Assignment 5: Data Visualization

Britney Pepper

OVERVIEW

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

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Work through the steps, **creating code and output** that fulfill each instruction.
3. Be sure to **answer the questions** in this assignment document.
4. When you have completed the assignment, **Knit** the text and code into a single PDF file.
5. After Knitting, submit the completed exercise (PDF file) to the dropbox in Sakai. Add your last name into the file name (e.g., "Fay_A05_DataVisualization.Rmd") prior to submission.

The completed exercise is due on Monday, February 14 at 7:00 pm.

##My Set Up

```
#pov: me hoping this works because nothing else does
#install.packages("formatR")
library(formatR)
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=60), tidy=TRUE, echo=TRUE)
```

Set up your session

1. Set up your session. Verify your working directory and load the tidyverse and 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_PeterPaul_Processed.csv`] version) and the processed data file for the Niwot Ridge litter dataset (use the [`NEON_NIWO_Litter_mass_trap_Processed.csv`] version).

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

```
# 1 setwd
getwd()
```

```
## [1] "/Users/britneypepper/Desktop/ENVIRON 872 and L/GitHubRepositories/Environmental_Data_Analytics_2022
```

```
setwd("/Users/britneypepper/Desktop/ENVIRON 872 and L/GitHubRepositories/Environmental_Data_Analytics_2022"

# install packages install.packages('tidyverse')
# install.packages('cowplot')

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(cowplot)

# load the data
peterpaullakes <- read.csv("/Users/britneypepper/Desktop/ENVIRON 872 and L/GitHubRepositories/Environmental
    stringsAsFactors = TRUE)
niwotridegelitter <- read.csv("/Users/britneypepper/Desktop/ENVIRON 872 and L/GitHubRepositories/Environmen
    stringsAsFactors = TRUE)


# 2 both set to characters instead of date == need to
# change
peterpaullakes$sampledate <- as.Date(peterpaullakes$sampledate,
    format = "%Y-%m-%d")
niwotridegelitter$collectDate <- as.Date(niwotridegelitter$collectDate,
    format = "%Y-%m-%d")
```

Define your theme

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

```
# 3
mytheme <- theme_classic(base_size = 14) + theme(axis.text = element_text(color = "black"))
theme_set(mytheme)
```

Create graphs

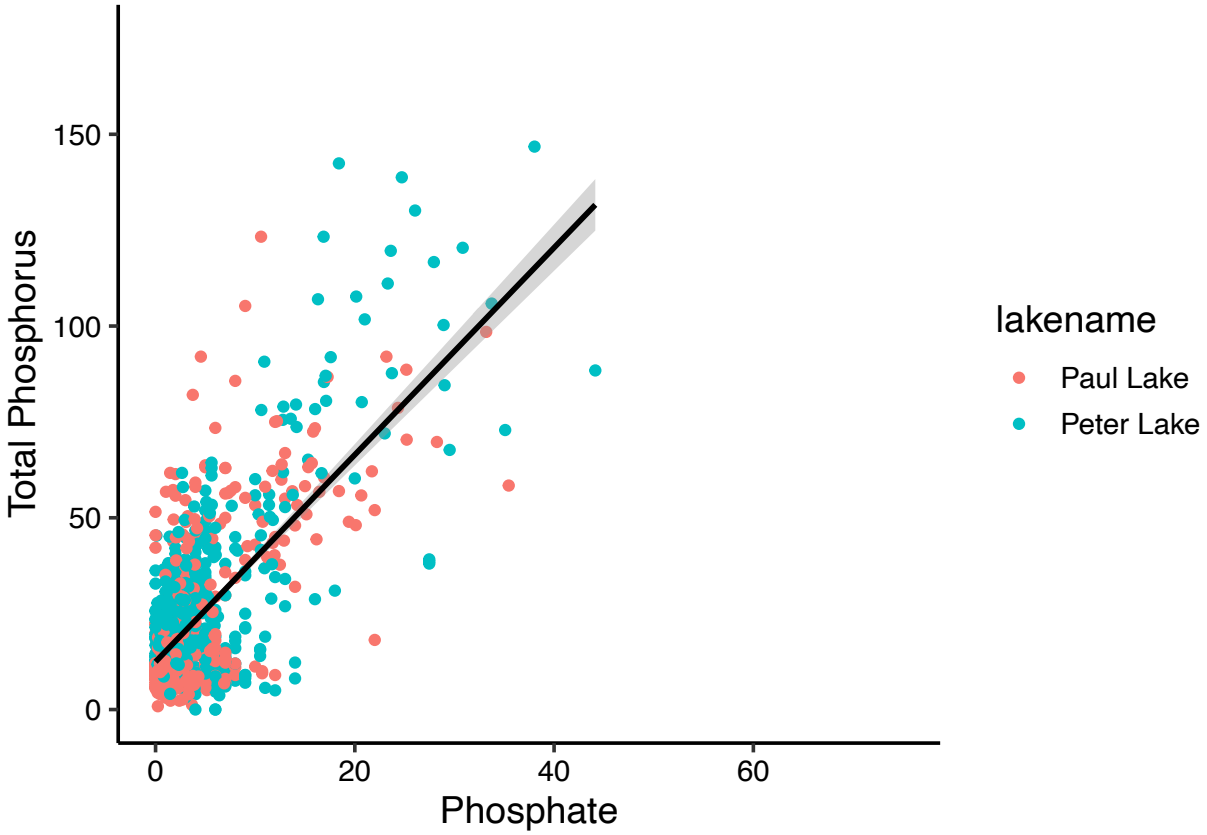
For numbers 4-7, create ggplot graphs and adjust aesthetics to follow best practices for data visualization. Ensure your theme, color palettes, axes, and additional aesthetics are edited accordingly.

