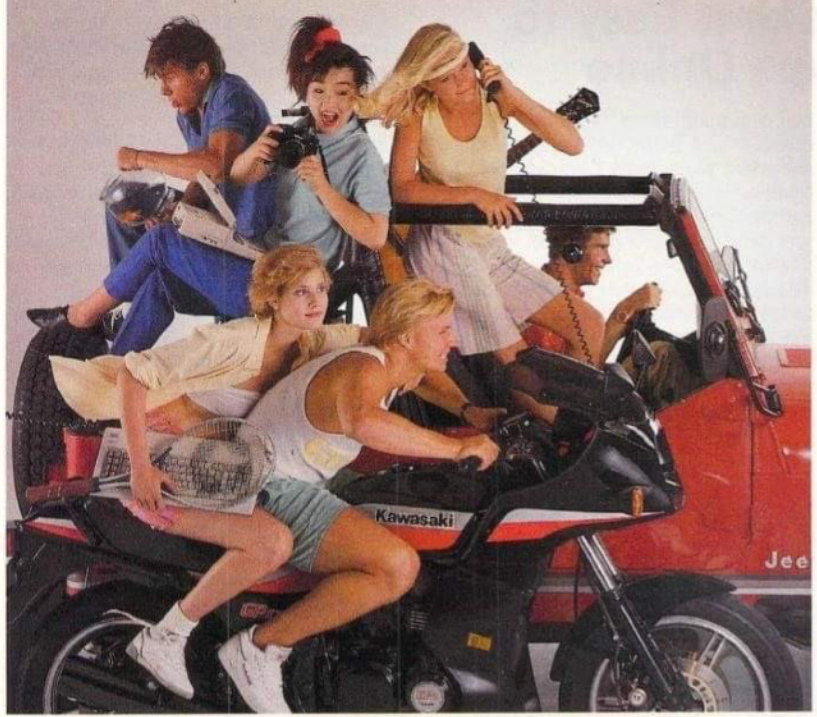
UNIVERSITÄT LEIPZIG

Forschungsseminar CSS – Scraping the web

GWZ H2 1.15, 28.10.2025

Felix Lennert, M.Sc.



TO BE YOUNG AND ONLINE

~~Youth~~ Explore Databases, Find Friends ~~in Forums~~

OUTLINE

- What does this even mean? How are we doing this?
- Intro to HTML
- Scraping – what do we have to bear in mind?
  - Is this legal?
  - How do we protect people's privacy?

(Today's slides are inspired by Étienne Ollion's slides for SICSS-Paris 2023, find all the materials here)

SCRAPING

- Scraping is describing the process of acquiring data from the world wide web
- Multiple ways of doing this exist – in descending order from most to least favorable/ convenient:
  - Data dumps – companies give out significant chunks of their data (e.g., Pushshift for Reddit data)
  - APIs (Application Programing Interfaces) – companies provide you a *structured* way of getting their data (e.g., Spotify, New York Times, etc.)
  - Screen-scraping
    - We write a program to grab raw content
    - Sometimes: we write a program to simulate a browser *and then* grab raw content
- Overview of packages

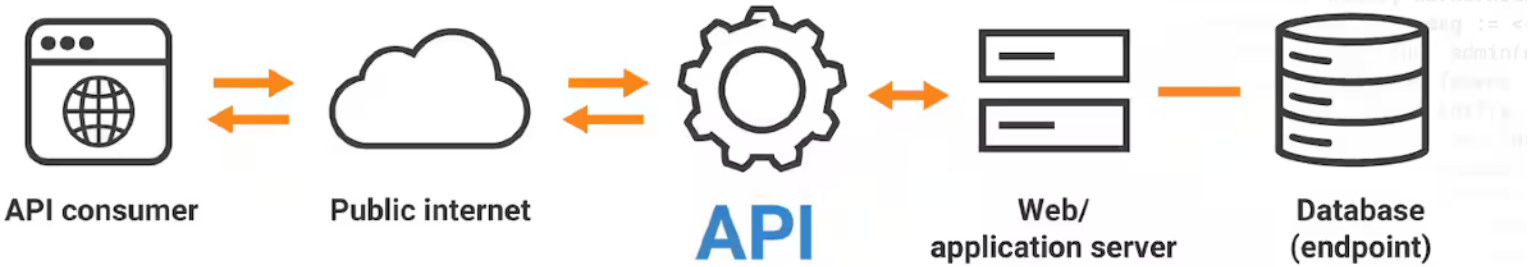
DATA DUMPS

- Data dumps: companies give out significant chunks of their data (e.g., Pushshift for Reddit data)
  - Always easy to acquire
  - Not always easy to process (e.g., SQL)
  - Clear legal implications

