

Data in- and export

Hauke Licht
University of Cologne

April 17, 2024

File Systems

Overview

- Understanding paths and file operations in R
- Key functions: `getwd()`, `file.path`, `basename`, `dirname`, `dir.exists`, `file.exists`, `create.dir`

File systems

- Files are organized in folders and subfolders in a hierarchical structure
- The **root folder** is the top-most directory in the hierarchy
- Paths are used to navigate and locate files in the system

Example

```
~  
├── Desktop  
│   ├── file.txt  
│   ├── subfolder  
│   │   ├── file2.txt  
│   │   └── file3.txt  
│   └── another_subfolder  
│       └── file4.txt  
└── ...
```

~ represents your **home directory**

```
1 # show the absolute path of your home directory  
2 path.expand("~/")
```

```
[1] "/Users/hlicht"
```

Paths can be *absolute* or *relative*

- **Absolute path:** Full path from the root directory,
e.g. `/home/hlicht/Desktop/file.txt`
- **Relative path:** Path relative to the current working directory,
e.g. `subfolder/file2.txt` (when `~/Desktop` is your working directory)

getwd()

- the **working directory** is the current directory where R searches for files
- `getwd()` retrieves the current working directory

Example

```
1 # print the current working directory
2 getwd()
```

```
[1] "/Users/hlicht/Dropbox/teaching/text_wrangling_in_r/slides"
```

Note: These slides are created with quarto, which always sets the working directory to the folder that contains the .qmd file. Hence, we are in the `slides/` folder.

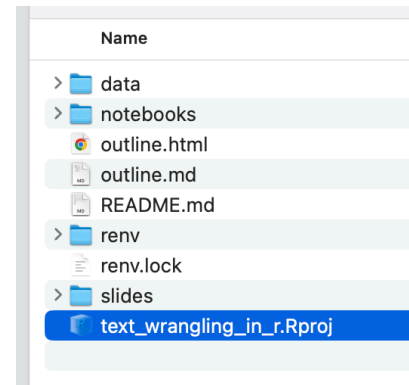
R Projects

- **R Projects** set the root directory to make paths compatible across user
- This makes the project folder the root folder
- so we can use relative paths to locate files in the project folder

Opening an R project

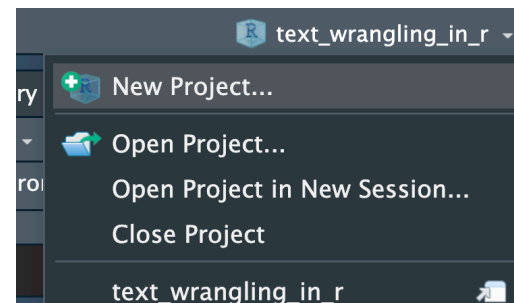
Option 1 🙋

1. locate the .Rproj file in a folder (e.g., “text_wrangling_in_r.Rproj”)
2. double-click the file to open the project in RStudio



