Web scraping part II

Programming for Statistical Science

Shawn Santo

Supplementary materials

Full video lecture available in Zoom Cloud Recordings

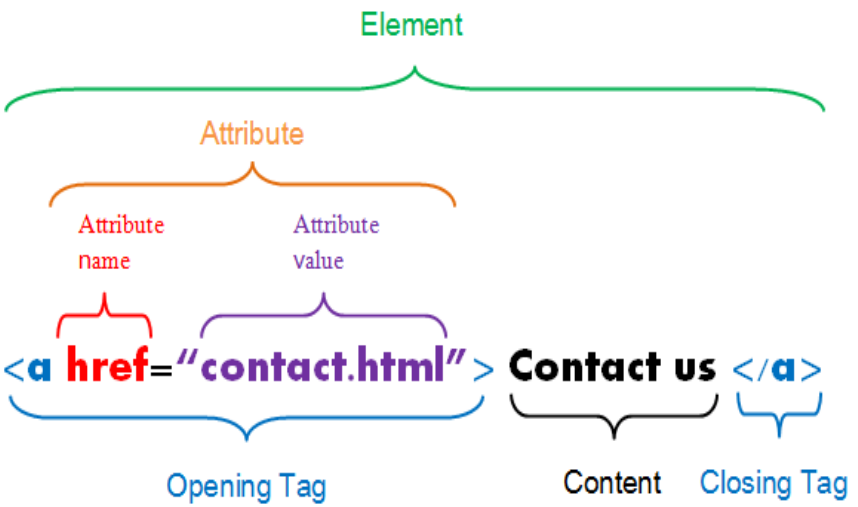
Additional resources

- Web scraping cheat sheet
- `RSelenium` website

Recall

Hypertext Markup Language

- HTML describes the structure of a web page; your browser interprets the structure and contents and displays the results.
- The basic building blocks include elements, tags, and attributes.
  - an element is a component of an HTML document
  - elements contain tags (start and end tag)
  - attributes provide additional information about HTML elements



HTML vs. XML

HTML snippet

```
<tr>
<td class=date>Jul 11, Sat</td>
<td class=where>
   <a class=mapq href="http://www.google.com/
   <span class=more>
   Wellington SP, West Shore Rd, Bristol, NH
   {43.639648,-71.779373}
   <a class="maplink" href="http://bing.com/m
   <a class="maplink" href="http://www.google
   <a class="maplink" href="https://www.mapqu
   <a class="maplink" href="http://www.openst
   </span>
</td>
<td class=name><a href="http://www.swimnewfou
<td class=distance>5 km, 10 km, 10 mi</td>
<td class=more><span class=time>7:00 AM</span
</tr>
```

