# Journey through R

## A motivating workshop

Julius Fenn

9th, 10th, 16th, 17th of May, 2025

# Workshop

This workshop consists of three parts:

1 Pep talk (*a speech which is intended to encourage someone to make more effort or feel more confident*)

- first some basics

2 Useful Software + Knowledge Management: How to enter the space shuttle and learn things?!

3 Introduction to R

- Overview
- Objects
- Data Structures
- ...

4 Amazing Applications of R

- typical analyses sequences in action
- bibliometric analysis

5 Templates (R Markdown)

# Part 1: Pep talk

# Part 1: Pep talk - Some Basics

# Definition: What is statistics?

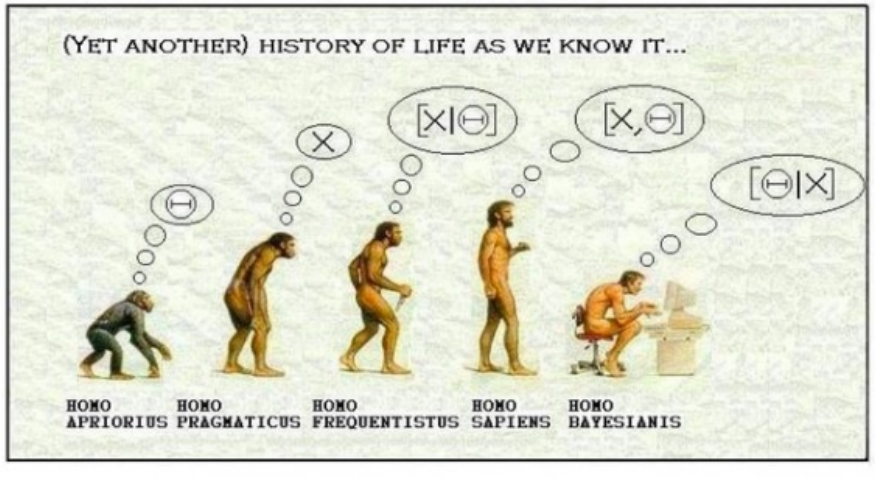
- Statistics is the science of responsible data analysis.
- Statistics is a cross-sectional discipline that is characterized by a combination of knowledge in
  - Mathematics (abstraction, modeling, stochastics, numerics),
  - Computer science (programming, scientific computing),
  - Fields of application (life, natural or economic sciences, etc.).
- Statistical modeling allows the description of stochastic phenomena and thus supports the finding of rational decisions under uncertainty.

Encyclopedia Britannica: Statistics is the art and science of gathering, analyzing and making inferences from data. Originally associated with numbers gathered for governments, the subject now includes large bodies of method and theory.

# The theoretical master-mind: The Statistician

**Statisticians:** theoretical driven, discussing terms like point estimates, margins of error, confidence intervals and are separated between “Frequentists” and “Bayesians”

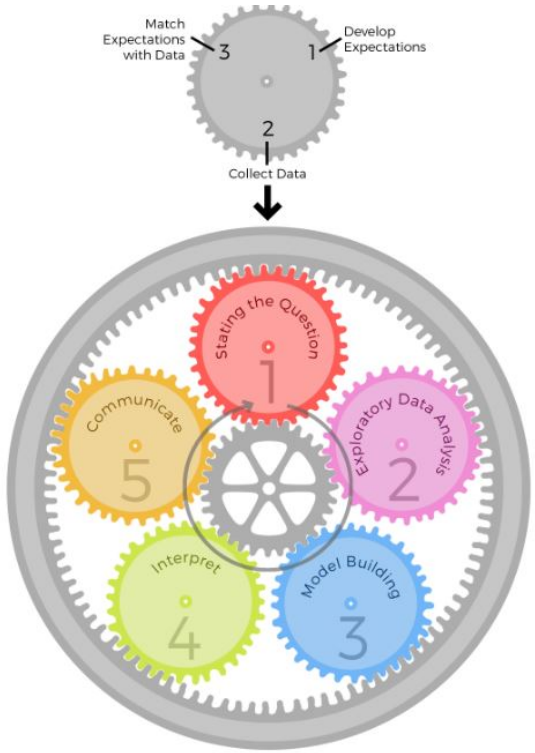
- The Frequentist approach to statistics (and testing) is a method which makes predictions on the underlying truths of the experiment, using only data from the current experiment.
- The Bayesian approach to statistics is a method that encodes past knowledge of similar experiments into a statistical device, known as prior. This prior is combined with current experiment data to make a conclusion on the test (knowledge accumulation).




https://cxl.com/blog/bayesian-frequentist-ab-testing/

# The modern (applied) statistician: The Data Scientist

**Data scientists:** *data analysis is an art*; a process of data ingest, data transformation, exploratory data analysis, model selection, model evaluation, and data storytelling




Epicycles of Analysis

see book: Peng, R. D., & Matsui, E. (2016). The Art of Data Science: A Guide for Anyone who Works with Data. Lulu.com. https://bookdown.org/rdpeng/artofdatascience/

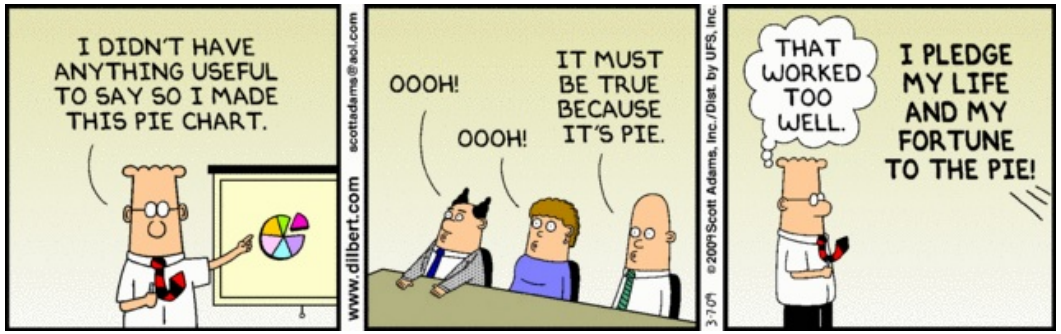
# Part 1: Pep talk - The Real Talk

# Why learn programming?!

(will I look like this cat?)

🍐🧺: https://osf.io/ytb8q

# arguable reason: to impress someone

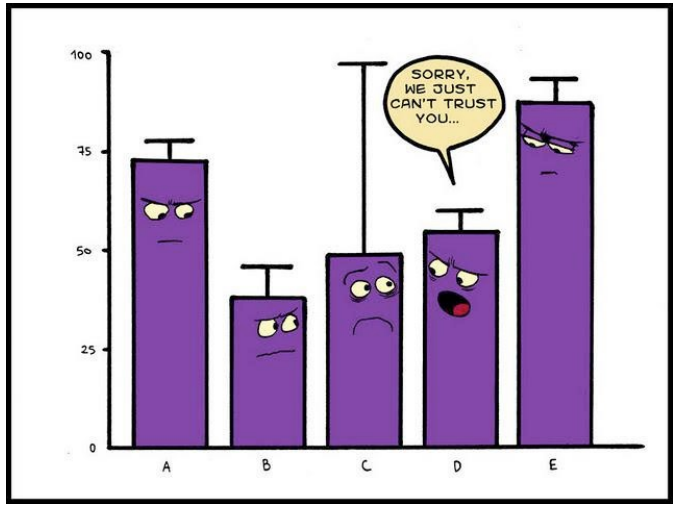



*Anyone wants to see impressive R Code in action?! > "getMeARoom.R"* , often 💩, sometimes 😄

https://stats.stackexchange.com/questions/423/what-is-your-favorite-data-analysis-cartoon

# better reason: to embrace the complexity of our world - prediction

to appreciate uncertainty of our predictions (e.g. prediction paradox)




> random variables, distributions, expectations, confidence intervall, variance,...

see book: Silver, N. (2015). The Signal and the Noise: Why So Many Predictions Fail--but Some Don't. Penguin Publishing Group.

# better reason: to embrace the complexity of our world - system theory

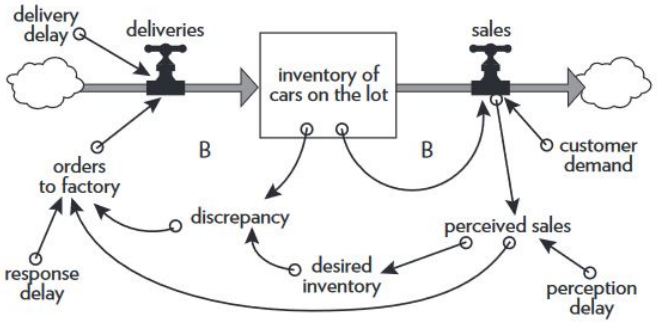



**Figure 31.** Inventory at a car dealership with three common delays now included in the picture—a perception delay, a response delay, and a delivery delay.

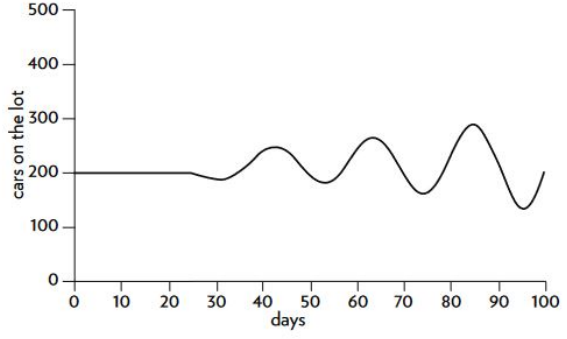



**Figure 32.** Response of inventory to a 10-percent increase in sales when there are delays in the system.

> computational modelling, simulation…

see book: Meadows, D. H. (2008). Thinking in Systems: A Primer. Chelsea Green Pub.

# down-to-earth reason: to get a job! - where to study?

- Göttingen: Master of Science (MSc) in Applied Statistics; https://www.uni-goettingen.de/en/421501.html
- Bamberg: Masterstudiengang Survey-Statistik; https://www.uni-bamberg.de/miss/
- Trier: Master of Science (MSc) in Applied Statistics; https://www.uni-trier.de/universitaet/fachbereiche-faecher/fachbereich-iv/faecher/volkswirtschaftslehre/professuren/wirtschafts-und-sozialstatistik/studieren/applied-statistics-msc
- Tübingen: Kognitionswissenschaft (MSc); https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/studium/studiengaenge/kognitionswissenschaft/
- Economics Master, ...

# Part 2: Useful Software + Knowledge Management

# Part 2: Useful Software

# if you want to publish

**Open Researcher and Contributer ID**

ORCID
Connecting Research and Researchers

https://orcid.org/

**ResearchGate, GoogleScholar**

ResearchGate

https://scholar.google.com/intl/de/scholar/citations.html https://www.researchgate.net/login

# if you want to promote your stuff

**set up your own webpage**

- Why not use R package blogdown: https://bookdown.org/yihui/blogdown/
- or create your website with **Quarto**: https://www.marvinschmitt.com/blog/website-tutorial-quarto/
- ...

**instant messaging program**

- Discord
- Slack
- ...

**social media**

- LinkedIn, Xing
- X (Mastodon)

# bring colloboration to a next level

**write your articles together**

- Google Docs (invite people and create a google account)
- (Microsoft: Microsoft 365 and OneDrive or SharePoint)
- **overleaf**: https://de.overleaf.com/

**share responsibilities**

- Taskade: https://www.taskade.com/
- ...

**share your code**


- **GitHub**: https://github.com/

**share your additional files (literature)**

- Nextcloud: https://nextcloud.com/
- Google Drive

# make your research transparent and reproducible

**upload all your files**


https://osf.io/

1. Register your research
2. Share data, material, and code
3. Publish preprint/share paper

**write all your scripts in annotated, reproducible documents**


- **Quarto** (R Markdown)
- use the R package **papaja**: https://github.com/crsh/papaja

# set up studies, statistical software [my favourites]

**set up studies**

- lab.js
  - https://lab.js.org/
  - Slack Channel: https://join.slack.com/t/nmbrcrnchrs/shared_invite/zt-1ukiirqar-fvuhI9FXE9xrIdVttY6TAQ
- SoSci Survey: GUI based (but flexible. . . if you know PHP), easy to use
  - Video Tutorials: https://www.soscisurvey.de/de/screencast

**statistical software**

- R
- JASP (Just Another Statistics Program): https://jasp-stats.org/
  - SPSS-like GUI for data analysis with R (unflexible!)
  - Includes Bayesian analysis, growing
- Python
  - Some similarities to R (interpreted, syntax), but more powerful
  - Not customized for statistics, popular for machine learning
- (Mplus)

- Power Analysis: G*Power (or simulation studies): https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower
- complex analyses - bwUniCluster 2.0: https://wiki.bwhpc.de/e/BwUniCluster2.0

# use A.I. tools [the future]

**List of AI Tools for Research Workflow in Academia**: https://docs.google.com/document/d/1mb4SWtqyi1iEGCn2uTnHkPHqW3UoQr8b0xv5_81a-4Y/edit?usp=sharing

get state of the art tools for low cost / for free as a student: https://education.github.com/pack

I personally using the following AIs:

- for programming within Visual Studio Code: https://github.com/features/copilot
- summarizing articles: https://www.chatpdf.com/

two conversational search engines:

- providing sources to questions: https://www.perplexity.ai/
- **ChatGPT**: https://chat.openai.com/

for reviews:

- (for literature reviews https://rayyan.ai/ and https://www.bibliometrix.org/home/)

# Part 2: Knowledge Management

# Check out the collection of materials!

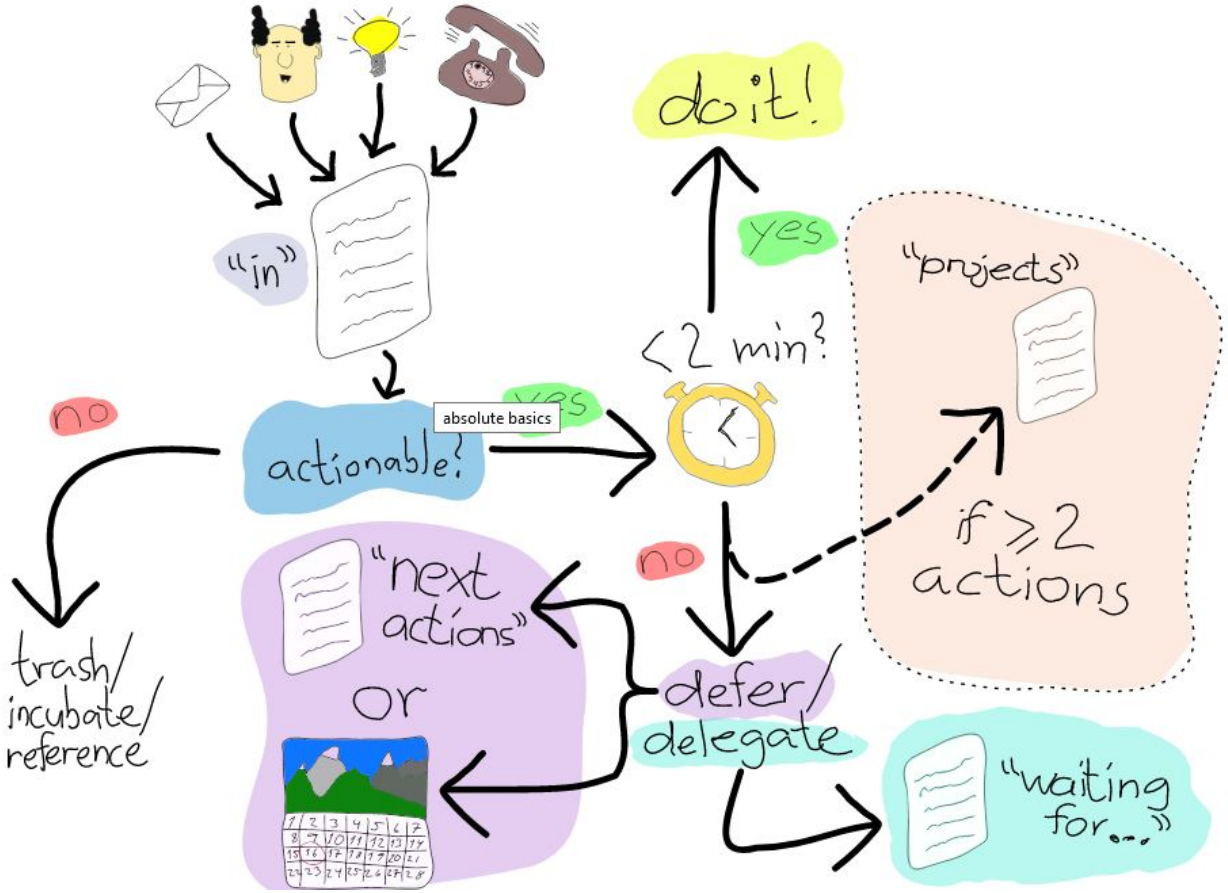
see: https://docs.google.com/document/d/1Z40Rkux_Ysq15VziCJJH21ca07ipwN52dA_LFYIsZ2g/edit?usp=sharing

start reading:

- Possible Learning Process
- Mixed -> Knowledge management and learning

# Getting Things Done - workflow

**workflow:**




see book: Allen, D. (2015). Getting Things Done: The Art of Stress-Free Productivity. Penguin.

