# Assignment 5: Data Visualization

Siyu Dong

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.

```
#1
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     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
```

```
library(lubridate)
library(here)
```

```
## here() starts at /home/guest/EDA_Spring2024
```

```
library(cowplot)
```

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

```
getwd()
```

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

```
PeterPaul.chem.nutrients <-
  read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv",
           stringsAsFactors = TRUE)
NiwotRidgeLitter <-
  read.csv("./Data/Processed/NEON_NIWO_Litter_mass_trap_Processed.csv",
           stringsAsFactors = TRUE)

#2
NiwotRidgeLitter$collectDate <-
  as.Date(NiwotRidgeLitter$collectDate, format = "%Y-%m-%d")
PeterPaul.chem.nutrients$sampledate <-
  as.Date(PeterPaul.chem.nutrients$sampledate, format = "%Y-%m-%d")
```

## 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
library(ggplot2)

ivy_theme <-  theme_gray()+
  theme(plot.title = element_text(color = "darkblue", size = 16, face = "bold", hjust = 0.5),
        axis.title.x = element_text(color = "darkblue", size = 12),
        axis.title.y = element_text(color = "darkblue", size = 12),
        legend.position = "right",
```

```
        legend.title = element_text(color = "darkblue", size = 12),
        legend.text = element_text(color = "darkblue", size = 10))

# Set custom theme as default theme
theme_set(ivy_theme)
```

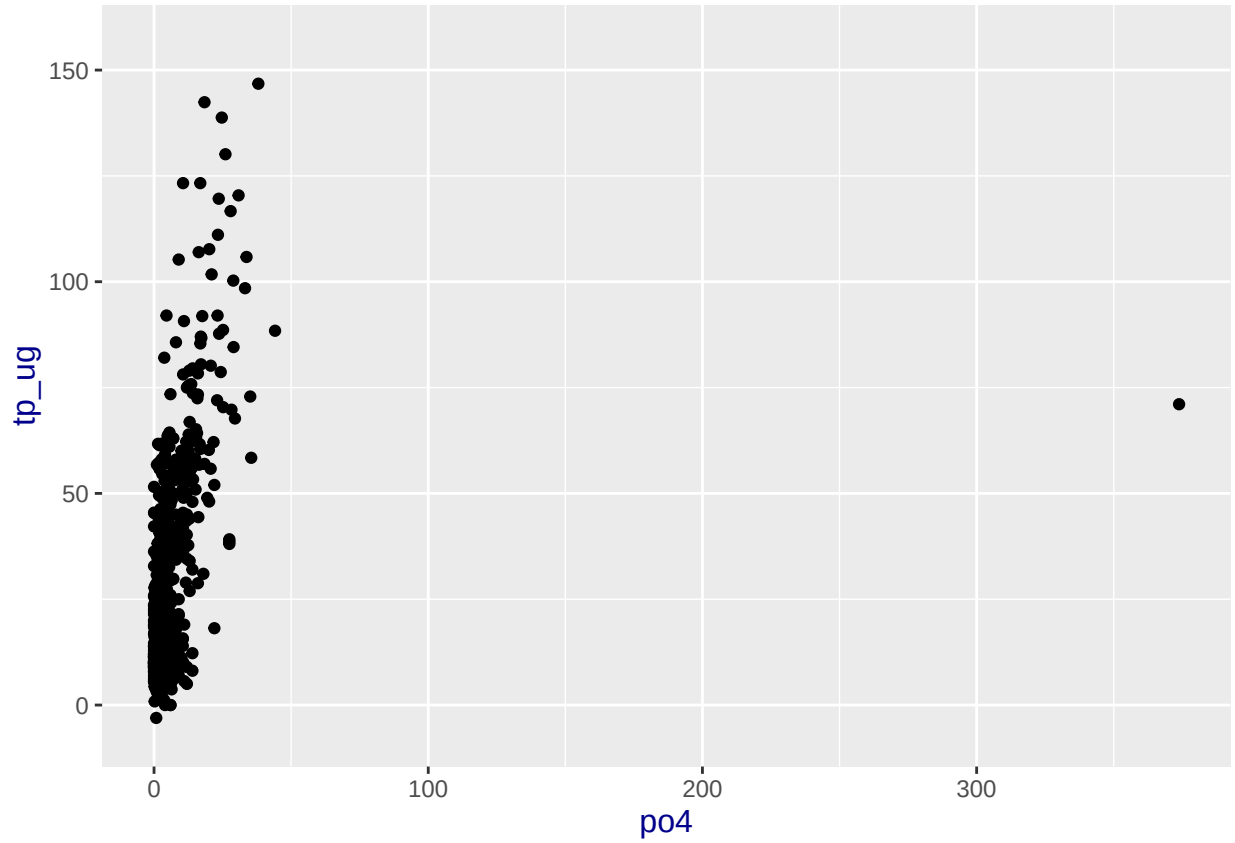
## Create graphs

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
# First preliminarily plot  total phosphorus (`tp_ug`) by phosphate (`po4`),
# with separate aesthetics for Peter and Paul lakes to check if there are any extreme values.
ggplot(PeterPaul.chem.nutrients, aes(x = po4, y = tp_ug)) +
  geom_point()
```

```
## Warning: Removed 21946 rows containing missing values (‘geom_point()‘).
```

