



THE LONDON SCHOOL
OF ECONOMICS AND
POLITICAL SCIENCE ■

Week 7: Scraping Webpages

MY472: Data for Data Scientists

<https://lse-my472.github.io/>

Autumn Term 2025

Ryan Hübert

Associate Professor of Methodology

Unstructured data from the web

Last lecture covered ways to get *structured* data from servers

- ▷ HTTP requests to download existing files
- ▷ APIs to return data in JSON format
- ▷ Connecting to cloud databases to download data

But a lot of data is available on the web in an *unstructured* format

- ▷ Data that is not tabular, very untidy, or fragmented across multiple sources

A very common example: data that is available on **webpages**

- ▷ Note terminology: webpage, website and (web)server

What is webscraping?

Webscraping: automated extraction of useful data from webpages

- ▷ Sometimes called **screen scraping** since it entails extracting data usually displayed on a (browser) screen

Why would you want to do this?

- ▷ Goal: transform extracted data to structured dataset
- ▷ Usually tabular, but also other possibilities

Entails: making requests for webpage data and parsing that data to extract information

The rules of the game

1. Respect the hosting site's wishes
 - ▷ Check if an API exists or if data is available for download
 - ▷ Respect copyright and ethics; what are you allowed to do?
 - ▷ 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 request ("polite delays")
 - ▷ Scrape only what you need, and just once
3. Read documentation
 - ▷ Is there a batch download option?
 - ▷ Are there any rate limits?
 - ▷ Can you share the data?

Outline

- 1 Source code for websites
- 2 Extracting data from a webpage
- 3 Webscraping in R
- 4 Webscraping ethics

Source code for websites

A server returns a combination of text and multimedia files (images, videos, etc.) that are used to display a website

Each “webpage” is typically a plain text file coded in a combination of languages: HTML, CSS and JavaScript

These plain text files contain instructions about how the webpage should look

- ▷ We refer to these files as the **source code** for the webpage

The purpose of a web browser (Chrome, Safari, Firefox, etc.) is to render source code so that a user sees something “nice”

- ▷ Can view/analyse source code in most browsers

Source code for websites

Hypertext Markup Language (HTML) is a markup language

- ▷ It defines structure and meaning of content on webpages
- ▷ It tells a browser what elements are on the page and how they're organized
- ▷ It only provides basic, static content

Cascading Style Sheets (CSS) is a style sheet language

- ▷ It describes formatting of HTML components

JavaScript is a programming language

- ▷ Adds dynamic behavior and interactivity to webpages
- ▷ Can modify content, structure, or style after page has loaded

HTML

An **HTML document** contains **elements**, which are the objects that make up a webpage

- ▷ E.g., text, images, links, scripts

Elements are written in the source code using **HTML tags**, which are pieces of markup enclosed in angle brackets and arranged in a tree-like structure

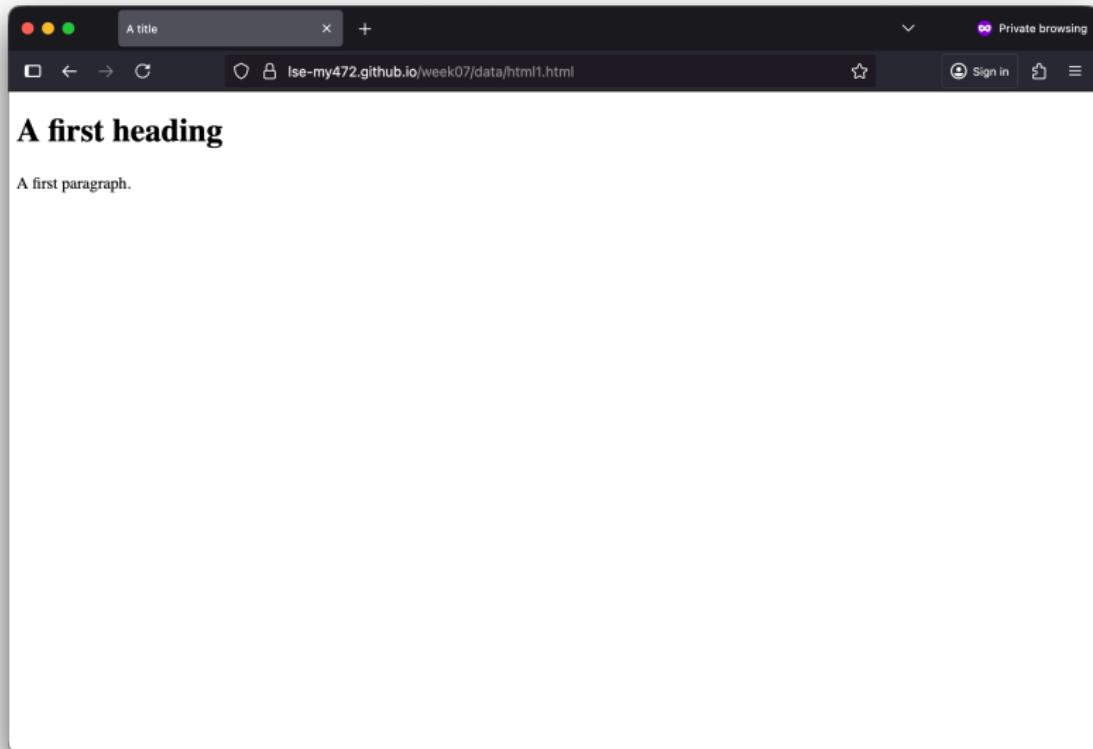
- ▷ Most elements need to be enclosed between an **opening tag** and a **closing tag**
- ▷ E.g., paragraph text like `<p>Hello world!</p>`
- ▷ **Void elements** are self-contained and do not have closing tags, such as ``, `<link>` and `
`

A very simple HTML file

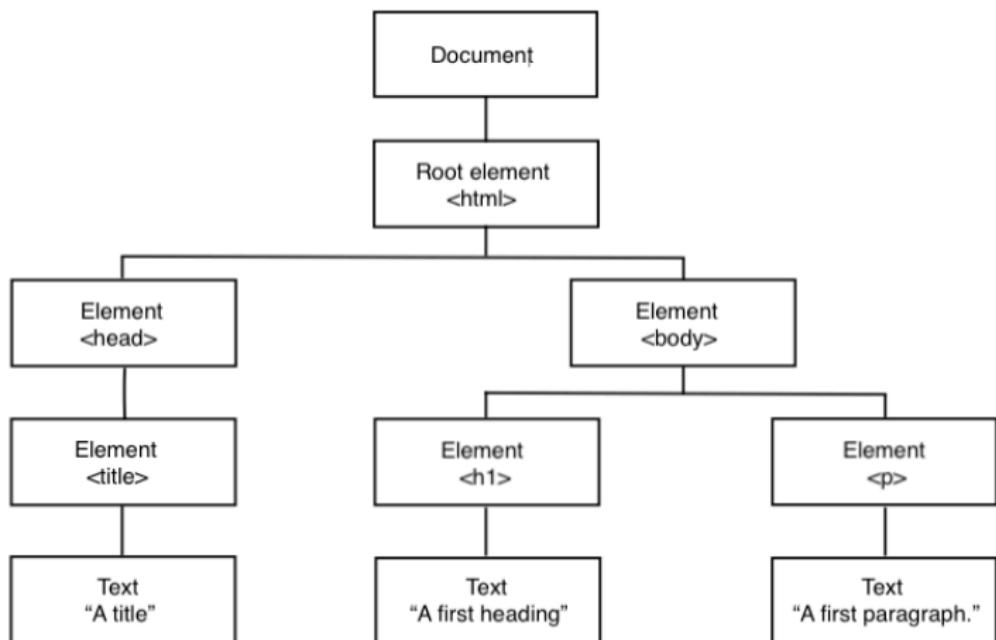
```
<!DOCTYPE html>
<html>
  <head>
    <title>A title</title>
  </head>
  <body>
    <h1>A first heading</h1>
    <p>A first paragraph.</p>
  </body>
</html>
```

Source: <https://lse-my472.github.io/week07/data/html1.html>

A very simple HTML file



HTML tree structure



HTML tree structure

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

Key vocabulary (similar to file paths):

- ▷ `<html>` is *the root element* of this HTML document
- ▷ `<head>` is *the parent element* for `<title>`
- ▷ `<h1>` is a *child element* for `<body>`
- ▷ `<head>` and `<p>` are both *descendant elements* of `<html>`

HTML attributes

HTML tags can contain **attributes** in them

- ▷ Attributes provide additional detail about how the tag should be treated by a browser

Some common attribute examples:

