Running **R** from the command line

Overview

- Running the **R**-interpreter
  - now: from the command line (Unix shell)
  - later: within a jupyter notebook
- Running **R**-scripts

Running **R** from the command line

- Sometimes one wants to run **R** without the overhead of first having to start a *jupyter* notebook (we will get to them in a moment)
  - for instance, you can use **R** as a kind of pocket calculator
- Start from the command prompt (here indicated by a $-sign):

```
$ R
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for
Statistical Computing ...
Type 'demo()' for some demos, 'help()' for
on-line help, or 'help.start()' for an HTML
browser interface to help.  Type 'q()' to quit R.
>
```

- The “>” is the prompt of the **R**-interpreter which is ready to receive user commands

Entering **R** commands interactively

```
>x <- 1:50
>print(x)
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 ...
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 ...
>y <- x^2
>plot(x,y)
>?print
```

- We will talk more about the **R** syntax later. However, you already got an idea how the assigment, plotting, and help works in **R**.
- Ending an **R**-session:
  `>q()` or `>q(save="no")`
  Using the optional argument `save` suppresses the interactive asking.
- The **R**-interpreter has build-in features (e.g., command history, TAB-completion) similar to the IPython interpreter.

Running **R**-scripts from the command line

- As soon as an R-program gets longer than a few line one does not want to keep typing the commands again and again. Scripts are useful here. These are just sequences of commands saved as an executable file, e.g.: `scriptname`.R.

```
$ R < testscript.R --no-save
```

- Here we have just written some **R** commands using a text editor in a file called `testscript`.R and saved it. This is our script. The script file is then executed by the **R**-interpreter.
  - note the capital ".R" as file extension indicating an R-program

Finally, a word of warning

- **R**-scripts (and Jupyter notebooks) contain executable code.
- Executing a script (or notebook) of unknown origin is dangerous as it can make changes to your local file system
- Using such a script, people can do anything a user can do on a system, e.g., modify or delete files, create backdoors, etc...
- Be particularly suspicious if there are calls to system routines in a script you didn't write
  - don't run it if you don't understand what it is doing