Reading and writing files and the concept of Working Directory

Silvie Cinková

2025-07-24

Table of contents

1 Working directory

- folder from which R sees other files and folders

Print path to your current Working Directory

```
 getwd() # in the console/R file
[1] "/lnet/aic/personal/cinkova/NPFL112_2025_ZS"
```

2 Set a different Working Directory

- ~ your Home
- .. one folder up
- . current folder

```
#setwd("~/folder/subfolder/")
```

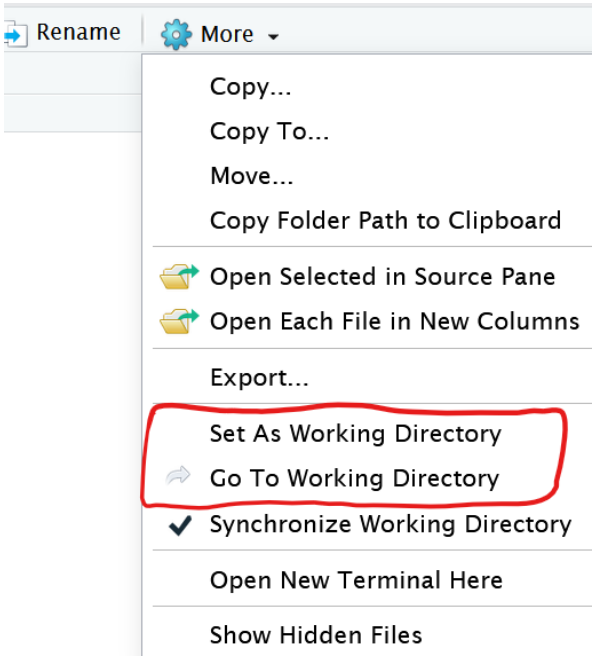

Figure 1: Interactive control of Working Directory location

3 RStudio Projects

- `.Rproj` file stores project configuration
- When you open this project next time, it tries to restore the work space from last time.

4 Working directory in Quarto code chunks

- working directory = **location of the qmd script file**

```
getwd()
```

```
[1] "/lnet/aic/personal/cinkova/SCRIPTS.NPFL112"
```

Quarto does not care which folder you said was your working directory. The working directory for Quarto is always where the qmd file is.

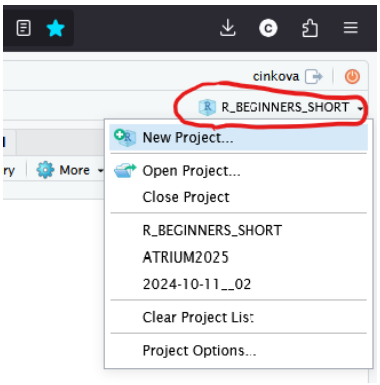

Figure 2: Project List in RStudio

5 How to render your copy of this file

- store part of file path in a variable
- combine it with file names by using `file.path()`

```
project_path <- "~/NPFL112_2025_ZS" # Silvie's path.
# Replace with something that would work for you.
file.path(project_path, "DATA.NPFL112")
```

```
[1] "~/NPFL112_2025_ZS/DATA.NPFL112"
```

```
datasaving_folder <- "DATA.NPFL112" # change this
# you must create your own folder you can write in!
```

I store all materials to this course in a subfolder and in this script I start to read files from other folders. If you decide to reuse parts of my scripts from the SCRIPTS.NPFL112 folder and still want to be able to render them, you need all paths to work for you in your setup. The best option is that you copy the script right to your home folder and replace this `project_path` variable with just "~".