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 `ylim()`).

```
# 4
phosphoorusVSphosphate <- ggplot(peterpaullakes, aes(x = po4,
    y = tp_ug)) + geom_point(aes(color = lakename)) + ylab("Total Phosphorus") +
    xlab("Phosphate") + xlim(0, 75) + ylim(0, 175) + geom_smooth(method = lm,
    color = "black")
print(phosphoorusVSphosphate)
```

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

## Warning: Removed 21948 rows containing non-finite values (stat_smooth).
```

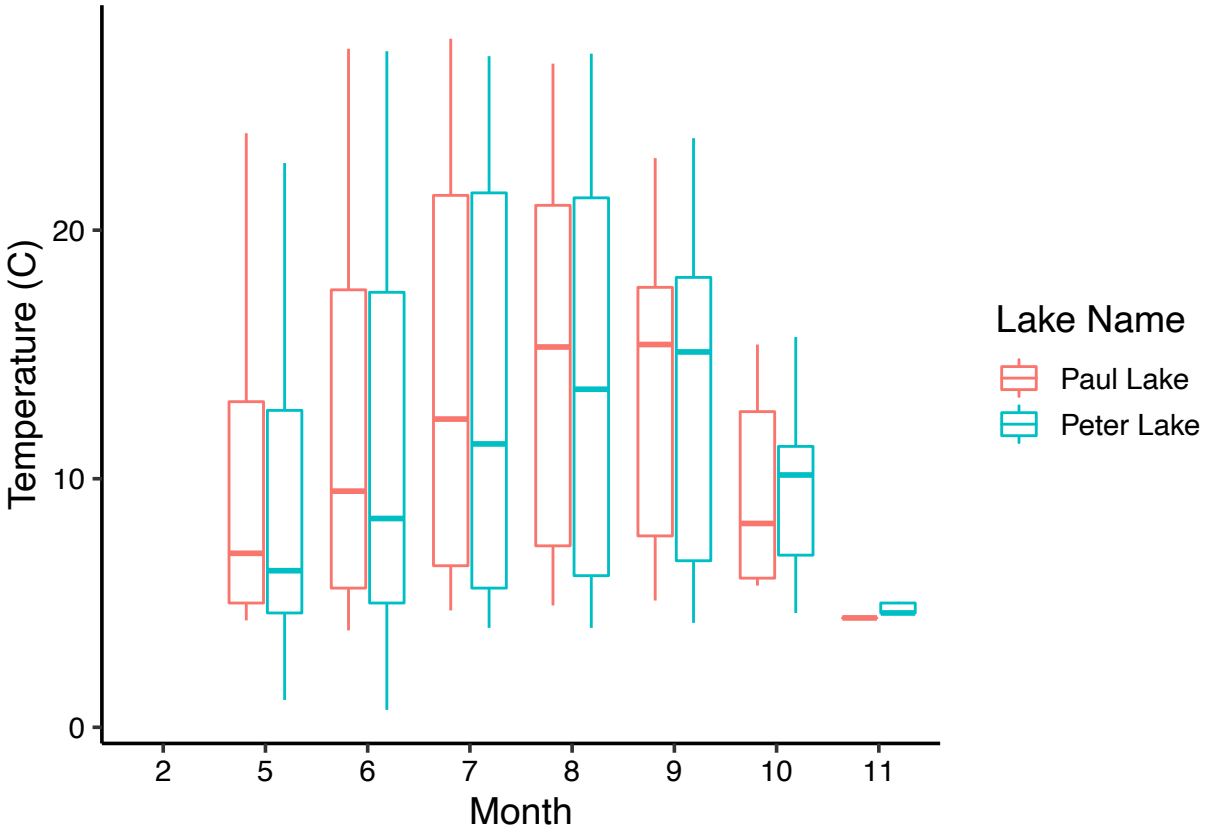
```
## Warning: Removed 21948 rows containing missing values (geom_point).
```



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.

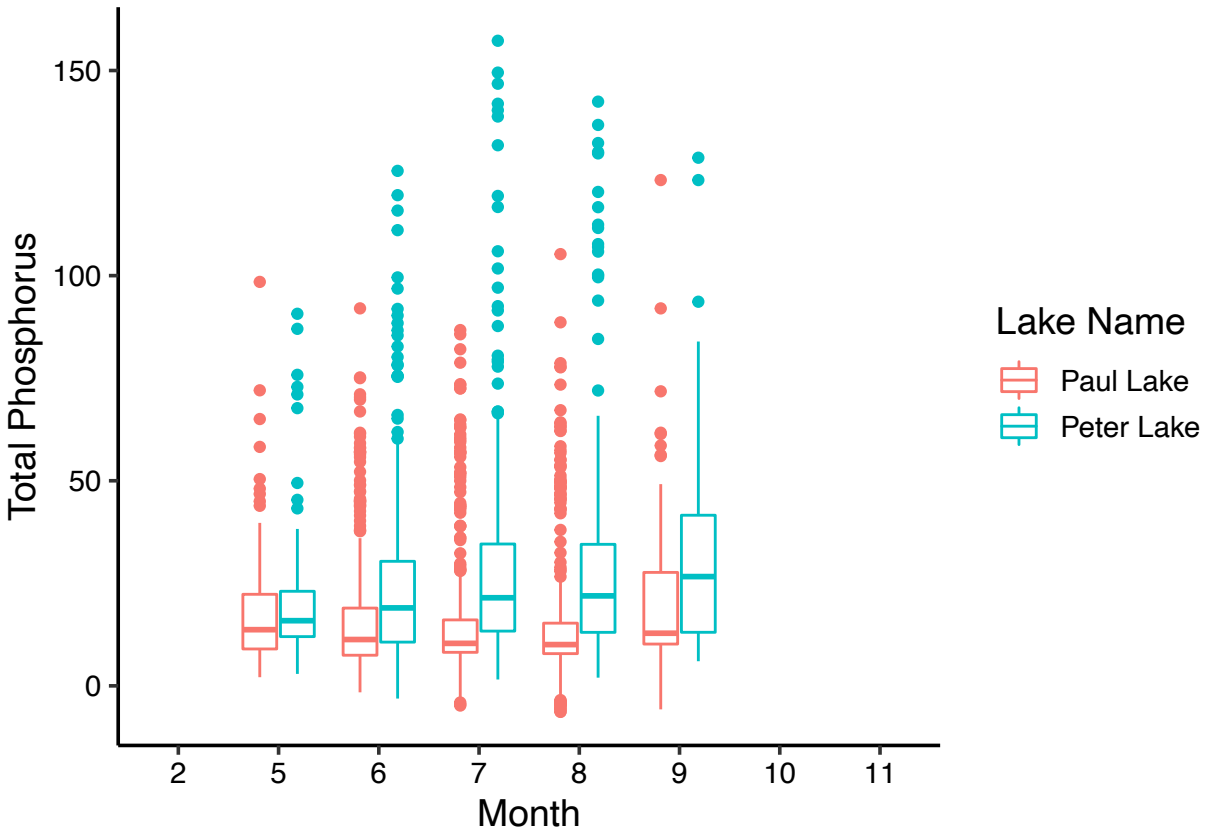
```
# 5

peterpaullakes$month <- as.factor(peterpaullakes$month)

