R Projects and data import

Applied Data Science using R, Session 7

Prof. Dr. Claudius Gräbner-Radkowitsch
Europa-University Flensburg, Department of Pluralist Economics
www.claudius-graebner.com | @ClaudiusGraebner | claudius@claudius-graebner.com

Europa-Universität Flensburg

Europa-Universität Flensburg
International Institute of Management and Economic Education
Department of Pluralist Economics

Goals for today

I. Learn how to set up an R project

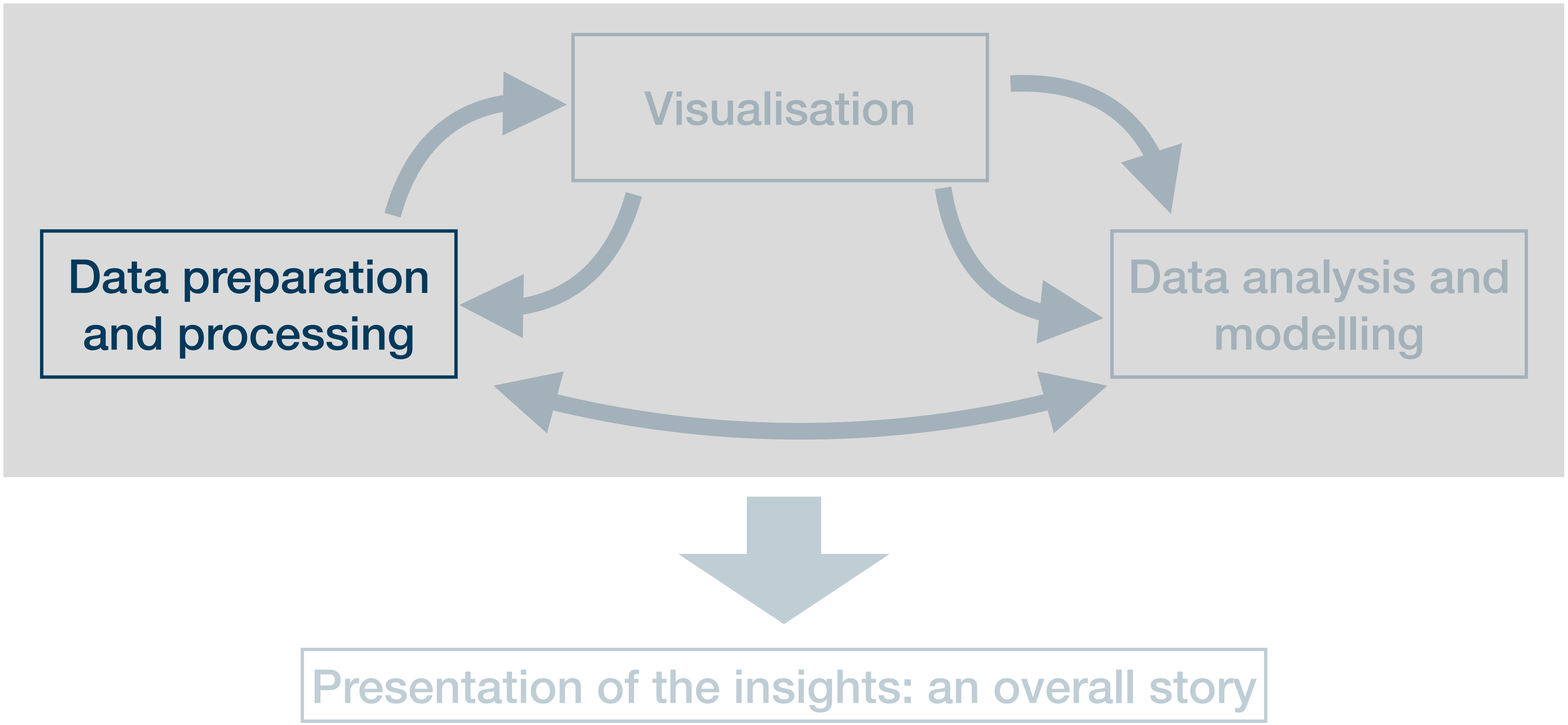
II. Learn how to use the here package and import data

