This worksheet is meant to be accompanied by http://publish.illinois.edu/johnrgallagher/scrape/

WEB-SCRAPING WORKSHOP WITHOUT CODING[1]

We're going to practice pulling data from a single webpage (in order to automate extracting multiple pages, you need a computer script or some sort of software). It's important to remember that web-scraping can present random problems because people build their websites with a variety of architecture and *ad hoc* structure. As a result, many websites require specific troubleshooting but the general ideas in this workshop will help you determine how, and in what ways, you might use web-scraping.

For this workshop, you'll gain experience with the following:

- Learn about the concepts of web-scraping and XPath query language
- Inspecting, and sorting through, webpage source code
- Calling webpage information into a spreadsheet (Google sheets)
- Manipulating data into basic visualizations

What exactly are we going to do?
We are going to use Google sheets and XPath query language (manipulating the information in XML documents such as webpages) to extract information from a website into a spreadsheet.

Note: this is the first step in web-scraping and practicing this can take a while—semester-long courses are taught on web-scraping.

Prelude (5 min): Copy and paste the table from the Game of Thrones character list (https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_characters) into a google sheet. Congratulations, you've performed the basic concept of web-scraping!

Question: what are some issues with what you've copy and pasted?

XPath language:

The above language is XPath query language.

- XPath is a language used for searching and identifying parts of a particular document. See our webpage for helpful links.
- XPath syntax summary
- Cheat sheet: https://devhints.io/xpath
- PDF cheat sheet

When importing XML, it's important to remember that double slashes (//) represents all elements of a specific HTML tag, regardless of where that tag is placed in the website architecture. Single slashes (/) represent an immediate child of that parent element. Therefore, if you use the query (//h3) you're researching for all "header 3" in your document. However, if you search for (//h3/p) you're performing an extraction for all header 3 with an immediate paragraph (not likely to exist because a "div" tag likely separates them due to formatting).

[1] For more scalable strategies, I recommend Python or R. These languages take about 2-3 months to learn. I recommend finding institutional resources as well.

This worksheet is meant to be accompanied by http://publish.illinois.edu/johnrgallagher/scrape/

Let's get started!

Step 1: We want to pull off parts of the webpage but not all of it. Therefore, we need more structure. Open up the previous Wikipedia page and find a particular piece of text. Right click and click on the "Inspect" button. Figures 1 and 2 demonstrate looking up the source code of the word "Main." Remember to open up the closed pathways to find some of the text!

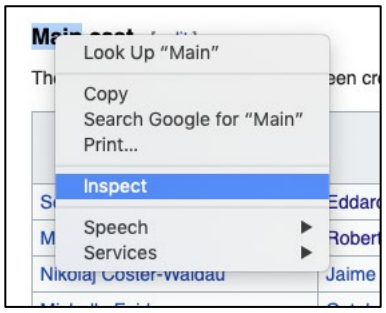

Figure 1: Inspect

Figure 2: Viewing the source code

Take a few minutes and practice looking through the source code. Sometimes, it's counterintuitive about where text is place within the website architecture. Once we've practiced

Step 2 (15 min): Open a Google sheet and, in cell A1, place the hyperlink for the Game of Thrones character list. This will serve as the reference point for calling information into the spreadsheet. XPath query language (using HTML and Microsoft's language) coupled with the "=IMPORTXML" function in Google sheets, can quickly and easily create a makeshift web-scraper. See Figure 3 below.

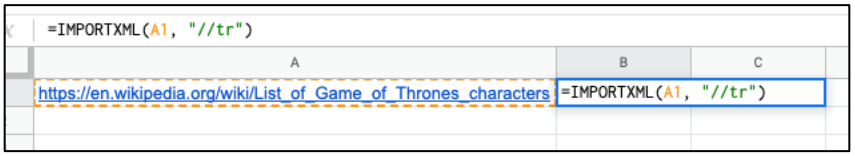

Figure 3: Setting up your google sheet

Practice using each of functions below on the following webpage. I recommend a new Google sheet for each function. I'll take us through each of these.

https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_characters (place this link into A1 in a Google spreadsheet)

```
=IMPORTXML(A1, "//tr")       (all row inside of a table)
=IMPORTXML(A1, "//p")        (all paragraph)
=IMPORTXML(A1, "//li")       (all list)
=IMPORTXML(A1, "//ul/li")    (all unordered list with the list inside all
                            of those unordered lists)
```

This worksheet is meant to be accompanied by http://publish.illinois.edu/johnrgallagher/scrape/

```
=IMPORTXML(A1, "//h3")        (header 3; will take all of the main
                              characters)
=IMPORTXML(A1, "//p/b")       (take all of the names from the paragraph
                              Descriptions)
```

Note the differences between the following two examples:

```
=IMPORTXML(A1, "//a/@href")         all links on this page
=IMPORTXML(A1, "//p//a/@href")      all links within a paragraph tag

=IMPORTXML(A1, "//h3//span[contains(@class, 'mw-headline')]")
                                 Pulls the header 3 names with just the name
```

Step 3 (10 min): Practice being a bit more precise in your XPath query call.

You can copy an XPath query path for a specific piece of information (John will show how) but, be warned, this does not always work. Take a few minutes and practice copying the XPath query, making sure to change the double quotation marks into single quotation marks. If you need help, refer to the cheat sheet: https://devhints.io/xpath.

[Let's look at three different examples that trying to get the list of names from the Game of Thrones character page]

You can use the following text to query one item on our Game of Thrones character page:

```
=IMPORTXML(A1,"//*[@id='mw-content-
text']/div/table[2]/tbody/tr[3]/td[1]/a")

=IMPORTXML(A1, "//img//@src")
```

Now try this:

```
=IMPORTXML(A1, "//*[@id="mw-content-text"]/div/div[14]/div/a/img")
```

The above errors…why?

…It errors because the copied XPath gives you double quotations marks and because you can't simply scrape an image using XPATH. You can scrape the source of the image, however. Here is the correct pathway:

```
=IMPORTXML(A1, "//*[@id='mw-content-text']/div/div[14]/div/a/img/@src")
```

This worksheet is meant to be accompanied by http://publish.illinois.edu/johnrgallagher/scrape/

Step 4: After you've done this, open the sample YouTube web-scraped page. This Google sheet has precise values specified already. Copy this sample into a new sheet and practice changing the URL to a new YouTube channel (but not the XPath language). You should be able to see the information change.

Be warned: YouTube is exponentially more complex than Wikipedia (lots of JavaScript)

Be warned, again: This kind of web-scraping won't work on Twitter…John will explain!

Step 4: John will now demonstrate how we can use basic visualizations to identify trends and such. (YouTube and Wikipedia article)

Step 5: (not in this workshop) Automate! Find institutional resources to help you automate (discuss)