Assignment 10: Data Scraping

Yangsen Zhang

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 Load required packages
library(tidyverse)
library(rvest)
library(here)

# Check working directory
here()
```

```
## [1] "/home/guest/EDE_Spring2026-yz"
```

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
- Set the year to 2024
- 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 <- "https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=03-32-010&year=2024"
webpage <- read_html(url)
```

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

- From the "1. System Information" section:
- Water system name
- Contact Person
- 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 "Reginald Hicks", the third "Municipality", and the last should be a vector of 12 numeric values (represented as strings)".

```
#3
get_value <- function(label){
  webpage %>%
    html_element(
      xpath = paste0(
        "//td[normalize-space()='", label, "']/following-sibling::td[1]"
      )
    ) %>%
    html_text(trim = TRUE)
}

system_name <- get_value("Water System Name:")

contact_person <- get_value("Contact Person:")

ownership <- get_value("Ownership:")

page_txt <- webpage %>%
  html_element("body") %>%
  html_text() %>%
  str_squish()

md_mat <- str_match_all(
  page_txt,
  "(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)\\s+([0-9]+\\.?[0-9]*)\\s+([0-9]+\\.?[0-9]*)"
)[[1]]

months_scraped <- md_mat[, 2]
```

```
max_day_use    <- md_mat[, 4]

system_name
```

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

```
contact_person
```

```
## [1] "Reginald Hicks"
```

```
ownership
```

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

```
max_day_use
```

```
##  [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...

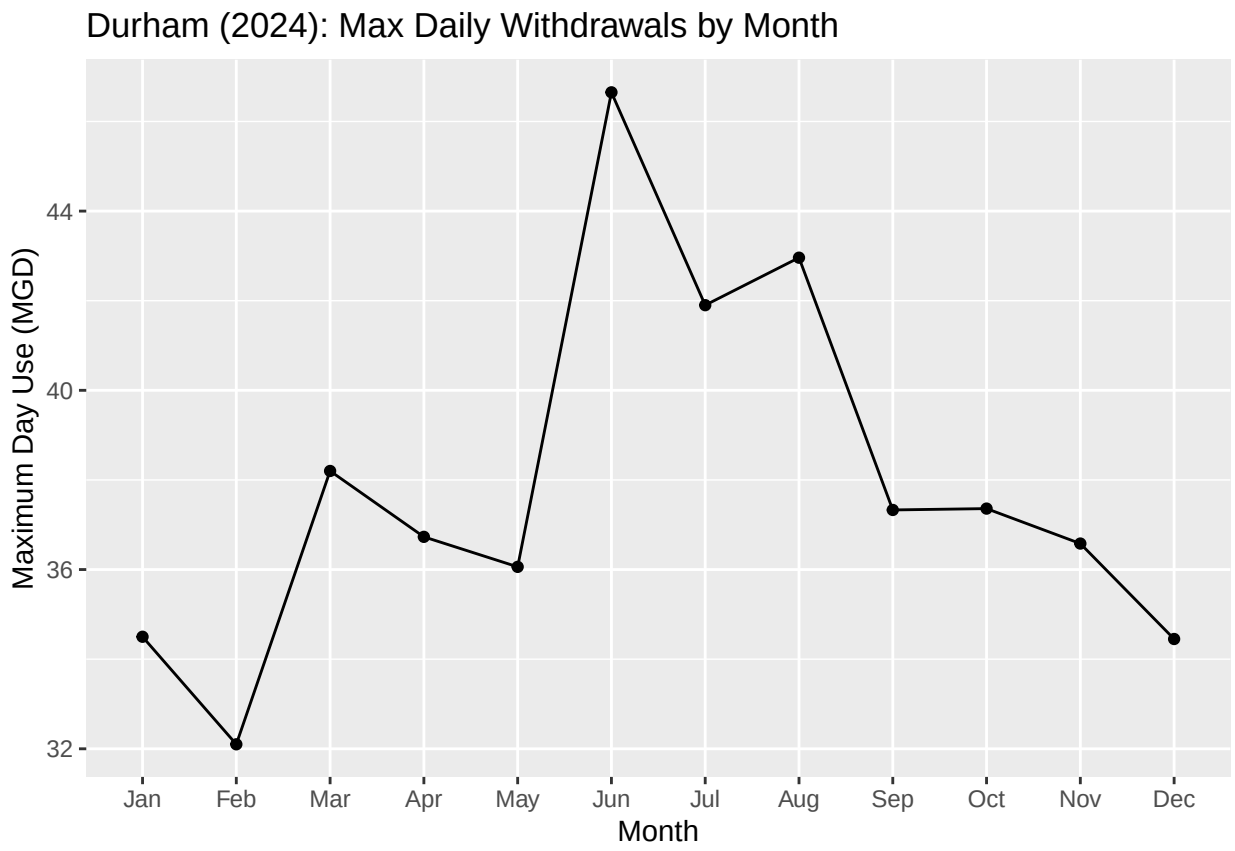
5. Create a line plot of the maximum daily withdrawals across the months for 2024, making sure, the months are presented in proper sequence.

