Assignment 09: Data Scraping

Melissa Merritt

OVERVIEW

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

Directions

1. Rename this file `<FirstLast>_A09_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 to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.

Set up

1. Set up your session:

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

```
# 1
getwd()  #use to check working directory
```

```
## [1] "/home/guest/Documents/EDA-Fall2022/Assignments"
```

```
library(tidyverse)
library(rvest)
library(lubridate)
library(viridis)
library(scales)

# Use the library() functions to load the
# packages that we need.

mytheme <- theme_bw(base_size = 10) + theme(axis.text = element_text(color = "black"),
    legend.position = "top", axis.text.x = element_text(angle = 60,
        hjust = 1))
theme_set(mytheme)

# I use the theme_bw() and theme() functions
# to define my theme. The theme_set()
# function allows me to set my theme for the
# entire document.
```

2. We will be scraping data from the NC DEQs Local Water Supply Planning website, specifically the Durham's 2021 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=2021

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=2021")

# Using the read_html() function allows us
# to read in the url as a webpage object.
```

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

- From the "1. System Information" section:
- Water system name
- PSWID
- Ownership
- From the "3. Water Supply Sources" section:
- Maximum Daily 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), with the first value being "27.6400".

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

```
## [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"
```

```
max.withdrawals.mgd <- URL %>%
    html_nodes("th~ td+ td") %>%
    html_text()
max.withdrawals.mgd
```

```
##  [1] "27.6400" "41.7900" "36.7200" "27.9700" "37.9500" "42.2400" "30.5400"
##  [8] "43.6200" "31.2800" "33.7600" "46.0800" "29.7800"
```

```
# To scrape the data we can use the pipe
# function with the URL, and then
# html_nodes() function with the location on
# the website, and then use the html_text()
# function to receive the test values.
```

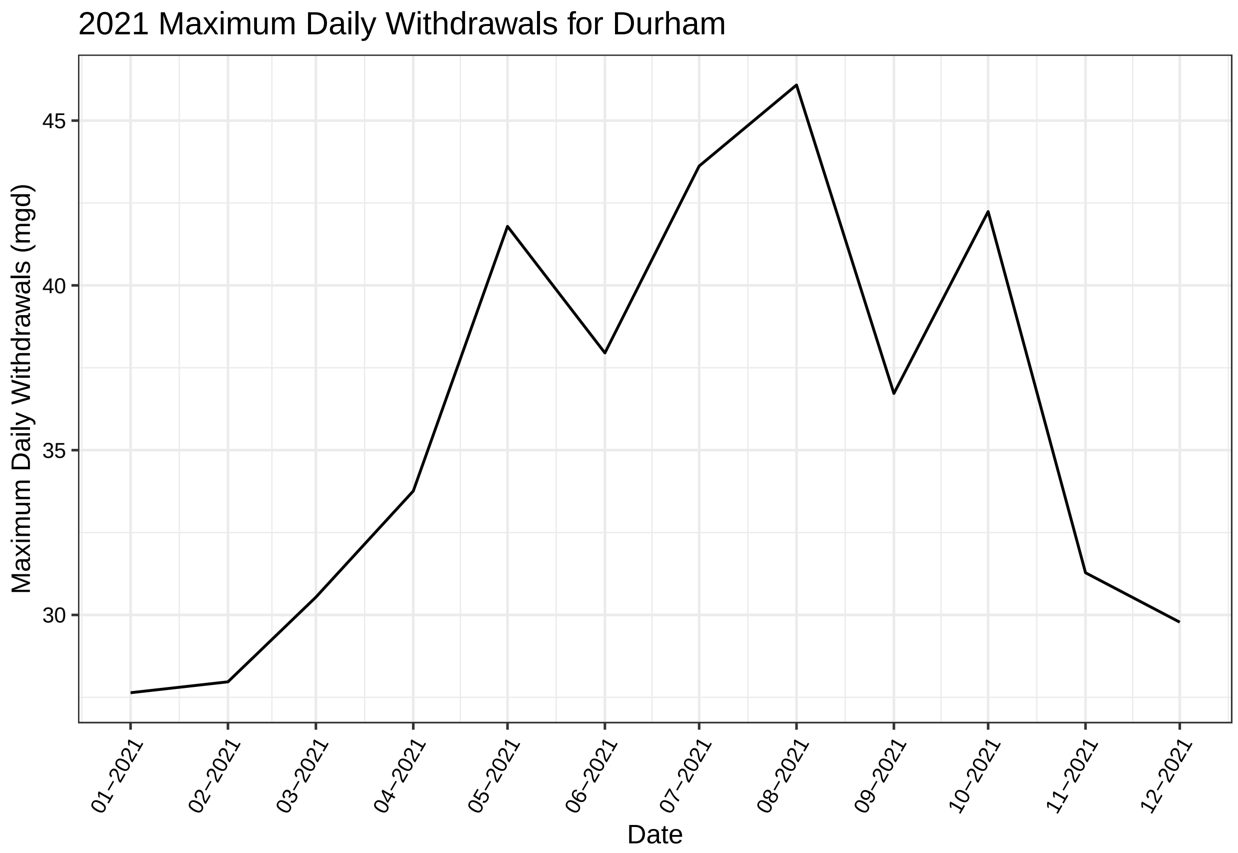
4. Convert your scraped data into a dataframe. This dataframe should have a column for each of the 4 variables scraped and a row for the month corresponding to the withdrawal data. Also add a Date column that includes your month and year in data format. (Feel free to add a Year column too, if you wish.)

   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 2021

