

# Processing and interpretation of neuroscience data:

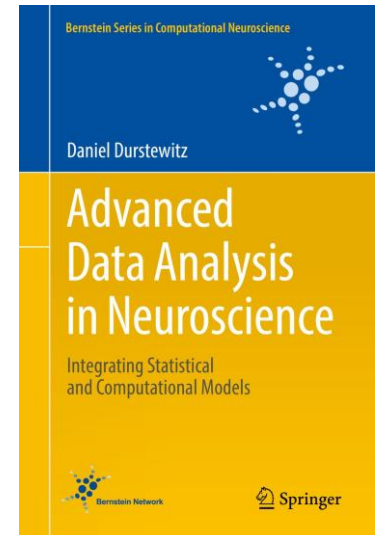
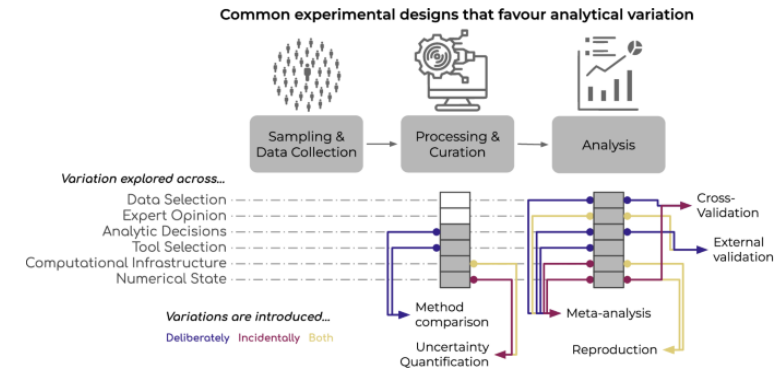
## Module 1 – Data overview and single cell sequencing

Dr. Kaja Moczulska & Dr. Łukasz Piszczek

Univ.-Prof. Dr. Wulf Haubensak  
Department of Neuronal Cell Biology

Center For Brain Research (Zentrum für Hirnforschung)

Module 2 – Data overview and single cell sequencing (18.11-09.12)  
Prof. Vered Kellner and Dr. Hugo Malagon-Vina



# Overview

## Resources needed:

- Laptop
- Internet access
- Materials:
  - VM machine
  - <https://github.com/HugoMalagon/NeuroData>
- [860.053-MUW](#)

## 21.10

- Introduction to R, basics
- Visual analytics

## 28.10

- dimensional reduction: PCA, UMAP
- Normalization/scaling
- clustering: k-means, knn

## 4.11

- Intro to Seurat
- Single cell RNA seq
- Dataset merging and preprocessing

## 11.11

- clustering in Seurat
- DEG interpretation

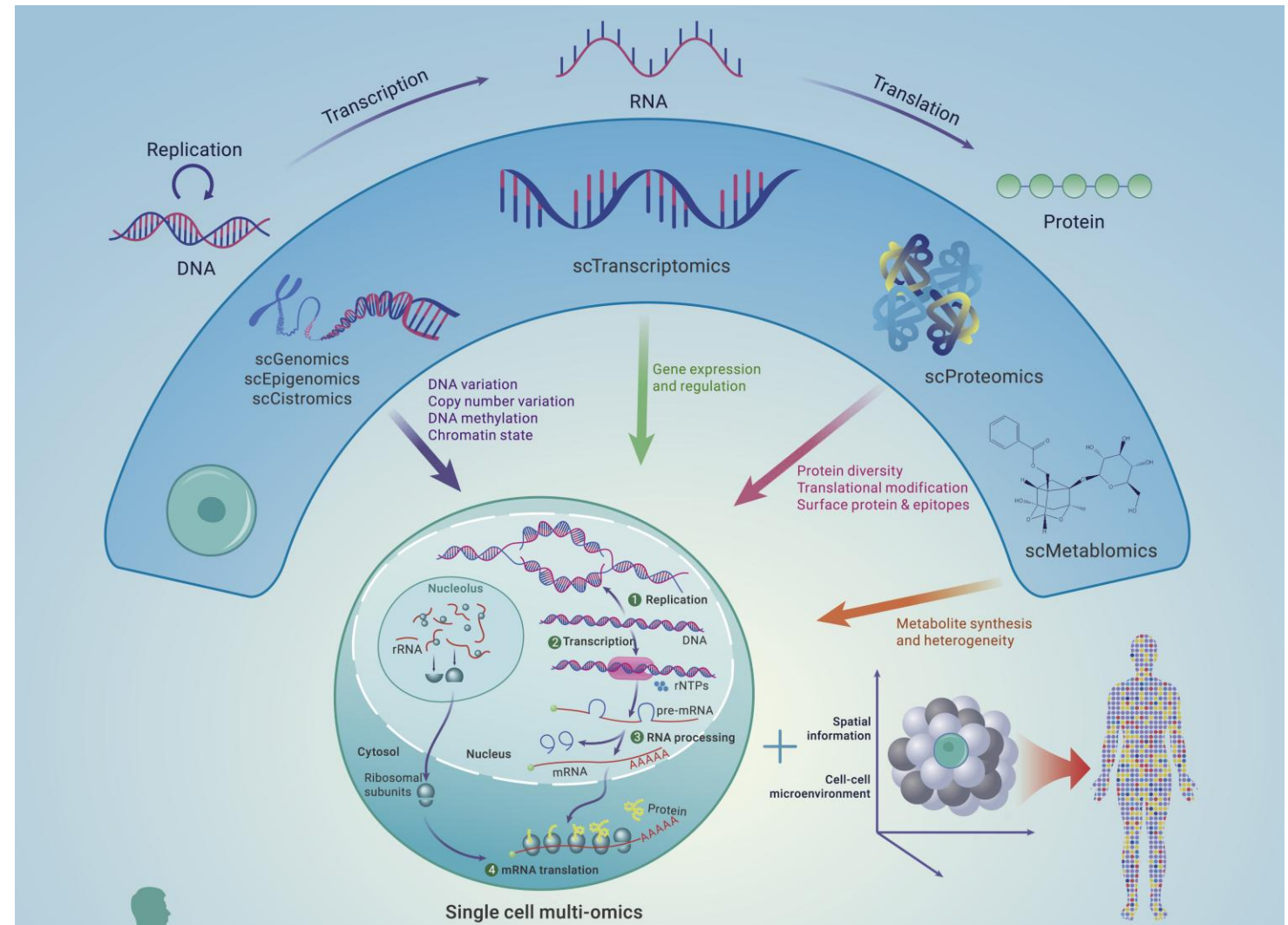
„Exam“: Assessment via email after each class

# Day 1

## Introduction to R Basics

# Why is single cell analysis interesting neuroscience?

- Neurogenomics and neurotranscriptomics are rapidly developing fields
- First sc transcriptome was published in 2009
- Single cell and spatial analysis techniques are constantly being developed and widely used in areas of biology
- Push to integrate multimodal data
- Valuable skillset: basics of data analysis tools and techniques are shared between different type of data
- Lot of data available: with the skillsets and ideas one can get new information out of already published available datasets



Wen et al, Innovation, 2022

# Introduction into R

Visualization

Open source

Data science



Platform agnostic


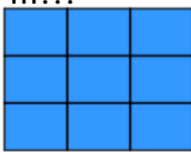

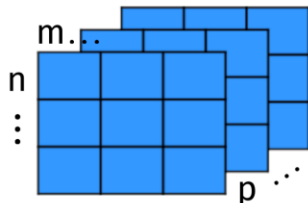
Computational  
statistics



**R:**

- is a GNU Project and licensed under the GNU General Public License.
- It is written primarily in C, Fortran, and R itself.
- Precompiled executables are provided for various operating systems.
- As an interpreted language, R has a native command line interface.

