A Primer to Web Scraping with R

Simon Munzert

Mannheim Centre for European Social Research

r-datacollection.com | @RDataCollection | @simonsaysnothin

October 2016

Introduction and Organizational Matters

Workshop outline

Time	**Topic**
08:30 a.m. - 10:15 a.m.	Introduction, setup, and overview
10.30 a.m. - 12.30 a.m.	Scraping static webpages with `rvest`
02.00 p.m. - 03.15 p.m.	Scraping with `RSelenium`; good practice
03.30 p.m. - 05.00 p.m.	Tapping APIs

Goals

After attending this course, . . .

- you have a fundamental overview of what's possible with R in terms of collecting data from the Web
- you are able to scrape information from static and dynamic websites using R
- you are able to access web services (APIs) with R

Goals

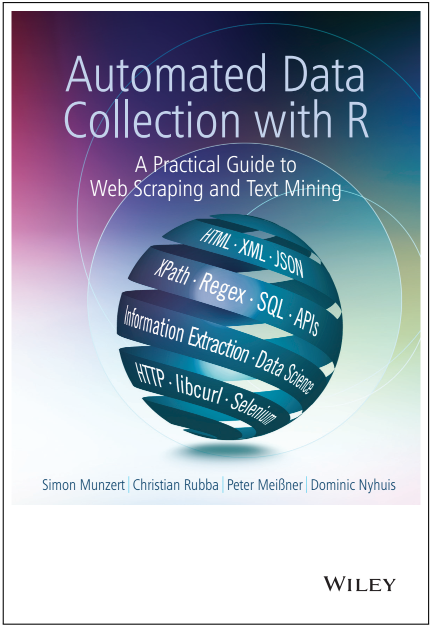
After attending this course, . . .

- you have a fundamental overview of what's possible with R in terms of collecting data from the Web
- you are able to scrape information from static and dynamic websites using R
- you are able to access web services (APIs) with R

→ the focus here is on **practical issues of web scraping**; see the **book** to get an overview of the **technical background** of web architecture!

The accompanying book

- contains most of which I tell you during the workshop (but much more, and presumably more accurate)
- written between 2012 and 2014 → not entirely up-to-date anymore, more on that later
- homepage with materials: www.r-datacollection.com


Automated Data Collection with R

A Practical Guide to Web Scraping and Text Mining

HTML · XML · JSON
XPath · Regex · SQL · APIs
Information Extraction · Data Science
HTTP · libcurl · Selenium

Simon Munzert | Christian Rubba | Peter Meißner | Dominic Nyhuis

WILEY

Web scraping. What? Why?

Web scraping

A.k.a. screen scraping, crawling, web harvesting; computer-aided collection of predominantly unstructured data (e.g., from HTML code)

The World Wide Web is full of various kinds of new data, e.g.:

- open government data
- search engine data
- services that track social behavior

Practical arguments

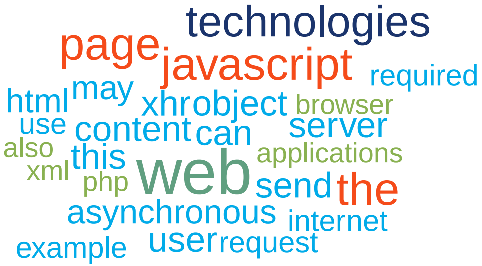
- financial resources are sparse
- . . . and so is our time
- reproducibility

Technical Setup

Please go to the following page now:

https://github.com/SocialScienceDataLab/Web-scraping-with-R-Extended-Edition

AJAX and Selenium



What's AJAX?

- HTML/HTTP are used for static display of content
- in order to display dynamic content, they lack
    1. mechanisms to detect user behavior in the browser (and not only on the server)
    2. a scripting engine that reacts on this behavior
    3. a mechanism for asynchronous queries
- **A**synchronous **J**avaScript **a**nd **X**ML' is a set of technologies that serve these purposes
- massively used in modern webpage design and architecture
- makes classical screen scraping more difficult

Example: https://twitter.com/POTUS

JavaScript

What's JavaScript?

- Programming language that connects well to web technologies
- W3C web standard
- native browser support
- extensible by many libraries
- *jQuery* library for DOM manipulation

JavaScript on the Web

How's JavaScript code embedded in HTML?

- between `<script>` tags
- as an external reference in the `scr` attribute of a `<script>` element
- directly in certain HTML attributes ('event handler')

JavaScript on the Web

DOM manipulation with JavaScript

- adding/removing HTML elements
- changing attributes
- modification of CSS styles
- ...

Example:

```
<script type="text/javascript" src="jquery-1.8.0.min.js"></script>
<script type="text/javascript" src="script1.js"></script>
```

Example

http://www.r-datacollection.com/materials/ajax/

Selenium

The problem reconsidered

- dynamic data requests are not stored in the static HTML page
- therefore, we cannot access them with classical methods and packages (`rvest`, `download.file()`, etc.)

The solution

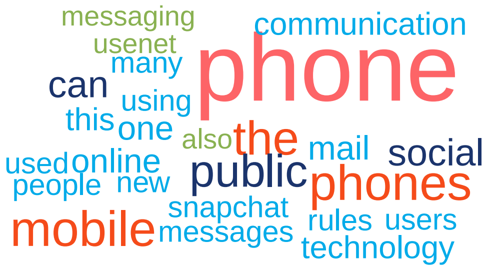
- initiate and control a web browser session with R
- let the browser do the JavaScript interpretation work and the manipulations in the live DOM tree
- access information from the web browser session

Selenium

What's Selenium?