- Examples:
  - Reddit (Pushshift)
  - Project Gutenberg (R package: *gutenbergr*) – books

APIS

- APIs (Application Programing Interfaces)



APIS

- APIs (Application Programing Interfaces)
  - Companies provide you a *structured* way of getting their data (e.g., Spotify, New York Times, etc.)
  - Not always easy to acquire (depending on documentation)
    ⟹ Packages/wrappers might exist! – overview
  - Post-processing usually straight-forward; clear legal implications
- Example request:

https://api.nytimes.com / svc/search/v2/articlesearch.json ? q=Trump&end_date=20161110&api-key=[yourkey]

base url | endpoint | headers/parameters

DATA DUMPS AND APIS

- Rather straightforward
- Data comes in pre-structured format

⟹ Usually in either XML or JSON format

- Data extraction performed using *xml2* (for XML documents) or *jsonlite* (JSON)
  - Tutorial for xml2: working with XML documents is covered in *rvest*
  - Tutorial for jsonlite: *jsonlite* makes it easy to transform json files to tibbles/lists – then you can use R/tidyverse functions to manipulate

SCREEN SCRAPING

- Packages: *rvest* (static pages), *selenium* (simulates a browser)
- Write a program to grab web content
  - Not always easy and convenient
  - Post-processing sometimes unclear
  - Mostly legal (check [url]/*robots.txt*)

```
# TYPO3-Pfade ausschließen
Disallow: /typo3_src/
Disallow: /typo3/
Disallow: /t3lib/
Disallow: /typo3conf/
Disallow: /search/
Disallow: /en/search/
```

```
# Sitemap
Sitemap: https://www.uni-leipzig.de/sitemap.xml

###Grunsätzliche Erlaubnis für alle Verzeichnisse/Ordner/Pfade
# beinhaltet auch Seitenstruktur im Frontend
User-agent: *
Allow: /
```

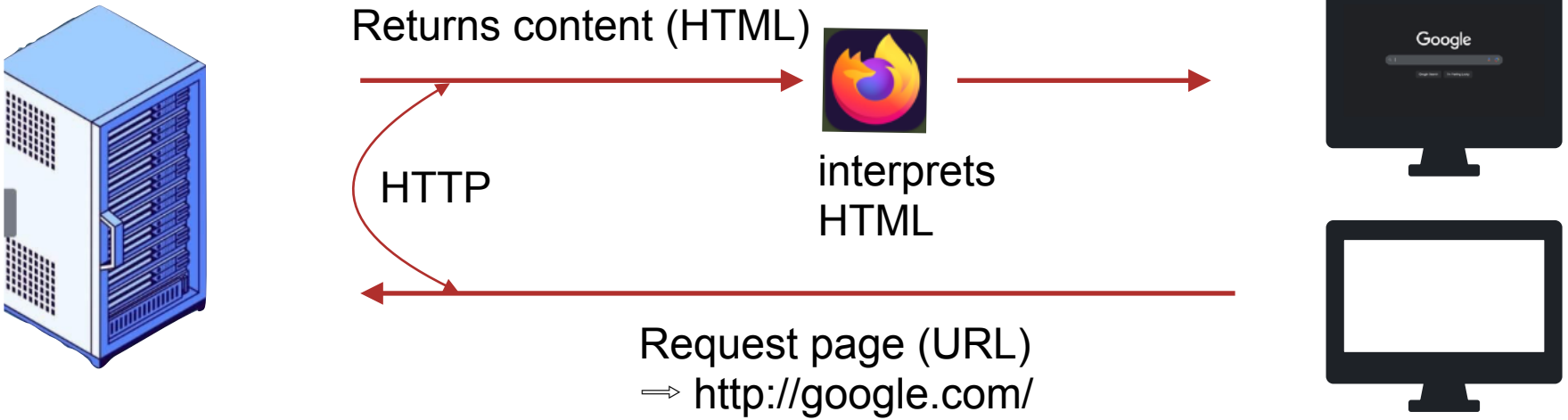
```
Disallow:
```

```
Disallow: /
```

SCREEN SCRAPING

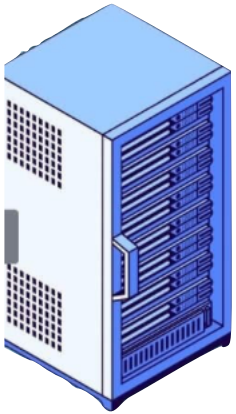
- Some basic terminology
  - HTML – *hypertext markup language*
    ⟹ Language that websites are written in
  - HTTP – *hypertext transfer protocol*
    ⟹ The protocol browsers use to communicate
  - URL – *uniform resource locator*
    ⟹ Path where web content is stored at
  - Browser – application that interprets and displays websites

