Assignment 5: Data Visualization

Sayra Martinez

Spring 2024

OVERVIEW

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

Directions

1. Rename this file `<FirstLast>_A05_DataVisualization.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 your code is tidy; use line breaks to ensure your code fits in the knitted output.
5. Be sure to **answer the questions** in this assignment document.
6. When you have completed the assignment, **Knit** the text and code into a single PDF file.

Set up your session

1. Set up your session. Load the tidyverse, lubridate, here & cowplot packages, and verify your home directory. Read in the NTL-LTER processed data files for nutrients and chemistry/physics for Peter and Paul Lakes (use the tidy `NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv` version in the Processed_KEY folder) and the processed data file for the Niwot Ridge litter dataset (use the `NEON_NIWO_Litter_mass_trap_Processed.csv` version, again from the Processed_KEY folder).

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

```
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=80), tidy=TRUE)
#1 Installing libraries, setting my directory and importing my datasets
library(tidyverse);library(lubridate);library(here); library(dplyr); library(cowplot); library(viridis)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## here() starts at /home/guest/EDA_Spring2024
##
##
## Attaching package: 'cowplot'
##
##
## The following object is masked from 'package:lubridate':
##
##     stamp
##
##
## Loading required package: viridisLite
```

```
getwd()
```

```
## [1] "/home/guest/EDA_Spring2024"
```

```
PeterPaul.chem.nutrients <-
  read.csv(here("Data/Processed_KEY/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv"),
           stringsAsFactors = T)

Litter <-
  read.csv(here("Data/Processed_KEY/NEON_NIWO_Litter_mass_trap_Processed.csv"),
           stringsAsFactors = T)

#2 Setting and verifying sampledate & collectDate as date
class(PeterPaul.chem.nutrients$sampledate)
```

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

```
PeterPaul.chem.nutrients$sampledate <- mdy(PeterPaul.chem.nutrients$sampledate)
```

```
## Warning: All formats failed to parse. No formats found.
```

```
class(PeterPaul.chem.nutrients$sampledate)
```

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

```
class(Litter$collectDate)
```

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

```
Litter$collectDate <- as.Date(Litter$collectDate)
class(Litter$collectDate)
```

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

Define your theme

3. Build a theme and set it as your default theme. Customize the look of at least two of the following:

- Plot background
- Plot title
- Axis labels
- Axis ticks/gridlines
- Legend

```
#3 Creating my theme

sayra.theme <-
  theme(axis.text = element_text(color = "black"),

        legend.position = "right",
        legend.direction = "vertical",
        plot.title =        element_text(color = "black", size = 10),
        axis.ticks =        element_line(6),
        axis.text.x = element_text(color = "darkred", size = 7),
        axis.text.y = element_text(color = "darkred", size = 7),
        plot.background = element_rect(fill = "gray90"),
        legend.background = element_rect(
      color="black"
      ),
    legend.title = element_text(
      color="royalblue", size = 7.5
    )
  )

theme_set(sayra.theme)
```

