



Workshop ‘Computational Text Analysis’

Session 1: Web-Scraping

Mirko Wegemann

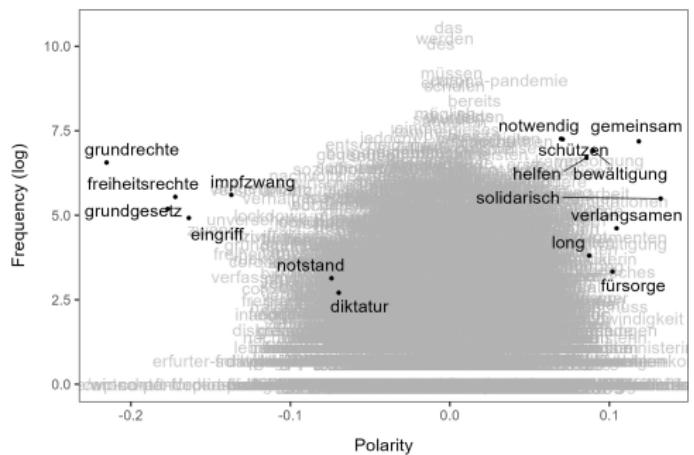
24 March 2025



About us II

Application of CTA in my own work

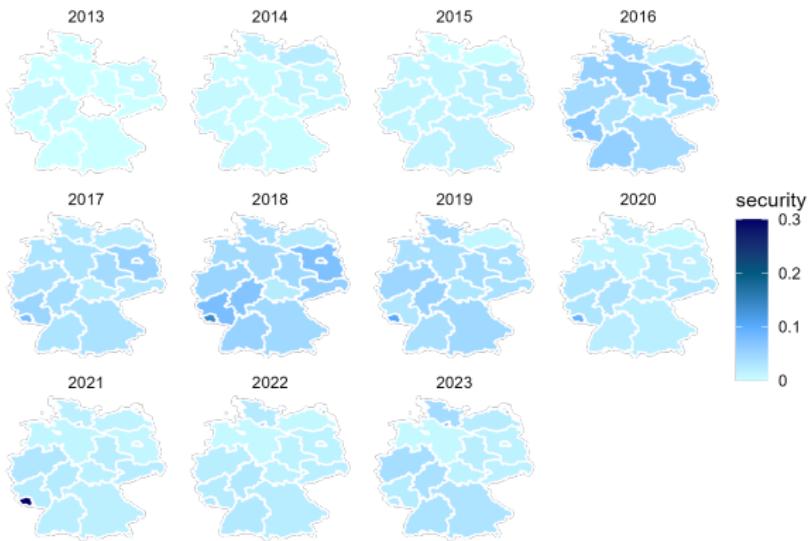
- scaling the Covid-19 discourse using Latent Semantic Scaling
(joint work with Rebecca Kittel)





About us III

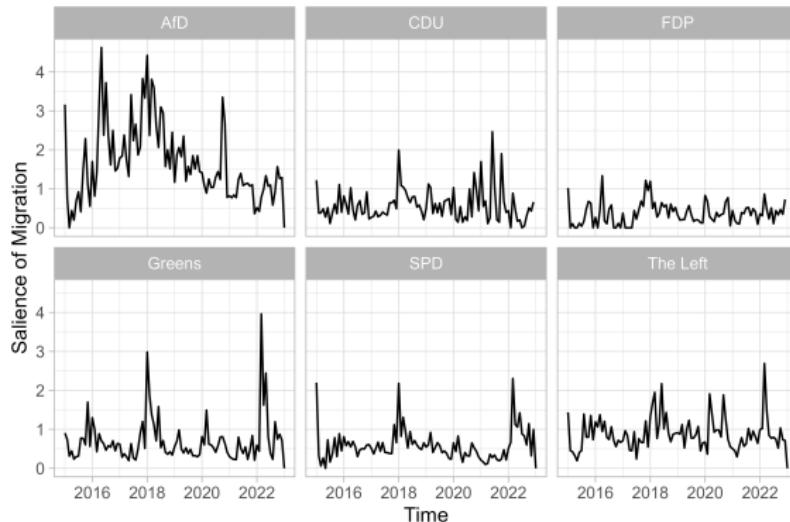
- classification of gender appeals by the radical right using a multilingual transformer model (joint work with Leon Küstermann)





