Assignment 5: Data Visualization

Melissa Merritt

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on Data Visualization

Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.

The completed exercise is due on Friday, Oct 14th @ 5:00pm.

Set up your session

1. Set up your session. Verify your working directory and load the tidyverse, lubridate, & cowplot packages. Upload the NTL-LTER processed data files for
nutrients and chemistry/physics for Peter and Paul Lakes (use the tidy [`NTL-LTER_Lake_Chemistry_Nutrients_PeterPau`
version) and the processed data file for the Niwot Ridge litter dataset (use the [`NEON_NIWO_Litter_mass_trap_Processed`
version).

2. Make sure R is reading dates as date format; if not change the format to date.

```
# 1
getwd()
```

```
## [1] "/home/guest/Documents/EDA-Fall2022"
```

```
# I used getwd() to check the working directory.

install.packages("tidyverse")
```

```
## Installing package into '/home/guest/R/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("lubridate")
```

```
## Installing package into '/home/guest/R/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("cowplot")
```

```
## Installing package into '/home/guest/R/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(cowplot)
```

```
##
## Attaching package: 'cowplot'
##
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
# Using the install.packages() and library() to get the packages needed to run
# the specific functions.

Chemistry_Nutrients_PeterPaul <- read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_
    stringsAsFactors = TRUE)

NiwotRidge_Litter <- read.csv("./Data/Processed/NEON_NIWO_Litter_mass_trap_Processed.csv",
    stringsAsFactors = TRUE)

# I used the read.csv() function to import the data files and I renamed them to
# something shorter and easier to work with.

# 2

class(Chemistry_Nutrients_PeterPaul$sampledate)
```

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

```
class(NiwotRidge_Litter$collectDate)
```

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

```
# Initially when I check the class of the date columns in each of the data
# frames they were factors, so I used the as.Date() function below to change
# the class to date.

Chemistry_Nutrients_PeterPaul$sampledate <- as.Date(Chemistry_Nutrients_PeterPaul$sampledate,
    format = "%Y-%m-%d")
NiwotRidge_Litter$collectDate <- as.Date(NiwotRidge_Litter$collectDate, format = "%Y-%m-%d")
```

```
class(Chemistry_Nutrients_PeterPaul$sampledate)
```

```
## [1] "Date"
```

```
class(NiwotRidge_Litter$collectDate)
```

```
## [1] "Date"
```

```
# I was able to recheck the class with the class() function to ensure that the
# date columns were now being read as date formats.
```

Define your theme

3. Build a theme and set it as your default theme.

```
# 3
mytheme <- theme_bw(base_size = 12) + theme(axis.text = element_text(color = "black"),
    legend.position = "top")

# I built my theme using the black and white theme. Base_size represents the
# base font, so I set that at 12. The theme() function allows me to change the
# color scheme of the axis text, and keep the legend on top of the plots.
```

