8: Data Visualization Basics

Environmental Data Analytics | Kateri Salk

Spring 2020

Objectives

1. Perform simple data visualizations in the R package `ggplot`
2. Develop skills to adjust aesthetics and layers in graphs
3. Apply a decision tree framework for appropriate graphing methods

Opening discussion

Effective data visualization depends on purposeful choices about graph types. The ideal graph type depends on the type of data and the message the visualizer desires to communicate. The best visualizations are clear and simple. My favorite resource for data visualization is Data to Viz, which includes both a decision tree for visualization types and explanation pages for each type of data, including links to R resources to create them. Take a few minutes to explore this website.

Set Up

```
getwd()
```

```
## [1] "C:/Users/senam/Box Sync/My Documents/MEM classes/Duke Spring 2020/DataAnalytics/Environmental_Da
```

```
library(tidyverse)
#install.packages("ggridges")
library(ggridges)
library(lubridate)

PeterPaul.chem.nutrients <-
  read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv")
PeterPaul.chem.nutrients.gathered <-
  read.csv("./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Processed.csv")
EPAair <- read.csv("./Data/Processed/EPAair_O3_PM25_NC1819_Processed.csv")

EPAair$Date <- as.Date(EPAair$Date, format = "%Y-%m-%d")
PeterPaul.chem.nutrients$sampledate <- as.Date(PeterPaul.chem.nutrients$sampledate, format = "%Y-%m-%d"
```

ggplot

ggplot, called from the package `ggplot2`, is a graphing and image generation tool in R. This package is part of tidyverse. While base R has graphing capabilities, ggplot has the capacity for a wider range and more sophisticated options for graphing. ggplot has only a few rules:

- The first line of ggplot code always starts with `ggplot()`
- A data frame must be specified within the `ggplot()` function. Additional datasets can be specified in subsequent layers.

- Aesthetics must be specified, most commonly x and y variables but including others. Aesthetics can be specified in the `ggplot()` function or in subsequent layers.
- Additional layers must be specified to fill the plot.

Geoms

Here are some commonly used layers for plotting in ggplot:

- geom_bar
- geom_histogram
- geom_freqpoly
- geom_boxplot
- geom_violin
- geom_dotplot
- geom_density_ridges
- geom_point
- geom_errorbar
- geom_smooth
- geom_line
- geom_area
- geom_abline (plus geom_hline and geom_vline)
- geom_text

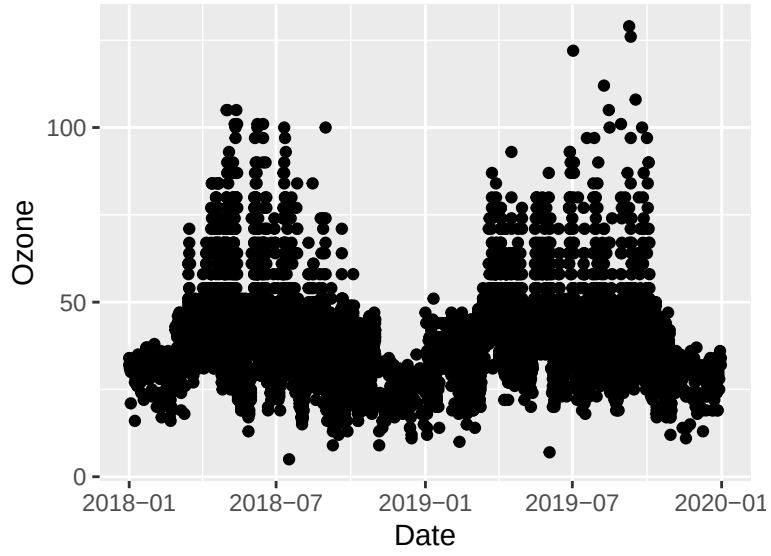
Aesthetics

Here are some commonly used aesthetic types that can be manipulated in ggplot:

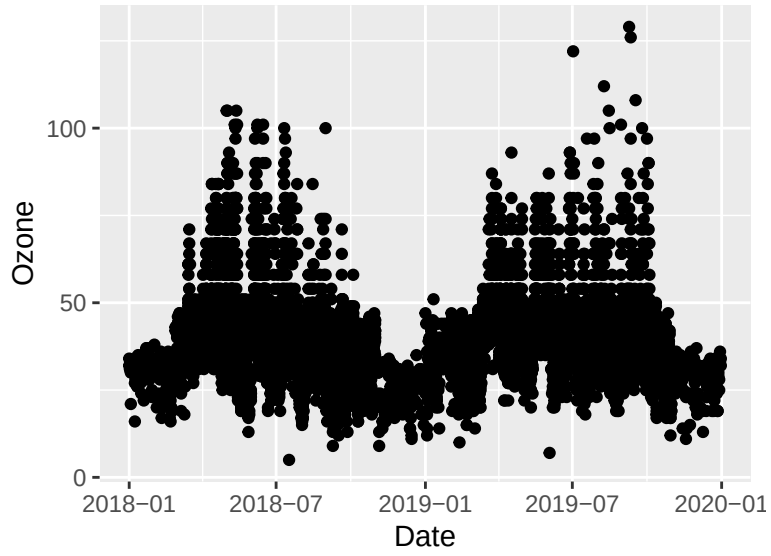
- color
- fill
- shape
- size
- transparency

