University of Zurich[UZH]

Epidemiology, Biostatistics
and Prevention Institute,
Biostatistics Department
www.biostat.uzh.ch

Statistical Programming in the Clinical Sciences

Get to know R

BSc. Klaus Steigmiller
PD Dr. Ulrike Held
Dr. Stefanie von Felten
Dr. Eva Furrer

Version 0.8 of November 10, 2019

Contents

1 Why R

R is a progamming language for statistical computing. It provides an environment for a wide variety of statistical analysis and graphics.

There are actually many reasons to use R. First of all, R is free to use with **open source code** and it runs on every operating system (Windows, Mac, Linux). Besides high-quality statistics, R also provides tools to produce all kinds of graphics that can be personalized to an arbitrary degree and can fulfill all demands for a publication.
Moreover, R increases the **reproducibility** of research findings since R code can be shared and rerun by other researchers and the exact same statistical analysis can be recalculated. RStudio offers tools that make this process easy. As a result, research becomes a transparent process.
Once developed, any R code can also easily be adapted or **extended** for other statistical analysis projects. Many other statistical softwares lack the range of possibilities that R is providing. It is very flexible and extensible. Additionally, R is compatible/connectable to other languages and has the reputation of beeing stable and reliable.

R is **easy to learn** by its **intuitive programming** concepts. At first sight, non-programmers need to get used to the idea of statistical programming, but our experience has shown that there is a steep learning curve. Moreover, the R and RStudio communities are quite large and very active. This community is not only working on the extension of the functionality of R, but is also helpful in supporting beginners by answering questions.

2 Aim of this introduction

There are a lot of introductions to R and RStudio availabale on the internet. For beginners, it is somehow difficult to get started and understand the instructions of those introductions. Therefore, in the following section a "get to know R and RStudio" and an explanation of the basic concepts is provided. This should get you in a position to understand existing tutorials more easily. At the end of this introduction a list of some openly accessible tutorials is given.

You should be aware that there are tutorials for beginners as well as for advanced users. If a tutorial is too difficult just choose an easier one.

In this guide, R together with the editor RStudio is introduced. To be precise, R is the actual programming language, but the user is writing the code in an editor, e.g. RStudio. Running the code means that the editor sends the code behind the scene to R which then sends the results back to the editor. To work more efficiently with RStudio we will provide also some shortcuts. Since

these shortcuts are slightly different for Windows and Mac, they are written in separate format, e.g. "(**Windows**: <ctrl> + <shift> + n / **Mac**: <cmd> + <shift> + n)"

3 First start

If you open RStudio for the first time you will see a collection of panels with different tabs as shown in Figure 1.

