Data in- and export

Hauke Licht

University of Cologne

April 18, 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**

```r
# show the absolute path of your home directory
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

```r
# print the current working directory
getwd()
```

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

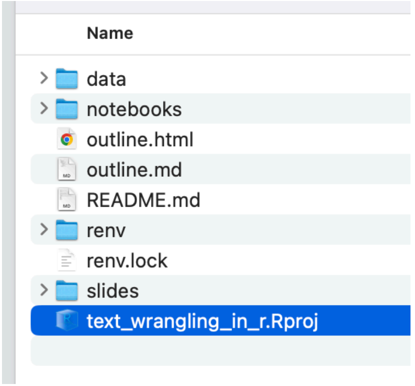
Note: These slides are created with quarto, which always sets the working directory to the folder that contains the .qmd file. Hence, we are in the `slides/` folder.

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

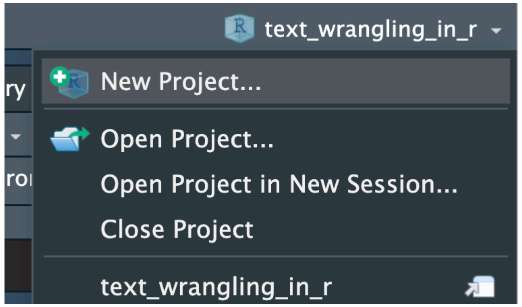
Option 1 👉



Name
data
notebooks
outline.html
outline.md
README.md
renv
renv.lock
slides
text_wrangling_in_r.Rproj

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 👉



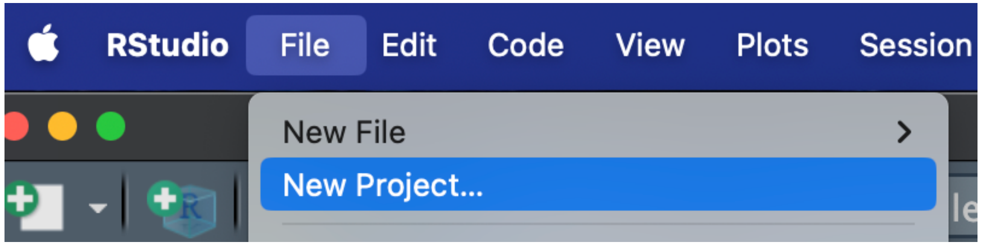
Select an existing R project in R Studio

R Projects

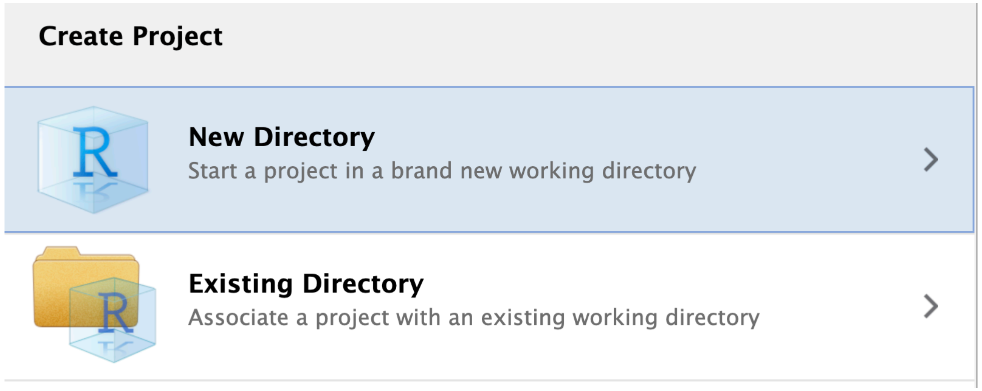
- **R Projects** set the root directory to make paths compatible across user
- This makes the project folder the root folder
- so we can use relative paths to locate files in the project folder

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 👆


Create Project

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

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

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

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

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

```r
# print directory path
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

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

```
[1] FALSE
```

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

```
[1] FALSE
```

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

```
[1] TRUE
```

```r
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

