Introduction to Web Scraping

Data Wrangling and Husbrandry

3/2/2020

A lot of data is presented in webpages, for which there may not be a nice delimited file to download. Downloading data from webpages can be quite complex, but we start with static pages, which are a bit easier. We call this computer-aided collection of predominantly unstructured data "web scraping."

HyperText Markup Language (HTML)

- markup language = plain text + markup
- standard for websites
- relevance for web scraping: web architecture is important because it determines where and how information is stored

browsing vs. scraping

- browsing
  1. you click on something
  2. browser sends request to server that hosts website
  3. server returns resource (often an HTML document)
  4. browser interprets HTML and renders it nicely
- scraping with R
  1. you manually specify a resource
  2. R sends request to server that hosts website
  3. server returns resource
  4. R parses HTML (i.e., interprets the structure), but does not render it nicely
  5. it's up to you to tell R which parts of the structure to focus on and what content to extract

Scraping with `rvest`

Basic workflow of scraping with the `rvest` package (installed as part of the `tidyverse` but not loaded with it)

1. specify URL
2. download static HTML behind the URL and parse it
3. extract specific nodes
4. extract content from nodes

rvest example

specify URL

```r
url <-
  "https://en.wikipedia.org/wiki/List_of_colleges_and_unive
browseURL(url) # not necessary for the code to work
```

download static HTML behind the URL and parse it

```r
library(rvest)
url %>%
  read_html()
```

```
## {html_document}
## <html class="client-nojs" lang="en" dir="ltr">
## [1] <head>\n<meta http-equiv="Content-Type" content="tex
## [2] <body class="mediawiki ltr sitedir-ltr mw-hide-empty
```

extract specific nodes

(this step is optional if you are extracting a table)

```
url %>%
  read_html() %>%
  html_nodes("table")
```

```
## {xml_nodeset (13)}
##  [1] <table class="box-Disputed plainlinks metadata ambo
##  [2] <table class="wikitable sortable" style="text-align
##  [3] <table class="wikitable sortable" style="text-align
##  [4] <table class="wikitable sortable" style="text-align
##  [5] <table class="wikitable sortable" style="text-align
##  [6] <table class="wikitable sortable" style="text-align
##  [7] <table class="wikitable sortable" style="text-align
##  [8] <table class="wikitable sortable" style="text-align
##  [9] <table class="wikitable sortable" style="text-align
## [10] <table class="nowraplinks hlist mw-collapsible auto
## [11] <table class="nowraplinks mw-collapsible autocolla
## [12] <table class="nowraplinks mw-collapsible autocolla
```

extract content from nodes

```
college_tables <-
  url %>%
  read_html() %>%
  html_nodes("table") %>%
  html_table(fill = TRUE)
length(college_tables)
```

```
## [1] 13
```

```
class(college_tables)
```

```
## [1] "list"
```

```
class(college_tables[[1]])
```

```
## [1] "data.frame"
```

If all you want are tables, you can skip the `html_nodes()` part. For example, here's some code that gets the tables from the page and looks at the first one.

```
url %>%
  read_html() %>%
  html_table(fill = TRUE) %>%
  .[[1]]
```

```
##   X1
## 1 NA
##
## 1 This article's factual accuracy is disputed. Relevant
```

Note that this produces a list of 13 data frames. If you know ahead of time (say, by looking at the webpage) you can restrict the node by using something like

```
url %>%
read_html() %>%
   html_nodes("table") %>% .[1:2] %>%
     html_table(fill = TRUE) %>% length()
```

```
## [1] 2
```

