TDS Facility

Introduction to R for Life Sciences

João Lourenço, Tania Wyss & Nadine Fournier

Translational Data Science – Facility

SIB Swiss Institute of Bioinformatics

With slides from Diana Marek, Thomas Junier, Wandrille Duchemin, Leonore Wigger

From: First steps with R in Life Sciences

The Translational Data Science Facility

- Part of the **SIB Swiss Institute of Bioinformatics**
- Located at the AGORA Cancer Research Center in **Lausanne**
- Provides **statistics, bioinformatics and computational expertise** to molecular biology and applied research labs.
- Participates in fundamental and translational research by providing expertise in **data analysis** of single-cell and bulk multi-omics, spatial transcriptomics, flow cytometry, etc

For core facility service inquiry: nadine.fournier@sib.swiss
https://agora-cancer.ch/scientific-platforms/translational-data-science-facility/
https://www.sib.swiss/raphael-gottardo-group

Tell us about yourself !

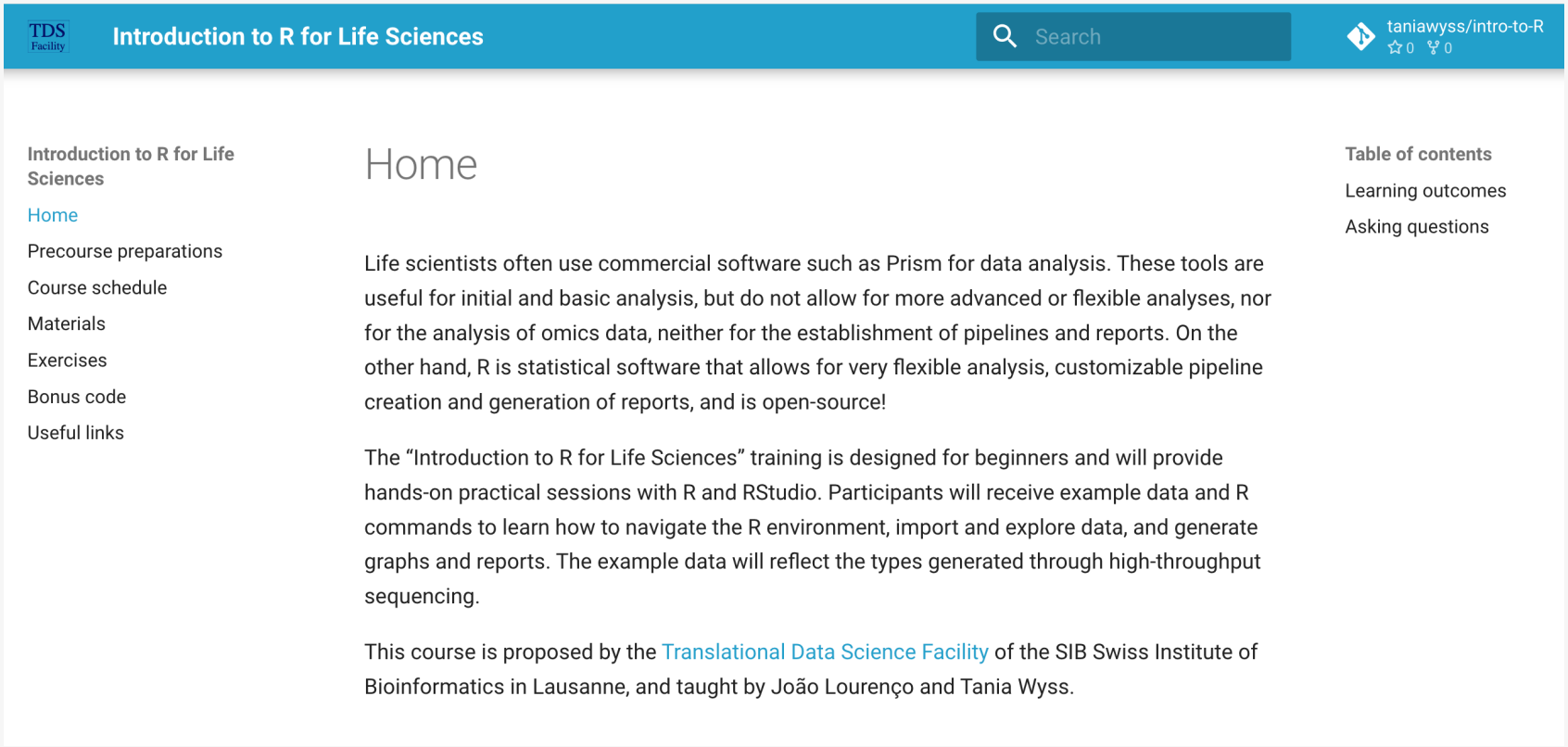
Share about yourself and your research, experience with programming, etc

Photo by National Cancer Institute, Unsplash

Photo by Scott Graham, Unsplash

Course material

1. Website
https://taniawyss.github.io/intro-to-R/



2. Ask us questions!

Outline & Schedule

Morning

01 **Introduction to R and the RStudio environment, working with scripts files**
Exercises
(9:00 – 10:30)
10:30 – 10:50 Coffee break

02 **Syntax, data types and structures, importing data**
Exercises
(10:50 -12:00)
12:00 – 13:00 Lunch break

Outline & Schedule

Afternoon

03 **Graphics**
Exercises
(13:00 – 15:30)

04
15:30 -15:50 Coffee break

04 **Statistics**
Exercises
(15:50 – 16:50)

16:50 - 17:00 Feedback and end of day

Course Content

R is vast and can't be learned in one day. The scope of this course is to:

- Give you a basic understanding of concepts behind R
- Allow you to import and manipulate data in R
- Show you how to create your first plots

This course is only the first step in your  journey!

01 Introduction to R and the RStudio environment

What is R ?

- R is a programming language and an environment for statistical computation and graphics.

  - A simple development environment with a console and a text editor

  - Facilities for data import, manipulation and storage
  - Functions for calculations on vectors and matrices
  - Large collections of data analysis tools
  - Graphical tools

https://www.r-project.org/

R's user community

- Group of core developers who maintain and upgrade the basic R installation. New version every 6 months.

- Anyone can contribute with add-on packages which provide additional functionality (thousands of such packages available) and help for each function.

