Session 2

statistiques descriptives et tests d'hypothèses, figures, paquets

Teachers: Guillaume Achaz, Claire Vandiedonck
Helpers: Natacha Cerisier, Jacques van Helden

Le script "DUBii_R_Session2.R" reprenant l'ensemble du code présenté dans ce diaporama est fourni

Plan de la session 2

1. Random variable and sampling
2. Figures with R
3. R Packages
4. Hypotheses and statistical tests
5. Tutorial: A first data analysis

Why using statistics ?

Making sense of data

↳ Aim: identify variables whose variation levels are associated with a phenotype or a covariate of interest
(eg: response to stress, to a treatment, survival, mutation, tumor class, time…)

Variable to explain ~ explanatory variables + covariates + residual error

Problems addressed by statistics:

1. estimation: of the effects of interest and of how they vary
2. testing: = assessing the statistical significance of the observed effects

1. Random variable and sampling

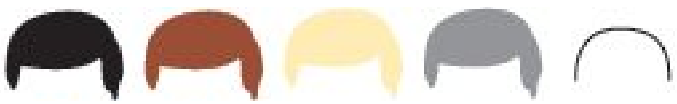
Some French-English terms

- random variable = variable aléatoire
- random/sampling fluctuation = variation d'échantillonnage
- sample = échantillon
- mean = moyenne
- variance = variance = dispersion des données autour de la moyenne
- standard deviation = écart type = racine carrée de la variance
- standard error = standard deviation of the mean = écart type de la moyenne = écart-type rapporté à la racine carrée de la taille de l'échantillon
- co-variate = covariable
- barplot = diagramme en bâtons
- density probability =densité de probabilité
- confidence interval (CI) = intervalle de confiance
- threshold = seuil
- significance = signification
- likely = probable
- power = puissance
- pairwise = apparié

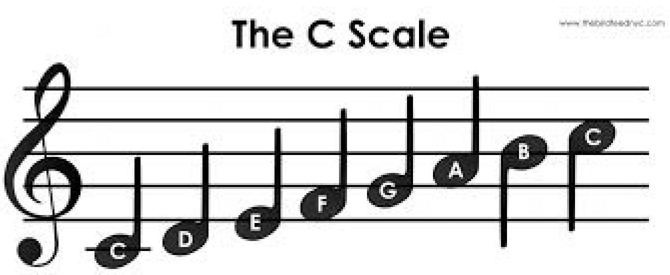
Traits

Qualitative

- Nominal = categorical

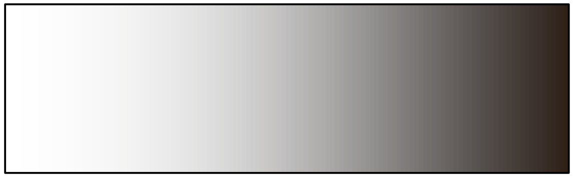


- Ordinal = rankable



Quantitative = variable

- continuous: uncountable items



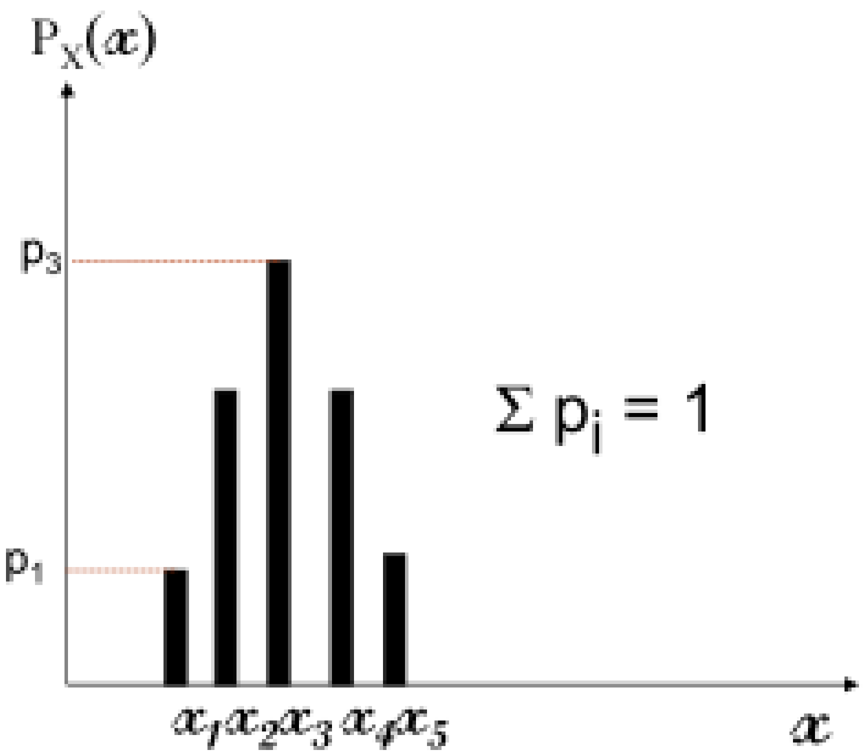
- discrete : countable items

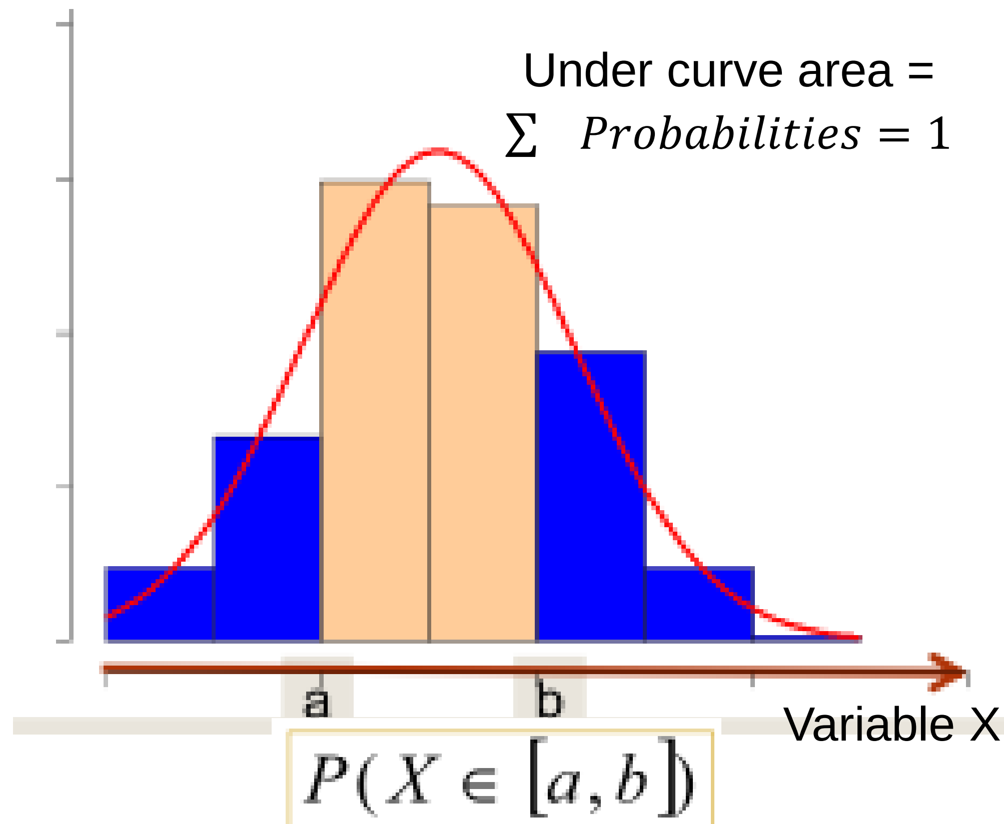
Random variable

Probability associated to the each value of the variable

↳ characterized by a distribution function of density probability

- discrete distributions = barplots
- continuous distributions





A population versus a sample

Population = homogeneous set

sample = subset from the population

X: random quantitative variable

x_i: observations of X in sample i of size n_i

population

μ = mean of a quantitative trait **X**

sample 1
of size n1

$\hat{\mu} = m_1$
= observed mean of X

x_2 x_1 x_i .... x_{n1}

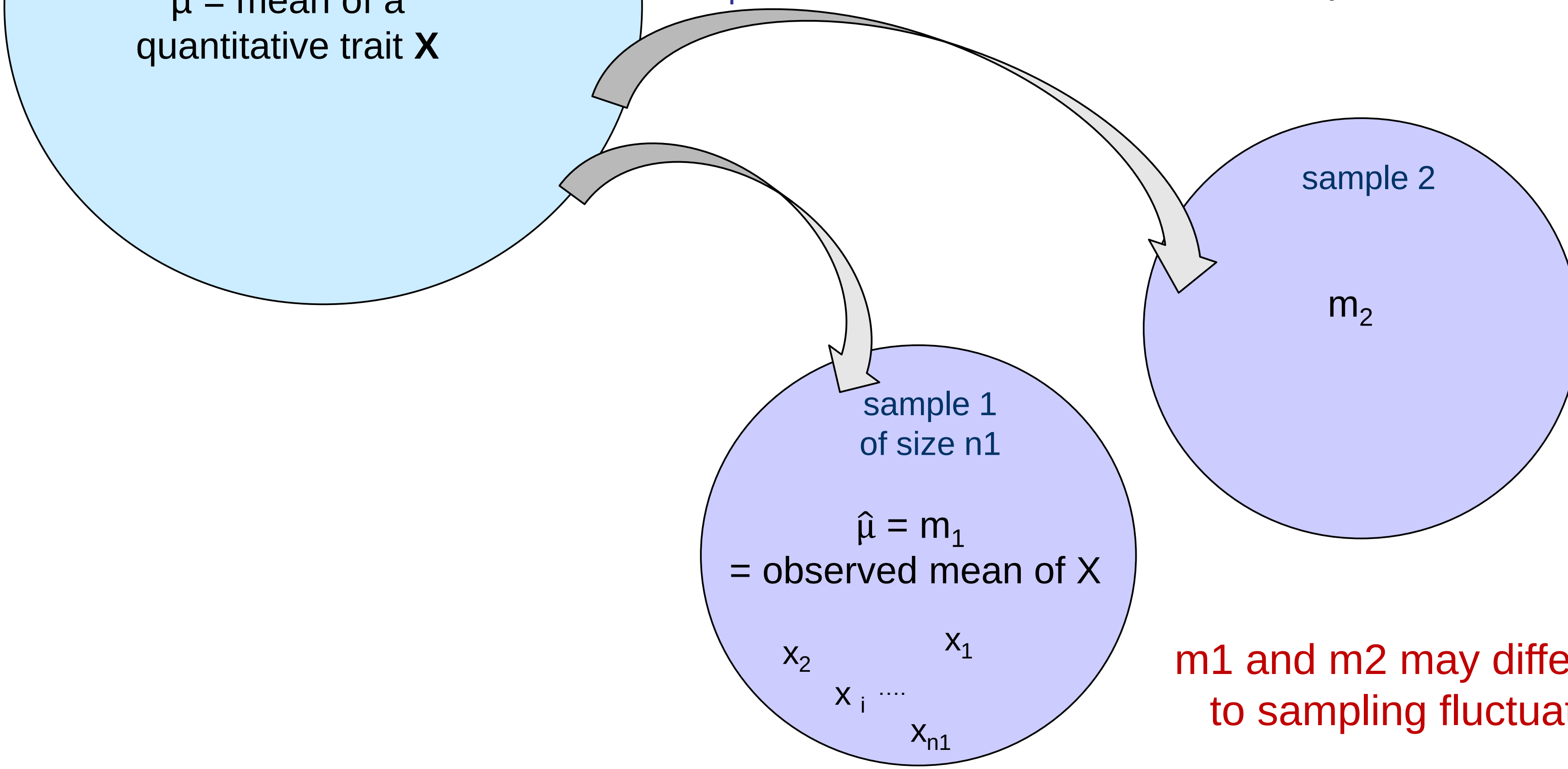
A population versus a sample

Population = homogeneous set

sample = subset from the population

X: random quantitative variable

x_i: observations of X in sample i of size n_i



m1 and m2 may differ due to sampling fluctuation

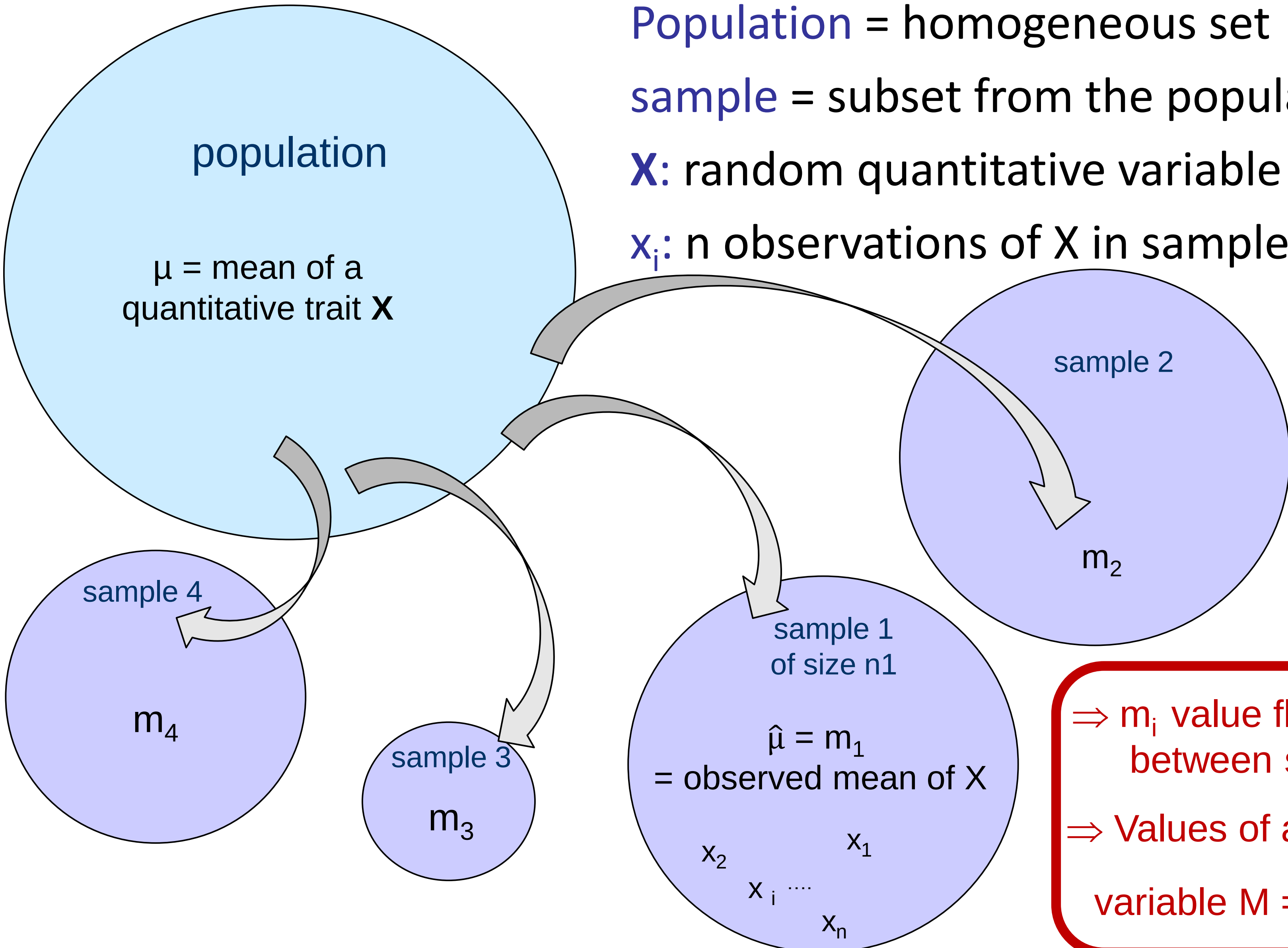
A population versus a sample

Population = homogeneous set

sample = subset from the population

X: random quantitative variable

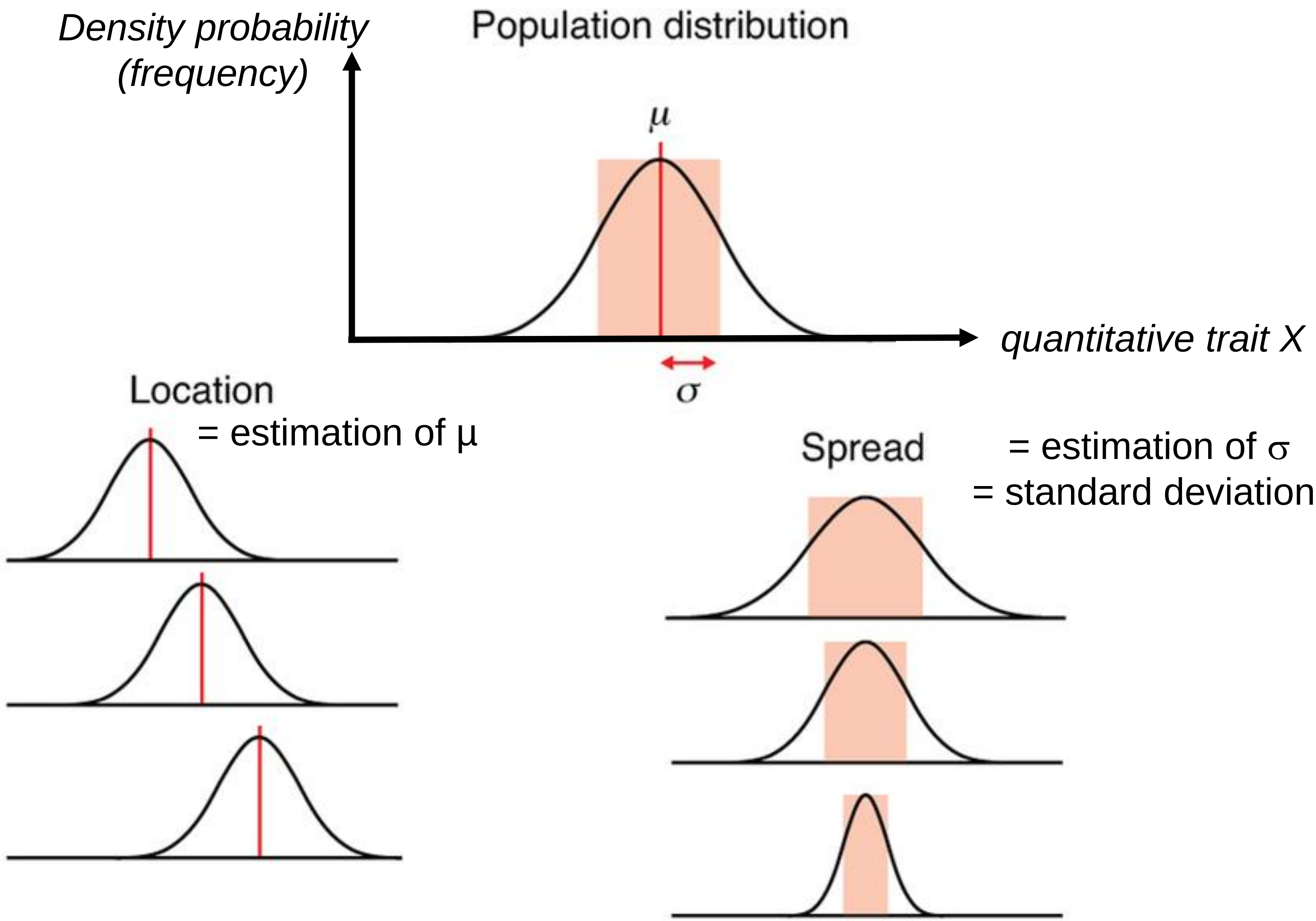
x_i: n observations of X in sample i of size n_i



