Assignment 5: Data Visualization

Caroline Watson

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).

```
#1
#viewing working directory
getwd()
```

```
## [1] "/Users/carolinewatson/Documents/Spring 2019/Environmental Data Analytics/Env_Data_Analytics"
```

```
#loading tidyverse package
suppressMessages(library(tidyverse))
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(viridis)
```

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

```
library(RColorBrewer)
library(colormap)

#reading in data files
PeterPaul.chem.nutrients <-
  read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv")

PeterPaul.nutrients.gathered <-
  read.csv("./Data/Processed/PeterPaul.gathered.processed.csv")

#write.csv(USGS.flow.data.complete, row.names = FALSE, file = "./Data/Processed/USGS_Stream_Flow_Proces

USGS.flow.data.complete <-
  read.csv("./Data/Processed/USGS_Stream_Flow_Processed.csv")

ecotox.neonic.data <-
  read.csv("./Data/Raw/ECOTOX_Neonicotinoids_Mortality_raw.csv")

#fixing format for column headings because they added .. when the data was imported
colnames(ecotox.neonic.data)[8:12] <- c("Duration", "Conc.Type", "Conc.Mean", "Conc.Units", "Pub.Year")

#2 updating the date for each table
PeterPaul.chem.nutrients$sampledate <- as.Date(PeterPaul.chem.nutrients$sampledate, format = "%Y-%m-%d"
PeterPaul.nutrients.gathered$sampledate <- as.Date(PeterPaul.nutrients.gathered$sampledate, format ="%Y
USGS.flow.data.complete$datetime <- as.Date(USGS.flow.data.complete$datetime, format = "%Y-%m-%d")
#ecotox.neonic.data$Pub.Year <- as.Date(ecotox.neonic.data$Pub.Year, format = "%Y")
```

Define your theme

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

```
#3
#creating my theme
caroline_theme <- theme_classic(base_size = 16) +
  theme(axis.text = element_text(color = "black"),
        legend.position = "right")

#setting my theme as the default theme throughout the code
theme_set(caroline_theme)
```

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.

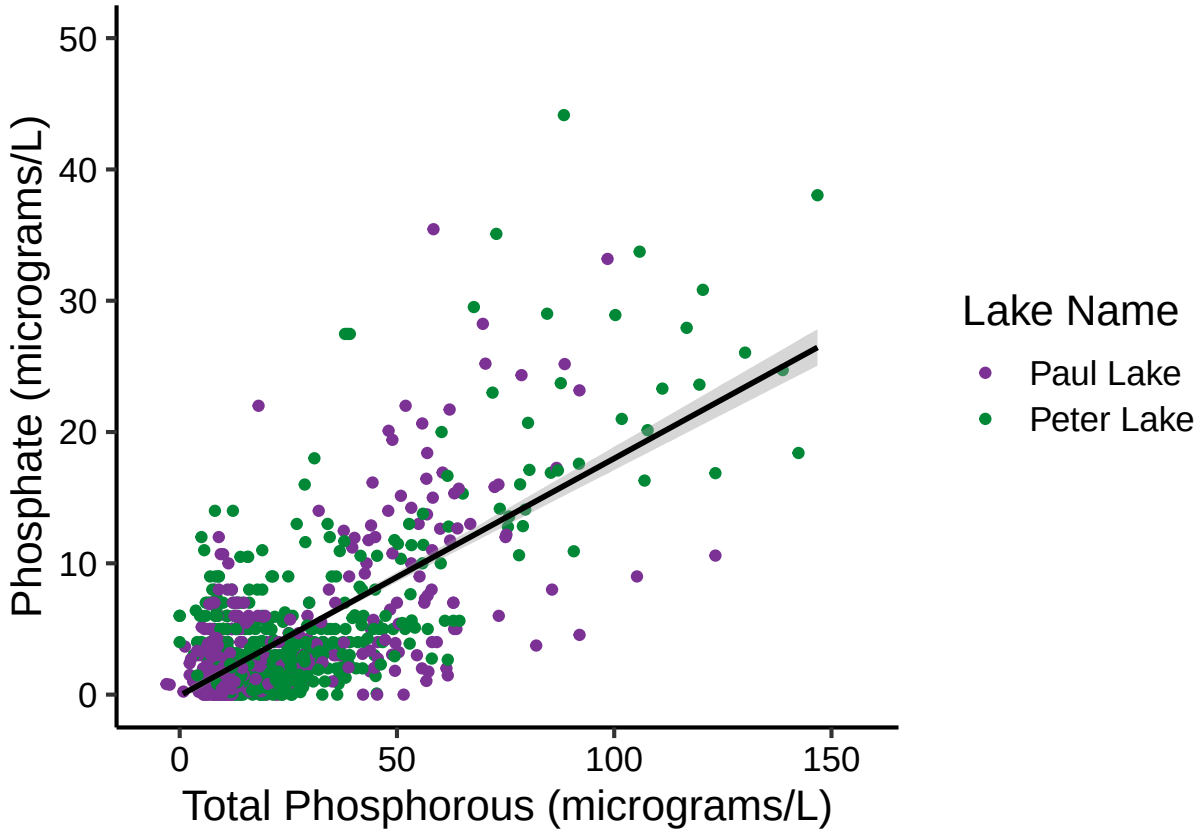
```
#4
#creating a graph of total phosphorous by phosphate
ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = po4, color = lakename)) +
```

```
      geom_point() +
  ylim(0,50) +
  geom_smooth(method = lm, color = "black") +
  labs(x = "Total Phosphorous (micrograms/L)", y = "Phosphate (micrograms/L)", color = "Lake Name") +
  scale_color_manual(values = c("#7b3294", "#008837"))
```

```
## Warning: Removed 22310 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 22310 rows containing missing values (geom_point).
```

```
## Warning: Removed 2 rows containing missing values (geom_smooth).
```

