Scraping

Erin Rossiter

This week

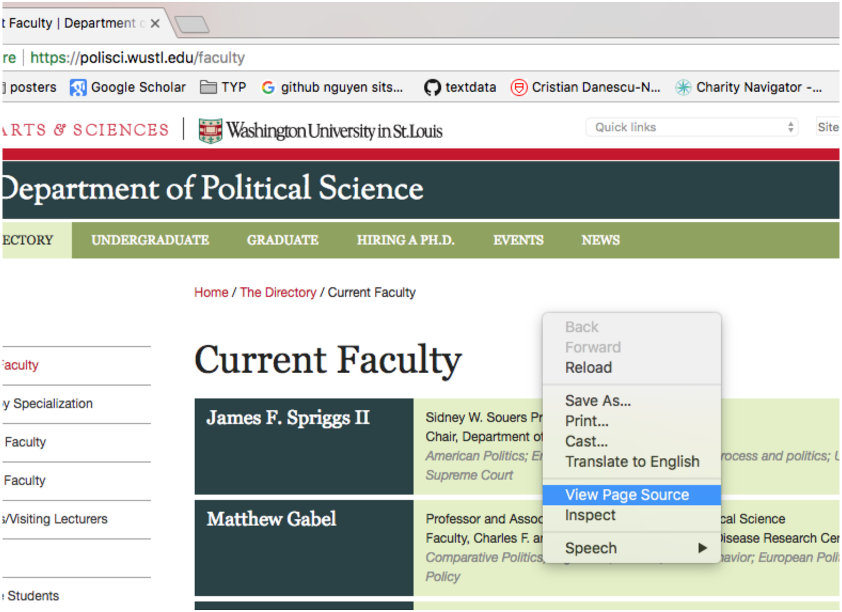
- I use what we go over today (and this week) a lot.
- Scraping, APIs, text handling is where Python beats R.
- The frustrating stuff from last week (exception handling!) is very useful here.

First things first: Page Source

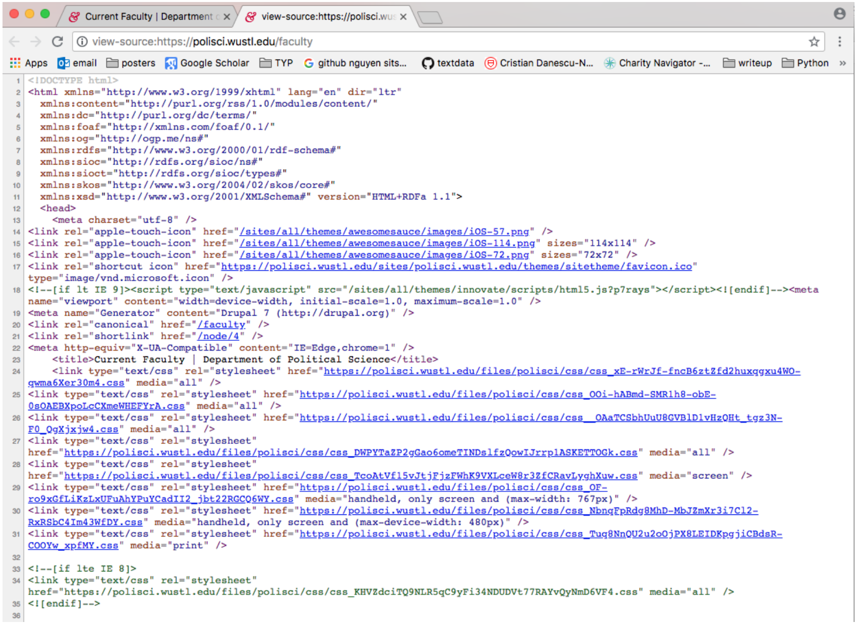
```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```



https://polisci.wustl.edu/faculty



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" dir="ltr"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:dc="http://purl.org/dc/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:og="http://ogp.me/ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:sioc="http://rdfs.org/sioc/ns#"
  xmlns:sioct="http://rdfs.org/sioc/types#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" version="HTML+RDFa 1.1">
  <head>
    <meta charset="utf-8" />
<link rel="apple-touch-icon" href="/sites/all/themes/awesomesauce/images/iOS-57.png" />
<link rel="apple-touch-icon" href="/sites/all/themes/awesomesauce/images/iOS-114.png" sizes="114x114" />
<link rel="apple-touch-icon" href="/sites/all/themes/awesomesauce/images/iOS-72.png" sizes="72x72" />
<link rel="shortcut icon" href="https://polisci.wustl.edu/sites/polisci.wustl.edu/themes/sitetheme/favicon.ico"
type="image/vnd.microsoft.icon" />
<!--[if lt IE 9]><script type="text/javascript" src="/sites/all/themes/innovate/scripts/html5.js?p7rays"></script><![endif]--><meta
name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0" />
<meta name="Generator" content="Drupal 7 (http://drupal.org)" />
<link rel="canonical" href="/faculty" />
<link rel="shortlink" href="/node/4" />
<meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1" />
    <title>Current Faculty | Department of Political Science</title>
    <link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_xE-rWrJf-fncB6ztZfd2huxqgxu4WO-
qwma6Xer30m4.css" media="all" />
<link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_OOi-hABmd-SMR1h8-obE-
0sOAEBXpoLcCXmeWHEFYrA.css" media="all" />
<link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css__OAaTCSbhUuU8GVBlD1vHzQHt_tgz3N-
F0_QgXjxjw4.css" media="all" />
<link type="text/css" rel="stylesheet"
href="https://polisci.wustl.edu/files/polisci/css/css_DWPYTaZP2gGao6omeTINDslfzQowIJrrplASKETTOGk.css" media="all" />
<link type="text/css" rel="stylesheet"
href="https://polisci.wustl.edu/files/polisci/css/css_TcoAtVf15vJtjFjzFWhK9VXLceW8r3ZfCRavLyghXuw.css" media="screen" />
<link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_OF-
ro9xGfLiKzLxUFuAhYPuYCadII2_jbt22RGCQ6WY.css" media="handheld, only screen and (max-width: 767px)" />
<link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_NbnqFpRdg8MhD-MbJZmXr3i7Cl2-
RxRSbC4Im43WfDY.css" media="handheld, only screen and (max-device-width: 480px)" />
<link type="text/css" rel="stylesheet" href="https://polisci.wustl.edu/files/polisci/css/css_Tuq8NnQU2u2oOjPX8LEIDKpgjiCBdsR-
COOYw_xpfMY.css" media="print" />

<!--[if lte IE 8]>
<link type="text/css" rel="stylesheet"
href="https://polisci.wustl.edu/files/polisci/css/css_KHVZdciTQ9NLR5qC9yFi34NDUDVt77RAYvQyNmD6VF4.css" media="all" />
<![endif]-->