```r
# create a directory
dir.create("new_folder")
# check
dir.exists("new_folder")
```

```
[1] TRUE
```

```r
# remove
unlink("new_folder", recursive = TRUE) # see https://stackoverflow.com/q/28097035
# check
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)

```r
# | warning: false
library(readr)
```

Examples

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

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

MS Excel files

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

Example

```r
library(readxl)
# read Excel file
fp <- file.path("..", "data", "tabular", "test.xlsx")
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

```r
library(officer)
# read Word document
fp <- file.path("..", "data", "files", "test_file.docx")
doc <- read_docx(fp)
content <- docx_summary(doc)
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

```r
library(pdftools)
# extract text from PDF
fp <- file.path("..", "data", "files", "test_file.pdf")
doc <- pdf_text(fp)
doc
```

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

Structured

JSON

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

Example

```r
library(jsonlite)
# read JSON file
fp <- file.path("..", "data", "nontabular", "test.json")
data <- read_json(fp)
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

```r
library(readr)
library(purrr)
library(jsonlite)
# read JSON lines and convert
fp <- file.path("..", "data", "nontabular", "test.jsonl")
lines <- read_lines(fp)
data <- map(lines, fromJSON)
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

```r
library(xml2)
# read XML file
fp <- file.path("..", "data", "files", "example.xml")
data <- read_xml(fp)
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

```r
library(xml2)
# read HTML file
fp <- file.path("..", "data", "files", "example.html")
data <- read_html(fp)
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 poltical 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: dowloading with the persistent file URL

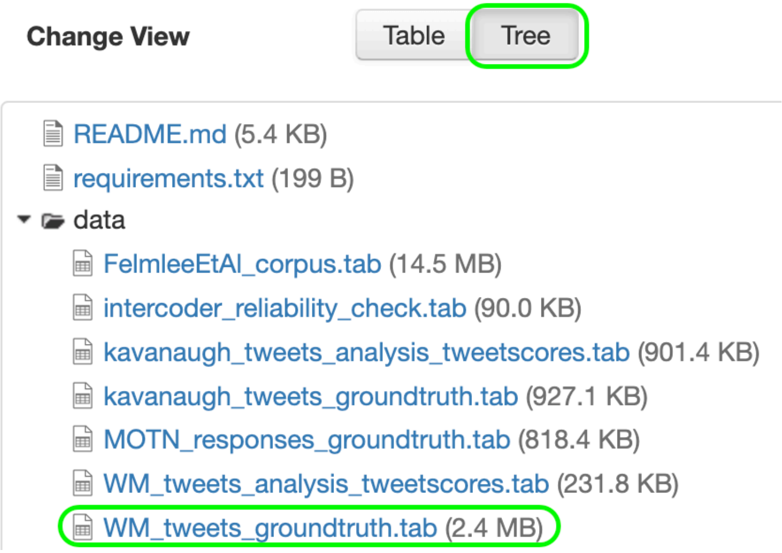
We will use the replication data of the article

> 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”



README.md (5.4 KB)
requirements.txt (199 B)
data
FelmleeEtAl_corpus.tab (14.5 MB)
intercoder_reliability_check.tab (90.0 KB)
kavanaugh_tweets_analysis_tweetscores.tab (901.4 KB)
kavanaugh_tweets_groundtruth.tab (927.1 KB)
MOTN_responses_groundtruth.tab (818.4 KB)
WM_tweets_analysis_tweetscores.tab (231.8 KB)
WM_tweets_groundtruth.tab (2.4 MB)

