Department for Transport

An Introduction to Web Scraping

Coffee and Coding

Department for Transport

To start…

- Who knows what web scraping is?
- Who has used it before?
- What is the main thing you want to learn today?
- This is an introduction to show what is possible and where to look for more guidance

Department for Transport

In this C&C talk

1. What is web scraping
2. When is web scraping useful
3. Legal implications
4. How web scraping works
5. Useful R packages
6. Example 1 – Simple HTML table
7. Example 2 – The Book Store

Department for Transport

What is web scraping?

"Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites."

Boeing & Waddell 2016

Two types of online data

- Structured data files ("easy" web scraping)
  - Plain-text data (e.g. csv) stored at secure or non-secure URLs
  - Data stored in a database with a well structured API (Application Programming Interface)
  - URL is all you need, multiple datasets are saved with defined URLs → looped downloads possible
- Not as well structure information ("difficult" web scraping)
  - Html tables
  - Text on websites
  - Information requiring navigation of web forms
  - Information needs to be identified: Either by html "node" (best way) or by string pattern (if nodes are not an option)

Department for Transport

When web scraping can be useful

- A lot of useful data is stored on webpages in unstructured format. We all have used the simplest form of web scraping – copy-pasting – but for more complex tasks, automation saves time and is more efficient.
- Certain data can be collected from the web and processed without spending too much resources – both time-wise and financially.

- Principles of reproducibility apply:
  - Fully document your web scraping
  - Find (inevitable) mistakes
  - Easy to re-run or update if webpage content changes
  - Can apply scraping methods to other webpages

Department for Transport

Legal implications

- Web scraping can put strain on the servers of a website and can cause issues for the provider
- Check the terms of service of a website before scraping
- Think about the use of the data (exploratory vs business)
- Some webpages will block access after a certain amount of website requests within a short amount of time (e.g. Google)

Department for Transport

How it works

- Web pages are coded in order to correctly display in browser
- Most common coding language is the “Hypertext Markup Language” (HTML)
- Text is “marked up” in html via the use of “tags” (e.g. formatted in **bold**)
- Querying the html code for those tags lets us find and extract the needed information
- Easiest way to explorer the html code of a webpage is the “Inspect” element in browsers (“right click” → “View page source”, or “Ctrl+U” in Chrome)

Tags (pink)

Tag attributes (blue)

Marked up text (black)

```
43
44
45
46      </head>
47
48      <body id="default" class="default">
49
50
51
52
53          <header class="header container-fluid">
54              <div class="page_inner">
55                  <div class="row">
56                      <div class="col-sm-8 h1"><a href="index.html">Books to Scrape</a><small> We lov
57  </div>
58
59
60                  </div>
61              </div>
62          </header>
63
64
65
66  <div class="container-fluid page">
67      <div class="page_inner">
68
69          <ul class="breadcrumb">
70              <li>
71                  <a href="index.html">Home</a>
72              </li>
73              <li class="active">All products</li>
74          </ul>
75
76              <div class="row">
```

Department for Transport

Web scraping in R with rvest

- **rvest** is a package for easy scraping (or “harvesting”) of data from webpages

- Main **rvest** functions:
  - **read_html()**: converts a website into an xml object
  - **html_node()**: extracts relevant “nodes” from an xml object, with an argument either for
    - Tags (e.g. “body”); or
    - Classes (e.g. “.header container-fluid”, the “.” signifies that the argument is a class)
  - **html_text()**: extracts the data from the selected nodes
  - **html_attrs()**: extracts the data from attributes (in case one is after the attribute and not the text)

Department for Transport

Key steps

1. Look at the HTML for the webpage you want to scrape (e.g. use Inspect Element in Chrome).
2. Request a URL with **rvest::read_html()**.
3. Extract the specific content nodes from the request with **rvest::html_nodes()**.
4. Convert the nodes to your desired R object type.
5. Clean the data.

