5: Part 1 - Data Visualization Basics

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

Fall 2024

List of Figures

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. #use this link for a decision tree of what you should use for data visualization

Set Up

```r
getwd()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```r
library(tidyverse);
library(lubridate);
library(here)
library(ggridges)
here()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```r
PeterPaul.chem.nutrients <-
  read.csv(here("Data/Processed_KEY/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv"), string
PeterPaul.chem.nutrients.gathered <-
  read.csv(here("Data/Processed_KEY/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Processed.csv"), stringsAs
EPAair <- read.csv(here("Data/Processed_KEY/EPAair_O3_PM25_NC1819_Processed.csv"), stringsAsFactors = T

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)
- 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
# Scatterplot
ggplot(EPAair, aes(x = Date, y = Ozone)) +
  geom_point() #if you want a point do this
```

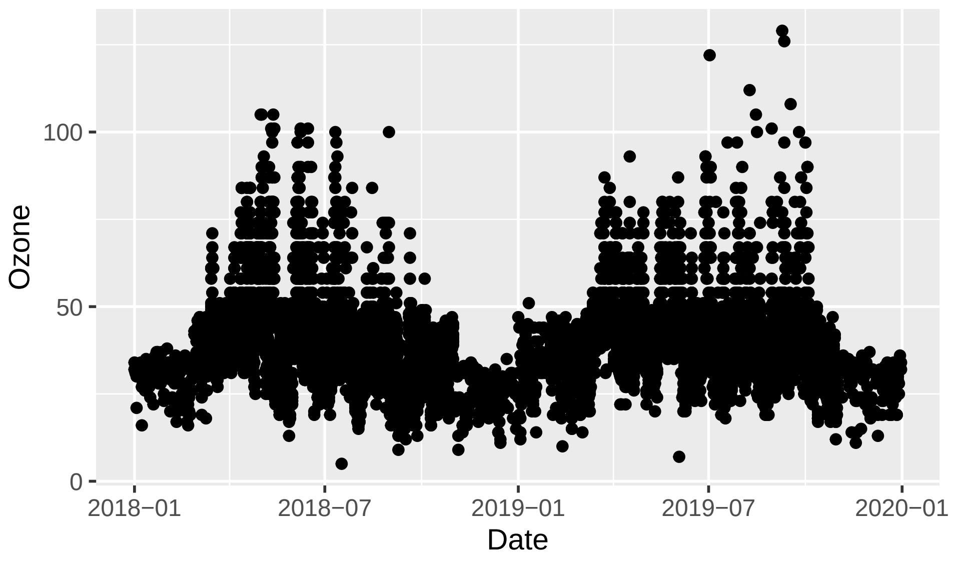


Figure 1: Scatter Plot

```r
O3plot <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
print(O3plot) #add two plots if you want and/or other layers you want and then just printing that objec
```

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

```r
# 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 certain attributes to different things
print(PMplot)
```

```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) #based off sites not colors
print(PMplot.faceted)
```