⇒ m_i value fluctuates between samples

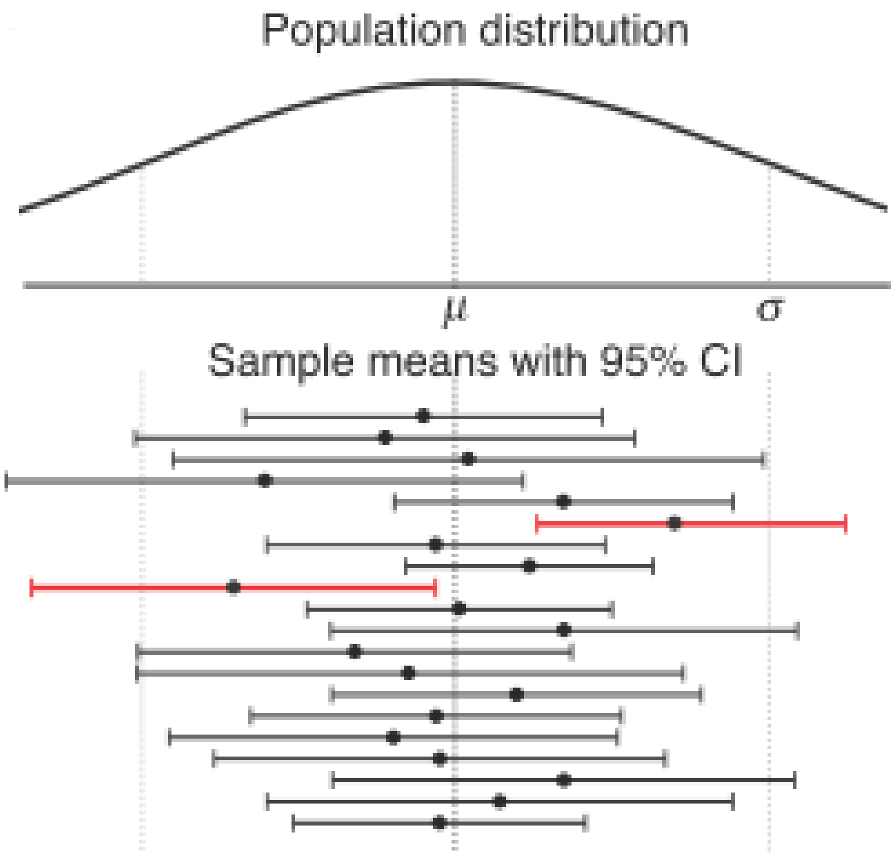
⇒ Values of a random variable M = $\bar{X} = \frac{\sum X}{n}$

1st aim: estimation of population parameters



Estimation with confidence intervals

95% of intervals are expected to span the mean
while the other 5% (in red here) do not



$$IC_{1-\alpha} \text{ of } \mu = \left[m \pm \varepsilon_\alpha \sqrt{\frac{s^2}{n}} \right]$$

Practical:

Sampling variation with a Shiny application

http://shiny.calpoly.sh/Sampling_Distribution/

2. Figures with R core

Start R again...together (demo on R studio)!

I saved into an .Rdata file the dataframe object called myDataf of session 1:

```
> save(myDataf, file="dataframe_session1.RData")
```

Load the data into a new R session:

```
> load("dataframe_session1.RData ")
> ls()
[1] "myDataf"
```

Some basic graphs

Scatter plot with the function plot()

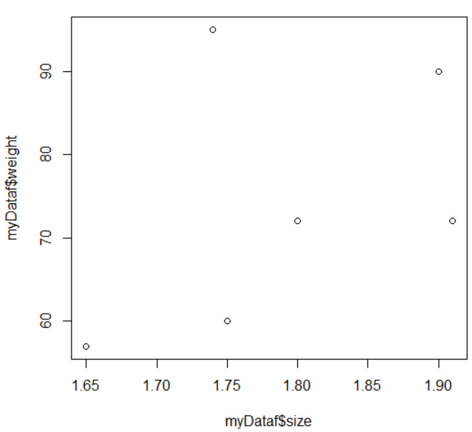
```
> plot(myDataf$weight~myDataf$size)  # Y~X is equivalent to X, Y
```

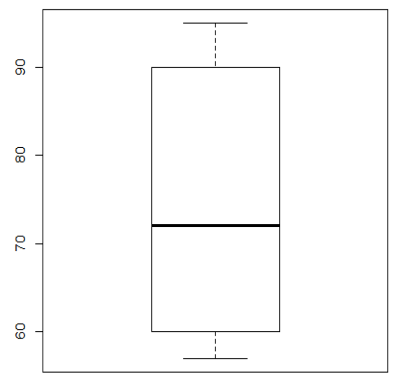
Distribution with the fonction boxplot()

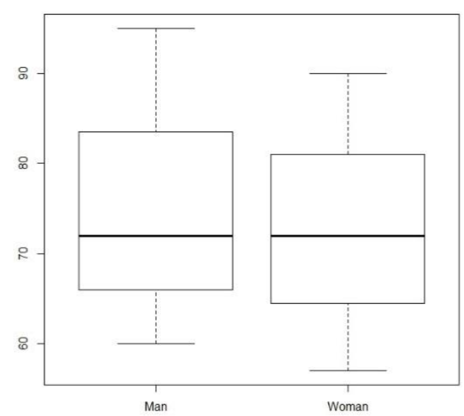
```
> boxplot(myDataf$weight)
> boxplot(myDataf$weight~myDataf$sex) # ~ to display depending on a categorical variable
```

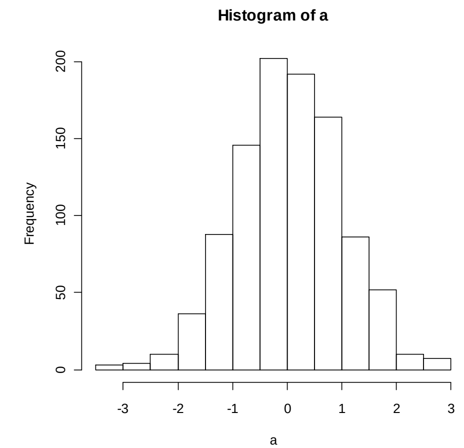
Histograms with the hist()

```
> a <- rnorm(1000) # sample randomly 1000 values from a normal distribution
> hist(a, breaks=20) # breaks to specify the number of intervals
```









Three-level graph functions

1. Primary graph functions = high-level graphical functions

to plot the most principal graphs in R

2. Secondary graph functions = low-level plotting commands

to complement an existing plot

3. Graphical parameters

to modify the presentation of the plots

- either as options within the above two kind of graphic functions
- or permanently with the par() function before plotting the graph

The primary graph functions

Examples of the most frequently used graphs in R

plot()	to plot points at given coordinates (x) or (x,y) ordered on the axes
pie()	to plot a circular pie chart of a qualitative variable
barplot()	to plot occurrences/frequencies of a qualitative variable
hist()	to plot the distribution of a quantitative variable as an histogram
boxplot()	to plot the distribution of a quantitative variable as a boxplot
stripchart()	to plot the values of a quantitative variable along an axis
pairs()	to draw pair-wise plots between the columns of a matrix

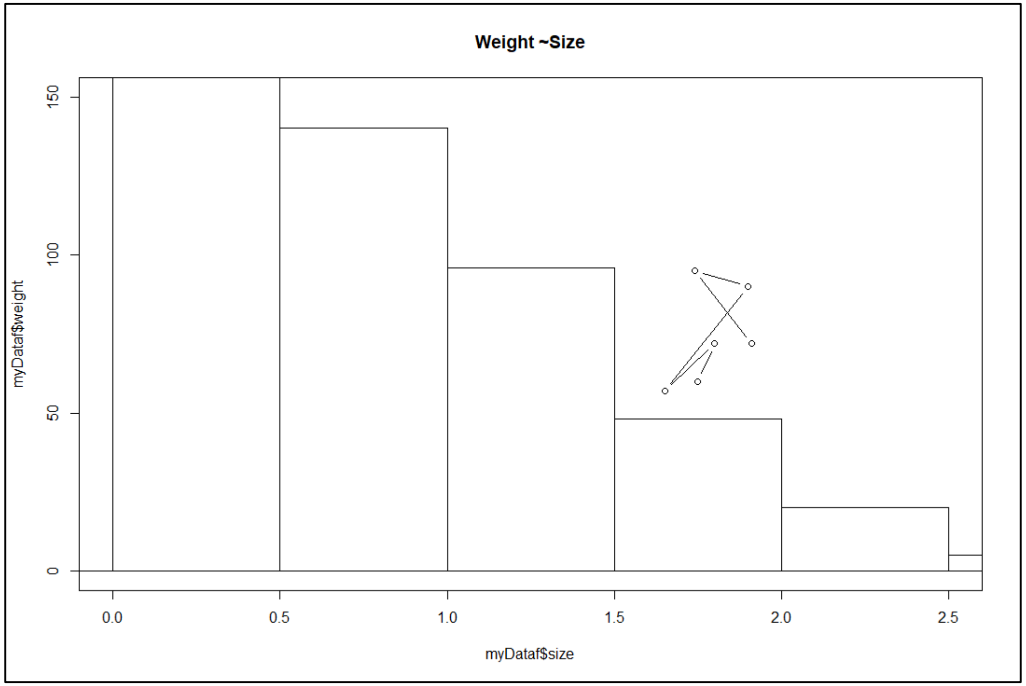
…

Some arguments/options are identical for several graph functions

eg. "main"	to specify the title
"xlim", "ylim"	to specify the limits of axes
"type"	to specify the type of plots ("p" for points, "l" for lines, "n" for none…)
"add"	to supperpose to the previous plot if TRUE

Example of primary graph functions

```
> plot(myDataf$weight~myDataf$size)
> plot(myDataf$weight~myDataf$size,
    main="Weight ~Size")
      # to add a title
> plot(myDataf$weight~myDataf$size,
    main="Weight ~Size", type="l")
      # to draw a line
> plot(myDataf$weight~myDataf$size,
    main="Weight ~Size", type="b")
      # to connect a line between points
> plot(myDataf$weight~myDataf$size,
    main="Weight ~Size", type="b",
    xlim= c(0,2.5), ylim=c(0,150))
      # to specify axis limits
```




```
> hist(a,breaks=20, add=T)
      # the add argument allows to draw the new plot
      # above the previously called plot
      # note: add does not work for plot, use points(), cf. secondary functions)
```

The secondary graph functions

Examples of the most frequently used low-level plotting functions in R

- complement an existing plot

eg. points()	to add new points
lines()	to add points connected to a line
abline()	to add a new line of given slop and interecpt
mtext()	to add text in a margin
axis()	to add axis with a given layout
legend()	to add a legend
title()	to add a global title

…

Graphical parameters

Examples of important parameters

« mar » size of margins

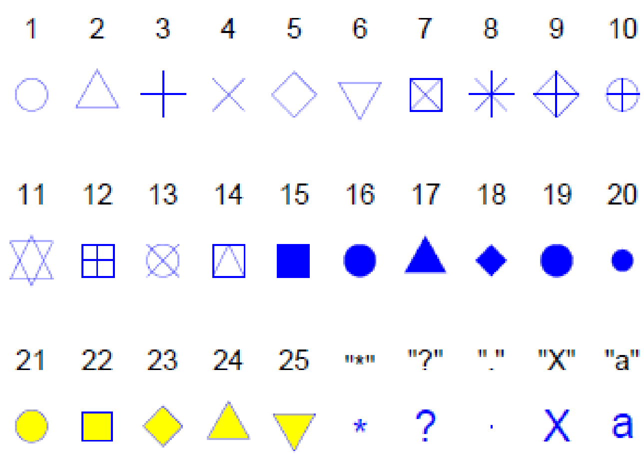
« mfrow and mfcol » to specify the display of plots (number of lines and columns) within the graph window

« cex » size of texts and symbols

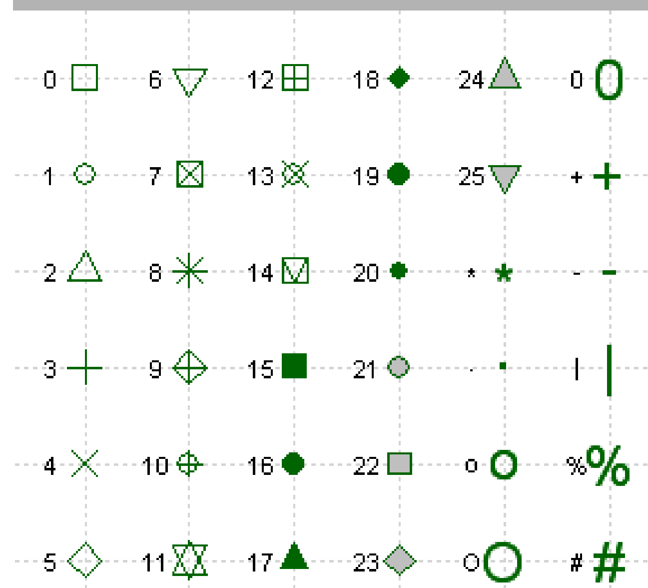
similarly, specific cex parameters for axis: cex.axis, for labels: cex.lab…

« pch » type of symbols

E. Paradis



QUICK R



« bg » background color (by default = "transparent", or "white" in Rstudio)

« col » color of symbols, texts…

similarly specific col parameters for axis: col.axis, for labels: col.lab

Example of secondary graph functions and parameters

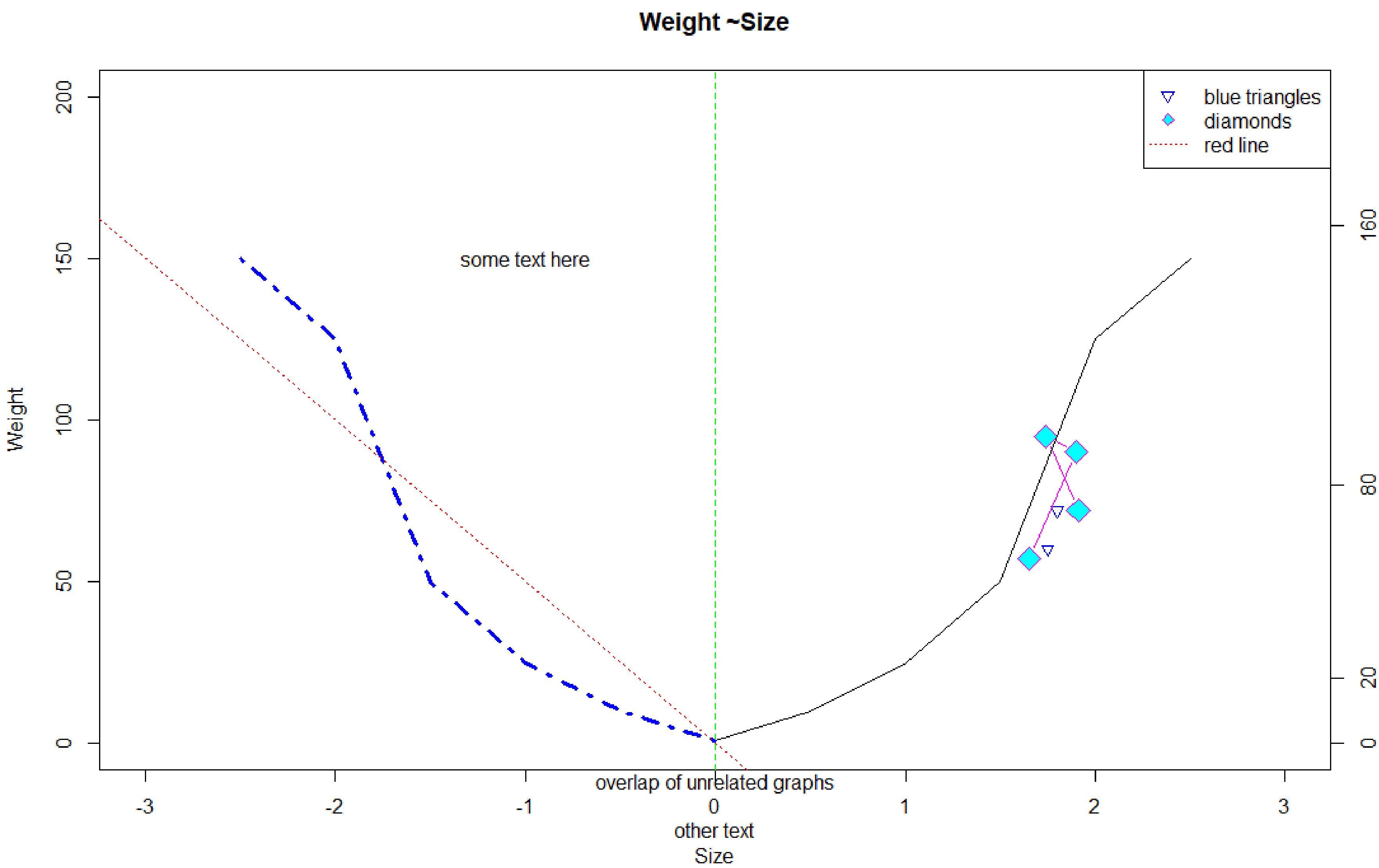
```
> plot(myDataf$weight~myDataf$size, main="Weight ~Size",
   xlim= c(-3,3), ylim=c(0,200), type="n", xlab="size", ylab="weight")
        # draw the frame of the plot but not the data with type="n«

> points(myDataf$weight[1:2]~myDataf$size[1:2], pch=6, col="blue")
        # points() allows to add the data to the existing plot
        # it is usefull to filter data to display points on different manners

> points(myDataf[3:6,"weight"]~myDataf$size[3:6], type="b", pch=23, col="magenta",
        bg="cyan", cex=2)
        # here for the last 4 points, I change the type and its color and background

> points(seq(0,2.5, 0.5), c(1, 10, 25, 50, 125, 150), type="l")
        # using type="l", I can aslo draw a line through the points
> lines(-seq(0,2.5, 0.5), c(1, 10, 25, 50, 125, 150), lty= "dotdash", col="blue", lwd=3)
        # lines() also draws a line. You can specify its type with lty and width with lwd
> abline(0, -50, lty=3, col="red")
> abline(v=0, lty=2, col="green")
        # abline is a further function to draw lines with a given slope, vertical or horizontal
```