III. Learn how to import data into R using `data.table::fread()`

Data wrangling in R

The role of data preparation

- Importing and preparing is the most fundamental task in data science
  - It is also largely under-appreciated 🙄


Visualisation
Data preparation and processing
Data analysis and modelling
Presentation of the insights: an overall story

Data wrangling is an essential skill

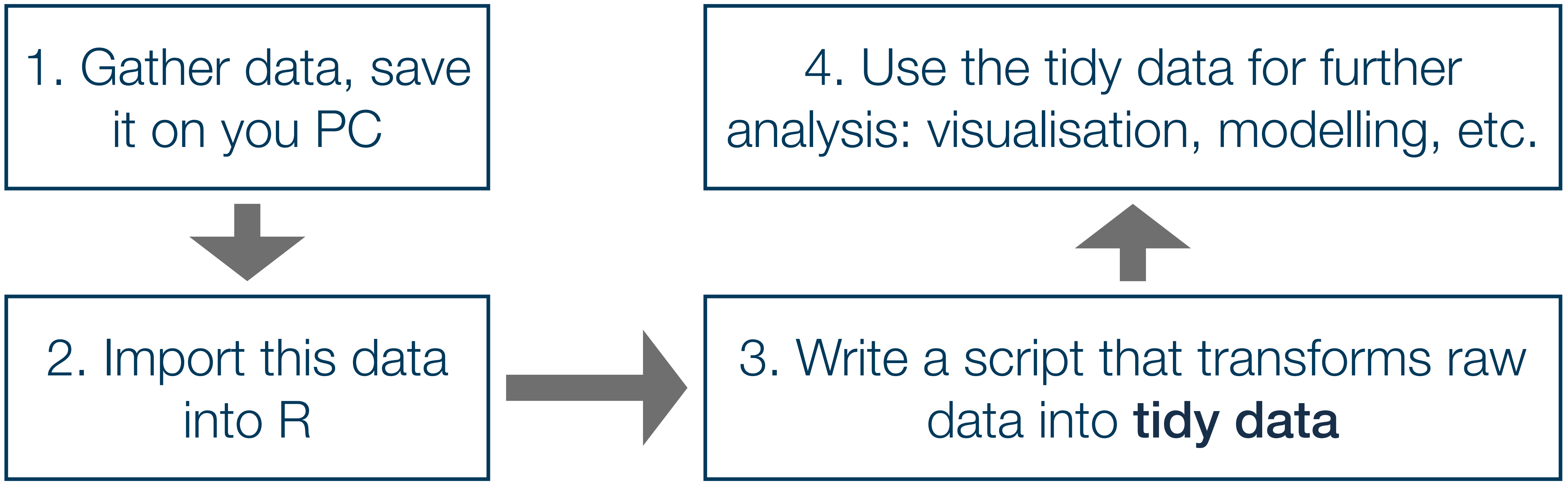
- According to several surveys, data scientists usually spend **about 80% of their working time** with importing and preparing data
- At the same time, few people learn how to do it properly
- Although it might sound a bit boring in the first place, few skills will
    - …save so much time
    - …save you so much nerves and frustration
    - …help you making so many new friends
- …in the medium run as skills in data wrangling!

Our goal

- The **goal**: learn about a **default directory structure** and a general way to **document everything** you do with your raw data
    - This will also facilitate the collaboration with future-you considerably
    - Nothing is worse than hating your past-you for not documenting correctly where data came from, or how it has been prepared
- Here we will learn a general workflow that, once mastered, helps you to avoid all editable problems **with certainty**
- A central idea is that all your research results must be **reproducible** from the raw data at any time
    - This implies that you **must not manipulate your raw data** at any cost
    - Raw data is what you download from the internet, gather through an experiment, or code yourself

How to keep your work transparent

- Raw data must not be changed, but is usually not in a state we can work with 🤨


1. Gather data, save it on you PC
2. Import this data into R
3. Write a script that transforms raw data into **tidy data**
4. Use the tidy data for further analysis: visualisation, modelling, etc.

