Assignment 10: Data Scraping

Sarah Sussman

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on data scraping.

Directions

1. Rename this file `<FirstLast>_A10_DataScraping.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure your code is tidy; use line breaks to ensure your code fits in the knitted output.
5. Be sure to **answer the questions** in this assignment document.
6. When you have completed the assignment, **Knit** the text and code into a single PDF file.

Set up

1. Set up your session:

- Load the packages `tidyverse`, `rvest`, and any others you end up using.
- Check your working directory

```
#1
library(tidyverse)
library(rvest)
library(lubridate)
library(here)

# Check working directory
here()
```

```
## [1] "/home/guest/EDA_Spring2025"
```

2. We will be scraping data from the NC DEQs Local Water Supply Planning website, specifically the Durham's 2024 Municipal Local Water Supply Plan (LWSP):

- Navigate to https://www.ncwater.org/WUDC/app/LWSP/search.php
- Scroll down and select the LWSP link next to Durham Municipality.
- Note the web address: https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=03-32-010&year=2024

Indicate this website as the as the URL to be scraped. (In other words, read the contents into an `rvest` webpage object.)

```
#2
URL <- read_html('https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=03-32-010&year=2024')
```

3. The data we want to collect are listed below:

- From the "1. System Information" section:
- Water system name
- PWSID
- Ownership
- From the "3. Water Supply Sources" section:
- Maximum Day Use (MGD) - for each month

In the code chunk below scrape these values, assigning them to four separate variables.

> HINT: The first value should be "Durham", the second "03-32-010", the third "Municipality", and the last should be a vector of 12 numeric values (represented as strings)".

```
#3
water_system <- URL %>%
  html_nodes("div+ table tr:nth-child(1) td:nth-child(2)") %>%
  html_text()
water_system
```

```
## [1] "Durham"
```

```
PWSID <- URL %>%
  html_nodes("td tr:nth-child(1) td:nth-child(5)") %>%
  html_text()
PWSID
```

```
## [1] "03-32-010"
```

```
ownership <- URL %>%
  html_nodes("div+ table tr:nth-child(2) td:nth-child(4)") %>%
  html_text()
ownership
```

```
## [1] "Municipality"
```

```
monthly_MGD <- URL %>%
  html_nodes("th~ td+ td") %>%
  html_text()
monthly_MGD
```

```
##  [1] "34.5000" "36.0600" "37.3300" "32.1000" "46.6500" "37.3600" "38.2000"
##  [8] "41.9000" "36.5800" "36.7300" "42.9600" "34.4500"
```

4. Convert your scraped data into a dataframe. This dataframe should have a column for each of the 4 variables scraped and a row for the month corresponding to the withdrawal data. Also add a Date column that includes your month and year in data format. (Feel free to add a Year column too, if you wish.)

   TIP: Use `rep()` to repeat a value when creating a dataframe.

   NOTE: It's likely you won't be able to scrape the monthly widthrawal data in chronological order. You can overcome this by creating a month column manually assigning values in the order the data are scraped: "Jan", "May", "Sept", "Feb", etc... Or, you could scrape month values from the web page...