Example of secondary graph functions and parameters

```
> mtext("overlap of unrelated graphs", side=1)
> mtext("other text", side=1, line=2)
            #mtext() is used to write text in the margins of the plot
> text(-1, 150, "some text here")
            # while the function text adds text at the given coordinates
> axis(side=4, labels=c(0, 20, 80, 160),  at=c(0, 20, 80, 160), tick=T)
             # axis is another way to drawe x, y axis but also an additional axis
            # on the right side
```

```
> legend("topright", c("blue triangles", "black  dots", "redline"),
            col=c("blue", "black", "red"),  pch=c(6, 1, NA), lty=c(0,0,3))
#  you specify within vectors the text of the different elements, their color, etc…
```

Example of secondary graph functions and parameters



Colors in R

Display current colors with palette()

Specify colors by their index, "name", "hexadecimal" or "rgb" values

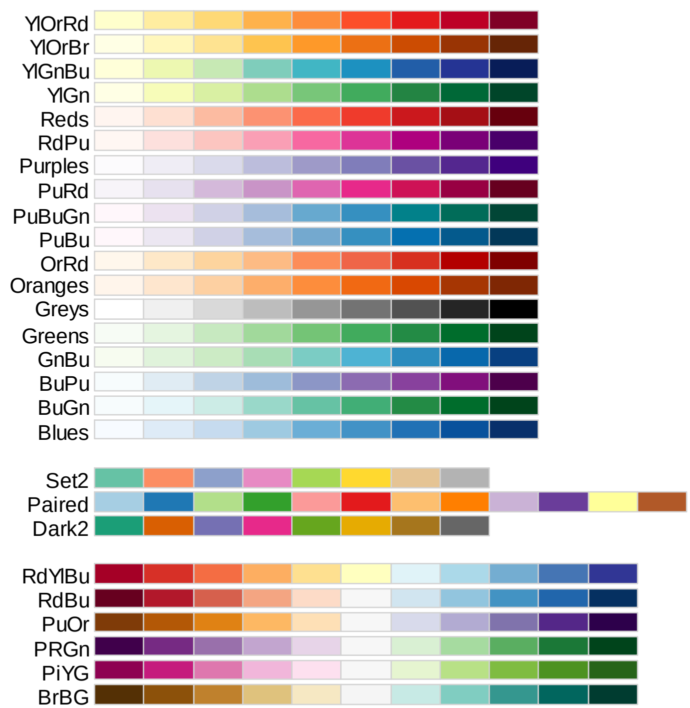
R Chart color at https://web.archive.org/web/20121202022815/http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf

white	#FFFFFF	255	255	255
aliceblue	#F0F8FF	240	248	255
antiquewhite	#FAEBD7	250	235	215
antiquewhite1	#FFEFDB	255	239	219
antiquewhite2	#EEDFCC	238	223	204
antiquewhite3	#CDC0B0	205	192	176
antiquewhite4	#8B8378	139	131	120
aquamarine	#7FFFD4	127	255	212
aquamarine1	#7FFFD4	127	255	212
aquamarine2	#76EEC6	118	238	198
aquamarine3	#66CDAA	102	205	170
aquamarine4	#458B74	69	139	116
azure	#F0FFFF	240	255	255
azure1	#F0FFFF	240	255	255
azure2	#E0EEEE	224	238	238
azure3	#C1CDCD	193	205	205
azure4	#838B8B	131	139	139
beige	#F5F5DC	245	245	220
bisque	#FFE4C4	255	228	196

Etc…

```
#install.packages("RColorBrewer")
library(RColorBrewer)
display.brewer.all(colorblindFriendly=TRUE)
```

Very useful package « **RColorBrewer** » with associated palettes including for colorblind



Tip: to find the color reference from an electronic document, use InstantEye Dropper

Graphical parameters with par()

```
> par()                  # displays the current parameters in a list!
> par()$cex              # displays the current cex parameter

> opar <- par()          # to save the current parameters VERY IMPORTANT
> par(bg=rgb(0, 51, 102, max=255), col="white", mfrow=c(2,3), cex=1.1)
             # new graphs will have a background of the same color as my slide titles
             # and 6 plots will be plotted on the same graph window (2 rows, 3 columns)
             # and the size of the text will be 10% larger than by default
```

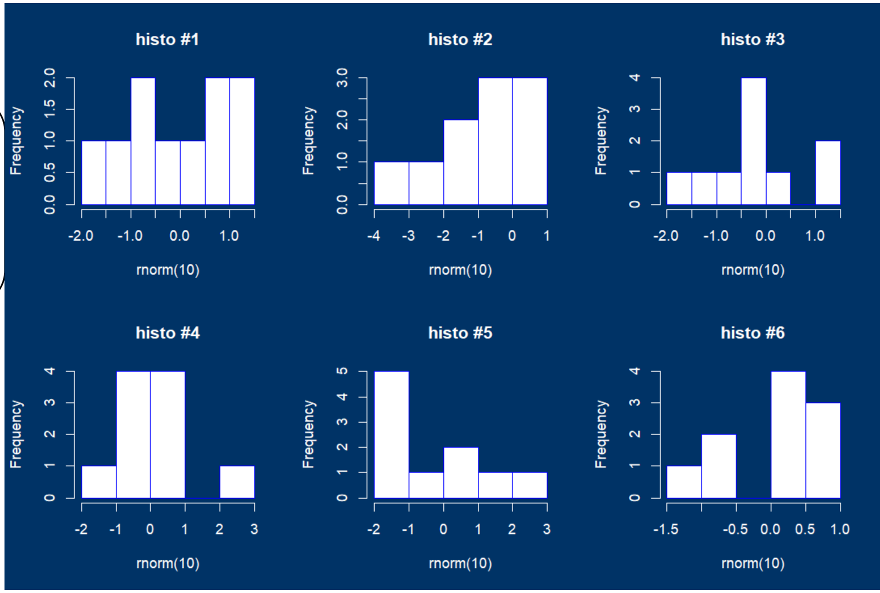
Then do your plots…

```
> hist(rnorm(10), col="white", border="blue“,
   col.axis="white", fg="white”,  col.lab="white",
   col.main="white")
```

…and 5 other plots

and finally restore the initial parameters

```
> par(opar)      # to restore default parameters
```



Saving figures in your working directory

Save figures in different formats with the appropriate function

bmp()	.bmp
jpeg()	.jpeg
tiff()	.tiff
png()	.png
postscript()	.eps

```
> png("MyPlot.png")

> hist(rnorm(10000, 0, 1),
freq=F)

> dev.off()
```

Three steps

1. Type the function with the name of the saved file as an argument with the correct extension

   Other arguments like « width » and « height » to specify dimensions

2. Do your plot -> it is directed to the file and not displayed in the graphical window within R
3. Close the graph by typing the following function

   dev.off()

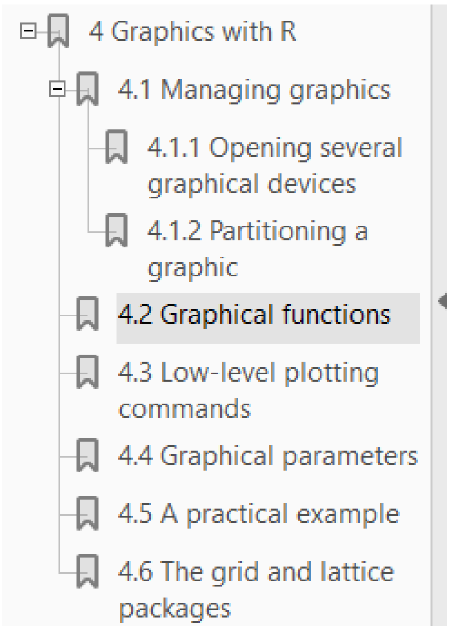
Specific case for pdf() to save graphs in a .pdf

- you may save each figure at a time
- or all several (all) figures generated with all the command lines entered between pdf() and dev.off()

Getting help

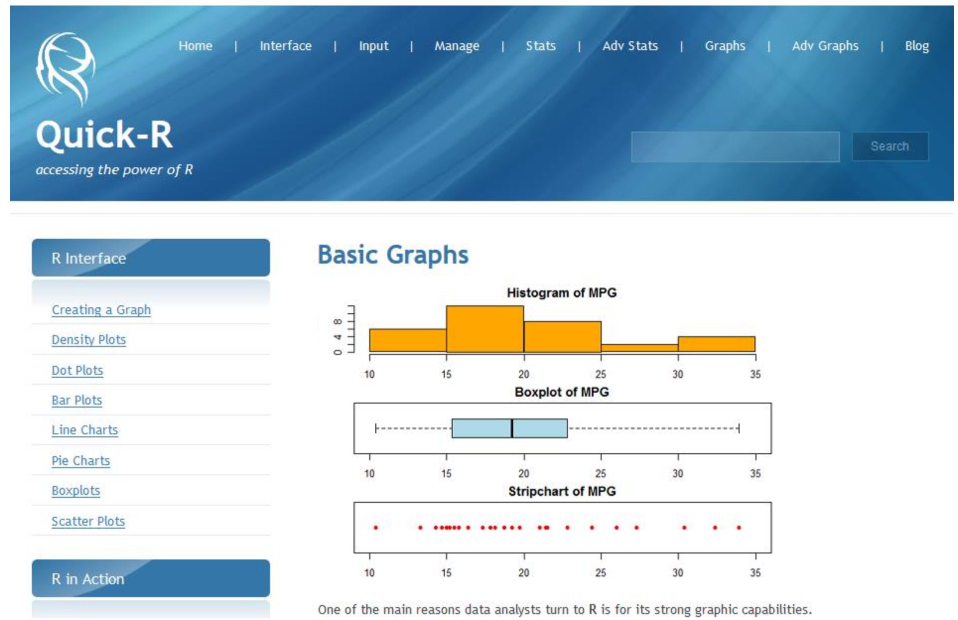
R for beginners E. Paradis

Chapter 4 for graphs quite exhaustive
in moodle in French and English


4 Graphics with R
4.1 Managing graphics
4.1.1 Opening several graphical devices
4.1.2 Partitioning a graphic
4.2 Graphical functions
4.3 Low-level plotting commands
4.4 Graphical parameters
4.5 A practical example
4.6 The grid and lattice packages

QUICK R:

http://www.statmethods.net/
basic and avdanced graphs
with main parameters



R gallely

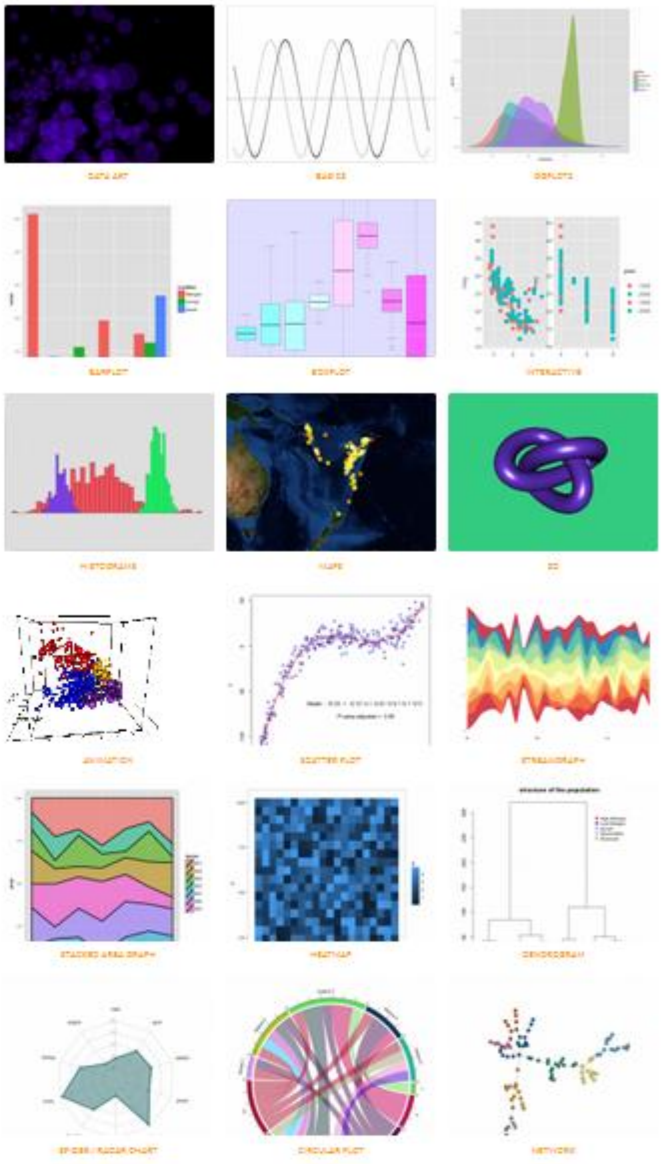
http://www.r-graph-gallery.com/all-graphs/
for specific kinds of graphs

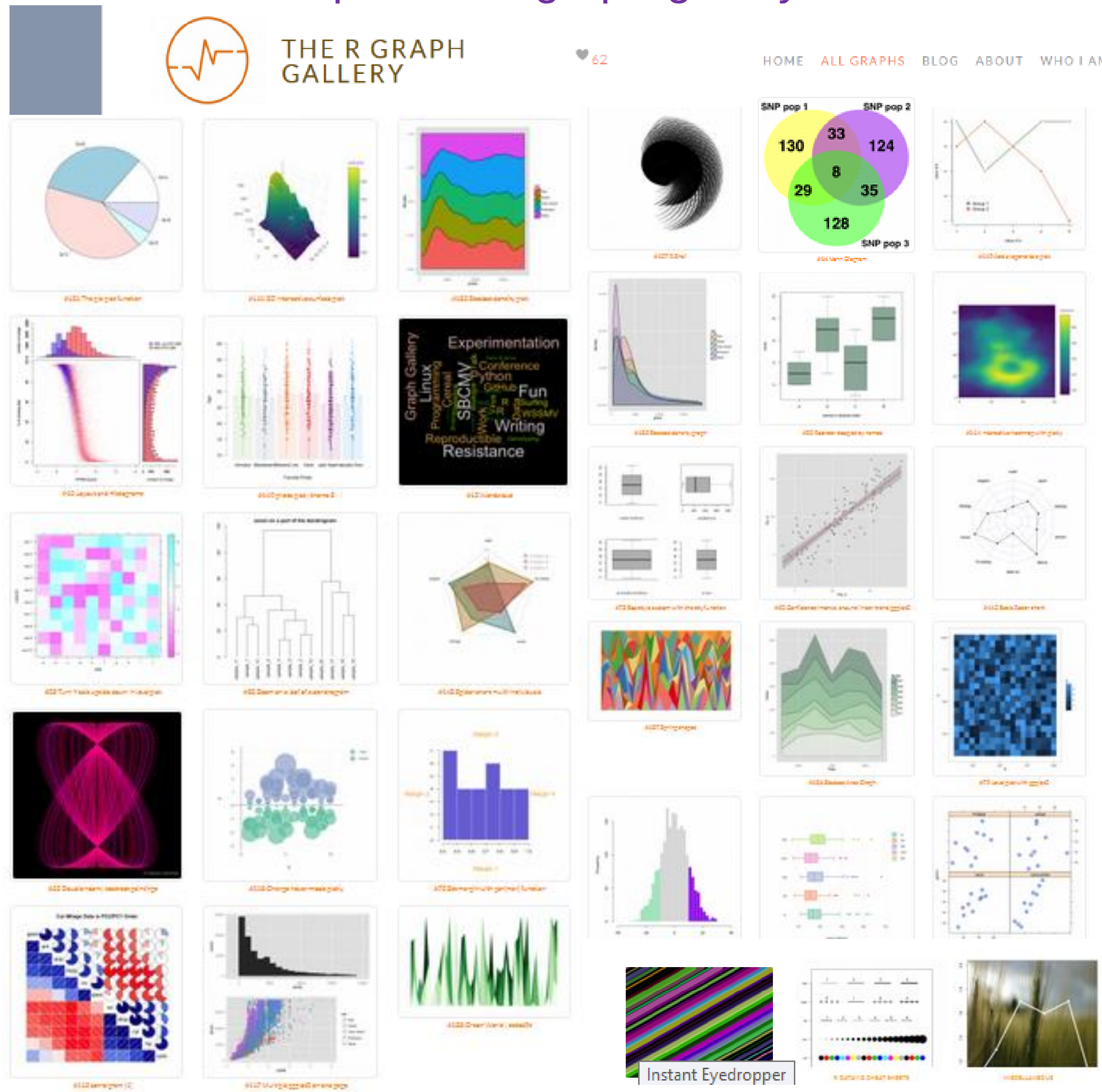
And some blogs for specific questions

https://www.stat.ubc.ca/~jenny/STAT545A/block14_colors.html#using-colors-in-r
https://danieljhocking.wordpress.com/2013/03/12/high-resolution-figures-in-r/