6 List files from DATA.NPFL112

```
list.files(file.path(project_path, "DATA.NPFL112"))
```

```
 [1] "billionaires_combined.tsv"
 [2] "DataGeographies-v2-by-Gapminder.xlsx"
 [3] "ddf--gapminder--systema_globalis_files_from_GitHubAPI.json"
 [4] "Founders_Network.csv"
 [5] "gapminder_billionaires_ddf--entities--person.csv"
 [6] "gapminder_countries.tsv"
 [7] "gapminder_ddf--concepts.csv"
 [8] "gapminder_ddf--datapoints--daily_income--by--person--time.csv"
 [9] "gapminder_firstmarriage.csv"
[10] "gapminder_geonames.xlsx"
[11] "gapminder_hourly_labour_cost_constant_2017_                          usd--by--geo--time.c
[12] "gapminder_hourly_labour_cost_constant_2017_usd--by--geo--time.csv"
[13] "gapminder_laborcost_cze_deu.csv"
[14] "gapminder_metadata_filenames.tsv"
[15] "GitHubURLs_Gapminder_SystemaGlobalis.tsv"
[16] "jrc_1.tsv"
[17] "jrc_2.tsv"
[18] "jrc_3.tsv"
[19] "jrc_4.tsv"
[20] "jrc_latin_4.tsv"
[21] "jrc_latin.tsv"
[22] "JRC_Names"
[23] "JRC_Names.tsv"
[24] "migrants.tsv"
[25] "titanic.csv"
```

7 Open a csv file from DATA.NPFL112

```
marriage <- read.csv(file = file.path(project_path, datasaving_folder,
 ↪  "gapminder_firstmarriage.csv"))
head(marriage)
```

```
      country    X2005
1 Afghanistan 17.83968
2     Albania 23.32651
3     Algeria 29.60000
4      Angola       NA
5   Argentina 23.26396
6     Armenia 22.98603
```

You have seen this last time in a bare-R file!

8 Individual files from (GitHub) URL

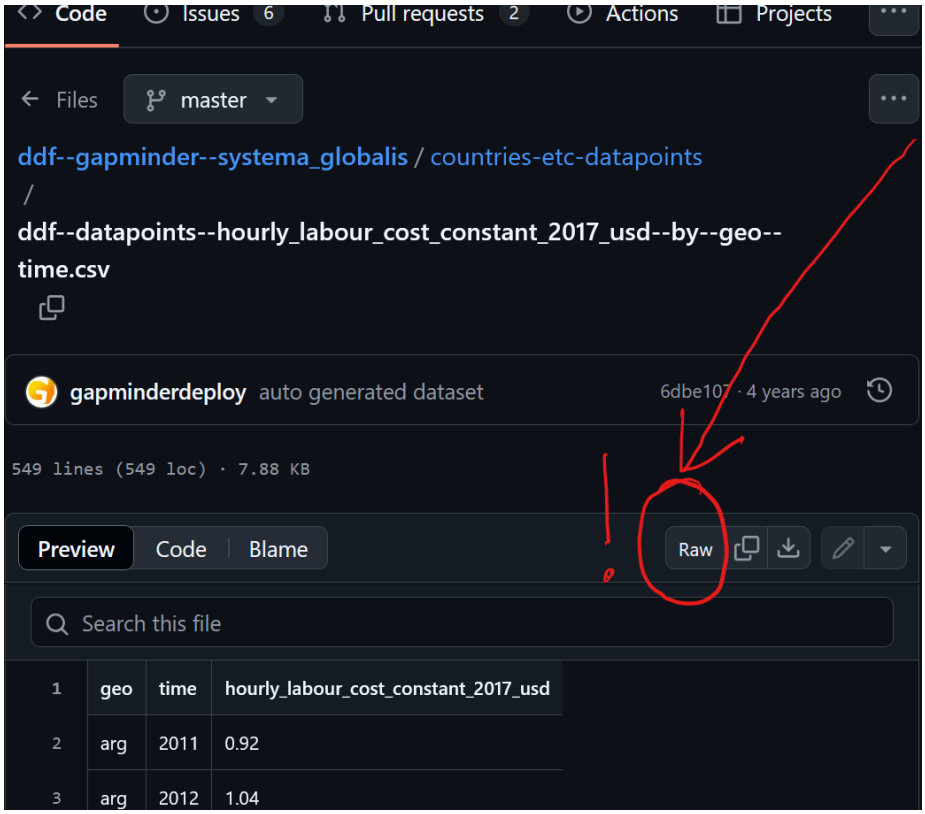

Figure 3: GitHub default view

When you look for interesting data sets, you will find plenty in the Internet. To download a file from the Internet, you must use its URL. In ordinary websites you just copy it from the address/URL bar at the top or you right-click your mouse at a link and select "Copy link".

