

5: Part 1 - Data Visualization Basics

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

Fall 2024

List of Figures

1	Figure 1	4
2	Figure 1	4
3	Figure 1	5
4	Figure 1	5
5	Figure 1	6
6	Figure 1	7
7	Figure 1	7

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

```
getwd()
```

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

```
library(tidyverse);library(lubridate);library(here)
library(ggribges) # package for ridge plot: looks distribution of a variable over time
here()
```

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

```
PeterPaul.chem.nutrients <-
  read.csv(here("Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv"), stringsAsFactors = FALSE)
PeterPaul.chem.nutrients.gathered <-
  read.csv(here("Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Processed.csv"), stringsAsFactors = FALSE)
EPAair <- read.csv(here("Data/Processed/EPAair_03_PM25_NC1819_Processed.csv"), stringsAsFactors = TRUE)

EPAair$date <- ymd(EPAair$date)
PeterPaul.chem.nutrients$sampldate <- ymd(PeterPaul.chem.nutrients$sampldate)
PeterPaul.chem.nutrients.gathered$sampldate <- ymd(PeterPaul.chem.nutrients.gathered$sampldate)
```

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

```
# layout for plots generated in this chunk

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

```
# generate the same plot in a different way
O3plot <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
print(O3plot)
```

```
# Fix this code (change dots color to blue)
O3plot2 <- ggplot(EPAair) +
  # geom_point(aes(x = Date, y = Ozone, color = "blue")) # however the color is not in blue. when speci
  geom_point(aes(x = Date, y = Ozone), color = "blue") # correct
print(O3plot2)
```

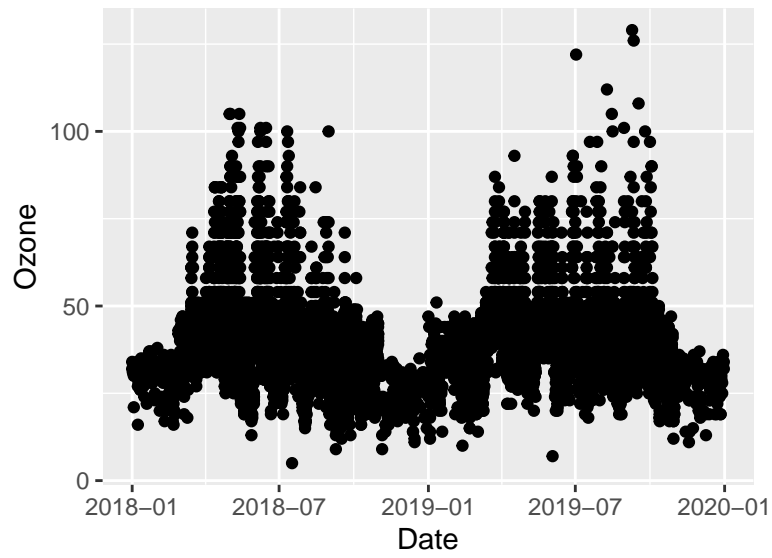


Figure 1: Figure 1

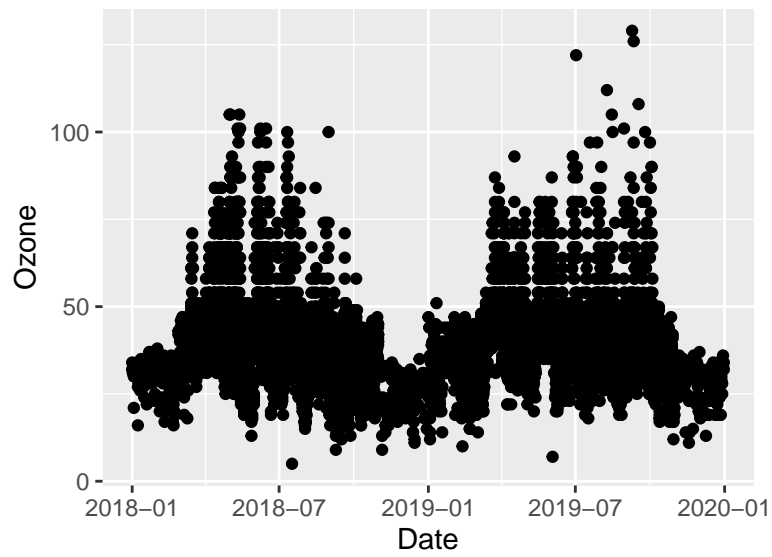


Figure 2: Figure 1

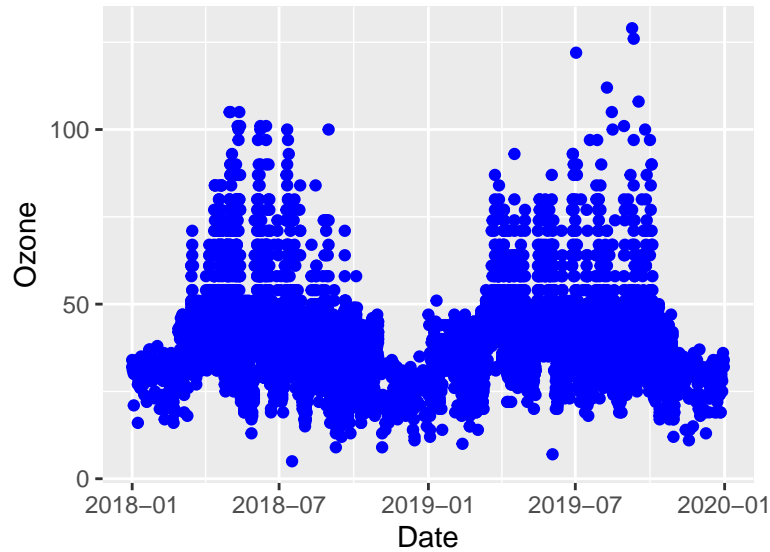


Figure 3: Figure 1