# Getting Things Done - projects

**projects:** *a project is anything we want to do that requires more than one action step. It's therefore a mechanism to remember that, when we finish that first action step, there will still be something more to do*

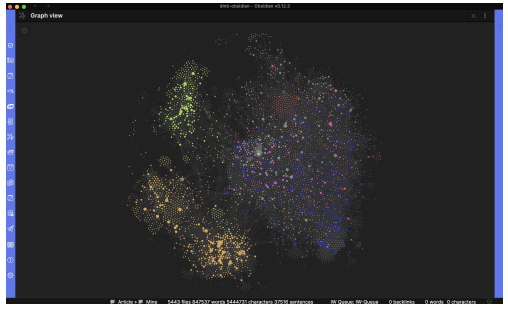
1. Set up a project list, which is an index, in no particular order, of all your open loops (to dos).
2. For every project define at least the first next action step (OR waiting for, or calendar action).
3. The Projects list and project plans are typically reviewed in your GTD Weekly Review, ensuring each project has at least one current next action, waiting for, or calendar item.
4. It's fine to have multiple next actions on any given project, as long as they are parallel and not sequential actions.
5. Projects are listed by the outcome you will achieve when you can mark it as done.
    - Effective project names motivate you toward the outcome you wish to achieve, and give you clear direction about what you are trying to accomplish.

=> that's how you set up a **learning plan**

# How to learn - One Approach

**Organize your knowledge as a "Zettelkasten":**

- use Anki: https://apps.ankiweb.net/
- organize the "Zettel" by using Obsidian: https://obsidian.md/




Check out YouTube Video SpiegelMining – Reverse Engineering von Spiegel-Online: https://www.youtube.com/watch?v=-YpwsdRKt8Q

# Part 3: Introduction to R

# Part 3: Introduction to R - Overview

*Setting up your first project*

# Main Literatur, use my templates

- easy readable introduction R and statistics: Field, Andy, Jeremy Miles, and Zoë Field. Discovering Statistics Using R. SAGE, 2012.
- Resource for improving coding skills and deepening (technical) understanding of R:
  - Wickham, Hadley. Advanced R, Second Edition. CRC Press, 2019. https://adv-r.hadley.nz/.
  - Jones, Owen, Robert Maillardet, and Andrew Robinson. Introduction to Scientific Programming and Simulation Using R, Second Edition. CRC Press, 2014. https://nyu-cdsc.github.io/learningr/assets/simulation.pdf.

plus collected materials / workshops...

**templates**

in the folder "templates" you find a template for...

- basic_R_file.R: a (not so) standard R file (a good start)
- folder "templates" > "advanced_modular_R_file.qmd" file: a advanced Quarto file (“Let us see how high we can fly before the sun melts the wax in our wings.”)

# Introduction to R

- R is a programming language and tool for statistical computing and data analysis
- consist of basic functionalities (i.e., objects and functions) as well as packages that allow for robust and efficient coding
- in R are multiple object-oriented programming (OOP) included like S3, R6, S4, ..., enables *polymorphism* (use the same function form for different types of input)

```
summary(c(TRUE, TRUE, FALSE, TRUE))
```

```
##    Mode   FALSE    TRUE
## logical       1       3
```

```
summary(rnorm(n = 100, mean = 0, sd = 20))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -39.558  -6.845   5.484   3.811  16.007  53.401
```

Important:

- Everything that exists is an object.
- Everything that happens is a function call.

# Side-Note: Polymorphism?

```
methods(generic.function = "summary")[1:20]
```

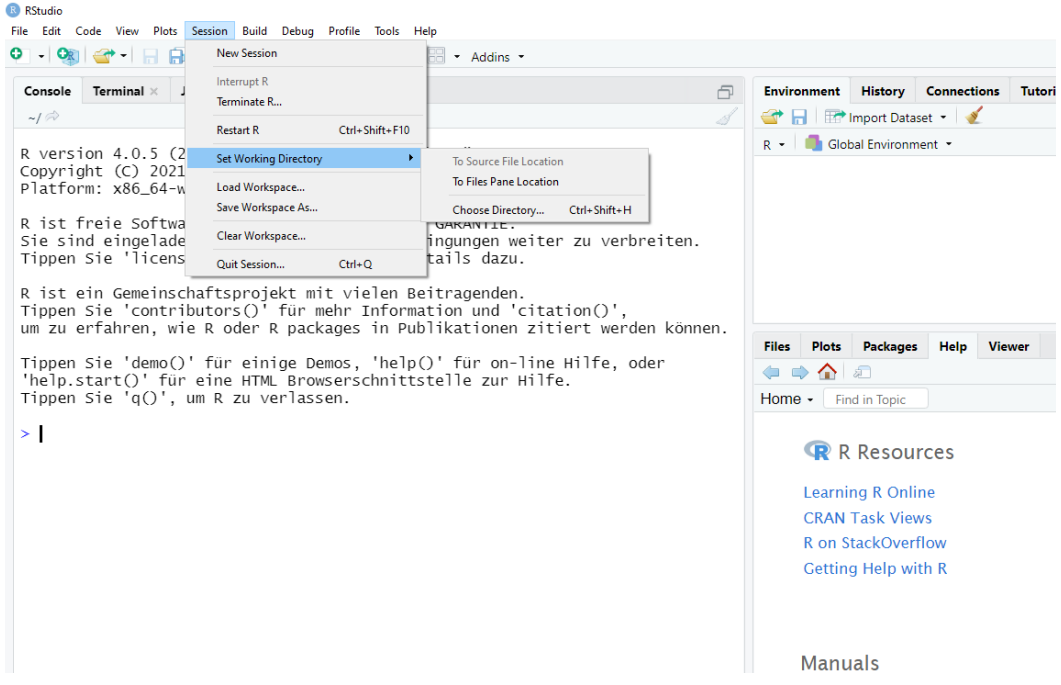
```
##  [1] "summary,ANY-method"                "summary,BFBayesFactor-method"
##  [3] "summary,BFBayesFactorTop-method" "summary,BFodds-method"
##  [5] "summary,BFprobability-method"    "summary,diagonalMatrix-method"
##  [7] "summary,sparseMatrix-method"     "summary.aareg"
##  [9] "summary.afex_aov"                "summary.agnes"
## [11] "summary.allFit"                  "summary.Anova.mlm"
## [13] "summary.aov"                     "summary.aovlist"
## [15] "summary.areg.boot"               "summary.aspell"
## [17] "summary.bcnPowerTransform"       "summary.bcnPowerTransformlmer"
## [19] "summary.boot"                    "summary.cch"
```

other important functions are:

```
?typeof
?class
?mode
```

# set your working directory

- every time when starting R set your working directory (or better create an R Project*)




*you could also use the R API:

```
# sets the directory of location of this script as the current directory
setwd(dirname(rstudioapi::getSourceEditorContext()$path))
```

# your workflow

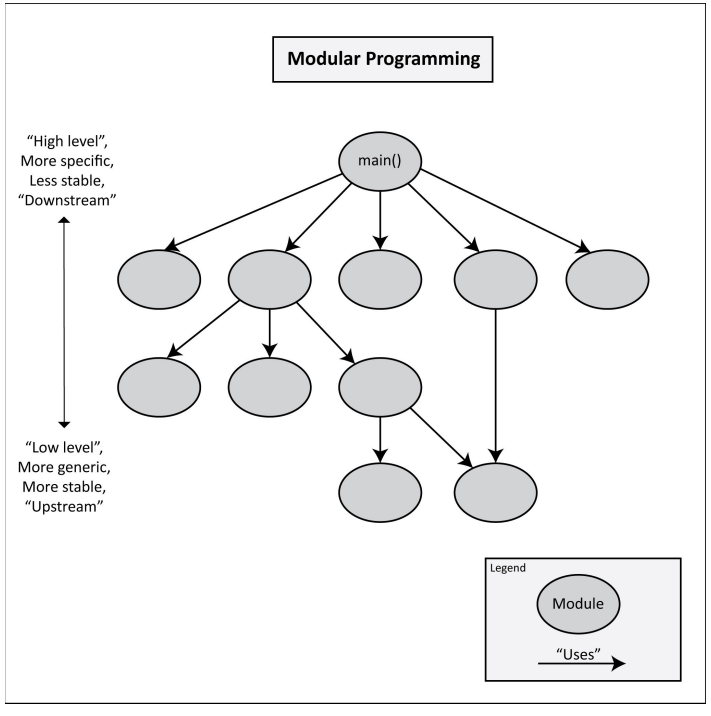
! **modular programming**

if using R projects:

- Create a project folder
- In this folder, organize your file in subfolders (e.g., data)
- All filepaths in your script(s) are specified relative to the folder's top level
- Thus, your working directory is always this top level

# Side-Note: modular programming?

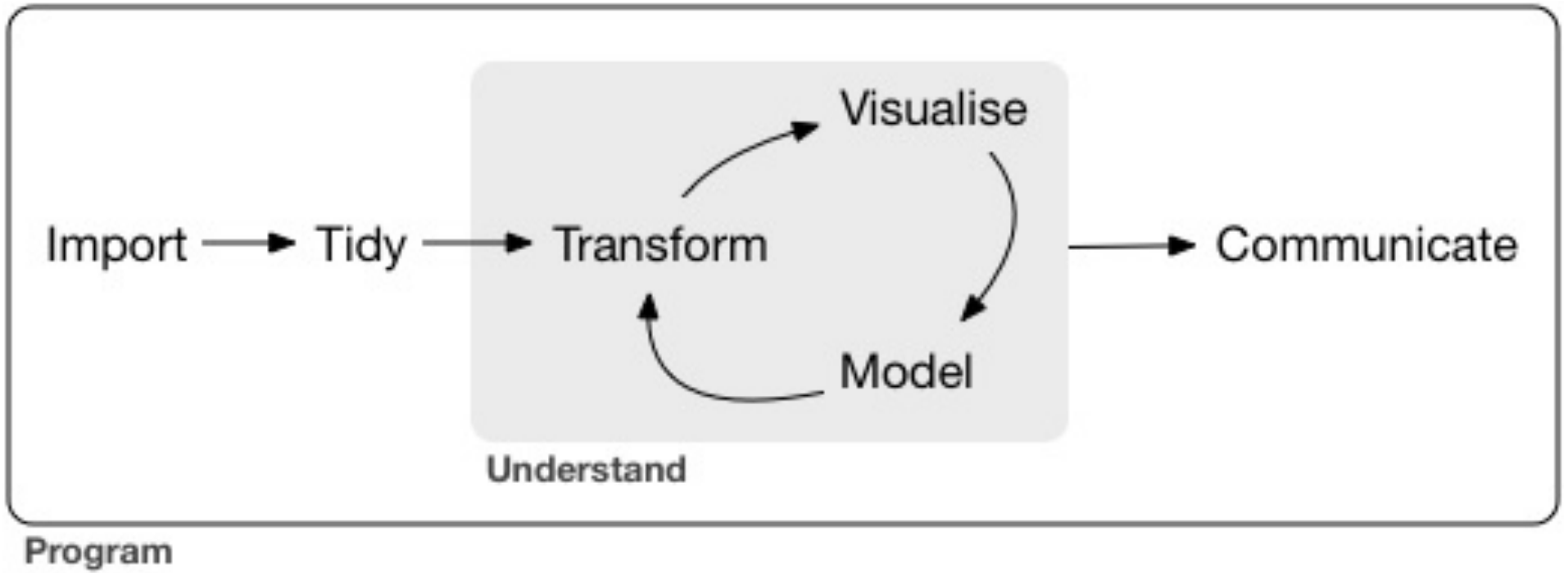
you separate your programs into separate functional units (modules). Each module does a well-defined task and the modules are called by one-another as needed. Typically, the state of each module is encapsulated and only supposed to be altered by the functions from that module.




> Modular programming decomposes a large program into modules!

# Side-Note: modular programming - Importance

without following the principle of modular programming (*or sometimes structured programming*) you cannot set up a typical data science project:




see in detail: https://r4ds.had.co.nz/introduction.html

# RStudio Projects

When using a project, RStudio will automatically set your working directory to the location of the project file.

Additionally, RStudio will

- load .RData files
- load .RHistory files
- open source files (scripts)
- restore RStudio settings

This will particularly benefit your workflow when collaborating with others/sharing your script, for example by using


https://codehorizons.com/making-your-first-github-r-project/

# hands-on: set up a project!

and remember the philosophy of R:

**Everything that exists is an object.**

**Everything that happens is a function call.**

# Part 3: Introduction to R - Objects

## *Using R as a sophisticated calculator*

# ??????! I am lost

Is anyone lost?

# the very basics

**Explanation:**

1 get help

2 assignments

3 operators

4 comparisons

5 comments: everything that follows #

> case sensitive: usage of CAPITAL and small letters matters!

**R commands:**

1

```
?topic, help(topic)
```

2

```
x <- 5 (recommended), x = 5, 5 -> x
```

3

```
+, -, *, /, ^, &, &&; see help("+")
```

4

```
== , != , >, >= , <, <=; see help("=")
```

5

```
# I am a comment
```

# basic data structures

**Explanation:**

1 integer

2 double

3 logical

4 character

5 missings

**R commands:**

1

```
1; 2; 301L
```

2

```
1.0; .141; 1.23e-3; NaN; Inf; -Inf
```

3

```
TRUE; FALSE #(not T, F!)
```

4

```
"hello"; "I'm a string"
```

5

```
NA
```

# basic data structures - construction & coercion I

**Explanation:**

Coercion: When you call a function with an argument of the wrong type, R will try to coerce values to a different type so that the function will work. R will convert from more specific types to more general types.

**R commands:**

you define a vector x as follows

```
x <- c(1, 2, 3, 4, 5)
x
```

```
## [1] 1 2 3 4 5
```

```
typeof(x); class(x)
```

```
## [1] "double"
```

```
## [1] "numeric"
```

**R commands:**

change the second element of the vector to the word “hat.”

```
x[2] <- "hat"
x
```

```
## [1] "1"   "hat" "3"   "4"   "5"
```

```
typeof(x); class(x)
```

```
## [1] "character"
```

```
## [1] "character"
```

# basic data structures - construction & coercion II

**Explanation:**

Coercion:

- coerce to a type xxx by as.xxx()
- check if xxx is a specific type by is.xxx()
- when combining different data types, they will be coerced to the most flexible type
- coercion often happens automatically

**R commands:**

```
x <- 1
is.numeric(x)
```

```
## [1] TRUE
```

```
as.logical(x)
```

```
## [1] TRUE
```

```
x <- c(FALSE, FALSE, TRUE)
as.numeric(x)
```

```
## [1] 0 0 1
```

# basic data structures - represent nothing

**Explanation:**

In R, there are three ways to represent 'nothing', but the reason for the missingness of the information can be distinguished:

- NA: missing sample values, impossible coercion, . . .
- NaN: undefined results in mathematical operation (e.g. log(-1), 1/0)
- NULL: null pointer, i.e. pointer in empty/ undefined memory

**R commands:**

```
c(3, NA)
```

```
## [1]  3 NA
```

```
c(3, 0/0)
```

```
## [1]   3 NaN
```

```
c(3, NULL)
```

```
## [1] 3
```

```
max(3, NA)
```

```
## [1] NA
```

# basic data structures - Infinity

**Explanation:**

Some mathematical operations can be performed with Inf and -Inf:

**R commands:**

```
 max(3, Inf)
```

```
## [1] Inf
```

```
 min(3, Inf)
```

```
## [1] 3
```

```
 c(Inf + Inf, (-Inf) * Inf, Inf - Inf)
```

```
## [1]  Inf -Inf  NaN
```

# hands-on: try out the basic data structures

The bored people can open the R Reference Card 2.0 and try out more fancy stuff: https://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf


check out also the folder "additional scripts -> Basic Vocabulary"

# Part 3: Introduction to R - Data Structures

*We are going to face the 10th dimension (wuuuuhu)*

# ??????! I am not lost but thirsty

Anyone needs a coffee / break?