Endless kinds of graphs with R

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

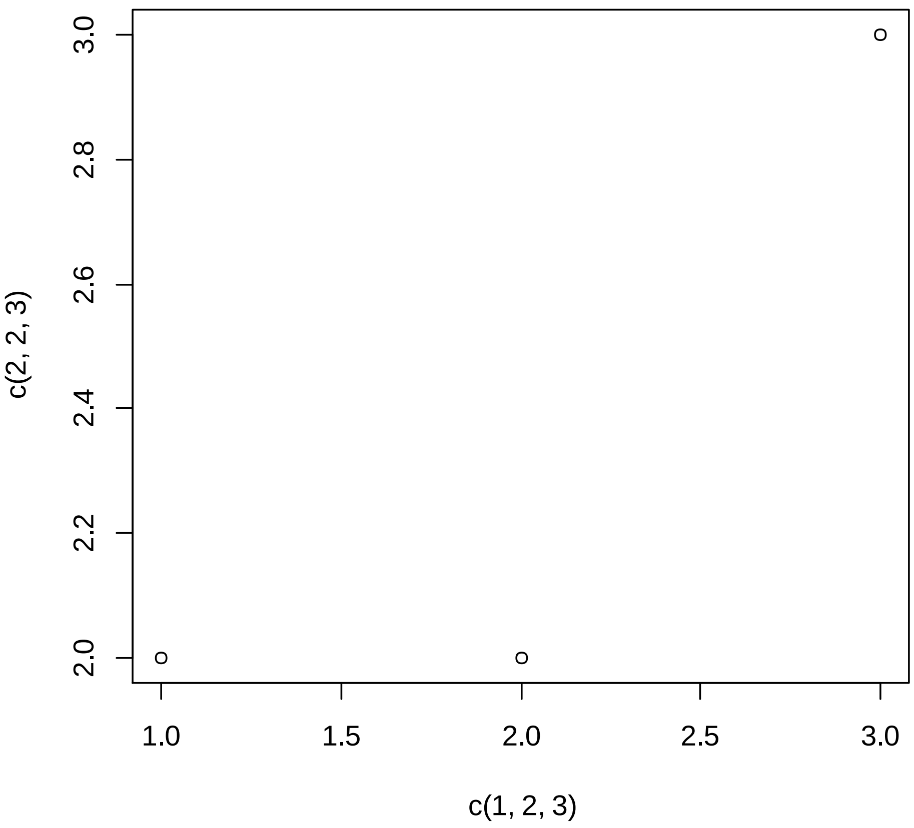


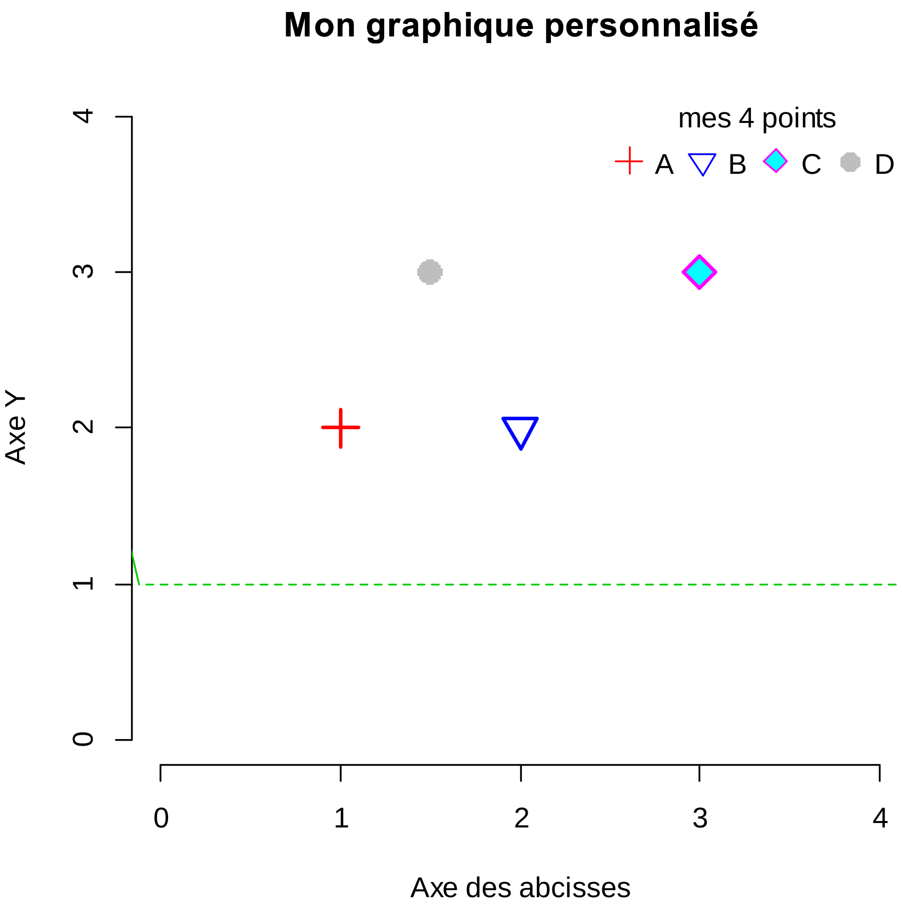


Practical:

generate a custom figure with the exercice 1 of the tutorial descriptive-statistics.html before

after





3. R packages

Packages in R

R packages:

- set of functions and sometimes of data aiming at fulfilling specific tasks or adressing sepcific problems
- uses core R functions
- may use other packages functions

  -> these other packages are called 'dependencies'

➢ use R packages rather than rewriting a function already written by someone else !

Data from packages

➢ Using data from an R package:

Loading data with the function data() with the argument « package »

```
> try(data(package="rpart" )) #list the data available from the package « rpart »
```

R data sets

```
Data sets in package 'rpart':

car.test.frame          Automobile Data from 'Consumer Reports' 1990
car90                   Automobile Data from 'Consumer Reports' 1990
cu.summary              Automobile Data from 'Consumer Reports' 1990
kyphosis                Data on Children who have had Corrective Spinal
                        Surgery
solder                  Soldering of Components on Printed-Circuit
                        Boards
stagec                  Stage C Prostate Cancer
```

```
> data(stagec, package="rpart") # load the dataset « stagec » corresponding to Stage C Prostate Cancer in R

> ls()

[1] "stagec"

> help(stagec, package="rpart") # to get help on the stagecdata
```

Which R packages are installed on my computer?

R programm itself is installed in a « bin » folder

R packages are installed in a « library » folder…there may be different library folders

- Getting the folders, i.e libraries, where R packages are installed using .libPaths() and corresponding packages with list.files()

```
> .libPaths()
[1] "C:/Users/claire/Documents/R/win-library/3.2"
[2] "C:/Program Files (x86)/R-3.2.1/library"

> list.files(.libPaths()[2])
 [1] "abind"            "acepack"          "annotate"
 [4] "AnnotationDbi"    "base"             "BH"
 [7] "Biobase"          "BiocGenerics"     "BiocInstaller"
[10] "BiocParallel"     "biomaRt"          "Biostrings"
[13] "bitops"           "boot"             "car"
[16] "caTools"          "chron"            "class"
[19] "cluster"          "codetools"        "colorspace"
[22] "compiler"         "corrplot"         "curl"
[25] "data.table"       "datasets"         "DBI"
[28] "DESeq"            "devtools"         "dichromat"
etc…
```

Which R packages are installed on my computer?

➢ Or getting the installed packages directly with the function installed.packages() that returns a matrix containing all packages with their version and location…

```
> colnames(installed.packages())
[1] "Package" "LibPath"  "Version" "Priority" "Depends"
[6] "Imports" "LinkingTo""Suggests""Enhances" "License"
[11] "License_is_FOSS" "License_restricts_use" "OS_type" "MD5sum" "NeedsCompilation"
[16] "Built"

> head(installed.packages()[,c(1,2,3)]) # to get the most useful columns
                  Package         LibPath                                    Version
AnnotationDbi "AnnotationDbi" "C:/Users/claire/Documents/R/win-library/3.3" "1.36.2"
backports     "backports"     "C:/Users/claire/Documents/R/win-library/3.3" "1.1.2"
base64enc     "base64enc"     "C:/Users/claire/Documents/R/win-library/3.3" "0.1-3"
BH            "BH"            "C:/Users/claire/Documents/R/win-library/3.3" "1.62.0-1
Biobase       "Biobase"       "C:/Users/claire/Documents/R/win-library/3.3" "2.34.0"
BiocGenerics  "BiocGenerics"  "C:/Users/claire/Documents/R/win-library/3.3" "0.20.0"
            # etc…
```

Loading installed R packages

- Loading an installed R package using the function **library()** and the name of the package as an argument, either with or without ". This is the recommended function to load a package. You might also see the function **require()** : sometimes preferred if within a function since it returns warnings instead of errors although it might be better to know the package is missing before using the function

```
> library(MASS)          # load the MASS library dedicated to statistics
> sessionInfo() # check loaded version of all loaded packages
R version 3.5.2 Patched (2019-01-02 r75949)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17134)

Matrix products: default

locale:
[1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
[3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
[5] LC_TIME=French_France.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] MASS_7.3-51.1

loaded via a namespace (and not attached):
[1] compiler_3.5.2
```

What happens if I try to load an uninstalled package?

```
library(tutu) # it returns an error
Error in library(tutu) : aucun package nommé 'tutu' n'est trouvé
require(tutu)    # it returns a warning
Le chargement a nécessité le package : tutu
Warning message:
In library(package, lib.loc = lib.loc, character.only = TRUE,  logical.return
= TRUE,   :
aucun package nommé 'tutu' n'est trouvé
```

- Check and install missing package before loading using **require()** since require returns (invisibly) a logical indicating whether the required package is available

```
require(tutu) == FALSE
Le chargement a nécessité le package : tutu
[1] TRUE                # TRUE here means require(tutu) returns the logical value FALSE
# Etc…
```

=> solution recommended when you pass your script to others

```
if (!require("RColoBrewer" , quietly = T)) {    # if the package "RColoBrewer" is already installed,
    install.packages(" RColoBrewer ")           # it will not be installed again, whereas if it was not
  }                                             # installed, it will be
library(RColorBrewer)
```

Installing new R packages

Packages are stored in several possible repositories:

1. CRAN -> the general R repository
2. GitHub -> geeks' repository…includes tools in many programming languages

   You may use git with gitHub or gitLab also for your own scripts. It is possible with Rstudio to push and pull documentsto or from Git -> excellent for versioning control
3. Bioconductor -> a repository for bioinformatics tools = the Bioconductor project

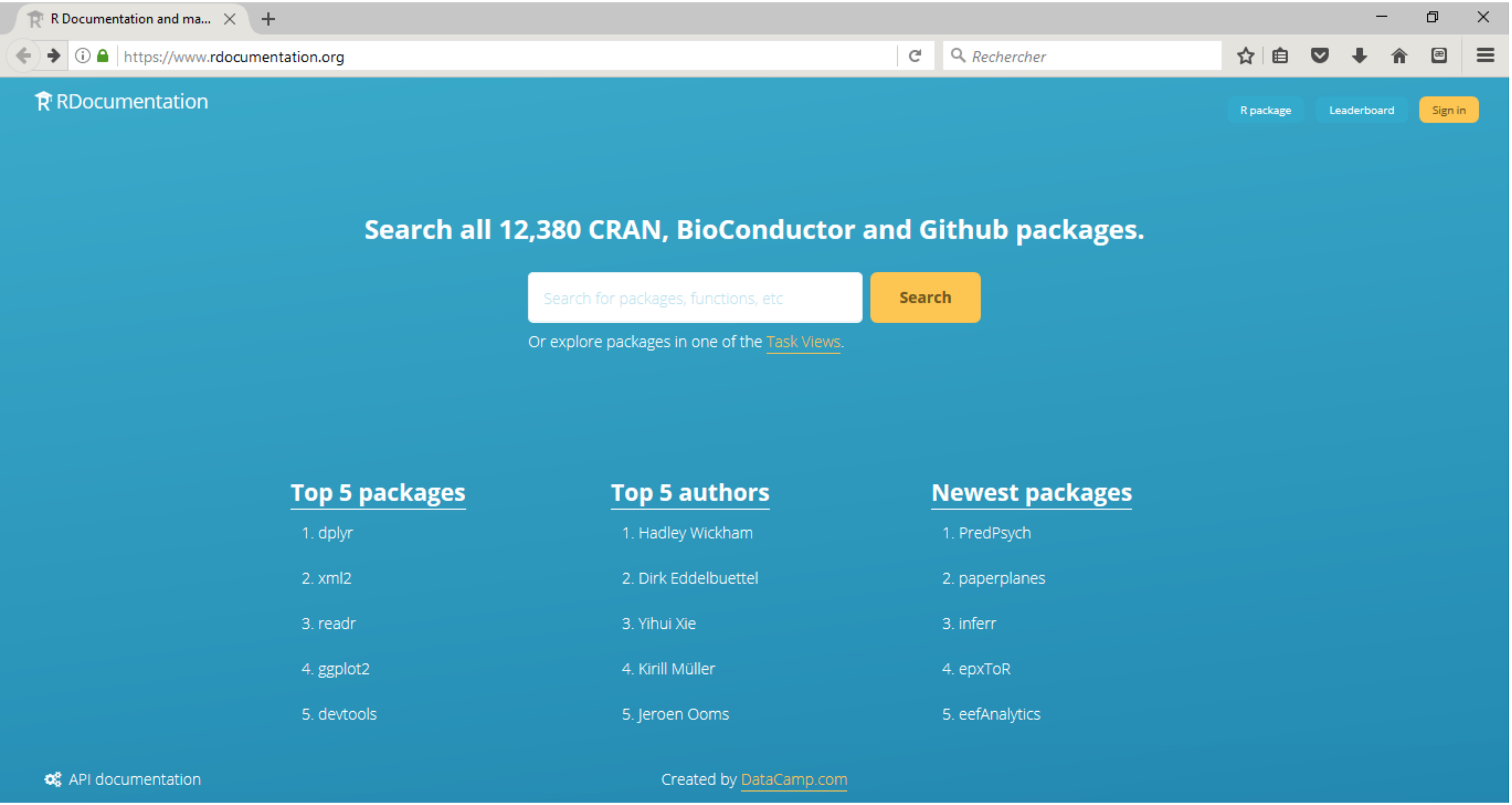
etc…

Packages are written for a specific minimal R version

Packages may require dependent packages

Functions and their corresponding packages in R

Finding the package corresponding to a given function using
https://www.rdocumentation.org/



Installing new R packages

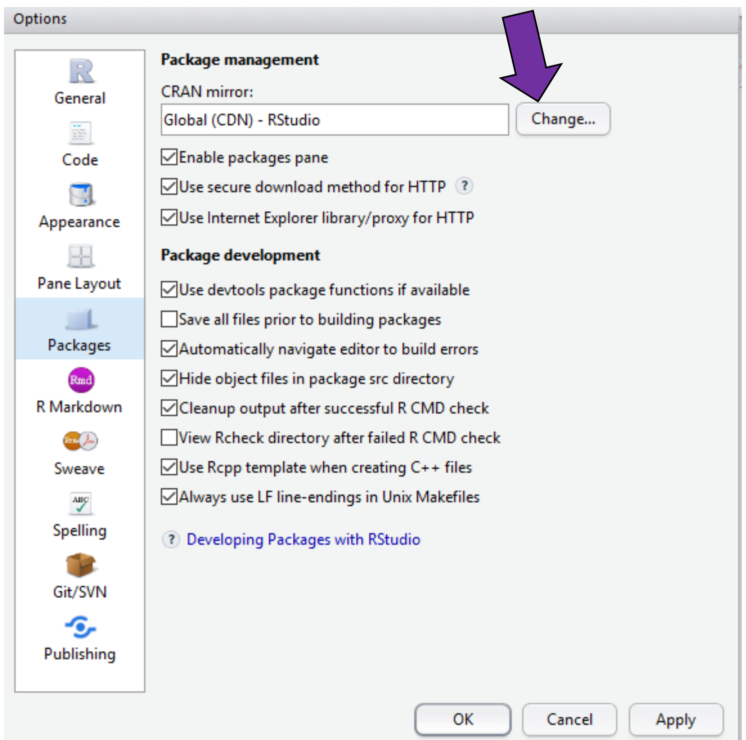
➢ Installing a package with the function install.packages()

with the name of the package between "quotes"

- by default from the CRAN miror repository of your choice. Historically, France(Lyon1) or France(Lyon2) were more exhaustive than France(Paris)

If working with Rstudio, by default Global (CDN) –Rstudio which is fine

Occasionally, you may change it by clicking in the Menu on Tools/Global options




```
> install.packages("qqman")
        # to install the qqman package
```

You may install several packages at once:

```
> install.packages(c("qqman", "MASS"))
# to install both qqman and MASS packages
```

getting all possible packages from CRAN using available.packages()

```
> dim(available.packages())[1]
# currently 15159 in Lyon1 and in Rstudio
```

Installing new R packages

➢ Installing a package via the **devtools** package

If you have to regularly install packages from different sources, the devtools package simplifies this process.

It includes specific functions for each repository including:

install_local()	from a local file
install_cran()	from CRAN
install_github()	from GitHub
install_url()	from a URL
install_bioc()	from BioConductor
…	

You may also use it to install a specific older version from CRAN:

```
install_version(package, version=NULL) # by default NULL installs the last version
```

And devtools is also a package to help packages developments!

Possible issues when installing package…and solutions!

1. Packages are not available for your current R version

You will have an error message when installing the library.

To overcome this issue, download either the source **tar.gz** if you are working on Unix, or the binaries for Windows or Mac if working on these OS.

Then rerun the installation by specifying the argument « repos=NULL » and providing the path of the downloaded file

You may also specify the library folder where to install it with the argument « lib » : see next issue

```
> install.packages("/mypath/qqman/qqman_0.1.2.tar.gz" , repos=NULL,
lib="mylibrarypath")
```

Possible issues when installing package…and solutions!

2. You are not allowed to install the library in the user library folder

You have not the rights to write within the folder. By default it starts with the first element returned by .libPaths(), then the second, etc…

In that case, by default R will offer you the possibility to install the library in a local user folder that it will create giving you the rights to write in

-> a question is asked to you: answer y for yes to allow this installation in your local/file/library folder

You may also want to install the package in a folder that already exists for which you have the rights to write in by specifying the argument « lib »

3. Errors occur when dependencies are not installed

The installation stops.

It often happens if the dependent packages are not available in your current R version. An error message will include the names of the packages that could not be loaded. Install them one by one as described in issues 1 and 2.

Managing R packages and their functions

To update packages to their latest version: update.packages()

To remove obsolete or useless packages: remove.packages()

Further considerations:

- If needed, you may have several R versions -> there will be several « bin » folders and their corresponding « library » folders

- If needed, you may have several versions of the same library:
Each version must be saved in a different folder. Then load the desired one with library() using its argument « lib.loc » to specify the folder of the library version

- If a function from a library does not perform exactly as wanted:
try to write your own function with its own name -> you may borrow most of the library function code: look at it by typing it without the () and adjust the function as needed *(example: treatment of NA values not always implemented…)*

Using installed R packages

- Using a function of an installed R package without loading the package using the notation packagename::functionname()
can be used if sporadic use of a few functions from the package instead of loading the full package

```
> gwasResults <- qqman::gwasResults # load preloaded simulated GWAS results in qqman
> qqman::qq(gwasResults) # calls the function qq from the package qqman to plot a
                         # qqplot of gwasResults
```

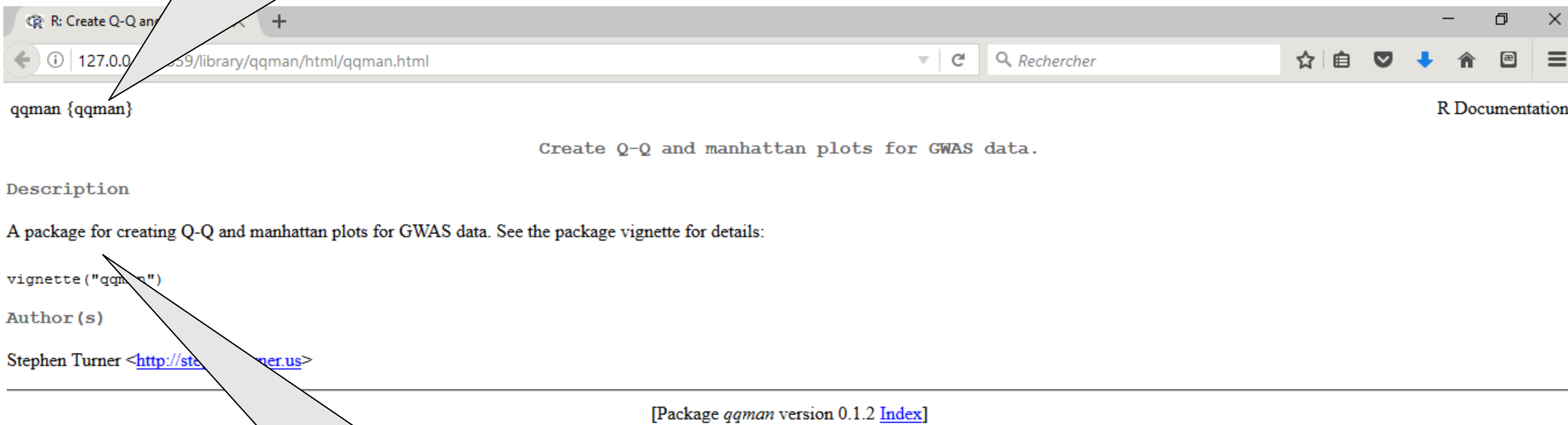
But to access to the documentation, you need to use library ()

```
> library(qqman)
> ?qqman              # only works for some packages
> qq(gwasResults$P) # same plot as previously, once the library is loaded
> manhattan(gwasResults) # manhattan plot of the results
```

Help on R packages

information on ‘qqman’ function

followed by {the package}


R: Create Q-Q an

127.0.0. 39/library/qqman/html/qqman.html

qqman {qqman} R Documentation

Create Q-Q and manhattan plots for GWAS data.