Example 1: dowloading 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

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

Import from *Harvard Dataverse*

Example 2: dowload with file persistent ID

We will use the replication data for the article

> Barberá, P., Boydstun, 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

```r
library(dataverse)
Sys.setenv("DATAVERSE_SERVER" = "dataverse.harvard.edu")
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

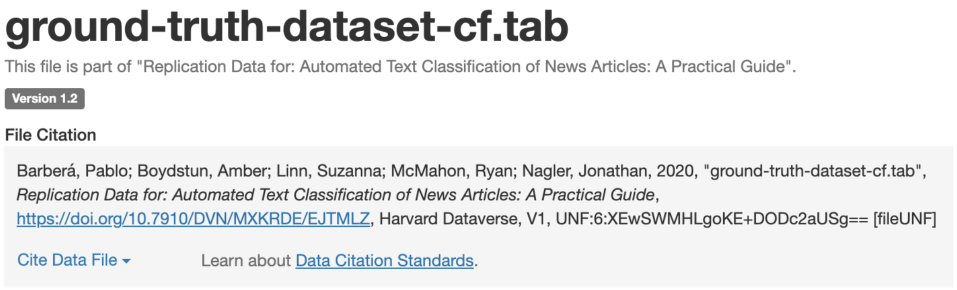
> Barberá, P., Boydstun, A. E., Linn, S., McMahon, R., & Nagler, J. (2021). Automated Text Classification of News Articles: A Practical Guide. *Political Analysis*, 29(1), 19–42.

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"

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



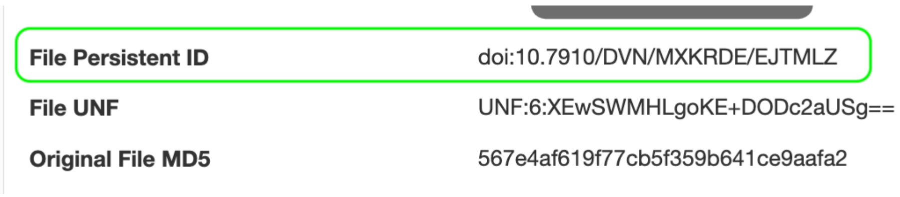
ground-truth-dataset-cf.tab

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

Version 1.2

File Citation

Barberá, Pablo; Boydstun, 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]



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

Example 2: dowload with file persistent ID

We will use the replication data for the article

> Barberá, P., Boydstun, 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

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

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

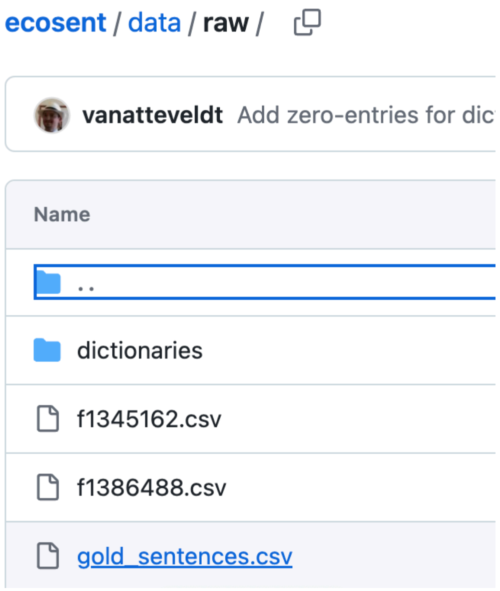
> van Atteveldt, W., van der Velden, M. A. C. G. & Boukes, M. (2021) The Validity of Sentiment Analysis: Comparing Manual Annotation, Crowd-Coding, Dictionary Approaches, and Machine Learning Algorithms. *Communication Methods and Measures*, 15(2), 121-140.

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

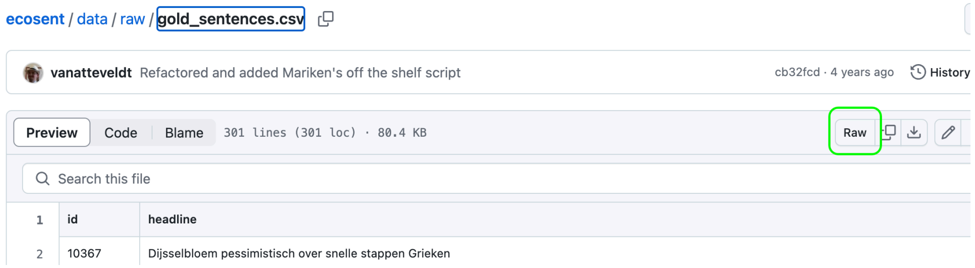
..

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

301 lines (301 loc) · 80.4 KB

id	headline
10367	Dijsselbloem pessimistisch over snelle stappen Grieken

