Making data on the web useful: scraping

Introduction

Many times data is not easily accessible - although it does exist. As much as we wish everything was available in CSV or the format of our choice - most data is published in different forms on the web. What if you want to use the data to combine it with other datasets and explore it independently? Scraping to the rescue!

Scraping describes the method to extract data hidden in documents - such as Web Pages and PDFs and make it useable for further processing. It is among the most useful skills if you set out to investigate data - and most of the time it's not especially challenging. For the most simple ways of scraping you don't even need to know how to write code.

This example relies heavily on Google Chrome for the first part. Some things work well with other browsers, however we will be using one specific browser extension only available on Chrome. If you can't install Chrome, don't worry the principles remain similar.

Extracting a table from a webpage using Google Spreadsheets

Knowing the structure of a website is the first step towards extracting and using the data. Let's get our data into a spreadsheet - so we can use it further. An easy way to do this is provided by a special formula in Google Spreadsheets.

Walkthrough: Importing HTML tables into Google Spreadsheets.

1. Go to http://drive.google.com, log in and create a new spreadsheet
2. Edit cell A1 (the top left cell)
3. Let's import the table of UK MPs
4. Enter the following formula into the cell: =importHTML("http://www.parliament.uk/mps-lords-and-offices/mps/","table",2) (The last number indicates the number of the table in the document, just try them out and find the matching one...)

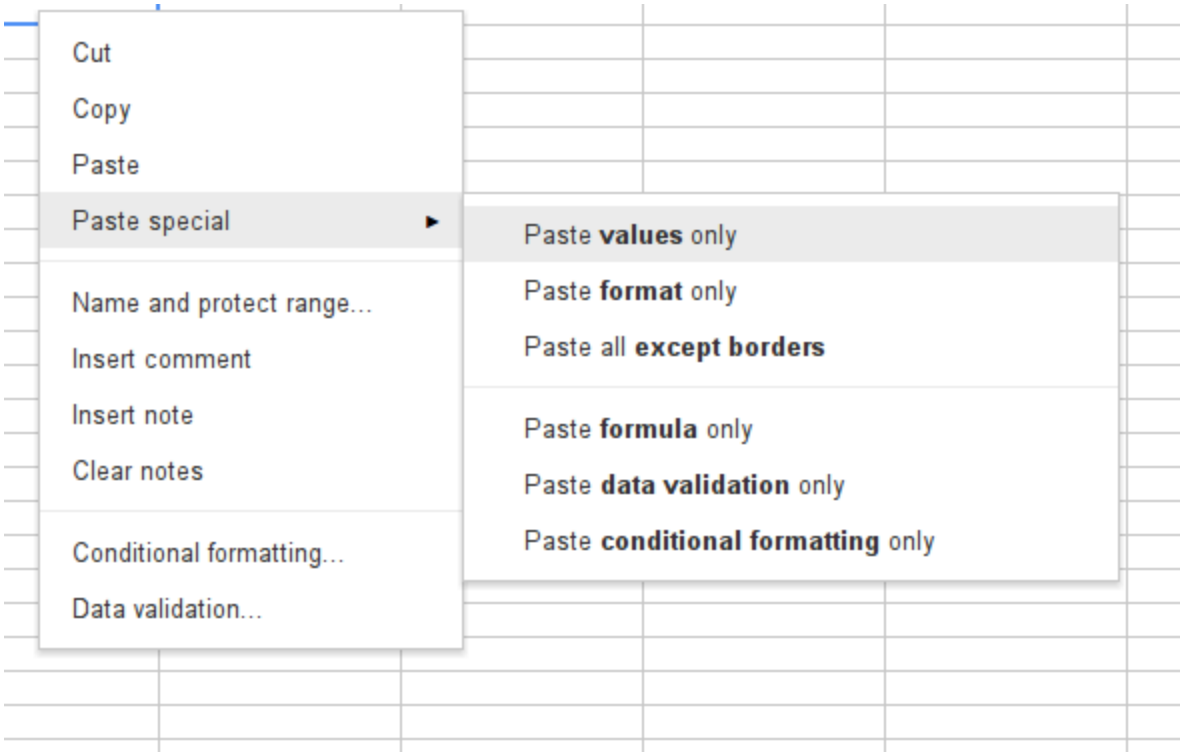
	A	B	C	D	E
1	=importHTML("http://www.parliament.uk/mps-lords-and-offices/mps/","table",2)				
2					
3					
4					
5					
6					
7					
8					