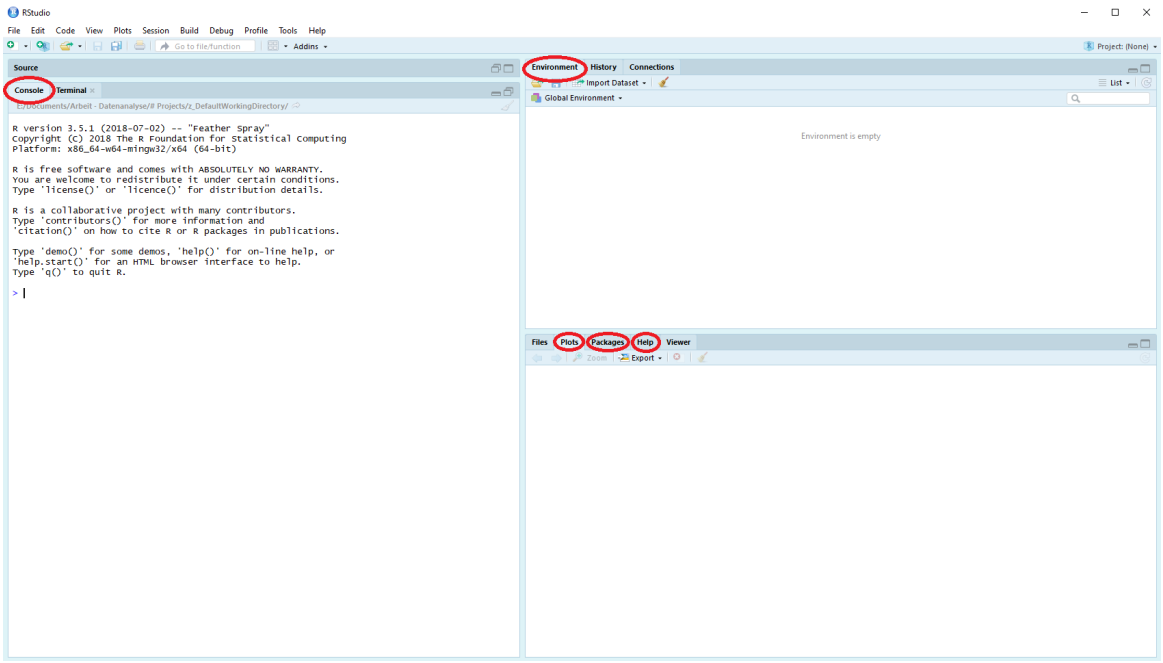

Figure 1: Default working environment.

Every tab has its own purpose:

Console
: Here you can see the output / results of your calculations. The console can also be used as an advanced calculator, entered commands will be executed directly. But this is not how you should use it!

Environment
: While programming, you read in datasets, produce new variables and handle various kinds of objects. These objects will be listed in the Environment, which is sometimes also called Workspace Environment.

Plots/Packages/Help
: In these tabs plots appear, you can get help on functions or you can handle R-packages. The general concept of packages will be introduced further below.

Script editor
: At the first start of RStudio the panel for the script editor is not shown. You can open this panel by clicking on *"File"* → *"New File"* → *"R Script"* (<ctrl> + <shift> + n / <cmd> + <shift> + n). In the upper left corner, the new panel for scripts opens and you see an empty script file named "Untitled1", see Figure 2.

In the script editor you can collect all commands and run them as a whole. Moreover, script files can be saved for later use. Note that the commands in the script file will not be executed automatically. The execution starts when the commands are sent to R. How this is done is shown further down in this introduction.
R is a so-called script language, i.e. every command in the script can be executed one after the other by a user. This is also very helpful in creating the analytical R code step by step.

