5: Part 1 - Data Visualization Basics

Environmental Data Analytics | John Fay and Luana Lima | Developed by Kateri Salk

Fall 2023

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. A good 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

```r
chooseCRANmirror(ind = 1)
library(tidyverse);library(lubridate);library(here)
install.packages("ggridges")
```

```
## package 'ggridges' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ziyaw\AppData\Local\Temp\RtmpG2Kmea\downloaded_packages
```

```r
library(ggridges)
here()
```

```
## [1] "C:/Users/ziyaw/Downloads/EDE_Fall2023"
```

```r
PeterPaul.chem.nutrients <-
  read.csv(here("Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv"), stringsAsFa
PeterPaul.chem.nutrients.gathered <-
  read.csv(here("Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Processed.csv"), stringsAsFact
EPAair <- read.csv(here("Data/Processed/EPAair_O3_PM25_NC1819_Processed.csv"), stringsAsFactors = TRUE)

EPAair$Date <- ymd(EPAair$Date)
PeterPaul.chem.nutrients$sampledate <- ymd(PeterPaul.chem.nutrients$sampledate)
PeterPaul.chem.nutrients.gathered$sampledate <- ymd(PeterPaul.chem.nutrients.gathered$sampledate)
```

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) #intersects a and b
- 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

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

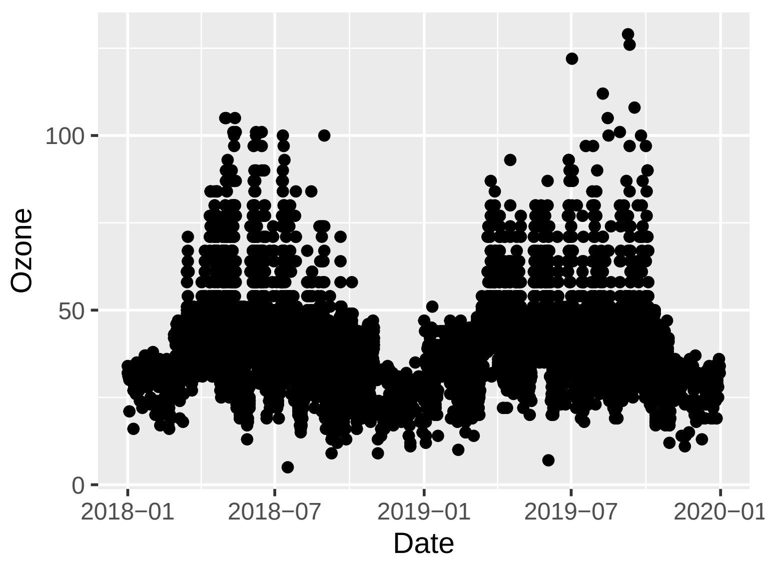


Figure 1: figure 1

```r
#aes can be specified inside of ggplot() or in geom_point()
O3plot <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
##useful when we want to have several plots together
print(O3plot)
```

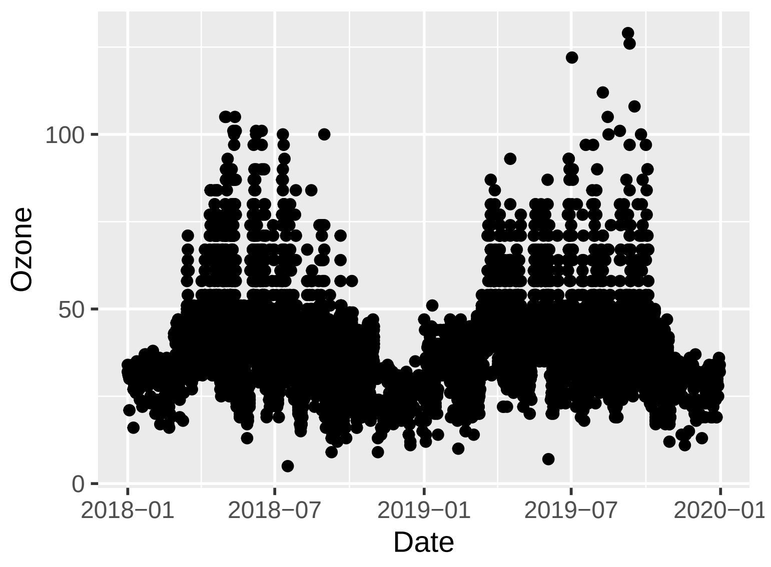


Figure 2: figure 1

```r
# Fix this code
O3plot2 <- ggplot(EPAair) +
  ##geom_point(aes(x = Date, y = Ozone, color = "blue")) ###when we do this, we are adding another laye
  geom_point(aes(x = Date, y = Ozone), color = "blue") ###we need to specify the color outside of aes
print(O3plot2)
```