- Saving the scripts in steps 2 & 3 makes your work fully reproducible
- By looking into the script you will always know what you did to your raw data → you can also heal basically every mistake you made, not harm done!

Outlook

- The remainder will be organised as follows:

Set up you project environment

This is done only once per project

Import data

Transform raw data into tidy data

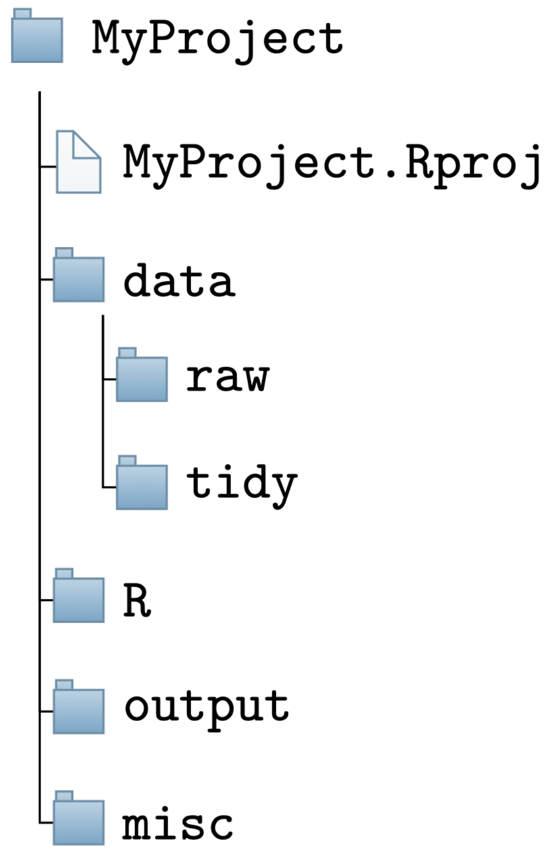
Save data

This might be done several times

Set up your R project

Setting up your working environment

- Before we talk about importing raw data we need to discuss where the raw data should actually be saved
- A prerequisite for a transparent, reproducible, and easy-to-work-with project is the right directory structure
- Thus, for every task in R you should set up your project like this:
- All the relevant steps to set this up, and the rationality for this structure are described in the respective tutorial

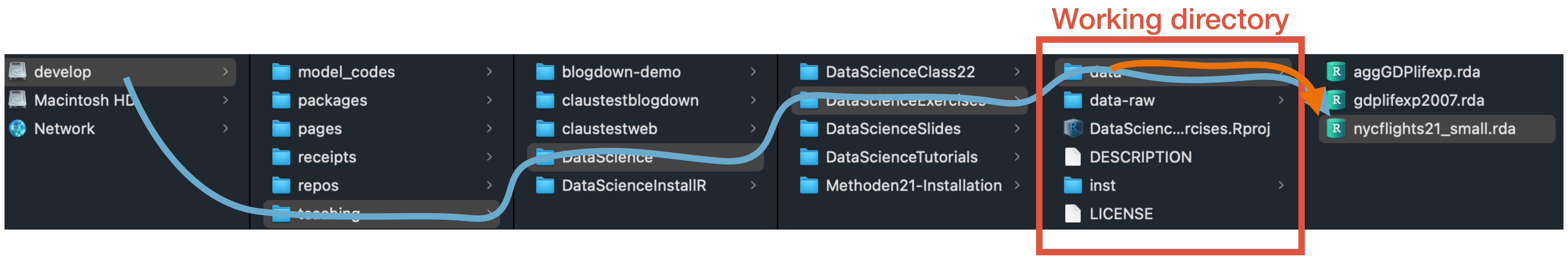


Creating an R project

See the tutorial on the course homepage for the documentation of the relevant steps (slides focus on selected background concepts only)