Create graphs

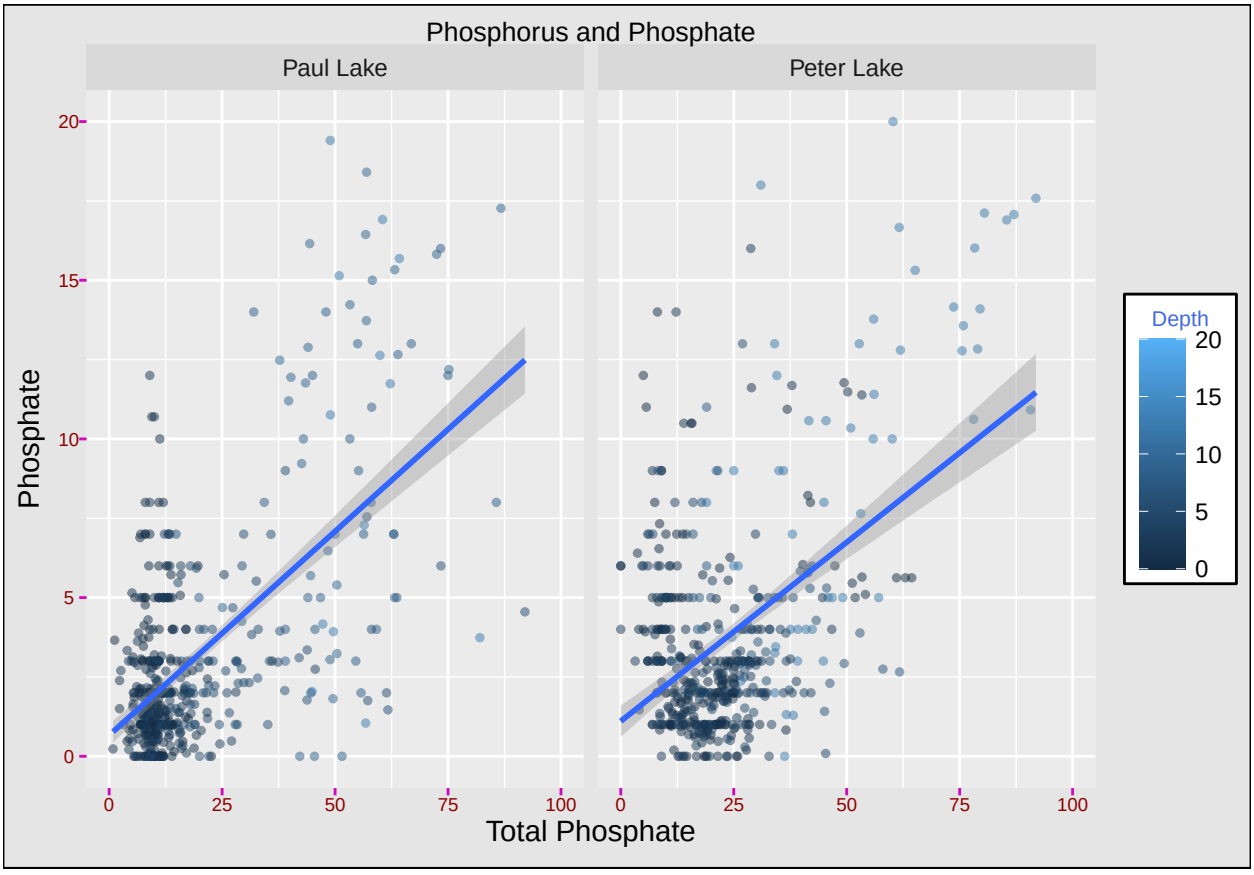
For numbers 4-7, create ggplot graphs and adjust aesthetics to follow best practices for data visualization. Ensure your theme, color palettes, axes, and additional aesthetics are edited accordingly.

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

```
# 4

Phosphorus_Phosphate <- ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = po4,
    color = depth)) + geom_point(alpha = 0.5, size = 1) + xlim(0, 100) + ylim(0,
    20) + facet_wrap(vars(lakename)) + ggtitle("Phosphorus and Phosphate") + labs(color = "Depth") +
    xlab(expression(paste("Total Phosphate"))) + ylab(expression(paste("Phosphate"))) +
    geom_smooth(method = lm)
print(Phosphorus_Phosphate)
```

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

```
## Warning: Removed 21986 rows containing non-finite values (`stat_smooth()`).

## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?

## Warning: Removed 21986 rows containing missing values (`geom_point()`).
```