Cleaning up

Tables imported from webpages almost always need some cleaning up.

```
public <- college_tables[[1]]
public$Public_Private <- "Public"
private <- college_tables[[2]]
private$Public_Private <- "Private"
nj_colleges <- bind_rows(public, private)
nj_colleges[1:2,]
```

```
##   X1
## 1 NA
## 2 NA
##
## 1 This article's factual accuracy is disputed. Relevant
## 2
##   Public_Private                        School       Locatic
## 1         Public                          <NA>          <NA
## 2        Private The College of New Jersey Ewing Townshi
```

```
nj_colleges <- rename(nj_colleges,
    Control = `Control[9]` , Type = `Type[9]`)

nj_colleges$Enrollment
```

```
##  [1] NA             "6,964"        "16,000+"      "21,115
##  [6] "10,130"       "5,200"        "18,484"       "65,000+'
## [11] "10,934 [12]" "10,250"
```

```
str_replace_all(nj_colleges$Enrollment, "[,+]", "")
```

```
##  [1] NA            "6964"        "16000"       "21115 [10]'
##  [6] "10130"       "5200"        "18484"       "65000"
## [11] "10934 [12]" "10250"
```

```
nj_colleges$Enrollment <- nj_colleges$Enrollment %>%
  str_replace_all("[,+]", "") %>%
  str_replace_all("\\[[0-9]+\\]", "")
nj_colleges$Enrollment
```

```
##  [1] NA        "6964"   "16000"  "21115 " "8550"   "1013
##  [9] "65000"  "8111"   "10934 " "10250"
```

```
nj_colleges$Enrollment <- as.integer(nj_colleges$Enrollment
```

```r
table(nj_colleges$Type)
```

```
##
##             Master's Research university
##                    7                   4
```

```r
nj_colleges$Type <-
  str_replace_all(nj_colleges$Type, "university", "Univers
nj_colleges <- nj_colleges %>%
  unite(Category, Public_Private, Type, remove = FALSE)
table(nj_colleges$Category)
```

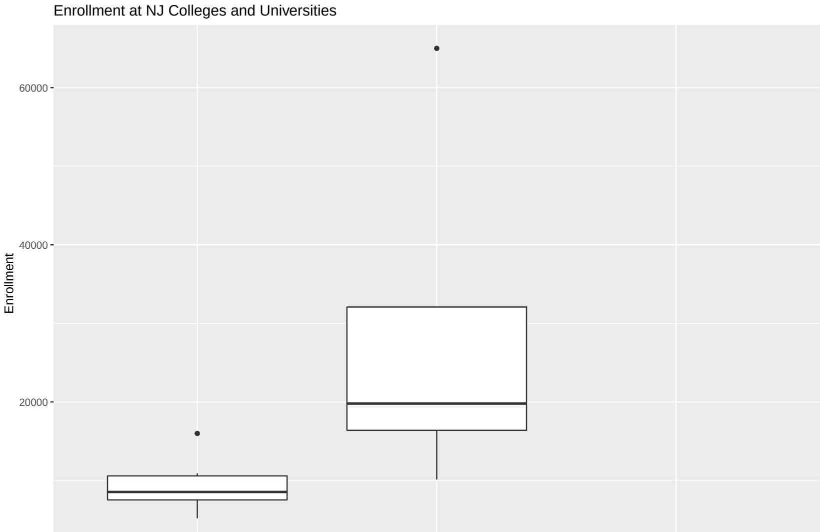
```
##
##             Private_Master's Private_Research University
##                            7                           4
##                    Public_NA
##                            1
```

```r
glimpse(nj_colleges)
```

```
## Observations: 12
## Variables: 11
## $ X1             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, N
## $ X2             <chr> "This article's factual accuracy
## $ Category       <chr> "Public_NA", "Private_Master's",
## $ Public_Private <chr> "Public", "Private", "Private", '
## $ School         <chr> NA, "The College of New Jersey",
## $ Location       <chr> NA, "Ewing Township", "Union and
## $ Founded        <int> NA, 1855, 1855, 1908, 1929, 1881,
## $ Control        <chr> NA, "Public", "Public", "Public",
## $ Type           <chr> NA, "Master's", "Master's", "Rese
## $ Enrollment     <int> NA, 6964, 16000, 21115, 8550, 10
## $ Accreditation  <chr> NA, "MSA", "MSA", "MSA", "MSA", '
```

```
ggplot(data = nj_colleges, aes(Category, Enrollment)) +
  geom_boxplot() +
  ggtitle("Enrollment at NJ Colleges and Universities")
```