Paths and the here-package

- There are two ways in which you tell your computer where a certain file is located:
  - Via an absolute path: description starts at the root directory 🌲
  - Via a relative path: description starts at your current position in the file system



- Assuming we are 'located' in the folder `DataScienceExercises`: and want to point to the file `nycflights21_small.rda`:
  - `"/Volumes/develop/teaching/DataScience/DataScienceExercises/data/nycflights21_small.rda"`
  - `"data/nycflights21_small.rda"`

Relative paths and setwd()

- The relative path seems nicer…
  - Its shorter 😇 and you can share code without forcing others to adjust the path
- Problem: how to set our location to the directory `DataScienceExercises`?
- We can do this using `setwd()`, providing the absolute path to `DataScienceExercises` as an argument:
  - `setwd("/Volumes/develop/teaching/DataScience/DataScienceExercises")`
  - Then we can use `"data/nycflights21_small.rda"`
- Many people put `setwd()` at the top of their scripts
  - **BUT YOU MUST NEVER EVER DO THIS!!!!!!!!!!!!!!!!!!!!!!!!!!!**

Why setwd() is evil and not to be used

- You should never ever use `setwd()` in your scripts
- First, it does not help avoiding absolute paths because you have to provide an absolute path to `setwd()` 🤯
- Second, it makes people hate you:

Abby writes amazing_script.R 👩🏼‍💻

```
setwd("/Volumes/Macintosh HD/Users/AbbysUserName/
          PathToFolderThatOnlyExistsHere/ProjectName”)
data_file <- data.table::fread(“data/file.csv”)
```

Sends file to Ellie 📧

↓

Ellie opens file and executes it 😀

↓

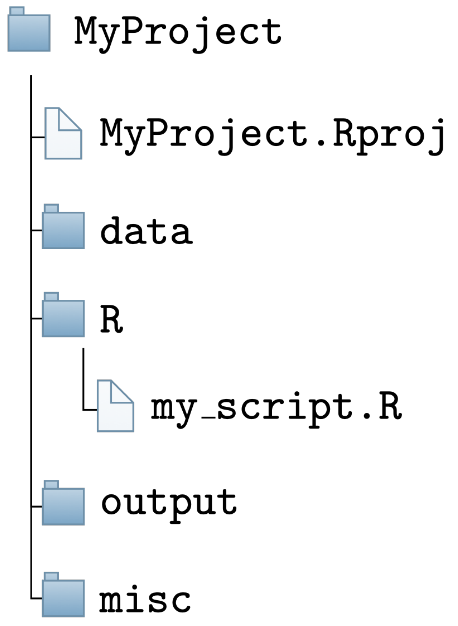
```
> setwd("/Volumes/Macintosh HD/Users/AbbysUserName/PathToF
olderThatOnlyExistsHere/ProjectName/file.txt")
Error in setwd("/Volumes/Macintosh HD/Users/AbbysUserName/
PathToFolderThatOnlyExistsHere/ProjectName/file.txt") :
  cannot change working directory
```

The better alternative to setwd() is `here`

- Thankfully, there is a very simple solution: the package **here**
- It allows you to set an anchor ⚓ in you project directory
- Then you can create paths relative to this anchor using the function `here::here()`
  - These commands will always work on every machine
- Always put `here::i_am()` into the first line of your scripts
  - As an argument, provide the location of the script relative to the project root
- From now on, only provide paths relative to this root using `here::here()`