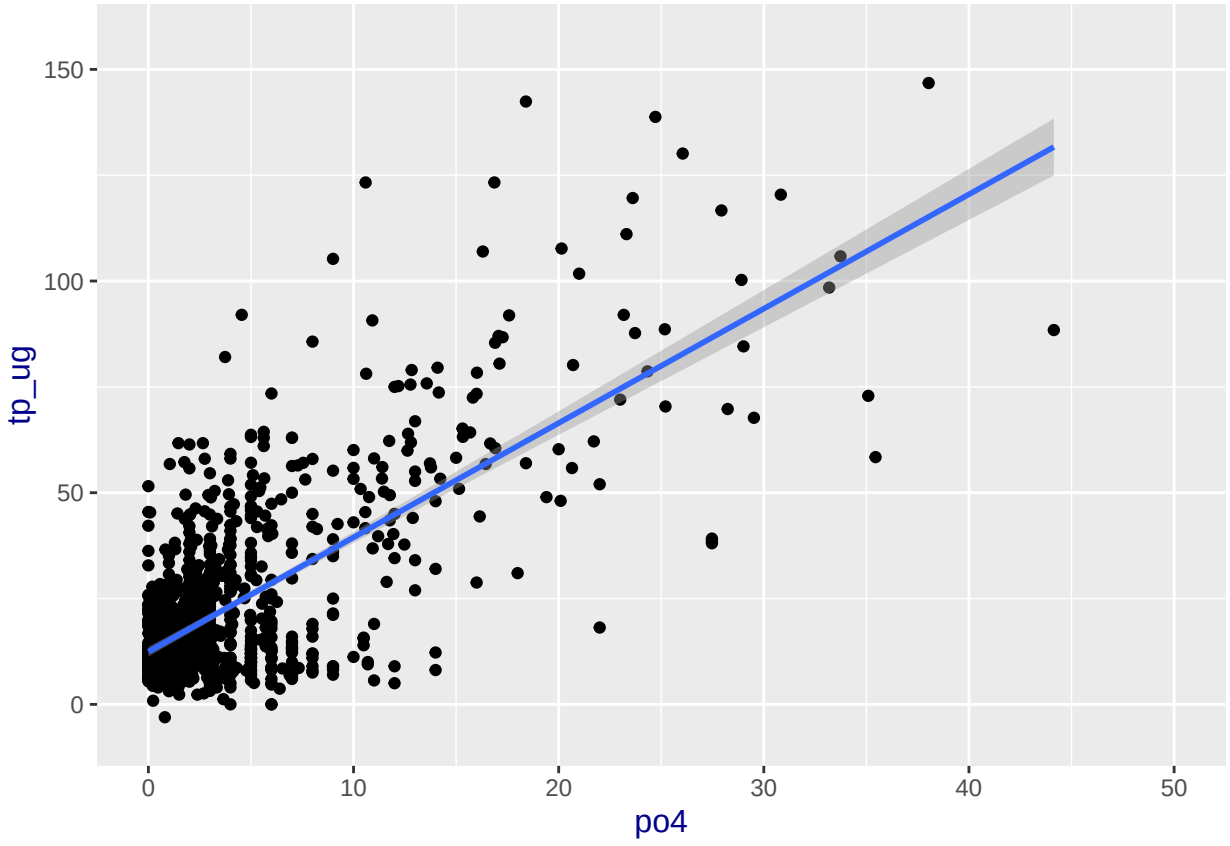


```
# Most of the observations have the po4 values within 50.
# The only extreme observation is around (po4=375, tp_ug=75),
# which influces the limit of x axis setting greatly.
# Therefore, here I set xlim to (0,50) and remain the same for tp_ug.
ggplot(PeterPaul.chem.nutrients, aes(x = po4, y = tp_ug)) +
  geom_point() +
  xlim(0, 50) +
  geom_smooth(method = lm)
```