```
# 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()
print(PMplot)
```

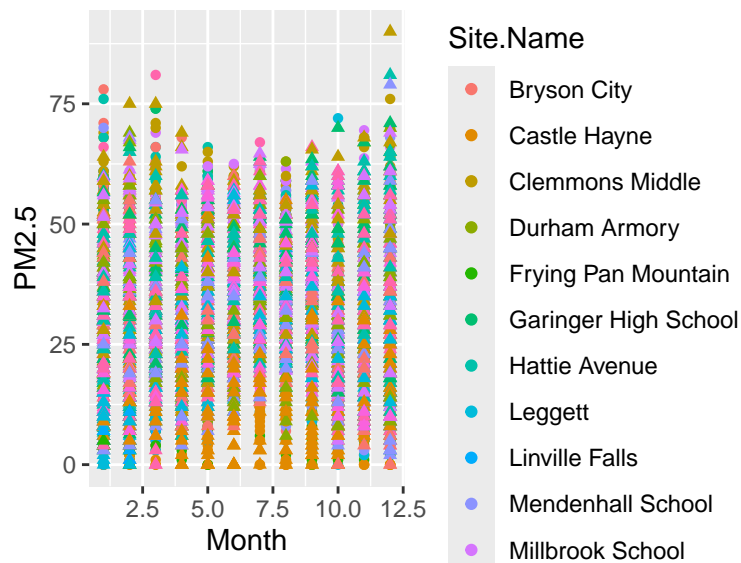


Figure 4: Figure 1

```
# 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)
```

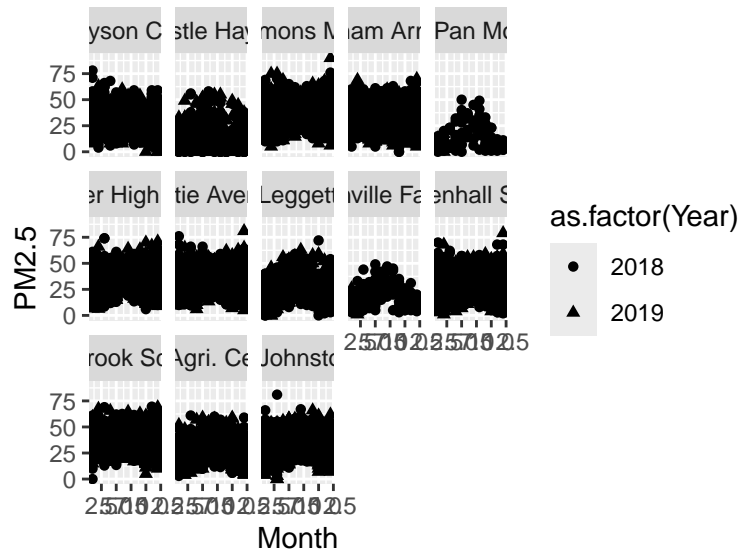


Figure 5: Figure 1