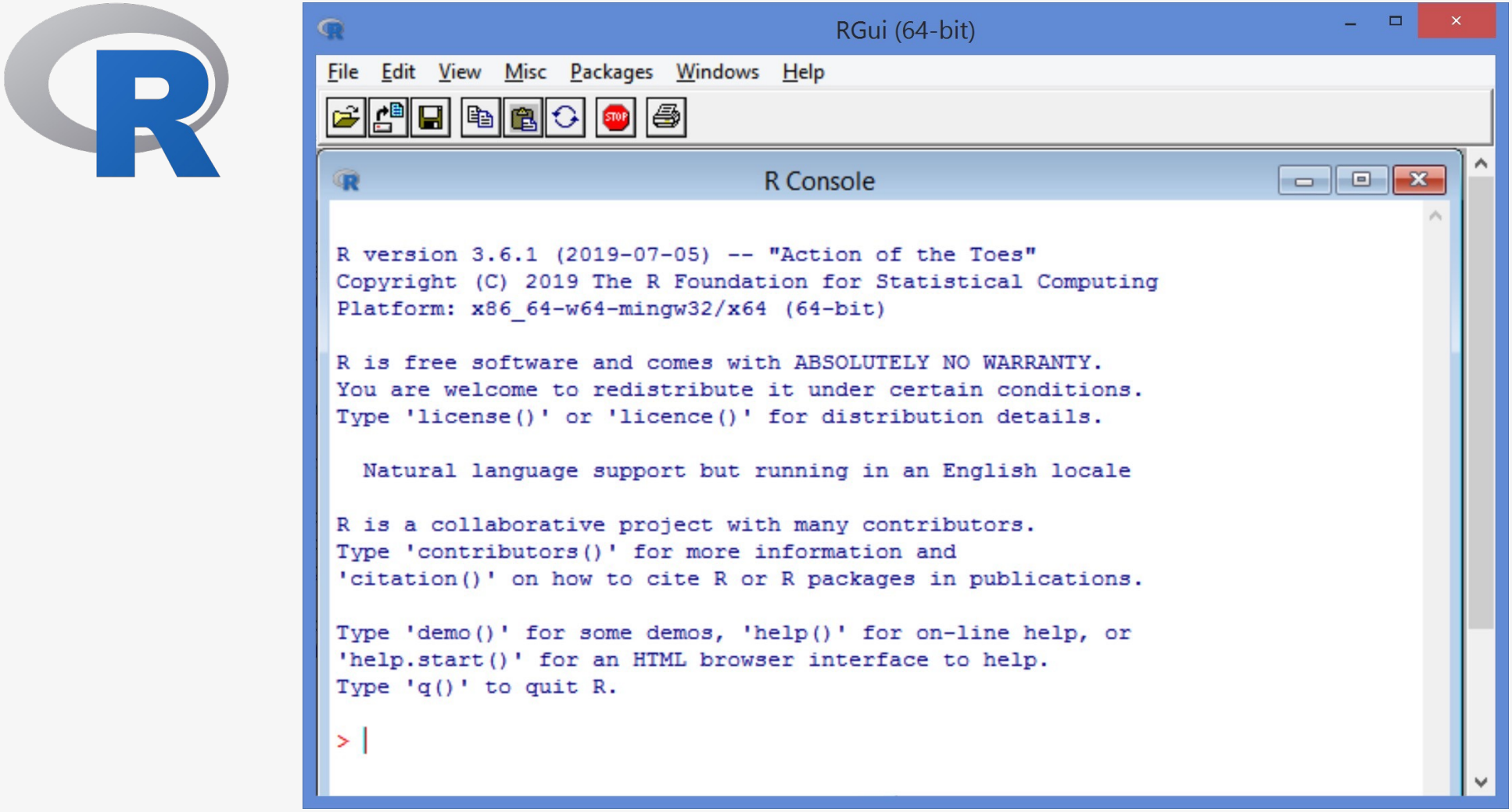
- Online help

  - in user group forums, *eg*:
  https://stat.ethz.ch/mailman/listinfo/r-help
  http://stackoverflow.com/questions/tagged/r

  - in countless online tutorials, books, blogs

RGui (**R** **G**raphical **u**ser **i**nterface)

- Together with the programming language, a (minimal) graphical user interface is installed.



R Combined with RStudio

https://posit.co/products/open-source/rstudio/

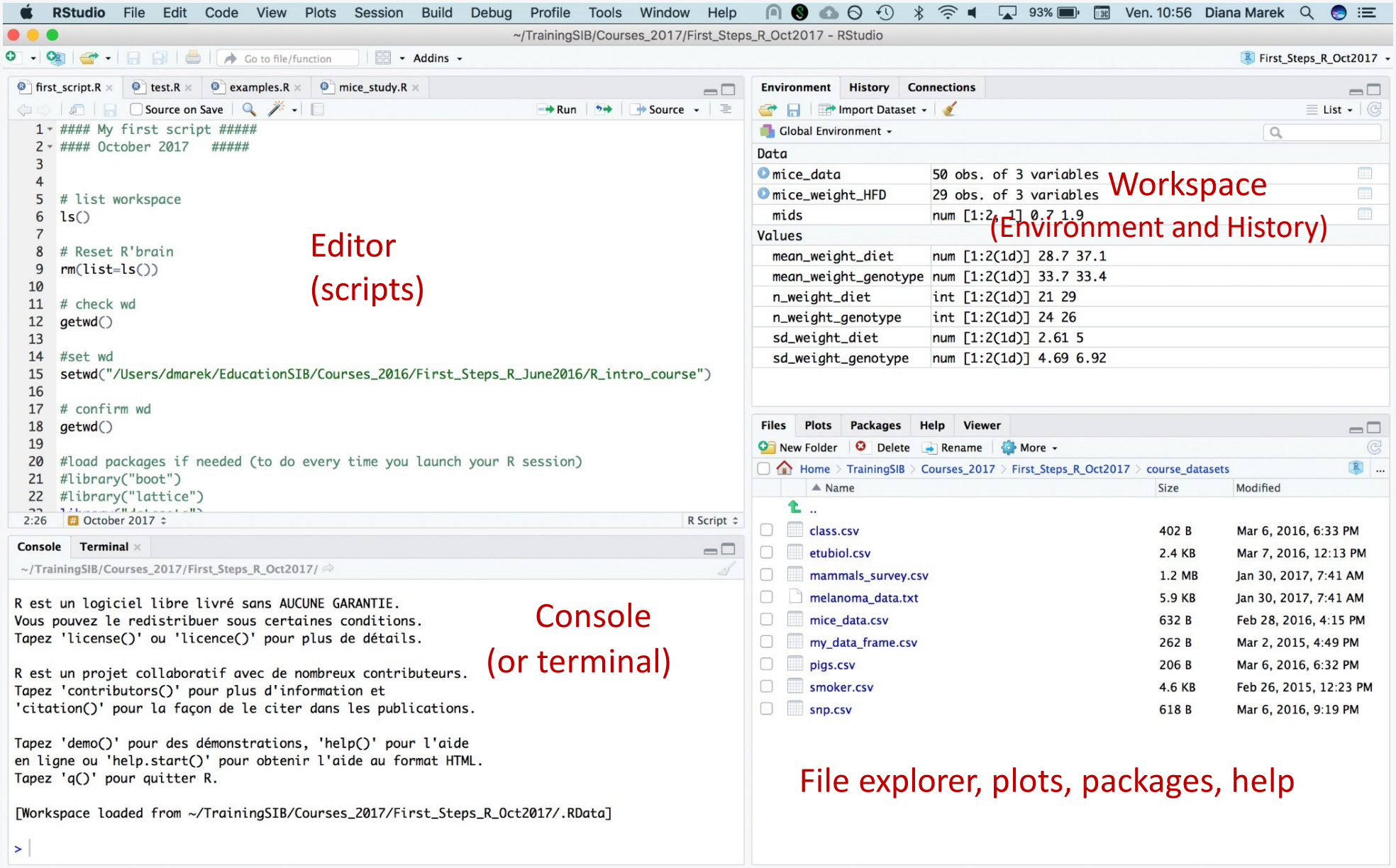
RStudio is an integrated development environment (IDE), designed to help you be more productive with R

