

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/>

The screenshot shows the homepage of the 'Introduction to R for Life Sciences' website. The header includes the TDS Facility logo, the page title, a search bar, and a navigation bar with links to Table of contents, Learning outcomes, and Asking questions. The main content area features a large heading 'Home', a brief introduction to R, and a detailed description of the course's purpose and content.

TDS Facility Introduction to R for Life Sciences Search taniawyss/intro-to-R
☆ 0 0

Introduction to R for Life Sciences

Home

Precourse preparations

Course schedule

Materials

Exercises

Bonus code

Useful links

Home

Life scientists often use commercial software such as Prism for data analysis. These tools are useful for initial and basic analysis, but do not allow for more advanced or flexible analyses, nor for the analysis of omics data, neither for the establishment of pipelines and reports. On the other hand, R is statistical software that allows for very flexible analysis, customizable pipeline creation and generation of reports, and is open-source!

The "Introduction to R for Life Sciences" training is designed for beginners and will provide hands-on practical sessions with R and RStudio. Participants will receive example data and R commands to learn how to navigate the R environment, import and explore data, and generate graphs and reports. The example data will reflect the types generated through high-throughput sequencing.

This course is proposed by the Translational Data Science Facility of the SIB Swiss Institute of Bioinformatics in Lausanne, and taught by João Lourenço and Tania Wyss.