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

## Warning: Removed 21947 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 21947 rows containing missing values (`geom_point()`).
```

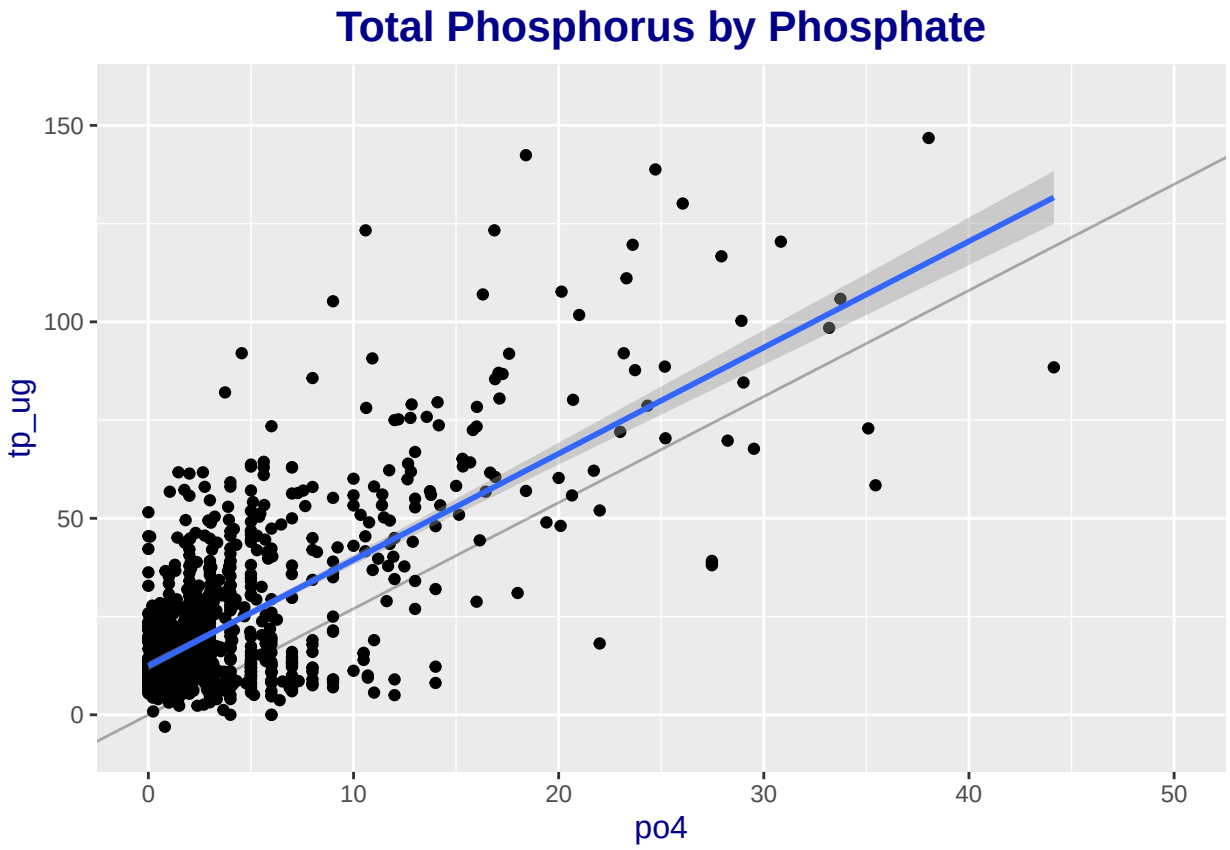



```
# Add an abline on the basis of previous plot and adjust aesthetics
P_PointLine<- ggplot(PeterPaul.chem.nutrients, aes(x = po4, y = tp_ug)) +
  geom_point() +
  xlim(0, 50) +
  geom_smooth(method = lm) +
  geom_abline(aes(slope = 2.7, intercept = 0), alpha = 0.3) +
  labs(title = "Total Phosphorus by Phosphate") +
  ivy_theme