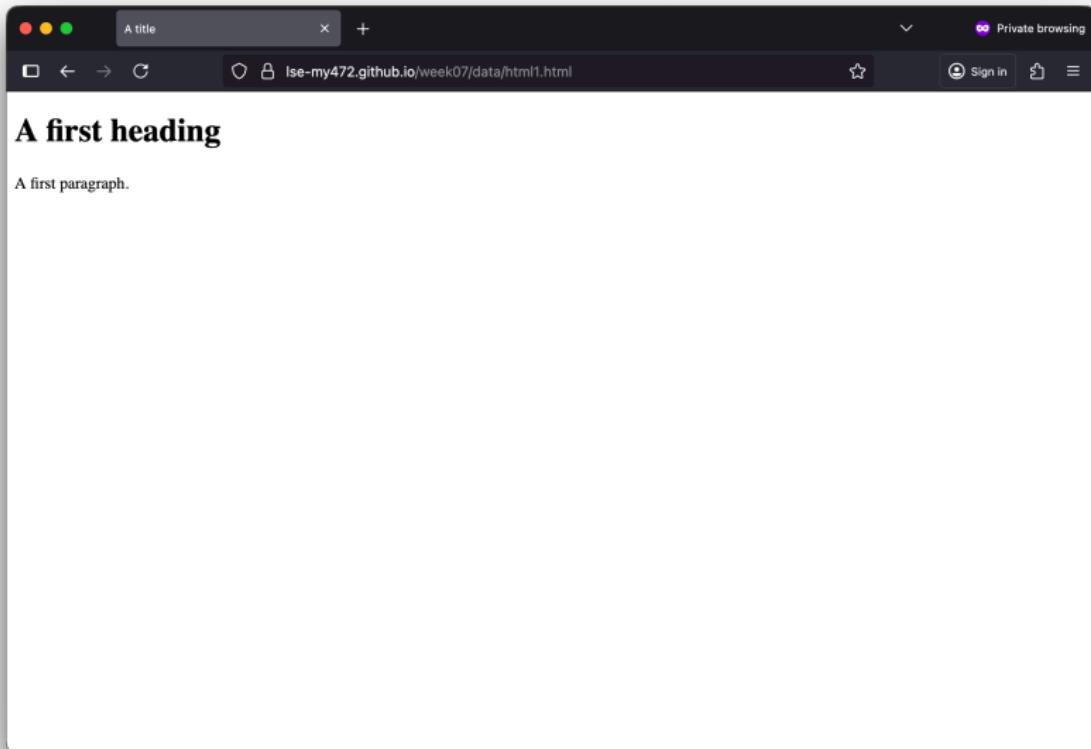
- ▷ `Link`
- ▷ ``
- ▷ `<div class="my_class">Some content here.</div>`
- ▷ `<p id="firstParagraph">My first paragraph.</p>`

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> and
      an image .</p>
  </body>
</html>
```

Source: <https://lse-my472.github.io/week07/data/html2.html>

Slightly more features



With some content divisions

```
<!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>
```

Source: <https://lse-my472.github.io/week07/data/html3.html>

With some content divisions

The screenshot shows a web browser window with a red header bar containing the LSE logo. The main content area displays two distinct sections separated by a horizontal line. The first section contains a heading and three paragraphs. The second section contains another heading and one paragraph.

A title

lse-my472.github.io/week07/data/html3.html

Private browsing

LSE

Heading of the first division

A first paragraph.

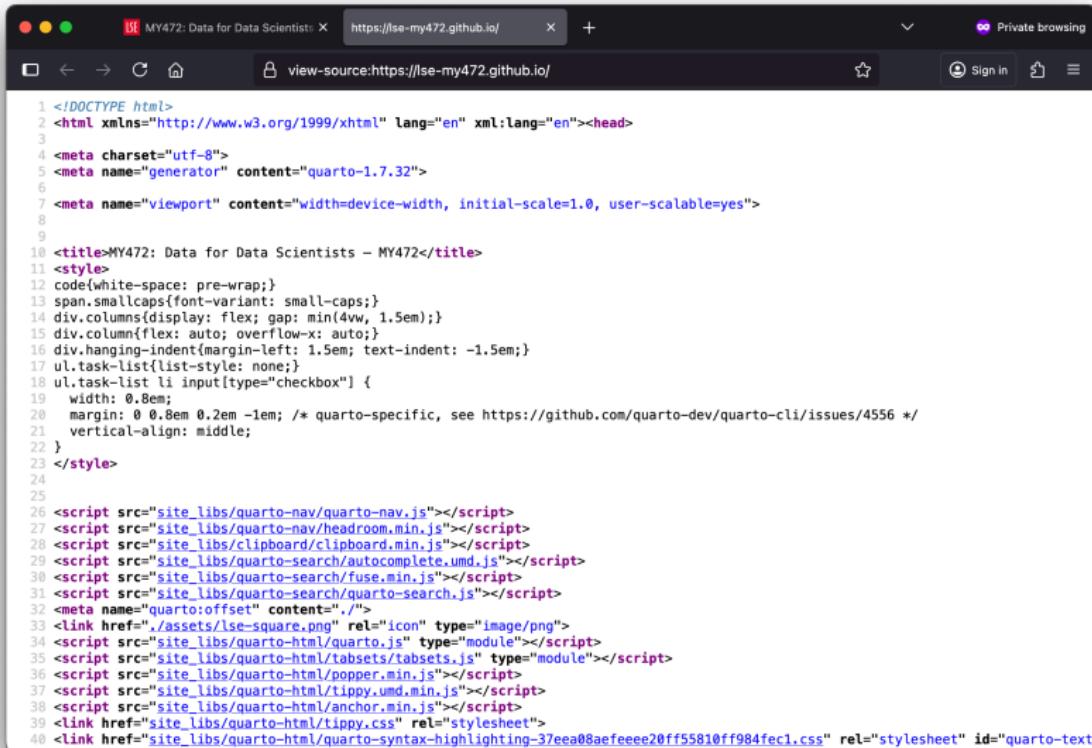
A second paragraph with some **formatted** text.

A third paragraph with a [hyperlink](#).

Heading of the second division

Another paragraph with some text.

What do CSS and JavaScript look like?



The screenshot shows a browser window with the URL <https://lse-my472.github.io/>. The title bar says "MY472: Data for Data Scientists". The browser interface includes a back button, forward button, search bar, and a "view-source" link. The main content area displays the source code of the page, which is a combination of HTML, CSS, and JavaScript.

```
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"><head>
3
4 <meta charset="utf-8">
5 <meta name="generator" content="quarto-1.7.32">
6
7 <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=yes">
8
9
10 <title>MY472: Data for Data Scientists - MY472</title>
11 <style>
12 code{white-space: pre-wrap;}
13 span.smallcaps{font-variant: small-caps;}
14 div.columns{display: flex; gap: min(4vw, 1.5em);}
15 div.column{flex: auto; overflow-x: auto;}
16 div.hanging-indent{margin-left: 1.5em; text-indent: -1.5em;}
17 ul.task-list{list-style: none;}
18 ul.task-list li input[type="checkbox"] {
19   width: 0.8em;
20   margin: 0 0.8em 0.2em -1em; /* quarto-specific, see https://github.com/quarto-dev/quarto-cli/issues/4556 */
21   vertical-align: middle;
22 }
23 </style>
24
25
26 <script src="site_libs/quarto-nav/quarto-nav.js"></script>
27 <script src="site_libs/quarto-nav/headroom.min.js"></script>
28 <script src="site_libs/clipboard/clipboard.min.js"></script>
29 <script src="site_libs/quarto-search/autocomplete.umd.js"></script>
30 <script src="site_libs/quarto-search/fuse.min.js"></script>
31 <script src="site_libs/quarto-search/quarto-search.js"></script>
32 <meta name="quarto:offset" content=".1">
33 <link href="/assets/lse-square.png" rel="icon" type="image/png">
34 <script src="site_libs/quarto-html/quarto.js" type="module"></script>
35 <script src="site_libs/quarto-html/tabssets/tabssets.js" type="module"></script>
36 <script src="site_libs/quarto-html/popper.min.js"></script>
37 <script src="site_libs/quarto-html/tippy.umd.min.js"></script>
38 <script src="site_libs/quarto-html/anchor.min.js"></script>
39 <link href="site_libs/quarto-html/tippy.css" rel="stylesheet">
40 <link href="site_libs/quarto-html/quarto-syntax-highlighting-37eea08aeffffe20ff55810ff984fec1.css" rel="stylesheet" id="quarto-text"
```

What do CSS and JavaScript look like?

The screenshot shows a code editor window with a dark theme. The title bar says "quarto-nav.js". The code is written in JavaScript and defines two functions: `headroomChanged` and `announceDismiss`. The `headroomChanged` function creates a custom event. The `announceDismiss` function removes an element by ID and sets a local storage item. The `announceRegister` function checks if an element exists, gets its ID, and checks local storage for a dismissed status. The code editor has a sidebar on the right showing other files and a status bar at the bottom.