Option 2 🙋

Select an existing R project in R Studio

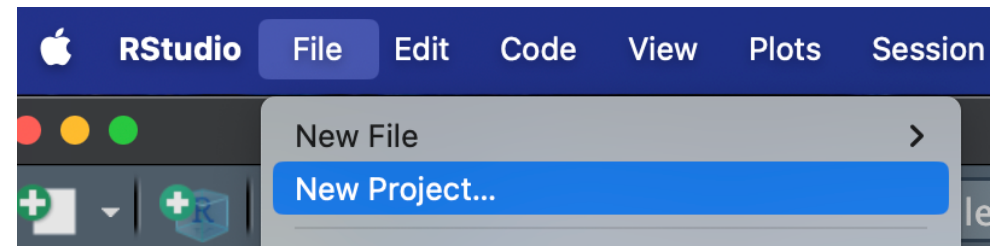


R Projects

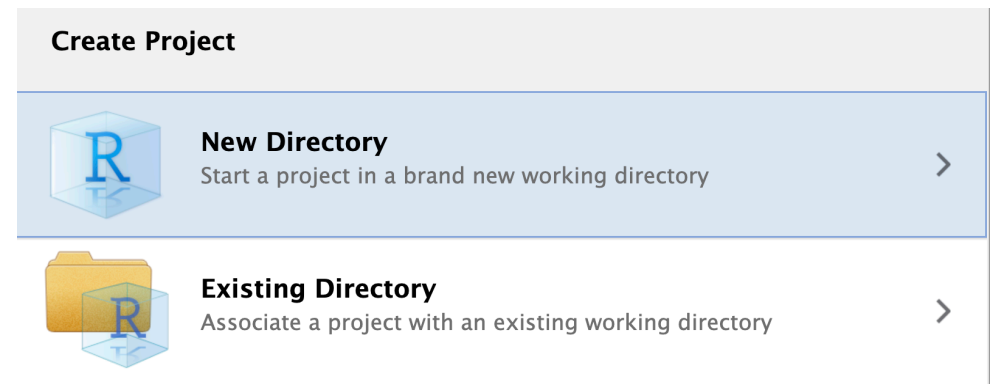
- **R Projects** set the root directory to make paths compatible across user
- This makes the project folder the root folder
- so we can use relative paths to locate files in the project folder

Creating a new R project

1. Open RStudio
2. In the program menu, click on “File” → “New Project”
3. Choose
 1. “Existing Directory” if you have already have a folder with R scripts → select the location of the folder
 2. “New Directory” otherwise → specify the location and name of the new folder
4. Click “Create Project”



Navigate in R Studio “File” Menu 🖱️



Choose create from new or existing directory 🖱️

Useful file system functions in R

`file.path`

- Generates system-specific paths
- Utilizes `.Platform$file.sep` for compatibility

Example

```
1 # create a path under the current system
2 fp <- file.path("folder", "subfolder", "file.txt")
3 fp
```

```
[1] "folder/subfolder/file.txt"
```

Useful file system functions in R

`basename` and `dirname`

- `basename` for obtaining the file name from path
- `dirname` for obtaining the directory part of path

Example

```
1 fp <- file.path("folder", "subfolder", "file.txt")
2 # print file name
3 basename(fp)
```

```
[1] "file.txt"
```

```
1 # print directory path
2 dirname(fp)
```

```
[1] "folder/subfolder"
```


Useful file system functions in R

`dir.exists` and `file.exists`

- Checks if directories and files exist
- `dir.exists` for directories, `file.exists` for files

Example

```
1 # check existence (in slides/ folder)
2 dir.exists("01-data_io_files")
```

```
[1] FALSE
```

```
1 dir.exists("yfgsx")
```

```
[1] FALSE
```

```
1 file.exists("01-data_io.qmd")
```

```
[1] TRUE
```

```
1 file.exists("yfgsx.txt")
```

```
[1] FALSE
```

Useful file system functions in R

`dir.create` and `unlink`

- Function to create and remove directories
- Handles the creation of non-existent directories

Example

```
1 # create a directory
2 dir.create("new_folder")
3 # check
4 dir.exists("new_folder")
```

```
[1] TRUE
```

```
1 # remove
2 unlink("new_folder", recursive = TRUE) # see https://stackoverflow.com/q/28097035
3 # check
4 dir.exists("new_folder")
```

```
[1] FALSE
```

File import from local

File Formats

tabular and non-tabular, structured and unstructured formats

- **Tabular:** “2-dimensional” data organized in rows and columns, e.g., CSV, TSV, Excel
- **Non-tabular:** Data in other formats, e.g., JSON, XML, HTML, PDF, Word
- **Structured:** Data with a defined schema, e.g., CSV, JSON, XML
- **Unstructured:** Data without a defined schema, e.g., PDF, Word

Tabular data

Overview

- Importance of managing tabular data
- CSV, TSV and their functions

Comma/Tab Separated

- `readr::read_csv` for reading *comma-separated* file (CSV) with extension “.csv”,
`readr::read_tsv` for reading *tab-separated* file (TSV) with extensions “.tsv”
- `readr::read_delim` for custom delimiters (e.g., “;” for semicolon-separated files)

```
1 # | warning: false
2 library(readr)
```

Examples

```
1 # read CSV file
2 fp <- file.path("../", "data", "tabular", "test.csv")
3 df <- read_csv(fp)
```

```
1 # read CSV file
2 fp <- file.path("../", "data", "tabular", "test.tsv")
3 df <- read_tsv(fp)
```

MS Excel files

- Using `readxl::read_excel` to read Excel files
- Handles “.xlsx” files effectively

Example

```
1 library(readxl)
2 # read Excel file
3 fp <- file.path("../data/tabular/test.xlsx")
4 df <- read_excel(fp)
```

Non-Tabular

Overview

- Unstructured and structured non-tabular data
- Handling formats like JSON, XML, HTML