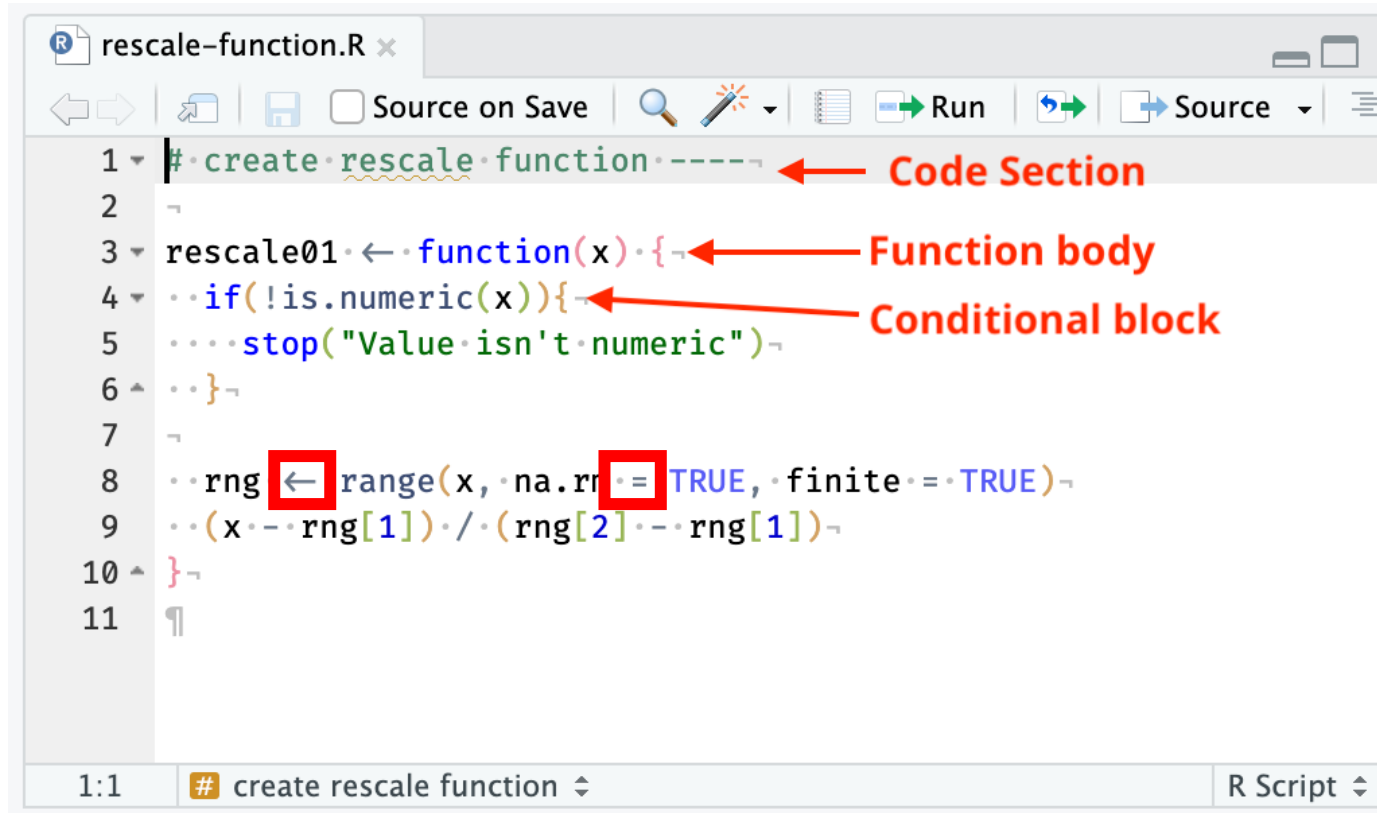
# Introduction into R: Data structures

	Dimensions	Mode (data "type")	Example
<b>Vector</b>	1 	Identical	<code>c(10,0.2,34,48,53)</code>
<b>Matrix</b>		Identical	<code>matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3)</code>
<b>Data frame</b>		Can be different	<code>data.frame(x = 1:3, y = 5:7)</code>
<b>Array</b>		Identical	<code>array(data = 1:3, dim = c(2,4,2))</code>
<b>List</b>	$\left\{ \begin{array}{l} \text{Vector} \\ \text{Matrix} \\ \text{Data frame} \\ \text{Array} \end{array} \right\}$	Can be different	<code>list(x = cars[,1], y = cars[,2])</code>

See Rstudio IDE cheatsheat

<https://r.qcbs.ca/workshop01/pres-en/workshop01-pres-en.html#1>

# Introduction into R: basic code structure



The screenshot shows an R script editor window titled 'rescale-function.R'. The script contains the following code:

```
1 # create rescale function -----
2
3 rescale01 <- function(x) {
4   if(!is.numeric(x)) {
5     stop("Value isn't numeric")
6   }
7
8   rng <- range(x, na.rm = TRUE, finite = TRUE)
9   (x - rng[1]) / (rng[2] - rng[1])
10 }
11
```

Annotations with red arrows point to specific parts of the code:

- Code Section**: Points to the comment line 1.
- Function body**: Points to the opening curly brace of the function definition on line 3.
- Conditional block**: Points to the `if` statement on line 4.

Red boxes highlight the assignment operators `<-` on lines 3 and 8.

Conventional  
assignment  
types

## Practical example

```
> mean(x=1:10)
[1] 5.5
> x
Error: object 'x' not found

> mean(x<-1:10)
[1] 5.5
> x
[1] 1 2 3 4 5 6 7 8 9 10
```

<https://docs.posit.co/ide/user/ide/guide/code/code-sections.html>

# Introduction into R

Operator	Primary Use	Scope/Behavior	Community Convention
<code>&lt;-</code>	Assignment	Assigns in current environment	Preferred for assignment
<code>=</code>	Assignment, Arguments	Assigns or sets function argument	Preferred for arguments only

