R Tutorial at the WZB

1 - Introduction

Markus Konrad

October 25, 2018

Motivational introductory example

TODO

About me

- Markus Konrad
  - markus.konrad@wzb.eu
  - tel -555
  - office D005
- studied Computer Science (MSc.) at HTW Berlin
- worked at HTW Berlin and at Excellence Cluster Topoi before
- working at WZB as Data Scientist in IT dept. since April 2016
- mainly working with Python and R

Important notes and documents

- weekly course, except for three weeks in November (see tutorial schedule)
- each Thursday 10am-12pm, B001 or B002/3
- usual structure: first input presentation, then some tasks to solve
- presentation slides, scripts, tasks, solutions and datasets at https://wzbsocialsciencecenter.github.io/wzb_r_tutori
- contact: markus.konrad@wzb.eu

If you don't understand something, please ask!

Literature and other sources

- Grolemund & Wickham 2017: R for Data Science (avail. online for free)
- Kabacoff 2015: R in Action
- Salganik 2017: Bit by Bit (avail. online for free)
- Chang 2013: R Graphics Cookbook
- interactive SWIRL tutorials
- R programming course by John Hopkins Univ. / Roger Peng at Coursera

Tutorial schedule

- today: Getting to know R and RStudio
- next week: R Basics I
- Week 3: R Basics II **(self-study, no tutorial at WZB)**
- Week 4: R Basics III **(self-study, no tutorial at WZB)**
- Week 5: Transforming data with R I **(self-study, no tutorial at WZB)**
- Week 6: Recap / Transforming data with R II / plotting with ggplot2
- Week 7: Working with geo-spatial data / Record linkage
- Week 8: Working with large datasets (replicating Michel et al. 2011)
- Week 9: Guest speaker Taylor Brown → Quantitative text analysis with R I

– Christmas and New Years Eve break –

Tutorial schedule cont.

TODO

What to expect

You'll learn modern R to do:

- "data wrangling" (transform data)
- record linkage (merging / joining datasets)
- explorative data analysis (EDA) with descriptive statistics and data visualizations
- quantitative text analysis
- "Big Data" API querying and integration

TODO: update this

What not to expect

This is not a statistics course.

→ we'll focus is on data preparation, EDA and visualization

For statistics with R see:

- Dalgaard 2008: Introductory Statistics with R
- Field & Miles 2012: Discovering Statistics using R
- Matloff 2017: Statistical Regression and Classification
- Kuhn & Johnson 2013: Applied Predictive Modeling

What not to expect

This is not an in-depth programming course.

→ we'll write short scripts and learn some fundamental concepts of programming

For programming with R see:

- Wickham 2014: Advanced R
- Grolemund 2014: Hands-On Programming with R: Write Your Own Functions and Simulations
- Matloff 2011: The Art of R Programming

What is R?

- a **free**, **open-source statistical programming language** and **computing environment**
- based on S language developed at Bell Labs
- initially developed in 1993 by **R**oss Ihaka and **R**obert Gentleman at University of Auckland, New Zealand
- currently 25th anniversary!

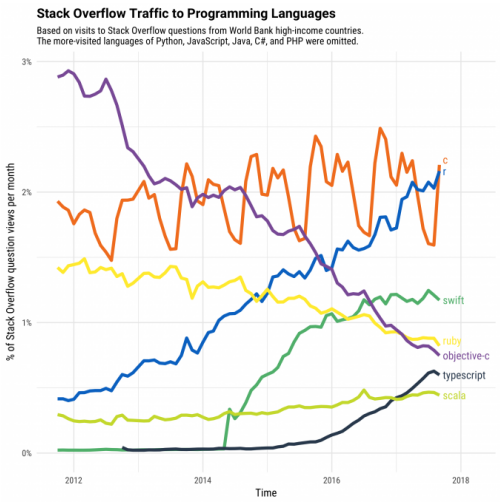
WZB
Wissenschaftszentrum Berlin für Sozialforschung

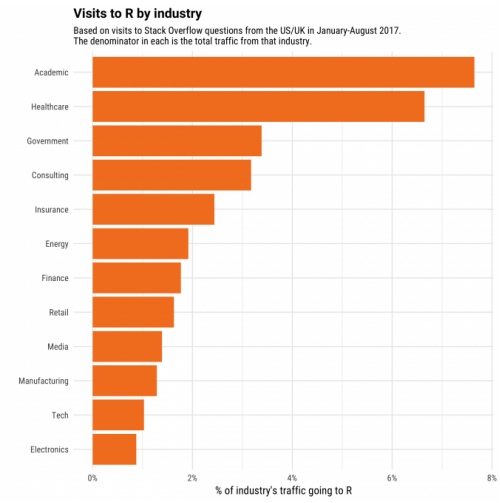
Why R?

