Assignment 10: Data Scraping

Sujay Dhanagare

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

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

here()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

2. We will be scraping data from the NC DEQs Local Water Supply Planning website, specifically the Durham's 2023 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=2023

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

```r
#2

url <- "https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=03-32-010&year=2023"

webpage <- read_html(url)
```

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)".

```r
#3
# Scrape the "Water System Name"
water_system_name <- webpage %>%
  html_nodes(xpath = "//th[contains(text(), 'Water System Name')]/following-sibling::td") %>%
  html_text(trim = TRUE)

# Scrape the "PWSID"
pwsid <- webpage %>%
  html_nodes(xpath = "//th[contains(text(), 'PWSID')]/following-sibling::td") %>%
  html_text(trim = TRUE)

# Scrape the "Ownership"
ownership <- webpage %>%
  html_nodes(xpath = "//th[contains(text(), 'Ownership')]/following-sibling::td") %>%
  html_text(trim = TRUE)

# Extract rows containing "Maximum Day Use (MGD)" data
max_day_use_rows <- webpage %>%
  html_nodes(xpath = "//table[@class='fancy-table']//tr") %>%
  html_text(trim = TRUE) %>%
  grep(pattern = "Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec", value = TRUE)

# Parse the data into a clean list of months and their corresponding maximum day use values
parsed_data <- strsplit(max_day_use_rows, "\r\n\t\t")  # Split rows into individual elements
parsed_data <- unlist(parsed_data)  # Flatten the list into a single vector

# Identify valid months
valid_months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
```

```
# Extract months
months <- parsed_data[parsed_data %in% valid_months]

# Extract numeric values
values <- suppressWarnings(as.numeric(parsed_data[!parsed_data %in% valid_months]))

# Filter only "Max Day Use (MGD)" values
max_day_use_values <- values[seq(2, length(values), by = 2)]  # Every second value corresponds to Max D
```

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.

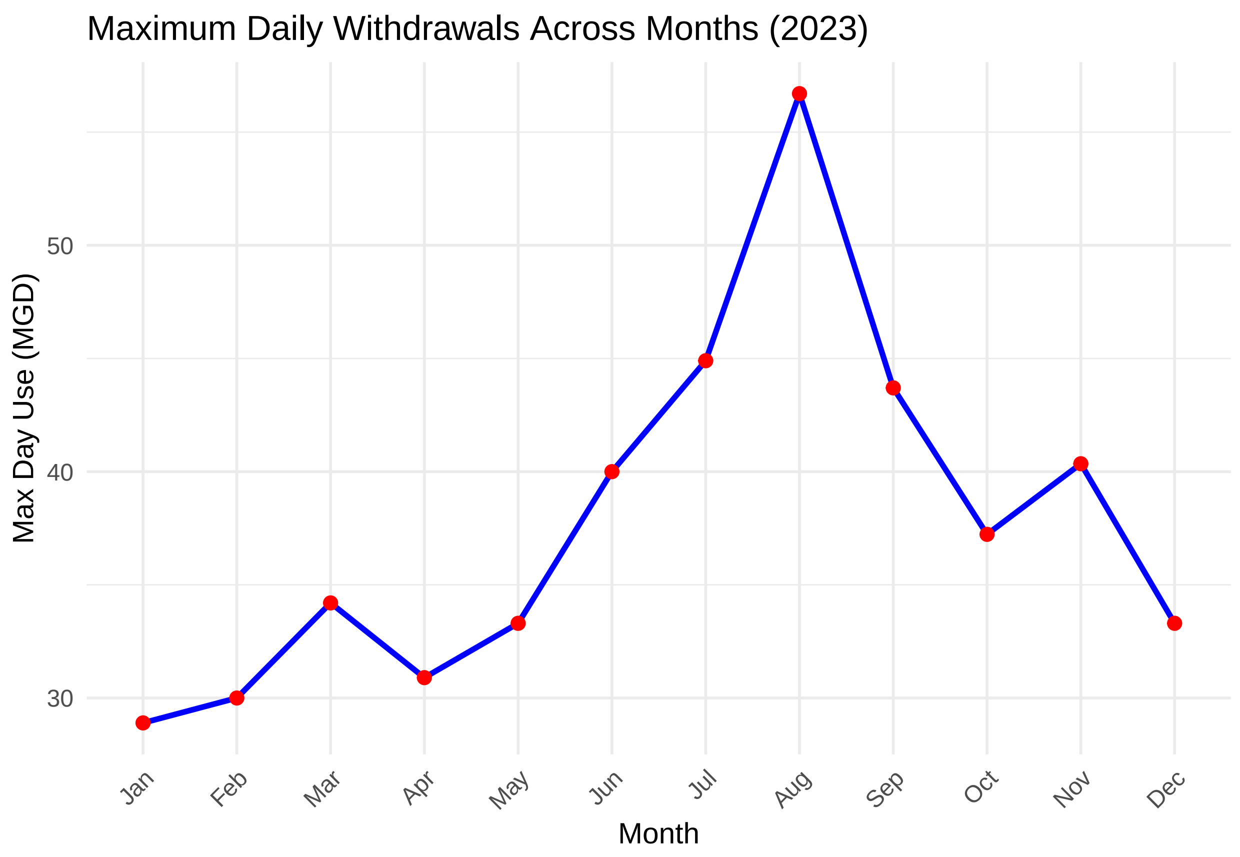
   NOTE: It's likely you won't be able to scrape the monthly widthrawal data in chronological order. You can overcome this by creating a month column manually assigning values in the order the data are scraped: "Jan", "May", "Sept", "Feb", etc… Or, you could scrape month values from the web page…