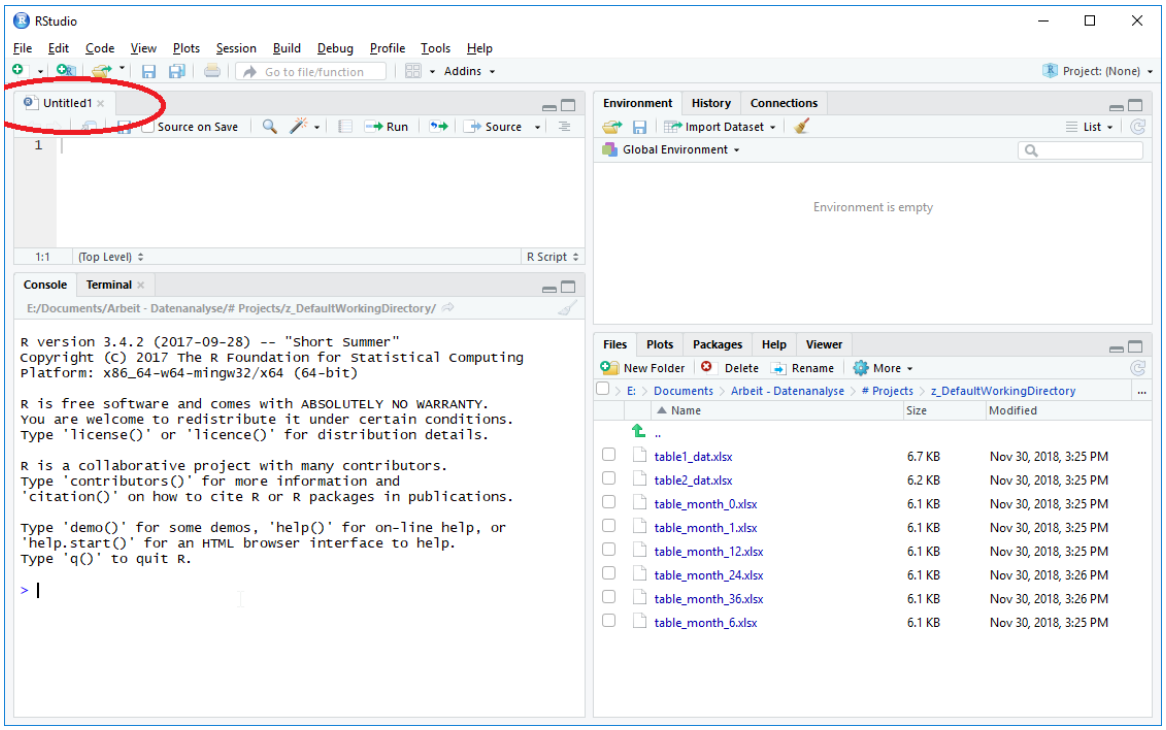

Figure 2: Default working environment.

4 Basic concepts

In this section, basic concepts of R and RStudio are introduced. If you are completely new to programming, then you are probably most interested in this part.

4.1 Executing commands

As mentioned before, RStudio is just an editor, the acutal execution of commands will be performed in the background. RStudio is sending the commands to R and displays the response of R.

To start very simple, the calculation of 3+5 is performed. Just write "3+5" into the console after the undermost ">" symbol and then press <enter>.

```
> 3+5
```

After executing your command, the result is shown in the next line after [1] which means that this is the first response. You again see another ">" symbol, ready for the input of another command. This is one way to send a command to R, see Figure 3.

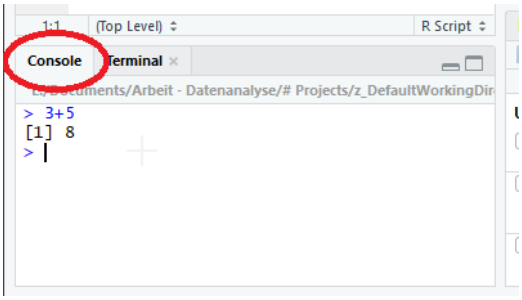

Figure 3: Console - executing 3+5.

As already stated, it is actually never recommended to code in the console, rather use a script. To open a new script, click on *"File"* → *"New File"* → *"R Script"*. Write "10/2" in one line of the script file, keep/place the cursor on this line and then "run the code" by clicking on "Run" (alternatively <ctrl> + <enter> / <cmd> + <enter>).

```
> 10/2
```

The code is now sent to R and the cursor jumps to the next line.

In order to execute several commands at once, just select all lines of interest and then click "run the code", see Figure 4.

