_sd/sqrt(x\_N)  
x\_ICmin<-x\_mu+x\_se\*qnorm(0.025)  
x\_ICmax<-x\_mu+x\_se\*qnorm(0.975)  
paste(signif(x\_ICmin,3),"<",signif(x\_mu,3),"<",signif(x\_ICmax,3))

## [1] "5.71 < 5.84 < 5.98"

The command signif(variable, number) print the variable with the number of significant value asked.

# Evaluation of probability law

With R, it is easy to evaluate a probability law:

* **r**mylaw ==> Simulate mylaw
* **d**mylaw ==> Calculate the density of mylaw
* **p**mylaw ==> Calculate the cumulative of density function (CDF) of mylaw
* **q**mylaw ==> Calculate the quantile of mylaw

Below the 3 mains laws:

## *Normal Law (Gauss Law)*

my\_mu<-20 # mean value of the normal law  
my\_sg<-sqrt(20) # std dev of the normal law  
dnorm(20,mean=my\_mu,sd=my\_sg) # point evaluation of the density at x=20 of a normal law centered at my\_mu and width sigma=my\_sg

## [1] 0.08920621

pnorm(20,mean=my\_mu,sd=my\_sg) # point evaluation of the CDF at q=20 of this same normal distribution

## [1] 0.5

qnorm(0.05,mean=my\_mu,sd=my\_sg) # quantile of this normal distribution considering a probability p=0.05

## [1] 12.64399

rnorm(10,mean=my\_mu,sd=my\_sg) # generate 10 random numbers following this normal distribution

## [1] 15.68801 21.18071 23.39987 17.25429 16.67350 21.04305 24.58422 12.39827  
## [9] 21.86074 22.85245

alpha=0.0001 # kind of "type I risk", the part of the plot I want to hide on the left/right sides.  
xmin=qnorm(alpha,mean=my\_mu,sd=my\_sg) # left value of the range to plot the distribution  
xmax=qnorm(1-alpha,mean=my\_mu,sd=my\_sg) # right value of the range to plot the distribution

## *Binomial law*

The Binary law can be positive or negative, but in the general cases, it is only the positive which is used. This is the law which is followed by a randomization selection when there are **ONLY TWO** possibility of results: A or B, alive or died. The probability to obtained each evenement must be constant (but not necessary equal). The probability to obtain the A event will be and B will be . So, with selection, the probability to obtain k event of A will follow the binomial law . The mean correspond to the expectation (*espérance in french*) and will be give by . the variance is given by .

n<-30 # population  
p<-0.1 # probability of success  
dbinom(x=1,size=n,prob=p) # Probability that I win exactly 1 time in 30 plays

## [1] 0.1413039

pbinom(q=2,size=n,prob=p) # CDF, probability to win 0,1 or 2 times in 30 plays

## [1] 0.4113512

qbinom(p=0.5,size=n,prob=p) # Quantile of 0.5 from the binomial law

## [1] 3

rbinom(n=10,size=n,prob=p) # Sample of size 10 from the binomial law

## [1] 2 6 2 4 1 1 1 3 3 2

The formula of the binomial law is:

Where is the binomial coefficient (n choose k) and can be calculate with the command choose(n,k) or by

## *Poisson law*

The Poisson law is used to described the probability of a rare event in an interval of time. It is an approximation of the binomial law when the when tends to zero (very law probability) and tends to (a lot of repetition). This law take the parameter and traduce the the expectation and the variance (which is logical since in the binomial law). That’s why the Poisson law is define by and represent the mean of the probability where the rare event is realized.

n<-30 # number of observation  
m <- 20 # Poisson law parameter  
dpois(x=1, lambda =m ) #

## [1] 4.122307e-08

ppois(q=2, lambda =m ) #

## [1] 4.55515e-07

qpois(p=0.5, lambda =m ) #

## [1] 20

rpois(n=n, lambda =m ) #

## [1] 16 30 18 21 25 23 14 21 18 24 23 14 17 16 15 12 21 17 14 20 29 29 18 27 14  
## [26] 19 19 21 16 21

The formaula of the Poisson law is:

*Note: factorial calculation in R:*

factorial(5)

## [1] 120

# Graphics

Some graphics can be use in R without package. But they are limited and don’t have a lot possibility to manage them. Those graphs can be useful for a first and fast exploration of the data but for a presentation, it must be interested to create more complex/beautful graphs. That why it is better to work with the package ggplot2.

