9: Data Visualization

Environmental Data Analytics | Kateri Salk

Spring 2019

LESSON 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

SET UP YOUR DATA ANALYSIS SESSION

```
getwd()
```

```
## [1] "C:/Users/Felipe/OneDrive - Duke University/1. DUKE/1. Ramos 2 Semestre/EOS-872 Env. Data Analyti
```

```
library(tidyverse) #ggplot is included here

PeterPaul.chem.nutrients <- read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Proce
PeterPaul.nutrients.gathered <- read.csv("./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Pr
PeterPaul.chem.nutrients.summaries <- read.csv("./Data/Processed/NTL-LTER_Lake_Summaries_PeterPaul_Proce
EPAair <- read.csv("./Data/Processed/EPAair_O3_PM25_NC1718_Processed.csv")

EPAair$Date <- as.Date(EPAair$Date, format = "%Y-%m-%d") #we always need to do the date thing again
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 #sometimes you can use other dataframes in the geom

Here are some commonly used layers for plotting in ggplot:

- geom_bar
- geom_histogram
- geom_freqpoly
- geom_boxplot
- geom_violin
- geom_dotplot

- geom_point
- geom_errorbar
- geom_smooth
- geom_line
- geom_area
- geom_abline (plus geom_hline and geom_vline)
- geom_text

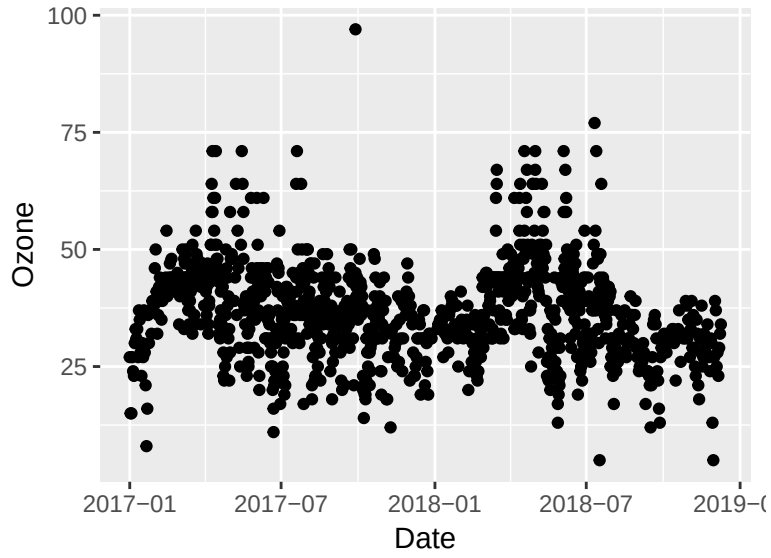
Aesthetics

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

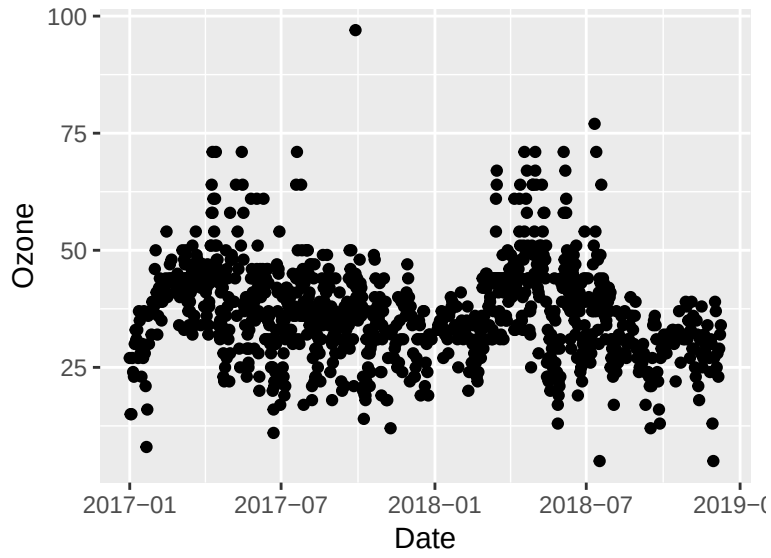
- color
- fill
- shape
- size
- transparency

Plotting continuous variables over time: Scatterplot

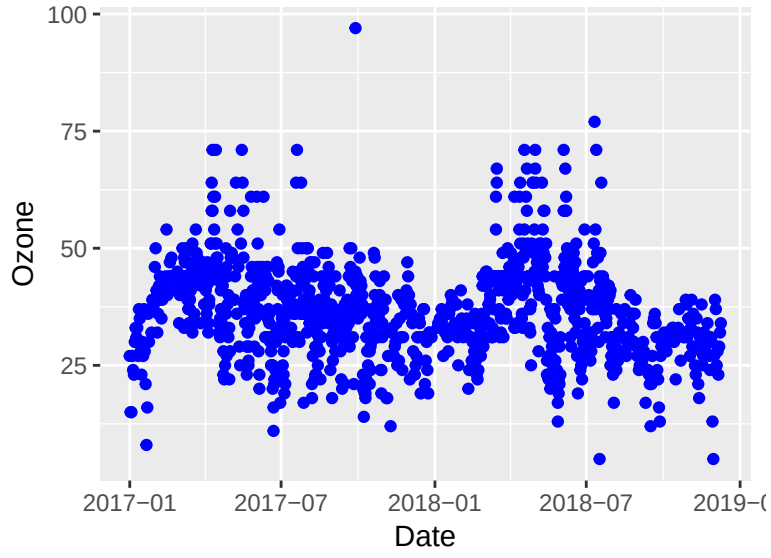
```
# Scatterplot
ggplot(EPAair, aes(x = Date, y = Ozone)) +
  geom_point()
```



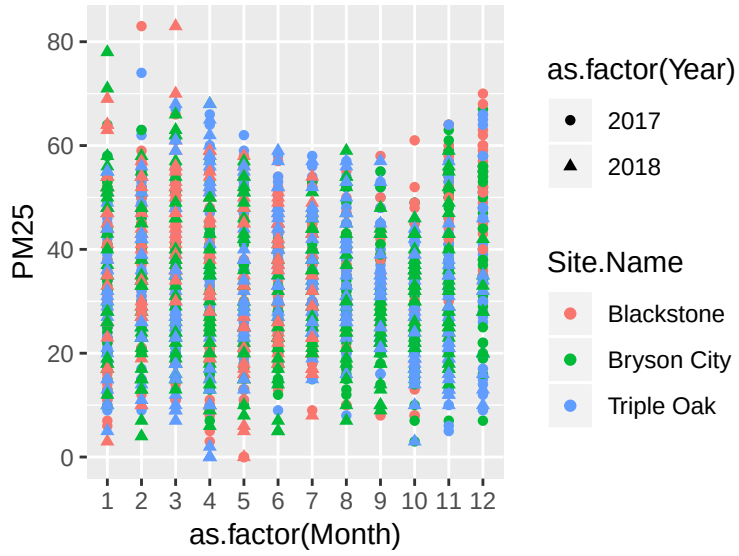