```
# 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) # Site.Name: row in plot, Year: column in plot
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)
```

Plotting the relationship between two continuous variables: Scatterplot

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

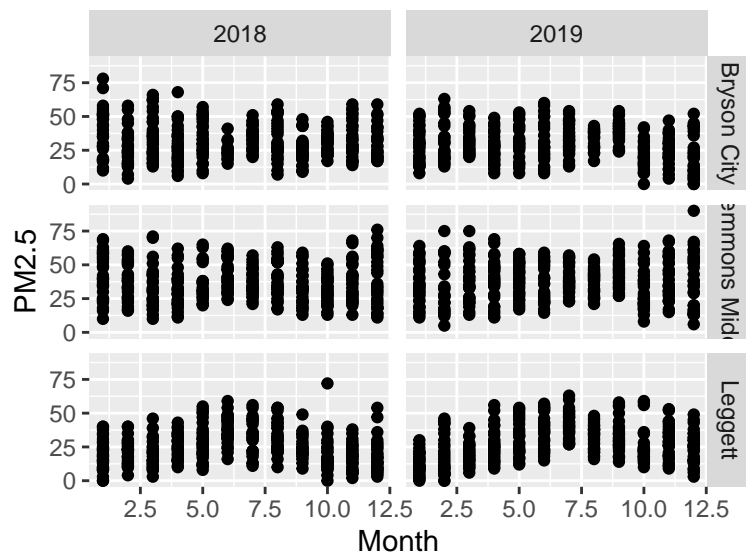