It includes:

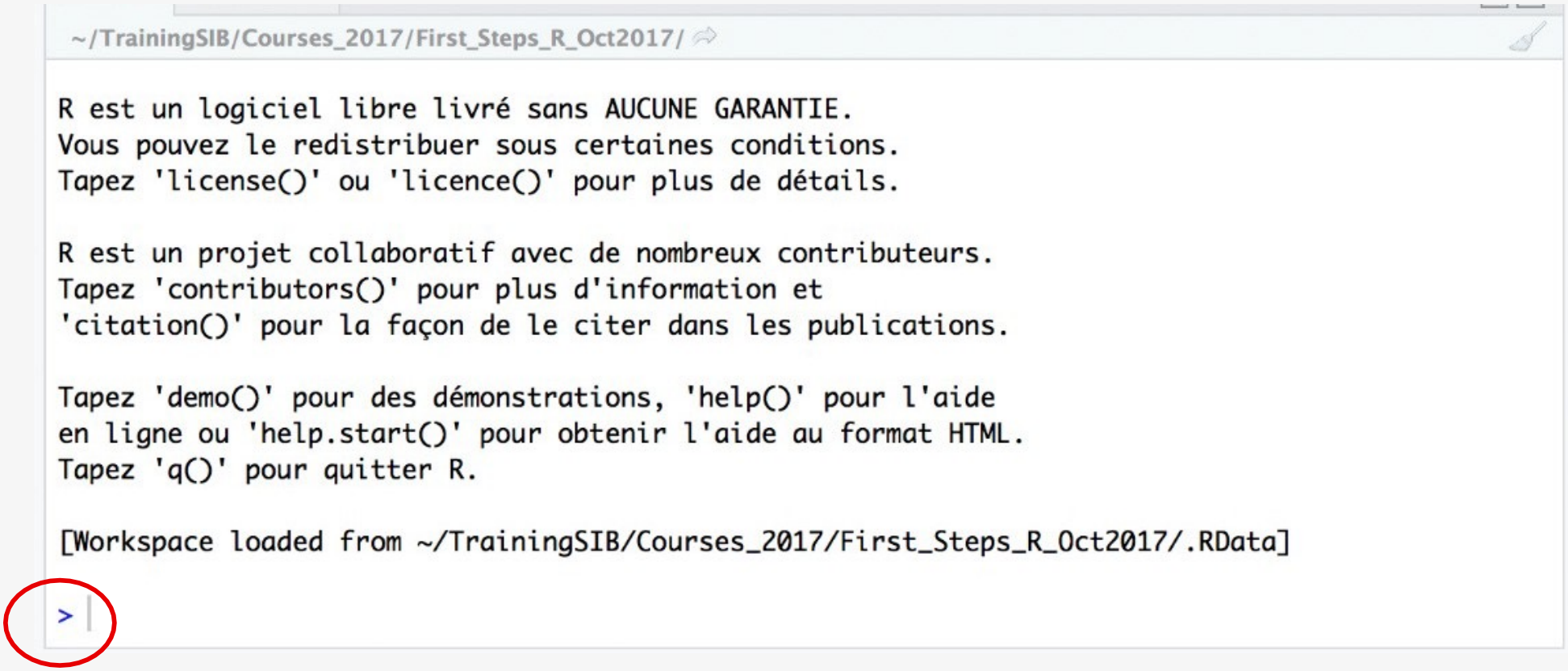
- A console
- A syntax-highlighting editor that supports direct code execution
- Tools for viewing the workspace and the history
- A file explorer, a package explorer, plot and help display areas

We suggest RStudio as a more powerful, more comfortable alternative to the RGUI

RStudio interface



Console: The Command Line



The prompt ">" indicates that R is waiting for you to type a command

Try it out...

Type the following at the command prompt:

Simple calculations

```
> 1 + 1
```

Assign values to a variable names

```
> x   <- 128.5
```

Display content of variables

```
> x
```

Pre-defined functions

```
> abs(-11)
```

The (not always helpful) help pages:

```
>  ?p.adjust
```

After each command, hit the return key.

This causes R to execute it.

Note the assignment operator <- with which we can keep values in the memory, by assigning a value and a name to a variable and store it in the session's memory.
We can use either <- or = to assign values to an object

Stick to one for consistency.