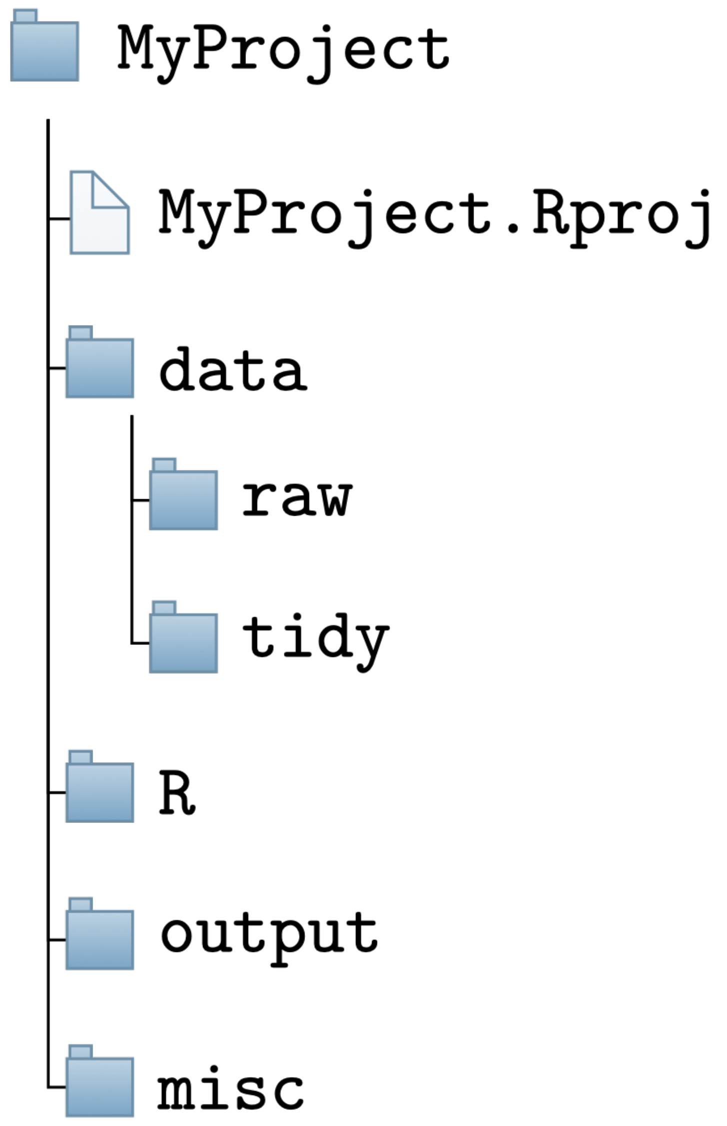
```r
here::i_am("R/my_script.R")
library(here)
library(ggplot2)
# Script content

```



Your turn

- Create a new R-Project on your computer
- Create all the required folders
- Write an R script, put it into the right directory, and make it usable for the `here`-package
- Check out what the function `here::here()` returns and experiment with its use



Importing data

Import functions

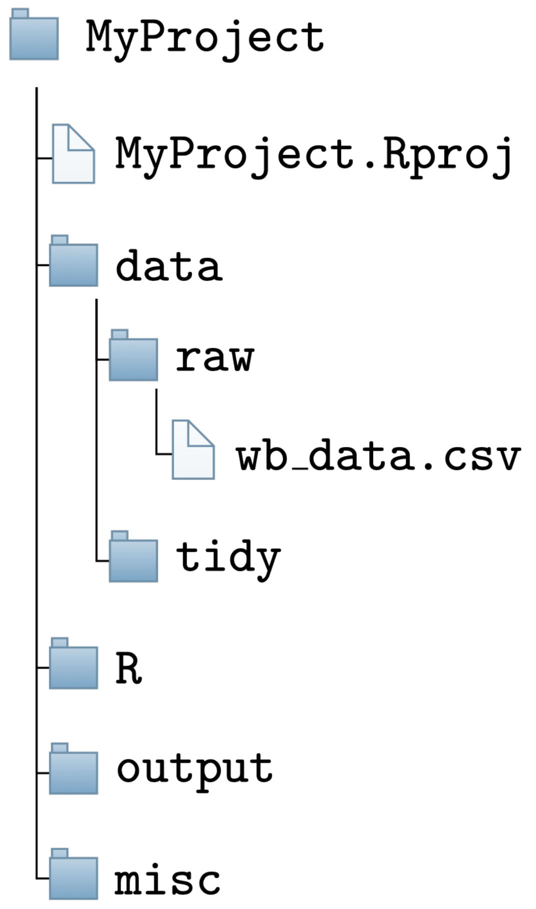
- Now that we have set up the project environment we can import data
- In the following we will assume that you raw data is stored in the folder `data/raw`
- The function we use to import a data set depends on the file type:

csv/tsv files	.Rds/RData files	Specific formats

`data.table::fread()`	`readRDS()`
	`load()`
		`haven::read_dta()`
		`haven::read_sas()`
		`haven::read_spss()`

