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] "/Users/emilymcnamara/Desktop/Env Data Analytics/Environmental_Data_Analytics_2020"
```

```
library(tidyverse)
library(ggridges)

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. You have to tell it which data set to look for within first ggplot function.
- 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 (type of graph you are making)

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

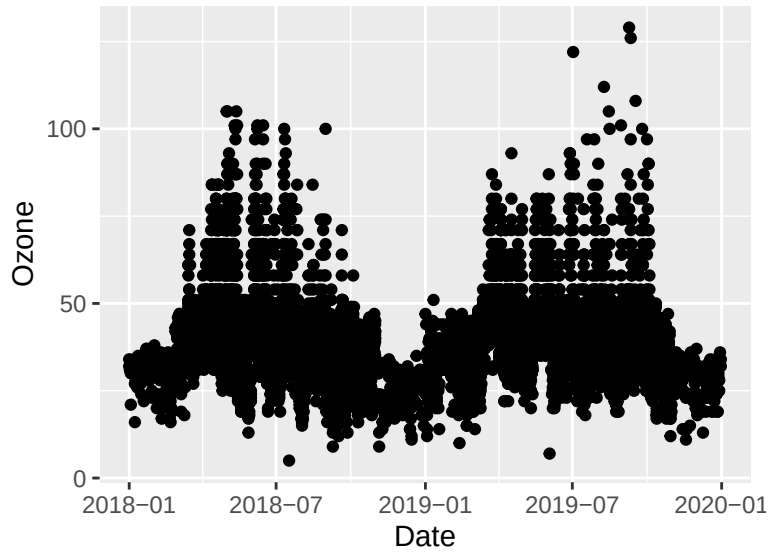
Aesthetics

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

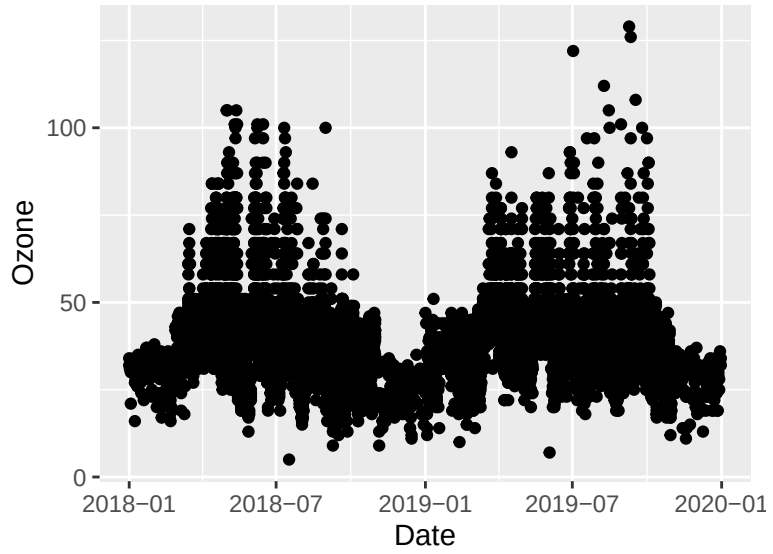
- color
- fill
- shape of points
- size
- transparency

Plotting continuous variables over time: Scatterplot and Line Plot

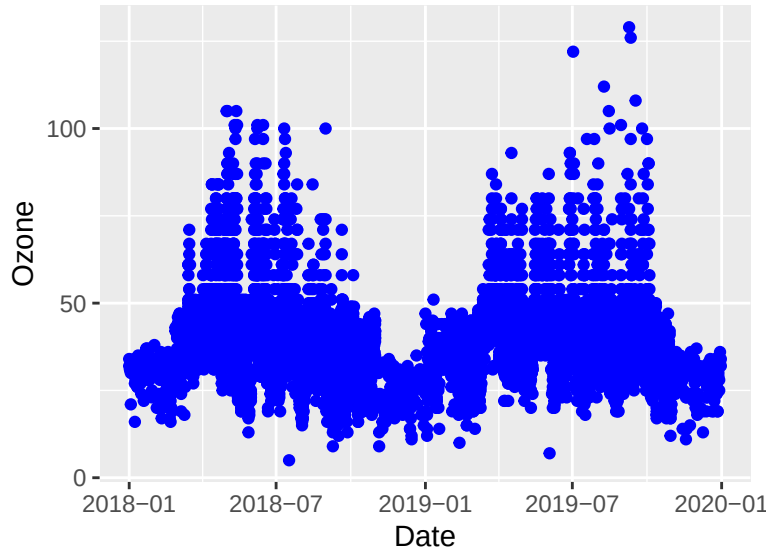
```
# Scatterplot

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