Description

A package for creating Q-Q and manhattan plots for GWAS data. See the package vignette for details:

```
vignette("qq n")
```

Author(s)

Stephen Turner <http://ste ner.us>

[Package *qqman* version 0.1.2 Index]

all packages have one or several vignettes describing their usage

= vignettes are the « user guides » ➡

vignette("package_name") directs to a webpage with package usage description

or to get all available vignettes of a package:

browseVignettes("package_name")

R packages from CRAN

Example with qqman:

CRAN - Package qqman

https://cran.r-project.org/web/packages/qqman/index.html

qqman: Q-Q and manhattan plots for GWAS data

Q-Q and manhattan plots for GWAS data

the minimal R version

Version:	0.1.2
Depends:	R (≥ 3.0.0)
Suggests:	knitr
Published:	2014-09-25
Author:	Stephen Turner
Maintainer:	Stephen Turner <vustephen at gmail.com>
License:	GPL-3
NeedsCompilation:	no
Materials:	README
CRAN checks:	qqman results

the manual describing each function within the package as when using help() or ?

the vignette describing the usage of the functions with some examples

Downloads:

Reference manual:	qqman.pdf
Vignettes:	Intro to the qqman package
Package source:	qqman_0.1.2.tar.gz
Windows binaries:	r-devel: qqman_0.1.2.zip, r-release: qqman_0.1.2.zip, r-oldrel: qqman_0.1.2.zip
OS X Mavericks binaries:	r-release: qqman_0.1.2.tgz, r-oldrel: qqman_0.1.2.tgz
Old sources:	qqman archive

other packages depending on this one

Reverse dependencies:

Reverse imports:	mrMLM, pweight
Reverse suggests:	solarius

the package source = that may be useful for custom installation

Linking:

R packages from CRAN

Example with ggplot2:

ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics

A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Version:	2.2.1
Depends:	R (≥ 3.1)
Imports:	digest, grid, gtable (≥ 0.1.1), MASS, plyr (≥ 1.7.1), reshape2, scales (≥ 0.4.1), stats, tibble, lazyeval
Suggests:	covr, ggplot2movies, hexbin, Hmisc, lattice, mapproj, maps, maptools, mgcv, multcomp, nlme, testthat (≥ 0.11.0), quantreg, knitr, rpart, rmarkdown, svglite
Enhances:	sp
Published:	2016-12-30
Author:	Hadley Wickham [aut, cre], Winston Chang [aut], RStudio [cph]
Maintainer:	Hadley Wickham <hadley at rstudio.com>
BugReports:	https://github.com/tidyverse/ggplot2/issues
License:	GPL-2 | file LICENSE
URL:	http://ggplot2.tidyverse.org, https://github.com/tidyverse/ggplot2
NeedsCompilation:	no
Citation:	ggplot2 citation info
Materials:	README NEWS
In views:	Graphics, Phylogenetics
CRAN checks:	ggplot2 results

dependencies if any

Downloads:

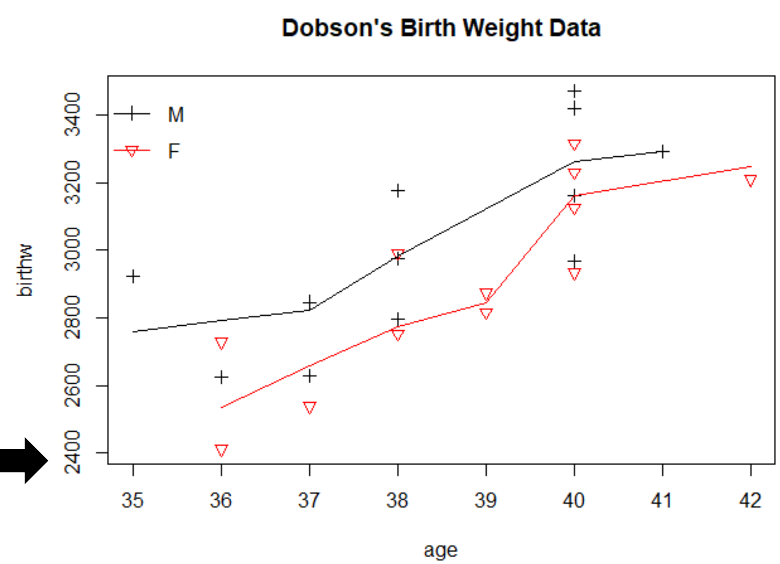
Reference manual:	ggplot2.pdf
Vignettes:	Extending ggplot2 Aesthetic specifications
Package source:	ggplot2_2.2.1.tar.gz
Windows binaries:	r-devel: ggplot2_2.2.1.zip, r-release: ggplot2_2.2.1.zip, r-oldrel: ggplot2_2.2.1.zip

Demo on R packages

- Some packages have a demo accessible with demo()

```
> demo(lm.glm, package="stats", ask=TRUE)


        demo(lm.glm)
        ---- ~~~~~~

Type  <Return>  to start :
```

```
> ### Examples from: "An Introduction to Statistical Modelling"
> ###                     By Annette Dobson
> ###
> ### == with some additions ==
>
> #  Copyright (C) 1997-2015 The R Core Team
>
> require(stats); require(graphics)

> ## Plant Weight Data (Page 9)
> ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)

> trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)

> group <- gl(2,10, labels=c("Ctl","Trt"))

> weight <- c(ctl,trt)

                              ...

> plot(age, birthw, col=as.numeric(sex), pch=3*as.numeric(sex),
+       main="Dobson's Birth Weight Data")
Hit <Return> to see next plot:
```



What's in Bioconductor?

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Search:

Home | Install | Help | Developers | About

Home » BiocViews

All Packages

Bioconductor version 3.4 (Release)

Autocomplete biocViews search:

- Software (1294)
  - AssayDomain (486)
  - BiologicalQuestion (462)
  - Infrastructure (277)
  - ResearchField (341)
  - StatisticalMethod (404)
  - Technology (815)
  - WorkflowStep (678)
- AnnotationData (939)
- ExperimentData (308)

Packages found under Software:

Show All entries

Search table:

Package	Maintainer	Title
a4	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Umbrella Package
a4Base	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Base Package
a4Classif	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Classification Package
a4Core	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Core Package
a4Preproc	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Preprocessing Package
a4Reporting	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Reporting Package
ABAEnrichment	Steffi Grote	Gene expression enrichment in human brain regions
ABarray	Yongming Andrew Sun	Microarray QA and statistical data analysis for Applied Biosystems Genome Survey Microrarray (AB1700) gene

Many packages in version 3.8

4 main Components

Software (1649)

AssayDomains (661)
BiologicalQuestion (668)
Infrastructure (360)
ResearchFiled (728)
StatisticalMethod (572)
Technology (1049)
WorkflowSetp (884)

Annotation Data (942)

ChipManufacturer (387)
ChipName (195)
CustomArray (2)
CustomDBSchema (4)
FunctionalAnnotation (29)
Organism (610)
SequenceAnnotation (1)

Experiment Data (360)

AssayDomainDara (61)
    including CNV, CpG, expression, SNPData…
DiseaseModel (86)
    including CancerData (83)
OrganismData (123)
    including A thaliana, E Coli, D Melanogaster, S Cerevisae, H Sapien, M musculus…
PackageTypeData (2)
RepositoryData(85)
    including ArrayExpres, ENCODE, GEO, 1KG…
ReproductibleResearch (16)
SpecimenSource (94)
    including CelleCulture, StemCell…
TechnologyData (230)
    including arrays, massspec, FACS, sequencing

Workflow (23)

AnnotationWorkflow (2)
BasicWorkflow (4)
EpigeneticsWorkflow (3)
GeneExpressionWorkflow (13)
GenomicVariantsWorkflow (13)
ImmunoOncology Workflow (2)
ResourceQueryingWorkflow (2)
SingleCellWorkflow (2)

A semi-annual release

Two coexisting versions both designed to work with a specific R version

- a released version
- a development version

Current: Bioconductor 3.10 October 31, 2019 working with with R>= 3.6

Previous versions archived for use with Bioconductor (R)

Release	Date	Software packages	R
3.10	October 30, 2019	1823	3.6
3.9	May 3, 2019	1741	3.6
3.8	October 31, 2018	1649	3.5
3.7	May 1, 2018	1560	3.5
3.6	October 31, 2017	1473	3.4
3.5	April 25, 2017	1383	3.4
3.4	October 18, 2016	1296	3.3
3.3	May 4, 2016	1211	3.3
3.2	October 14, 2015	1104	3.2
3.1	April 17, 2015	1024	3.2
3.0	October 14, 2014	934	3.1
2.14	April 14, 2014	824	3.1
2.13	October 15, 2013	749	3.0
2.12	April 4, 2013	671	3.0
2.11	October 3, 2012	610	2.15
2.10	April 2, 2012	554	2.15
2.9	November 1, 2011	517	2.14
2.8	April 14, 2011	466	2.13
2.7	October 18, 2010	418	2.12
2.6	April 23, 2010	389	2.11
2.5	October 28, 2009	352	2.10
2.4	April 21, 2009	320	2.9
2.3	October 22, 2008	294	2.8
2.2	May 1, 2008	260	2.7
2.1	October 8, 2007	233	2.6
2.0	April 26, 2007	214	2.5
1.9	October 4, 2006	188	2.4
1.8	April 27, 2006	172	2.3

Etc…

https://www.bioconductor.org/about/release-announcements/#release-announcements

Installing a bioconductor package

Obsolete: R versions <3.5

Installing the package -> it automatically adapts to your R version

```
# first install the Bioconductor installer package called "biocLite"
source("http://bioconductor.org/biocLite.R")
biocLite()# to install the minimum set of packages
biocLite("affy")# to install a specific package like "affy"
```

For R versions >= 3.5

Installing the package -> it automatically adapts to your R version

```
if (!requireNamespace("BiocManager"))
     install.packages("BiocManager") # to install the installer
BiocManager::install() # to install the minimum set of packages
BiocManager::install("affy")# to install a specific package like "affy"
```

Loading the package

```
library(affy)# load the package
library(affy,lib.loc=.libPaths()[1])#load the package from specific path
```

Some widely-used R functions and packages in genomics

For genomic intervals and annotations

- the rle() function: groups of consecutive values and counts their numbers
- IRanges: to store, manipulate and aggregate intervals on sequences
- GenomicRanges: serves as the foundation for representing genomic locations within the Bioconductor project
- biomaRt: to get genomic annotations tables and cross them
- Rctracklayer: to export/import/manipulate genome browser tracks in different formats

For genetic association studies:

- qqman: to perform QCs on GWAS data (manhattan and qqplots)

For microarray analyses:

- affy: to read affymetrix array data, to perform microarray normalisations
- limma: to perform differential expression analysis on microarrays (the gold-standard method) and now on RNASeq data

For NGS data:

- Rsamtools: as samtools in Unix to handel sam/bam files
- edgeR: normalization and differential expression of RNASeq data
- DESeq: normalization and differential expression of RNASeq data

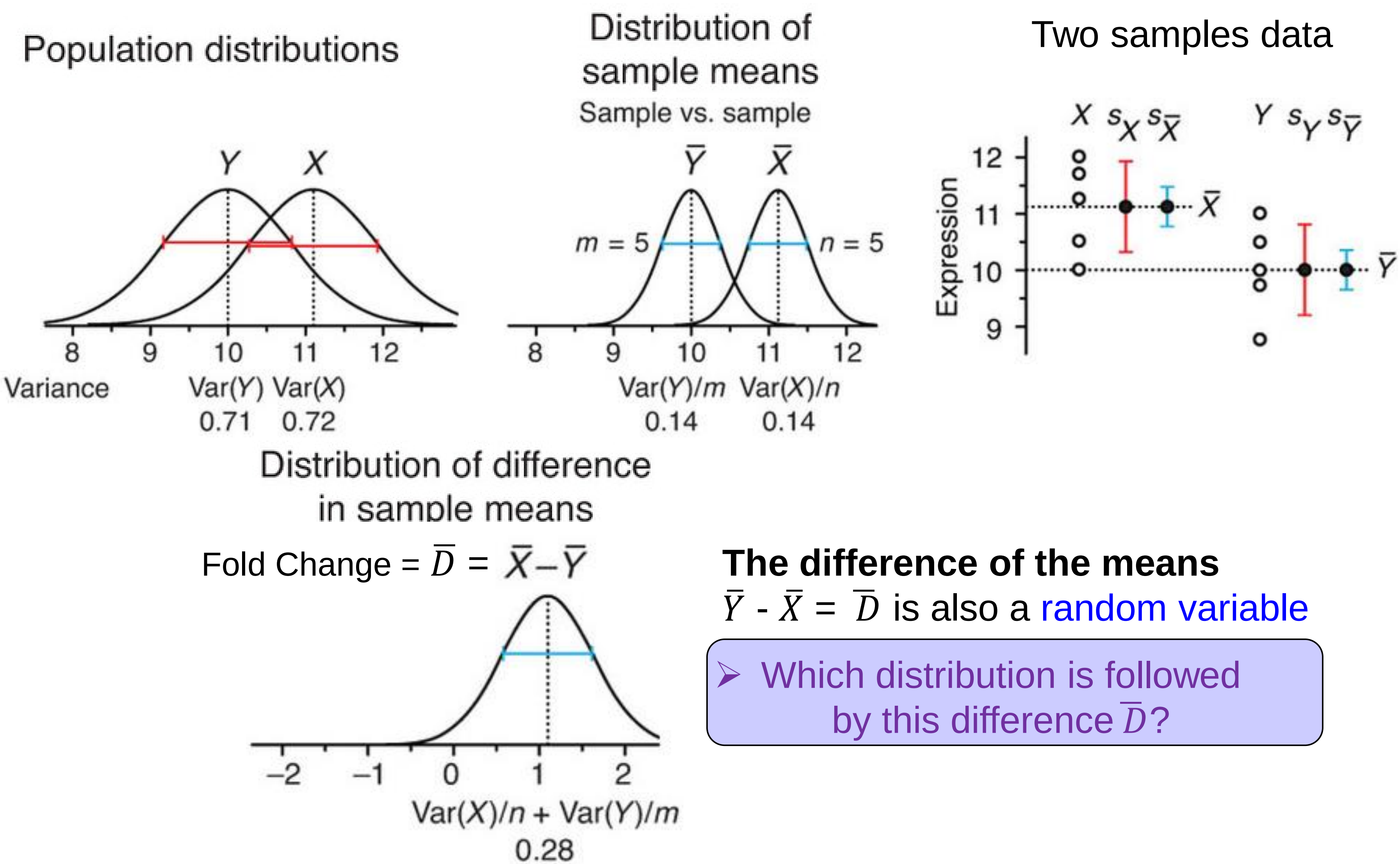
Practical:

install the package dabestr
https://github.com/ACCLAB/dabestr

4. Statistical tests

2nd aim = comparing population parameters

Comparing 2 populations X and Y with different means

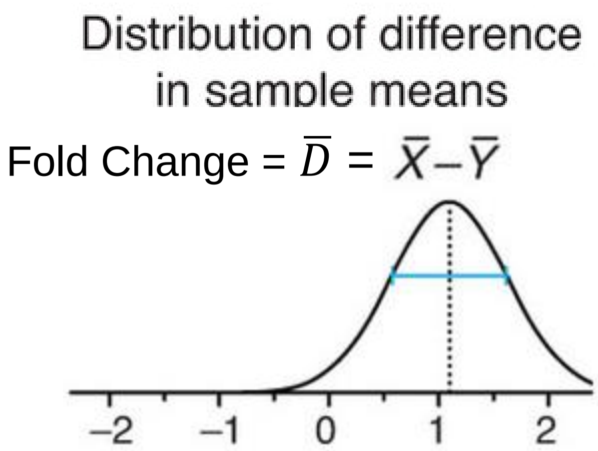


The difference of the means
$\bar{Y}$ - $\bar{X}$ = $\bar{D}$ is also a random variable

➢ Which distribution is followed by this difference $\bar{D}$?

2^{nd} aim = comparing population parameters

Comparing 2 populations X and Y with different means



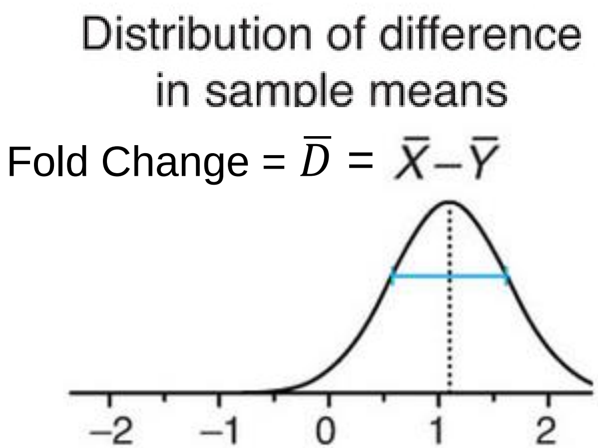
The difference of the means
$\overline{Y}$ - $\overline{X}$ = $\overline{D}$ is also a random variable

➢ Which distribution is followed by this difference $\overline{D}$?

H0: no difference
H1: there is a difference

2^{nd} aim = comparing population parameters

Comparing 2 populations X and Y with different means



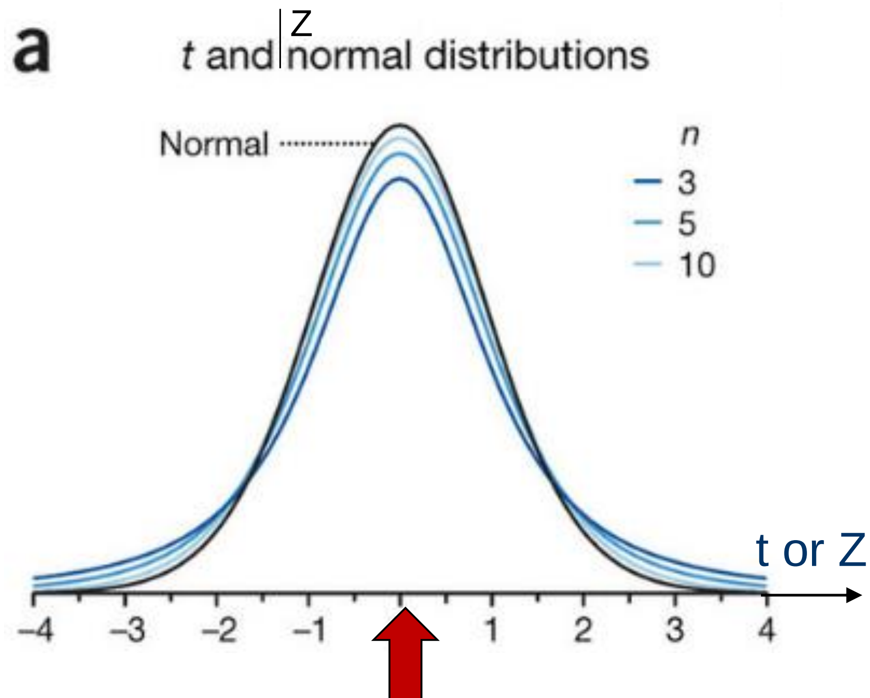
The difference of the means
$\bar{Y}$ - $\bar{X}$ = $\bar{D}$ is also a random variable

➢ Which distribution is followed by this difference $\overline{D}$?