S and APL had its own keyboards

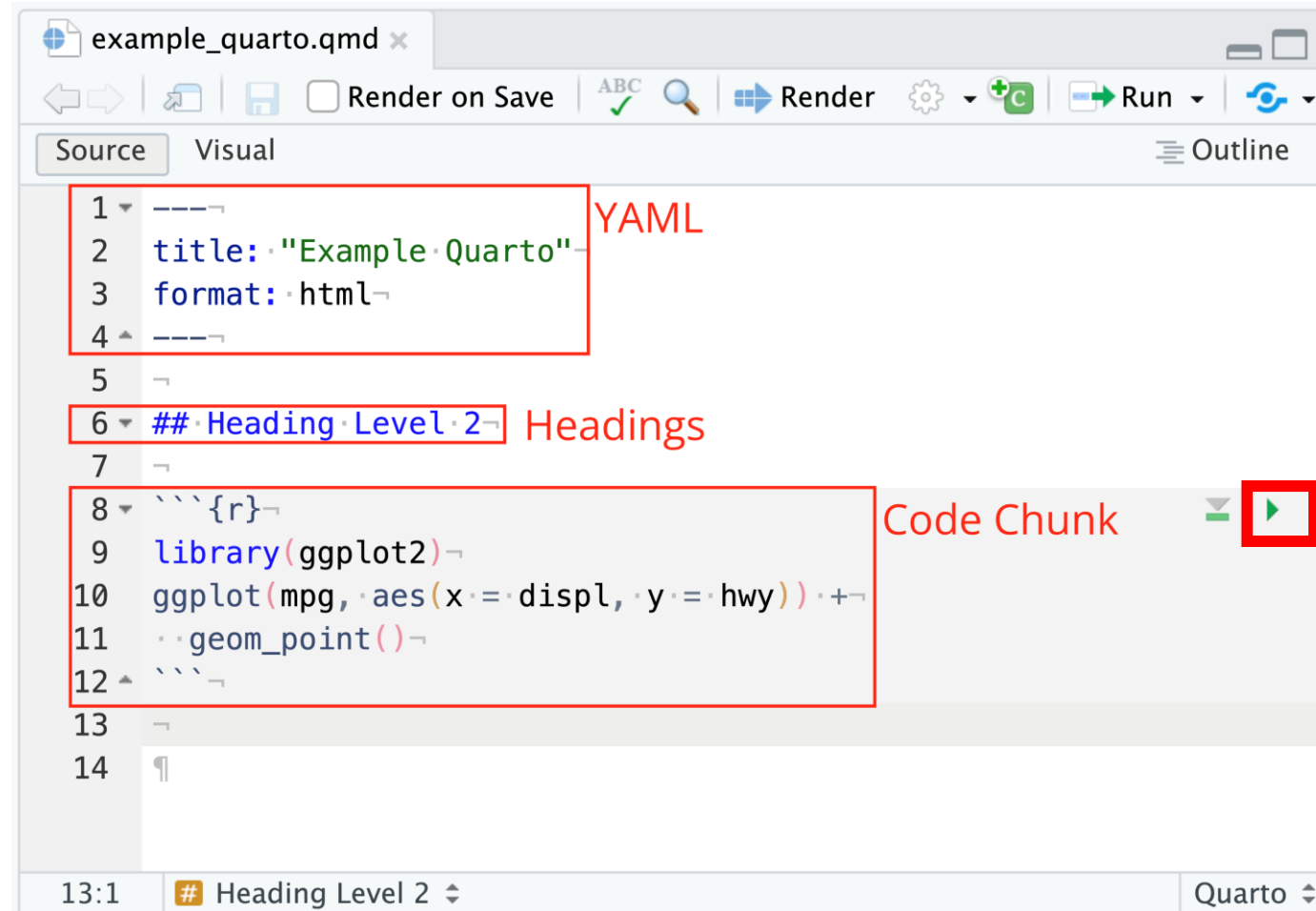


S language didn't have `==` for equality testing, so that was left to the single equal sign



# Introduction into R: basic code structure

YAML is a human-readable and easy to write language to define data structures. Rmd makes it possible to use a YAML header to specify certain parameters right at the beginning of the document. Built-in YAML parameters make it easier to create more organized and informative reports.



The screenshot shows the Quarto IDE interface with a file named 'example\_quarto.qmd'. The editor is in 'Source' mode. The document content is as follows:

```
1 ---
2 title: "Example Quarto"
3 format: html
4 ---
5
6 ## Heading Level 2
7
8 ```{r}
9 library(ggplot2)
10 ggplot(mpg, aes(x = displ, y = hwy)) +
11   geom_point()
12 ```
13
14
```

Annotations in the image:

- A red box highlights lines 1-4, labeled 'YAML'.
- A red box highlights line 6, labeled 'Headings'.
- A red box highlights lines 8-12, labeled 'Code Chunk'.
- A red box highlights the 'Run' button (a green play icon) on the right side of the code chunk, with the text 'Run Code Chunk' next to it.

The status bar at the bottom shows '13:1' and '# Heading Level 2'.

Run Code Chunk

<https://docs.posit.co/ide/user/ide/guide/code/code-sections.html>

# Introduction into R: RStudio layout

The screenshot displays the RStudio interface with four main panes:

- Script (red border):** Contains R code for creating a ggplot2 scatter plot. The code is:

```
1 library(ggplot2)
2 ggplot(mpg, aes(x = displ, y = hwy)) +
3   geom_point(aes(color = class))
4 mpg
5
```
- Environment (blue border):** Shows the Global Environment, which is currently empty.
- Console (yellow border):** Shows the execution of the R code, resulting in a tibble of 234 rows and 11 columns. The first 10 rows are displayed:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
	<chr>	<chr>	<dbl>	<int>	<int>	<chr>	<chr>	<int>	<int>	<chr>	<chr>
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
- Output (purple border):** Displays a scatter plot of highway mileage (hwy) versus engine displacement (displ), colored by car class. The legend indicates the following classes: 2seater (red), compact (yellow), midsize (green), minivan (teal), pickup (blue), subcompact (purple), and suv (pink).

# Introduction into R: RStudio layout

## Script files

- Saves your script
- Allows code & comments
- Can have multiple files open at a time

## Console/Command line

- Can use as calculator
- Does not save code
- This is where your output is displayed

The screenshot displays the RStudio interface with the following components:

- Script editor (top left):** Contains R code for a function `block_summary`. The code is as follows:

```
231 }
232 }
233
234
235 # Provides the summary for the block containing units m through n, n > m
236 # t = time for firsts unit
237 # m = lower bound unit of production block
238 # n = upper bound unit of production block
239 # r = learning curve rate
240 block_summary <- function(t, m, n, r, na.rm = FALSE){
241
242   if(!is.numeric(t) | !is.numeric(m) | !is.numeric(n) | !is.numeric(r)){
243     stop('This function only works for numeric inputs!\n',
244         'You have provided objects of the following classes:\n',
245         't: ', class(t), '\n',
246         'm: ', class(m), '\n',
247         'n: ', class(n), '\n',
248         'r: ', class(r))
249   }
250 }
```
- Environment pane (top right):** Shows the workspace environment with the following values:

Variable	Value
m	10
n	100
r	0.77
t	5
- Console (bottom left):** Displays the output of the `block_summary` function calls:

```
$`block hours`
[1] 3668.436

$`midpoint unit`
[1] 44.03189

$`midpoint hours`
[1] 40.31249

> ?sum
> m + n
[1] 110
> m + n * t^r
[1] 355.3082
> |
```
- Viewer pane (bottom right):** Shows the R documentation for the `mean` function, including the title "Arithmetic Mean", a description, usage examples, and arguments.