```
const headroomChanged = new CustomEvent("quarto-headroomChanged", {  
    detail: {},  
    bubbles: true,  
    cancelable: false,  
    composed: false,  
});  
  
const announceDismiss = () => {  
    const annEl = window.document.getElementById("quarto-announcement");  
    if (annEl) {  
        annEl.remove();  
  
        const annId = annEl.getAttribute("data-announcement-id");  
        window.localStorage.setItem(`quarto-announce-${annId}`, "true");  
    }  
};  
  
const announceRegister = () => {  
    const annEl = window.document.getElementById("quarto-announcement");  
    if (annEl) {  
        const annId = annEl.getAttribute("data-announcement-id");  
        const isDismissed =  
            window.localStorage.getItem(`quarto-announce-${annId}`) || false;  
        if (isDismissed) {  
            annEl.remove();  
        }  
    }  
};
```

Line 1, Column 48 31 misspelled words Spaces: 2 JavaScript

Outline

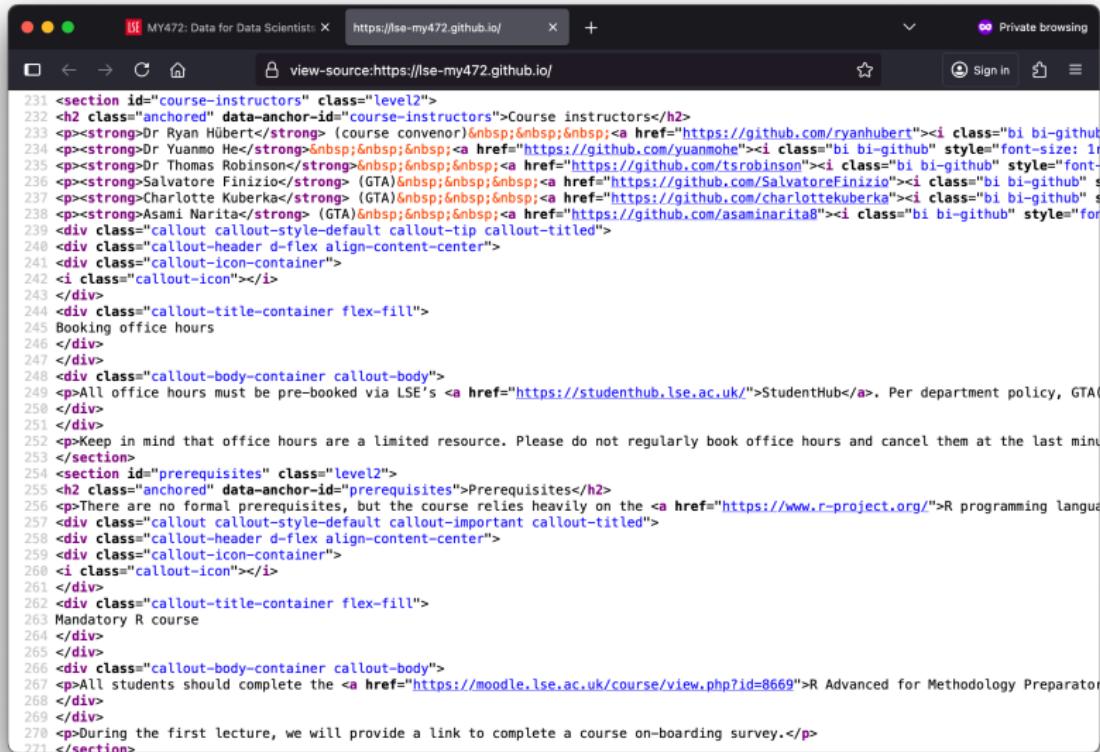
- 1 Source code for websites
- 2 Extracting data from a webpage
- 3 Webscraping in R
- 4 Webscraping ethics

What do you “scrape”?

The screenshot shows a web browser window with the following details:

- Title Bar:** MY472: Data for Data Scientists
- Address Bar:** lse-my472.github.io
- Page Header:** LSE MY472 General Information Assessments Course Topics
- Page Content:**
 - Section:** MY472: Data for Data Scientists (Autumn Term 2025)
 - Section:** Course instructors
 - Dr Ryan Hubert:** course convenor, Associate Professor, Department of Methodology
 - Dr Yuanmo He:** LSE Fellow, Department of Methodology
 - Dr Thomas Robinson:** Assistant Professor, Department of Methodology
 - Salvatore Finizio:** GTA, PhD Student, Department of Geography and Environment
 - Charlotte Kuberka:** GTA, PhD Student, Department of Government
 - Asami Narita:** GTA, PhD Student, Department of Methodology
- On this page sidebar:** Course instructors, Prerequisites, Technical requirements, Registration and auditing, Course format and scheduling, Course-related communications, Administrative support, Attribution statement

Is the desired content in the source code?



The screenshot shows a browser window with the URL <https://lse-my472.github.io/>. The page content is displayed in a monospaced font, representing the raw HTML source code. The code includes sections for course instructors, booking office hours, prerequisites, and mandatory R courses, along with various CSS classes and GitHub links.