# boxplot: temp
temp <- ggplot(peterpaullakes, aes(x = month, y = temperature_C)) +
    geom_boxplot(aes(color = lakename)) + xlab("Month") + ylab("Temperature (C)") +
    labs(color = "Lake Name")
print(temp)
```

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



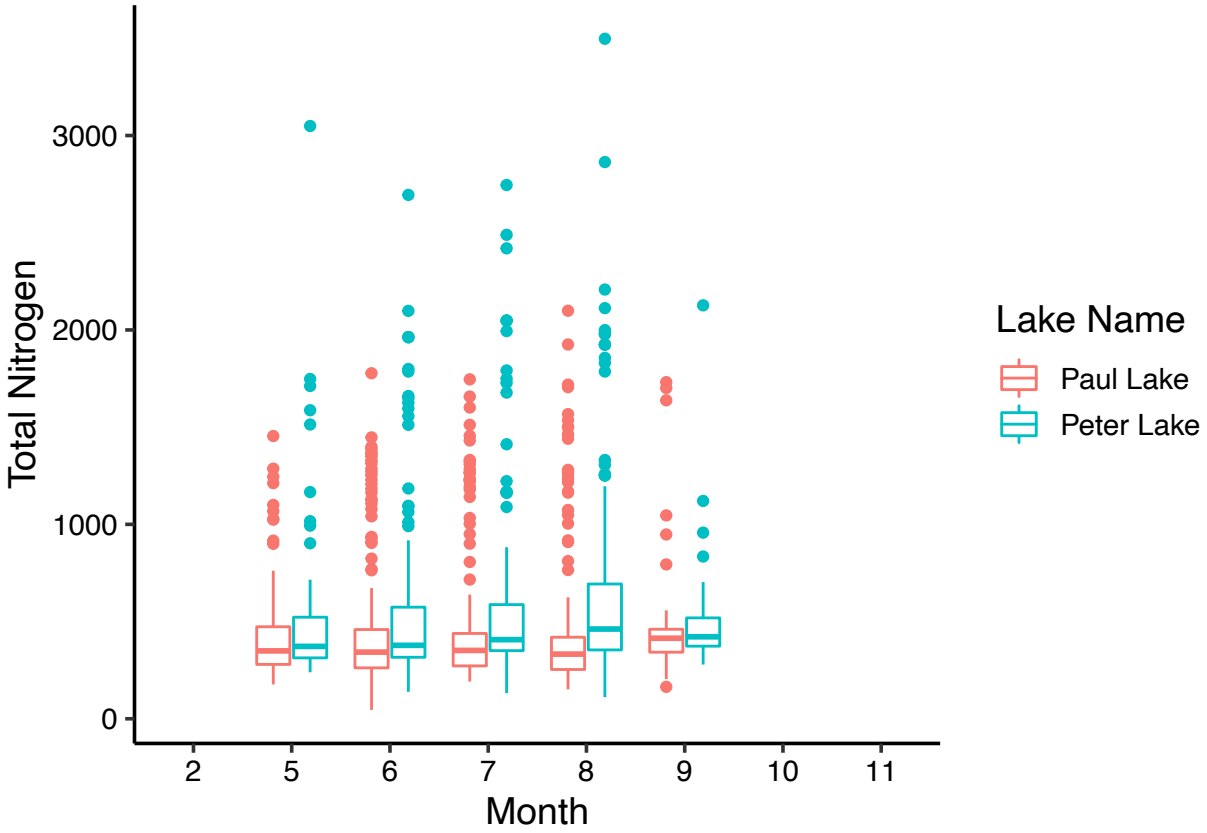

```
# boxplot: TP
TP <- ggplot(peterpaullakes, aes(x = month, y = tp_ug)) + geom_boxplot(aes(color = lakename)) +
    xlab("Month") + ylab("Total Phosphorus") + labs(color = "Lake Name")
print(TP)
```

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




