Assignment 5: Data Visualization

Sarah Ko

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics (ENV872L) on data wrangling.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Use the lesson as a guide. It contains code that can be modified to complete the assignment.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document. Space for your answers is provided in this document and is indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">". You should notice that the answer is highlighted in green by RStudio.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file. You will need to have the correct software installed to do this (see Software Installation Guide) Press the `Knit` button in the RStudio scripting panel. This will save the PDF output in your Assignments folder.
6. After Knitting, please submit the completed exercise (PDF file) to the dropbox in Sakai. Please add your last name into the file name (e.g., "Salk_A04_DataWrangling.pdf") prior to submission.

The completed exercise is due on Tuesday, 19 February, 2019 before class begins.

Set up your session

1. Set up your session. Upload the NTL-LTER processed data files for chemistry/physics for Peter and Paul Lakes (tidy and gathered), the USGS stream gauge dataset, and the EPA Ecotox dataset for Neonicotinoids.

2. Make sure R is reading dates as date format, not something else (hint: remember that dates were an issue for the USGS gauge data).

```
#load packages
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.2
```

```
## -- Attaching packages ----------------------------------------------------------- tidyverse 1.2.1
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
## Warning: package 'readr' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.2
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
## -- Conflicts -------------------------------------------------------------- tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 3.5.2
```

```
## Loading required package: viridisLite
```

```
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.5.2
```

```
library(colormap)
```

```
## Warning: package 'colormap' was built under R version 3.5.2
```

```
library(tidyr)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 3.5.2
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
#1

# get working directory
getwd()
```

```
## [1] "C:/Users/Sarah/Documents/Duke/Year 2/Spring 2019/Data Analytics/Environmental_Data_Analytics"
```

```
# set wd to the filepath of Environmental_Data_Analytics to use relative filepath

PeterPaul.Gathered <- read.csv("./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Processed.csv

Stream.Raw <- read.csv("./Data/Raw/USGS_Site02085000_Flow_Raw.csv")

Neonicotinoids.Raw <- read.csv("./Data/Raw/ECOTOX_Neonicotinoids_Mortality_raw.csv")

# check that the dates fall within the reasonable timeframe

head(PeterPaul.Gathered$sampledate)
```

```
## [1] 1991-05-20 1991-05-20 1991-05-20 1991-05-20 1991-05-20 1991-05-20
## 778 Levels: 1991-05-20 1991-05-27 1991-05-28 1991-06-03 ... 2016-08-16
```

```
head(Stream.Raw$datetime)
```

```
## [1] 1/1/28 1/2/28 1/3/28 1/4/28 1/5/28 1/6/28
## 33216 Levels: 1/1/00 1/1/01 1/1/02 1/1/03 1/1/04 1/1/05 1/1/06 ... 9/9/99
```

```
head(Neonicotinoids.Raw$Pub..Year)
```

```
## [1] 2013 2017 2013 2013 2016 2016
```

```
# fix the dates in Stream.Raw

# change the class to date
Stream.Raw$datetime <- as.Date(Stream.Raw$datetime, format = "%m/%d/%y")

# format the dates as 6 character string: year/month/date
Stream.Raw$datetime <- format(Stream.Raw$datetime, "%y%m%d")

# create a function.
# paste0 amends the components together
# checks if the date is after Dec 31, 2018 (the last possible day in the dataset). if its over, the dat
create.early.dates <- (function(d) {
       paste0(ifelse(d > 181231,"19","20"),d)
       })

# create the new dates, reformat
Stream.Raw$datetime <- create.early.dates(Stream.Raw$datetime)
Stream.Raw$datetime <- as.Date(Stream.Raw$datetime, format = "%Y%m%d")

#2

# check the class of the date columns

class(PeterPaul.Gathered$sampledate)
```

```
## [1] "factor"
```

```
class(Stream.Raw$datetime)
```

```
## [1] "Date"
```

```
class(Neonicotinoids.Raw$Pub..Year)
```

```
## [1] "integer"
```

```
# change class date. must define original format of data
PeterPaul.Gathered$sampledate <- as.Date(PeterPaul.Gathered$sampledate, format = "%Y-%m-%d")

# the Neonicotinoid column Pub..Year is already an integer, and does not have a month/day associated wi

# confirm class of the date columns

class(PeterPaul.Gathered$sampledate)
```

```
## [1] "Date"
```

```
class(Stream.Raw$datetime)
```

```
## [1] "Date"
```

Define your theme

3. Build a theme and set it as your default theme.

```
#3

SKotheme <- theme_gray(base_size = 15) +
  theme(axis.text = element_text(color = "black"),
```

```
        legend.position = "right",
        plot.title = element_text(hjust = 0.5))

theme_set(SKotheme)
```

Create graphs

For numbers 4-7, create graphs that follow best practices for data visualization. To make your graphs "pretty," ensure your theme, color palettes, axes, and legends are edited to your liking.