print(P_PointLine)
```

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

```
## Warning: Removed 21947 rows containing non-finite values (`stat_smooth()`).
## Removed 21947 rows containing missing values (`geom_point()`).
```



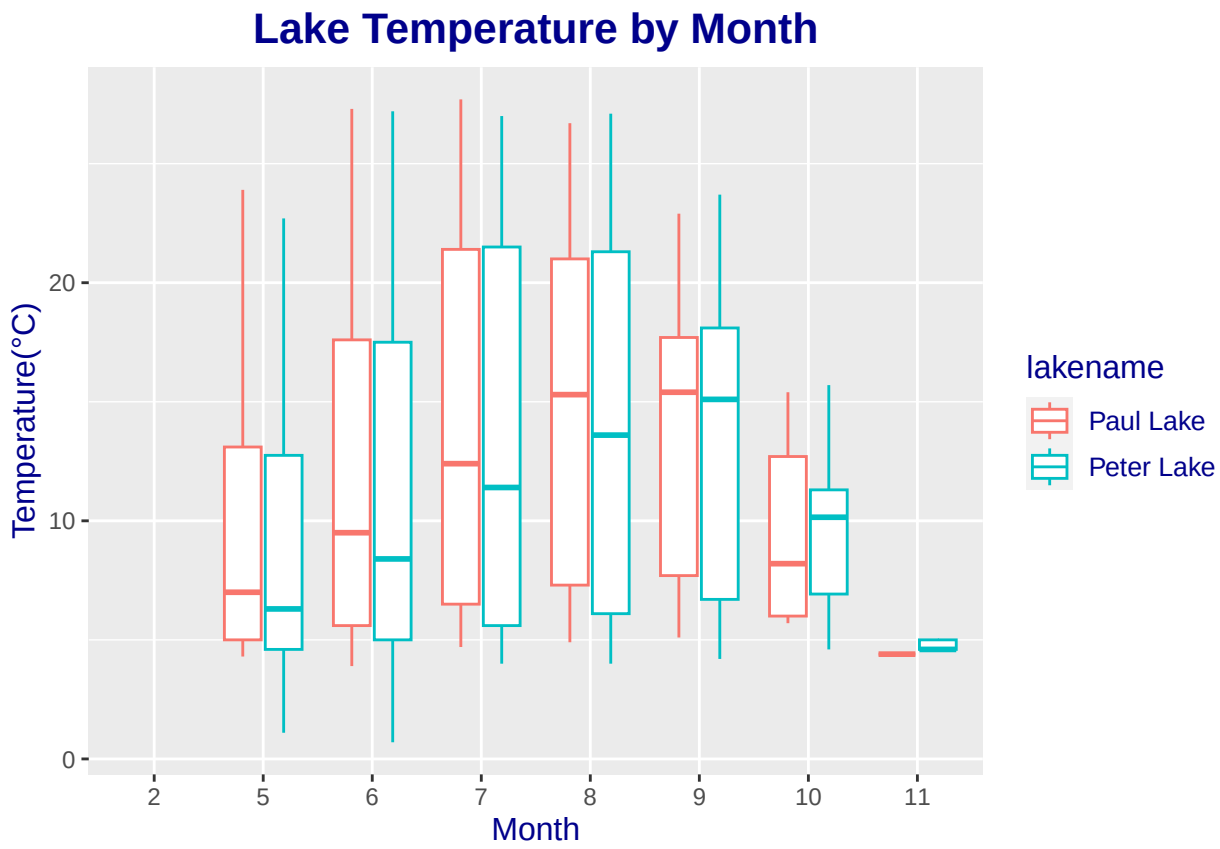

5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and (c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

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
NTL_LTER_BoxA <-
  ggplot(PeterPaul.chem.nutrients,
         aes(x = factor(month), y = temperature_C, color = lakename))+
  geom_boxplot() +
  labs(title = "Lake Temperature by Month", x = "Month", y = "Temperature(°C)") +
  ivy_theme

print(NTL_LTER_BoxA)
```

