Webscraping

We are going to start webscraping today by doing an exercise that goes through getting information on PhD students at UCSB economics. We are going to utilize 3 main functions today in the `rvest` package:

- `rvest::read_html` - reads an html. Uses link as argument.
- `rvest::html_elements` - takes in a CSS selector as arugment and returns all elements with that selector. To be used after `read_html`
- `rvest::html_text` - gives all text inside the element above. To be used after `html_elements`

There are also a couple other functions we'll explore a little too, although these will be used far less frequently:

- `rvest::html_children` - gives all "children" of the element selected. To be used after `html_elements`
- `rvest::html_table` - specific function for gathering tables. See example.
- `rvest::html_attr` - gets a particular attribute from an element. Common are "href", "img"

A common workflow for scraping is to use the following process:

```
## common workflow - do not run
read_html(link) %>%
  html_elements(css_selector) %>%
  html_text()
```

We'll see this in action in the next example.

Here is a helpful link to get you familiar with html elements and attributes... because it's very easy to forget.

Example 1: Scraping Students from UCSB

```
read_html("https://econ.ucsb.edu/people/students") %>%
  html_elements(".views-row") %>%
  html_text(trim= T)
```

Looks pretty good right? But the trick to webscraping is to get small pieces of information at a time to avoid errors since webpages tend to have missing pieces.

Let's get the names only:

```
page_info <- read_html("https://econ.ucsb.edu/people/students") %>%
  html_elements(".views-row") %>%
  html_elements(".group-second") %>%
  html_text2()


## getting only people
```

```
student_info <- page_info %>%
  as_tibble() %>%
  separate(value, into = c("student", "other"), sep = "\n", extra = "merge") %>%
  extract(other, "office_number",  "(\\d\\d\\d\\d)")
```

Now we can make some fun graphs... but for some other time:

```
student_info %>%
  count(office_number, sort = T)
```

What if we want to get the hyperlinks for each of the people?

```
links <- read_html("https://econ.ucsb.edu/people/students") %>%
  html_elements(".group-first") %>%
  html_elements("a") %>%
  html_attr("href") %>%
  as_tibble() %>%
  rename(link = value)
```

```
## combining together
all_info <- student_info %>%
  bind_cols(links)
```

Notice that you can image looping through all of these links and grabbing subsequent information... maybe this can be a homework assignment.

Example 2: Craigslist Housing

We're going to use the following link:

- https://santabarbara.craigslist.org/search/apa?search_distance=15&postal=93111&min_price=&max_price=&availabilityMode=0&sale_date=all+dates

Let's try and get description, bedrooms, price and html.

Starting with description:

```
description <- read_html("https://santabarbara.craigslist.org/search/apa?search_distance=15&postal=9311
  html_elements(".result-info") %>%
  html_elements(".result-heading") %>%
  html_text2() %>%
  as_tibble() %>%
  rename(description = value)
```

Now let's do price:

```
prices <- read_html("https://santabarbara.craigslist.org/search/apa?search_distance=15&postal=93111&min
  html_elements(".result-info") %>%
  html_elements(".result-price") %>%
  html_text2() %>%
  as_tibble() %>%
  rename(price = value)
```

Your turn! Do bedrooms and html!

```
bedrooms <- read_html("https://santabarbara.craigslist.org/search/apa?search_distance=15&postal=93111&m
  html_elements(".result-info") %>%
  html_elements(".housing") %>%
  html_text2() %>%
  as_tibble() %>%
  rename(bedrooms = value) %>%
  extract(bedrooms, "bedrooms", "(^\\d{1,})") %>%
  mutate(bedrooms = readr::parse_number(bedrooms))
```

Getting HTML Tables

Recall that I told you we may looks at the `html_table` function. Here's an opportunity.

Let's look at basketball data (sorry for non-sports peoples).

```
read_html("https://www.basketball-reference.com/players/p/poolejo01/gamelog/2021") %>%
  html_elements("#all_pgl_basic") %>%  ## get rid of this - why does this happen??
  html_table() %>%
  pluck(1) %>%
  janitor::clean_names()
```