```r
#x and y are the column names. If run first line of code, it just makes a nice graph without any data.
# AQI values on y-axis for ozone, time on x-axis. time/date will go on x-axis
# Not in environment, so it's not stored. Have to give it a name like example below.

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

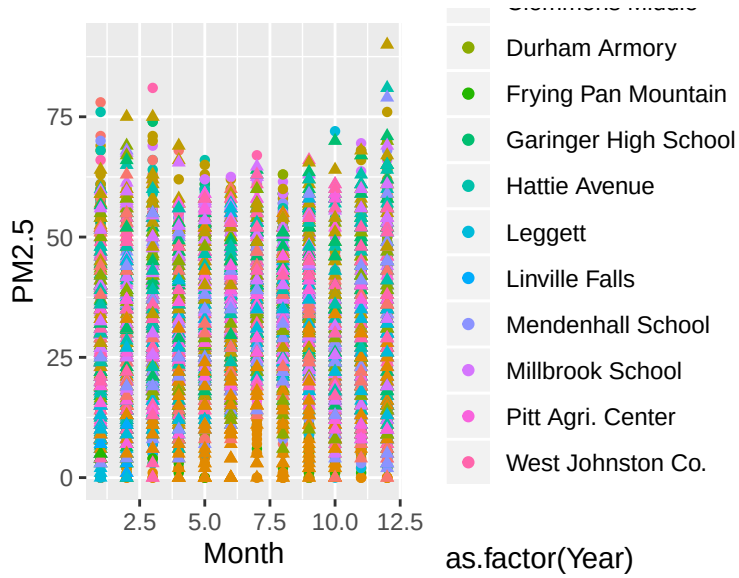



```r
# print function will knit it when go to pdf

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




```r
# The parentheses were wrong. Color shouldn't be inside the asethetic layer. Need to add a parenthesis
# Can put your asethetics in the beginning line like in line 80 or in the the gemo_point line. Doesn't
```

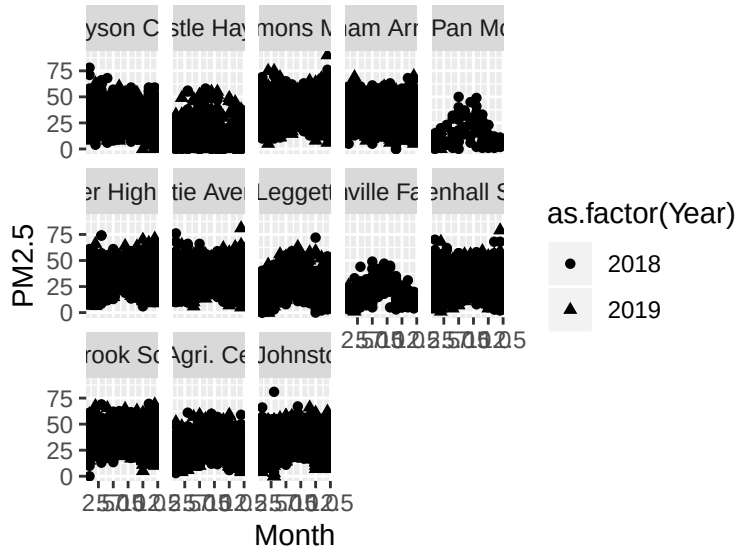
```
# Add additional variables
PMplot <-
  ggplot(EPAair, aes(x = Month, y = PM2.5, shape = as.factor(Year),
                     color = Site.Name)) +
  geom_point()
print(PMplot)
```



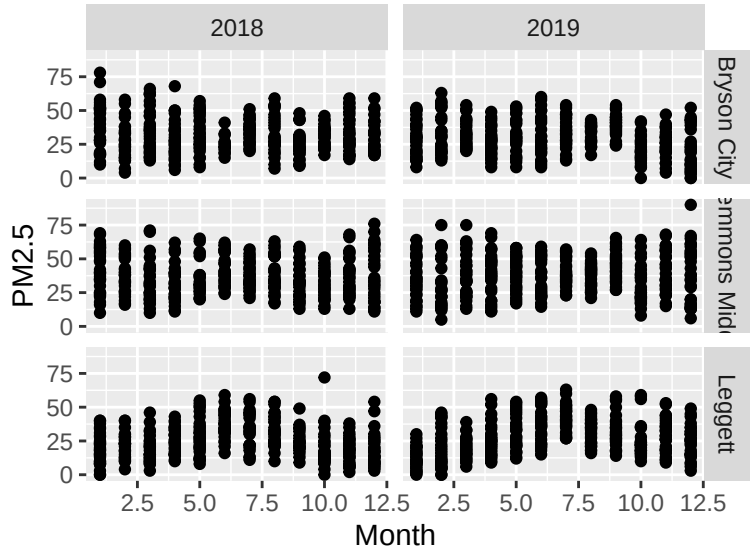