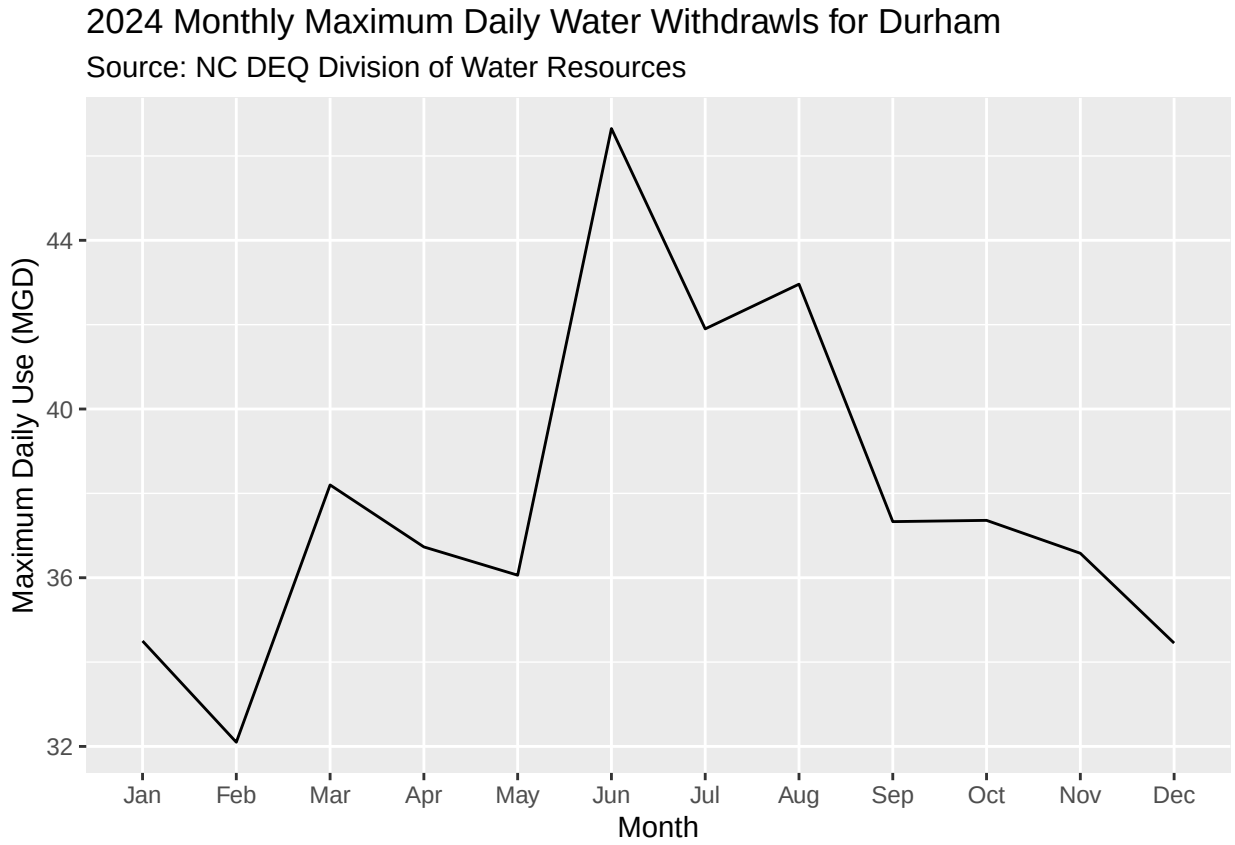
5. Create a line plot of the maximum daily withdrawals across the months for 2024, making sure, the months are presented in proper sequence.

```r
#4
# Scrape for month
month <- URL %>%
  html_nodes(".fancy-table:nth-child(30) tr+ tr th") %>%
  html_text()

# Create dataframe
df_H2Owithdrawls <- data.frame("Monthly MGD" = as.numeric(monthly_MGD),
                               "Ownership" = ownership,
                               "PWSID" = PWSID,
                               "Water system" = water_system,
                               "Month" = month)

# Set months to be in chronological order
df_H2Owithdrawls$Month <- factor(df_H2Owithdrawls$Month,
                                 levels = c("Jan", "Feb", "Mar", "Apr",
                                 "May", "Jun", "Jul", "Aug", "Sep", "Oct",
                                 "Nov", "Dec"))

#5
ggplot(df_H2Owithdrawls, aes(x = Month,
                             y = Monthly.MGD,
                             group = 1)) +
  geom_line() +
  labs(title = "2024 Monthly Maximum Daily Water Withdrawls for Durham",
       subtitle = "Source: NC DEQ Division of Water Resources",
       x = "Month",
       y = "Maximum Daily Use (MGD)")
```



6. Note that the PWSID and the year appear in the web address for the page we scraped. Construct a function with two input - "PWSID" and "year" - that:

- Creates a URL pointing to the LWSP for that PWSID for the given year
- Creates a website object and scrapes the data from that object (just as you did above)
- Constructs a dataframe from the scraped data, mostly as you did above, but includes the PWSID and year provided as function inputs in the dataframe.
- Returns the dataframe as the function's output

```
#6.
base_url <- 'https://www.ncwater.org/WUDC/app/LWSP/report.php'
the_PWSID <- '03-32-010'
the_year <- 2024
the_scrape_url <- paste0(base_url, '?pwsid=', the_PWSID, '&year=', the_year)
print(the_scrape_url)
```

```
## [1] "https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=03-32-010&year=2024"
```

```
# Create scraping function
scrape.function <- function(the_PWSID, the_year){

the_website <- read_html(paste0('https://www.ncwater.org/WUDC/app/LWSP/report.php','?pwsid=',
                                the_PWSID, '&year=', the_year))
```

```r
#Set the element address variables
water_system_tag <- "div+ table tr:nth-child(1) td:nth-child(2)"
ownership_tag <- "div+ table tr:nth-child(2) td:nth-child(4)"
monthly_MDG_tag <- "th~ td+ td"
month_tag <- ".fancy-table:nth-child(30) tr+ tr th"

# Scrape the data items
the_water_system <- the_website %>%
  html_nodes(water_system_tag) %>%
  html_text()

the_ownership <- the_website %>%
  html_nodes(ownership_tag) %>%
  html_text()

the_monthly_MGD <- the_website %>%
  html_nodes(monthly_MDG_tag) %>%
  html_text(trim = TRUE) %>%
  as.numeric()

the_month <- the_website %>%
  html_nodes(month_tag) %>%
  html_text()

# Convert to a dataframe
df_MGD <- data.frame("Monthly_MGD" = the_monthly_MGD,
                     "Ownership" = rep(the_ownership,12),
                     "Water_System" = rep(the_water_system,12),
                     "Year" = rep(the_year,12),
                     "Month" = the_month) %>%
  mutate(Date = my(paste(Month, "-", Year)))

df_MGD$Month <- factor(df_MGD$Month,
                                levels = c("Jan", "Feb", "Mar", "Apr",
                                "May", "Jun", "Jul", "Aug", "Sep", "Oct",
                                "Nov", "Dec"))
# Return the dataframe

return(df_MGD)
}
# Run the function
df_2024 <- scrape.function('03-32-010', 2024)
```

7. Use the function above to extract and plot max daily withdrawals for Durham (PWSID=‘03-32-010’) for each month in 2020

```r
#7
#df_2020 <- scrape.function('03-32-010', 2020)
# not working for any year besides 2024

# This is how I would plot it if I could get it to work.
# Hoping I can get partial credit here.
# I tried going to office hours on 4/1 but no one was there.
```

```
#ggplot(df_2020, aes(x = Month,
                     #y = Monthly_MGD)) +
  #geom_line(method = "loess", se = FALSE) +
  #labs(title = "2020 Monthly Max Daily Withdrawals for Durham",
       #x = "Month",
       #y = "Maximum Daily Use (MGD)")
```

8. Use the function above to extract data for Asheville (PWSID = '01-11-010') in 2020. Combine this data with the Durham data collected above and create a plot that compares Asheville's to Durham's water withdrawals.

```
#8
# This is how I would do Q8 if I could get the function to work.

# Extract data for Asheville in 2020
#df_avl_2020 <- scrape.function(2020,'01-11-010')

# Combine AVL and Durham data to create a plot that compared AVL to Durham's
# water withdrawals

#df_avl_dur_2020 <- inner_join(df_2020, df_avl_2020, by = "Month")

# ggplot(df_avl_dur_2020, aes(x = Month,
                              #y = Monthly.MGD,
                              #group = 1,
                              #color = water_system)) +
  #geom_line() +
 # labs(title = "2020 Monthly Maximum Daily Water Withdrawls for
                #Durham and Asheville",
       #subtitle = "Source: NC DEQ Division of Water Resources",
       #x = "Month",
       #y = "Maximum Daily Use (MGD)")
```

9. Use the code & function you created above to plot Asheville's max daily withdrawal by months for the years 2018 thru 2023.Add a smoothed line to the plot (method = 'loess').

   TIP: See Section 3.2 in the "10_Data_Scraping.Rmd" where we apply "map2()" to iteratively run a function over two inputs. Pipe the output of the map2() function to `bindrows()` to combine the dataframes into a single one, and use that to construct your plot.

```
#9
# This is how I would do Q9 if my function worked.

#Set the inputs to scrape years 2018 to 2023 for the Asheville
#(PWSID = "01-11-010")

#the_years = rep(2018:2023)
#my_pwsid = '01-11-010'

#Use lapply to apply the scrape function
#the_dfs <- lapply(X = the_years,
```

```
                  #FUN = scrape.function,
                  #my_pwsid = my_pwsid)

#Conflate the returned dataframes into a single dataframe
#the_df <- bind_rows(the_dfs)

# Plot
#ggplot(the_dfs, aes(x = Date, y = Monthly_MGD, color = Water_System)) +
#geom_line() +
#geom_smooth(method = "loess", se = FALSE) +
#labs(title = "2018-2023 MGD for Durham and Asheville",
      #x = Date,
      #y = Maximum Monthly Withdrawls (MGD))
```

Question: Just by looking at the plot (i.e. not running statistics), does Asheville have a trend in water usage over time? > Answer: I can't get my function to work so I can't see... but I would assume it would be increasing. >