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**

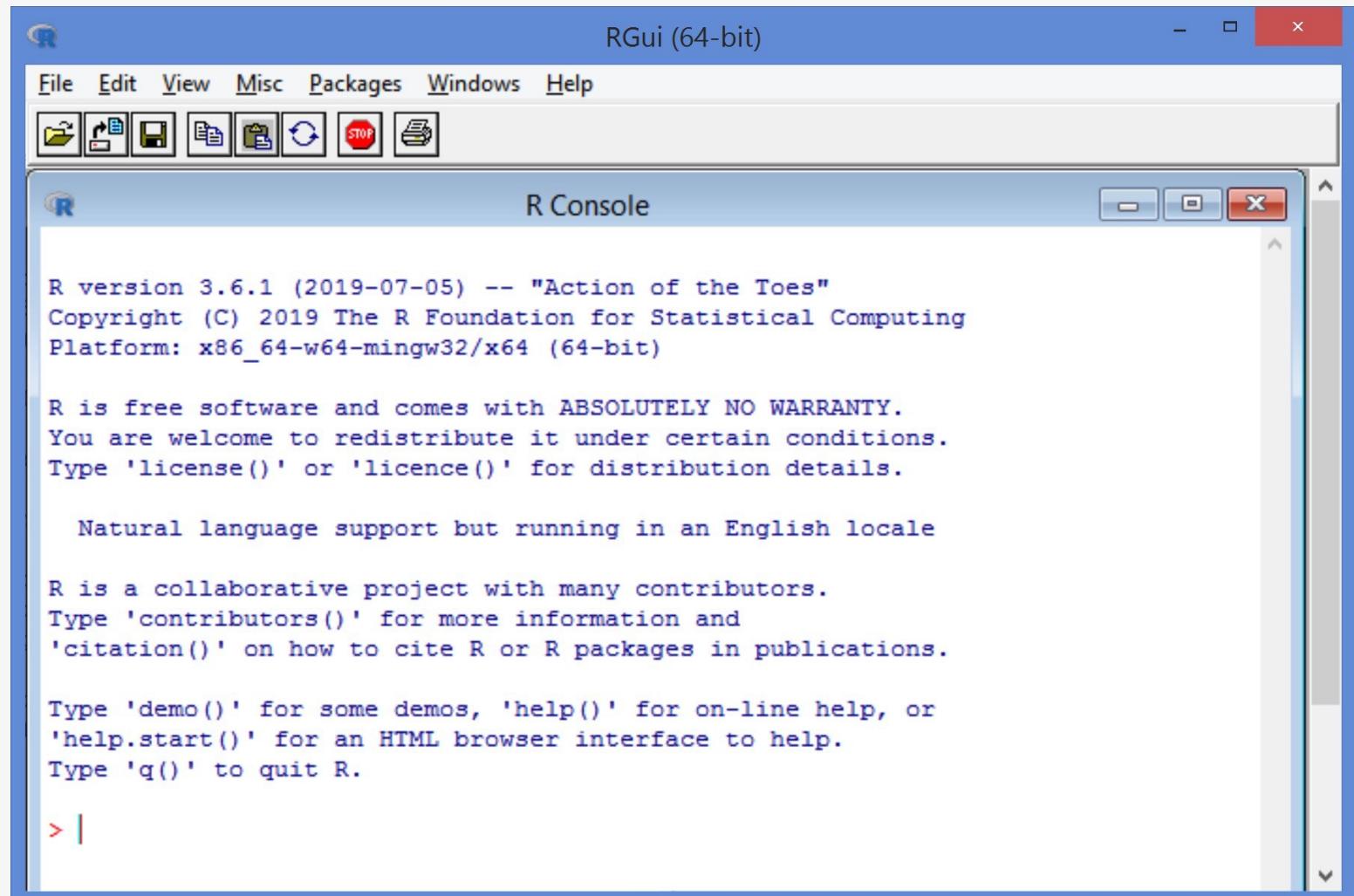
<https://www.r-project.org/>

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 Graphical user interface)

- 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 R GUI

RStudio interface

The screenshot shows the RStudio interface with several red annotations:

- Editor (scripts)**: Points to the top-left pane where an R script named "first_script.R" is open, containing code related to setting the workspace and loading packages.
- Console (or terminal)**: Points to the bottom-left pane displaying the R startup message and help text.
- File explorer, plots, packages, help**: Points to the bottom-right pane which includes a file browser, a plots section, and package management tools.
- Workspace (Environment and History)**: Points to the top-right pane showing the R environment, listing global variables like "mice_data" and "mice_weight_HFD".

Code in the Editor:

```
1 ## My first script #####
2 ##### October 2017 #####
3
4
5 # list workspace
6 ls()
7
8 # Reset R'brain
9 rm(list=ls())
10
11 # check wd
12 getwd()
13
14 #set wd
15 setwd("/Users/dmarek/EducationSIB/Courses_2016/First_Steps_R_June2016/R_intro_course")
16
17 # confirm wd
18 getwd()
19
20 #load packages if needed (to do every time you launch your R session)
21 #library("boot")
22 #library("lattice")
```

Console output:

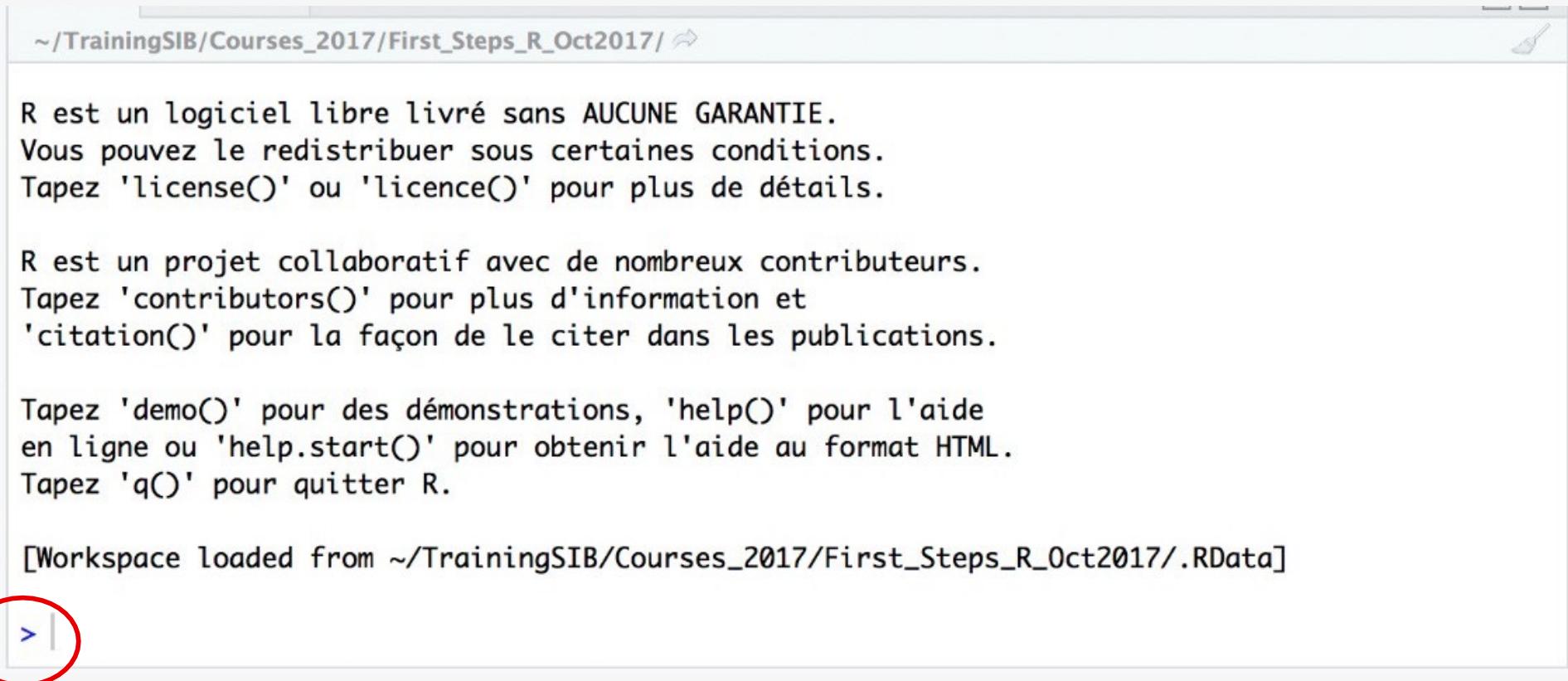
```
R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/TrainingSIB/Courses_2017/First_Steps_R_Oct2017/.RData]
```

Console: The Command Line



The screenshot shows an R console window with the following text:

```
~/TrainingSIB/Courses_2017/First_Steps_R_Oct2017/ ↗
```

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Workspace loaded from ~/TrainingSIB/Courses_2017/First_Steps_R_Oct2017/.RData]

> |

A red circle highlights the blue prompt character '>'. The cursor is positioned immediately after it.

