A Primer to Web Scraping with R

Simon Munzert

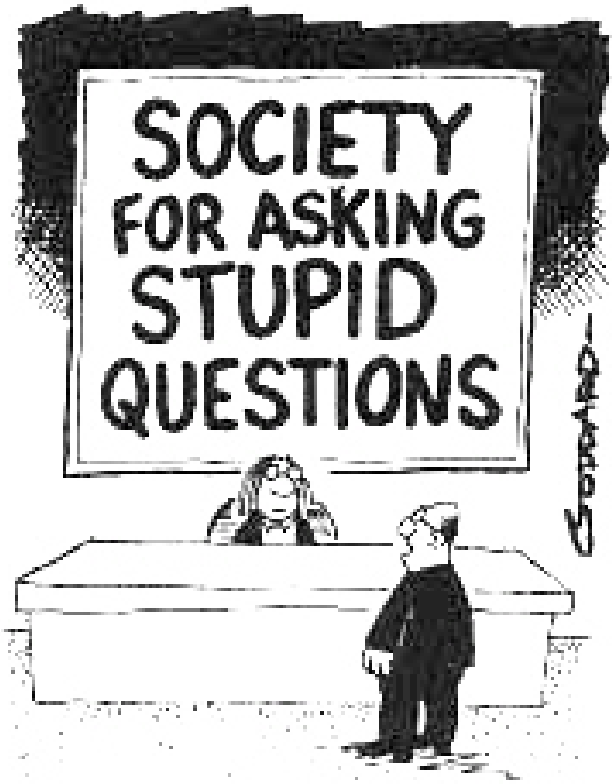
Mannheim Centre for European Social Research

r-datacollection.com | @RDataCollection | @simonsaysnothin

July 2016

Introduction and Organizational Matters

First: ask questions! No matter what...



"Excuse me, but is this The Society for Asking Stupid Questions?"

Workshop outline

Time	Topic
Slot 1, 08:30 a.m. - 10:15 a.m.	Introduction, setup, and overview
Slot 2, 10.30 a.m. - 12.30 a.m.	Scraping static webpages with rvest
Slot 3, 02.00 p.m. - 03.15 p.m.	Advanced scraping with RSelenium, good practice
Slot 4, 03.30 p.m. - 05.00 p.m.	Tapping APIs