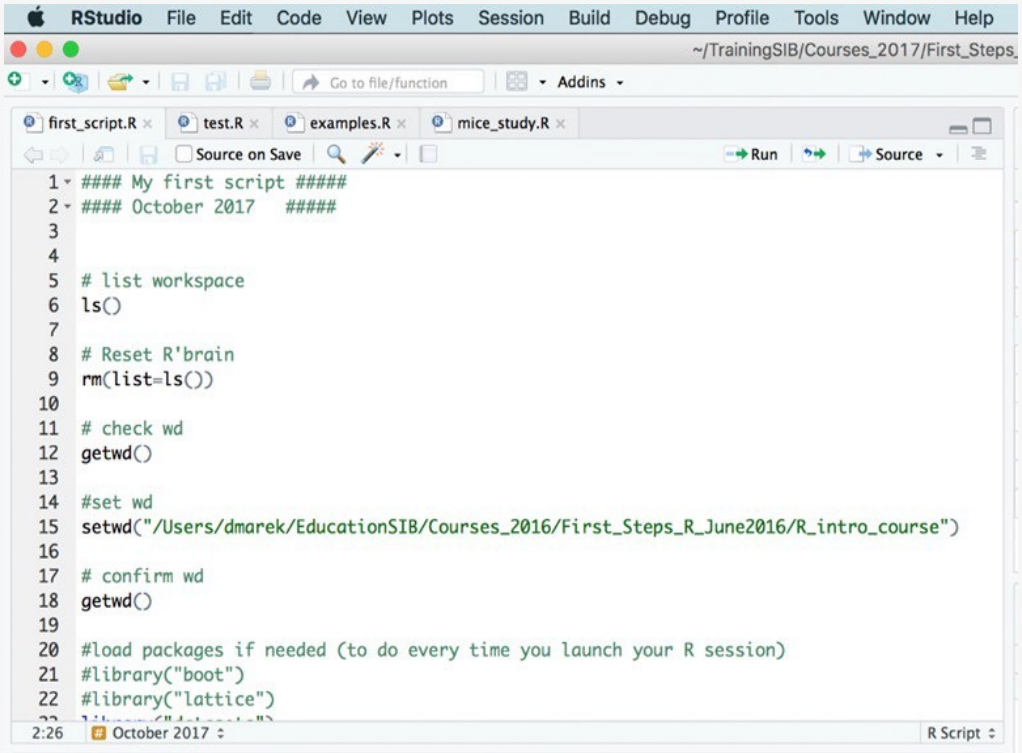
Working with script files

Editor: Write code to a script file

A script is a file that contains commands to be executed in succession.

Write your code into a script and save it

- to have documentation later of what you did
- to be able to re-use the code and create variations
- for easy execution



Notice the syntax highlighting

Create a new script and type code

- Create a new script using File > New File > R script. **Don't forget to save your script often.**
- By default, scripts are saved to the working directory.
- Files can be saved to other locations (File -> Save As…)
- Start **Typing code** at the top of the script

```
# My first command:
2 + 3
```

- **Notice the syntax highlighting**
- **Comments** : "#" at the beginning of a line or before a command: helping text ; everything that follows is ignored by the during executing ; R does not support multi-line comments

Send Code From a Script to the Console

Run individual lines, one by one:

- In RStudio: put the cursor anywhere in a line, hit

Ctrl + enter (Windows)

Cmd + return (Mac)

or click the "Run" button

Tip: Run part of a line or multiple lines: Highlight the code, then proceed as above

Save, close and open scripts

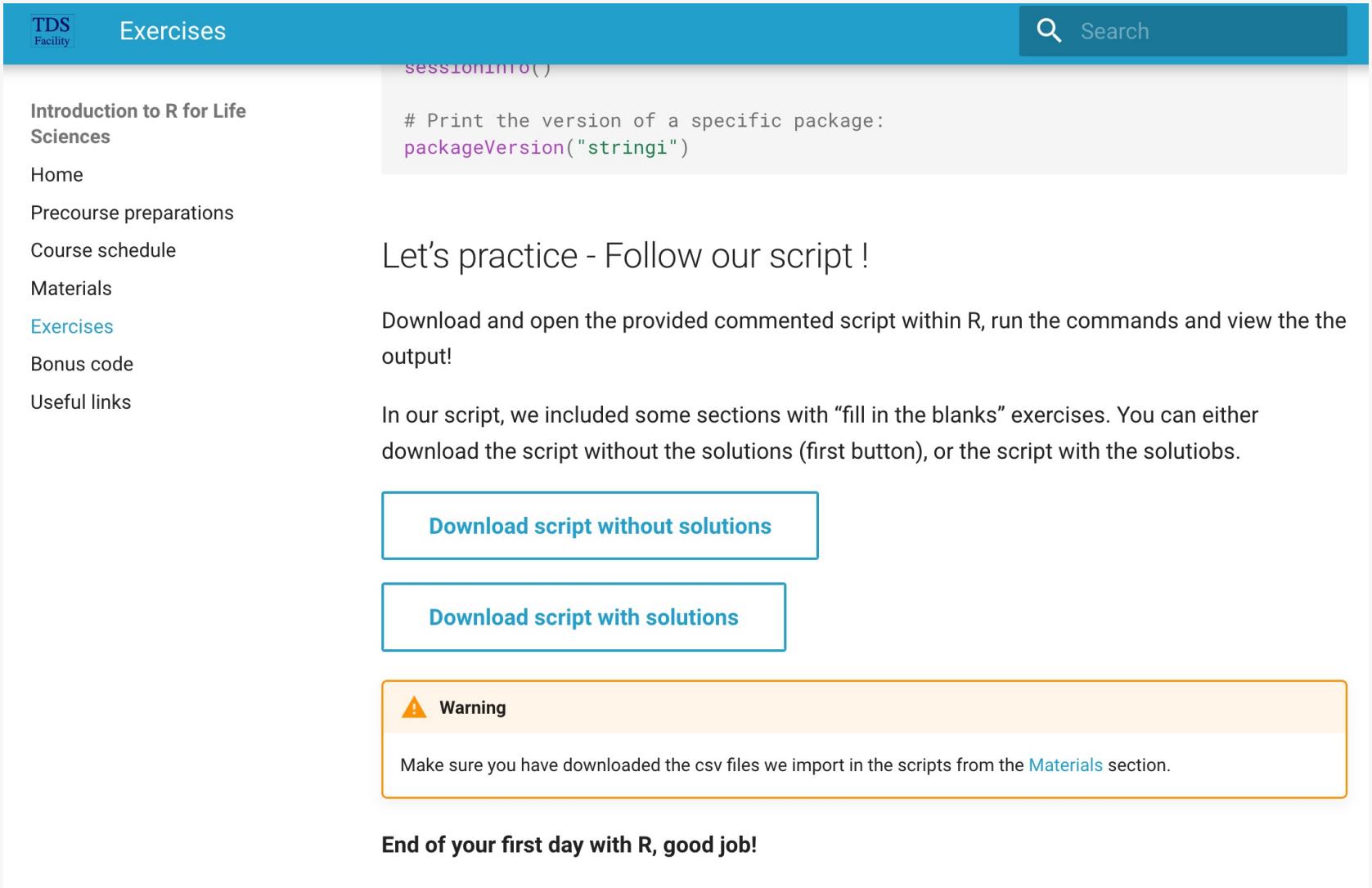
- **Save a script:** File > Save or 
- **Close and open a script:** File > Close and File > Open File

Tips:

- Most of your code should be developed and saved in scripts.
  - You can execute individual lines of code interactively while you are writing it.
  - You can run the entire script once it is ready and debugged.

Let's work with the provided script!