About us IV

- tracing the migration discourse in Germany using large language models





About us V

Eva Krejcova (she/her)

- Junior Lecturer at the Swiss Graduate School of Public Administration (IDHEAP) at the University of Lausanne, Switzerland
- introduces you to images-as-data



About us VI

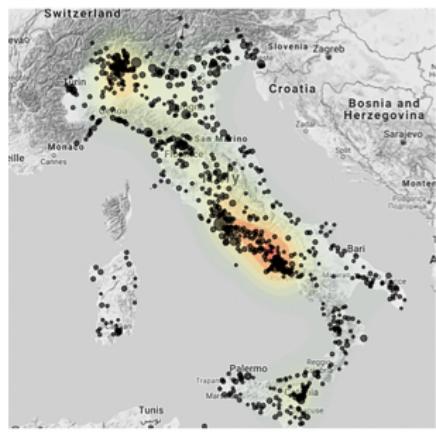
Sara Dybesland (she/her)

- 2nd-year SPS PhD Researcher
- regularly uses web-scraping and topic models in R
- analyses parliamentary speeches
- TA for the lab sessions



How can we utilize it in research? II

A



B



Bischof and Kurer (2023) show how local campaigning of M5S mobilizes voters



How can we utilize it in research? III

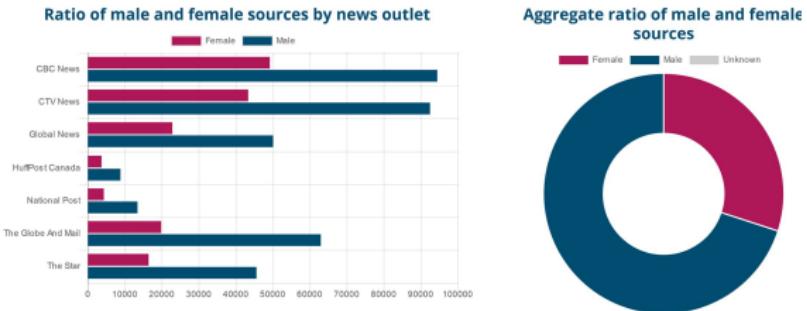


Fig 1. The Gender Gap Tracker online dashboard page. Reprinted from <https://gendergaptracker.informedopinions.org/> under a CC BY license, with permission from Informed Opinions, original copyright 2018.

Asr et al. (2021) on how Canadian newspaper quote men more often than women (**still online!**)



How can we utilize it in research? IV

Web scraping is a first step in our text-as-data query: we access the data. However, neither is scraping the only way to access data nor is text analysis the only type of analysis we can apply.



Types of web-scraping

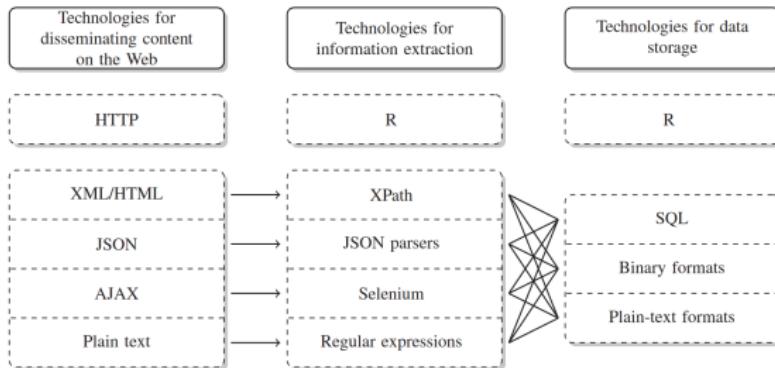


Figure 1.4 Technologies for disseminating, extracting, and storing web data

Munzert (2015, p. 10)



What we'll cover...