```
O3plot <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
print(O3plot) # you get the same result. Print() prints it in a pdf.
```



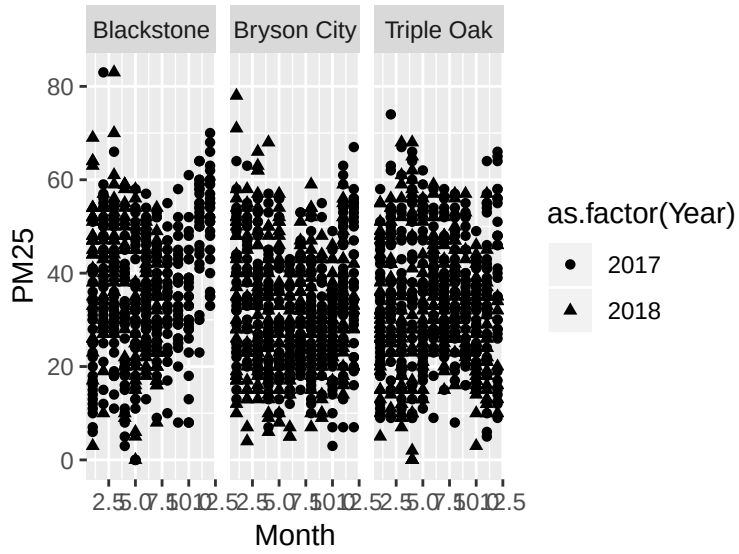

```
# Fix this code
O3plot2 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone), color = "blue") # fixed
print(O3plot2)
```



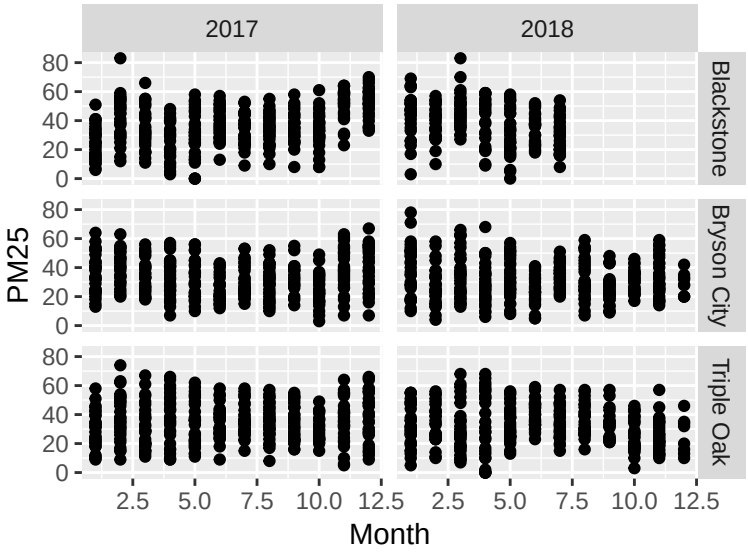