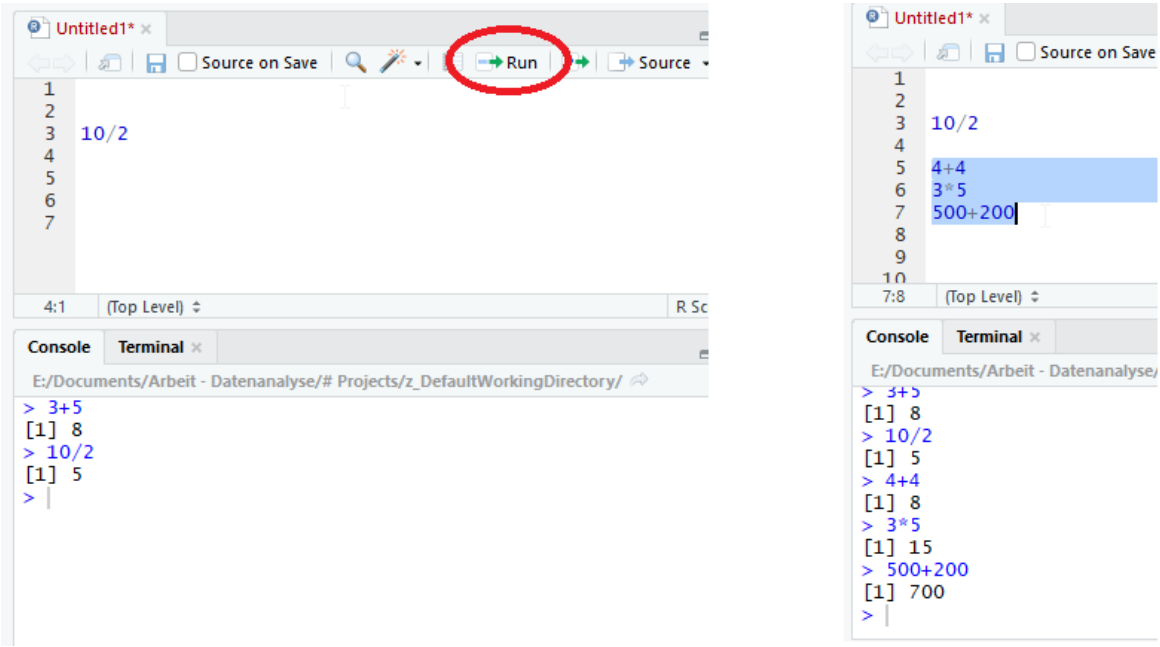

Figure 4: Console - executing several commands.

4.2 Environment

The Environment displays all created objects and functions.
You can create a variable with a specific value by writing:

```
var.1 <- 3
```

This means that the value 3 is assigned to the variable with the name "var.1". You can see that the Environment changes. The new variable "var.1" is shown there and also the assigned value 3. All objects (datasets, vectors, lists, etc.) are listed in there, see Figure 5.

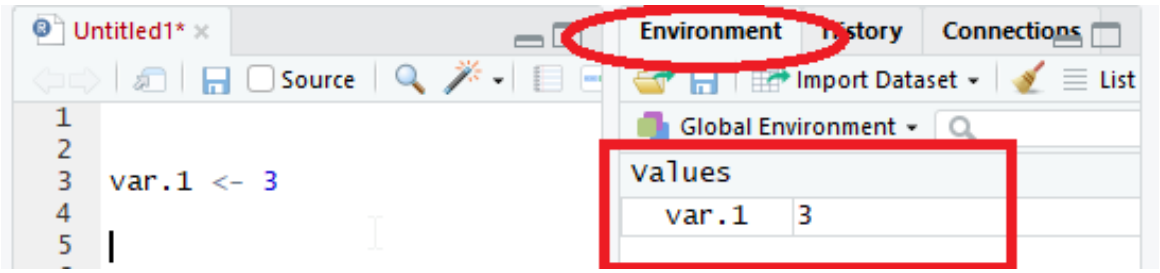

Figure 5: Environment in RStudio.

4.3 Plots

Next it is shown how you can create and view a plot. Just write and execute

```
plot(x = 1:10, y = 1/(1:10))
```

Note that this is just a very basic example, the plot can be arbitrarily modified in appearance. Automatically, the tab *Plots* opens in the lower right panel and you can view the result, see Figure 6.

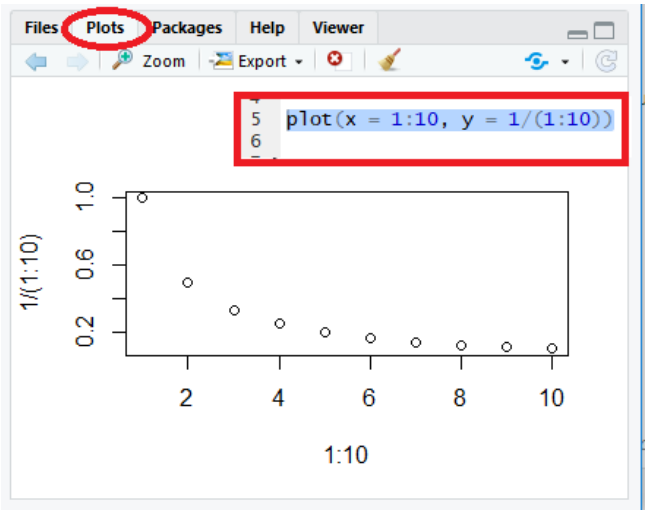

Figure 6: Plots in RStudio.

4.4 Built-in help

RStudio has a built-in help. The tab *Help* can be found in the lower right panel, next to the tabs *Plots* and *Packages*. If you need to know how to use a specific function or command you can use the search in there. RStudio then automatically retrieves the R documentation for the specific function or command. Alternatively, you can directly write in the script file "?" followed by the name of the function. E.g., if you want to know how to use the function "median", just write "?median" and

execute this line (<ctrl> + <enter> / <cmd> + <enter>), see Figure 7.

