Introduction to Web Scraping in Python

Jan Knuf

July 24, 2019

Why Should You Consider Learning How to Web Scrape?

1) Quite simple
2) Sounds fancy
3) Might make you think differently about available data sources
4) Access to highly-specific information (of tailored frequency)
5) Potentially saves a ton of time with tedious tasks

Web Scraping as a Tool for Economic Research

- *Price data:* online retailers, menus of restaurants, flight and train tickets
- *User-produced data* (social media, online auctions, blog posts, forum entries)
- Access to *natural experiments* that are otherwise not documented or very tedious to digitize
- *Person-specific data* on career achievements, media mentions, etc.

Some Examples

Cavallo (2018)	High-frequency price data from online retailers to measure price stickiness
Card and DellaVigna (2013)	Google scholar citation counts on 95% of all econ papers to identify historical developments of publishing in econ
Zimmerman (2019)	Linking career information of top executives in Chile to administrative records
Lowe (2019, WP)	Identifying treated MP's in a weekly natural experiment in the British parliament to capture exogenous variation in their visibility to party leaders

Table of Contents

1. Basics

2. Tutorial

3. Constraints

Basics

The Web Scraping Process

Fetch	Download the HTML-file from the server and store it locally
Parse	Search and navigate through the content and structure of the HTML file
Extract	Filter out the relevant bits of the website
(Manipulate)	Interact with the website

Relevant Packages

Requests	Fetch the HTML-file
lxml, html5lib	Parsing engine
BeautifulSoup	Navigation, Searching, (Manipulation)
re	Need regex very often

Extensions & Alternatives

Selenium:

- When interaction with the website is required (entry in search form, login, load more results,...)
- Replaces **requests**
- Simulates a browser (web driver)

Scrapy

- Entire web scraping framework (fetch, parse, manipulate)
- Documentation here
- Program a "spider" (or bot) to automatically fetch and extract a or multiple websites
- Combines **requests** and **BeautifulSoup** (fetch and parse)

Tutorial

Tutorial

Constraints

Constraints

Time: In most applications you will likely parse a large number of webpages, downloading each of them and search for the objects of interest.
Solutions: Pickle HTML-files, parallelization

Irregularities: (i) the web-developer changes the structure of the HTML-file or (ii) a content producer makes (human) errors when entering the data that lead to deviations from the expected regex patterns.
Solutions: Robust algorithm, exception rules, testing

Reproducibility: As if, some stranger has access to the data-folder of your project and can add, change or delete variables and observations.
Solutions: Store HTML-files locally and make them available with code

Structure or Regex?

- Often both is necessary and helpful
- Searching by *Regex* almost always produces False Positives, if we are searching through the/multiple entire website(s)
- Searching by *structure* is not always stable over time / websites
- It is crucial to make the algorithm robust:
  1. Define a set of possible (structure) locations
  2. Define a set of possible regex pattern
  3. Define case-specific extraction rules depending on where and which pattern was found
  4. Carefully implement state variables and set exception rules

The Issue of Reproducibility

- The raw data can be manipulated over time or disappear
- Even if HTML-source files are stable from initial scrape to review/post publication: might make reproducibilty even more tedious if the data-set has to be assembled again, before replication analysis can be conducted

Storing HTML-source files locally and supplying them together with other data, can help.
Which makes the pickling-step even more useful.
Ideally, some not manipulable format including time stamp for the original HTML-file.

References

- Card, D. and S. DellaVigna (2013). "Nine facts about top journals in economics". In: *Journal of Economic Literature* 51.1, pp. 144–61.
- Cavallo, A. (2018). "Scraped data and sticky prices". In: *Review of Economics and Statistics* 100.1, pp. 105–119.
- Zimmerman, S. D. (2019). "Elite colleges and upward mobility to top jobs and top incomes". In: *American Economic Review* 109.1, pp. 1–47.