First of all, install the package ggplot2 then upload it in R thanks to library command:

library(ggplot2)

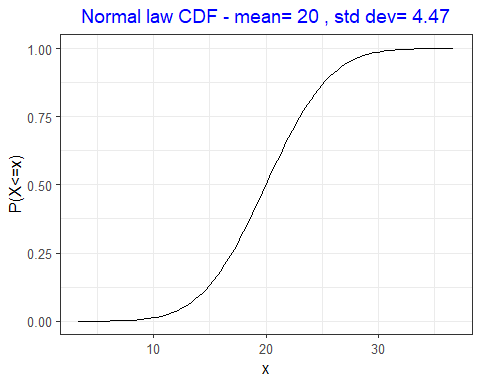
ggplot2 is defined at least by a **data set**, some **aes,** which represent the “aesthetic” of the graphic, and the type of the graphic thanks to the function **geom.**. The **aes** is defined at least by a **x** and a **y** values. It’s possible to add also the **color**, the **fill**, the **group**. Some example will be showed below.

All the addition of function is add thanks the **+**

For more information: <https://bookdown.org/ael/rexplor/chap8.html> (in French)

Graphic can caught in a variable but it is recommended to have one and unique block per graphic:

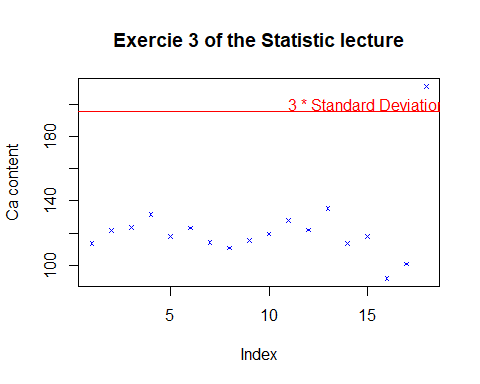
# A beautiful plot of this normal law CDF  
ggplot()+ # start plot  
 xlim(xmin,xmax)+ # define the range to plot  
 stat\_function(fun=pnorm,args=c(mean=my\_mu,sd=my\_sg))+ # define the normal law density  
 xlab("x")+ # decoration of the x axis  
 ylab("P(X<=x)")+ # decoration of the y axis  
 labs(title=paste("Normal law CDF - mean=",my\_mu,", std dev=",signif(my\_sg,3)))+ # title of the plot  
 theme\_bw(base\_size=12)+ # ...the final touch  
 theme(plot.title = element\_text(hjust = 0.5, color = "blue"))



Here stat\_function(fun=pnorm,args=c(mean=my\_mu,sd=my\_sg) is used to draw a function (*the CDF of a normal law in this case*).

## The point\_plot:

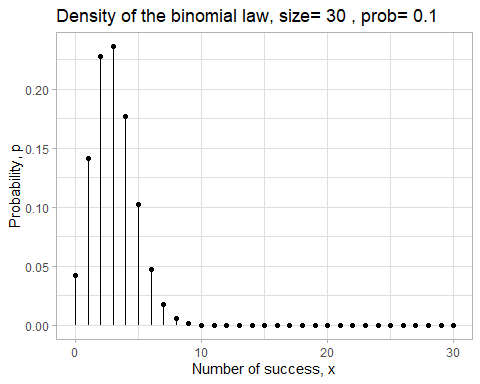
plot(Ca,   
 type = "p", # type de tracé: points ("p"), lignes ("l"), les deux ("b" ou "o"),  
 col = "blue", # couleur, tapez `colours()` pour la liste complète  
 pch = 4, # type de symboles, un chiffre entre 0 et 25, tapez `?points`  
 cex = 0.5, # taille des symboles  
 lty = 3, # type de lignes, un chiffre entre 1 et 6  
 lwd = 1.2, # taille de lignes  
 ylab = "Ca content", # titre pour l'axe des y  
 main="Exercie 3 of the Statistic lecture")  
abline(h=mean(Ca)+3\*sd(Ca), col="red") # Add a line on the graphic  
text(15,200, "3 \* Standard Deviation", col="red") # Add a text on the graphic