NORMAL BROWSING



SCRAPING

extraction and post-processing



Returns content (HTML); *xml2* transforms it to XML

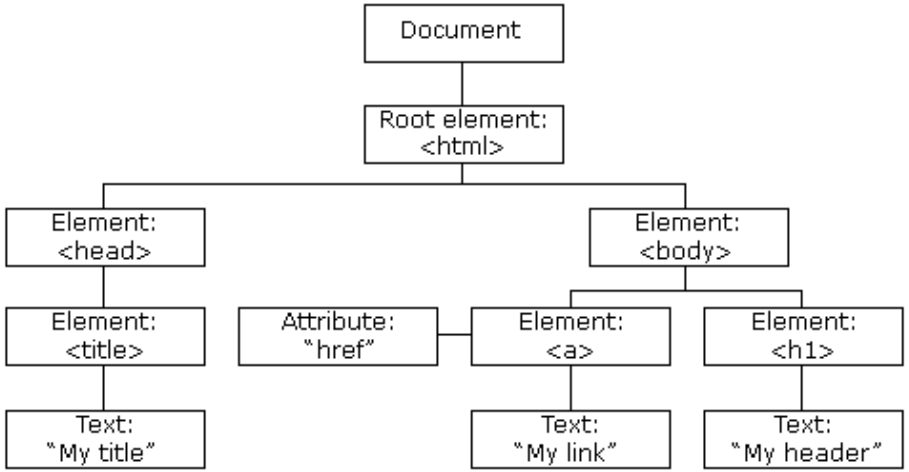
HTTP (powered by *httr* in R)

Request page (URL)
xml2::read_html("http://google.com/")

HTML BASICS

- We never want the *entire content* of a page
- We need to extract relevant stuff
- One solution: REGEX – **tedious**
- But: we can choose parts of a page harnessing how the web is written

⟹ HTML tags



[PLACEHOLDER VIDEO DEMONSTRATION INSPECT ELEMENT]

HTML BASICS

```
<html>
 <head>
  <title>Page title</title>
 </head>
 <body>
  <h1 id="first">A heading</h1>
  <p class="paragraph">Some text &amp;amp; <b>some bold text.</b></p>
  <a> Some more <i> italicized text which is not in a paragraph. </i> </a>
  <a class="paragraph">even more text &amp;amp; <i>some italicized text.</i></p>
  <a id="link" href="www.nyt.com"> The New York Times </a>
 </body>
```

HTML BASICS

```
<html>
 <head>
  <title>Page title</title>
 </head>
 <body>
  <h1 id="first">A heading</h1>
  <p class="paragraph">Some text &amp;amp; <b>some bold text.</b></p>
  <a> Some more <i> italicized text which is not in a paragraph. </i> </a>
  <a class="paragraph">even more text &amp;amp; <i>some italicized text.</i></p>
  <a id="link" href="www.nyt.com"> The New York Times </a>
 </body>
```

A heading

Some text & **some bold text.**

Some more *italicized text which is not in a paragraph*. even more text & *some italicized text*.

The New York Times

This looks shit. Why? ⟹ No CSS (cascading style sheet)

CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
h1           {color: blue;}
p            {color: red;}
a.paragraph {color: pink;}
#link        {
              color: black;
              font-family: courier;
              }
</style>
</head>
<body>
<body>
  <h1 id="first">A heading</h1>
  <p class="paragraph">Some text &amp;amp; <b>some bold text.</b></p>
  <a> Some more <i> italicized text which is not in a paragraph. </i> </a>
  <a class="paragraph">even more text &amp;amp; <i>some italicized text.</i></p>
  <a id="link" href="www.nyt.com"> The New York Times </a>
</body>
</html>
```

CSS

A heading

Some text & **some bold text.**

Some more *italicized text which is not in a paragraph*. even more text & *some italicized text*.

The New York Times

CSS

- Responsible for the “styling” of the page
- Useful for us, since we can use these tags to grab content
  - Intro to the basics of CSS selectors in R script
- Do we need to go inside the document all the time? – HELL NO
- Selectorgadget to the rescue

[PLACEHOLDER VIDEO DEMONSTRATION SELECTORGADGET]

ENOUGH FOR STATIC SITES

- Caveat: not all of these sites are as “basic” static pages
- Challenges you might encounter:
  - Dynamic content (Javascript)
  - Authentification
  - Captchas (making sure there is a human in front of the screen)
- All these things might be above *rvest*’s capabilities
- Solution: we control a browser via code – *selenium*

[PLACEHOLDER VIDEO DEMONSTRATION SELENIUM]

NEXT TIME

- Ethics
- How we do this in R/Python

UNIVERSITÄT LEIPZIG

MERCI!

Felix Lennert
Institut für Soziologie

felix.lennert@uni-leipzig.de
www.uni-leipzig.de