# complex data types

complex data structures in R can be organized by their dimensionality and if all their contents are of the same type (homogeneous), or not (heterogeneous)

**homogeneous:**

- 1d atomic vector

```
1:10; c(1,2,3,4,5,6,7,8,9,10)
```

- 2d matrix

```
matrix(data = NA, nrow = 2, ncol = 3)
```

- nd array

```
array(1:60, dim=c(3,4,5))
```

**heterogeneous:**

- 1d list

```
list(1:10, letters)
```

- 2d matrix data frame

```
data.frame(id = 1:26,
           letters = letters,
           constant = "Hello World")
```

> we need depending on the issue we are facing different database paradigms!

check out Fireship YouTube Video: https://youtu.be/W2Z7fbCLSTw

# complex data types - very rare in psychology (social science)

often we have "simple" rectangular data, which are made of (values are associated with a variable and a observation):

- **column**, which represents a variable (like ID)
- **rows**, which represents an instance of data in the data set (like a participant)

```
DT::datatable(dat_twins, options = list(pageLength = 5))
```

Show 5 entries Search:

| | ID | IDZP | GR | GES | OG | VG | CC | KU | IQ | KG | IQ.cat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1913.88 | 1005 | 6.08 | 54.7 | 96 | 57.607 | below average |
| 2 | 2 | 1 | 2 | 2 | 1684.89 | 963 | 5.73 | 54.2 | 89 | 58.968 | below average |
| 3 | 3 | 2 | 1 | 2 | 1902.36 | 1035 | 6.22 | 53 | 87 | 64.184 | below average |
| 4 | 4 | 2 | 2 | 2 | 1860.24 | 1027 | 5.8 | 52.9 | 87 | 58.514 | below average |
| 5 | 5 | 3 | 1 | 2 | 2264.25 | 1281 | 7.99 | 57.8 | 101 | 63.958 | medium |

Showing 1 to 5 of 20 entries Previous 1 2 3 4 Next

# every data set should be accompanied by a codebook

Sample size N = 20 Variables (columns):

- ID: Identifier of the individual person
- IDZP: Identifier of the twin pair
- GR: Birth order
- GES: Sex (1 = male, 2 = female)
- OG: Surface area of the cerebral cortex in cm2
- VG: Volume of the forebrain in cm3
- CC: Area of the corpus callosum in cm2
- KU: Circumference of head in cm
- IQ: Intelligence quotient
- KG: Body weight in kg
- additionally IQ.cat: grouped intelligence quotient in 3 groups

Study: https://n.neurology.org/content/50/5/1246

# Side-Note: the most scientific study ever

Hypothesis: the size and shape of the human forebrain predict intelligence!

the mighty correlation matrix:

```
round(x = cor(x = dat_twins[, c("VG", "CC", "KU", "KG", "IQ")],
              method = "spearman"), digits = 2)
```

```
##      VG   CC   KU   KG   IQ
## VG 1.00 0.80 0.63 0.36 0.11
## CC 0.80 1.00 0.59 0.13 0.34
## KU 0.63 0.59 1.00 0.22 0.21
## KG 0.36 0.13 0.22 1.00 0.07
## IQ 0.11 0.34 0.21 0.07 1.00
```

> What is your opinion?

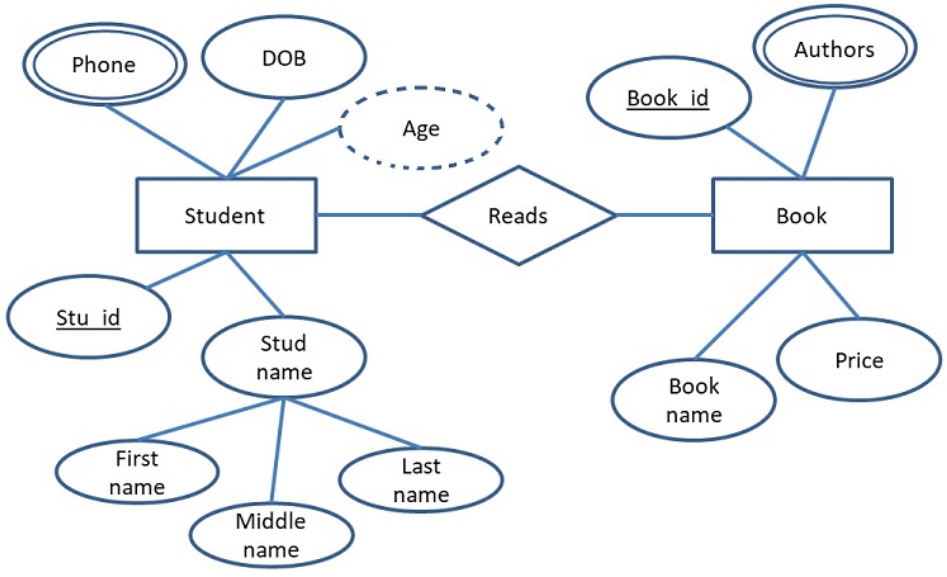
- Spearman correlation coefficient only maps monotone relationships; r_SP > 0, if in tendency x large -> also y large and vice versa; indicate a monotonic relationship in the same direction (not linear relationship!)
  - to be applied especially when the data are not normally distributed, the scales have unequal answering options, for very small sample sizes (or better use concordance measures)

# complex data types - the SQL side of the story

for huge data sets you could use, for example SQL (Structured Query Language), which is a domain-specific language used in programming and designed for managing data held in a relational database management system; check out: https://www.dbis.informatik.uni-goettingen.de/Mondial/

> multiple relational databases are matched by a primary key / multiple primary keys

and complex data bases can be depicted for example as entity relationship models:




there are multiple database paradigms out there: https://www.youtube.com/watch?v=W2Z7fbCLSTw

# complex data types - add attributes

All objects can have arbitrary additional attributes, used to store metadata about the object

- can be thought of as a named list (with unique names); other frequently encountered attributes: "dimnames", "names", "class"(!)
- can be accessed individually with attr() or all at once (as a list) with attributes()
- arrays are simply vectors with a "dim"-attribute.
- factor is a vector with attribute levels

```
dat <- data.frame(id = 1:26,
            letters = letters,
            constant = "Hello World")
head(dat)
```

```
##   id letters    constant
## 1  1       a Hello World
## 2  2       b Hello World
## 3  3       c Hello World
## 4  4       d Hello World
## 5  5       e Hello World
## 6  6       f Hello World
```

```
attr(x = dat, which = "names")
```

```
## [1] "id"       "letters"  "constant"
```

```
attributes(x = dat)
```

```
## $names
## [1] "id"       "letters"  "constant"
##
## $class
## [1] "data.frame"
##
## $row.names
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
## [26] 26
```

# hands-on: try out the different data structures

Hint: we have the following data structures:

**homogeneous:**

- 1d atomic vector

```
1:10; c(1,2,3,4,5,6,7,8,9,10)
```

- 2d matrix

```
matrix(data = NA, nrow = 2, ncol = 3)
```

- nd array

```
array(1:60, dim=c(3,4,5))
```

**heterogeneous:**

- 1d list

```
list(1:10, letters)
```

- 2d matrix data frame

```
data.frame(id = 1:26,
           letters = letters,
           constant = "Hello World")
```

If you want to save your data structures to an R object e.g. write:

```
myFirstVector <- 1:10
```

# Part 3: Introduction to R - Subsetting

*From the 10th dimension down to spaceship earth again.*

# ??????! When you had to debug your first error messages

recommended TV shows: I'm a Celebrity … Get Me Out of Here! (also valuable: Get the F*ck out of my House)

# Subsetting - Theory

- there are three subsetting operators: [, [[, and $
- the three types of subsetting:
  - Positive integers return elements at the specified positions.
  - Logical vectors select elements where the corresponding
  - logical value is TRUE; application of logical expressions.
- character vectors to return elements with matching names.
- important differences in behavior of different objects (e.g., vectors, lists, factors, matrices, and data frames)
- more advanced subsetting, in particular in combination with complex logical expressions, can be done using the functions subset() and which().
- (the default drop=TRUE simplifies the data type of the result.)

you "only" need to learn the behavior of

- Positive Integers
- Negative integers
- Logical vectors
- Character vectors

on different types of data!

# Subsetting - Atomic Vector I

```
# define an R Object
x <- c(2.1, 4.2, 3.3, 5.4)
```

**Positive Integers:**

```
x[c(3, 1)]
#> [1] 3.3 2.1
x[order(x)]
#> [1] 2.1 3.3 4.2 5.4

# Duplicate indices will duplicate values
x[c(1, 1)]
#> [1] 2.1 2.1

# Real numbers are silently truncated to inte
x[c(2.1, 2.9)]
#> [1] 4.2 4.2
```

**Negative integers:** exclude elements at the specified positions

```
x[-c(3, 1)]
#> [1] 4.2 5.4
```

**Logical vectors:** select elements where the corresponding logical value is TRUE

```
x[c(TRUE, TRUE, FALSE, FALSE)]
#> [1] 2.1 4.2
x[x > 3]
#> [1] 4.2 3.3 5.4
```

# Subsetting - Atomic Vector II

```
# define an R Object
x <- c(2.1, 4.2, 3.3, 5.4)
```

**Logical vectors:** select elements where the corresponding logical value is TRUE

```
x[x>5]
```

```
## [1] 5.4
```

```
x>5
```

```
## [1] FALSE FALSE FALSE  TRUE
```

**Character vectors:** return elements with matching names

```
(y <- setNames(x, letters[1:4]))
#>   a   b   c   d
#> 2.1 4.2 3.3 5.4
y[c("d", "c", "a")]
#>   d   c   a
#> 5.4 3.3 2.1

# Like integer indices, you can repeat indice
y[c("a", "a", "a")]
#>   a   a   a
#> 2.1 2.1 2.1

# names are always matched exactly
z <- c(abc = 1, def = 2)
z[c("a", "d")]
#> <NA> <NA>
#>   NA   NA
```

# Subsetting - Matrices I

Setting up your matrix:

```
# ?matrix
a <- matrix(data = 1:9, nrow = 3) # ncol, ...
colnames(a) <- c("A", "B", "C")
a
```

```
##      A B C
## [1,] 1 4 7
## [2,] 2 5 8
## [3,] 3 6 9
```

# Subsetting - Matrices II

```
a <- matrix(1:9, nrow = 3)
colnames(a) <- c("A", "B", "C")
```

```
a[1:2, ]
#>      A B C
#> [1,] 1 4 7
#> [2,] 2 5 8
a[c(TRUE, FALSE, TRUE), c("B", "A")]
#>      B A
#> [1,] 4 1
#> [2,] 6 3
a[0, -2]
#>      A C
```

more advanced: you can also subset higher-dimensional data structures with an integer matrix (or, if named, a character matrix). Each row in the matrix specifies the location of one value, and each column corresponds to a dimension in the array

```
vals <- outer(1:5, 1:5, FUN = "paste", sep =
select <- matrix(ncol = 2, byrow = TRUE, c(
  1, 1,
  3, 1,
  2, 4
))
vals[select]
#> [1] "1,1" "3,1" "2,4"
```

# Subsetting - Data frames I

Setting up your data frame:

```
# ?data.frame
df <- data.frame(x = 1:3, y = 3:1, z = letters[1:3]) # stringsAsFactors !
df
```

```
##   x y z
## 1 1 3 a
## 2 2 2 b
## 3 3 1 c
```

Important to note: there is a equivalent of data.frames in more "modern" R frameworks called "tibble" (has enhanced printing, ...):

```
# ? tibble::tibble
tibble::tibble(x = 1:3, y = 3:1, z = letters[1:3])
```

```
## # A tibble: 3 × 3
##       x     y z
##   <int> <int> <chr>
## 1     1     3 a
## 2     2     2 b
## 3     3     1 c
```

# Subsetting - Data frames II

The most important subsetting to get data of certain objects of observation:

```
df[df$x == 2, ]
#>   x y z
#> 2 2 2 b
df[c(1, 3), ]
#>   x y z
#> 1 1 3 a
#> 3 3 1 c
```

the dollar sign is a shorthand operator: "x$y" is roughly equivalent to "x[["y"]]"; often used to access variables in a data frame

```
df$x
```

```
## [1] 1 2 3
```

```
# There are two ways to select columns from a
# Like a list
df[c("x", "z")]
#>   x z
#> 1 1 a
#> 2 2 b
#> 3 3 c
# Like a matrix
df[, c("x", "z")]
#>   x z
#> 1 1 a
#> 2 2 b
#> 3 3 c
```

# Subsetting - Lists I

Setting up your list:

```
# ?list
x <- list(1:3, "a", 4:6)
x
```

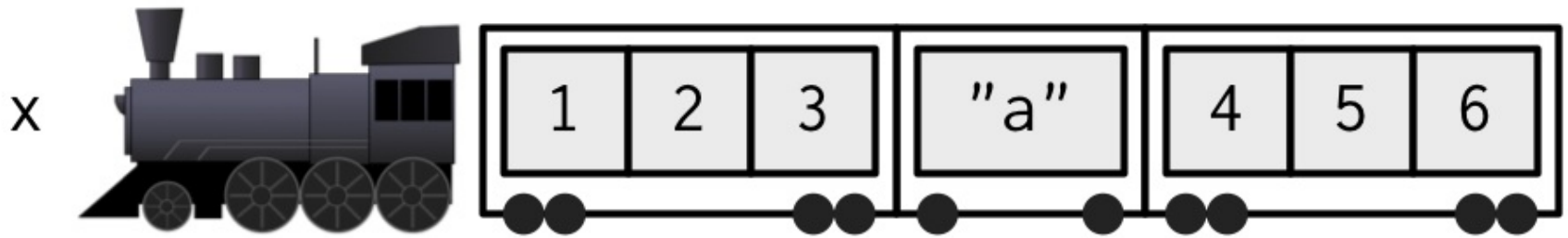
```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] 4 5 6
```

# Subsetting - Lists II

```
x
```

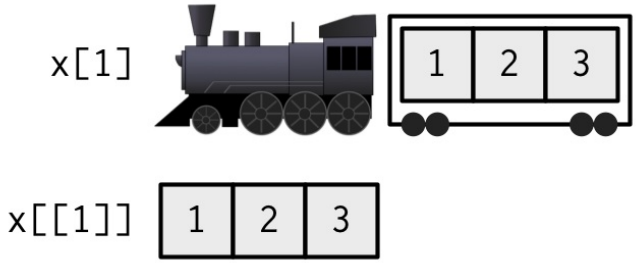
```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] 4 5 6
```

is equivalent to the following representation:

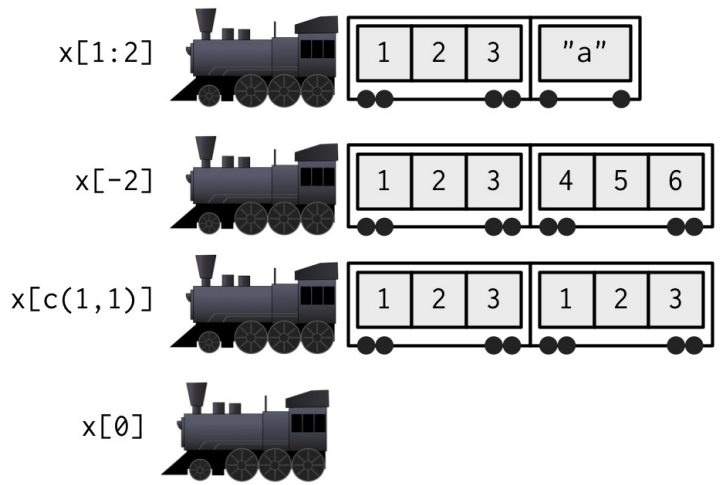


# Subsetting - Lists III

When extracting a single element, you have two options: you can create a smaller train, i.e., fewer carriages, or you can extract the contents of a particular carriage. This is the difference between [ and [[:




When extracting multiple (or even zero!) elements, you have to make a smaller train:



# Subsetting - simplifying vs. preserving

- **simplifying subsets** simplest possible data structure, that can represent the output
- **preserving subsets** keeps the structure of the output the same as the input (better for programming)

Example with lists, which return the object inside the list, not a single element list

```
y <- list(a = 1:2, b = letters[1:3])
str(y[1])
```

```
## List of 1
##  $ a: int [1:2] 1 2
```

```
str(y[[1]])
```

```
##  int [1:2] 1 2
```

# Subsetting - data frames most useful I

Do you remember our twin pair data set?

```
DT::datatable(dat_twins, options = list(pageLength = 5))
```

Show 5 entries Search:

| | ID | IDZP | GR | GES | OG | VG | CC | KU | IQ | KG | IQ.cat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1913.88 | 1005 | 6.08 | 54.7 | 96 | 57.607 | below average |
| 2 | 2 | 1 | 2 | 2 | 1684.89 | 963 | 5.73 | 54.2 | 89 | 58.968 | below average |
| 3 | 3 | 2 | 1 | 2 | 1902.36 | 1035 | 6.22 | 53 | 87 | 64.184 | below average |
| 4 | 4 | 2 | 2 | 2 | 1860.24 | 1027 | 5.8 | 52.9 | 87 | 58.514 | below average |
| 5 | 5 | 3 | 1 | 2 | 2264.25 | 1281 | 7.99 | 57.8 | 101 | 63.958 | medium |

Showing 1 to 5 of 20 entries Previous 1 2 3 4 Next

Three questions:

- How do we get only the twins with IQ higher than 120?
- How do we get only the twins with IQ lower than 100 and body weight higher than 100?
- How can we order the data set according to IQ?

