STATISTICS & ML WITH R

Intro to R and data analysis

2024

M. Chiara Mimmi & Luisa M. Mimmi

WORKSHOP SCHEDULE

- Modules
  - 1. Intro to R and data analysis
  - 2. Statistical inference & hypothesis testing
  - 3. Modeling correlation and regression
  - 4 Mapping causal & predictive approaches
  - 5. Machine Learning
  - 6. Extra topics:
    - MetaboAnalyst;
    - Power Analysis

- Each day will include:
    - Frontal class (MORNING)
    - Practical training with R about the topics discussed in the morning. (AFTERNOON)

DAY 1 – LECTURE OUTLINE

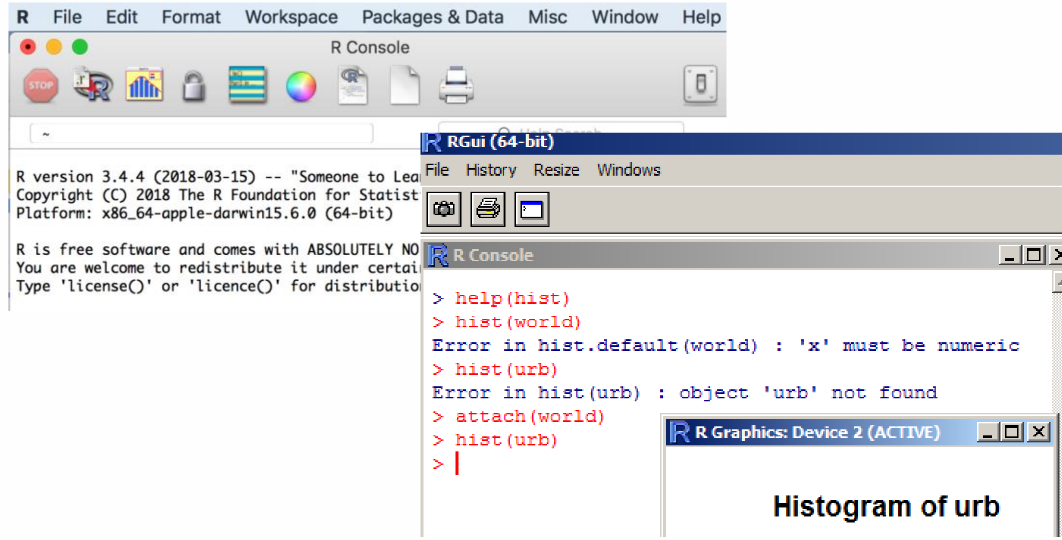
- Introduction to R and R-studio
  - Why R?
  - Principles of reproducible analysis with R + RStudio
- R objects, functions, packages
- Understanding different types of variables
  - Principles of "tidy data"
  - Data cleaning and manipulation
- Descriptive statistics
  - measures of central tendency, measures of variability (or spread), and frequency distribution
- Visual data exploration
  - {ggplot2}

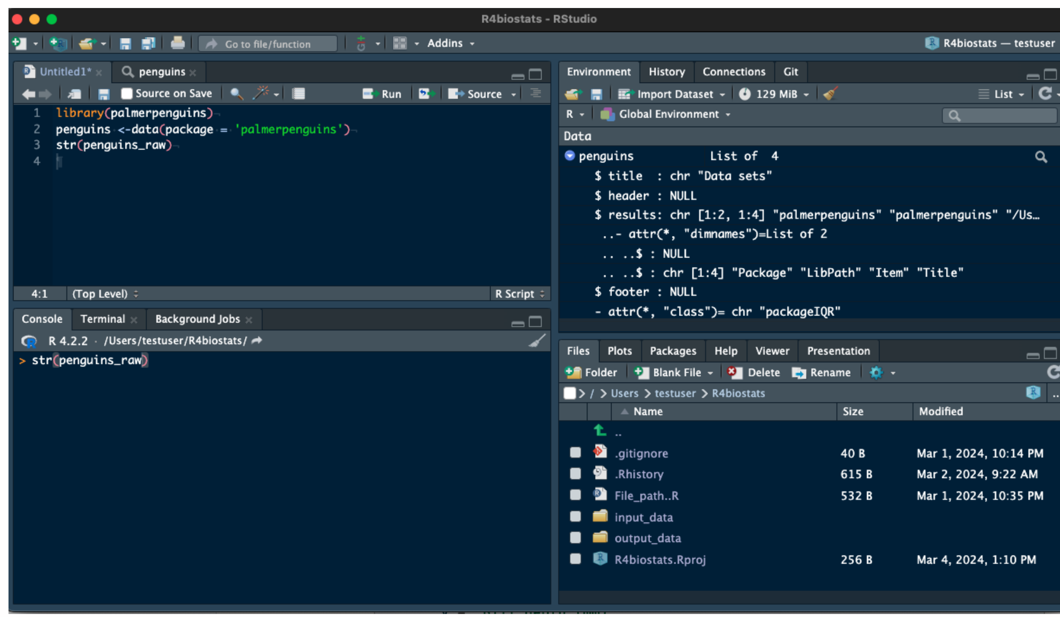
WHY learning and using R ?

- R is an analytical tool created over 25 years ago by and for statisticians
- It is free and open-source, i.e. built by a large, vibrant community of users (including from life science/epidemiology/healthcare fields), in academia and industry
- Beyond statistical computing, the R ecosystem enables many different tasks: data visualization, academic publishing, web scraping, automated reporting, website building…
- …*while* it inherently promotes reproducible research:
  - It is a “scripted” language (better than “point & click tools”)
  - It is OS agnostic
  - It integrates files’ version control systems (git)
  - It integrates literate programming tools (.Rmd, .qmd) for combining code + comments

Different ways to code with R

1. The native **R GUI** (Graphic User Interface)
2. Your **CLI** (command line interface) – on Mac terminal, On PC shell
3. Practically any **Code Editor** (VS Code, Sublime Text, Jupyter Notebook, etc.)
4. Our favorite: **Rstudio IDE** (Interactive Development Environment)!





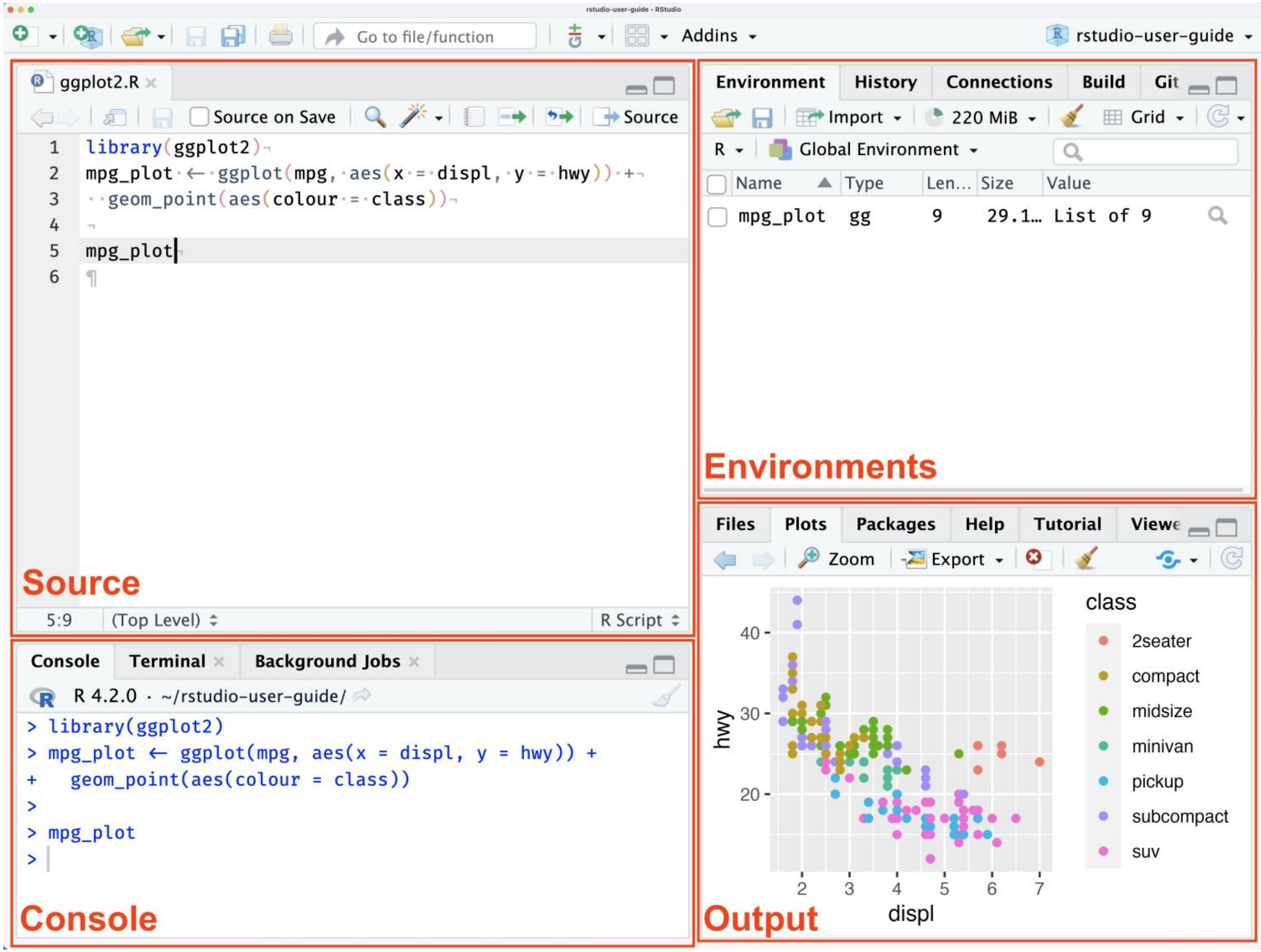
How do I install R and RStudio?

1. To install **R** go to https://cloud.r-project.org/
    - Download appropriate version (Windows, Linux, macOS)

2. (Then) you can install **Rstudio Desktop** going to https://posit.co/download/rstudio-desktop/
    - Download appropriate version (Windows, Linux, macOS)
    - “Desktop” is the free version (there is also a paid pro one)
    - No need to have the latest version, but I recommend v2022.07.1 or later (it includes Quarto*)

Check out the RStudio “CHEATSHEET”!
https://rstudio.github.io/cheatsheets/rstudio-ide.pdf

We will use RStudio Desktop as our IDE

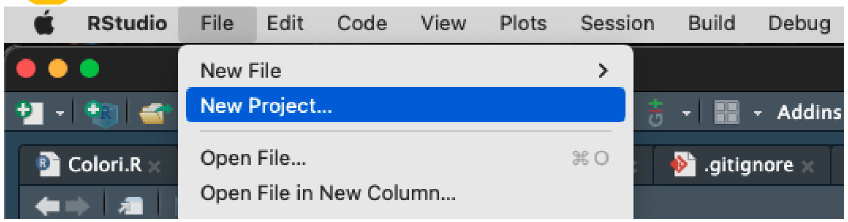


The 4 primary panes in the Rstudio interface

Source: https://docs.posit.co/ide/user/ide/guide/ui/ui-panes.html

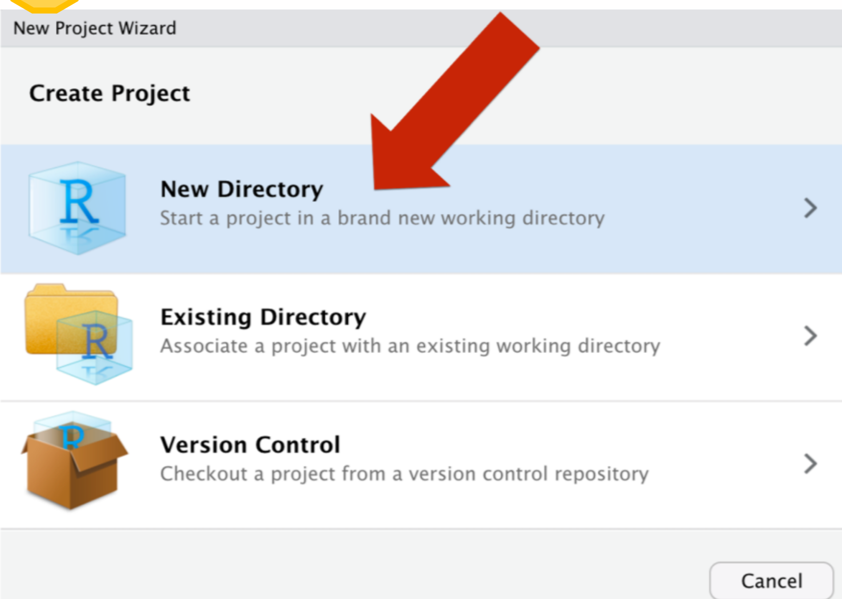
How do I create an Rproject in RStudio?