- free and open-source
- runs on all major Operating Systems
- well tested and trusted software base
- combines flexible programming model with wide range of statistical methods
- active development and broad community
- easily extensible through R packages

Why R?

increasingly popular, esp. in the science community





source: StackOverflow

"Base R" and the "tidyverse"

"Base R" or "R Core": Core functions of the R language without additional packages

- syntax of R "historically grown" since 25 years → many ambiguities, differing concepts
- can be awkward and confusing for beginners

```
with(airquality, sapply(split(Ozone, Month), mean, na.rm = TRU
```

"Base R" and the "tidyverse"

tidyverse: set of packages that share the same "design philosophy, grammar, and data structures"

source: tidyverse.org

- https://www.tidyverse.org/
- tries to modernize R language; fosters better readable code

```
airquality %>%
  group_by(Month) %>%
  summarize(m_oz = mean(Ozone, na.rm = TRUE))
```

"Base R" and the "tidyverse"

Base R:

```
with(airquality, sapply(split(Ozone, Month), mean, na.rm = TRU
```

tidyverse:

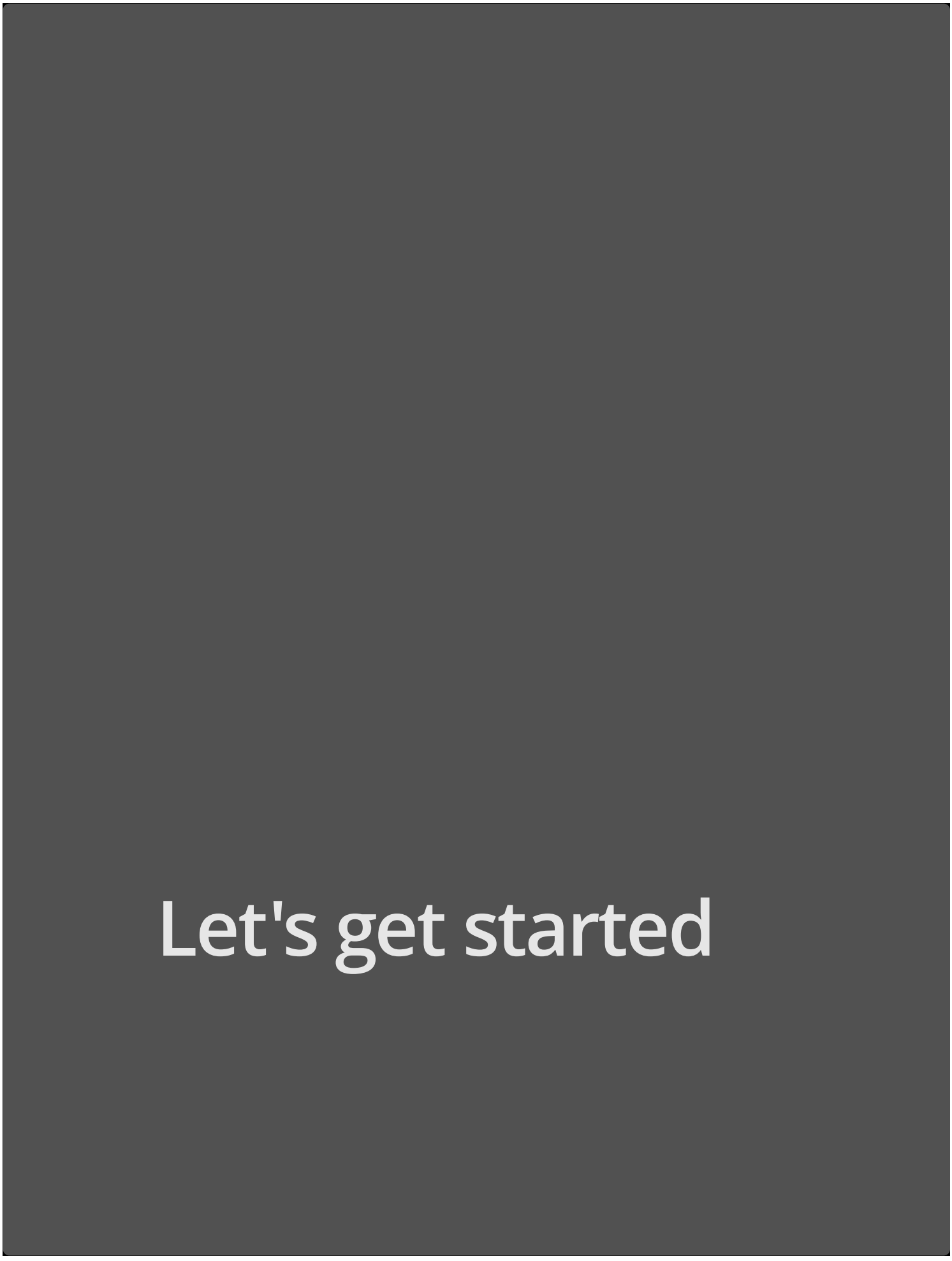
```
airquality %>%
  group_by(Month) %>%
  summarize(m_oz = mean(Ozone, na.rm = TRUE))
```