5. Press Enter
6. Wait for a while and see the a table magically appear

fx =CONTINUE(A1, 2, 1)

	A	B
1	Surname, First name	Constituency
2	*A*	
3	Abbott, Diane (Lab)	Hackney North and Stoke Newington
4	Abrahams, Debbie (Lab)	Oldham East and Saddleworth
5	Adams, Nigel (Con)	Selby and Ainsty
6	Afriyie, Adam (Con)	Windsor
7	Ainsworth, Bob (Lab)	Coventry North East
8	Aldous, Peter (Con)	Waveney
9	Alexander, Danny (LD)	Inverness, Nairn, Badenoch and Strathspey
10	Alexander, Douglas (Lab)	Paisley and Renfrewshire South

7. Now let's do a little cleanup: Delete all heading rows (the ones containing *A* *B* and so on)
8. Notice how if you work with the sheet, the deleted rows appear again and again? This is because the formula keeps refreshing the content.
9. In order to change the content or delete it, we'll need to copy the content of the first sheet into another sheet
10. Select the two columns that were imported in the spreadsheet
11. Copy them using "ctrl"+"c"
12. Create a new sheet in your spreadsheet
13. Right click on the top left cell and select "paste special" - "paste values only"

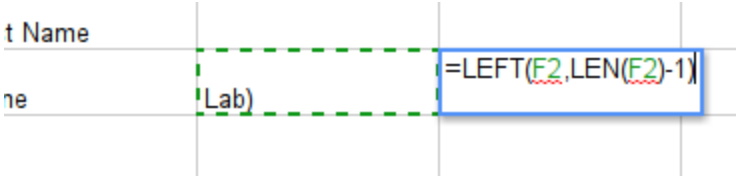


14. Once you've done this, filter the constituency field for "(blank)" values - this will give you all the letter headings (If you're not sure how to do this - check out this course (link to spreadsheets intro sort&filter)
15. Delete the rows (you can simply select all and right click on the rows to delete them)
16. Turn off the filter to see the remaining rows
17. Notice how the first column contains three pieces of information (Surname, Name and Party). Let's split this up
18. The surname is separated from the name and the party by a simple comma - let's split it here.
19. The formula you're going to use is =SPLIT(). It takes two values: 1. the String to be split 2. the delimiter.
20. Label the third column "Surname". This is where our surname is going to be
21. In the second row enter =SPLIT(A2,",") and press enter

	Surname, First name	Constituency	Surname	First
2	Abbott, Diane (Lab)	Hackney North and Stoke Newington	=split(A2,",")	
3	Abrahams, Debbie (Lab)	Oldham East and Saddleworth		

22. This will split up the second row for Surname and Firstname (Party)
23. Extend the formula all the way down.
24. Now you can do the same with the first name (split with "(" as delimiter)
25. Last let's remove the trailing parenthesis from the party name
26. We are going to combine two formulas: =LEFT and =LEN. =LEFT gives us n characters

of a text and LEN gives us the length of the text. We want all characters except the last one - so the formula is =LEFT(F2,LEN(F2)-1)



27. If you hide the columns you're not interested in, you will end up with a sheet having Constituency, Surname and Name and Party of the MP.

Task: Find a website with a table and scrape the information from it. Share your result on the datahub.io (make sure to tag your dataset with schoolofdata.org)

Detour - a short introduction to HTML

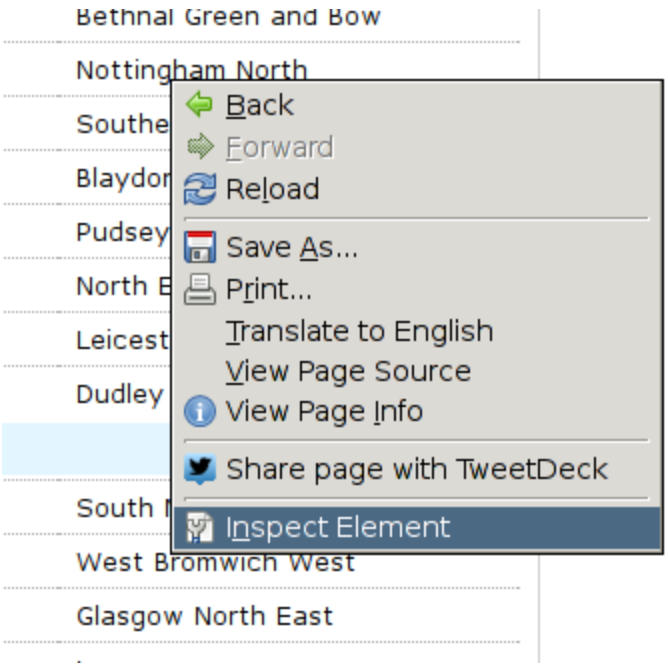
Getting data from websites might seem a little complicated at first - but rest assured, once you've done it a couple of times it will be similar. To extract data from websites we need to peek under the hood and look at the underlying HTML code. Don't worry you don't need to understand every detail of it just to be able to do so.