Plotting continuous variables over time: Scatterplot and Line Plot

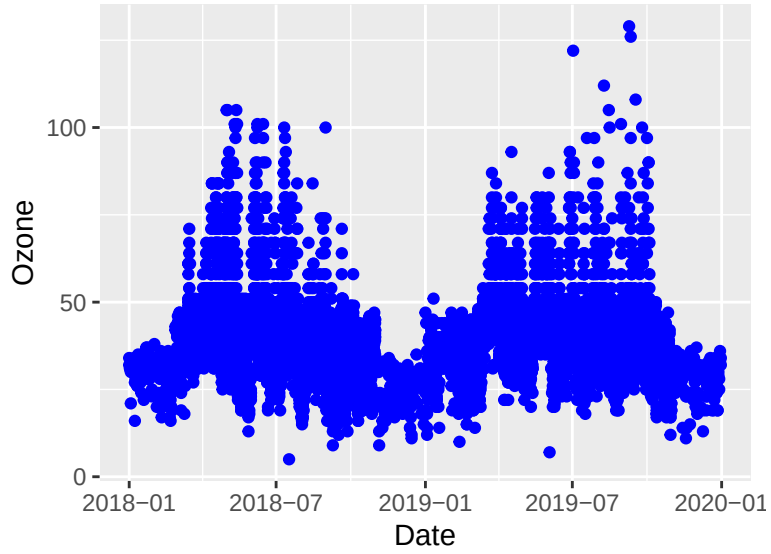
```
# Scatterplot
ggplot(EPAair, aes(x = Date, y = Ozone)) +
  geom_point()
```



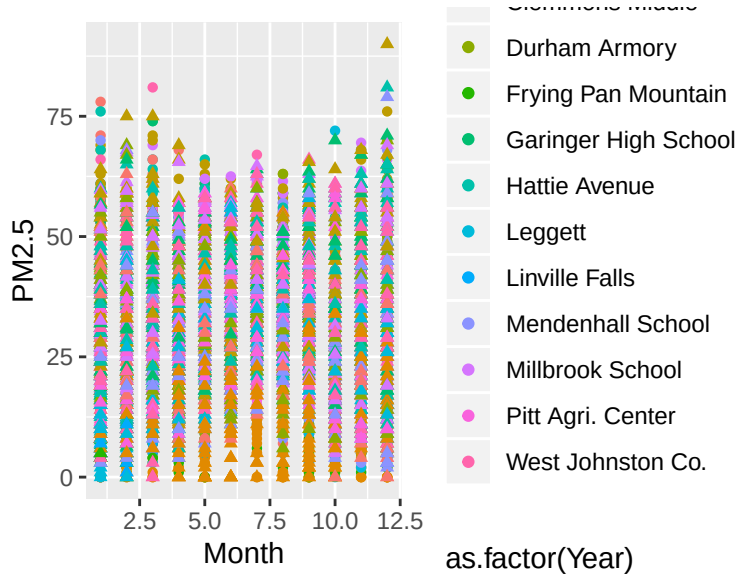

```
O3plot <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
print(O3plot)
```



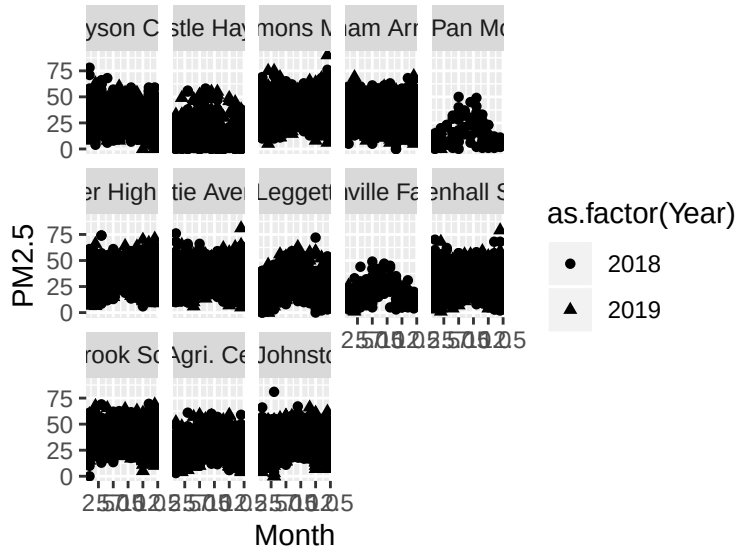

```
# Fix this code
O3plot2 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone), color = "blue")
print(O3plot2)
```



