**Scraping without coding**

* How do journalists get data?
  + From humans: **Ask nicely** or **FOIA**
  + From computers: **Download** or **scrape**
    - *Scraping is the computer version of filing the FOIA – you know they have the info and you know you want it, so play hardball*
  + **Scraping** is a term for getting data off the web
    - **“Extractors”**pull one web page
    - **“Crawlers”**pull multiple web pages
    - **“Pagination”**is when the data is on multiple pages
* Basic HTML
  + **HTML** is under the hood of websites. It’s what your computer is seeing.
  + Right click and **“view source”** in your browser and you can pull it up in a separate window, or right click and **“inspect element”** to see it in the same window.
  + In HTML, content is wrapped in **tags** that look like this:
    - <> content content </>
    - If you see **<table>** in the web HTML that means you have a table
  + Here’s an HTML dictionary: [https://www.w3schools.com/tags/](https://protect-us.mimecast.com/s/X8dVBluL2KbMip?domain=w3schools.com)
* Use Google Sheets formula to scrape a table or list:
  + Let’s say we want to scrape the table on this Wikipedia page: <https://en.wikipedia.org/wiki/List_of_parishes_in_Louisiana>
  + We need to use the following formula in Google sheets
    - **=ImportHTML(“url”, “element”, numberElement)**
  + The “element” means we need to tell Google sheets whether this is a table or a list that we want to scrape. In this case, it’s a table.
  + Right click and “view source” on the webpage if you need to figure out what element and number you need to refer to.
  + The first element number is 0 and the second is 1. So the first table on the page is table 0 and the second table is table 1. It’s a weird computer thing.
  + Our formula will look like this (don’t forget quotation marks):
    - **=ImportHTML(“**[**https://en.wikipedia.org/wiki/List\_of\_parishes\_in\_Louisiana**](https://en.wikipedia.org/wiki/List_of_parishes_in_Louisiana)**”, “table”, 0)**
  + Using this method, you can only scrape tables and lists, and only from pages with distinct URLs. You need a more advanced formula or tool to scrape anything else.
  + Note that any time the website changes, the changes show up in your spreadsheet. That can be good if you want to keep it up-to-date, but bad if you want to save historical versions of the data on a website.
* Basic XML
  + To do some scraping that’s a little bit more advanced in Google sheets, you need to scrape something called the **XML** code, rather than the HTML code. This will let you scrape different parts of a webpages that aren’t tables or lists.
  + We are going to be using the **XPath** in the XML code which looks like this:
    - **//table[@class=’wikitable’]/tr[1]/td[4]**
    - It always begins with two slashes, then the html element, then potentially an optional identifier in square brackets, then it will specify which row and column it is. This is telling us we’re looking at a table on a Wikipedia page, specifically the first row and fourth column.
    - Note that this works like a URL: The address starts big and then goes small to tell you where you are in the code.
  + You can scrape everything at a particular XPath using this Google Sheets formula, and don’t forget the quotation marks:
    - **=importXML(“source”, “XPath”)**
  + Let’s say we want to scrape all the headers on our Wikipedia page. We just need to know what the XPath is for those headers. Fortunately, that’s easy.
    - **=importXML(“URL”, “//h2”)**
    - **=importXML(“https://en.wikipedia.org/wiki/List\_of\_parishes\_in\_Louisiana”, “//h2”)**
  + Sometimes it’s less easy to tell what the XPath is. In chrome, you can highlight the part of the code you want to scrape, click “inspect” and it will bring up that part of the code.
  + Let’s say we want to scrape the information on when the Wikipedia page was last edited, which is way down at the bottom.
    - Right click on the information, click “inspect”, and see that it shows you where this information exists in the code.
    - Hover over the code that highlights the information we want to scrape. Right click it, then press copy, and then copy XPath. Paste this somewhere on your Google sheet so you can take a look at it.
    - In this case, the XPath is: **//\*[@id="footer-info-lastmod"]**
    - Important note: If the X Path contains double quotation marks in it, you need to **switch them to single quotation marks** or you will get an error! The whole XPath needs to be surrounded in the formula by double quotes, so Google sheets gets confused if you have double quotes inside those quotes. That means our XPath should look like this: **//\*[@id=’footer-info-lastmod’]**
    - So, our formula should look like this: **=importXML("https://en.wikipedia.org/wiki/List\_of\_parishes\_in\_Louisiana","//\*[@id='footer-info-lastmod']")**
  + The XML formula works 99% of the time, but the HTML formula doesn’t work for that many use cases. You can also do the HTML formula using the ImportXML function so you can go straight to that.
* Scraping an RSS feed
  + Here’s a definition of an [**RSS feed**](https://protect-us.mimecast.com/s/7GXDB7tqgaXpfL?domain=support.squarespace.com): “An RSS feed is an .xml file that contains your newest content. News aggregators, feed readers, email subscriptions, and podcast lists all pull content from RSS feeds.”
  + The formula is simple: **=importFeed(“URL”)**
  + If you want to pull specific information in the feed, you can do that by adding other information in the formula, but we won’t worry about that now.
  + You can use this formulafor scraping a feed at regular intervals. Here’s a tutorial: [https://support.google.com/docs/answer/3093337?hl=en](https://protect-us.mimecast.com/s/EJRLB5hko357sn?domain=support.google.com)
  + Rarely does this work perfectly. Then troubleshoot, troubleshoot, troubleshoot.
  + Let’s say we want to scrape one of these New York Times newsletters: <http://rss.nytimes.com/services/xml/rss/nyt/HomePage.xml>
  + We can use the URL that points directly to the RSS feed: <http://rss.nytimes.com/services/xml/rss/nyt/HomePage.xml>
  + The formula looks like this:
    - **=ImportFeed(“http://rss.nytimes.com/services/xml/rss/nyt/HomePage.xml”)**
* Other tools for scraping
  + **OutWit Hub** is a desktop app that can identify each HTML element on a webpage and scrape it – you can see all the tables, all the lists, all the links, etc. Then export it.
  + [**Import.io**](https://protect-us.mimecast.com/s/RKgVB2szOAgXh7?domain=import.io) is a desktop app with a function that scrapes static webpages. It’ll probably work on a simple webpage like with HTML tables. There is not much troubleshooting for this program, unfortunately, but if you know XPath you can use it to build a manual extractor. It also lets you build crawlers! BUT each data point must have the same XPath or it won’t work.
    - You can also create a **“connector”**where you record yourself on the webpage (a “macro”) and it repeats it as often as necessary
    - There is a free service for data journalists where import.io people will do things for you.
  + **Power BI:** Free tool, download it, “get data” > “from web” > add link
  + Other tools: **Zapier** (website), **Web Scraper** (chrome extension), **CloudScrape** (web service)