```
231 <section id="course-instructors" class="level2">
232 <h2 class="anchored" data-anchor-id="course-instructors">Course instructors</h2>
233 <p><strong>Ryan Hubert</strong> (course convenor)<br/><a href="https://github.com/ryanhubert"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Ryan Hubert</a></p>
234 <p><strong>Dr Yuanmo He</strong><br/><a href="https://github.com/yuanmohhe"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Dr Yuanmo He</a></p>
235 <p><strong>Dr Thomas Robinson</strong><br/><a href="https://github.com/tsrobinson"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Dr Thomas Robinson</a></p>
236 <p><strong>Salvatore Finizio</strong> (GTA)<br/><a href="https://github.com/SalvatoreFinizio"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Salvatore Finizio</a></p>
237 <p><strong>Charlotte Kuberka</strong> (GTA)<br/><a href="https://github.com/charlottekuberka"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Charlotte Kuberka</a></p>
238 <p><strong>Asama Narita</strong> (GTA)<br/><a href="https://github.com/asamarita8"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Asama Narita</a></p>
239 <div class="callout callout-style-default callout-tip callout-titled">
240 <div class="callout-header d-flex align-content-center">
241 <div class="callout-icon-container">
242 <i class="callout-icon"></i>
243 </div>
244 <div class="callout-title-container flex-fill">
245 Booking office hours
246 </div>
247 </div>
248 <div class="callout-body-container callout-body">
249 <p>All office hours must be pre-booked via LSE's <a href="https://studenthub.lse.ac.uk/">StudentHub</a>. Per department policy, GTA(250
251 </div>
252 <p>Keep in mind that office hours are a limited resource. Please do not regularly book office hours and cancel them at the last minute(253
254 <section id="prerequisites" class="level2">
255 <h2 class="anchored" data-anchor-id="prerequisites">Prerequisites</h2>
256 <p>There are no formal prerequisites, but the course relies heavily on the <a href="https://www.r-project.org/">R programming language(257
257 <div class="callout callout-style-default callout-important callout-titled">
258 <div class="callout-header d-flex align-content-center">
259 <div class="callout-icon-container">
260 <i class="callout-icon"></i>
261 </div>
262 <div class="callout-title-container flex-fill">
263 Mandatory R course
264 </div>
265 </div>
266 <div class="callout-body-container callout-body">
267 <p>All students should complete the <a href="https://moodle.lse.ac.uk/course/view.php?id=8669">R Advanced for Methodology Preparation(268
268 </div>
269 </div>
270 <p>During the first lecture, we will provide a link to complete a course on-boarding survey.</p>
271 </section>
```

Three different scenarios

Static websites

- ▷ Server sends “fixed” HTML files
- ▷ Client renders the HTML file in a browser

Dynamic websites

- ▷ Server or client (via JavaScript) builds the HTML on the fly
- ▷ Client browser still ends up receiving and displaying HTML
- ▷ Enables integration of real-time data (from a database or API)

Single-page apps (or SPAs)

- ▷ Client loads a minimal HTML skeleton
- ▷ Embedded JavaScript injects more HTML dynamically

Three different scenarios

The screenshot shows a web browser window with the following details:

- Title Bar:** MY472: Data for Data Scientists
- Address Bar:** lse-my472.github.io
- Header:** LSE MY472 General Information Assessments Course Topics
- Content Area:**
 - Section Header:** MY472: Data for Data Scientists (Autumn Term 2025)
 - Section:** Course instructors
 - Dr Ryan Hubert (course convenor):** Associate Professor, Department of Methodology
 - Dr Yuanmo He:** LSE Fellow, Department of Methodology
 - Dr Thomas Robinson:** Assistant Professor, Department of Methodology
 - Salvatore Finizio (GTA):** PhD Student, Department of Geography and Environment
 - Charlotte Kuberka (GTA):** PhD Student, Department of Government
 - Asami Narita (GTA):** PhD Student, Department of Methodology
- Right Sidebar:** On this page (Course instructors, Prerequisites, Technical requirements, Registration and auditing, Course format and scheduling, Course-related communications, Administrative support, Attribution statement)

Three different scenarios

The New York Times - Breaking + Private browsing

www.nytimes.com

Sunday, November 9, 2025 Today's Paper

U.S. INTERNATIONAL CANADA ESPAÑOL 中文 SUBSCRIBE FOR £0.50/WEEK LOG IN

The New York Times

Nasdaq -0.21% +

U.S. World Business Arts Lifestyle Opinion Video Audio Games Cooking Wirecutter The Athletic

Families in Limbo After Supreme Court Order Interrupts Food Stamp Payments

The fate of SNAP is again in question after the Supreme Court temporarily agreed to allow the Trump administration to withhold full aid under the program.

5 MIN READ



Marco Postigo/Stonel for The New York Times

A Timeline of the Legal Saga Surrounding SNAP Payments

Weeks of uncertainty during the longest government shutdown in American history have left some states struggling to issue payments to food stamp recipients.

4 MIN READ

Down to \$1.18: How Families Are Coping With Worries Mount as Air Traffic Delays Stretch to

6,907 Miles in 165 Days: How 2 Women Made History by Rowing the Pacific Ocean

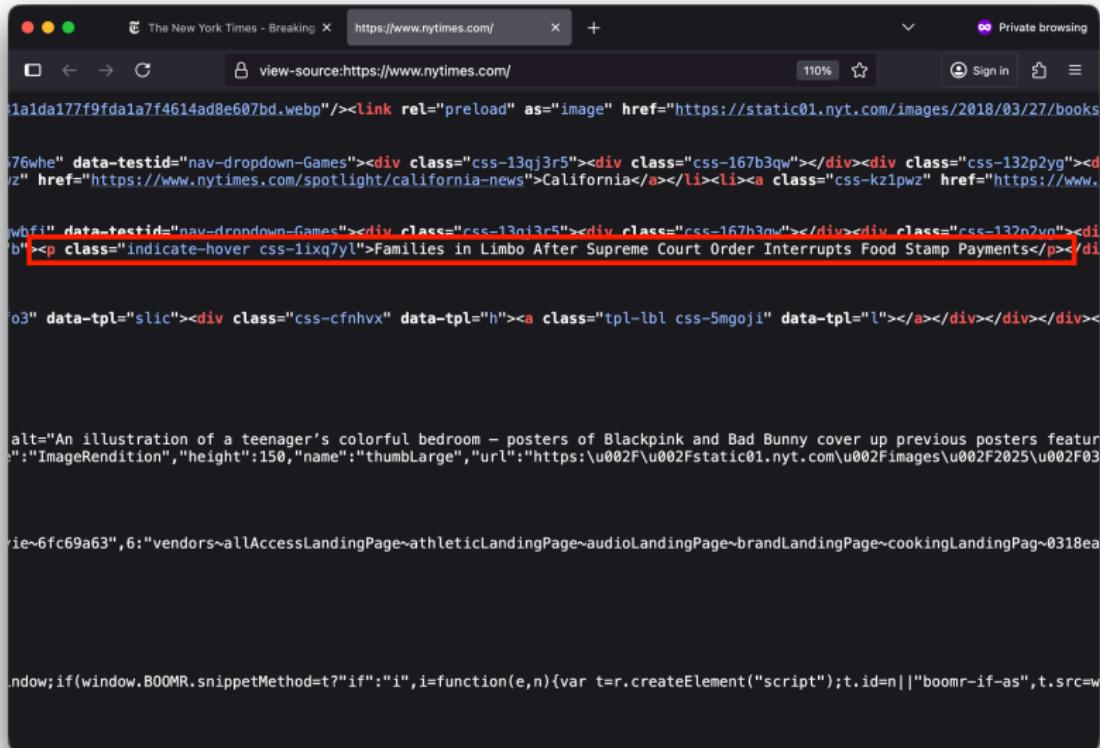
Brian Cassey/Agence France-Presse — Getty Images

For Jess Rowe and Miriam Payne, the hardest task of all during six months at sea was consuming 5,000 calories each a day.

FROM THE ATHLETIC



Three different scenarios

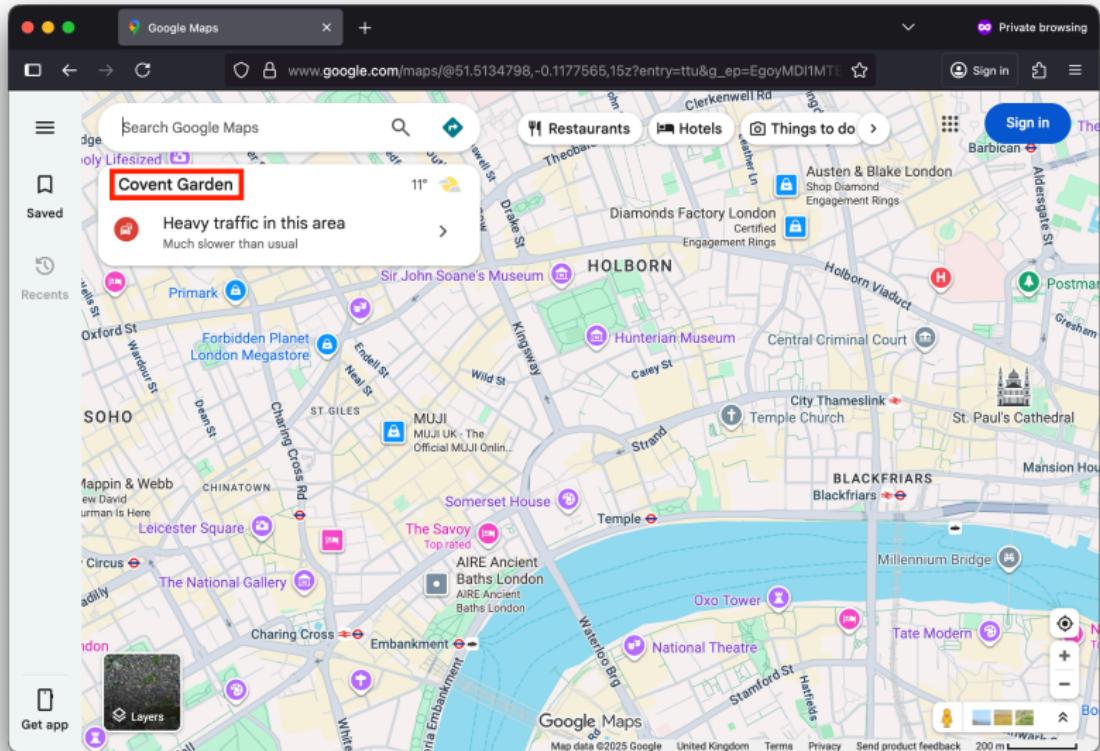


The screenshot shows a browser window with the URL <https://www.nytimes.com/>. The page content is visible at the top, and the source code is displayed below it. A specific line of code is highlighted with a red box:

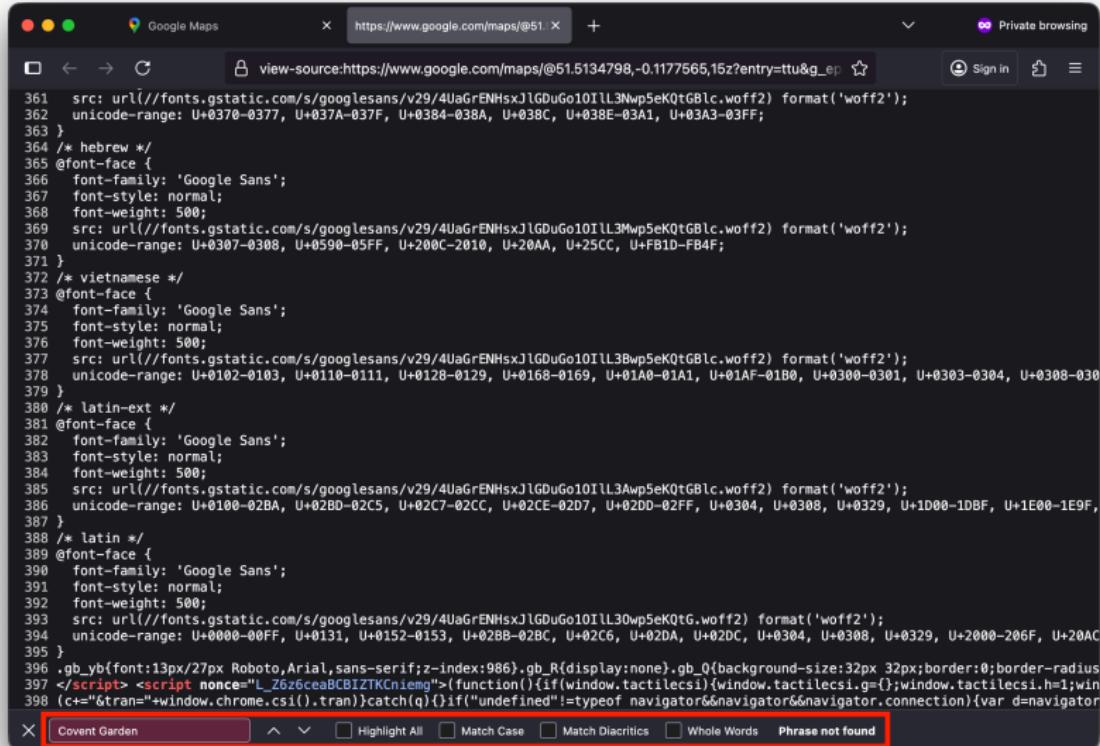
```
<p class="indicate-hover css-1ixq7yl">Families in Limbo After Supreme Court Order Interrupts Food Stamp Payments</p></div></div></div>
```

The source code also includes several CSS classes and IDs, such as `css-13qj3r5`, `css-167b3qw`, `css-132p2yg`, `css-13n2vn`, and `css-cfnhv`.

Three different scenarios



Three different scenarios



Google Maps https://www.google.com/maps/@51.5134798,-0.1177565,15z?entry=ttu&g_e_p Private browsing