R packages and CRAN

R's functionality can be extended by packages which are available in the Comprehensive R Archive Network (CRAN).

Popular packages include:

- ggplot2 (data visualization)
- dplyr and tidyr (data manipulation)
- foreign (read/write data from Stata, SPSS, SAS, etc.)
- RColorBrewer (popular color schemes from colorbrewer)
- caret and Keras (advanced regression and machine learning models)


Let's get started

RStudio

- RStudio is an **Integrated development environment (IDE)** for R
- it's a comfortable **interface** to R
- analogy: if R is the engine, then RStudio is the car around it
- offers:
  - interactive console
  - script editor with error checking
  - package manager
  - data, plot and file viewers
  - ...

RStudio Server

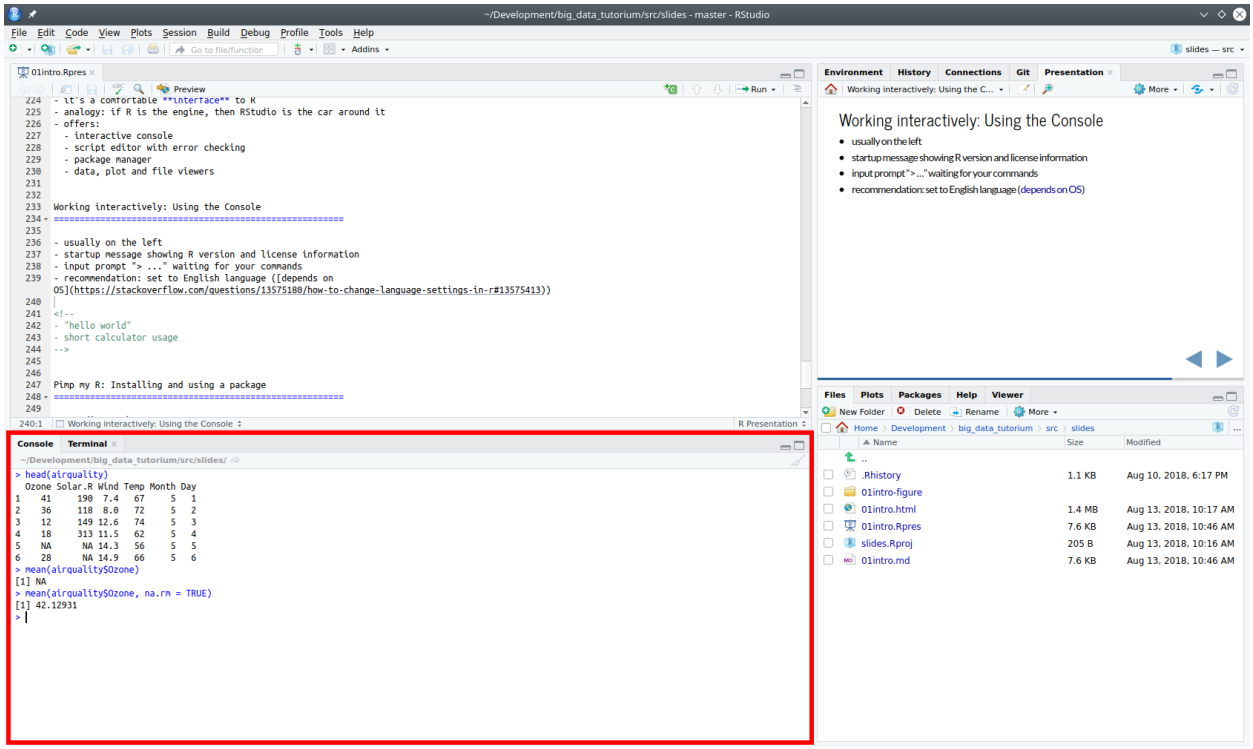
Only for WZB staff:

For those who can't / don't want to install RStudio on their computer there's an option to use RStudio via the browser:

https://rstudio.wzb.eu

Use your WZB login there.

Working interactively: Using the Console



Working interactively: Console tips & tricks

- usually on the (lower) left
- startup message showing R version and license information
- input **prompt** "> …" waiting for your commands
  (commands are issued using ENTER)
- output: depends on data type
- **general hint: all commands are case sensitive**
- recommendation: set to English language (depends on OS)

Some general R syntax rules

- a syntax describes the general rules of a programming language
- it's like a grammar in a natural language, however, a programming language is **very** strict about the grammar

Some general rules:

1. Each line is a statement ("command"), several statements are evaluated from top to bottom.

```
c <- a + b
d <- sqrt(c)
```

Exception: If an expression is not closed (see paranthesis rule below), it can span several lines:

```
a * (b
+ c
+ d)
```

This is the same as `a * (b + c + d)`.

Some general R syntax rules

2. Spaces are generally ignored.

These are all equivalent:

```
a+b
a + b
a     +   b
```

Use spaces and indents to make your code more readable.

Some general R syntax rules

3. Expressions must be closed.

There are different special characters, that mark the beginning and end of something, e.g. the beginning and end of a character string or an expression:

```
"hello world"
a * (b + c)
x[1]
```

More complex statements contain nested expressions. Nested expressions are evaluated from inner to outer.

```
y[c(1, 3)]
```

For each opened paranthesis, quotation mark, etc. there must be a closing counterpart in the correct order. This would be wrong:

```
y[c(1, 3])
## Error: unexpected ']'
```

Some general R syntax rules

4. Comma and dots

Comma split things: Mainly arguments (parameters) of functions.

```
log(x, 5)
```

→ passes the parameters x and 5 to compute the base 5 logarithm of x.