- Basic procedure the same in all cases → focus on reading csv files here

How to import data



- Good practice: save path to file in a vector:

```
data_path <- here("data/raw/wb_data.csv")
```

- Since its a csv file we use `data.table::fread()`:

```
data.table::fread(file = data_path)
```

- This uses default options to import the file
    - Works often for clean data files
    - But for the sake of transparency and since data files are often not clean, we should specify several optional arguments
- In contrast to the exposition in Wickham and Grolemund (2022) I strongly recommend using `data.table::fread()`

Your turn

- Download the zip file `fread_expls.zip` from the course homepage
- Extract the zip file within the folder `data/raw/` in your R project
- Write a script that imports the data set saved in the file `fread_expls-1.csv` into your session

How to use data.table::fread()

- *For documentation of the next steps, please refer to the tutorial on data import on the course page*
- In the following we will learn when and how to use the following arguments of `data.table::fread()`:
    - `file`: the relative path to the csv file you want to read → use `here::here()`
    - `sep`: symbol that separates columns
    - `dec`: symbol used as decimal sign
    - `colClasses`: what object type should be used for the columns?
- For other widely used commands check the tutorial and do the exercises
    - But note that there are even more specification options → `help(fread)`

How to use data.table::fread()

- *For documentation of the next steps, please refer to the tutorial on data import on the course page*
- In the following we will learn when and how to use the following arguments of `data.table::fread()`:
  - `file`: the relative path to the csv file you want to read → use `here::here()`
  - `sep`: **symbol that separates columns**
  - `dec`: symbol used as decimal sign
  - `colClasses`: what object type should be used for the columns?
- For other widely used commands check the tutorial and do the exercises
  - But note that there are even more specification options → `help(fread)`

How to use data.table::fread()

Specifying column delimiters using sep

```
c_code; year; exports; unemployment
AT; 2013; 53.44; 5.34
AT; 2014; 53.39; 5.62
DE; 2013; 45.4; 5.23
DE; 2014; 45.64; 4.98
```

- Especially in Germany, columns are often separated via `;` instead of `,`
- We can pass a string to `sep` indicating how the columns are separated
    - In the above case: `sep = ";"`

How to use data.table::fread()

- *For documentation of the next steps, please refer to the tutorial on data import on the course page*
- In the following we will learn when and how to use the following arguments of `data.table::fread()`:
    - `file`: the relative path to the csv file you want to read → use `here::here()`
    - `sep`: symbol that separates columns
    - `dec`: **symbol used as decimal sign**
    - `colClasses`: what object type should be used for the columns?
- For other widely used commands check the tutorial and do the exercises
    - But note that there are even more specification options → `help(fread)`