```
# Want to look at it over 'month' so make that x-axis, and then PM2.5 values will be y-axis, and what t
# have to say 'as.factor(Year)' because year can be an integer like below, and need it to be a shape ae
class(EPAair$Year)
```

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

```
# PMplot looks cool, but it isn't effective because too many sites, months in integer values, and can't


# 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)
```

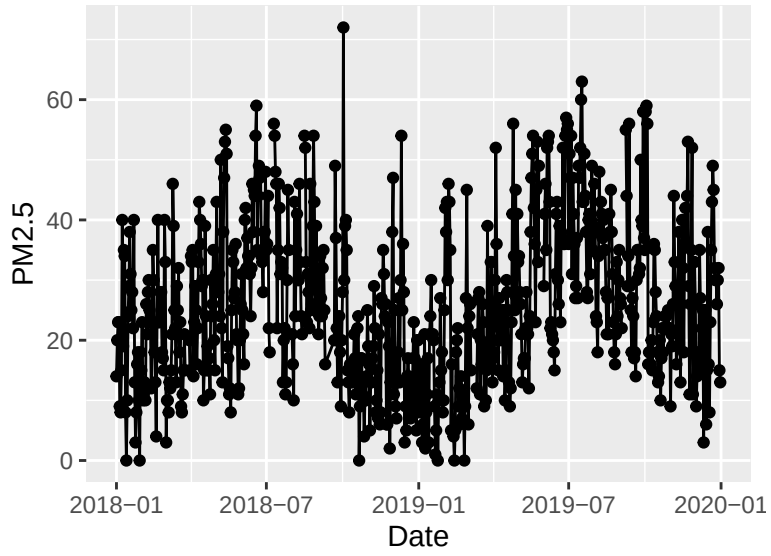



```
# Want to include everything except color aesthetic. then tell it to make a geom_point. THEN say make d
# If want them to be free can let the two different options differ rather than have them fixed.
# Takes all 13 sites, plots them in alphabetical order and you can see differences among sites, seasona
# nrow = 3, means for all facets you want 3 rows of graphs.