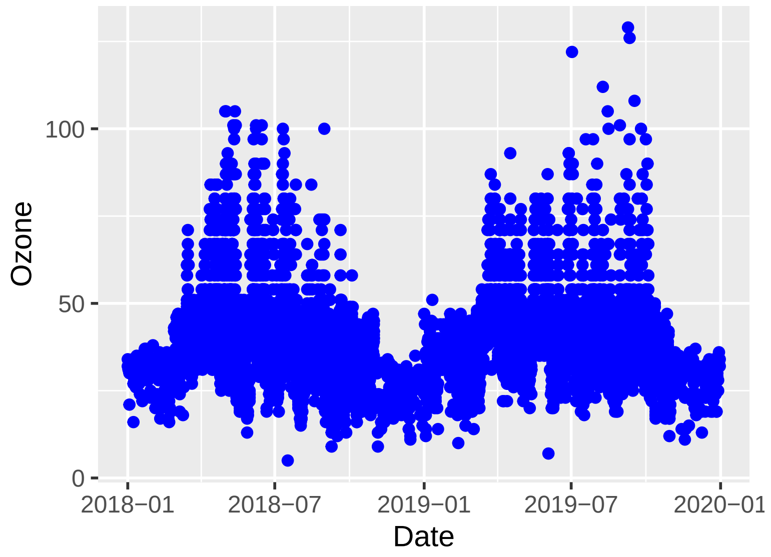


Figure 3: figure 1

```
# Add additional variables
# How could you automatically assign a marker color to a variable?
PMplot <-
  ggplot(EPAair, aes(x = Month, y = PM2.5, shape = as.factor(Year), color = Site.Name)) +
  geom_point() ##assign shapes depending on the year
print(PMplot)
```

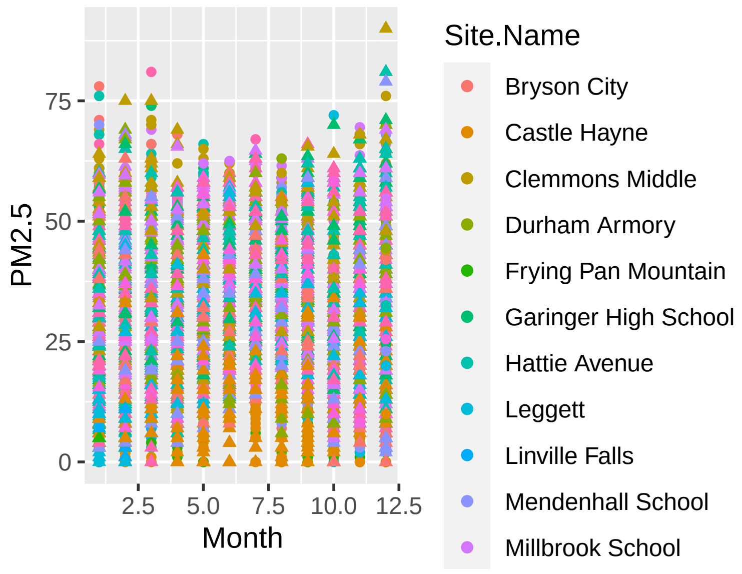


Figure 4: figure 1

```r
# 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) ###facet_wrap: creating different plots for different sites, 3 r
print(PMplot.faceted)
```

