Web Scraping using R

Jakob Tures

2025-03-06

Introduction

This is an introduction to web scraping using R, designed to accompany a seminar held at the University of Potsdam in winter 2023/24.

What is web scraping?

Web scraping encompasses methods of data collection from online sources. This introduction will focus on scraping websites, but other forms on online data, e.g. databases or files, are scrapable as well.

Instead of copying and pasting content from a browser window, in web scraping the process is automated through a programming language as R or Python. This is less error prone than manual collection and enables us to collect large amounts of data efficiently or regularly repeat the scraping process to update the data without investing much time. In web scraping your time is invested in planning the scraping process, programming your software and analysing the retrieved data – aka the fun stuff – rather than in time consuming and repetitive manual work.

Web scraping enables us to access data, that we could not retrieve using traditional methods of data collection. Mainly because some types of data would not exist without the internet. Think of consumer reviews or social media posts. In other cases, web scraping enables us to collect data much more efficiently compared to traditional methods. Imagine you want to create a data file with demographic information of parliament members. This was obviously possible and done before web scraping became a possibility but included a lot of manual work using printed almanacs and typing the information into a table. Time consuming and error prone work. With web scraping we can set up scripts that gather this data directly from the website of a parliament including the possibility to rerun the collection process after the next election. But we don’t have to stop there. Maybe we want to analyse the sentiment of twitter messages written by parliament members by some socio-demographic variables as gender, age or educational background. We could retrieve both the tweets and the socio-demographic information using web scraping and analyse the gathered data in combination.

At this point, your alarm bells should hopefully be ringing. Collecting data from social media profiles sounds like an Orwellian nightmare, and if it is done inconsiderate, it is. We should not and will not collect data for the sake of data collection. Before scraping data on the internet we have to think about what data we actually need to answer our specific research question. Next we have to check if we are actually allowed to collect certain types of data and even if we are, if we should collect it. Do we need the names of people or users if we are collecting data? Do we actually need all socio-demographic variables available or can we keep it at a minimum, ensuring that individuals in our data remain unidentifiable? We are not web spiders, collecting every bit of data that is available, whether we need it or not. We are scientists and have to behave in a responsible and reasonable manner when collecting data.

Course contents

Part I serves as an introduction to R, the programming language we will use for web scraping as well as analysis of the data gathered. This introduction can necessarily only cover the basics of R, as our focus lies on introducing the concepts of web scraping. The introduction to R will be problem-oriented, i.e. we learn what we need for web scraping, but there will be a lot left out or only covered cursorily. See this as a starting point for your R journey, not its destination. Additional resources for deepening your R knowledge will be provided.

Part II forms the main part of the course and introduces basic and intermediate web scraping concepts. First we have to understand the structure of websites before we can start picking out their specific parts. We will learn how to identify data of interest on HTML pages, select the data using CSS selectors and extract it using the R package **rvest**. We will also look at scraping tables, content generated by dynamic websites and scraping of multi-page websites. We close with a closer look on web scraping ethics and good practice, already briefly discussed above, and what we can do to make sure we scrape responsibly.

Part III will cover the practical application of the acquired scraping knowledge to a small real world data analysis project, spanning the process from generating a research question over scraping appropriate data to cleaning and analysing the data with the goal of answering the formulated question. In this process, we will also deepen our R knowledge by looking at data cleaning, data transformation and graphical analysis in R using the **tidyverse** packages.

Part IV is a basic outlook into regular expressions. These are used for handling, transforming and analysing text data, a data type we will commonly encounter in web scraping. Some starting points for working with regular expressions, as well as pointers to additional external resources, are given.

The internet is an ever changing landscape. Where once every page we encountered was a simple HTML document, now many modern pages are less simplistic. Content is generated dynamically for each individual user and displayed information is not actually “present” in the HTML files but loaded through JavaScript or database connections. This also makes scraping harder. While still being possible, this requires advanced techniques that go beyond the scope of this introduction. You will still receive some pointers for where to look, should you want to follow these paths further. Also, when we begin scraping raw text from websites, we will need to learn about advanced handling of strings and regular expressions, which can also only be covered briefly here.

This introduction focuses on using **rvest** for scraping in R. This will suffice for many intermediate and advanced scraping projects, but alternatives exist within R and in other languages. While these also can not be covered here, you will gain a solid basis to pursue alternatives, should you so desire.

Technical requirements

All scraping in this introduction is done in R and RStudio using the **tidyverse** packages. Information on how to install the software and handle packages will be provided in chapters 1 and 2.