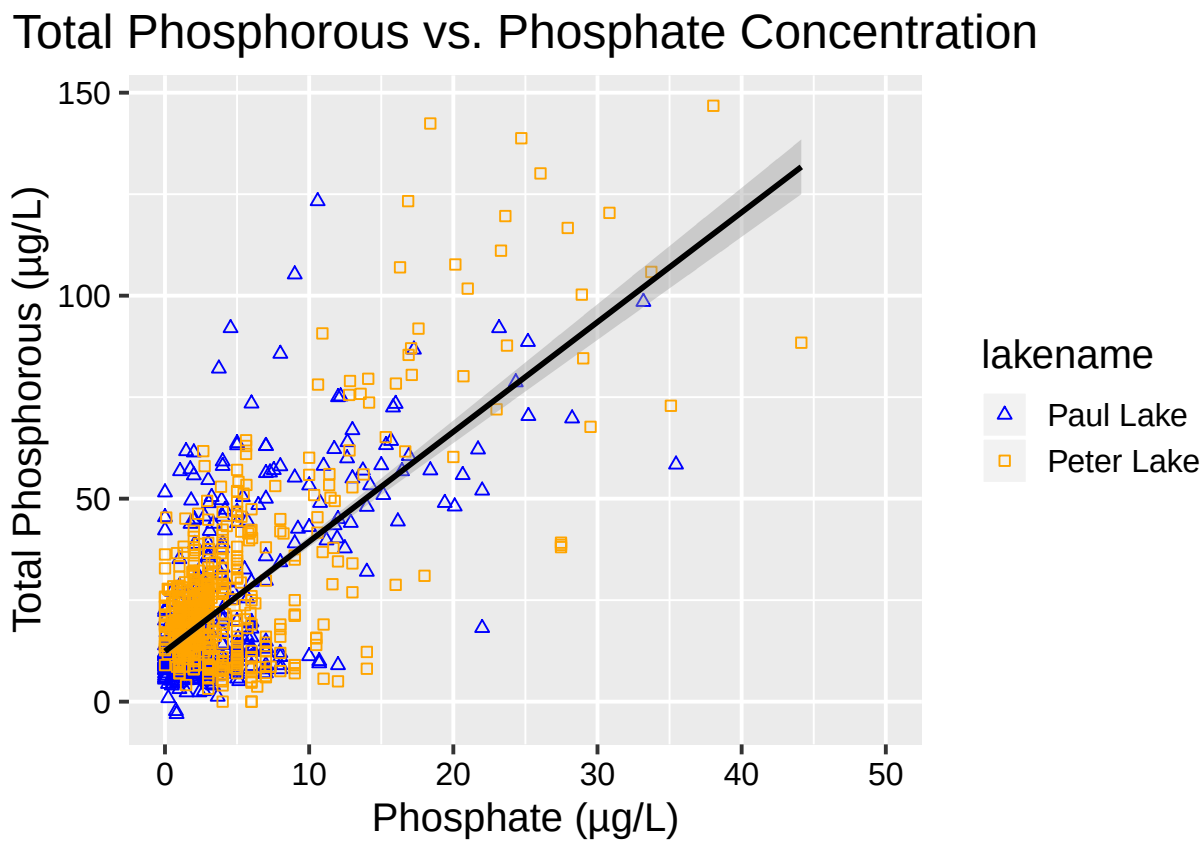
Hint: a good way to build graphs is to make them ugly first and then create more code to make them pretty.