# Subsetting - data frames most useful II

How do we get only the twins with IQ higher than 120?

```
dat_twins[dat_twins$IQ > 120, ]
```

```
##    ID IDZP GR GES      OG   VG   CC   KU  IQ     KG        IQ.cat
## 9   9    5  1   2 1743.04 1034 6.48 53.1 127 62.143 above average
## 10 10    5  2   2 1709.30 1070 6.43 54.8 126 83.009 above average
## 20 20   10  1   1 1971.63 1160 7.67 58.5 124 72.576 above average
```

```
## identical function
subset(dat_twins, IQ > 120)
```

```
##    ID IDZP GR GES      OG   VG   CC   KU  IQ     KG        IQ.cat
## 9   9    5  1   2 1743.04 1034 6.48 53.1 127 62.143 above average
## 10 10    5  2   2 1709.30 1070 6.43 54.8 126 83.009 above average
## 20 20   10  1   1 1971.63 1160 7.67 58.5 124 72.576 above average
```

# Subsetting - data frames most useful III

How do we get only the twins with IQ lower than 100 and body weight higher than 100?

```
 dat_twins[dat_twins$IQ < 100 & dat_twins$KG > 100, ] # ! only single logical operator
```

```
##    ID IDZP GR GES      OG   VG   CC   KU IQ      KG        IQ.cat
## 8  8     4  2   2 1850.64 1079 6.84 55.3 96 107.503 below average
```

> such expressions could be of any complexity

# Subsetting - data frames most useful IV

How can we order the data set according to IQ?

```
 order(dat_twins$IQ)
```

```
##  [1] 16  3  4 14  2 13 15  1  8 12 17  5 11  6  7 19 18 20 10  9
```

```
 head(dat_twins[order(dat_twins$IQ), ])
```

```
##    ID IDZP GR GES      OG   VG   CC   KU IQ     KG        IQ.cat
## 16 16    8  1   1 2154.67 1439 7.62 57.2 85 99.792 below average
## 3   3    2  1   2 1902.36 1035 6.22 53.0 87 64.184 below average
## 4   4    2  2   2 1860.24 1027 5.80 52.9 87 58.514 below average
## 14 14    7  1   1 2018.92 1104 6.32 57.2 88 79.380 below average
## 2   2    1  2   2 1684.89  963 5.73 54.2 89 58.968 below average
## 13 13    7  2   1 2136.37 1067 6.32 57.2 93 83.916 below average
```

```
 # dat_twins[, order(names(dat_twins))] # order data set alphabetically
```

> ordering according to the primary key (ID variable) is often necessary before matching data

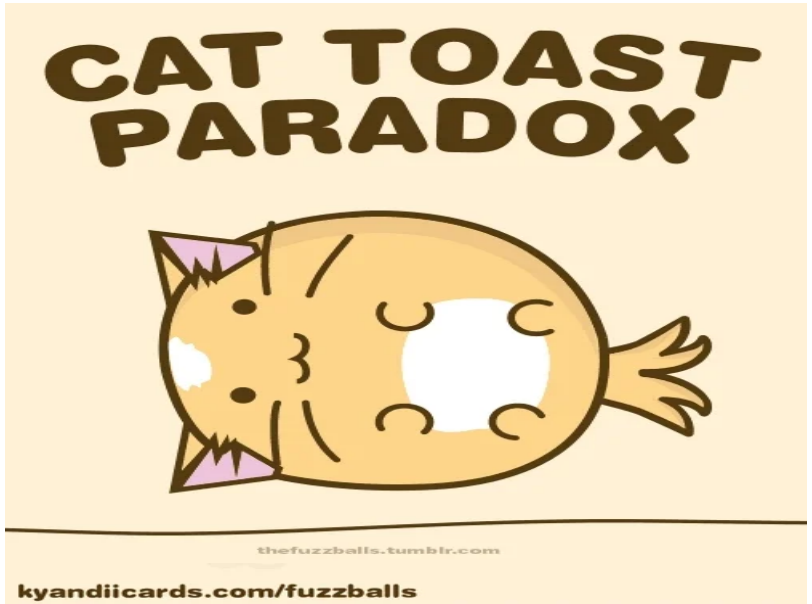
# Part 3: Introduction to R - Flow control

*That's how adults play Looping Louie!*

# ??????! Before running circles we need a break right?!

Time to learn p-hacking for experts (in German we say "T-Test rechnen bis die Tasten glühen")


Instead of p-hacking why not just starting a research program to solve finally this paradox:



# Flow control - vocabulary

There are some new words to learn if you want to become a flow control master:

```
# Control flow
if - &&, || (short circuiting)
for, while
next, break
switch
ifelse

# Debugging
stop
warning
message
```

# Flow control - if statement I

Normally we want to combine if statements with loops (plus functions).

**if statements** have the following form:

```
## abstract representation
if (condition) true_action
if (condition) true_action else false_action
```

in action:

```
x1 <- if (TRUE) 1 else 2 # the experts write code like this!
x1
```

```
## [1] 1
```

```
x2 <- if (FALSE) 1 else 2
x2
```

```
## [1] 2
```

# Flow control - if statement II

```
greet <- function(name, birthday = FALSE) {
  paste0(
    "Hi ", name,
    if (birthday) { # but I prefer this way of writing code
      " and HAPPY BIRTHDAY"
    } else{
      " and you do not have BIRTHDAY"
    }
  )
}
greet("Maria", FALSE)
```

```
## [1] "Hi Maria and you do not have BIRTHDAY"
```

```
greet("Jaime", TRUE)
```

```
## [1] "Hi Jaime and HAPPY BIRTHDAY"
```

# Flow control - if statement III

If statements can be arbitrarily complex:

```
x <- 20

if (x > 30) {
  print("x > 30")
} else if (x <= 30 && x > 20) {
  print("x <= 30 && x > 20")
} else{
  print("everything else")
}
```

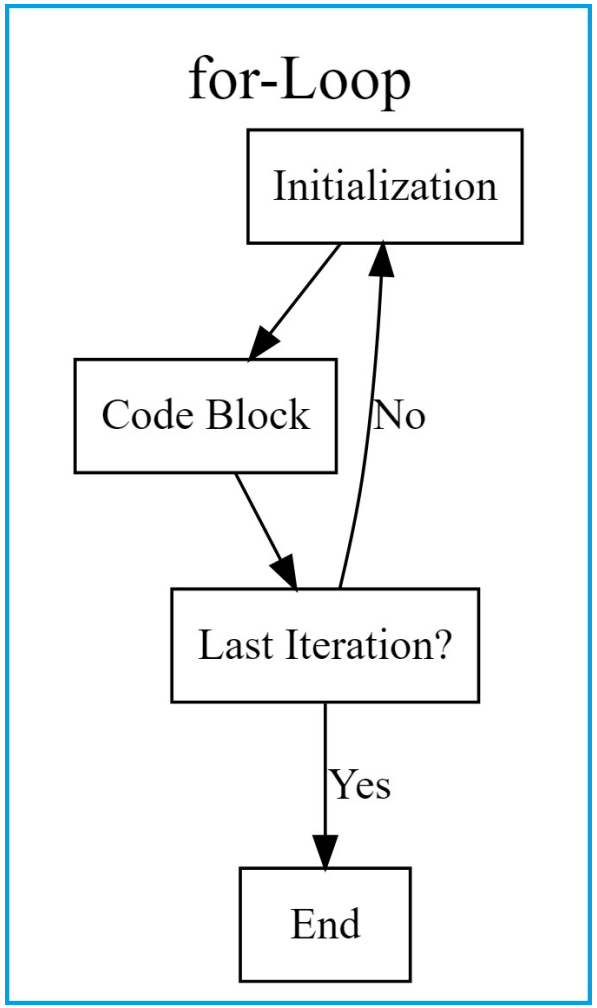
```
## [1] "everything else"
```

An important function is also the ifelse() function:

```
x <- 1:10
ifelse(x %% 2 == 0, "even", "odd")
```

```
##  [1] "odd"  "even" "odd"  "even" "odd"  "even" "odd"  "even" "odd"  "even"
```

# Flow control - For Loop




```r
for(i in 1:10) { # Head of for-loop
  x1 <- i^2 # Code block
  print(x1) # Print results
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
## [1] 36
## [1] 49
## [1] 64
## [1] 81
## [1] 100
```

see multiple examples: https://statisticsglobe.com/for-loop-in-r

# Flow control - Why Awesome?! I

Imagine we want to drawn for the prices of diamonds for every different quality of the cut; data set with over 50.000 diamonds

```
# ? diamonds
# How many cuts?
levels(diamonds$cut)
```

```
## [1] "Fair"      "Good"      "Very Good" "Premium"   "Ideal"
```

```
# Is there a difference in price?
diamonds %>%
  group_by(cut) %>%
  summarize(mean(price))
```

```
## # A tibble: 5 × 2
##   cut       `mean(price)`
##   <ord>             <dbl>
## 1 Fair              4359.
## 2 Good              3929.
## 3 Very Good         3982.
## 4 Premium           4584.
## 5 Ideal             3458.
```

# Flow control - Why Awesome?! II

Imagine we want to drawn for the prices of diamonds for every different quality of the cut; data set with over 50.000 diamonds

```
for(c in levels(diamonds$cut)) {
  cat("quality of cut: ", c, "\n")
  tmp <- diamonds$price[diamonds$cut == c]
  cat("    > # of diamonds: ", length(tmp), "\n")
  cat("    > skewness: ", moments::skewness(tmp), "\n")
  cat("    > kurtosis: ", moments::kurtosis(tmp), "\n")
  hist(tmp, main = c, xlab = "price")
  abline(v = mean(tmp), col = "red") # add mean

  if(length(tmp) < 5000){
    message("Oh no there are not even 5000 diamonds.")
  }
}
```

> by using print() or cat() you can print additional information in the console
>
> by adding if() statements you can implement a complex logical structure in your for loop

# Subsetting - for loop useful example I

In the twin data set we have the variable "IDZP" (identifier of the twin pair)

```
DT::datatable(dat_twins, options = list(pageLength = 5))
```

Show 5 entries Search:

| | ID | IDZP | GR | GES | OG | VG | CC | KU | IQ | KG | IQ.cat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1913.88 | 1005 | 6.08 | 54.7 | 96 | 57.607 | below average |
| 2 | 2 | 1 | 2 | 2 | 1684.89 | 963 | 5.73 | 54.2 | 89 | 58.968 | below average |
| 3 | 3 | 2 | 1 | 2 | 1902.36 | 1035 | 6.22 | 53 | 87 | 64.184 | below average |
| 4 | 4 | 2 | 2 | 2 | 1860.24 | 1027 | 5.8 | 52.9 | 87 | 58.514 | below average |
| 5 | 5 | 3 | 1 | 2 | 2264.25 | 1281 | 7.99 | 57.8 | 101 | 63.958 | medium |

Showing 1 to 5 of 20 entries Previous 1 2 3 4 Next

```
table(dat_twins$IDZP)
```

```
##
##  1  2  3  4  5  6  7  8  9 10
##  2  2  2  2  2  2  2  2  2  2
```

# Subsetting - for loop useful example II

Compute the mean difference of IQ for every twin pair:

```
mean_twinPairs <- double(10) # empty vector made up of 0 to fill up
h=1 # index variable
for(t in levels(dat_twins$IDZP)) {
  # cat("twin pair: ", t, "\n")
  tmp <- dat_twins$IQ[dat_twins$IDZP == t]
  # cat("    > IQ values: ", tmp, "\n")

  mean_twinPairs[h] <- diff(tmp)
  h=h+1 # increase index variable by 1
}

mean_twinPairs
```

```
##  [1] -7  0  2 -7 -1 -5 -5 -9 17 11
```

# Part 3: Introduction to R - Writing Functions

*Statisticians are lazy as hell, that's why we write code only once!*

# ?!!! Never get completely lost anymore

Otherwise it could get dark (at least your mood when writing code)!

# Writing Functions - vocabulary

Also here are some new words to learn if you want to program your own function:

```
# Functions to do with functions
function
missing
return, invisible
```

# Writing Functions - Theory I

- R, at its heart, is a functional programming language. This means that it provides many tools for the creation and manipulation of functions.
- The structure of a function is given by

```
function ( <arglist> ){ <body> }
```

- the keyword function indicates the beginning of a function
- arguments are defined by comma-separated key-value-pairs
- is a block of R commands which are executed by the function

# Writing Functions - Theory II

All R functions are objects and consist of three parts:

- body() the code inside the function.
- formals() the list of arguments which controls how you can call the function.
- environment() the 'map' of the location of the function's variables. If the environment isn't displayed, it means

```
f <- function(x){
  square <- x^2 # compute the square
  return(square) # return the value
}
```

```
formals(f)
```

```
## $x
```

```
body(f)
```

```
## {
##     square <- x^2
##     return(square)
## }
```

```
environment(f)
```

```
## <environment: R_GlobalEnv>
```

# Writing Functions - Human Interaction is Incoming I

Recommendation of Xie (chief software engineer of R):

> Avoid the commands that require human interaction: human input can be highly unpredictable; e.g., we do not know for sure which file the user will choose if we pop up a dialog box asking the user to choose a data file...

The user breaks our function:

```
 f
```

```
## <srcref: file "" chars 1:6 to 4:1>
```

```
 f(x = "I am a string!")
```

```
## Error in x^2: nicht-numerisches Argument für binären Operator
```

# Writing Functions - Human Interaction is Incoming II

If you writing functions it is good practice to control for the input:

```
f <- function(x){
  if(is.numeric(x)){
    square <- x^2 # compute the square
    return(square) # return the value
  }else{
    message("You should provide a numeric value / numeric vector.")
    return(NULL)
  }

}

f(x = "I am a string!")
```

```
## You should provide a numeric value / numeric vector.

## NULL
```

# Writing Functions - Human Interaction is Incoming III

If you have a "character" argument like "sex = c("Male", "Female")" you can check the provided values by:

- match.arg matches arguments against a table of candidate values specified by choices

```
match.arg(arg, choices, several.ok = FALSE)
```

- missing argument will be replaced with the first candidate

```
talk_about_sex <- function(sex = c("Male", "Female", "Non-binary")) {
  match.arg(sex)
}

talk_about_sex("F") # partial matching (not good practice)
```

```
## [1] "Female"
```

```
talk_about_sex()
```

```
## [1] "Male"
```

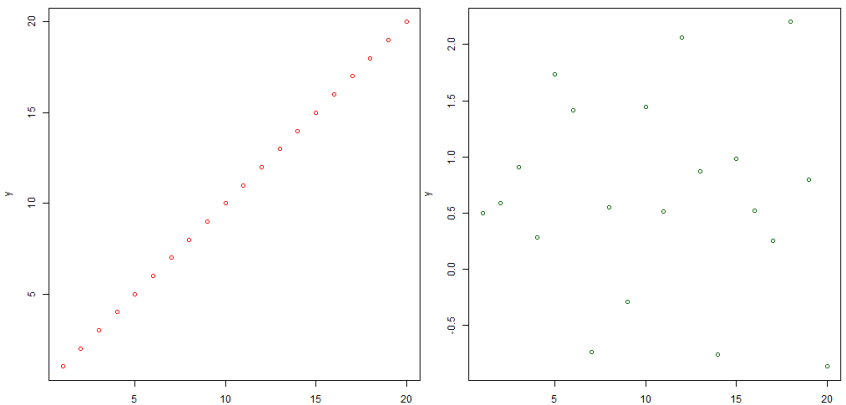
```
talk_about_sex("W")
```

```
## Error in match.arg(sex): 'arg' should be one of "Male", "Female", "Non-binary"
```

# Writing Functions - Function Arguments

- formals() the list of **arguments** which controls how you can call the function.
  - Function arguments in R can have default values.
  - They can be also defined in terms of other arguments.
  - You can determine if an argument was supplied or not with the missing() function.

```
myplot <- function(x, y, mycol = 'red') {
  if (missing(y)) {
    y <- x
    x <- 1:length(y)
    }
  plot(x, y, col = mycol)
}
par(mar = c(4, 4, .1, .1))
myplot(x = 1:20)
myplot(x = 1:20, y = rnorm(20), mycol='darkgreen')
```



# Writing Functions - examples I

Using the **Cairo** package you could save your graphics directly as files to your computer in great resolution:

```
#####################################################################
##### save_graphic()
# save graphic object as png file
#####################################################################
save_graphic <- function(filename){
  tmp <- paste(filename, ".png", sep = "")
  Cairo::Cairo(file=tmp,
               type="png",
               units="px",
               width=2500,
               height=1700,
               pointsize=44, # text size is artifically shrunked when saved
               dpi= "auto",
               bg = "white")
}

dev.off()
save_graphic(filename = "myFirstPlot")
plot(cars$speed, cars$dist)
dev.off()
```

(also possible to use for example ggsave() from the **ggplot2** package)

# Writing Functions - examples II

Using the **stargazer** package you could save your easily set up summary statistics in the APA 7 format:

```
#####################################################################
##### getDescriptives()
# get multiple summary statistics to R console AND / OR to HTML
#####################################################################
### functionals:
# dataset = dat_twins
# variables = c("IQ", "KG")
# nameAPAtable = "myFirstTable"
getDescriptives <- function(dataset = NULL,
                            variables = NULL,
                            nameAPAtable = NULL){
  x <- dataset[, variables]
  ## table
  tmp_descriptives <- sapply(x, function(x) c(
    "Mean"= mean(x, na.rm=TRUE),
    "SD" = sd(x, na.rm=TRUE),
    "Median" = median(x, na.rm=TRUE),
    "CoeffofVariation" = sd(x, na.rm=TRUE)/mean(x, na.rm=TRUE),
    "Minimum" = min(x, na.rm=TRUE)
```

# Part 3: Introduction to R - File and Data Management

*Of all the things I've lost, I miss my mind the most.*
*(Mark Twain)*

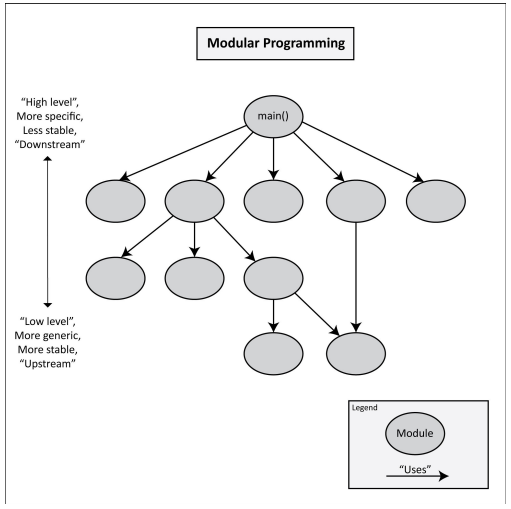
# ! we are now on track for our first sophisticated program

The management of your files, your data, your code is key. Keep on riding...