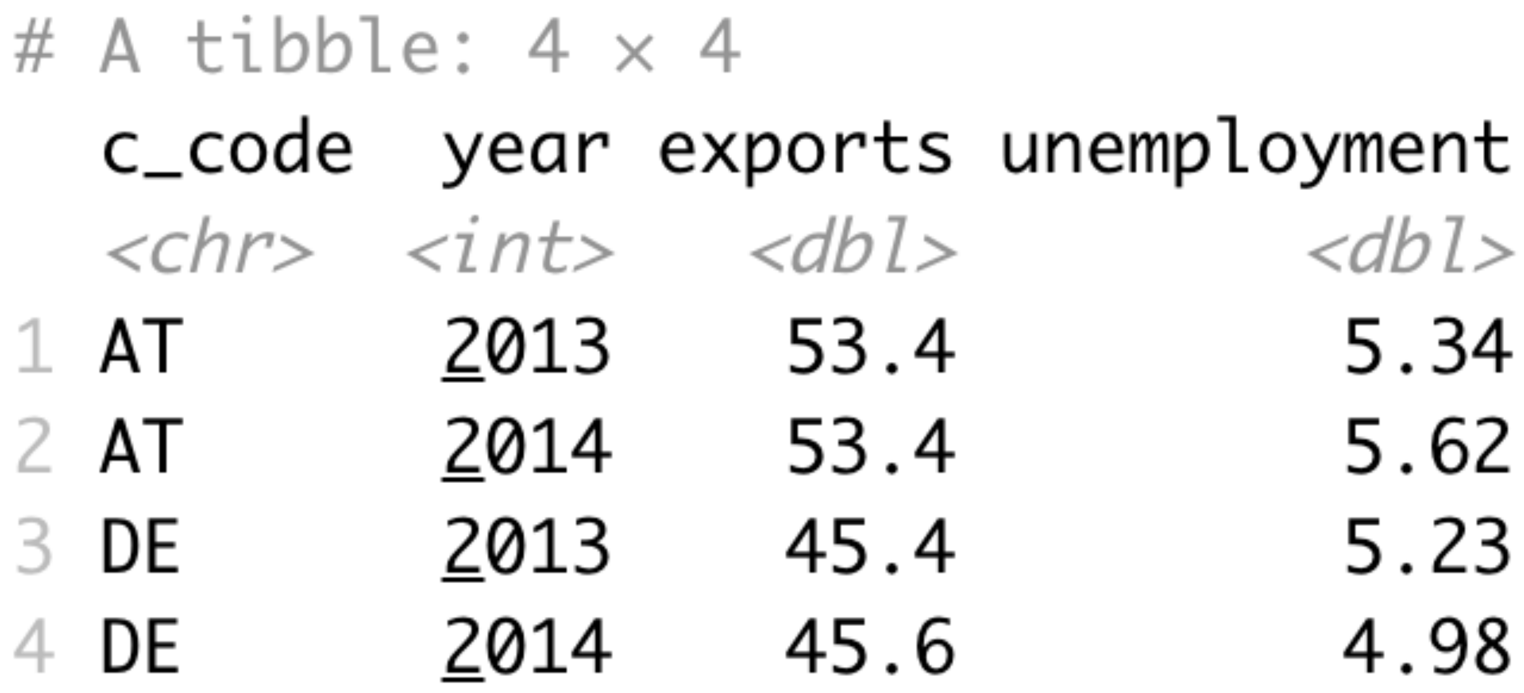
How to use data.table::fread()
Specifying decimal separators using dec

```
c_code; year; exports; unemployment
AT; 2013; 53,44; 5,34
AT; 2014; 53,39; 5,62
DE; 2013; 45,4; 5,23
DE; 2014; 45,64; 4,98
```

- Again in Germany, decimal places are often separated via `,` instead of `.`
- We can pass a string to `dec` indicating how the columns are separated
  - In the above case: `dec = ","`

Your turn

- Copy the file `fread_expls-2.csv` into the data directory of your R project
- Write a script that imports the data set into your session such that the following `tibble` results:



```
# A tibble: 4 × 4
  c_code  year exports unemployment
  <chr>  <int>   <dbl>        <dbl>
1 AT      2013    53.4         5.34
2 AT      2014    53.4         5.62
3 DE      2013    45.4         5.23
4 DE      2014    45.6         4.98
```

How to use data.table::fread()

- *For documentation of the next steps, please refer to the tutorial on data import on the course page*
- In the following we will learn when and how to use the following arguments of `data.table::fread()`:
  - `file`: the relative path to the csv file you want to read → use `here::here()`
  - `sep`: symbol that separates columns
  - `dec`: symbol used as decimal sign
  - `colClasses`: **what object type should be used for the columns?**
- For other widely used commands check the tutorial and do the exercises
  - But note that there are even more specification options → `help(fread)`

How to use data.table::fread()