```
# 4
withdrawal_df <- data.frame(Month = c("Jan", "May",
    "Sept", "Feb", "Jun", "Oct", "Mar", "Jul",
    "Nov", "Apr", "Aug", "Dec"), Year = rep(2021),
    Maximum.Daily.Use.mgd = as.numeric(max.withdrawals.mgd)) %>%
    mutate(Water.System = !!water.system.name,
        PWSID = !!pwsid, Ownership = !!ownership,
        Date = my(paste(Month, "-", Year)))

# To create the data frame with these
# values, we can use the data.frame()
# function. The first step in creating this
# function is to set the months up, and to
# do this we need to manually add the months
# to match the data from the website. Then
# we need to set the year using the rep()
# function. We can also set the max daily
# use as a numeric using the as.numeric()
# function for the values. Using the pipe
# and mutate functions we can add the rest
# of the columns. The !! points to the
# variable, and the my() function can create
# a new column with the month and year.

# 5
Max.withdrawals.plot <- ggplot(withdrawal_df,
    aes(x = Date, y = Maximum.Daily.Use.mgd)) +
    geom_line() + ylab("Maximum Daily Withdrawals (mgd)") +
    scale_x_date(date_breaks = "1 month", labels = date_format("%m-%Y")) +
    labs(title = "2021 Maximum Daily Withdrawals for Durham")
print(Max.withdrawals.plot)
```



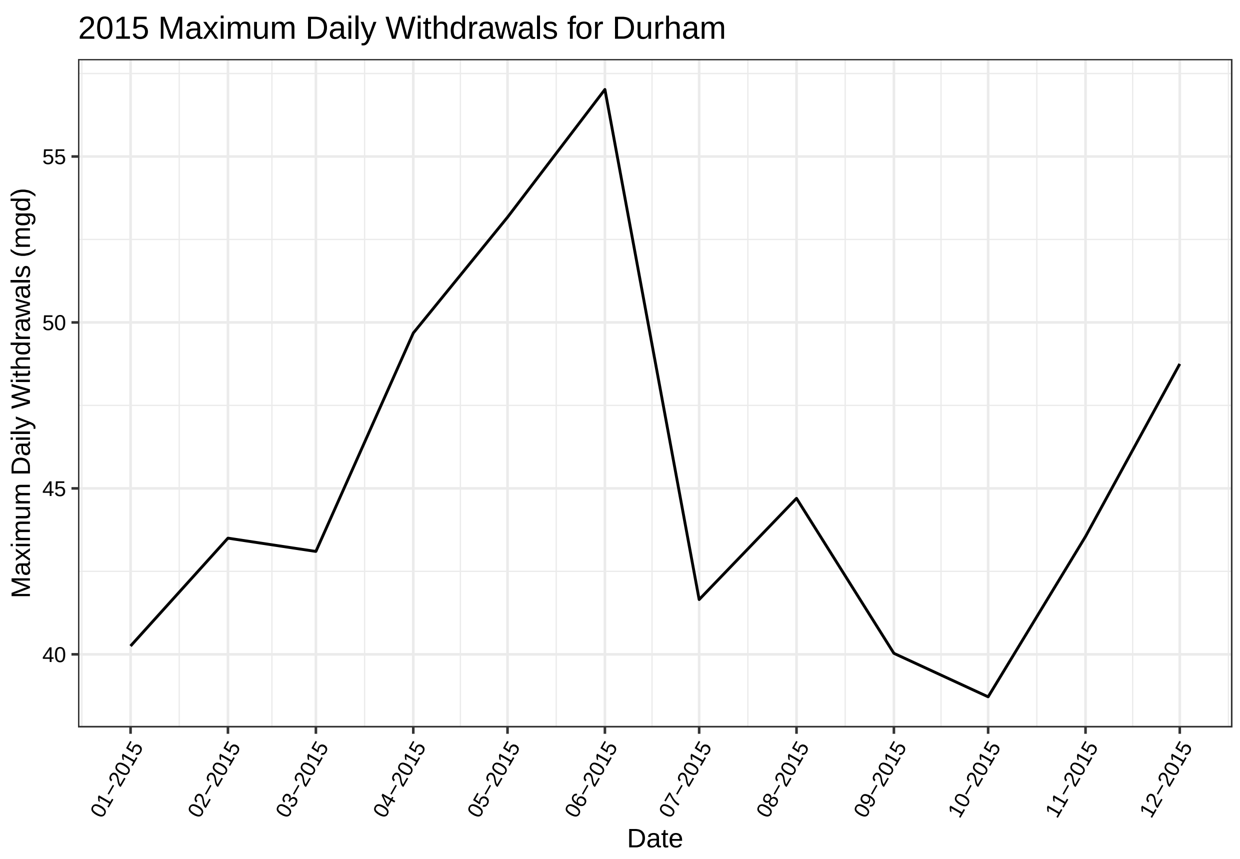
```
# To create this plot we can use the
# ggplot() function. Within this function we
# can define the data frame we are using as
# well as the x and y values using aes(). We
# can specify that we want a line graph
# using the function geom_line(). Using the
# ylab() we can label the y-axis. The
# scale_x_date() function allows us to
# include each month in the y-axis labeling
# to make the graph more clear. Finally, we
# use the print() function to view the
# graph.
```