```
# Add additional variables
PMplot <-
  ggplot(EPAair, aes(x = as.factor(Month), y = PM25, shape = as.factor(Year), color = Site.Name)) +
  geom_point() # here makes sense color inside aes # month goes to 12.5. not pretty. as factor solves i
print(PMplot)
```



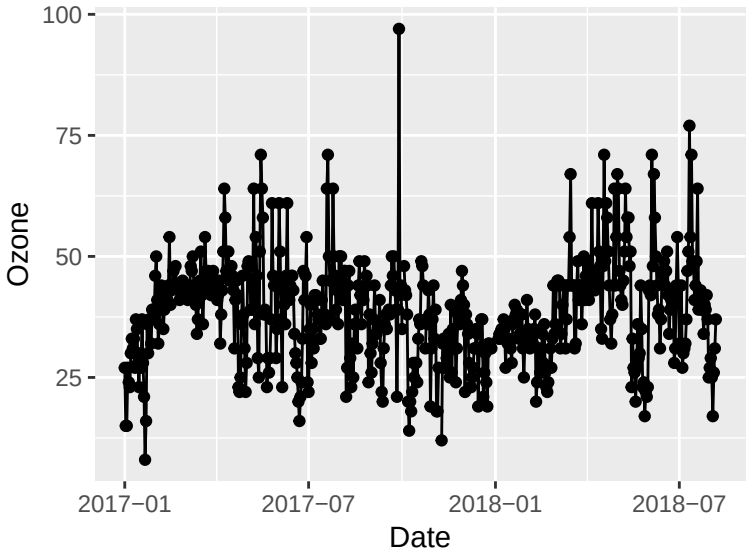

```
# Separate plot with facets
PMplot.faceted <-
  ggplot(EPAair, aes(x = Month, y = PM25, shape = as.factor(Year))) +
  geom_point() +
  facet_wrap(vars(Site.Name)) #new layer. three rows. three graphs. if not default is columns.
print(PMplot.faceted)
```




```
PMplot.faceted2 <-
  ggplot(EPAair, aes(x = Month, y = PM25)) +
  geom_point() +
  facet_grid(Site.Name ~ Year) # tilda is by
print(PMplot.faceted2)
```

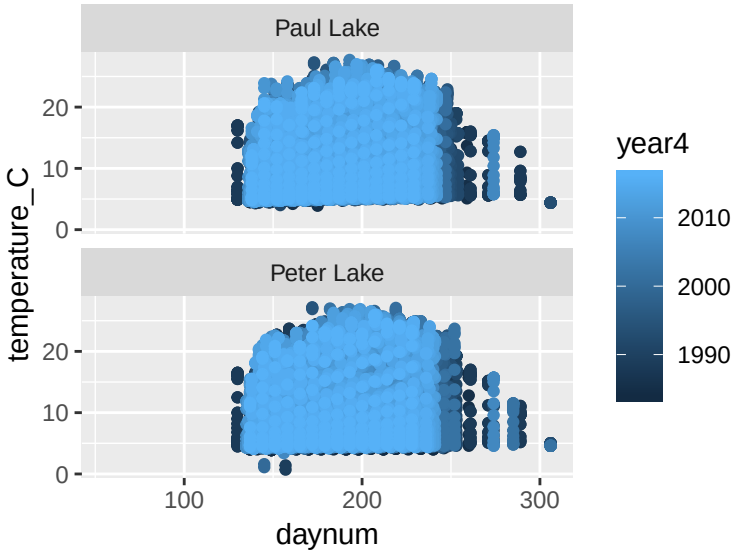



```
# Filter dataset within plot building
O3plot.Blackstone <-
  ggplot(subset(EPAair, Site.Name == "Blackstone"), aes(x = Date, y = Ozone)) +
  geom_point() +
  geom_line() # puts a line between the dots
print(O3plot.Blackstone) #subset of only Blackstone
```

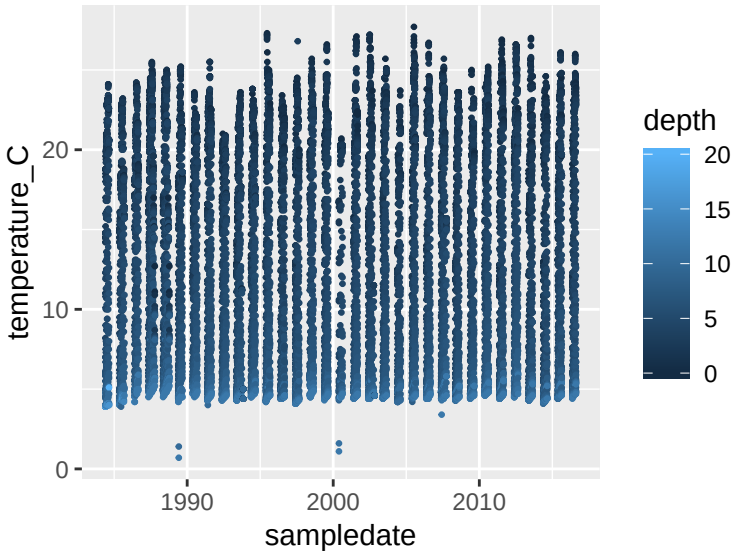



```
# Exercise: build your own scatterplots of PeterPaul.chem.nutrients

# 1.
# Plot surface temperatures by day of  year.
# Color your points by year, and facet by lake in two rows.

PeterPaul.chem.nutrients.faceted <-
  ggplot(PeterPaul.chem.nutrients, aes(x = daynum, y = temperature_C, color = year4)) +
  geom_point() +
  facet_wrap(vars(lakename), nrow = 2)
```

