Web Scraping and Data Tools

Macroeconomics 3: TA class #6

Andrea Pasqualini

Bocconi University

15 March 2021

Web Scraping: What Is It?

Definition

From Wikipedia: *Web Scraping [...] is used for extracting data from websites. [...] The term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.*

Objective: Build a dataset from potentially unstructured data stored on a remote (web) server
Tools: Computer programs that navigate, identify and organize data into a structured dataset

Web Scraping: Why?

- Limited availability of structured data
  - Structured data are records assembled by somebody (e.g., gov't agency)
  - Cost of assembling records is high (e.g., data entry, data verification, methodologies)
  - Benefit must outweigh the costs

- The Internet as a medium of information exchange
  - The Internet grew fast because it's a business opportunity (e.g., Amazon)
  - Information is available for consumption reasons
  - Assembling structured *public* datasets carries little value for stakeholders

- The Internet as a platform for user-generated content
  - Users generate massive amounts of coded information (e.g., eBay)
  - Users generate massive amounts of uncoded information (e.g., Twitter)

Catch-all reason: uncover new evidence with novel data
Relevant reading: [Edelman, 2012]

Web Scraping: Examples (from Edelman, 2012)

Microeconomics

- **Patrick Bajari and Ali Hortacsu.** 2003. “The Winner’s Curse, Reserve Prices, and Endogenous Entry: Empirical Insights from eBay Auctions.” *RAND Journal of Economics* 34(2): 329–55.
- Bid data from coin sales on eBay reveal bidder behavior in auctions, including the magnitude of the winner’s curse

Macroeconomics

- **Alberto Cavallo.** 2015. “Scraped Data and Sticky Prices.” *NBER Working Paper*
- Daily price data from online supermarkets reveal price adjustment and price stickiness

Financial Economics

- **Werner Antweiler and Murray Z. Frank.** 2004. “Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards.” *Journal of Finance* 59(3): 1259–94.
- Finds that online discussions help predict market volatility; effects on stock returns are statistically significant but economically small

Web Scraping: Which Tools and When to Use Them?

HTTP programming (HTTP: HyperText Transfer Protocol)

- Carefully craft a willful URL
- Obtain and manage response from server
- Most popular: HTTP Application Programming Interfaces (API)

HTML parsing (HTML: HyperText Markup Language)

- Write a program to “surf” specific HTML code
- Navigate to specific points
- Read and write info on separate data storage

Browser automation (DOM: Document Object Model)

- Write a program to hijack your browser
- Make the browser navigate webpages, read and write info
- Useful for dynamically-generated HTML pages (e.g., Twitter)

The Internet: A Primer at Light Speed (1/3)

What happens when you write a URL in the address bar and press Enter?

1. Your browser sends a **HTTP request** for data to the destination web server
2. The web server receives the request and... serves it, sending a **HTTP response** back
3. Your browser receives the data, analyzes the metadata, and displays a payload

A **HTTP request** is a text file with the following components

- A request line
- Request header fields
- An empty line
- (Optional) A message body

A **HTTP response** is a text file with the following components

- A status line
- Response header fields
- An empty line
- (Optional) A message body

The Internet: A Primer at Light Speed (2/3)

Example HTTP request

```
GET / HTTP/1.1
Host: www.example.com

```

Example HTTP response

1 — Status line
2–9 — Header fields
11–18 — Message body

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a HTML document.</p>
  </body>
</html>
```

Status	Meaning
1xx	Information
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

The Internet: A Primer at Light Speed (3/3)

Why do we care about HTTP requests and responses?

HTTP programming

- The response message body is the data we are after (e.g., API)
- Can be a { JSON, CSV, XLS, . . . } file

HTML parsing

- The response message body is (static) HTML code
- We use a program to parse the HTML code

Browser automation

- The response message body may contain a `<script>` tag (e.g., JavaScript)
- The HTML code changes dynamically depending on circumstances
- Cannot handle this with a simple HTML parser

Luckily, Python can handle each and every of these scenarios!

Web Scraping: The Tools in the Python Toolbox

HTTP programming

- `import requests`
- Craft custom requests
- Manage responses
- https://requests.readthedocs.io/en/master/

HTML parsing

- `import bs4`
- Access the HTML code in a response's message body
- Navigate the HTML document with a "Pythonic" interface
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Browser automation

- `import selenium`
- Hijack your web browser, emulate human behavior with a browser
- Navigate the potentially-changing HTML document, trigger functions in web scripts
- https://selenium-python.readthedocs.io/

HTTP Programming: An Example

Go to https://xkcd.com/2434/info.0.json (write a HTTP request)

The message body of the HTTP response is (this is the pretty-print)