HTML is the acronym for Hypertext Markup Language and is the language used to describe (markup) web pages. It is the underlying language to structure web-page content. HTML itself does not determine the way things look - it only helps to classify and structure content. So let's peek at some websites.

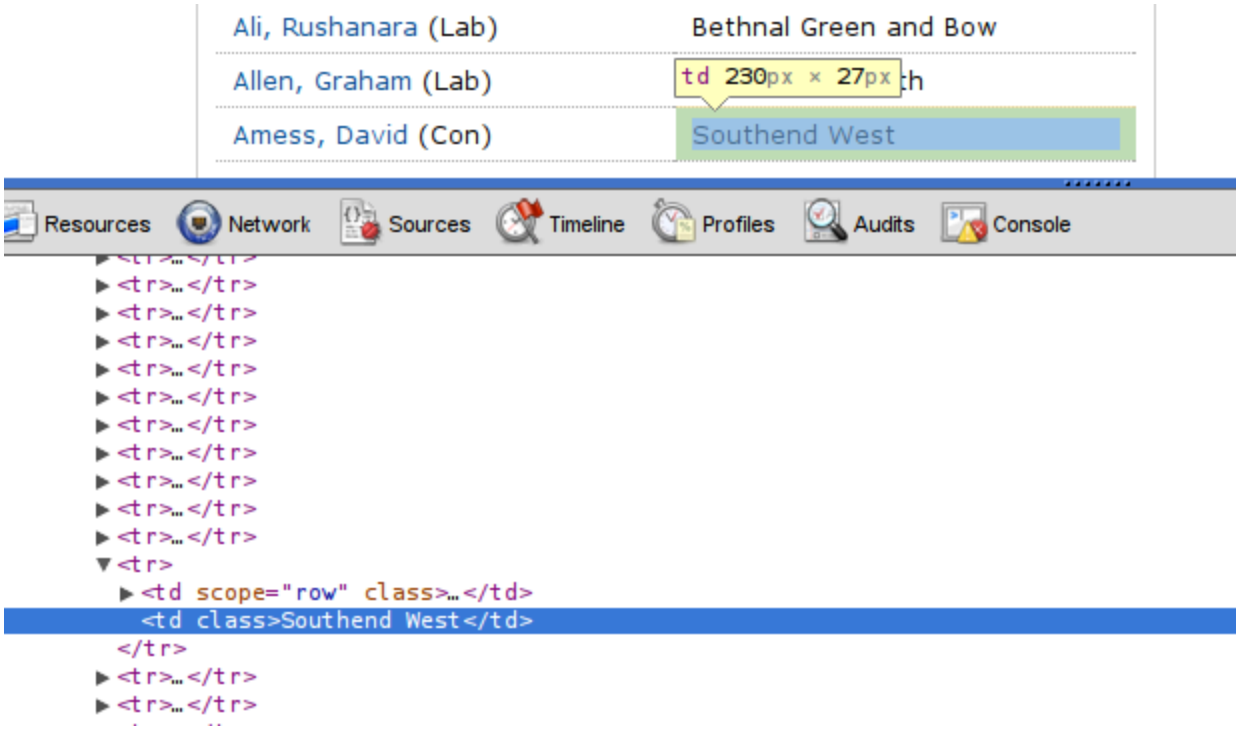
I'm always interested in politics - and parliaments websites are great places to go and start scraping. They usually have listings for members of parliament, their parties etc. Let's look at a sample: The MP listing of the UK parliament:

Walkthrough: Exploring HTML with Google Chrome

1. Open the website listing all MPs for the UK Parliament at http://www.parliament.uk/mps-lords-and-offices/mps/ in Chrome
2. Scroll down to the list of MPs
3. Right click on one of the entries
4. Select "Inspect Element"