```
print(PeterPaul.chem.nutrients.faceted)
```

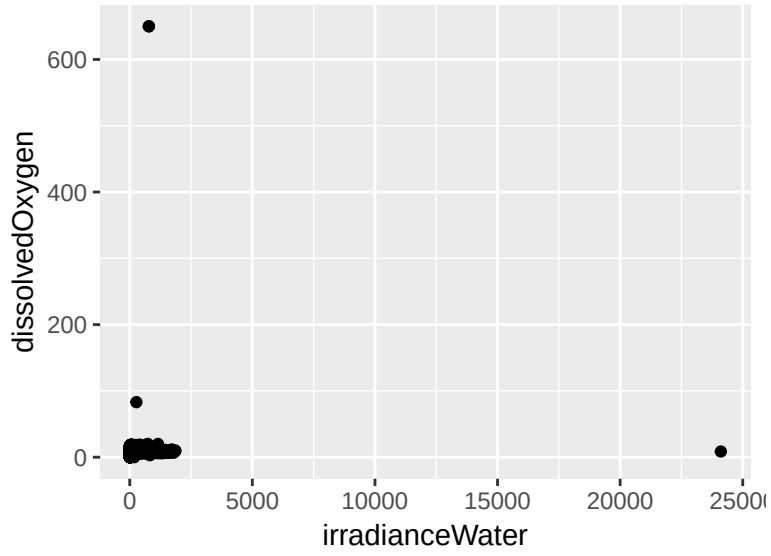



```
#2.
# Plot temperature by date. Color your points by depth.
# Change the size of your point to 0.5

PeterPaul.chem.nutrients.faceted2 <-
  ggplot(PeterPaul.chem.nutrients, aes(x = sampledate, y = temperature_C, color = depth)) +
  geom_point(size = 0.5)
  print(PeterPaul.chem.nutrients.faceted2)
```

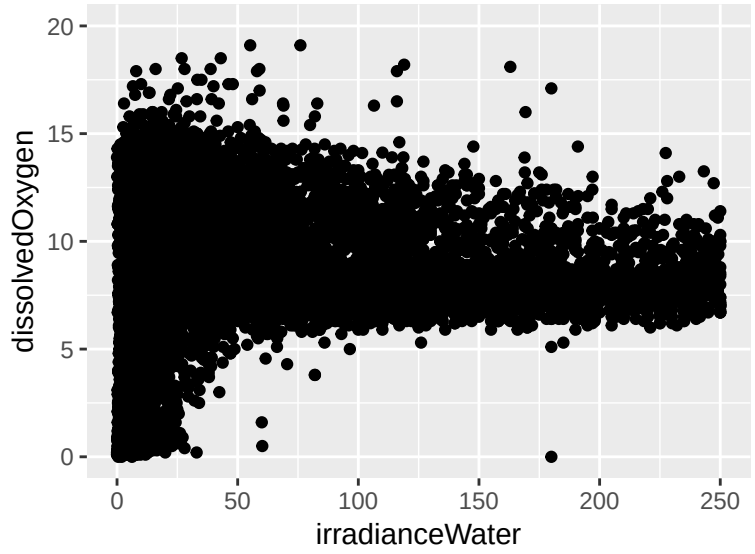


Plotting the relationship between two continuous variables: Scatterplot

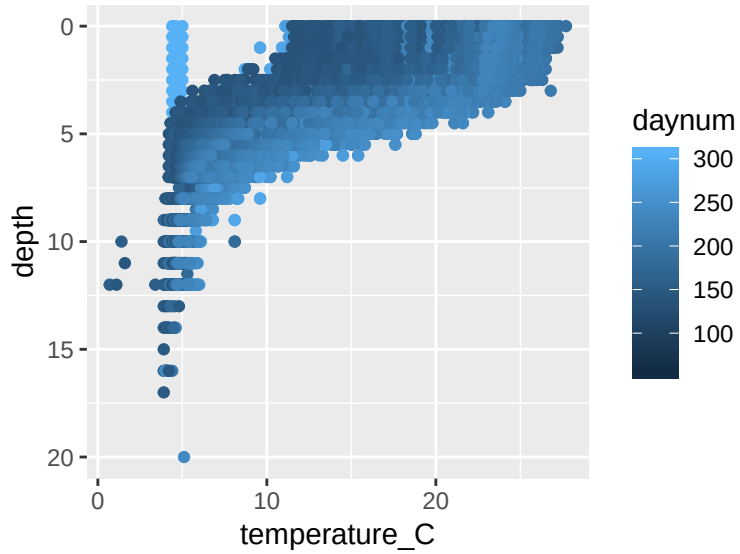
```
# Scatterplot
lightvsDO <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
  geom_point()
print(lightvsDO)
```



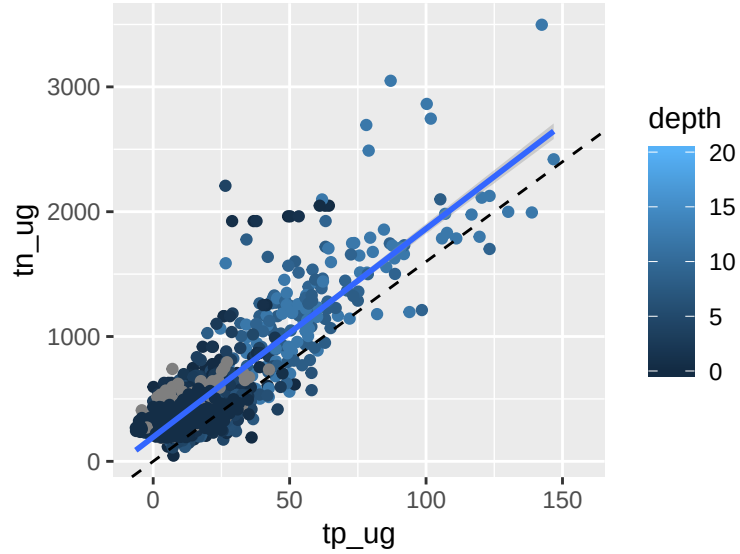

