Rbasics

PhD toolbox - 39th PhD cycle

Part IV - Tips and Tricks

Summary

- Organize your working directory and your script
- Manage files & folders
- Tips on Warning/errors
- R documentation and manuals

Practical tips for organizing scripts, files and folders

- working directory concept
  - setwd("path/to/folder/")

- a working directory for each research project/experiment/analysis ... I suggest to avoid use the default R working directory. After a bit you will have an incomprensible multitude of input files, outputs, from different projects, analysis, errors and reconsiderations ... trust me I already lived it XD.

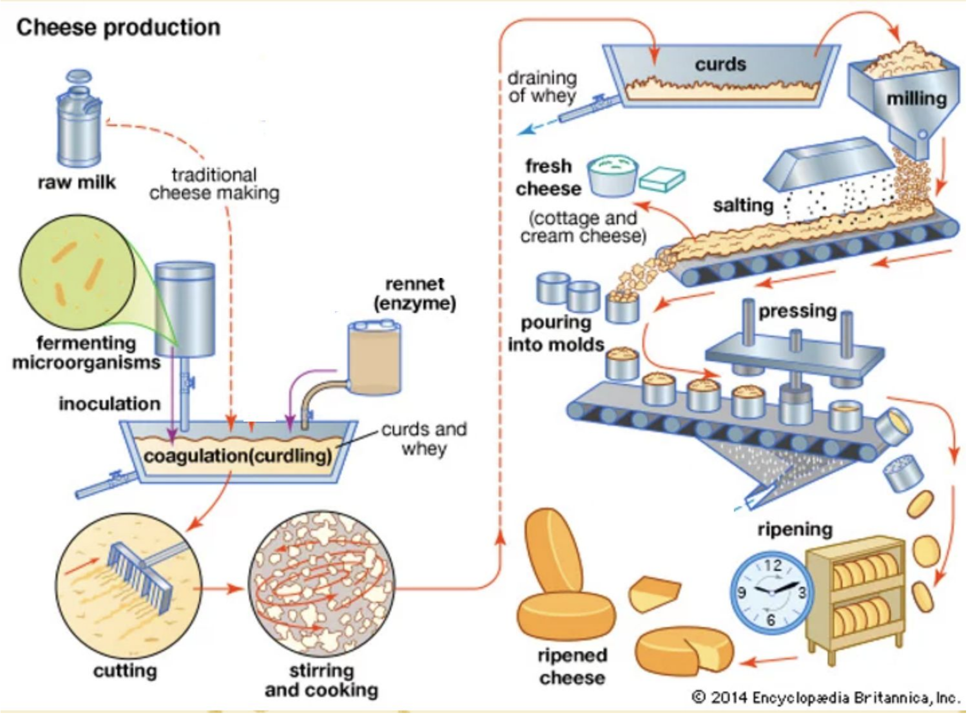
- long analysis need order (teutonic strengthness!) it is easy to create bugs in mis-organized scripts.

  - script "chapterization" using

    ```
    # 1. Chapter 1 ------------------------------
    ## 1.1 Sub-chapter --------------------------
    or
    #### 1. Chapter 1 ####
    ##### 1.1 Sub-chapter #####
    ```

    this will create tracking points in your script

practical tips for organizing scripts, files and folders



© 2014 Encyclopædia Britannica, Inc.

- long analysis needs order (tip2). Long scripts (I mean > 500 rows) are good for braggarts: several scripts generating intermediate results are better.

- only you know where to stop! my suggestion (actually is an example)

script 1: data loading + basic transformations + preliminary data surveys

script 2: main analysis with transformed data

script 3: cool figures

Your project folder

README file
explains the content of the project and the folders/files contained in the project's main directory

RAW data folder
contains the untouched raw data. Ideally should have read-only permissions.
They must be stored in a very safe place, and possibly have a recovery backup.
Have you made them available after publication? This prevents to have your materials lost and is increasingly required by journals

Scripts folder
The tidy collection of all your script used in the several steps of data analysis

Analysis (or Results) folder
Contains the outputs of your analyses (elaborated tables, figures, etc..)

practical tips for organizing scripts, files and folders

- R scripts should be published to address reproducible research requirements. Don’t do the assholes, sharing is good for scientific community!

- You can add a nice header to your scripts including:
  - Paper title
  - Script author(s)
  - A brief explanation on its usage, if it is needed
  - R and Rstudio version you used
  - List of packages and their version

practical tips for organizing scripts, files and folders

```
# Herbonaute
# Adamo M., Marmeisse R.
# R script by M. Adamo

# R version 4.2.2 Patched (2022-11-10 r83330) -- "Innocent and Trusting"
# Copyright (C) 2022 The R Foundation for Statistical Computing
# Platform: x86_64-pc-linux-gnu (64-bit)

# used libraries -------------------------------------------------------
library("raster")        # Geographic data analysis and modeling v3.5-29
library("ggbiplot")      # A ggplot2 based biplot v0.55
library("factoextra")    # Multivariate data analysis v1.0.7
…
```

Repositories

Some good places where to deposit your data (raw, elaborated or scripts)