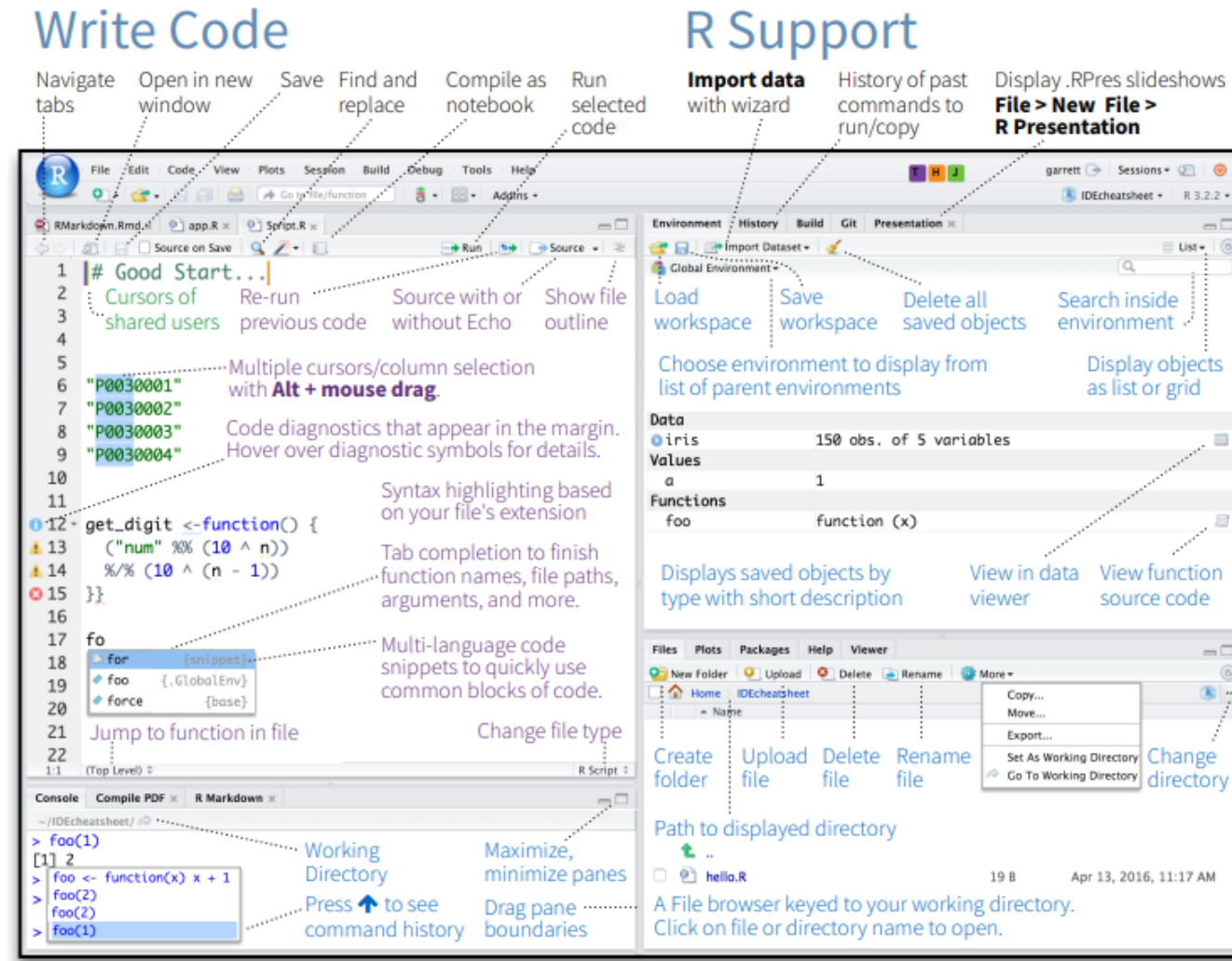
## Workspace environment

- Holds your objects
- Can review history

## Misc - Displays:

- files in working directory
- plots when produced
- help files/search

# Introduction into R: RStudio layout



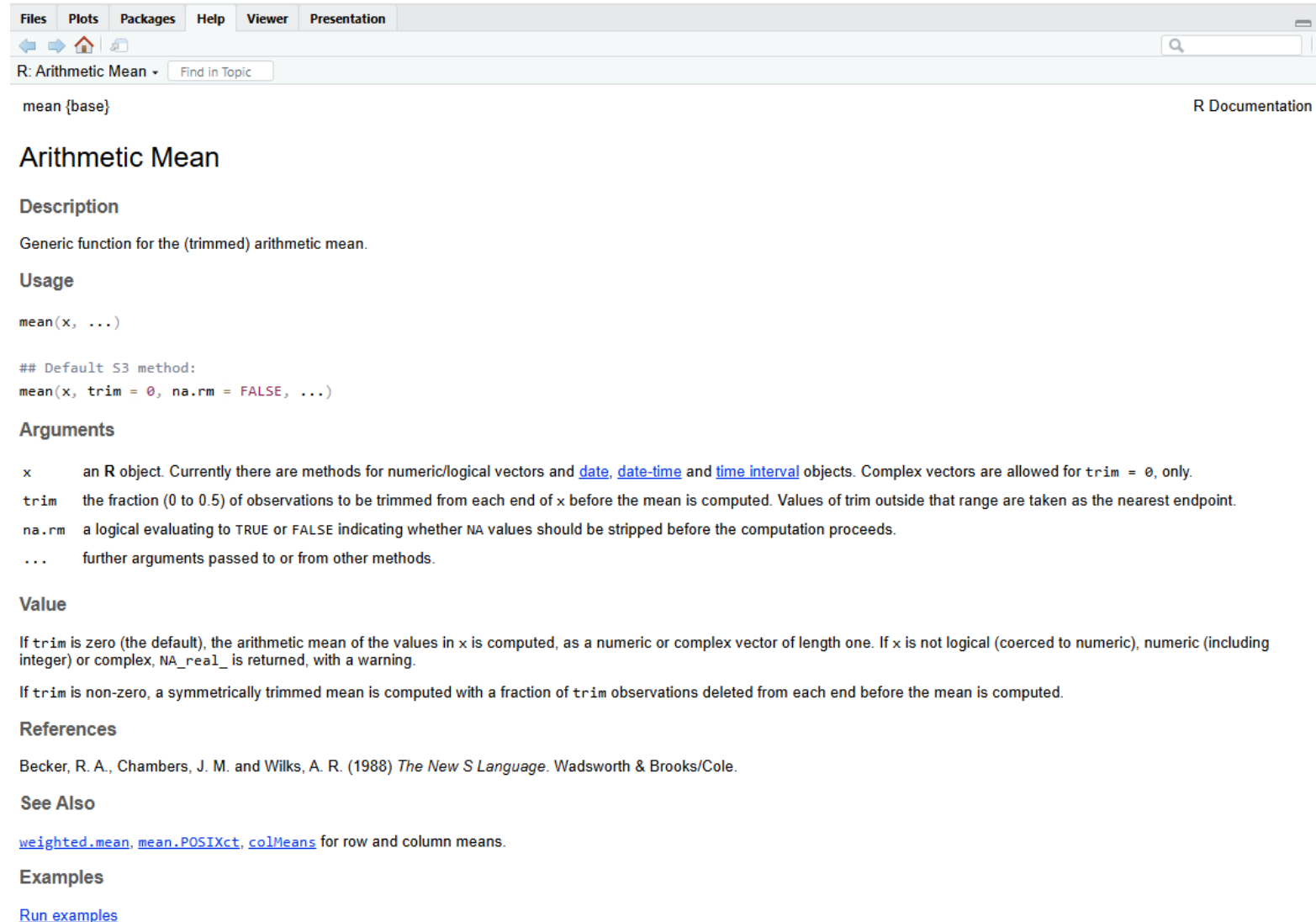
See Rstudio IDE cheatsheet

# Introduction into R: help

Not sure what a function is doing or how to use it?

Use `?function_name` or `help(function_name)`

```
> ?mean  
>
```



The screenshot shows the R Documentation page for the `mean` function. The page has a navigation bar at the top with tabs for Files, Plots, Packages, Help, Viewer, and Presentation. Below the navigation bar is a search bar and a "Find in Topic" button. The main content area is titled "Arithmetic Mean" and includes a "Description" section stating it is a generic function for the (trimmed) arithmetic mean. The "Usage" section shows the function signature `mean(x, ...)`. The "Arguments" section lists the parameters: `x` (an R object), `trim` (the fraction of observations to be trimmed), and `na.rm` (a logical evaluating to TRUE or FALSE). The "Value" section explains that the function returns the arithmetic mean of the values in `x`. The "References" section cites Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. The "See Also" section lists `weighted.mean`, `mean.POSIXct`, and `colMeans`. The "Examples" section includes a link to "Run examples".

Files Plots Packages Help Viewer Presentation

R: Arithmetic Mean Find in Topic

mean {base}

R Documentation

## Arithmetic Mean

### Description

Generic function for the (trimmed) arithmetic mean.

### Usage

```
mean(x, ...)
```

## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)

### Arguments

`x` an R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

`trim` the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

`na.rm` a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.

... further arguments passed to or from other methods.

### Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

### References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

### See Also

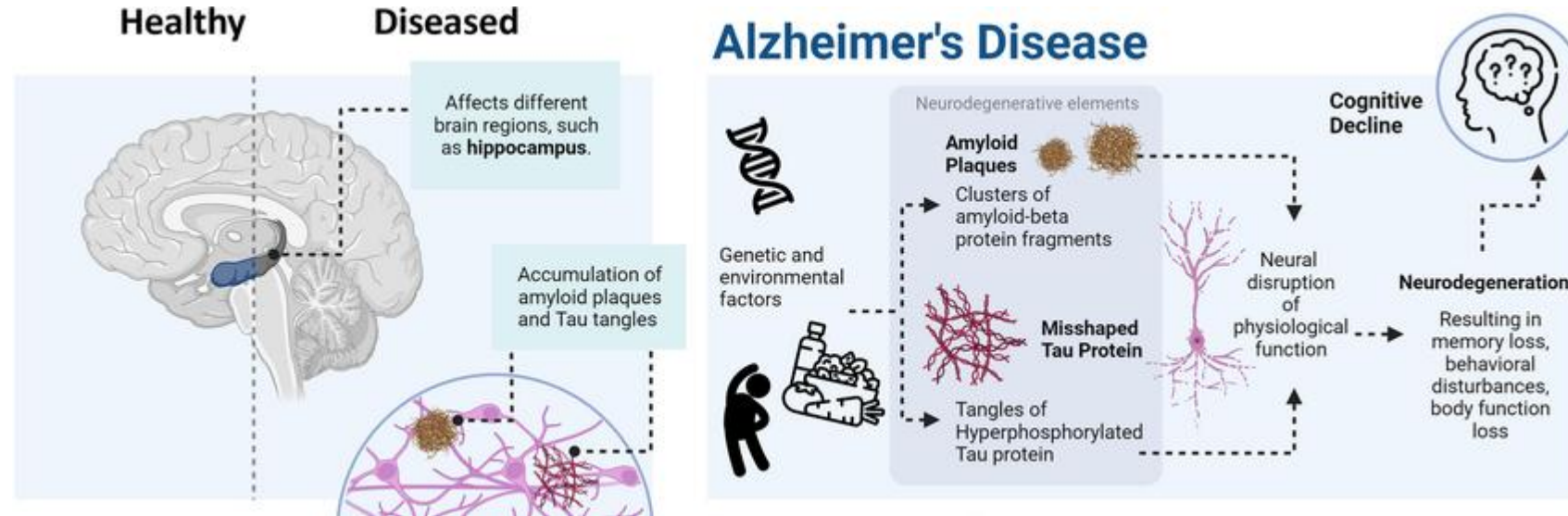
[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