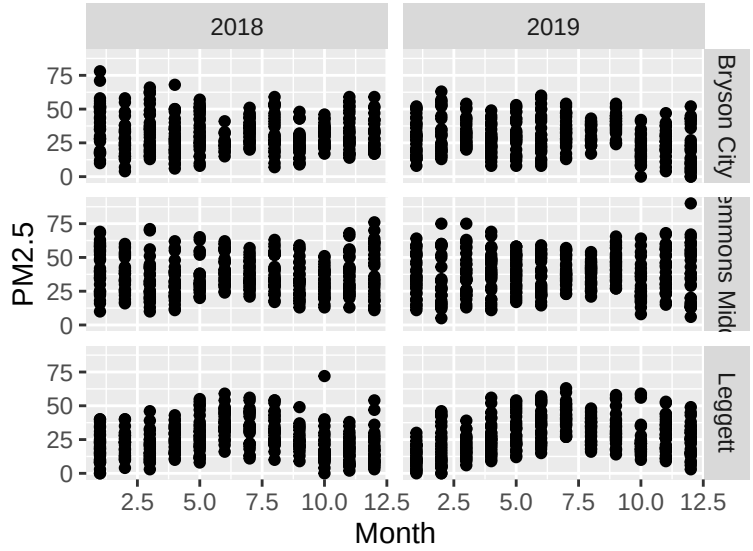

```
# Add additional variables
PMplot <-
  ggplot(EPAair, aes(x = Month, y = PM2.5, shape = as.factor(Year), color = Site.Name)) +
  geom_point()
print(PMplot)
```



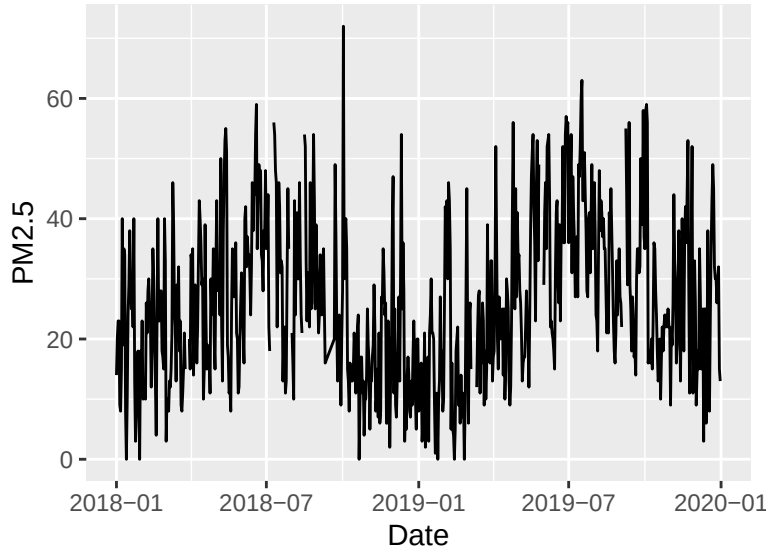

```
# Separate plot with facets
PMplot.faceted <-
  ggplot(EPAair, aes(x = Month, y = PM2.5, shape = as.factor(Year))) +
  geom_point() +
  facet_wrap(vars(Site.Name), nrow = 3)
print(PMplot.faceted)
```



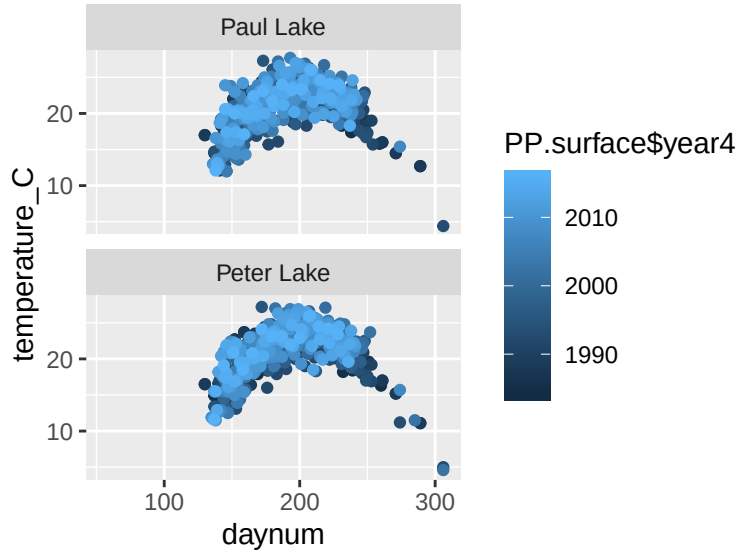

```
# Filter dataset within plot building and facet by multiple variables
PMplot.faceted2 <-
  ggplot(subset(EPAair, Site.Name == "Clemmons Middle" | Site.Name == "Leggett" |
                  Site.Name == "Bryson City"),
         aes(x = Month, y = PM2.5)) +
  geom_point() +
  facet_grid(Site.Name ~ Year)
print(PMplot.faceted2)
```




```
# Plot true time series with geom_line
PMplot.line <-
  ggplot(subset(EPAair, Site.Name == "Leggett"),
         aes(x = Date, y = PM2.5)) +
  geom_line()
print(PMplot.line)
```