GitHub
figshare
zenodo
GitLab
DRYAD
NCBI

panic! - warnings *vs* errors -

# Warnings: Your function was able to run but there are some conditions that needs to be checked to avoid possible issues

```
> matrix(data = sample(seq(1,30)), nrow = 10, ncol = 5, byrow = F)
     [,1] [,2] [,3] [,4] [,5]
 [1,]    3   19    9    3   19
 [2,]    7   27   25    7   27
 [3,]   13    1   22   13    1
 [4,]   17   16    6   17   16

Warning message:
In matrix(data = sample(seq(1, 30)), nrow = 10, ncol = 5, byrow = F) :
  data length differs from size of matrix: [30 != 10 x 5]
```

panic! - warnings *vs* errors -

# Errors: Your command cannot be executed

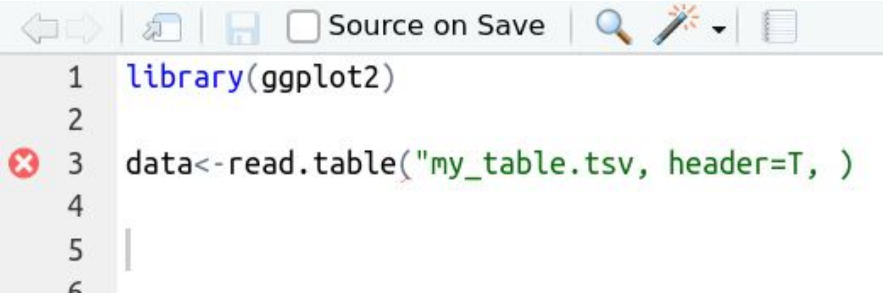
```
> library(unicorns)
Error in library(unicorns) : there is no package called ‘unicorns’
```

Several reasons behind:

- there is a typo in your command
- the function you typed does not exist or the library containing it has not be loaded
- the object name you typed has some typos or does not still exist
- your command does not comply with the R language syntax
  - e.g. there are unclosed brackets/quotation marks

panic! - warnings *vs* errors

Some useful tips:



```
library(ggplot2)

data<-read.table("my_table.tsv, header=T, )

```

- RStudio most of the times tells you that something is wrong with the syntaxis

- sometimes function names overlap between different packages. You need to get aware of that and force the function from the right package:

`dplyr::filter`

- functions can be updated over the time and their usage can have changed from the last time you used it. No way back: the best solution is to integrate the new function in the script.

panic! - Other strategies

- bug fixing
  - it is usually possible when problems are simple and/or you’re experienced
  - check the presence of NAs or other lacks in the data
  - check the format of your input data
- manuals
  - almost all packages have the help page
  - some package have dedicated web-pages and online tutorials. *e.g.*:

    http://www.ggtern.com/docs/

panic! - Other strategies

- forums
    - "Google knows!"
    - you will probably ground in one of the several R-specialized forums
- AI chat
    - ChatGPT is able to create and explain good parts of R code … Ask it! …

      Otherwise you must know R to understand if the reply is good or not

I get an error message that I don’t understand:

- Googling the error message
- Check Stack Overflow forum pages
- Ask your skilled colleagues (do not abuse of this solution)
- Ask on R-help mailing list

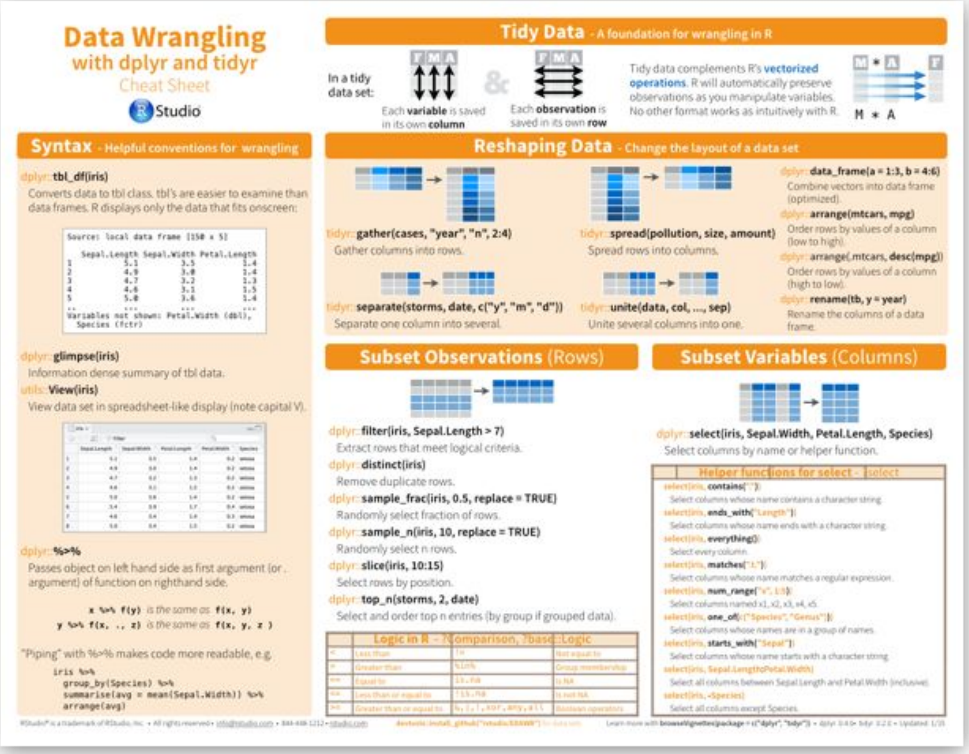
Note: Most of the time some other else already encountered your same issue, or a bug is already known (and not fixed, sure!)