The help files may be hard to read and understand at the beginning, but you will learn that they are always structured in the same way.

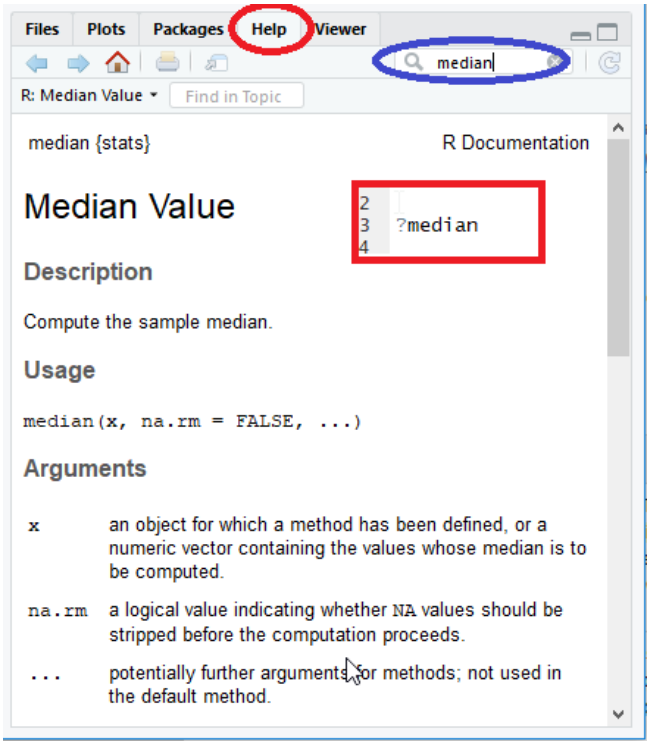

Figure 7: Built-in help in RStudio.

4.5 Packages/Libraries

R can be extended by a lot of packages. While most of the basic statistical functions are automatically loaded if you start RStudio/R, the functions of the so called "base" package, you can install and load several additional ones if needed. E.g. the package "tableone" is very useful for descriptive statistics, "epitools" for training and practicing epidemiologists, etc. Such additional packages need to be installed once before they can be used.
You can install a package by clicking on the tab *Packages* in the lower right panel. Then click on "Install" and now write the name of the package (e.g. "tableone") into the line for packages. See Figure 8.

Installed packages are not loaded automatically. To use them, you can then load a package by executing the command "library(<Name of package>)". To be precise, a package is a collection of functions and other elements, the location of the package is called library. Since this is not relevant for the regular use of R, the expressions package and libraries are often used interchangeable in most literature and therefore also in this introduction.

To understand the concept of packages better consider an example: If you run

```
> ?CreateTableone

## No documentation for 'CreateTableone' in specified packages and libraries:
## you could try '??CreateTableone'
```

R will not find this function. But after loading the package with the command

```
> library(tableone)
```

you can re-run

```
> ?CreateTableone
```

and the corresponding help page will be shown now, see Figure 8.

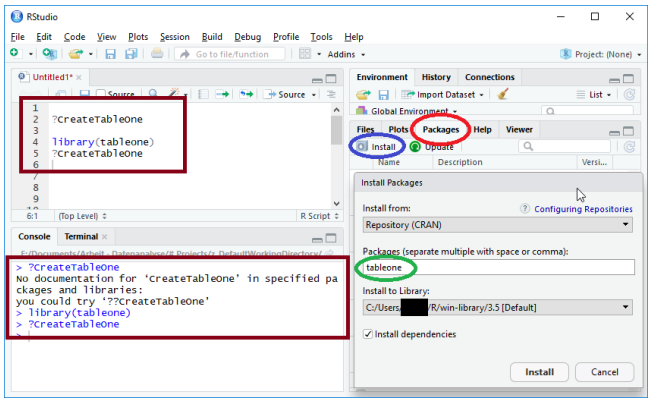

Figure 8: Installing and loading packages.