We will also need a browser for scraping. While in principle any popular browser should do, I would recommend using Chrome or Chromium, as all examples using the build in Web Developer Tools, are done in Chromium. You should make sure, that the URL bar in your browser shows the full URLs. In Chrome/Chromium this is done by right-clicking the URL bar and choosing “Always show full URLs” in the context menu.

While not strictly necessary, I would also advice using a decent text editor for opening HTML files. The options that come with your operating systems work, but why not do yourselves a favour and use something you can actually work with. Options are aplenty, but some of the more accessible while still powerful text editors I personally would recommend are Notepad++ on Windows (https://notepad-plus-plus.org/) and Atom on all operating systems (https://atom.io/). If your feel you are up to it, you can also dip your feet into Emacs, also available for every OS (https://www.gnu.org/software/emacs/), but note that this is an advanced option. Proceed at your own risk.

Conventions

All code on the website is set in this `code font`. This is true for R as well as HTML code. R code that is included in text paragraphs will not always be runnable and often serve illustrative purposes. All R code written in code blocks is runnable and look like this:

```
print("Hello World!")
## [1] "Hello World!"
```

If an R command produces relevant output, you will see the output following directly after the command, written in red and introduced by `##`.

You should in general try to run the code blocks yourself in RStudio. The only way to learn what is happening in R and web scraping, is doing it yourself. Run the examples and experiment with them. Every problem you will run into while changing the code is a learning opportunity. Notice, that code blocks are copyable. Mouse over the code block and you will see a copy symbol, or simply copy & paste like you are used to. But, I would strongly advice on writing as much code as you can yourself. Copying code examples by hand may not be intellectually challenging, but if your aim is to learn the R functions and syntax you have to write a lot of it before it can become second nature.

Names of packages are written in bold and exactly as they are named, e.g. **tidyverse**.

Acknowledgements

I would like to thank the authors of **bookdown**, the **tidyverse** and all R packages used in the creation of this course.

Very special thanks go to Lukas Höttges for his support in creating the website and in organizing the seminar at the University of Potsdam. To Hannah Gehrmann for the invaluable feedback on an early draft. To Fabian Class for technical support and numerous answers to my many questions. And to Sophia Varma for translating the original German draft and the great support.