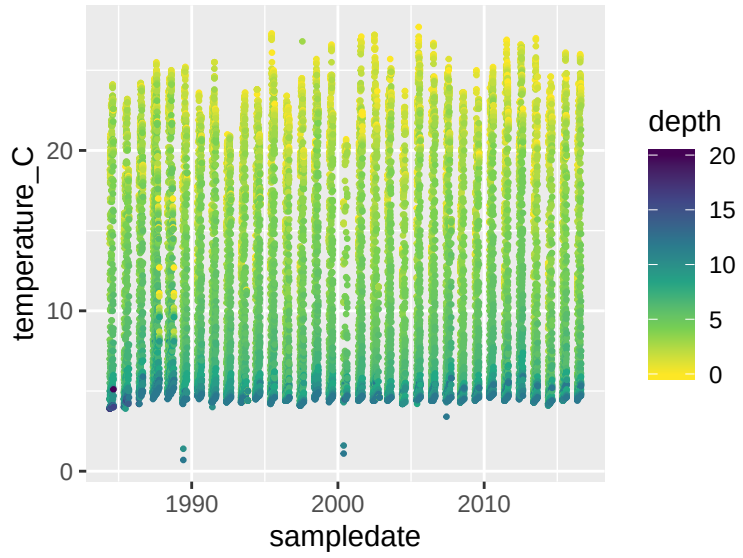



```
# Exercise: build your own scatterplots of PeterPaul.chem.nutrients

# 1.
# Plot surface temperatures by day of  year.
# Color your points by year, and facet by lake in two rows.
PP.surface <- PeterPaul.chem.nutrients%>%
  filter(depth ==0)

ggplot(PP.surface)+
  geom_point(aes(x=daynum, y = temperature_C, color = PP.surface$year4))+
  facet_wrap(PP.surface$lakename, nrow = 2)
```



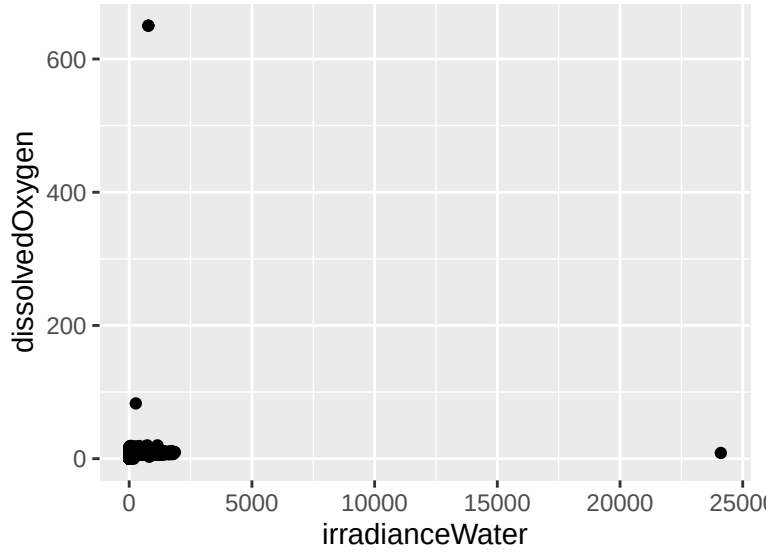

```
# can't get the subset version to work here...
#ggplot(subset(PeterPaul.chem.nutrients, depth == 0))+
#  geom_point(aes(x = daynum, y = temperature_C, color = year4))+
#  facet_wrap(PeterPaul.chem.nutrients$lakename)
```

```r
#2.
# Plot temperature by date. Color your points by depth.
# Change the size of your point to 0.5
ggplot(PeterPaul.chem.nutrients, aes(x=sampledate, y = temperature_C, color = depth))+
  geom_point(size = 0.5)+
  scale_color_viridis_c(direction = -1)
```



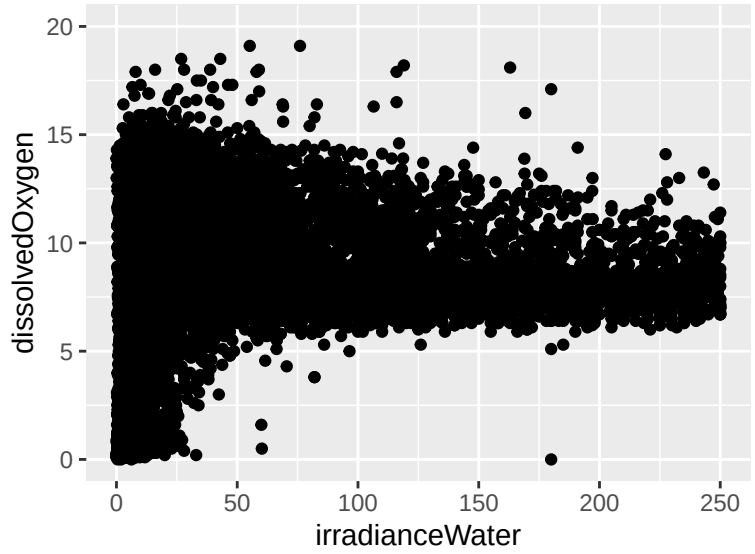
Plotting the relationship between two continuous variables: Scatterplot

```r
# Scatterplot
lightvsDO <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
  geom_point()
print(lightvsDO)
```

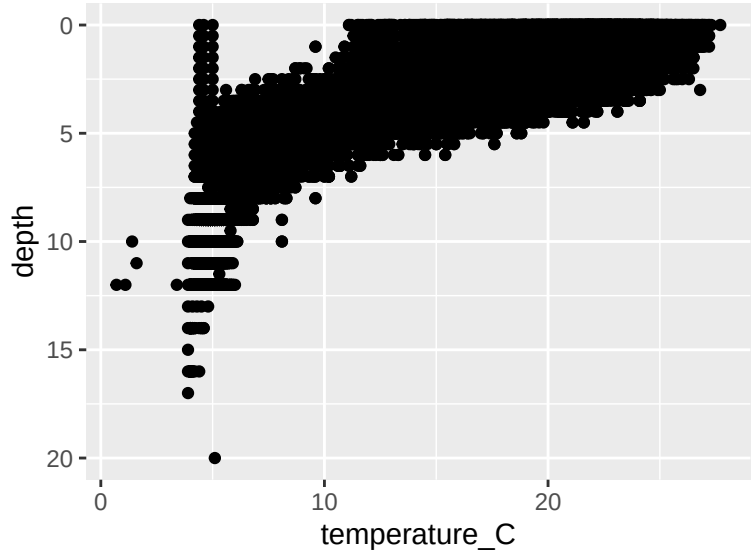



```r
# Adjust axes
lightvsDOfixed <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
```

```
  geom_point() +
  xlim(0, 250) +
  ylim(0, 20)
print(lightvsDOfixed)
```