4.6 Functions

Functions are a fundamental concept in every programming language. E.g. for calculating the mean of a variable, the function mean(x) automatically calculates the average of a variable (vector of values) in R. This is a simple example of a function. A function can have several parameters, e.g. t.test(...), furthermore, a function can itself make again a function call, e.g. the function t.test() calls the function mean() in the background. This may sound complicated but it will get clearer for you when you see some more examples.

4.7 Workflow

The normal workflow for a statistical analysis in clinical research is to read in the dataset (i.e. load it into the environment) and then write all commands that transform the data and produce the results into a script file. The content of the script file will be sent to R where all commands will be executed in the same order as in the script file. R then produces your desired outputs. Of course, one cannot write down all commands for a complex analysis in one go. As mentioned before, R is a script language and the code can therefore be developed step by step. Also, all parts of a script file can be executed repeatedly, hence it is possible to undo performed changes by running the code again from the start until the desired point.

4.8 Closing RStudio

Save your R-script before you close the RStudio session. R-scripts have the extension *.R*. You may get asked whether you want to "Save workspace image?". This is optional, it saves all elements in the Environment into a file readable only in R. We do not recommend to save it.

4.9 Best practice

Before you begin a new project or just after re-opening RStudio after you closed it, it is advisable to clean the environment. This ensures, that first no "old" data or variables are remaining if you are working on different projects at the same time. All additional packages will be unloaded.
Clean the environment by clicking on the "broom" symbol, see Figure 9. Unload all packages by clicking on the menu bar "Session" and then on "Restart R" (<ctrl> + <shift> + F10 / <cmd> + <shift> + F10).

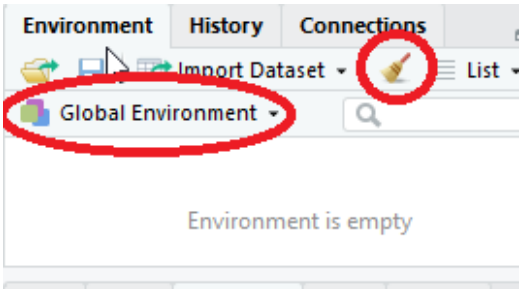

Figure 9: Clear environment.

5 Useful additions

5.1 Programming principles

Good code is also well structured and follows certain style standards. Those standards are not unique, some examples for style guides are Google's R Style Guide and The tidyverse style guide. In the following, some important and basic style principles are presented.

1. Write comments! It is essential to write comments into the R-script file and explain what you are doing in each analysis step. You can also add how and why you are doing it. In R you can write comments by putting a "#" symbol. Everything in this line after that symbol is ignored when executing the code.

This not only helps other readers, you will have an easier time if you want to re-use parts of your code for different projects after some time. But most importantly, it will help you to keep track of what you are doing.

As your code grows and the content gets more complex it is really important to keep an overview of your analysis steps. Keep in mind that there are never too many comments! Be generous, it is worth the effort.

2. Organize and structure your code! Similar to an essay/a text, one cannot/should not write code without a red thread. Therefore, plan your analysis on a scratch paper first and then implement your strategy as code. For example, by writing the following code into the script file one can create sections in a script file. Click on the "Hide document outline" symbol (<ctrl> + <shift> + O / <cmd> + <shift> + O) to view the sections (Figure 10):

```
# 1) Read in datasets ====
# 2) Check datasets and First overview ====
# 3) Descriptive statistics ====
# 3) a) Tables ====
# 3) b) Plots ====
# 4) Hypothesis Tests ====
```

The four equality signs have their reason. By using them, RStudio automatically recognizes them as a section and you can view them as shown in Figure 10.

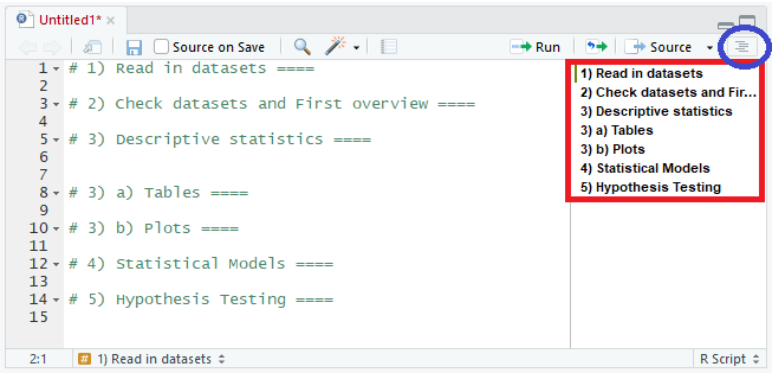

Figure 10: Creating sections in the script file.