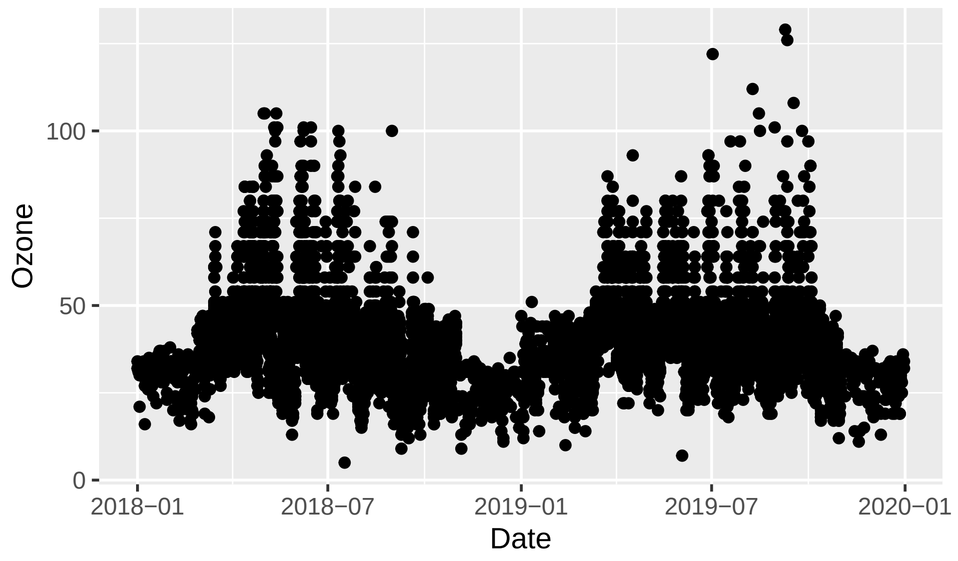


Figure 2: Scatter Plot

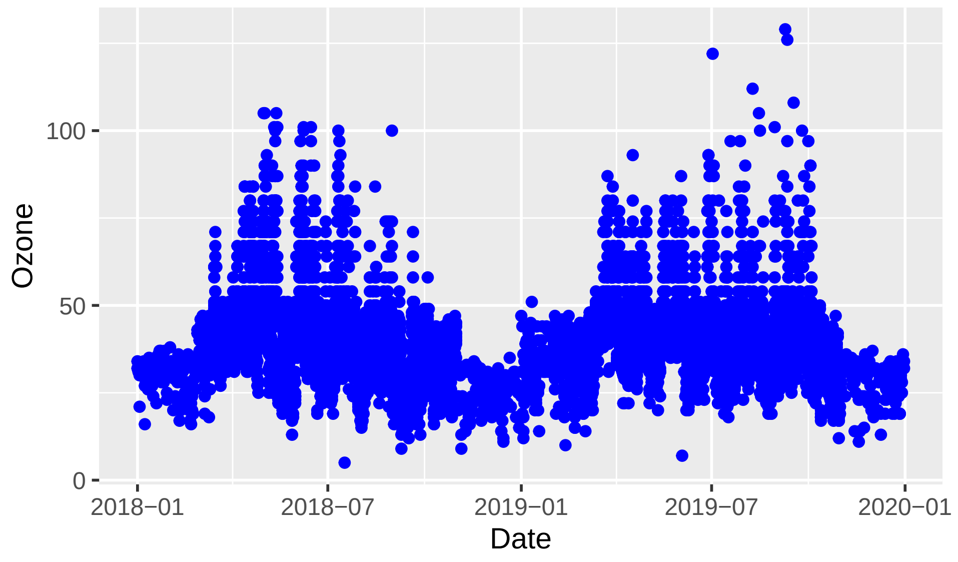


Figure 3: Scatter Plot

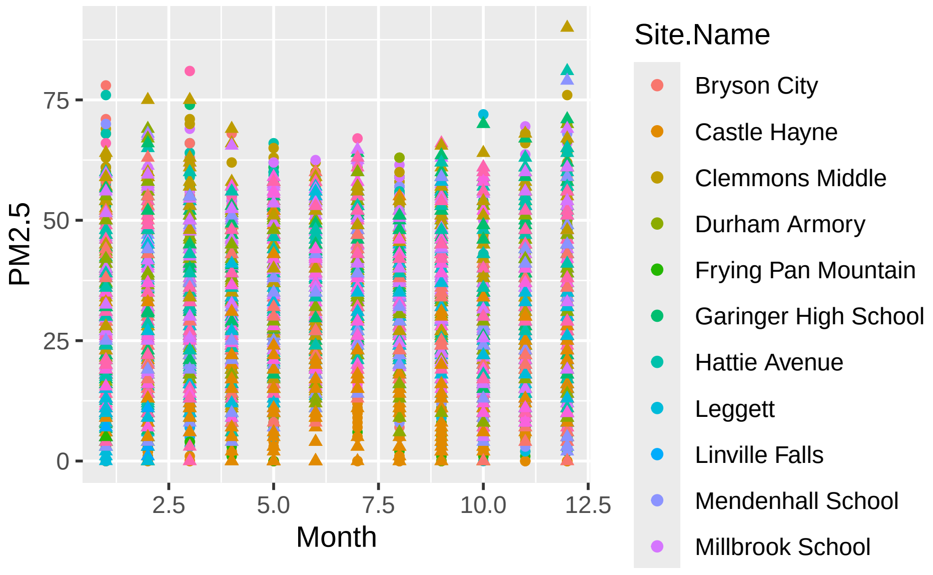


Figure 4: Scatter Plot