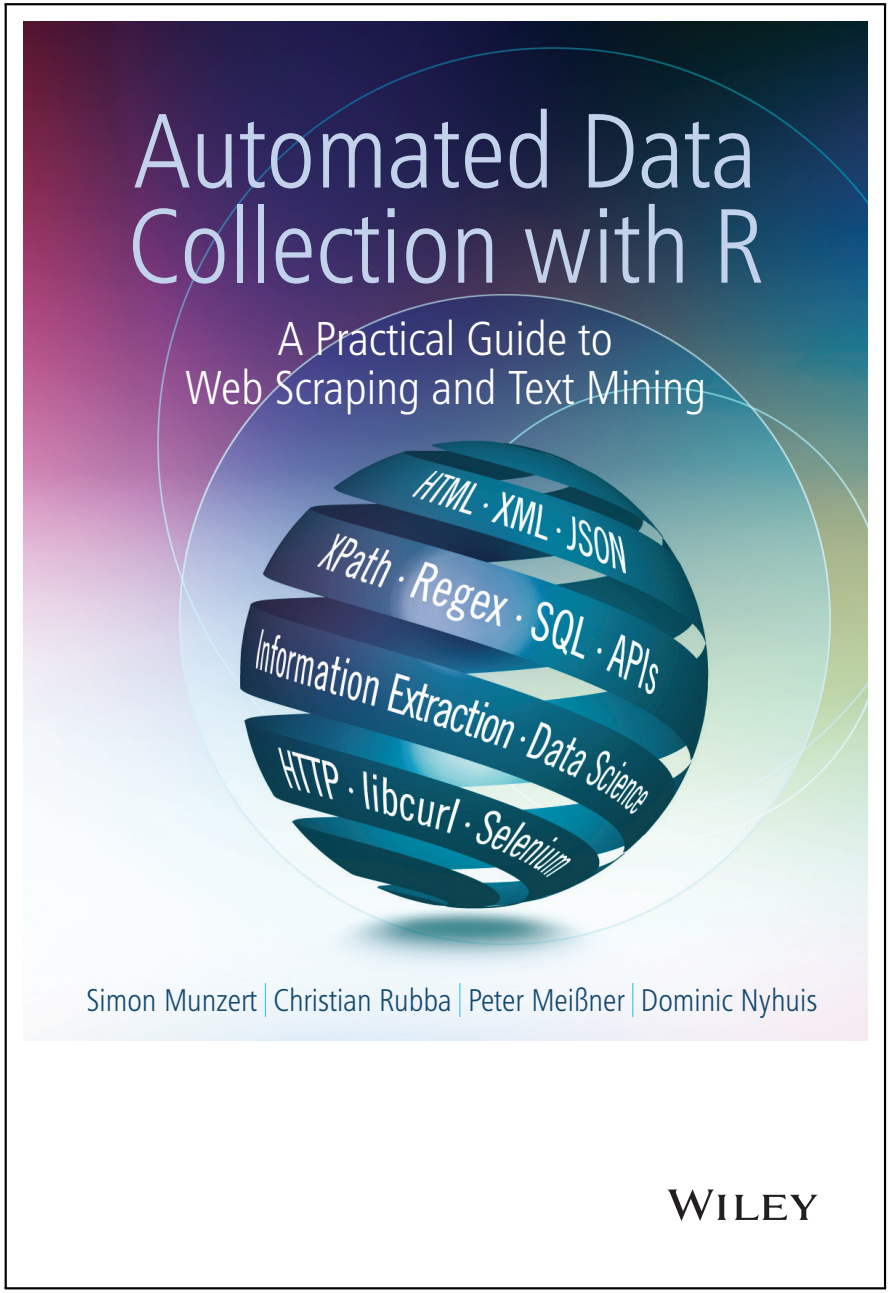
Goals

After attending this course, you ...

- have basic knowledge of web technologies
- you are able to scrape information from static and dynamic websites using R
- you are able to access web services (APIs) with R
- you can build up and maintain your own original data sets

The accompanying book

- contains most of which I tell you during the workshop (but much more, and much more accurate)
- written between 2012 and 2014 → not entirely up-to-date anymore, more on that later
- homepage with materials: www.r-datacollection.com
- manuscript you have is a modified excerpt
- if you find any errors in the book, please tell us!



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

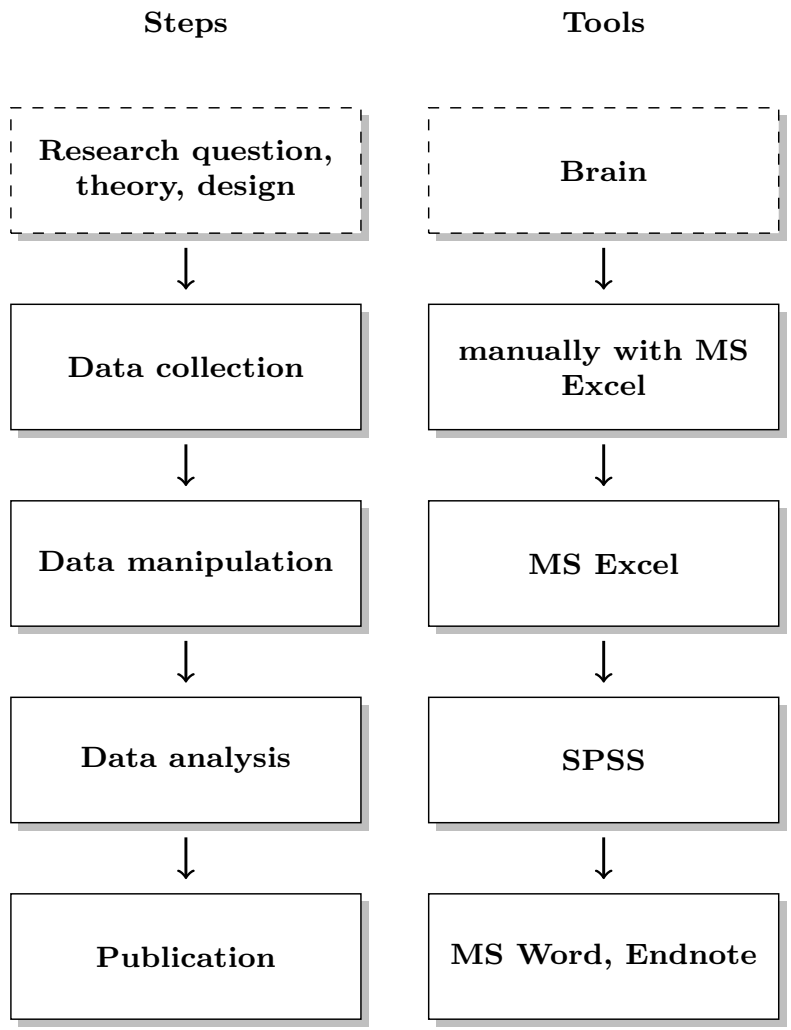
- financial resources are sparse
- . . . and so is our time
- reproducibility