abline & text functions can add a line and a text respectively in the graphic.

With ggplot2 use geom.point():

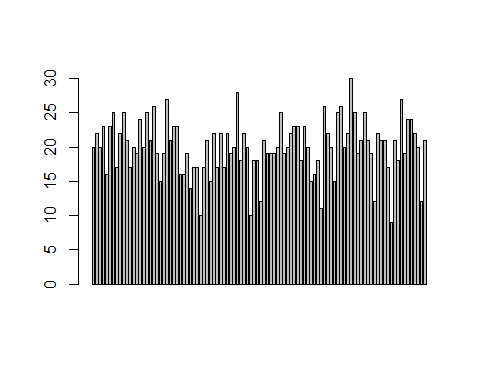
# Example of data frame  
df\_binom<-data.frame(x=c(0:n),p=dbinom(x=c(0:n),size=n,prob=p))  
# Genaration of a graph  
ggplot(data=df\_binom)+  
 aes(x=x,y=p)+  
 geom\_point()+  
 geom\_segment(aes(x=x,y=0,xend=x,yend=p))+  
 theme\_light()+  
 xlab("Number of success, x")+  
 ylab("Probability, p")+  
 labs(title=paste("Density of the binomial law, size=",n,", prob=",p))



## The Barplot

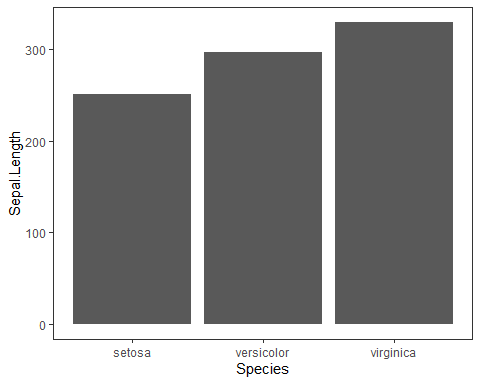
Basic R code for to generate a barplot of a Poison law:

barplot(rpois(100, lambda = m))



With ggplot2 use geom.histogram():

ggplot(iris, aes(x = Species, y = Sepal.Length))+  
 geom\_bar(stat = "identity")+  
 theme\_test()



It is possible to have multigroup in the barplot. It’s just needed to add the parameter fill = group2 in the aes(). By default the group are stacked. To have the barplot side by side, add position=position\_dodge() in the geom\_bar parameter.

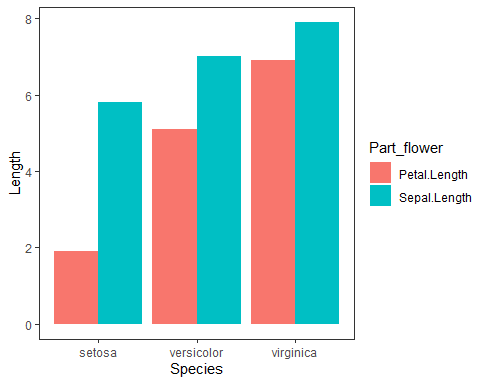
More complicate, the case of 2 continue variables y. It is not possible at this step with geom\_bar function, which allow only one x for one y. To solve this issue, it’s needed to reorganized the data to transform a table with y columns to a table with only two columns. This transformation can be done with the function gather(data, key, value, columns\_to\_gather) of the library tidyr.