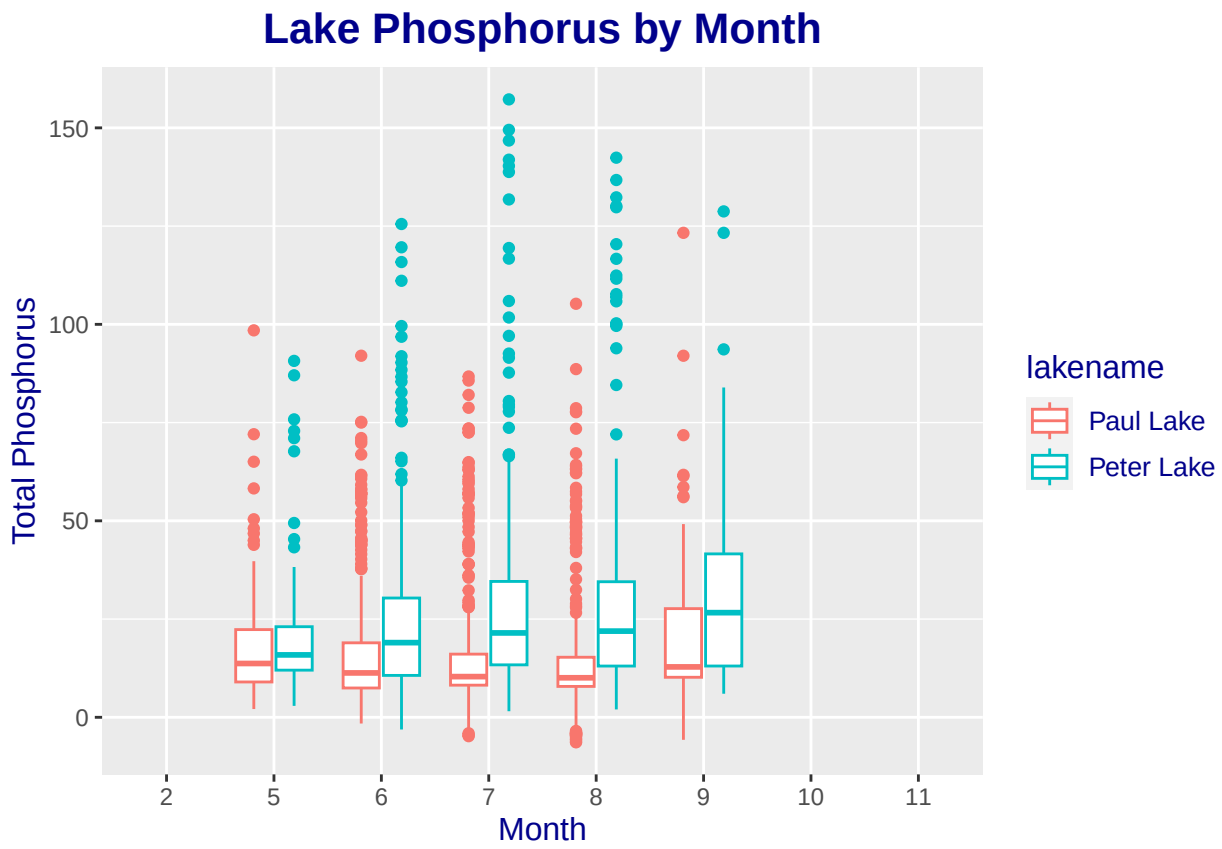
```
## Warning: Removed 3566 rows containing non-finite values (`stat_boxplot()`).
```




```
NTL_LTER_BoxB <-
  ggplot(PeterPaul.chem.nutrients,
         aes(x = factor(month), y = tp_ug, color = lakename))+
  geom_boxplot() +
  labs(title = "Lake Phosphorus by Month", x = "Month", y = "Total Phosphorus") +
  ivy_theme

print(NTL_LTER_BoxB)
```

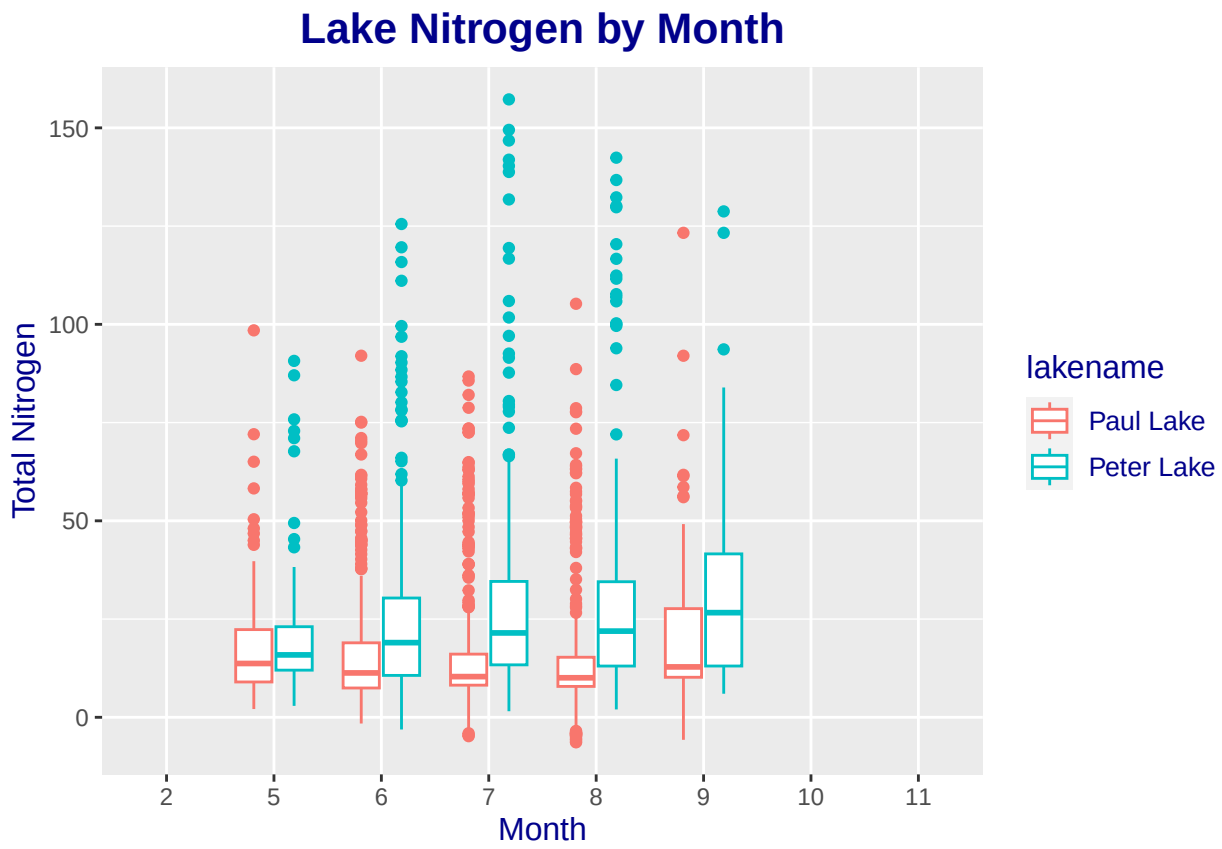
```
## Warning: Removed 20729 rows containing non-finite values (‘stat_boxplot()‘).
```




```
NTL_LTER_BoxC <-
  ggplot(PeterPaul.chem.nutrients,
         aes(x = factor(month), y = tp_ug, color = lakename))+
  geom_boxplot() +
  labs(title = "Lake Nitrogen by Month", x = "Month", y = "Total Nitrogen") +
  ivy_theme

print(NTL_LTER_BoxC)
```