1

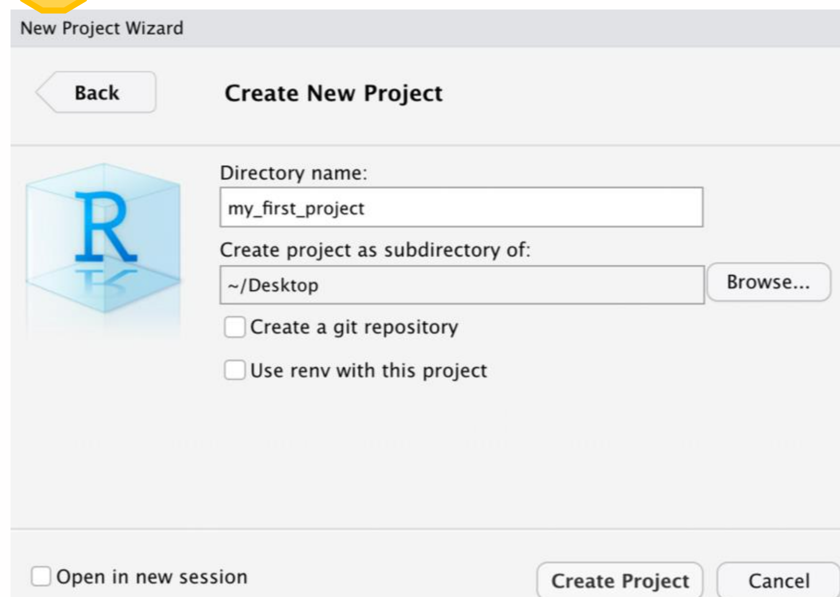


- Easy, just follow RStudio prompts
- Create a new directory for each project
- Select parent folder

2

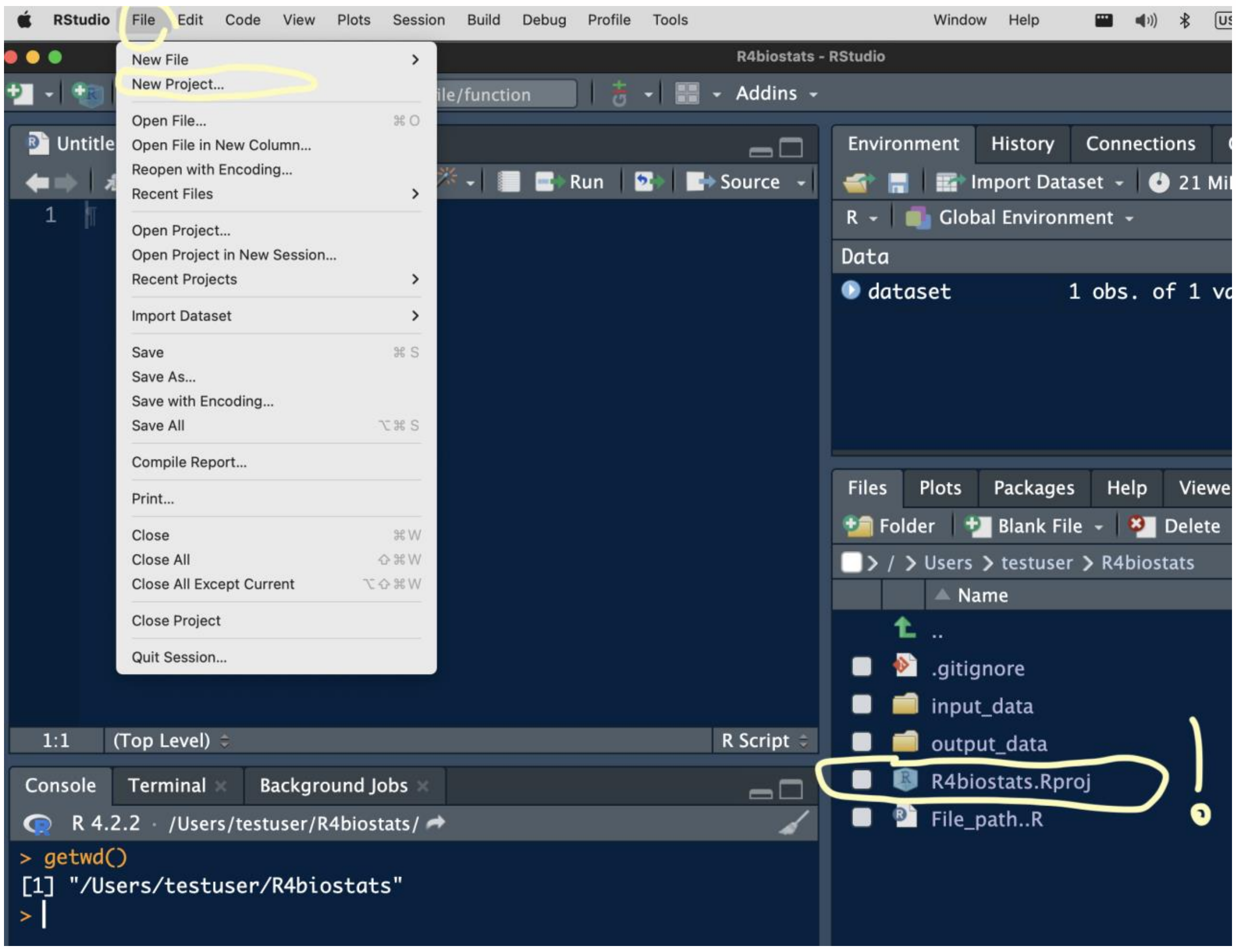


3

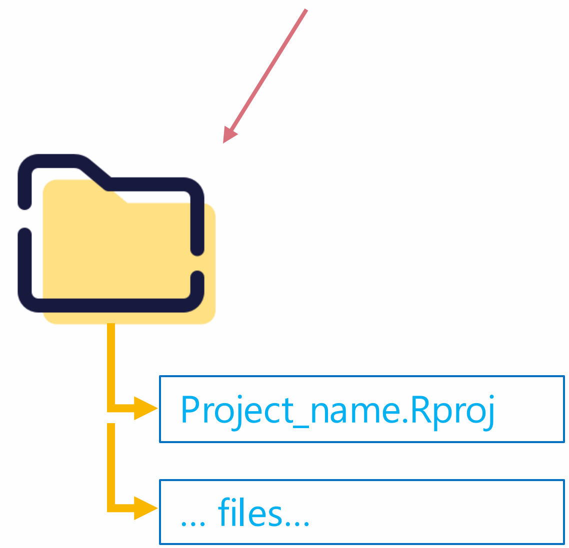


What does the Rproject do?

- Organizes all R code + complimentary files associated to a project
- Set the project's "**working directory**" as the folder containing the "***.Rproj**" file



Working Directory
key to identify/define
FILE PATHS



Projects' files organization with Rproject

typical files' organization system in data-driven research project

- **data/**
  - raw_data/
  - tidy_data/
  - codebook.md
- **code/**
  - raw_code/
  - final_code/
- **figures/**
  - exploratory_figures/
  - explanatory_figures/
- **products/**
  - writing/
    - journal_article
    - presentation
- .gitignore
- README.md
- project.Rproj

3 CRITICAL PIECES!

- ✓ **.gitignore**
  - a list of files to be ignored by version control systems (protected data, very large files…)
- ✓ **README.md**
  - a file that describes the project's content, its purpose, and guidelines to cite, contribute etc.
- ✓ **project.Rproj**
  - an RStudio system file telling R that all folder's files belong together
  - it automatically sets the working directory

Organizing data/folder (for reproducible analysis)

Future-YOU will thank you!

- **data/**
  - raw_data/
  - tidy_data/
  - codebook.md
- **code/**
  - raw_code/
  - final_code/
- **figures/**
  - exploratory_figures/
  - explanatory_figures/
- **products/**
  - writing/
    - journal_article
    - presentation
- .gitignore
- README.md
- project.Rproj

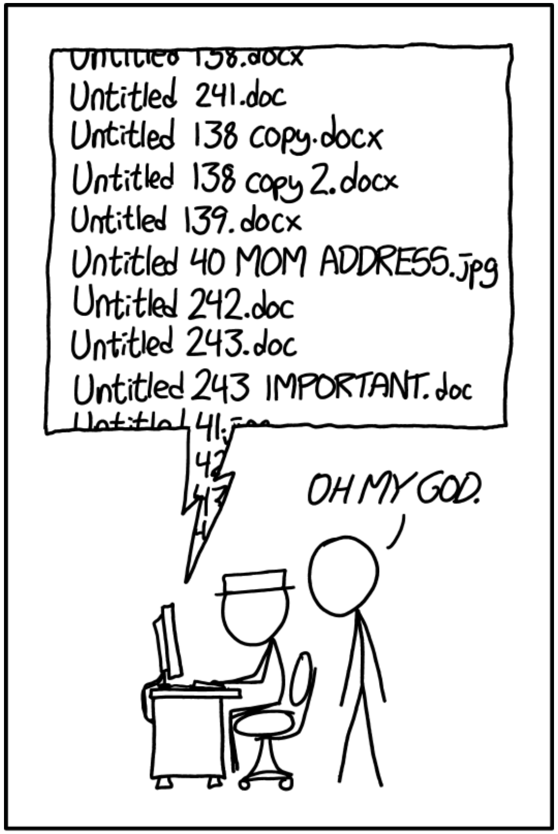
REMEMBER:

✓ **raw_data/…**
- must be saved **BEFORE** being processed in any way
- …then treated as **READ ONLY**!!

✓ You need a **data-codebook** file
- with all **metadata** on your variables (name, type, range of values, missing values, etc)

✓ **tidy_data/…** or **output_data/…**
- must be kept separately from raw data
- ideally in "**tidy**" form

A file naming system is crucial too…



Source: https://xkcd.com/1459

> "Pick a filename convention.
> Any convention.
> Just pick one."
>
> Jenny Bryan, statistics professor and R developer

Bad naming	Good naming
1 my file.md	01_file_labA.csv
10.my.tenth.file.md	02_file_labB.csv
2 (actual) second?-file.md	YYYY_MM_DD_version.md
Extremely-long LOOOONG toolong-name.csv	YYYY_MM_DD_author_report.md
M@ybe THE-worst! Name 3.md	reportA_v01.txt
	reportA_v02.txt

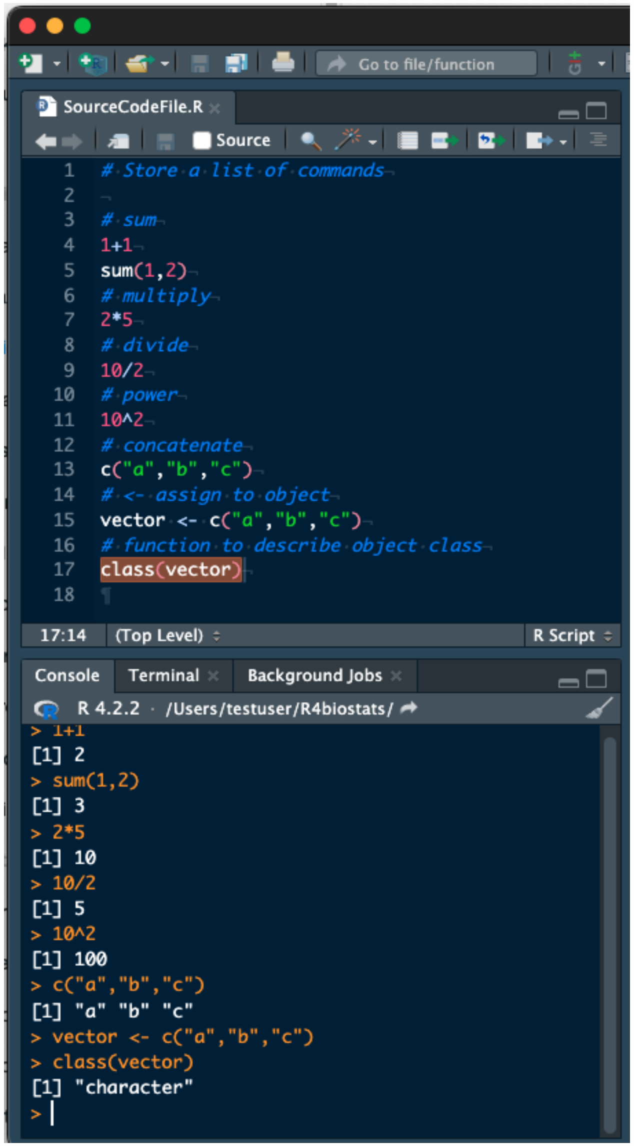
human-readable & machine-readable

Other techniques for reproducible analysis

1. Analyze data via "scripts" (to store code)—instead of manually or with point-&-click tools leaving no trace of analytical steps
   - eg. with R, Python, Stata, Excel macros
2. Automate repeated tasks
   - "DRY" principle (Don't Repeat Yourself!)
   - Organize procedures into dedicated functions (that clean data,split datasets, create graphs…)
3. Use a version control system (VCS)
   - Git, Github, OSF files
4. Use open source software (where possible)
   - Yay R!
5. Use and create open data (where possible)

How to execute R commands (2 ways)

1



2

entering 3 or more characters of a command into the console or a script will trigger autocomplete!

1) In the **Source** pane, you write/save code scripts (***.R** or ***.Rmd, *.qmd**) including:

- R commands
- `#comments' lines`

2) In the **Console** pane, you write R commands interactively

- & see the output just below

DAY 1 – LECTURE OUTLINE

- Introduction to R and R-studio
  - Why R?
  - Principles of reproducible analysis with R + RStudio
- R objects, functions, packages
- Understanding different types of variables
  - Principles of “tidy data”
  - Data cleaning and manipulation
- Descriptive statistics
  - measures of central tendency, measures of variability (or spread), and frequency distribution
- Visual data exploration
  - {ggplot2}

R objects

- Everything in R is an **object!** (*hence it's called Object-Oriented-Language*)
- The ← («assign») operator gives a name to a value (object) and saves it in your workspace

Object Type	Meaning	Example
integer	whole numbers	1
logical	values of true or false	TRUE
double	floating point non-whole numbers	2.5
character	character strings	"R is very cool"
function	code defining a function	func <- plot()
vector	values (of same type) stored in an object together	intg_vector <- c(1, 2, 5) char_vector <- c("blue", "red", "green")
matrix	Rectangular data object where every element is of the same type	matrix_data <- cbind(1:3, 4:5)
data.frame	Rectangular data object that can contain different object types (numeric, character, factor, dates, etc.)	data_frame_data <- dplyr::starwars **name | height | homeworld** Luke Skywalker | 172 | Tatooine C-3PO | 167 | Tatooine
list	Objects which contain elements of different types (numbers, strings, vectors, dataframe or even list) …	list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)

R functions

- A function is a set of statements organized together to perform a specific task
- R functions:
  - take 1 or more inputs (*arguments*)
  - contain operation to be executed inside **{…}**
  - either produce an ouput (*return value/object*) or perform a task (i.e. save a file)
- R functions can be:
  - built-in
  - user defined (and shared by other programmers inside packages)
  - created by you!

DEFINING +
CALLING YOUR OWN function !