5. Create a line plot of the maximum daily withdrawals across the months for 2023, making sure, the months are presented in proper sequence.

```
#4
# Consolidate months and max day use values into a dataframe
raw_data <- data.frame(
  Month = rep(months, each = 2)[seq(2, length(values), by = 2)],  # Duplicate months align with Max Day
  Max_Day_Use_MGD = max_day_use_values
)

# Remove duplicates: take the first occurrence of each month
final_data <- raw_data[!duplicated(raw_data$Month), ]

# Add a Year column and create a Date column
final_data$Year <- 2023
final_data$Date <- ym(paste(final_data$Year, final_data$Month, sep = "-"))

# Reorder the dataframe chronologically
final_data <- final_data[order(match(final_data$Month, c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                                         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))), ]

# Print the final dataframe
print(final_data)
```

```
##    Month Max_Day_Use_MGD Year       Date
## 1    Jan           28.90 2023 2023-01-01
## 4    Feb           30.00 2023 2023-02-01
## 7    Mar           34.20 2023 2023-03-01
## 10   Apr           30.90 2023 2023-04-01
## 2    May           33.30 2023 2023-05-01
## 5    Jun           40.00 2023 2023-06-01
## 8    Jul           44.90 2023 2023-07-01
## 11   Aug           56.70 2023 2023-08-01
## 3    Sep           43.70 2023 2023-09-01
## 6    Oct           37.23 2023 2023-10-01
## 9    Nov           40.35 2023 2023-11-01
## 12   Dec           33.30 2023 2023-12-01
```

```r
#5
# Create the line plot
ggplot(final_data, aes(x = Month, y = Max_Day_Use_MGD, group = 1)) +
  geom_line(color = "blue", linewidth = 1) +   # Use `linewidth` instead of `size` for the line
  geom_point(color = "red", size = 2) +        # Points for each month
  labs(
    title = "Maximum Daily Withdrawals Across Months (2023)",
    x = "Month",
    y = "Max Day Use (MGD)"
  ) +
  theme_minimal() +                            # Use a minimalistic theme
  scale_x_discrete(limits = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



6. Note that the PWSID and the year appear in the web address for the page we scraped. Construct a function using your code above that can scrape data for any PWSID and year for which the NC DEQ has data, returning a dataframe. **Be sure to modify the code to reflect the year and site (pwsid) scraped**.

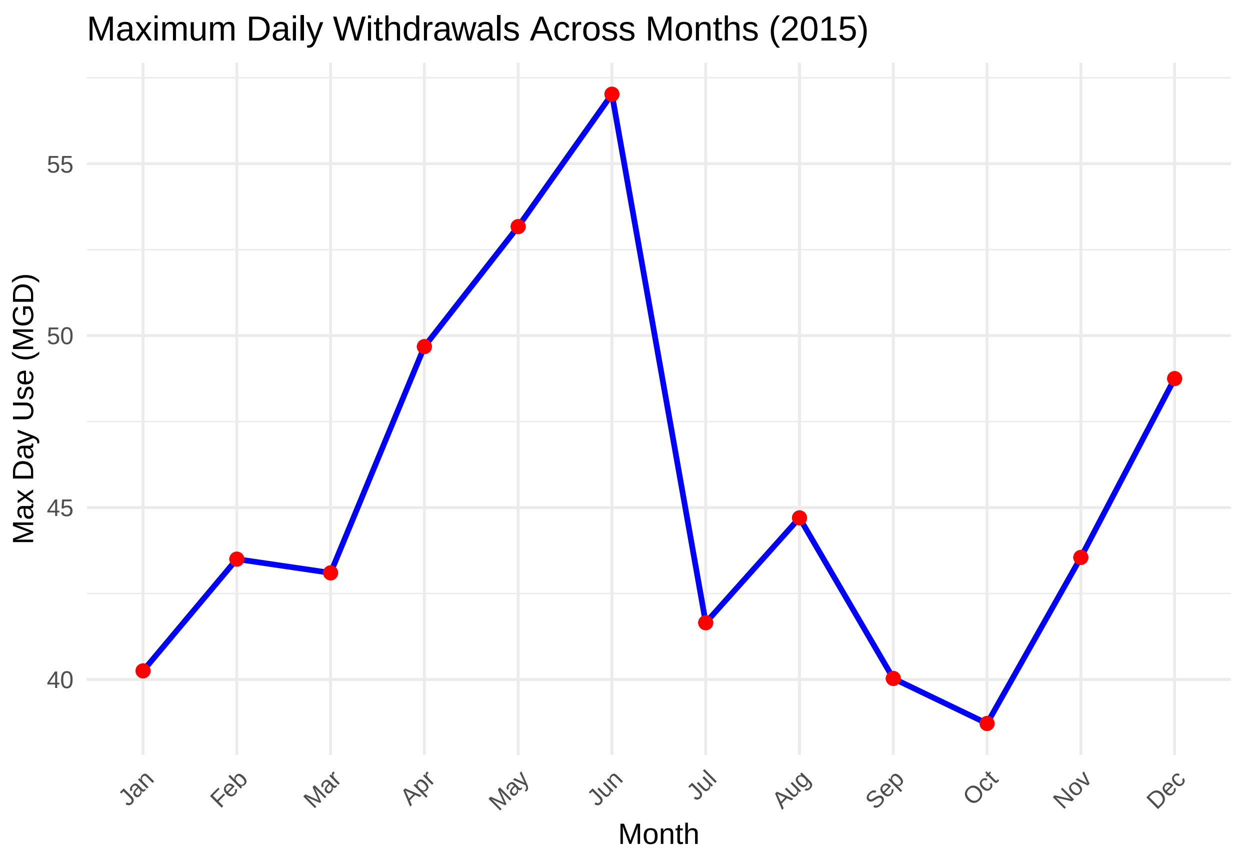
```r
#6.
scrape_max_day_use <- function(pwsid, year) {
  # Construct the URL dynamically
  url <- paste0("https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=", pwsid, "&year=", year)

