17: Data Scraping

Environmental Data Analytics | Kateri Salk

Spring 2020

Objectives

1. Acquire and scrape data from web sources
2. Process web-scraped data into reproducible formats

Set up

```r
getwd()
```

```
## [1] "/Users/ks501/Box/Courses/Environmental Data Analytics/2020/Lessons"
```

```r
library(tidyverse)
library(viridis)
#install.packages("rvest")
library(rvest)
#install.packages("ggrepel")
library(ggrepel)

# Set theme
mytheme <- theme_classic() +
  theme(axis.text = element_text(color = "black"),
        legend.position = "top")
theme_set(mytheme)
```

Data scraping

Sometimes, there may be data that we can access online but are not available in downloadable formats. Data scraping is a technique that allows us to convert unstructured data (e.g., those presented in a webpage) into a structured format (e.g., a csv file).

Methods for scraping data (from Analytics Vidhya data scraping guide):

- Manual copy-paste
- API (retrieve data from standard code; data must be in prescribed format)
- DOM parsing

We will be scraping today using DOM parsing. First, we need to install a tool on our web browser to be able to call the web text we need. The tool is called a Selector Gadget, which for Chrome can be found here.

Now that our selector gadget is in operation, we can start the data scraping process with the `rvest` package.

Example: Environmental Protection Agency's Impaired Waters list

The U.S. Environmental Protection Agency maintains a website that summarizes water quality assessment reports from each state, detailing waters impaired due to nutrient-related causes under Sections 305(b) and

303(d) of the Clean Water Act. There are several useful tables on this website:

1. River quality assessment results
2. Lake and reservoir quality assessment results
3. Bay and estuary quality assessment results

However, these tables aren't in a usable form to be able to analyze and visualize in R. Copying and pasting this information would be tedious, so let's explore the ability for `rvest` to scrape the data directly. (Note: I realize there is a data download link for each of these tables in the webpage; this is meant to be an illustrative exercise).

```
# Specify website to be scraped
url <- "https://www.epa.gov/nutrient-policy-data/waters-assessed-impaired-due-nutrient-related-causes"

# Reading the HTML code from the website
webpage <- read_html(url)
```

Now that R knows which website we want to access, we need to tell it what content to scrape.

1. In Chrome, navigate to the website and activate the Selector Gadget by clicking the icon.
2. Components of the website should now glow orange as you hover over them with your mouse.
3. For each column of the table, we want to highlight just the cells that correspond to that column (and no others). Correctly highlighted items will be either green or yellow. For instance, let's build a new dataset that includes information for the estuary dataset.
4. The first column is "State." Click on "Alabama" and note the following things:

- Alabama is highlighted in green (this is good)
- Subsequent states in the table are highlighted in yellow (this is good)
- The rest of the table is highlighted in yellow (this is not good; we want only the state column)
- The river and lake tables are also highlighted in yellow (this is not good; we want only the estuary table)

5. Now, we need to de-highlight the columns we don't want. De-highlight columns by clicking on website components until they are either de-highlighted or are highlighted red. This may take some trial and error to obtain the correct subset.
6. Copy the text that appears in the window at the bottom of the screen and insert it in the following lines of code.

```
# Grab specific components of the website
State = webpage %>% html_nodes("table:nth-child(20) td:nth-child(1)") %>% html_text()
Estuaries.Assessed.mi2 = webpage %>% html_nodes("table:nth-child(20) td:nth-child(2)") %>% html_text()
Estuaries.Assessed.percent = webpage %>% html_nodes("table:nth-child(20) td:nth-child(3)") %>% html_tex
Estuaries.Impaired.mi2 = webpage %>% html_nodes("table:nth-child(20) td:nth-child(4)") %>% html_text()
Estuaries.Impaired.percent = webpage %>% html_nodes("table:nth-child(20) td:nth-child(5)") %>% html_tex
Estuaries.Impaired.percent.TMDL = webpage %>% html_nodes("table:nth-child(20) td:nth-child(6)") %>% html

# Note: each of these objects has 23 items in it; consistency is good!
# If lengths were not consistent you could use matrix subsetting to make consistent

# Each of these objects is just a value, so we need to convert to a data frame
# Coerce into a data frame
Estuary <- data.frame(State, Estuaries.Assessed.mi2, Estuaries.Assessed.percent,
                          Estuaries.Impaired.mi2, Estuaries.Impaired.percent,
                          Estuaries.Impaired.percent.TMDL)
```