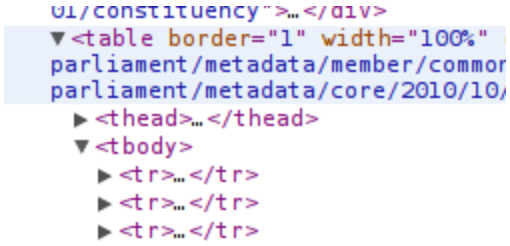
5. Chrome will open a second area on the bottom of the page showing the underlying HTML code - focussed on the element you clicked



6. The pointy brackets are the HTML tags.
7. Now move your mouse up and down and notice how chrome tells you which element is which
8. You can expand and collapse certain sections by clicking on the triangles
9. Did you notice something? Every row in the long list of MPs is within one <tr></tr>

section. <tr> indicates a table row.

10. The names and the constituency are in <td></td> tags - td indicates table data. So we're dealing with a table here?
11. If you scroll up the list you'll notice a <table> element, followed by a <tbody> element - so yes this is a proper HTML table.



12. Go ahead and explore!

HTML is no mystery. If you want to know more about it and how to build webpages with it - visit the School of Webcraft for a gentle introduction.

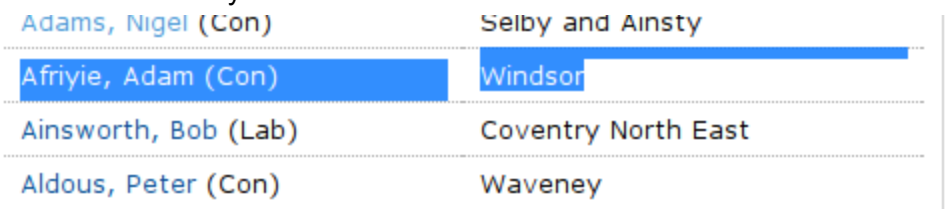
Task: Identify a website and look at the HTML code using Inspect Element. Did you find something interesting?

Scraping websites using the Scraper extension for Chrome

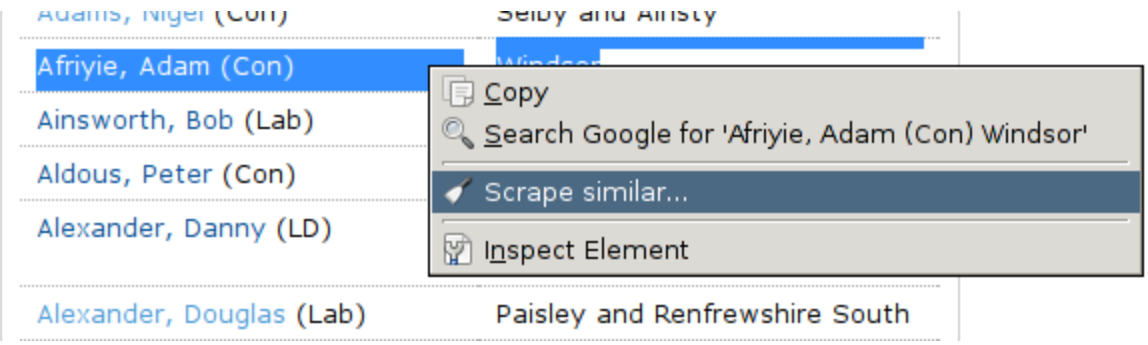
If you are using Google Chrome there is a browser extension for scraping web pages. It's called "Scraper" and it is easy to use. It will help you scrape a website's content and upload the results to google docs.

Walkthrough: Scraping a website with the Scraper extension

1. Open Google Chrome and click on Chrome Web Store
2. Search for "Scraper" in extensions
3. The first search result is the "Scraper" extension
4. Click the add to chrome button.
5. Now let's go back to the listing of UK MPs
6. Open http://www.parliament.uk/mps-lords-and-offices/mps/
7. Now mark the entry for one MP



8. Right click and select "scrape similar..."



9. A new window will appear - the scraper console

	Surname, First name	Constituency
1	A	
2	Abbott, Diane (Lab)	Hackney North and Stoke Newington
3	Abrahams, Debbie (Lab)	Oldham East and Saddleworth
4	Adams, Nigel (Con)	Selby and Ainsty
5	Afriyie, Adam (Con)	Windsor
6	Ainsworth, Bob (Lab)	Coventry North East
7	Aldous, Peter (Con)	Waveney
8	Alexander, Danny (LD)	Inverness, Nairn, Badenoch and Strathspey
9	Alexander, Douglas (Lab)	Paisley and Renfrewshire South

10. In the scraper console you will see the scraped content
11. Click on “Save to Google Docs...” to save the scraped content as a Google Spreadsheet.