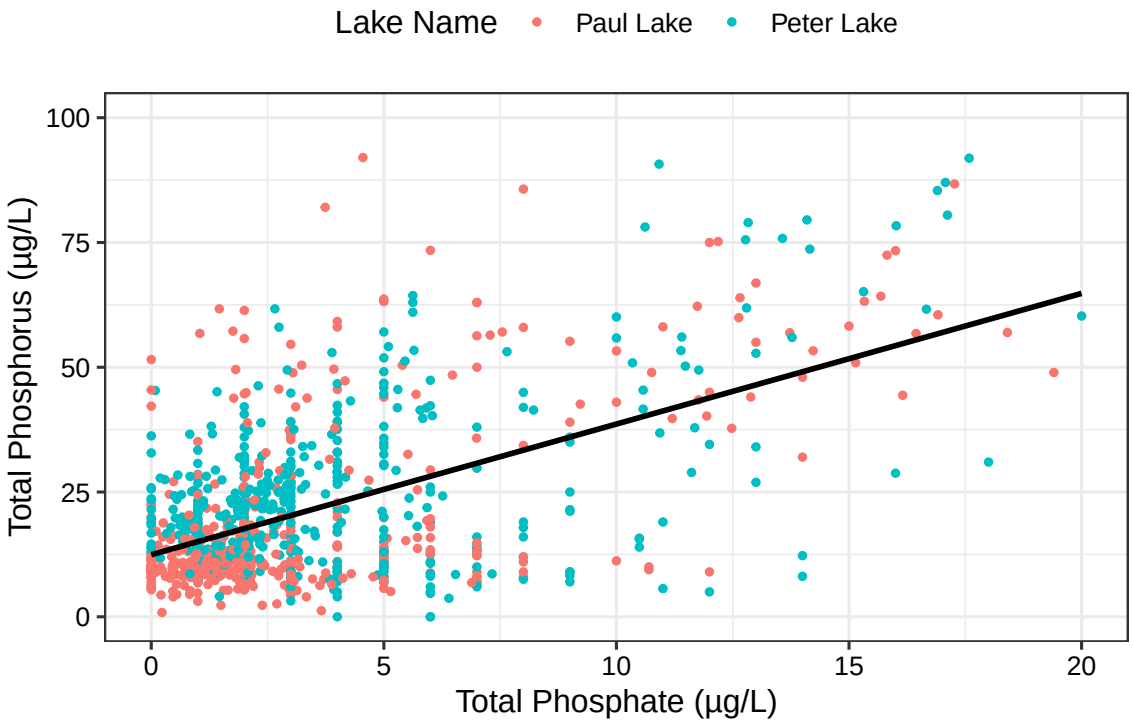
Create graphs

For numbers 4-7, create ggplot graphs and adjust aesthetics to follow best practices for data visualization. Ensure your theme, color palettes, axes, and additional aesthetics are edited accordingly.

4. [NTL-LTER] Plot total phosphorus (`tp_ug`) by phosphate (`po4`), with separate aesthetics for Peter and Paul lakes. Add a line of best fit and color it black. Adjust your axes to hide extreme values (hint: change the limits using `xlim()` and/or `ylim()`).

```
# 4
Phosphorus_Phosphate <- ggplot(Chemistry_Nutrients_PeterPaul, aes(x = po4, y = tp_ug,
    color = lakename)) + geom_point(size = 1) + xlim(c = 0, 20) + ylim(c = 0, 100) +
    xlab("Total Phosphate (µg/L)") + ylab("Total Phosphorus (µg/L)") + geom_smooth(method = lm,
    color = "black", se = F) + labs(color = "Lake Name") + mytheme
print(Phosphorus_Phosphate)
```

