Text as Data

Session 5: Data Collection – Scraping I

Mirko Wegemann

Universität Münster
Institut für Politikwissenschaft

20 May 2026

What is web scraping?

Web scraping describes the systematic collection of (often unstructured) data from the internet to store it in a structured format.

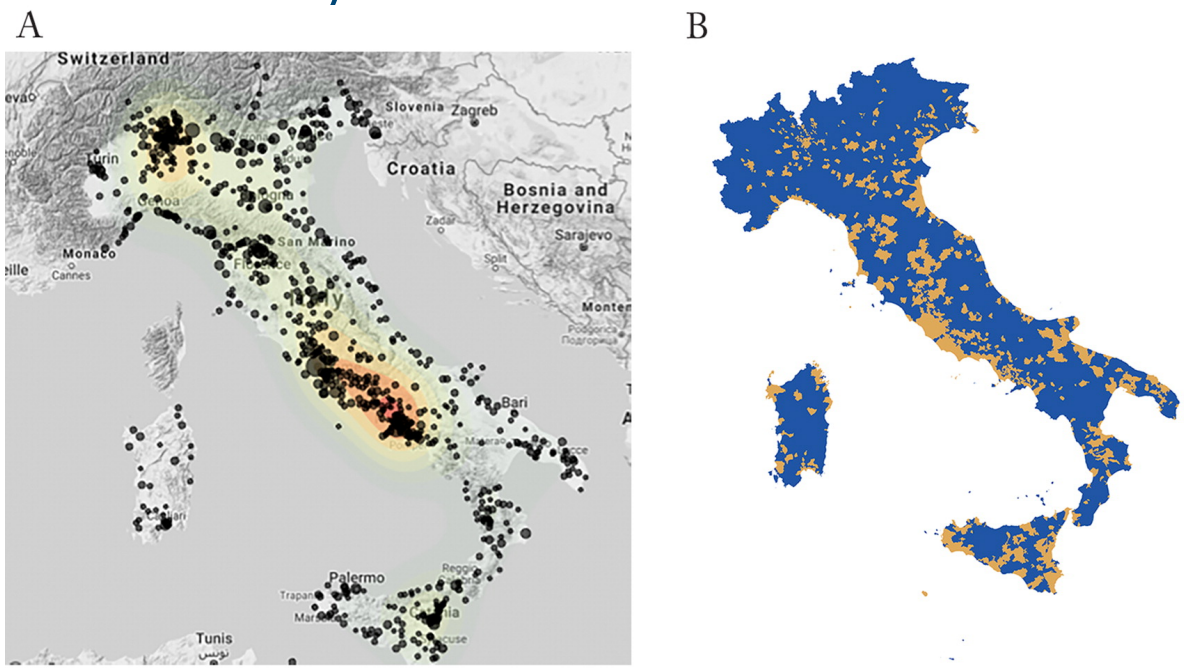
Why should we do it? I

Figure 1.1 Location of UNESCO World Heritage Sites in danger (as of March 2014). Cultural sites are marked with triangles, natural sites with dots

Figure: Overview of the World Heritage in Danger

From a Wikipedia table, we can create a visual representation.

Why should we do it? II



Bischof and Kurer (2023) provide data on Italian Five Star Movements campaign events to measure the impact of mobilization.