Department for Transport

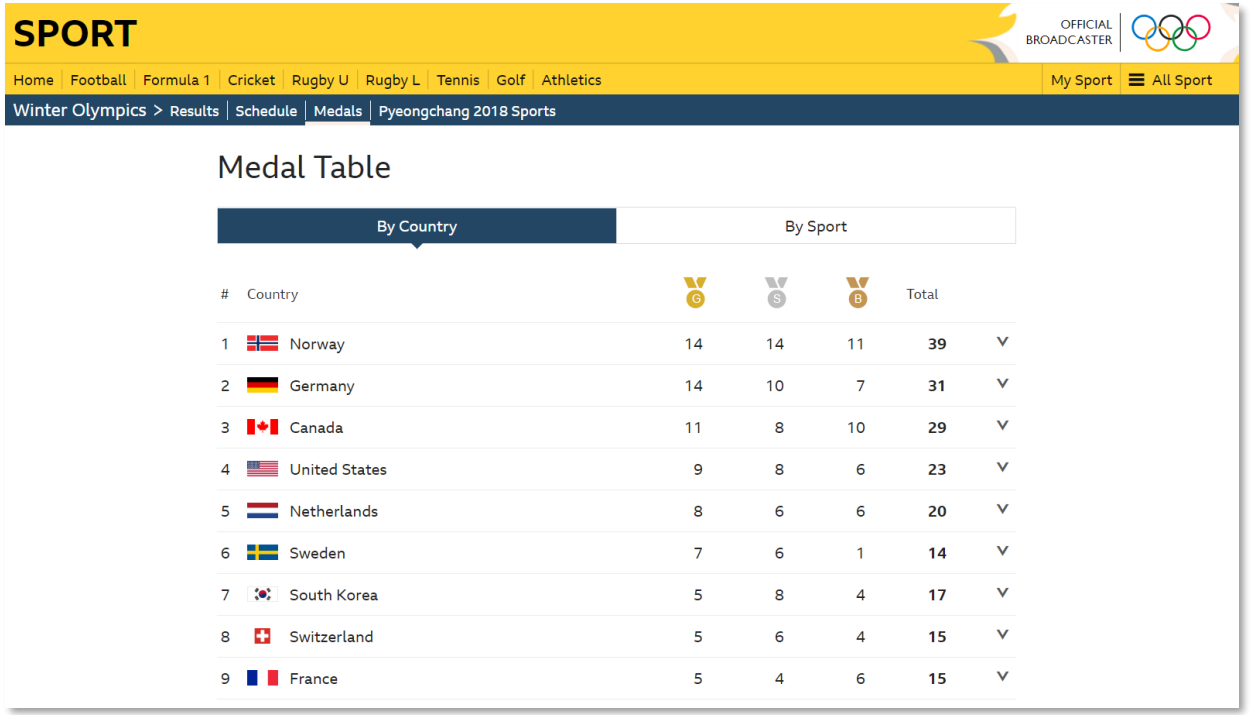
Example 1 – HTML table

https://www.bbc.co.uk/sport/winter-olympics/medals/countries

```
library(tidyverse)
library(rvest)

URL <- paste0("https://www.bbc.co.uk/sport/winter-olympics",
        "/medals/countries")

medals_table <- URL %>% read_html() %>%
                html_nodes('medals-table__table') %>%
                html_table() %>%
                as.data.frame
```


SPORT

OFFICIAL BROADCASTER

Home | Football | Formula 1 | Cricket | Rugby U | Rugby L | Tennis | Golf | Athletics | My Sport | All Sport

Winter Olympics > Results | Schedule | Medals | Pyeongchang 2018 Sports

Medal Table

By Country | By Sport

#	Country	G	S	B	Total
1	Norway	14	14	11	39
2	Germany	14	10	7	31
3	Canada	11	8	10	29
4	United States	9	8	6	23
5	Netherlands	8	6	6	20
6	Sweden	7	6	1	14
7	South Korea	5	8	4	17
8	Switzerland	5	6	4	15
9	France	5	4	6	15

Department for Transport

Example 2 – The Book Store

- Have a look at http://books.toscrape.com/ and investigate the page and the page source
- Open up the “R script - web scraping example 2” file on github if you want to run the code alongside

Department for Transport

Relevant coding areas

- Tidy data
  - Web scraped data might not come in a tidy format and might have to be converted
  - See **dplyr** package
- Character encoding
  - Some html text might be in latin1 and needs recoding to UTF-8 with the **iconv** function
- Stringr and regular expressions
  - For identifying and manipulating strings
  - See **stringr** package
- Functions
  - For structuring more complex web scraping and to keep the code modular and easy to test and improve