  # Read the webpage content
  webpage <- read_html(url)

  # Extract rows containing "Maximum Day Use (MGD)" data
  max_day_use_rows <- webpage %>%
    html_nodes(xpath = "//table[@class='fancy-table']//tr") %>%
    html_text(trim = TRUE) %>%
    grep(pattern = "Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec", value = TRUE)

  # Parse the data into a clean list of months and their corresponding maximum day use values
  parsed_data <- strsplit(max_day_use_rows, "\r\n\t\t") %>% unlist()  # Split and flatten

  # Identify valid months
  valid_months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")

  # Extract months and numeric values
  months <- parsed_data[parsed_data %in% valid_months]
  values <- suppressWarnings(as.numeric(parsed_data[!parsed_data %in% valid_months]))

  # Filter only "Max Day Use (MGD)" values
  max_day_use_values <- values[seq(2, length(values), by = 2)]  # Every second value corresponds to Max

  # Create a dataframe
  raw_data <- data.frame(
    Month = rep(months, each = 2)[seq(2, length(values), by = 2)],  # Align months with Max Day Use
    Max_Day_Use_MGD = max_day_use_values
  )

  # Remove duplicates and finalize the dataframe
  final_data <- raw_data[!duplicated(raw_data$Month), ]
  final_data$Year <- year
  final_data$Date <- ym(paste(final_data$Year, final_data$Month, sep = "-"))

  # Reorder the dataframe chronologically
  final_data <- final_data[order(match(final_data$Month, valid_months)), ]

  return(final_data)
}
```

7. Use the function above to extract and plot max daily withdrawals for Durham (PWSID='03-32-010') for each month in 2015

```r
#7
final_data_2015 <- scrape_max_day_use("03-32-010", 2015)

# Print the dataframe for verification
print(final_data_2015)
```

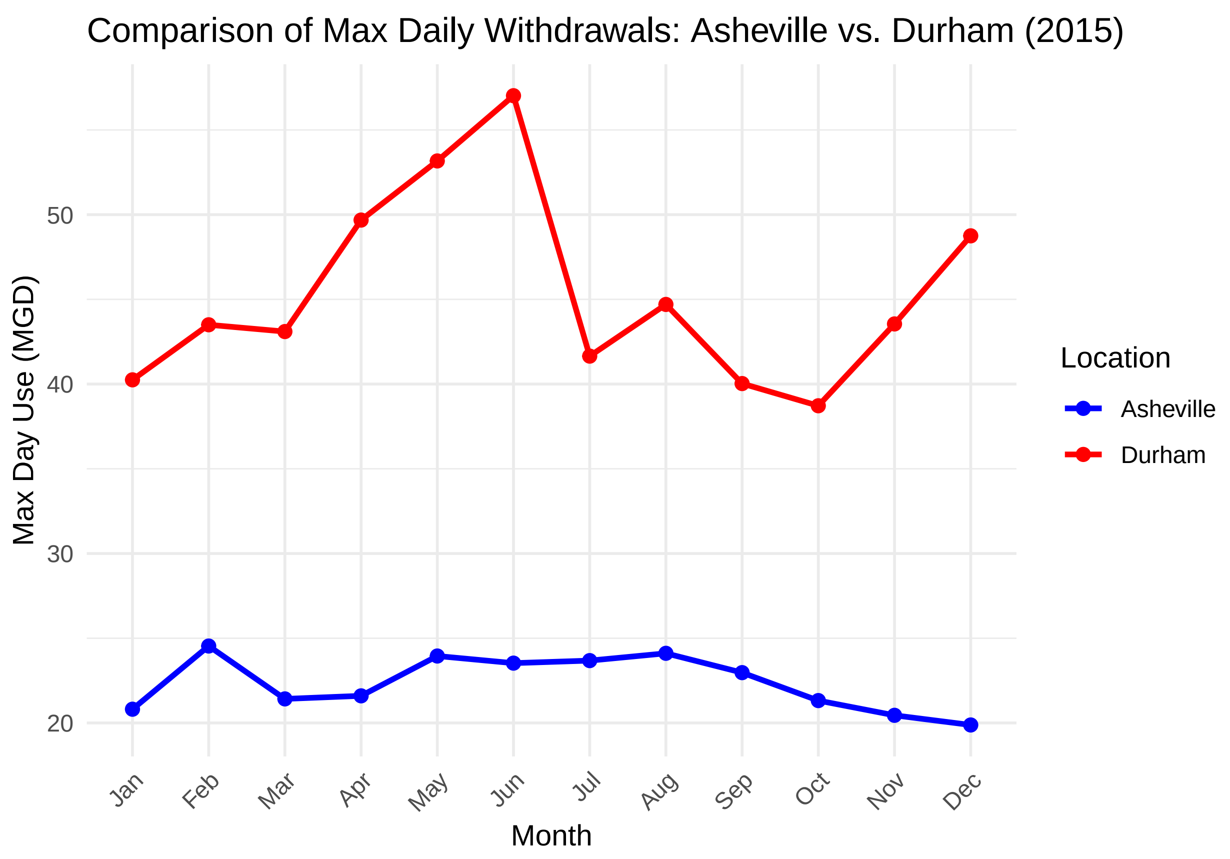
```
##    Month Max_Day_Use_MGD Year       Date
## 1    Jan           40.25 2015 2015-01-01
## 4    Feb           43.50 2015 2015-02-01
## 7    Mar           43.10 2015 2015-03-01
## 10   Apr           49.68 2015 2015-04-01
## 2    May           53.17 2015 2015-05-01
## 5    Jun           57.02 2015 2015-06-01
## 8    Jul           41.65 2015 2015-07-01
## 11   Aug           44.70 2015 2015-08-01
## 3    Sep           40.03 2015 2015-09-01
## 6    Oct           38.72 2015 2015-10-01
## 9    Nov           43.55 2015 2015-11-01
## 12   Dec           48.75 2015 2015-12-01
```