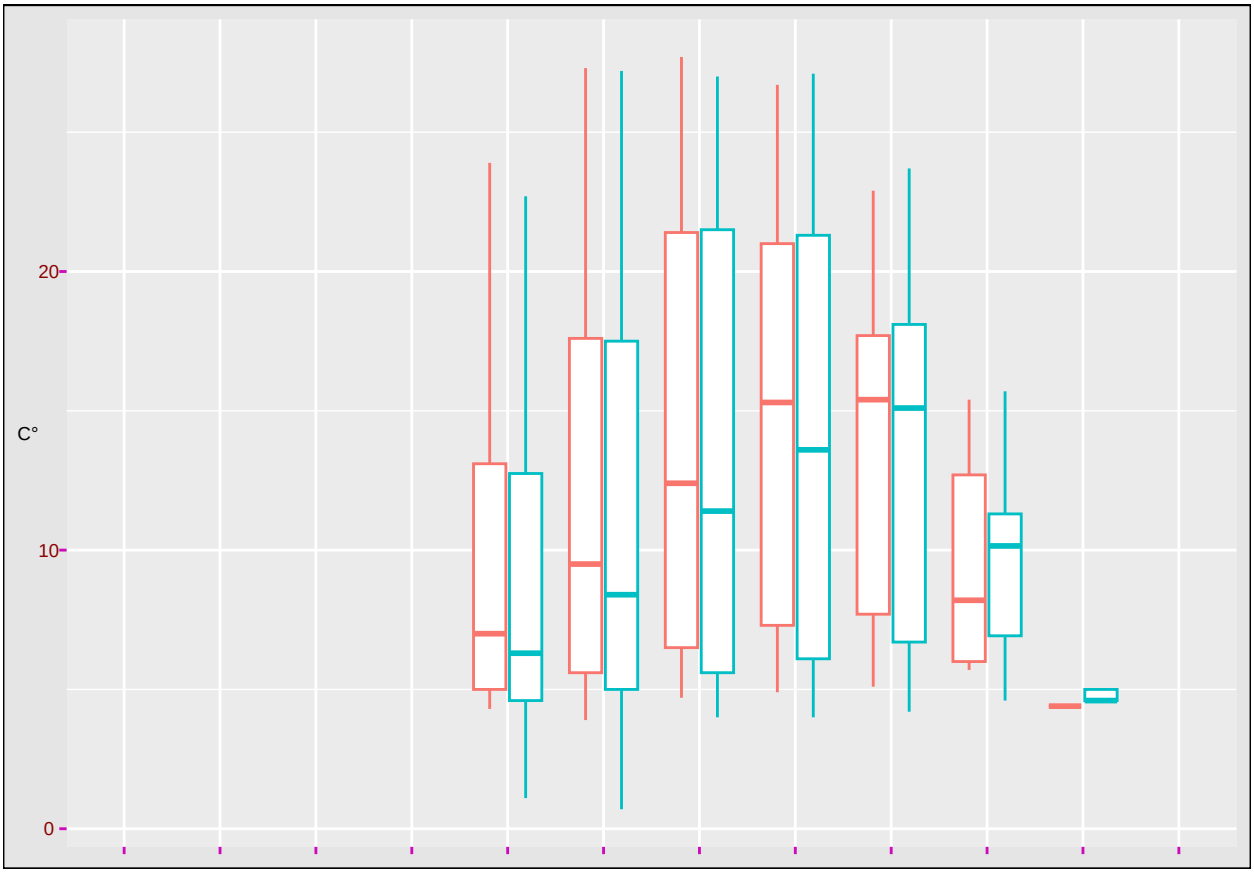


5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and (c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

Tips: * Recall the discussion on factors in the lab section as it may be helpful here. * Setting an axis title in your theme to `element_blank()` removes the axis title (useful when multiple, aligned plots use the same axis values) * Setting a legend's position to "none" will remove the legend from a plot. * Individual plots can have different sizes when combined using `cowplot`.

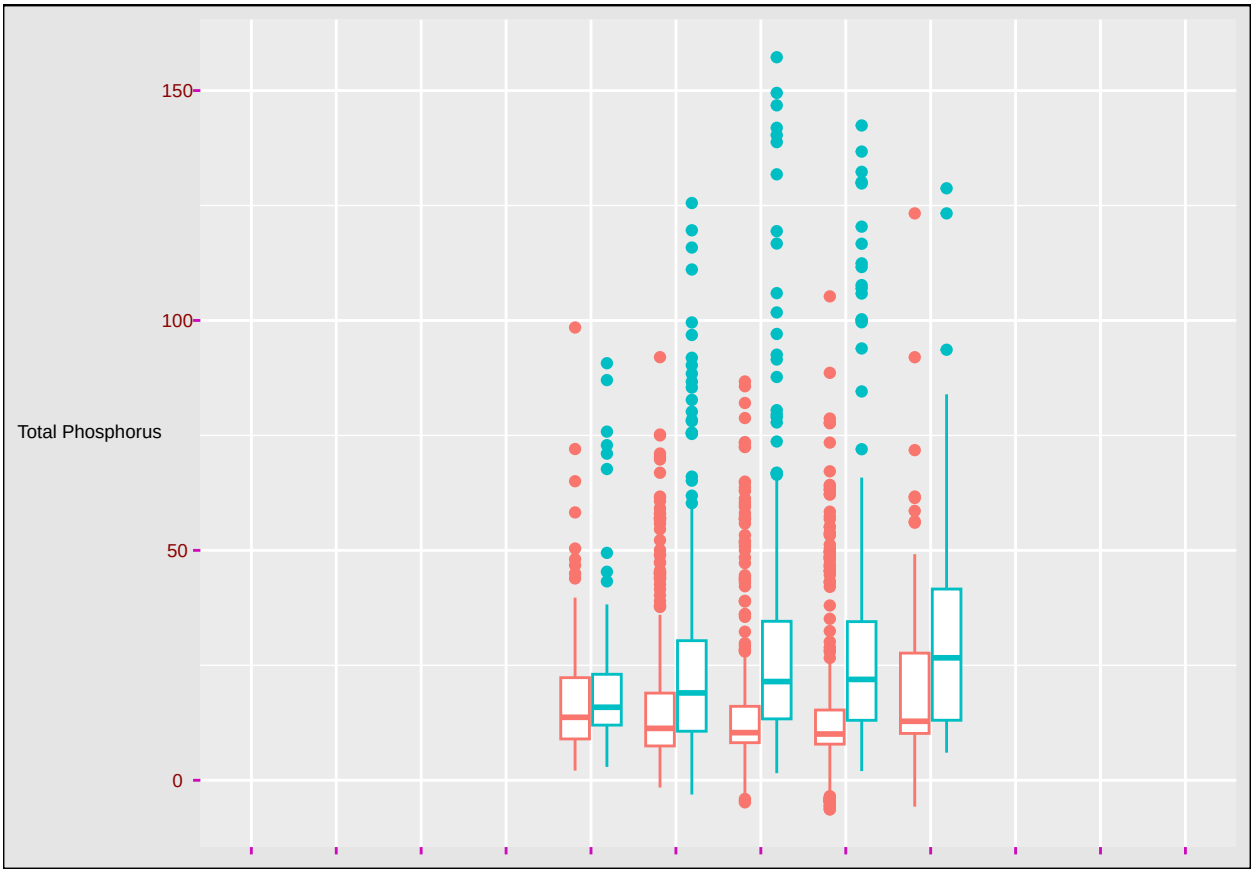
```
# 5 I create the plotboxes, but first I put month as a factor

PeterPaul.chem.nutrients$month_f <- factor(PeterPaul.chem.nutrients$month, levels = 1:12,
    labels = month.abb)

Temp.Boxplot <- ggplot(PeterPaul.chem.nutrients, aes(x = month_f, y = temperature_C)) +
    geom_boxplot(aes(color = lakename)) + scale_x_discrete(drop = F) + ylab(expression(paste("C°"))) +
    theme(axis.text.x = element_blank(), legend.position = "none", axis.title.x = element_blank(),
        axis.title.y = element_text(size = 7))
print(Temp.Boxplot)
```

```
## Warning: Removed 3566 rows containing non-finite values (‘stat_boxplot()‘).
```