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



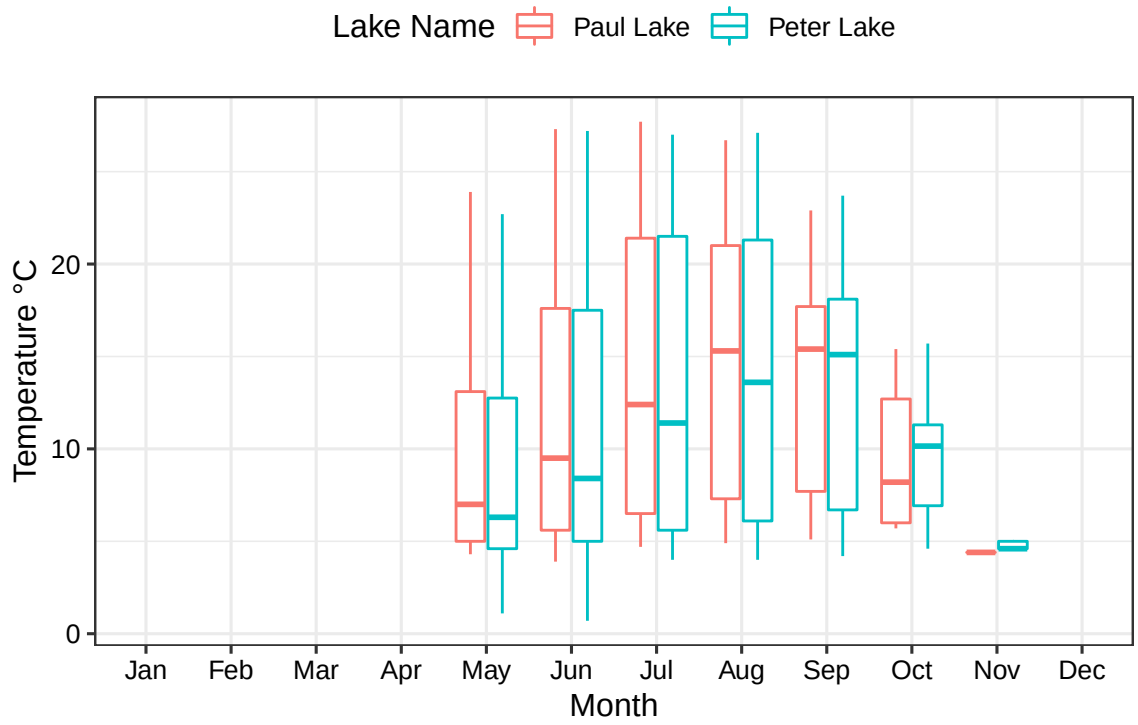

```
# I am using the ggplot() function to assign po4 to x and tp_ug to y to create
# a plot of phosphorus (tp_ug) BY phosphate (po4). To make the colors by lake,
# I set color = lakename. I am doing a scatter plot because it is the best
# representation of the data, especially since we are adding a regression line.
# Since there was an obvious extreme value, I set the x limit to 20. I chose to
# go to 20 because I think it better represents the entirety of the data and
# provides a regression line that better represents the trend. I chose for the
# y axis to go to 100 to get rid of extreme values. The xlab() and ylab()
# functions allow me to label the axes. geom_smooth() creates the regression
# line (with the method = lm). I use 'se=F' to get rid of the gray confidence
# interval. Lastly, I add my theme to the plot, and print it, so I can look at
# the outcome.
```

5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and

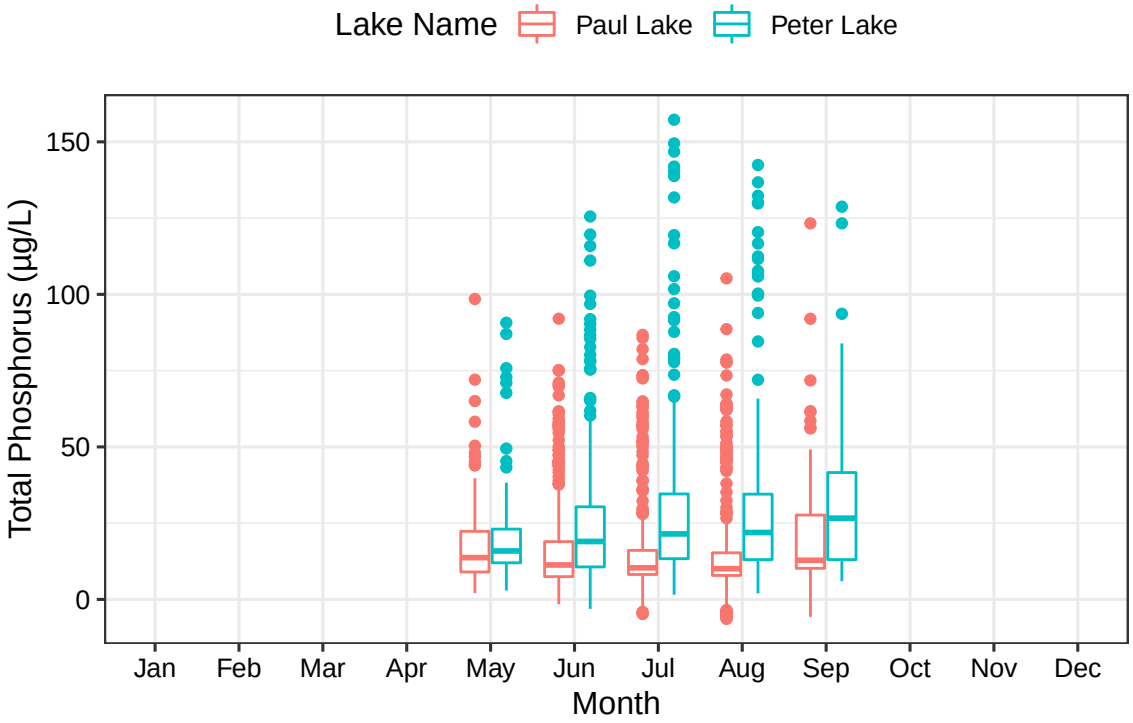
(c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