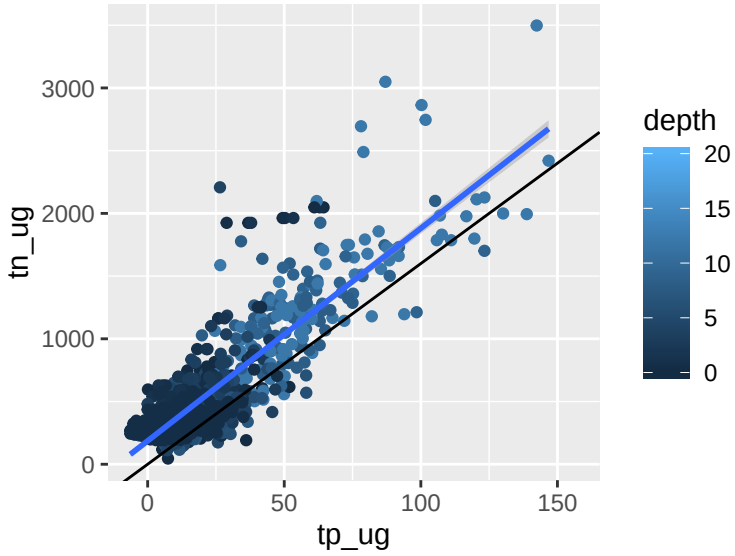



```
# Depth in the fields of limnology and oceanography is on a reverse scale
tempvsdepth <-
  ggplot(PeterPaul.chem.nutrients, aes(x = temperature_C, y = depth)) +
  #ggplot(PeterPaul.chem.nutrients, aes(x = temperature_C, y = depth, color = daynum)) +
  geom_point() +
  scale_y_reverse()
print(tempvsdepth)
```

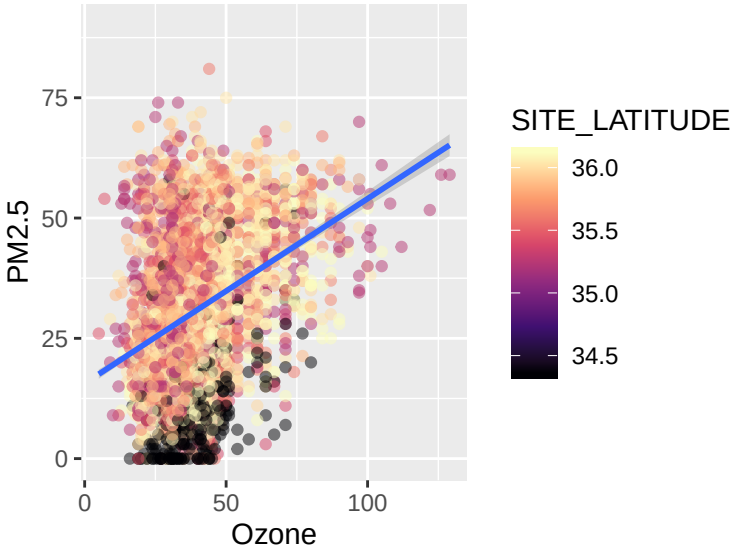



```
NvsP <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth)) +
  geom_point() +
  geom_smooth(method = lm) +
```

```
  geom_abline(aes(slope = 16, intercept = 0))
print(NvsP)
```




```
# Exercise: Plot relationships between air quality measurements

# 1.
# Plot AQI values for ozone by PM2.5, colored by latitude
# Make the points 50 % transparent
# Add a line of best fit for the linear regression of these variables.
ggplot(EPAair, aes(x = Ozone, y = PM2.5, color = SITE_LATITUDE))+
  geom_point(alpha = 0.5)+
  scale_color_viridis_c(option = "magma")+
  geom_smooth(method = lm)
```

