Assignment 5: Data Visualization

Caroline Reents

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
getwd()
```

```
## [1] "/Users/carolinereents/Desktop/Data Analytics/EnvironmentalDataAnalytics"
```

```
library(tidyverse)
```

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

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

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

```
library(readr)
library(viridis)
```

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

```
library(RColorBrewer)
library(colormap)
library(gridExtra)
```

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

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

```
library(stringr)

PeterPaul.ChemPhys.processed <- read.csv("./Data/Processed/NTL-LTER_Lake_ChemistryPhysics_PeterPaul_Pro


PeterPaul.ChemNut.processed <- read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_P


USGS_Flow_Raw <- read_csv("./Data/Raw/USGS_Site02085000_Flow_Raw.csv")
```

```
## Parsed with column specification:
## cols(
##   agency_cd = col_character(),
##   site_no = col_integer(),
##   datetime = col_character(),
##   `165986_00060_00001` = col_double(),
##   `165986_00060_00001_cd` = col_character(),
##   `165987_00060_00002` = col_character(),
##   `165987_00060_00002_cd` = col_character(),
##   `84936_00060_00003` = col_character(),
##   `84936_00060_00003_cd` = col_character(),
##   `84937_00065_00001` = col_character(),
##   `84937_00065_00001_cd` = col_character(),
##   `84938_00065_00002` = col_character(),
##   `84938_00065_00002_cd` = col_character(),
##   `84939_00065_00003` = col_character(),
##   `84939_00065_00003_cd` = col_character()
## )
```

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


colnames(USGS_Flow_Raw) <- c("agency_cd", "site_no", "datetime",
                             "discharge.max", "discharge.max.approval",
                             "discharge.min", "discharge.min.approval",
                             "discharge.mean", "discharge.mean.approval",
                             "gage.height.max", "gage.height.max.approval",
                             "gage.height.min", "gage.height.min.approval",
                             "gage.height.mean", "gage.height.mean.approval")

names (ECOTOX_Neonicotinoids_Mortality_raw) <- str_replace_all(names(ECOTOX_Neonicotinoids_Mortality_ra


#2
class(PeterPaul.ChemPhys.processed$sampledate)
```

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

```
class(PeterPaul.ChemNut.processed$sampledate) #already reads as date
```

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

```
class(USGS_Flow_Raw$datetime)
```

```
## [1] "character"
```

```
PeterPaul.ChemPhys.processed$sampledate <-as.Date(PeterPaul.ChemPhys.processed$sampledate, format = "%Y
USGS_Flow_Raw$datetime <- as.Date(USGS_Flow_Raw$datetime, format = "%m/%d/%y")
USGS_Flow_Raw$datetime <- as.Date(ifelse(USGS_Flow_Raw$datetime > Sys.Date(),
  format(USGS_Flow_Raw$datetime, "19%y-%m-%d"),
  format(USGS_Flow_Raw$datetime)))
```

Define your theme

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

```
#3
mytheme <- theme_classic(base_size = 14) +
  theme(axis.text = element_text(color = "black"),
        legend.position = "top")
theme_set(mytheme)
```

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
class(PeterPaul.ChemNut.processed$tp_ug)
```