Figure 6: Figure 1

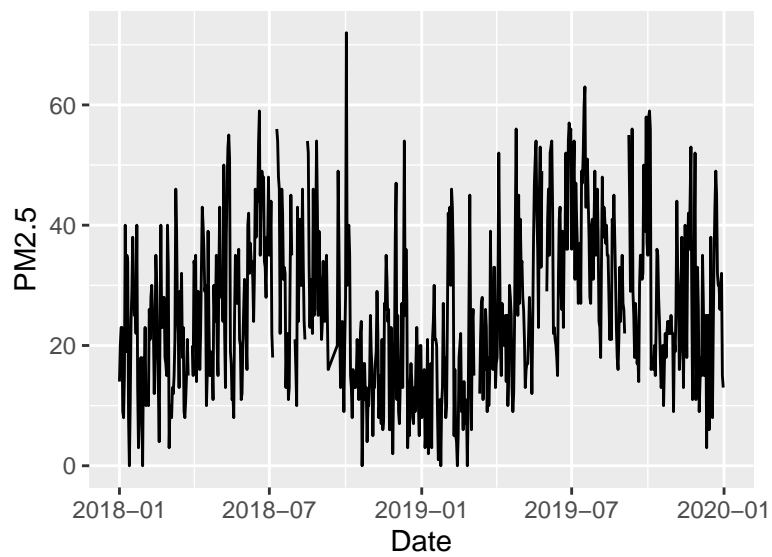
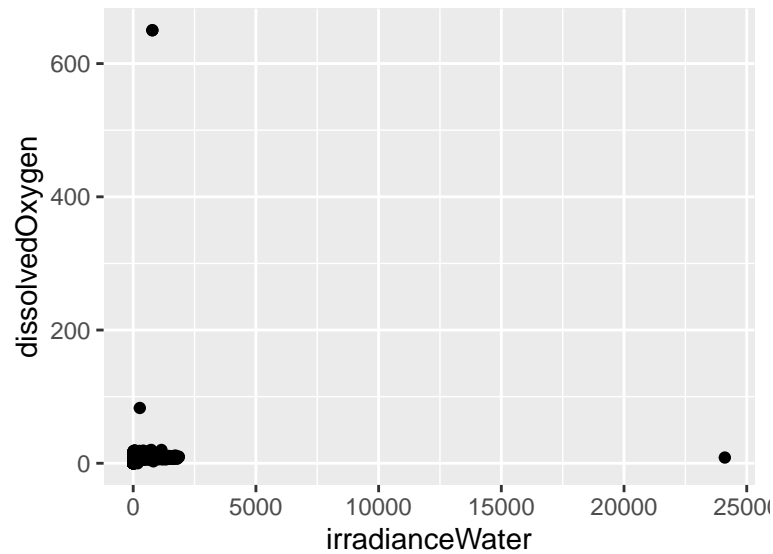
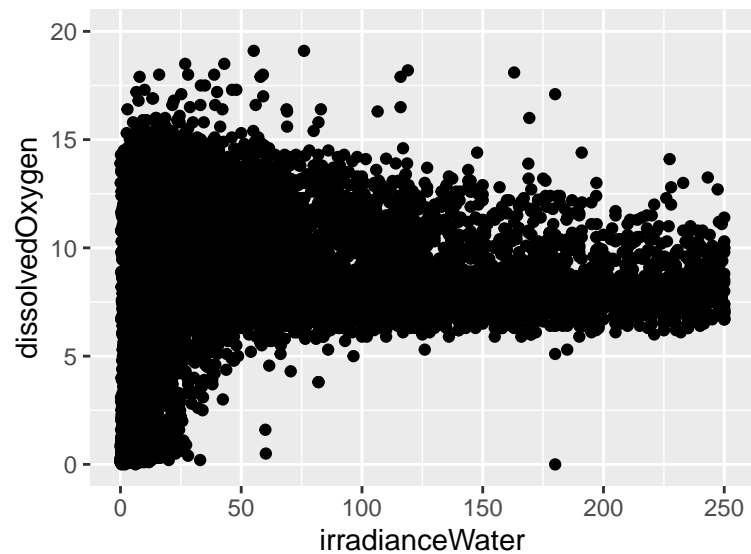


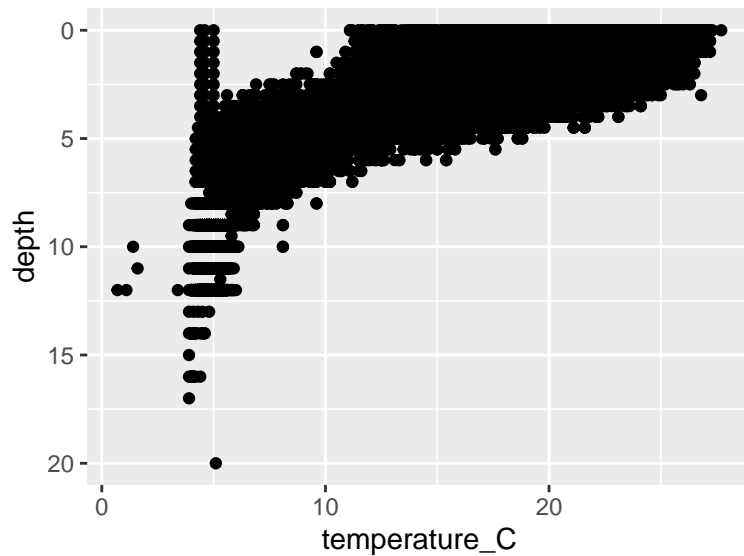
Figure 7: Figure 1



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

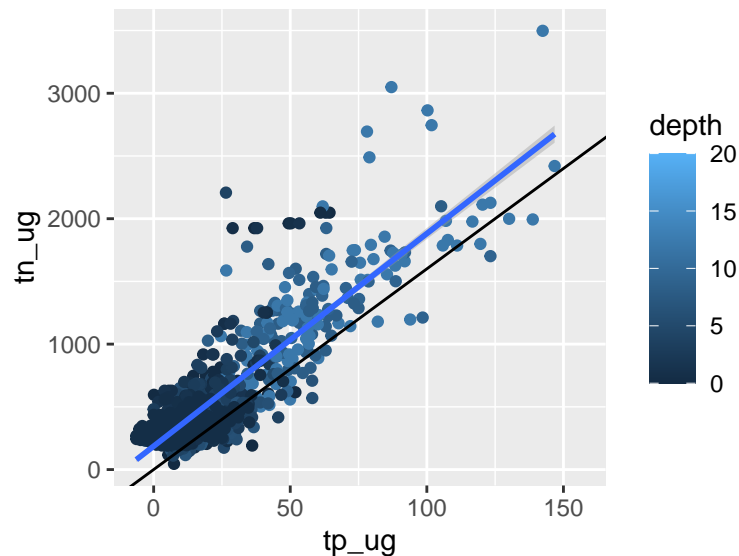