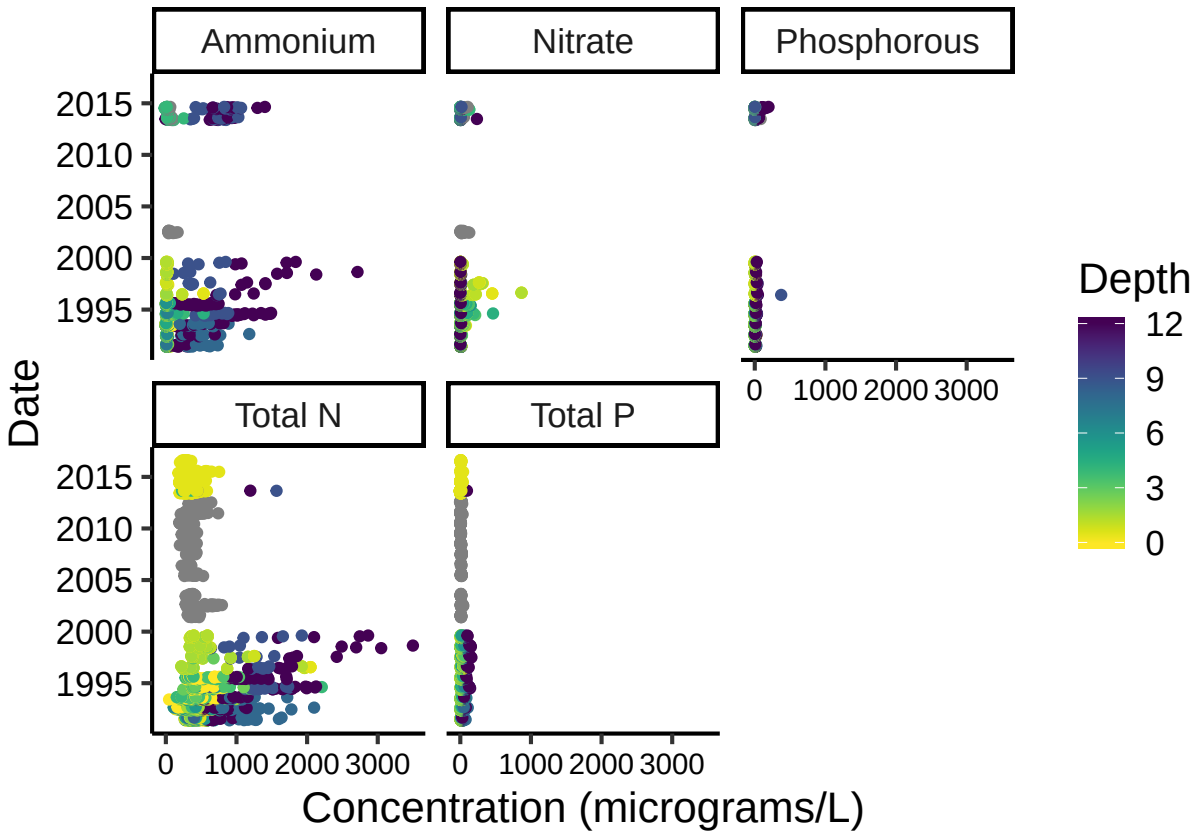


5. [NTL-LTER] Plot nutrients by date for Peter Lake, with separate colors for each depth. Facet your graph by the nutrient type.

```
#5 #updating the nutrient names so they will display correctly in the graph
levels(PeterPaul.nutrients.gathered$nutrient) <- c("Ammonium", "Nitrate",
                                                   "Phosphorous", "Total N", "Total P")

#plotting nutrients by date for Peter Lake with seperate colors for each depth