* **data** ==> data frame, example iris
* **key** ==> name of the group
* **value** ==> name of the y axis
* **columns\_to\_gather** ==> y variables of the original data frame that it needed to be merged, separate by ,

library(tidyr)  
reformat <- gather(iris, key = "Part\_flower", value = "Length", Sepal.Length, Petal.Length)  
head(reformat)

## Sepal.Width Petal.Width Species Part\_flower Length  
## 1 3.5 0.2 setosa Sepal.Length 5.1  
## 2 3.0 0.2 setosa Sepal.Length 4.9  
## 3 3.2 0.2 setosa Sepal.Length 4.7  
## 4 3.1 0.2 setosa Sepal.Length 4.6  
## 5 3.6 0.2 setosa Sepal.Length 5.0  
## 6 3.9 0.4 setosa Sepal.Length 5.4

ggplot(reformat, aes(x = Species, y = Length, fill=Part\_flower))+  
 geom\_bar(stat = "identity", position=position\_dodge())+  
 theme\_test()



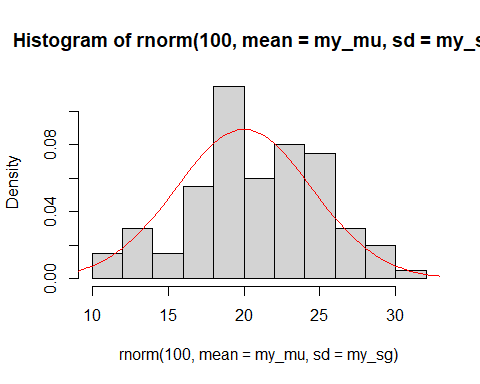
More inspiring barplot : <https://www.datanovia.com/en/fr/lessons/ggplot-barplot/>

*Remark: it is not the best graph to describe the iris database*

## The Histogram

Basic R code for to generate an histogram of a nominal law:

hist(rnorm(100,mean=my\_mu,sd=my\_sg), freq=F)   
curve(dnorm(x,mean=my\_mu,sd=my\_sg),from = xmin,to = xmax,ylab="densité",add=T,col="red")

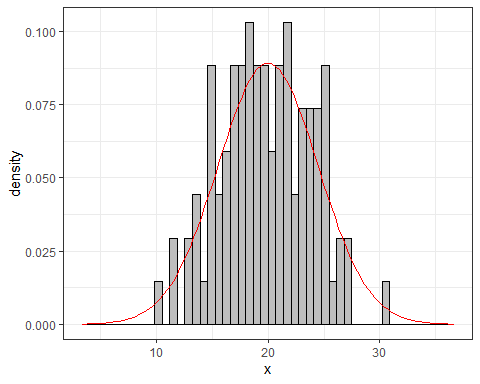


With ggplot2 use geom.histogram():

data <- data.frame(x = rnorm(100,mean=my\_mu,sd=my\_sg))  
ggplot(data, aes(x))+  
 geom\_histogram(aes (y=after\_stat(density)), #after\_stat(density) normalize to 1  
 fill = "grey",  
 color = "black",  
 bins = 50)+ # bins number of interval un the histogram  
 stat\_function(fun=dnorm,args=c(mean=my\_mu,sd=my\_sg),  
 color ="red",   
 size = 0.1)+  
 xlim(xmin,xmax)+  
 theme\_bw()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## Warning: Removed 2 rows containing missing values (`geom\_bar()`).



## The box-plot

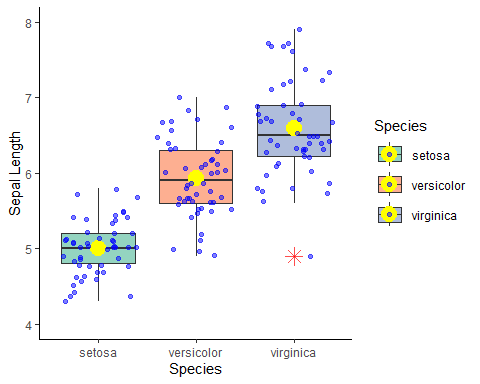
The boxplot can represent the distribution of the observation in function of the group. It gives also the quantile, the mean and the median information.