```
#4 Convert scraped values into a dataframe (1 row per month)
durham_2024 <- tibble(
  month = months_scraped,
  max_day_mgd = as.numeric(max_day_use),
  system_name = rep(system_name, length(months_scraped)),
  contact_person = rep(contact_person, length(months_scraped)),
  ownership = rep(ownership, length(months_scraped)),
  year = 2024
) %>%
  mutate(
    month = factor(month, levels = month.abb, ordered = TRUE),
    month_num = match(as.character(month), month.abb),
    date = as.Date(sprintf("%d-%02d-01", year, month_num))
  ) %>%
  arrange(month)

durham_2024
```

```
## # A tibble: 12 x 8
##    month max_day_mgd system_name contact_person ownership     year month_num
##    <ord>       <dbl> <chr>       <chr>          <chr>        <dbl>     <int>
##  1 Jan          34.5 Durham      Reginald Hicks Municipality  2024         1
##  2 Feb          32.1 Durham      Reginald Hicks Municipality  2024         2
##  3 Mar          38.2 Durham      Reginald Hicks Municipality  2024         3
##  4 Apr          36.7 Durham      Reginald Hicks Municipality  2024         4
##  5 May          36.1 Durham      Reginald Hicks Municipality  2024         5
##  6 Jun          46.6 Durham      Reginald Hicks Municipality  2024         6
##  7 Jul          41.9 Durham      Reginald Hicks Municipality  2024         7
##  8 Aug          43.0 Durham      Reginald Hicks Municipality  2024         8
##  9 Sep          37.3 Durham      Reginald Hicks Municipality  2024         9
## 10 Oct          37.4 Durham      Reginald Hicks Municipality  2024        10
## 11 Nov          36.6 Durham      Reginald Hicks Municipality  2024        11
## 12 Dec          34.4 Durham      Reginald Hicks Municipality  2024        12
## # i 1 more variable: date <date>
```

```r
#5 Line plot (months in proper sequence)
ggplot(durham_2024, aes(x = month, y = max_day_mgd, group = 1)) +
  geom_line() +
  geom_point() +
  labs(
    x = "Month",
    y = "Maximum Day Use (MGD)",
    title = "Durham (2024): Max Daily Withdrawals by Month"
  )
```



6. Note that the PWSID and the year appear in the web address for the page we scraped. Construct a function with two inputs, "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 Function to scrape LWSP given PWSID and year
scrape_lwsp <- function(PWSID, year){

  url <- paste0(
    "https://www.ncwater.org/WUDC/app/LWSP/report.php?pwsid=",
    PWSID, "&year=", year
  )

  webpage <- read_html(url)

  # helper for label-value cells using XPath
  get_value <- function(label){
    webpage %>%
      html_element(xpath = paste0("//td[normalize-space()='", label, "']/following-sibling::td[1]")) %>%
      html_text(trim = TRUE)
  }

  system_name <- get_value("Water System Name:")
  contact_person <- get_value("Contact Person:")
  ownership <- get_value("Ownership:")

  # extract monthly max day use from body text (base R)
  page_txt <- webpage %>%
    html_element("body") %>%
    html_text(trim = TRUE)
  page_txt <- gsub("\\s+", " ", page_txt)

  md_mat <- regmatches(
    page_txt,
    gregexpr(
      "(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)\\s+([0-9]+\\.?[0-9]*)\\s+([0-9]+\\.?[0-9]*)",
      page_txt
    )
  )[[1]]

  months_scraped <- sub("^([A-Za-z]{3}).*$", "\\1", md_mat)
  max_day_use <- sub(
    "^(?:[A-Za-z]{3})\\s+(?:[0-9]+\\.?[0-9]*)\\s+([0-9]+\\.?[0-9]*).*$",
    "\\1",
    md_mat
  )

  df <- tibble(
    pwsid = PWSID,
    year = year,
```