Why should we do it? III

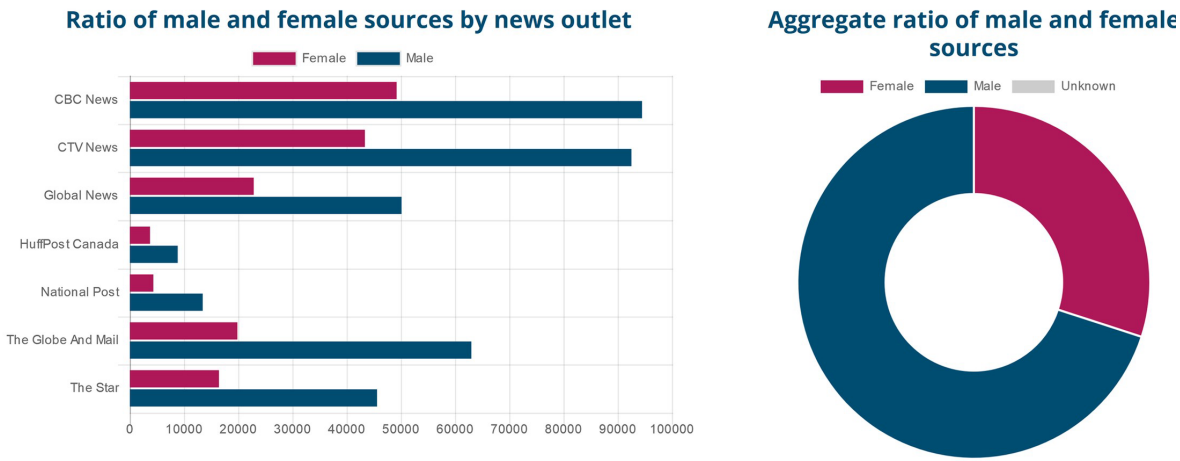


Fig 1. The Gender Gap Tracker online dashboard page. Reprinted from https://gendergaptracker.informedopinions.org/ under a CC BY license, with permission from Informed Opinions, original copyright 2018.

Asr et al. (2021) collect data from Canadian newspapers to assess how often it relates to gender-related topics.

→ Web scraping is often the first step for a subsequent analysis, not necessarily a text analysis.

Different types of web scraping

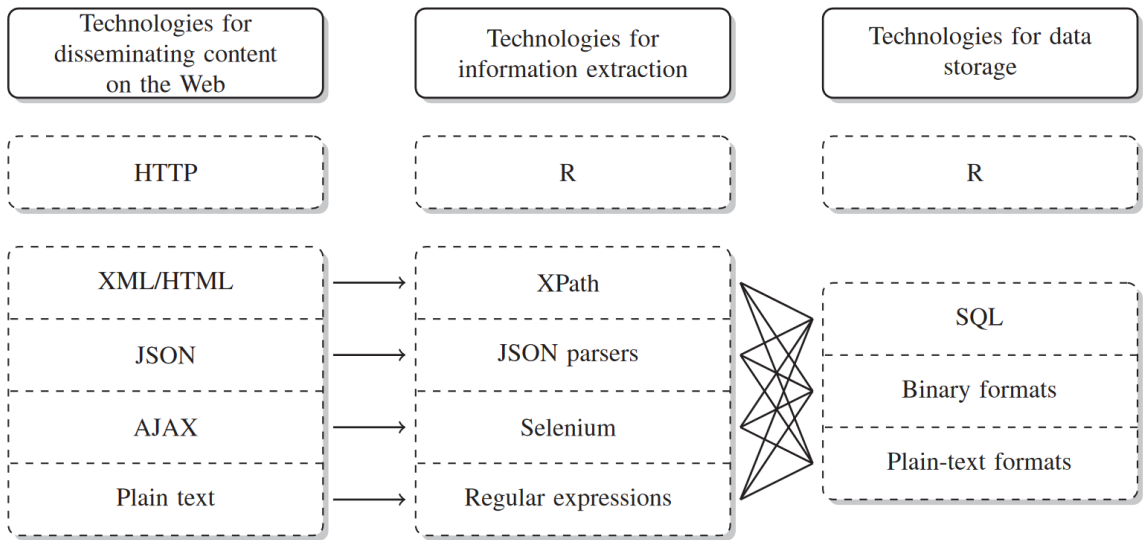


Figure 1.4 Technologies for disseminating, extracting, and storing web data

Munzert (2015, p. 10)

Our goals

1. **Static HTML structures** can be accessed through (`rvest`)
   - The content that interests us is visible in the HTML source code.
2. **Dynamic websites** can be accessed via (`RSelenium`)
   - The content is not immediately visible in the HTML source but requires interaction (e.g., through mouse clicks).

Before data scraping

1. Do you need to use the data at all?
   - Can you use it as intended for your research goal?
   - Do the data meet relevant qualitative standards?
   - Are the data already available in other locations?
     - Through direct data downloads or APIs?
   - Are there any legal concerns?

Legal considerations I

Is web scraping permitted? **It depends...**

- Does it violate the terms of use of the website provider?

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.