1. 'static' HTML structures (`rvest`)
 - all content is available by parsing the HTML
2. 'dynamic' webpages (`RSelenium`)
 - content is not visible in HTML source code but becomes available after interacting with the webpage



Before you scrape data

1. carefully consider whether this is the data you need for your research question
 - Do the data allow you to measure what you need to?
 - Do they comply with academic quality standards?
 - Is it really worth the effort?
2. check whether the data is directly available (e.g., someone else downloaded it, there is a direct download link, there is API access)
3. consider legal constraints



Disclaimer: Legal considerations I

Is web-scraping even allowed?

It depends...

- violation against terms of service?



Disclaimer: Legal considerations II

Without limitation, you shall not use (and you shall also not facilitate, authorise or permit the use of) the Guardian Site and/or any Guardian Content (including, any caption information, keywords or other associated metadata) for any other purpose without our prior written approval - this includes, without limitation, that you shall not use, copy, scrape, reproduce, alter, modify, collect, mine and/or extract the Guardian Content: (a) for any machine learning, machine learning language models and/or artificial intelligence-related purposes (including the training or development of such technologies); (b) for any text and data aggregation, analysis or mining purposes (including to generate any patterns, trends or correlations); or (c) with any machine learning and/or artificial intelligence technologies to generate any data or content or to synthesise or combine with any other data or content; or (d) for any commercial use.

Terms of Service of 'The Guardian'



Disclaimer: Legal considerations III

Is web-scraping even allowed?

It depends...

- violation against terms of service?
- gathering of personally-identifiable information may violate EU [General Data Protection Regulation \(GDPR\)](#)
- even if you do not collect personal data, you may infringe copyright regulations
- check [robots.txt](#)



Disclaimer: Legal considerations IV



- hiQ scraped data from LinkedIn public profiles
- LinkedIn tried to prevent that
- initial ruling: scraping allowed; later rulings: hiQ breached User Agreement → settlement

In the end, “[l]egalities depend a lot on where you live. However, as a general principle, if the data is public, non-personal, and factual, you’re likely to be ok” (Wickham et al. 2023)



HTML Basics

Websites are based on **HyperText Markup Language**

- HTML contains information about the structure of a web-page
- it's responsible for how content is graphically displayed

An example



HTML Elements and Attributes

- HTML consists of elements, tags and attributes
 - elements are the different components of a webpage (e.g. headlines, text, images)
 - elements are mostly embedded into tags (`<element>content</element>`) but some come without start and end tags
 - attributes are additional information of an element (e.g., the size of an image, the font type etc.)

We will only cover HTML in a superficial way, but try this [tutorial](#)



HTML head vs body

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>"EUI Website"</title>
5   </head>
6   <body>
7     <h1>"New Academic Year"</h1>
8   </body>
9 </html>
```

HTML documents consist of a head (with meta information on the webpage) and a body (with content) → mainly interested in the body!



Important elements

- *h1, h2, h3*, etc.: headline elements
- *p*: paragraph elements
- *a*: hyperlink elements
- *img*: image elements



Important attributes

- *href*: weblink, always comes with a element
- *src*: source of an image



How to use HTML for scraping?

We need to identify the CSS selector of an element of interest.
There are **two** options:

- manual approach: cursor on element of interest > right-click > Inspect
- semi-automatic approach: download **SelectorGadget** or save as bookmark



Basic pipeline

Setup

- install SelectorGadget
- R library: rvest



Step 1

Download HTML source with `rvest`