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. **Be sure to modify the code to reflect the year and site (pwsid) scraped**.

```
# 6.
scrape.it <- function(the_year, pwsid) {

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

    water.system.name_tag <- the_website %>%
        html_nodes("div+ table tr:nth-child(1) td:nth-child(2)") %>%
        html_text()
    ownership_tag <- the_website %>%
        html_nodes("div+ table tr:nth-child(2) td:nth-child(4)") %>%
        html_text()
    max.withdrawals.mgd_tag <- the_website %>%
        html_nodes("th~ td+ td") %>%
        html_text()

    withdrawal_df <- data.frame(Month = c("Jan",
        "May", "Sept", "Feb", "Jun", "Oct", "Mar",
        "Jul", "Nov", "Apr", "Aug", "Dec"), Year = rep(the_year,
        12), PWSID = pwsid, Maximum.Daily.Use.mgd = as.numeric(max.withdrawals.mgd_tag)) %>%
        mutate(Water.System = !!water.system.name_tag,
            Ownership = !!ownership_tag, Date = my(paste(Month,
                "-", Year)))

    return(withdrawal_df)
}

# In order to create the scrape.it function
# we first set the variables that are
# changing in the function() brackets. Then
# we need to define the function first using
# the read_html() function to read in the
# website URL that has the variables defined
# as the ones we set. Then we can copy the
# scraping and dataframe functions from
# above. The differences were just that we
# needed to make both pwsid and the_year
# apart of the data.frame() function instead
# of defining it as a variable with the !!.
# The last step is to use the return()
# function to define the new function. We
# can use the next question to ensure the
# scrape.it() function is working correctly.
```

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

