R-Course: Day 1

Lukas Mödl, Matthias Becher, Erin Sprünken

Institut für Biometrie und Klinische Epidemiologie

Charité - Universitätsmedizin Berlin, Berlin

erin-dirk.spruenken@charite.de

November 9, 2022

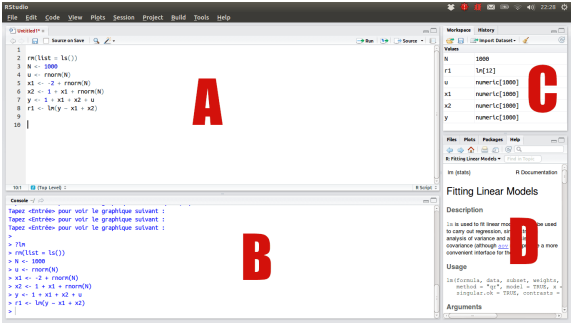
Motivation

- Statistical Analyses
- Possibilities compared to SPSS, SAS, Stata
- Open Source

Installation

- R (Software)
- RStudio (GUI = Graphical User Interface)
- Alternatives (but not recommended by us): xcode, Visual Studio, Texteditor

Interface of RStudio



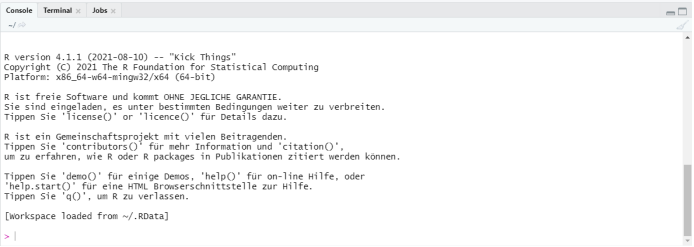
A Script window

B Console

C Workspace, Data

D Plots, Help, Filebrowser, Packet manager

Console and Pocket Calculator



- Only „Console“ relevant
- We can write commands and execute them with enter
- Example pocket calculator:
  - 3 + 2
  - 5 * 4
  - 20 - 8
  - 12 / 16

Errors, Warnings, NA, NaN, Inf

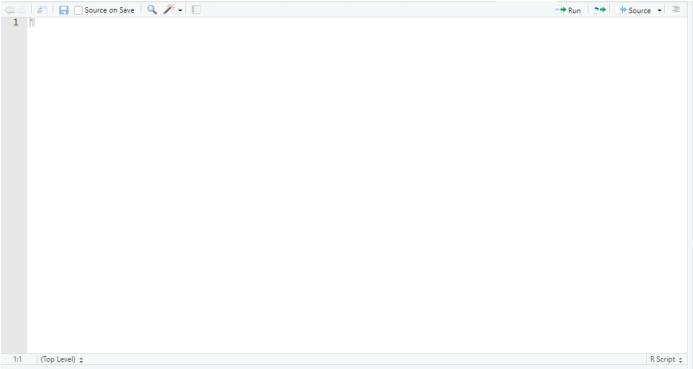
What happens if we write `5!` ?
R gives us an error. It is important that we know what to tell R. Thus: `factorial(5)`.

- 3 / 0 $\Rightarrow$ `Inf`; R solves this numerically and ends up with a limit (here `Inf` $= \infty$).
- 0/0 $\Rightarrow$ `NaN`; stands for „Not a Number“
- NA stands for "Not Available", i.e. a missing value. The main reason for this to occur (aside from raw data) is, if we conduct operations and don't tell R how to handle missing data.

Mathematical Expressions

- `factorial(x)` $= x!$
- `exp(x)` $= e^x$
- `log(x)` $= log(x)$
- `sqrt(x)` $= \sqrt{x}$
- `abs(x)` $= |x|$
- `x^n` $= x^n$

Script window



- It is inconvenient to write every command line by line into the console
- Here, we can write as many commands as we want after another and execute whole blocks
- On the technical side, R just hands over the executed lines from the script to the console line by line
- Scripts can be saved and reused

Assigning variables

Often, we want to save results or data for later use. To do so, we must assign variables. Historically, the left-pointing arrow is used in R: `<-`. The common = works just as well. Variables are case-sensitive!

- `x <- 3.1415`
- `b <- 4`
- `B = 5`
- `d <- factorial(5)`

Other

R offers several usefule functions. Two specifically useful are `?` and `rm()`

- If we want to know how a functions works, we can type the question mark followed by the name of the function to call the documentation, e.g.: `?rm`
- If we do so, we get the information about `rm()`: We can write objects into the parentheses that we want to remove. From the former slide we have `x`, `b`, `B` and `d`
- We want to remove `B` and `b`, thus: `rm(B, b)`

Datatypes

What types of data exist in R?

- Numeric ⇔ Numbers, `is.numeric()`
- Character ⇔ Letters/Words, `is.character()`
- Factor ⇔ Categorical Variables, `is.factor()`
- Date ⇔ Date/Time
- Logical ⇔ True/False, `is.logical()`

Vector

R is a vector-language: Most of the data constructions are types or expansions of a vector.
A vector in R can only contain a single datatype.

- `c()`
- `seq()`
- `rep()`
- `is.vector()`

Matrix

A matrix is a chaining of vectors (side-by-side, row-by-row).
A matrix in R can only contain a single datatype.

- `matrix()`
- `cbind()`
- `rbind()`
- `is.matrix()`

Algebraic operations

It is simple to calculate with vectors and matrices in R. By default, R computes everything elementwise. What do we get using the following command?

```
c(3.1415, 5, 1, 2/3) * seq(1, 8, 2)
```

R computes as follows:

$$\begin{pmatrix} 3.1415 \\ 5 \\ 1 \\ \frac{2}{3} \end{pmatrix} * \underbrace{\begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix}}_{\text{seq(1, 8, 2) = c(1, 3, 5, 7)}} = \begin{pmatrix} 3.1415 \cdot 1 \\ 5 \cdot 3 \\ 1 \cdot 5 \\ \frac{2}{3} \cdot 7 \end{pmatrix}$$

List

Maybe we want to store different datatypes in a vector, what can we do now?
Answer: List.
Lists are generalized forms of the classical vector, as elements of a list are not restricted to the same datatype. Lists are extremely flexible and can be nested into each other, i.e. a list can contain lists.

- `list()`
- `c()`
- `$`

Data Frame

However, lists can become confusing. A special type of a list is the Data Frame. Visually, it looks just like a matrix, but the Data Frame is allowed to store different datatypes in different columns. However, a single column must contain data from only one datatype.

- `data.frame()`
- `rbind()`
- `cbind()`
- `$`

Saving

Typically, we don't work on something only once. For this it seems useful that we save data. R offers a file format to save data for later use: `*.RData`

- `save()`
- `write.table()`
- `write.csv()`