This example is slightly more complex but particularly useful for you as data scientists. What you see is not an ordinary website, but the user interface of **GitHub**. GitHub is a public repository of files, like Google Drive, but optimized for programmers to facilitate collaboration on software. On GitHub, you must switch to a "Raw" view to obtain the correct URL. Otherwise you would download the html code of the entire interface website you are viewing, instead of the selected file!

Downloading individual files is not the typical use case for GitHub (cf. `git` and GitHub's API), but that would be an entirely different topic.

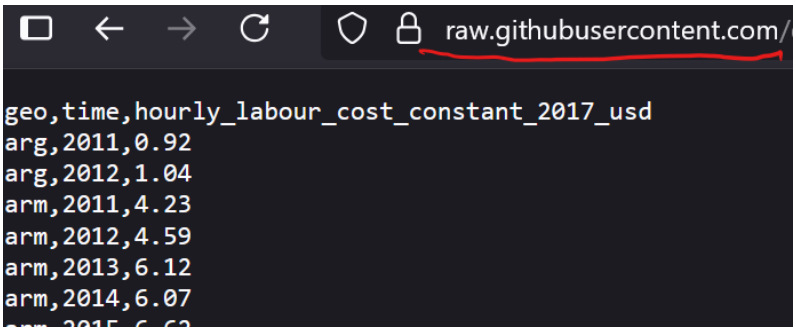

Figure 4: Switched to raw file URL

9 Download a file (e.g. from GitHub)

```
library(glue) # enables multiline with \\
URL <- glue(
 ↪  "https://raw.githubusercontent.com/open-numbers/ddf--gapminder--systema_globalis/refs/heads/mast
 ↪  )
my_destination <- glue(file.path(project_path, datasaving_folder, "\\
gapminder_hourly_labour_cost_constant_2017_\\
usd--by--geo--time.csv"))
```

```
download.file(
  url = URL,
  destfile = my_destination
  )
```

This is a code that downloads a file from its URL to a destination in your computer (or to your user account on a cloud). The important piece is the `download.file` function. Do not bother about the `glue` library. I had to use it to fit long strings such as URL on a slide.

The `download.file` function is universal to download any file from anywhere. Sometimes you can copy a download link from a website and use this URL to download the file programmatically.

This is how to download some data from GitHub, which is a bit specific. Here I work with data from Gapminder on Github. Their repository is very large and this was a largely random pick: `https://github.com/open-numbers/ddf-gapminder-systema_globalis/tree/master/countries-etc-datapoints`. This repository contains a table that explains each data set, but I am going to select one that is intelligible without reading much metadata. It is going to be a table about average labor cost in a given country in a given

year: https://raw.githubusercontent.com/open-numbers/ddf–gapminder–systema_globalis/refs/heads/master/countries-etc-datapoints/ddf–datapoints–hourly_labour_cost_constant_2017_usd–by–geo–time.csv.

Manually navigate to the file you want and copy its URL. Mind to use the URL that appears when you hit the `Raw` button (starting with `https://raw.githubusercontent`) to download the contents of the file. On the default `https://github.com/….` you would only download a html file of the website you are seeing.

Use the `download.file` function. Leave all arguments at default, except `url` and `destfile`. Put the file into the the folder where you can store your own data. Use the end part of the original file name and give it a prefix `gapminder_` and keep doing this with all files that you happen do download from this source. This will help you keep a system in your files.