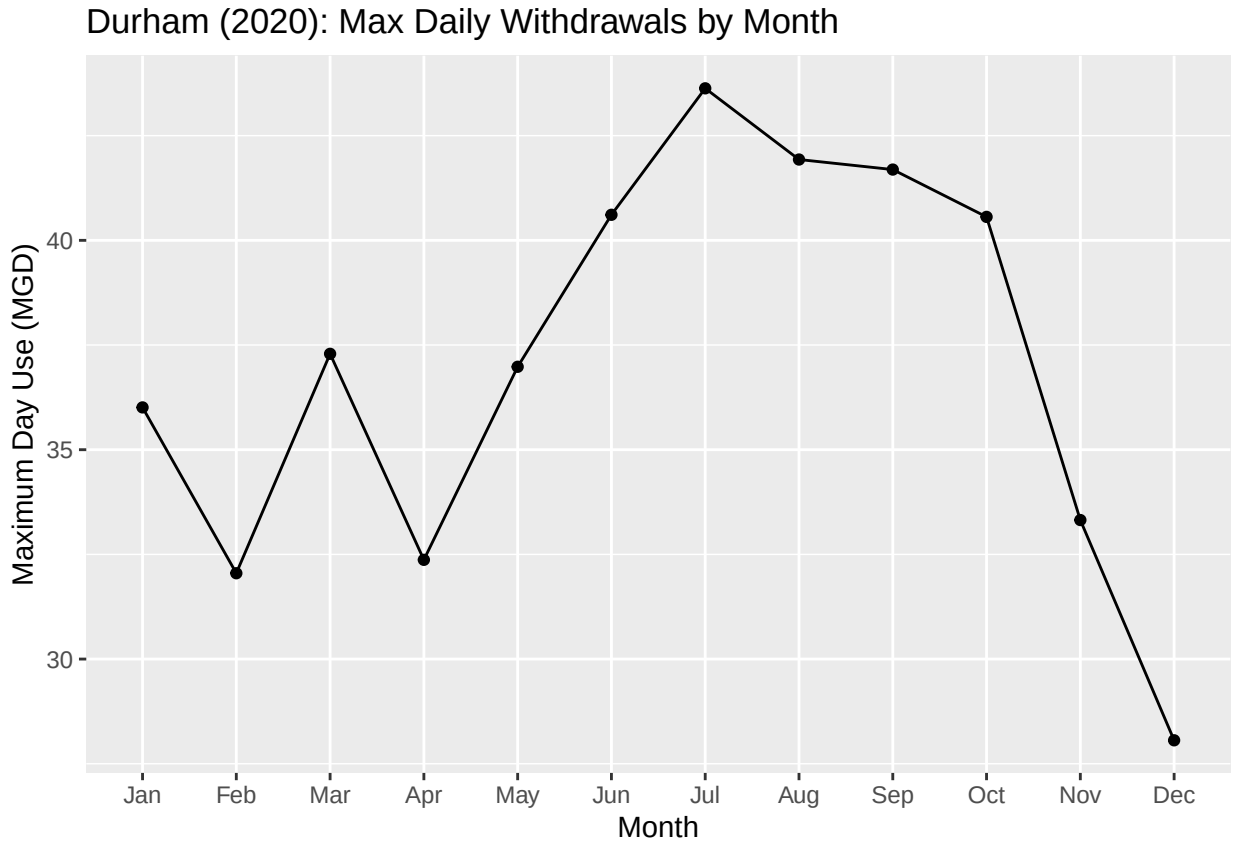
```
    month = months_scraped,
    max_day_mgd = as.numeric(max_day_use),
    system_name = rep(system_name, length(months_scraped)),
    contact_person = rep(contact_person, length(months_scraped)),
    ownership = rep(ownership, length(months_scraped))
  ) %>%
    mutate(
      month = factor(month, levels = month.abb, ordered = TRUE),
      month_num = match(as.character(month), month.abb),
      date = as.Date(sprintf("%d-%02d-01", year, month_num))
    ) %>%
    arrange(month)

  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. Then show the structure of the dataframe with either the `str()` or the `glimpse()` function.

```
#7 Durham 2020
durham_2020 <- scrape_lwsp(PWSID = "03-32-010", year = 2020)

ggplot(durham_2020, aes(x = month, y = max_day_mgd, group = 1)) +
  geom_line() +
  geom_point() +
  labs(
    x = "Month",
    y = "Maximum Day Use (MGD)",
    title = "Durham (2020): Max Daily Withdrawals by Month"
  )
```