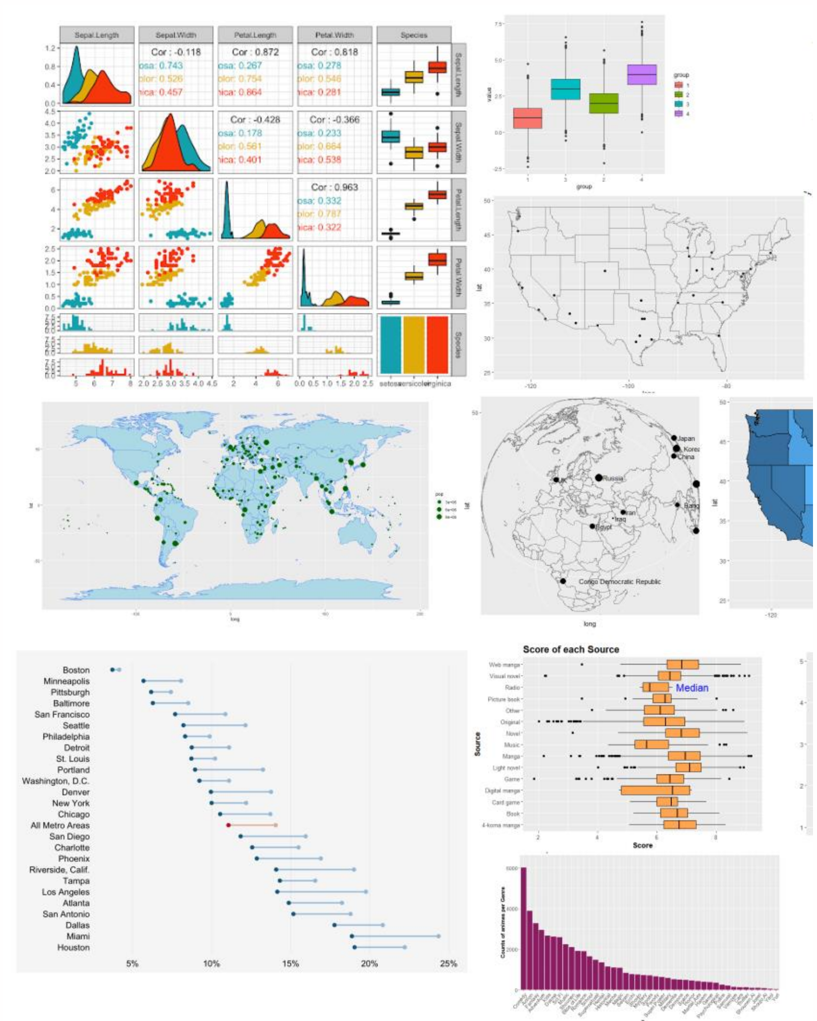
```r
# Define a function that takes input `temp_F` ----
fun_fahrenheit_2_celsius <- function(temp_F) {
    # operations to be executed inside `{}`
    temp_C <- (temp_F - 32) * 5 / 9
    # returned output
    return(temp_C)
}

# Call the function ----------------------------
fun_fahrenheit_2_celsius(75)

# response
# [1] 23.88889
```

R packages

- R has many useful functions built in…

- …or you can add custom libraries or "packages" = coherent *collections of functions* for specific needs

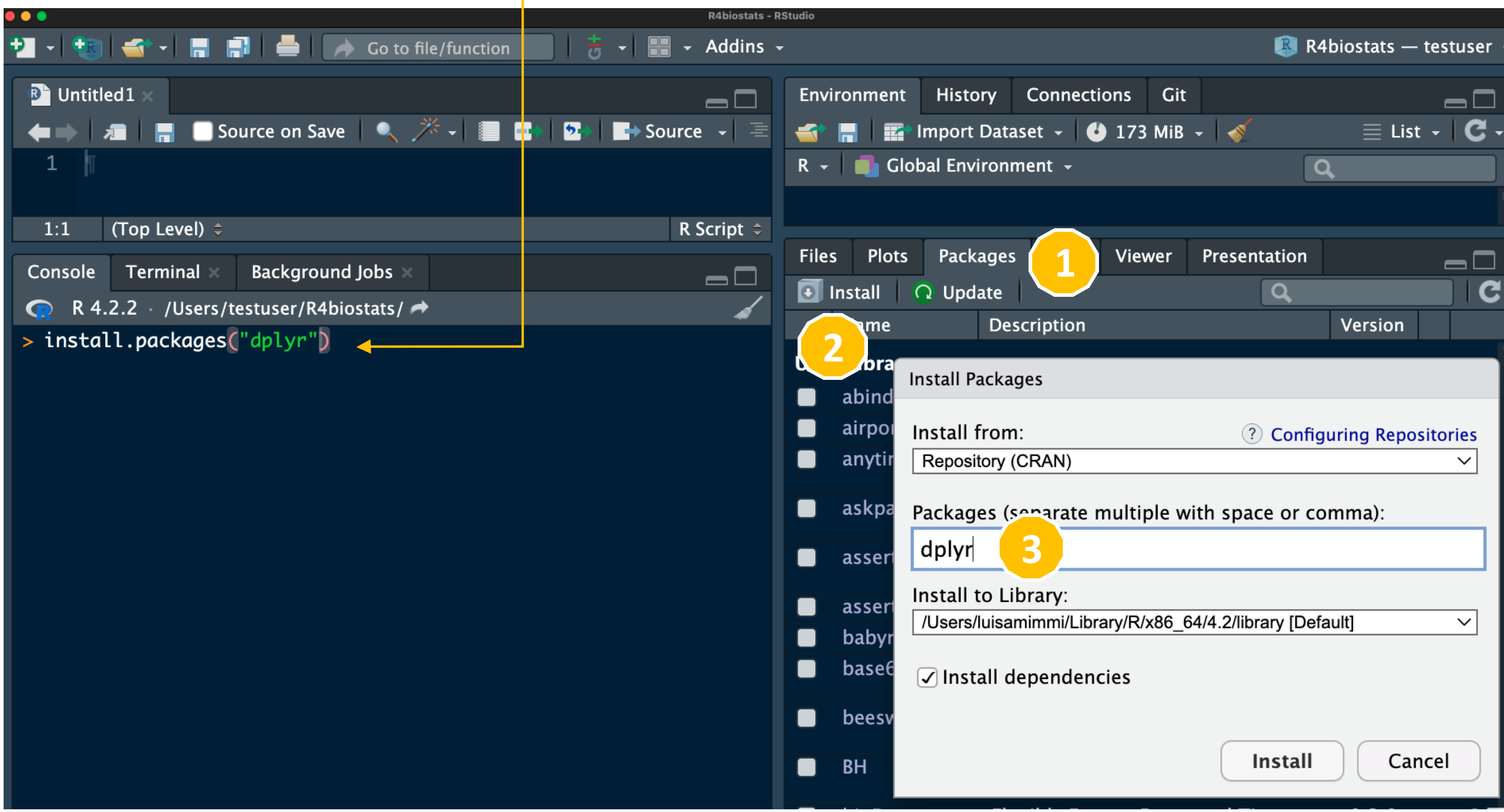
  - EXAMPLE: ***ggplot2*** is an R package which introduces a variety of powerful visualisation and graphing options
    - (we will use it in our Lab session)



R package installation in RStudio: 2 ways

1) Writing the command in the console

2) Using the "Packages" tab



R resources

- There are a number of good resources on the web for learning R and seeking answers to questions… ~~copying code~~ 😉

- Many such resources are also free and open and range from:
  - **books**
  - **courses** & **tutorials**
  - **R packages**' documentation and examples
  - even **podcasts**!

- Check out the workshop website "*Acknowledgement*" page for some relevant resources https://r4biostats.com/license_etc.html

DAY 1 – LECTURE OUTLINE

- Introduction to R and R-studio
  - Why R?
  - Principles of reproducible analysis with R + RStudio
- R objects, functions, packages
- Understanding different types of variables
  - Principles of “tidy data”
  - Data cleaning and manipulation
- Descriptive statistics
  - measures of central tendency, measures of variability (or spread), and frequency distribution
- Visual data exploration
  - {ggplot2}

Qualitative and quantitative variables

- **Qualitative variables** express a qualitative attribute
  - E.g. hair color, religion, gender, favorite movie, etc.
- The values of a qualitative variable do not imply a numerical ordering
  - E.g. value of the variable “eye color” differ qualitatively; no ordering of eye color is implied
- Qualitative variables are also referred to as categorical variables

- **Quantitative variables**, instead, are measured in terms of numbers
  - E.g. height, weight, shoe size, etc.

Discrete and continuous (quantitative) variables

Quantitative

- **Discrete variables** can only take certain (countable) values
  - E.g. “children in a household” could be 3 or 6, but never 4.53 children

- **Continuous variables** can take any value within the range of the scale
  - [they are measured in units that can be subdivided]
  - E.g. “time to respond to a question” could be 1.64 seconds, because it is measured on a scale that is continuous and not made up of discrete steps

Another classification of variables (within experimental settings)

- When conducting research, experimenters often manipulate variables
  - E.g. if comparing the effectiveness of 4 types of antidepressants, the treatment variable is "type of antidepressant"

- When a variable is manipulated by an experimenter, it is called an independent (or *explanatory*) variable
  - Our experiment seeks to determine the effect of the independent variable on the effect in terms of "relief from depression"

- In this example, the measured effect/outcome is a dependent (or *response*) variable

In general, the independent variable is manipulated by the researcher and its effects on the dependent variable are measured

Variables' levels of measurement

informative value

- **Nominal** variables assign a label/name to each of the possible response categories, but there is no natural order
  - e.g. gender, blood type, favorite color, religion
- **Ordinal** variables assign a label/name to each of the possible response categories, but the ranking of responses is meaningful
  - e.g. consumer satisfaction levels, military rank, class ranking

Qualitative

- **Interval** variables assign ordered values that have equal intervals, but the zero-point is arbitrary (i.e. your can have negative values)
  - e.g. Celsius temperature scale, pH
- **Ratio** variables contain all the informative qualities of the *nominal*, *ordinal*, and *interval* scales, plus have a true zero point (i.e. zero has a meaning, like absence of the phenomenon)
  - e.g. age, weight, height, dose amount, pulse

Quantitative

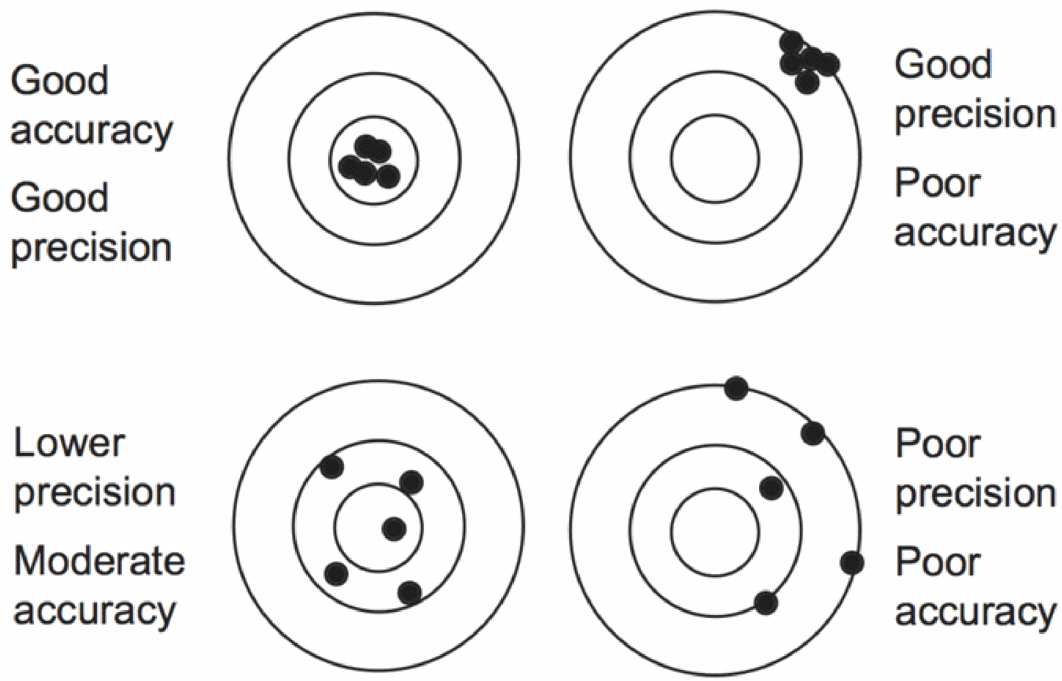
The measurement scale matters for data analysis

Knowing the types of data and levels of measurement is crucial for choosing the appropriate statistical tools to analyze data

	Level of measurement			
OK to compute....	**Nominal**	**Ordinal**	**Interval**	**Ratio**
Frequency distribution	Yes	Yes	Yes	Yes
Median and percentiles	No	Yes	Yes	Yes
Add or subtract	No	No	Yes	Yes
Mean, standard deviation, standard error of the mean	No	No	Yes	Yes
Ratios, coefficient of variation	No	No	No	Yes

Measurement accuracy and precision

- **Accuracy** – how close a measurement is to the true value of whatever it is you are trying to measure.
- **Precision** – how repeatable a measure is, irrespective of whether it is close to the actual value.
  - The ideal is the top left target in the diagram,
  - 2nd best would be bottom left - measurements that are reasonably accurate even though not too precise



Source: https://tuos-bio-data-skills.github.io/intro-stats-book/data-variables.html

Error and bias

- **Error** is present in almost all biological data, but not all error is equally problematic. The worst error is
  - **bias** = a systematic lack of accuracy, i.e. *all data deviate from the true measurements in the same direction*

- Common causes of data measurement containing bias:
  - **Non-random sampling**
  - **Conditioning of biological material**
  - **Interference by the process of investigation**
  - **Investigator bias**

Data types in R corresponding to types of variables

Type of Variable	Data types in R	Value type	Example	Notes
continuous	***Numeric***	decimals	`num_dec <- c(3.4, 7.1, 2.9)`	
numeric	***Integer*** *(special case of numeric)*	whole numbers	`num <- c(3, 7, 2)`	*use if you are sure that the numbers you store will never contains decimals*

nominal	***Character***	text contained in ""	`char <- "some text"`
			`also_char <- "2.3"`
			`also_char_2 <- c("text", 1, 3.72, 4)`
				IF a number or an element in a vector is inside **""** *--> R will store as character*

ordinal	***Factor*** *(special case of character)*	text contained in ""	`ranking_char <- c("low","medium","high")` `ranking_factor <- factor(ranking_char, levels = c("low","medium","high"))`	

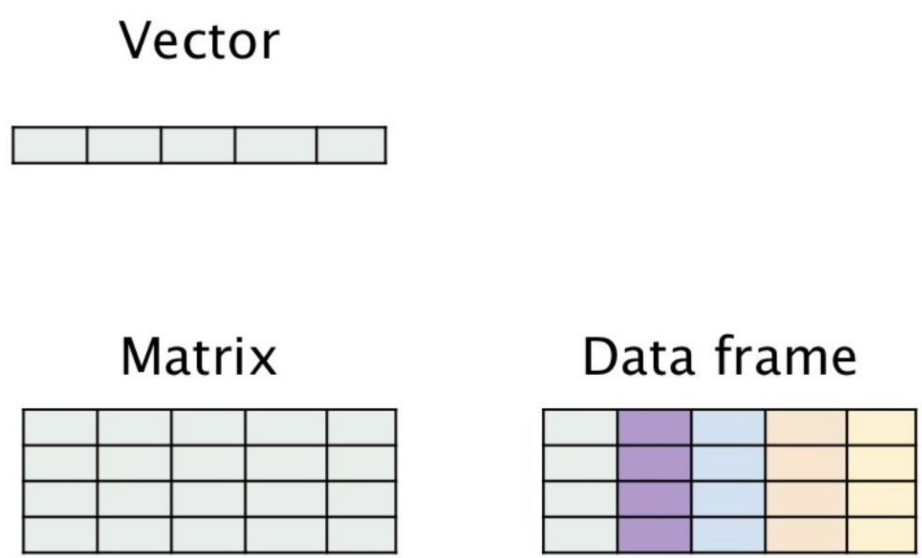
boolean	***Logical***	TRUE or FALSE	`greater <- 1 > 10`
			`greater`
			`[1] FALSE`	