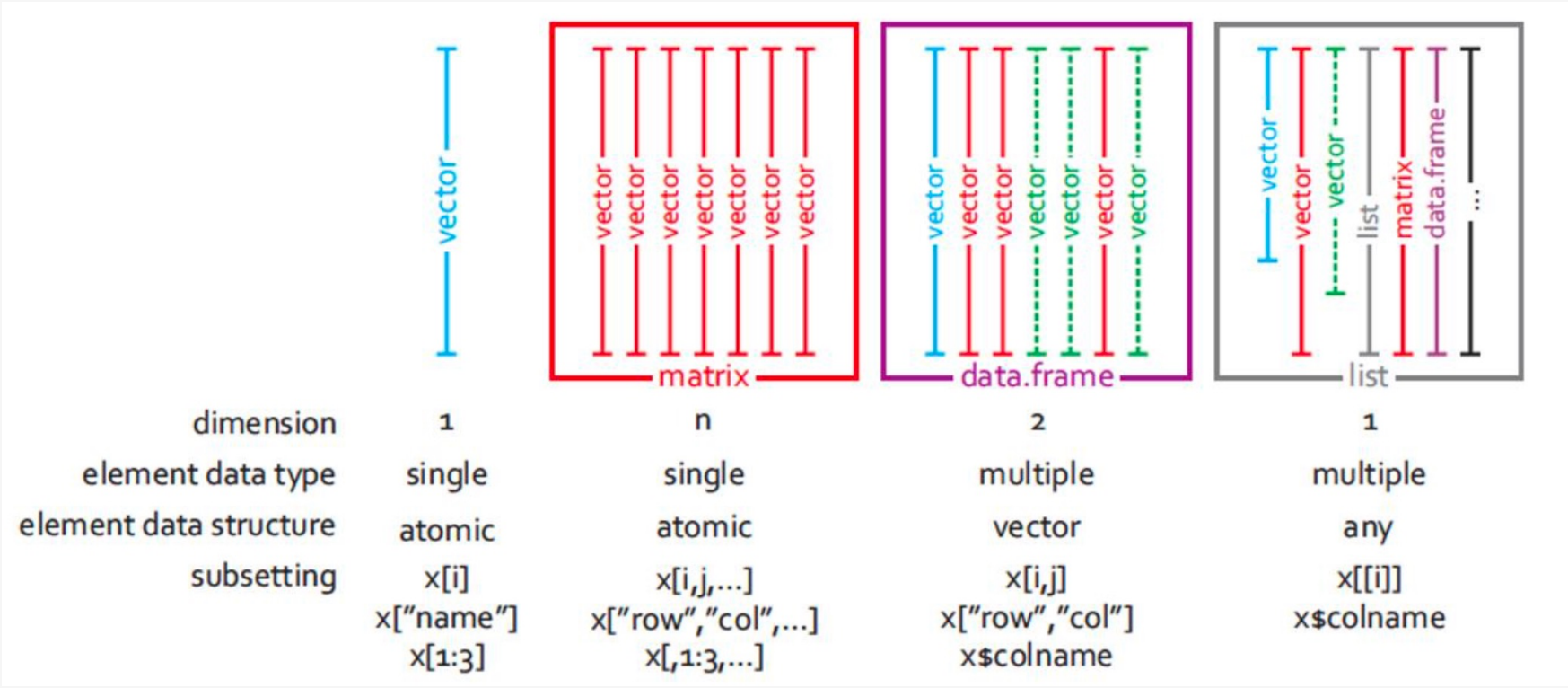
Download it from the bottom of the Exercise page, and open it in R



02

Syntax, data types and structures, importing data

Common object classes



	vector	matrix	data.frame	list
dimension	1	n	2	1
element data type	single	single	multiple	multiple
element data structure	atomic	atomic	vector	any
subsetting	x[i] x["name"] x[1:3]	x[i,j,...] x["row","col",...] x[,1:3,...]	x[i,j] x["row","col"] x$colname	x[[i]] x$colname

Example of a well-formated dataset

	A	B	C	D
1	Sample_ID	Age	Sex	Disease
2	M417	71	male	Healthy
3	M244	73	female	Tumor
4	M255	60	male	Healthy
5	M229	75	male	Tumor
6	M420	68	female	Healthy
7	M368	73	male	Healthy
8	M403	68	male	Tumor
9	M230	56	male	Tumor
10	M370	84	male	Tumor
11	M406	69	male	Tumor
12	M245	70	male	Tumor
13	M409	NA	female	Tumor
14	M395AR_dm	67	male	Tumor
15	PB	57	male	Healthy
16	M318	62	male	Healthy
17	M423	72	female	Tumor
18	M398_DMOS	61	female	Tumor
19	M233	74	male	Tumor
20	M381	57	male	Healthy
21	M408	65	male	Tumor
22	M402	68	male	Healthy

Example of a spreadsheet in Excel

- A header line with variable names (4 variables, 1 in each column)
- No blank spaces in variable names (use _ instead)
- Variable names do not contain symbols other than _
- One observation per row
- No comments or other content around the data table
- Indicate missing values with NA

Graphics

R graphics

R is powerful for plotting graphs and figures. Several plotting systems, including:

- base (i.e. graphics package, widely used, comes with basic R installation)
- ggplot2 (widely used in omics data analysis and others, implements the Grammar of Graphics, Wilkinson, Springer 2005)

They have very different syntaxes, cannot be mixed, and need to be learned separately.

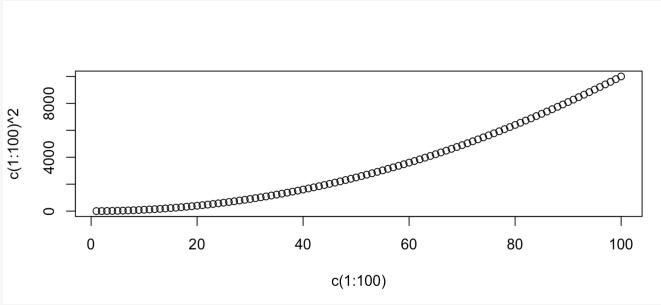
R base plotting system

Plots are built up step by step with multiple function calls.

