Plan for today

- Introduction
- Some key features of the internet
- HTML and CSS
- Fundamentals of web scraping
- Guided coding session
Introduction
Examples

An increasing amount of data is available on the web

- Speeches, biographical information ...
- Social media data, articles, press releases ...
- Geographic information, conflict data ...

These datasets are often provided in an unstructured format

Web scraping is the process of extracting this information automatically and transforming it into a structured dataset
Why automate?

Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets

In contrast, automated web scraping

1. Scales well for large datasets
2. Is reproducible
3. Involved adaptable techniques
4. Facilitates detecting and fixing errors

When to scrape?

1. Trade-off between your time today and your time in the future. Invest in your future self
2. Computer time is often cheap; human time more expensive
Obtaining data from the web: Two approaches

Two different approaches

1. **Screen scraping** Extract data from source code of website, with html parser and/or regular expressions
   - `rvest` package in R

2. **Web APIs** (application programming interfaces): A set of structured http requests that return JSON or XML data
   - `httr` package to construct API requests
   - Packages specific to each API: For example [WDI](https://cran.r-project.org/web/views/WorldData.html), [Rfacebook](https://www.r-library.com)
   - Check CRAN Task View on [Web Technologies and Services](https://cran.r-project.org/web/views/WebTechnologiesAndServices.html) for examples
   - More on APIs in week 5
Some key features of the internet
Client-server model
Client-server model

- Client: User computer, tablet, phone, software application, etc.
- Server: Web server, mail server, file server, Jupyter server, etc.

1. Client makes request to the server
   - Depending on what you want to get, the request might be
     - HTTP: Hypertext Transfer Protocol
     - HTTPS: Hypertext Transfer Protocol Secure
     - SMTP: Simple Mail Transfer Protocol
     - FTP: File Transfer Protocol

2. Server returns response
Request and response in the case of HTTP

From StackOverflow
Simple example: MY472 website

Let’s see a very simple example of https://lse-my472.github.io

LSE MY472 Data for Data Scientists
Course Handout web page for Michaelmas Term 2020

MY472 Data for Data Scientists
Michaelmas Term 2020

Prerequisites
All students are required to complete the preparatory course ‘R Advanced for Methodology’ early in Michaelmas Term, ideally in weeks 0 and 1. You will be auto-enrolled into the R course when enrolling into MY472 on Moodle.

Instructors
Office hour slots to be booked via LSE’s StudentHub
Simple example: MY472 website

<table>
<thead>
<tr>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request URL:</strong></td>
</tr>
<tr>
<td><strong>Request Method:</strong></td>
</tr>
<tr>
<td><strong>Status Code:</strong></td>
</tr>
<tr>
<td><strong>Remote Address:</strong></td>
</tr>
<tr>
<td><strong>Referrer Policy:</strong></td>
</tr>
</tbody>
</table>
Simple example: Request headers

- **:authority:** lse-my472.github.io
- **:method:** GET
- **:path:** /
- **:scheme:** https
- **accept:** text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
- **accept-encoding:** gzip, deflate, br
- **accept-language:** en-US,en;q=0.9,ja;q=0.8,zh-CN;q=0.7,zh-TW;q=0.6,zh;q=0.5
- **upgrade-insecure-requests:** 1
- **user-agent:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.67 Safari/537.36
Simple example: Response headers

- **Response Headers**
  - accept-ranges: bytes
  - access-control-allow-origin: *
  - age: 21
  - cache-control: max-age=600
  - content-encoding: gzip
  - content-length: 7753
  - content-type: text/html; charset= utf-8
  - date: Fri, 19 Oct 2018 12:51:30 GMT
  - etag: W/"5bc841de-5085"
  - expires: Fri, 19 Oct 2018 12:45:38 GMT
  - last-modified: Thu, 18 Oct 2018 08:18:30 GMT
  - server: GitHub.com
  - status: 200
  - strict-transport-security: max-age=31556952
  - vary: Accept-Encoding
  - via: 1.1 varnish
  - x-cache: HIT
  - x-cache-hits: 1
  - x-fastly-request-id: b4184e64b5a061bce2a6b9a85a94b41d80683e90
  - x-github-request-id: ADB4:1E3D:EE3370:1362A72:5BC9CF96
  - x-served-by: cache-lcy19238-LCY
  - x-timer: S1539953490.243899,VS0,VE1
Simple example: Response content

