Data Science with R

Part I: Introduction and Command Line

Raphael Schleutker

February 21, 2020

The most important thing in the programming language is the name. A language will not succeed without a good name. I have recently invented a very good name and now I am looking for a suitable language.

— Donald E. Knuth

Table of contents

Organizational Stuff

Contents of this workshop

1. Introduction
2. Primitive data structures and assignments
3. Functions I and how to get help
4. Non-primitive data structures
5. Subsetting
6. Data input/output
7. Working with data frames
8. Graphics I
9. Control structures
10. Working with text information
11. Functions II
12. Graphics II
13. Functions III / Scoping rules
14. Communicating results and outlook

⋮

Procedure of this workshop

- We will meet weekly, if possible at the same time always.

Procedure of this workshop

- We will meet weekly, if possible at the same time always.
- I will give a lecture on a specific topic.

Procedure of this workshop

- We will meet weekly, if possible at the same time always.
- I will give a lecture on a specific topic.
- These lectures will build up on top of each other. So if you miss one week, you would have to work yourself through the lecture in order to keep track.

Procedure of this workshop

- We will meet weekly, if possible at the same time always.
- I will give a lecture on a specific topic.
- These lectures will build up on top of each other. So if you miss one week, you would have to work yourself through the lecture in order to keep track.
- Additionally, I will hand out weekly exercises. If you do them, you have good chances to learn something. If you don't do them, you will have no chance to learn anything (my personal experience).

Procedure of this workshop

- We will meet weekly, if possible at the same time always.
- I will give a lecture on a specific topic.
- These lectures will build up on top of each other. So if you miss one week, you would have to work yourself through the lecture in order to keep track.
- Additionally, I will hand out weekly exercises. If you do them, you have good chances to learn something. If you don't do them, you will have no chance to learn anything (my personal experience).
- Each sheet will have some exercises from the last lecture and some informations/exercises for the upcoming lecture.

Procedure of this workshop

- We will meet weekly, if possible at the same time always.
- I will give a lecture on a specific topic.
- These lectures will build up on top of each other. So if you miss one week, you would have to work yourself through the lecture in order to keep track.
- Additionally, I will hand out weekly exercises. If you do them, you have good chances to learn something. If you don't do them, you will have no chance to learn anything (my personal experience).
- Each sheet will have some exercises from the last lecture and some informations/exercises for the upcoming lecture.
- The exercises will shortly be discussed in our meetings. I will also upload sample solutions with explanations.

Where to get the material

I will upload everything to github, a repository for shared development of code.

What other resources could you use?

There are a lot of resource available for learning R like books, workshops, online tutorials, so called data camps.

- Some books are good for learning R, others are not. A good one is for instance The Art of R Programming. Books that i read in the beginning of learning R and that I found nice these days are Hand-On Programming with R, R for Data Science, and Advanced R. I do not recommend these books cause they do not teach you R but how to use the packages written by the authors (the tidyverse packages including dplyr, ggplot2, tidyr, etc.).

What other resources could you use?

There are a lot of resource available for learning R like books, workshops, online tutorials, so called data camps.

- Some books are good for learning R, others are not. A good one is for instance The Art of R Programming. Books that i read in the beginning of learning R and that I found nice these days are Hand-On Programming with R, R for Data Science, and Advanced R. I do not recommend these books cause they do not teach you R but how to use the packages written by the authors (the tidyverse packages including dplyr, ggplot2, tidyr, etc.).
- The same applies to many online resources. Often, they show you how to use the tidyverse packages. This does not replace a solid understanding of base R.

What other resources could you use?

There are a lot of resource available for learning R like books, workshops, online tutorials, so called data camps.

- Some books are good for learning R, others are not. A good one is for instance The Art of R Programming. Books that i read in the beginning of learning R and that I found nice these days are Hand-On Programming with R, R for Data Science, and Advanced R. I do not recommend these books cause they do not teach you R but how to use the packages written by the authors (the tidyverse packages including dplyr, ggplot2, tidyr, etc.).
- The same applies to many online resources. Often, they show you how to use the tidyverse packages. This does not replace a solid understanding of base R.
- As soon as you have acquired a certain understanding the best way to improve is practicing, recapitulating old code and reading source code from professional packages.