# 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)
```

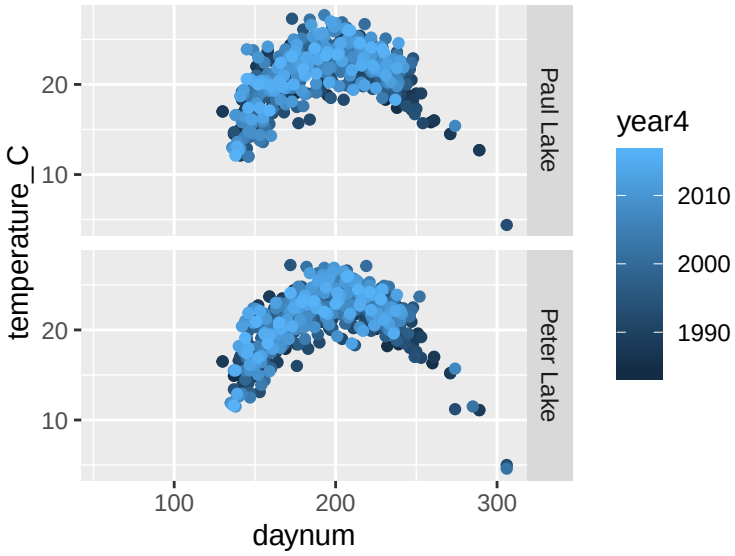



```
# apply whatever you would do for a filter function, but instead you're subsetting. could also do: "(EP
# Then use same aesthetics. Make geom_point
# Facet_grid allows you to facet 2 different variables instead of having years be two different shapes
```

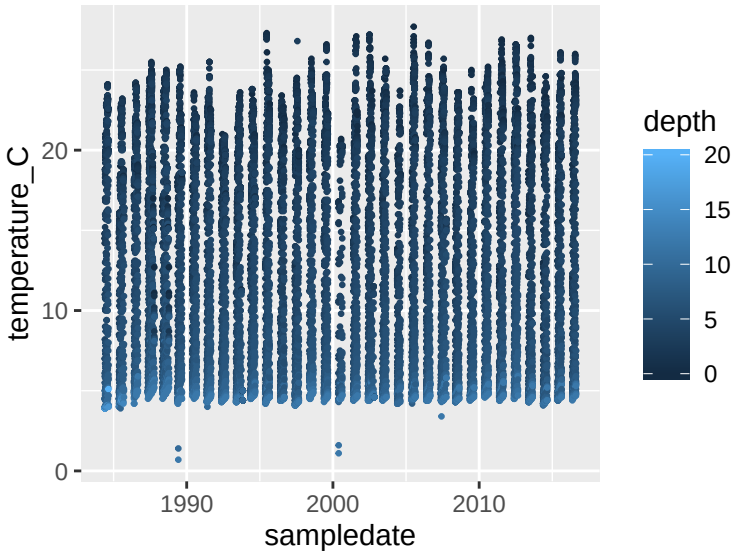
```r
# Plot true time series with geom_line
PMplot.line <-
  ggplot(subset(EPAair, Site.Name == "Leggett"),
         aes(x = Date, y = PM2.5)) +
  geom_line() +
  geom_point()
print(PMplot.line)
```




```r
# time series: don't want to combine each of the sites because tracking air quality over time. have to
# might want to make a subset to just look at one site name. Could also change colors if want to look a
# If want to add actual points, then after geom_line add geom_point

# 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.temp <-
  ggplot(subset(PeterPaul.chem.nutrients, depth == 0)) +
  geom_point(aes(x = daynum, y = temperature_C, color = year4)) +
  facet_grid(vars(lakename))
print(PP.temp)
```



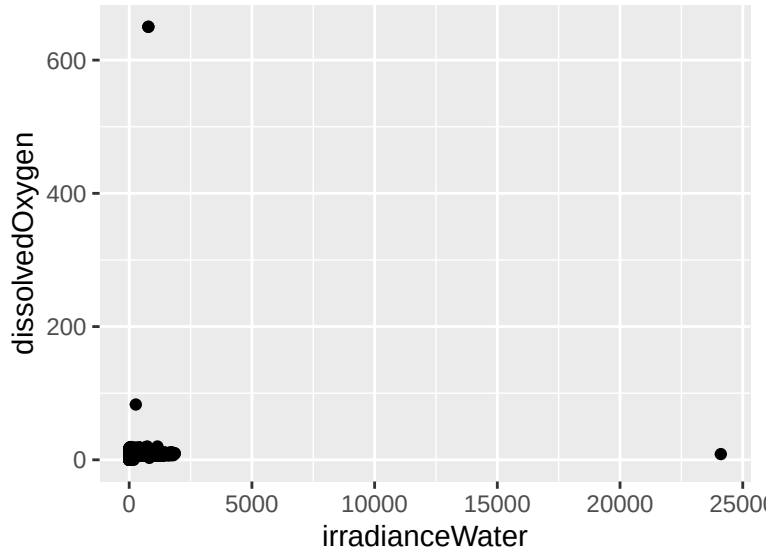