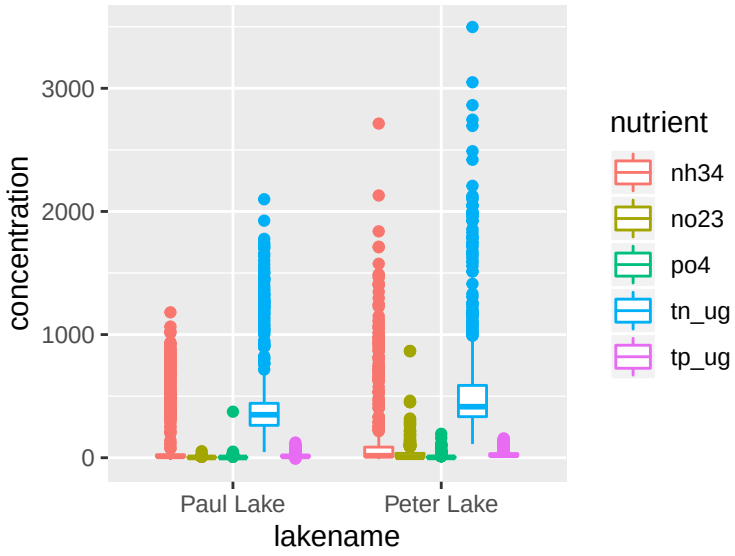


Plotting continuous vs. categorical variables

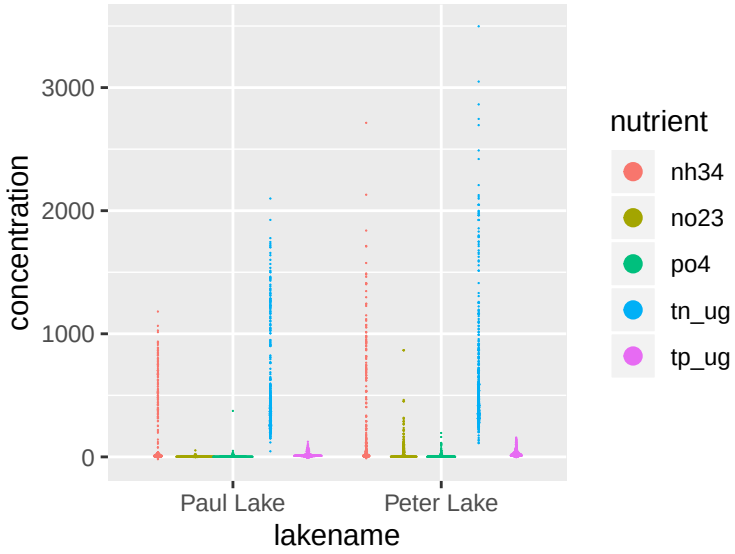
A traditional way to display summary statistics of continuous variables is a bar plot with error bars. Let's explore why this might not be the most effective way to display this type of data. Navigate to the Caveats page on Data to Viz (https://www.data-to-viz.com/caveats.html) and find the page that explores barplots and error bars.

What might be more effective ways to display the information? Navigate to the boxplots page in the Caveats section to explore further.

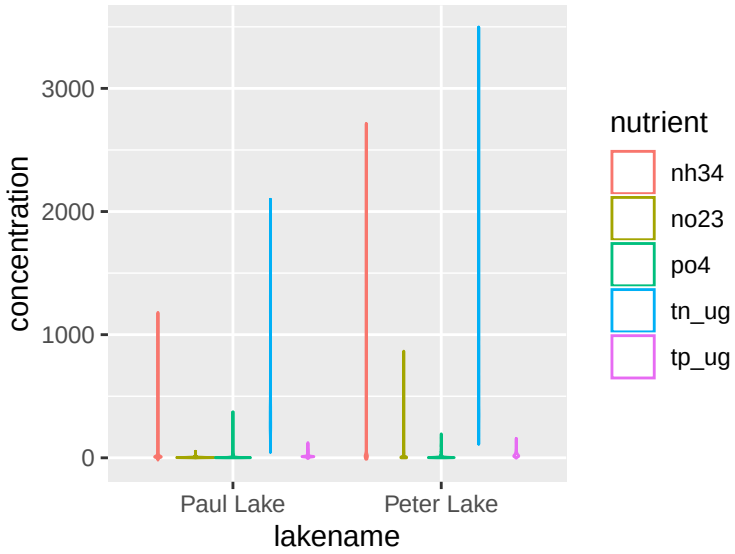
```
# Box and whiskers plot
Nutrientplot3 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_boxplot(aes(color = nutrient)) # Why didn't we use "fill"?
print(Nutrientplot3)
```




```
# Dot plot
Nutrientplot4 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_dotplot(aes(color = nutrient, fill = nutrient), binaxis = "y", binwidth = 1,
               stackdir = "center", position = "dodge", dotsize = 2) #
print(Nutrientplot4)
```

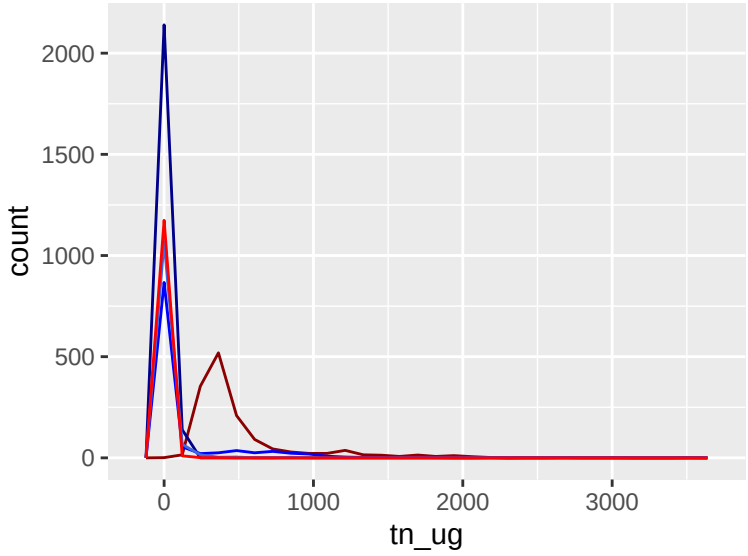



```
# Violin plot
Nutrientplot5 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_violin(aes(color = nutrient)) #
print(Nutrientplot5)
```