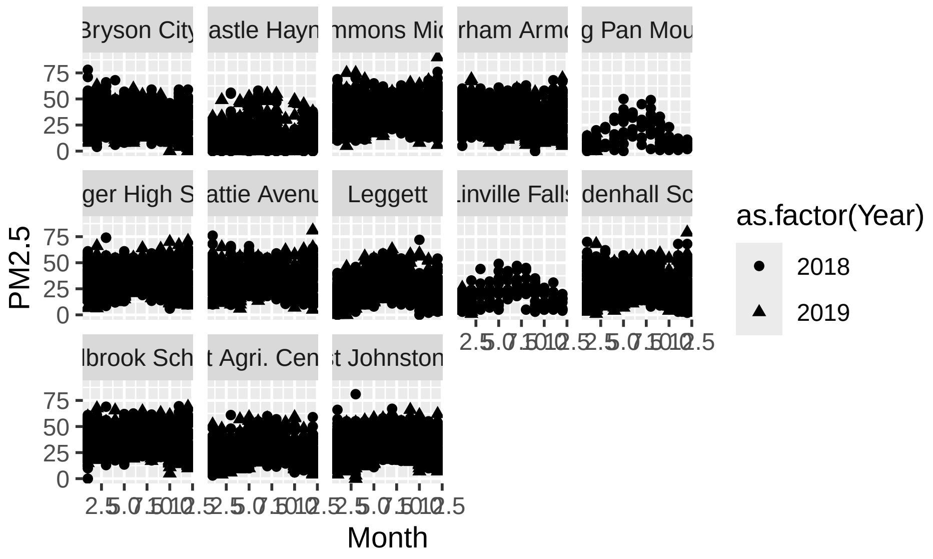


Figure 5: Scatter Plot

```r
# Filter dataset within plot building and facet by multiple variables. How to select only certain sites
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)
```