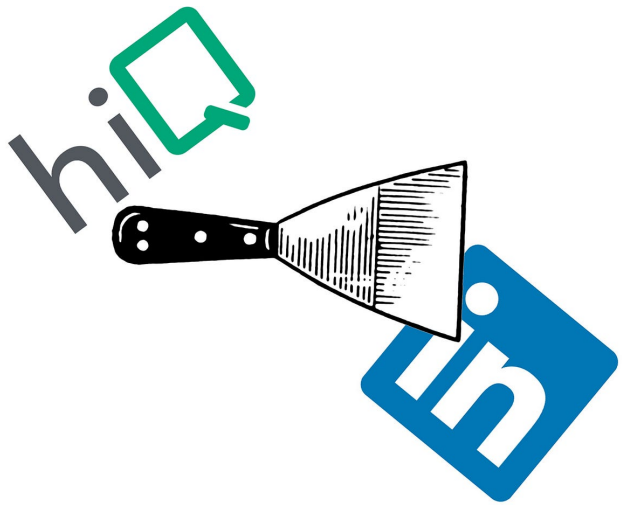
Figure: Terms of use for The Guardian

Legal considerations III

It depends...

- Does collecting personal data violate the European Data Protection Directive (GDPR)?
- Even if no personal data is collected, copyright laws may be violated.

Legal considerations IV



- hiQ collected data from public LinkedIn profiles.
- LinkedIn tried to prevent this.
- First instance: Scraping is allowed; second instance: hiQ violated terms

HTML basics

Webpages are built on the **HTML** markup language.

- HTML contains information about the structure of a webpage.
- HTML ensures that content is visually presented.

An example

HTML Elements and Attributes

- HTML consists of elements, tags, and attributes
  - Elements are the various components of a web page (e.g., headings, text, images)
  - Elements are usually embedded in tags (<*element*>content</*element*>), though some do not have start and end tags
  - Attributes provide additional information about an element (e.g., image size, font, etc.)

We are only covering HTML very superficially; if you want to delve deeper, try this **tutorial**.

HTML: head vs. body

```
1  <!DOCTYPE html>
2  <html lang="de" class="no-js">
3  <head>
4  <title>Universitaet Muenster</title>
5  </head>
6  <body>
7  ...
8  </body>
9  </html>
```

HTML documents consist of a **header** (the *header*, containing meta information about the webpage) and a **body** (containing content) → we are mainly interested in the *body*!

Recurring Elements

- *h1*, *h2*, *h3*, etc.: Headings
- *p*: Paragraphs
- *a*: Hyperlinks
- *img*: Images

Recurring Attributes

- *href*: Web link, always appears with the *a* element
- *src*: Source of an image

HTML and Scraping

We need to identify the CSS selector of the desired element. There are **two** options for this:

- Manual method: Hover over the desired element > Right-click > Inspect
- (Semi-)automatic method: Download **SelectorGadget** or save it as a bookmark