H0: no difference
H1: there is a difference

0
Under H0: $\Delta = \mu 1 - \mu 2 = 0$
= the expected value (esperance) when there is no difference

Distribution of the difference of the means when there is none



0

Under H0: $\Delta = \mu 1 - \mu 2 = 0$
= the expected value (esperance)
when there is no difference

$\overline{D}$ can be centered on Δ
and reduced by its standard deviation

$$\text{Z or } t = \frac{\overbrace{(\bar{X}-\bar{Y})}^{\overline{D}} - \overbrace{(\mu_1-\mu_2)}^{\Delta}}{s_{\bar{X}-\bar{Y}}}$$

where $s^2_{\bar{X}-\bar{Y}} = s^2_{\bar{X}} + s^2_{\bar{Y}} \approx s^2_p/n + s^2_p/m$

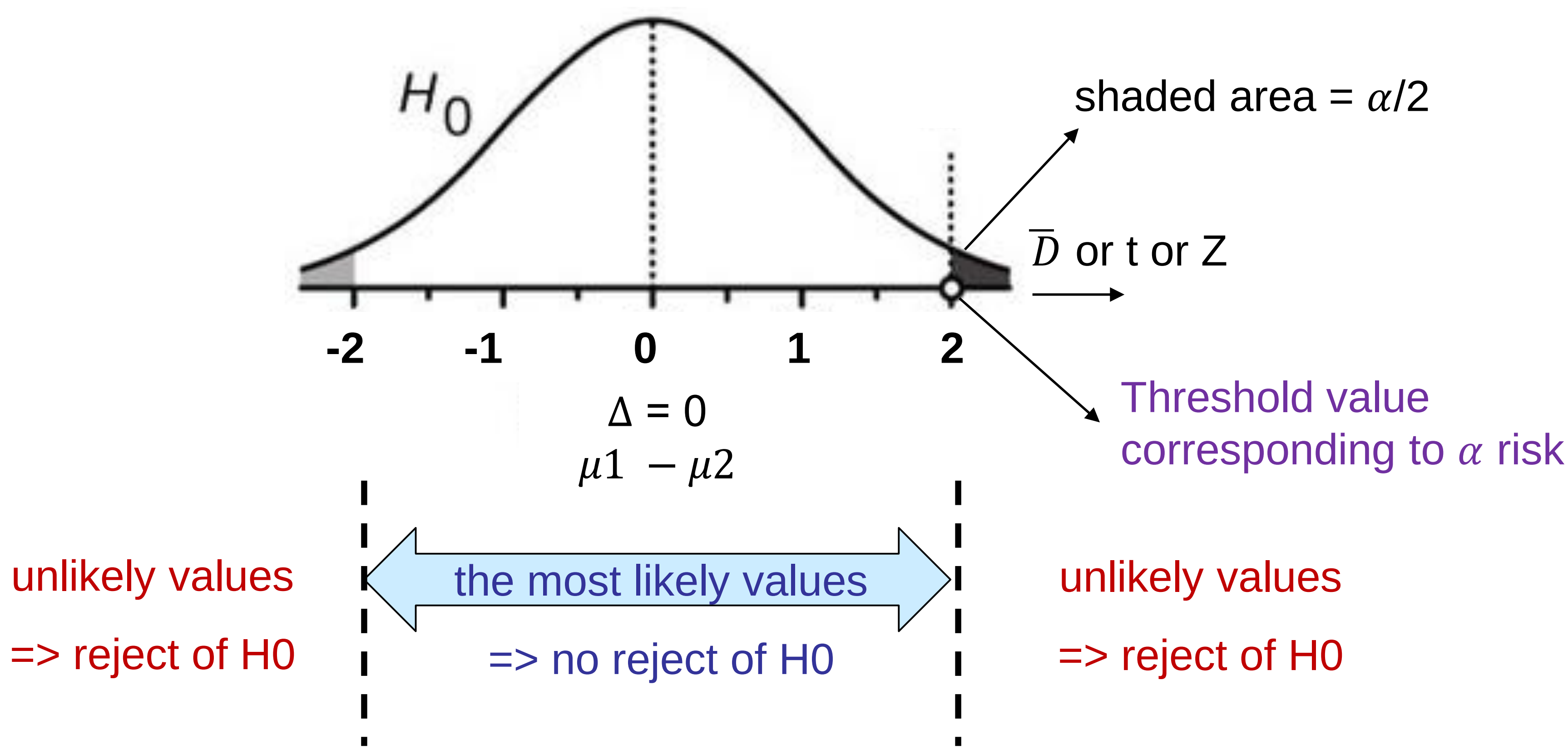
H0: no difference
H1: there is a difference

⇒ Z or t is a also random variable centered on 0 under H0

↳ How likely under the null hypothesis is the difference/statistics you observe?

Test theory : rejection criteria

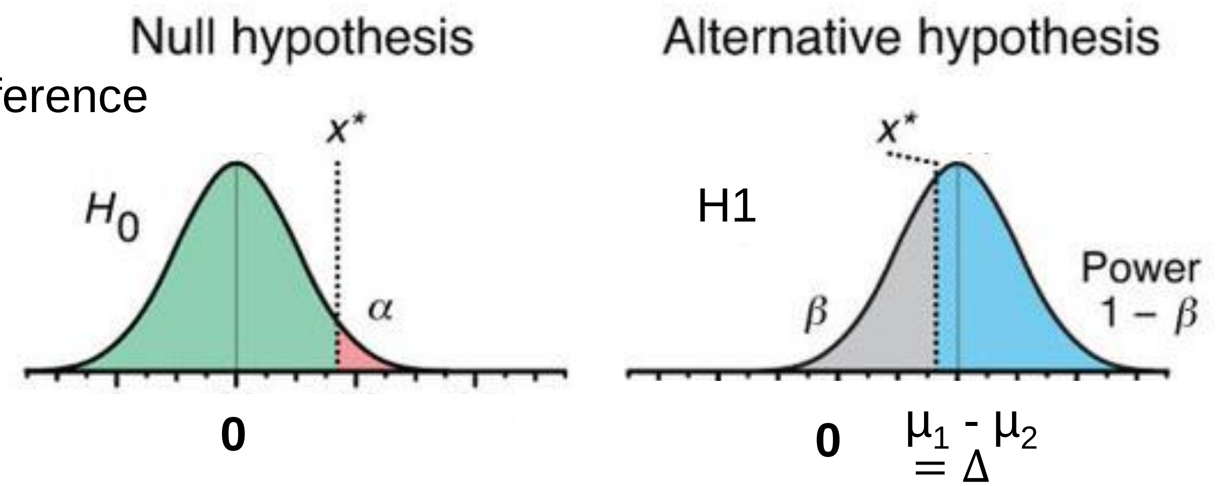
Probability of observing $\overline{D}$ or t or Z under H_0



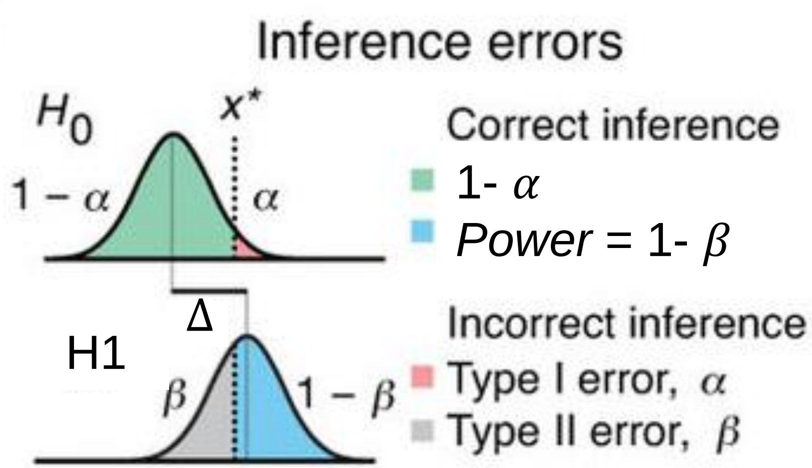
- ➢ Boundaries of the no reject area determined by alpha risk

Test theory : alpha and beta risks

= no difference
$\mu_1 = \mu_2$
$\Delta = 0$



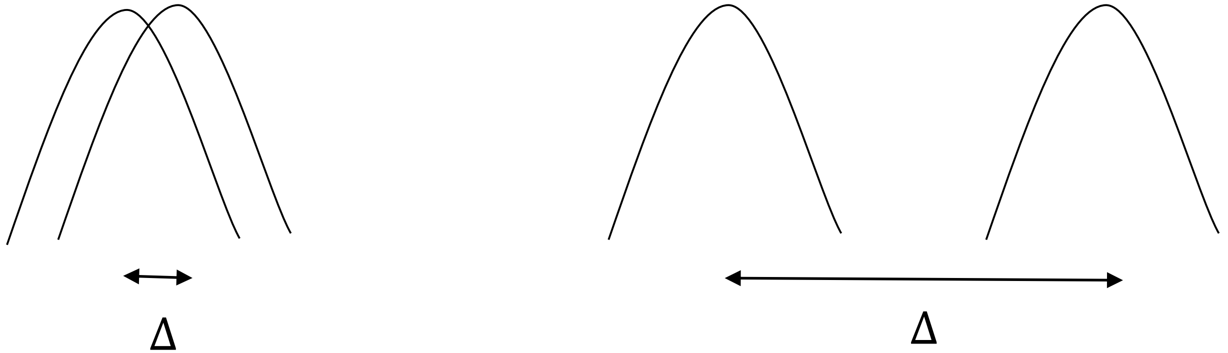
= difference
$\mu_1 \neq \mu_2$
$\Delta \neq 0$



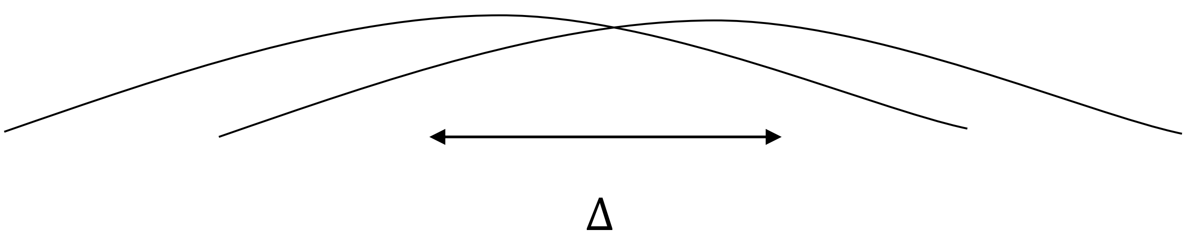
➢ Test decision	Reality H_0	Reality H_1
no reject of H_0	$1 - \alpha$ (TN)	β (FN)
reject of H_0	α (FP)	$1 - \beta$ (TP)

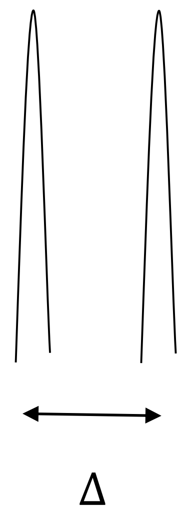
Impact on power

Power increases with effect size (Δ)



Power increases when standard deviation decreases



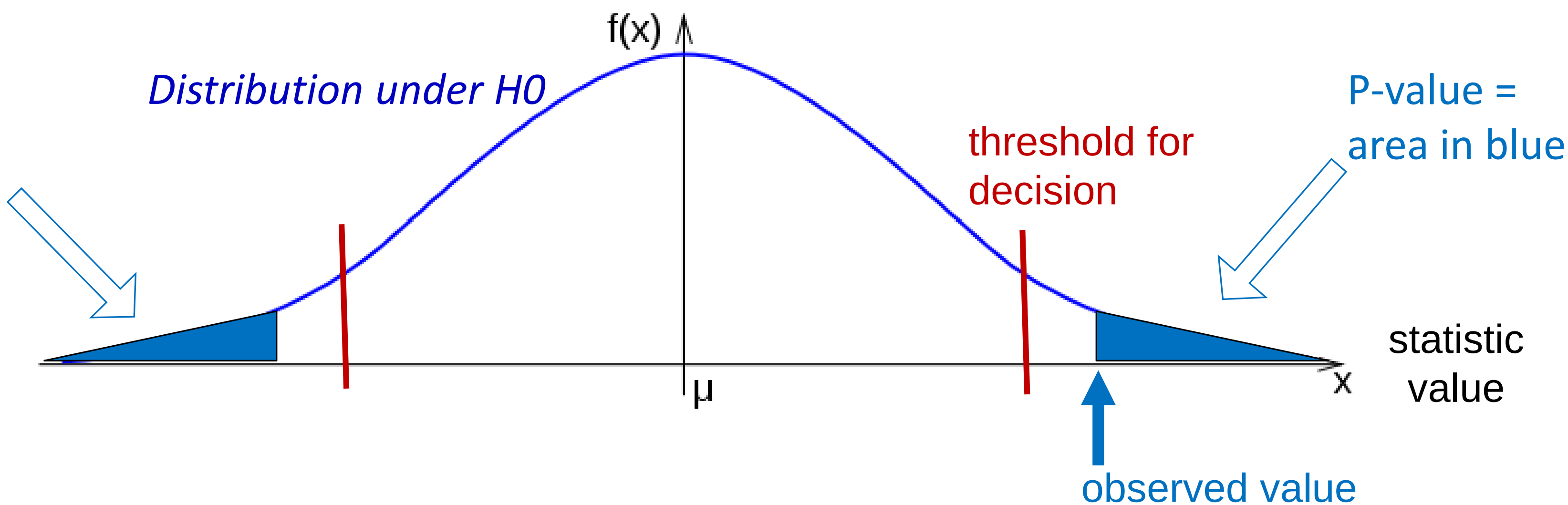


P-value

The p-value is defined as the probability to obtain, under H0, a value of the statistic (Student t, Z, Chi²…) at least as extreme as the observed value

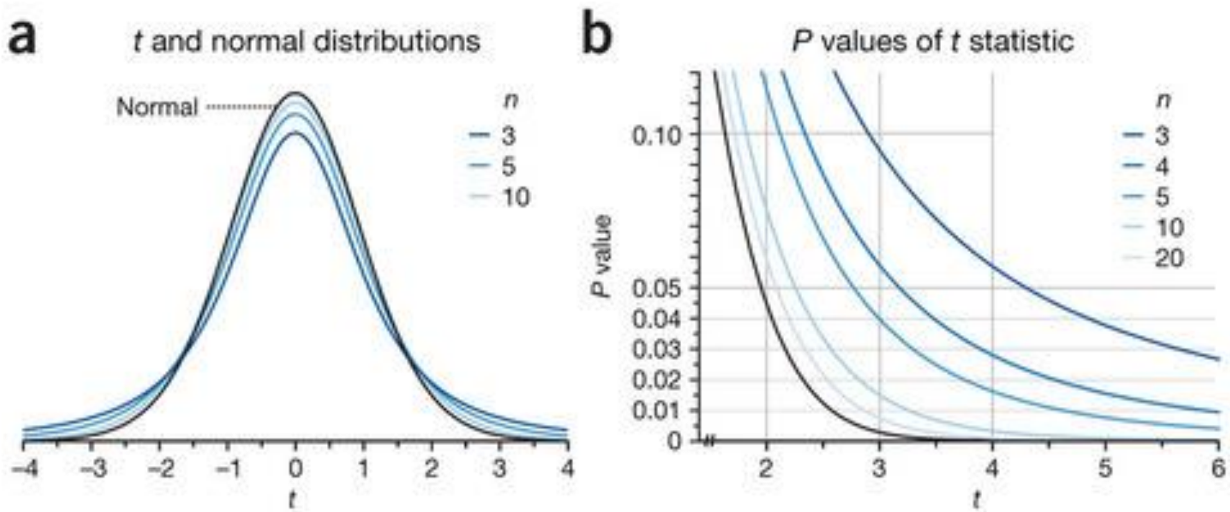
$$P(\ |statistics\ | > \text{observed value} / H_0) \leq \alpha$$

- report always your stat to have the direction effect + give CI of estimated effect size
- p-value is automatically computed by software but only to report if reject of H0, i.e significant test at the α risk (otherwise report NS for not significant)
- the higher your |stat| , the lower your-pvalue



P-value in a student test

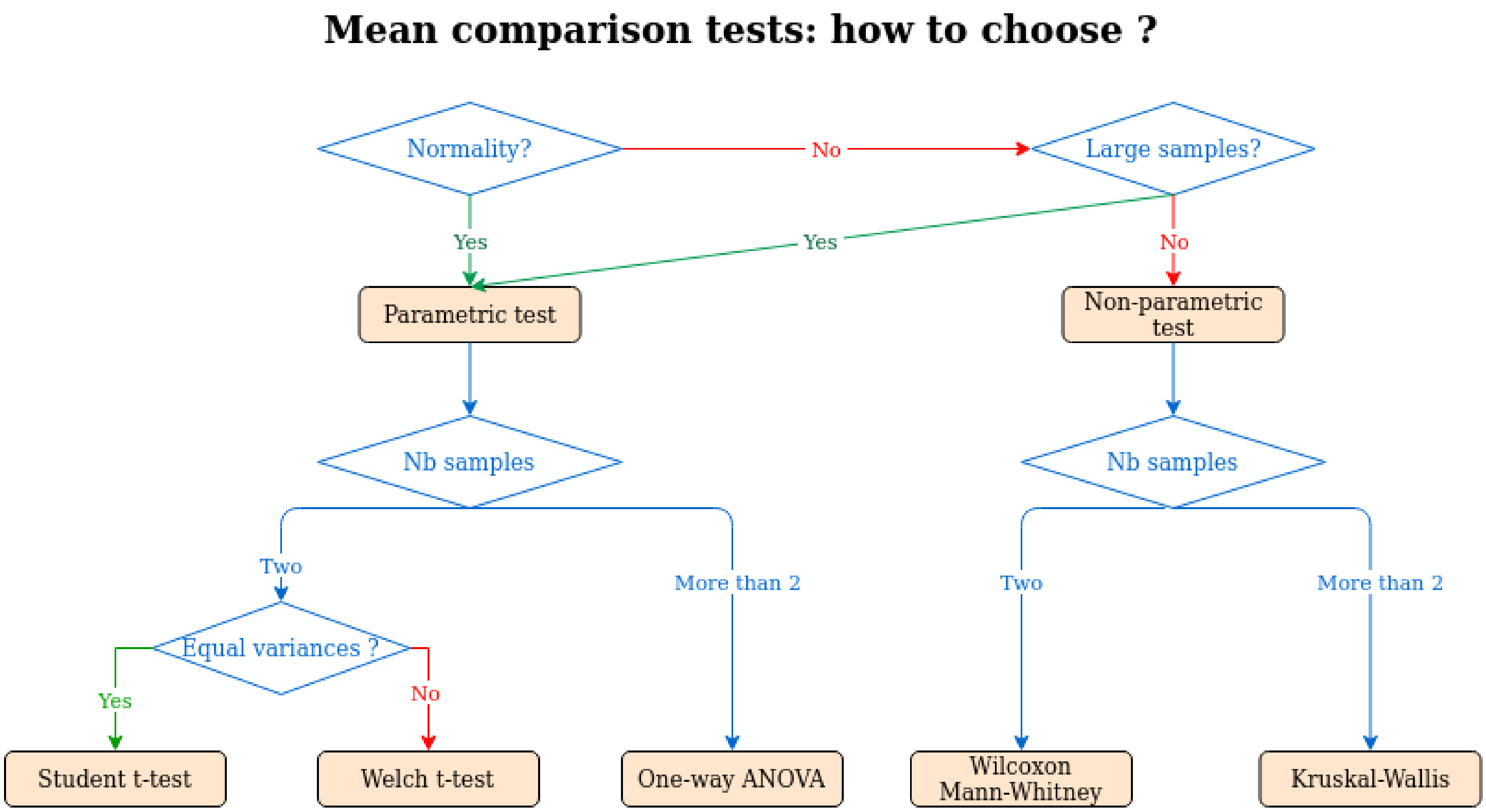
- the higher your stat (eg.|t|) , the lower your p-value, the higher your significacnce



t Table

cum. prob	$t_{.50}$	$t_{.75}$	$t_{.80}$	$t_{.85}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$	$t_{.999}$
one-tail	**0.50**	**0.25**	**0.20**	**0.15**	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**	**0.001**
two-tails	**1.00**	**0.50**	**0.40**	**0.30**	**0.20**	**0.10**	**0.05**	**0.02**	**0.01**	**0.002**
df										
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144

Which statistical to use?