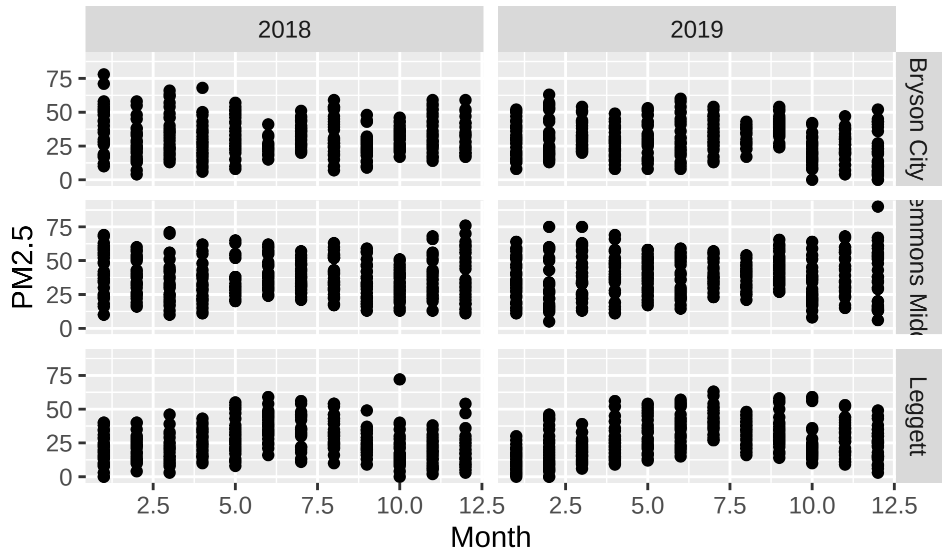


Figure 6: Scatter Plot

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

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 so outliers dont skew the graph and make it hard to read
lightvsDOfixed <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
  geom_point() +
```

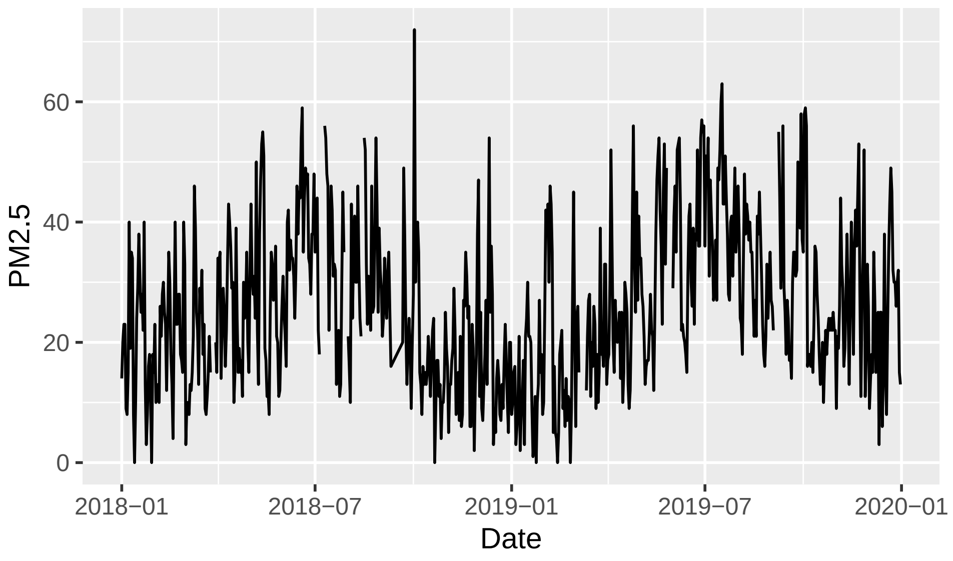


Figure 7: Scatter Plot

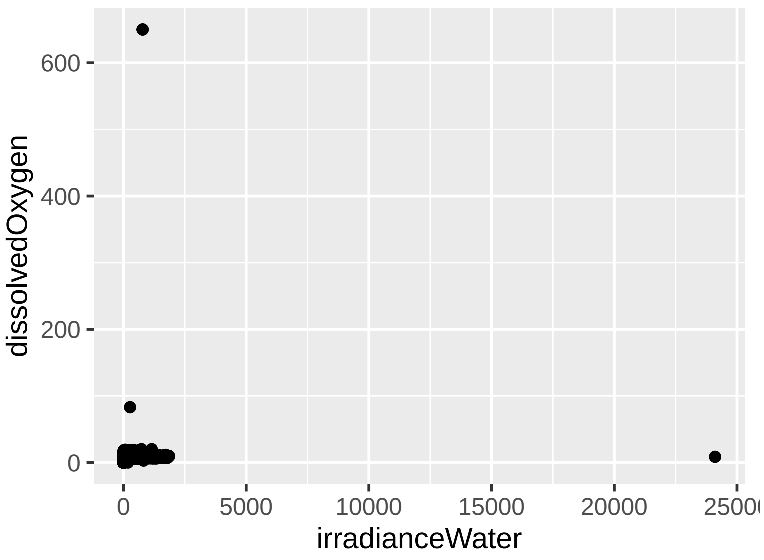


Figure 8: Another scatter plot

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

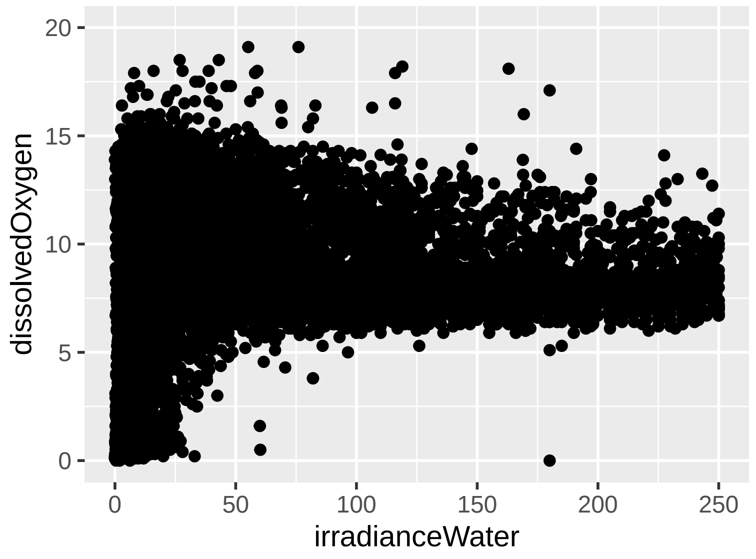


Figure 9: Another scatter plot

```
# 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() #if you do not specify then your data points will be plotted from 0 up, but we want
print(tempvsdepth)
```

```
#ombre among plot points from dark blue (Deepest) to light blue (shallow)
NvsP <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth)) +
  geom_point() +
  geom_smooth(method = lm) + #trend line
  geom_abline(aes(slope = 16, intercept = 0)) #can add a line with slope and intercept
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.