High-level graphics functions:

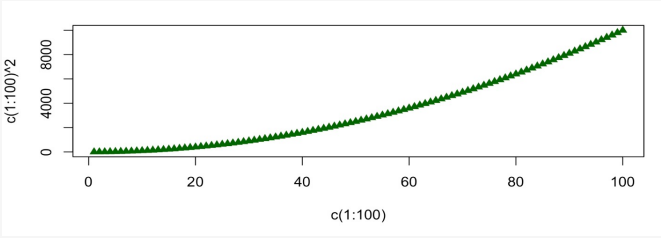
- Draw a new plot.

```
> plot(c(1:100), c(1:100)^2)
```



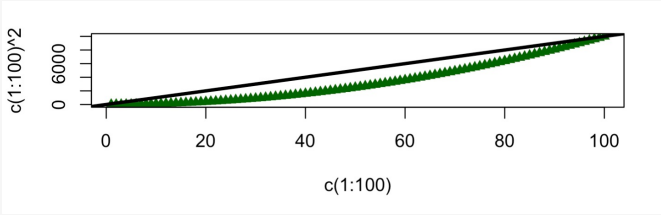
- Tailor its appearance with optional arguments.

```
> plot(c(1:100), c(1:100)^2,
       col="darkgreen", pch=17)
```



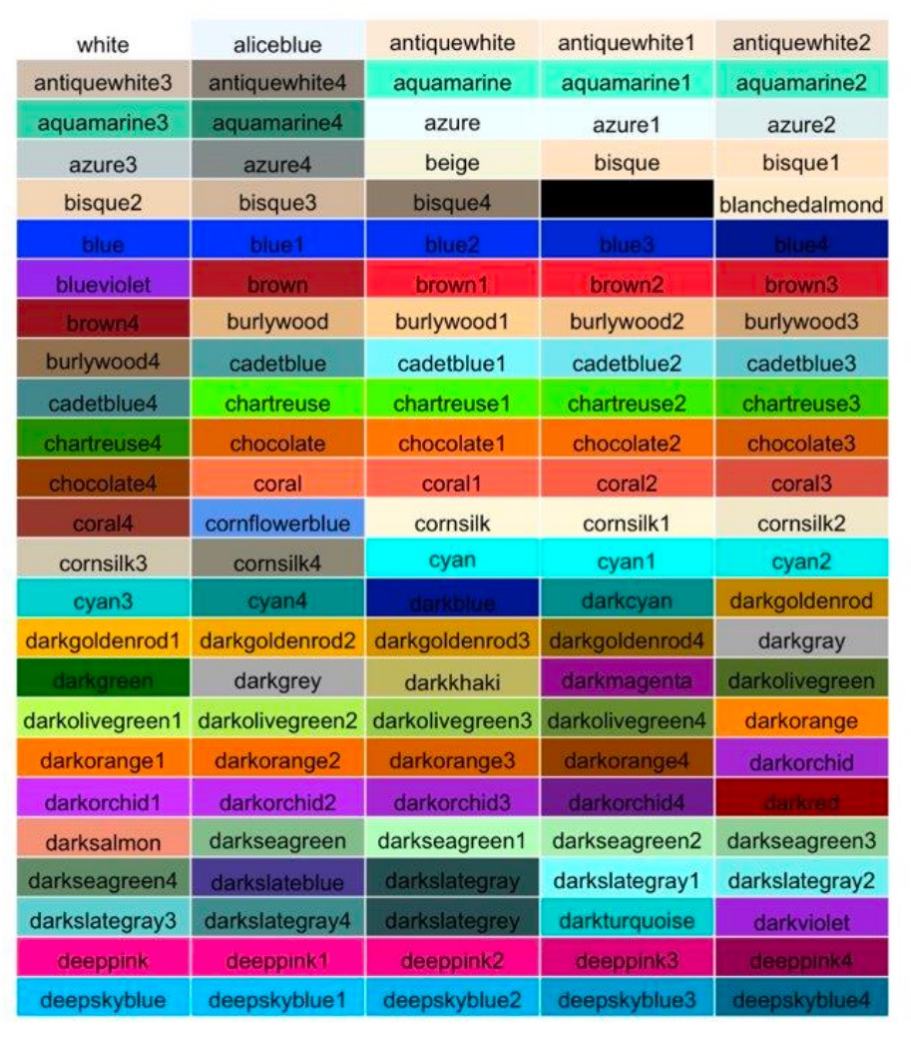
Low-level graphics functions: add graphical elements to an existing plot, piece by piece.

```
> plot(c(1:100), c(1:100)^2, col="darkgreen", pch=17)
> abline(a=0, b=100, lwd=3)
```



R colors (col)

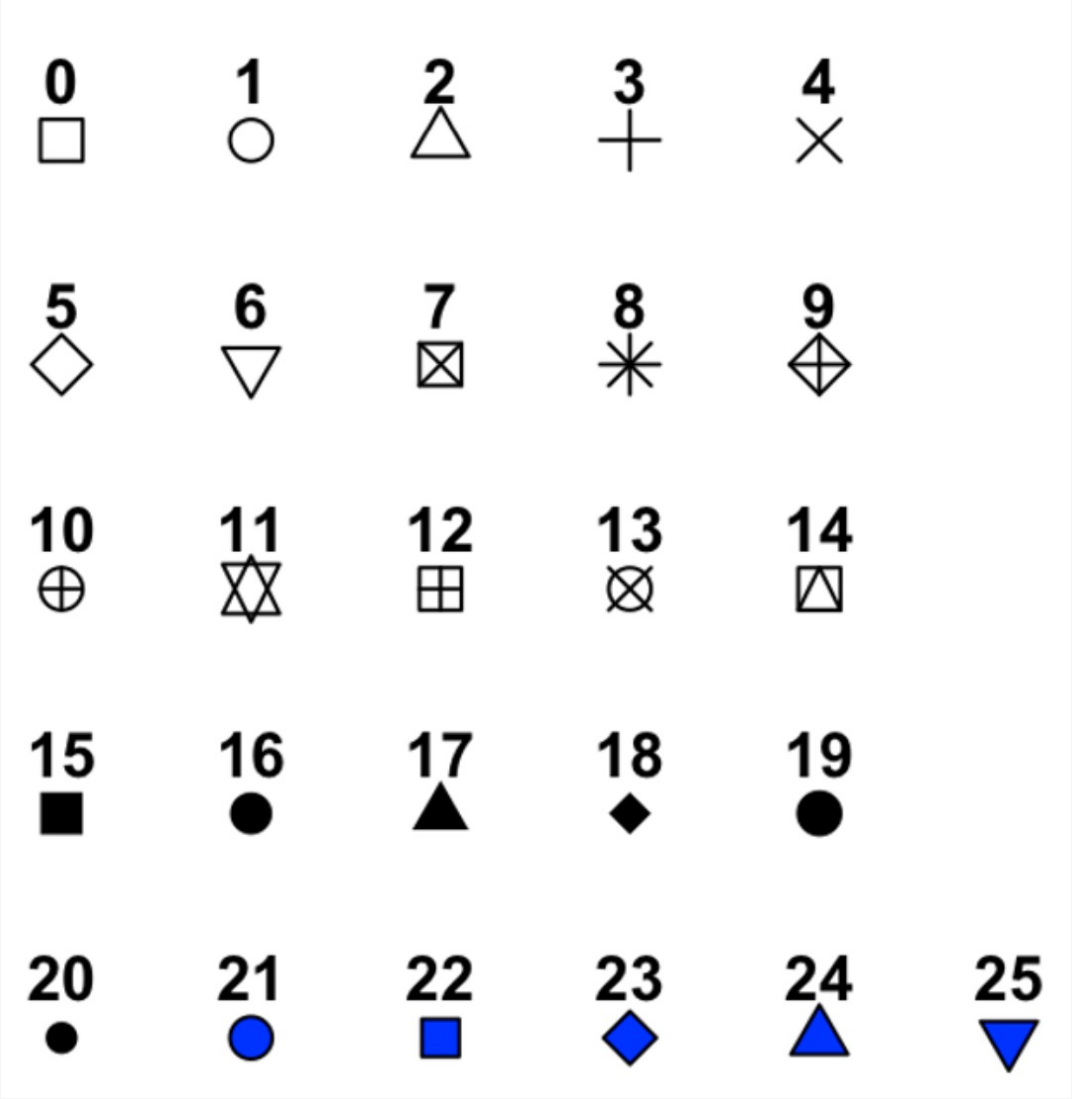
657 built-in color names
Here is a subset -->