ggplot(subset(PeterPaul.nutrients.gathered, lakename = "Peter Lake"),
       aes(x = concentration, y = sampledate, color = depth)) +
  geom_point() +
  labs(x = "Concentration (micrograms/L)", y = "Date", color = "Depth") +
  scale_color_viridis(option = "viridis", direction = -1) +
  facet_wrap(vars(nutrient)) +
  theme(axis.text.x = element_text(size = 11))
```

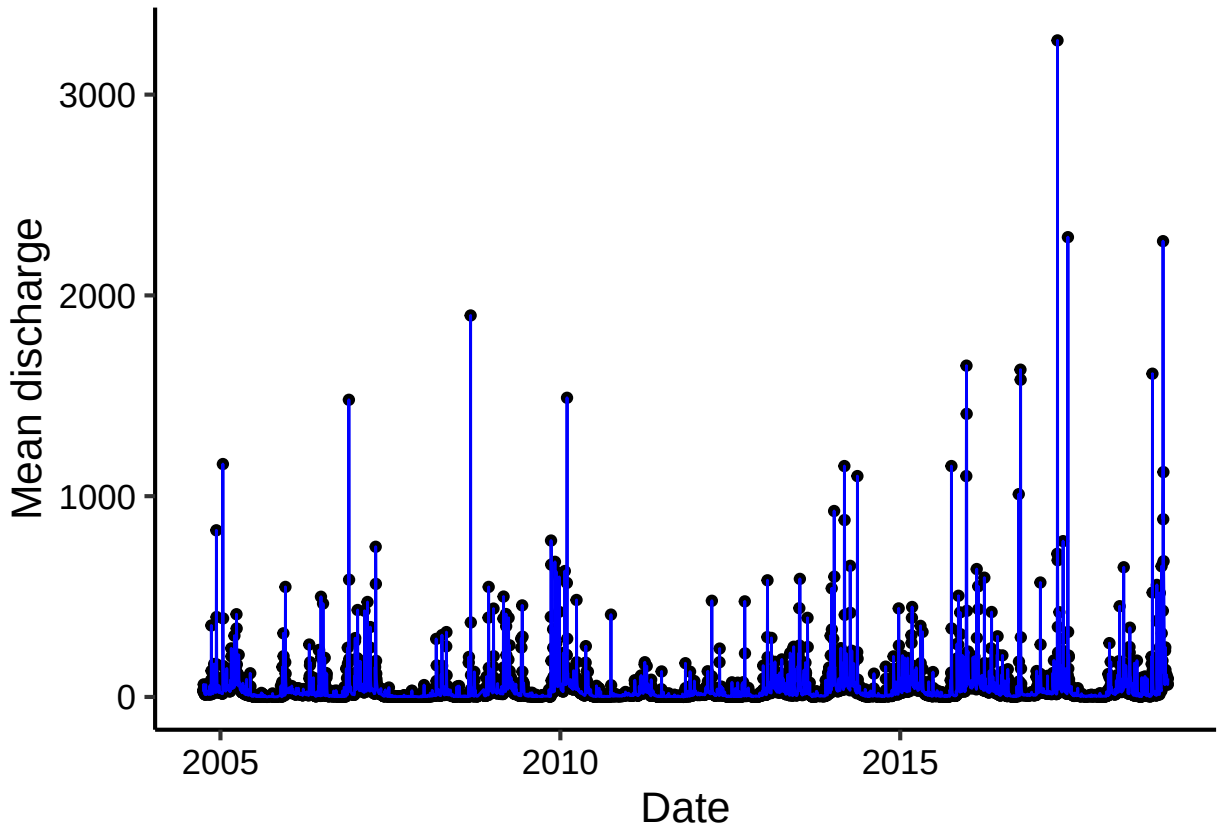
```
## Warning: Removed 5853 rows containing missing values (geom_point).
```