```
1 > library(rvest)
2 > url <- "https://press.un.org/en/content/press-release"
3 > html <- read_html(url)
4 > html
5 {html_document}
6 <html lang="en" dir="ltr">
7 [1] <head>\n<meta http-equiv="Content-Type"
     content="text/html; charset=UTF-8">\n<meta
     charset="utf-8">\n<link rel="canonical" href="https:
     ...
8 [2] <body class="layout-one-sidebar
      layout-sidebar-first page-view-home-press
      path-content">\n      <div
      class="visually-hidden-focusable bg- ..
```



Step 2

Example: **UN press releases**

The screenshot shows the homepage of the United Nations Press Release website at press.un.org/en/content/press-release. The page title is "Meetings Coverage and Press Releases". The navigation menu includes "Home", "Secretary-General", "General Assembly", "Security Council", "Economic and Social Council", and "International Court of Justice". Below the menu, a "Press Release" section displays a recent release from the Security Council dated 22 May 2024. To the right, there are "Highlights" sections for "Highlights of the Security Council Activities" and "Highlights of General Assembly 78th Session".



Step 3

United Nations | Meetings Coverage and Press Releases | SEARCH

Home Secretary-General General Assembly Security Council Economic and Social Council International Court of Justice

Press Release

22 May 2024

Security Council 2140 Committee Considers Panel of Experts' Midterm Update

On 3 May 2024, the Security Council Committee established pursuant to resolution 2140 (2014) held informal update on the midterm update of the Panel of Experts, submitted in accordance with paragraph 3 of resolution 2707 (2023).

copy CSS selector

click on element of interest

Activities of the Secretary-General in Bahrain, 15-17 May

United Nations Secretary-General António Guterres flew from Muscat, Oman, to Manama, Bahrain, 15 May 2024.

Highlights

BC/15705 Highlights of the Security Council Activities

ights of General Assembly 78th Session New York, 5 Sep. - 22 Dec. 2023

GW Session of the Commission on the Status of Women

Clear (1) Toggle Position XPath ? X

Try it out yourself!



Headline

Here, we retrieve every level-1 headline of the webpage.

```
1 > library(rvest)
2 > (top_level_headline <- read_html(url)
3 +     %>% html_elements("h1")
4 +     %>% html_text())
5 [1] "Press Release"
```



Text

Here, we retrieve every paragraph on the webpage.

```
1 > library(rvest)
2 > (paragraphs <- read_html(url) %>%
3 +     html_elements("p") %>%
4 +     html_text())
5 [1] "On 3 May 2024, the Security Council Committee
     established pursuant to resolution 2140 (2014) held
     informal consultations to consider the midterm
     update of the Panel of Experts, submitted in
     accordance with paragraph 3 of resolution 2707
     (2023)."
6 [2] "United Nations Secretary-General Antonio Guterres
     flew from Muscat, Oman, to Manama, Bahrain, in the
     early evening of Wednesday, 15 May."
```



Links

If we want to access links, we need to retrieve the `a` element first, and then call its attribute `href`

```
1 > (pr_urls <- read_html(url) %>%
2 +     html_elements(".field_item a") %>%
3 +     html_attr("href"))
4 [1] "/en/2024/sc15705.doc.htm"
      "/en/2024/sgt3388.doc.htm"
      "/en/2024/sgt3387.doc.htm"    "/en/2024/3386.doc.htm"
```



Tables

rvest has a pre-defined function `read_table()` to extract information from HTML tables

```
1 > html <- read_html(url2)
2 > table <- html %>%
3 +   html_element(".wikitable:nth-child(4)") %>%
4 +   html_table()
```



Images

For images, it is a bit more complicated.

1. open a session
2. retrieve image source link
3. download the image to your directory



Images II