```
# Depth in the fields of limnology and oceanography is on a reverse scale
tempvsdepth <-
  ggplot(PeterPaul.chem.nutrients, aes(x = temperature_C, y = depth)) +
  #ggplot(PeterPaul.chem.nutrients, aes(x = temperature_C, y = depth, color = daynum)) +
  geom_point() +
  scale_y_reverse() # y scale reversed: 0-20
print(tempvsdepth)
```

```
# relationship of tp_ug and tn_ug
NvsP <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth)) +
  geom_point() +
  geom_smooth(method = lm) + # lm: linear model, calculate trend line
  geom_abline(aes(slope = 16, intercept = 0)) # already know the slope
print(NvsP)
```

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



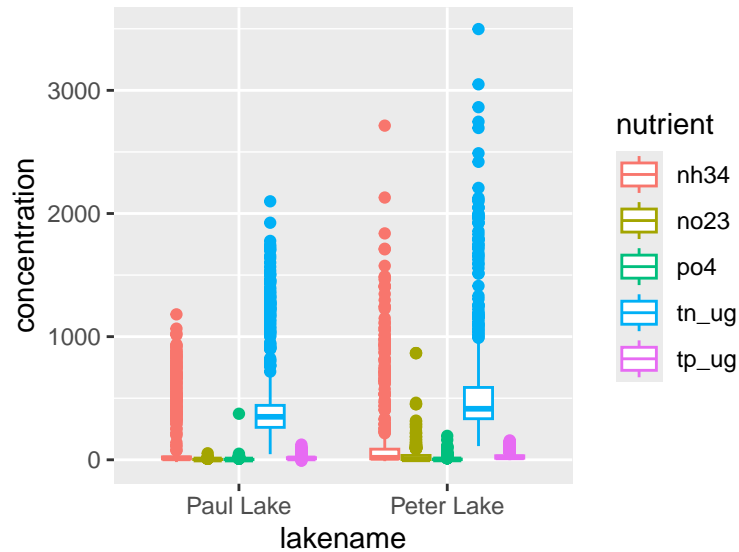
Plotting continuous vs. categorical variables

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

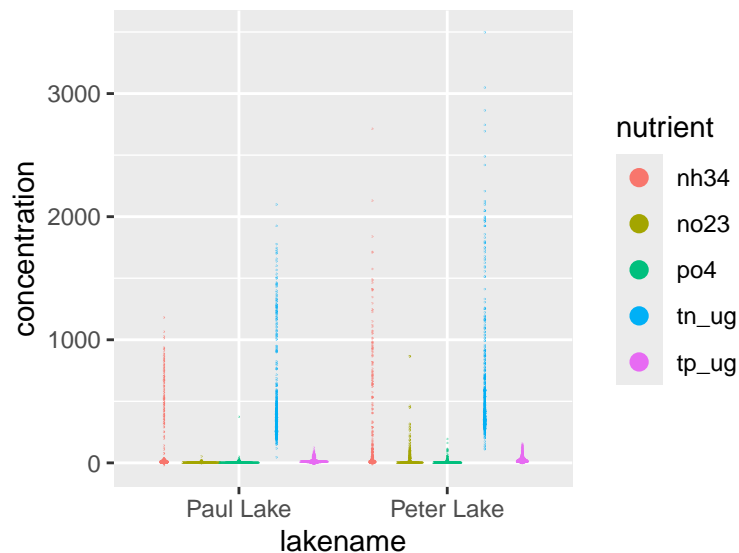
page on Data to Viz (<https://www.data-to-viz.com/caveats.html>) and find the page that explores barplots and error bars.

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

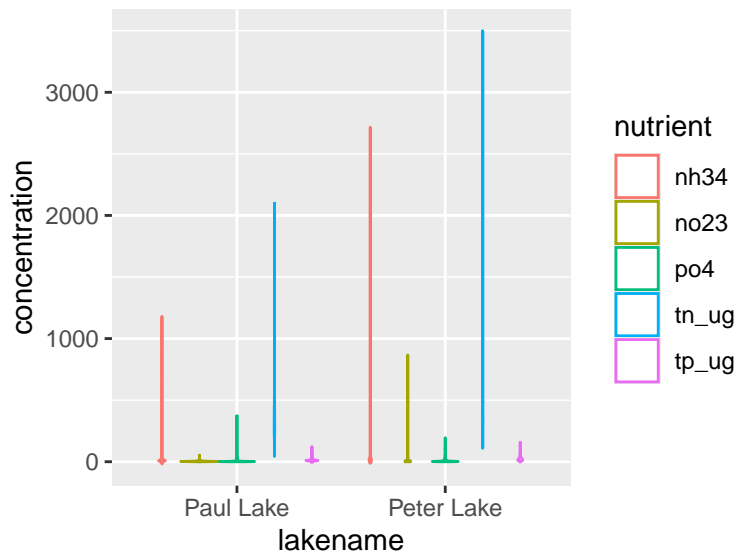
```
# Box and whiskers plot
Nutrientplot3 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_boxplot(aes(color = nutrient)) # Why didn't we use "fill"? color:contour; fill:inside of the box
print(Nutrientplot3)
```