<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Begin Jekyll SEO tag v2.5.0 -->
    <title>lse-my472.github.io | Course handout web page for LSE MY472, Data for Data Scientists (Michaelmas Term 2018).</title>
    <meta name="generator" content="Jekyll v3.7.4" />
    <meta property="og:title" content="lse-my472.github.io" />
    <meta property="og:locale" content="en_US" />
    <meta name="description" content="Course handout web page for LSE MY472, Data for Data Scientists (Michaelmas Term 2018)." />
    <meta property="og:description" content="Course handout web page for LSE MY472, Data for Data Scientists (Michaelmas Term 2018)" />
    <link rel="canonical" href="https://lse-my472.github.io/" />
    <meta property="og:url" content="https://lse-my472.github.io/" />
    <meta property="og:site_name" content="lse-my472.github.io" />
    <!-- End Jekyll SEO tag -->
    <script type="application/ld+json">
      
      <!--
     一体 headlines: "lse-my472.github.io","type":"WebSite","url":"https://lse-my472.github.io","name":"lse-my472.github.io","description"
      -->
    </script>
  </head>
  <body>
    <div class="container-lg px-3 my-5 markdown-body">
HTML and CSS
HTML

HTML: Hypertext Markup Language

- HTML displays mostly **static** content
- Many contents of dynamic webpages cannot be found in HTML
  - Example: Google Maps
- Understanding what is static and dynamic in a webpage is a crucial first step for web scraping
Beyond plain HTML

1. **Cascading Style Sheets (CSS)**  Style sheet language which describes formatting of HTML components, useful for us because of selectors

2. **Javascript**: Adds functionalities to the websites, e.g. change content/structure after website has been loaded
A very simple HTML file

```html
<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <h1>A first heading</h1>
    <p>A first paragraph.</p>
  </body>
</html>
```

From: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro
HTML tree structure
Slightly more features

<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <h1>A first heading</h1>
    <p>A first paragraph.</p>
    <p>A second paragraph with some <b>formatted</b> text.</p>
    <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
  </body>
</html>
With some content divisions

```html
<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <div>
      <h1>Heading of the first division</h1>
      <p>A first paragraph.</p>
      <p>A second paragraph with some <b>formatted</b> text.</p>
      <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
    </div>
    <div>
      <h1>Heading of the second division</h1>
      <p>Another paragraph with some text.</p>
    </div>
  </body>
</html>
```
Adding some simple CSS (1/2)

```html
<!DOCTYPE html>
<html>
  <head>
    <!-- CSS start -->
    <style>
      p {
        color: green;
      }
    </style>
    <!-- CSS end -->

    <title>A title</title>
  </head>
  <body>
    <div>
      <h1>Heading of the first division</h1>
      <p>A first paragraph.</p>
      <p>A second paragraph with some <b>formatted</b> text.</p>
      <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
    </div>
    <div>
      <h1>Heading of the second division</h1>
      <p>Another paragraph with some text.</p>
    </div>
  </body>
</html>
```
Adding some simple CSS (2/2)

```html
<!DOCTYPE html>
<html>
    <head>
        <!-- CSS start -->
        <style>
            .text-about-web-scraping {
                color: orange;
            }
            .division-two h1 {
                color: green;
            }
        </style>
        <!-- CSS end -->
    </head>
    <body>
        <div>
            <h1>Heading of the first division</h1>
            <p>A first paragraph.</p>
            <p>A second paragraph with some <b>formatted</b> text.</p>
            <p class="text-about-web-scraping">A third paragraph now containing some text about web scraping ...</p>
        </div>
        <div class="division-two">
            <h1>Heading of the second division</h1>
            <p>Another paragraph with some text.</p>
            <p class="text-about-web-scraping">A last paragraph discussing some web scraping ...</p>
        </div>
    </body>
</html>
```
Fundamentals of web scraping
### International court