About R

R - A programming language for statistics and graphics

There are several hundreds of different programming languages out there. What does qualify R to be learned by data scientists?

- It was designed for statistics and graphics and therefore has many built-in functions that are only available as third-party packages in other languages.

R - A programming language for statistics and graphics

There are several hundreds of different programming languages out there. What does qualify R to be learned by data scientists?

- It was designed for statistics and graphics and therefore has many built-in functions that are only available as third-party packages in other languages.
- It is an interpreted language and therefore much easier to learn for beginners.

R - A programming language for statistics and graphics

There are several hundreds of different programming languages out there. What does qualify R to be learned by data scientists?

- It was designed for statistics and graphics and therefore has many built-in functions that are only available as third-party packages in other languages.
- It is an interpreted language and therefore much easier to learn for beginners.
- It is completely free...

R - A programming language for statistics and graphics

There are several hundreds of different programming languages out there. What does qualify R to be learned by data scientists?

- It was designed for statistics and graphics and therefore has many built-in functions that are only available as third-party packages in other languages.
- It is an interpreted language and therefore much easier to learn for beginners.
- It is completely free...
- It is heavily used by scientists and therefore many functionalities were developed for analyzing all kinds of biological data.

R - A programming language for statistics and graphics

There are several hundreds of different programming languages out there. What does qualify R to be learned by data scientists?

- It was designed for statistics and graphics and therefore has many built-in functions that are only available as third-party packages in other languages.
- It is an interpreted language and therefore much easier to learn for beginners.
- It is completely free...
- It is heavily used by scientists and therefore many functionalities were developed for analyzing all kinds of biological data.
- It comes with all the fun of programming ☺

R - A programming language for statistics and graphics

Cons about R include

- R is considered slow. It is indeed not blazingly fast but fast enough for everything you will do. Also, R is often slow because the code was written by people that do not have a formal education in programming and that is therefor very inefficient.

R - A programming language for statistics and graphics

Cons about R include

- R is considered slow. It is indeed not blazingly fast but fast enough for everything you will do. Also, R is often slow because the code was written by people that do not have a formal education in programming and that is therefor very inefficient.
- R is not a general purpose language. It is designed for statistics and graphics. If you discover the programmer within you and you want to write general software (like the database application I'm working on) you would have to learn another language.

R - A programming language for statistics and graphics

Cons about R include

- R is considered slow. It is indeed not blazingly fast but fast enough for everything you will do. Also, R is often slow because the code was written by people that do not have a formal education in programming and that is therefor very inefficient.
- R is not a general purpose language. It is designed for statistics and graphics. If you discover the programmer within you and you want to write general software (like the database application I'm working on) you would have to learn another language.
- It does not have a graphical user interface (GUI). You will have to write the code yourself and eventhough some IDEs like RStudio support you with GUI functionalities you still have to write that code into your script.

R and RStudio

A common misconception arises from differentiating R and RStudio.

- R is a programming language, a theoretical construct consisting of syntax and semantics. It comes along with an interpreter that interprets and translates your code into a machine readable form. Besides the standard interpreter others are available.

R and RStudio

A common misconception arises from differentiating R and RStudio.

- R is a programming language, a theoretical construct consisting of syntax and semantics. It comes along with an interpreter that interprets and translates your code into a machine readable form. Besides the standard interpreter others are available.
- RStudio is an integrated development environment (IDE) that should make the work with R as easy as possible. The editor in which you have written your code is of absolutely no importance (and thus should not be cited).