```
## [1] "numeric"
```

```
class(PeterPaul.ChemNut.processed$po4)
```

```
## [1] "numeric"
```

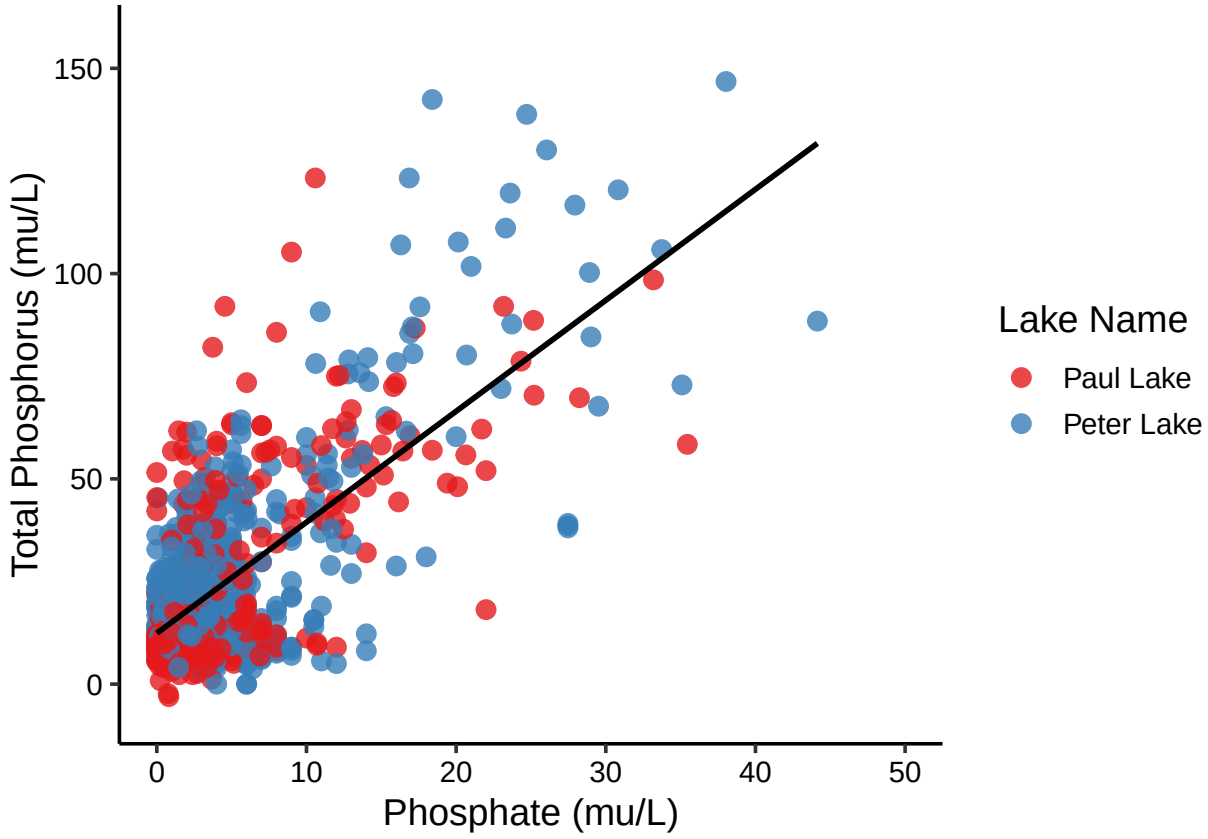
```
PeterPaul.ChemNut.processed$tp_ug <- as.numeric(PeterPaul.ChemNut.processed$tp_ug)
PeterPaul.ChemNut.processed$po4 <- as.numeric(PeterPaul.ChemNut.processed$po4)

View(PeterPaul.ChemNut.processed)
graph.num.4 <- ggplot(PeterPaul.ChemNut.processed, aes(x = po4, y = tp_ug, color= lakename))  +
  xlim(0,50) +
  labs(x="Phosphate (mu/L)", y="Total Phosphorus (mu/L)" , color="Lake Name" )+
  theme(legend.position = "right")+
  scale_color_brewer (palette = "Set1")+
  geom_point(alpha = 0.8, size = 3)+
  geom_smooth(method=lm, color="black", se=FALSE)
```

```
print(graph.num.4)
```

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