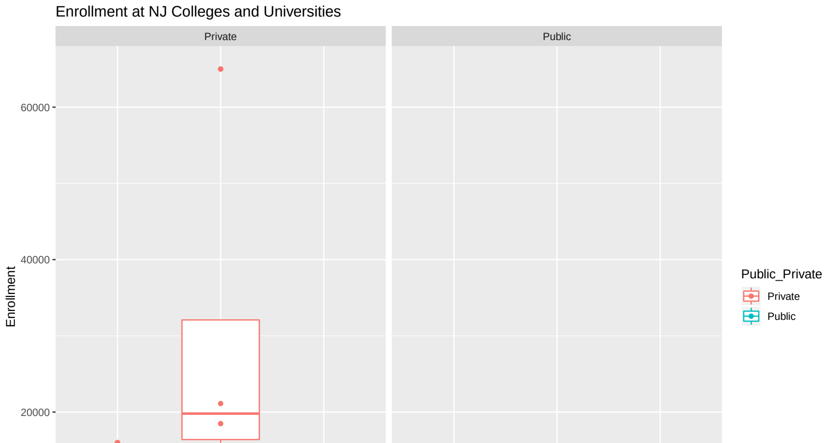
```
## Warning: Removed 1 rows containing non-finite values (st
```



```
ggplot(nj_colleges, aes(Type, Enrollment, colour = Public_P
  geom_boxplot() + geom_point() +
  ggtitle("Enrollment at NJ Colleges and Universities") +
  facet_wrap(~ Public_Private, ncol = 2)
```

```
## Warning: Removed 1 rows containing non-finite values (st

## Warning: Removed 1 rows containing missing values (geom_
```



Another example

Extract title and duration from Youtube trending list:
https://www.youtube.com/feed/trending

Information from Youtube's official page:
https://support.google.com/youtube/answer/7239739?hl=en

There has been controversy over how Youtube's videos are selected:
https://www.cnbc.com/2018/02/22/youtube-trending-algorithm-needs-reform.html

```r
youtube <- "https://www.youtube.com/feed/trending" %>%
read_html() %>% html_nodes("h3") %>%
  html_text()
youtube <- tibble(text = youtube)
head(youtube)
```

```
## # A tibble: 6 x 1
##   text
##   <chr>
## 1 "\n      Best of YouTube\n    "
## 2 "Watch Queue"
## 3 "Queue"
## 4 "Drake - When To Say When & Chicago Freestyle - Durat
## 5 "Real Madrid 2 - 0 FC Barcelona - HIGHLIGHTS & GOALS -
## 6 "The OFFICIAL GENDER REVEAL Of THE ROYALTY FAMILY! **
```

Separating and cleaning

We need to separate the title and duration and do some cleaning.

```
youtube <- "https://www.youtube.com/feed/trending" %>%
read_html() %>% html_nodes("h3") %>%
  html_text()
youtube <- tibble(text = youtube)
youtube <-  youtube %>% separate(text, c("title", "duratio
subset(!is.na(duration))
head(youtube)
```

```
## # A tibble: 6 x 2
##   title
##   <chr>
## 1 "Drake - When To Say When & Chicago Freestyle "
## 2 "Real Madrid 2 - 0 FC Barcelona - HIGHLIGHTS & GOALS -
## 3 "The OFFICIAL GENDER REVEAL Of THE ROYALTY FAMILY! **
## 4 "Lil Uzi Vert - That Way [Official Audio] "
## 5 "Rolling balloon swipe acrylic pouring with water dro
## 6 "Heart shape acrylic pour love - fluid art demo with
```

Extracting other text

By changing the type of node, you can extract other parts of webpage. For example, `h1` refers to the top level header.

```
url %>%
  read_html() %>%
     html_nodes("h1")
```

```
## {xml_nodeset (1)}
## [1] <h1 id="firstHeading" class="firstHeading" lang="en'
```

```
url %>%
  read_html() %>%
     html_nodes("h1") %>% html_text()
```

```
## [1] "List of colleges and universities in New Jersey"
```

HTML tags

In HTML one tags the content with code like `<h1>` to start and `</h1>` to finish. The most common tags are

- `<h1>`, `<h2>`,...,`<h6>`: Largest heading, second largest heading, and so on
- `<p>`: Paragraph elements
- `<ul>`: Unordered bulleted list
- `<ol>`: Ordered list
- `<li>`: Individual List item
- `<div>`: Division or section
- `<table>`: Table