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




```
library(cowplot)

# Combine the individual plots
combined_boxes <- plot_grid(
  NTL_LTER_BoxA + theme(legend.position = "none", axis.title.x = element_blank()),
  NTL_LTER_BoxB + theme(legend.position = "none", axis.title.x = element_blank()),
  NTL_LTER_BoxC + theme(axis.title.x = element_blank()),
  align = "v", rel_heights = c(1, 1, 1))
```

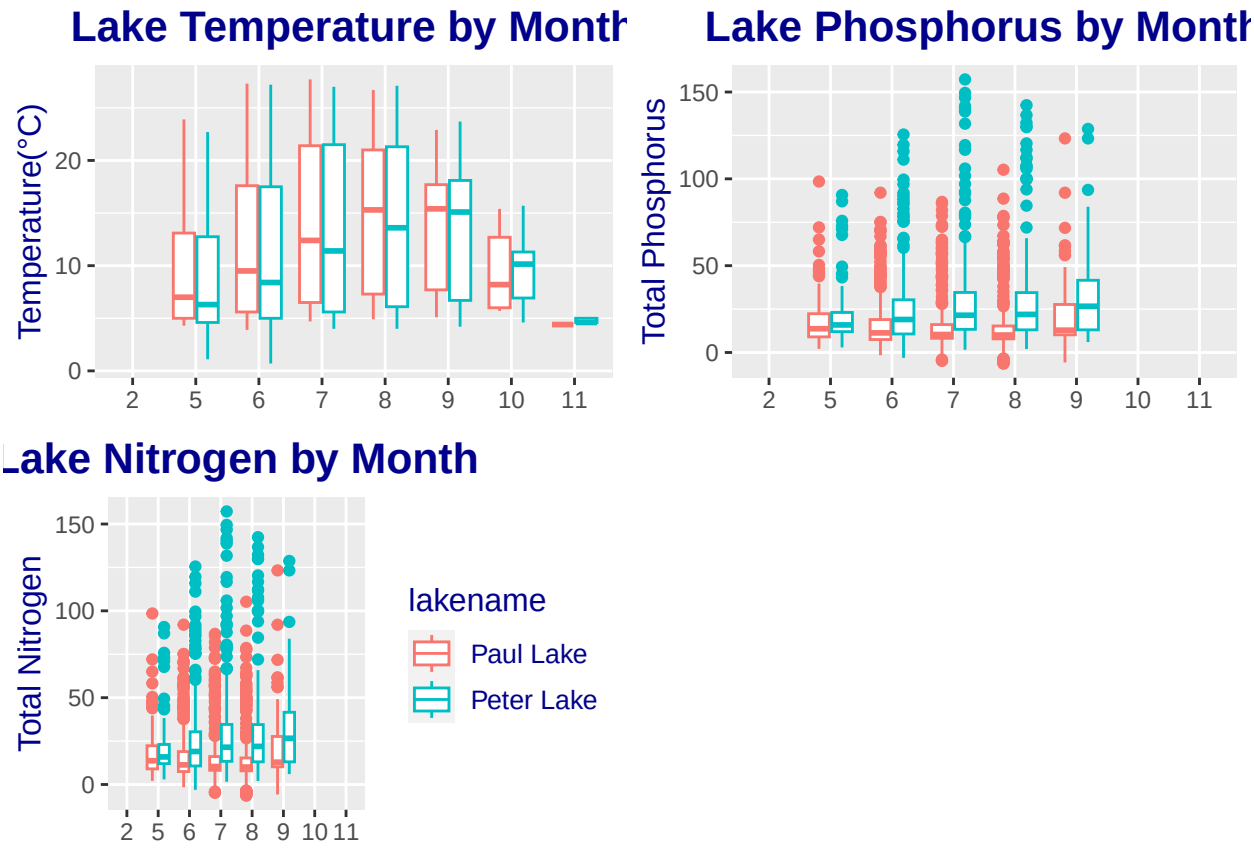
```
## Warning: Removed 3566 rows containing non-finite values (`stat_boxplot()`).
```