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)

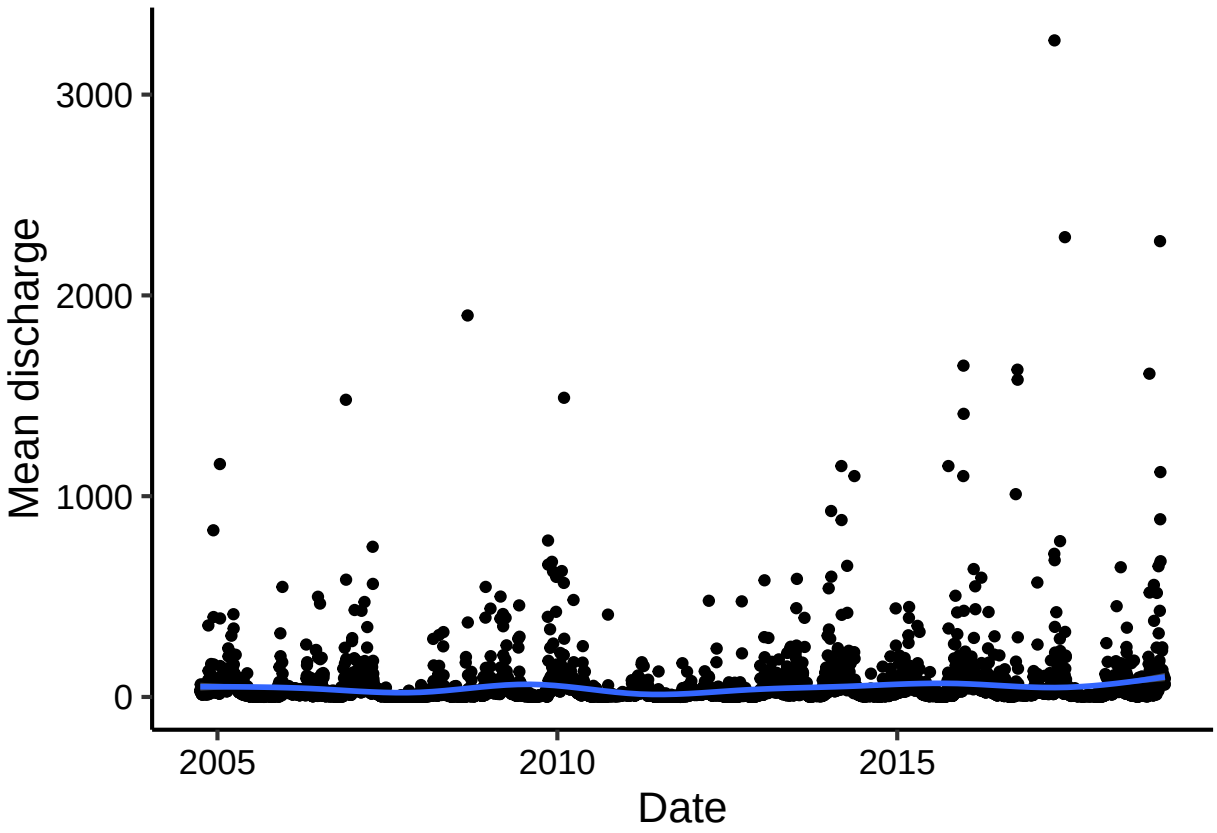
```
#6

#plot of discharge by date geom_point with geom_line
dischargeline <- ggplot(USGS.flow.data.complete, aes(x = datetime, y = discharge.mean)) +
  geom_point() +
  geom_line(color = "blue") +
  labs(x = "Date", y = "Mean discharge")
print(dischargeline)
```