Why R?

- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow

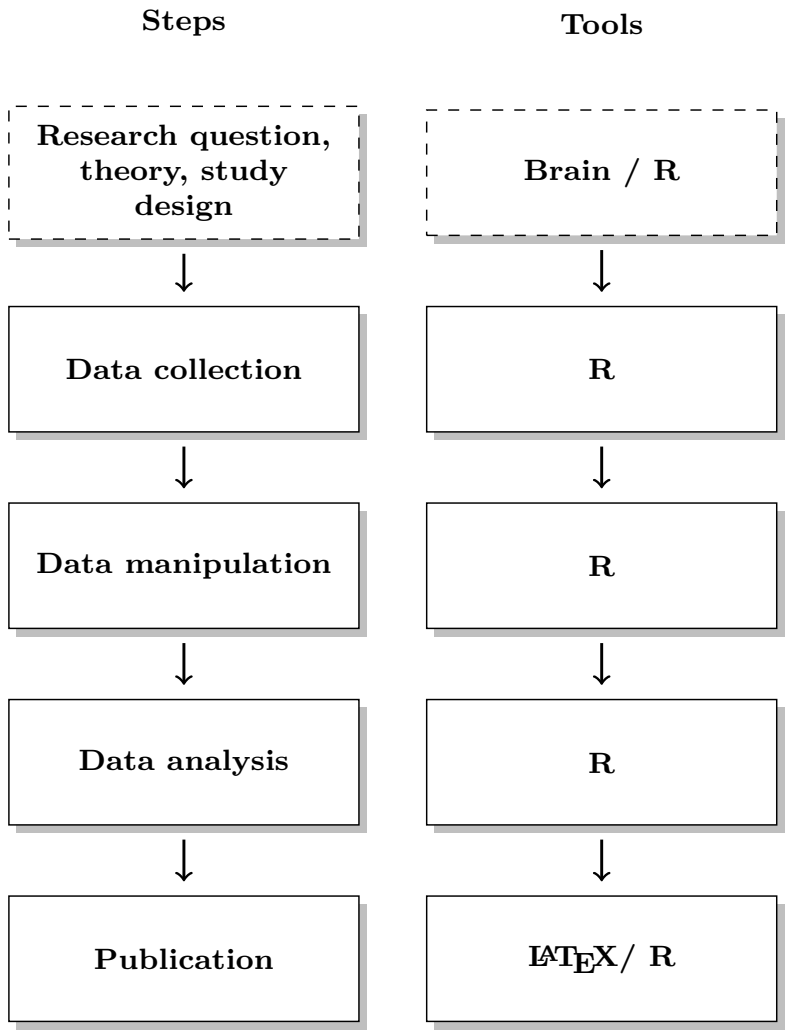
Why R?

- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow

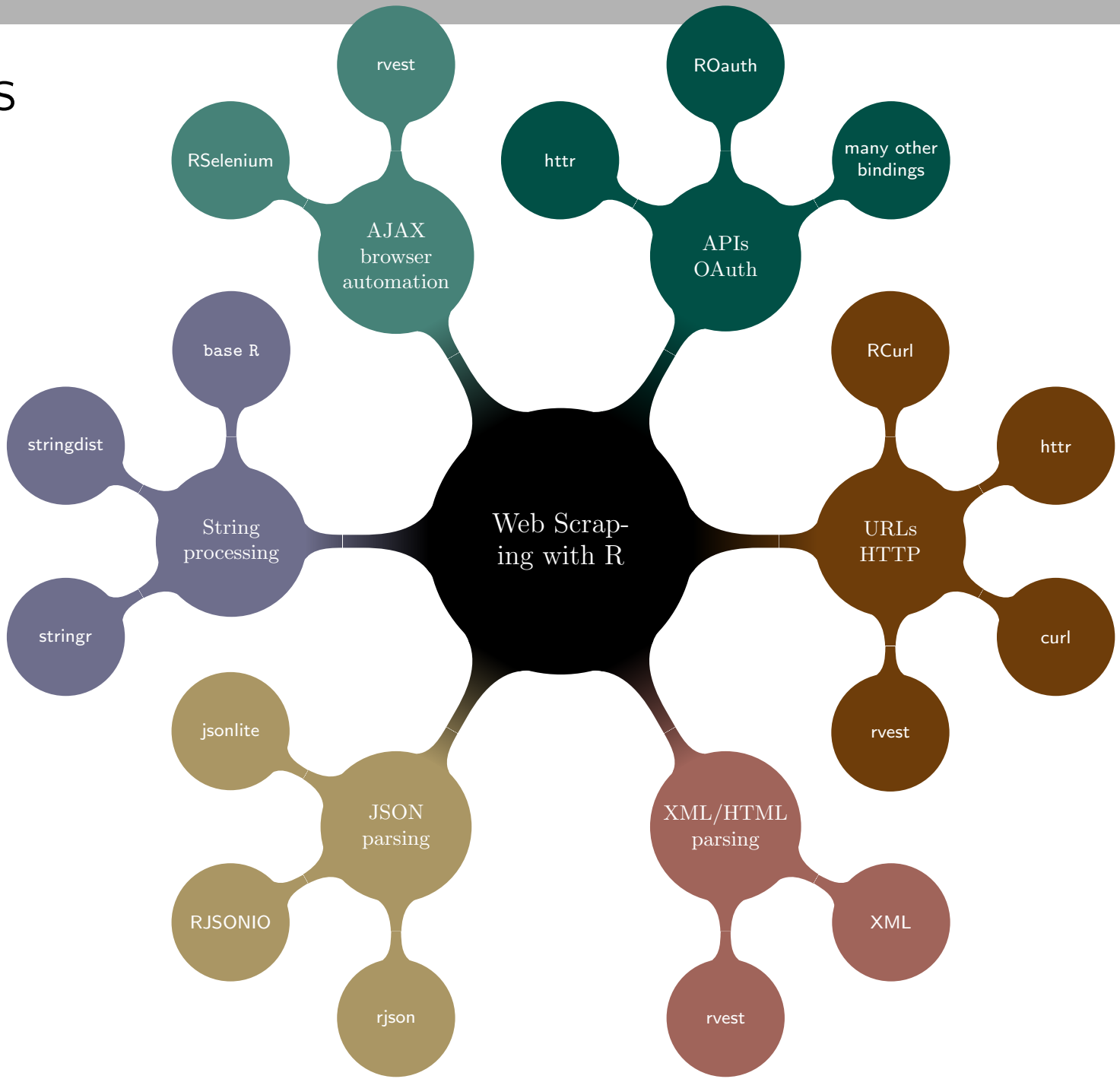


Why R?