Unstructured

MS Word files (.docx)

- Reading MS Word documents using `officer::read_docx`
- Handling .docx files in data analysis

Example

```
1 library(officer)
2 # read Word document
3 fp <- file.path("../", "data", "files", "test_file.docx")
4 doc <- read_docx(fp)
5 content <- docx_summary(doc)
6 content
```

	doc_index	content_type	style_name	
1	1	paragraph	Title	
2	2	paragraph	Author	
3	3	paragraph	First Paragraph	

		text
1		Test file
2		Hauke Licht
3	This is just a text document for illustrating how to read word and PDF files into R.	

	level	num_id
1	NA	NA
2	NA	NA
3	NA	NA

Unstructured PDF files (.pdf)

- Extract text from PDF using `pdftools::pdf_text`
- Useful for text processing

Example

```
1 library(pdftools)
2 # extract text from PDF
3 fp <- file.path("../data/files/test_file.pdf")
4 doc <- pdf_text(fp)
5 doc
```

```
[1] "
Test file\n
is just a text document for illustrating how to read word and PDF files into R.\n"
```

```
Hauke Licht\n\nThis
```

Structured JSON

- Reading JSON files with `jsonlite::read_json`
- Common in web data and configurations

Example

```
1 library(jsonlite)
2 # read JSON file
3 fp <- file.path("../", "data", "nontabular", "test.json")
4 data <- read_json(fp)
5 data
```

```
$null_field
NULL
```

```
$logical_field
[1] TRUE
```

```
$numeric_field
[1] 1
```

```
$string_field
[1] "a value"
```

```
$list_field
$list_field[[1]]
[1] "a"
```

```
$list_field[[2]]
[1] "list"
```

```
$list_field[[3]]
```

```
[1] "of"
```

```
$list_field[[4]]
```

Structured JSONlines (.jsonl)

- Combining `readr::read_lines`, `purrr::map`, `jsonlite::fromJSON`
- Efficient for large sets of JSON objects

Example

```
1 library(readr)
2 library(purrr)
3 library(jsonlite)
4 # read JSON lines and convert
5 fp <- file.path("../", "data", "nontabular", "test.jsonl")
6 lines <- read_lines(fp)
7 data <- map(lines, fromJSON)
8 data
```

```
[[1]]
[[1]]$id
[1] "001"
```

```
[[1]]$text
[1] "I'm sorry, I don't understand. Can you try again?"
```

```
[[2]]
[[2]]$id
[1] "002"
```

```
[[2]]$text
[1] "What is the average length of an elephant's ear?"
```

Structured XML files

- `xml2::read_xml` to read XML files
- Widely used in web data, configurations

Example

```
1 library(xml2)
2 # read XML file
3 fp <- file.path("../", "data", "files", "example.xml")
4 data <- read_xml(fp)
5 data
```

```
{xml_document}
<library>
[1] <book id="1">\n  <title>The Great Gatsby</title>\n  <author>F. Scott Fitz ...
[2] <book id="2">\n  <title>To Kill a Mockingbird</title>\n  <author>Harper L ...
[3] <book id="3">\n  <title>1984</title>\n  <author>George Orwell</author>\n  ...
```

Structured HTML

- `xml2::read_html` to read HTML content
- Useful for web scraping, data extraction from websites

Example

```
1 library(xml2)
2 # read HTML file
3 fp <- file.path("../", "data", "files", "example.html")
4 data <- read_html(fp)
5 data
```

```
{html_document}
<html lang="en">
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
[2] <body>\n    <h1>Library Catalog</h1>\n    <table>\n<thead><tr>\n<th>ID</t ...
```

Data import from external sources

Overview

Many commonly used political (text) dataset are available online

- *ParlSpeech2*
- the *Manifesto Project* corpus
- the *Comparative Agendas Project* (CAP) corpora

For replicability and version control purposes, it's a **best practice** to program the download of these data (instead of manually downloading and saving them)

Import from *Harvard Dataverse*

Many replication materials for articles published in political science journals are available through [Harvard Dataverse](#):

Many journals have their own “dataverses”. Here some:

- *American Political Science Review* (APSR):
https://dataverse.harvard.edu/dataverse/the_review
- *Political Analysis*: <https://dataverse.harvard.edu/dataverse/pan>
- *The Journal of Politics* (JOP): <https://dataverse.harvard.edu/dataverse/jop>
- *Political Science Research & Methods* (PSRM): <https://dataverse.harvard.edu/dataverse/PSRM>