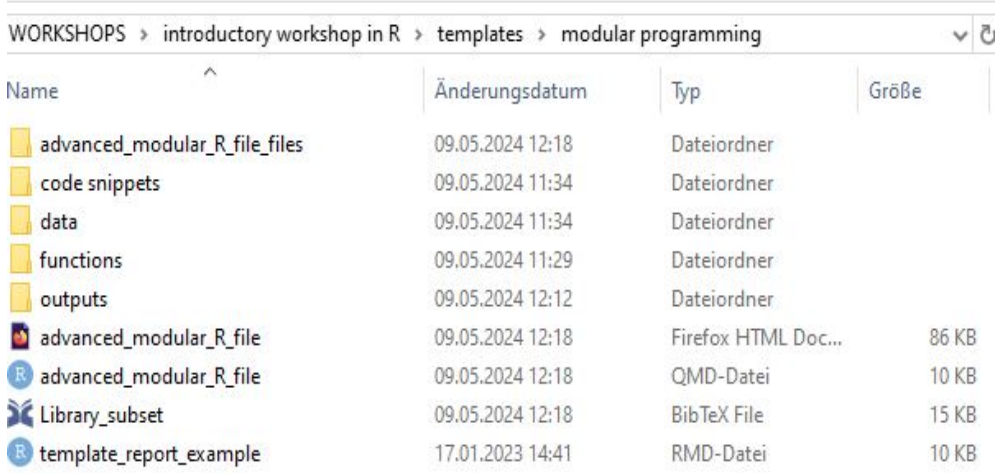
# organize your program in modules I

at best use R projects and separate your programs / code / files into separate functional units (modules) / folders:

- Create a project folder
- In this folder, organize your file in subfolders (e.g., data)
- All file paths in your script(s) are specified relative to the folder's top level



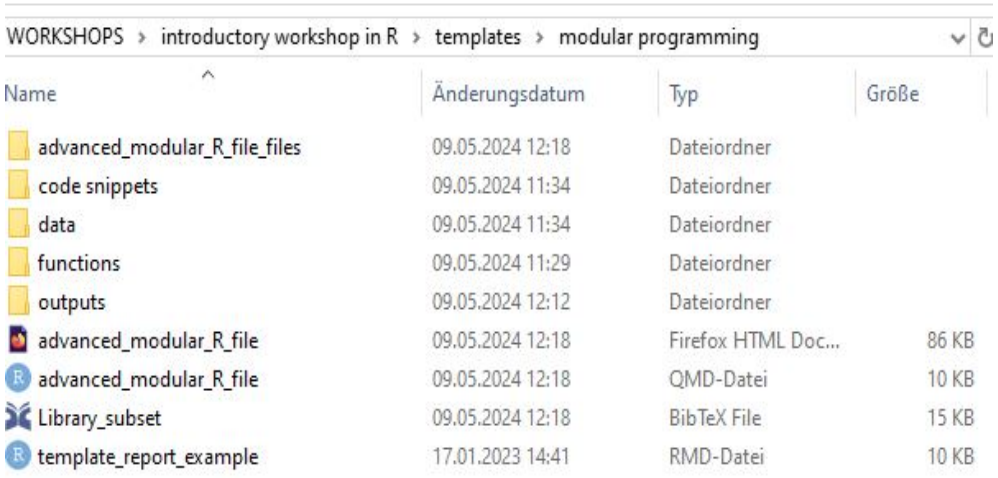

check out the folder "templates"




> click on the R Project file "templates" to open the project

# organize your program in modules II

check out the folder "additional scripts -> project template"




> click on the R Project file "templates" to open the project

- code snippets: R Code you do not need at the moment, but maybe in future
- data: your data files and **codebook** (provides information on the structure, contents, and layout of a data file)
- functions: functions you have written
- *advanced_modular_R_file_files: can be ignored (output from advanced_modular_R_file.qmd)*
- advanced_modular_R_file.qmd: an Quarto template, which allows you to create documents that serve as a neat record of your analysis combining codes and texts

# Side-Note: the lazy expert

managing many data files and multiple outputs experts use the following R functions:

| Function | Important arguments | Description |
|---|---|---|
| **Functions for folder management** | | |
| `dir()` | `path` | Returns character vector of the names of files and directories in `path` |
| `dir.create()` | `path, recursive` | Creates new directory in `path` (only the last element). If `recursive=T` all elements on the path are created. |
| **Functions for file management** | | |
| `file.path()` | `...` | Constructs a file path of character elements |
| `file.info()` | `path` | Returns character vector with information about a file in `path` |
| `file.exists()` | `path` | Returns logical value whether a file already exists in `path` |
| `file.access()` | `names, mode` | Tests files in `names` for existence (`mode=0`), execute permission (`mode=1`), writing permission (`mode=2`), read permission (`mode=4`). Returns integer with values 0 for success and −1 for failure |
| `file.rename()` | `from, to` | Renames a file in path `from` to a name in `to` |
| `file.remove()` | `path` | Deletes a file in `path` from the hard drive |
| `file.append()` | `file1, file2` | Appends contents in `file2` to `file1` |
| `file.copy()` | `from, to` | Creates a copy of a file in path `from` to path `to` |
| `basename()` | `path` | Returns the lowest level in a `path` |
| `dirname()` | `path` | Returns all but the lower level in a `path` |
| **Functions for working with compressed files** | | |
| `zip()` | `zipfile, files` | Create a zip file in path `zipfile` of `files` |
| `unzip()` | `zipfile, files` | Extracts specific `files` (all when unspecified) from a zip file in path `zipfile` |

Path arguments may usually be passed as complete or incomplete paths. In the latter case, paths are expanded to the working directory (`getwd()`). File paths may be passed in abbreviated form without the user's home directory. (`path.expand()` is used to replace a leading tilde by the user's home directory.)

Chapter 11 in Munzert, S., Rubba, C., Meißner, P., & Nyhuis, D. (2015). Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining. John Wiley & Sons.

# Data Import and Export

- Data Import and Export can be a frustrating task and can take far more time than the statistical analysis itself (sometimes about 80% of an analysis project)
- majority of the data will be given in the format of spreadsheet-like data (rectangular data)
- it is often easier to import badly formatted data than explain what a "good" format is)
- R is not particularly well suited to manipulations of large-scale data) or relational databases (! but possible)
  - more than a (few) hundred megabytes in size can cause R to run out of memory (depending on your RAM)
- in practice, you will be often faced with data corrections and updates), e.g. by a project management

# Data Import Vocabulary

```
# Reading data
data # loads specified data sets
count.fields # counts the number of fields
read.table # Reads a file in table format
read.fwf # Read a table of fixed width formatted data
load # Reload 'RData' datasets
library(foreign) # Read Data Stored by Minitab, SAS, SPSS, ...

# Files and directories
setwd, getwd # set and get current working directory
dir # names of files or directories in a directory
dirname # returns the directory name
file.path # reads path to a file from components
normalizePath # convert file paths to canonical form
file.choose # choose a file interactively
download.file # download a file from the Internet

# low-level interface to the computer's file system:
file.copy, file.create, file.remove, file.rename, dir.create, file.exists, file.info
```

# Data Import - spreadsheet-like data I

To read spreadsheet-like data you can use functions like

- read.table()
- and variations like read.csv(), ...

These functions often have the following arguments:

- fileEncoding: use latin1 for data produced in Windows, and utf-8 for data from Unix
- header: names of variables (columns) and observations (rows)
- sep: field separator vs. record separator
- quote: protecting separators appearing in strings
- na.strings: which string (or number) represents missing values?
- skip: number of lines skipped before beginning to read data
- ...

# Data Import - spreadsheet-like data II

**Example:**

```
 # if no idea how data is structured
 head(readLines("data/tramo1998etal_twins.csv"))
```

```
## [1] "ID,IDZP,GR,GES,OG,VG,CC,KU,IQ,KG"
## [2] "1,1,1,2,1913.88,1005,6.08,54.7,96,57.607"
## [3] "2,1,2,2,1684.89,963,5.73,54.2,89,58.968"
## [4] "3,2,1,2,1902.36,1035,6.22,53,87,64.184"
## [5] "4,2,2,2,1860.24,1027,5.8,52.9,87,58.514"
## [6] "5,3,1,2,2264.25,1281,7.99,57.8,101,63.958"
```

```
 # load csv2 (seperated by ",")
 dat_twins <- read.csv2("data/tramo1998etal_twins.csv", header = TRUE, sep = ",")
```

# Data Import - from other statistical softwares

if you want to import (or export) data, for example, from SPSS you could use the **haven** R package:

- SPSS: "read_sav()" reads .sav files; "write_sav()" writes .sav files.

```
# ?haven::read_spss

haven::read_sav(
  file,
  encoding = NULL,
  user_na = FALSE,
  col_select = NULL,
  skip = 0,
  n_max = Inf,
  .name_repair = "unique"
)
```

# Side-Note: naming convention

outside of the scope of the workshop are processes of data cleaning to get *tidy data*, but there is a useful function from the **janitor** R package to get the right naming convention for variables:

```
x <- data.frame(caseID = 1, DOB = 2, Other = 3)
# use camelCase variable names:
janitor::clean_names(x, "lower_camel")
```

```
##   caseId dob other
## 1      1   2     3
```

and possible convention (use one!):

- **lowerCamelCase** for functions: single word names consist of lower case letters and in names consisting of more than one word all, except the first word, are capitalized as in "colMeans" or "suppressPackageStartupMessage"
- **period.separated** for functions or parameters: all letters are lower case and multiple words are separated by a period. This naming convention is unique to R and used in many core functions such as "as.numeric" or "read.table"

see: The State of Naming Conventions in R: https://scilifelab.github.io/courses/r_programming/1703/files/Rnaming.pdf

# Data Export Vocabulary

```
# Output
print, cat # `cat' performs much less conversion
# than `print'
dput # Writes an ASCII text representation of
# an R object to a file
sink, capture.output # Evaluates its arguments with the output
# being returned as a character string or
# sent to a file.

# Writing data
write # The data (usually a matrix) are written
# to file
write.table, write.csv # converts the object to a data frame and
# prints it to a file
save # writes an external representation of
# R objects
```

# Data Export - fundamentals and example

- exporting results from R is usually a less contentious task than importing
- normally a .txt (text) or .csv file will be the most convenient interchange vehicle with "write.table" or "write.csv"
  - **foreign**::write.foreign enables the export of a data set to another statistical software (e.g. SPSS, SAS, ..)
  - **haven**::write_ also enables the export of a data set to another statistical software

the XLSX document file is the Microsoft Excel standard format file type for Excel2007 and above, which can be normally open by everyone

```
library(xlsx)
xlsx::write.xlsx2(x = dat_twins, file = "dat_twings.xlsx", sheetName = "mytwins")
```

# Data Import and Export - the R Data Format: RDS and RDATA

Save and load one object to a file

```
# Save an object to a file
saveRDS(object, file = "my_data.rds")
# Restore the object
readRDS(file = "my_data.rds")
```

Save and read multiple objects to a file

```
# Save multiple objects
save(data1, data2, file = "data.RData")
# To load the data again
load("data.RData")
```

Save your entire workspace

```
save.image(file = "my_work_space.RData")
```

see: http://www.sthda.com/english/wiki/saving-data-into-r-data-format-rds-and-rdata

# Part 3: Introduction to R - Adding Packages

*Statisticians are lazy as hell, that's why we add packages to avoid writing any code!*

# !! Smart kids let the computer do the job

But please do not pet my cat!!!

# add packages

- the R community's package development means it has the most prewritten functionality of any data analysis software

New packages are installed via

```
install.packages('package_name') # mind the quotes
```

and need to be loaded at the beginning of each session (when using them):

```
library(package_name) # no quotes necessary
```

cool kidz use functions:

```
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
usePackage("tidyverse")
```

# The (art) gallery of the most impressionistic packages I - **tidyverse**

**tidyverse** is a collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures: https://www.tidyverse.org/

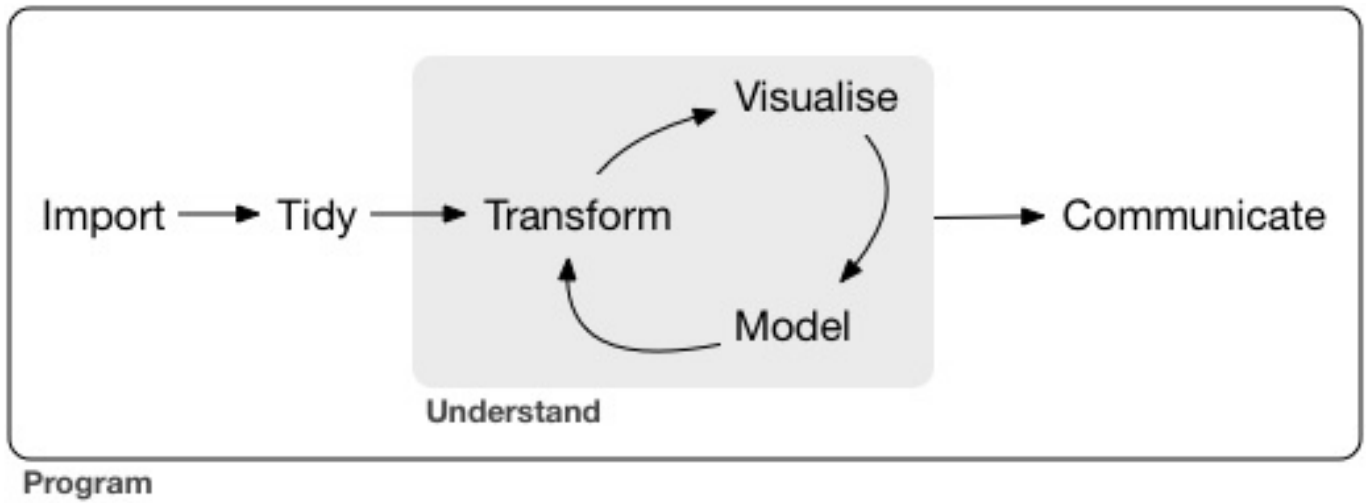

just write:

```
install.packages("tidyverse")
```

# The (art) gallery of the most impressionistic packages II - **tidyverse**

Which packages are included in tidyverse? see: https://www.tidyverse.org/packages/

Using these collection of R package you have everything to analyze the complete data analysis pipeline:




great introductory book: Wickham, H., & Grolemund, G. (2017). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media, Inc. https://r4ds.had.co.nz/

> if you confronted in the future with hughe datasets (>> 100mb) then it could be reasonable to teach yourself the R package **data.table**; e.g. watch: https://www.youtube.com/watch?v=SfrjF5YSj0Y

# The (art) gallery of the most impressionistic packages III - **psych**

Psychometric theory in one central package in R called **psych**: https://personality-project.org/r/book/

*Psychometrics is that area of psychology that specializes in how to measure what we talk and think about (focused on problems in measurement)*

What is possible in R?!

check out: Mair, P. (2018). Modern Psychometrics with R. Springer International Publishing. https://doi.org/10.1007/978-3-319-93177-7

# The (art) gallery of the most impressionistic packages IV - entering the world of hypothesis tests

**Statistical hypothesis test** is a method of statistical inference used to decide whether the data at hand sufficiently support a particular hypothesis; allows us to make probabilistic statements about population parameters.

Set of great R packages:

- **afex**: convenience functions for analyzing factorial experiments using ANOVA or mixed models (multiple wrapper functions)
- **ggstatsplot**: extension of **ggplot2** package for creating graphics with details from statistical tests included in the information-rich plots themselves
- **BayesFactor**: enables the computation of Bayes factors in standard designs, such as one- and two- sample designs, ANOVA designs, and regression (any evidence for the H0 out there?!)

# The (art) gallery of the most impressionistic packages V - entering the world of regression models

**Regression models** provide a function that describes the relationship between one or more independent variables and a response, dependent, or target variable of any type (binary, numeric, ...)

Set of great R packages:

- **lme4**: analyze grouped data and complex hierarchical structures using mixed-effects models
  - **lmerTest**: provides p-values in type I, II or III anova and summary tables for lmer model fits
- **nlme**: fits a nonlinear mixed-effects model

- **blme**: Bayesian Linear Mixed-Effects Models

great starting book: Fox, J. (2016). Applied Regression Analysis and Generalized Linear Models. SAGE Publications.