To extract the text of all the paragraph elements, for example, you could use (note that I'm only showing the first 50 characters)

```r
url %>%
  read_html() %>%
     html_nodes("p") %>% html_text() %>%
       map_chr(str_sub, end = 50)
```

```
## [1] "\n"
## [2] "As of 2014[update], the State of New Jersey recogn
## [3] "New Jersey was the only British colony to permit t
## [4] "On August 22, 2012, the New Jersey governor Chris
## [5] "There are three law schools in the state accredite
## [6] "New Jersey has a system of 19 public community col
## [7] "Not all of the county colleges were founded by the
## [8] "Theological schools are typically classified as \"
```

In class exercise

- Go to Youtube's trending page https://www.youtube.com/feed/trending and extract the rank, duration, and view count of each video. Plot view counts against rank. Plotarataion against rank. Can you also get the date it was posted?
- Go to the Wikipedia page on Women's World Cup Alpine Skiing and form a data frame of the annual results
- Clean up the names by dropping the parentheses and numbers
- Find the top three champions in each event

The Structures of a Web Page

Web pages use a variety of ways of organizing their information—to understand how to scrape data from them we have to understand at least some of the structures used for that information.

The four most important technology formats are HTTP, JSON, XML, and AJAX.

HTML

As we've discussed, HTML is the primary language for presenting content on the web. It is primarily composed of

- *elements*, which typically have a start tag, content, and an end tag, and
- *attributes*, which are a feature of tags.

The code

```
<a href = "http://msds-stat.rutgers.edu/">MSDS Home
Page</a>
```

has the anchor tag `<a>` with the attribute of a link. `</a>` is the end tag.

HTML with CSS

In recent years, the look of webpages has devolved to Cascading Style Sheets (CSS). The CSS specifies the look of, say, `<div>` elements of a specified class.

This can be quite useful for scraping because those elements will be labeled, for example with

`<div class = "major"> ... </div>`.

Because HTML is used both to create the look of the page as well as to include information, scraping a page with only HTML can be challenging.

Scraping with rvest and selector.gadget

Consider the Statistics Dept list of faculty at https://www.stat.rutgers.edu/people-pages/faculty. Can we extract the names of the professors?

```
library(rvest)
stat_faculty_page <- read_html("https://www.stat.rutgers.ed
```

To extract the names, we can use selectorgadget to determine which css selector matches the data we want. You can find out more about selectorgadget in the R context with `vignette("selectorgadget")`. It looks like we want `.newstitle`. Use `html_nodes()` to find the nodes that matches that selector (use `html_node()` if you want just the first) and extract its contents with html_text()

```r
library(rvest)
stat_faculty_page <- read_html("https://www.stat.rutgers.ed

stat_faculty_page %>%
  html_nodes(".newstitle") %>%
  html_text() %>%
  str_replace_all("^\\s+|\\s+$", "")
```

```
##  [1] "Waheed U. Bajwa"             "Pierre Bellec"
##  [3] "Javier Cabrera"              "Steve Buyske"
##  [5] "Jerry Cheng"                 "Rong Chen"
##  [7] "Harry Crane"                 "Lee Dicker"
##  [9] "Ruobin Gong"                 "Tirthankar DasGupta"
## [11] "Derek Gordon"                "Regina Y. Liu"
## [13] "Edwin Green"                 "Zijian Guo"
## [15] "Qiyang Han"                  "Donald R. Hoover"
## [17] "Ying Hung"                   "Jason M. Klusowski"
## [19] "John Kolassa"                "Robert Kopp"
## [21] "William Kostis"              "Eun-Young Mun"
## [23] "Neville O'Reilly"            "Anand D. Sarwate"
```

In class Assignment

Get the names of all the associate editors of the Annals of Statistics.

The page https://www.govtrack.us/congress/members/NJ#representatives lists the US Representatives for NJ. Please extract their names (just the names).

Use SelectorGadget to create a data frame from the table at https://www.billboard.com/charts/hot-100 . Include the ranking, the song title, the artist, and last week's ranking

The site https://spotifycharts.com/regional somewhat surprisingly uses html tables, so the chart there can be extracted using `html_table()` without having to use SelectorGadget for help. Get the data frame. If time allows, clean the entries.