How to make easier to help you

- Use the **correct terms** to describe your problem
- Try to **generalize** what you are doing
- Include the output of **sessionInfo**() in your requests
- Most of the forums allow user to add **example code sections** that can be easily handled by other participants.
- Try to share a **reproducible example**, including your input data!

  see here for further ideas: http://adv-r.had.co.nz/Reproducibility.html

Cheatsheets



Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**
Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**
Information dense summary of tbl data.

utils::**View(iris)**
View data set in spreadsheet-like display (note capital V).

dplyr::**%>%**
Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` is the same as `f(x, y)`
`y %>% f(x, ., z)` is the same as `f(x, y, z )`

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**
Gather columns into rows.

tidyr::**spread(pollution, size, amount)**
Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**
Separate one column into several.

tidyr::**unite(data, col, ..., sep)**
Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**
Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**
Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**
Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**
Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.

dplyr::**distinct(iris)**
Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.

dplyr::**slice(iris, 10:15)**
Select rows by position.

dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic			
<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.

select(iris, **everything()**)
Select every column.

select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

RStudio® is a trademark of RStudio, Inc. • All rights reserved • info@rstudio.com • 844-448-1212 • rstudio.com

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with **browseVignettes(package = c("dplyr", "tidyr"))** • dplyr 0.4.0 • tidyr 0.2.0 • Updated: 1/15

You cannot known in detail all the function of a given package:

Cheatshets ("bigliettino") help you in summarizing the main functions and their usage!

Rbasics

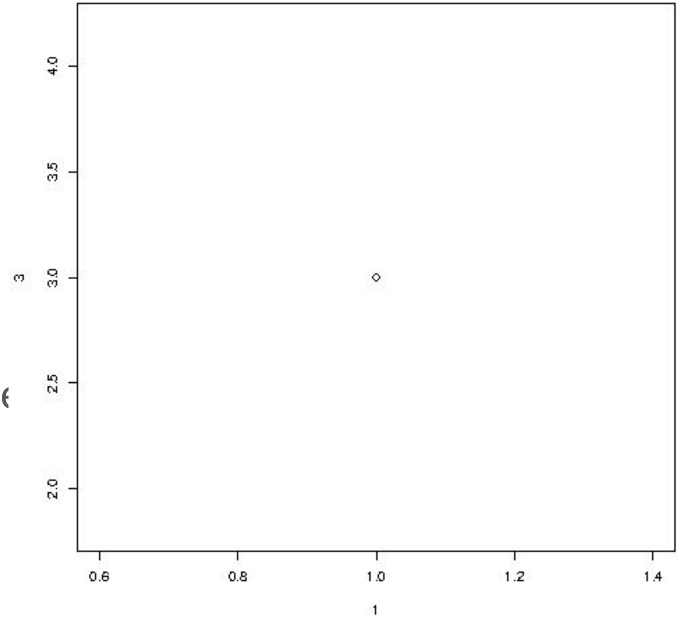
PhD toolbox - 39th PhD cycle

Part V - Base graphics in R (some tips)

Part V - Base graphics in R (some tips)

the base function to create graphics is plot() it simply creates a Cartesian plane where you can plot your data.

```
>plot(1,3)

or

>plot(x,y)
```



EXERCISE:

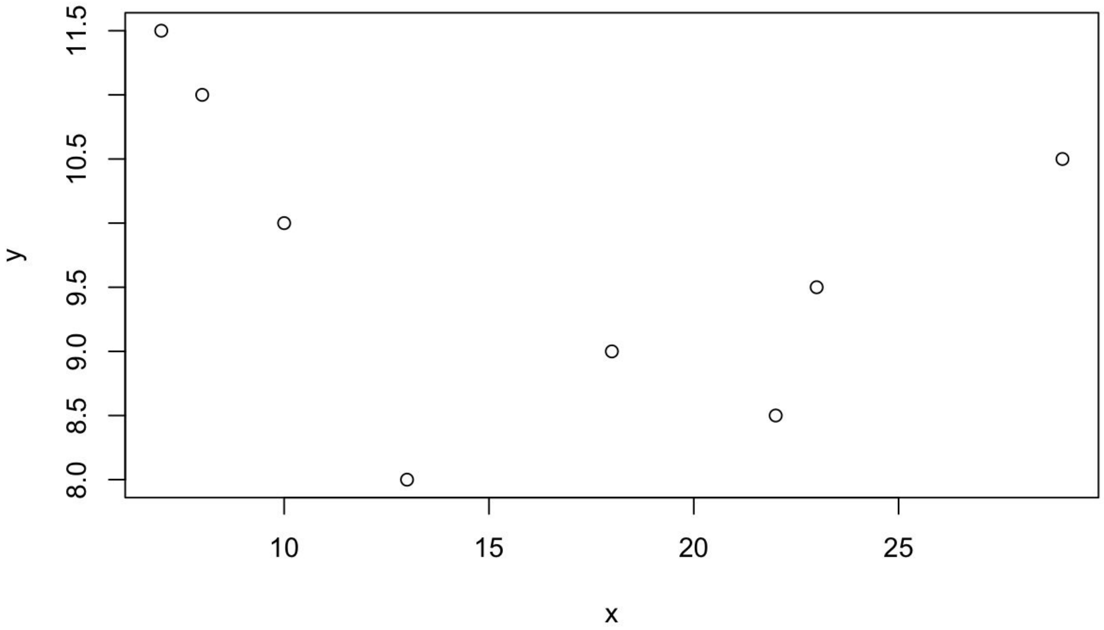
Substitute x and y with colname_2 and colname_3 ve

from the *df* object.