Tip: R has a build in variable called `month.abb` that returns a list of months; see https://r-lang.com/month-abb-in-r-with-example

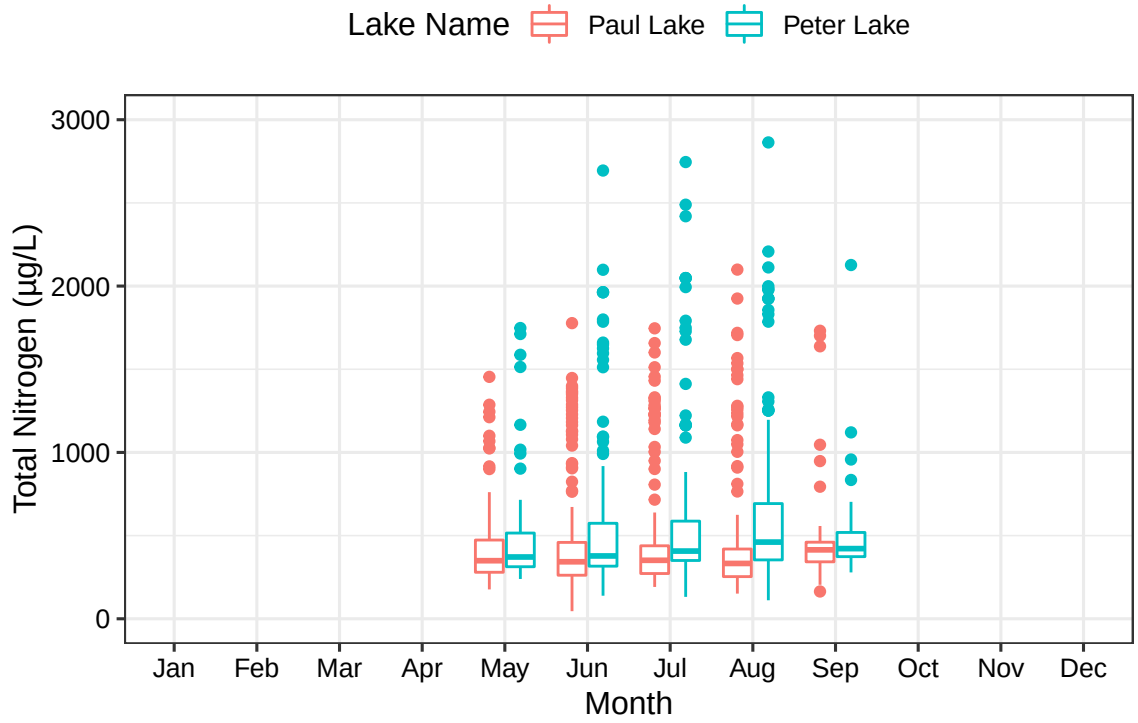
```
# 5

Temperature_Month <- ggplot(Chemistry_Nutrients_PeterPaul, aes(x = factor(month,
    levels = c(1:12)), y = temperature_C, color = lakename)) + geom_boxplot() + scale_x_discrete(labels
    drop = FALSE) + xlab("Month") + ylab("Temperature °C") + labs(color = "Lake Name") +
    mytheme
print(Temperature_Month)
```




```
TP_Month <- ggplot(Chemistry_Nutrients_PeterPaul, aes(x = factor(month, levels = c(1:12)),
    y = tp_ug, color = lakename)) + geom_boxplot() + scale_x_discrete(labels = month.abb,
    drop = FALSE) + xlab("Month") + ylab("Total Phosphorus (µg/L)") + labs(color = "Lake Name") +
    mytheme
print(TP_Month)
```