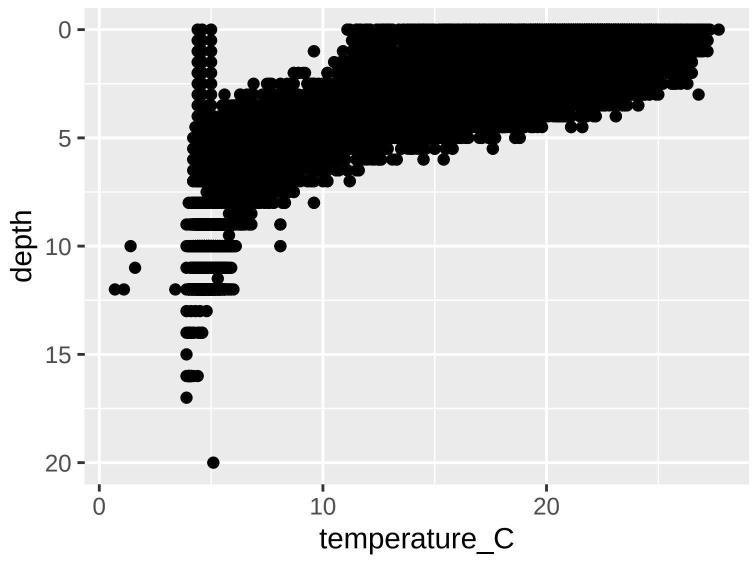


Figure 10: Another scatter plot

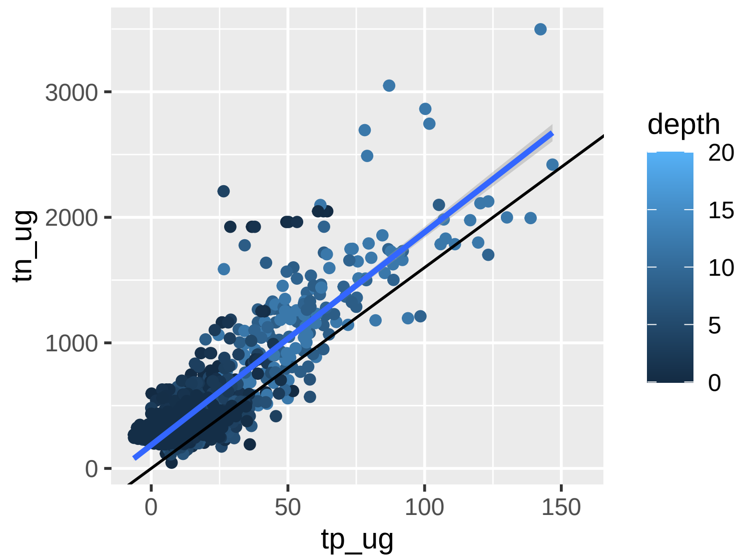
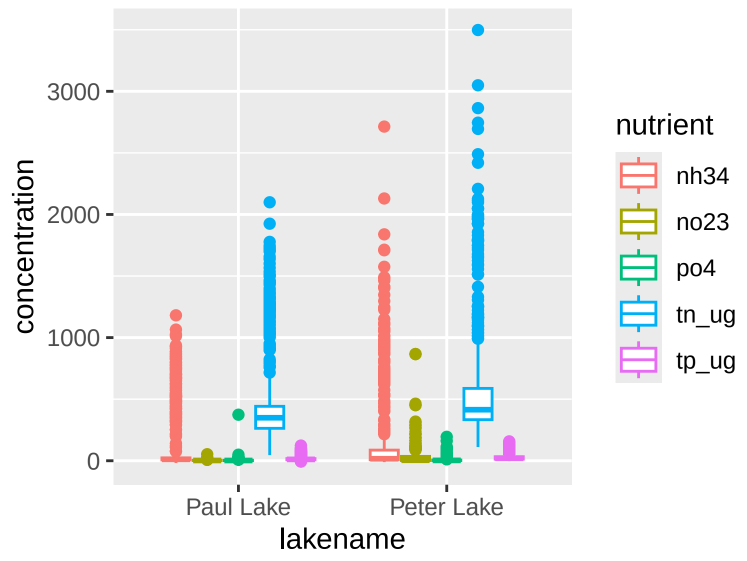