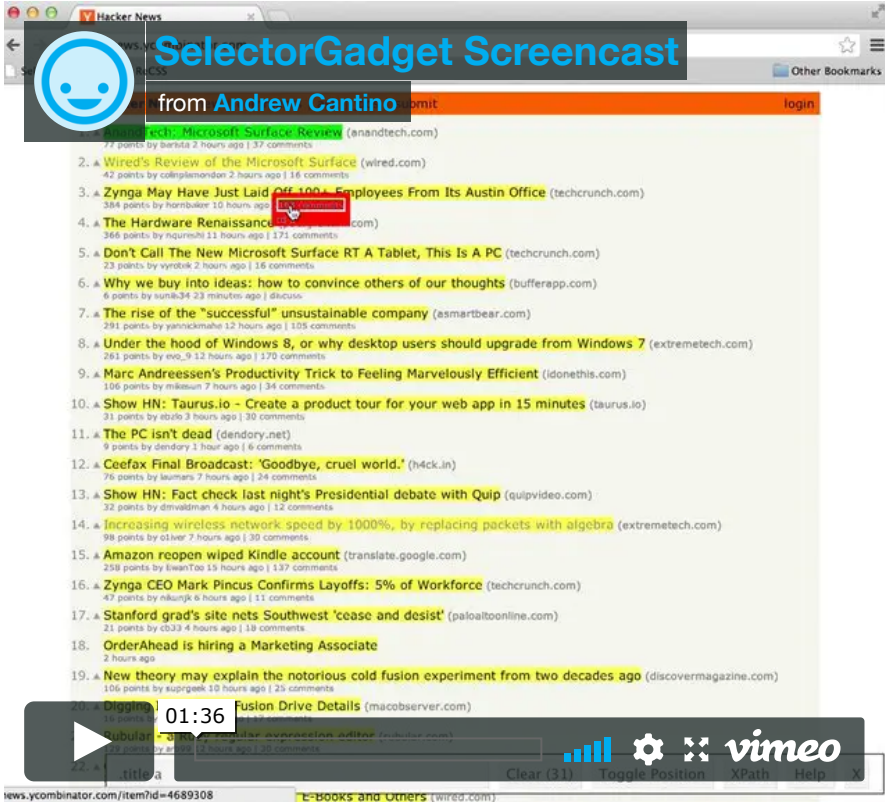
XML snippet

```
<swim>
  Swim with a Mission
  <location>
    Wellington SP, West Shore Rd, Bristol, NH
  </location>
  <link>
    http://www.swimnewfoundlake.com/
  </link>
  <date>
    Jul 11, Sat
  </date>
  <distance>
    5 km, 10 km, 10 mi
  </distance>
</swim>
```

SelectorGadget

In CSS, selectors are patterns used to select the element(s) you want to style.

SelectorGadget makes identifying the CSS selector you need as easy as clicking on items on a webpage.



Web scraping workflow

1. Understand the website's hierarchy and what information you need.
2. Use SelectorGadget to identify relevant CSS selectors.
3. Read html by passing a url and subset the resulting html document using CSS selectors.

```
read_html(url) %>%
  html_nodes(css = "specified_css_selector")
```

4. Further extract attributes, text, or tags by adding another layer with

```
read_html(url) %>%
  html_nodes(css = "specified_css_selector") %>%
  html_*()
```

where `*` is `text`, `attr`, `attrs`, `name`, or `table`.