Base graphics in R

Exercise Solution:

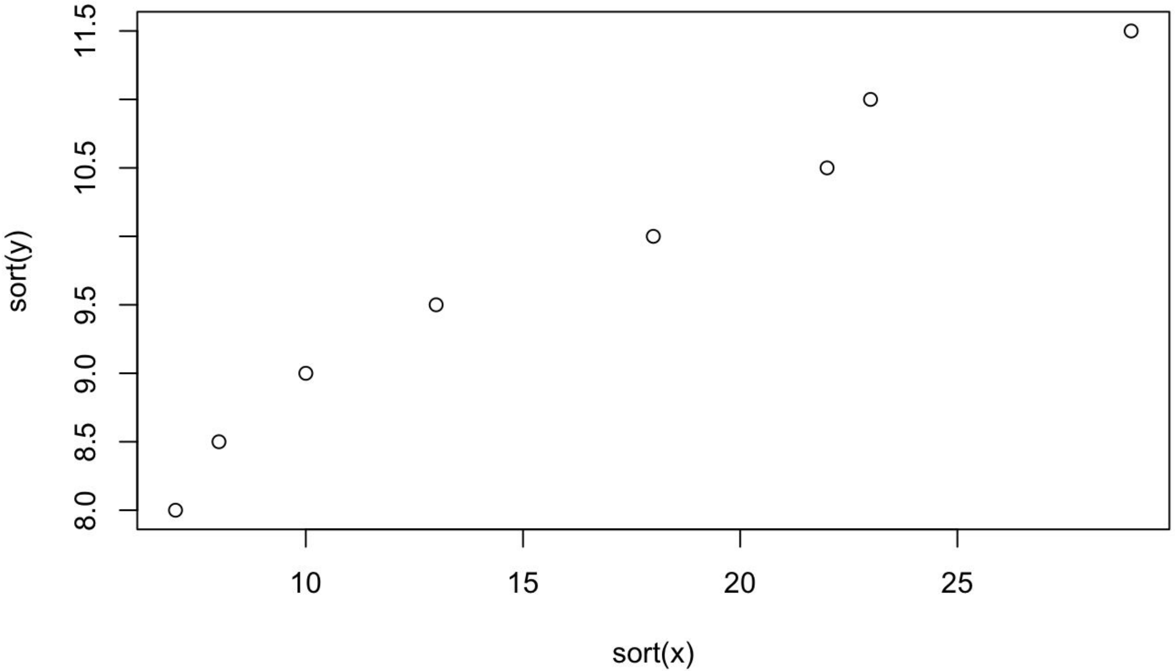
```
> x = df$colname_2

> y = df$colname_3

> plot(x,y)
```



Base graphics in R

you can modify vectors directly before plotting to ameliorate the graphical output

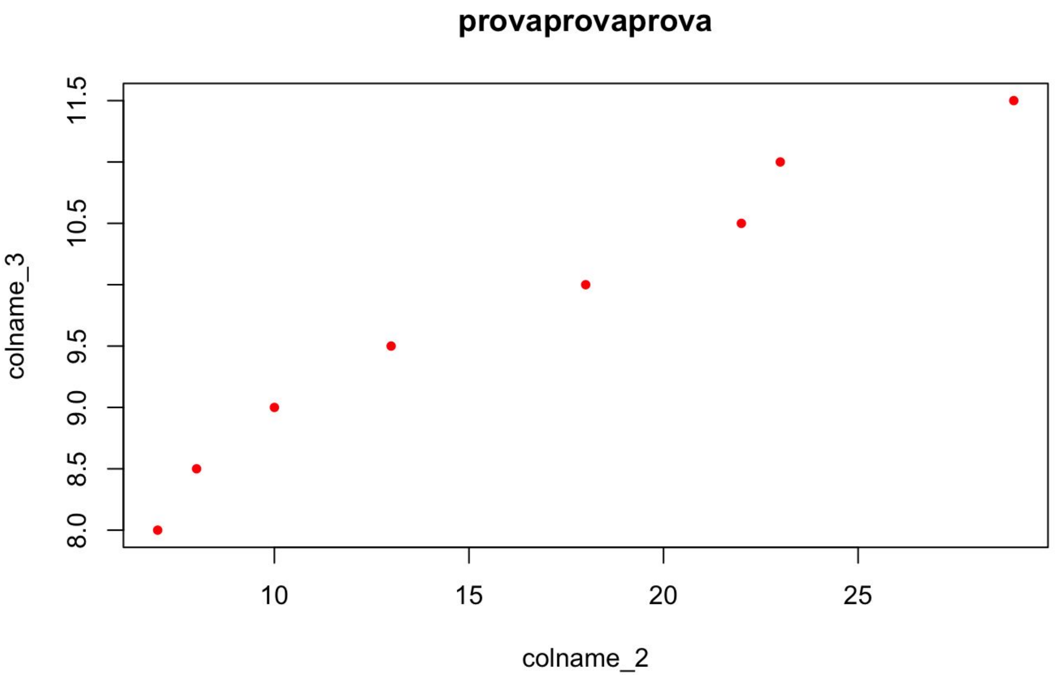
```
> plot(sort(y)~sort(x))
```



Base graphics in R

Plots can be modified in many different ways (most of Radvance program)