```
1 > session <- session(url)
2 >
3 > # access links for image sources
4 > imgsrc <- session %>%
5 +     read_html() %>%
6 +     html_nodes("img") %>%
7 +     html_attr("src")
8 >
9 > # access page of source image (only retrieve the
   first image here)
10 > img <- session_jump_to(session, paste0(root_url,
    imgsrc[[1]]))
11 >
12 > # write to our project's directory
13 > writeBin(img$response$content, basename(imgsrc[1]))
```

Let's do it in R



Loops I

Often, we want to automatize these steps for multiple pages.

Two options:

1. creating empty objects and filling them in a `for-loop`
2. defining a function, apply it and retrieve objects of interest from list

→ usually, functions are more versatile and can be easier run in parallel



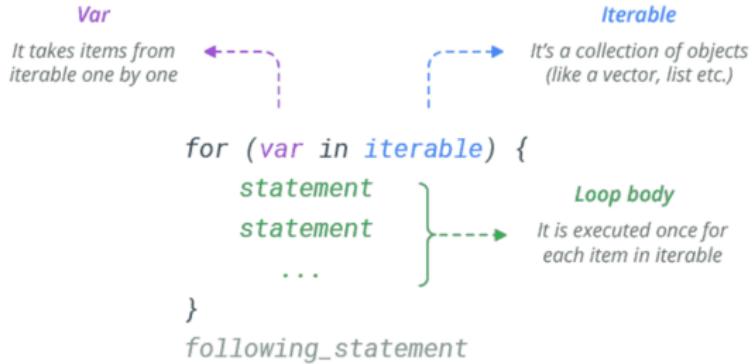
Loops II

Before creating a loop/function

1. check the pagination structure of a website (e.g. [United Nations](#) use '?page=#' to list results)
2. make sure which elements you need to retrieve (often, you just want links but some information, such as date, may not be available on sub-pages, so you want to gather these as well)
3. test the pipeline on a single element before using it in a loop



For-loops



Tutorial on for-loops
Graph and guide to for-loops



For-loops for automation

```
1  
2 > urls <- c("https://www.eui.eu/en/home",  
3   "https://www.ceu.edu/")  
4 > links <- c()  
5 > for(url in 1:length(urls)){  
6   +   html <- read_html(urls[[url]])  
7   +   links[url] <- html %>%  
8   +     html_node("h1") %>%  
9   +     html_text()  
9 >  
10 > links  
11 [1] "\nOpening doors to the future\n" "Apply Now for  
2024-2025!"
```

Let's do it in R



Functions in R

```
Function name  
An identifier by which the function is called
```

```
name <- function(args){  
  statement  
  statement  
  ...  
  return(value)  
}
```

```
Arguments  
Contains a list of values passed to the function
```

```
Function body  
This is executed each time the function is called
```

```
Return value  
Ends function call & sends data back to the program
```

Tutorial on Functions Graph and guide to functions



Functions for automation

```
1 > h1_scrape <- function(url) {  
2 +     html <- read_html(urls[[url]])  
3 +     links[url] <- html %>%  
4 +         html_node("h1") %>%  
5 +             html_text()  
6 + }  
7 >  
8 > (links <- sapply(1:length(urls), h1_scrape))  
9 [1] "\nOpening doors to the future\n" "Apply Now for  
    2024-2025!"
```

Let's do it in R



Post processing

Regular expressions

Often, the data you retrieve are not in the desired format.

Regular expressions are handy for data processing, they can match/extract/remove a pattern in text

```
1 > library(stringr)
2 > date_time <- "2009/03/12 20:12:31"
3 > str_extract(date_time, "\\\d{4}\\\\d{2}\\\\d{2}")
4 [1] "2009/03/12"
```



Regular expressions in R

- base R includes regular expressions in functions like `gsub` (replacement), `regmatches` (extraction) or `regexpr` (detection)
- `stringr` is the tidyverse approach to regex and provides equivalent functions such as `str_replace`, `str_extract` or `str_detect`



More information on regular expressions

- You can try out regular expressions in online tools like [these](#)
- [StringR cheat sheet](#)
- [A guide to regular expressions](#)



Dynamic Webpages

Sometimes, webpages change only when we interact with them in a browser session (e.g. by clicking on specific objects). For these **webpages**, rvest is not applicable.

Usually, you can identify these pages by their usage of active classes