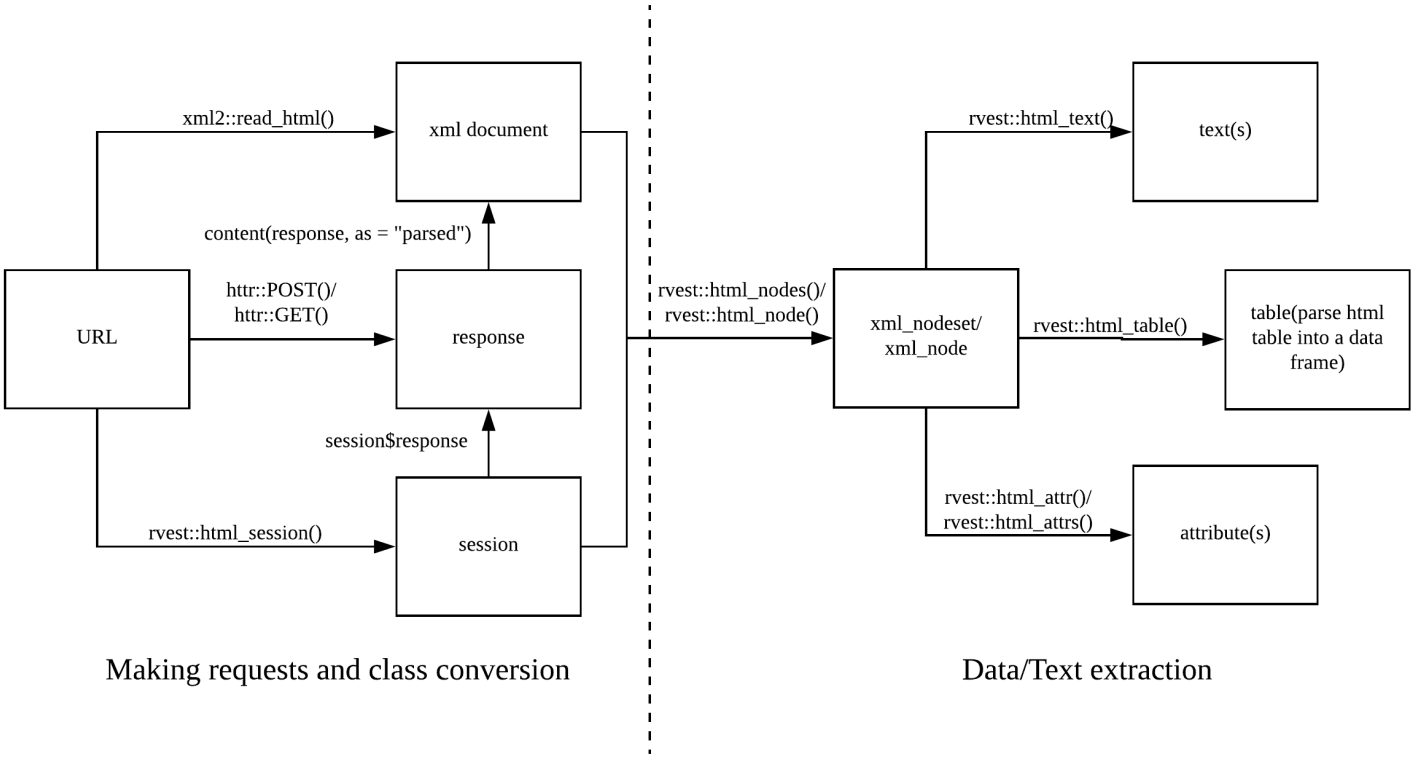
Example with `html_table()`

http://www.tornadohistoryproject.com/tornado/North-Carolina/2017/table

```
library(rvest)
library(tidyverse)

url <- "http://www.tornadohistoryproject.com/tornado/North-Carolina/2017/table"

nc_tornado <- read_html(url) %>%
  html_nodes("#results") %>%
  html_table(header = TRUE) %>%
  .[[1]] %>%
  janitor::clean_names() %>%
  select(date:lift_lon)

glimpse(nc_tornado)
```

```
#> Rows: 40
#> Columns: 15
#> $ date              <chr> "2017-02-15", "2017-03-31", "2017-05-01", "2017-05-…
#> $ time              <chr> "10:53:00 3", "16:15:00 3", "13:54:00 3", "01:12:00…
#> $ state_s           <chr> "North Carolina", "North Carolina", "North Carolina…
#> $ fujita            <chr> "1", "1", "0", "0", "1", "0", "0", "0", "1", "Fujit…
#> $ fatalities        <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "Fatal…
#> $ injuries          <chr> "0", "0", "0", "0", "0", "0", "0", "0", "1", "Injur…
#> $ width             <chr> "60", "100", "50", "375", "250", "100", "75", "30",…
#> $ length            <chr> "3.18", "4.8", "2.67", "3.26", "1", "11.71", "8.12"…
#> $ affected_counties <chr> "Brunswick", "Bertie", "Catawba", "Rockingham", "Gr…
#> $ damage            <chr> "$80000", "$250000", "$10000", "$40", "$100000", "$…
#> $ crop_loss         <chr> "-", "-", "-", "-", "-", "-", "-", "-", "-", "Crop …
#> $ touch_lat         <chr> "34.006", "36.2081", "35.61", "36.4869", "36.365", …
#> $ touch_lon         <chr> "-78.6088", "-76.9334", "-81.2", "-79.7452", "-78.5…
#> $ lift_lat          <chr> "34.009", "36.2213", "35.64", "36.534", "36.3792", …
#> $ lift_lon          <chr> "-78.5534", "-76.8488", "-81.17", "-79.7494", "-78.…
```