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

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

```
# Both horizontal and vertical alignment could not work...

# Print the combined plot
print(combined_boxes)
```




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

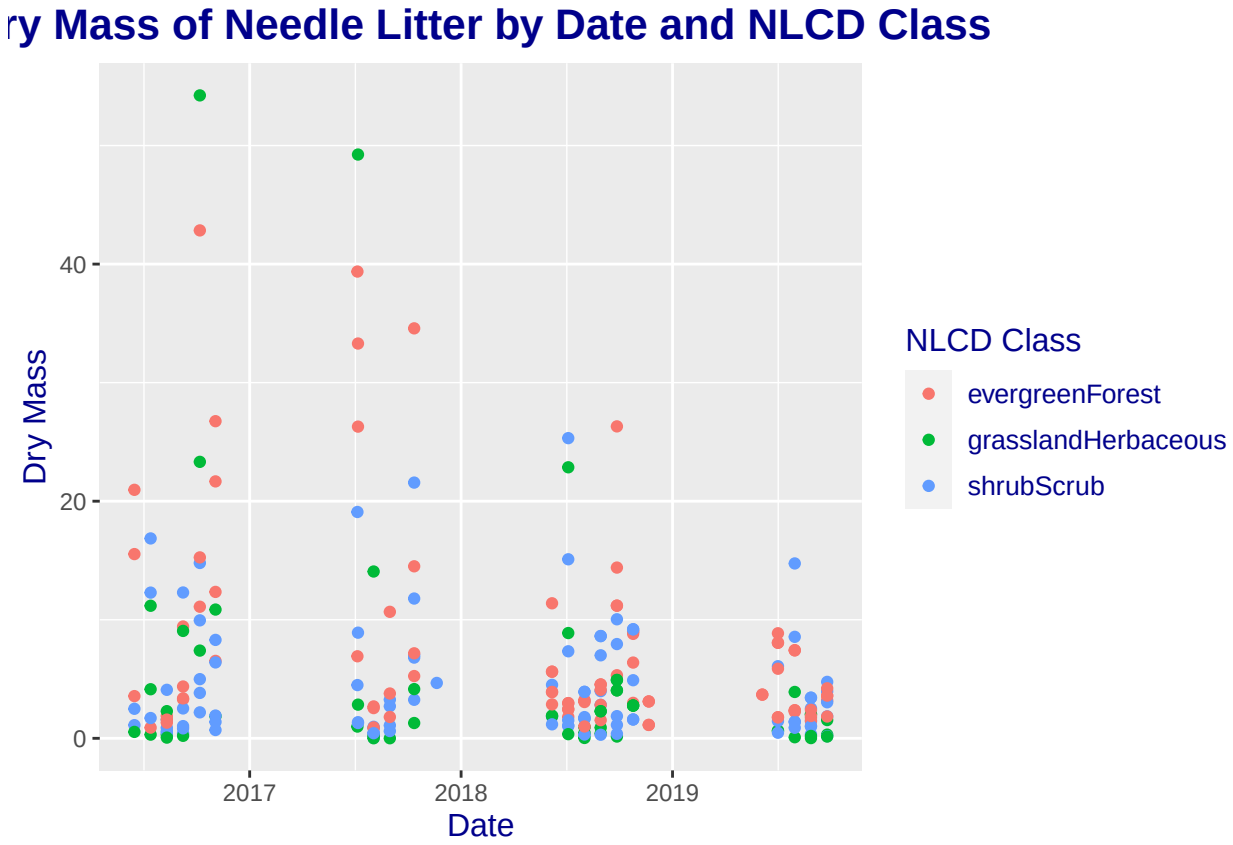
> Answer: Both of Peter Lake and Pual Lake have the higher temperatures in summer and lower ones in spring and fall (no observations in winter). When the temperature goes up, the amount of Phophorus and Nitrogen generally decline for Pual Lake, while it is the opposite situation for Peter Lake.

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.

