Web Scraping with `{rvest}`

Data Aquisition and Distribution

Dr Zak Varty

Aquiring Data by Web Scraping

The data you need will not always be delivered to you, sometimes you have to go out and get it for yourself.

- **Web scraping** is one common way to obtain this data
- **APIs** are another common way to both obtain and distribute data

This video we will focus of the former: extracting data from a HTML webpage.

1. What is a webpage?

What is a webpage?

Like with LaTeX, content and presentation are handled separately. With webpages, this separation is even more extreme.

- **HyperText Markup Language** (HTML) files store content of a webpage.
- **Cascading Style Sheet** (CSS) files describe how that content should be displayed.

A Basic HTML page - Code

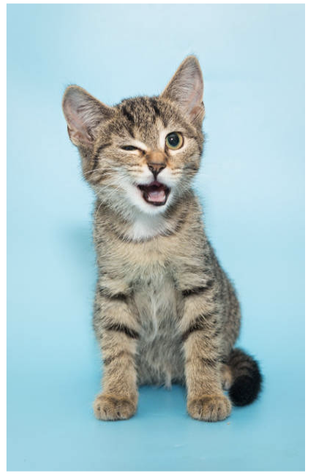
```
<html>
<head>
  <title>Page title</title>
</head>
<body>
  <h1 id='first'>A level 1 heading</h1>
  <p>Hello World!</p>
  <p>Here is some plain text &amp;amp; <b>some bold text.</b></p>
  <img src='myimg.png' width='100' height='100'>
</body>
```

A Basic HMTL Page - In Browser

A level 1 heading

Hello World!

Here is some plain text & **some bold text.**



CSS Zen Garden - multiple CSS files formatting the same HTML.

A Basic HTML Page - Structure

```
<html>
<head>
  <title>Page title</title>
</head>
<body>
  <h1 id='first'>A level 1 heading</h1>
  <p>Hello World!</p>
  <p>Here is some plain text &amp;amp; <b>some bold text.</b></p>
  <img src='myimg.png' width='100' height='100'>
</body>
```

- Escape characters: > is `>`, < is `<`, & is `&`, ...

Important HTML Elements

- The `<html>` element must enclose every HTML page:
  - `<head>` element contains metadata,
  - `<body>` element contains content displayed in browser.
- Block elements: headers `<h1>`, ..., `<h6>`, paragraphs `<p>`, lists `<ol>` & `<ul>`.
- Inline elements: bold `<b>`, italic `<i>`, emphasis `<em>`, hyperlinks `<a>`.

Resources: MDN Web Docs & W3schools website.

HTML Attributes

HTML attributes are contained within the opening tag.

```
<tag attribute1='value1' attribute2='value2'>element contents</tag>
```

- `id` and `class` attributes are usually the most important in relation to web scraping.

CSS Selectors

- CSS has it's own system for selecting elements of a webpage: **selectors**.

CSS Selectors can work on the level of an element type, a class, or a tag and these can be used in a nested (or **cascading**) way.

- The `p` selector will select all paragraph `<p>` elements.
- The `.title` selector will select all elements with class "`title`".
- The `p.special` selector will select all`<p>` elements with class "`special`".
- The `#title` selector will select the element with the id attribute "`title`".

Which Attributes and Selectors Do You Need?

Before you can scrape data, you first need to be able to describe what you want to scrape!

- right click + “inspect page source” (F12)
- right click + “inspect”
- Rvest Selector Gadget (very useful but fallible)

Start simple and build your confidence. Dynamic webpages can be more difficult.

2. Reading HTML with {rvest}

Reading HTML with {rvest}

{rvest} gives us funcitonality just like {readr}.

```
1 html <- rvest::read_html("https://www.zakvarty.com/professional/teaching.html")
2 class(html)
```

```
[1] "xml_document" "xml_node"
```

Rather than a tibble we get an xml_document, an object from {xml2}.

```
1 html
```

```
{html_document}
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
[2] <body class="nav-fixed">\n\n<div id="quarto-search-results"></div>\n  <he ...
```

Extracting HTML elements

```
 1 library(rvest)
 2 html %>% html_element("h1")
 3 ## {html_node}
 4 ## <h1>
 5 html %>% html_elements("h2")
 6 ## {xml_nodeset (2)}
 7 ## [1] <h2 id="toc-title">On this page</h2>
 8 ## [2] <h2 class="anchored" data-anchor-id="course-history">Course History</h2>
 9 html %>% html_elements("p")
10 ## {xml_nodeset (2)}
11 ## [1] <p>I am fortunate to have had the opportunity to teach in a variety of ro ...
12 ## [2] <p>I am an associate fellow of the Higher Education Academy, which you ca ...
```

```
1 html %>% html_elements("p a,h2")
```

```
{xml_nodeset (3)}
[1] <h2 id="toc-title">On this page</h2>
[2] <a href="https://www.advance-he.ac.uk/fellowship/associate-fellowship">he ...
[3] <h2 class="anchored" data-anchor-id="course-history">Course History</h2>
```

Extracting data from HTML elements

- We have the HTML elements we care about… now what?
- Depends if you are interested in the contents or the attributes of those elements.
- Ideally, someone else will have put everything into a table for you already!