### Examples

[Run examples](#)

See Rstudio IDE cheatsheet

# Alzheimer's disease datasets for this module



Balestri et al., J Neuoinfl 2024

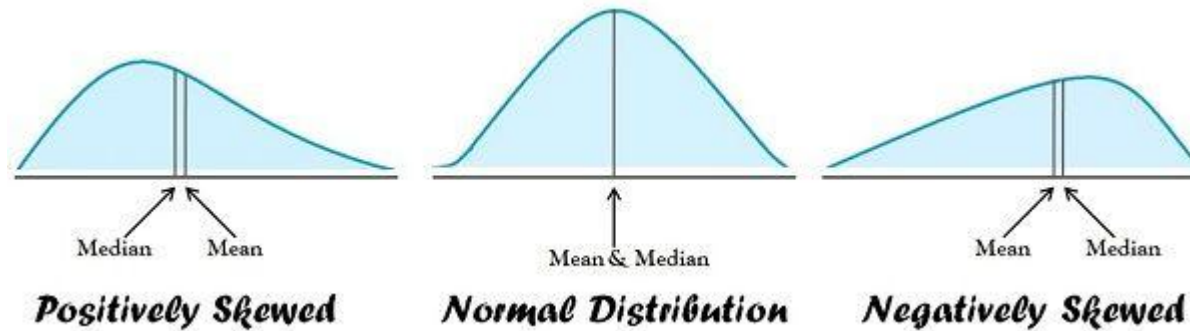
- Meeting 1 and 2: human patient data
  - Microarray
  - Selected genes expression, 3 brain regions, 2 sexes, control and AD patients
  - <https://europepmc.org/article/MED/23595620>
- Meeting 3 and 4: preclinical mouse model (ApoE KO) data
  - Single cell RNA seq
  - All detected genes, hippocampus, control and AD mice
  - <https://www.sciencedirect.com/science/article/pii/S0168010221002182?via%3Dihub>

# Look at the data

- Get some info with what you work on. Useful 1<sup>st</sup> glimpse
  - `dim(df)` -> dimensions of data frame
  - `nrows()` -> number of rows
  - `ncols()` -> number of columns
  - `head(df, 20)` -> first 20 entries
  - `tail(df, 20)` -> last 20 entries
  - ...
- Libraries
  - `dlookr`
  - `skimr`
  - ...



# Basic statistics



Normal distribution: test means  
t-test  
Skewed distribution: test medians  
Wilcoxon rank sum test

Useful functions in R:

`mean(x)` – calculates arithmetic mean

`median(x)` – calculates median

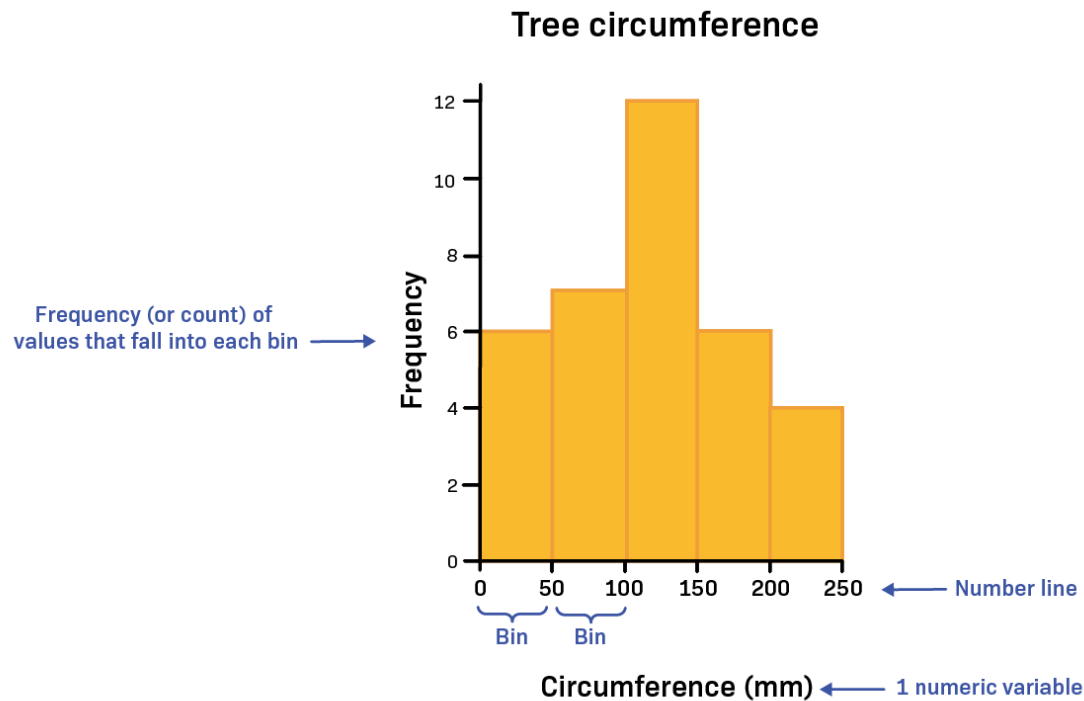
`ttest(x, y)` – calculates t-test (of the means)

`wilcox.test()` – calculates Wilcoxin test (of the medians)

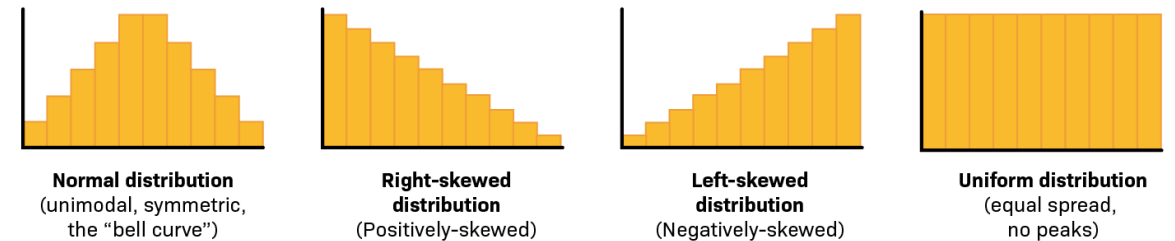


# How to check if your distribution is normal

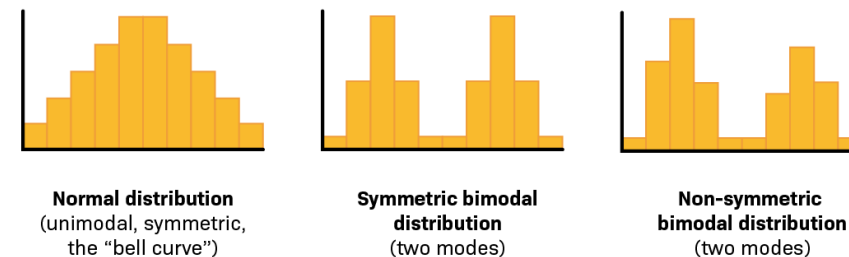
- Statistics (ex. shapiro.test())
- Visual (histogram)