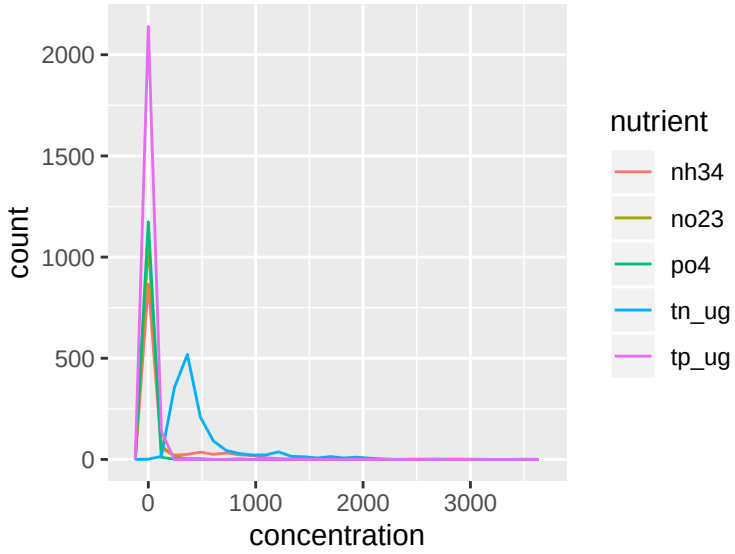

```
# Frequency polygons
# Using a tidy dataset
Nutrientplot6 <-
  ggplot(PeterPaul.chem.nutrients) +
  geom_freqpoly(aes(x = tn_ug), color = "darkred") +
  geom_freqpoly(aes(x = tp_ug), color = "darkblue") +
  geom_freqpoly(aes(x = nh34), color = "blue") +
  geom_freqpoly(aes(x = no23), color = "royalblue") +
  geom_freqpoly(aes(x = po4), color = "red")
print(Nutrientplot6)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Using a gathered dataset
Nutrientplot7 <-
  ggplot(PeterPaul.chem.nutrients.gathered) +
  geom_freqpoly(aes(x = concentration, color = nutrient))
print(Nutrientplot7)
```

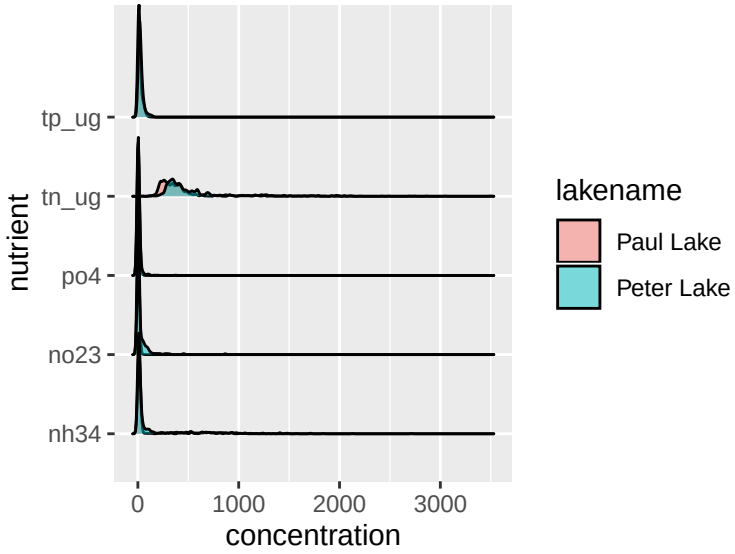
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Frequency polygons have the risk of becoming spaghetti plots.
# See https://www.data-to-viz.com/caveat/spaghetti.html for more info.

# Ridgeline plot
```

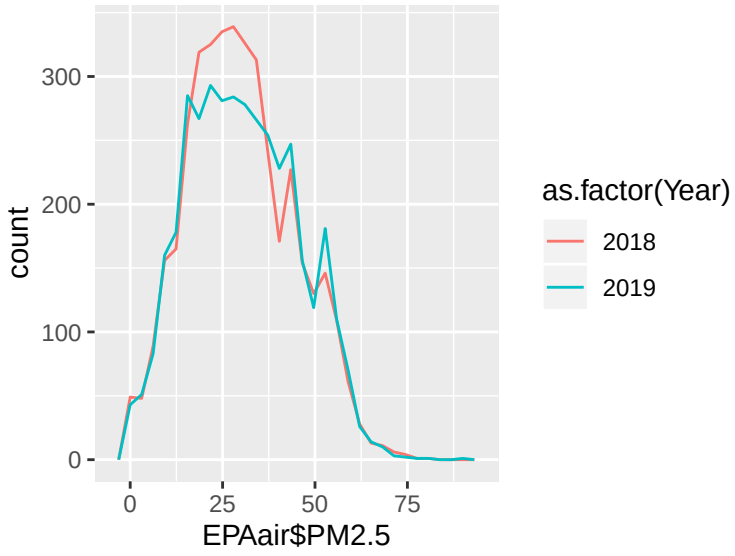
```
Nutrientplot6 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(y = nutrient, x = concentration)) +
  geom_density_ridges(aes(fill = lakename), alpha = 0.5)
print(Nutrientplot6)
```