IMPORTANT: In the URLs listed above, the last part behind the last “/” is called “Dataverse ID” – we need this to automatically download files from a journals dataverse

Import from *Harvard Dataverse*

Example 1: downloading with the persistent file URL

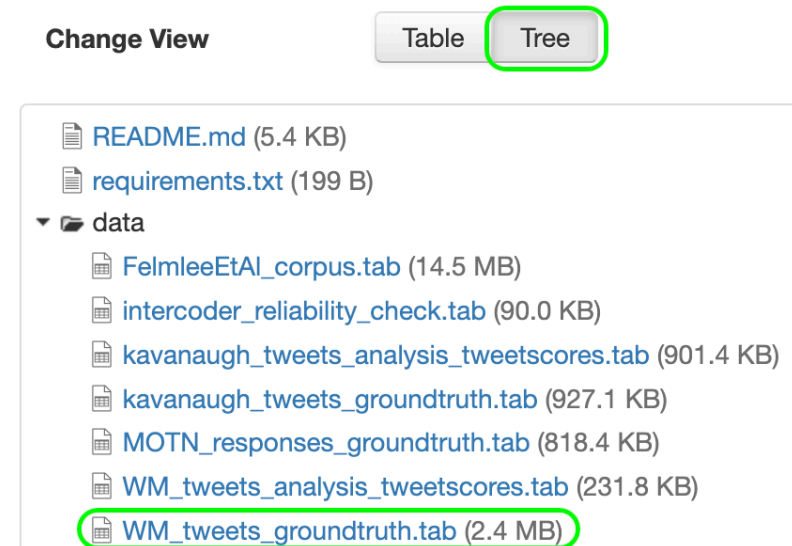
We will use the replication data of the article

Bestvater, S., & Monroe, B. (2023). Sentiment is Not Stance: Target-Aware Opinion Classification for Political Text Analysis. *Political Analysis*, 31(2), 235-256.

The repository is <https://doi.org/10.7910/DVN/MUYYG4>

Step 1. locate the file we want to download

1. go to <https://doi.org/10.7910/DVN/MUYYG4>
2. in the “Files” panel, click “Tree”
3. in the data folder, find and click on the file ‘WM_tweets_groundtruth.tab’
4. on the files page, go to the “Metadata” tab
5. get the value in the field “Download URL”



Example 1: downloading with the persistent file URL

We will use the replication data of the article

Bestvater, S., & Monroe, B. (2023). Sentiment is Not Stance: Target-Aware Opinion Classification for Political Text Analysis. *Political Analysis*, 31(2), 235-256.

The repository is <https://doi.org/10.7910/DVN/MUYYG4>

Step 2. read the file

```
1 file_url <- "https://dataverse.harvard.edu/api/access/datafile/5374866"
2 # we use `read_tsv` because the file we want to download is a .tab file, i.e. "tab-separated"
3 df <- read_tsv(file_url)
```

Import from *Harvard Dataverse*

Example 2: download with file persistent ID

We will use the replication data for the article

Barberá, P., Boydston, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated Text Classification of News Articles: A Practical Guide. *Political Analysis*, 29(1), 19–42.

The repository is <https://doi.org/10.7910/DVN/MXKRDE>

Step 1. load the ‘dataverse’ package and set the necessary environment variables

```
1 library(dataverse)
2 Sys.setenv("DATAVERSE_SERVER" = "dataverse.harvard.edu")
3 Sys.setenv("DATAVERSE_ID" = "pan") # set to Political Analysis dataverse ID !
```

Example 2: dowload with file persistent ID

We will use the replication data for the article

Barberá, P., Boydston, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated Text Classification of News Articles: A Practical Guide. *Political Analysis*, 29(1), 19–42.

The repository is <https://doi.org/10.7910/DVN/MXKRDE>

Step 2. locate the file we want to download

- 1. go to <https://doi.org/10.7910/DVN/MXKRDE>
- 2. search for the file ‘ground-truth-dataset-cf.tab’
- 3. on the files page, go to the “Metadata” tab
- 4. get the value in the field “File Persistent ID”

```
1 persistent_id <- "doi:10.7910/DVN/MXKRDE/EJTMLZ"
```

ground-truth-dataset-cf.tab

This file is part of "Replication Data for: Automated Text Classification of News Articles: A Practical Guide".