```
glimpse(durham_2020)
```

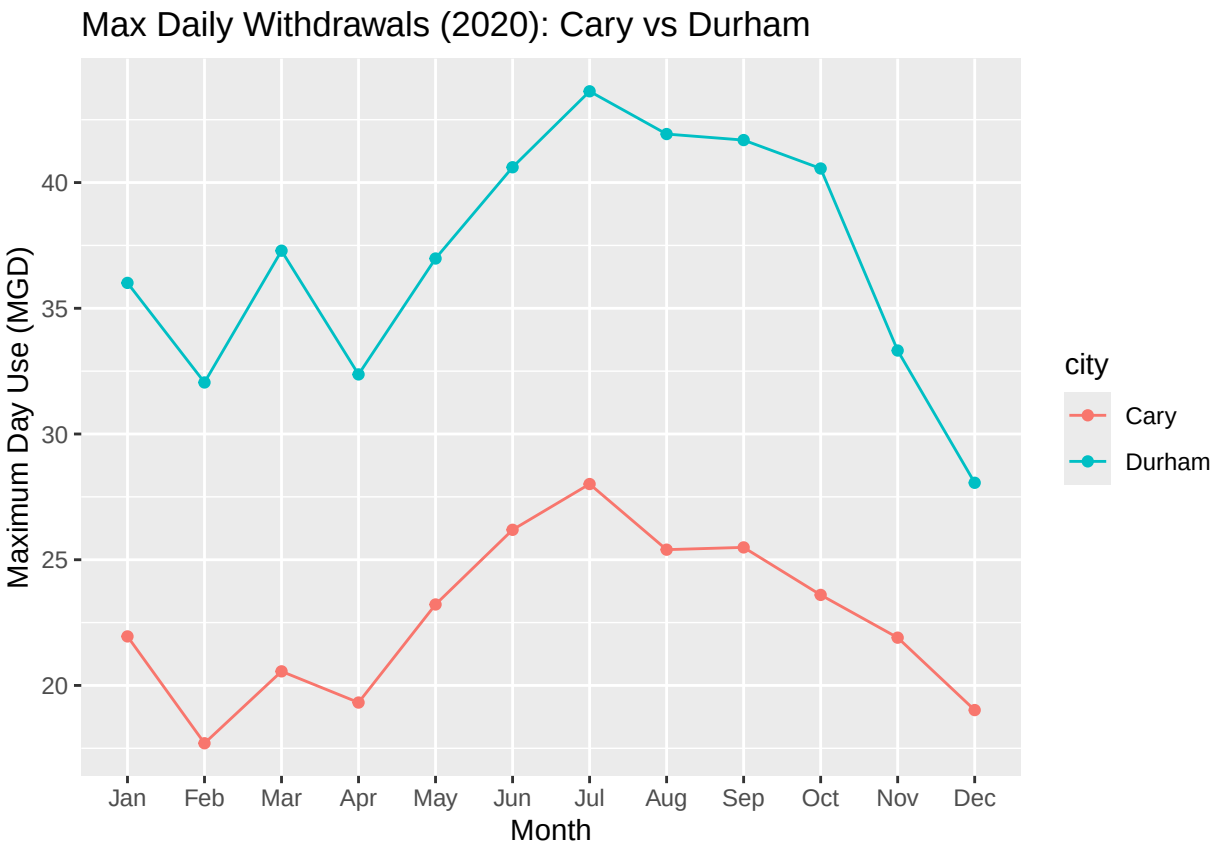
```
## Rows: 12
## Columns: 9
## $ pwsid          <chr> "03-32-010", "03-32-010", "03-32-010", "03-32-010", "03~
## $ year           <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2~
## $ month          <ord> Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, ~
## $ max_day_mgd    <dbl> 36.01, 32.05, 37.29, 32.37, 36.98, 40.61, 43.63, 41.93,~
## $ system_name    <chr> "Durham", "Durham", "Durham", "Durham", "Durham", "Durh~
## $ contact_person <chr> "Reginald Hicks", "Reginald Hicks", "Reginald Hicks", "~
## $ ownership      <chr> "Municipality", "Municipality", "Municipality", "Munici~
## $ month_num      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
## $ date           <date> 2020-01-01, 2020-02-01, 2020-03-01, 2020-04-01, 2020-05~
```

8. Use the function above to extract data for Cary (PWSID = '03-92-020') in 2020. Combine this data with the Durham data collected above and create a plot that compares Cary's to Durham's water withdrawals.