3. Naming variables All objects should use proper names. This not only makes coding more easy, it also avoids mistakes and helps other people to understand your code.

- Use meaningful names (not too long, not too short). E.g.

  Good: *female*, *date.of.birth*, *type.disease*
  Bad: *f*, *date*, *this.variable.encodes.all.types.of.diseases*

- Do not use a space character or special symbols e.g. $\pm$, $\geq$, &, #, %, *, etc. Limit it to letters, dots ("."), underscore ("_") or numbers (put numbers never at the beginning, only in the middle or end). In general, do not use other special characters except dots and underscore. E.g.

  Good: *blood.pressure.geq.100*, *blood_sugar*, *systolic.month24*, *anaesthesia*
  Bad: *bloodpressure$\geq$100*, *blood sugar*, *24month.systolic*, *anästhesie*

- It is more comfortable to use lowercase letters, since you have to write some of those variables repetitively, but use capital letters if it helps you to distinguish. Note: variable names are case sensitive in R, i.e. the names *sysbp* and *SysBp* would refer to two different variables.
- Sometimes its useful to mention the type of object or specify to what your variable is related in a name. E.g. "*vars.symptoms_list* <- ..."

4. Keep your code clean Only keep those parts of your code in the script file that are needed for the current analysis project. You should not keep parts of code in there that are not working. If you do not want to lose some code for later use or for learning purpose just create a second script file *code_snippets.R* and store it there.

5. Warning and error messages Sometimes it happens that functions give warnings, or your code has some mistakes and an error occurs (usually in red color). Make sure that you figure out what is going wrong. Never just ignore it.

5.2 Useful shortcuts

5.2.1 Windows

to comment single line or region
<ctrl> + <shift> + C

to complete code
<tabulator> (try this out in several circumstances, e.g. it works within a function, after the "$" sign of dataframes, etc.)

to cancel execution
<esc> (sometimes needed if there is an error in the code or the calculation needs too long)

to save your script
<ctrl> + S

to switch to console
<ctrl> + 1

to switch to script editor
<ctrl> + 2

to clean console
<ctrl> + L

to create a new script
<ctrl> + <shift> + N

5.2.2 Mac

to comment single line or region
<cmd> + <shift> + C

to complete code
<tabulator> (try this out in several circumstances, e.g. it works within a function, after the "$" sign of dataframes, etc.)

to cancel execution
<esc> (sometimes needed if there is an error in the code or the calculation needs too long)

to save your script
<cmd> + S

to switch to console
<cmd> + 1

to switch to script editor
<cmd> + 2

to clean console
<cmd> + L

to create a new script
<cmd> + <shift> + N

5.3 Trouble shooting

If you encounter problems you will very likely not be the first one. Extensively use google/bing/etc. and search for more information. There are plenty of forums out there that explain how to deal with various issues! Programming includes to be able to solve problems on your own, this includes searching information on the web, ask colleagues, post a question in a forum, etc.

It's quite safe to say that every programmer encounters problems sooner or later and it takes sometimes a lot of time to troubleshoot. But you will get faster each time. Don't give up: Everyone can learn how to program!

6 Additional Material

All links were last accessed (automatically by script) on 2019-11-10. Please note that we do not take any responsibility for the content of the links/tutorials listed below.