Department for Transport

References and further sources

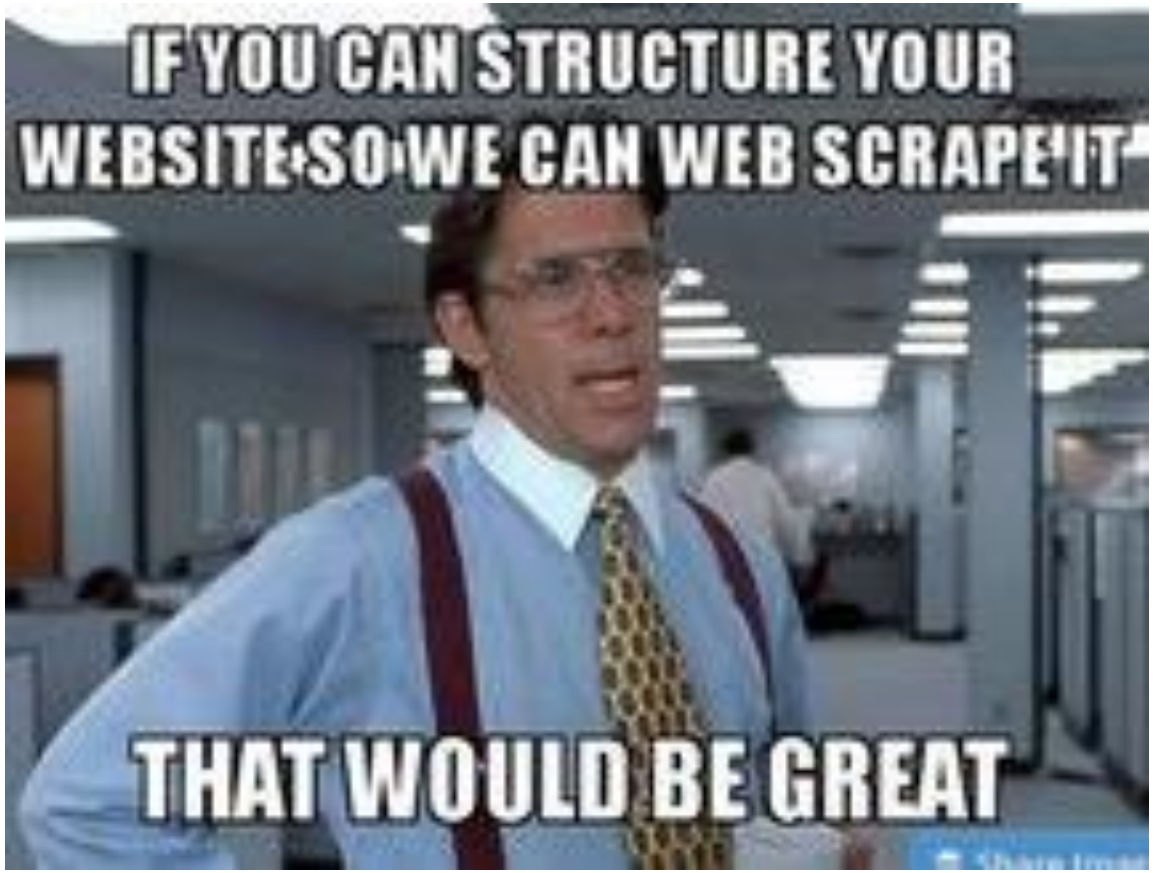
- **Rstudio blogpost on rvest package (2014)**
  - Good introduction into main functionality (how to work with html code in R)
  - Link: https://blog.rstudio.com/2014/11/24/rvest-easy-web-scraping-with-r/
- **Free data camp tutorial on web scraping (2018)**
  - Useful overview over the process of identifying the location of information and how to loop the harvest process over a collection of pages
  - Detailed usage of functions → Crucial when building one's own custom built process
  - Link: https://www.datacamp.com/community/tutorials/r-web-scraping-rvest
- **Regular expressions training materials**
  - https://stackoverflow.com/questions/4736/learning-regular-expressions/2759417#2759417
  - https://regexone.com/
- **Digital Methods Initiative at Amsterdam University (2019)**
  - Internet studies research group
  - Useful collection of web scraping tools (many work as standalone tools)
  - Link: https://wiki.digitalmethods.net/Dmi/ToolDatabase