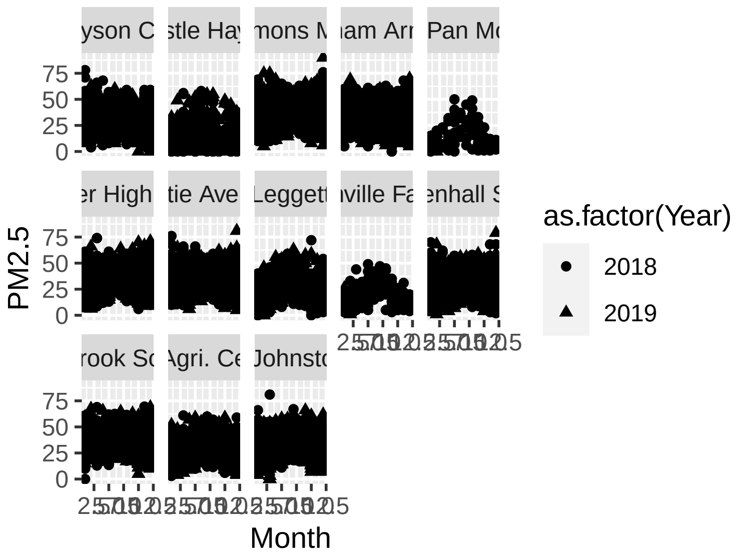


Figure 5: figure 1

```r
 # 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"), ###only include those three sites
         aes(x = Month, y = PM2.5)) +
  geom_point()
  facet_grid(Site.Name ~ Year) ###column: year, rows:site name
```

```
## <ggproto object: Class FacetGrid, Facet, gg>
##     compute_layout: function
##     draw_back: function
##     draw_front: function
##     draw_labels: function
##     draw_panels: function
##     finish_data: function
##     init_scales: function
##     map_data: function
##     params: list
##     setup_data: function
##     setup_params: function
##     shrink: TRUE
##     train_scales: function
##     vars: function
##     super:  <ggproto object: Class FacetGrid, Facet, gg>
```

```
print(PMplot.faceted2)
```

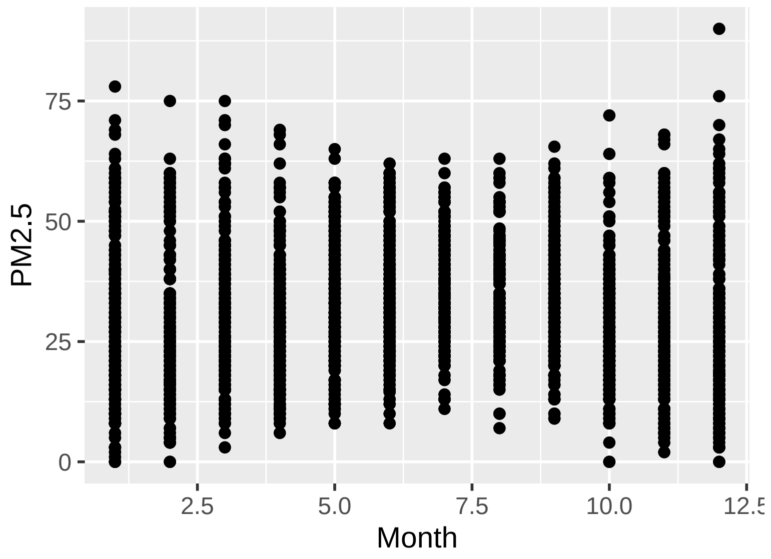


Figure 6: figure 1

```
# 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)
```