Symmetric (normal) vs skewed and uniform distributions

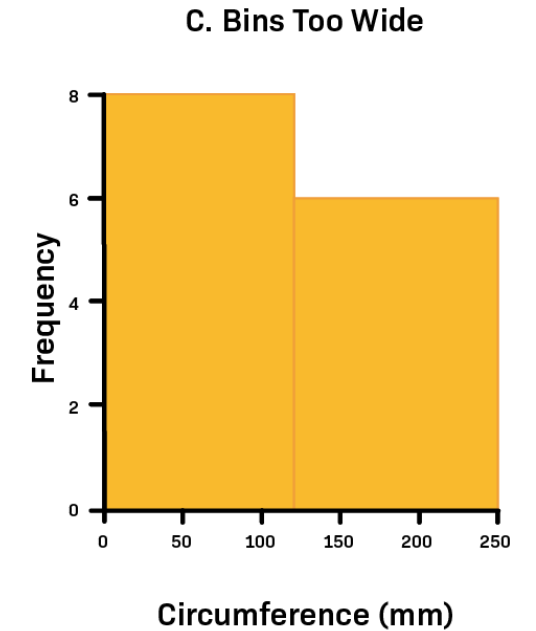
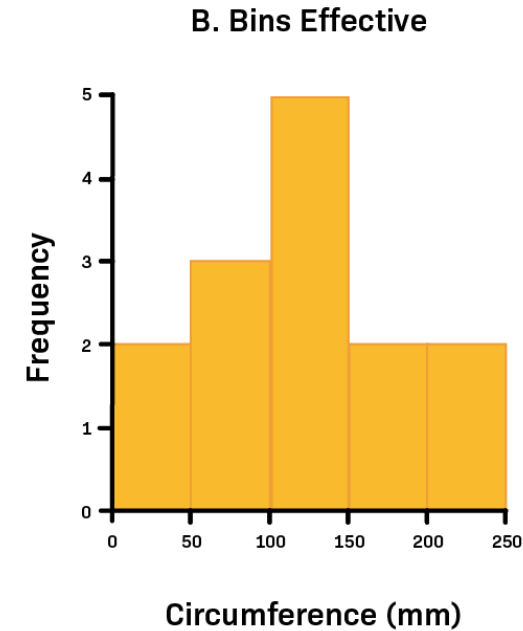
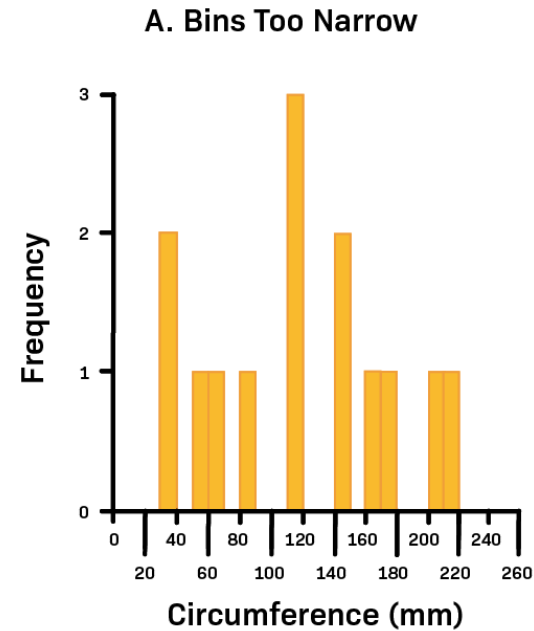
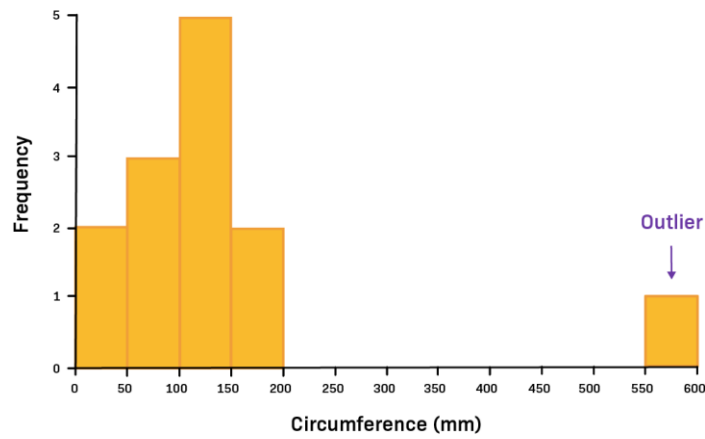


Unimodal vs bimodal distributions



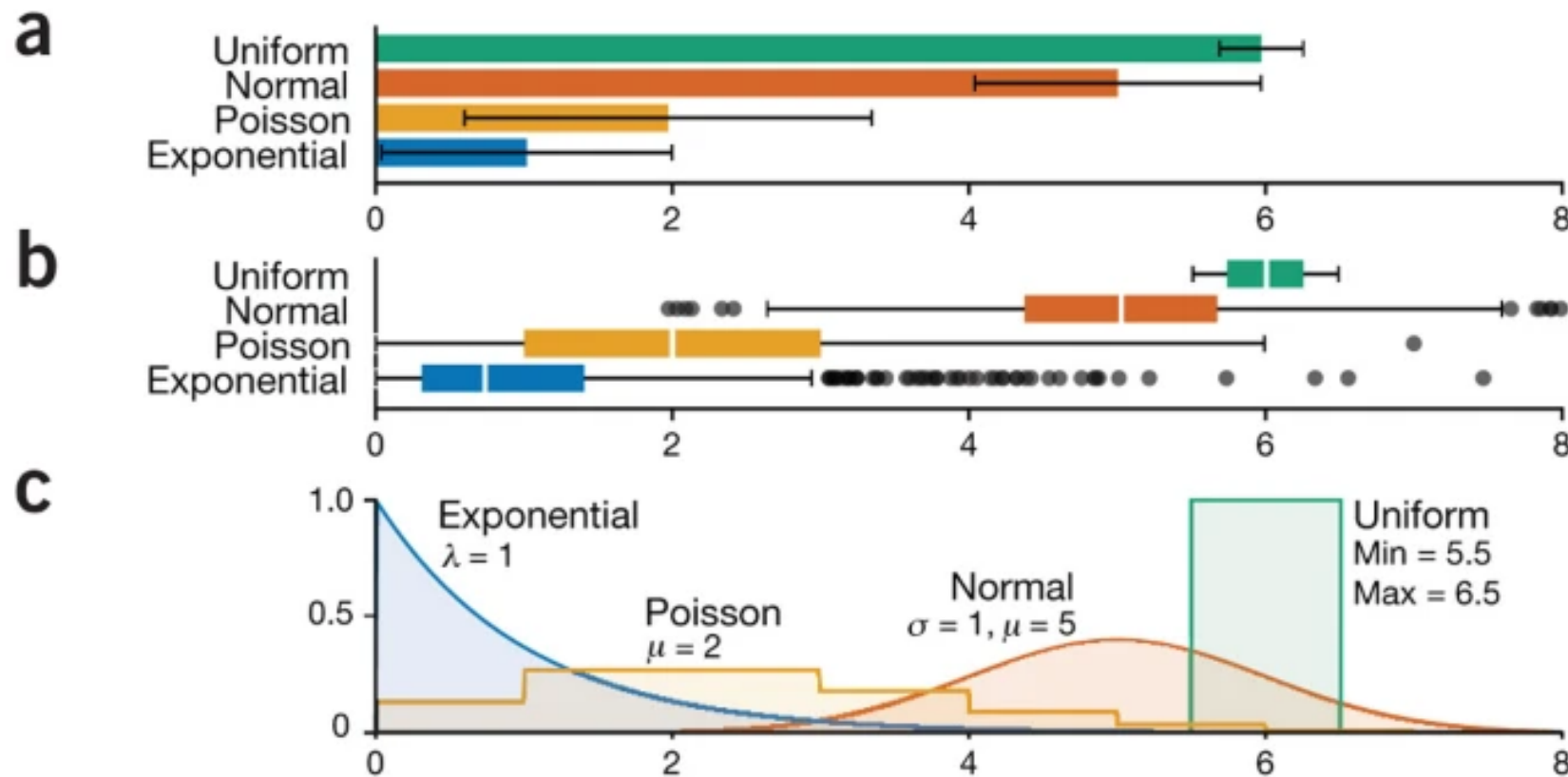
<https://www.labxchange.org/library/items/lb:LabXchange:10d3270e:html:1>