From Wikipedia, the free encyclopedia

#### List of international courts

<table>
<thead>
<tr>
<th>Name</th>
<th>Scope</th>
<th>Years active</th>
<th>Subject matter</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Court of Justice</td>
<td>Global</td>
<td>1945–present</td>
<td>General disputes</td>
</tr>
<tr>
<td>International Criminal Court</td>
<td>Global</td>
<td>2002–present</td>
<td>Criminal prosecutions</td>
</tr>
<tr>
<td>Permanent Court of International Justice</td>
<td>Global</td>
<td>1922–1946</td>
<td>General disputes</td>
</tr>
<tr>
<td>Appellate Body</td>
<td>Global</td>
<td>1995–present</td>
<td>Trade disputes within the WTO</td>
</tr>
<tr>
<td>International Tribunal for the Law of the Sea</td>
<td>Global</td>
<td>1994–present</td>
<td>Maritime disputes</td>
</tr>
<tr>
<td>African Court of Justice</td>
<td>Africa</td>
<td>2009–present</td>
<td>Interpretation of AU treaties</td>
</tr>
<tr>
<td>African Court on Human and Peoples’ Rights</td>
<td>Africa</td>
<td>2006–present</td>
<td>Human rights</td>
</tr>
<tr>
<td>COMESA Court of Justice</td>
<td>Africa</td>
<td>1998–present</td>
<td>Trade disputes within COMESA</td>
</tr>
<tr>
<td>ECOWAS Community Court of Justice</td>
<td>Africa</td>
<td>1996–present</td>
<td>Interpretation of ECOWAS treaties</td>
</tr>
<tr>
<td>East African Court of Justice</td>
<td>Africa</td>
<td>2001–present</td>
<td>Interpretation of EAC treaties</td>
</tr>
<tr>
<td>SADC Tribunal</td>
<td>Africa</td>
<td>2005–2012</td>
<td>Interpretation of SADC treaties</td>
</tr>
</tbody>
</table>
Scenario 2: Data in unstructured format

POLICE NILO GHUSS (bribe)

What will happen to this country... police mamu's govt income: 30,000 per month. Per day GHUSS income 5000 (per passport verification). Imagine they t... Read more

How to Get a Passport Verified in Ghaziabad

Corruption due to vague rules

At Chembur near Eastern Expressway traffic cop stopped me and started checking docs...all was fine but puc expired...then he pointed out film... He took... Read more

Things to Know on Traffic Offences and Respective Penalties

Bribe collected by Staff of Enrollment agency

UIDAI has to take a stand on fees to be paid to enrolment agencies for processing Adhaar
Scenario 3: Hidden behind web forms
Three main scenarios

1. Data in *table* format
   - Automatic extraction with *rvest* or select specific table with *inspect element* in browser

2. Data in *unstructured* format
   - Element identification key in this case
     - *Inspect element* in browser
     - *selectorGadget*
   - Identify the target e.g. with *CSS* (this week) or *xpath* selector (next week)
   - Automatic extraction with *rvest*

3. Data hidden *behind web forms* (next week)
   - Element identification to e.g. find text boxes, buttons, and results
   - Automation of web browser behavior with *RSelenium*
Identifying elements via CSS selector notation (1/2)

- Selecting by tag-name
  - Exemplary html code: `<h3>This is the main item</h3>`
  - Selector: h3

- Selecting by class
  - Exemplary html code: `<div class='itemdisplay'>This is the main item</div>`
  - Selector: .itemdisplay

- Selecting by id
  - Exemplary html code: `<div id='maintitle'>my main title</div>`
  - Selector: #maintitle
Identifying elements via CSS selector notation (2/2)

- Selecting by tag structure
  
  - Exemplary html code (hyperlink tag a inside div tag): `<div><a href='https://www.google.com'>Google Link</a></div>`
  
  - Selector: `div a`

- Selecting by nth child of a parent element (pseudo-class)
  
  - Exemplary html code: `<body><p>First paragraph</p><p>Second paragraph.</p></body>`
  
  - Selector of second paragraph: `body > p:nth-child(2)`

Reference and further examples: https://www.w3schools.com/cssref/css_selectors.asp
The rules of the game

1. Respect the hosting site’s wishes
   - Check if an API exists or if data are available for download
   - Respect copyright; check whether republishing is allowed or not
   - Keep in mind where data comes from and give credit
   - Some websites disallow scrapers via robots.txt file

2. Limit your bandwidth use
   - Wait some time after each hit
   - Scrape only what you need, and just once

3. When using APIs, read documentation
   - Is there a batch download option?
   - Are there any rate limits?
   - Can you share the data?
Guided coding session
Markdown files this week

- 01-selecting-elements.Rmd
- 02-scrapping-tables.Rmd