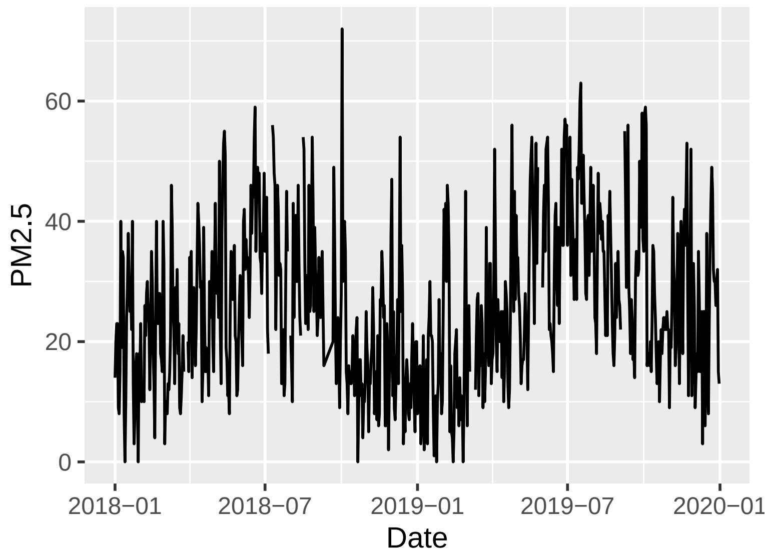
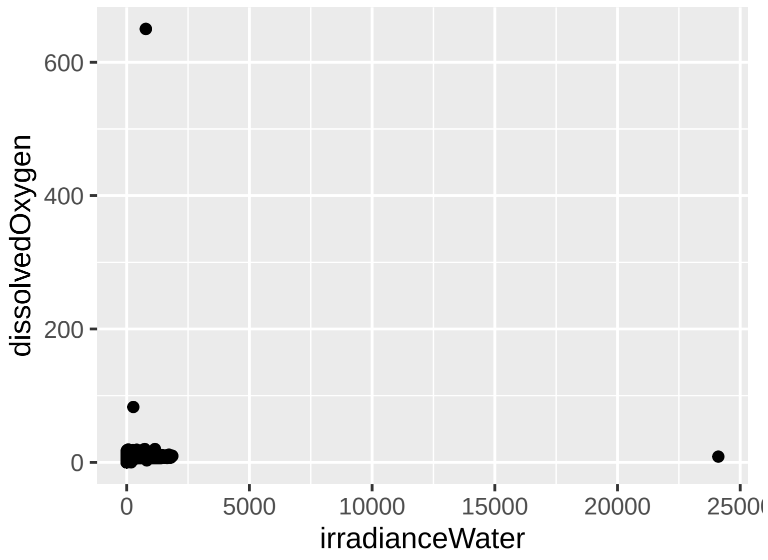


Figure 7: figure 1

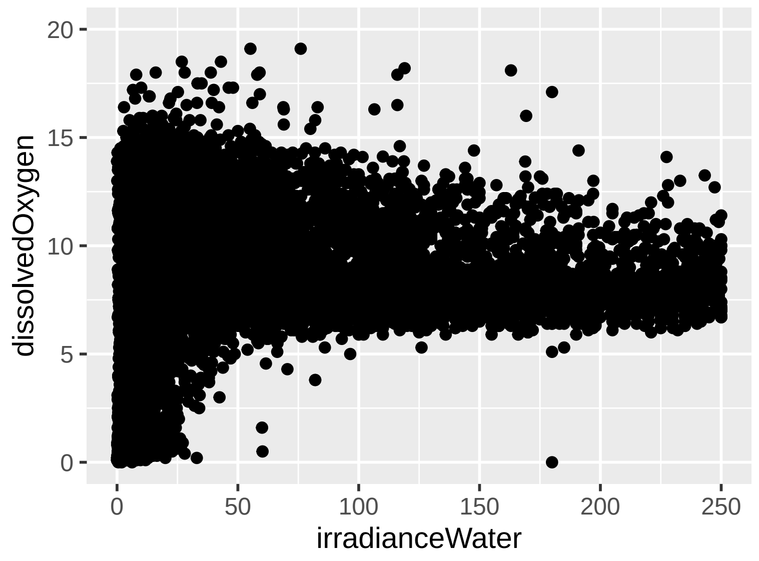
```
##time series plot
```

Plotting the relationship between two continuous variables: Scatterplot

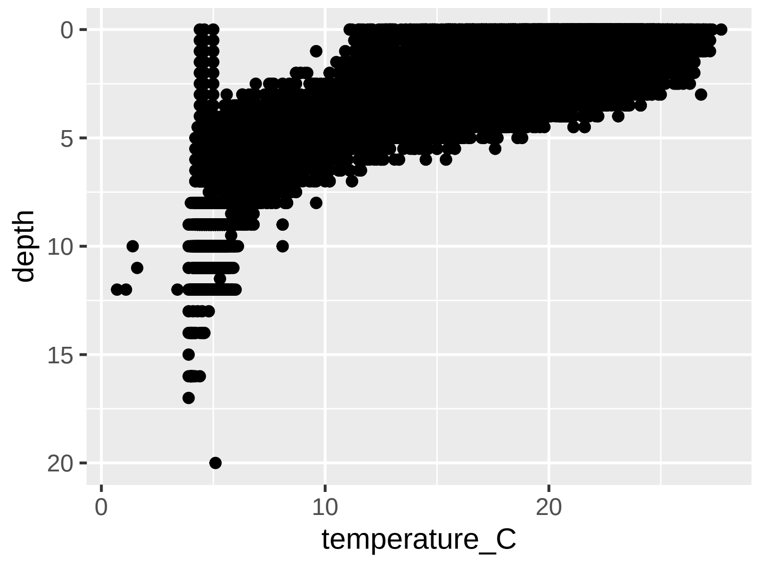
```
# Scatterplot
lightvsDO <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) + ##visualizing 2 vai
  geom_point()
print(lightvsDO) ##too far from each other
```