# Histograms



<https://www.labxchange.org/library/items/lb:LabXchange:10d3270e:html:1>

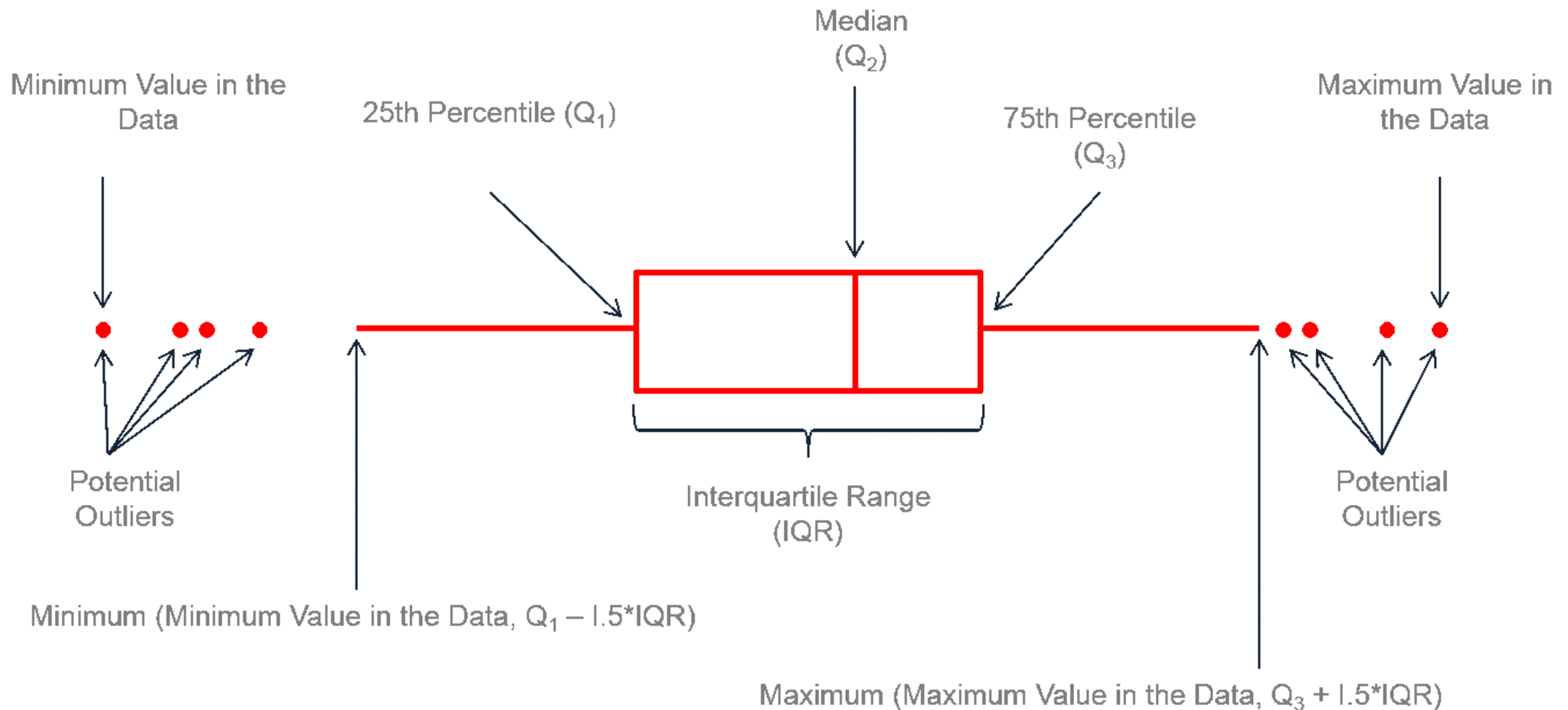
# Bar plot vs box plot



- (a) Bar chart showing sample means ( $n = 1,000$ ) with standard-deviation error bars.
- (b) Box plot ( $n = 1,000$ ) with whiskers extending to  $\pm 1.5 \times \text{IQR}$ .
- (c) Probability density functions of the distributions in **a** and **b**.  $\lambda$ , rate;  $\mu$ , mean;  $\sigma$ , standard deviation.

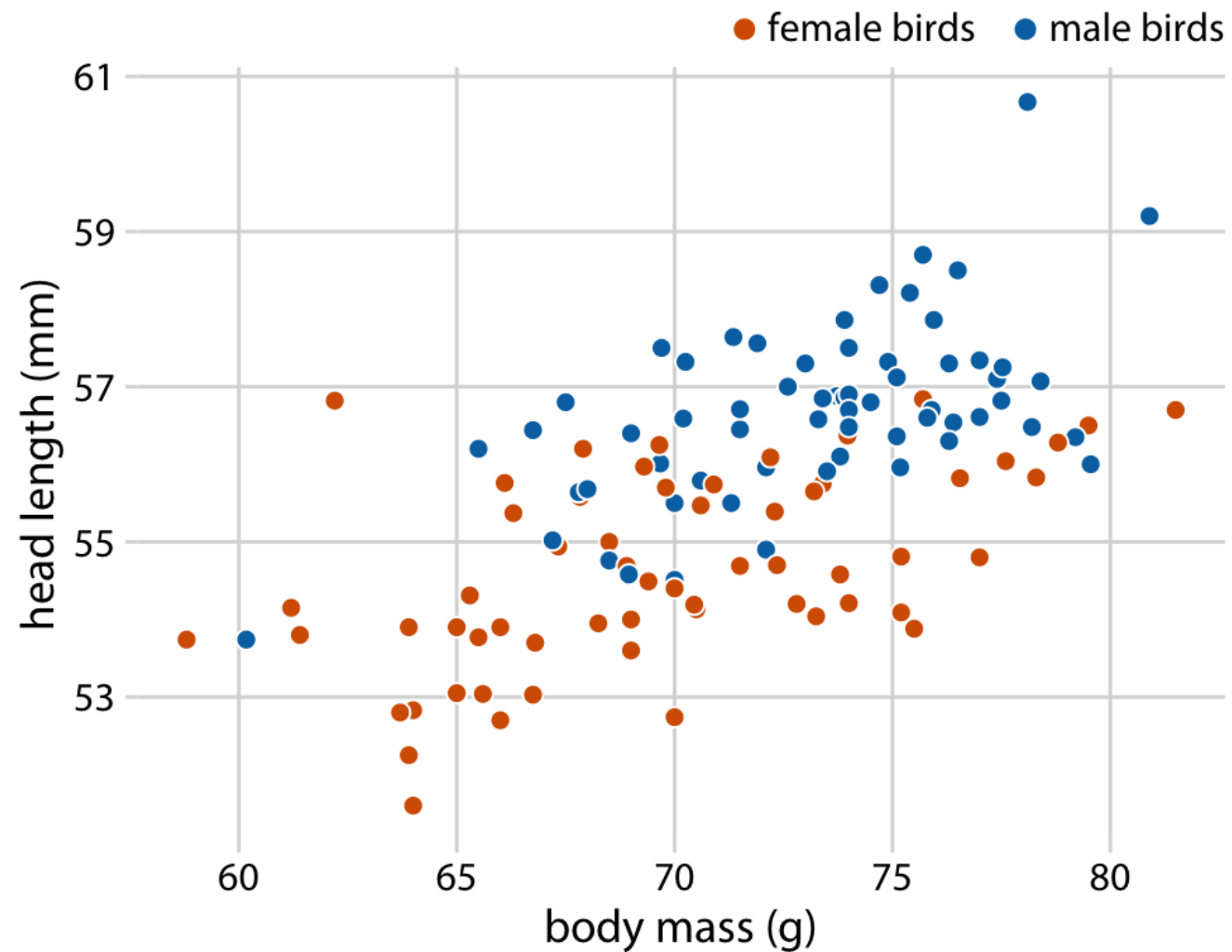
<https://www.nature.com/articles/nmeth.2807>