Extracting text

Extract text using `rvest::html_text()` or `rvest::html_text2()`.

```
1 html %>%
2   html_elements("#teaching li") %>%
3   html_text2()
```

```
[1] "one-to-one tuition for high school students;"
[2] "running workshops and computer labs for undergraduate and postgraduate modules;"
[3] "delivering short courses on scientific communication and LaTeX;"
[4] "supervising an undergraduate research project;"
[5] "developing and lecturing postgraduate modules in statistics and data science."
```

Do you want to extract like the HTML file or like the browser display?

Extracting Attributes

- Attributes can also contain useful data that you want to extract (links, image sizes, image paths,...).
- Get twitter link from the icon in the footer of the webpage using Selector Gadget.

```
1 html %>% html_element(".compact:nth-child(1) .nav-link")
```

```
{html_node}
<a class="nav-link" href="https://www.twitter.com/zakvarty">
[1] <i class="bi bi-twitter" role="img">\n</i>
```

Extracting Attributes (2)

- Extract the `href` attribute.

```
1 html %>%
2   html_elements(".compact:nth-child(1) .nav-link") %>%
3   html_attr("href")
```

```
[1] "https://www.twitter.com/zakvarty"
```

Note: Attributes are always extracted as strings, so may need reformatting before analysis.

HTML Tables

There are four main elements to know about that make up an HTML table:

- `<table>`,
- `<tr>` (table row),
- `<th>` (table heading),
- `<td>` (table data).

Example HTML Table

```
<table>
  <tr>
    <th>Name</th>
    <th>Number</th>
  </tr>
  <tr>
    <td>A</td>
    <td>1</td>
  </tr>
  <tr>
    <td>B</td>
    <td>2</td>
  </tr>
  <tr>
    <td>C</td>
    <td>3</td>
  </tr>
</table>
```

Extracting HTML Tables

`{rvest}` has a useful function `html_table()` to help us.

```
1 html_2 %>%
2   html_element("table") %>%
3   html_table()
```

```
# A tibble: 3 × 2
  Name  Number
  <chr>  <int>
1 A          1
2 B          2
3 C          3
```

Extracting HTML Tables (2)

```
1 html %>%
2   html_element("table") %>%
3   html_table()
```

```
# A tibble: 25 × 3
   Year      Course                                  Role
   <chr>     <chr>                                   <chr>
 1 "2021-22" Supervised Learning                     Lecturer
 2 ""        Ethics in Data Science I                Lecturer
 3 ""        Ethics in Data Science II               Lecturer
 4 "—"       —                                       —
 5 "2020-21" MATH562/582: Extreme Value Theory       Lecturer
 6 ""        MATH331: Bayesian Inference             Graduate teaching assista…
 7 ""        MATH330: Likelihood Inference           Graduate teaching assista…
 8 "2019-20" DSCI485: Introduction to LaTeX          Co-leading short course
 9 ""        MATH566: Longitudinal Data Analysis     Graduate teaching assista…
10 "2018-19" STOR-i Internship: Introduction to LaTeX Co-leading short course
# … with 15 more rows
```

Tip when building tibbles

- Aim: build a tibble with 1 row per observation unit in the HTML (table row, list item, etc).

1. Use `html_elements()` to select the elements that contain each observation unit;
2. Use `html_element()` to extract the variables from each of those observations.

This avoids issues with missing values.

Tip Example: Star Wars

Example from the star wars dataset.

```
starwars_html <- minimal_html("
  <ul>
    <li><b>C-3PO</b> is a <i>droid</i> that weighs <span class='weight'>167 kg</span><
    <li><b>R2-D2</b> is a <i>droid</i> that weighs <span class='weight'>96 kg</span></
    <li><b>Yoda</b> weighs <span class='weight'>66 kg</span></li>
    <li><b>R4-P17</b> is a <i>droid</i></li>
  </ul>
  ")
```

Tip Example: what not to do

Do not try to extract each element directly: vectors of differing lengths.

```
starwars_html %>% html_elements("b") %>% html_text2()
```

```
[1] "C-3PO"  "R2-D2"  "Yoda"   "R4-P17"
```

```
starwars_html %>% html_elements("i") %>% html_text2()
```

```
[1] "droid" "droid" "droid"
```

```
starwars_html %>% html_elements(".weight") %>% html_text2()
```

```
[1] "167 kg" "96 kg"  "66 kg"
```

Tip Example: correct approach

First select the elements that contain each observation unit.

```
starwars_characters <- starwars_html %>% html_elements("li")
```

Then extract the variables from each of those observations.

```
tibble::tibble(
  name = starwars_characters %>% html_element("b") %>% html_text2(),
  species = starwars_characters %>% html_element("i") %>% html_text2(),
  weight = starwars_characters %>% html_element(".weight") %>% html_text2()
)
```

```
# A tibble: 4 × 3
  name    species weight
  <chr>   <chr>   <chr>
1 C-3PO   droid   167 kg
2 R2-D2   droid   96 kg
3 Yoda    <NA>    66 kg
4 R4-P17  droid   <NA>
```

Wrapping Up

1. Structure of webpages.

2. Using `{rvest}` package to extract the elements of interest.

3. Formatting those extracted into useful formats.