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

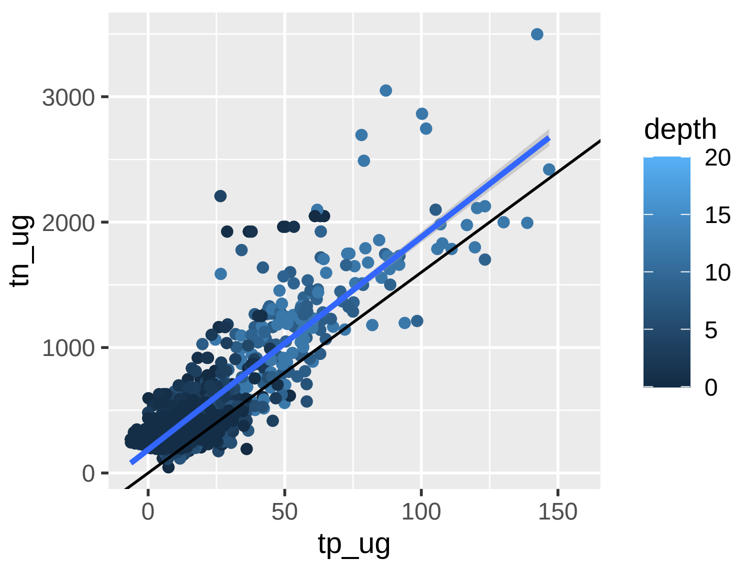


```r
# Depth in the fields of limnology and oceanography is on a reverse scale
tempvsdepth <-
  ggplot(PeterPaul.chem.nutrients, aes(x = temperature_C, y = depth)) +
  #ggplot(PeterPaul.chem.nutrients, aes(x = temperature_C, y = depth, color = daynum)) +
  geom_point() +
  scale_y_reverse() ###reversing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



```r
NvsP <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth)) +
  geom_point() +
  geom_smooth(method = lm) + ###trendline: linear
  geom_abline(aes(slope = 16, intercept = 0)) #intercept: the x value it intercepts
print(NvsP)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