4. [NTL-LTER] Plot total phosphorus by phosphate, with separate aesthetics for Peter and Paul lakes. Add a line of best fit and color it black.

```
#4

# Spread nutrient data into separate columns
PeterPaul.spread <- spread(PeterPaul.Gathered, nutrient, concentration)

# Remove row if there is an NA in either the po4 or tp_ug column
PeterPaul.spread.PlotP <-
  PeterPaul.spread %>%
  drop_na(po4, tp_ug)

tP_vs_PO4_plot <- ggplot(PeterPaul.spread.PlotP, aes(x = po4, y = tp_ug)) +
  geom_point(aes(shape = lakename, color = lakename)) +
  scale_shape_manual(values=c(2, 0)) + # define shapes for lake names
  geom_smooth(method=lm, colour="black") + # add line of best fit
  xlim(0, 50) + # zoom into concentration of points
  ggtitle("Total Phosphorous vs. Phosphate Concentration") + # add main title
  xlab("Phosphate (\U003BCg/L)") + # format labels with UOM
  ylab("Total Phosphorous (\U003BCg/L)") +
  scale_colour_manual(values = c("Peter Lake" = "Orange", "Paul Lake" = "Blue"))
print(tP_vs_PO4_plot)
```

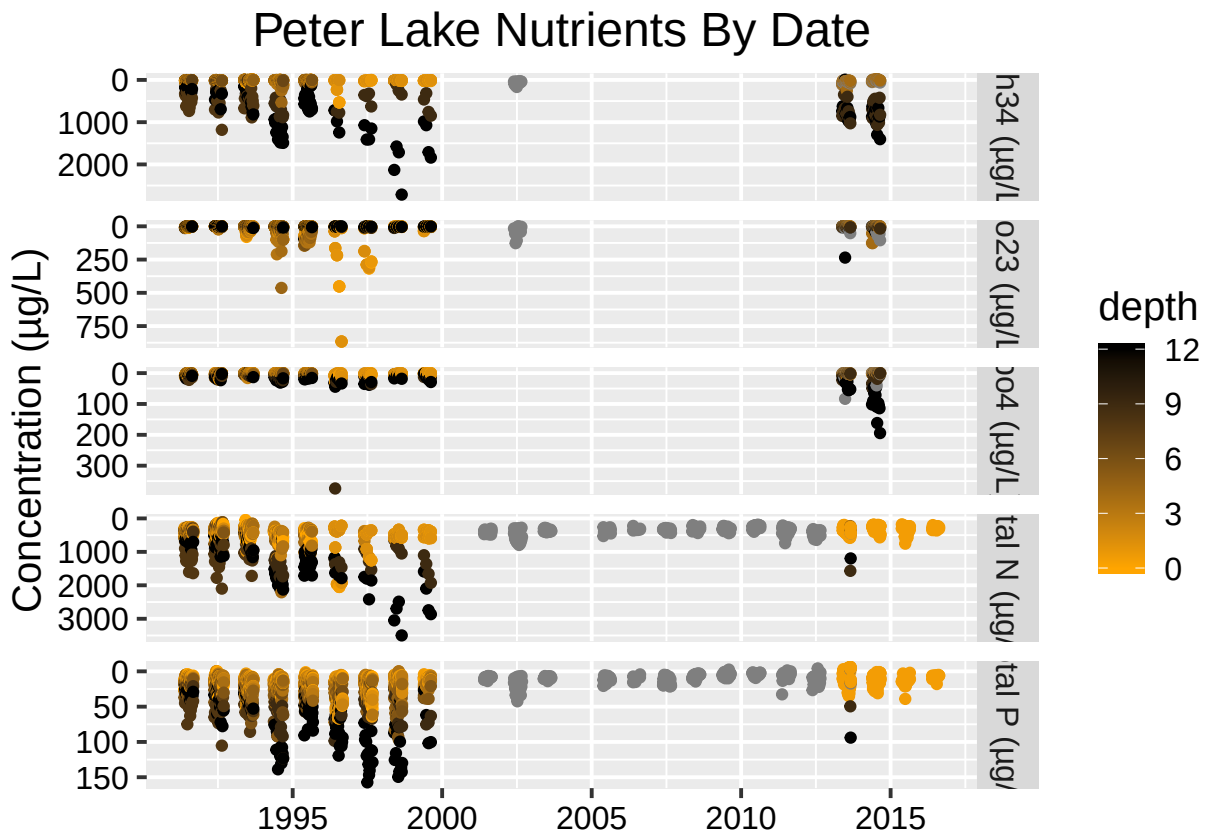
```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



5. [NTL-LTER] Plot nutrients by date for Peter Lake, with separate colors for each depth. Facet your graph by the nutrient type.