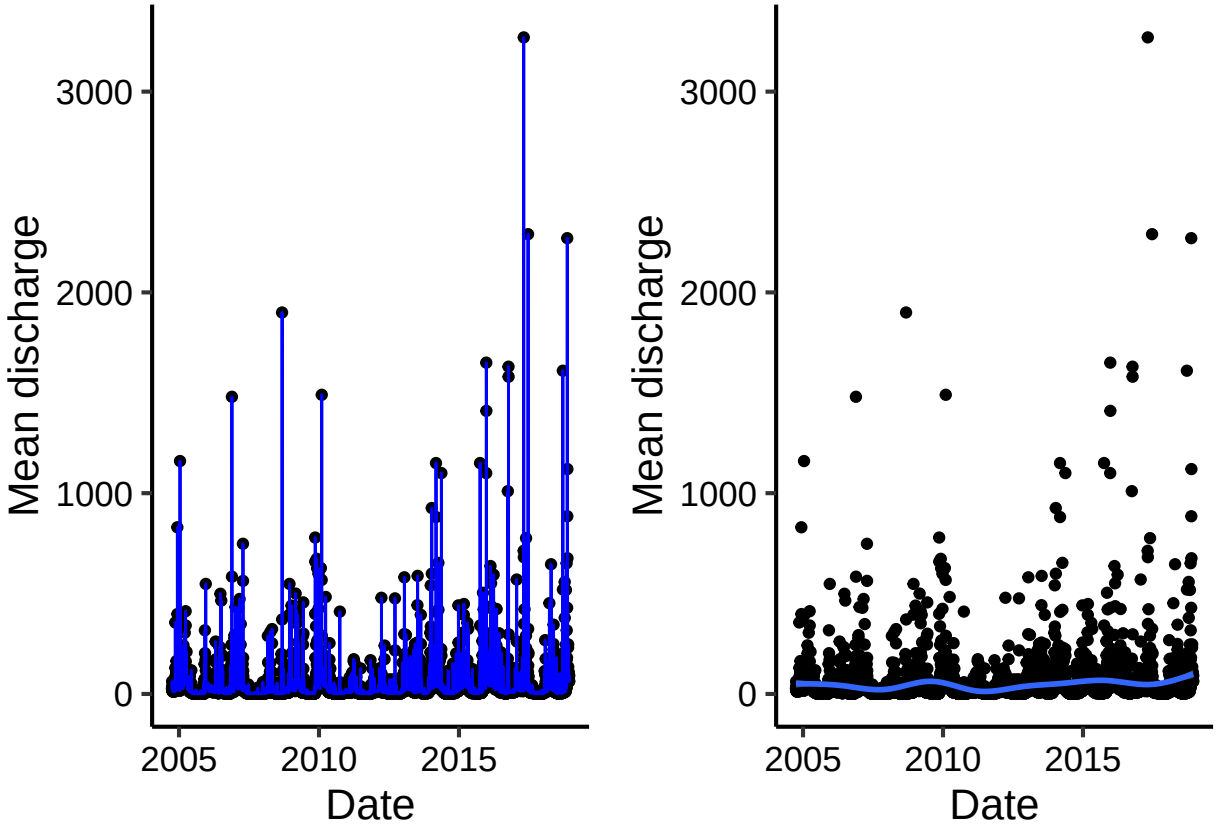

```
#plot of discharge by date using geom_smooth
dischargesmooth <- ggplot(USGS.flow.data.complete, aes(x = datetime, y = discharge.mean)) +
  geom_point() +
  geom_smooth(method = "auto") +
  labs(x = "Date", y = "Mean discharge")
print(dischargesmooth)
```

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




```
#plotting graphs together using grid.arrange
grid.arrange(dischargeline, dischargesmooth, nrow = 1)
```