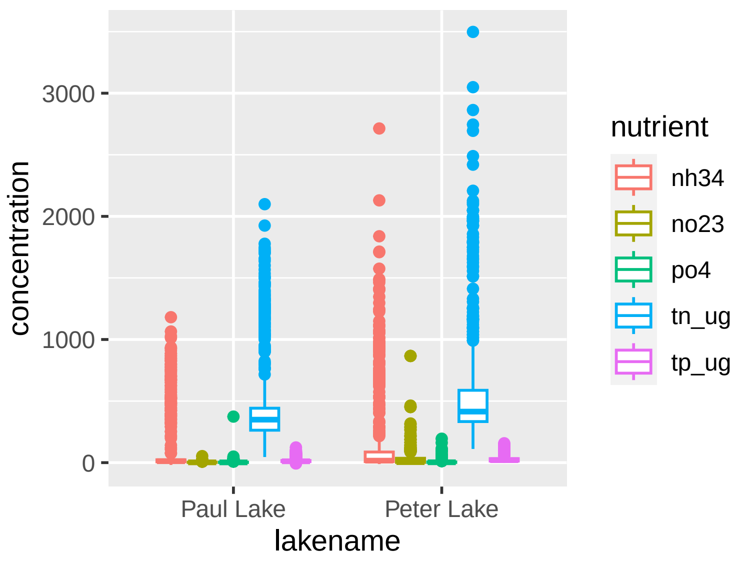


Plotting continuous vs. categorical variables

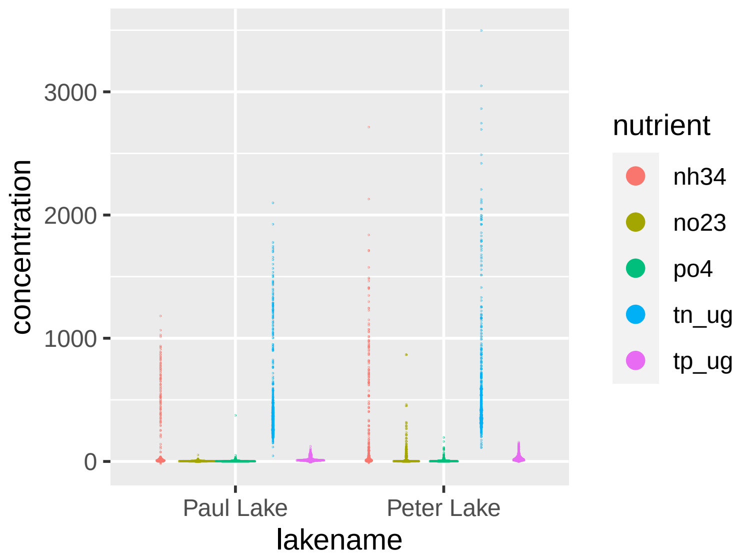
A traditional way to display summary statistics of continuous variables is a bar plot with error bars. Let's explore why this might not be the most effective way to display this type of data. Navigate to the Caveats page on Data to Viz (https://www.data-to-viz.com/caveats.html) and find the page that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

```r
# 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"? ##fill is the filling of the boxplot,
print(Nutrientplot3)
```

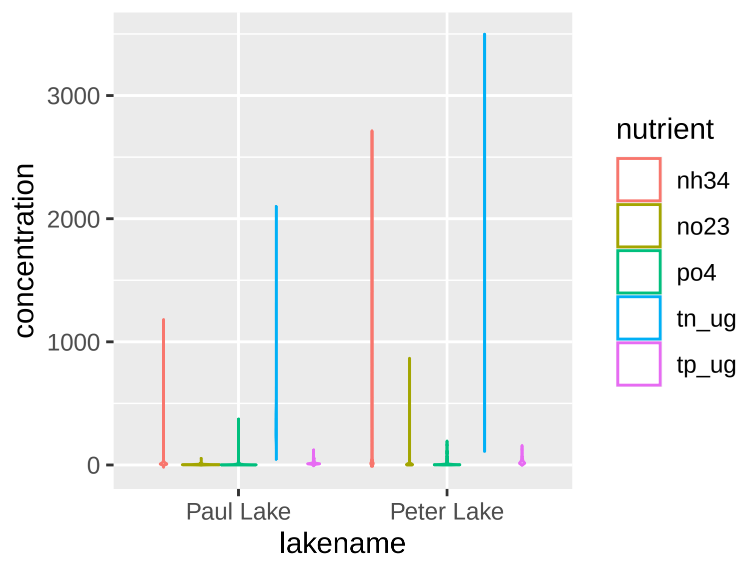


```r
# Dot plot
Nutrientplot4 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_dotplot(aes(color = nutrient, fill = nutrient), binaxis = "y", binwidth = 1,
               stackdir = "center", position = "dodge", dotsize = 2) #
print(Nutrientplot4)
```



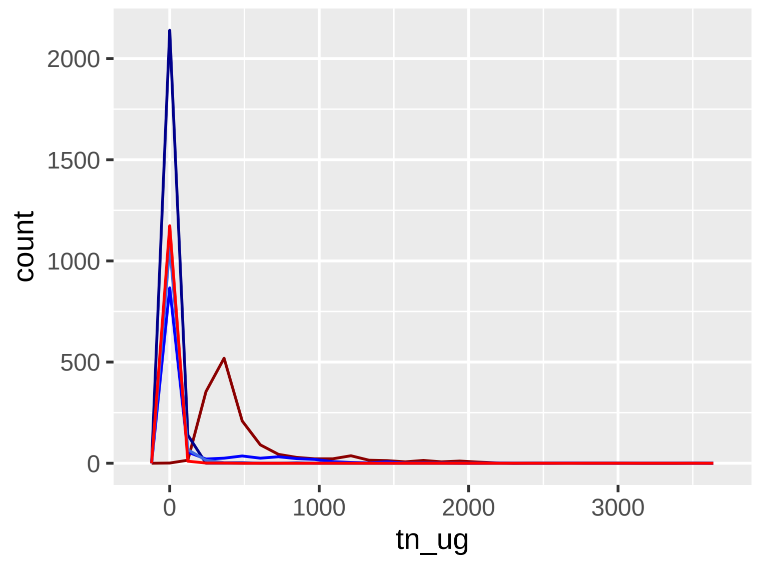
```r
####better representation of the data
##binaxis: stacking the observations over y
#### stackdir: where to stack the points
#### position: dodge: so that the dots do not overlay, they go to the side (hence horizontal "lines" in

# Violin plot ###also shows density plot in addition to summary stats that boxplots provide
Nutrientplot5 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_violin(aes(color = nutrient)) #
print(Nutrientplot5)
```