```
#5

#create facet titles
nutrient_names <- c(
                    `nh34` = "nh34 (\U003BCg/L)",
                    `no23` = "no23 (\U003BCg/L)",
                    `po4` = "po4 (\U003BCg/L)",
                    `tn_ug` = "Total N (\U003BCg/L)",
                    `tp_ug` = "Total P (\U003BCg/L)"
                    )

Peter_nutrients_bydate <- ggplot(PeterPaul.Gathered, aes(x = sampledate, y = concentration, group = nut
  geom_point() +
  ggtitle("Peter Lake Nutrients By Date") + # add main title
  facet_wrap(vars(nutrient), nrow = 5) +
  xlab(NULL) + # remove x label
  ylab("Concentration (\U003BCg/L)") +
  geom_blank(data=PeterPaul.Gathered, aes(sampledate, concentration)) +
  facet_grid(nutrient ~ ., scales="free", labeller = as_labeller(nutrient_names)) +
  scale_color_gradient(low = "orange", high = "black") +
  scale_y_reverse()
print(Peter_nutrients_bydate)
```

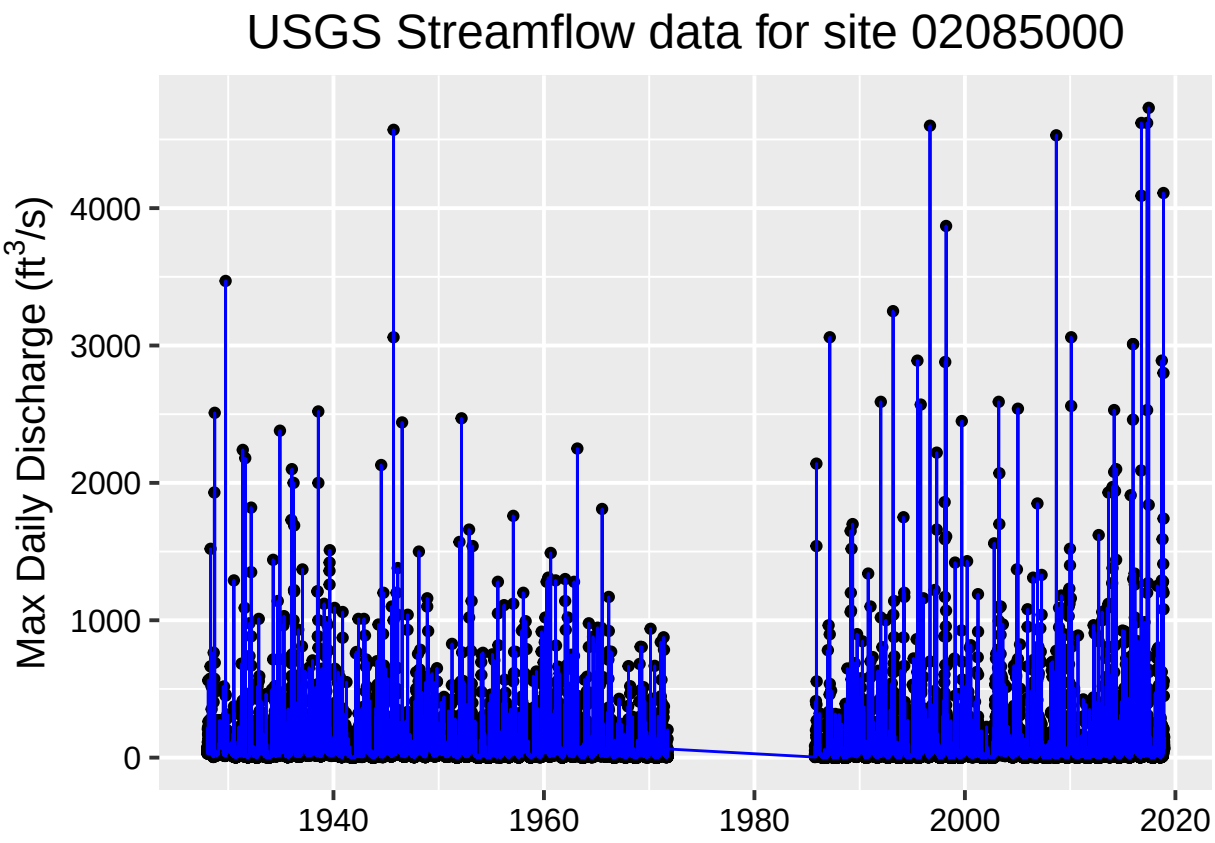


6. [USGS gauge] Plot discharge by date. Create two plots, one with the points connected with geom_line and one with the points connected with geom_smooth (hint: do not use method = "lm"). Place these graphs on the same plot (hint: ggarrange or something similar)