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



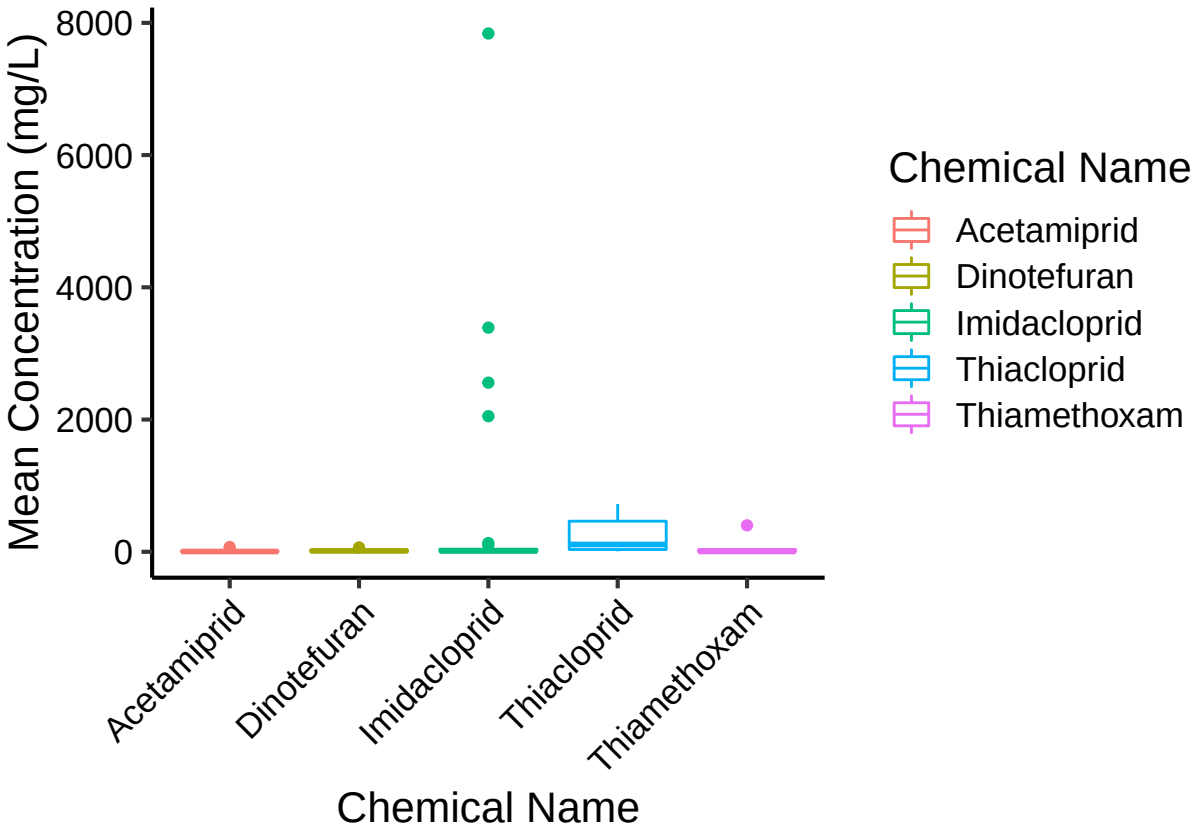
Question: How do these two types of lines affect your interpretation of the data?

> Answer: Geom_line creates a line between the values and is ordered by the x-value whereas geom_smooth creates a line of best-fit. These lines affect the interpretation of the graph because geom_line makes it seem like all the points are connected, whereas geom_smooth, shows that there is no trend between data points.

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

```
#7
#filtering data to only have ecotox data measured in mg/L
ecotox_filtered <- ecotox.neonic.data %>%
  filter(Conc.Units == "mg/L")

#plot of concentration divided by chemical name
ggplot(ecotox_filtered, aes(x = Chemical.Name, y = Conc.Mean, color = Chemical.Name)) +
  geom_boxplot() +
  labs(x = "Chemical Name", y = "Mean Concentration (mg/L)", color = "Chemical Name") +
  theme(axis.text.x = element_text(angle = 45,  hjust = 1))
```


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