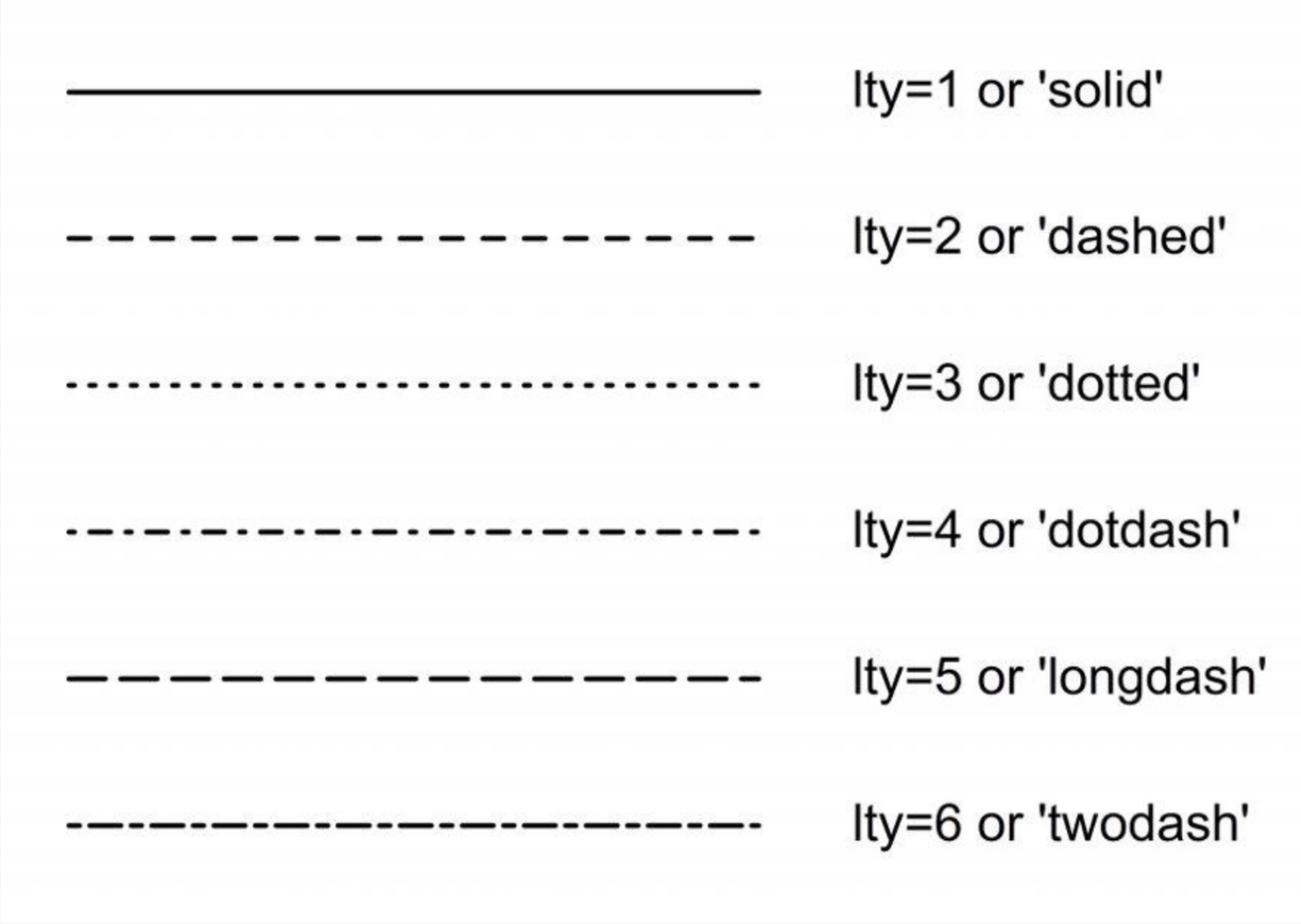
https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf

http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

R plotting characters (pch)



R line types (lty)