10 https://www.gapminder.org/

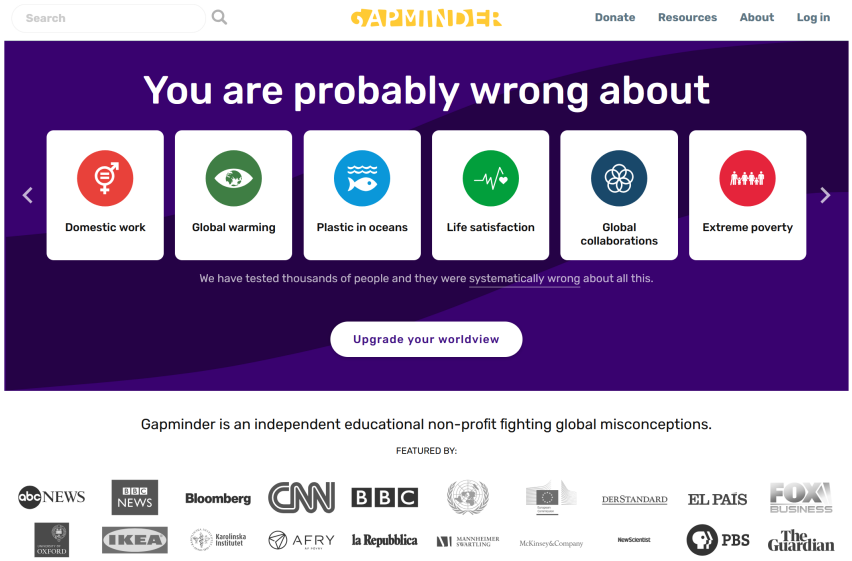

Figure 5: Gapminder

11 Read files with base R or `tidyverse`

- File too big to open in a text editor?
- Inspect it reading it as text lines (first 3 lines)

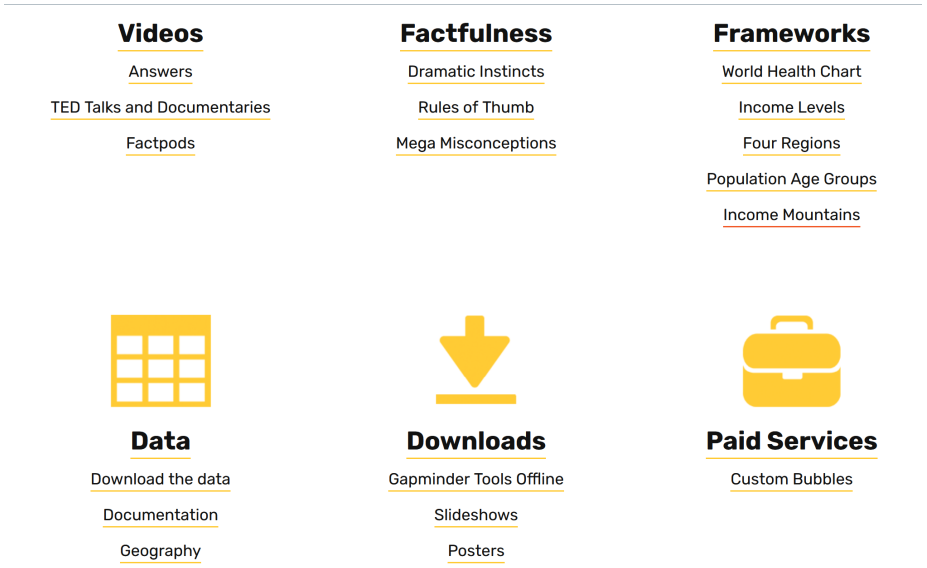

Figure 6: Introducing Gapminder

```
mypath <- glue(file.path(project_path, datasaving_folder,
 ↪  "gapminder_hourly_labour_cost_constant_2017_\\
usd--by--geo--time.csv"))
```

```
library(readr)
read_lines(
  file = mypath,
  n_max = 3)
```

```
[1] "geo,time,hourly_labour_cost_constant_2017_usd"
[2] "arg,2011,0.92"
[3] "arg,2012,1.04"
```

```
readLines(
  con = mypath,
  n = 3)
```

```
[1] "geo,time,hourly_labour_cost_constant_2017_usd"
[2] "arg,2011,0.92"
[3] "arg,2012,1.04"
```

What you are seeing are the first three lines of a tabular file we have just read as a text file, assuming no columns or headers. This comes handy when you do

not know the structure of the file and it is too large to open interactively in a text editor, for instance. So you read lines of text. In plain text files, each line ends when the author hit Enter. (So in our terms it is rather a paragraph than a line!)