Storing a variable as the wrong data type can generate errors in R programming

Data types in R… and the objects storing them

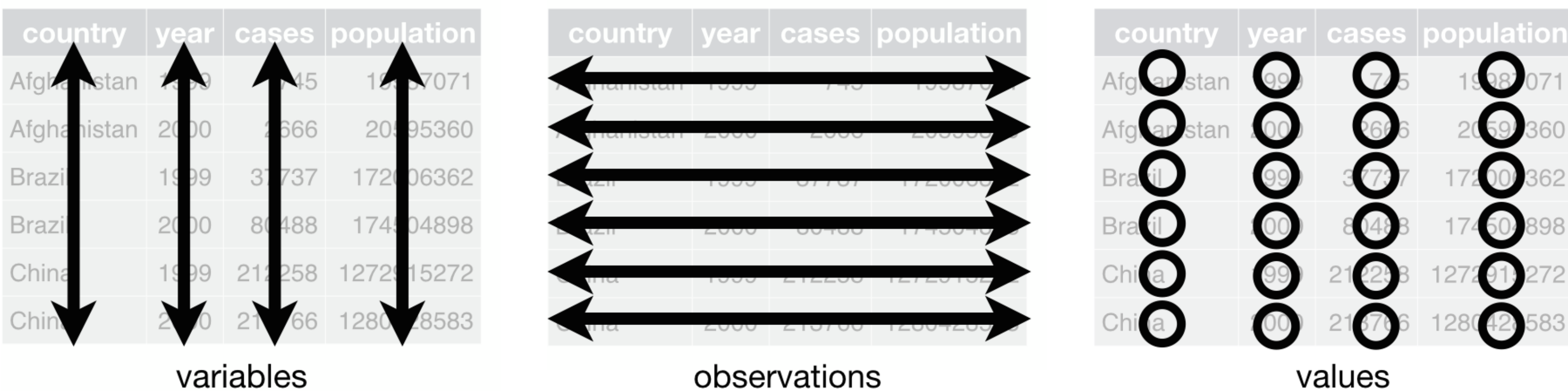
Data type	Type
vector	1D collection of variables of the same type
matrix	2D collection of variables of the same type
data.frame	2D collection of variables of multiple types

Variables	Example
integer	100
numeric	0.05
character	“hello”
logical	TRUE
factor	“Green”



Tidy (rectangular) data makes analysis easier

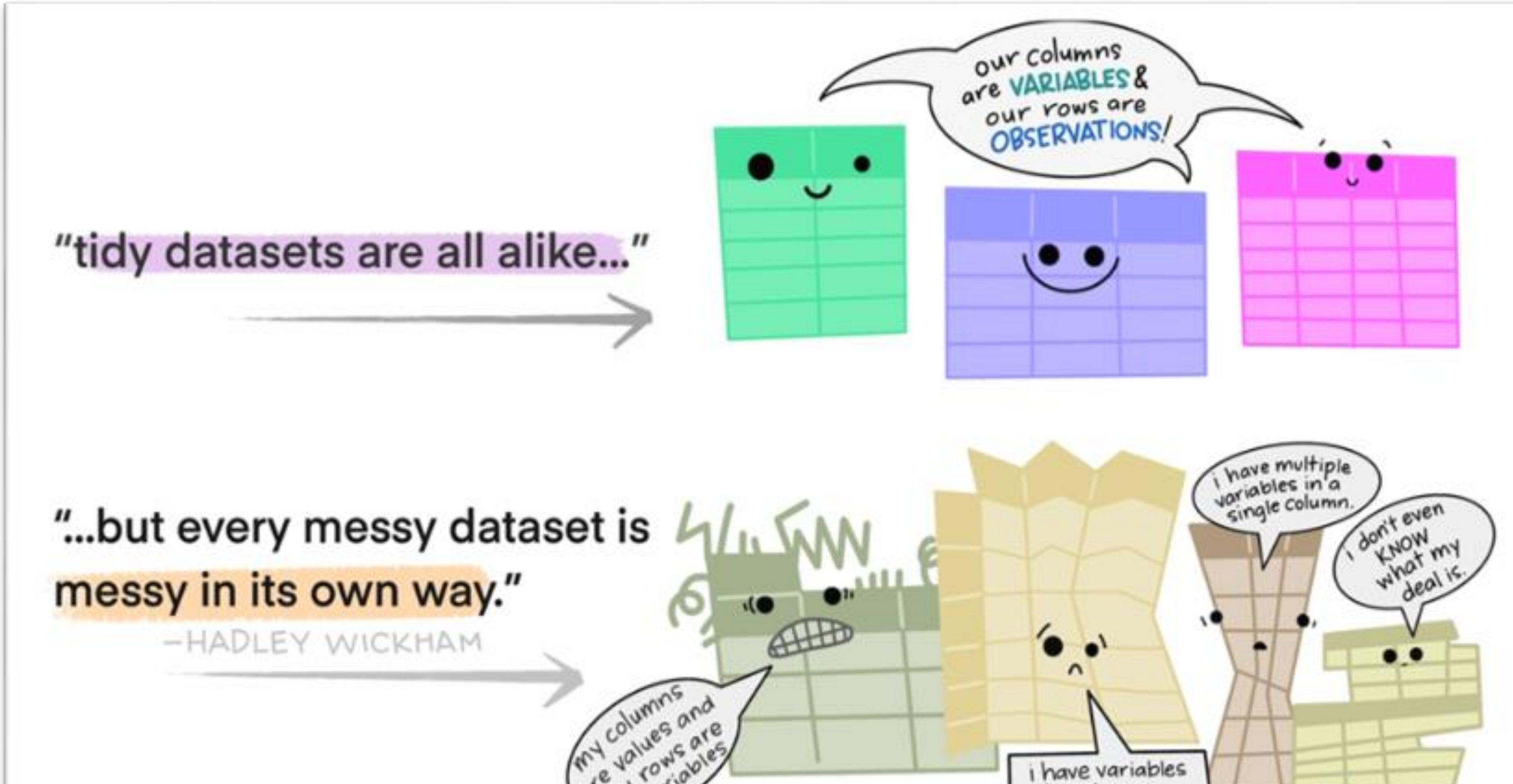
- In R, we are working with tables, which are stored in special data structures called "data frames" (or "tibbles").
  - a typical example is a table in a relational database
- In (**tidy**) rectangular data structures:
  - **columns** will represent different variables associated with each data point or instance e.g. Name, ID, location, time, value…
  - **rows** will represent instances or individual observations e.g. data points, patients, events, samples, etc. while
  - each **cell** (intersection of row and column) will contain one and only value (i.e. the state of a variable measured for one specific individual/unit)



Source: https://r4ds.hadley.nz/data-tidy.html

Why do we care about tidy data form?

- uniformity & standardization
- readability
- ease to clean/analyze/model/plot data



Source: art by [Allison Horst]

An example of “tidy”, rectangular data frame

```r
bmc_data <- data.frame (name = c("Alice", "Bob", "Carol", "David"),
                        gender = as.factor(c("Female", "Male", "Female", "Male")),
                        disorder = c("autism", "anxiety", "autism", "depression"),
                        age = c(20, 45, 15, 12),
                        biomarker1 = c(5.70, 4.96, 1.37, 10.44),
                        clinicalstage = c("1b", "1a", "1a", "2"),
                        stringsAsFactors = FALSE)
```

Untitled1* bmc.data

Filter

	fname	gender	disorder	age	biomarker1	clinicalstage
1	Alice	Female	autism	20	5.70	1b
2	Bob	Male	anxiety	45	4.96	1a
3	Carol	Female	autism	15	1.37	1a
4	David	Male	depression	12	10.44	2

Reshaping the data: from wide to long form

Suppose we have 3 patients with ids A, B, and C, and we take 2 blood pressure measurements on each patient

- We want our new dataset to have three variables: patient **id**, **measurement**, and **value**

You typically need data in "long" form for modeling and plotting

id	bp1	bp2
A	100	120
B	140	115
C	120	125

id	measurement	value
A	bp1	100
A	bp2	120
B	bp1	140
B	bp2	115
C	bp1	120
C	bp2	125

Source: https://r4ds.hadley.nz/data-tidy.html

Reshaping the data: from long to wide form

You may also need to convert a dataset from long to wide format

- Data stored in 1 column (variable **year**) is now splitted in **separate variables**, so that the table has only 1 row per country

You typically need data in "wide" form for better readability

country	year	cases
Angola	1999	800
Angola	2000	750
Angola	2001	925
Angola	2002	1020
India	1999	20100
India	2000	25650
India	2001	26800
India	2002	27255
Mongolia	1999	450
Mongolia	2000	512
Mongolia	2001	510
Mongolia	2002	586

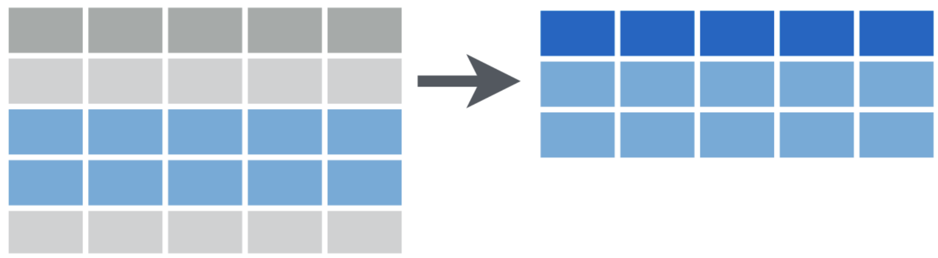
country	1999	2000	2001	2002
Angola	800	750	925	1020
India	20100	25650	26800	27255
Mongolia	450	512	510	586

Source: https://epirhandbook.com/

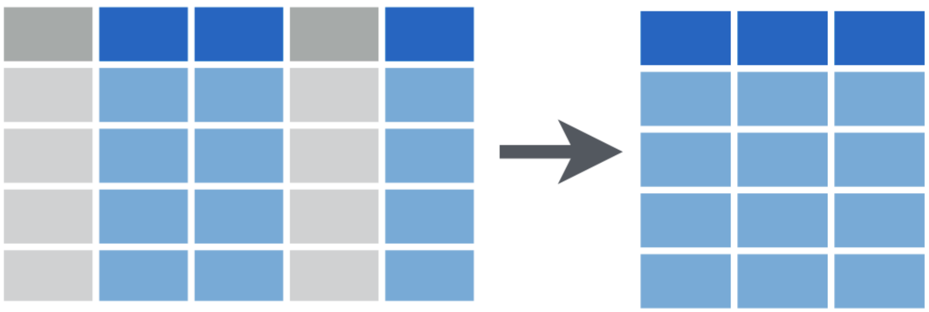
Subsetting the data

- Subsetting is a fundamental action that serves many purposes:
  - viewing and getting a quick sense of the variables
  - analyzing a sub-group of individuals
  - splitting samples
  - etc.
- We will see in the practice sessions that there are many ways to subset a dataframe in R.

Subsetting Observations (rows)



Subsetting Variables (columns)



Dealing with missing data

- Missing data = the value of the variables of interest are not measured or recorded for all subjects in the sample

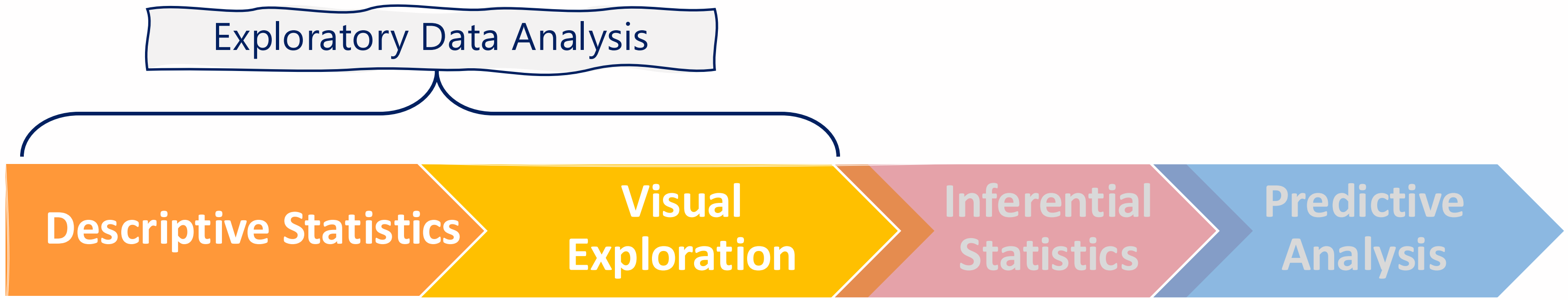
- In clinical research, missing data can be handled with different approaches:
  - complete-case analysis -> subsetting complete cases only
  - mean-value imputation -> missing values are replaced with the mean/median/mode value of that variable
  - model-based imputation -> missing values imputed based on a linear regression model or K-Nearest Neighbors predictive model, interpolation, etc.…
  - multiple imputation (MI) -> multiple plausible values are imputed or filled in for each subject who has missing data for that variable and then multiple datasets are analyzed
  - domain-specific imputation -> justified imputation based on external data (i.e. blood type when you have family history)

1. It is SUPER important to understand the reason some data are missing (random or not) and to what extent they are related to the dependent variable
2. Dealing with missing data can lead to biased estimates of statistics (e.g., of regression coefficients) and/or confidence intervals that are artificially narrow.