# The (art) gallery of the most impressionistic packages VI - entering the world of dynamic documents

**Dynamic analysis documents** combine code, rendered output (such as figures), and text using the **rmarkdown** package

- R Markdown allows you to create documents that serve as a neat record of your analysis
- enables reproducible research (appendix to a paper, upload it to an online repository, keep as a personal record, ...)
- Quarto file (.qmd); when you knit the Quarto file, the Markdown formatting and the R code are evaluated, and an output file (HTML, PDF, etc) is produced
- R Markdown makes use of *Markdown* syntax

great introductiory book: Xie, Y., Allaire, J. J., & Grolemund, G. (2018). R Markdown: The Definitive Guide. Chapman and Hall/CRC. https://doi.org/10.1201/9781138359444

great book to get an overview: Xie, Y. (2017). Dynamic Documents with R and knitr. Chapman and Hall/CRC, https://duhi23.github.io/Analisis-de-datos/Yihue.pdf

check out also the file "additional scripts -> rmarkdown package"

# Part 4: Amazing Applications of R

# Rriding a bycle for the first time was also difficult? wasn't it?

If you replace pain by pleasure (or better joy) you get what I mean...

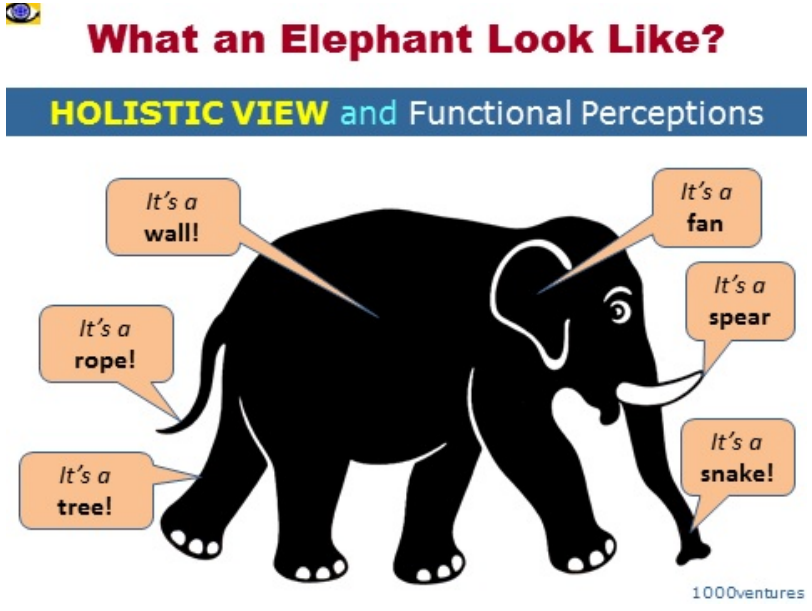
# Part 4: Amazing Applications of R - analyses sequences

*The most important skill for the applied statistician (some people call such a person "data scientist") is to learn sequences of analyses and the assumptions and relationships of different statistical models (quote by J.F.).*

# Motivation to learn the dependencies between different statistical models / frameworks I

**A story of the six blind men and the elephant.**

Six blind men were discussing exactly what they believed an elephant to be, since each had heard how strange the creature was, yet none had ever seen one before. So the blind men agreed to find an elephant and discover what the animal was really like...




from http://www.1000ventures.com/business_guide/crosscuttings/knowing_people_perceptions_elephant.html

# Motivation to learn the dependencies between different statistical models / frameworks II

**concept of Generalizability Theory:**

$$Var(\hat{d})_{\text{Total}} = Var(\hat{d})_{\text{Persons}} + Var(\hat{d})_{\text{Items}} + Var(\hat{d})_{\text{Models}} + Var(\hat{d})_{\text{Occasions}}$$

Sources of variability of results:

- persons
- items (there is a potential universe of possible items for the query of individual knowledge areas)
- statistical models
- time point of measurement

recommended book: Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). Experimental and quasi-experimental designs for generalized causal inference. Houghton, Mifflin and Company.

and article: https://www.researchgate.net/publication/227580118_Generalizability_Theory_Overview

# descriptive summary statistics I - base R functions

In the world of base R functions we can apply great functions:

```
# boxplot and summary contains identical information - 5 point summary statistics
boxplot(dat_twins$IQ)
summary(dat_twins$IQ)

# boxplot can be grouped by any non-numeric variable
boxplot(dat_twins$CC ~ dat_twins$IQ.cat)

# super useful are correlation matrices, coupled with the function pairs
cor(x = dat_twins[, c("VG", "CC", "KU", "KG", "IQ")],
              method = "pearson")
pairs(dat_twins[, c("VG", "CC", "KU", "KG", "IQ")])
```

many more...

it quite helps if you "google" the test procedure you want to apply with the letter R, e.g. "unpaired t test AND R" to get an overview of possible useful summary statistics

# descriptive summary statistics II - tidyverse universe

In the world of **tidyverse** we can apply even greater functions (! advanced):

```
 dat_twins %>%
   group_by(IDZP) %>%
   summarise(M=mean(IQ), SD=sd(IQ), difference=diff(IQ)) %>%
   print(n=3)
```

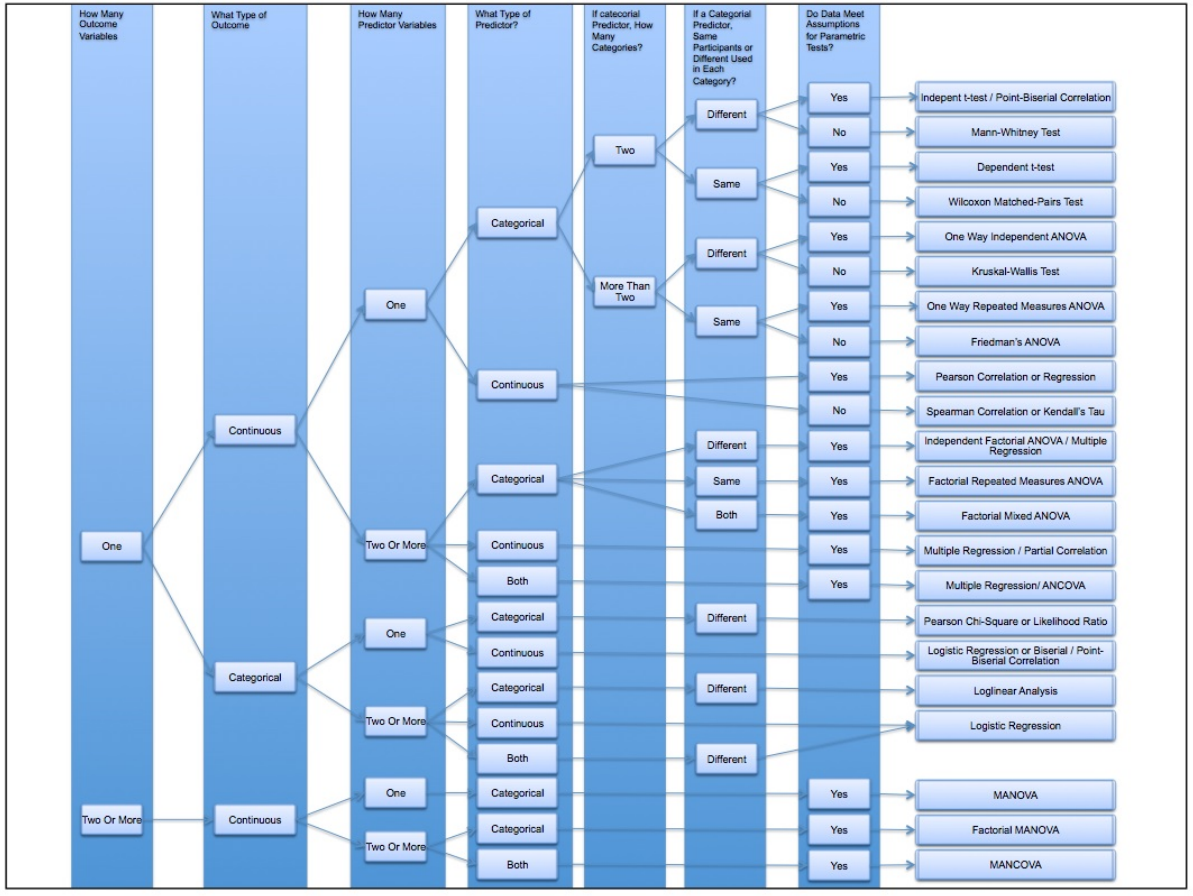
```
## # A tibble: 10 × 4
##    IDZP     M    SD difference
##   <int> <dbl> <dbl>      <int>
## 1     1  92.5  4.95         -7
## 2     2  87    0             0
## 3     3 102    1.41          2
## # ℹ 7 more rows
```

- by using group_by() the unit of analysis from the complete data set to individual groups is changed
- together group_by() and summarise() provide one of the tools that you'll use most commonly when working with grouped summaries
- the "%>% are called pipes, which link multiple operations from top-to-bottom

great introductory book: Wickham, H., & Grolemund, G. (2017). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media, Inc. https://r4ds.had.co.nz/

# hypothesis tests I - decision tree

Which hypothesis test should I use?




see: https://www.utwente.nl/en/bms/m-store/step-by-step-guide/statistical-tests/choice-test-2/

# hypothesis tests II - in R

Which hypothesis test are implemented in R?

```
 apropos(what = "\\.test$") # Find Objects by (Partial) Name
```

```
##  [1] "agostino.test"         "ansari.test"           "anscombe.test"
##  [4] "bartlett.test"         "binom.test"            "bonett.test"
##  [7] "Box.test"              "chisq.test"            "chisq.test"
## [10] "cor.test"              "fisher.test"           "fisher.test"
## [13] "fligner.test"          "friedman.test"         "jarque.test"
## [16] "kruskal.test"          "ks.test"               "mantelhaen.test"
## [19] "mauchly.test"          "mcnemar.test"          "mood.test"
## [22] "oneway.test"           "pairwise.prop.test"    "pairwise.t.test"
## [25] "pairwise.wilcox.test" "poisson.test"          "power.anova.test"
## [28] "power.prop.test"       "power.t.test"          "PP.test"
## [31] "prop.test"             "prop.trend.test"       "quade.test"
## [34] "shapiro.test"          "t.test"                "var.test"
## [37] "wilcox.test"
```

```
 # library(afex)
 apropos("^aov") # ANOVAs
```

```
## [1] "aov"     "aov_4"   "aov_car" "aov_ez"
```

```
 # many more!
```

# hypothesis tests III - example t-Tests

**in general**: The goal of hypothesis testing is to test a hypothesis about the population using a sample; there are two hypotheses:

- Null hypothesis $H_0$: hypothesis to be tested (contains "critical" assumption).
- Alternative hypothesis $H_1$: Opposite to the null hypothesis

! Decision rule of classical significance tests (frequentist perspective!): Reject the $H_0$ only if it is very unlikely to be true.

**t-Test**: statistical hypothesis test in which the test statistic follows a Student's t-distribution under the null hypothesis; most common application is to test whether the means of two populations are different

for relative equal variances:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{S_p * \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

# hypothesis tests IV - always consider non-parametric tests

**parametric tests:**

- information about the distribution of the population is known and is based on a fixed set of parameters

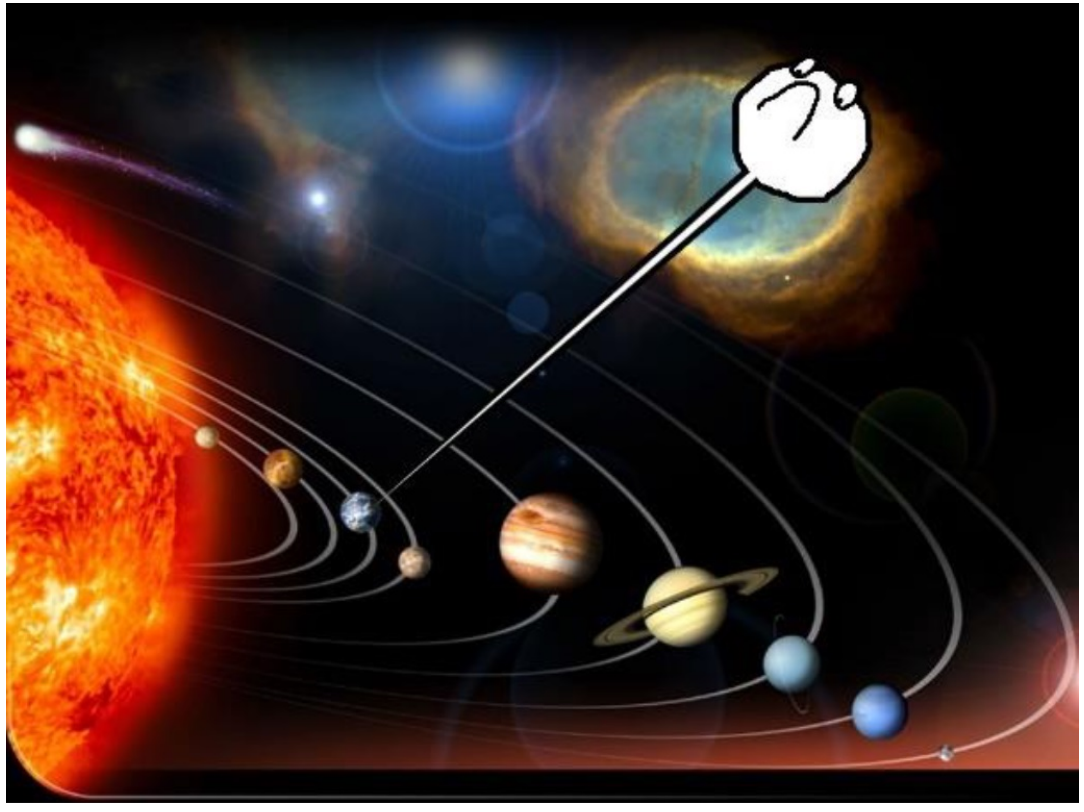
**non-parametric tests:**

- information about the distribution of a population is unknown, and the parameters are not fixed
  - no requirement for any distribution of the population in the non-parametric test (also known as distribution-free testing)
- usually performed when the independent variables are non-metric or central assumptions are violated (e.g. normality)

| Data Setup | Parametric test | Non-Parametric test |
|---|---|---|
| 1 Variable<br>2 Categories<br>Between Subjects | independent t-test | Mann-Whitney U test |
| 1 Variable<br>2 Categories<br>Within-Subjects | paired t-test | Wilcoxon Signed Rank Test |
| 1 Variable<br>>2 Categories<br>Between Subjects | One-way ANOVA | Kruskal Wallis Test |
| 1 Variable<br>>2 Categories<br>Within Subjects | repeated measures ANOVA | Friedman test<br>Mood's median test |
| 1 Variable<br>(Correlation) | Pearson's r | Spearman's ρ (rho) |

# jumping to outer space for a second!

*Embrace yourself we will enter the top floor of crazy tower for a second!*



# Bayesian Hypothesis Testing

**frequentist hypothesis testing:**

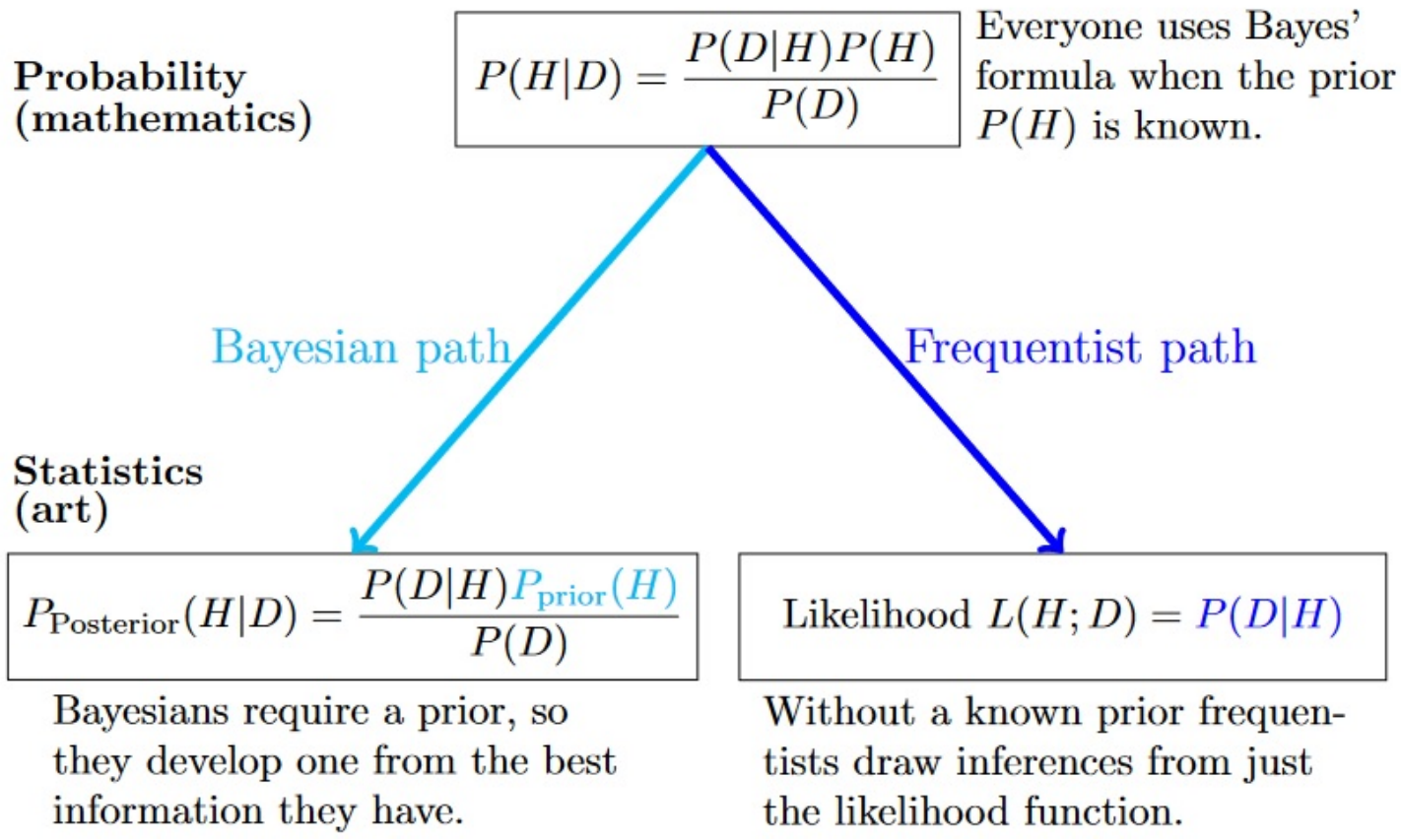
- focuses on null hypothesis significance testing (*NHST*), whereby p-values can only reject the null hypothesis
  - whether to reject a hypothesis, accept it as true (! categorical decision) or to withhold judgement because no decision can be made (if we assuming insufficient power)
  - relative strength of an effect can be made by *effect sizes*