- free
- open source
- large community
- powerful tools for statistical analysis
- powerful tools for visualization
- flexible in processing all kinds of data/languages
- useful in every step of the workflow



R tools

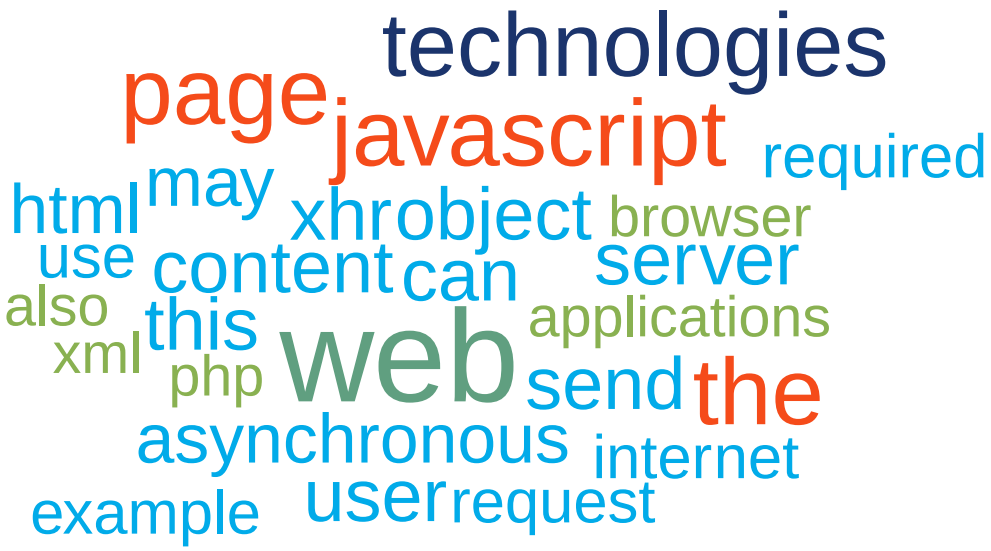

Web Scrap-
ing with R
AJAX
browser
automation
rvest
RSelenium
APIs
OAuth
ROauth
httr
many other
bindings
String
processing
base R
stringdist
stringr
URLs
HTTP
RCurl
httr
curl
rvest
JSON
parsing
jsonlite
RJSONIO
rjson
XML/HTML
parsing
XML
rvest

Technical Setup

Please go to the following page now:

https://github.com/simonmunzert/rscraping-jsm-2016

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:

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

JavaScript on the Web

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

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

<head>
<title>Collected R wisdoms</title>
</head>

<body>
<div id="R Inventor" lang="english" date="June/2003">
  <h1>Robert Gentleman</h1>
  <p><i>'What we have is nice, but we need something very different'</i></p>
  <p><b>Source: </b>Statistical Computing 2003, Reisensburg</p>
</div>

<div lang="english" date="October/2011">
  <h1>Rolf Turner</h1>
  <p><i>'R is wonderful, but it cannot work magic'</i> <br><emph>answering a request for automatic generation of '
        data from a known mean and 95% CI'</emph></p>
  <p><b>Source: </b><a href="https://stat.ethz.ch/mailman/listinfo/r-help">R-help</a></p>
</div>

<address><a href="http://www.r-datacollection.com"><i>The book homepage</i></a></address>
</body>
</html>
```

JavaScript on the Web

A JavaScript code snippet