```
# boxplot: TN
TN <- ggplot(peterpaullakes, aes(x = month, y = tn_ug)) + geom_boxplot(aes(color = lakename)) +
    xlab("Month") + ylab("Total Nitrogen") + labs(color = "Lake Name")
print(TN)
```

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



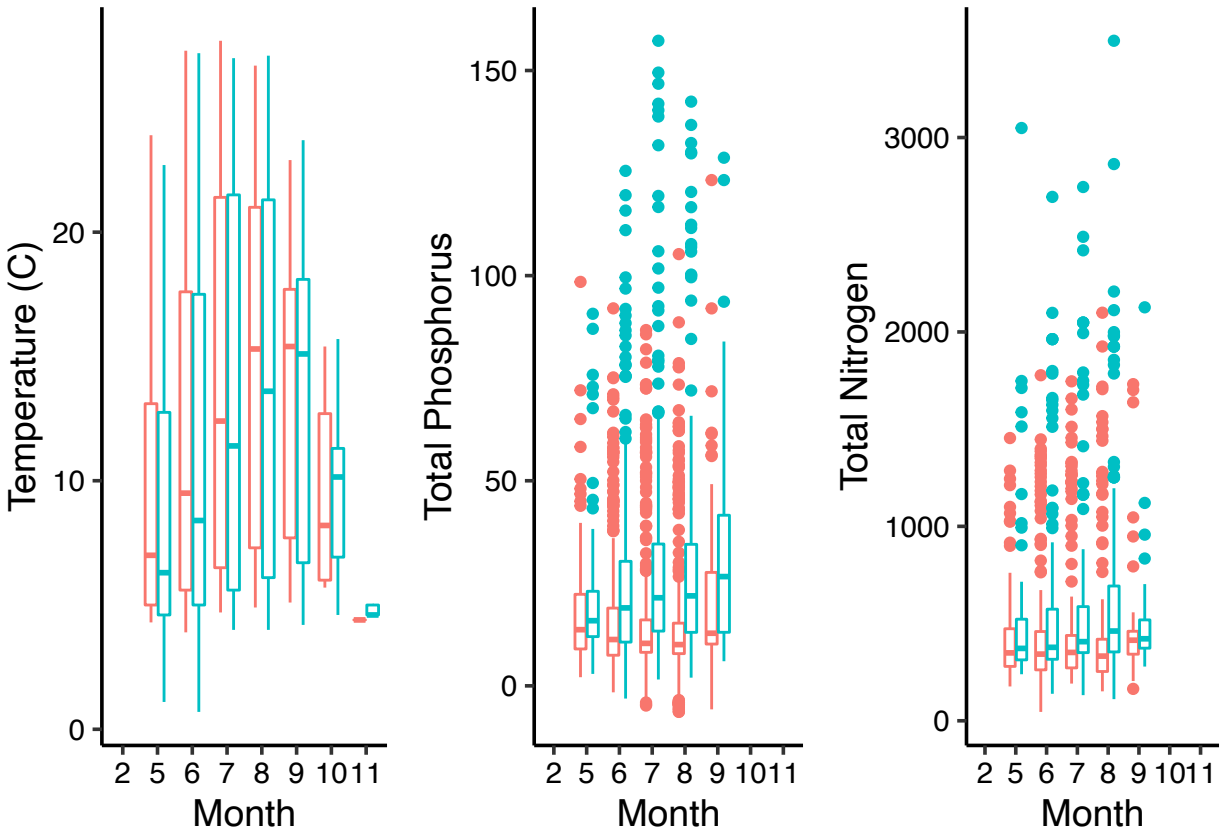

```
# cowplot save three plots as object
all_plots <- plot_grid(temp + theme(legend.position = "none"),
    TP + theme(legend.position = "none"), TN + theme(legend.position = "none"),
    nrow = 1)
```

```
## 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).
```

```
print(all_plots)
```



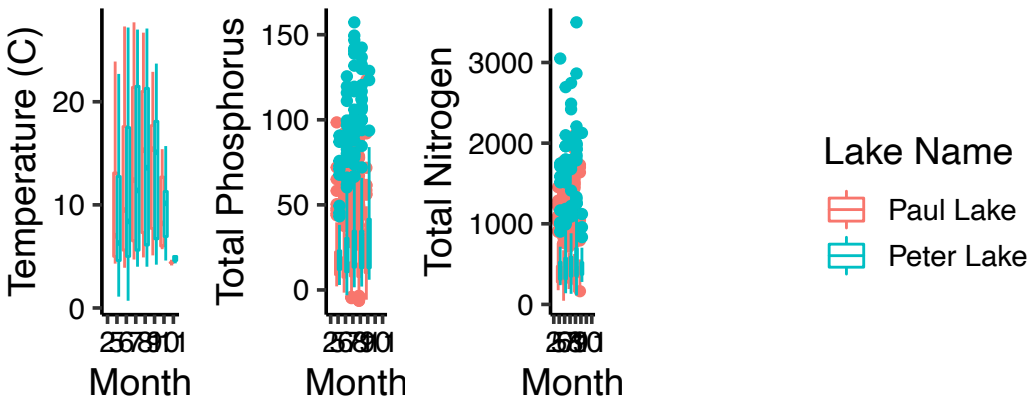

```
# create another object just for legend
cowplot_legend <- get_legend(temp)
```

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