```
361 src: url("//fonts.gstatic.com/s/googlesans/v29/4UaGrENHsxJlGDuGo10I1l3Nwp5eKQtGBlc.woff2) format('woff2');  
362 unicode-range: U+0370-0377, U+037A-037F, U+0384-038A, U+038C, U+038E-03A1, U+03A3-03FF;  
363 }  
364 /* hebrew */  
365 @font-face {  
366   font-family: 'Google Sans';  
367   font-style: normal;  
368   font-weight: 500;  
369   src: url("//fonts.gstatic.com/s/googlesans/v29/4UaGrENHsxJlGDuGo10I1l3Mwp5eKQtGBlc.woff2) format('woff2');  
370   unicode-range: U+0307-0308, U+0590-05FF, U+200C-2010, U+20AA, U+25CC, U+FB1D-FB4F;  
371 }  
372 /* vietnamese */  
373 @font-face {  
374   font-family: 'Google Sans';  
375   font-style: normal;  
376   font-weight: 500;  
377   src: url("//fonts.gstatic.com/s/googlesans/v29/4UaGrENHsxJlGDuGo10I1l3Bwp5eKQtGBlc.woff2) format('woff2');  
378   unicode-range: U+0102-0103, U+0110-0111, U+0128-0129, U+0168-0169, U+01A0-01A1, U+01AF-01B0, U+0300-0301, U+0303-0304, U+0308-030  
379 }  
380 /* latin-ext */  
381 @font-face {  
382   font-family: 'Google Sans';  
383   font-style: normal;  
384   font-weight: 500;  
385   src: url("//fonts.gstatic.com/s/googlesans/v29/4UaGrENHsxJlGDuGo10I1l3Awp5eKQtGBlc.woff2) format('woff2');  
386   unicode-range: U+0100-02BA, U+02B0-02C5, U+02C7-02CC, U+02CE-02D7, U+02DD-02FF, U+0304, U+0308, U+0329, U+1D00-1DBF, U+1E00-1E9F,  
387 }  
388 /* latin */  
389 @font-face {  
390   font-family: 'Google Sans';  
391   font-style: normal;  
392   font-weight: 500;  
393   src: url("//fonts.gstatic.com/s/googlesans/v29/4UaGrENHsxJlGDuGo10I1l30wp5eKQtG.woff2) format('woff2');  
394   unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02BB-02BC, U+02C6, U+02DA, U+02DC, U+0304, U+0308, U+0329, U+2000-206F, U+20AC  
395 }  
396 .gb_yb{font:13px/27px Roboto,Arial,sans-serif;z-index:986}.gb_R{display:none}.gb_Q{background-size:32px 32px;border:0; border-radius:  
397 </script><script nonce="L_2626ceabCBIZTKniemg">(function(){if(window.tactilecsi){window.tactilecsi.g={};window.tactilecsi.h=1;win  
398 (c+="&tran=")+window.chrome.csil.tran}catch(q){}if("undefined"!=typeof navigator&&navigator&&navigator.connection){var d=navigator
```

X Covent Garden ⌂ ⌂ Highlight All ⌂ Match Case ⌂ Match Diacritics ⌂ Whole Words ⌂ Phrase not found

Query languages for webscraping

Typical use case for webscraping: get information from webpages

- ▷ Requires that you be able to find and extract relevant info
- ▷ Ideally: use structure of HTML to identify key elements

There are two main **query languages** used to identify elements

- ▷ **CSS selectors** (read more [here](#))
- ▷ **XPath** (or **XML Path Language**, read more [here](#))

Analogy between regex and query languages

For most things, you can use either CSS selectors or XPath

- ▷ But XPath is (probably) more flexible/powerful

Identifying elements by tag

```
<h3>This is the main item</h3>
```

These get this element:

- ▷ CSS Selector: `h3`
- ▷ XPath: `//h3`

This is most useful if there is only *one* `<h3>` tag

XPath resembles file paths, but note:

- ▷ An **absolute XPath** begins with one slash, e.g., `/html`
- ▷ A **relative XPath** begins with two slashes, e.g. `//h3`

Identifying elements using tree structure

```
<div>
  <p><a href="https://www.google.com">Google</a></p>
</div>
```

These get the paragraph element:

- ▷ CSS Selector: `div > p`
- ▷ XPath: `//div/p`

These get the link element:

- ▷ CSS Selector: `div a`
- ▷ XPath: `//div//a`

Key distinction: extracting *direct* child versus *any* child—so, neither `div > a` nor `//div/a` will get the link!

Identifying elements by an index

```
<body>
  <h2>Title</h2>
  <p>First paragraph.</p>
  <p>Second paragraph.</p>
</body>
```

These get the *second* paragraph element:

- ▷ CSS Selector: `body > p:nth-of-type(2)`
- ▷ XPath: `//body/p[2]`

These get the *first* paragraph element:

- ▷ CSS Selector: `body > p:nth-child(2)`
- ▷ XPath: `//body/*[2]`

Identifying elements by an attribute

```
<div id="maintitle">My main title</div>
<div class="itemdisplay">This is the main item.</div>
<p class="itemdisplay">This is a subsidiary item.</p>
```

These get the first `div` element:

- ▷ CSS Selector: `div#maintitle` or `#maintitle`
- ▷ XPath: `//div[@id='maintitle']` or
`//*[@id='maintitle']`

Identifying elements by class or id attributes

```
<div id="maintitle">My main title</div>
<div class="itemdisplay">This is the main item.</div>
<p class="itemdisplay">This is a subsidiary item.</p>
```

These get the second div element:

- ▷ CSS Selector: `div.itemdisplay`
- ▷ XPath: `//div[@class='itemdisplay']`

These get both the second div and the paragraph elements:

- ▷ CSS Selector: `.itemdisplay`
- ▷ XPath: `//*[@class='itemdisplay']`

Selecting tags by other attributes

```
<link href = "css1.css" id = "some_id">  
<link href = "css2.css">
```

These get both links:

- ▷ CSS Selector: `link` or `link[href]`
- ▷ XPath: `//link` or `//link[@href]`

These get the first link:

- ▷ CSS Selector: `link[id]` or `link[id="some_id"]` or
`link[href="css1.css"]` or `link[id*="some"]`
- ▷ XPath: `//link[@id]` or `//link[@id="some_id"]` or
`//link[@href="css1.css"]` or `//link[contains(@id, 'some')]`

Selecting tags by displayed text

```
<div>
  <h1>Heading of the first division</h1>
</div>
<div>
  <h1>Heading of the second division</h1>
</div>
```

These get the first `h1` element:

- ▷ CSS Selector: `h1:contains("Heading of the first division")`
- ▷ XPath: `//h1[text()="Heading of the first division"]`

Figuring out your query by inspecting elements

The screenshot shows a web browser window displaying the LSE MY472 course page. The title of the page is "MY472: Data for Data Scientists". The browser's developer tools are open, specifically the "Inspector" tab, which is focused on the `<h1>` element. The element's bounding box is highlighted, and its dimensions are listed as 691 x 38.6667. The right side of the developer tools interface shows the element's properties, including its class and style definitions. The "Computed" tab is selected, showing the final styles applied to the element.