```
# or facet_wrap(vars(lakename), nrow= 2)
# aes are in geom_point instead of plot but would be the same. Personal preference is to put aesthetics
# have to do subset( depth == 0) because it asks for surface temps

#2.
# Plot temperature by date. Color your points by depth.
# Change the size of your point to 0.5

PP.temp2 <-
  ggplot(PeterPaul.chem.nutrients) +
  geom_point(aes(x = sampledate, y = temperature_C, color = depth),
             size = 0.5)
print(PP.temp2)
```



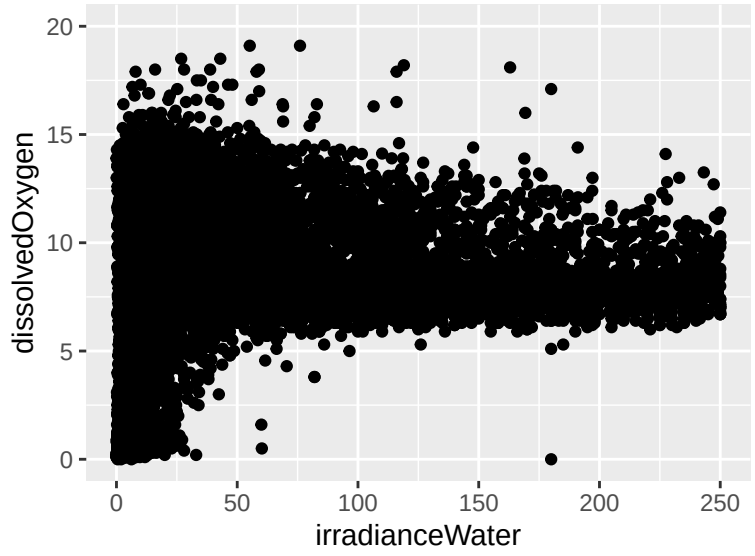

```
# this plot is not a true time series because there are two lakes and at multiple days. For true time s
```

Plotting the relationship between two continuous variables: Scatterplot

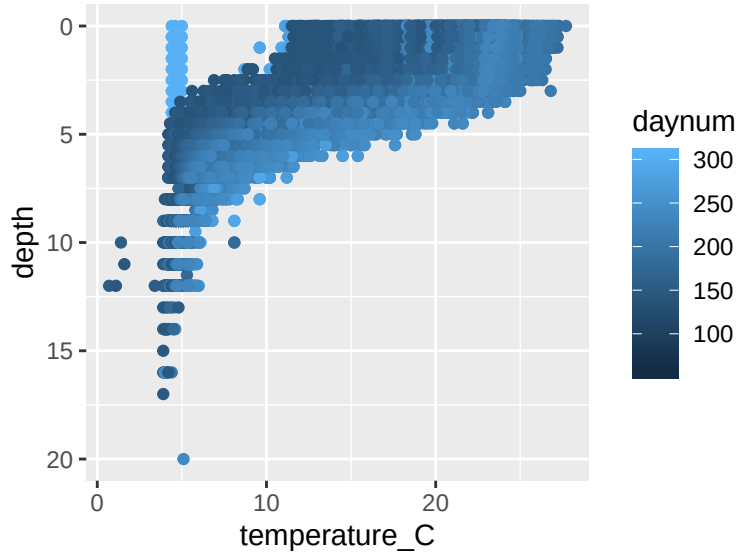
```
# Scatterplot
lightvsDO <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
  geom_point()
print(lightvsDO)
```