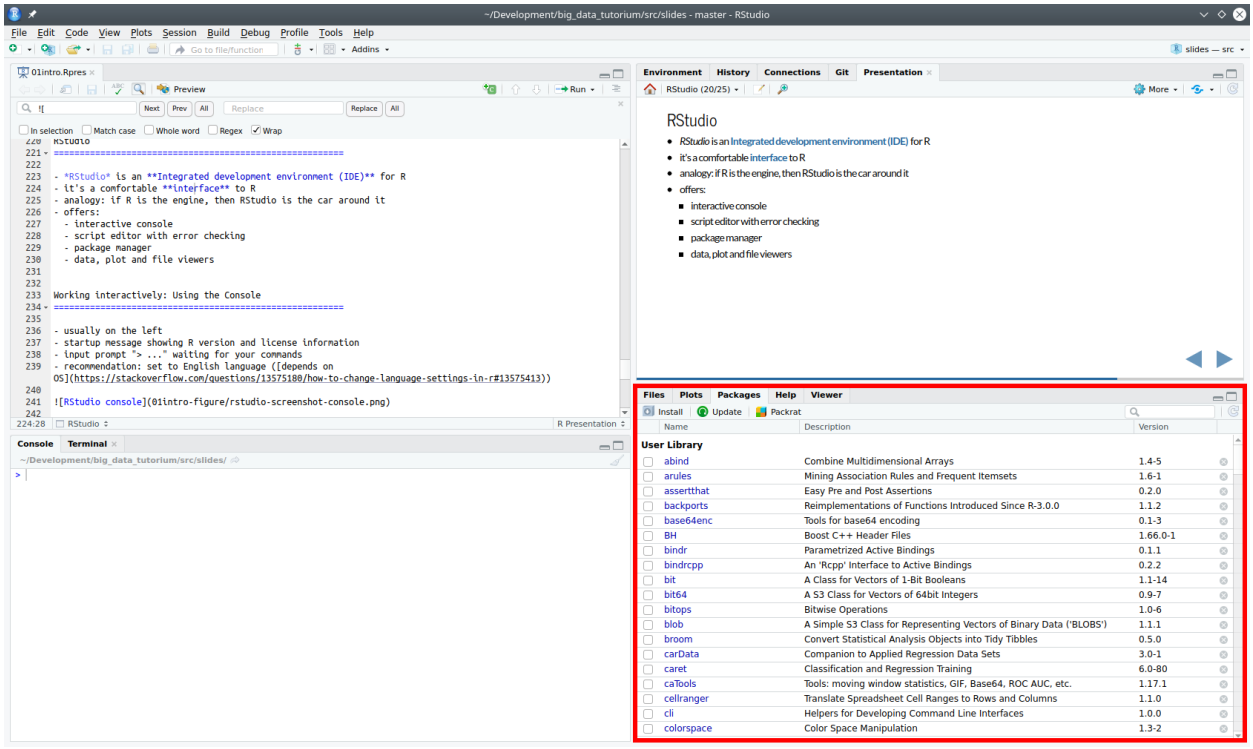
Comma **cannot** be used to group digits in large numbers:

```
population <- 3,350,000
## Error: unexpected ',' in "population <- 3,"
```

A dot is used as decimal point:

```
3.1415
```

Pimp my R: Installing and using a package



WZB
Wissenschaftszentrum Berlin für Sozialforschung

Pimp my R: Package manager in RStudio

- packages (aka "libraries") extend R's functionality
- on the right, "Packages" tab
- allows to view, install and update R packages from CRAN
- **first task for you:** install the following packages
  - tidyverse (this is a meta-package containing lots of other packages – it will take a while)
  - swirl (this is package for interactive exercises that we'll use later)

Pimp my R: Package manager tips & tricks

- alternative: use command on Console:

  `install.packages("<PACKAGE_NAME>")`

- then, to load a package:

  `library(<PACKAGE_NAME>)` (without quotation marks!)

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

Pimp my R: Package manager tips & tricks

If you forget to load a package, you will be confronted with errors like these:

```
qplot()
## Error in qplot() : could not find function "qplot"
diamonds
## Error: object 'diamonds' not found
```

Knowing where you R: The working directory concept

- the working directory or path is the location on your computer's drive, at which your current R session is working
- reading files, writing files, etc. is **relative to this path**
- finding out the current working path: `getwd()`
- setting the working path: `setwd("<PATH>")`
- **absolute path:** path starts with `/` (MacOS / Unix) or `C:\`
  - depends on your personal folder structure
- **relative path:** path starts directly with a file or folder name
  - relative from some other path, e.g. the current working path

