Assignment 10: Data Scraping

Natalie Holsclaw

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(dplyr)
library(viridis)
```

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()
pwsid <- url %>% html_nodes("td tr:nth-child(1) td:nth-child(5)") %>% html_text()
owner <- url %>% html_nodes("div+ table tr:nth-child(2) td:nth-child(4)") %>%
  html_text()
daily_use <- url %>% html_nodes("th~ td+ td") %>% html_text()
```

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.)

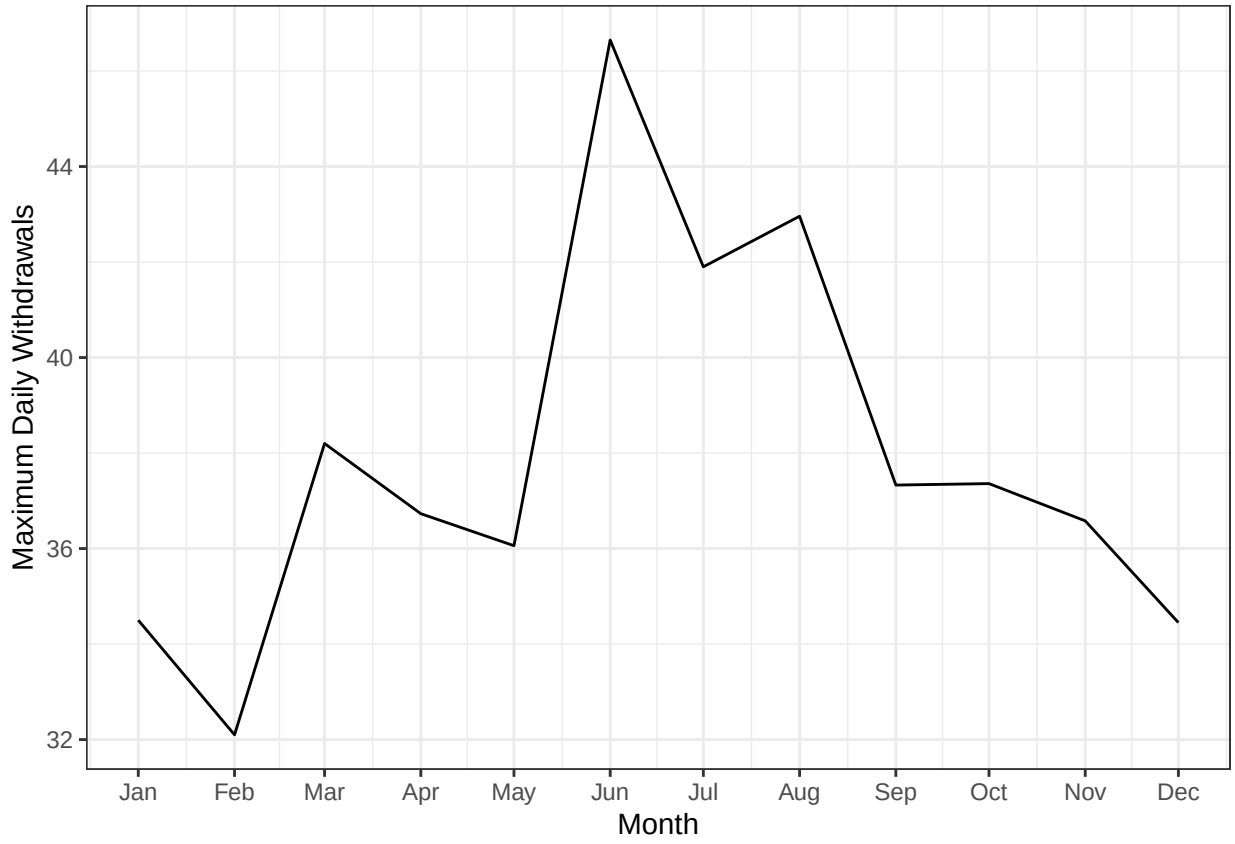
   > TIP: Use `rep()` to repeat a value when creating a dataframe.

   > 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.

```
#4
df <- data.frame("Month" = c("Jan", "May", "Sept", "Feb", "Jun", "Oct", "Mar",
                             "Jul", "Nov", "Apr", "Aug", "Dec"),
                 "Year" = rep(2024, 12),
                 "Max_daily_use" = as.numeric(daily_use),
                 "Owner" = rep(owner, 12),
                 "PSWID" = rep(pwsid, 12),
                 "Water_system" = rep(water_system, 12)) %>%
  mutate(Date = my(paste(Month,"-",Year))) %>%
  arrange(Date)
```

```r
#5
ggplot(df, aes(x = Date, y = Max_daily_use))+
  geom_line()+
  scale_x_date(date_labels = "%b", date_breaks = "1 month")+
  theme_bw()+
  labs(x = "Month", y = "Maximum Daily Withdrawals")
```