Overview



Source: https://github.com/yusuzech/r-web-scraping-cheat-sheet/blob/master/README.md

Recall previous exercise

Go to http://books.toscrape.com/catalogue/page-1.html and scrape the first five pages of data on books with regards to their

1. title
2. price
3. star rating

Organize your results in a neatly formatted tibble similar to below.

```
# A tibble: 100 x 3
   title                                                price rating
   <chr>                                                <chr> <chr>
 1 A Light in the Attic                                 £51.… Three
 2 Tipping the Velvet                                   £53.… One
 3 Soumission                                           £50.… One
 4 Sharp Objects                                        £47.… Four
 5 Sapiens: A Brief History of Humankind                £54.… Five
 6 The Requiem Red                                      £22.… One
 7 The Dirty Little Secrets of Getting Your Dream J…    £33.… Four
 8 The Coming Woman: A Novel Based on the Life of t…    £17.… Three
 9 The Boys in the Boat: Nine Americans and Their E…    £22.… Four
10 The Black Maria                                      £52.… One
# … with 90 more rows
```

Web scraping considerations

Best practices

- Abide by a site's terms and conditions.
- Respect robots.txt.
  - https://www.facebook.com/robots.txt
  - https://www.wegmans.com/robots.txt
  - https://www.google.com/robots.txt
- Cache your `read_html()` chunks. Isolate these chunks.
- Avoid using `read_html()` in code that is iterated.
- Do not overload the server at peak hours.
  - Implement delayed crawls: `Sys.sleep(rexp(1) + 4)`
- If available, use a site's API.
- Do not violate any copyright laws.

Other considerations

- Disguise your IP address.
  - `httr::use_proxy()`

- Avoid scraping behind pages protected by log-in, unless it is permitted by the site.
  - `html_session()`

- Watch out for honey pot traps - invisible links to normal visitors, but present in HTML code and found by web scrapers.

Beyond `rvest` and static sites

Limitations of using `rvest` functions

- It is difficult to make your code reproducible long term. When a website or the HTML changes, your code may no longer work.
    - CSS selectors change
    - Contents are moved
    - Switch from HTML to JavaScript

- Websites that rely heavily on JavaScript

What is JavaScript?

- Scripting language for building interactive web pages
- Basis for web, mobile, and network applications
- Every browser has a JavaScript engine that can execute JavaScript code.
    - Chrome and Edge: V8
    - Safari: JavaScriptCore
    - Firefox: SpiderMonkey

If `read_html()` is meant for HTML, what can we do?

Possible solutions

1. Use your browser's developer tools (Chrome is easiest)
2. Execute JavaScript in R
3. Use package `Rselenium` or other web drivers
    - http://ropensci.github.io/RSelenium/

We'll focus on the first option...

> In order for the information to get from their server and show up on a page in your browser, that information had to have been returned in an HTTP response somewhere.
>
> It usually means that you won't be making an HTTP request to the page's URL that you see at the top of your browser window, but instead you'll need to find the URL of the AJAX request that's going on in the background to fetch the data from the server and load it into the page.
>
> Hartley Brody

Live demo

Exercise

http://quotes.toscrape.com/ provides quotes for scraping. This site scraping sandox provides various endpoints that present different scraping challenges. Try and scrape the first 50 quotes and authors at the following endpoints.

- http://quotes.toscrape.com/scroll
- http://quotes.toscrape.com/js/

First, try using the typical approach with `rvest` to understand what is going on.

References

1. Scraping Sandbox. (2020). http://toscrape.com/.

2. SelectorGadget: point and click CSS selectors. (2020). Selectorgadget.com. https://selectorgadget.com/.

3. yusuzech/r-web-scraping-cheat-sheet. (2020). https://github.com/yusuzech/r-web-scraping-cheat-sheet.