```r
# Plot the data
ggplot(final_data_2015, aes(x = Month, y = Max_Day_Use_MGD, group = 1)) +
  geom_line(color = "blue", linewidth = 1) +   # Use `linewidth` for the line
  geom_point(color = "red", size = 2) +       # Points for each month
  labs(
    title = "Maximum Daily Withdrawals Across Months (2015)",
    x = "Month",
    y = "Max Day Use (MGD)"
  ) +
  theme_minimal() +                           # Use a minimalistic theme
  scale_x_discrete(limits = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



8. Use the function above to extract data for Asheville (PWSID = 01-11-010) in 2015. Combine this data with the Durham data collected above and create a plot that compares Asheville's to Durham's water withdrawals.

```r
#8
# Extract data for Asheville (PWSID = 01-11-010) in 2015
asheville_data_2015 <- scrape_max_day_use("01-11-010", 2015)

# Extract data for Durham (PWSID = 03-32-010) in 2015
durham_data_2015 <- scrape_max_day_use("03-32-010", 2015)

# Add a column to identify the location
asheville_data_2015$Location <- "Asheville"
durham_data_2015$Location <- "Durham"

# Combine the two datasets
combined_data <- rbind(asheville_data_2015, durham_data_2015)

# Plot the combined data
ggplot(combined_data, aes(x = Month, y = Max_Day_Use_MGD, color = Location, group = Location)) +
  geom_line(linewidth = 1) +  # Line for each location
  geom_point(size = 2) +      # Points for each month
  labs(
    title = "Comparison of Max Daily Withdrawals: Asheville vs. Durham (2015)",
    x = "Month",
    y = "Max Day Use (MGD)"
  ) +
  theme_minimal() +
  scale_x_discrete(limits = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +  # Rotate month labels
  scale_color_manual(values = c("Asheville" = "blue", "Durham" = "red"))
```



9. Use the code & function you created above to plot Asheville's max daily withdrawal by months for the years 2018 thru 2022.Add a smoothed line to the plot (method = 'loess').