Comparing more than 2 samples

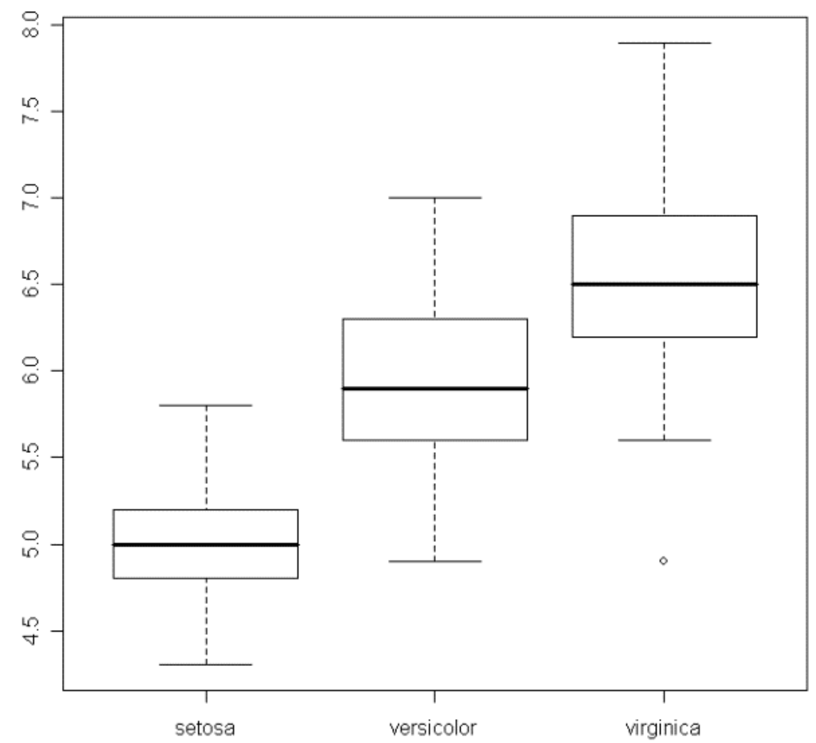
1. Perform a global test

= one-way ANOVA

H_0 : all population means are equal

H_1: at least one of the means differs

- the test compares the ratio of the variance among the sample means to the variance of each sample



2. If significant, perform pair-wise comparisons = post-hoc tests

Linear regression = perfect for more complex situations

It is useful to consider a model for the observed data (on a single trait)

$$Y = \mu + \alpha + \beta + \gamma + \ldots + \text{error}$$

eg. Microarray expression of a single gene Y =log2(intensity)

μ is the mean over all samples (all conditions)

error is the random error that is a mixture of measurement error and biological variability

the other terms are systematic deviations from the mean, due to the factors of interest (treatments, tissue…) and technical effects (batch, platform,…)

- We test the simplest model:

$$H_0: Y = \mu + \text{error} \quad \text{while } \boldsymbol{\alpha, \beta \ldots = 0}$$

=> Extendable to more complicated models with several factors and interactions

Example: testing a genetic variant on expression

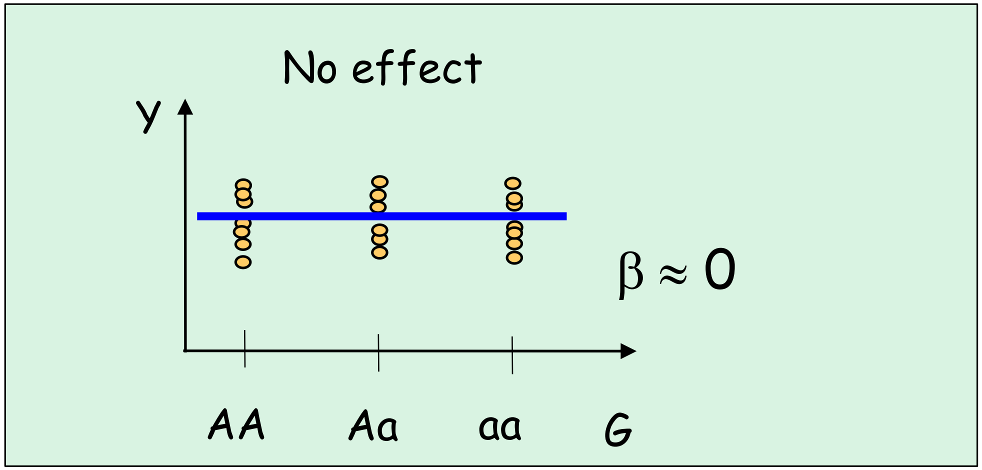
Y = expression
G = genotypes of a biallelic variant

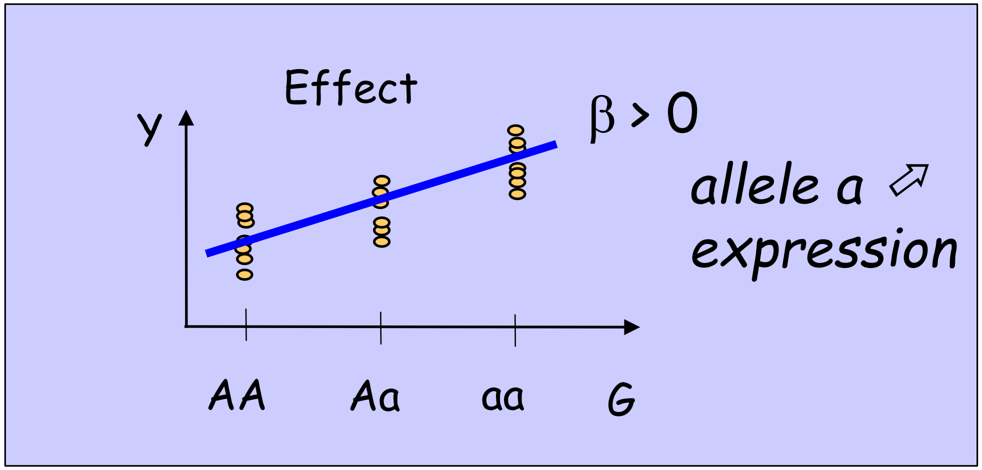
Model: $Y = \alpha + \beta G$

➢ **Test:**

H0: no effect (β=0)
H1: effect ($\beta \neq 0$)

$$t_{n-2} \sim \beta \,/\, \sigma^2_\beta$$





Further explanations on correlation and regression

See Document:

CovCorReg.pdf

Stats with R

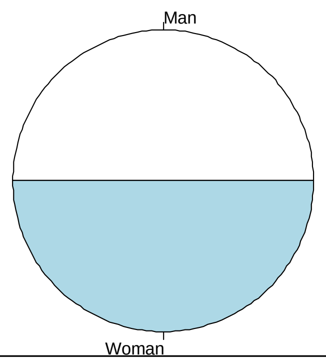
Some graph examples for qualitative variables

Cross-tabulations of occurrences using table()

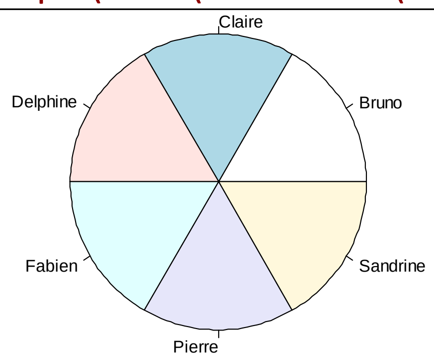
```
> table(myDataf$sex)
Man  Woman
  3      3
> table(myDataf$sex, row.names(myDataf)) # can be done on two or more variables
        Bruno Claire Delphine Fabien Pierre Sandrine
Man         1      0        0      1      1        0
Woman       0      1        1      0      0        1
```

Display proportions using pie() or barplot()

```
> pie(table(myDataf$sex))
```

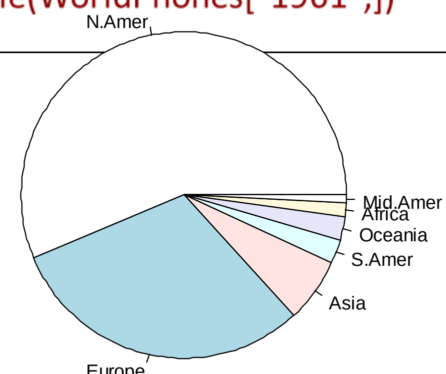



```
> pie(table(row.names(myDataf)))
```

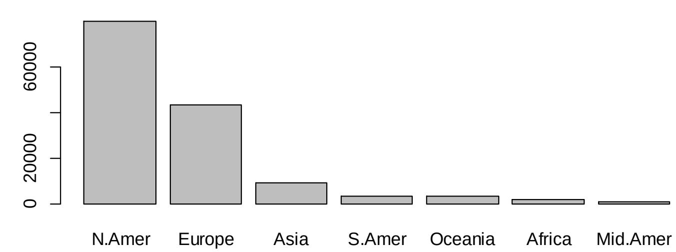



```
> data(WorldPhones)
> tail(WorldPhones)
     N.Amer Europe Asia S.Amer Oceania Africa Mid.Amer
1956  60423  29990 4708   2568    2366   1411      733
1957  64721  32510 5230   2695    2526   1546      773
1958  68484  35218 6662   2845    2691   1663      836
1959  71799  37598 6856   3000    2868   1769      911
1960  76036  40341 8220   3145    3054   1905     1008
196179831   43173 9053   3338    3224   2005     1076
```

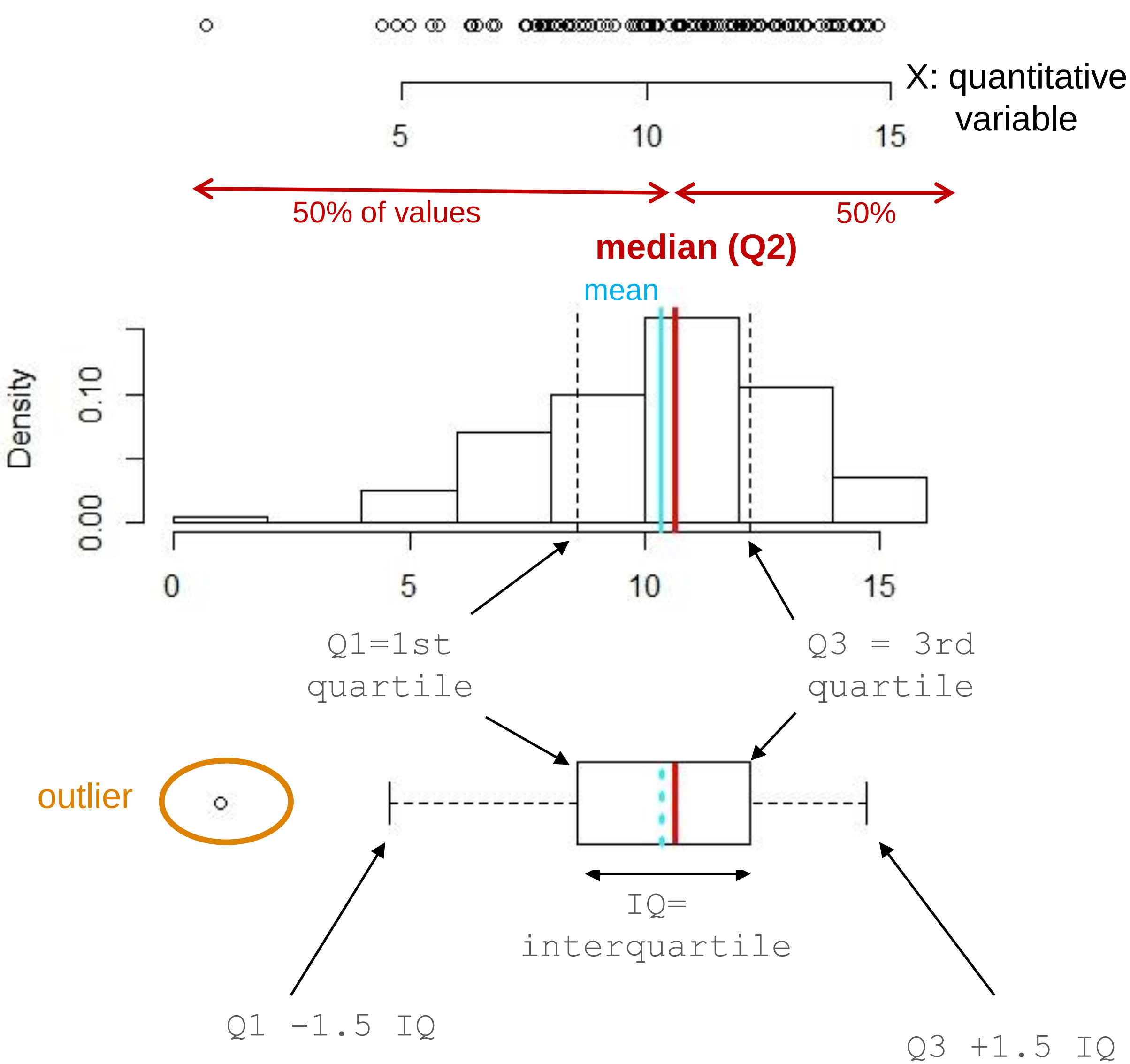
```
> pie(WorldPhones["1961",])
```




```
> barplot(WorldPhones["1961",])
```



Plotting distributions for continuous quantitative variables



stripchart()

« vertical » =F by default

hist()

« freq » = T by default to display counts while = F to display density

boxplot()

does not display the mean but the median

« range »=1.5 by default = k*IQ distance of whisker edges if 0: up to min and max, no outliers

« outlines »=T to display outliers by default, F to hide outliers

Some graph examples for quantitative variables

Example: the old faithful geyser
in Yellowstone National Park, USA

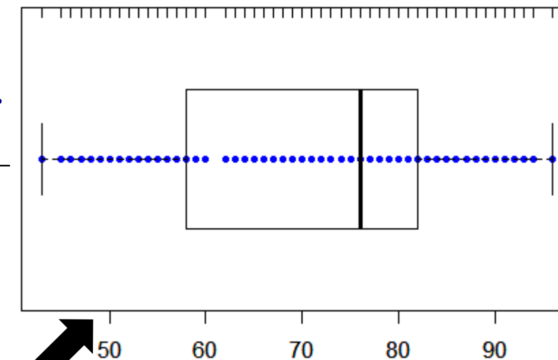
```
> data(faithful)
> str(faithful)
'data.frame':   272 obs. of  2 variables:
 $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
 $ waiting  : num  79 54 74 62 85 55 88 85 51 85 .
> ?faithful
```

Format

A data frame with 272 observations on 2 variables.

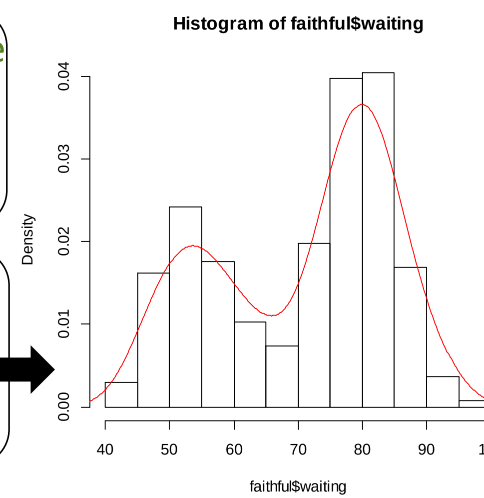
[,1] eruptions numeric Eruption time in mins

[,2] waiting numeric Waiting time to next eruption (in mins)




```
> stripchart(faithful$waiting, col="blue", pch=20) # col is a parameter used inside
> boxplot(faithful$waiting, horizontal=T, add=T) # add=T to superpose graphs
> rug(faithful$waiting, side=3) # example of secondary function
```

```
> hist(faithful$waiting, freq=F)      # freq=T to display counts,
                                      # freq =F to display frequencies!
> lines(density(faithful$waiting),col="red")
```

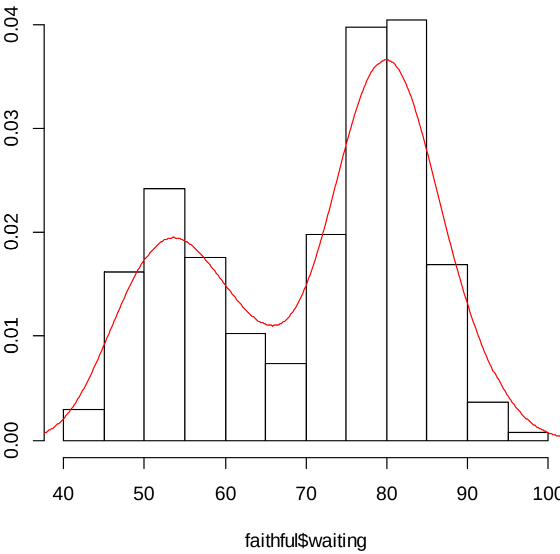


Frequency distributions of quantitative variables

Descriptive statistics:

add na.rm=T if NA values except for summary

```
> range(faithful$waiting)
[1] 43 96
> mean(faithful$waiting)
[1] 70.89706
> sd(faithful$waiting)
[1] 13.59497
> summary(faithful$waiting)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   43.0    58.0    76.0    70.9    82.0    96.0
> median(faithful$waiting)
[1] 76
```



Getting the quantile values of a distribution with the quantile() function

```
> quantile(faithful$waiting, probs=0.5)
[1] 76
> quantile(faithful$waiting, 0.1)
10%
 51
> quantile(faithful$waiting, c(0.1,0.9))
10% 90%
 51  86
> quantile(faithful$waiting, seq(0,1,0.1))
  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
  43   51   55   60   71   76   78   81   83   86   96
```

Why not using barplots for quantitative data?