DAY 1 – LECTURE OUTLINE

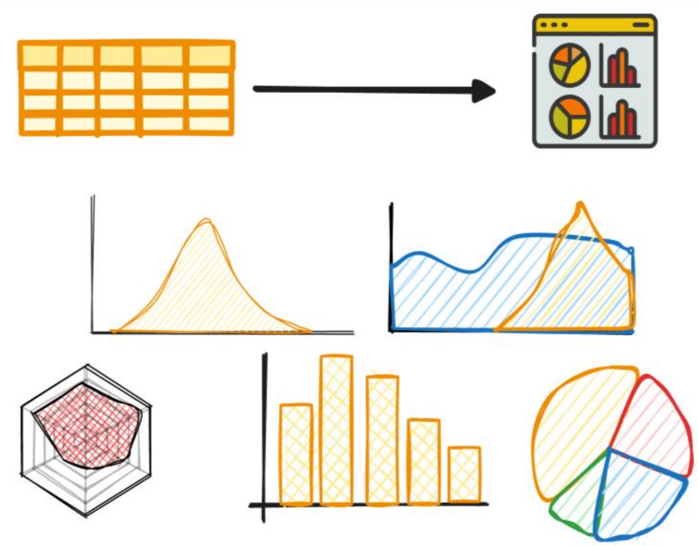
- Introduction to R and R-studio
    - Why R?
    - Principles of reproducible analysis with R + RStudio
- R objects, functions, packages
- Understanding different types of variables
    - Principles of “tidy data”
    - Data cleaning and manipulation
- Descriptive statistics
    - measures of central tendency, measures of variability (or spread), and frequency distribution
- Visual data exploration
    - {ggplot2}

Describe dataset to: familiarize with it(!), check quality, generate hypotheses before inferential modeling



Quantitative & visual approach to learn:

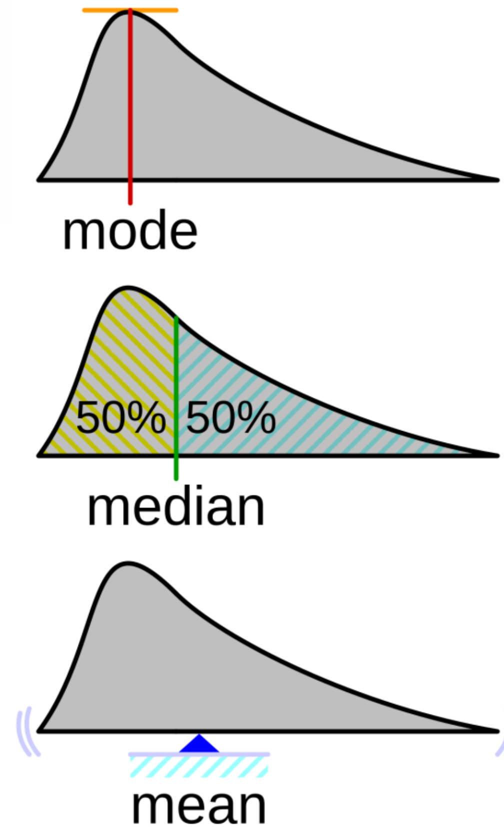
- **VARIABLES' MEASURES OF CENTER & DISPERSION**
  - Mean / Median / Mode
  - Min & max
  - Standard deviation
  - Variance
  - Range / Interquartile Range (IQR)
- **VARIABLES' DISTRIBUTION AND PATTERNS**
  - Frequency distributions
  - Geographical distribution
  - Correlation and join variability
  - Missing observations, anomalies
  - Outliers



Source: https://www.kdnuggets.com/7-steps-to-mastering-exploratory-data-analysis

Measures of central tendency: mean, median, mode

- Mean (sum of all observations divided by n of observations)
- Median (value at the midpoint of a frequency distribution of observed values)
- Mode (the value that is most often observed in the data)



Numerical Variables and Measures of Central Tendency

- **Mean**
  - Simple
  - Only one
  - Susceptible to extreme values
  - Not necessarily observed in data
- **Median**
  - Simple
  - Only one
  - Not susceptible to extreme values
  - Not necessarily observed in data
- **Mode**
  - Simple
  - May be none
  - May be more than one
  - Observed in data by definition

Source: https://www.r4epi.com/measures-of-central-tendency

Mean and median equations

Sample mean $$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

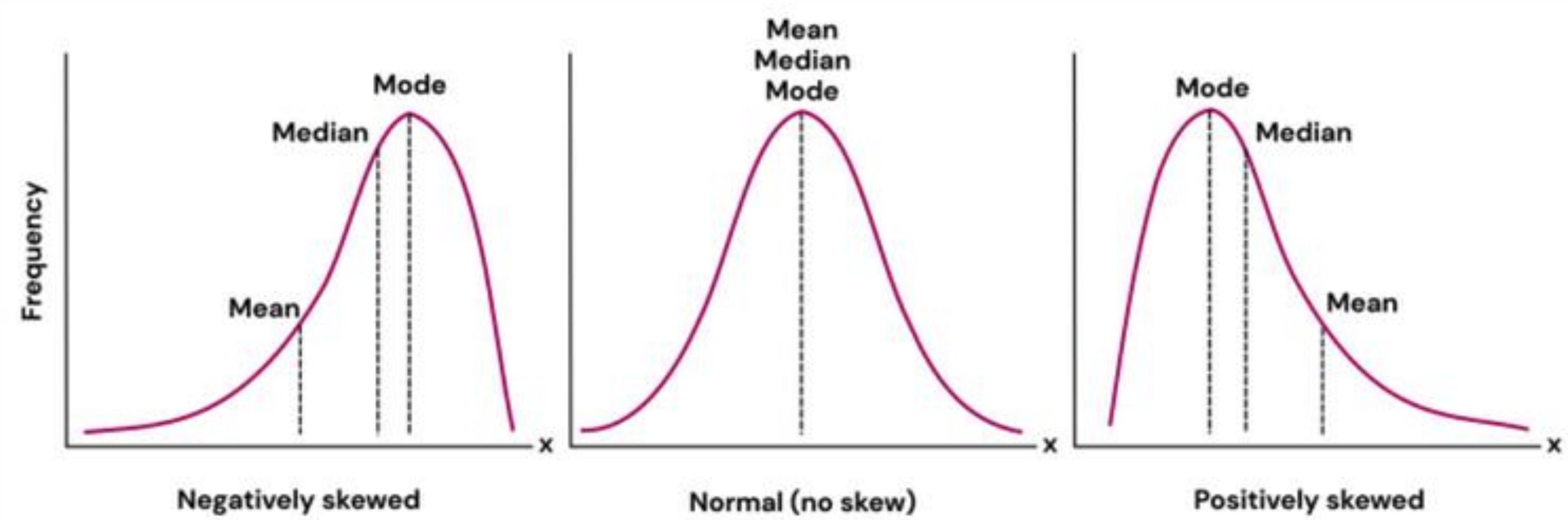
Population mean $$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

Sample Median for odd *n* $$Mdn = x_{\left[\frac{n+1}{2}\right]}$$

Sample Median for even *n* $$Mdn = \frac{x_{n/2} + x_{[(n/2)+1]}}{2}$$

Mean and Median features

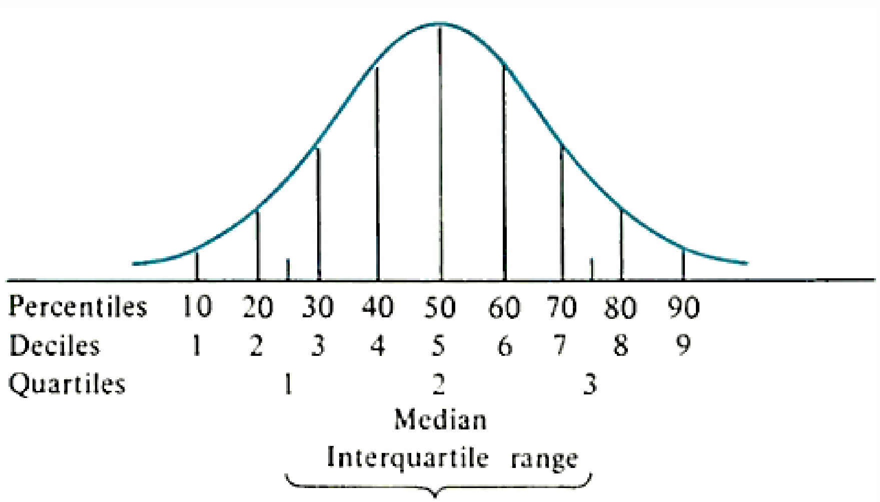
The median is <u>not</u> influenced by:
- variability in the data
- extreme values/outliers



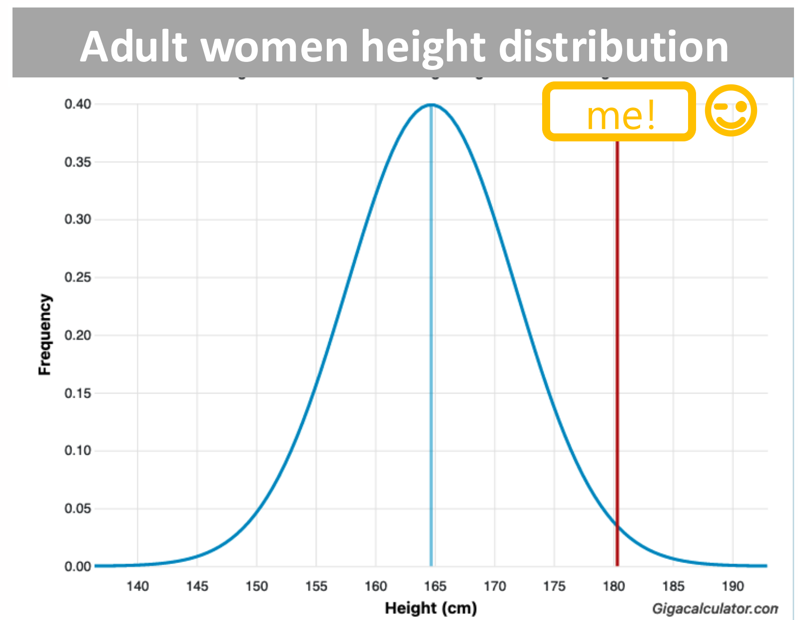
Partition values (quantiles)

- Partition values partition the same collection of observations in several ways
- Similar to the median, depending on their position, quantiles contain portion of observations in the frequency distribution of a variable:
  - quartiles: distribution divided into quarters
  - quintiles: distribution divided into fifths
  - deciles: distribution divided into tenths
  - percentiles : distribution divided into hundredths

For example, I am in the top 2nd percentile of world's adult women for height!

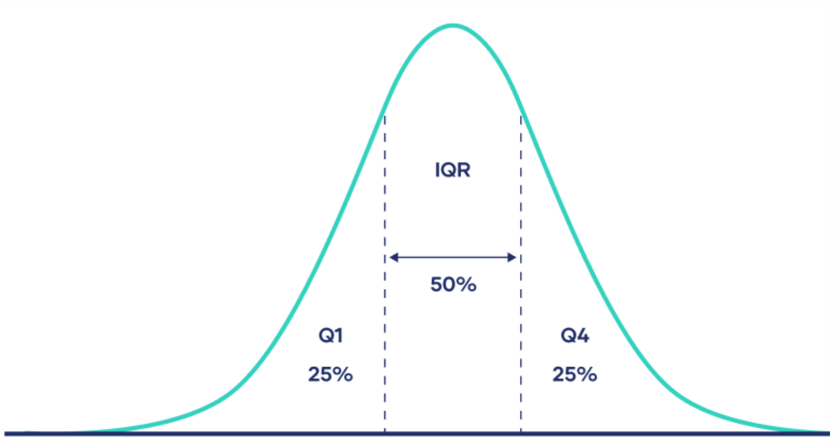


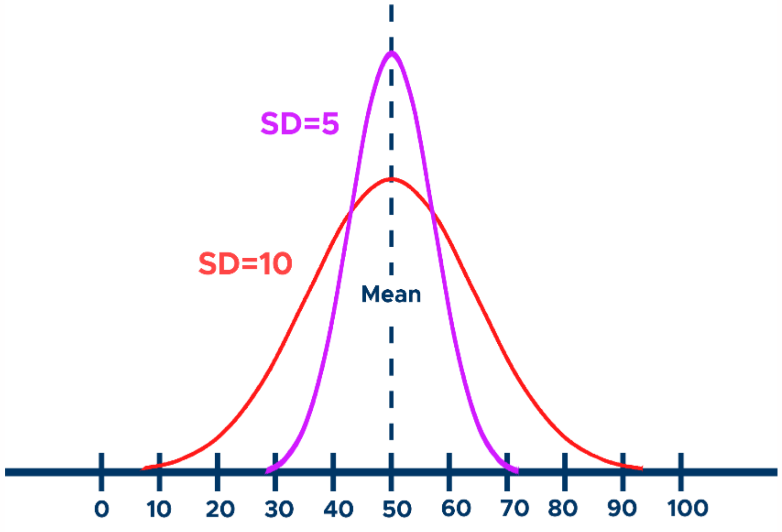
Source: https://itfeature.com/statistics/quantiles-or-fractiles/



Measures of variability or dispersion

- Capable of expressing in one measure the elements of heterogeneity of (quantitative) data:
  - Range: difference between the maximum and minimum values
  - Interquartile Range (IQR): range of the middle half of a distribution
  - Variance: average squared deviation from the mean
  - Standard deviation: square root of the variance
  - Coefficient of variation





Variance and Standard deviation equations

Population variance $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$

Sample variance $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Population standard deviation $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$

Sample standard deviation $s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$

Examples of {base} R built-in functions

(central tendency and dispersion)

Examples of {base} R functions to summarize your data

Built-in {base} R functions

- summary(bmc_data) # can take dataframe as argument
- summary(bmc_data$age) # results change according to var type

- table (bmc_data$age, useNA = "ifany")
- table (bmc_data$gender, useNA = "ifany")

- mean (bmc_data$age, na.rm = TRUE) # sample mean $\overline{x} = \frac{\sum(x_i))}{n}$
- median (bmc_data$age, na.rm = TRUE) # observation corresponding to index $\frac{n+1}{2}$ (when values are in order low to high)

Examples of user-defined function to summarize your data

Our own custom function to calculate **mode**

```r
# Create custom function to calculate statistical mode

f_calc_mode  <- function(x) {
  # `unique` returns a vector of unique values
  uni_x <- unique(x)
  # `match` returns the index positions of 1st vector against 2nd vector
  match_x <- match(x, uni_x)
  # `tabulate` count the occurrences of integer values in a vector.
  tab_x  <-  tabulate(match_x)
  # returns element of uni_x that corresponds to max occurrences
  uni_x[tab_x == max(tab_x)]
}
```