Basic pipeline

Setup

- Install SelectorGadget
- R library: `rvest`

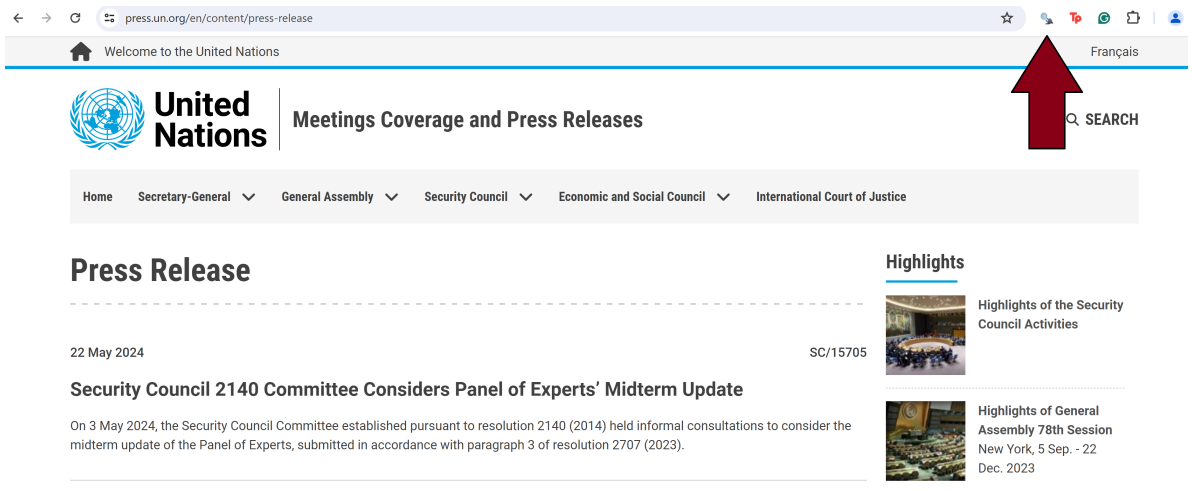
Step 1

Download the HTML source using `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**



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
SC/15705
Security Council 2140
mittee Considers Panel of Experts' Midterm Update
On 3 May 2024, the Security Council Committee established pursuant to resolution 2140 (2014) held informal c
midterm update of the Panel of Experts, submitted in accordance with paragraph 3 of resolution 2707 (2023).
21 May
Activities of the Secretary-General in Bahrain, 15-17 May
United Nations Secretary-General António Guterres flew from Muscat, Oman, to Manama, Bah
Highlights
Highlights of the Security Council Activities
lights of General
embly 78th Session
New York, 5 Sep. - 22 Dec. 2023
68th Session of the Commission on the Status of Women
click on element of interest
copy CSS selector
h1
Clear (1)
Toggle Position
XPath
?
X

Try it yourself!

And now in R

Heading

Here we retrieve every Level 1 heading from the website.

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

Text

Here we retrieve every paragraph on the website.

```
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 must first retrieve the a element and then access its `href` attribute.

```
(pr_urls <- read_html(url) %>%
+     html_elements(".field__item a") %>%
+     html_attr("href"))
[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 predefined function `html_table()` to extract information from HTML tables.

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

Images

With images, it's a bit more complicated.

1. Open a session
2. Retrieve the image source
3. Download the image to your directory

Bilder II

```
> session <- session(url)
>
> # save links to image source
> imgsrc <- session %>%
+     read_html() %>%
+     html_nodes("img") %>%
+     html_attr("src")
>
> # open image source
> img <- session_jump_to(session, paste0(root_url,
    imgsrc[[1]]))
>
> # write into project directory
> writeBin(img$response$content, basename(imgsrc[1]))
```

Loops I

We often want to automate these steps across multiple pages.
Two options:

1. Create empty objects and fill them in a `for-loop`
2. Define a function, apply it, and retrieve the desired objects from a list

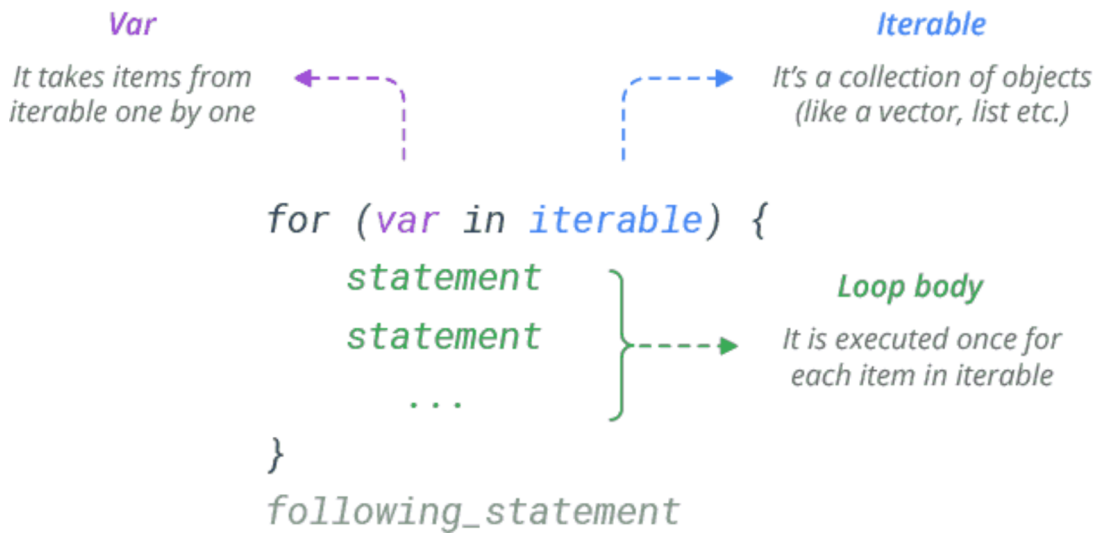
→ Functions are usually more versatile and can be executed in parallel more easily.

