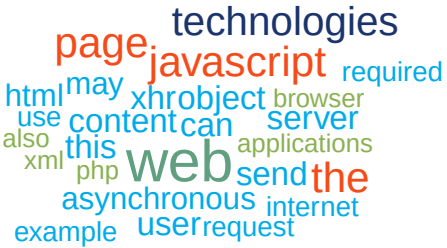
Introduction to Web Scraping with R

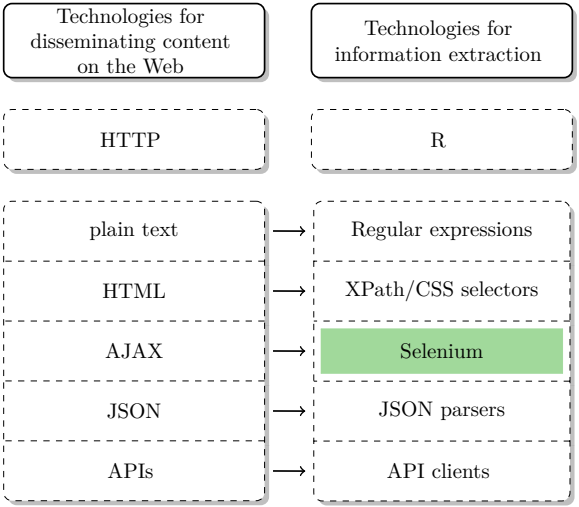
Selenium: Basics



Simon Munzert | IPSDS

Selenium

Technologies of the World Wide Web



Scraping dynamic webpages

The problem reconsidered

- dynamic data requests are not stored in the static HTML page
- therefore, we cannot access them with classical methods and packages (`httr`, `rvest`, `download.file()`, etc.)
- R is not a browser with a JavaScript rendering engine

Scraping dynamic webpages

The problem reconsidered

- dynamic data requests are not stored in the static HTML page
- therefore, we cannot access them with classical methods and packages (`httr`, `rvest`, `download.file()`, etc.)
- R is not a browser with a JavaScript rendering engine

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

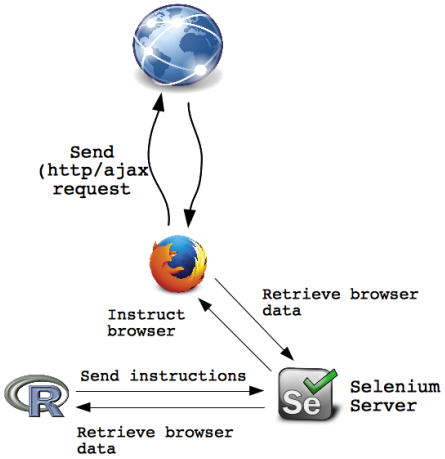
What's Selenium?

- free software environment for automated web application testing
- webpage: http://www.seleniumhq.org
- several modules for different tasks; most important for our purposes: Selenium WebDriver
- enables automated browsing via scripts

Selenium and R

The scraping workflow with Selenium and R

1. Selenium—an external, Java-based program—is launched
2. via the R package `RSelenium`, we remote-control Selenium and let it start a virtual server
3. we let Selenium start a browser session (e.g., Chrome)
4. everything we do in the browser—open a page, click on links or buttons, enter information—is described in R and sent to the server via the "remote-control" Selenium
5. we can gather the live HTML tree at any instance



Selenium and R

Software requirements

- Java, https://www.java.com/de/download/
- Selenium Standalone Server, newest version available here: http://www.seleniumhq.org/download/ or via RSelenium and `rsDriver()`
- browser (on most systems, both Chrome and Firefox seem to work reliably)
- `RSelenium` package