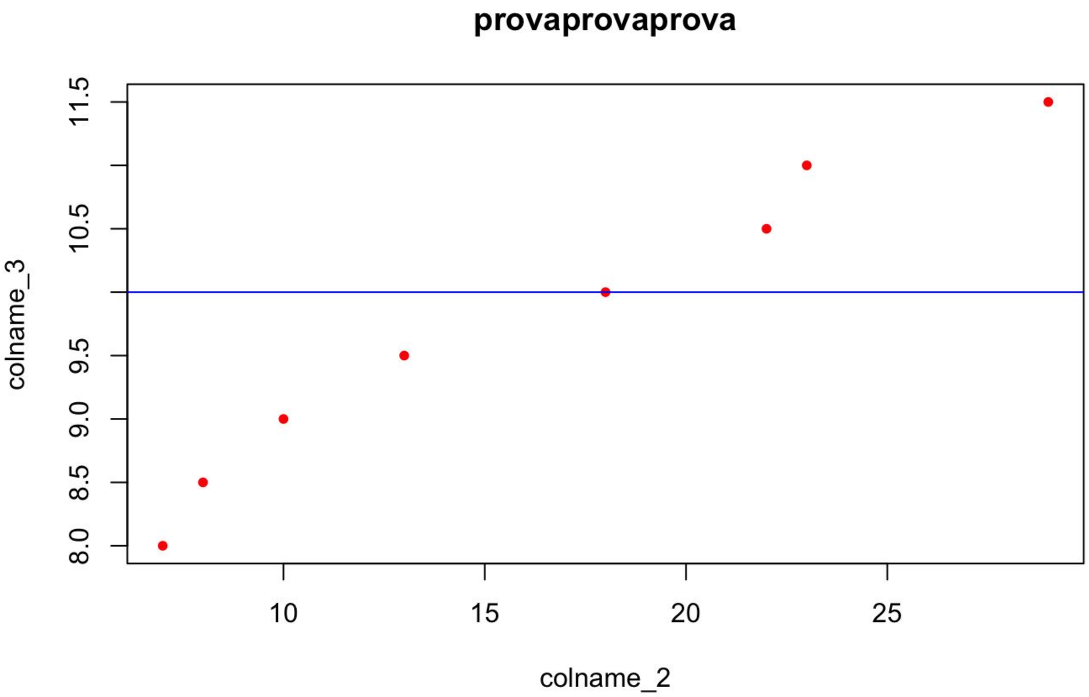
```
> plot(sort(y)~sort(x),

     col = "red",

     pch = 20,

     main = "provaprovaprova",

     ylab = "colname_3",

     xlab = "colname_2")
```



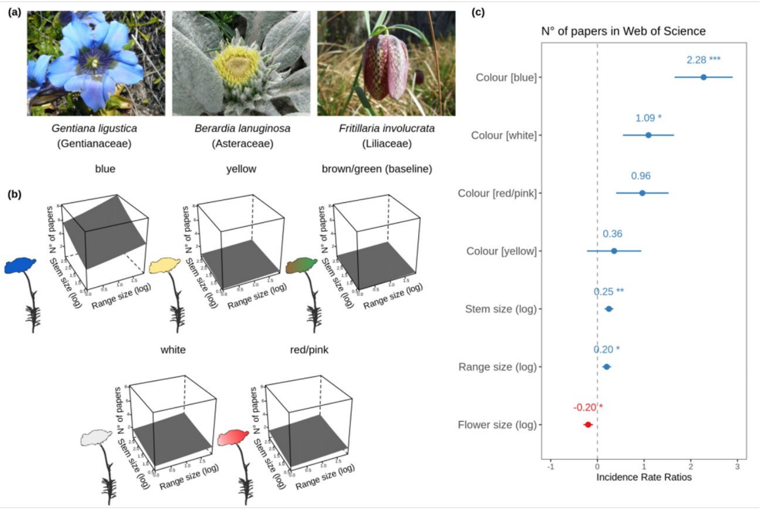
Base graphics in R

plots() is a canvas on which you can draw secondary elements, such as lines and legends

```
> plot(sort(y)~sort(x),
      col = "red",
      pch = 20,
      main = "provaprovaprova",
      ylab = "colname_3",
      xlab = "colname_2")
> abline(10,0, color = "blue")
```



benefits of scientific graphics in R



PROS

1. **Understanding**: steep learning curve

2. **Efficiency**: display different information in small space.