```
## Picking joint bandwidth of 10.9
```

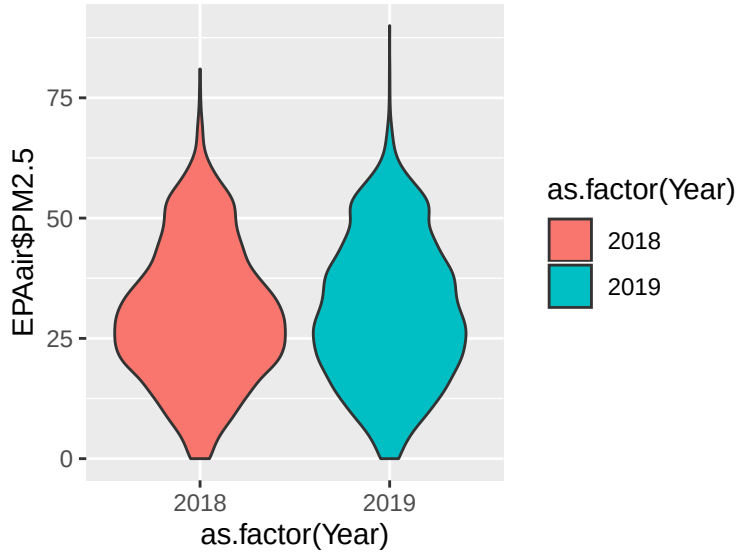



```
# Exercise: Plot distributions of AQI values for EPAair

# 1.
# Create several types of plots depicting PM2.5, divided by year.
# Choose which plot displays the data best and justify your choice.

ggplot(EPAair, aes(x= EPAair$PM2.5, color = as.factor(Year)))+
  geom_freqpoly()
```

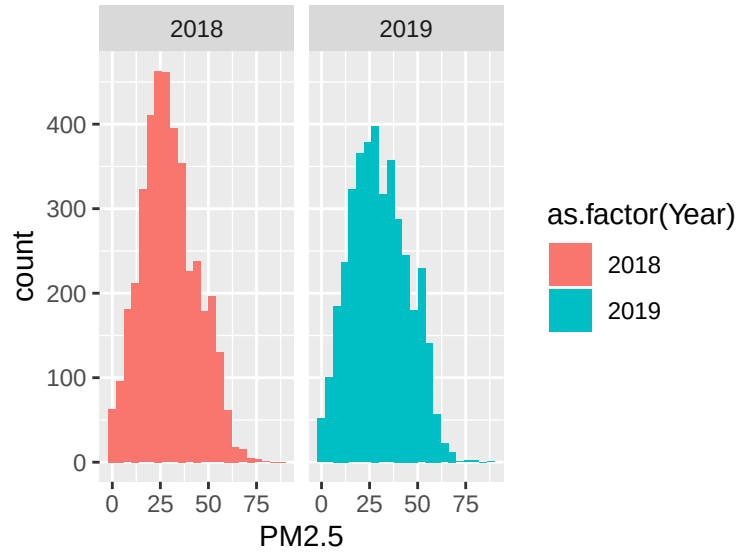
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
ggplot(EPAair)+
  geom_violin(aes(x = as.factor(Year), y = EPAair$PM2.5, fill = as.factor(Year)))
```

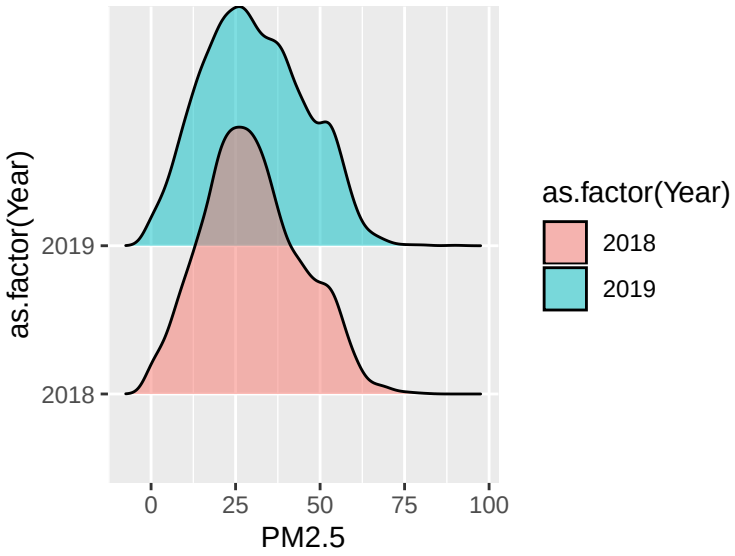



```
ggplot(EPAair)+
  geom_bar(aes(x=PM2.5, fill = as.factor(Year)), position = "dodge", binwidth = 4)+
  facet_wrap(EPAair$Year)
```




```
ggplot(EPAair)+
  geom_density_ridges(aes(y = as.factor(Year), x = PM2.5, fill = as.factor(Year)), alpha = 0.5)
```

```
## Picking joint bandwidth of 2.5
```


as.factor(Year)
2019
2018
0
25
50
75
100
PM2.5
as.factor(Year)
2018
2019