**bayesian hypothesis testing:**

- Bayes factor can state evidence for both the null and the alternative hypothesis, making confirmation of hypotheses possible
  - Bayes factor is the ratio of the likelihood of one particular hypothesis to the likelihood of another. It can be interpreted as a measure of the strength of evidence in favor of one theory among two competing theories

> you could use the statistic program JASP to compute Baysian Hypothesis Tests: https://jasp-stats.org/

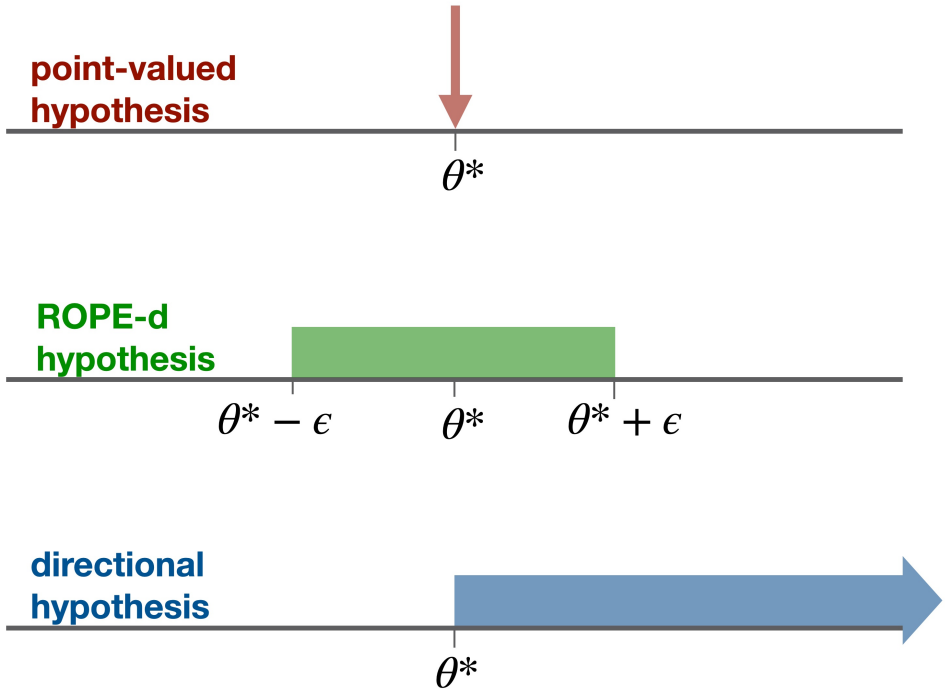
# Bayesian vs. Frequentist Hypothesis Testing




slide from MIT: https://math.mit.edu/~dav/05.dir/class17-slides-all.pdf

# Fancy ways to formulate informative hypotheses

Three common types of hypotheses anchored to a point-value of interest of a parameter (! smallest effect size of interest):




see in more detail: book online: https://michael-franke.github.io/intro-data-analysis/

article: Lakens, D., Scheel, A. M., & Isager, P. M. (2018). Equivalence Testing for Psychological Research: A Tutorial. Advances in Methods and Practices in Psychological Science, 1(2), 259–269. https://doi.org/10.1177/2515245918770963

# hypothesis tests V - steps of sequences

1 get an overview of your data:

```
data("sleep")
DT::datatable(sleep, options = list(pageLength = 5))
```

Show 5 entries Search:

| | extra | group | ID |
|---|---|---|---|
| 1 | 0.7 | 1 | 1 |
| 2 | -1.6 | 1 | 2 |
| 3 | -0.2 | 1 | 3 |
| 4 | -1.2 | 1 | 4 |
| 5 | -0.1 | 1 | 5 |

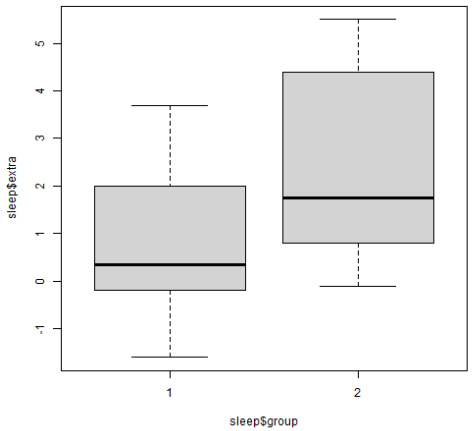
Showing 1 to 5 of 20 entries Previous 1 2 3 4 Next

> paired sample t-test is needed; consist of a sample of matched pairs of similar units (e.g. using Propensity Score Matching), or one group of units that has been tested twice

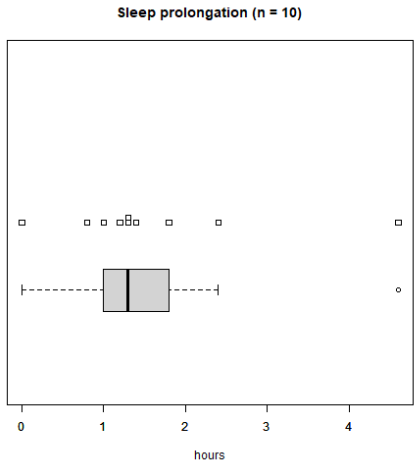
# hypothesis tests VI - steps of sequences

2 compute summary statistics / visualize your data I - simple plots:

```
## grouped boxplot
boxplot(sleep$extra ~ sleep$group)
```




```
stripchart(sleep1, method = "stack", xlab = "
           main = "Sleep prolongation (n = 10
boxplot(sleep1, horizontal = TRUE, add = TRUE
        at = .6, pars = list(boxwex = 0.5, st
```




```
## compute mean deviations:
sleep1 <- with(sleep, extra[group == 2] - ext
summary(sleep1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.05    1.30    1.58    1.70    4.60
```

# hypothesis tests VI - steps of sequences

2 compute summary statistics / visualize your data II - more sophisticated plots (**ggplot2**) I:

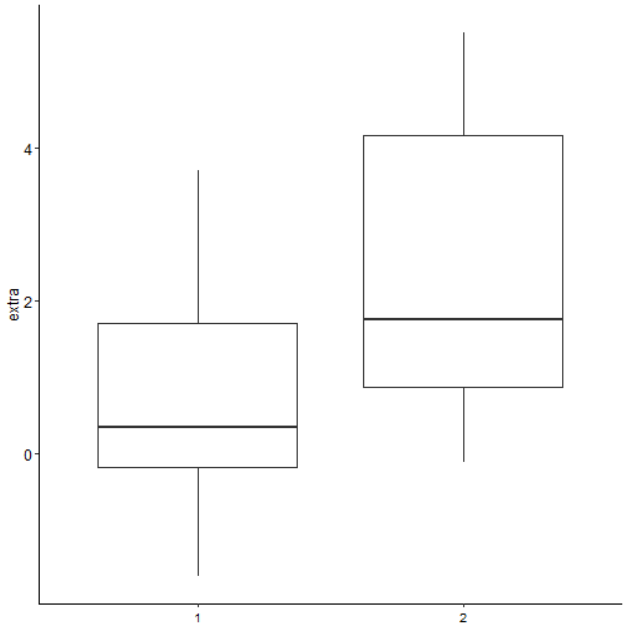
template for adjusting the format of ggplot2 figures to APA style:

```
ggplot_theme <- theme(axis.title.x = element_blank(),
                      axis.title.y = element_text(size=12),
                      axis.text.x = element_text(size=10,hjust=0.5,vjust=0.5,face="plain",
                                                 colour = "black"),
                      axis.text.y = element_text(size=12,face="plain", colour = "black"),
                      panel.border = element_blank(),
                      axis.line = element_line(colour = "black"),
                      panel.grid.major = element_blank(),
                      panel.grid.minor = element_blank(),
                      panel.background = element_blank(),
                      legend.position="none")  # no legend
```

# hypothesis tests VI - steps of sequences

2 compute summary statistics / visualize your data II - more sophisticated plots (**ggplot2**) II:

```
# grouped boxplot
ggplot(sleep, aes(x=group, y=extra)) +
    geom_boxplot() + ggplot_theme
```



# hypothesis tests VI - steps of sequences

2 compute summary statistics / visualize your data II - more sophisticated plots (**ggstatsplot**) I:

Using the type argument you can specify the statistical approach:

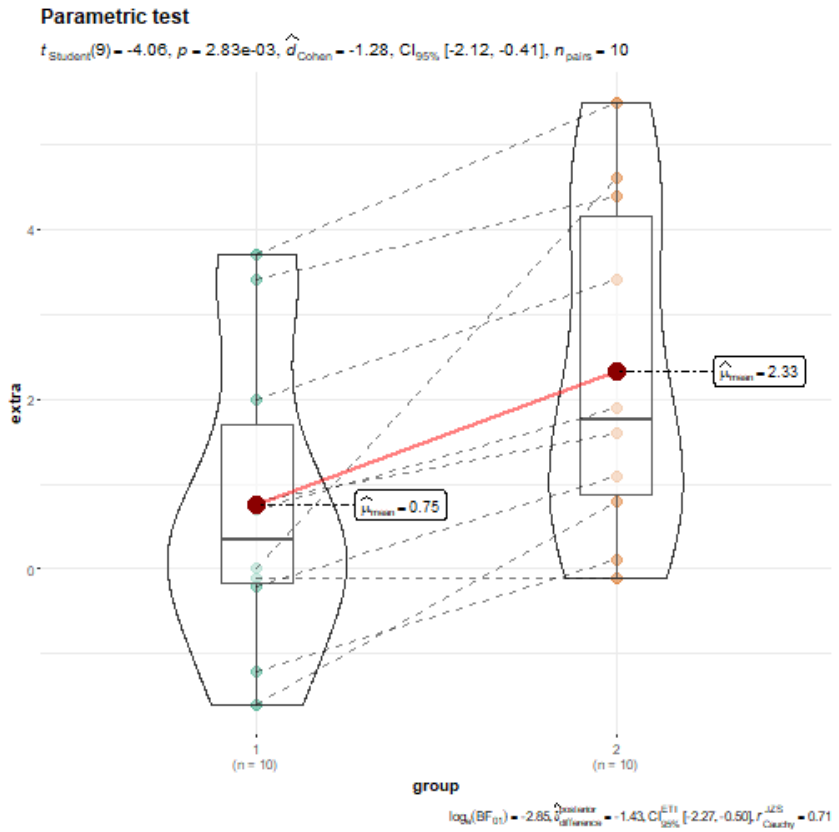
- "parametric"
- "nonparametric"
- "robust"
- "bayes"

```
## parametric t-test
p1 <- ggwithinstats(
  data = sleep,
  x = group,
  y = extra,
  type = "p",
  effsize.type = "d",
  conf.level = 0.95,
  title = "Parametric test"
)
```

# hypothesis tests VI - steps of sequences

2 compute summary statistics / visualize your data II - more sophisticated plots (**ggstatsplot**) II:

```
## parametric t-test
p1
```

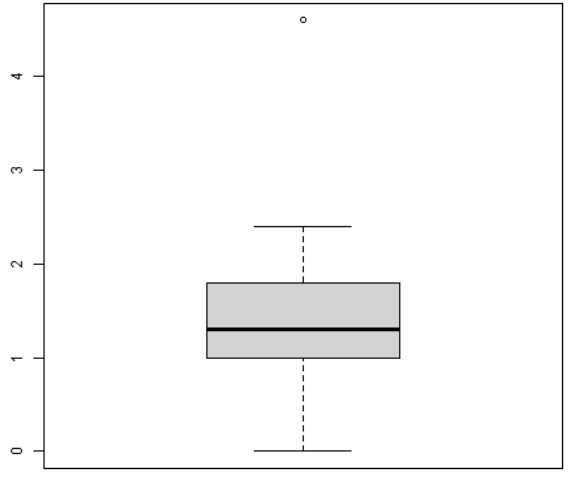


# hypothesis tests VII - steps of sequences

3 check the assumptions of the paired sample t-test:

1. Independence: Each observation should be independent of every other observation.
2. Normality: The differences between the pairs should be approximately normally distributed.
    - No Extreme Outliers: There should be no extreme outliers in the differences.
    - Remark: quite often in practice, if $n >> 30$ this assumption is not checked

```
## check outlier assumption
boxplot(sleep1) # to check outlier assumption
```



# hypothesis tests VII - steps of sequences

3 check the assumptions of the paired sample t-test:

applying Kolmogorov-Smirnov OR Shapiro-Wilk-Test to check normality; sample size sensitive: tests become more sensitive to minimal deviations from the normal distribution as sample size increases

Use Shapiro-Wilk normality test as described at: Normality Test in R.

- Null hypothesis: the data are normally distributed
- Alternative hypothesis: the data are not normally distributed

```
# Shapiro-Wilk normality test for the differences
shapiro.test(sleep1)
```

```
##
##     Shapiro-Wilk normality test
##
## data:  sleep1
## W = 0.82987, p-value = 0.03334
```

# hypothesis tests VII - steps of sequences

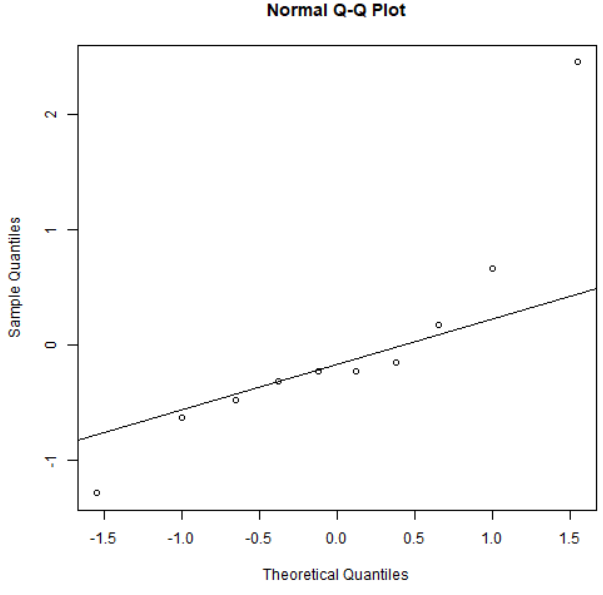
3 check the assumptions of the paired sample t-test:

because of the sample size sensitivity it is often better to visually check the assumption of normality:

drawing **Q-Q Plot**:

```
sleep1_sc <- scale(sleep1) # z standardize mean differences

qqnorm(sleep1_sc) # QQ Plot
qqline(sleep1_sc) # vertical line
```



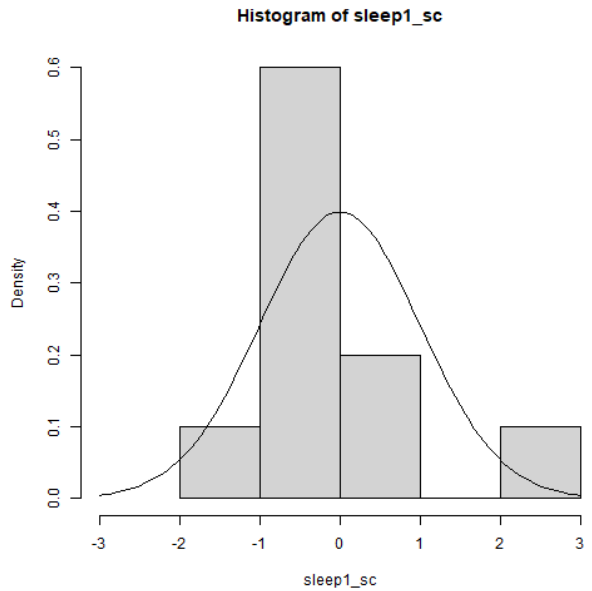
# hypothesis tests VII - steps of sequences

3 check the assumptions of the paired sample t-test:

because of the sample size sensitivity it is often better to visually check the assumption of normality:

drawing **Histogram** and overlay normal distribution:

```
hist(sleep1_sc, freq = FALSE, xlim = c(-3,3))
x<-seq(-4,+4,by=0.02)
curve(dnorm(x), add=TRUE)
```



# hypothesis tests VII - steps of sequences

4 compute paired sample t-test:

```
## Formula interface
t.test(extra ~ group, data = sleep, paired = TRUE)
```

```
##
##     Paired t-test
##
## data:  extra by group
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -2.4598858 -0.7001142
## sample estimates:
## mean difference
##           -1.58
```

outputs:

- t is the t-test statistic value
- df is the degrees of freedom
- p-value is the significance level of the t-test
- conf.int is the confidence interval (conf.int) of the mean differences at 95%
- sample estimates is the mean differences between pairs

# hypothesis tests VII - steps of sequences

4 compute paired sample t-test:

compute the Bayes factor:

```
 ## Traditional interface
 with(sleep, ttestBF(extra[group == 1], extra[group == 2], paired=TRUE))
```

```
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 : 17.25888 ±0%
##
## Against denominator:
##   Null, mu = 0
## ---
## Bayes factor type: BFoneSample, JZS
```

> Bayse factor of 17.25888

# hypothesis tests VII - steps of sequences

4 compute paired sample t-test:

evaluate the Bayes factor:

> Bayse factor of 17.25888

Jeffreys' scale of evidence is a subdivision of the possible values of the Bayes factor into categories (or grades).