Colophon

The website was built with:

```
sessioninfo::session_info()
```

```
## ─ Session info ───────────────────────────────────────────────────────────────
##  setting  value
##  version  R version 4.4.3 (2025-02-28)
##  os       Ubuntu 24.04.2 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       Europe/Berlin
##  date     2025-03-06
##  pandoc   3.2 @ /usr/lib/rstudio/resources/app/bin/quarto/bin/tools/x86_64/ (v
##  quarto   1.5.57 @ /usr/lib/rstudio/resources/app/bin/quarto/bin/quarto
##
## ─ Packages ───────────────────────────────────────────────────────────────────
##  package     * version date (UTC) lib source
##  bookdown      0.42    2025-01-07 [1] CRAN (R 4.4.3)
##  bslib         0.8.0   2024-07-29 [1] CRAN (R 4.4.1)
##  cachem        1.1.0   2024-05-16 [1] CRAN (R 4.4.1)
##  chromote      0.3.1   2024-08-30 [1] CRAN (R 4.4.2)
##  cli           3.6.3   2024-06-21 [1] CRAN (R 4.4.1)
##  colorspace    2.1-1   2024-07-26 [1] CRAN (R 4.4.1)
##  digest        0.6.37  2024-08-19 [1] CRAN (R 4.4.1)
##  dplyr       * 1.1.4   2023-11-17 [1] CRAN (R 4.4.0)
##  echarts4r   * 0.4.5   2023-06-16 [1] CRAN (R 4.4.2)
##  evaluate      0.24.0  2024-06-10 [1] CRAN (R 4.4.1)
##  fansi         1.0.6   2023-12-08 [1] CRAN (R 4.4.0)
##  fastmap       1.2.0   2024-05-15 [1] CRAN (R 4.4.1)
##  forcats     * 1.0.0   2023-01-29 [1] CRAN (R 4.4.0)
##  generics      0.1.3   2022-07-05 [1] CRAN (R 4.4.0)
##  ggplot2     * 3.5.1   2024-04-23 [1] CRAN (R 4.4.0)
##  glue          1.8.0   2024-09-30 [1] CRAN (R 4.4.1)
##  gtable        0.3.5   2024-04-22 [1] CRAN (R 4.4.0)
##  hms           1.1.3   2023-03-21 [1] CRAN (R 4.4.0)
##  htmltools     0.5.8.1 2024-04-04 [1] CRAN (R 4.4.0)
##  htmlwidgets   1.6.4   2023-12-06 [1] CRAN (R 4.4.0)
##  httpuv        1.6.15  2024-03-26 [1] CRAN (R 4.4.1)
##  httr           1.4.7   2023-08-15 [1] CRAN (R 4.4.0)
##  jquerylib      0.1.4   2021-04-26 [1] CRAN (R 4.4.0)
##  jsonlite       1.8.8   2023-12-04 [1] CRAN (R 4.4.0)
##  knitr        * 1.48    2024-07-07 [1] CRAN (R 4.4.1)
##  later          1.3.2   2023-12-06 [1] CRAN (R 4.4.1)
##  lifecycle      1.0.4   2023-11-07 [1] CRAN (R 4.4.0)
##  lubridate    * 1.9.3   2023-09-27 [1] CRAN (R 4.4.0)
##  magrittr       2.0.3   2022-03-30 [1] CRAN (R 4.4.0)
##  mime           0.12    2021-09-28 [1] CRAN (R 4.4.0)
##  munsell        0.5.1   2024-04-01 [1] CRAN (R 4.4.0)
##  pillar         1.9.0   2023-03-22 [1] CRAN (R 4.4.0)
##  pkgconfig      2.0.3   2019-09-22 [1] CRAN (R 4.4.0)
##  processx       3.8.4   2024-03-16 [1] CRAN (R 4.4.0)
##  promises       1.3.2   2024-11-28 [1] CRAN (R 4.4.2)
##  ps             1.8.0   2024-09-12 [1] CRAN (R 4.4.1)
##  purrr        * 1.0.2   2023-08-10 [1] CRAN (R 4.4.0)
##  R6             2.5.1   2021-08-19 [1] CRAN (R 4.4.0)
##  Rcpp           1.0.13  2024-07-17 [1] CRAN (R 4.4.1)
##  readr        * 2.1.5   2024-01-10 [1] CRAN (R 4.4.0)
##  rlang          1.1.4   2024-06-04 [1] CRAN (R 4.4.1)
##  rmarkdown      2.28    2024-08-17 [1] CRAN (R 4.4.1)
##  rstudioapi     0.16.0  2024-03-24 [1] CRAN (R 4.4.0)
##  rvest        * 1.0.4   2024-02-12 [1] CRAN (R 4.4.0)
##  sass           0.4.9   2024-03-15 [1] CRAN (R 4.4.0)
##  scales         1.3.0   2023-11-28 [1] CRAN (R 4.4.0)
##  sessioninfo    1.2.3   2025-02-05 [1] CRAN (R 4.4.3)
##  shiny          1.10.0  2024-12-14 [1] CRAN (R 4.4.2)
##  stringi        1.8.4   2024-05-06 [1] CRAN (R 4.4.1)
##  stringr      * 1.5.1   2023-11-14 [1] CRAN (R 4.4.0)
##  tibble       * 3.2.1   2023-03-20 [1] CRAN (R 4.4.0)
##  tidyr        * 1.3.1   2024-01-24 [1] CRAN (R 4.4.0)
##  tidyselect     1.2.1   2024-03-11 [1] CRAN (R 4.4.0)
##  tidyverse    * 2.0.0   2023-02-22 [1] CRAN (R 4.4.0)
##  timechange     0.3.0   2024-01-18 [1] CRAN (R 4.4.0)
##  tzdb           0.4.0   2023-05-12 [1] CRAN (R 4.4.0)
##  utf8           1.2.4   2023-10-22 [1] CRAN (R 4.4.0)
##  vctrs          0.6.5   2023-12-01 [1] CRAN (R 4.4.0)
##  websocket      1.4.2   2024-07-22 [1] CRAN (R 4.4.2)
##  withr         3.0.1   2024-07-31 [1] CRAN (R 4.4.1)
##  xfun          0.51    2025-02-19 [1] CRAN (R 4.4.3)
##  xml2          1.3.6   2023-12-04 [1] CRAN (R 4.4.0)
##  xtable        1.8-4   2019-04-21 [1] CRAN (R 4.4.2)
##  yaml          2.3.10  2024-07-26 [1] CRAN (R 4.4.1)
##
##  [1] /home/jtures/R/x86_64-pc-linux-gnu-library/4.4
##  [2] /usr/local/lib/R/site-library
##  [3] /usr/lib/R/site-library
##  [4] /usr/lib/R/library
##  * —— Packages attached to the search path.
##
## ──────────────────────────────────────────────────────────────────────────────
```