```
#6

# remove NAs in Stream.Raw
Stream.Raw_noNA <- Stream.Raw %>% drop_na(X165986_00060_00001)

# find max value of max discharge (cubic feet per second)
max(Stream.Raw_noNA$X165986_00060_00001)
```

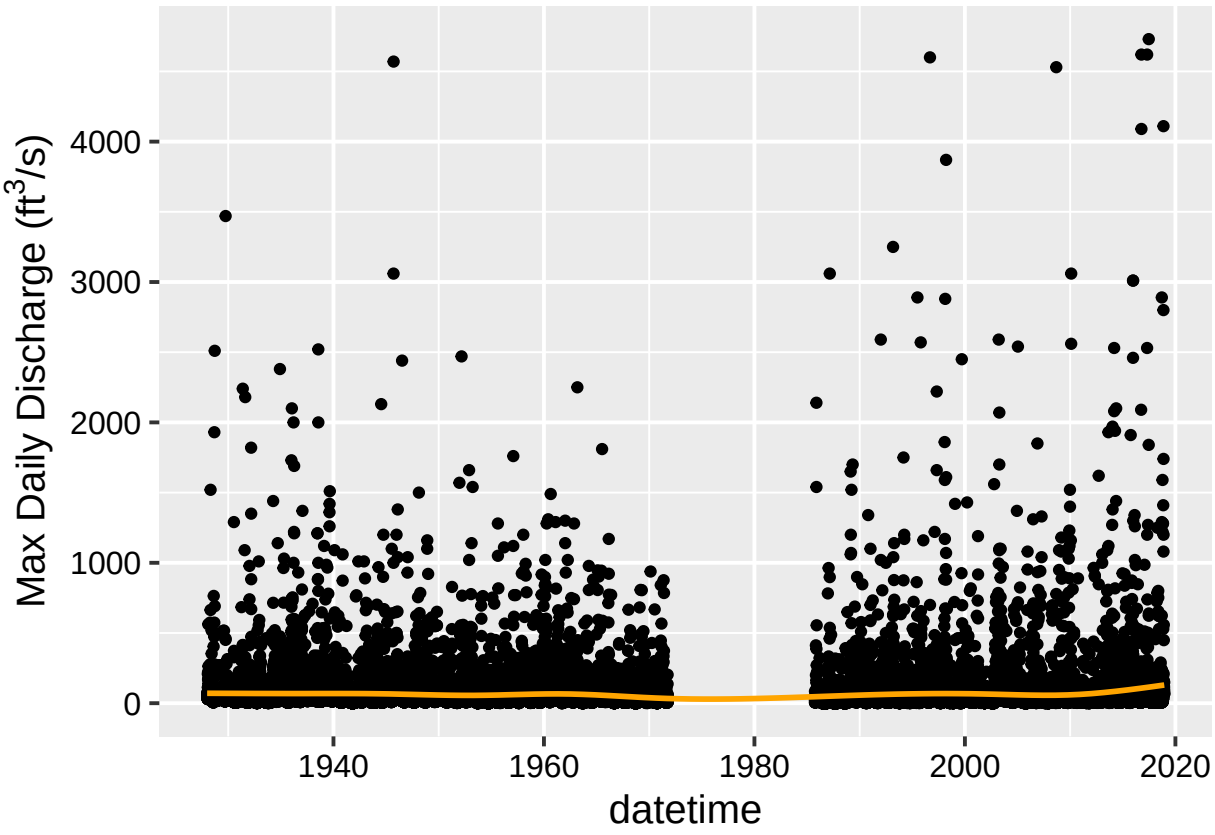
```
## [1] 4730
```

```
# plot graph with geom_line
Discharge_bydate_line <- ggplot(Stream.Raw_noNA, aes(x = datetime, y = X165986_00060_00001)) +
  geom_point() +
  geom_line(color = "blue") +
  xlab(NULL) +
  ylab(expression(paste("Max Daily Discharge (", ft^3, "/", s,")", sep=""))) +
  ggtitle("USGS Streamflow data for site 02085000")
print(Discharge_bydate_line)
```