```
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

```r
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

```r
sentences_df <- read_csv(gold_sentences_texts_url)
labels_df <- read_csv(gold_sentences_labels_url)

colnames(sentences_df)
```

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

```r
colnames(labels_df)
```

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

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

```r
library(dplyr)

# compute number of labels per headline
labels_df |>
  group_by(id) |>
  summarise(n_labels = n()) |>
  # count numbers of labels per headlines
  count(n_labels)
```

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

Note: each of 284 headlines has only one label

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

Download and extracting from a ZIP file

We can download ZIP archives and extract selected files in R

Example

We will use the replication data for the article

> Benoit, K., Conway, D., Lauderdale, B. E., Laver, M., & Mikhaylov, S. (2016). Crowd-sourced Text Analysis: Reproducible and Agile Production of Political Data. *American Political Science Review*, 110(2), 278–295.

The Github repository is here https://github.com/kbenoit/CSTA-APSR

```r
url <- "https://github.com/kbenoit/CSTA-APSR/raw/4bc6cbc48a4eeff557cbb03b8ede73b29e36aa00/Data%20-%20CF%20jobs/CFj

# download
temp <- tempfile()
download.file(url, temp, quiet = TRUE)

# list contents
conts <- unzip(temp, list = TRUE)$Name
head(conts, 4)
```

```
[1] "f240807.csv" "f246554.csv" "f246609.csv" "f263548.csv"
```

```r
# extract and read selected file
read_csv(unz(temp, "f240807.csv"))
```

```
# A tibble: 2,971 × 28
   `_unit_id` `_created_at`   `_golden` `_canary`  `_id` `_missed` `_started_at`
        <dbl> <chr>           <lgl>     <lgl>      <dbl> <lgl>     <chr>
```

```
 1 323988547 9/25/2013 01:2… FALSE     NA        1.06e9 NA        9/25/2013 01…
 2 323988547 9/25/2013 11:3… FALSE     NA        1.06e9 NA        9/25/2013 11…
 3 323988547 9/26/2013 04:4… FALSE     NA        1.06e9 NA        9/26/2013 04…
 4 323988547 9/26/2013 16:0… FALSE     NA        1.06e9 NA        9/26/2013 16…
 5 323988547 9/26/2013 16:2… FALSE     NA        1.06e9 NA        9/26/2013 16…
 6 323988548 9/25/2013 11:2… FALSE     NA        1.06e9 NA        9/25/2013 11…
 7 323988548 9/25/2013 12:0… FALSE     NA        1.06e9 NA        9/25/2013 12…
 8 323988548 9/26/2013 03:5… FALSE     NA        1.06e9 NA        9/26/2013 03…
 9 323988548 9/26/2013 16:1… FALSE     NA        1.06e9 NA        9/26/2013 16…
10 323988548 9/26/2013 19:0… FALSE     NA        1.06e9 NA        9/26/2013 19…
# ℹ 2,961 more rows
# ℹ 21 more variables: `_tainted` <lgl>, `_channel` <chr>, `_trust` <dbl>,
#   `_worker_id` <dbl>, `_country` <chr>, `_region` <chr>, `_city` <chr>,
#   `_ip` <chr>, econ_scale <dbl>, econ_scale2 <lgl>, policy_area <dbl>,
#   soc_scale <dbl>, soc_scale2 <lgl>, orig__golden <lgl>,
#   econ_scale_gold <lgl>, policy_area_gold <dbl>, post_sentence <chr>,
#   pre_sentence <chr>, sentence_text <chr>, sentenceid <dbl>, …
```

```r
# remove ZIP in temporary location
unlink(temp, recursive = TRUE)
```