```
1 $(document).ready(function() {
2     $("p").hide();
3     $("h1").click(function(){
4         $(this).nextAll().slideToggle(300);
5     });
6 });
```

- $() operator: addresses DOM elements
- `ready()`: JavaScript execuption starts when the complete DOM is ready, i.e. fetched from the server
- `hide()`: element is hidden at first place
- `click` Event: identifies mouse click and executes a certain action
- `nextAll()`: all subsequent elements in the DOM are addressed
- `slideToggle()`: Toggle effect, 300 milli-seconds

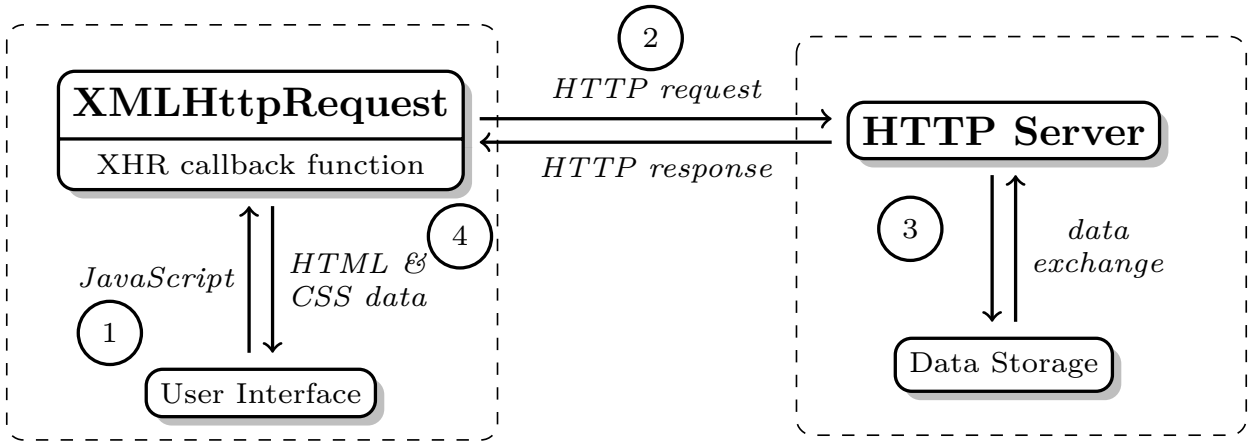
JavaScript on the Web

```
library(XML)
(fortunes1 <- htmlParse("../materials/fortunes1.html"))
## <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
## <html>
## <head>
## <script type="text/javascript" src="jquery-1.8.0.min.js"></script><script type="text/java
## </head>
## <body>
## <div id="R Inventor" lang="english" date="June/2003">
##   <h1>Robert Gentleman</h1>
##   <p><i>'What we have is nice, but we need something very different'</i></p>
##   <p><b>Source: </b>Statistical Computing 2003, Reisensburg</p>
## </div>
##
## <div lang="english" date="October/2011">
##   <h1>Rolf Turner</h1>
##   <p><i>'R is wonderful, but it cannot work magic'</i> <br><emph>answering a request for
##   <p><b>Source: </b><a href="https://stat.ethz.ch/mailman/listinfo/r-help">R-help</a></p
## </div>
##
## <address><a href="http://www.r-datacollection.com"><i>The book homepage</i></a></address
## </body>
## </html>
##
```

XHR

What's XHR?

- **X**ML**H**ttp**R**equest
- interface for dynamic HTTP client-server communication
- classic HTTP: synchronous, XHR: asynchronous



XHR

Example: dynamic import of HTML

```
library(XML)
(fortunes2 <- htmlParse("../materials/fortunes2.html"))
## <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
## <html>
## <head>
## <script type="text/javascript" src="jquery-1.8.0.min.js"></script><script type="text/java
## </head>
## <body>
## <address><a href="http://www.r-datacollection.com"><i>The book homepage</i></a></address>
## </body>
## </html>
##
```

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

XHR

Example: processing of JSON data

```
1 $("#quoteButton").click(function(){
2     $.getJSON("quotes/all_quotes.json", function(data){
3         $.each(data, function(key, value){
4             $("body").prepend("<div date='"+value.date+"'><h1>"+value.author+"</h1><
                  p><i>'"+value.quote+"'</i></p><p><b>Source: </b>'"+value.source+"
                  '</p></div>");
5         });
6     });
7 });
```

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 WebDiver 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

```
1 User-agent: Googlebot
2 Disallow: /images/
3 Disallow: /private/
```

```
1 User-agent: *
2 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