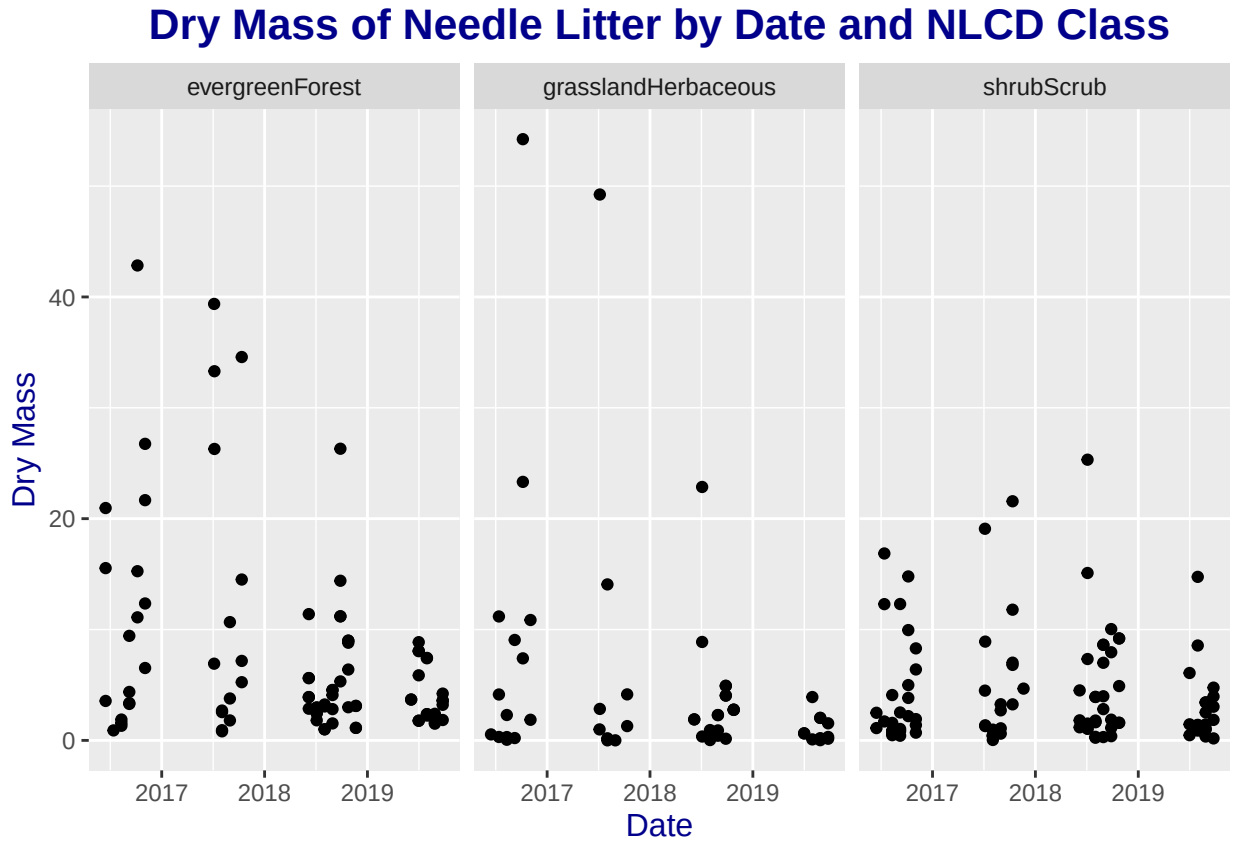
```
#6
Needles_DryMass <-
  ggplot(NiwotRidgeLitter[NiwotRidgeLitter$functionalGroup == "Needles",],
         aes(x = collectDate, y = dryMass, color = nlcdClass)) +
  geom_point() +
  labs(title = "Dry Mass of Needle Litter by Date and NLCD Class",
       x = "Date", y = "Dry Mass",
       color = "NLCD Class") +
  ivy_theme

print(Needles_DryMass)
```




```
#7
Needles_DryMass_Facet <-
  ggplot(NiwotRidgeLitter[NiwotRidgeLitter$functionalGroup == "Needles",],
         aes(x = collectDate, y = dryMass)) +
  geom_point() +
  facet_wrap(vars(nlcdClass)) +
  labs(title = "Dry Mass of Needle Litter by Date and NLCD Class",
       x = "Date", y = "Dry Mass") +
  ivy_theme

print(Needles_DryMass_Facet)
```




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

Answer: I think it really depends. If I want to have a more straight comparison among different NLCD classed, I would choose the plot 6 (Needles_DryMass). If I want to have a better understanding of each NLCD class and their respective trend, I would choose the plot 7 (Needles_DryMass_Facet).