```
# plot graph with geom_smooth
Discharge_bydate_smooth <- ggplot(Stream.Raw_noNA, aes(x = datetime, y = X165986_00060_00001)) +
  geom_point() +
  geom_smooth(method = "auto", color = "orange") +
  ylab(expression(paste("Max Daily Discharge (", ft^3, "/", s,")", sep="")))
print(Discharge_bydate_smooth)
```

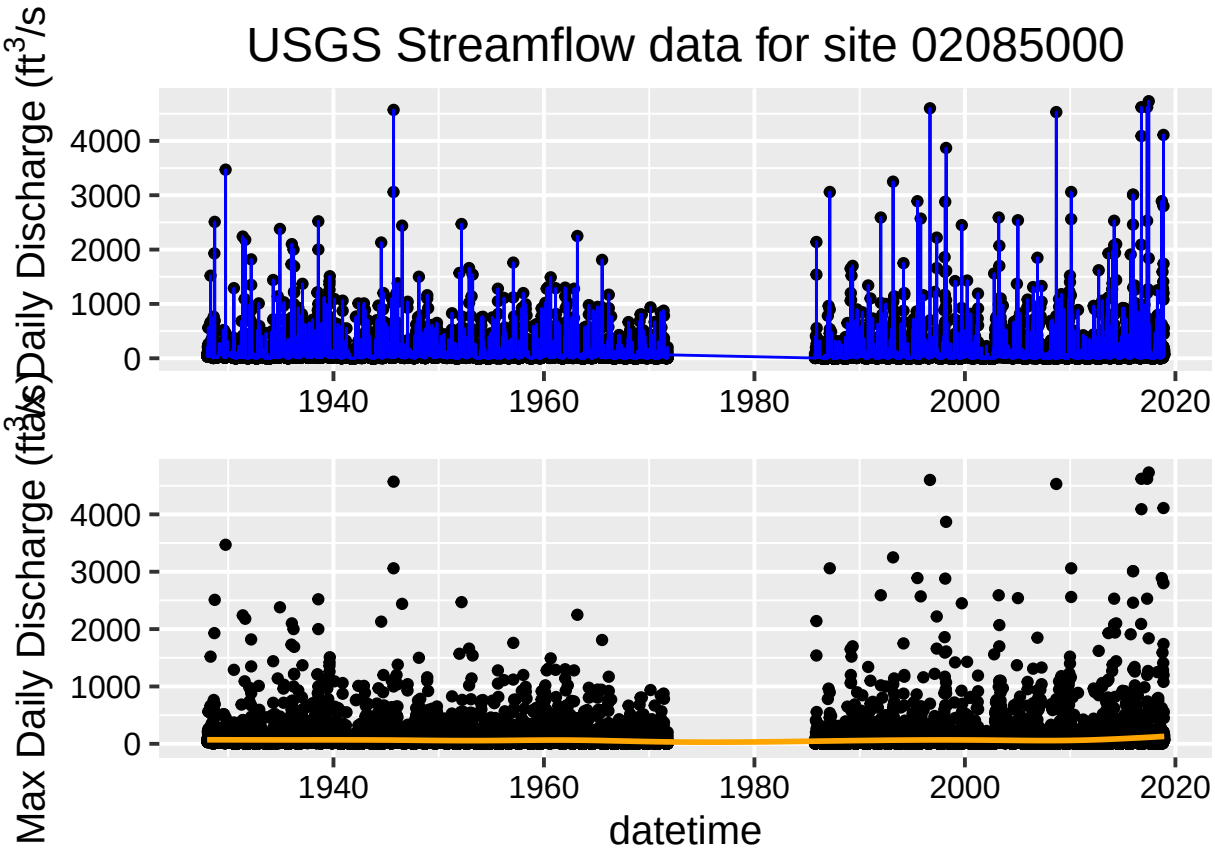
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# create combined figure
plot_stream <- ggarrange(Discharge_bydate_line, Discharge_bydate_smooth,
                         ncol = 1, nrow = 2)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
print(plot_stream)
```