```
{
  "month": "3",
  "num": 2434,
  "link": "",
  "year": "2021",
  "news": "",
  "safe_title": "Vaccine Guidance",
  "transcript": "",
  "alt": "I can't wait until I'm fully vaccinated and can safely send chat
    messages in all caps again.",
  "img": "https://imgs.xkcd.com/comics/vaccine_guidance.png",
  "title": "Vaccine Guidance",
  "day": "8"
}
```


All we need to do is

- Translate this JSON text into a Python dictionary
- Feed the dictionary into a `pandas.DataFrame`

HTML Parsing / Browser Automation: Identifying Information

All website content is embedded into its HTML document

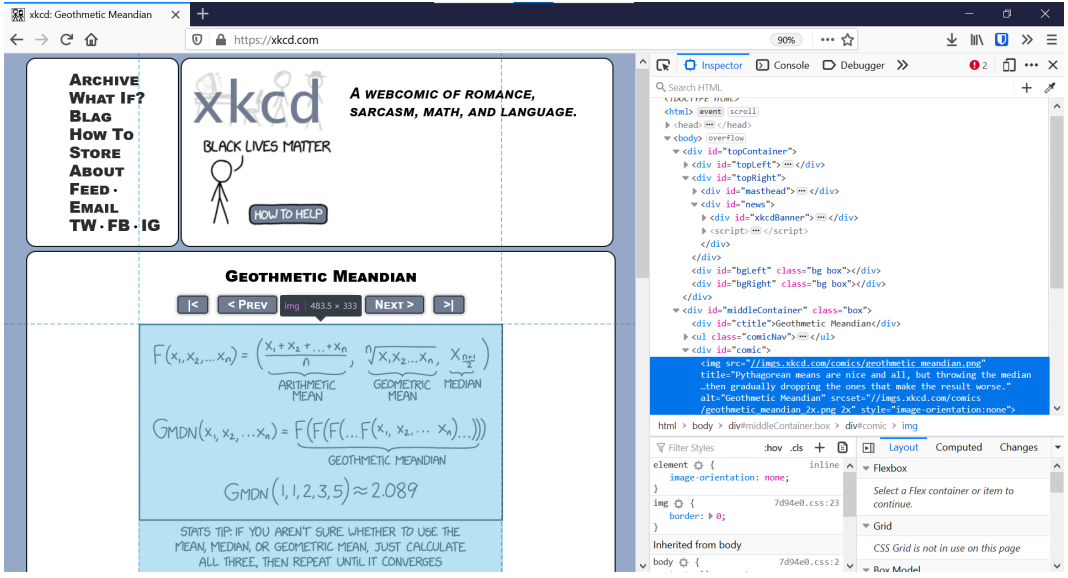
Objective: identify information of interest in the HTML tree

- Use *Web Developer Tools* in your browser to easily identify elements in the HTML tree
- Take notes: what HTML tags are used? How are elements *uniquely* identified?

My experience

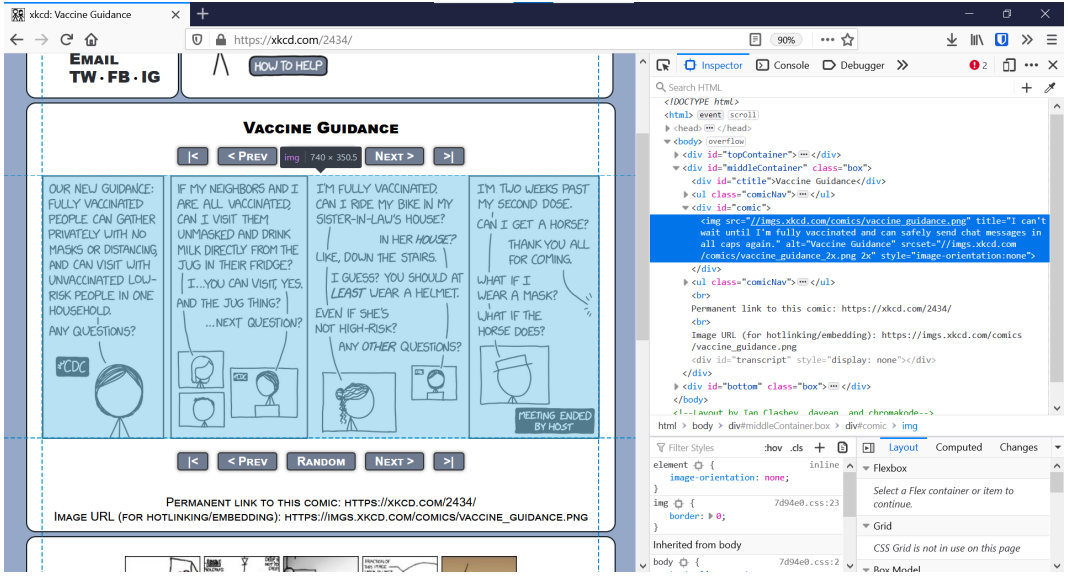
- Tidy websites uniquely identify elements with CSS classes
- Messy websites require smart strategies (e.g., find table whose caption is [smth])
- Some web developers intentionally make seemingly messy HTML code
  - Web dev laziness, and/or
  - Sophisticated back-end design (e.g., Facebook), and/or
  - Deliberate attempt at making scraping difficult

HTML Parsing / Browser Automation: Example (1/2)



I have permission to do this: see `https://xkcd.com/license.html`.

HTML/Browser automation: Example (2/2)



I have permission to do this: see `https://xkcd.com/license.html`.

Web Scraping: Beware!

THIS SLIDE DOES NOT CONSTITUTE LEGAL ADVICE.

- Web scraping may be illegal in your jurisdiction
- Web scraping may be forbidden by a website's Terms and Conditions
- Web scraping may be used to collect sensitive personal information

It is important to take adequate precautions

- Read the Terms and Conditions
- Ask for permission to the website owner (see, "webmaster")
- Seek advice from your University / Institution / Human Studies committee

Web Scraped! . . . Now What?

- Keep the code for scraping separate from the rest