```
## Warning: Removed 22310 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
Nutrients_PeterPaulGathered_Processed <- read_csv("./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGa
```

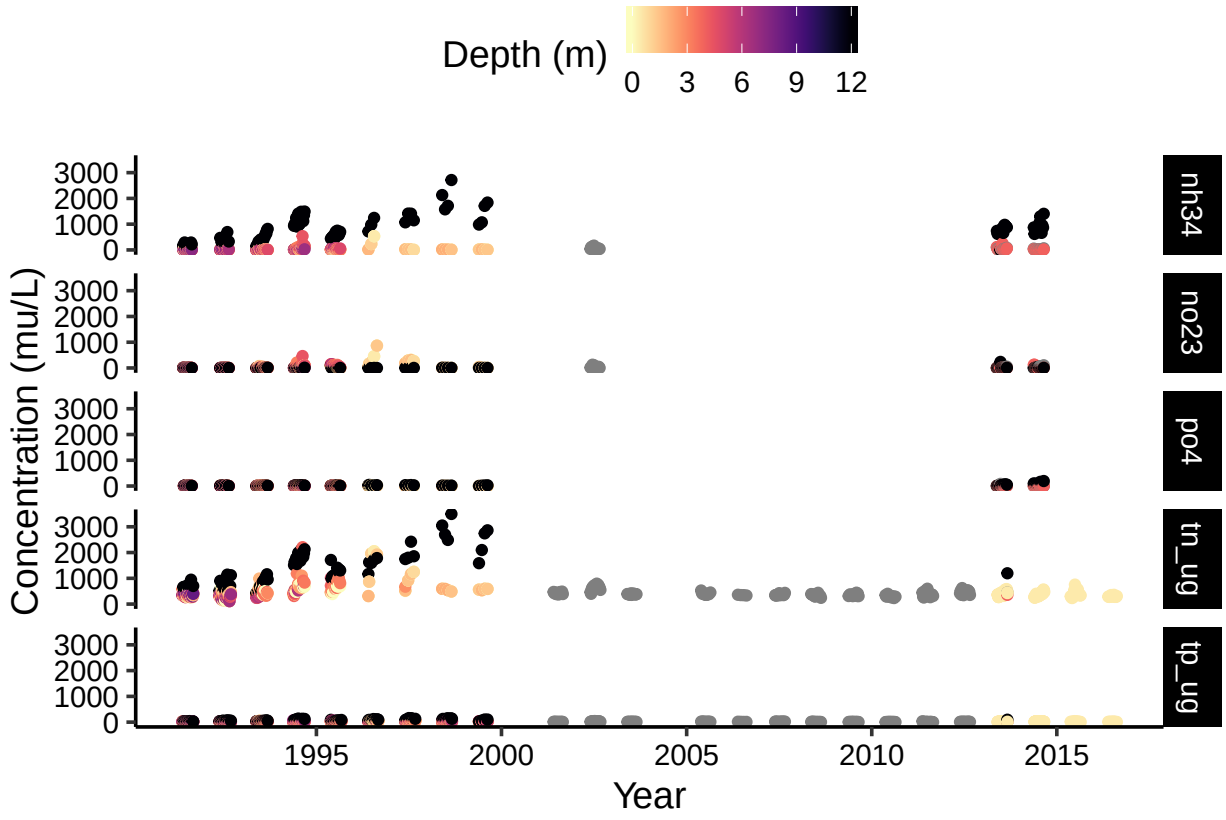
```
## Parsed with column specification:
## cols(
##   lakename = col_character(),
##   daynum = col_integer(),
##   year4 = col_integer(),
##   sampledate = col_date(format = ""),
##   depth = col_double(),
##   nutrient = col_character(),
##   concentration = col_double()
## )
```

```
Nutrients_PeterGathered_Processed <- filter(Nutrients_PeterPaulGathered_Processed, lakename == "Peter La
View(Nutrients_PeterGathered_Processed)
Peter.faceted <- ggplot (Nutrients_PeterGathered_Processed, aes(x= sampledate, y=concentration, color=de
  geom_point() +
```

```
  facet_grid(nutrient~.) +
  theme(strip.background = element_rect(fill = "black"), strip.text = element_text(color = "white")) +
  scale_color_viridis(option = "magma", direction=-1)+
  labs(x="Year", y= "Concentration (mu/L)", color= "Depth (m)")