```
# 7

Durham2015_df <- scrape.it(2015, "03-32-010")
view(Durham2015_df)

# Using the scrape.it() function created
# above, we can create a new dataframe with
# the data for Durham in 2015. We can use
# the view() function to see the dataframe.

Max.withdrawals.plot_Durham2015 <- ggplot(Durham2015_df,
    aes(x = Date, y = Maximum.Daily.Use.mgd)) +
    geom_line() + ylab("Maximum Daily Withdrawals (mgd)") +
    scale_x_date(date_breaks = "1 month", labels = date_format("%m-%Y")) +
    labs(title = "2015 Maximum Daily Withdrawals for Durham")
print(Max.withdrawals.plot_Durham2015)
```

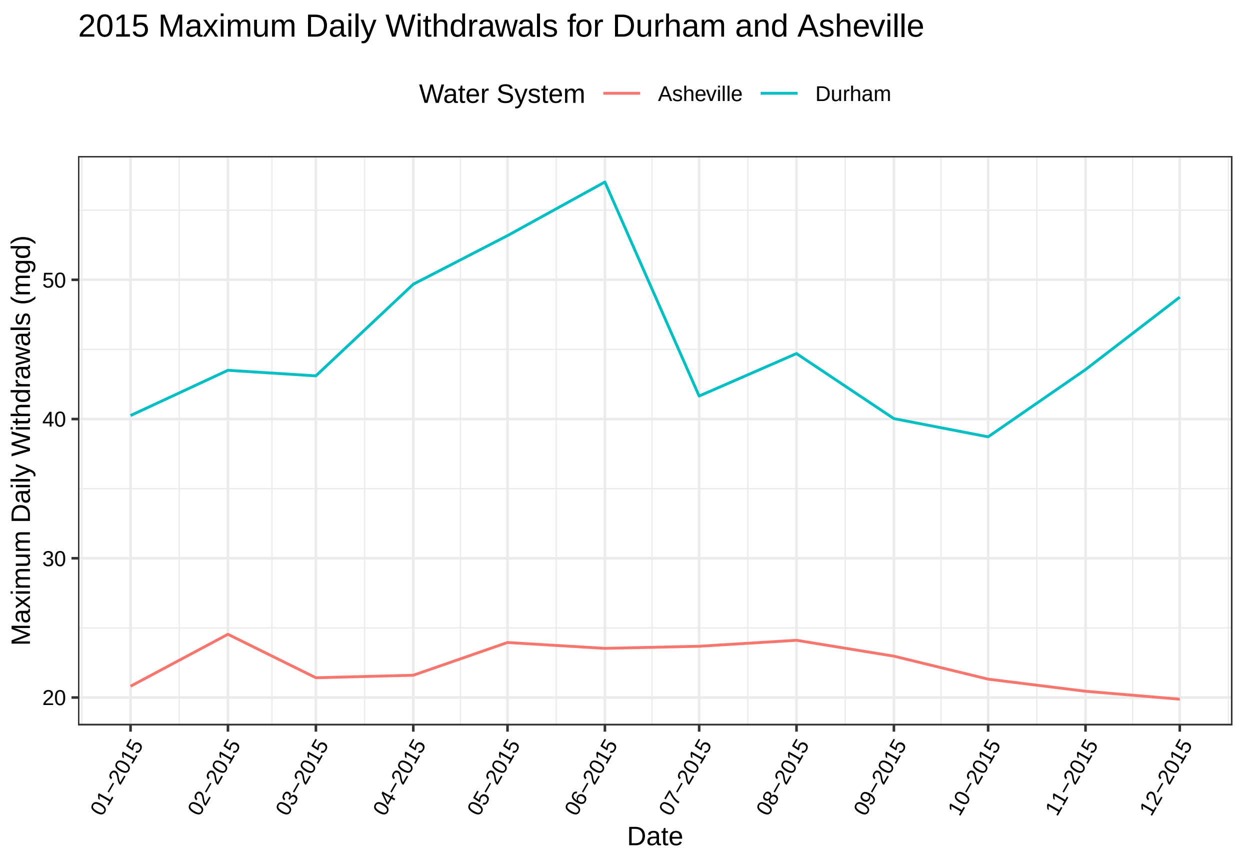