```
# coule flip x and y because time isn't involved
# huge outliers in this graph. Because we know these aren't possible data points, can adjust axis


# Adjust axes
lightvsDOfixed <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
  geom_point() +
  xlim(0, 250) +
  ylim(0, 20)
print(lightvsDOfixed)
```

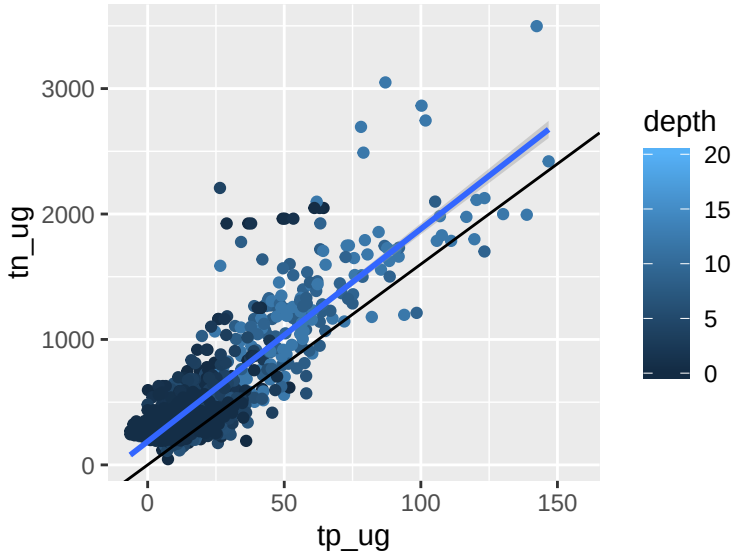



```
# This zooms in on the part of the graph that's of interest.
# shows that across different irradiance, see a pretty similar amount of dissolved oxygen. at lower lev

# 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)
```




```
# know it's dark at greater depth, so we can flip scales, so that zero depth is at top of y-axis.
# aes = temp and depth. then make geom_point to show data points. then use 'scale_y_reverse' to flip ax
# shows the deeper, the colder
# second ggplot line (line 196) shows how temp changes over days/seasons

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)
```



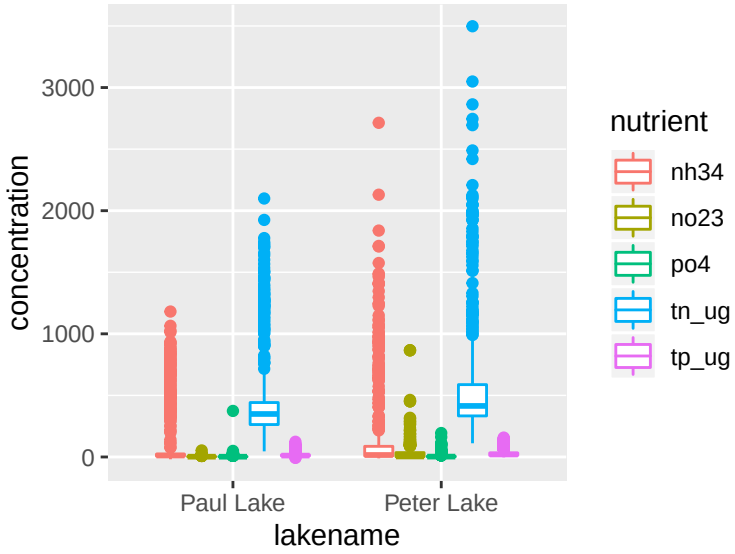

```
# geom_smooth: allows u to draw a line of best fit. default method takes moving address of whatever you
# total P = x-axis, total N = y-axis. it's colored by depth. blue line shows linear relationship. shadi
#geom_abline is y = mx + b so have to provide a slope and an intercept. added 16 and 0 because it's not