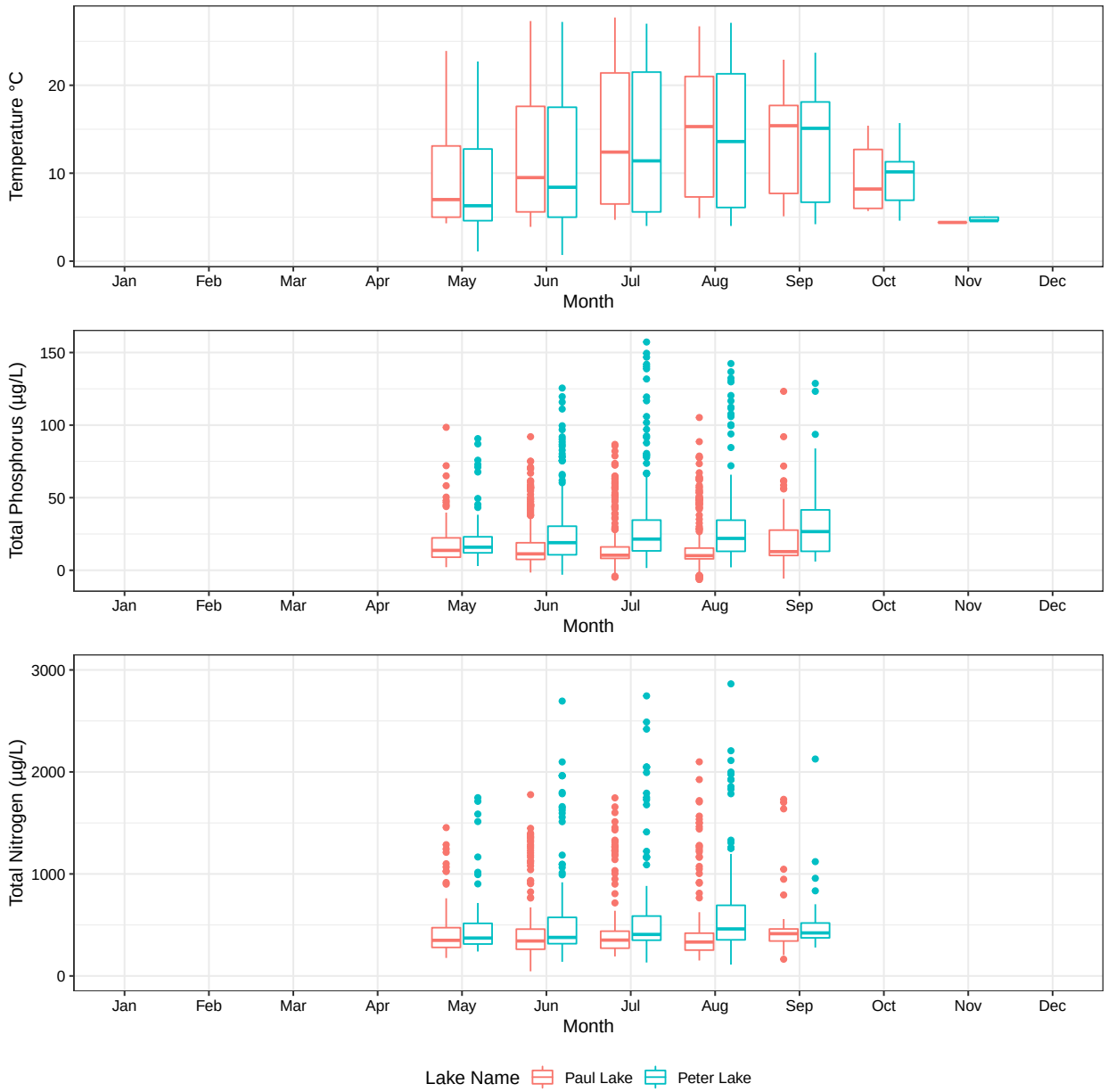



```
TN_Month <- ggplot(Chemistry_Nutrients_PeterPaul, aes(x = factor(month, levels = c(1:12)),
    y = tn_ug, color = lakename)) + geom_boxplot() + scale_x_discrete(labels = month.abb,
    drop = FALSE) + xlab("Month") + ylab("Total Nitrogen (µg/L)") + ylim(c = 0,
    3000) + labs(color = "Lake Name") + mytheme
print(TN_Month)
```




```
# For all three of the plots above I am using the ggplot() function to plot
# boxplots for each of the different variables. For each plot the x =
# factor(month) allows me to change the month values to factors going from 1 to
# 12. Each plot has a different y variable, and the color is based on the lake
# name. The scale_x_discrete() function allows me to change the labels on the
# x-axis from numbers to the actual month names (abbreviated), and drop = False
# prevents the plot from dropping the names with no values attach (i.e. keeping
# all the month names). I then use the xlab(), ylab(), and labs() functions to
# label the x-axis, y-axis, and legend respectively. For the total nitrogen
# graph, I added a y limit because there were a few extreme values beyond 3000
# that made the data difficult to see. I changed the figure height and width to
# better represent the plots when knitting.
```

```
Chem_Nutrient_Plot_Grid <- plot_grid(Temperature_Month + theme(legend.position = "none"),
    TP_Month + theme(legend.position = "none"), TN_Month + theme(legend.position = "bottom"),
    nrow = 3, ncol = 1, align = "v", rel_heights = c(7, 7, 10))
print(Chem_Nutrient_Plot_Grid)
```




```
# For this plot I used the plot_grid() function to put all three of the plots
# above into one. I used the theme() function to make sure only one of the
# plots had a legend (since the legends are the same). The nrow and ncol
# allowed me to ensure that the plots were all stacked on top of each other.
# The align = 'v' allowed me to make sure all the plots were vertically
# aligned, and the rel_heights allowed me to manually input how tall each plot
# was based on the information it was portraying. I decided to put this plot in
# a new r chunk, so I could change the figure height and width, so it would
# print more clearly in the knitted PDF.
```