- Store the resulting (raw) dataset on disk, label it with a date
- Clean the dataset (will take a lot of time)
- Keep the code for cleaning separate from the rest
- Store the resulting (clean) dataset on disk
- Research away!

Practice Time

Moving to a Jupyter Notebook

Wrapping up my TA Classes: What to Remember for the Exam

- Solving Bellman Equation for the fixed point $V(\cdot)$
  - Value Function Iter., Policy Function Iter. (i.e., Howard's Improvement), Direct Projection
  - Mind the *Curse of Dimensionality*
  - Policy functions are generally well-behaved (e.g., capital accumulation almost linear)
- Adding exogenous stochastic variables
  - Tauchen, Tauchen-Hussey, Rouwenhorst are limited to AR(1) processes
  - Literature has come up with approaches for general continuous Markov processes
- Solving for the equilibrium prices
  - Define the net excess demand
  - Take it to zero with a zero-finding routine
- Solving for equilibrium prices with heterogeneous agents
  - Combine exogenous transition matrix with policy functions
  - Obtain endogenous transition matrix across the whole vector of state variables
  - Compute the ergodic distribution
  - Aggregate agents and define aggregate demand minus aggregate supply
- Combining idiosyncratic and aggregate shocks
  - Computationally expensive, but feasible (esp. by mixing projection and perturbation methods)
  - Consider MIT shocks: IRF to "parameters" using only idiosyncratic uncertainty

Wrapping up my TA Classes: What to Remember for the Profession

I hope I gave you

- A glimpse into numerical methods in Economics
- A clear understanding of our reliance on computing, as Economists
- A solid foundation into projection methods and models with heterogeneous agents
- A sense of curiosity for “robustness” exercises
- An idea of how much Python can be flexible and powerful

Tip: when you work with code, do like this guy: https://github.com/michaelstepner/healthinequality-code/blob/master/code/readme.md

References

Edelman, B. (2012).
Using Internet Data for Economic Research.
Journal of Economic Perspectives, 26(2):189–206.