Notice there are some instances where symbols appear in cells where there should just be numbers or NAs. Let's fix this.

```r
# Filter out states with no data
Estuary <- Estuary %>%
  filter(State != "Mississippi" & State != "Oregon" & State != "Washington")

# Use str_replace to remove non-numeric characters
Estuary$Estuaries.Assessed.mi2 <- str_replace(Estuary$Estuaries.Assessed.mi2,
                                               pattern = "([,])", replacement = "")
Estuary$Estuaries.Assessed.percent <- str_replace(Estuary$Estuaries.Assessed.percent,
                                               pattern = "([%])", replacement = "")
Estuary$Estuaries.Assessed.percent <- str_replace(Estuary$Estuaries.Assessed.percent,
                                               pattern = "([*])", replacement = "")
Estuary$Estuaries.Impaired.mi2 <- str_replace(Estuary$Estuaries.Impaired.mi2,
                                               pattern = "([,])", replacement = "")
Estuary$Estuaries.Impaired.percent <- str_replace(Estuary$Estuaries.Impaired.percent,
                                           pattern = "([%])", replacement = "")
Estuary$Estuaries.Impaired.percent.TMDL <- str_replace(Estuary$Estuaries.Impaired.percent.TMDL,
                                                pattern = "([%])", replacement = "")
Estuary$Estuaries.Impaired.percent.TMDL <- str_replace(Estuary$Estuaries.Impaired.percent.TMDL,
                                                pattern = "([±])", replacement = "")

# Does R know that the numeric columns are numbers?
str(Estuary)
```

```
## 'data.frame':    20 obs. of  6 variables:
##  $ State                          : Factor w/ 23 levels "Alabama","Alaska",..: 1 2 3 4 5 6 7 8 9 10
##  $ Estuaries.Assessed.mi2         : chr  "734" "31" "904" "612" ...
##  $ Estuaries.Assessed.percent     : chr  "100" "0" "42" "100" ...
##  $ Estuaries.Impaired.mi2         : chr  "0" "1" "30" "305" ...
##  $ Estuaries.Impaired.percent     : chr  "0" "2" "3" "50" ...
##  $ Estuaries.Impaired.percent.TMDL: chr  "0" "100" "" "59" ...
```

```r
Estuary$Estuaries.Assessed.mi2 <- as.numeric(Estuary$Estuaries.Assessed.mi2)
Estuary$Estuaries.Assessed.percent <- as.numeric(Estuary$Estuaries.Assessed.percent)
Estuary$Estuaries.Impaired.mi2 <- as.numeric(Estuary$Estuaries.Impaired.mi2)
Estuary$Estuaries.Impaired.percent <- as.numeric(Estuary$Estuaries.Impaired.percent)
Estuary$Estuaries.Impaired.percent.TMDL <- as.numeric(Estuary$Estuaries.Impaired.percent.TMDL)
str(Estuary)
```

```
## 'data.frame':    20 obs. of  6 variables:
##  $ State                          : Factor w/ 23 levels "Alabama","Alaska",..: 1 2 3 4 5 6 7 8 9 10
##  $ Estuaries.Assessed.mi2         : num  734 31 904 612 30 ...
##  $ Estuaries.Assessed.percent     : num  100 0 42 100 7 100 7 65 65 5 ...
##  $ Estuaries.Impaired.mi2         : num  0 1 30 305 29 ...
##  $ Estuaries.Impaired.percent     : num  0 2 3 50 98 32 22 83 17 0 ...
##  $ Estuaries.Impaired.percent.TMDL: num  0 100 NA 59 10 NA 100 NA 22 NA ...
```