Question: What do you observe about the variables of interest over seasons and between lakes?

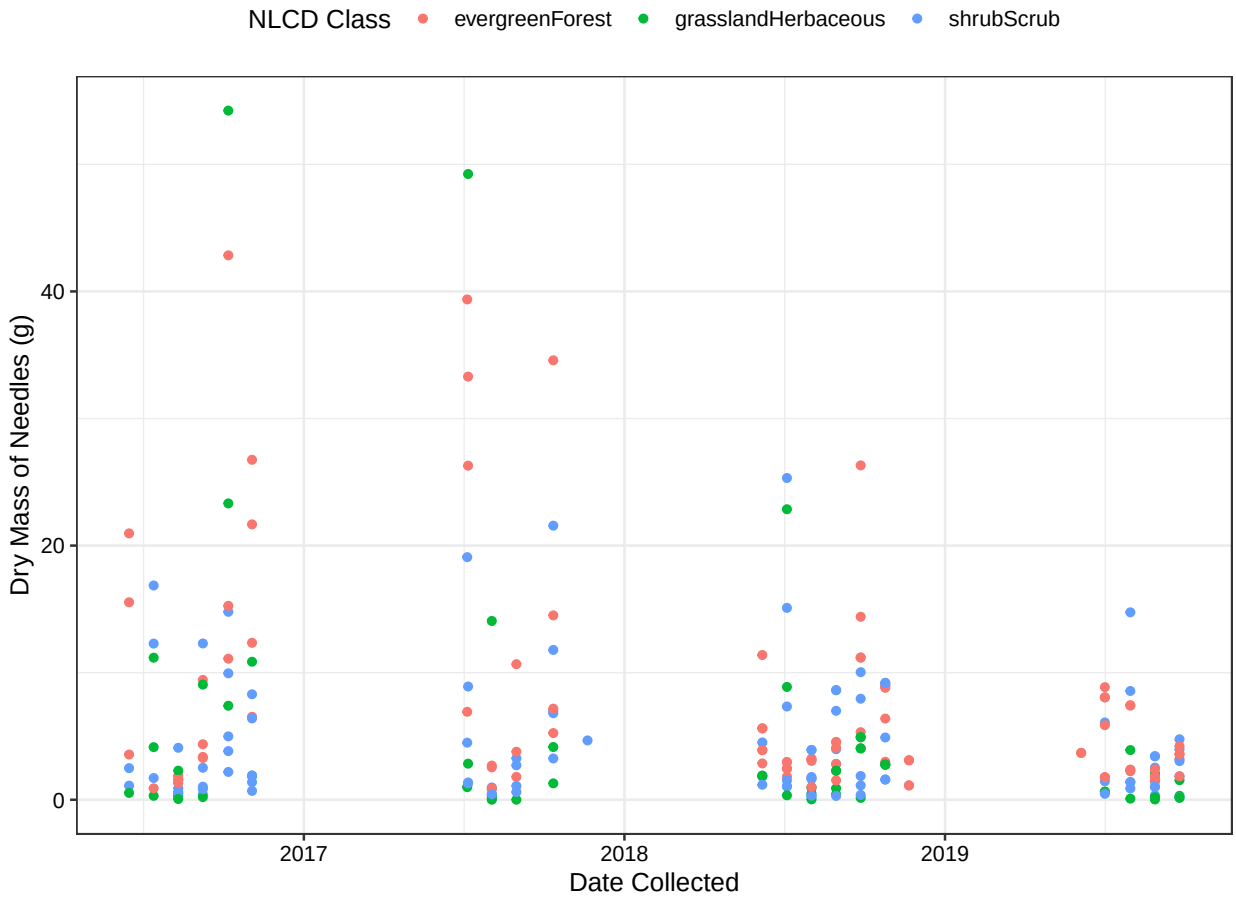
> Answer: Overall, it appears that Peter Lake generally has higher levels for each variable. For both lakes, temperature peaks in July and August, which makes sense for the region. Total phosphorous in Peter Lake gradually increases with the temperature. Total phosphorous in Paul