```
# Adjust axes
lightvsDOfixed <-
  ggplot(PeterPaul.chem.nutrients, aes(x = irradianceWater, y = dissolvedOxygen)) +
  geom_point() +
  xlim(0, 250) + # we set our axis to known data normal limits.
  ylim(0, 20)
print(lightvsDOfixed)
```




```
# 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()    #Reverse Axis
print(tempvsdepth)
```




```
#CLASS 2 ###############################################################################
NvsP <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth)) +
  geom_point() +
  geom_smooth(method = lm) + #without the inside uses smooth function
  geom_abline(aes(slope = 16, intercept = 0), lty = 2) # lty = linetype. The points are above the line.
print(NvsP)   # there are grey points. points with no value for the color.
```




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

# 1.
# Plot AQI values for ozone by PM2.5, colored by site.

O3vsPM25 <-
  ggplot(EPAair, aes(x = PM25, y = Ozone, color = Site.Name)) +
  geom_point() #+
  geom_smooth(method = lm) #+
```

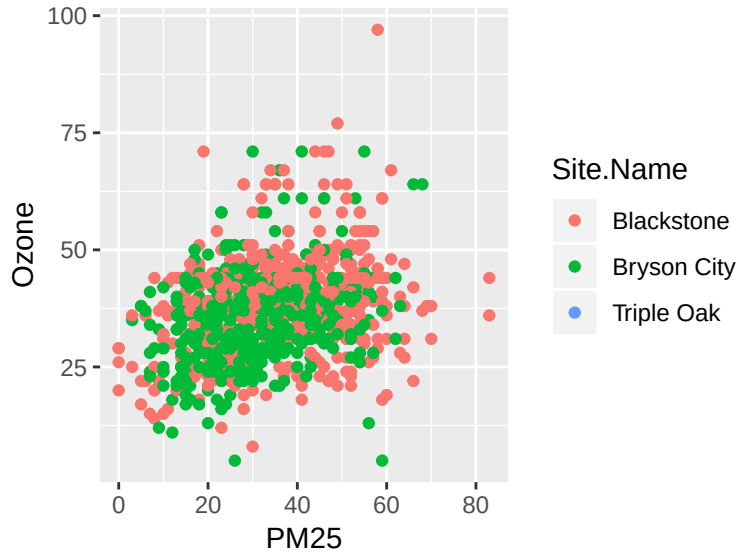
```
## geom_smooth: na.rm = FALSE, se = TRUE
## stat_smooth: na.rm = FALSE, se = TRUE, method = function (formula, data, subset, weights, na.action,
## {
##     ret.x <- x
##     ret.y <- y
##     cl <- match.call()
##     mf <- match.call(expand.dots = FALSE)
##     m <- match(c("formula", "data", "subset", "weights", "na.action", "offset"), names(mf), 0)
##     mf <- mf[c(1, m)]
##     mf$drop.unused.levels <- TRUE
##     mf[[1]] <- quote(stats::model.frame)
##     mf <- eval(mf, parent.frame())
##     if (method == "model.frame")
##         return(mf)
##     else if (method != "qr")
##         warning(gettextf("method = '%s' is not supported. Using 'qr'", method), domain = NA)
##     mt <- attr(mf, "terms")
##     y <- model.response(mf, "numeric")
##     w <- as.vector(model.weights(mf))
##     if (!is.null(w) && !is.numeric(w))
##         stop("'weights' must be a numeric vector")
##     offset <- as.vector(model.offset(mf))
##     if (!is.null(offset)) {
##         if (length(offset) != NROW(y))
##             stop(gettextf("number of offsets is %d, should equal %d (number of observations)", lengt
##     }
##     if (is.empty.model(mt)) {
##         x <- NULL
##         z <- list(coefficients = if (is.matrix(y)) matrix(NA, 0, ncol(y)) else numeric(), residuals =
##         if (!is.null(offset)) {
##             z$fitted.values <- offset
##             z$residuals <- y - offset
##         }
##     }
##     else {
##         x <- model.matrix(mt, mf, contrasts)
##         z <- if (is.null(w))
##             lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...)
##         else lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok, ...)
##     }
##     class(z) <- c(if (is.matrix(y)) "mlm", "lm")
##     z$na.action <- attr(mf, "na.action")
##     z$offset <- offset
##     z$contrasts <- attr(x, "contrasts")
##     z$xlevels <- .getXlevels(mt, mf)
##     z$call <- cl
##     z$terms <- mt
##     if (model)
##         z$model <- mf
##     if (ret.x)
##         z$x <- x
##     if (ret.y)
##         z$y <- y
##     if (!qr)
```

```
##         z$qr <- NULL
##     z
## }, formula = y ~ x
## position_identity
```

```
  geom_abline(aes(slope = 16, intercept = 0), lty = 2)
```

```
## mapping: slope = 16, intercept = 0
## geom_abline: na.rm = FALSE
## stat_identity: na.rm = FALSE
## position_identity
```