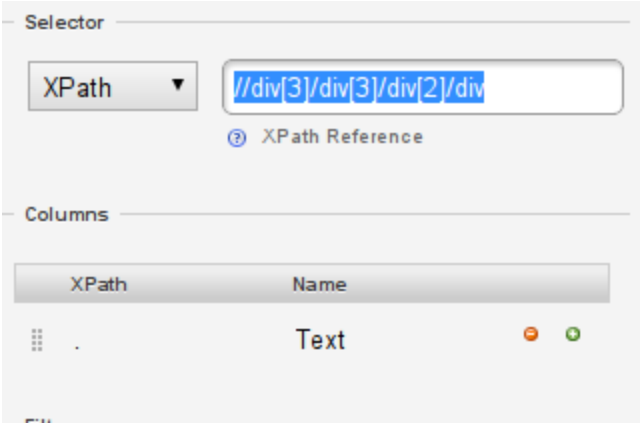
Easy wasn’t it? Now let’s do something a little more complicated. Let’s say we’re interested in the roles a specific actress played. The source for all kinds of data on this is the IMDB (You can also search on sites like DBpedia or Freebase for this kinds of information; however, we’ll stick to IMDB to show the principle)

Waltkthrough: extended scraping with the Scraper extension

1. Let’s say we’re interested in creating a timeline with all the movies the Italian actress Asia Argento ever starred; where do we start?
2. The IMDB has a quite comprehensive archive of actors. Asia Argento’s site is: http://www.imdb.com/name/nm0000782/
3. If you open the page you’ll see all the roles she ever played, together with a title and the year - let’s scrape this information
4. Try to scrape it like we did above
5. You’ll see the list comes out garbled - this is because the list here is structured quite

differently.

6. Go to the scraper console. Notice the small box on the upper left, saying XPath?
7. XPath is a query language for HTML and XML.
8. XPath can help you find the elements in the page you're interested in - all you need to do is find the right element and then write the xpath for it.
9. Now let's assemble our table.
10. You'll see that our current Xpath - the one including the whole information is "//div[3]/div[3]/div[2]/div"



11. Xpath is very simple it tells the computer to look at the HTML document and select element number 3, then in this the third one, the second one and then all elements (which if you count down our list, results in exactly where you are right now.
12. However, we'd like to have the data separated out.
13. To do this use the columns part of the scraper console...
14. Let's find our title first - look at the title using Inspect Element

15. See how the title is within a <b> tag? Let's add the tag to our xpath.
16. The expression seems to work well: let's make this our first column
17. In the "Columns" section, change the name of the first column to "title"
18. Now let's add the XPATH for the title to it
19. The xpaths in the columns section are relative, that means "./b" will select the <b> element
20. add "./b" to the xpath for the title column and click "scrape"

21. See how you only get titles?
22. Now let's continue for year? Years are within one
23. Create a new column by clicking on the small plus next to your "title" column
24. Now create the "year" column with xpath "./span"

25. Click on scrape and see how the year is added
26. See how easily we got information out of a less structured webpage?

Task: Find a webpage having information you are interested in and scrape it! Don't forget to post your results on datahub.io.

Scraping more than one webpage: Scraperwiki

Until now we've only scraped data from a single webpage. What if there are more? Or you want to scrape complex databases? You'll need to learn how to program - at least a bit. Fortunately for you there is a good website for programming scrapers: Scraperwiki.com

Scraperwiki has two main functions: You can write scrapers - which are optionally run regularly and the data is available to everyone visiting - or you can request them to write scrapers for you. The latter costs some money - however it helps to contact the Scraperwiki community (Google Group) someone might get excited about it and help you!.

If you are interested in writing scrapers with Scraperwiki, check out this sample scraper - scraping the MP data like we did in the examples above: https://scraperwiki.com/scrapers/members_of_the_uk_parliament/ - click View source to see the details. Also check out the scraperwiki documentation: https://scraperwiki.com/docs/python/

Summary:

In this course we've covered Web scraping and how to extract data from websites. The main function of scraping is to convert data that is semi-structured into structured data and make it easily useable for further processing. While this is a relatively simple task with a bit of programming - for single webpages it is also feasible without any programming at all. We've introduced =importHTML and the Scraper extension for your scraping needs.

Further Reading

See blogpost: http://schoolofdata.org/2012/12/04/the-webpage-is-the-api-scraping-resources/