print(Peter.faceted)
```



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)

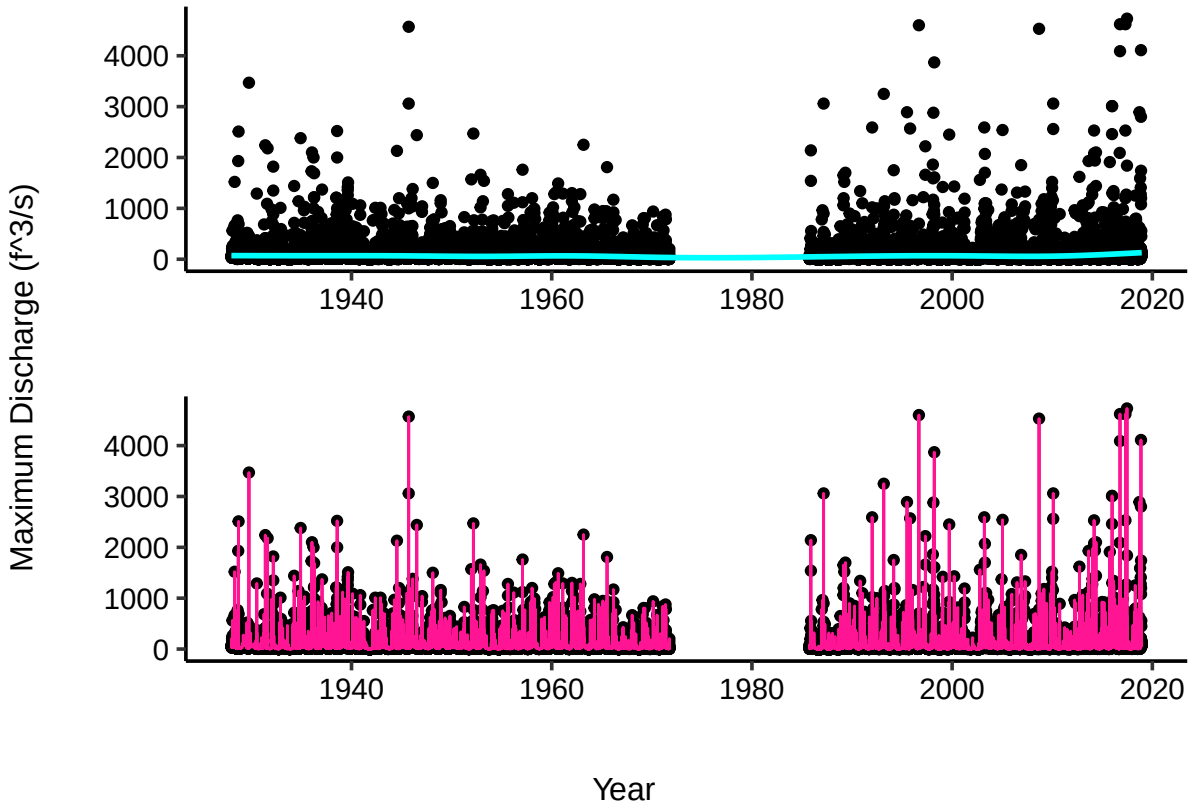
```
#6


USGSplot <- ggplot(USGS_Flow_Raw, aes(x = datetime, y =  discharge.max)) +
  geom_point()+
  labs(x="", y="")+
  geom_smooth(method="auto", color="cyan")

USGSplot1 <- ggplot(USGS_Flow_Raw, aes(x = datetime, y =  discharge.max)) +
  geom_point()+
  labs(x="", y="") +
  geom_line(color="deeppink1")
```

```
graph.num.6<-grid.arrange(USGSplot, USGSplot1, top="", bottom="Year",
                left="Maximum Discharge (f^3/s)", right="")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 5113 rows containing non-finite values (stat_smooth).
## Warning: Removed 5113 rows containing missing values (geom_point).
## Warning: Removed 5113 rows containing missing values (geom_point).
```




```
print(graph.num.6)
```

```
## TableGrob (4 x 3) "arrange": 6 grobs
##   z     cells    name                grob
## 1 1 (2-2,2-2) arrange      gtable[layout]
## 2 2 (3-3,2-2) arrange      gtable[layout]
## 3 3 (1-1,2-2) arrange text[GRID.text.357]
## 4 4 (4-4,2-2) arrange text[GRID.text.358]
## 5 5 (1-4,1-1) arrange text[GRID.text.359]
## 6 6 (1-4,3-3) arrange text[GRID.text.360]
```

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

> Answer:geom_smooth interprets as both variables being seperate and continuous and geom_line interpreets them as related by a continuous function

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

```
#7
class(ECOTOX_Neonicotinoids_Mortality_raw$`Conc..Mean.(Std)`)
```

```
## [1] "NULL"
```

```
graph.num.7 <- ggplot(ECOTOX_Neonicotinoids_Mortality_raw, aes(x=Pub..Year , y= "Conc..Mean.(Std)", col
  geom_point()


print(graph.num.7)
```