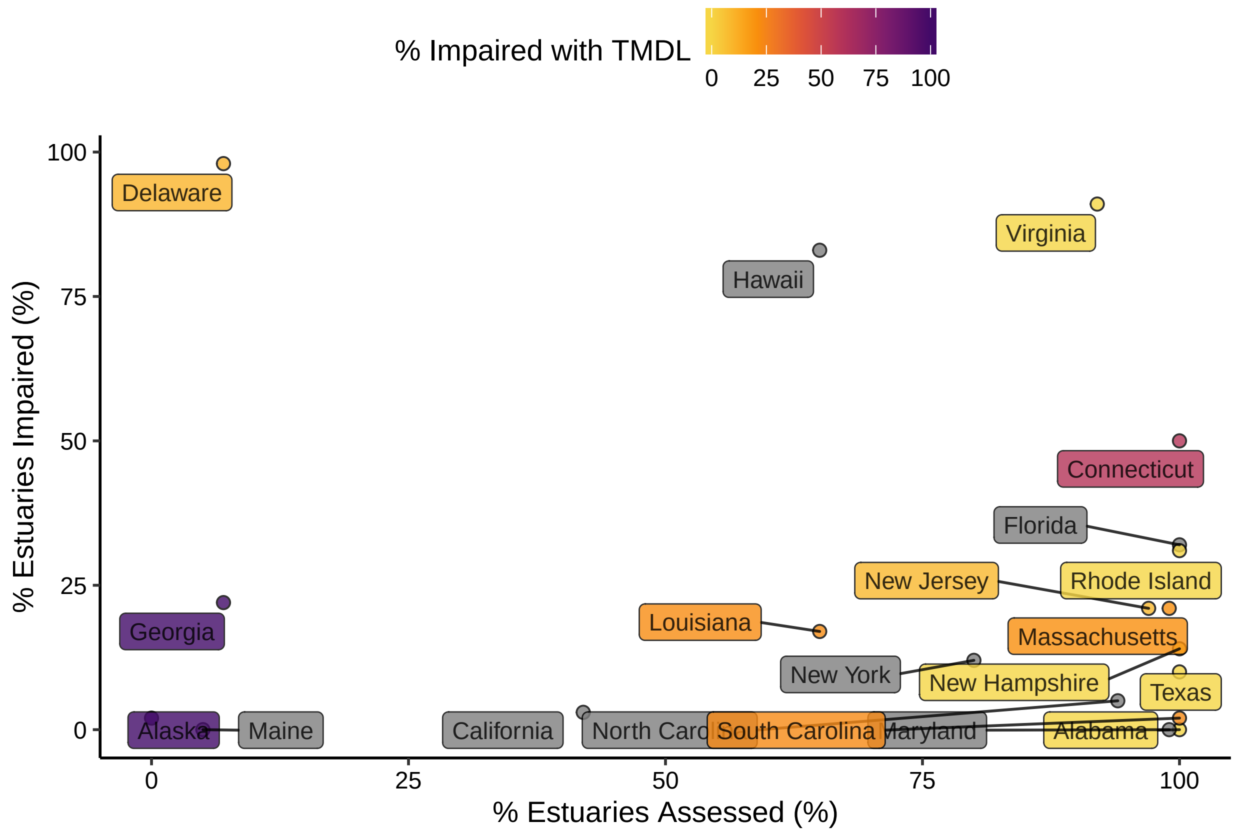
Visualizing scraped data

What might be the relationship between estuary assessment and estuary impairment across the coastal states?