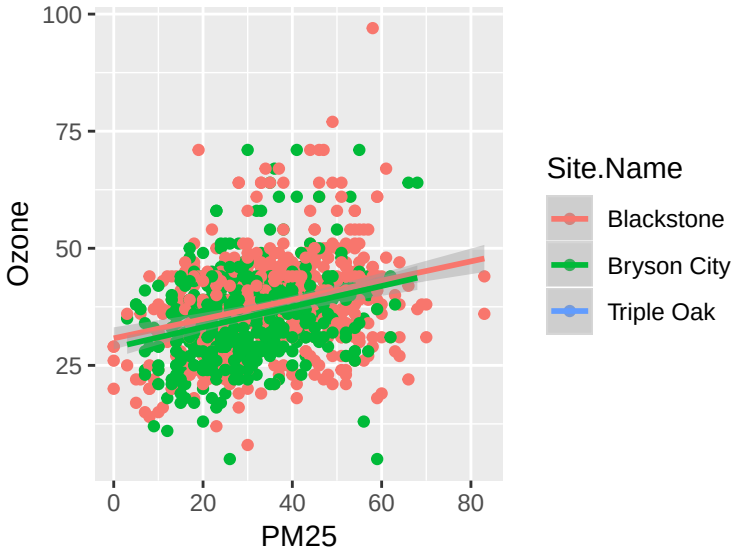
```
print(O3vsPM25)
```




```
# Add a line of best fit for the linear regression of these variables.
O3vsPM25 <-
  ggplot(EPAair, aes(x = PM25, y = Ozone, color = Site.Name)) +
  geom_point() + # if you put here as(color=Site.Name) it gives one line for everything.
  geom_smooth(method = lm) #+
  geom_abline(aes(slope = 16, intercept = 0), lty = 2)
```

```
## mapping: slope = 16, intercept = 0
## geom_abline: na.rm = FALSE
## stat_identity: na.rm = FALSE
## position_identity
```

```
print(O3vsPM25)
```