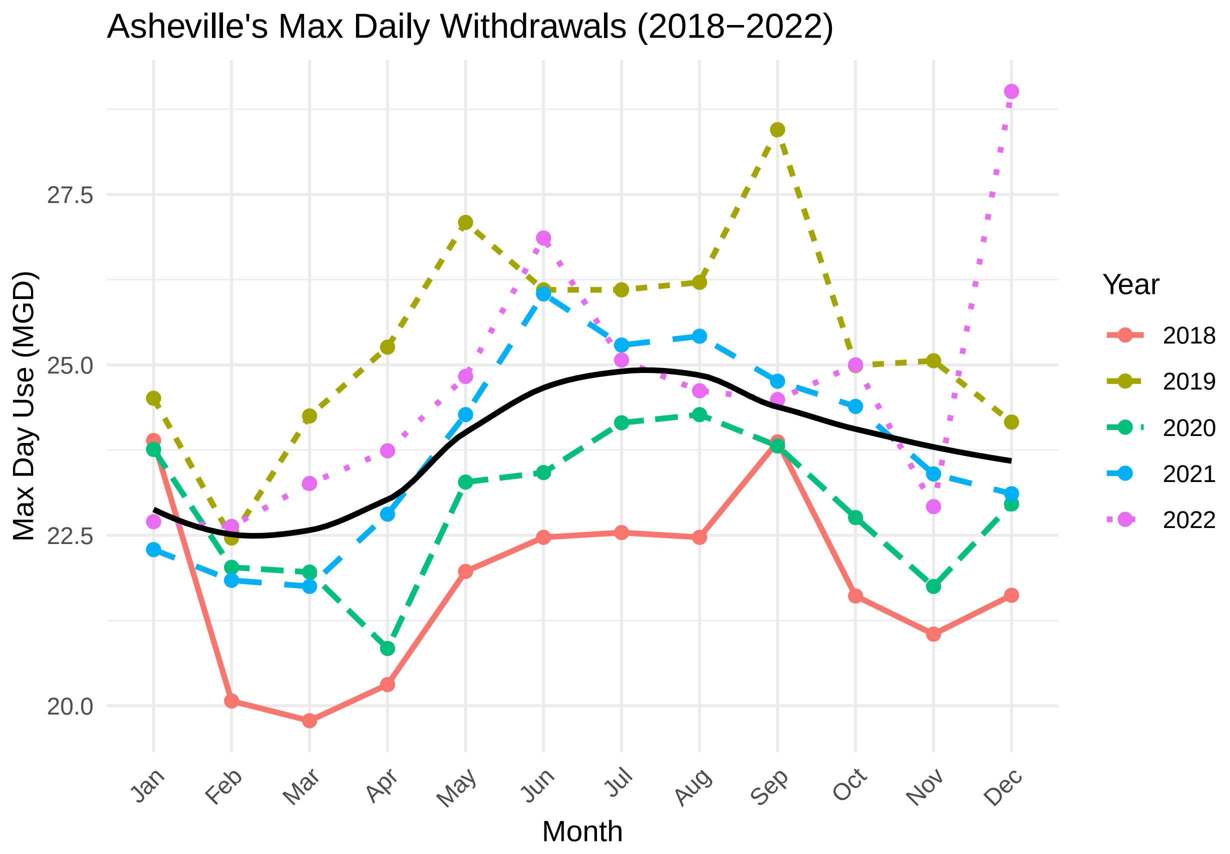
   TIP: See Section 3.2 in the "10_Data_Scraping.Rmd" where we apply "map2()" to iteratively run a function over two inputs. Pipe the output of the map2() function to `bindrows()` to combine the dataframes into a single one.

```r
#9
library(purrr)  # For map2()

# Define years and PWSID
years <- 2018:2022
pwsid <- "01-11-010"

# Use map2() to iteratively scrape data for each year
asheville_data <- map2(years, rep(pwsid, length(years)), ~ scrape_max_day_use(.y, .x)) %>%
  bind_rows()  # Combine all dataframes into one

# Print the combined dataframe
print(asheville_data)
```

```
##    Month Max_Day_Use_MGD Year       Date
## 1    Jan           23.89 2018 2018-01-01
## 2    Feb           20.07 2018 2018-02-01
## 3    Mar           19.78 2018 2018-03-01
## 4    Apr            20.31 2018 2018-04-01
## 5    May            21.97 2018 2018-05-01
## 6    Jun            22.47 2018 2018-06-01
## 7    Jul            22.54 2018 2018-07-01
## 8    Aug            22.47 2018 2018-08-01
## 9    Sep            23.87 2018 2018-09-01
## 10   Oct            21.61 2018 2018-10-01
## 11   Nov            21.05 2018 2018-11-01
## 12   Dec            21.62 2018 2018-12-01
## 13   Jan            24.51 2019 2019-01-01
## 14   Feb            22.46 2019 2019-02-01
## 15   Mar            24.25 2019 2019-03-01
## 16   Apr            25.26 2019 2019-04-01
## 17   May            27.09 2019 2019-05-01
## 18   Jun            26.10 2019 2019-06-01
## 19   Jul            26.10 2019 2019-07-01
## 20   Aug            26.21 2019 2019-08-01
## 21   Sep            28.45 2019 2019-09-01
## 22   Oct            24.99 2019 2019-10-01
## 23   Nov            25.06 2019 2019-11-01
## 24   Dec            24.16 2019 2019-12-01
## 25   Jan            23.76 2020 2020-01-01
## 26   Feb            22.03 2020 2020-02-01
## 27   Mar            21.96 2020 2020-03-01
## 28   Apr            20.84 2020 2020-04-01
## 29   May            23.28 2020 2020-05-01
## 30   Jun            23.42 2020 2020-06-01
## 31   Jul            24.15 2020 2020-07-01
## 32   Aug            24.27 2020 2020-08-01
## 33   Sep            23.81 2020 2020-09-01
## 34   Oct            22.76 2020 2020-10-01
## 35   Nov            21.75 2020 2020-11-01
## 36   Dec            