Use new function

```r
f_calc_mode(bmc_data$age)
f_calc_mode(bmc_data$biomarker1)
```

DISPERSION: range, min, max, IQR

- range = difference between the largest and smallest value in a distribution
- IQR (interquartile range) = difference between the first quartile (the 25th percentile) and the third quartile (the 75th percentile) of a dataset

Built-in {base} R functions

- min(bmc_data$age)
- max(bmc_data$age)
- range(bmc_data$age)
- IQR(bmc_data$age, na.rm=TRUE) # Q3 – Q1 (where Q1 and Q3 are the 25th and 75th percentiles)
- var(bmc_data$age) # variance (for sample of size n)

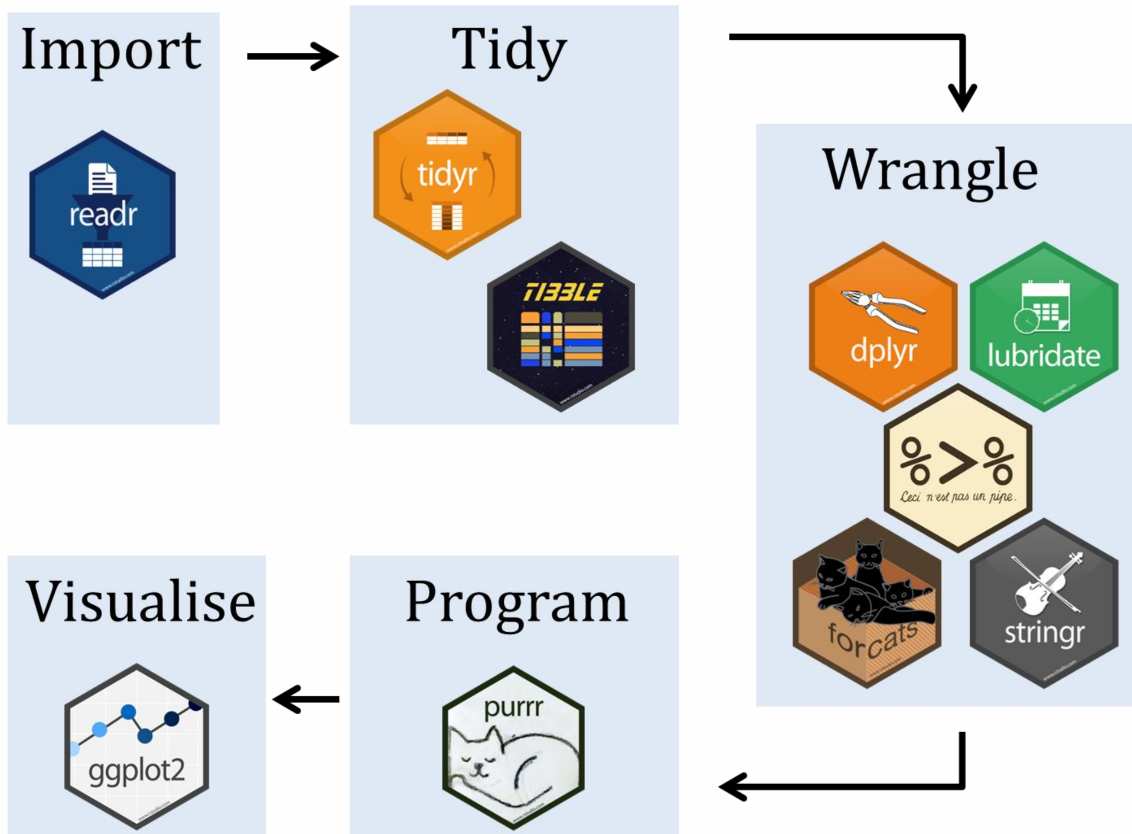
$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- sd(bmc_data$age) # standard deviation (for sample of size n)

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

The {tidyverse} paradigm

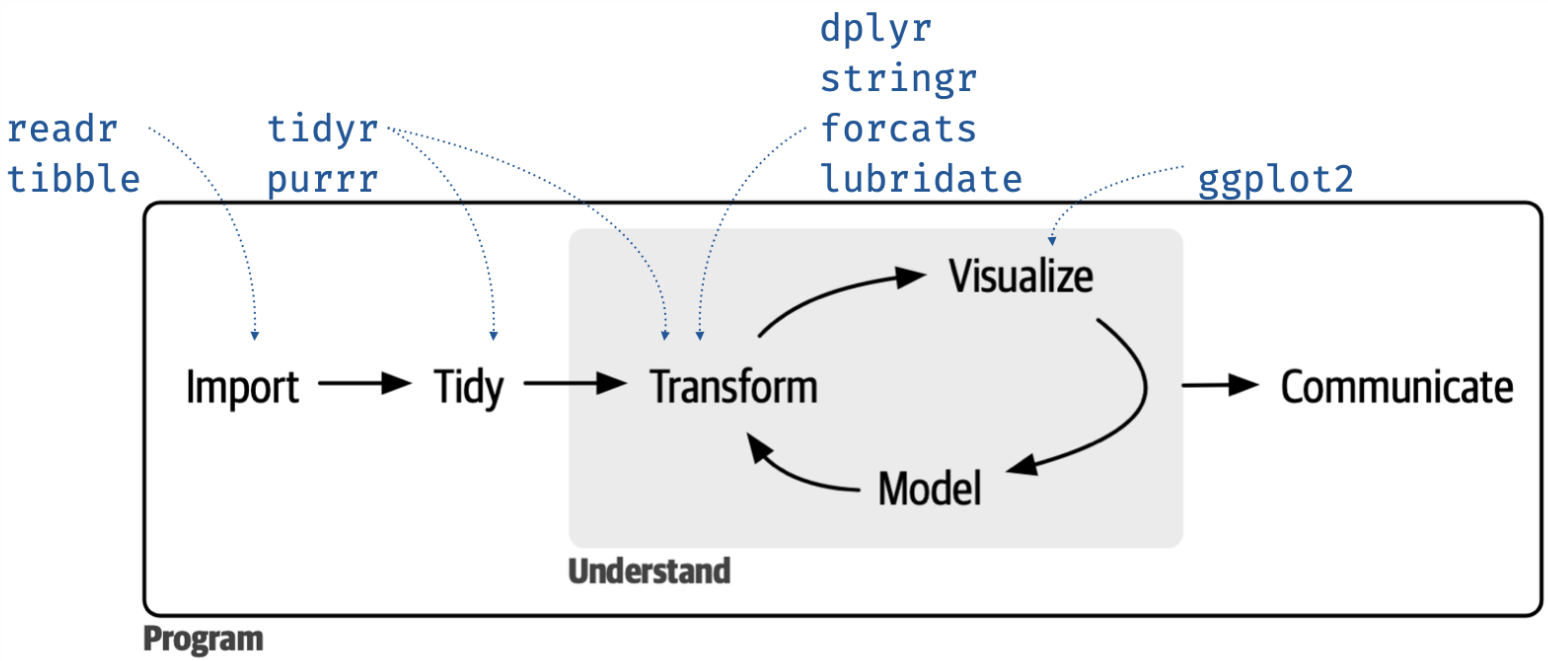
A coherent collection of R packages to carry out the entire analytical process



Source: https://ajsmit.github.io

The {tidyverse} "frame of mind" for data transformation, analysis & plotting

- Besides {base} R functions, many R programmers (especially beginners) enjoy using {tidyverse}, a collection of packages with:
  - consistent philosophy, grammar, and data structures
  - the **pipe** **%>%** linking functions!!
- An alternative package/framework for data analysis (*we won't look at*) is {**data.table**}



Source: https://www.tidyverse.org/blog/2023/08/teach-tidyverse-23/

Some {tidyverse} R functions to transform data

A great package for data wrangling is **{dplyr}**

These **dplyr functions** (reminiscent of SQL commands) reflect the fact what that they 'do' to data.

For example:

- **select** obtains a subset of variables,
- **mutate** constructs new variables,
- **filter** obtains a subset of rows,
- **arrange** reorders rows,
- **group_by** groups based on a categorical variable… then
- **summarise** calculates information about groups.

All {tidyverse} functions can be chained by the pipe %>% operator!

- This operator (from the {magrittr} package) is similar to the one used in other programming languages (e.g. | in the Shell or Terminal)
- %>% allows to easily and intuitive chain sequences of functions:
  - the output of one function becomes the input used by the next function
  - instead of `f(g(x))`, we write our code as `x %>% g() %>% f()`

From

```
leave_house(get_dressed(get_out_of_bed(wake_up(me, time =
"8:00"), side = "correct"), pants = TRUE, shirt = TRUE), car
= TRUE, bike = FALSE)
```

To

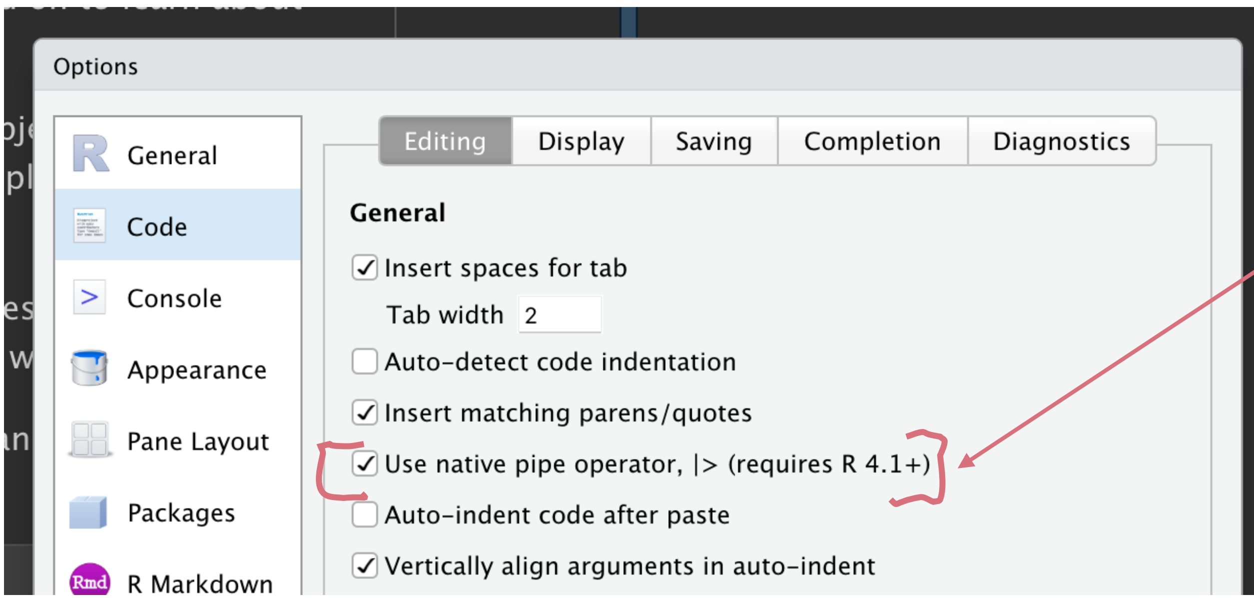
```
me %>%
  wake_up(time = "8:00") %>%
  get_out_of_bed(side = "correct") %>%
  get_dressed(pants = TRUE, shirt = TRUE) %>%
  leave_house(car = TRUE, bike = FALSE)
```

Source: https://talks.andrewheiss.com/2021-seacen/01_welcome-tidyverse/slides/01_tidyverse-dplyr.html#65

R has a new native pipe |> which can replace the {magrittr} packages's one (%>%)

- **R 4.1.0** introduced a native pipe operator, |>, which behaves very similarly to %>%

In RStudio options, you can choose to use the native |> here



Examples of {dplyr} functions for summarizing data

Functions: filter, group_by, summarise

```
# input the dataframe name
bmc_data %>%
  # filter some rows
  dplyr::filter(age >= 15) %>%
  # split by some column
  dplyr::group_by(gender) %>%
  # summarise in each group
  dplyr::summarise(mean(biomarker1))  # returns a dataframe!
```

Results

<fct>	<dbl>
1 Female	3.54
2 Male	4.96

3 alternative R packages/paradigms for data manipulation

OPERATION	PARADIGM		
	base	tidyverse	data.table
Supported data class	data.frame	tibble	data.table
Reading data	read.csv	read_csv	fread
Subset by column	[, …]	select()	[,… ,]
Subset by rows	[… ,]	filter()	[… , ,]
Create new column	df$y = …	mutate(tb, y = …)	[, y := …,]
Delete a column	df$y = NULL	select(tb, -y)	[, y := NULL,]
Summarize	apply(df[, y], 2, …)	summarise()	[, …(y),]
Grouping	aggregate()	group_by()	[, , by = …]
Pivot to long	reshape()	pivot_longer()	melt()
Pivot to wide	reshape()	pivot_wider()	dcast()
Joining tables	merge()	left_join()	dt1[dt2, on = …]

Source: https://jtr13.github.io/cc21fall2/comparison-among-base-r-tidyverse-and-datatable.html#summary-of-key-functions

DAY 1 – LECTURE OUTLINE

- Introduction to R and R-studio
  - Why R?
  - Principles of reproducible analysis with R + RStudio
- R objects, functions, packages
- Understanding different types of variables
  - Principles of "tidy data"
  - Data cleaning and manipulation
- Descriptive statistics
  - measures of central tendency, measures of variability (or spread), and frequency distribution
- Visual data exploration
  - {ggplot2}

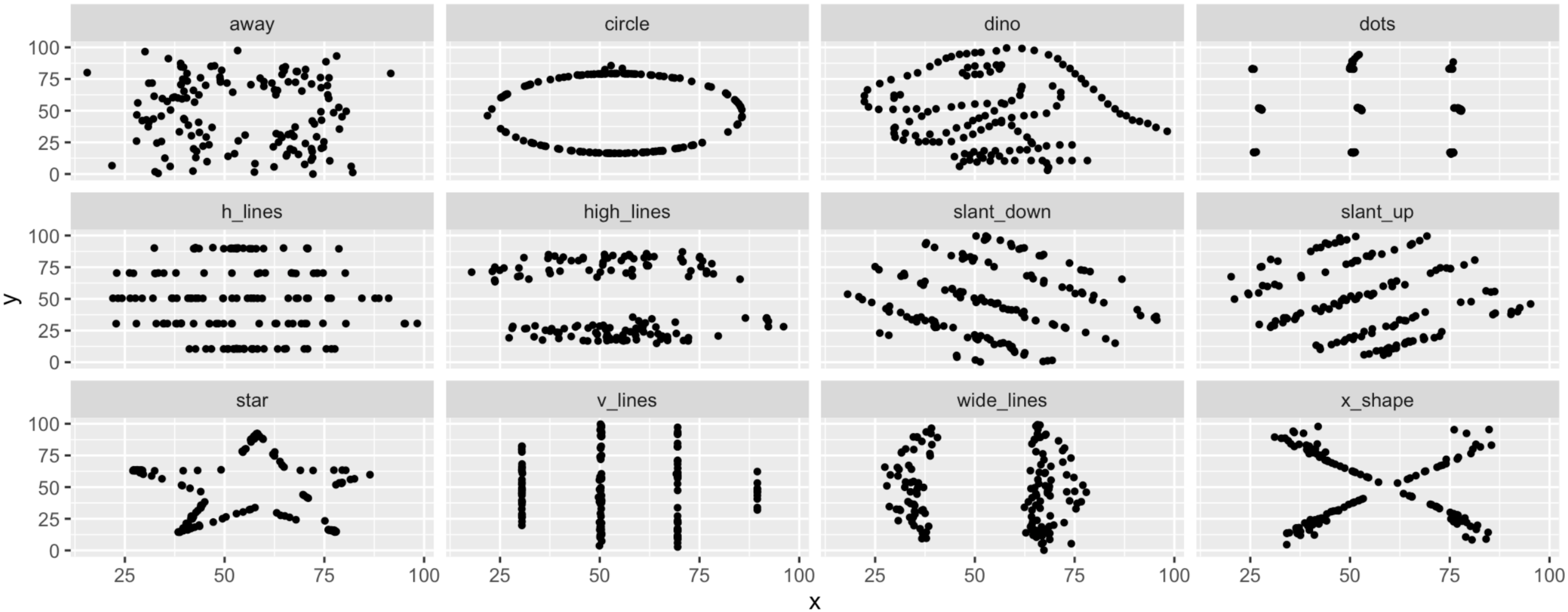
Why pay attention to data visualization?