6.1 Short notes

- *CheatSheets* `https://www.rstudio.com/resources/cheatsheets/`
- *Daten einlesen mit R (ETH)* `https://stat.ethz.ch/Teaching/WBL/R-Einstieg/Daten_Einlesen_mit_R.pdf`
- *Basic knowledge in R / RStudio* `https://stat.ethz.ch/education/semesters/as2013/anova/Exercises/R-tutorial.pdf`

6.2 Detailed introduction to R

6.2.1 General:

- *'Hands-On Progarmming with R' by Garrett Grolemund* `https://d1b10bmlvqabco.cloudfront.net/attach/ighbo26t3ua52t/igp9099yy4v10/igz7vp4w5su9/OReilly_HandsOn_Programming_with_R_2014.pdf`
- *Official Introduction by cran: 'An Introduction to R'* `https://cran.r-project.org/doc/manuals/R-intro.pdf`
- *'A tutorial to R' by Viktor Müller (ETH)* `http://www.tb.ethz.ch/content/dam/ethz/special-interest/usys/ibz/theoreticalbiology/education/learningmaterials/701-1424-00L/Rintro.pdf`
- *'Einführung in die Statistik-Umgebung R' by Christian Keller et al. (ETH)* `https://stat.ethz.ch/Teaching/WBL/R-Einstieg/R-Skript_Kap1-4.pdf`

- *'Einführung zu R' (University Ulm)* https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ss08/stat1/R-skript.pdf
- *'R for Beginners' by Emmanuel Paradis* https://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- *'Einführung in die Statistik-Umgebung R' by Werner Stahel (ETH)* https://stat.ethz.ch/~stahel/courses/R/usingr-script-d.pdf
- *'R resources (free courses, books, tutorials, and cheat sheets)' by Paul van der Laken* https://paulvanderlaken.com/2017/08/10/r-resources-cheatsheets-tutorials-books/

6.2.2 Graphics:

- *'Visualisierung statistischer Daten mit R' by Sebastian Jeworutzki (Ruhr Universität Bochum)* http://www.stat.ruhr-uni-bochum.de/teaching/vis/visu.pdf
- *'Grafiken mit R' by Mike Kühne (TU Dresden)* https://tu-dresden.de/gsw/phil/iso/mes/ressourcen/dateien/prof/lehre/freieS/Dateien/Grafiken.pdf?lang=de

6.3 Online learning platforms

- http://www.r-tutor.com/r-introduction/data-frame
- https://www.datamentor.io/r-programming/
- https://www.guru99.com/r-tutorial.html
- http://www.sthda.com/english/wiki/r-basics-quick-and-easy
- http://www.sthda.com/english/wiki/descriptive-statistics-and-graphics

6.4 Web Tutorials / learning websites

- https://stat.ethz.ch/~stahel/courses/R/
- https://www.cyclismo.org/tutorial/R/types.html
- https://stat.ethz.ch/Teaching
- https://stat.ethz.ch/~stahel/dataanalysis/

6.5 Related links

These pages include a lot of links which lead to other helpful pages.

- http://scs.math.yorku.ca/index.php/R:_Getting_started_with_R#R_lessons

6.6 Youtube

- *R programming for beginners - Statistic with R* https://www.youtube.com/watch?v=ANMuuq502rE
- *Data Camp (Online Learning Platform) - very extensive.* https://www.youtube.com/watch?v=SWxoJqTqo08&list=PLjgj6kdf_snYBkIsWQYcYtUZiDpam7ygg
- *Introduction to Data Science with R - Data Analysis Part 1* https://www.youtube.com/watch?v=32o0DnuRjfg
- *R Programming Tutorial* https://www.youtube.com/watch?v=s3FozVfd7q4

6.7 Help for self-help

- *Use search engines: Google, Bing, DuckDuckGo, etc. (good search terms are: R, tutorial, cran, cheat sheet, vignettes, example, etc.)*
- *If you want to imitate an existing plot, use the picture search of a search engine*
- *Forums, e.g. stackexchange*
- *Books, e.g. A Handbook of Statistical Analyses using R* by Hothorn and Everitt (2014) (Online Access)
- *Conversations with / ideas of colleagues, friends, experts, etc.*

7 References

Hothorn, T. and Everitt, B. (2014). *A Handbook of Statistical Analyses using R*. CRC Press.
URL `https://books.google.ch/books?id=-4-9BwAAQBAJ`

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
URL `https://www.R-project.org/`

R version and packages used to generate this report:
R version: R version 3.6.0 (2019-04-26)
Base packages: stats, graphics, grDevices, utils, datasets, methods, base
Other packages: tableone 0.10.0, RCurl 1.95-4.12, bitops 1.0-6, reporttools 1.1.2, xtable 1.8-4, knitr 1.23
This document was generated on November 10, 2019 at 03:16.