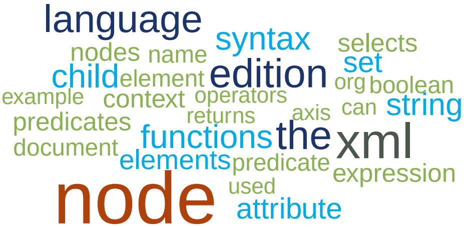
Introduction to Web Scraping with R

The Scraping Workflow



Simon Munzert | IPSDS

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools

2. you are able to parse HTML into R with `rvest`

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools
2. you are able to parse HTML into R with `rvest`
3. you are able to speak XPath

Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools
2. you are able to parse HTML into R with `rvest`
3. you are able to speak XPath
4. you are able to apply XPath expressions with `rvest`

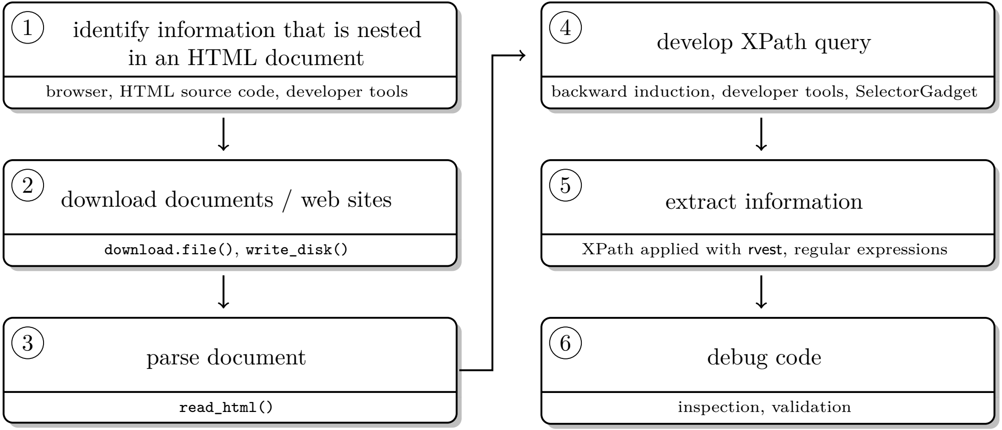
Where you stand now

You have learned the main tools necessary to scrape static webpages with R

1. you are able to inspect HTML pages in your browser using the web developer tools
2. you are able to parse HTML into R with `rvest`
3. you are able to speak XPath
4. you are able to apply XPath expressions with `rvest`
5. you are able to tidy web data with your R skills and regular expressions

The scraping workflow

The scraping workflow



1 identify information that is nested in an HTML document

browser, HTML source code, developer tools

2 download documents / web sites

`download.file()`, `write_disk()`

3 parse document

`read_html()`

4 develop XPath query

backward induction, developer tools, SelectorGadget

5 extract information

XPath applied with `rvest`, regular expressions

6 debug code

inspection, validation

Downloading HTML files

Stay modest when accessing lots of data

- content on the web is publicly available, ...
- but accessing the data causes server traffic
- stay polite by querying resources as sparsely as possible

Downloading HTML files

Stay modest when accessing lots of data

- content on the web is publicly available, ...
- but accessing the data causes server traffic
- stay polite by querying resources as sparsely as possible

Two easy-to-implement practices

1. do not bombard the server with requests—and if you have to, do at a reasonable speed
2. download HTML files first, then parse

Downloading HTML files

R code

```r
for (i in 1:length(list_of_urls)) {
    if (!file.exists(paste0(folder, file_names[i]))) {
        download.file(list_of_urls[i], destfile = paste0(folder, file_names[i]))
        Sys.sleep(runif(1, 1, 2))
    }
}
```

end

The code snippet explained

- loop over a list of urls
- `!file.exists()` checks whether a file does not yet exist in the local folder
- `download.file()` downloads the file to a folder; file name has to be specified
- `Sys.sleep()` suspends the execution of R code for a given time interval. Here: random interval between 1 and 2 seconds

Staying identifiable

Don’t be a phantom

- downloading massive amounts of data may arouse attention from server adminstrators
- assuming that you’ve got nothing to hide, you should stay identifiable beyond your IP address

Staying identifiable

Don't be a phantom

- downloading massive amounts of data may arouse attention from server adminstrators
- assuming that you've got nothing to hide, you should stay identifiable beyond your IP address

Two easy-to-implement practices

1. personally get in touch with website owners
2. use HTTP header fields `From` and `User-Agent` to provide information about yourself

Staying identifiable

R code

```r
url <- "http://a-totally-random-website.com"
session <- html_session(url, add_headers(From = "my@email.com", `User-Agent` = R.Version()$version.string))
headlines <- session %>% html_nodes(xpath = "p//a") %>%  html_text()
```

end

The code snippet explained

- `rvest`'s `html_session()` creates a session object that responds to HTTP and HTML methods
- here, we provide our email address and the current R version as User-Agent information
- this will pop up in the server logs—the webpage adminstrator has the chance to easily get in touch with you

Summary

Summary

- the basic scraping workflow with R is straightforward
- with great power comes great responsibility: stay polite on the web when scraping lots of data!
- more complexity is added when you want to gather data from multiple websites, or when dynamic elements such as forms or JavaScript content is involved
- we will consider such cases in upcoming sessions