Department for Transport

Time for Memes

Department for Transport

Time for Memes

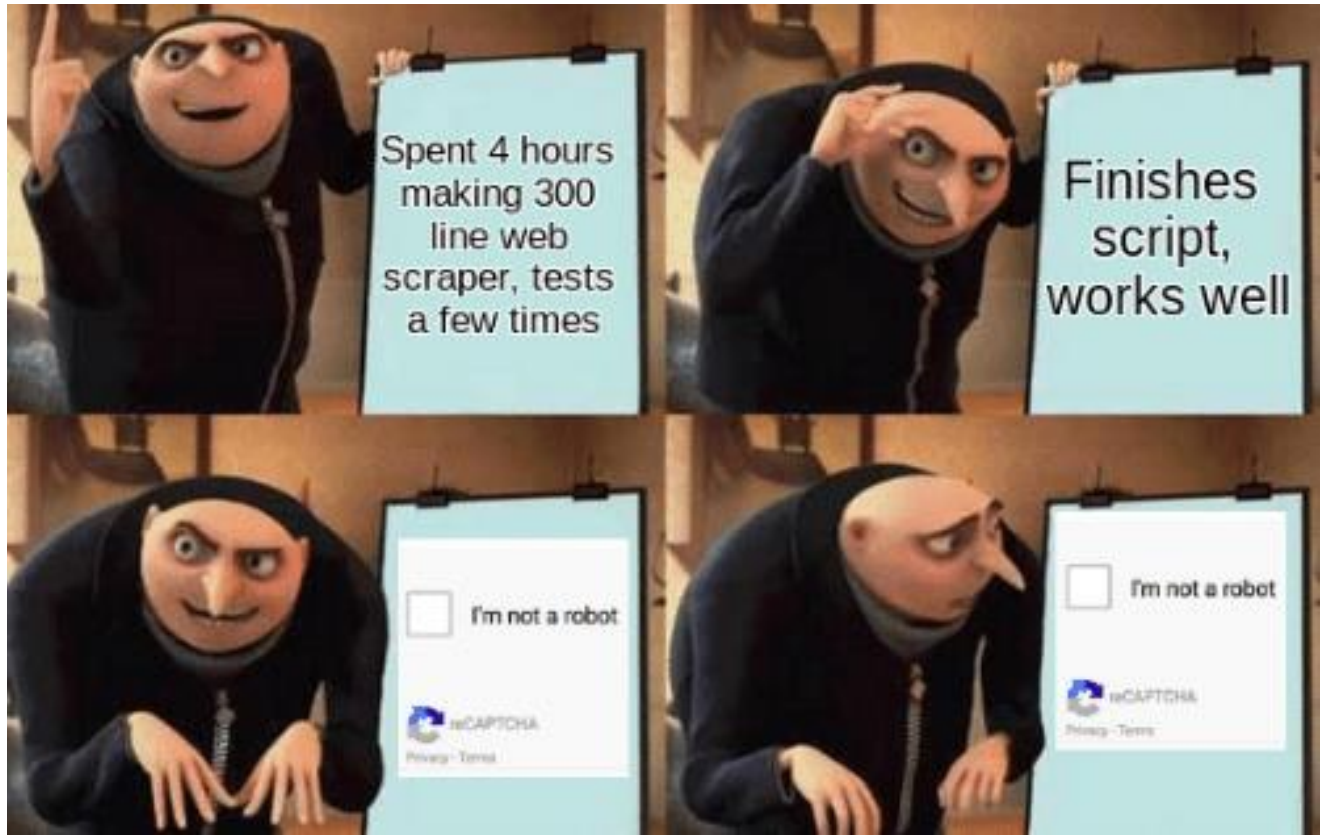


Department for Transport

Time for Memes

Department for Transport

Time for Memes



Department for Transport

Thank you