```r
# To plot the values we can use the ggplot()
# function to define the dataframe we are
# using and the x and y values. We can use
# the geom_line() function to specify that
# we want a line plot. Using the ylab()
# function allows us to label the y axis.
# The scale_x_date() function allows us to
# include each month in the y-axis labeling
# to make the graph more clear. Finally, we
# use the print() function to view the
# graph.
```

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

Asheville2015_df <- scrape.it(2015, "01-11-010")
view(Asheville2015_df)

# Using the scrape.it() function created
# above, we can create a new dataframe with
# the data for Asheville in 2015. We can use
# the view() function to see the dataframe.

Combined_df <- rbind(Asheville2015_df, Durham2015_df)
```

```r
# To combine the two dataframes we can use
# the rbind() function which combines the
# rows.

Max.withdrawals.plot_DurhamAsheville2015 <- ggplot(Combined_df,
    aes(x = Date, y = Maximum.Daily.Use.mgd, color = Water.System)) +
    geom_line() + scale_x_date(date_breaks = "1 month",
    labels = date_format("%m-%Y")) + ylab("Maximum Daily Withdrawals (mgd)") +
    labs(color = "Water System") + labs(title = "2015 Maximum Daily Withdrawals for Durham and Ashevill
print(Max.withdrawals.plot_DurhamAsheville2015)
```