- To explore data
- To diagnose data patterns & preliminary insights
- To display results and reinforce messages shared with readers of a publication

Believe it or not…

Each of the datasets depicted below (sets of x,y coordinates) has the same **mean**, **standard deviation**, **variance**, and **correlation!**

- x mean = 54.26 y mean = 47.83
- x sd = 16.76 y sd = 26.93
- correlation (x,y) = -0.06



Source: https://talks.andrewheiss.com/2021-seacen/02-ggplot2.html

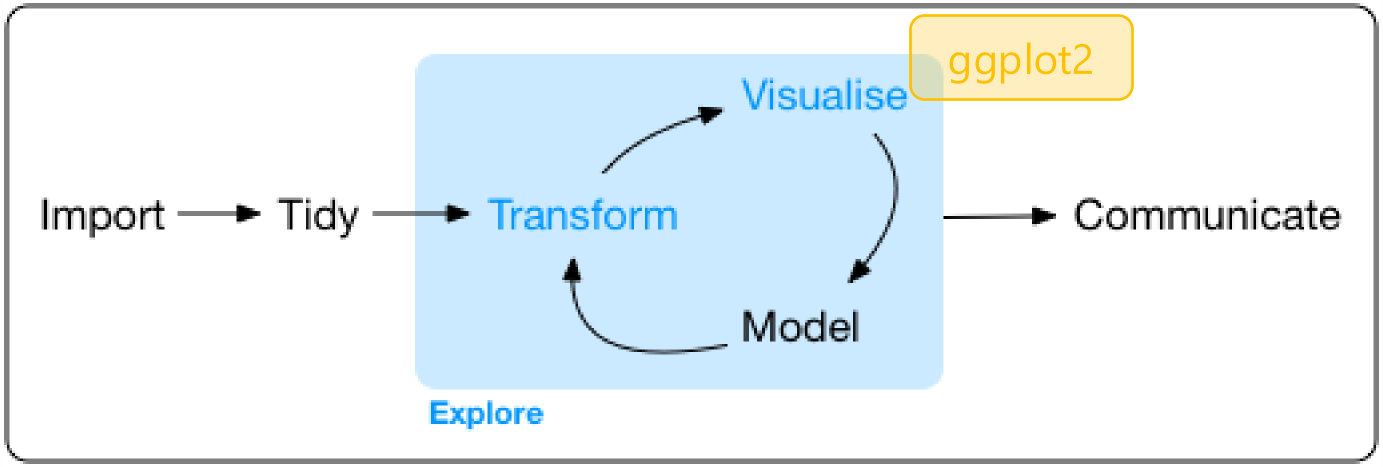
The {tidyverse} package {ggplot2} is very helpful for the visualization of data

ggplot2's conception of plot builds on some fundamental parts:

- data is a data frame
- aesthetics is used to indicate x and y variables. It can also be used to control the color, the size or the shape of points, the height of bars, etc.
- geometry defines the type of graphics (*histogram, box plot, line plot, density plot, dot plot*, ...)

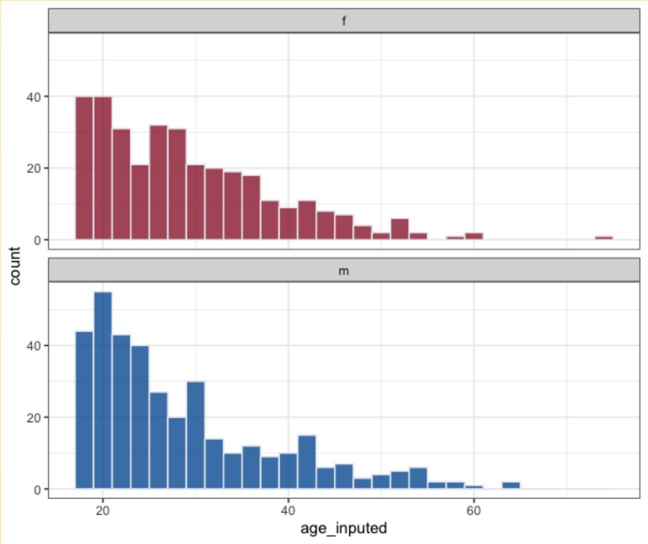
with the addition of:

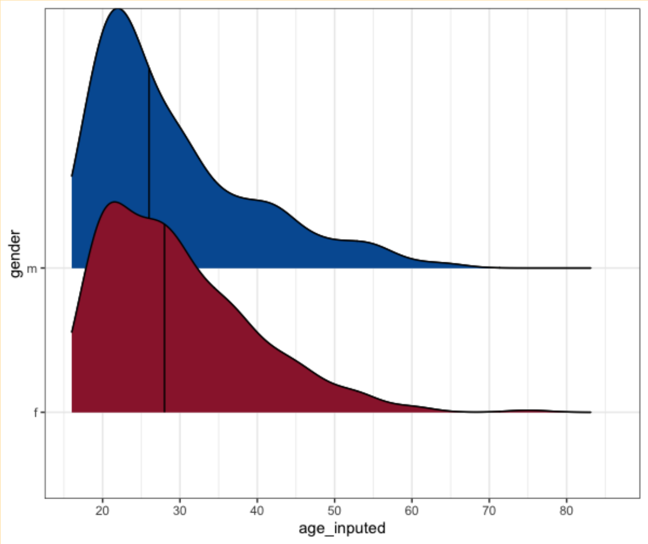
- statistics transform data on the way to visualization
- scales manipulate the labels, breaks, transformations and palettes
- facets to split the plot by a category
- coordinate systems to flip the graph
- labels for clarity
- theme for esthetic improvements

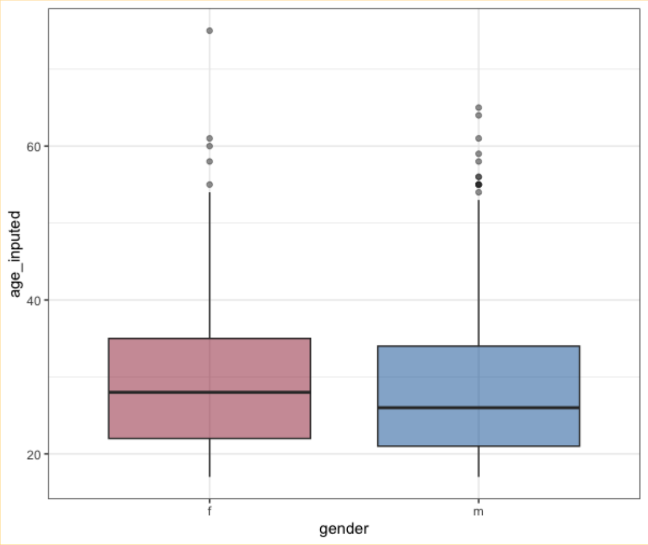


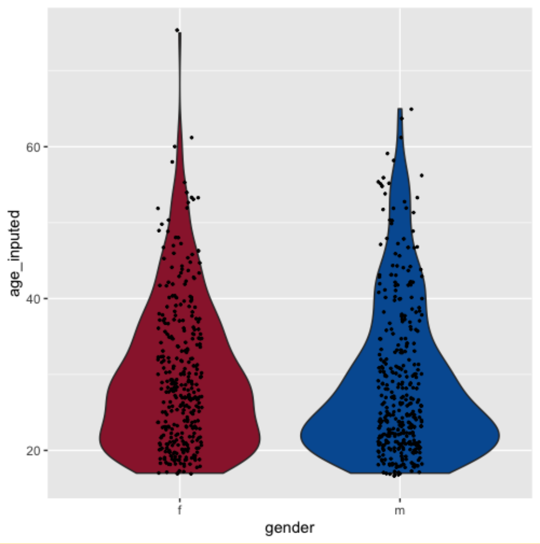
Source: https://www.tidyverse.org/blog/2023/08/teach-tidyverse-23/

Different ways to plot a frequency distribution of a continuous variable: histogram / density plot / boxplot / violin plot





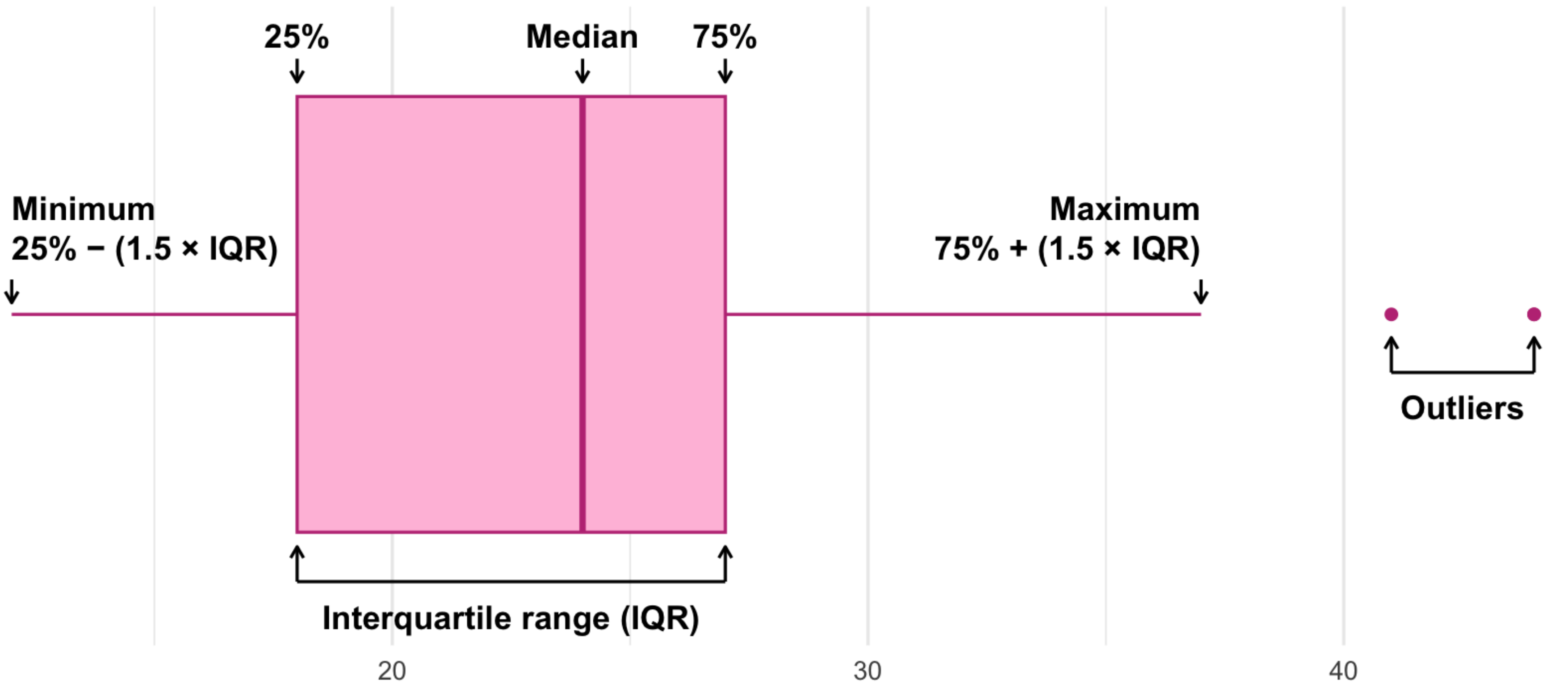




Source: autism dataset (see lab)