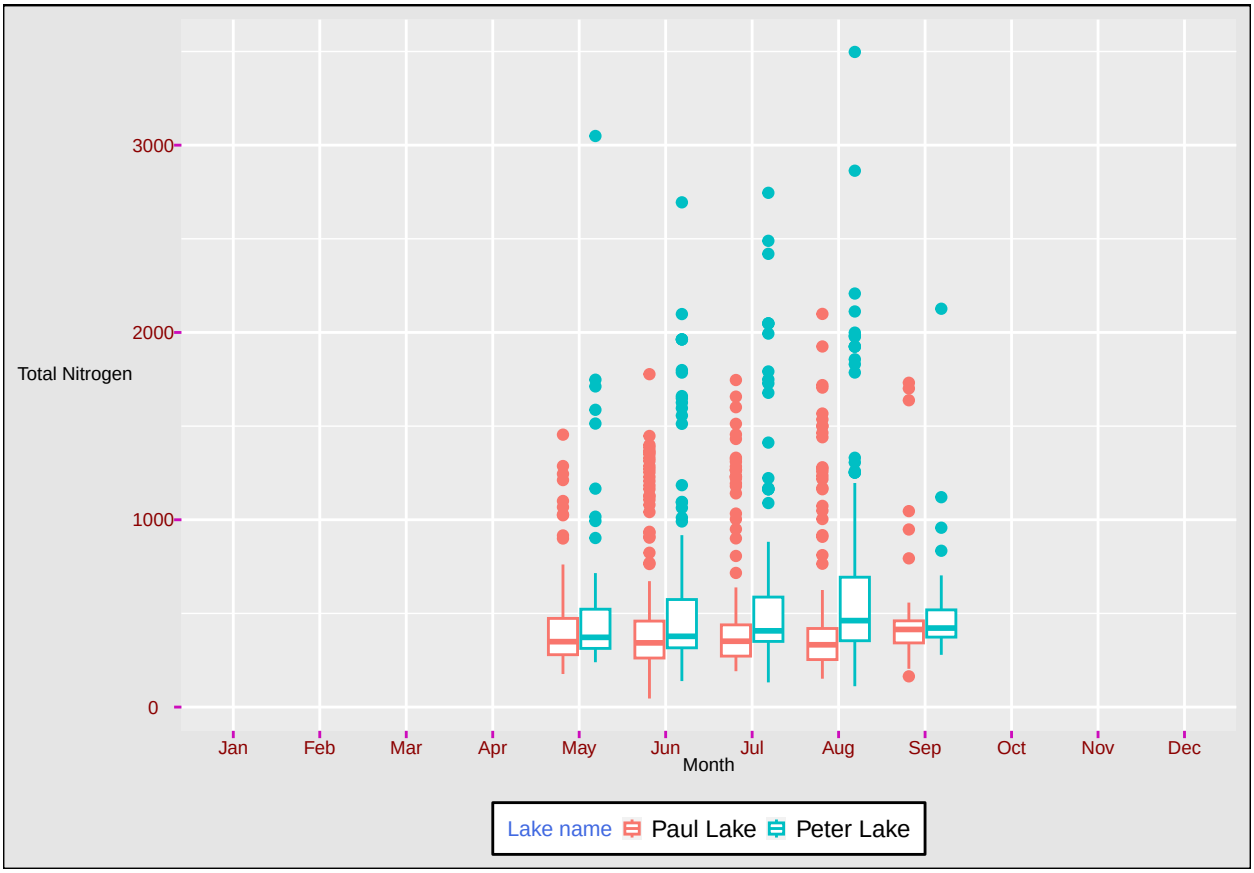

```
Phospho.boxplot <- ggplot(PeterPaul.chem.nutrients, aes(x = month_f, y = tp_ug)) +
    geom_boxplot(aes(color = lakename)) + scale_x_discrete(drop = F) + ylab(expression(paste("Total Phos
    theme(axis.text.x = element_blank(), legend.position = "none", axis.title.x = element_blank(),
        axis.title.y = element_text(size = 7))
print(Phospho.boxplot)
```

```
## Warning: Removed 20729 rows containing non-finite values (‘stat_boxplot()‘).
```




```
Nitro.boxplot <- ggplot(PeterPaul.chem.nutrients, aes(x = month_f, y = tn_ug)) +
    geom_boxplot(aes(color = lakename)) + scale_x_discrete(drop = F) + labs(color = "Lake name") +
    xlab(expression(paste("Month"))) + ylab(expression(paste("Total Nitrogen"))) +
    theme(legend.position = "bottom", legend.direction = "horizontal", axis.title.x = element_text(size
        axis.title.y = element_text(size = 7), legend.key.size = unit(0.5, "line"))
print(Nitro.boxplot)
```