```
</li>
  > <li> ... </li>
  > <li> ... </li>
  > <li> ... </li>
</ul>
</nav>
</div>
<!--main-->
<main id="quarto-document-content" class="content">
  <header id="title-block-header" class="quarto-title-block default">
    <div class="quarto-title">
      <h1 class="title">MY472: Data for Data Scientists</h1>
      <p class="subtitle lead">Autumn Term 2025</p>
    </div>
    <div class="quarto-title-meta">...</div>
  </header>
  <section id="course-instructors" class="level1">
    <h2>Course instructors</h2>
    <div>
      <h3>Dr Ryan Hubert (course convenor)</h3>
      <img alt="ORCID icon" data-bbox="398 481 413 496"/>
      <img alt="Email icon" data-bbox="418 481 433 496"/>
      <img alt="ORCID icon" data-bbox="438 481 453 496"/>
      <p>Associate Professor, Department of Methodology</p>
    </div>
  </section>
</main>
```

On this page

- Course instructors
- Prerequisites
- Technical requirements
- Registration and auditing
- Course format and scheduling
- Course-related communications
- Administrative support

Outline

- 1 Source code for websites
- 2 Extracting data from a webpage
- 3 Webscraping in R
- 4 Webscraping ethics

Webscraping in R

Webscraping in R entails:

1. Making requests for webpages
 - ▷ Simple `GET` requests for webpages: `{httr}` package
 - ▷ Automated browser requests: `{RSelenium}` package
2. Parsing html using `{rvest}` package
 - ▷ Convert raw HTML to tree object using `read_html()`
 - ▷ Find elements using `html_elements()`
 - ▷ Extract text/tables: `html_text()`, `html_tables()`

Note: `{httr}` and `{rvest}` are tightly integrated

- ▷ If you supply a URL as the argument for `read_html()` it will make `GET` request then convert raw HTML from response

Two webscraping approaches

Simple **GET** requests work for:

- ▷ Static HTML pages
- ▷ Dynamic HTML pages that are rendered server-side
- ▷ Pages requiring “simple” interaction

Automated browser is needed for:

- ▷ Pages with client-side rendering (e.g., JavaScript)
- ▷ Pages with infinite scroll
- ▷ Pages requiring user login or script-based interaction
- ▷ Sites with “malfunctioning” bot detection

A simple example

The screenshot shows a web browser window with the following details:

- Title Bar:** A title bar with three colored window control buttons (red, yellow, green) on the left, a tab labeled "A title" in the center, and a "Private browsing" indicator on the right.
- Address Bar:** An address bar containing the URL "lse-my472.github.io/week07/data/css2.html".
- Content Area:** The main content area displays the following elements:
 - A red square logo with the letters "LSE" in white.
 - Heading of the first division:** A bold black heading.
 - A first paragraph: "A first paragraph."
 - A second paragraph with some **formatted text**: "A second paragraph with some **formatted text**."
 - A third paragraph now containing some text about web scraping ...": "A third paragraph now containing some text about web scraping ..."
 - Heading of the second division:** A bold green heading.
 - Another paragraph with some text: "Another paragraph with some text."
 - A last paragraph discussing some web scraping ...": "A last paragraph discussing some web scraping ..."

A simple example

```
library("rvest")
url <- "https://lse-my472.github.io/week07/data/css2.html"

html <- read_html(url)

{html_document}
<html>
[1] <head>\n<meta http-equiv="Content-Type" content="text/h ...
[2] <body>\n          <div>\n               p:nth-of-type(3)") |>
  html_text()

[1] "A third paragraph now containing some text about web s ..."
```

A simple example

```
library("rvest")
url <- "https://lse-my472.github.io/week07/data/css2.html"

html <- read_html(url)

{html_document}
<html>
[1] <head>\n<meta http-equiv="Content-Type" content="text/h ...
[2] <body>\n          <div>\n                  
  html_text()

[1] "A third paragraph now containing some text about web s ..."
```

Extracting tables from webpages

The screenshot shows a web browser window with the title "Assessments and marking - MY" and the URL "lse-my472.github.io/assessments.html". The page content is as follows:

LSE MY472 General Information Assessments Course Topics

Assessments and marking

Assessment schedule

There are three assessments in this course.

Type	Due date
1 Formative in-class exercises	Weekly
2 Formative data science project (part 1)	24 November 2025, noon London time
Formative data science project (part 2)	19 December 2025, noon London time
3 Summative in-person practical test (100%)	11 December 2025

Additional details for each assessment will be sent via course announcements.

Disability accommodations

If you have an official disability accommodation related to exams, please contact the

On this page

- [Assessment schedule](#)
- Disability accommodations
- Extension requests
- Marking
- Academic integrity
- Generative AI

Extracting tables from webpages

```
# This will GET and parse HTML in one step
t <- read_html("https://lse-my472.github.io/assessments.html")
Sys.sleep(0.5) # a polite sleep
print(t)

{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 ...
[2] <body class="nav-fixed quarto-light">\n\n<div id="qua ...
```

Extracting tables from webpages

The `html_table()` function returns a list of tables found in the HTML based on HTML table syntax ([read more about this syntax](#))

```
tables <- html_table(t)
length(tables)      # only 1 table
```

```
[1] 1
```

```
af <- tables[[1]] # assessments info
print(af)
```

```
# A tibble: 4 x 3
  `` Type          `Due date` 
  <int> <chr>        <chr>    
1     1 Formative in-class exercises Weekly  
2     2 Formative data science project (part 1) 24 Novemb~ 
3     NA Formative data science project (part 2) 19 Decemb~ 
4     3 Summative in-person practical test (100%) 11 Decemb~
```

Extracting unstructured data from webpages

The screenshot shows a browser window with the URL `lse-my472.github.io`. A dashed blue box highlights the sidebar area on the left, which contains the title "Course instructors" and a list of three instructors with their contact information. To the right of the sidebar, the main content area displays a sidebar menu titled "On this page" with several links: "Course instructors", "Prerequisites", "Technical requirements", "Registration and auditing", "Course format and scheduling", "Course-related communications", "Administrative support", and "Attribution statement". At the bottom of the browser window, the developer tools are open, showing the "Inspector" tab with the DOM tree and the "Layout" tab of the "Elements" panel.

Course instructors

Dr Ryan Hübert (course convener)
Associate Professor, Department of Methodology

Dr Yuanmo He
LSE Fellow, Department of Methodology

Dr Thomas Robinson
Assistant Professor, Department of Methodology

Salvatore Finizio (GTA)

On this page

- Course instructors
- Prerequisites
- Technical requirements
- Registration and auditing
- Course format and scheduling
- Course-related communications
- Administrative support
- Attribution statement

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application ...

Search HTML

```
<div id="quarto-margin-sidebar" class="sidebar margin-sidebar sidebar-unpinned" style="top: 0px; max-height: 100vh;">...</div>
<!-main-->
<main id="quarto-document-content" class="content">
  <header id="title-block-header" class="quarto-title-block default">...</header>
  <section id="course-instructors" class="level2">
    <h2 class="anchored" data-anchor-id="course-instructors">...</h2>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <p>...</p>
    <div class="callout callout-style-default callout-tip callout-titled">...</div>
    <p>...</p>
```

Filter Styles

element `:hover .cls`

`:be99ccc48519b6ed9ce.min.css:5`

`*, ::before, ::after { box-sizing: border-box; }`

Inherited from body

`:be99ccc48519b6ed9ce.min.css:5`

`body { font-family: var(--bs-body-font-family, "Source Sans Pro", sans-serif); font-size: var(--bs-body-font-size, 16px); font-weight: var(--bs-body-font-weight, 400); line-height: var(--bs-body-line-height, 1.5); }`

Layout

Select a Flex container or item to continue.

Grid

Overlay Grid

`div@quarto-content.quarto-container.page-columns.page-rows-contents.page-layout-article.page-navbar`

`div.quarto-title-meta`

Grid Display Settings

Display line numbers

Display area names

Extend lines infinitely

her.page... > main#quarto-document-content.content > section#course-instructors.level2 >

Extracting unstructured data from webpages

```
ins <- read_html("https://lse-my472.github.io") |>  
  html_elements(css = "section#course-instructors")  
print(ins)
```



```
{xml_nodeset (1)}  
[1] <section id="course-instructors" class="level2"><h2 c ...
```

Extracting unstructured data from webpages

The screenshot shows a web browser window displaying a page titled "Course instructors". The page lists three course instructors: Dr Ryan Hubert (course convener), Dr Yuanmo He, and Dr Thomas Robinson. Each instructor has a profile picture, a name, their title, and a link to their GitHub repository. To the right of the page, there is a sidebar with links to "On this page" sections such as "Prerequisites", "Technical requirements", and "Attribution statement". Below the page content, the browser's developer tools are open, specifically the "Inspector" tab. The "Elements" panel shows the HTML structure of the page, including the sidebar and main content area. The "Style Editor" panel is visible at the bottom, showing CSS rules for various elements like ".quarto-content" and ".quarto-title-meta". The "Layout" panel is also visible.

Course instructors

Dr Ryan Hubert (course convener)
Associate Professor, Department of Methodology

Dr Yuanmo He
LSE Fellow, Department of Methodology

Dr Thomas Robinson
Assistant Professor, Department of Methodology

Salvatore Finizio (GTA)

On this page

Course instructors

Prerequisites

Technical requirements

Registration and auditing

Course format and scheduling

Course-related communications

Administrative support

Attribution statement

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application ...

Search HTML