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
```

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.

**After each command,
hit the return key.**



This causes R to execute it.

02

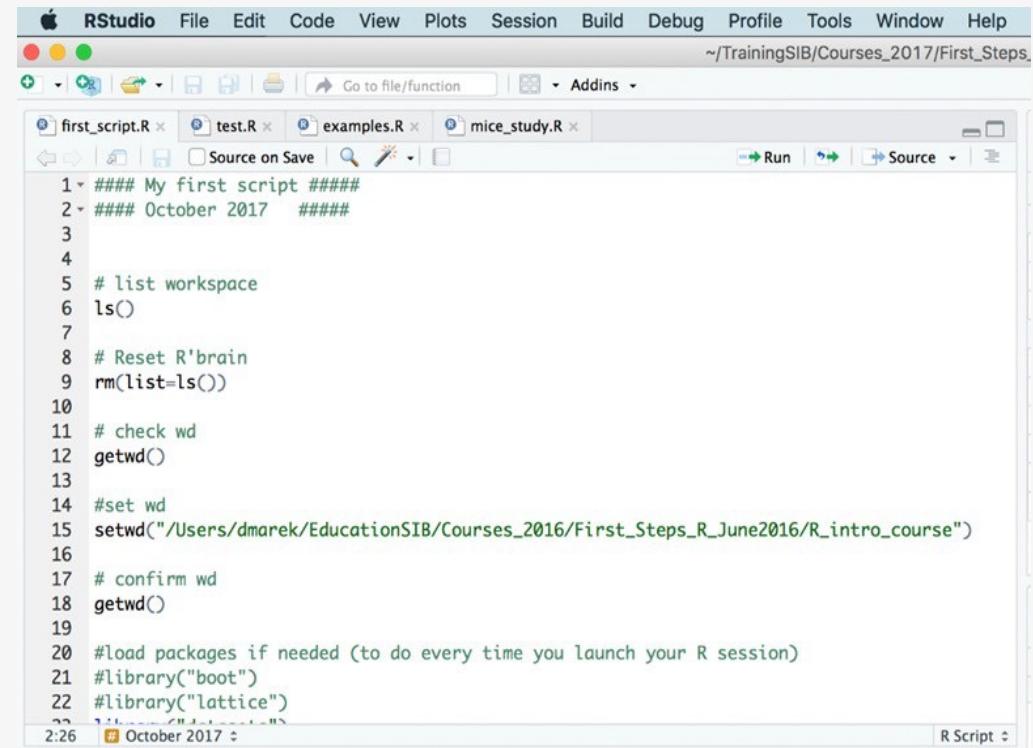
Working with script files

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



The screenshot shows the RStudio interface with the title bar "RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help". Below the title bar, the current working directory is shown as " ~/TrainingSIB/Courses_2017/First_Steps.". The main area is a code editor with four tabs: "first_script.R", "test.R", "examples.R", and "mice_study.R". The "first_script.R" tab is active, displaying the following R code:

```
1 ## My first script #####
2 #### October 2017 #####
3
4
5 # list workspace
6 ls()
7
8 # Reset R'brain
9 rm(list=ls())
10
11 # check wd
12 getwd()
13
14 #set wd
15 setwd("/Users/dmarek/EducationSIB/Courses_2016/First_Steps_R_June2016/R_intro_course")
16
17 # confirm wd
18 getwd()
19
20 #load packages if needed (to do every time you launch your R session)
21 library("boot")
22 library("lattice")
23
24 #### October 2017 #####
25
```

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

The screenshot shows a web page with a blue header bar. On the left, there's a sidebar with links: 'Introduction to R for Life Sciences', 'Home', 'Precourse preparations', 'Course schedule', 'Materials', 'Exercises' (which is the active tab), 'Bonus code', and 'Useful links'. The main content area has a search bar and some R code examples. A large text box contains the following R code:

```
sessionInfo()  
# Print the version of a specific package:  
packageVersion("stringi")
```

Below the code, there's a section titled 'Let's practice - Follow our script !' with instructions: 'Download and open the provided commented script within R, run the commands and view the output!'. It also notes that the script includes 'fill in the blanks' exercises. At the bottom, there are two buttons: 'Download script without solutions' and 'Download script with solutions'. A warning box at the very bottom says: '⚠ Warning: Make sure you have downloaded the csv files we import in the scripts from the Materials section.' The footer of the page says 'End of your first day with R, good job!'

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	$x[i]$ $x["name"]$ $x[1:3]$	$x[i,j,...]$ $x["row","col",...]$ $x[,1:3,...]$	$x[i,j]$ $x["row","col"]$ xcolname$	$x[[i]]$ xcolname$

The diagram illustrates the structure of four common R objects:

- vector:** A single vertical line labeled "vector".
- matrix:** A vertical column of seven horizontal lines, each labeled "vector", enclosed in a red rectangular border at the bottom labeled "matrix".
- data.frame:** A vertical column of eight horizontal lines. The first four are solid blue and labeled "vector"; the next three are dashed green and labeled "vector"; the last one is a dash-dot red and labeled "vector". This column is enclosed in a purple rectangular border at the bottom labeled "data.frame".
- list:** A vertical column of five horizontal lines. The first two are solid blue and labeled "vector"; the third is a dashed green and labeled "vector"; the fourth is a solid grey and labeled "list"; the fifth is a solid red and labeled "matrix"; the sixth is a solid purple and labeled "data.frame"; and the seventh is a solid black and labeled "...". This column is enclosed in a grey rectangular border at the bottom labeled "list".

Example of a well-formatted 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

- 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

Example of a spreadsheet in Excel

03

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.

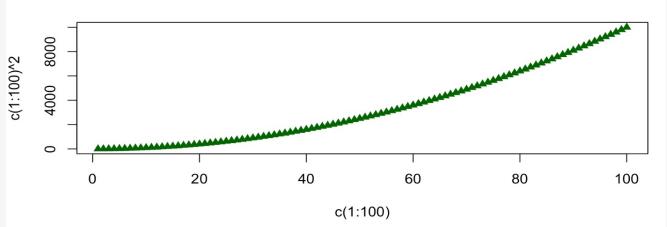
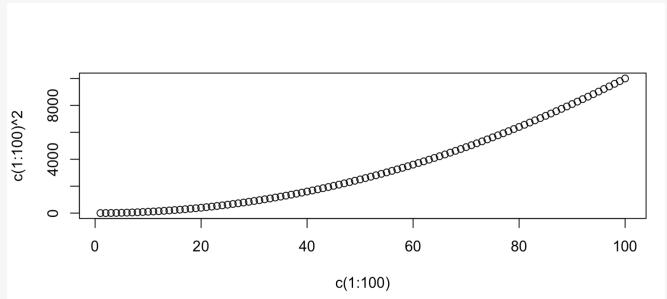
R base plotting system

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

High-level graphics functions:

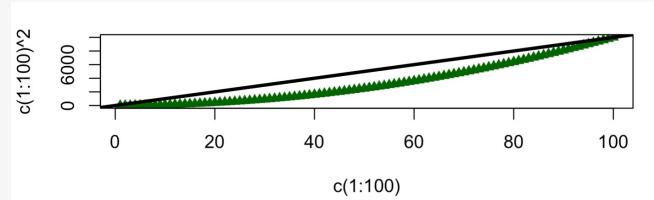
- Draw a new plot.
 > `plot(c(1:100), c(1:100)^2)`
- Tailor its appearance with optional arguments.

```
> 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 -->

white	aliceblue	antiquewhite	antiquewhite1	antiquewhite2
antiquewhite3	antiquewhite4	aquamarine	aquamarine1	aquamarine2
aquamarine3	aquamarine4	azure	azure1	azure2
azure3	azure4	beige	bisque	bisque1
bisque2	bisque3	bisque4		blanchedalmond
blue	blue1	blue2	blue3	blue4
blueviolet	brown	brown1	brown2	brown3
brown4	burlywood	burlywood1	burlywood2	burlywood3
burlywood4	cadetblue	cadetblue1	cadetblue2	cadetblue3
cadetblue4	chartreuse	chartreuse1	chartreuse2	chartreuse3
chartreuse4	chocolate	chocolate1	chocolate2	chocolate3
chocolate4	coral	coral1	coral2	coral3
coral4	cornflowerblue	cornsilk	cornsilk1	cornsilk2
cornsilk3	cornsilk4	cyan	cyan1	cyan2
cyan3	cyan4	darkblue	darkcyan	darkgoldenrod
darkgoldenrod1	darkgoldenrod2	darkgoldenrod3	darkgoldenrod4	darkgray
darkgreen	darkgrey	darkkhaki	darkmagenta	darkolivegreen
darkolivegreen1	darkolivegreen2	darkolivegreen3	darkolivegreen4	darkorange
darkorange1	darkorange2	darkorange3	darkorange4	darkorchid
darkorchid1	darkorchid2	darkorchid3	darkorchid4	darkred
darksalmon	darkseagreen	darkseagreen1	darkseagreen2	darkseagreen3
darkseagreen4	darkslateblue	darkslategray	darkslategray1	darkslategray2
darkslategray3	darkslategray4	darkslategrey	darkturquoise	darkviolet
deeppink	deeppink1	deeppink2	deeppink3	deeppink4
deepskyblue	deepskyblue1	deepskyblue2	deepskyblue3	deepskyblue4