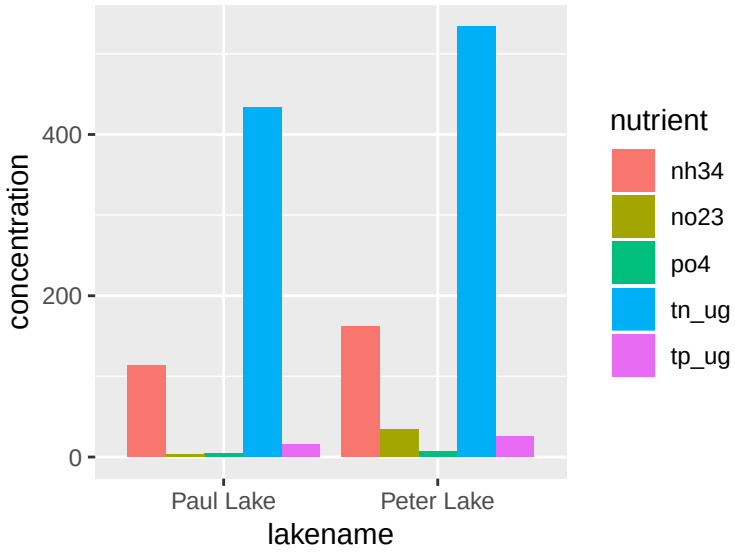



```
class(EPAair$Site.Name)
```

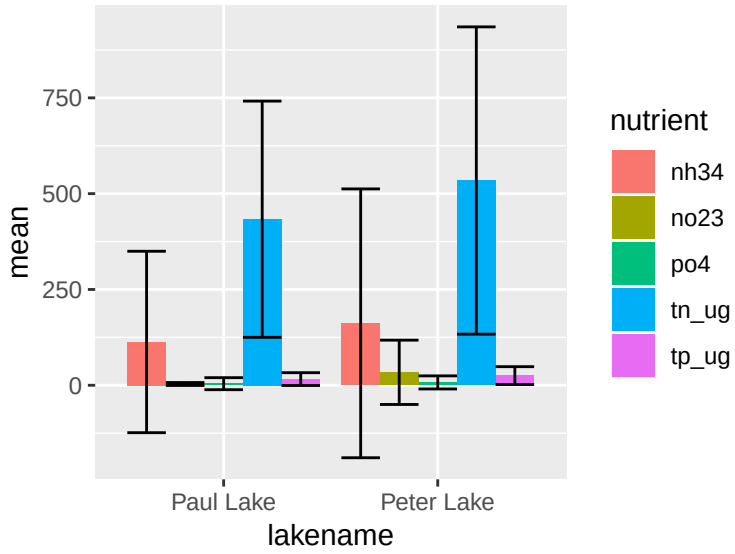
```
## [1] "factor"
```

Plotting continuous vs. categorical variables

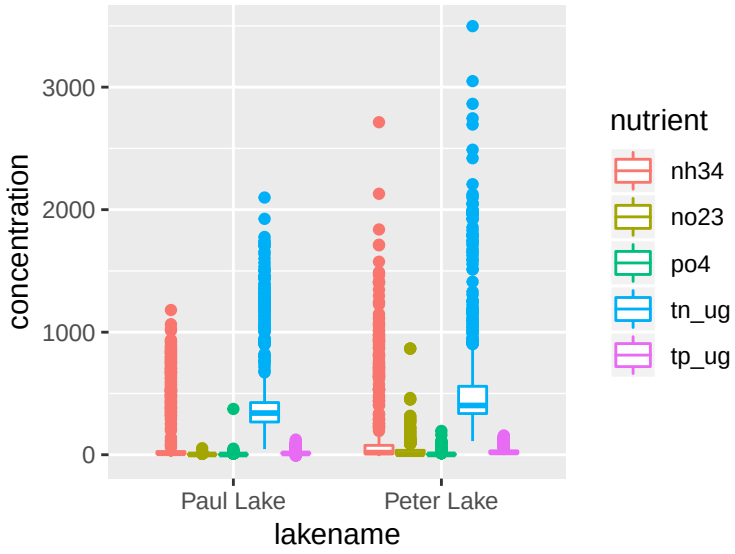
```
 # Barplot + error bars
PeterPaul.nutrient.summaries <- PeterPaul.nutrients.gathered %>%
  group_by(lakename, nutrient) %>%   # we are grouping by two things
  summarise(sd = sd(concentration),
            mean = mean(concentration))

Nutrientplot <-
  ggplot(PeterPaul.nutrients.gathered) +
  geom_bar(aes(x = lakename, y = concentration, fill = nutrient), # why did we use fill? Instead of col
           position = "dodge", stat = "summary", fun.y = "mean")  # what's happening here? Dodge = stuc
print(Nutrientplot)
```



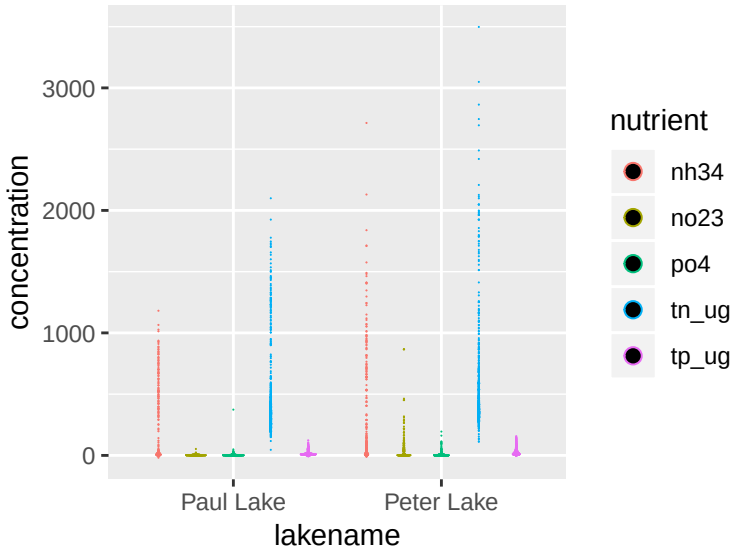

```
Nutrientplot2 <-
  ggplot(PeterPaul.nutrient.summaries, aes(x = lakename, y = mean, fill = nutrient)) + #
  geom_bar(stat = "identity", position = "dodge") + # what does the stat command do? The indentity of t
  geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd), # how do we specify error bars?
                position = "dodge") #you need dodge
print(Nutrientplot2) # not useful. Error goes below to zero. Not real value.
```



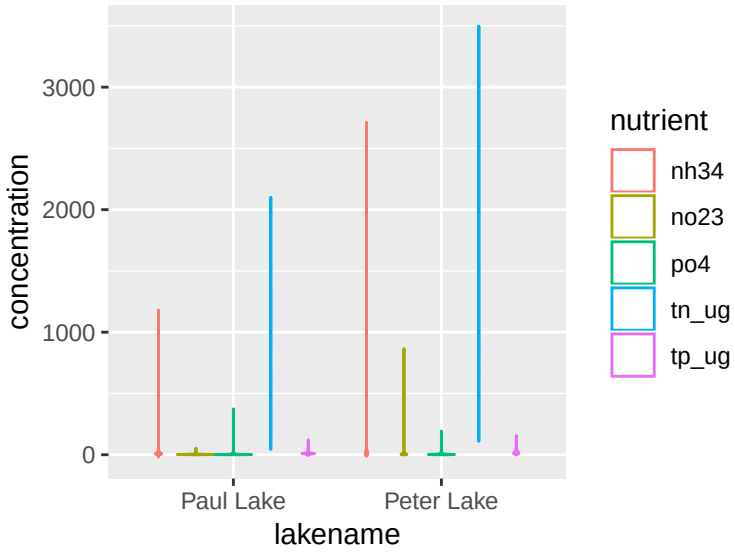

```
# Are there more effective ways to produce summary stats for categories? Alternatives-->

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




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

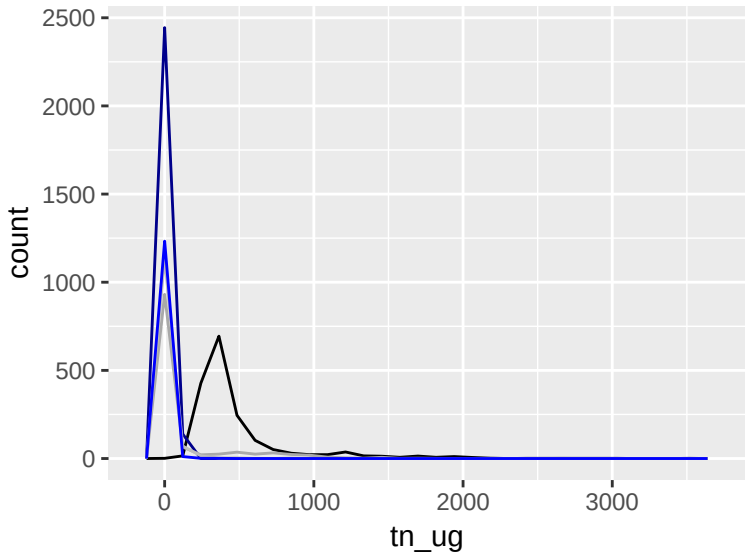