# 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.
```

Plotting continuous vs. categorical variables

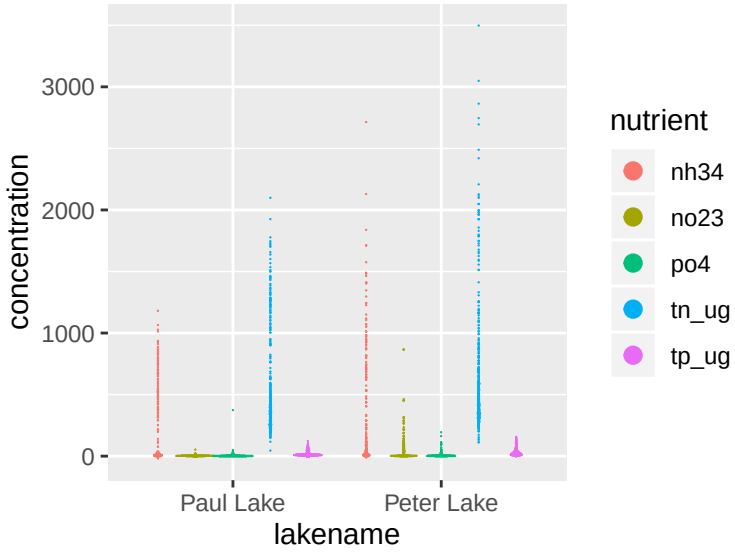
A traditional way to display summary statistics of continuous variables is a bar plot with error bars. Let's explore why this might not be the most effective way to display this type of data. Navigate to the Caveats page on Data to Viz (https://www.data-to-viz.com/caveats.html) and find the page that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

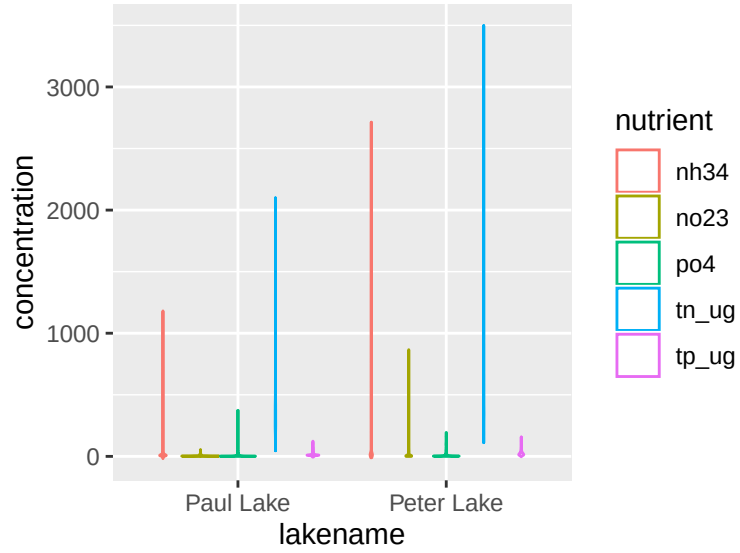



```
# plots different nutrients in lakes P and P, but want to separate by lake name. so use gathered datase
# tell it to make geom_boxplot and can split aesthetics by saying color = nutrient
# we didn't use fill because a lot of our concentrations are near 0, so can't actually see what that fi