<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)

0	1	2	3	4
□	○	△	+	×
5	6	7	8	9
◇	▽	⊗	*	◇
10	11	12	13	14
⊕	⊗	田	⊗	□
15	16	17	18	19
■	●	▲	◆	●
20	21	22	23	24
●	●	■	◆	▲
25				▽

R line types (lty)



`lty=1` or 'solid'



`lty=2` or 'dashed'



`lty=3` or 'dotted'



`lty=4` or 'dotdash'



`lty=5` or 'longdash'



`lty=6` or 'twodash'

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()



discrete x , continuous y
`f <- ggplot(mpg, aes(class, hwy))`

- `f + geom_col()`, x, y, alpha, color, fill, group, linetype, size
- `f + geom_boxplot()`, x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- `f + geom_dotplot(binaxis = "y", stackdir = "center")`, x, y, alpha, color, fill, group
- `f + geom_violin(scale = "area")`, x, y, alpha, color, fill, group, linetype, size, weight

Additional layers for full customizations
are then added:

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



COLOR AND FILL SCALES (CONTINUOUS)
`o <- c + geom_dotplot(aes(fill = x))`

- `o + scale_fill_distiller(palette = "Blues")`
- `o + scale_fill_gradient(low = "red", high = "yellow")`
- `o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25)`
- `o + scale_fill_gradientn(colors = topo.colors(6))`
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES
`p <- e + geom_point(aes(shape = fl, size = cyl))`

- `p + scale_shape() + scale_size()`
- `p + scale_shape_manual(values = c(3:7))`
- `p + scale_radius(range = c(1,6))`
- `p + scale_size_area(max_size = 6)`

ggplot2

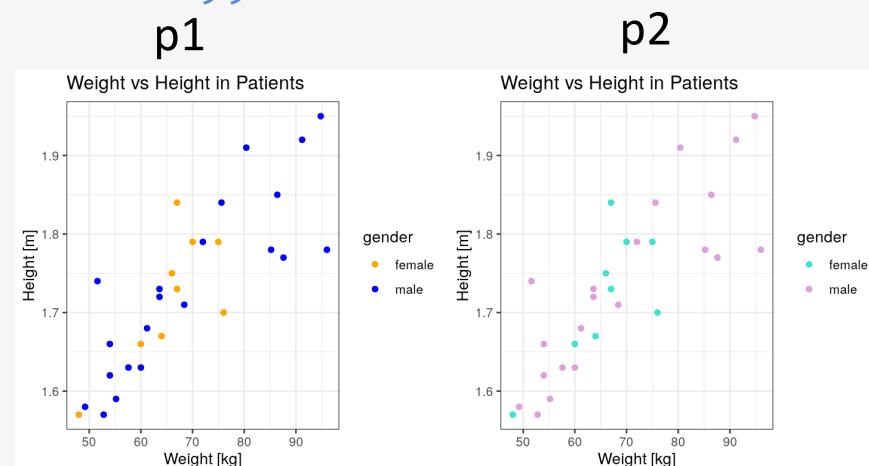
Many other packages offer additional ggplot2 functionalities

- 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)
```



- 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
- ...

04

Basic Statistics in R

Statistical hypothesis testing

Two hypotheses in competition:

- H_0 : the NULL hypothesis (usually the most conservative – e.g., “no difference”)
- H_1 : the alternative hypothesis (usually the one we are actually interested in)

Example:

- H_0 : « There is no difference in the expression of a gene between two given subtypes of melanoma»
- H_1 : « 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 H_0 , 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>

Glitr.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
```

Sepal.Length Sepal.Width Petal.Length Petal.Width Species
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