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

```r
#6.
func <- function(pwsid, the_year){
  website <- read_html(paste0("https://www.ncwater.org/WUDC/app/LWSP/report.php?",
                              'pwsid=', pwsid, '&year=', the_year))
  water_system <- website %>% html_nodes("div+ table tr:nth-child(1) td:nth-child(2)") %>%
  html_text()
pwsid2 <- website %>% html_nodes("td tr:nth-child(1) td:nth-child(5)") %>% html_text()
owner <- website %>% html_nodes("div+ table tr:nth-child(2) td:nth-child(4)") %>%
```

```
  html_text()
daily_use <- website %>% html_nodes("th~ td+ td") %>% html_text()

df <- data.frame("Month" = c("Jan", "May", "Sept", "Feb", "Jun", "Oct", "Mar",
                             "Jul", "Nov", "Apr", "Aug", "Dec"),
                 "Year" = rep(the_year, 12),
                 "Max_daily_use" = as.numeric(daily_use),
                 "Owner" = rep(owner, 12),
                 "PSWID" = rep(pwsid2, 12),
                 "Water_system" = rep(water_system, 12)) %>%
  mutate(Date = my(paste(Month,"-",Year))) %>%
  arrange(Date)

return(df)
}
```

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

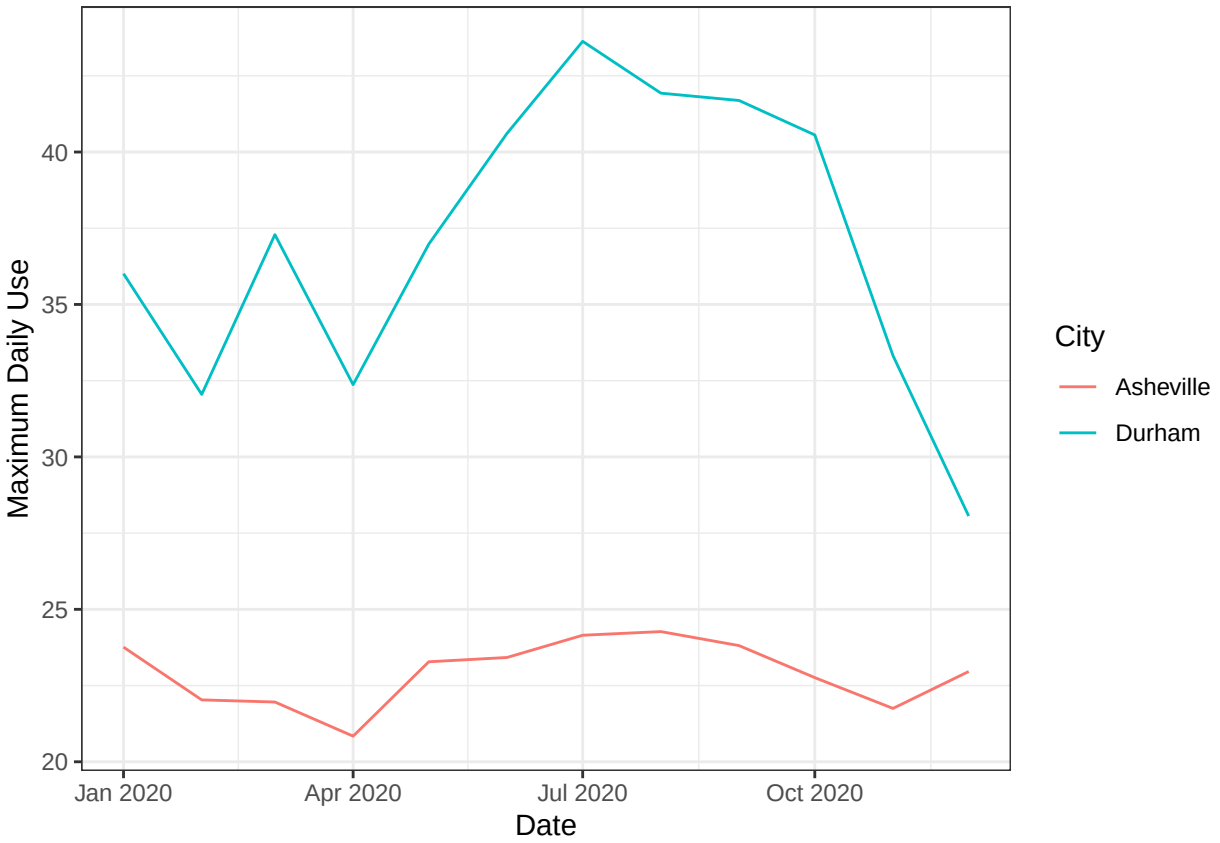
```
#7
df2 <- func('03-32-010', 2020)
view(df2)
```