Project Organization 101

How to structure your project on the computer

Some basic rules

- Your project should be self-contained

How to structure your project on the computer

Some basic rules

- Your project should be self-contained
- Use relative path descriptions

Some basic rules

- Your project should be self-contained
- Use relative path descriptions
- Use a standardized naming convention for your files

How to structure your project on the computer

Some basic rules

- Your project should be self-contained
- Use relative path descriptions
- Use a standardized naming convention for your files
- Never user spaces in file or folder names

Some basic rules

- Your project should be self-contained
- Use relative path descriptions
- Use a standardized naming convention for your files
- Never user spaces in file or folder names
- Avoid uppercase letters

How to structure your project on the computer

Some basic rules

- Your project should be self-contained
- Use relative path descriptions
- Use a standardized naming convention for your files
- Never user spaces in file or folder names
- Avoid uppercase letters
- Do not mix up your raw data with analysis

How to structure your project on the computer

Some basic rules

- Your project should be self-contained
- Use relative path descriptions
- Use a standardized naming convention for your files
- Never user spaces in file or folder names
- Avoid uppercase letters
- Do not mix up your raw data with analysis
- Document your project structure and what you have done

Some basic rules

- Your project should be self-contained
- Use relative path descriptions
- Use a standardized naming convention for your files
- Never user spaces in file or folder names
- Avoid uppercase letters
- Do not mix up your raw data with analysis
- Document your project structure and what you have done
- Evaluate your behaviour regulary and think of what could be improved

Your project should be self-contained

Self-containedness

As far as possible, all files of one project and only those should live within one folder. Do not spread project files around and do not mix them up with files/folders from other projects (or private stuff).

Benefits

- It is much easier to find everything.
- Your life will be much easier, if you have to migrate to a new computer.
- If your done, just drag the folder into an archive (zip, gzip, ...) and that's it.

Use relative path descriptions

Relative path descriptions

A path description should never depend on the computer. Imagine a script file that lives in the root folder of your project and you want to refer to a file in the `data` subfolder.

~~`C:/Users/Raphael/Documents/project_a/data/raw/data.txt`~~

`./data/raw/data.txt`

- `./` refers to the current location of the file.
- `../` refers to the parent folder in a folder hierarchy.

Use a standardized naming convention for your files

Standardized naming convention

Files of the same type should have a common naming structure like `<yyyy-mm-dd>_<genotype>_<experimentID>.<extension>`. Choose a character as a separator between the information chunks (like underscores _) and never use this character for anything else in your file and folder names.

Following this rule allows it to easily get the meta information saved in the file name with R or other programming languages. A typical procedure is to split the file name at every appearance of the separator. Then the first element is always the date, the second is always the genotype and so forth.

Never user spaces in file or folder names

Spaces

Spaces are a common source of ambiguity in programming code. But in contrast to humans programs do not try to guess what you could have meant but create an error instead. So please, do not use spaces.

Avoid uppercase letters

Uppercase letters

Some programs differentiate between uppercase and lowercase letters and some do not. To avoid unexpected issues in your programs always use lowercase letters.

```
data <- c(1, 3.1, 4.67, 9, 32.98)
data

# [1]  1.00  3.10  4.67  9.00 32.98

Data

# Error in eval(expr, envir, enclos):  object 'Data' not found
```

Do not mix up your raw data with analysis

Raw data

Keep your raw data separated from any kind of analysis. Do not save your results along with your raw data in the same folder and never change your raw data.

This is of particular importance for everyone who's using excel. Well, what else can I say...

Document your project structure and what you have done

Documentation

I expect everyone here to be quite clever but no one can keep track of dozens, hundreds or even thousands of files. So create a README text file to document the folder structure and what kind of files each folder contains.

An example structure

```
patency_index
├── data
│   └── microscopy
│       ├── 20200-02-15_aka_m6-f.tiff
│       └── ⋮
├── output
│   └── figure_1.pdf
├── src
│   ├── R
│   │   ├── utils.R
│   │   └── graphics.R
│   └── imagej
│       └── measure_cells.ijm
├── analysis.R
└── README.md
```

Evaluate your behaviour regulary and think of what could be improved

You steadily try to become a better biologist. Why not steadily trying to become a better data scientist as well?