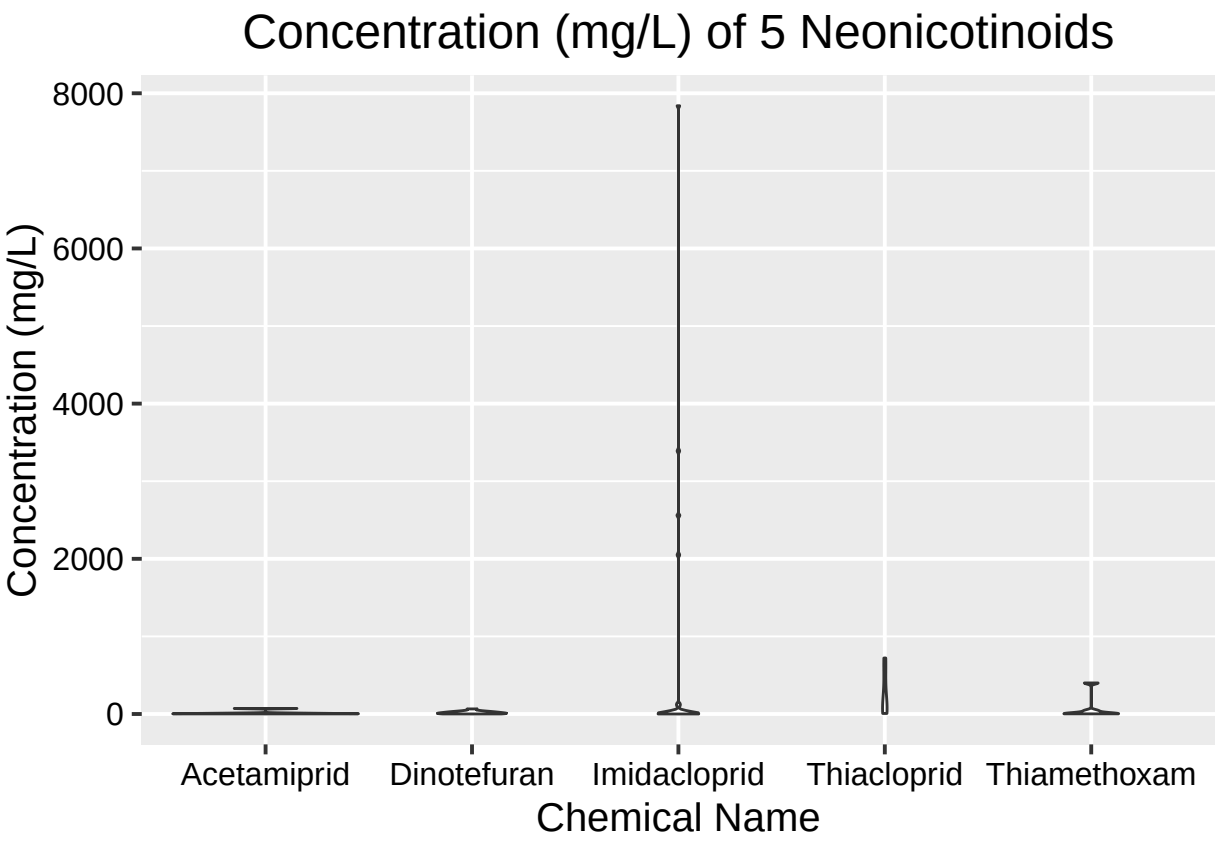


Question: How do these two types of lines affect your interpretation of the data?

> Answer: Interpreting the data with 'geom_line' makes the data appear to be highly variable because this line type connects each data point with the subsequent data point. Interpreting the data with geom_smooth makes the data seem relatively constant over time because this type creates a line based on the frequency of data points at each value. For this stream gauge, most of the data points were clustered around very low data points, the trend line created is mostly constant, and sits at a low daily discharge value.

7. [ECOTOX Neonicotinoids] Plot the concentration, divided by chemical name. Choose a geom that accurately portrays the distribution of data points.

```
#7

# filter to include only mg/L data
Neonicotinoids_mgL <-
  Neonicotinoids.Raw %>%
  filter(Conc..Units..Std. == "mg/L")

Neonicotinoids_plot <-
  ggplot(Neonicotinoids_mgL, aes(x = Chemical.Name, y = Conc..Mean..Std.)) +
  geom_violin() +
  xlab("Chemical Name") +
  ylab("Concentration (mg/L)") +
  ggtitle("Concentration (mg/L) of 5 Neonicotinoids")
print(Neonicotinoids_plot)
```


Concentration (mg/L) of 5 Neonicotinoids
Concentration (mg/L)
8000
6000
4000
2000
0
Acetamiprid
Dinotefuran
Imidacloprid
Thiacloprid
Thiamethoxam
Chemical Name