Loops II

Before creating loops/functions

1. Check the structure of the page pagination (e.g., the UN uses "?page=#"" to display results)
2. Check which elements need to be retrieved (often only links are needed, but some information, such as the date, may not be available on subpages and should therefore also be collected)
3. Test the pipeline on a single element before incorporating it into the loop

For Loops



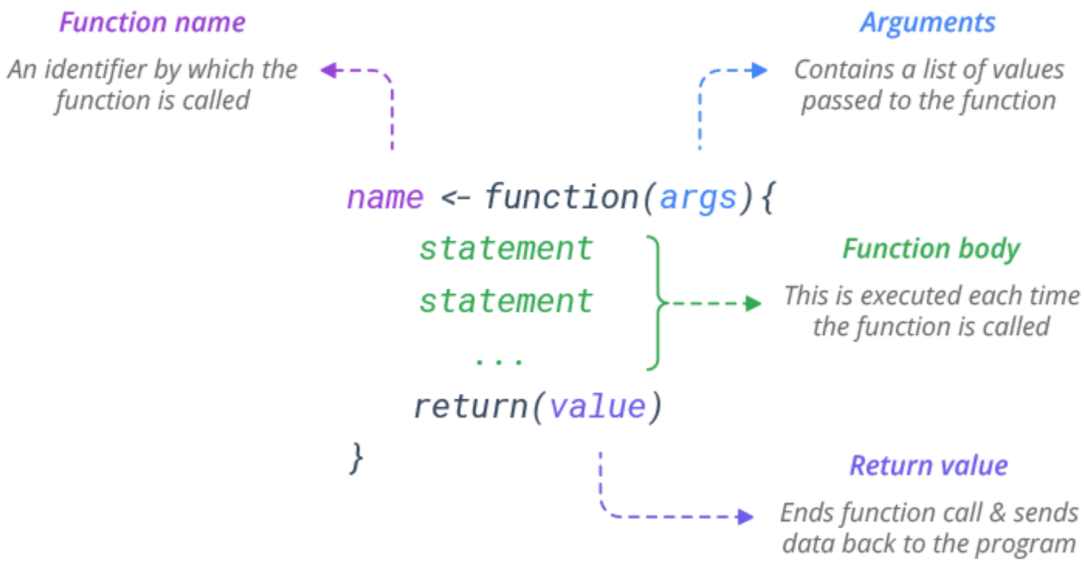
Tutorial on for-loops
Graphic and instructions for for-loops

for-Loops for Automation

```
> urls <- c("https://www.uni-muenster.de/de/",
    "https://www.uni-osnabrueck.de/startseite/")
> links <- c()
> for(i in 1:length(urls)){
+     html <- read_html(urls[[i]])
+     links[i] <- html %>%
+         html_node("h1") %>%
+         html_text()
+ }
> links
[1] "Universitaet Muenster" "Main Content"
```

And now in R

Functions in R



Tutorial on functions
Graphics and instructions for functions

Functions for automation

```
> h1_scrape <- function(url){
+     html <- read_html(urls[[url]])
+     links[url] <- html %>%
+         html_node("h1") %>%
+         html_text()
+ }
>
> (links <- sapply(1:length(urls), h1_scrape))
[1] "Universitaet Muenster" "Hauptinhalt"
```

Practice file in R (scraping_exercises_empty.Rmd)

Outlook

- Today we downloaded some simple websites
- Next week we'll look at websites like the one here
- Also: An introduction to data collection via APIs

For the session in two weeks

- Next week, there's Whitsun Holiday: no session!
- If applicable, upload your own web scraping application to the submission tool; prepare for the session in two weeks so you can present it to the group
- Download Java and save the installation as an environment variable
- Download RTools
- Sign up for the NYT API and the Manifesto Project API

Literatur 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.

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, June). *R for Data Science*. O'Reilly Media, Inc. https://r4ds.hadley.nz/