A MUST READ THREAD:

https://twitter.com/T_Weissgerber/status/1040576802979233793

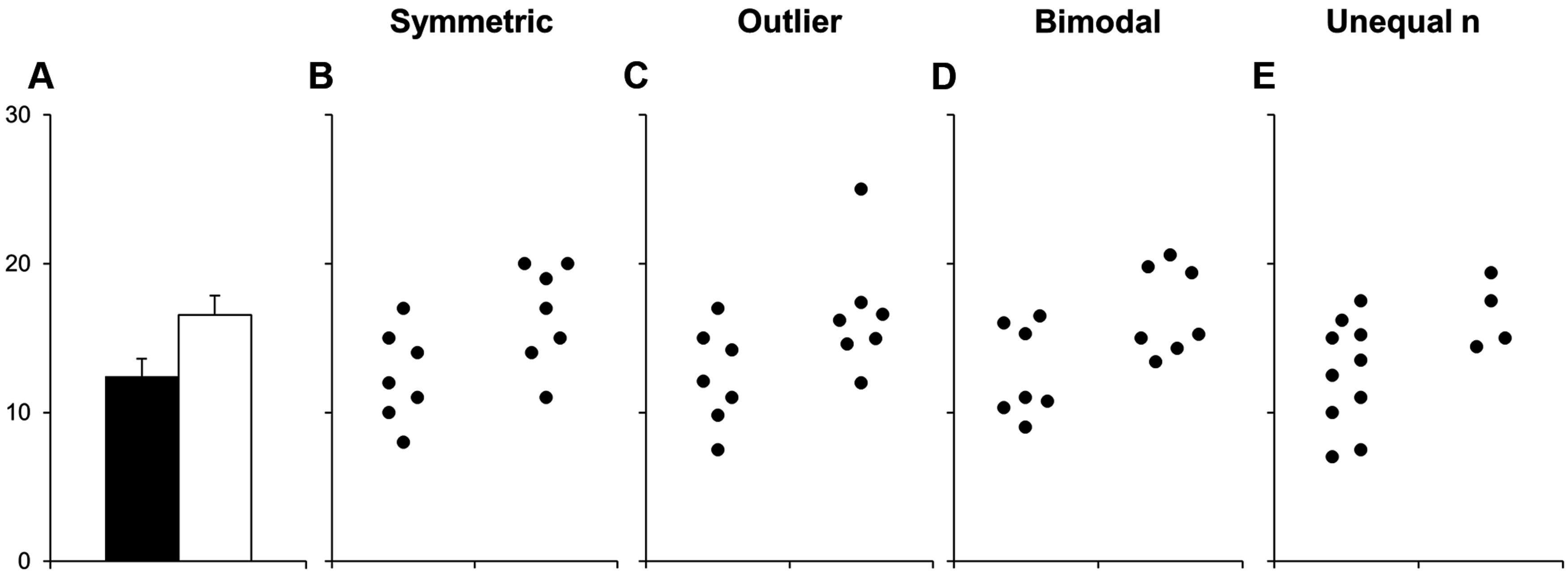
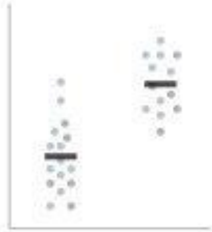
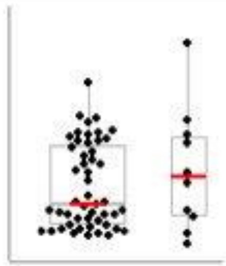
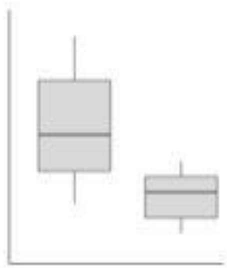
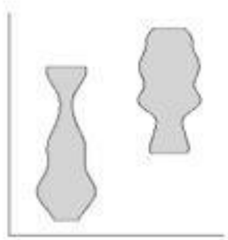
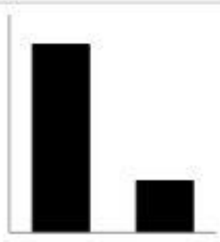


Figure Types	Example	Type of Variable	What the Plot Shows	Sample Size	Data Distribution	Best Practices
Dot plot		Continuous	Individual data points & mean or median line Other summary statistics (i.e. error bars) can be added for larger samples	Very small OR small; can also be useful with medium samples	Sample size is too small to determine data distribution OR Any data distribution	• Make all data points visible - use symmetric jittering • Many groups: Increase white space between groups, emphasize summary statistics & de-emphasize points • Only add error bars if the sample size is large enough to avoid creating a false sense of certainty • Avoid "histograms with dots"
Dot plot with box plot or violin plot		Continuous	Combination of dot plot & box plot or violin plot (see descriptions above and below)	Medium	Any	• Make all data points visible (symmetric jittering) • Smaller n: Emphasize data points and de-emphasize box plot, delete box plot and show only median line for groups with very small n • Larger n: Emphasize box plot and de-emphasize points
Box plot		Continuous	Horizontal lines on box: 75^{th}, 50^{th} (median) and 25^{th} percentile Whiskers: varies; often most extreme data points that are not outliers Dots above or below whiskers: outliers	Large	Do not use for bimodal data	• List sample size below group name on x-axis • Specify what whiskers represent in legend
Violin plot		Continuous	Gives an estimated outline of the data distribution. The precision of the outline increases with increasing sample size.	Large	Any	• List sample size below group name on x-axis • The violin plot should not include biologically impossible values
Bar graph		Counts or proportions	Bar height shows the value of the count or proportion	Any	Any	• **Do not use for continuous data**

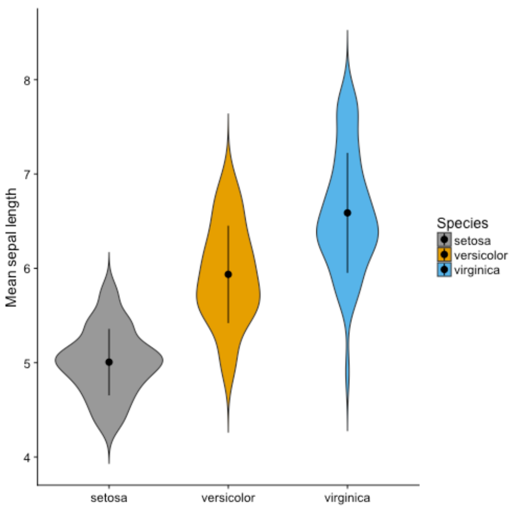
Source: Tracey Weissgerber
Twitter: @T_Weissgerber

Alternative to barplots

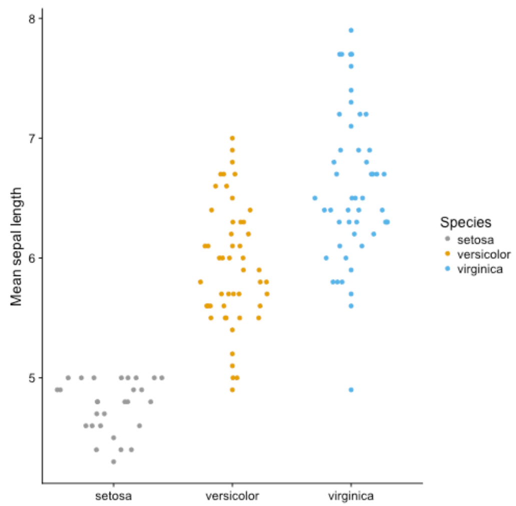
https://audhalbritter.com/alternatives-to-barplots/

https://cran.r-project.org/web/packages/sinaplot/vignettes/SinaPlot.html

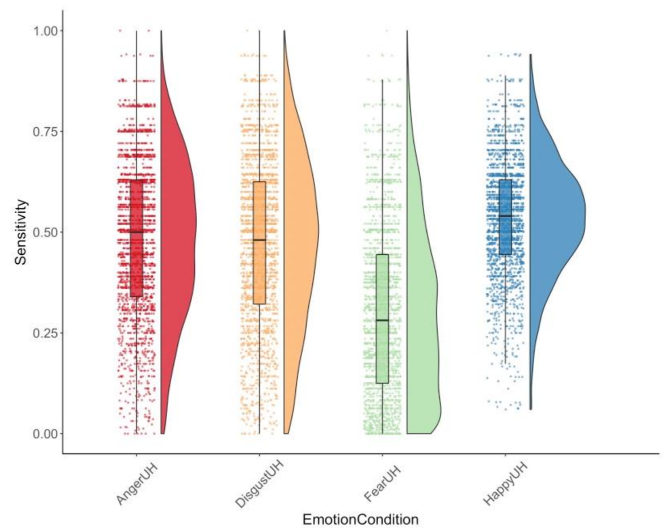
https://micahallen.org/2018/03/15/introducing-raincloud-plots/



violon plots



sina plots



raincloud plots

Known probablility laws of random variables

For a given probability law, the corresponding R name is:

(cf Quick R: http://www.statmethods.net/advgraphs/probability.html)

Discrete Distributions	R name
Binomial	binom
Poisson	pois
Negative binomial	nbinom

Ccontinuous Distributions	R name
Uniform	unif
Normal	norm
Student t	t
Chisquare	chisq
Fisher F	f
Exponential	exp

Getting **random** values drawn from the law using r*name*()

Getting the **quantile** values of a known probability law using q*name*()

Getting the **density** function using d*name*()

Getting the **cumulative distribution function** using p*name*()

Examples for discrete laws

Getting **random values** drawn from the law using r*name*()

```
> rbinom(n=10,size=3,prob=0.5)      # returns 10 values (results) from a binomial distribution of size 3 (nb of
[1] 1 2 2 2 1 1 3 2 2 3             # of attempts) with a probability of sucess of each attempt of 0.5
> rpois(10, 0.2)                    # returns 10 values from a poisson distribution of parameter lambda=0.2
 [1] 0 0 0 0 0 0 0 0 0 1
```

Getting the **density function** using d*name*()

-> returns the probability of a specific discrete value k : P(X = k)

```
> dbinom(2, 3, 0.5)
[1] 0.375              # the probability of getting 2 from a binom of size 3 and proba 0.5
> dpois(1, 0.2)
[1] 0.1637462          # the probability of getting 1 from a poisson distribution of lambda 0.2
```

Getting the **cumulative density function** using p*name*()

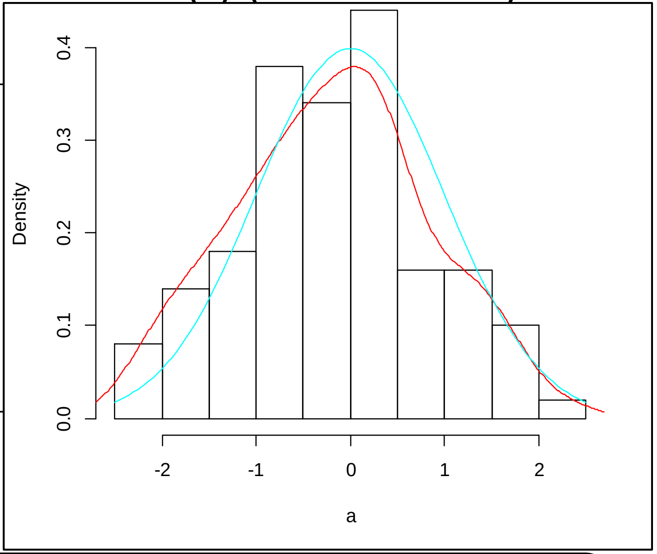
-> returns the cumulative probability P(X ≤ k) = P(X=0)+(PX=1)+…+P(X=k)

```
> pbinom(2, 3, 0.5)
[1] 0.875        # the probability of getting values ≤ 2 from a binom of size 3 and proba 0.5
> ppois(3, 0.2)
[1] 0.9999432    # the probability of getting  values ≤ 3 from a poisson distribution of lambda 0.2
```

Examples of continuous variables

Getting the **density** function using d*name*()

-> returns the distribution = the value of the probability distribution f(x) (on the Y axis) for x (on the x axis)




```
> a <- rnorm(100)
> hist(a,freq=F)
> lines(density(a), col="red")       # the density of random data
                                     # drawn from a normal distribution
> curve(dnorm(x),add=T, col="cyan") # the norm distribution itself!
```

Getting the values corresponding to the **quantiles**

```
> qnorm(p=c(0.025,0.5),mean=0,sd=1, lower.tail=T)
[1] -1.959964  0.000000  # values k such as P(x ≤ k) = 2.5% or 50% of the data
```

Getting the **cumulative distribution function** of a known probability law using p*name*()

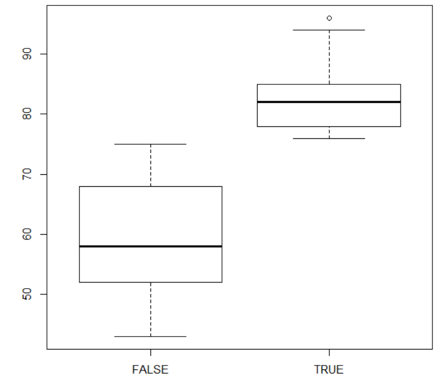
```
> pnorm(c(1.96,0),mean=0,sd=1, lower.tail=F)
[1] 0.0249979 0.5000000   # probabilities of getting a value of X ≤ k, P(X ≤ k)
```

Statistics examples for a continuous variable

Mean comparison

example with faithful data split in two categoriez according to the median value

```
> faithful$category <- faithful$waiting >= median(faithful$waiting)
> boxplot(faithful$waiting ~ faithful$category)
```



- parametric t test:

```
> t.test(faithful$waiting ~ faithful$category)
        Welch Two Sample t-test

data:  faithful$waiting by faithful$category
t = -25.605, df = 189.77, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -24.66955 -21.14053
sample estimates:
mean in group FALSE  mean in group TRUE 
           59.27612            82.18116 
> 2*pt(-25.605, 189.77,lower.tail=T) # returns the pvalue and not just < 2.2e-16
[1] 1.789806e-63
```

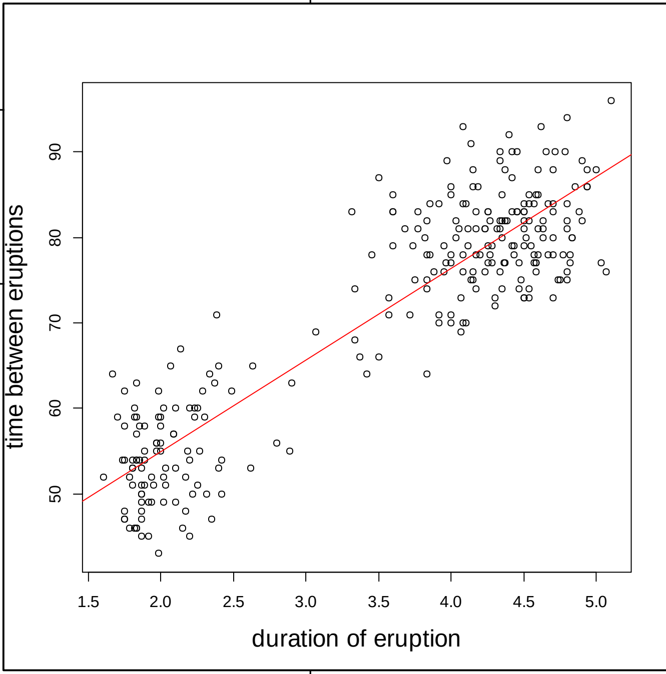
- non-parametric t test:

```
> wilcox.test(faithful$waiting ~ faithful$category)
        Wilcoxon rank sum test with continuity correction

data:  faithful$waiting by faithful$category
W = 0, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Graph and statistical test examples for quantitative variables

Scatter plot:

```
> plot(faithful$eruptions, faithful$waiting, xlab="duration of eruption", ylab="time between
     eruptions" , cex.lab=1.5)
> abline(lm(faithful$waiting~faithful$eruptions), col="red")
```



Linear Regression model

```
> summary(lm(faithful$waiting~faithful$eruptions))

Call:
lm(formula = faithful$waiting ~ faithful$eruptions)

Residuals:
     Min       1Q   Median       3Q      Max
-12.0796  -4.4831   0.2122   3.9246  15.9719

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         33.4744     1.1549   28.98   <2e-16 ***
faithful$eruptions  10.7296     0.3148   34.09   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.914 on 270 degrees of freedom
Multiple R-squared:  0.8115,    Adjusted R-squared:  0.8108
F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

Graph and statistical test examples for quantitative variables

Correlation tests between two continuous varaibles

- parametric test:

```
> cor.test(faithful$eruptions, faithful$waiting) # by default parametric Pearson correlation test

        Pearson's product-moment correlation

data:  faithful$eruptions and faithful$waiting
t = 34.089, df = 270, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8756964 0.9210652
sample estimates:
      cor
0.9008112
> str(cor.test(faithful$eruptions, faithful$waiting))
List of 9
 $ statistic  : Named num 34.1
  ..- attr(*, "names")= chr "t"
 $ parameter  : Named int 270
  ..- attr(*, "names")= chr "df"
 $ p.value    : num 0
 $ estimate   : Named num 0.901
  ..- attr(*, "names")= chr "cor«
$ conf.int   : atomic [1:2] 0.876 0.921
...
> cor.test(faithful$eruptions, faithful$waiting)$estimate^2 # the same determination coeff as with lm!
      cor
0.8114608
```

Graph and statistical test examples for quantitative variables

Correlation tests between two continuous variables

- non -parametric test:

```
> cor.test(faithful$eruptions, faithful$waiting, method="s") # Spearman is the non-parametric cor test

        Spearman's rank correlation rho

data:  faithful$eruptions and faithful$waiting
S = 744660, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.7779721

Warning message:
In cor.test.default(faithful$eruptions, faithful$waiting, method = "s")
:
  Cannot compute exact p-value with ties
```

Practicals:

A_first_data_analysis.html

and finish

descriptive-statictics.html