# 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)
```



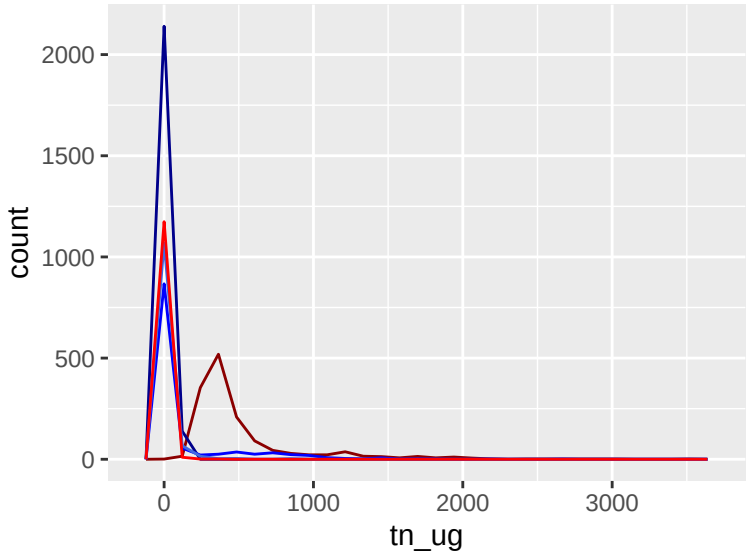

```
# geom_dot plot instead of geom_box plot. needs a bunch of different info. can see their meaning in "he
# every measurement in dataset is plotted as a point. for data that is the same, it plots it side-by-si

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

```
  geom_violin(aes(color = nutrient)) #
print(Nutrientplot5)
```



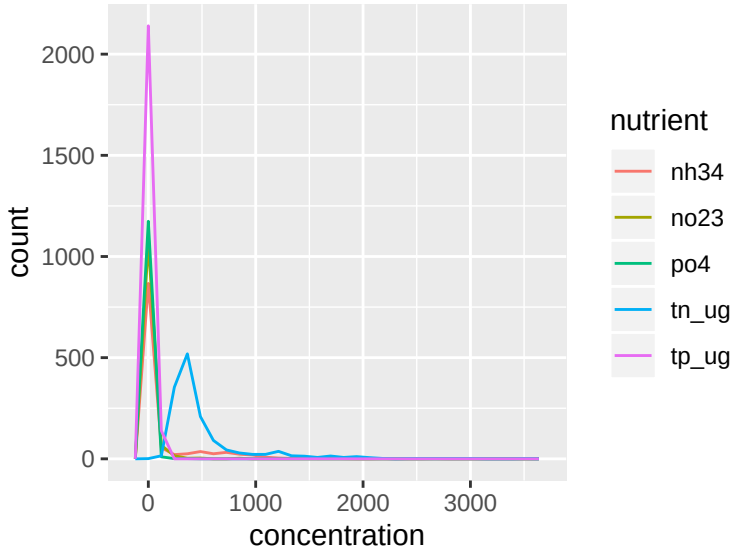

```
# looks similar to dot plot. wider or skinnier based on how many values are centered around a point

# Frequency polygons
# Using a tidy dataset (aka not gathered) would have to add a layer for each dataset and a color outsid
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)
```

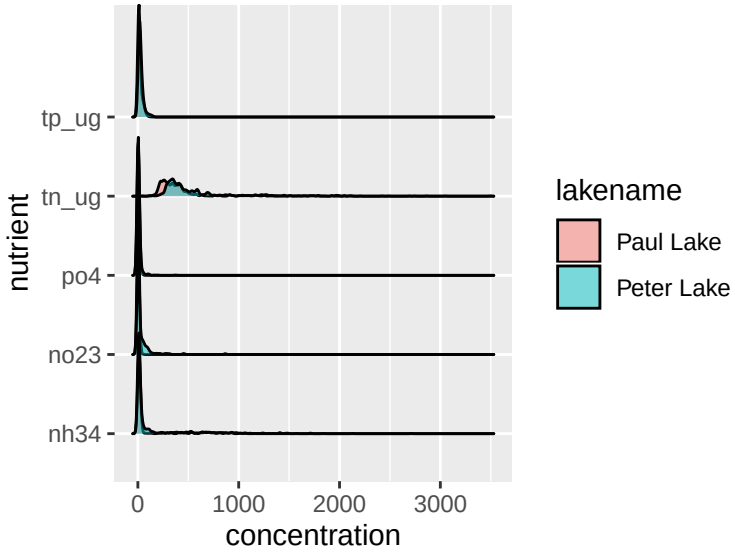
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# same data as frequency polygon (line 256) but much easier and more clear

# 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
```




```
# alpha = transparency piece and specified outside of aesthetic

# 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.
```