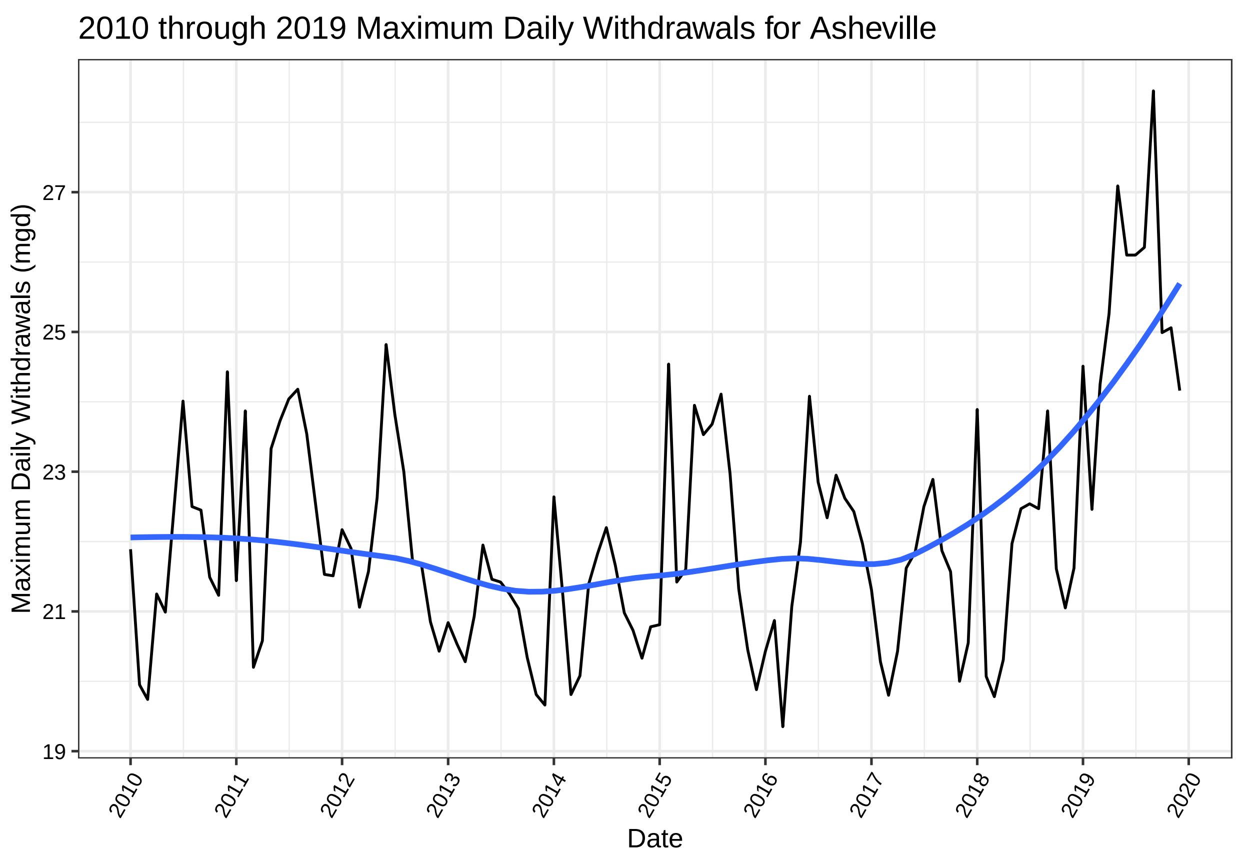
```r
# We can use the ggplot() function to plot
# the data from the combined data frame. We
# can use the aes() function to specify the
# x and y values and that the lines will be
# plotted in different colors based on the
# water.system. We use the geom_line()
# function to specify that we want a line
# plot. The scale_x_date() function allows
# us to include each month in the y-axis
# labeling to make the graph more clear. We
# use the ylab() and lab() functions to
# specify the labels for the y axis and the
# legend. Finally, we use the print()
# function to view the graph.
```

9. Use the code & function you created above to plot Asheville's max daily withdrawal by months for the years 2010 thru 2019.Add a smoothed line to the plot.

   TIP: See Section 3.2 in the "09_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.

```
# 9

the_years = rep(2010:2019)
my_pwsid = "01-11-010"

# We can set the values of our variables
# using the equal sign.

Asheville1019_dfs <- map2(the_years, my_pwsid,
    scrape.it) %>%
    bind_rows()

# To create the dataframe we can use the
# map2() function since we have a two
# argument case, and within this function we
# can specify the two variables we are using
# and the scrape.it function we made. Then
# using the pipe function %>% we can use the
# bind_rows() function to combine all of the
# variables.

Asheville1019_plot <- ggplot(Asheville1019_dfs,
    aes(x = Date, y = Maximum.Daily.Use.mgd)) +
    geom_line() + ylab("Maximum Daily Withdrawals (mgd)") +
    geom_smooth(method = "loess", se = FALSE) +
    scale_x_date(date_breaks = "1 year", labels = date_format("%Y")) +
    labs(title = "2010 through 2019 Maximum Daily Withdrawals for Asheville")
print(Asheville1019_plot)
```

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



```
# Using the ggplot() function we can define
# the dataset that we want to plot using the
# aes() function to specify the x and y
# values.  We use the geom_line() function
# to specify that we want a line plot. we
# can use the ylab() function to label the y
# axis. The geom_smooth() function allows us
# to add a smoothed line to the graph, and
# within this function we can define the
# method as 'loess' so it follows the line
# graph, and the se=FALSE gets rid of the
# confidence interval. The scale_x_date()
# function allows us to include each month
# in the y-axis labeling to make the graph
# more clear. Finally, we use the print()
# function to view the graph.
```

Question: Just by looking at the plot (i.e. not running statistics), does Asheville have a trend in water usage over time? Just looking at the graph it does appear that Asheville's water usage was relatively even until it spiked up in 2019, so it does look like it may start having an increasing trend.