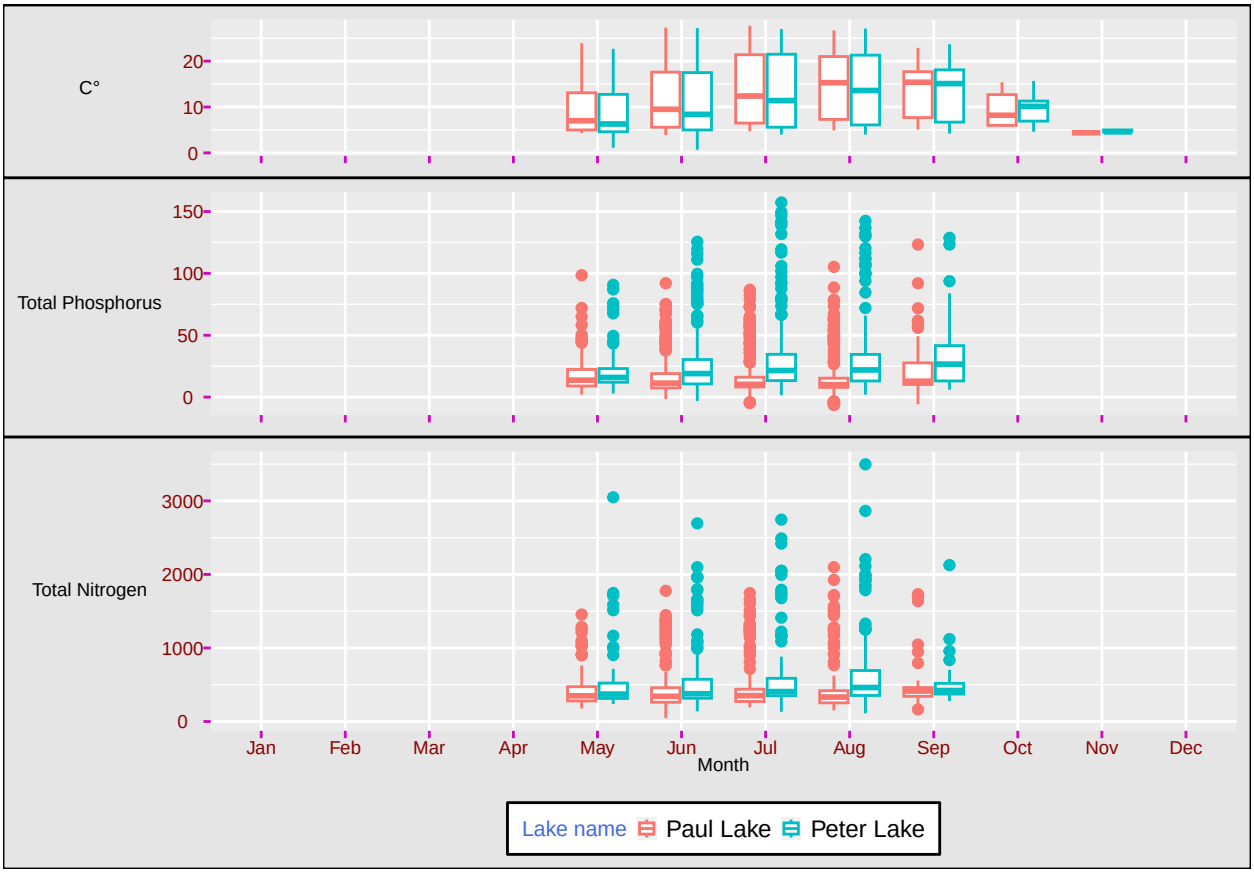
```
## Warning: Removed 21583 rows containing non-finite values (‘stat_boxplot()‘).
```




```
Three.Nutrients <- plot_grid(Temp.Boxplot, Phospho.boxplot, Nitro.boxplot, nrow = 3,
    rel_heights = c(1, 1.5, 2.5), align = T)
```

```
## Warning: Removed 3566 rows containing non-finite values (‘stat_boxplot()‘).

## Warning: Removed 20729 rows containing non-finite values (‘stat_boxplot()‘).

## Warning: Removed 21583 rows containing non-finite values (‘stat_boxplot()‘).

## Warning: Graphs cannot be horizontally aligned unless the axis parameter is
## set. Placing graphs unaligned.
```

```
print(Three.Nutrients)
```