```
#8 Cary 2020 + compare with Durham 2020
cary_2020 <- scrape_lwsp(PWSID = "03-92-020", year = 2020)

compare_2020 <- bind_rows(
  durham_2020 %>% mutate(city = "Durham"),
  cary_2020 %>% mutate(city = "Cary")
)
```

```
ggplot(compare_2020, aes(x = month, y = max_day_mgd, color = city, group = city)) +
  geom_line() +
  geom_point() +
  labs(
    x = "Month",
    y = "Maximum Day Use (MGD)",
    title = "Max Daily Withdrawals (2020): Cary vs Durham"
  )
```



9. Use the code & function you created above to plot Cary'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 "06_Data_Scraping.Rmd" where we apply "map2()" to iteratively run a function over two inputs (where one of the two inputs does not change). Pipe the output of the map2() function to `bind_rows()` to combine the dataframes into a single one, and use that to construct your plot.

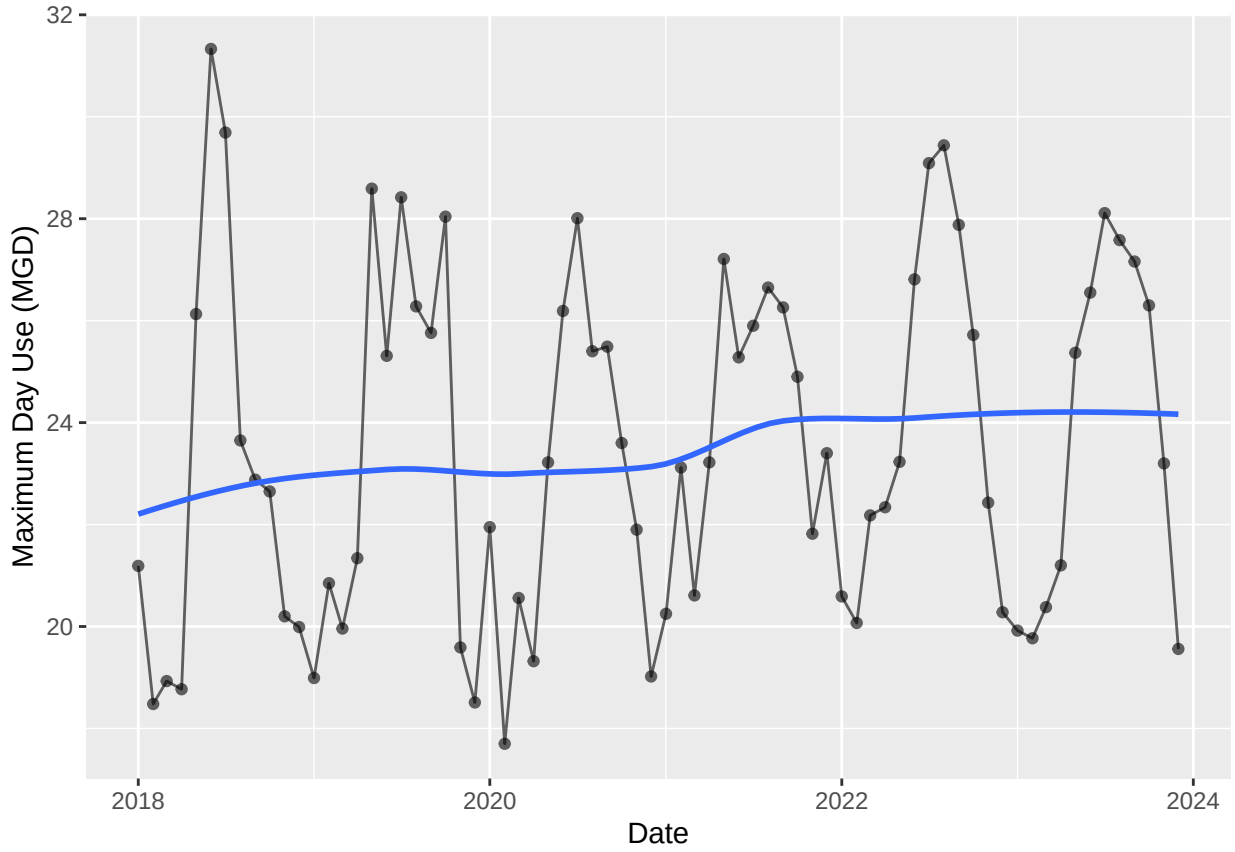
```
#9  Cary 2018-2023 with loess trend
years <- 2018:2023

cary_multi <- map2(
  .x = years,
  .y = rep("03-92-020", length(years)),
  ~ scrape_lwsp(PWSID = .y, year = .x)
```

```
) %>% bind_rows()

ggplot(cary_multi, aes(x = date, y = max_day_mgd)) +
  geom_line(alpha = 0.6) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(
    x = "Date",
    y = "Maximum Day Use (MGD)",
  )
```

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



Question: Just by looking at the plot (i.e. not running statistics), does Cary have a trend in water usage over time? > Answer:Yes, Cary's maximum daily withdrawals show a slight upward trend from 2018 to about 2021/2022, and then roughly flat through 2023.