22.96 2020 2020-12-01
## 37   Jan            22.29 2021 2021-01-01
## 38   Feb            21.84 2021 2021-02-01
## 39   Mar            21.75 2021 2021-03-01
## 40   Apr            22.81 2021 2021-04-01
## 41   May            24.27 2021 2021-05-01
## 42   Jun            26.04 2021 2021-06-01
## 43   Jul            25.29 2021 2021-07-01
## 44   Aug            25.42 2021 2021-08-01
## 45   Sep            24.76 2021 2021-09-01
## 46   Oct            24.39 2021 2021-10-01
## 47   Nov            23.40 2021 2021-11-01
## 48   Dec            23.11 2021 2021-12-01
## 49   Jan            22.70 2022 2022-01-01
## 50   Feb            22.63 2022 2022-02-01
## 51   Mar            23.26 2022 2022-03-01
## 52   Apr            23.74 2022 2022-04-01
## 53   May            24.83 2022 2022-05-01
## 54   Jun            26.86 2022 2022-06-01
## 55   Jul            25.07 2022 2022-07-01
## 56   Aug            24.62 2022 2022-08-01
## 57   Sep            24.49 2022 2022-09-01
## 58   Oct           25.00 2022 2022-10-01
## 59   Nov           22.92 2022 2022-11-01
## 60   Dec           29.01 2022 2022-12-01
```

```
# Plot the data with a smoothed line
ggplot(asheville_data, aes(x = Month, y = Max_Day_Use_MGD, group = Year, color = as.factor(Year))) +
  geom_line(aes(linetype = as.factor(Year)), linewidth = 1) +   # Lines for each year with linetype as
  geom_point(size = 2) +                                        # Points for each month
  geom_smooth(aes(group = 1), method = "loess", color = "black", se = FALSE) +  # Smoothed line
  labs(
    title = "Asheville's Max Daily Withdrawals (2018-2022)",
    x = "Month",
    y = "Max Day Use (MGD)",
    color = "Year",
    linetype = "Year"
  ) +
  theme_minimal() +
  scale_x_discrete(limits = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## ‘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:The plot reveals that Asheville’s maximum daily water usage generally increased over the years from 2018 to 2022 when compared on a monthly basis. However, 2019 stands out as abnormally high in several months, particularly during the summer, where usage spikes far exceed patterns observed in other years. Despite this anomaly, there is a clear upward trend in water usage across the months over the observed period.