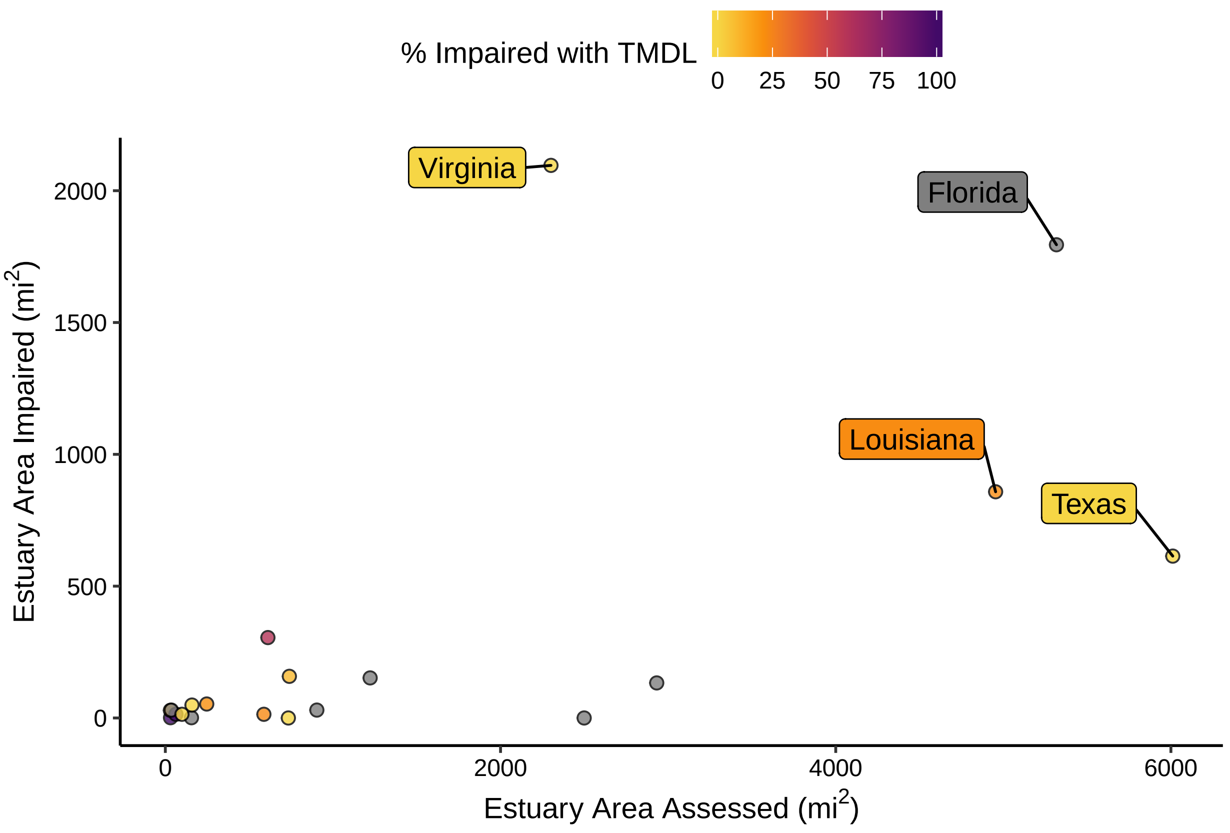
I've added some code to adjust the default settings in ggplot. Uncomment lines to see what changes. Add your own comments as you work through the code, particularly for functions you haven't encountered before.

Note: the fill aesthetic shows up gray for some states. Why?

```
ggplot(Estuary, aes(x = Estuaries.Assessed.percent, y = Estuaries.Impaired.percent,
                    fill = Estuaries.Impaired.percent.TMDL)) +
  geom_point(shape = 21, size = 2, alpha = 0.8) +
  scale_fill_viridis_c(option = "inferno", begin = 0.2, end = 0.9, direction = -1) +
  geom_label_repel(aes(label = State), nudge_x = -5, nudge_y = -5,
                   size = 3, alpha = 0.8) +
  labs(x = "% Estuaries Assessed (%)",
       y = "% Estuaries Impaired (%)",
       fill = "% Impaired with TMDL")
```



```
ggplot(Estuary, aes(x = Estuaries.Assessed.mi2, y = Estuaries.Impaired.mi2,
                    fill = Estuaries.Impaired.percent.TMDL)) +
  geom_point(shape = 21, size = 2, alpha = 0.8) +
  scale_fill_viridis_c(option = "inferno", begin = 0.2, end = 0.9, direction = -1) +
  geom_label_repel(data = subset(Estuary, State %in% c("Virginia", "Florida", "Louisiana", "Texas")),
            aes(label = State), nudge_x = -500, nudge_y = 200) +
  labs(x = expression("Estuary Area Assessed (mi"^"2"*")"),
       y = expression("Estuary Area Impaired (mi"^"2"*")"),
       fill = "% Impaired with TMDL")
```



Further tools: data scraping from PDFs

Online tutorial

GitHub Repo from R Ladies session