Specifying column types using colClasses

- Whenever numbers should be saved as character, the guessing algorithm of `data.table::fread()` often fails:

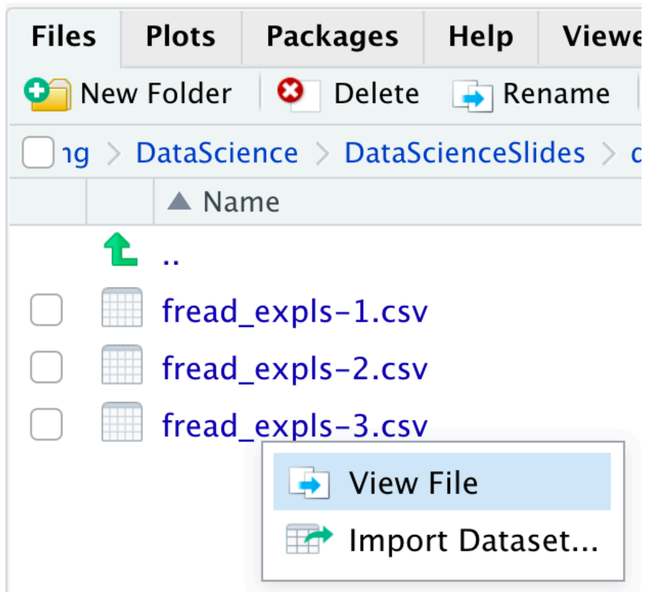
```
c_code,year,exports, PROD_CODE
AT, 2013, 53.44, 0011
AT, 2014, 53.39, 0011
DE, 2013, 45.4, 0011
DE, 2014, 45.64, 0011
```

```
# A tibble: 4 × 4
  c_code   year exports PROD_CODE
  <chr>   <int>   <dbl>     <int>
1 AT       2013    53.4        11
2 AT       2014    53.4        11
3 DE       2013    45.4        11
4 DE       2014    45.6        11
```

- We can specify the column types explicitly by passing a vector to `colClasses`:
  - `colClasses = c("character", rep("double", 2), "character")`
- Usually, this is often a good idea to make your code more transparent
- You can also combine it with `select` and only read selected columns (see tutorial)

A final exercise…

- Now read in the file `fread_expls-3.csv` and use all the arguments you consider to be necessary
- Make sure that the column `cgroup` is stored as a `factor`
- Then talk to you neighbour and compare your solutions
- Hint:
  - To get an idea about the raw data, click on the file and select "View File" to see it in its raw form → helps you to choose the right arguments:
  - Infeasible for very large files → use `nrows` and `select` to read a representative subset (see tutorial)



And what about saving data?

- Saving data is much easier than reading data
- The only relevant question is about the format
  - If there are no good arguments for using a different format, go for csv
- This can be achieved by `data.table::fwrite()`with the main arguments:
  - `x`: the name of the object to be saved
  - `file`: the file name under which the object should be saved
- Example: save object `exp_tab` to file `data/exp_tab.csv`:

```
data.table::fwrite(
    x = exp_tab,
    file = here::here("data/exp_tab.csv")
)
```

Data import as part of data preparation

Make yourself comfortable before reading in data - expect frustration and pain!

- General idea: you import the data and bind it to an R object - usually a `data.frame`
- Then you proceed with transforming this `data.frame` until it satisfies the demands for tidy data
- Then you save the data under a new name, save the script, and celebrate yourself 🎉🍾🥂
- We will cover the transformation steps in the next session

Summary and outlook

- We now know how to organise our working directory and how to import data
- Next time we will learn how to transform imported raw data into tidy data
  - This is the kind of data that is the vantage point for visualisation and modelling

→ Learn to produce the data you have used as input for visualisations yourself

Tasks until next time:

1. Fill in the **quick feedback survey** on Moodle
2. Read the **tutorial on data import and project management**
3. Do the **exercises** provided on the course page and **discuss problems** and difficulties via the Moodle forum
4. **Create a project** that you can use in the next session and already download and allocate the raw data posted on the course page