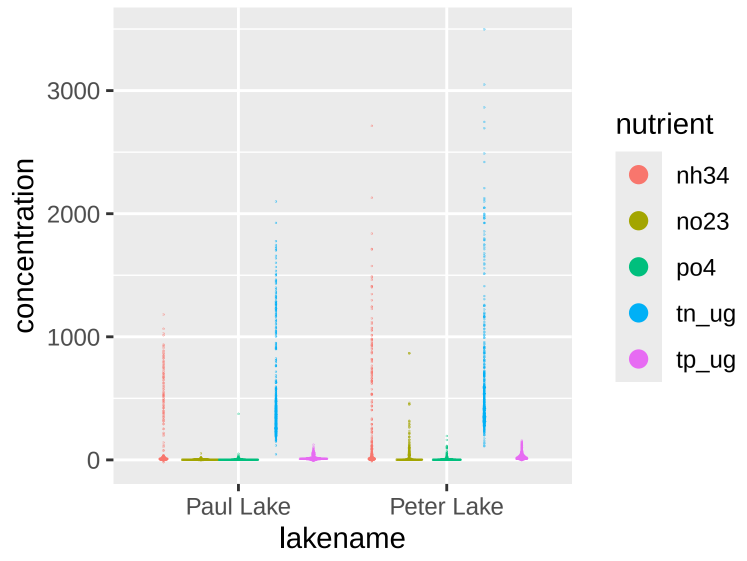
Figure 11: Another scatter plot

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

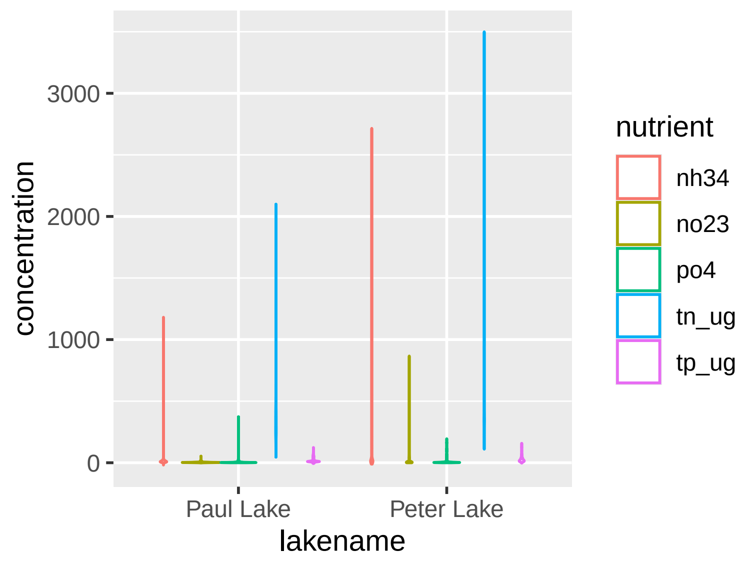
```r
# Box and whiskers plot
Nutrientplot3 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_boxplot(aes(color = nutrient)) # Why didn't we use "fill"?
print(Nutrientplot3)
```



```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) #dodge makes sure there are no overlying points, binaxis how the dots are stacking
```