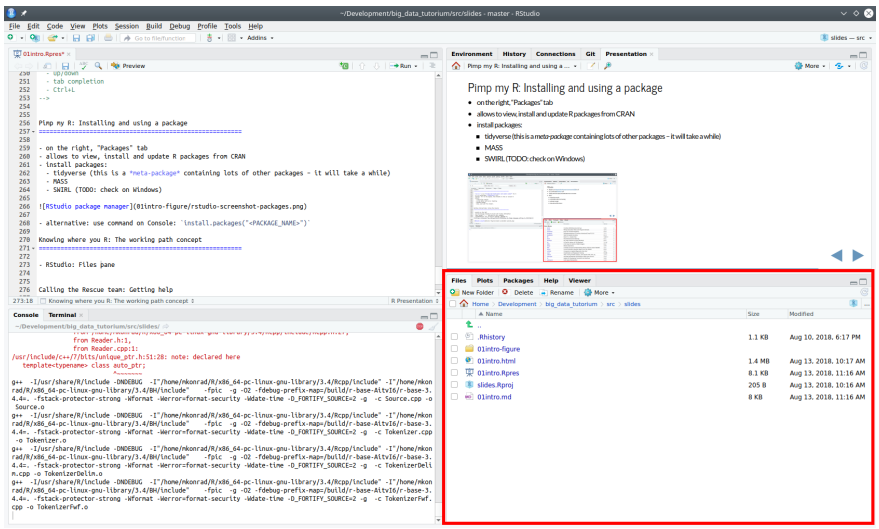
Knowing where you R: An example

- `getwd()` returns `"/Users/NoName/Documents"`
- the file you want to load is at `/Users/NoName/Documents/MyProject/data.csv`
- you can load the file with: `read.csv("MyProject/data.csv")`
- what if the working path were at...
  - `/Users/NoName/Documents/MyProject`?
  - `/Users/NoName/Research`?

Tips for file and folder names

- do not use spaces (use _ instead)
    - "funny file name .xlsx" – how many spaces do you count?
- try to stick to the English alphabet, avoid special characters
- keep it short
- can be case sensitive
- for a single project, use the same root directory for scripts and data
- do not use absolute paths in your code → it will only run on your computer!

RStudio file manager



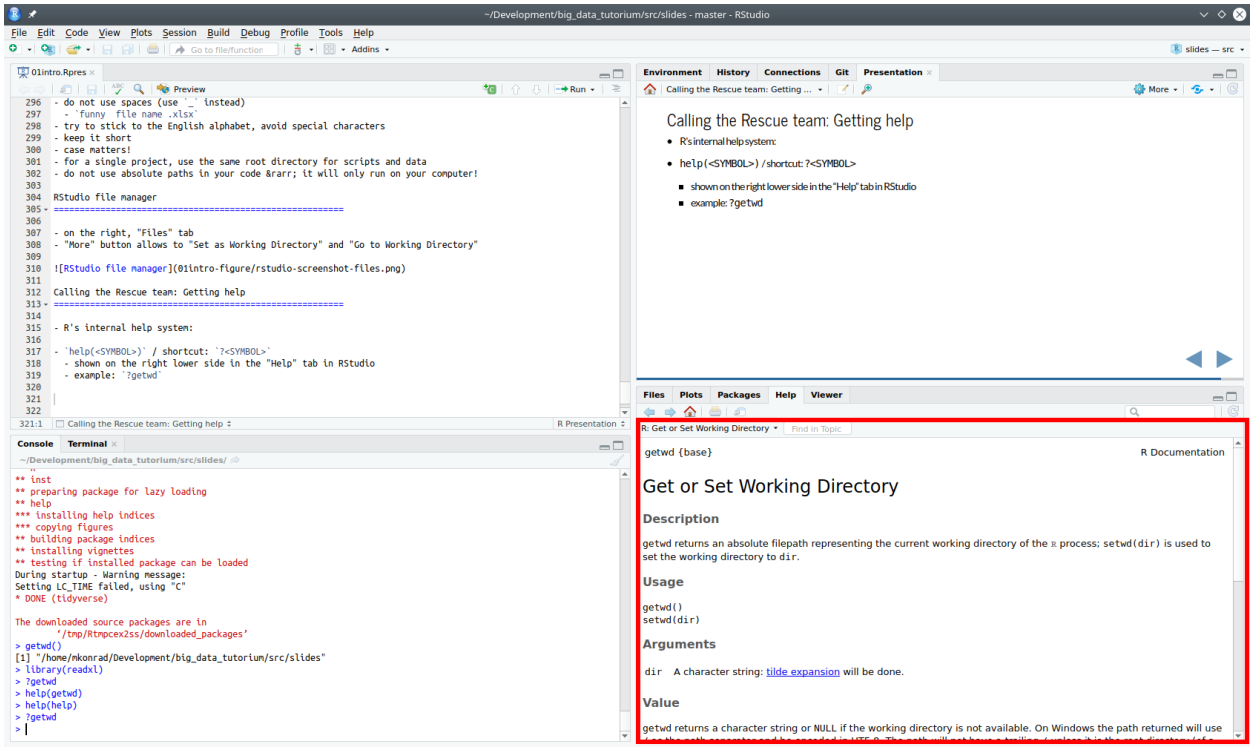
- on the right, "Files" tab
- "More" button allows to "Set as Working Directory" and "Go to Working Directory"