```
# 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")
print(Nutrientplot4)
```

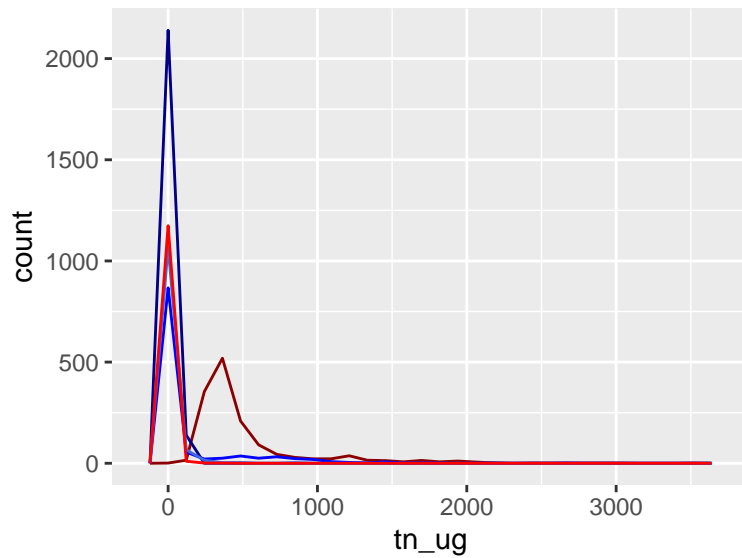


```
# Violin plot
Nutrientplot5 <-
  ggplot(PeterPaul.chem.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_violin(aes(color = nutrient)) # shows density plot
print(Nutrientplot5)
```



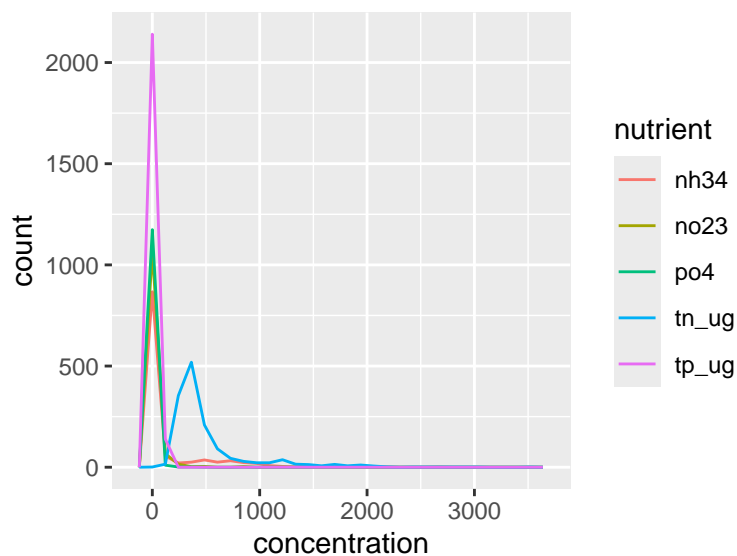
```
# Frequency polygons
# Using a tidy dataset
Nutrientplot6 <-
  ggplot(PeterPaul.chem.nutrients) +
  geom_freqpoly(aes(x = tn_ug), color = "darkred") +
  geom_freqpoly(aes(x = tp_ug), color = "darkblue") +
  geom_freqpoly(aes(x = nh34), color = "blue") +
  geom_freqpoly(aes(x = no23), color = "royalblue") +
  geom_freqpoly(aes(x = po4), color = "red") #specified variable on x and colors
print(Nutrientplot6) # no legend; not a good way to generate frequency plot! Just overlaying on one another
```

```
## '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, where we put all nutrients and concentrations together. Better approach
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) # categorized y by nutrient: 5 plots. One color
print(Nutrientplot6)
```

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