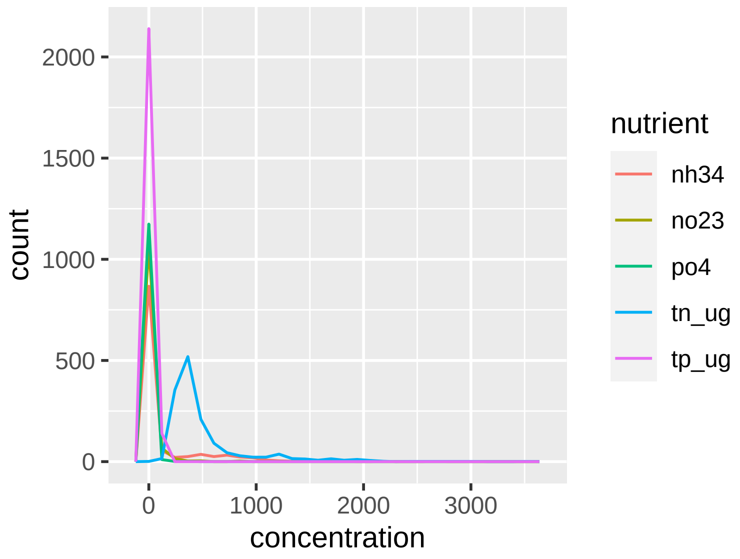
```r
# Frequency polygons
# Using a tidy dataset
Nutrientplot6 <- ######not a good way to do this
  ggplot(PeterPaul.chem.nutrients) +
  geom_freqpoly(aes(x = tn_ug), color = "darkred") +
  geom_freqpoly(aes(x = tp_ug), color = "darkblue") +
  geom_freqpoly(aes(x = nh34), color = "blue") +
  geom_freqpoly(aes(x = 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'.
```



```r
# Using a gathered dataset ###gathered has all the nutrients in one column and conc in antoher: longer
Nutrientplot7 <-
  ggplot(PeterPaul.chem.nutrients.gathered) +
  geom_freqpoly(aes(x = concentration, color = nutrient))
print(Nutrientplot7)
```

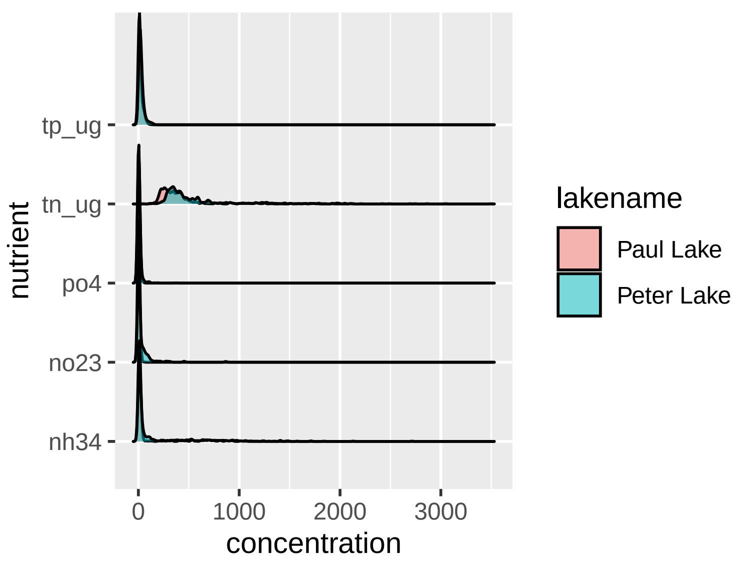
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
# 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) ###fill: fil below the curve
```

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


tp_ug
tn_ug
po4
no23
nh34
nutrient
0
1000
2000
3000
concentration
lakename
Paul Lake
Peter Lake