```
# use plot grid again with two rows one for object with
# three plots and one for the object with the legend
plot_all_grid <- plot_grid(all_plots, cowplot_legend, nrow = 2,
    rel_height = c(1, 0.1))
```

```
## Warning in as_grob.default(plot): Cannot convert object of class numeric into a
## grob.
```

```
print(plot_all_grid)
```



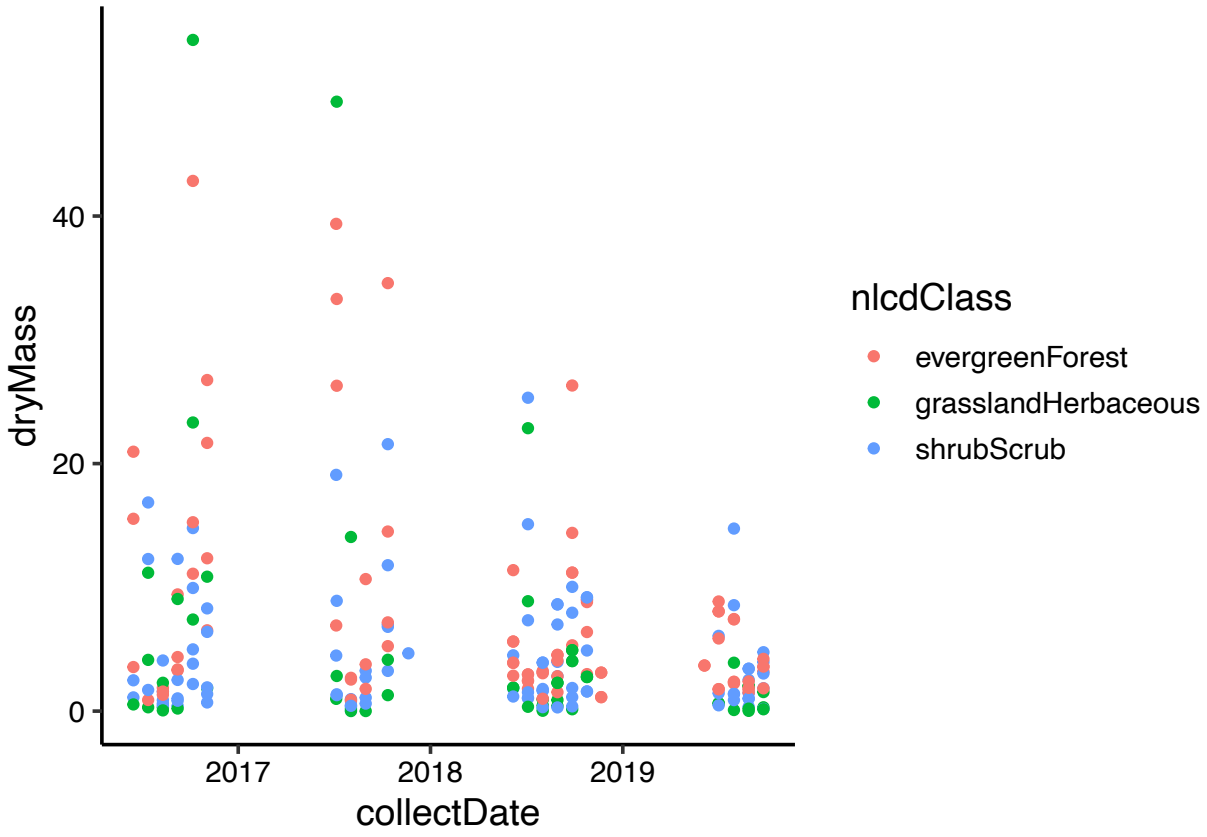
```
# Note: I tried everything to get the legend to print below
# the graphs, but I could not get it to :(
```

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

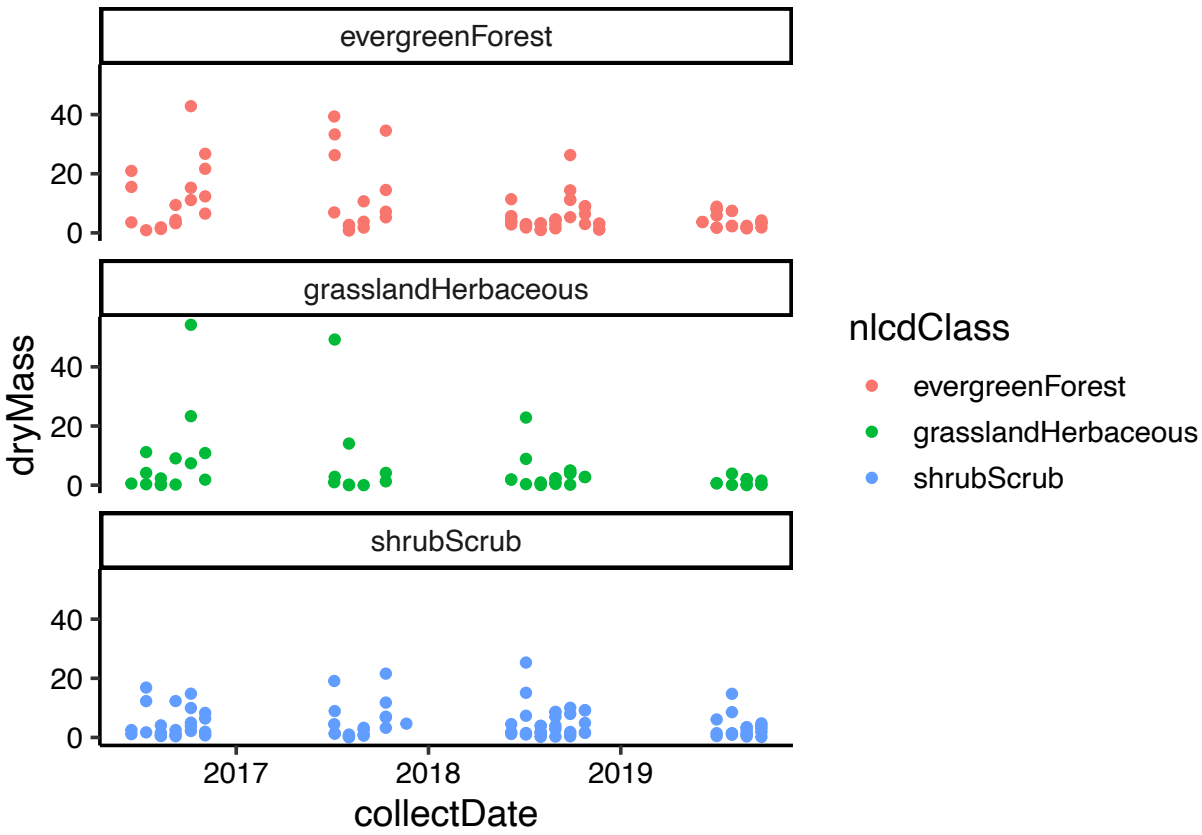
> Answer: The average temperature was consistently higher for Paul Lake than Peter Lake until the late fall/winter months. The temperatures also had more variability in the summer months with a wider range. The average total amount of phosphorus was consistenly greater for Peter Lake and Paul Lake, and the concentration seems to increase in range in July. The average total nitrogen content almost always greater than in Peter Lake than in Paul Lake. The range seemed to increase in the summer months, although the range for Peter Lake appeared to be much larger.

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
questionSix <- ggplot(subset(niwotridegelitter, functionalGroup ==
    "Needles"), aes(x = collectDate, y = dryMass)) + geom_point(aes(color = nlcdClass))
print(questionSix)
```




```
# 7
questionSeven <- ggplot(subset(niwotridegelitter, functionalGroup ==
    "Needles"), aes(x = collectDate, y = dryMass)) + geom_point(aes(color = nlcdClass)) +
    facet_wrap(vars(nlcdClass), nrow = 3)
print(questionSeven)
```



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

Answer: I think plot 7 was more effective. It separated the the NLCD classes into three separate facets, which allowed you to compare each individual facet across the years. It was much easier to analyze the data because I was not trying to see trends in the data by just using color differences on one graph like in plot 6.