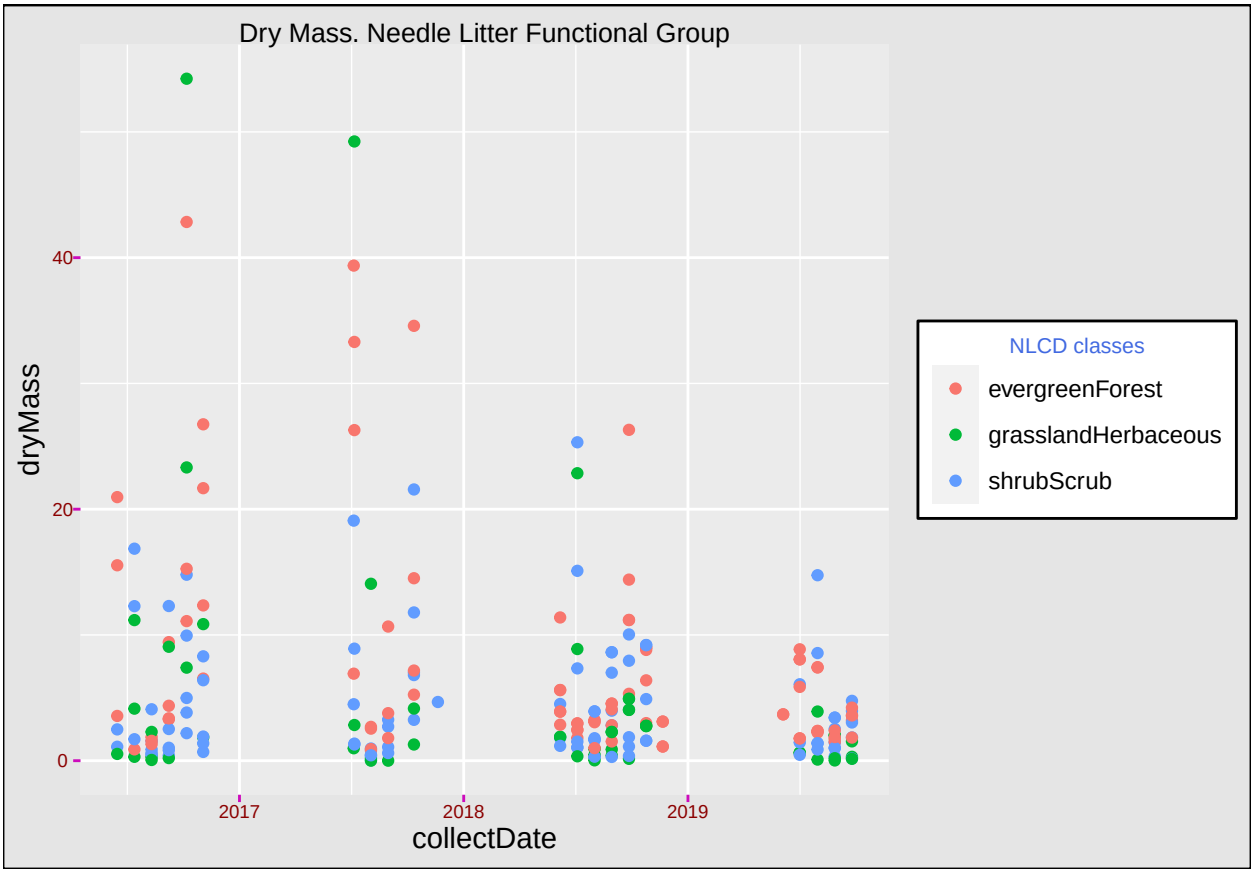
Question: What do you observe about the variables of interest over seasons and between lakes?