```
<!--sidebar-->
<!--margin-sidebar-->
<div id="quarto-margin-sidebar" class="sidebar margin-sidebar sidebar-unpinned" style="top: 0px; max-height: 100vh;">...</div>
<!--main-->
<main id="quarto-document-content" class="content">
  <header id="title-block-header" class="quarto-title-block default">...</header>
  <section id="course-instructors" class="level2">
    <h2 class="anchored" data-anchor-id="course-instructors">...</h2>
    <p>
      <strong>Dr Ryan Hubert</strong>
      (course convener)
      <a href="https://github.com/ryanhubert" data-original-href="https://github.com/ryanhubert">...</a>
      (whitespace)
      <a href="mailto:R.Hubert@lse.ac.uk" data-original-...</a>
    </p>
  </section>
</main>
```

Filter Styles

:hov .cls + * ()

element {

.0e99ccc48519b6ed9ce.min.css:5

p {

margin-top: 0;

margin-bottom: 1rem;

.0e99ccc48519b6ed9ce.min.css:5

*::before, *::after {

box-sizing: border-box;

Inherited from body

.0e99ccc48519b6ed9ce.min.css:5

body {

font-family: var(--bs-body-font-family, -apple-system, BlinkMacSystemFont, "Segoe UI", "Helvetica Neue", Arial, sans-serif);

Layout Computed Changes Compatibility

Select a Flex container or item to continue.

Grid

Overlay Grid

div#quarto-content.quarto-container.page-columns.page-rows-contents.page-layout-article.page-navbar

div.quarto-title-meta

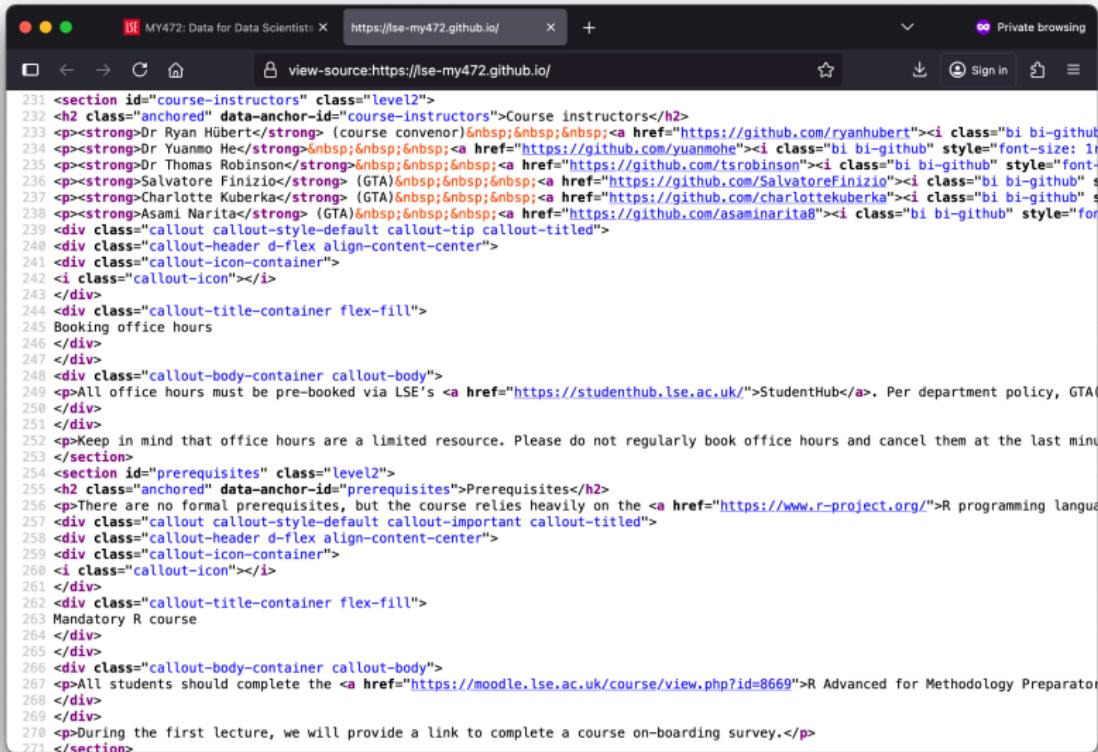
Grid Display Settings

Display line numbers

Display area names

Extend lines infinitely

Extracting unstructured data from webpages



The screenshot shows a browser window with the URL <https://lse-my472.github.io/>. The page content is displayed in a code editor-like interface, showing the raw HTML source code. The code is annotated with line numbers (231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271) and various CSS classes and GitHub links.

```
231 <section id="course-instructors" class="level2">
232   <h2 class="anchored" data-anchor-id="course-instructors">Course instructors</h2>
233   <p><strong>Dr Ryan Hubert</strong> (course convenor)<br/><a href="https://github.com/ryanhubert"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Dr Ryan Hubert</a></p>
234   <p><strong>Dr Yuanmo He</strong><br/><a href="https://github.com/yuanmohhe"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Dr Yuanmo He</a></p>
235   <p><strong>Dr Thomas Robinson</strong><br/><a href="https://github.com/tsrobinson"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Dr Thomas Robinson</a></p>
236   <p><strong>Salvatore Finizio</strong> (GTA)<br/><a href="https://github.com/SalvatoreFinizio"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Salvatore Finizio</a></p>
237   <p><strong>Charlotte Kuberka</strong> (GTA)<br/><a href="https://github.com/charlottekuberka"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Charlotte Kuberka</a></p>
238   <p><strong>Asami Narita</strong> (GTA)<br/><a href="https://github.com/asaminarita8"><i class="bi bi-github" style="font-size: 1em; vertical-align: middle; margin-right: 0.1em;"></i> Asami Narita</a></p>
239   <div class="callout callout-style-default callout-tip callout-titled">
240     <div class="callout-header d-flex align-content-center">
241       <div class="callout-icon-container">
242         <i class="callout-icon"></i>
243       </div>
244     <div class="callout-title-container flex-fill">
245       Booking office hours
246     </div>
247   </div>
248   <div class="callout-body-container callout-body">
249     <p>All office hours must be pre-booked via LSE's <a href="https://studenthub.lse.ac.uk/">StudentHub</a>. Per department policy, GTA(250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
627
628
629
629
630
631
632
633
634
635
636
637
637
638
639
639
640
641
642
643
644
645
645
646
647
647
648
649
649
650
651
652
653
654
655
656
656
657
658
658
659
659
660
661
662
663
664
665
666
666
667
668
668
669
669
670
671
672
673
674
675
675
676
677
677
678
678
679
679
680
681
682
683
684
685
685
686
687
687
688
688
689
689
690
691
692
693
694
695
695
696
697
697
698
698
699
699
700
701
702
703
704
705
705
706
707
707
708
708
709
709
710
711
712
713
714
715
715
716
717
717
718
718
719
719
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684

```

Extracting unstructured data from webpages

CSS selectors are like regex: lots of ways to find the same thing

In this case: easiest way to do this is to look for specific text

```
# Get all `p` tags
ins <- html_elements(ins, css = "p")

# Only keep ones containing text "Dr" or "(GTA)"
library("tidyverse")
ins <- ins[str_detect(html_text(ins), "(Dr | [()GTA[)]))")]
print(ins)
```

Extracting unstructured data from webpages

Now let's collect data

```
name <- ins |>
  html_elements(css = "strong") |>
  html_text()
print(name)
```

```
[1] "Dr Ryan Hübert"      "Dr Yuanmo He"
[3] "Dr Thomas Robinson" "Salvatore Finizio"
[5] "Charlotte Kuberka"   "Asami Narita"
```

Extracting unstructured data from webpages

```
title <- ins |>
  html_text() |>
  str_extract("\\s\\\\s+(.+?)", group = 1)
print(title)
```

```
[1] "Associate Professor" "LSE Fellow"
[3] "Assistant Professor" "PhD Student"
[5] "PhD Student"           "PhD Student"
```

```
dept <- ins |>
  html_text() |>
  str_extract("\\s\\\\s+.+?, *(.+)$", group = 1)
print(dept)
```

```
[1] "Department of Methodology"
[2] "Department of Methodology"
[3] "Department of Methodology"
[4] "Department of Geography and Environment"
[5] "Department of Government"
[6] "Department of Methodology"
```

Extracting unstructured data from webpages

```
email <- ins |>
  html_elements(css = "a[href*='@lse.ac.uk'])" |>
  html_attr("href")
print(email)
```

```
[1] "mailto:R.Hubert@lse.ac.uk"
[2] "mailto:Y.He54@lse.ac.uk"
[3] "mailto:T.Robinson7@lse.ac.uk"
[4] "mailto:S.Finizio@lse.ac.uk"
[5] "mailto:L.Kuberka@lse.ac.uk"
[6] "mailto:A.Narita@lse.ac.uk"
```

```
email <- email |>
  str_replace("mailto[:]", "")
print(email)
```

```
[1] "R.Hubert@lse.ac.uk"    "Y.He54@lse.ac.uk"
[3] "T.Robinson7@lse.ac.uk" "S.Finizio@lse.ac.uk"
[5] "L.Kuberka@lse.ac.uk"   "A.Narita@lse.ac.uk"
```

Extracting unstructured data from webpages

And we can turn this into a nice tibble

```
df <- tibble(name, title, dept, email)
print(df)
```

```
# A tibble: 6 x 4
  name              title            dept      email
  <chr>             <chr>           <chr>    <chr>
1 Dr Ryan Hübert   Associate Professor Department o~ R.Hu~
2 Dr Yuanmo He     LSE Fellow       Department o~ Y.He~
3 Dr Thomas Robinson Assistant Professor Department o~ T.Ro~
4 Salvatore Finizio PhD Student     Department o~ S.Fi~
5 Charlotte Kuberka PhD Student     Department o~ L.Ku~
6 Asami Narita     PhD Student     Department o~ A.Na~
```

Dynamic webscraping: a motivational example

In previous years: we scraped data from the U.S. Social Security Administration (at <https://www.ssa.gov/>)

A change in the website has caused this to stop working:

```
library("httr")
ua <- "LSE-MY472/2025 (educational use; contact: R.Hubert@lse.ac.uk"
url <- "https://www.ssa.gov/oact/babynames/numberUSbirths.html"
response <- GET(url, user_agent(ua))
response
```

```
Response [https://www.ssa.gov/oact/babynames/numberUSbirths.html]
Date: 2025-11-07 10:33
Status: 403
Content-Type: text/html
Size: 409 B
<HTML><HEAD>
<TITLE>Access Denied</TITLE>
</HEAD><BODY>
<H1>Access Denied</H1>
```

Dynamic webscraping: a motivational example

Doesn't seem to be due to change in policy—see `robots.txt`:

User-Agent: *

```
Disallow: /admin/
# Disallow: /agency/shutdown/
Disallow: /agency/commissioner/assets/materials/coss-letter-to-senator-warren
Disallow: /agency/commissioner/assets/materials/COSS-Open-Letter-90th-Annive
Disallow: /cgi-bin/
Disallow: /dev/
Disallow: /hlp/
Disallow: /info/
Disallow: /javascript/
Disallow: /myaccount/help/
Disallow: /myaccount/materials/pdfs/SSA-7004.pdf
Disallow: /onlineservices/eservices/
Disallow: /ssaexpress/
Disallow: /w3c/
Disallow: /agency/shutdown/materials/contingency-plan-08-05-19.pdf
Disallow: /agency/shutdown/materials/contingency-plan-08-05-19.html
Disallow: /news/mip/
Disallow: /menu/
Disallow: /news/materials/pdfs/2023-psrw-letter.pdf
```

Scraping with Selenium

Selenium is software that enables browser automation

- ▷ <https://www.selenium.dev/>
 - ▷ Originally developed for web testing purposes
- {RSelenium}: an R wrapper for Selenium
- ▷ Launches a browser session and all communication will be routed through that browser session

There is also a Python wrapper for Selenium: the `selenium` module

Selenium WebDrivers

Selenium uses WebDrivers to automate browsers

- ▷ WebDrivers are APIs for web browsers

Selenium's WebDrivers allow for:

- ▷ Automation of normal browsers like Chrome, Firefox, etc.
 - Selenium launches them just like a human would, and they're visible on screen
 - Useful if you need to watch the automated web browsing
- ▷ Automation of **headless browsers**
 - Full browser functionality, but no visible browser window
 - Useful when you do not have a graphical interface or do not want to clutter your interface

Set up and useful information

To run `{RSelenium}` we are going to use Firefox because it tends to be less buggy than Chrome

Selenium requires you to install the “Java Development Kit” from Oracle, which you can find here:

<https://www.oracle.com/java/technologies/downloads/>

In this course, you will learn how to use Selenium through R, but it's (probably) better to use Python

- ▷ Very similar process, just have to learn slightly different syntax

{RSelenium}: basic navigation

```
# Load the `RSelenium` library
library("RSelenium")

# Create browser instance
rD <- rsDriver(browser=c("firefox"), phantomver = NULL)
browser <- rD$client

# Navigate to a url
browser$navigate("https://www.lse.ac.uk/")

# Navigate back
browser$goBack()

# End a browser instance
browser$close()    # Close browser window
rD$server$stop()   # Stop server
```

{RSelenium}: interacting with a webpage

```
# Get the page source (HTML)
browser$getPageSource() # returns list object

# Find an element on a webpage
some_element <- browser$findElement(using = "xpath",
                                         value = "...")

# Click on an element you found (if button)
some_element$clickElement()

# Type text into an element (if text box)
search_box <- browser$findElement(using = "xpath",
                                         value = "...")
search_box$sendKeysToElement(list("some text"))

# Press enter key (e.g. for text box)
search_box$sendKeysToElement(list(key = "enter"))
```

Getting the SSA data

We can get the SSA data using `{RSelenium}`

```
library("RSelenium")
rD <- rsDriver(browser=c("firefox"), phantomver = NULL)
browser <- rD$client
browser$navigate("https://www.ssa.gov/oact/babynames/numberUSbirths.html")
html <- browser$getPageSource()
html <- read_html(html[[1]]) # Convert raw HTML to tree object
table <- html_table(html)
table <- table[[1]]
head(table)
```

```
# A tibble: 6 × 4
`Year of birth` Male Female Total
<int> <chr>   <chr>   <chr>
1      1880 118,399 97,604 216,003
2      1881 108,276 98,855 207,131
3      1882 122,031 115,694 237,725
4      1883 112,475 120,061 232,536
5      1884 122,738 137,585 260,323
6      1885 115,945 141,947 257,892
```

Outline

- 1 Source code for websites
- 2 Extracting data from a webpage
- 3 Webscraping in R
- 4 Webscraping ethics

Webscraping has gotten more difficult

Partly due to AI and partly due to cyber threats:

- ▷ Websites' terms of service have become much more stringent
- ▷ Their servers are more aggressive about blocking automation

It is increasingly infeasible to collect data via webscraping

Do not get in the habit of trying to *hide* who you are to evade a server's policies (e.g., by using {RSelenium})

- ▷ It's unethical (and potentially illegal)
- ▷ It isn't always successful

Webscraping has gotten more difficult

Something went wrong. Refresh or try again later.
[Learn more](#)

Play K
0:35 / 37:19

● PMQs LIVE: Prime Minister's Questions - 29 October 2025

UK Parliament  [Subscribe](#) 89K views Streamed 8 days ago Watch PMQs with British Sign Language (BSL) - <https://youtube.com/live/lbqoih-13aU?...>

Prime Minister's Question Time, also referred to as PMQs, takes place every Wednesday the ...more

Live chat replay is not available for this video.

● PMQs LIVE: Prime Minister's Questions - 5 ...
UK Parliament  39:16 64K views · Streamed 1 day ago New

Sky News Press Preview | Alex Deane and Christin...
Sky News  5.7K views · Streamed 2 days ago New

REVEALED: More MPs will LOSE to Nigel Farage at ...
GBNews  29:07 19K views · 4 hours ago New

Ed Conway analysis: Is raising taxes the only ...
Sky News  139K views · 2 days ago New

GET BRITAIN WORKING
Kemi Badenoch CALLS out Rachel Reeves in ...
Conservatives 

Webscraping has gotten more difficult

Permissions and Restrictions

You may access and use the Service as made available to you, as long as you comply with this Agreement and the law. You may view or listen to Content for your personal, non-commercial use. You may also show YouTube videos through the embeddable YouTube player.

The following restrictions apply to your use of the Service. You are not allowed to:

1. access, reproduce, download, distribute, transmit, broadcast, display, sell, license, alter, modify or otherwise use any part of the Service or any Content except: (a) as specifically permitted by the Service; (b) with prior written permission from YouTube and, if applicable, the respective rights holders; or (c) as permitted by applicable law;
2. circumvent, disable, fraudulently engage, or otherwise interfere with the Service (or attempt to do any of these things), including security-related features or features that: (a) prevent or restrict the copying or other use of Content; or (b) limit the use of the Service or Content;
3. access the Service using any automated means (such as robots, botnets or scrapers) except: (a) in the case of public search engines, in accordance with YouTube's robots.txt file; (b) with YouTube's prior written permission; or (c) as permitted by applicable law;

What should you do?

This is an area where you need to develop good judgement

Servers are increasingly paranoid about cyberattacks, so they might think you are attacking even if you are compliant

You can use strategies to get around clunky server blocks, but:

- ▷ Check documentation and TOS—are you breaking rules?
- ▷ Are you being blocked because you're being a jerk and sending requests to the server too quickly?

Again, **use good judgement**—there will be consequences if you are too cavalier and get caught

For you while at LSE

Different people have different attitudes about risk

But you will have to explain how you collected your data

If you use a webscraper to collect data, you will be asked to explain whether you violated any terms of service

- ▷ A potential red flag: needing to use {RSelenium}

If you violate TOS: the bar is very high (but not impossible) for justifying why your data collection is necessary

You could fail your capstone/dissertation (or worse)

What should I do to get data I want?

1. Is there a direct download or an API? Then use that!
2. If not: ask yourself if collecting this data seems sketchy: are you stealing someone else's work? are you revealing personal information? are you breaking laws?
3. If you decide you can proceed: consult the `robots.txt` file and the site's terms of service to see if you can use automated methods
4. Create a bot-like webscraper that identifies who you are and your goal, and that does not overwhelm the remote webserver
5. If you get server responses indicating you do not have permission (e.g., 403 Forbidden), re-evaluate your answer to #2
 - ▷ Sometimes purely *technical* reasons for blocking
6. Use browser automation if you are sure it is ethical