Calling the Rescue team: Getting help

Using R's internal help system:

- `help(<SYMBOL>)` / shortcut: `?<SYMBOL>`
- `<SYMBOL>` can be anything: a function, a package, a data set
- shown on the right lower side in the "Help" tab in RStudio
- example: `?getwd` or `?mean`

Calling the Rescue team: Getting help



Other useful help functions

- show example usages: `example(<SYMBOL>)`

```
example(mean)
## mean> x <- c(0:10, 50)
## mean> xm <- mean(x)
## mean> c(xm, mean(x, trim = 0.10))
## [1] 8.75 5.50
```

- list all available functions containing a keyword: `apropos("<SEARCH>")`

```
apropos('matrix')
## [1] "anyDuplicated.matrix"      "as.data.frame.matrix" ...
## [4] "as.matrix"                 "as.matrix.data.frame" ...
```

Other useful help functions: Vignettes and online help

Vignettes provide a short introduction to a specific package, function or topic. Not all packages offer a vignette.

- `vignette()` shows all available vignettes
- `vignette('<TOPIC>')` openes a vignette for a specific topic (e.g. `vignette('dplyr')` → introduction to the dplyr package in the help viewer)
- packages have info page on CRAN (search online for "cran ")
  - example: ggplot2 CRAN page
- many packages have own websites / online documentation, especially the tidyverse packages (tidyverse.org)

Handling problems and frustration

- R has a steep learning curve
- but it's worth the effort!
- programming languages are not fault tolerant, they're relentless in case of typos, syntax errors, etc.
- you need to be exact
- if you know R, you can learn other programming languages easier
- BUT: better don't try to learn more than one programming language at once

In case of fire, do not run

If you encounter an error:

- look closely and/or let someone else look closely
- break into smaller pieces and repeat
- use minimal data to reproduce error
- have a look at examples that use the similar functions or make similar calculations
- learn to use a debugger *
- search for help online (see tips on next slide)
    - StackOverflow
    - R-help mailinglist

* Maybe we'll have a session on this later in the semester.

Getting help online

Web search query patterns:

- `"r <PACKAGE> <PROBLEM>"`
- `"r <PROBLEM>"`

Reduce error messages to the general problem:

```
summarize(airquality, m_oz = mean(SolarR))
## Error in summarise_impl(.data, dots): Evaluation error:
## object 'SolarR' not found.
```

→ possible search query: `"r dplyr summarize object not found"`

Getting help online

Example 2:

```
mean(airquality$Ozone)
## [1] NA
```

→ possible search query: `"r mean always returns NA"`

Example 3:

Sometimes, error messages provide hints:

```
filter(airquality, Month = 7)
## Error: `Month` (`Month = 7`) must not be named, do you need
```

Tasks

Tasks

source: attackofthecute.com

1. If not done already, install the packages, tidyverse, swirl and MASS
2. Load the packages MASS and tidyverse. Loading these packages will produce some messages on the console. What do you think do they mean? (If you don't know, just make a wild guess!)
3. Load the builtin dataset "cats" provided by the package MASS (Hint: Run `data(cats)` to load the data)
4. Inform yourself about the data using R's help system – What are the variables in the dataset?
5. View the data using 4 different perspectives:
   - Issue simply the command `cats` at the console – What generally happens when you simply use an object's name as command?
   - Using the functions `head` and `tail`.
   - Using RStudios `View` function (use the function from the console and also check out the small table icon in the "Environment" tab in the top right pane)
6. Construct a scatter plot of the data using `qplot` from the ggplot2 package (incl. in tidyverse)
   - inform yourself on the Web, about what a scatter plot is
   - see the documentation for `qplot` in R's help system
   - plot `Bwt` (body weight) on the x-axis and `Hwt` (heart weight) on the y-axis
   - Hint: a scatterplot can be generated with a command like this:
   - `qplot(<VARIABLE ON X>, <VARIABLE ON Y>, data = <DATASET>)`

WZB Wissenschaftszentrum Berlin für Sozialforschung