> Answer: In both cases, as water's temperature increases, during the summer (from June to August), the TP and TN levels are also higher, specially in the case of Peter Lake.

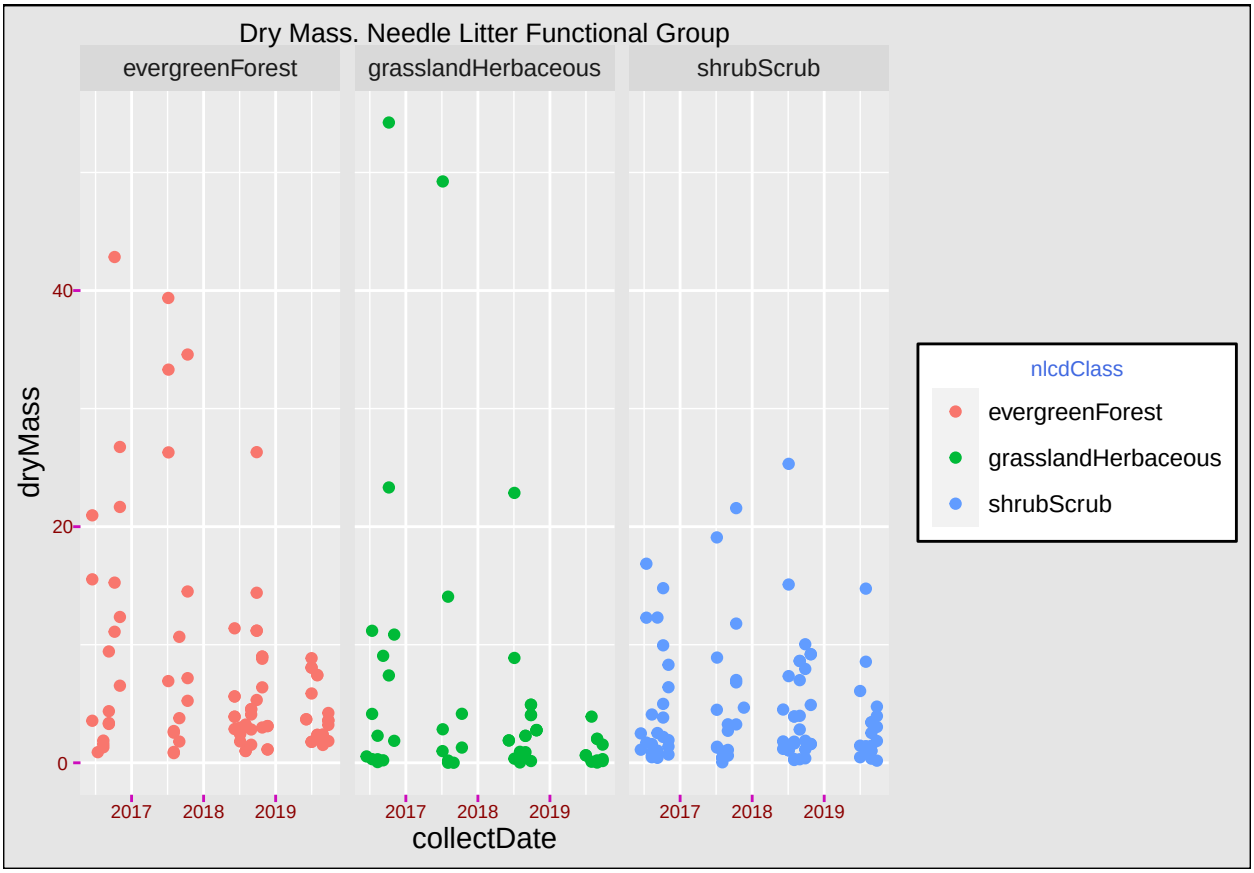
6. [Niwot Ridge] Plot a subset of the litter dataset by displaying only the "Needles" functional group. Plot the dry mass of needle litter by date and separate by NLCD class with a color aesthetic. (no need to adjust the name of each land use)