3. **Location**: it integrates mapping directly in graphs

4. **Cost**: R is free country to many other graphic tools

CONS

1. **Time**: especially first times could be time-consuming

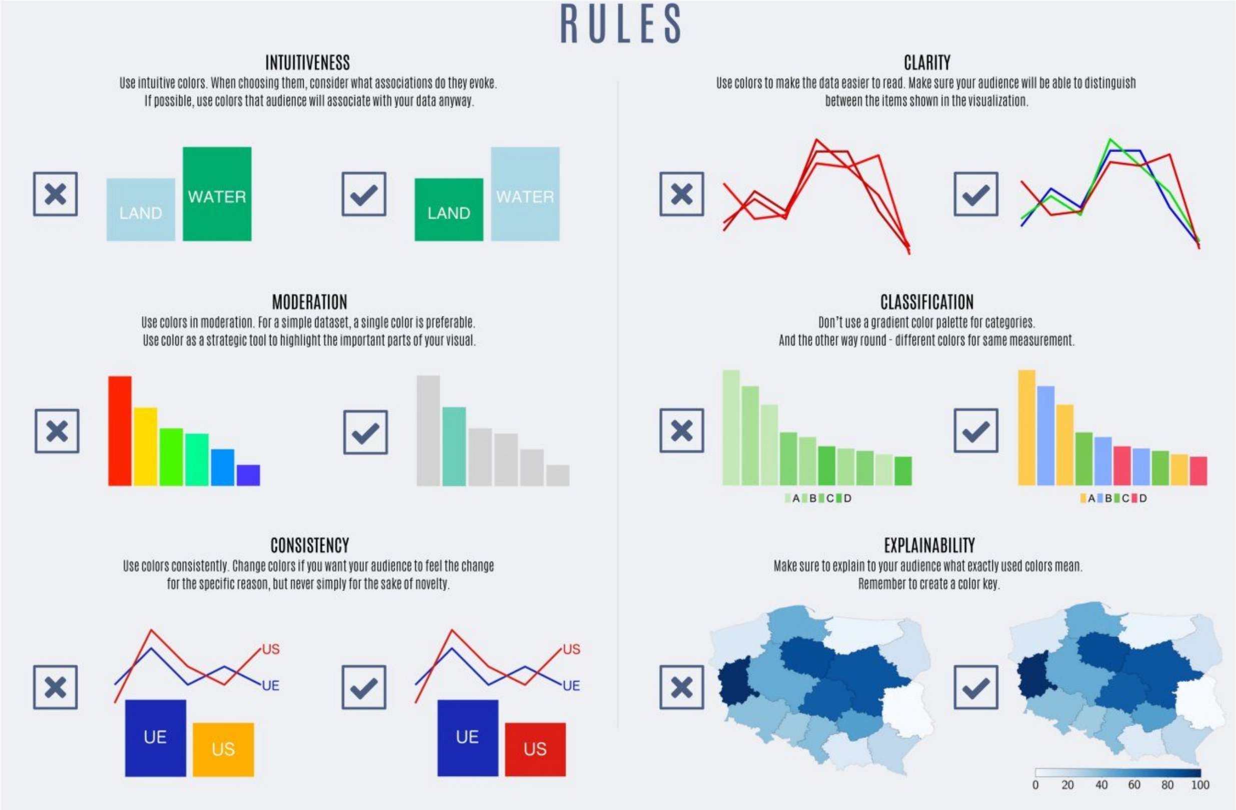
2. **Distraction**: you can build complex and fancy graphics-rich reports and charts, focusing more on the form than the function.

graphical notes for scientific data plotting

Easy/natural color associations

Use as few colors as you can

Use the same color for the same object through the whole report



RULES

INTUITIVENESS

Use intuitive colors. When choosing them, consider what associations do they evoke. If possible, use colors that audience will associate with your data anyway.

CLARITY

Use colors to make the data easier to read. Make sure your audience will be able to distinguish between the items shown in the visualization.

MODERATION

Use colors in moderation. For a simple dataset, a single color is preferable. Use color as a strategic tool to highlight the important parts of your visual.

CLASSIFICATION

Don't use a gradient color palette for categories. And the other way round - different colors for same measurement.

CONSISTENCY

Use colors consistently. Change colors if you want your audience to feel the change for the specific reason, but never simply for the sake of novelty.