# Box plot anatomy



<https://www.data-to-viz.com/caveat/boxplot.html>

# Scatter plot, correlation

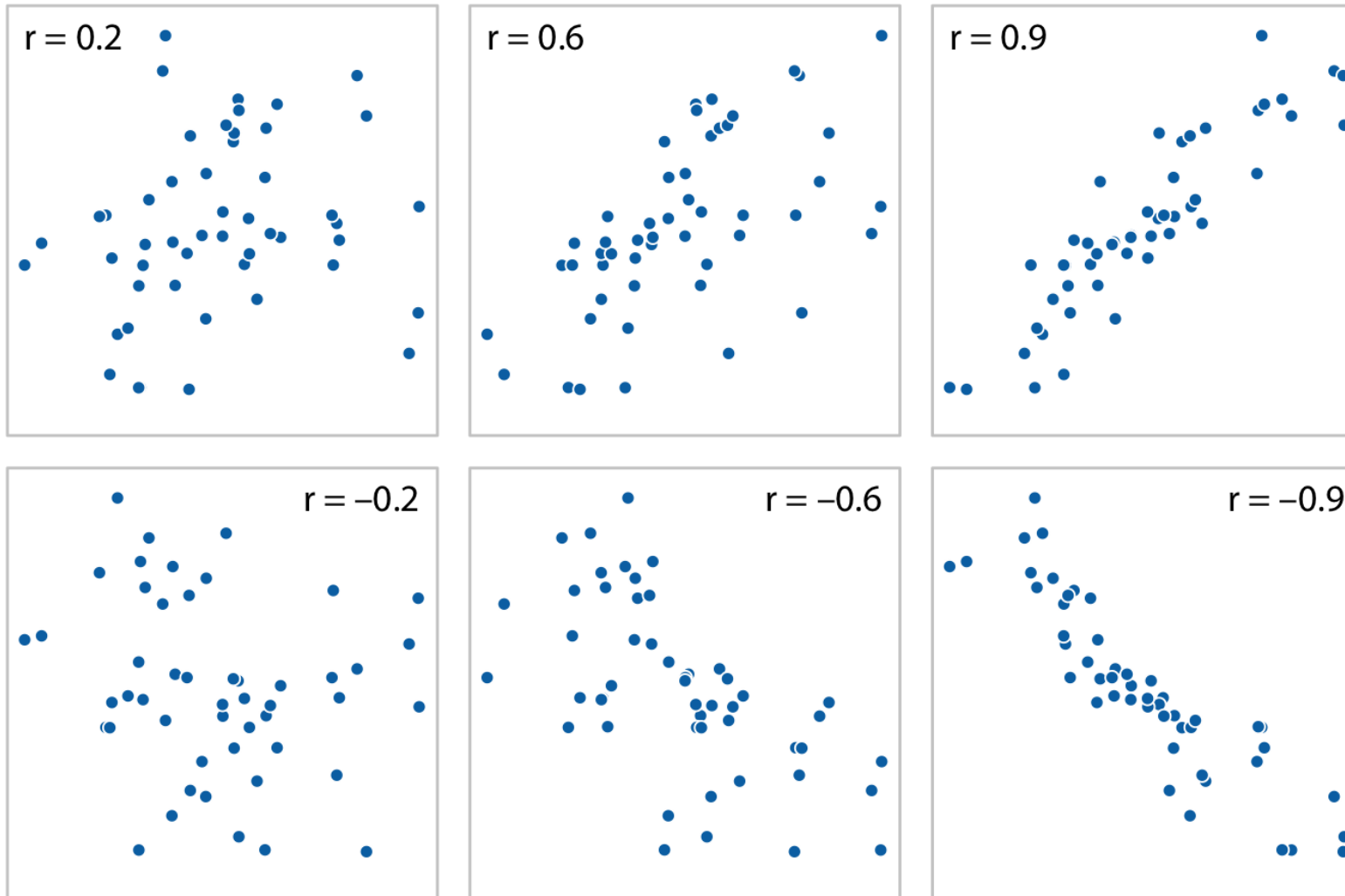


Blue jays



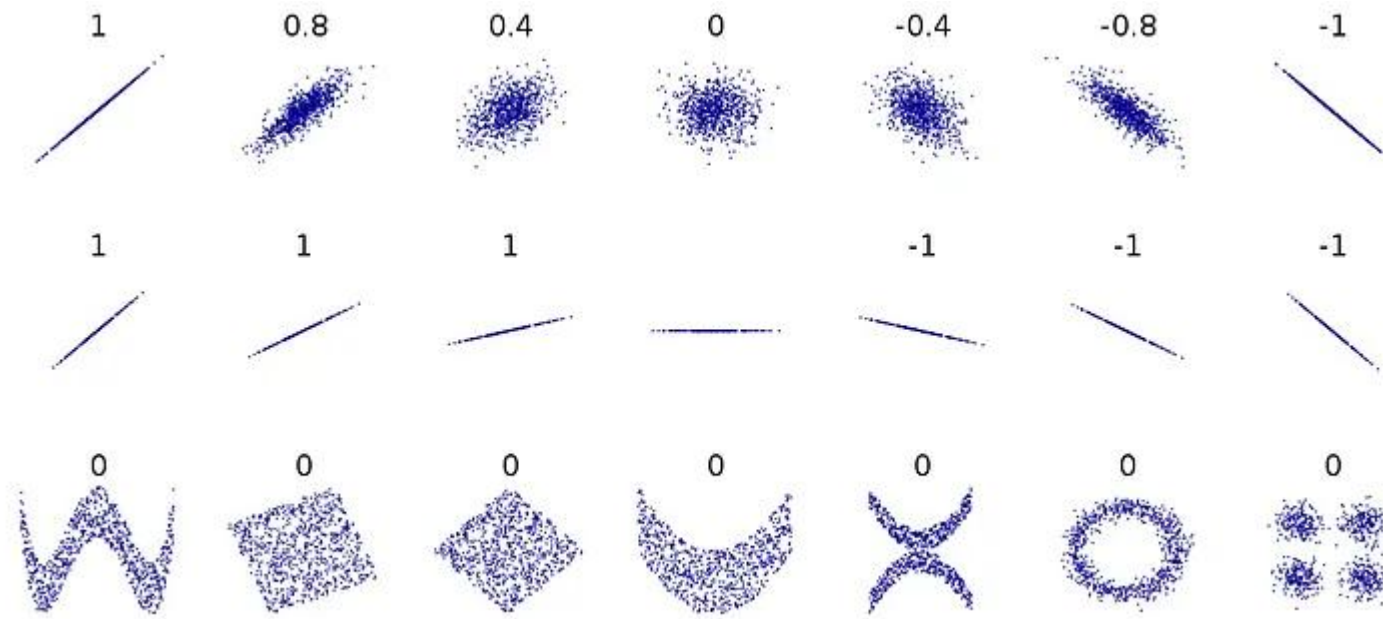
<https://clauswilke.com/dataviz/visualizing-associations.html>

# Correlation coefficient (r)



<https://clauswilke.com/dataviz/visualizing-associations.html>

# Why look at the data when we have correlation coefficient?



<https://medium.com/@becaye-balde/visualizing-correlations-scatter-matrix-and-heat-map-d597436b7d23>

# Thank you for your attention



# Now wake up and write some code!



For people working from home, go through the R-code lesson 1 intro, please.

There is a small assessment in the end of the R markdown file, where we ask you to send answers by email 😊