A tabular file is a plaintext file where each line is one table row and the columns are on each line separated by the same character (throughout the file). The best-known tabular format is **comma-separated values** (`csv`). The original U.S. format uses comma. The European csv uses semicolons because comma is often reserved for the decimal operator (vs. decimal point in the U.S.). To skip these issues altogether, you better save your files as `tsv` (tab-separated values).

In the code above you see two functions that look similar and whose output looks exact the same. One is a base-R function, the other is from a `tidyverse` package called `readr`. Feel free to choose either and just make a mental note that there is an alternative. Sometimes, when a file is tricky to read in with one function, it goes well with the other.

Look at the Help to either function and explore its other arguments using the file you have just loaded.

12 Reading a table with `readr`

- `read_csv`, `read_csv2`, `read_tsv`: tailored to the common separators `,`, `;`, tab
- `read_delim` : you name the separator (aka delimiter), more arguments

```
read_csv(file = mypath,
         n_max = 3) #just top 3 rows
```

```
Rows: 3 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): geo
dbl (2): time, hourly_labour_cost_constant_2017_usd

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

# A tibble: 3 x 3
  geo    time hourly_labour_cost_constant_2017_usd
  <chr> <dbl>                                <dbl>
1 arg    2011                                 0.92
2 arg    2012                                 1.04
3 arm    2011                                 4.23
```

13 Other arguments in `read_csv`

```
read_csv(file = mypath,
         col_names = c("country", "year", "USD_hour_2017"),
         n_max = 3)
```

```
Rows: 3 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): country, year, USD_hour_2017

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# A tibble: 3 x 3
  country year  USD_hour_2017
  <chr>   <chr> <chr>
1 geo     time  hourly_labour_cost_constant_2017_usd
2 arg     2011  0.92
3 arg     2012  1.04
```

14 Read directly from URL

```
URL2 <- glue(
 ↪  "https://raw.githubusercontent.com/open-numbers/ddf--gapminder--\\
systema_globalis/refs/heads/master/countries-etc-datapoints/\\
ddf--datapoints--hourly_labour_cost_constant_2017_usd--by--geo--\\
time.csv")
read_csv(file = URL2,
         n_max = 3)
```

```
Rows: 3 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): geo
dbl (2): time, hourly_labour_cost_constant_2017_usd

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# A tibble: 3 x 3
  geo    time hourly_labour_cost_constant_2017_usd
  <chr> <dbl>                                <dbl>
1 arg    2011                                 0.92
2 arg    2012                                 1.04
3 arm    2011                                 4.23
```

15 Download an Excel file

```
URL3 <- glue("https://docs.google.com/spreadsheets/d/1qHalit8s\\
XC0R8oVXibc2wa2gY7bkwGzOybEMTWp-08o/export?format=xlsx")
download.file(url = URL3,
              destfile =
                file.path(project_path, datasaving_folder,
 ↪  "gapminder_geonames.xlsx"),
              mode = "wb") # mind the mode
```

With Windows formats and on Windows-operated computers, set `mode` to `wb`. Otherwise the file may get corrupted during the transmission.

16 Read Excel

- `readxl` reads only local file paths, not URLs.

```
library(readxl)
read_xlsx(path = file.path(project_path, datasaving_folder,
 ↪  "gapminder_geonames.xlsx"),
          n_max = 3) # just three rows
```

```
New names:
* `` -> `...2`
* `` -> `...3`
* `` -> `...5`
```

```
# A tibble: 3 x 7
  Data: Geographies - v~1 ...2  ...3  Free data from www.g~2 ...5  id    version
  <chr>                   <chr> <lgl> <chr>                  <lgl> <chr> <chr>
1 Updated: July 1, 2021   <NA>  NA    CC BY 4.0 LICENCE      NA    geo   v2
2 Concept:                Geog~ NA    Are you seeing this o~ NA    <NA>  <NA>
3 Unit:                   <NA>  NA    gapm.io/datageo        NA    <NA>  <NA>
# i abbreviated names: 1: `Data: Geographies - v2`,
#   2: `Free data from www.gapminder.org`
```

```
# readxl::read_xlsx(path =
 ↪  "datasets_ATRIUM/DataGeographies-v2-by-Gapminder.xlsx") #the same
 ↪  file
```