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
df3 <- func('01-11-010', 2020)

df_combined <- bind_rows(df2,df3)

ggplot(df_combined, aes(x = Date, y = Max_daily_use, color = Water_system))+
  geom_line()+
  theme_bw()+
  labs(y = "Maximum Daily Use", color = "City")
```



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
pwsid3 <- '01-11-010'
pswids <- rep(pwsid3, 6)
the_years <- c(2018:2023)

df_ex <- cross2(pswids, the_years) %>%
  map(lift(func)) %>%
  bind_rows()
```

```
## Warning: `cross2()` was deprecated in purrr 1.0.0.
## i Please use `tidyr::expand_grid()` instead.
## i See <https://github.com/tidyverse/purrr/issues/768>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: `lift()` was deprecated in purrr 1.0.0.
```

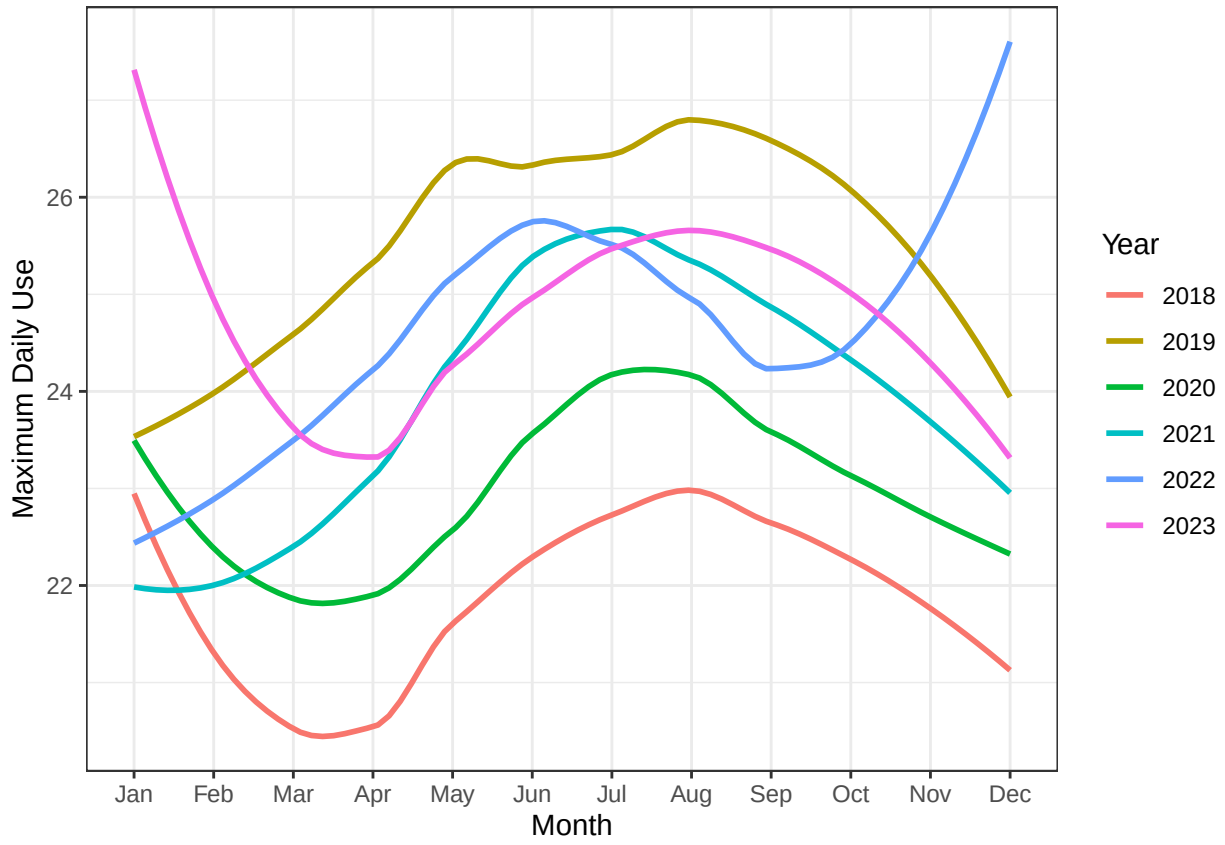
```
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

```r
# Extract month from Date column
df_ex$Month <- month(df_ex$Date, label = TRUE, abbr = TRUE)

# Ensure Year is a factor
df_ex$Year <- as.factor(df_ex$Year)

ggplot(df_ex, aes(x = Month, y = Max_daily_use, color = Year, group = Year))+
  geom_smooth(method="loess",se=FALSE)+
  labs(y = "Maximum Daily Use")+
  theme_bw()
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



Question: Just by looking at the plot (i.e. not running statistics), does Asheville have a trend in water usage over time? > Answer:No, water usage has gone up and down over the years with no clear trend but it does seem to peak in the summer months most years. >