The easyiest way to have a boxplot is to use the basic R command: boxplot(quantitative\_variable~quatative\_variable, labels). Between the quantitative\_variable & the quantative\_variable there is a (**~**).  
Example:

boxplot(iris$Sepal.Length~iris$Species,ylab="Sepal Length",xlab="Species")

With ggplot2 use geom.boxplot():

p <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=Species))+  
 ylim(4,8)+  
 ylab("Sepal Length")+   
 geom\_boxplot(alpha=0.7, #alpha for the transparence,  
 outlier.colour="red",  
 outlier.shape=8,outlier.size=4) +  
 stat\_summary(fun = mean, geom="point", shape=20, size=8, color="Yellow") +  
 geom\_point(position="jitter", color="blue", alpha=.5)+  
 scale\_fill\_brewer(palette="Set2")+  
 theme\_classic()  
p



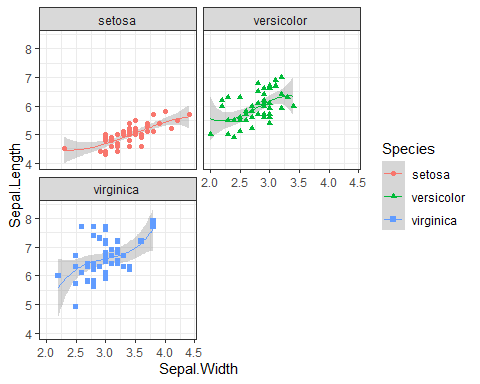
The function stat\_summary(fun.y = mean) display the mean value.

## Multiple grahs

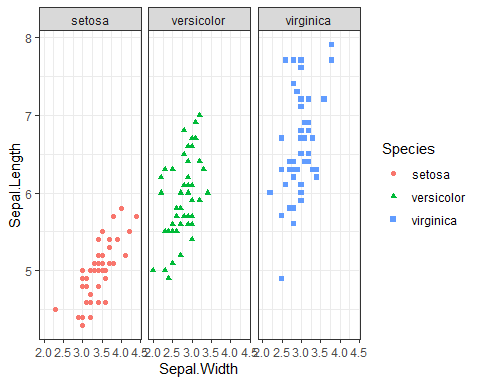
As it is showed, it is possible to combine several graphic in one by addition of goem functions. But sometimes, it is needed to separate the graphics. It is called the faceting. Two function can be used:

* facet\_warp(qualitative\_variable, nrow = number of ligne)
* facet\_grid(ligne ~column)

ggplot(iris, aes(x=Sepal.Width, y=Sepal.Length, shape = Species, color=Species))+  
 theme\_bw()+  
 geom\_smooth(method="lm", formula= y~poly(x,3), se=TRUE, size = 0.5) +  
 geom\_point() +  
 facet\_wrap(iris$Species, nrow = 2)



ggplot(iris, aes(x=Sepal.Width, y=Sepal.Length, shape = Species, color = Species))+  
 theme\_bw()+  
 geom\_point() +   
 facet\_grid(~Species)



In the first facing graph, the function geom\_smooth(method="lm", formula= y\~poly(x,3), se=TRUE, size = 0.5) has been introduced. This function is used to add a smooth regression curve to show the general trend of the dataset. Here the method used to draw this curve is a fitting linear model **lm.**

For more inspired graphs:  
<https://rstudio-pubs-static.s3.amazonaws.com/578122_5e69256788bb4dcca6157d2bcfa7694e.html>  
<https://www.charlesbordet.com/fr/faire-beaux-graphiques-ggplot2/#it%C3%A9ration-4---ajouter-des-couleurs-pour-chaque-groupe>

# Statistics Test

## Test de **Student**

## Test de **Fisher**

voir la library moment kurtosis skweness