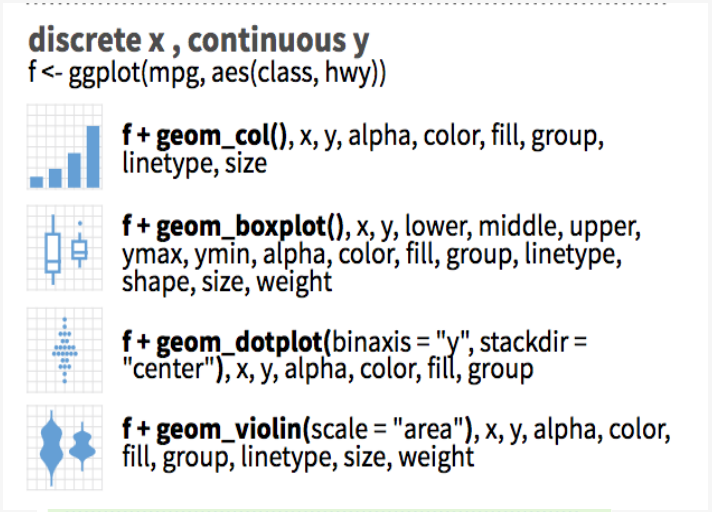


ggplot2

https://ggplot2.tidyverse.org/

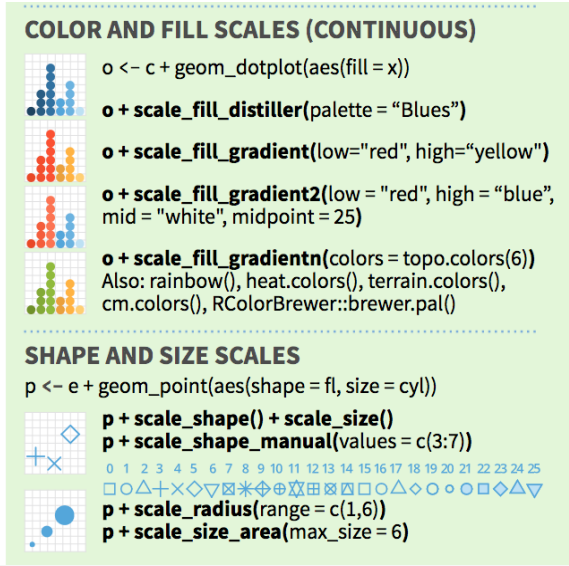
- The syntax (grammar) is very different from base R plotting functions.
- It builds a plot by adding layers of functions using the + sign
- The basic ggplot2 functions specify the data frame, the x,y coordinates, and the type of plot:

```
ggplot(dataframe, aes(x, y)) +
    geom_type()
```



Additional layers for full customizations are then added:

```
ggplot(dataframe, aes(x, y, color=factor)) +
    geom_type() +
    additional_layers()
```



ggplot2 cheatsheet https://rstudio.github.io/cheatsheets/data-visualization.pdf

ggplot2

Many other packages offer additional ggplot2 functionalities

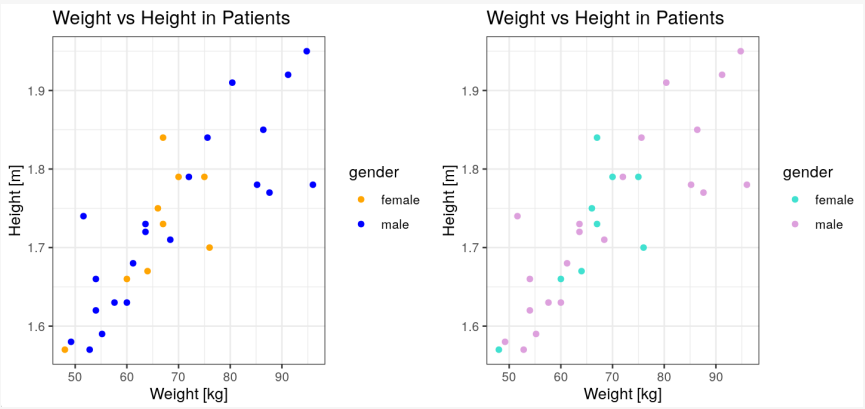
- Cowplot: Multi-panel figures; save the plot to an object, then display

```
p1 <- ggplot(dataset, aes(x, y, color=fact)) +
   geom_type() +
   additional_layers()

p2 <- ggplot(dataset, aes(x, y, color=fact)) +
   geom_type() +
   additional_layers()

# install.packages("cowplot")
library(cowplot)
plot_grid(p1, p2, nrow=1)
```

p1 p2



- ggpubr: publication-ready plot customization, e.g. add T-test result on plot
- ggrepel: to avoid overlapping data point labels, e.g. in volcano plots
- …

Basic Statistics in R

Statistical hypothesis testing

Two hypotheses in competition:

- H0: the NULL hypothesis (usually the most conservative – e.g., “no difference”)
- H1: the alternative hypothesis (usually the one we are actually interested in)

Example:

- H0: « There is no difference in the expression of a gene between two given subtypes of melanoma»
- H1: « The average expression of a gene is different in two given subtypes of melanoma»

Statistical test:

- Calculate test statistic
- Calculate associated p-value
- Check if p-value is small enough to reject H0, according to pre-defined significance level

t-test

Goal:

- Compare a continuous measure between two groups
- Is the difference between the two group means statistically significant?

Assumptions:

- Observations are independent
- The two groups follow a normal distribution
- Homogeneity of variances (R uses Welch's t-test, which does not assume equal variance)

Correlation

Measures the strength and direction of the relationship between two variables.

Correlation Coefficient (r):

- Ranges from -1 to +1
- Direction
  - Positive correlation ($r>0$): As one variable increases, the other tends to increase
  - Negative correlation ($r < 0$): As one variable increases, the other tends to decrease
- Strength
  - Perfect correlation: $|r| = 1$
  - Strong correlation: $0.7 < |r| < 1$
  - Moderate correlation: $0.3 < |r| \leq 0.7$
  - Weak correlation: $0 < |r| \leq 0.3$
  - No correlation (no consistent relationship between variables): $r = 0$

REMEMBER: Correlation does not imply causation !

Linear regression

Statistical method used to model the relationship between a dependent variable (Y) and one (or more) independent variable(s) (X).

Line of Best Fit: $Y = \beta_0 + \beta_1 X$

- β_0: Y-intercept (value of Y when X = 0)
- β_1: Slope (change in Y for a one-unit increase in X)

Least Squares Method: Minimizes the sum of squared residuals

R-squared (R^2):

- Measures how well the model fits the data
- Ranges from 0 to 1
- Higher values indicate better fit

Additional learning and practicing

Wandrille Duchemin's First Steps with R in Life Sciences (2 days):
It includes more on statistics!
https://github.com/sib-swiss/first-steps-with-R-training/tree/master

Introduction to statistics with R (3 days), for R beginners also:
https://sib-swiss.github.io/Introduction-to-statistics-with-R/day1/

Introduction to R for Cancer Scientists
https://bioinformatics-core-shared-training.github.io/r-intro/index.html

Glittr.org

How to get data for practicing and playing

R contains many practice data sets (data frames), great for trying out functions.

Display names of available data sets

```
> data(package = .packages(all.available = TRUE)) # lists
data set names available in all installed packages
```

Load and use a data set

```
>data(iris) # load the iris data (overwrite existing
>?iris      variable) #
>head(iris) # get information about the iris data
            # display top few lines of the iris data frame
  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
```

How to get data for practicing and playing

R can easily simulate data drawn from a given distribution. The function rnorm() generates normally distributed data.

Example:

```
>rnorm(10)   #numeric vector with 10 values  #drawn
             from normal distribution, #mean=0, sd=1
             (function defaults)
[1]  1.1053564  0.7937635  0.2743762  0.3574477 -0.7677099
[2]  0.5838973  0.6616164  0.1203090 -0.4060265  0.2778585
```

```
>rnorm(10, mean=10, sd=2) #customized mean and sd
 [1]  6.253392  9.527140  9.398857 11.932284 11.472909
 [2] 10.714245  7.656026 11.302829  9.332930 10.264157
```

If you want data from other distributions than normal:
rpois() for poisson, rbinom() for binomial (see R help)

sd: standard deviation