...

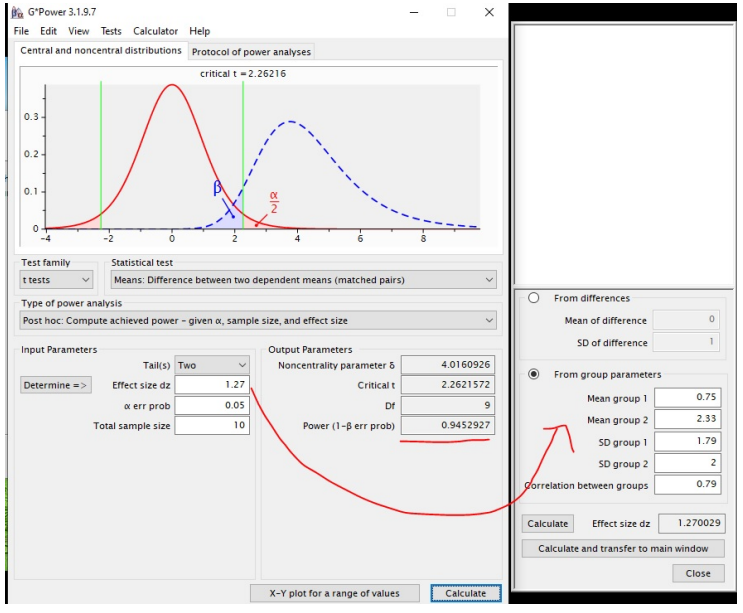
Kass and Raftery's scale.

| $B_{1,2}$ | $2 \ln(B_{1,2})$ | Grades of evidence |
|---|---|---|
| 1 to 3 | 0 to 2 | Barely worth mentioning |
| 3 to 20 | 2 to 6 | Positive |
| 20 to 150 | 6 to 10 | Strong |
| > 150 | > 10 | Very strong |

copied from: https://www.statlect.com/fundamentals-of-statistics/Jeffreys-scale

# hypothesis tests VIII - steps of sequences - missing parts

We have forgot to run a **power analysis**




if you want to install / use / read more about G*Power: https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower

# hypothesis tests VIII - steps of sequences - missing parts

We have forgot to compute a appropriate **effect size**

The standardized mean difference effect size for within-subjects designs is referred to as Cohen's $d_z$, which can be computed as:

```
## by hand:
means <- with(sleep, tapply(extra, group, mea
md <- means[1]-means[2] # get mean difference
sleep_wide <- cbind(sleep$extra[sleep$group==

(md / sqrt(sum(((sleep_wide[,1] - sleep_wide[
```

```
##         1
## -1.284558
```

```
## by package "rstatix"
sleep %>% cohens_d(extra ~ group, paired = TR
```

```
## # A tibble: 1 × 7
##   .y.   group1 group2 effsize    n1    n2 magnitud
## * <chr> <chr>  <chr>     <dbl> <int> <int> <ord>
## 1 extra 1      2        -1.28    10    10 large
```

see: Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and ANOVAs. Frontiers in Psychology, 4. https://www.frontiersin.org/articles/10.3389/fpsyg.2013.00863

# multiple linear regression - assumptions

Assumption of a linear regression (A. 8 is normally never taught!):

1. Number of predictors is smaller than number of cases (identifiability)
2. Assumption of linearity
3. The expected value of the errors is 0: $E(\epsilon_i) = 0$
4. The variances of the errors are equal (homoscedasticity): $Var(\epsilon_i) = \sigma^2$
5. No correlation between the errors: $Cov(\epsilon_i, \epsilon_j) = 0$, $foralli \neq j$
6. No exact multicollinearity of predictors
7. Normal distribution of errors: $\epsilon_i \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2)$
8. Reliability: The predictors are measured without error ($Rel(x_j) = 1$).

If assumptions 2-6 apply, the model is **BLUE** (smallest variance, linear, not biased).

> All models are wrong, but some are useful" - George Box. The aphorism acknowledges that statistical models always fall short of the complexities of reality but can still be useful nonetheless.

# Part 4: Amazing Applications of R - Bibliometrix

*Why read literature if you can use R?!*

# Bibliometrix - types

**systematic reviews**: search available literature for evidence with which to address a research question

**meta-analyses**: quantitatively assess statistical evidence found through systematic reviews -> dependent on good systematic review

**goals**:

- integrate past literature
- critically analyse the existing literature
- identify issues central to a field

**center often on one single area**:

- summarize / list findings of individual primary studies
  - .. methods used to carry out research
  - .. theories meant to explain the same or related phenomena
  - .. practices programs, treatments, .. being used in applied context

# Bibliometrix - search strategies

different modes should be used concurrently and used search strategies should be made explicit

**Footnote Chasing**
*Cooper 1985*
References in review papers written by others
References in books by others
References in nonreview papers from journals you subscribe to
References in nonreview papers you browsed through at the library
Topical bibliographies compiled by others
*Mann 1993*
Searches through published bibliographies (including sets of footnotes in relevant subject documents)
Related Records searches

**Consultation**
*Cooper 1985*
Communication with people who typically share information with you
Informal conversations at conferences or with students
Formal requests of scholars you knew were active in the field (e.g., solicitation letters)
Comments from readers/reviewers of past work
General requests to government agencies
*Mann 1993*
Searches through people sources (whether by verbal contact, E-mail, electronic bulletin board, letters, etc.)

**Searches in Subject Indexes**
*Cooper 1985*
Computer search of abstract data bases (e.g., *ERIC*, *Psychological Abstracts*)
Manual search of abstract data bases
*Mann 1993*
Controlled-vocabulary searches in manual or printed sources
Key word searches in manual or printed sources
Computer searches—which can be done by subject heading, classification number, key word. . .

**Browsing**
*Cooper 1985*
Browsing through library shelves
*Mann 1993*
Systematic browsing

**Citation Searches**
*Cooper 1985*
Manual search of a citation index
Computer search of a citation index (e.g., *SSCI*)
*Mann 1993*
Citation searches in printed sources
Computer searches by citation

# Bibliometrix - main concern of literature retrieval

return all of the studies relevant to the review (‘recall’) without retrieving irrelevant studies (‘precision’)

**Recall**:

- expresses (as a percentage) the ratio of relevant documents retrieved to all those in a collection that should be retrieved
- pure fiction: if we could identify all existing relevant documents in order to count them, we could retrieve them all, and so recall would always be 100 percent

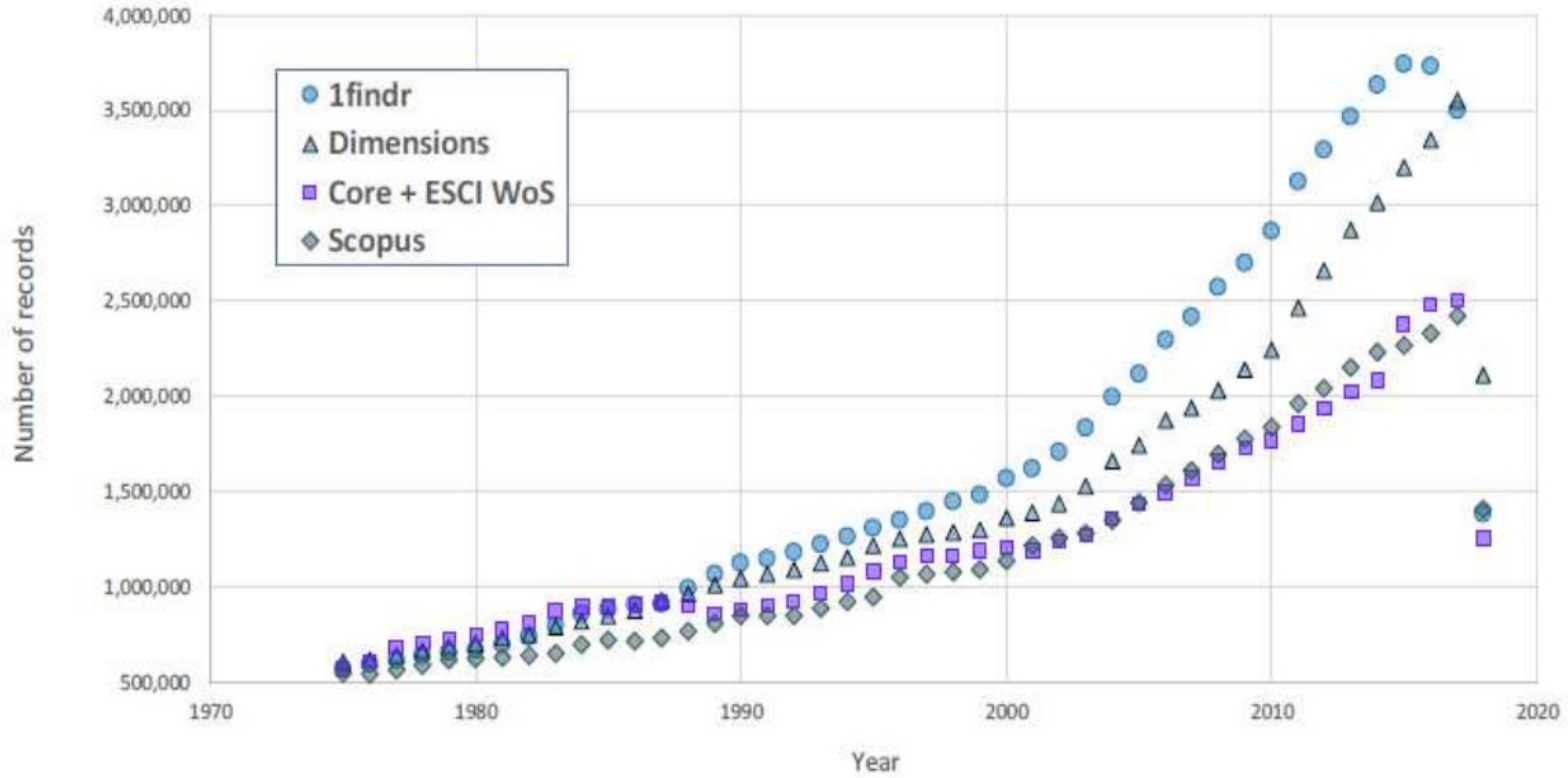
**Precision**:

- expresses (as a percentage) the ratio of documents retrieved and judged relevant to all those actually retrieved
- measures how many irrelevant documents-false positives-one must go through to find the true positives or hits

-> trade-off: high recall leads to degrading precision

# Bibliometrix - main concern of literature retrieval II

**Figure 8: Articles indexed from academic & scientific journals – 1findr, Dimensions, Core + ESCI WoS and Scopus, 1975-2018 (Courtesy of Eric Archambault)**




see: https://www.stm-assoc.org/2018_10_04_STM_Report_2018.pdf

# Bibliometrix - main concern of literature retrieval III

**solution: need for high recall !!!**

- „there is no way of ascertaining whether the set of located studies is representative of the full set of existing studies on the topic, the best protection against an unrepresentative set is to locate as many of the existing studies as is possible“
- „The point is to avoid missing a useful paper that lies outside one’s regular purview, thereby ensuring that one’s habitual channels of communication will not bias the results of studies obtained by the search… history of science and scholarship is full of examples of mutually relevant specialties that were unaware of each other for years because their members construed their own literatures too narrowly and failed to ask what other researchers did“

recommended book: Cooper, Hedges und Valentine (2009): The handbook of research synthesis and meta-analysis. Second Edition.

# Bibliometrix - traditional methods

Traditional methods often include (from my perspective) manual literature searches in arbitrary databases without clear inclusion criteria. Results of the research are often gathered in Excel.

however, there are immensely important procedures in the traditional approach:


How to use Google Scholar: https://www.wur.nl/en/article/How-to-use-Google-Scholar.htm

# Bibliometrix - definition, databases

The branch of library science concerned with the application of mathematical and statistical analysis to bibliography; the statistical analysis of books, articles, or other publications the use of statistical methods to analyse books, articles and other publications by means of

- simple descriptive measures
- complex indices
- network theory

there are three principal databases (increasing number of sources, but lower data quality):

- Web of Science: www.webofknowledge.com
- Scopus: https://www.scopus.com/
- Google Scholar: https://scholar.google.com/

# Bibliometrix - working process

1. download found sources as plain text from www.webofknowledge.com (when multiples times collect single txt-files with unique name in a folder)
2. use the ShinyApp of the R-package **bibliometrix** to check the found sources for possible errors and visualize your sources
3. use the R-package **litsearchr** to identify further search terms
4. use the R-package **scholar** to further investigate the publications of key authors

# Bibliometrix - bibliometrix package

most advanced is **bibliometrix**, which allows to compute:

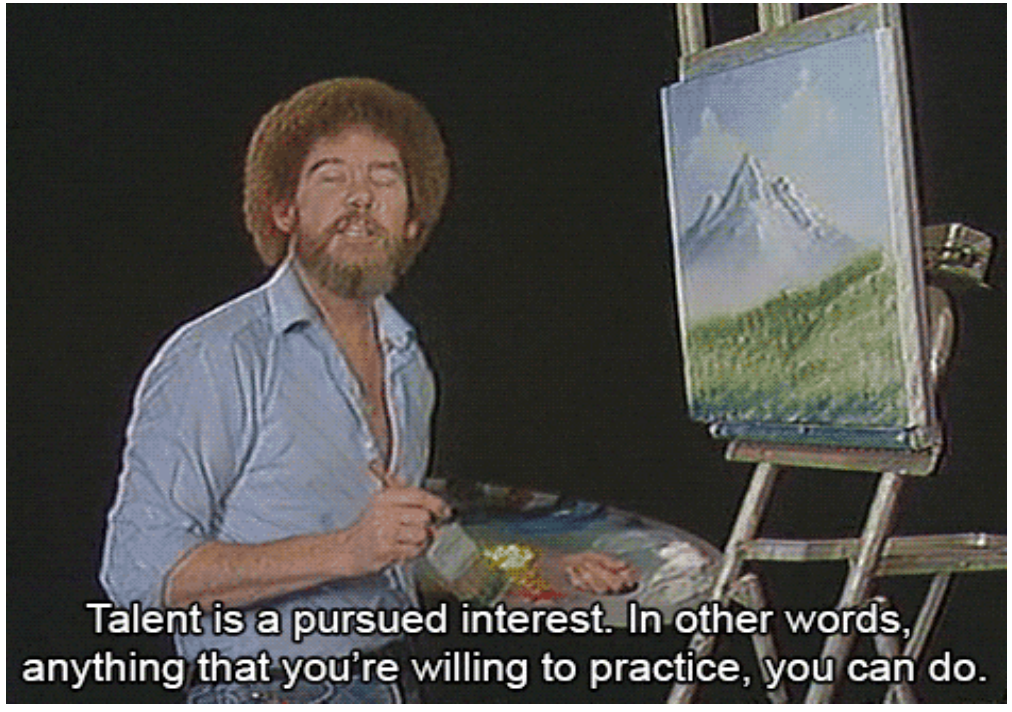
- descriptive statistics
- Bibliographic coupling: occurs when two works reference a common third work in their bibliographies (helpful to detect connections between research groups, ...)
- Co-citation: frequency with which two documents are cited together by other documents (helpful in detecting a shift in paradigms and schools of thought - using betweenness, ...)
- ...

```
## to run shiny app:
library(bibliometrix)
biblioshiny()
```

also check out the webpage: https://www.bibliometrix.org/home/

# Who wants to learn statistics and programming now? - Thank you for your attention :-)!

The final slide is reserved for Bob Ross!




To relax you can watch Bob Ross motivating painting videos: https://www.youtube.com/watch?v=2lBQUqzcbEg