17 Excel sheets listed

- `read_xlsx` reads the first sheet by default
- Have the spreadsheets listed:

```
readxl::excel_sheets(path = file.path(project_path, datasaving_folder,
 ↪  "gapminder_geonames.xlsx"))
```

```
[1] "ABOUT"                 "list-of-countries-etc" "list-of-regions"
[4] "list-of-income-levels" "global"                "geo-names"
```

```
readxl::read_xlsx(path = file.path(project_path, datasaving_folder,
 ↪  "gapminder_geonames.xlsx"), sheet = 2,
                  n_max = 3) # or sheet = "list-of-countries-etc"
```

```
# A tibble: 3 x 13
  geo   name    four_regions eight_regions six_regions members_oecd_g77 Latitude
  <chr> <chr>   <chr>        <chr>         <chr>       <chr>                <dbl>
1 aus   Austra~ asia         east_asia_pa~ east_asia_~ oecd                 -25
2 brn   Brunei  asia         east_asia_pa~ east_asia_~ g77                    4.5
3 khm   Cambod~ asia         east_asia_pa~ east_asia_~ g77                   13
# i 6 more variables: Longitude <dbl>, `UN member since` <dttm>,
#   `World bank region` <chr>, `World bank, 4 income groups 2017` <chr>,
#   `World bank, 3 income groups 2017` <chr>, UNHCR <chr>
```

18 Google sheets

- inspect it manually and pick one worksheet

```
library(googlesheets4)
shURL <- glue("https://docs.google.com/spreadsheets/d/1qHalit8sXC\\

 ↪  0R8oVXibc2wa2gY7bkwGzOybEMTWp-08o/edit?gid=425865495#gid=425865495"
 ↪  )
gs4_deauth() # skip logging in at GoogleDrive
googlesheets4::read_sheet(shURL, sheet = 2,
                          n_max = 3)
```

```
v Reading from "Data Geographies - v2 - by Gapminder".

v Range ''list-of-countries-etc''.

# A tibble: 3 x 13
  geo   name    four_regions eight_regions six_regions members_oecd_g77 Latitude
  <chr> <chr>   <chr>        <chr>         <chr>       <chr>                <dbl>
1 aus   Austra~ asia         east_asia_pa~ east_asia_~ oecd                 -25
2 brn   Brunei  asia         east_asia_pa~ east_asia_~ g77                    4.5
3 khm   Cambod~ asia         east_asia_pa~ east_asia_~ g77                   13
# i 6 more variables: Longitude <dbl>, `UN member since` <dttm>,
#   `World bank region` <chr>, `World bank, 4 income groups 2017` <chr>,
#   `World bank, 3 income groups 2017` <chr>, UNHCR <chr>
```

19 Saving tabular files

```
gapminder_countries <- readxl::read_xlsx(file.path(project_path,
  datasaving_folder, "gapminder_geonames.xlsx"),
                                         sheet = 2,
                                         n_max = 3)
readr::write_tsv(x = gapminder_countries,
                 file = file.path(project_path,
                   datasaving_folder, "gapminder_countries.tsv"))
```

20 Create a folder

- create a file to save your exercise scripts

```
dir.create(path = "HOMEWORK",
mode = '750', recursive = TRUE )
```

```
Warning in dir.create(path = "HOMEWORK", mode = "750", recursive = TRUE):
'HOMEWORK' already exists
```

`mode` = **octal notation** (access rights to file, just Unix)With `mode = '750'` you allow other students and teachers to see and execute files in this folder.

21 List files into a vector

```
list.files(path = project_path, recursive = FALSE,
  include.dirs = FALSE, pattern = "qmd", full.names = TRUE)
```

```
character(0)
```

- list files in a folder
- just those with `qmd` in their names
- `recursive`: search in subfolders?

22 Set up a file path

- creates a character vector, not a file!

```
file.path(project_path, "a_new_file.XXX")
```

```
[1] "~/NPFL112_2025_ZS/a_new_file.XXX"
```