Anatomy of a boxplot

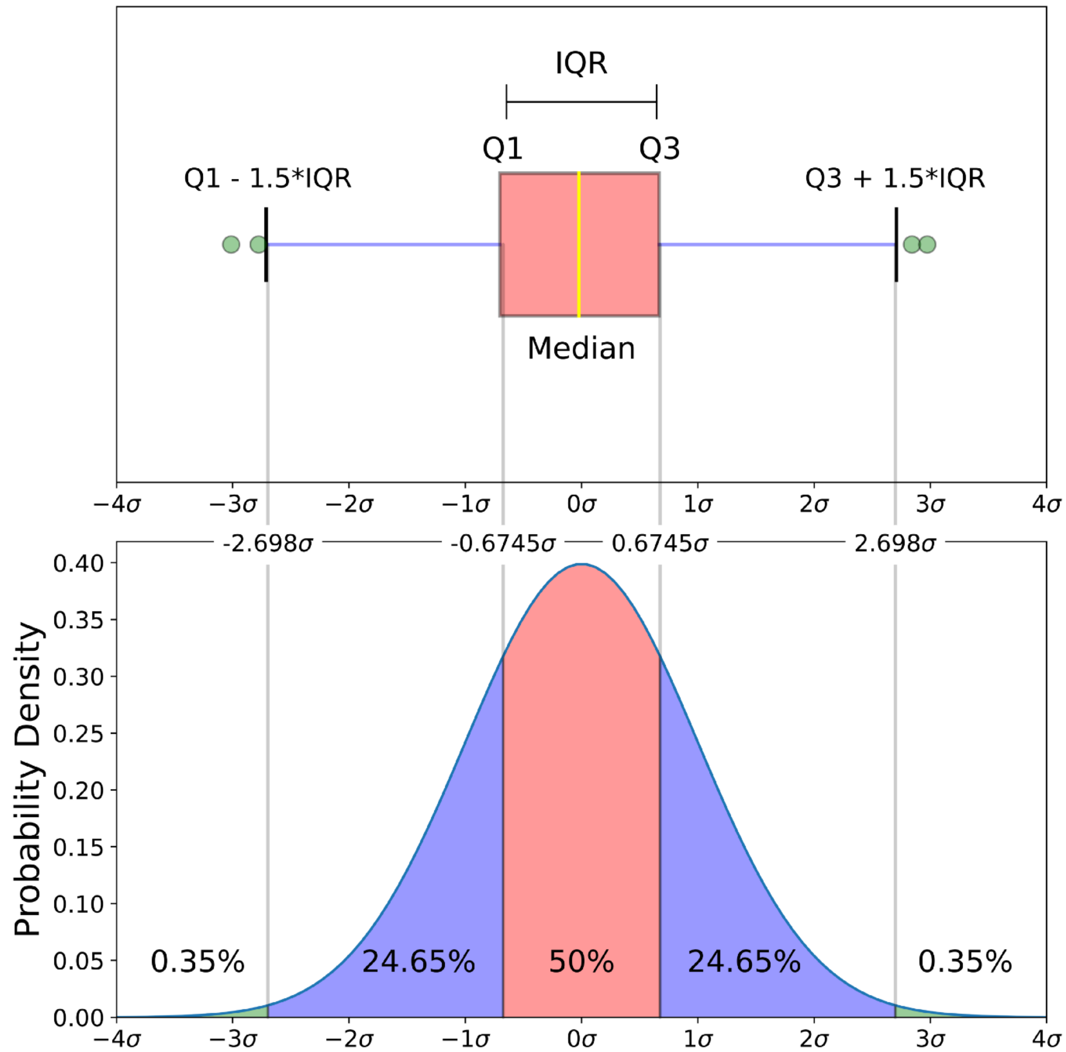
Box plots show the distribution of a variable by highlighting specific details, like the 25th, 50th (median) and 75th percentile, as well as the assumed minimum, assumed maximum, and outliers.



Source: https://datavizf23.classes.andrewheiss.com/lesson/06-lesson.html#boxes-violins-and-dots

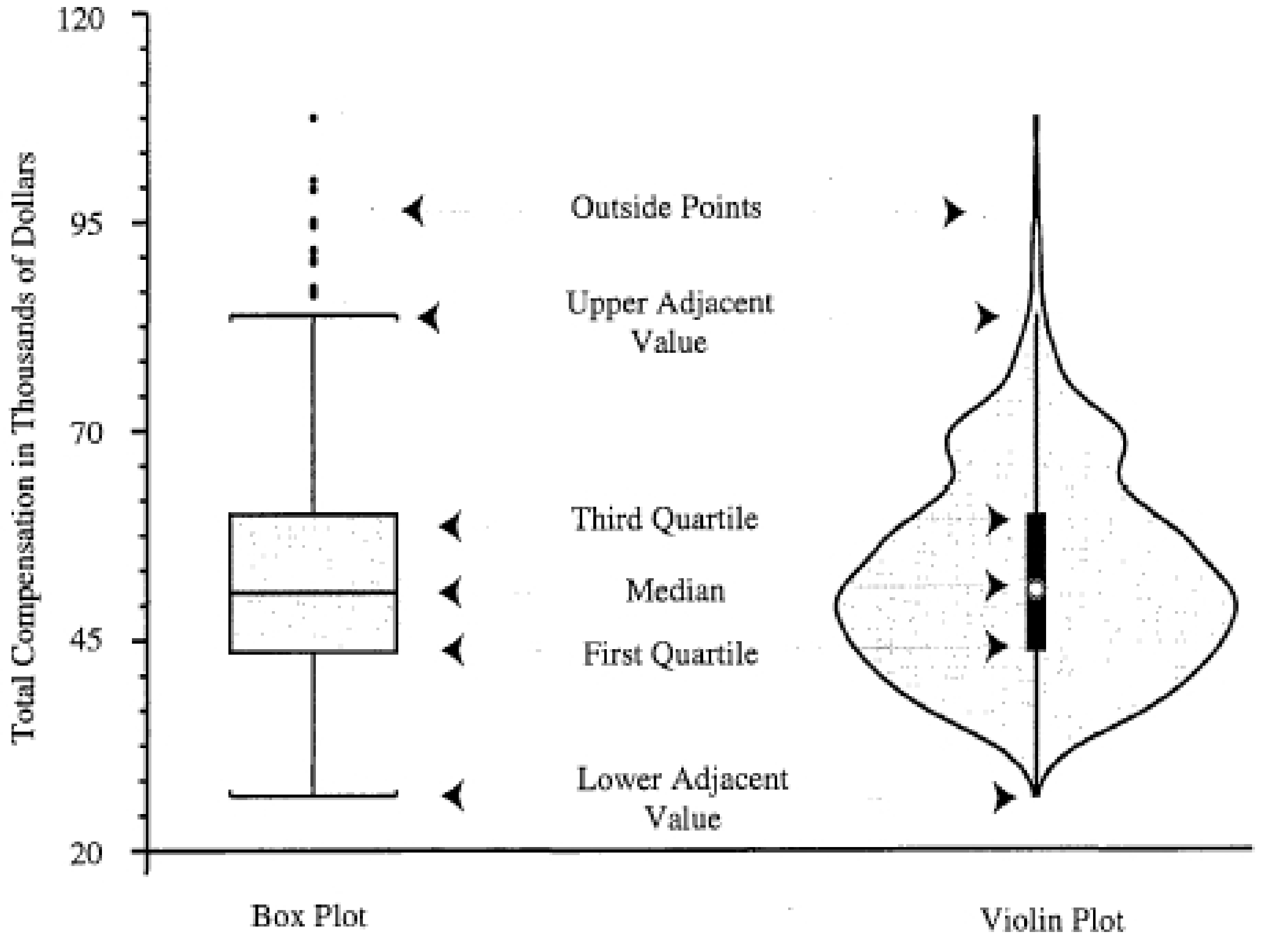
Boxplot (partitions) and distributions

Comparison of a boxplot of a nearly normal distribution and a probability density function (pdf) for a normal distribution

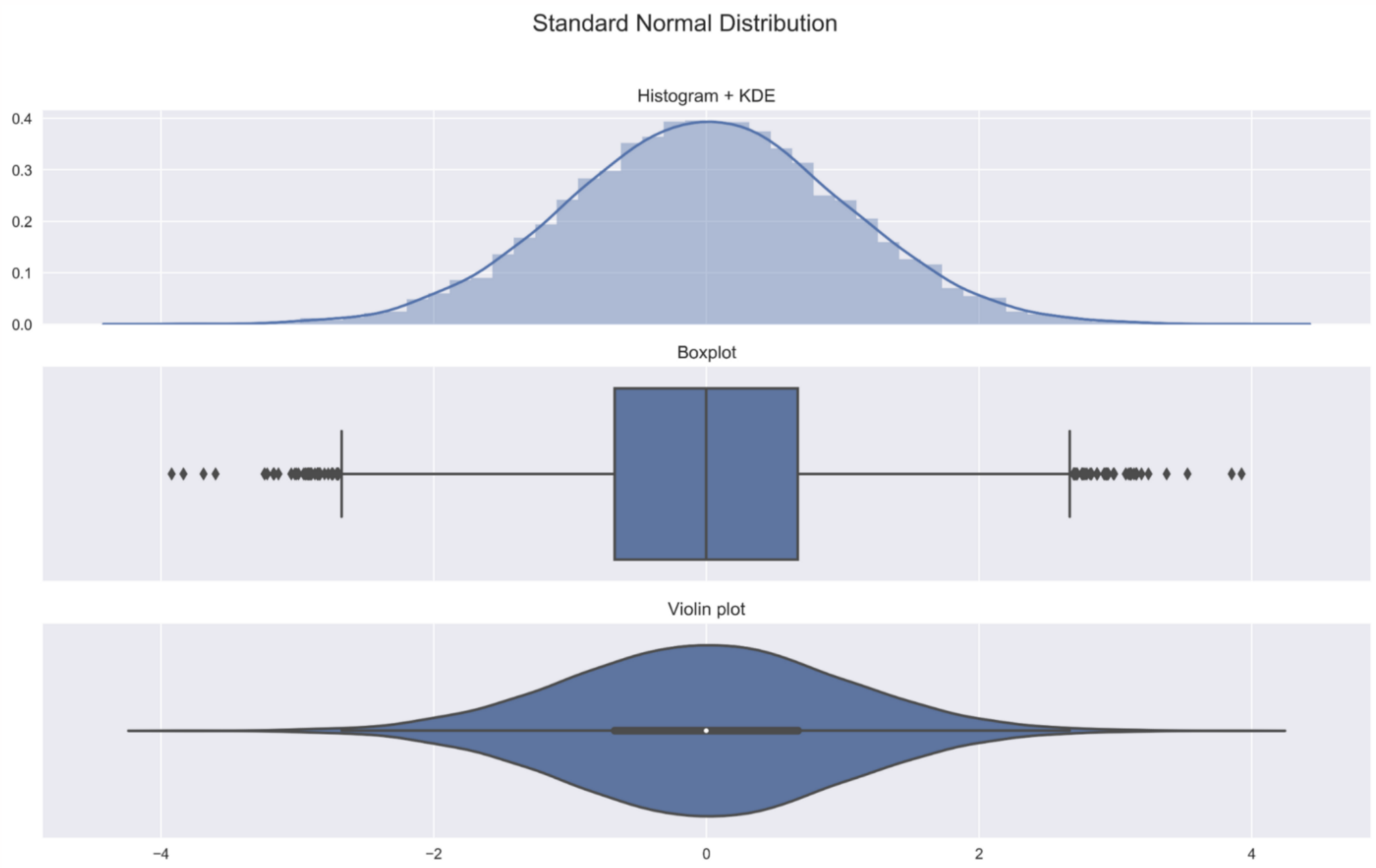


Boxplot and violin plots comparison

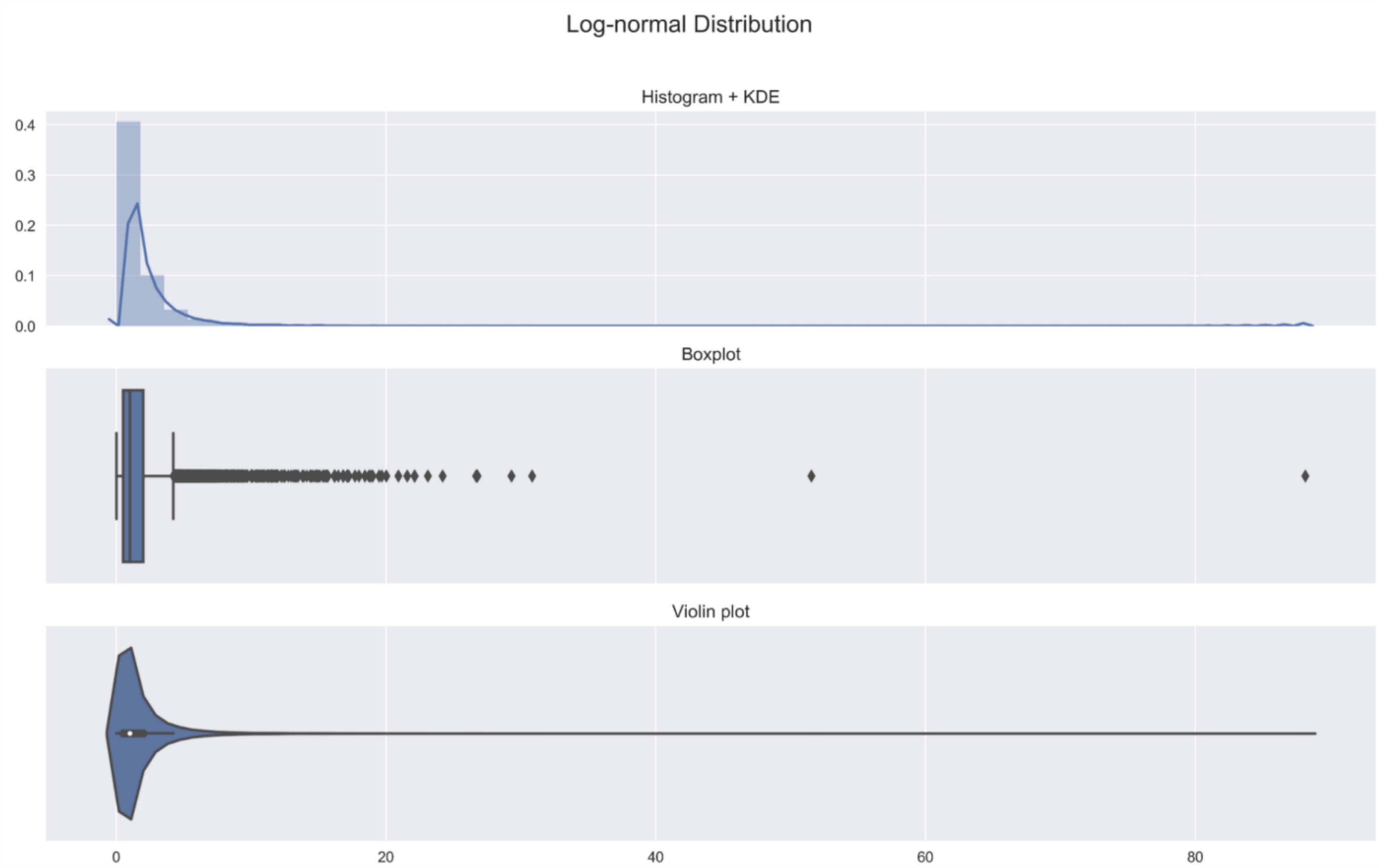
Common components of box plot and violin plot



Comparisons for a normal distribution 1/3



Comparisons for a log-normal distribution 2/3



Comparisons for a bimodal distribution 3/3