```
# Violin plot
Nutrientplot5 <-
  ggplot(PeterPaul.nutrients.gathered, aes(x = lakename, y = concentration)) +
  geom_violin(aes(color = nutrient)) #
print(Nutrientplot5)
```



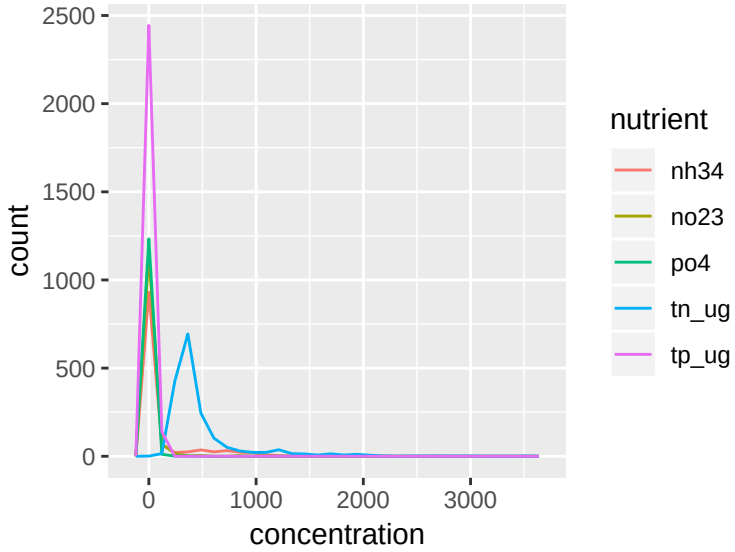

```
# Frequency polygons
# Using a tidy dataset
Nutrientplot6 <-
  ggplot(PeterPaul.chem.nutrients) +
  geom_freqpoly(aes(x = tn_ug), color = "black") +   # if you dont put colors, all black
  geom_freqpoly(aes(x = tp_ug), color = "darkblue") +
  geom_freqpoly(aes(x = nh34), color = "darkgray") +
  geom_freqpoly(aes(x = no23), color = "gray") +
  geom_freqpoly(aes(x = po4), color = "blue")
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`.
```

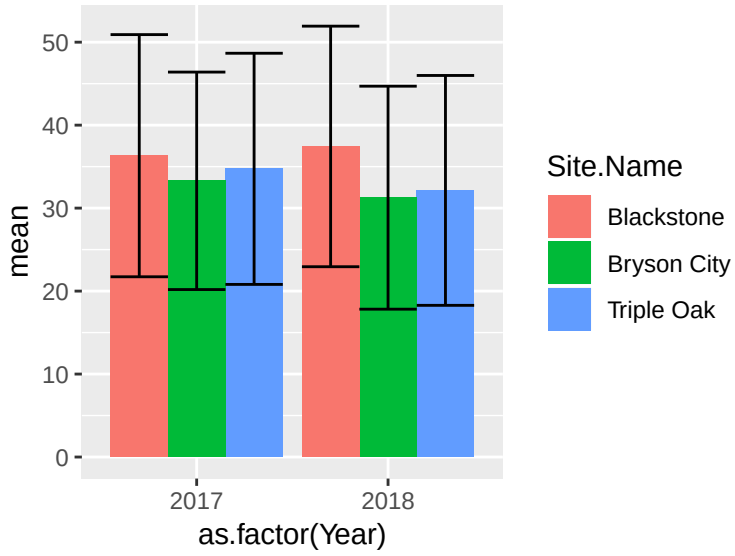


```
# Using a gathered dataset    #gathered versus tidy dataset
Nutrientplot7 <-
  ggplot(PeterPaul.nutrients.gathered) +
  geom_freqpoly(aes(x = concentration, color = nutrient))
print(Nutrientplot7)
```

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




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

# 1.
# Create a bar chart plus standard deviation error bars for PM2.5, divided by year.
# Create separate bars for each site.

EPAairsummary <- EPAair %>%
  group_by(Site.Name, Year) %>%   #key for the summary
  summarise(mean=mean(PM25,na.rm = TRUE),
            sd=sd(PM25,na.rm = TRUE))

AQIValuesEPAairbarplot <-
  ggplot(EPAairsummary, aes(x = as.factor(Year), y = mean, fill = Site.Name)) + #
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd),
                position = "dodge")
print(AQIValuesEPAairbarplot)
```




```
# 2.
# Create a new plot that better depicts the distribution of PM2.5 concentrations.
# Divide your graph by year and site.
```