Lake seems not to follow the same pattern, but it is hard to say when only looking at the box plot (the interquartile range with the extreme values does not always represent the data accurately). When looking at the Nitrogen values, they follow a very similar trend as the phosphorous for both of the lakes. For Peter Lake, it is clear that both Nitrogen and Phosphorous increase with temperature. For Paul Lake, it is less clear, but it seems as though the values for Phosphorous and Nitrogen decrease with an increase in temperature. I think it would be essential to look at other plots (i.e., a violin plot) to get a better picture of the trends in Paul Lake.

6. [Niwot Ridge] Plot a subset of the litter dataset by displaying only the "Needles" functional group. Plot the dry mass of needle litter by date and separate by NLCD class with a color aesthetic. (no need to adjust the name of each land use)

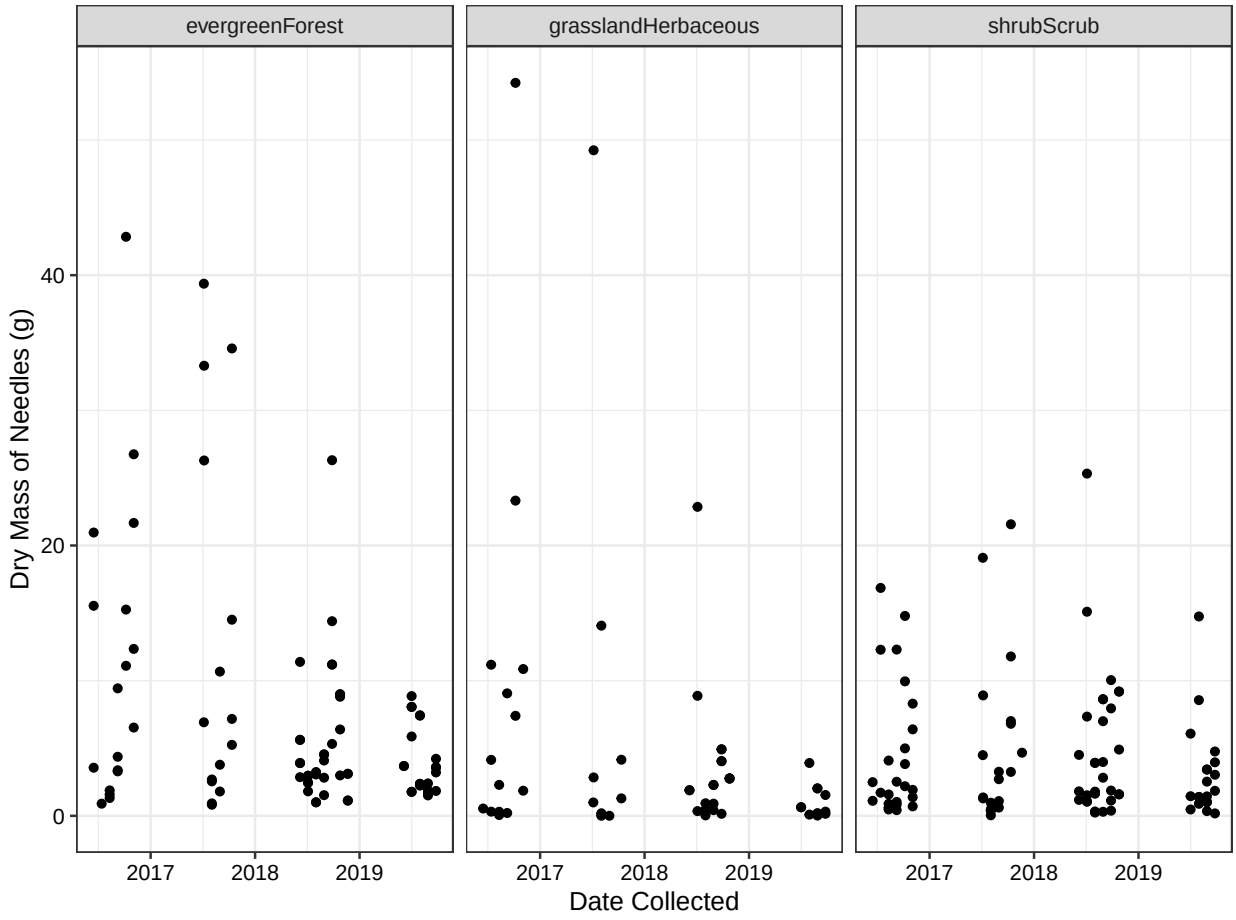
7. [Niwot Ridge] Now, plot the same plot but with NLCD classes separated into three facets rather than separated by color.

```
# 6
drymass_date <- ggplot(subset(NiwotRidge_Litter, functionalGroup == "Needles"), aes(x = collectDate,
    y = dryMass, color = nlcdClass)) + geom_point() + xlab("Date Collected") + ylab("Dry Mass of Needles
    labs(color = "NLCD Class") + mytheme
print(drymass_date)
```




```
# To create this plot first, I used the ggplot() function to define the data I
# was plotting. I used the subset() function to ensure that I am only plotting
# points that are in the functional group of needles. After I get the subset, I
# am able to define the x and y axis as the date and the dry mass respectively.
# Using the color = nlcdClass allows me to assign a color to the points
```

```
# representing each of the nlcd classes. The geom_point() allows me to graph
# the scatter plot. The xlab() and ylab() allow me to name the axes
# respectively. The labs() function is similar, but this allows me to name the
# legend. Lastly, I add my theme to the plot and use the print() function to
# view it. For both of these plots, I change the figure height and width to
# show the entire plot when knitting.

# 7
drymass_date_facet <- ggplot(subset(NiwotRidge_Litter, functionalGroup == "Needles"),
    aes(x = collectDate, y = dryMass)) + geom_point() + facet_wrap(vars(nlcdClass)) +
    ylab("Dry Mass of Needles (g)") + xlab("Date Collected") + mytheme
print(drymass_date_facet)
```




```
# To create this plot I used the same subset as above. The only difference to
# this plot is the facet_wrap() function which allows me to separate the plots
# by the different nlcd classes.
```

Question: Which of these plots (6 vs. 7) do you think is more effective, and why?

> Answer: I think plot 7 is more sufficient for viewing the data because you can more effectively see each of the different NLCD classes and the data points of needle dry mass that fall within each class. The first plot (plot 6) makes it harder to see the differences between the NLCD classes, even though they are colored differently. You are not able to see trends. In plot 7, the similarities and differences between the data are more prominent.