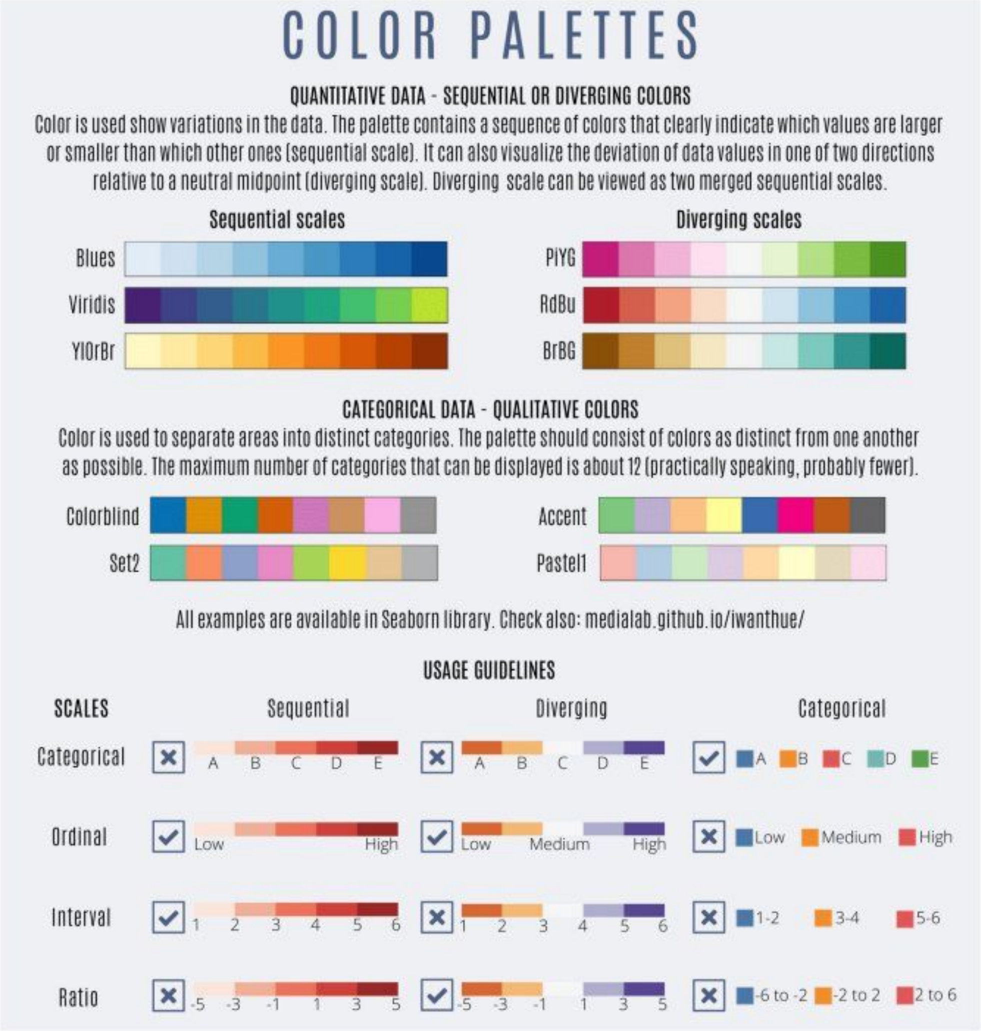
EXPLAINABILITY

Make sure to explain to your audience what exactly used colors mean. Remember to create a color key.

Each part should be easy readable

No gradients for categories

Legends are gold as well as measure units



COLOR PALETTES

QUANTITATIVE DATA - SEQUENTIAL OR DIVERGING COLORS

Color is used show variations in the data. The palette contains a sequence of colors that clearly indicate which values are larger or smaller than which other ones (sequential scale). It can also visualize the deviation of data values in one of two directions relative to a neutral midpoint (diverging scale). Diverging scale can be viewed as two merged sequential scales.

CATEGORICAL DATA - QUALITATIVE COLORS

Color is used to separate areas into distinct categories. The palette should consist of colors as distinct from one another as possible. The maximum number of categories that can be displayed is about 12 (practically speaking, probably fewer).

All examples are available in Seaborn library. Check also: medialab.github.io/iwanthue/

USAGE GUIDELINES

SCALES	Sequential	Diverging	Categorical
Categorical	☒ A B C D E	☒ A B C D E	☑ A B C D E
Ordinal	☑ Low High	☑ Low Medium High	☒ Low Medium High
Interval	☑ 1 2 3 4 5 6	☒ 1 2 3 4 5 6	☒ 1-2 3-4 5-6
Ratio	☒ -5 -3 -1 1 3 5	☑ -5 -3 -1 1 3 5	☒ -6 to -2 -2 to 2 2 to 6

Colors are useful make your graphs readable, but they must be used in a "correct way"

Journals are increasingly asking for **color-blind** readable figures because
~8% of the global population is affected by colorblindness (mainly males)

there are many packages to create palettes on R:

the most famous = RColorBrewer

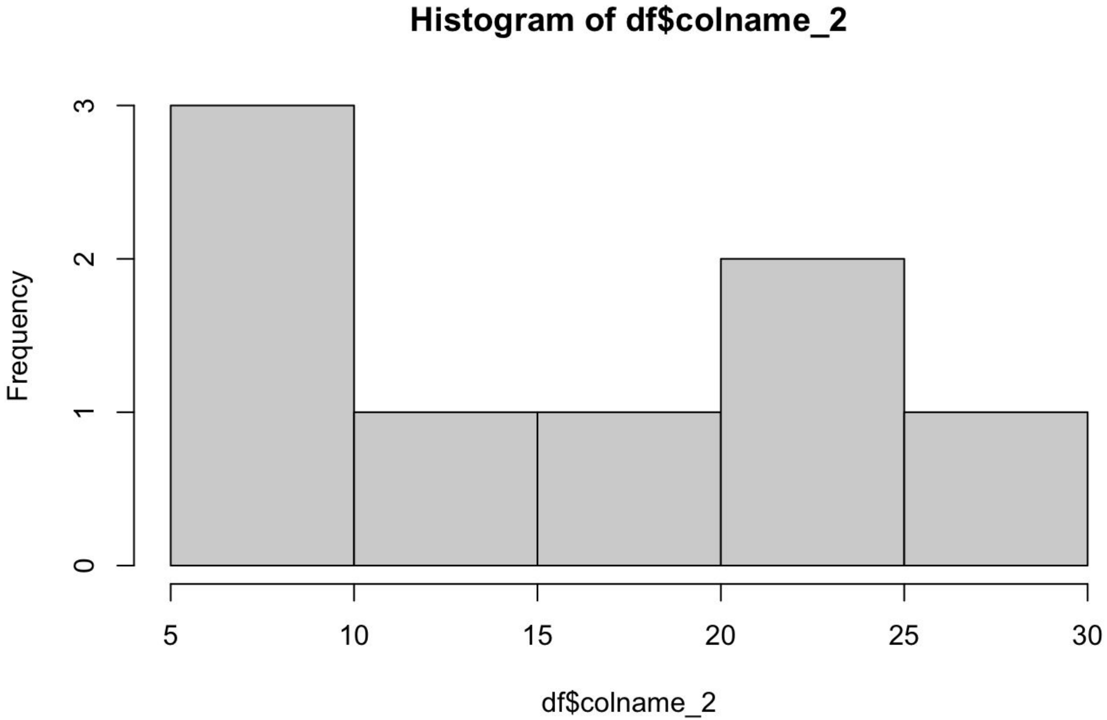
the most complete = paletteer

Additionally to colors there are shadings, line styles, point styles

main base graph functions - histograms

hist() is useful for visualize frequencies

```
> hist(df$colname_2)
```



main base graph functions - boxplots

```
> str(df)
'data.frame':  8 obs. of  3 variables:
 $ colname_1: chr  "A" "A" "B" "B" ...
 $ colname_2: int  13 22 18 23 10 29 8 7
 $ colname_3: num  8 8.5 9 9.5 10 10.5 11 11.5
```

Boxplots are useful to see a variable response to a specific factor … than you need to verify that you actually have a factor!

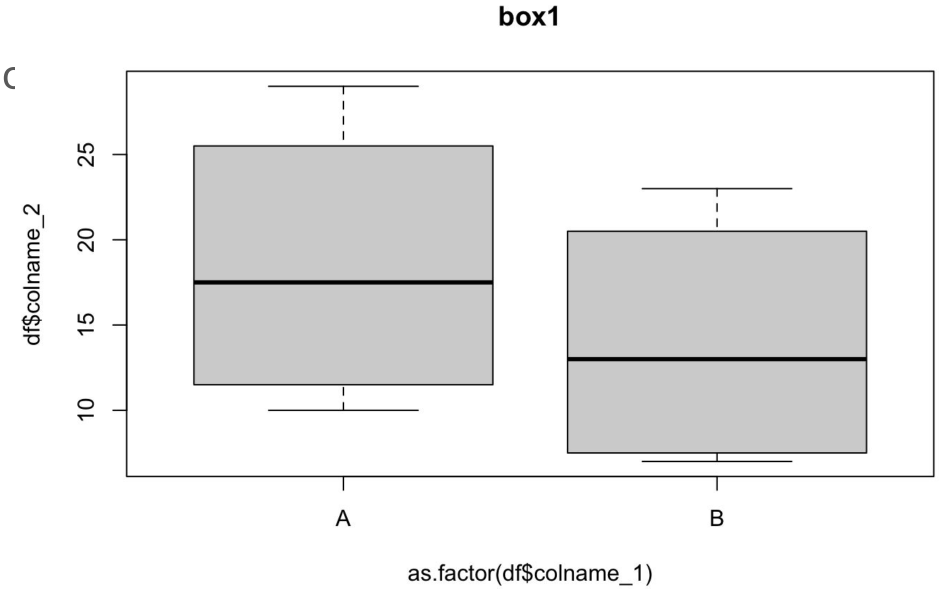
main base graph functions - boxplots

```
> boxplot(df$colname_2 ~ as.factor(df$colname_1), main = "box1")
```

Factors must be in the second argument

you can see multiple plots using the functic

```
> par(mfrow=c(plots x row,plots x col))
```



Exercise:

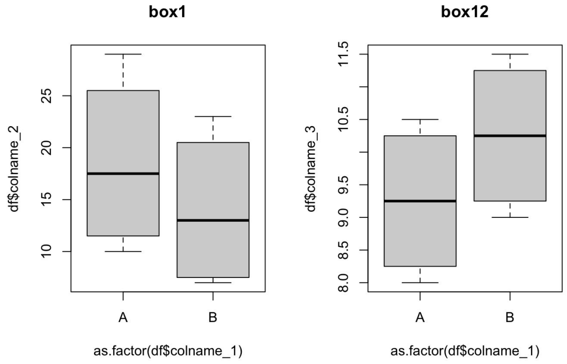
Visualize the two possible boxplot from df

in a single image.

main base graph functions - boxplots

Solution:

```
> par(mfrow=c(1,2))

> boxplot(df$colname_2 ~ as.factor(df$colname_1), main =
"box1")

> boxplot(df$colname_3 ~ as.factor(df$colname_1), main =
"box2")
```



Saving figures

Find your-own way, but remember that:

- export tool from Rstudio dependents on the resolution of your screen.
- figure sizes will depend from the plot window size (by default in Rstudio)
- you can avoid this steps saving images by using the command line (specific functions)
- journals want high resolutions figures (usually 300 dpi or higher), exporting *.pdf figures you save vectorial figures corresponding to infinite dpi!
- post-edit figure as few as you can
- post-edit figures with appropriate softwares (NO POWERPOINT!)

More hints in Stream 2 lessons!

PhD Toolbox - Get ready for Stream 2!

- Working with **lists**
- More advanced stuff on graphics (**ggplot2**)
- composite graphs panels (**gridExtra**, …)
- Exporting figures
- **Plotting Maps using R**

Aula 1

Friday January 26	**h 9-13**	**(sede di Viale Mattioli 25 - Botanical Garden)**
Monday January 29	**h 9-13**	**(sede di Viale Mattioli 25 - Botanical Garden)**
Tuesday January 30	**h 9-13**	**(sede di Viale Mattioli 25 - Botanical Garden)**