```

Scraping HTML

`urllib2`

- "crawler"; in the background
- navigate to url

`BeautifulSoup`

- download HTML
- parse it for info

Great approach when...

- Info is contained in HTML (older websites are great!)
- Webpage names have a pattern
- Example:
  `http://www.presidency.ucsb.edu/press_briefings.php?year=2018`

Not helpful when...

- Info comes from javascript call, or info is not in HTML for some other reason (fancier websites)
- Example:
  `https://www.oyez.org/cases/2017/17-586`
- Your task is not structured (e.g., using a search engine)
- Example:
  `http://www.spencerdailyreporter.com/`

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
    - They are incredibly fast...

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast...
  - ... but also easier to detect and block

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast...
  - ... but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast...
  - ... but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast...
  - ... but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
  - Simulates behavior of human (e.g., point, click, type)

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
    - They are incredibly fast...
    - ... but also easier to detect and block
    - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
    - Simulates behavior of human (e.g., point, click, type)
    - Less likely to be tracked and blocked

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast…
  - … but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
  - Simulates behavior of human (e.g., point, click, type)
  - Less likely to be tracked and blocked
  - Flexible to undefined tasks

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast…
  - … but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
  - Simulates behavior of human (e.g., point, click, type)
  - Less likely to be tracked and blocked
  - Flexible to undefined tasks
  - Cons:

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast…
  - … but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
  - Simulates behavior of human (e.g., point, click, type)
  - Less likely to be tracked and blocked
  - Flexible to undefined tasks
  - Cons:
    - slow

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
  - They are incredibly fast…
  - … but also easier to detect and block
  - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
  - Simulates behavior of human (e.g., point, click, type)
  - Less likely to be tracked and blocked
  - Flexible to undefined tasks
  - Cons:
    - slow
    - dependent on Internet connection

A second approach: Remote driver

- Beautiful Soup and Urllib are “crawlers” in the “background”.
    - They are incredibly fast…
    - … but also easier to detect and block
    - Plus, sometimes they aren’t amenable to your task
- Selenium is a “remote driver” of your favorite browser
    - Simulates behavior of human (e.g., point, click, type)
    - Less likely to be tracked and blocked
    - Flexible to undefined tasks
    - Cons:
        - slow
        - dependent on Internet connection
        - start-up costs (time, patience) in getting it set up on your machine

Some tricks and advice to become a pro-scraper

- Always check terms of service first.

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task
- I prefer Google Chrome to view page source and insect elements

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task
- I prefer Google Chrome to view page source and insect elements
- Use time breaks to avoid being blocked

Some tricks and advice to become a pro-scraper

- Always check terms of service first.
- Inspect source code and structure of website... think about what approach will work best for your task
- I prefer Google Chrome to view page source and insect elements
- Use time breaks to avoid being blocked
- Expect your code to break–websites change :(

Today

1. How to read and write files
2. How to navigate to url & parse HTML – approach #1
3. Example of selenium – approach #2
4. Lab to practice approach #1