```
<html lang="en"> event scroll
  > <head> ... </head>
  ▼ <body lang="en" data-jsactive="true" data-n
    <input id="dev_product_uri" type="hidden"
    ▼ <div id="website"> overflow
      > <div id="website-header" class="ep-header">
        <!--END OF REFACTORING-->
```



Functionality of RSelenium

Solution?



RSelenium

RSelenium was initially built to test webpages, we use it to send commands to a virtual browser window (e.g. clicking a button)



Before using RSelenium

Whenever possible, try to use the static approach, it's easier and less prone to error

- check other webpages which may contain the same information
- look for other sub-directories that may store the same content
- check the [Wayback Machine](#)
- use the 'Search' bar of a webpage to list results



Pipeline of RSelenium and wdman

A typical pipeline looks as follows:

- configuration of a session (now more complex, we need wdman for this)
- opening browser
- navigate to URL
- accept/decline cookies
- identify css selector for button, click on it (repeat if necessary)
- retrieve object of interest (usually links)



Setup

- configure driver
- extract client
- navigate to webpage

```
1 > url <- "https://www.europarl.europa.eu/news/en"  
2 > rd <- rsDriver(browser = "firefox",  
3 +           chromever = NULL,  
4 +           port = sample(1:65535, 1),  
5 +           verbose = FALSE)  
6 > browser <- rd[["client"]]  
7 > browser$navigate(url)
```



Identifying and clicking on a button

1. Find an element by its css selector
2. store it as an object
3. use the clickElement() function

```
1 > cookies <- browser$findElement(using = 'css  
    selector', value='.epjs_agree:nth-child(1) span')  
2 > cookies$clickElement()
```



How to extract the text of a dynamic webpage

- identify elements by css selector
- elements contain lots of other information
- to extract plain text, loop through the elements and extract only the text

```
1 > paragraphs <- browser$findElements(using = 'css
   selector', value='p')
2 > text <- c()
3 > for(i in 1:length(paragraphs)) {
4 +     text[i] <- paragraphs[[i]]$getElementText() [[1]]
5 + }
6 > text[1]
7 [1] "On Thursday afternoon, the Eurovision debate
      between the lead candidates for the presidency of
      the European Commission took place in the European
      Parliament."
```



How to extract the link of a dynamic webpage

```
1 > urls_euparl <- browser$findElements(using = 'css
  selector', value='.ep_title > a')
2 > urls_euparl2 <- c()
3 > for(i in 1:length(urls_euparl)){
4   +   urls_euparl2[i] <-
  urls_euparl[[i]]$getElementAttribute('href')[[1]]
5 }
```

Let's do it in R



Processing data

Once we have downloaded individual links to pages, we can often proceed in `rvest` (which is easier to handle).



Concluding remarks

That's it for today. We covered the first aspect of text analysis on how to retrieve data.

- in the lab session: practice scraping on your own projects
- tomorrow: bags-of-words approaches
 - topic modelling
 - prediction tasks

Thank you for your attention!

References I

- Asr, F. T., Mazraeh, M., Lopes, A., Gautam, V., Gonzales, J., Rao, P., & Taboada, M. (2021). The Gender Gap Tracker: Using Natural Language Processing to measure gender bias in media. *PLOS ONE*, 16(1), e0245533.
<https://doi.org/10.1371/journal.pone.0245533>
- Bischof, D., & Kurer, T. (2023). Place-Based Campaigning: The Political Impact of Real Grassroots Mobilization. *The Journal of Politics*, 85(3), 984–1002.
<https://doi.org/10.1086/723985>
- Munzert, S. (2015). *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining* (1st ed.). Wiley.
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science*. O'Reilly Media, Inc.
<https://r4ds.hadley.nz/>