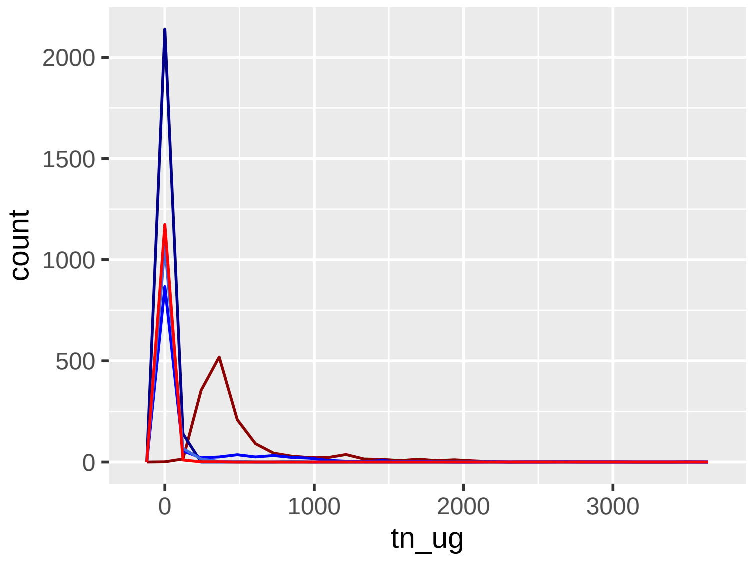


```r
# Violin plot
Nutrientplot5 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_violin(aes(color = nutrient)) #
print(Nutrientplot5)
```



```r
# Frequency polygons
# Using a tidy dataset. dont use, use gathered 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)
```

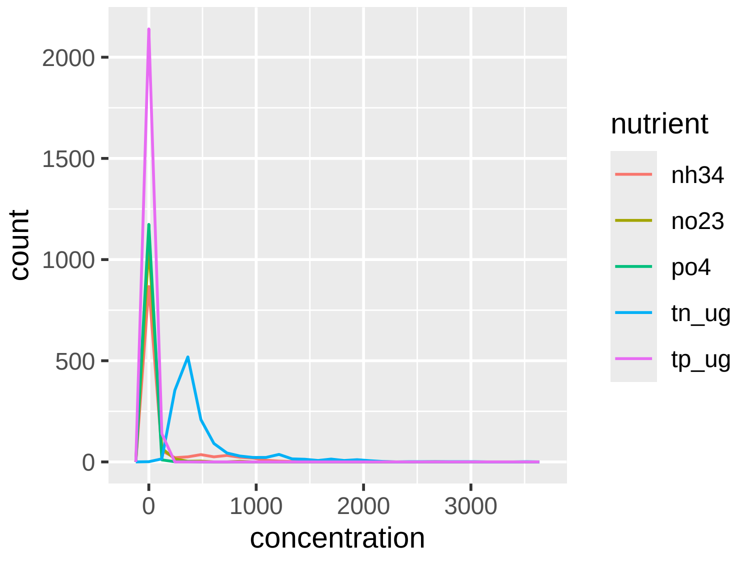
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using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



```
#no legend, dont know what they are, x axis isnt just tn_ug so its not accurate. Use gathered data set

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

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

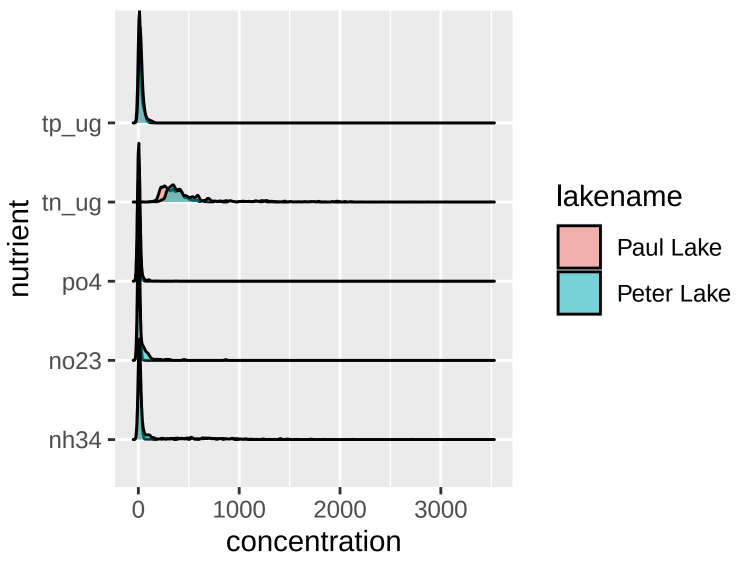


```
# Frequency polygons have the risk of becoming spaghetti plots.
# See <https://www.data-to-viz.com/caveat/spaghetti.html> for more info.

# Ridgeline plot
Nutrientplot6 <-
```

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

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


tp_ug
tn_ug
po4
no23
nh34
nutrient
concentration
lakename
Paul Lake
Peter Lake