- http://www.seleniumhq.org
- free software environment for automated web application testing
- several modules for different tasks; most important for our purposes: Selenium WebDriver
- Selenium WebDriver starts a server instance (as proxy) and passes commands (posed in R in our case) to the browser
- automated browsing via scripts

Good Practice



Is web scraping legal?

- no unambiguous yes or no in any country according to current jurisdiction
- so far, court cases (especially in the US) often (but not always) dealt with commercial interest and often (but not always) huge masses of data
    - eBay vs. Bidder's Edge
    - AP vs. Meltwater
    - Facebook vs. Pete Warden
    - United States vs. Aaron Swartz

A (not very useful) recommentation for your work

1. you take all the responsibility for your web scraping work
2. take all copyrights of a country's jurisdiction into account
3. if you publish data, do not commit copyright fraud
4. if in doubt, ask the author/creator/provider of data for permission—if your interest is entirely scientific, chances aren't bad that you get data
5. consult current jurisdiction, e.g. on http://blawgsearch.justia.com or from a laywer specialized on internet law

robots.txt

What's robots.txt?

- 'Robots Exclusion Protocol', informal protocol to prohibit web robots from crawling content
- located in the root directory of a website, e.g., http://www.google.com/robots.txt)
- documents which bot is allowed to crawl which resources (and which not)
- not a technical barrier, but a sign that asks for compliance

Examples:

- Google
- NYTimes

Syntax in `robots.txt`

Syntax

- not and official W3C standard, partly inconsistent syntax
- rules listed bot by bot
- general, bot-independent rules under '*' (most interesting entry for R-based crawlers)
- directories/folders listed separately

```
User-agent: Googlebot
Disallow: /images/
Disallow: /private/
```

```
User-agent: *
Disallow: /private/
```

Syntax in robots.txt

Universal ban

```
User-agent: *
Disallow: /
```

Separation of bots by empty line

```
User-agent: Googlebot
Disallow: /images/

User-agent: Slurp
Disallow: /images/
```

Allow declaration

```
User-agent: *
Disallow: /images/
Allow: /images/public/
```

Syntax in `robots.txt`

Crawl-delay (in seconds)

```
User-agent: *
Crawl-delay: 2
User-Agent: Googlebot
Disallow: /search/
```

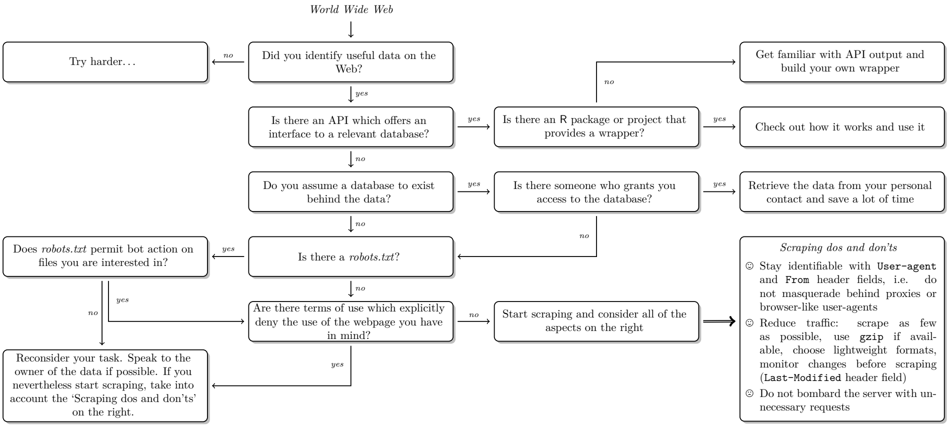
Robots `<meta>` tag

```
<meta name="robots" content="noindex,nofollow" />
```

How to deal with `robots.txt`?

- not clear if `robots.txt` is legally binding or not, and if yes for which activities
- originally not thought of as protection against small-scale web scraping applications, but against large-scale indexing bots
- guide to a webmaster's preferences with regards to visibility of content
- my advice: take `robots.txt` into account! If the data you are interested in are excluded from crawling: contact webmaster
- for crawling purposes: have a look at the new CRAN package `robotstxt`

Scraping etiquette



World Wide Web
Did you identify useful data on the Web?
no
Try harder...
yes
Is there an API which offers an interface to a relevant database?
yes
Is there an R package or project that provides a wrapper?
no
Get familiar with API output and build your own wrapper
yes
Check out how it works and use it
no
Do you assume a database to exist behind the data?
yes
Is there someone who grants you access to the database?
yes
Retrieve the data from your personal contact and save a lot of time
no
no
Is there a *robots.txt*?
yes
Does *robots.txt* permit bot action on files you are interested in?
no
yes
no
Are there terms of use which explicitly deny the use of the webpage you have in mind?
no
Start scraping and consider all of the aspects on the right
yes
Reconsider your task. Speak to the owner of the data if possible. If you nevertheless start scraping, take into account the 'Scraping dos and don'ts' on the right.

Scraping dos and don'ts

- ☺ Stay identifiable with `User-agent` and `From` header fields, i.e. do not masquerade behind proxies or browser-like user-agents
- ☺ Reduce traffic: scrape as few as possible, use `gzip` if available, choose lightweight formats, monitor changes before scraping (`Last-Modified` header field)
- ☹ Do not bombard the server with unnecessary requests