Version 1.2

File Citation

Barberá, Pablo; Boydston, Amber; Linn, Suzanna; McMahon, Ryan; Nagler, Jonathan, 2020, "ground-truth-dataset-cf.tab", *Replication Data for: Automated Text Classification of News Articles: A Practical Guide*, <https://doi.org/10.7910/DVN/MXKRDE/EJTMLZ>, Harvard Dataverse, V1, UNF:6:XEWsWmHLgoKE+DODc2aUSg== [fileUNF]

[Cite Data File](#)

[Learn about Data Citation Standards.](#)

File Persistent ID	doi:10.7910/DVN/MXKRDE/EJTMLZ
File UNF	UNF:6:XEWsWmHLgoKE+DODc2aUSg==
Original File MD5	567e4af619f77cb5f359b641ce9aafa2

Example 2: download with file persistent ID

We will use the replication data for the article

Barberá, P., Boydston, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated Text Classification of News Articles: A Practical Guide. *Political Analysis*, 29(1), 19–42.

The repository is <https://doi.org/10.7910/DVN/MXKRDE>

Step 3. download the file and read it into R

```
1 df <- get_dataframe_by_doi(  
2   # use the file persistent ID to specify which file to download  
3   filedoi = persistent_id,  
4   # pass the appropriate file reading function (from the readr package)  
5   .f = read_tsv  
6 )
```

Download from GitHub

- Github is a code sharing and open-source collaboration platform.
- Some researchers use it to store make available the replication materials

We will use the example of the article

van Atteveldt, W., van der Velden, M. A. C. G. & Boukes, M. (2021) The Validity of Sentiment Analysis: Comparing Manual Annotation, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms. *Communication Methods and Measures*, 15(2), 121-140.


The repository is <https://github.com/vanatteveldt/ecosent>

Download from GitHub

Step 1. locate the files we want to download

1. go to
<https://github.com/vanatteveldt/ecosent>
2. click on the “data” folder
3. get gold sentences’ texts: in the ‘raw’ subfolder,
 1. find the file ‘gold_sentences.csv’
 2. click on the file
 3. click on the “Raw” button
 4. copy the URL of the raw file

[ecosent](#) / [data](#) / [raw](#) /

 **vanatteveldt** Add zero-entries for dic

Name
..
dictionaries
f1345162.csv
f1386488.csv
gold_sentences.csv

[ecosent](#) / [data](#) / [raw](#) / [gold_sentences.csv](#)

 **vanatteveldt** Refactored and added Mariken's off the shelf script cb32fcd · 4 years ago [History](#)

Preview

Code


Blame

301 lines (301 loc) · 80.4 KB

[Raw](#)







	id	headline
1		
2	10367	Dijsselbloem pessimistisch over snelle stappen Grieken

```
1 gold_sentences_texts_url <- "https://raw.githubusercontent.com/vanatteveldt/ecosent/master/data/raw/gold_sentences"
```

Download from GitHub

Step 1. locate the files we want to download (*continued*)

4. get gold sentences' expert codings: in the intermediate 'subfolder'

1. find the file 'gold.csv'
2. click on the file
3. click on the "Raw" button
4. copy the URL of the raw file

```
1 gold_sentences_labels_url <- "https://raw.githubusercontent.com/vanatteveldt/ecosent/master/data/intermediate/gold"
```

Download from GitHub

Step 2. download the files and combine them

```
1 sentences_df <- read_csv(gold_sentences_texts_url)
2 labels_df <- read_csv(gold_sentences_labels_url)
3
4 colnames(sentences_df)
```

```
[1] "id"          "headline"    "google"      "deepl"
[5] "dutch_lemmas" "google_lemmas" "deepl_lemmas"
```

```
1 colnames(labels_df)
```

```
[1] "id"      "value"
```

Note: we use `read_csv` because the file we want to download is a .csv file

```
1 library(dplyr)
2
3 # compute number of labels per headline
4 labels_df |>
5   group_by(id) |>
6   summarise(n_labels = n()) |>
7   # count numbers of labels per headlines
8   count(n_labels)
```

```
# A tibble: 1 × 2
  n_labels      n
  <int> <int>
1         1  284
```

Note: each of 284 headlines has only one label

```
1 # combine texts and labels
2 gold_df <- inner_join(labels_df, sentences_df, by = "id")
```