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

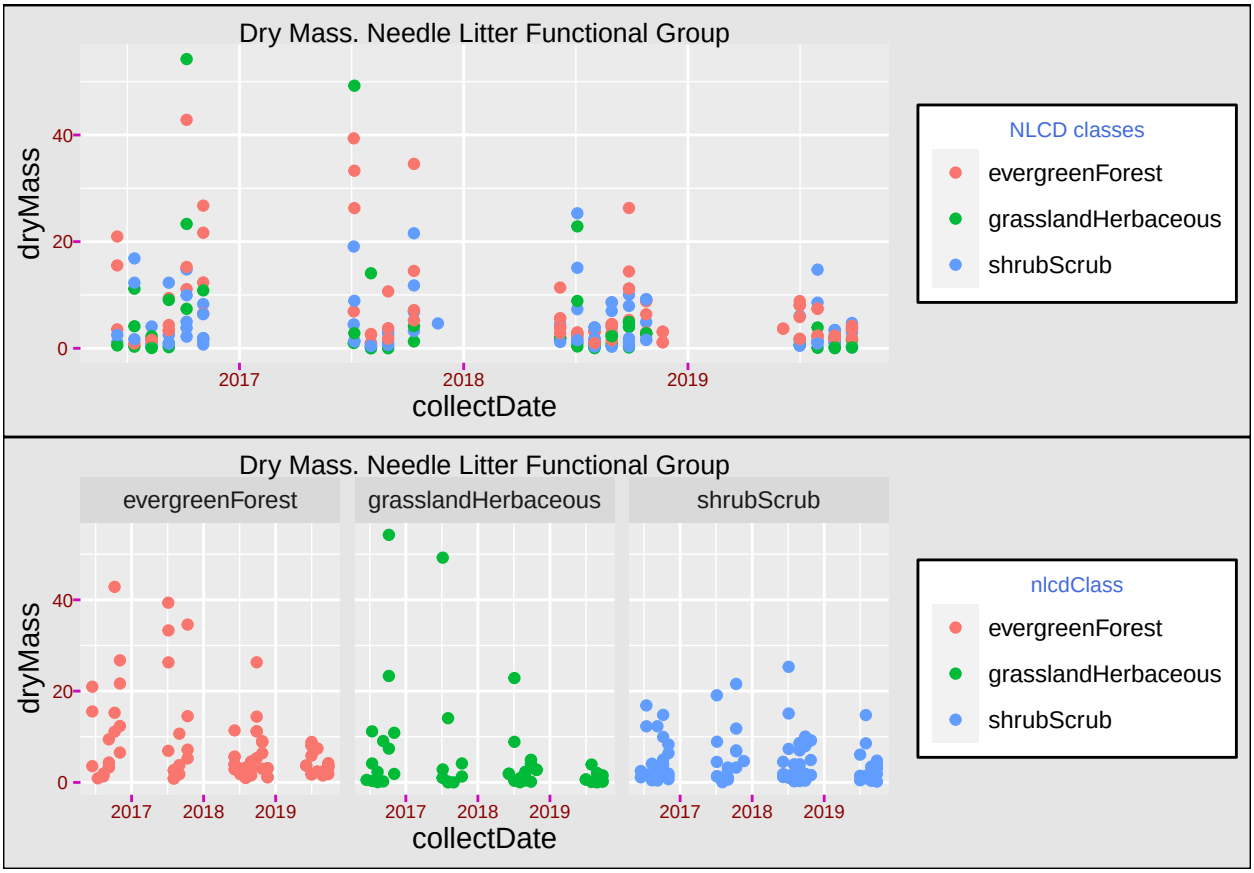
```
# 6 Create subgroup and plot, using NLCD classes as colors

Litter.Needles <- filter(Litter, functionalGroup == "Needles")

Needles.plot <- Litter.Needles %>%
    filter(functionalGroup == "Needles") %>%
    ggplot(aes(x = collectDate, y = dryMass, color = nlcdClass)) + geom_point() +
    labs(title = "Dry Mass. Needle Litter Functional Group") + theme(legend.position = "right",
    legend.direction = "vertical") + labs(color = "NLCD classes")
Needles.plot
```




```
# 7 Plot with NLCD separated

Litter.Needles <- filter(Litter, functionalGroup == "Needles")

Needles.plot2 <- Litter.Needles %>%
    filter(functionalGroup == "Needles") %>%
    ggplot(aes(x = collectDate, y = dryMass, color = nlcdClass)) + geom_point() +
    facet_wrap(vars(nlcdClass)) + labs(title = "Dry Mass. Needle Litter Functional Group") +
    theme(legend.position = "right", legend.direction = "vertical")
Needles.plot2
```




```
# I put them together to answer
Needles.Together <- plot_grid(Needles.plot, Needles.plot2, nrow = 2, rel_heights = c(1.2,
    1.2))
print(Needles.Together)
```



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

Answer: The one separating NLCD classes is more effective in this case, because it allows to observe clearly the Dry Mass of each of them; while in the one that only put them as colors, they overlap and it is no clear the observations.