Assignment 5: Data Visualization

Jazmine Pritchett

Fall 2023

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.

```r
#1

#project set up by install of packages and importing of data.

library(tidyverse)
```

```
## -- 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 1.9.2     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 error
```

```
library(lubridate)
library (cowplot)
```

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

```
library(here)
```

```
## here() starts at /home/guest/EDE_Fall2023
```

```
getwd()
```

```
## [1] "/home/guest/EDE_Fall2023"
```

```
PeterPaul.NTL <- read.csv(
  file=here("Data/Processed_KEY/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv"),
  stringsAsFactors = TRUE)

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


#2

#Dates were read as factors.

class(PeterPaul.NTL$sampledate)
```

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

```
PeterPaul.NTL$sampledate <- ymd(PeterPaul.NTL$sampledate)

class(NR.Litter$collectDate)
```

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

```
NR.Litter$collectDate <- ymd(NR.Litter$collectDate)
```

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

#Set a theme that aesthetically pleasing and that doesn't take away
#from data values.

library(ggthemes)
```

```
##
## Attaching package: 'ggthemes'

## The following object is masked from 'package:cowplot':
##
##     theme_map
```

```
my_theme <- theme_base() + theme (legend.background = element_rect(
      color='black',
      fill = 'darkgrey'),
    axis.ticks = element_line(
      color='darkgrey', linewidth = 0.5))
```

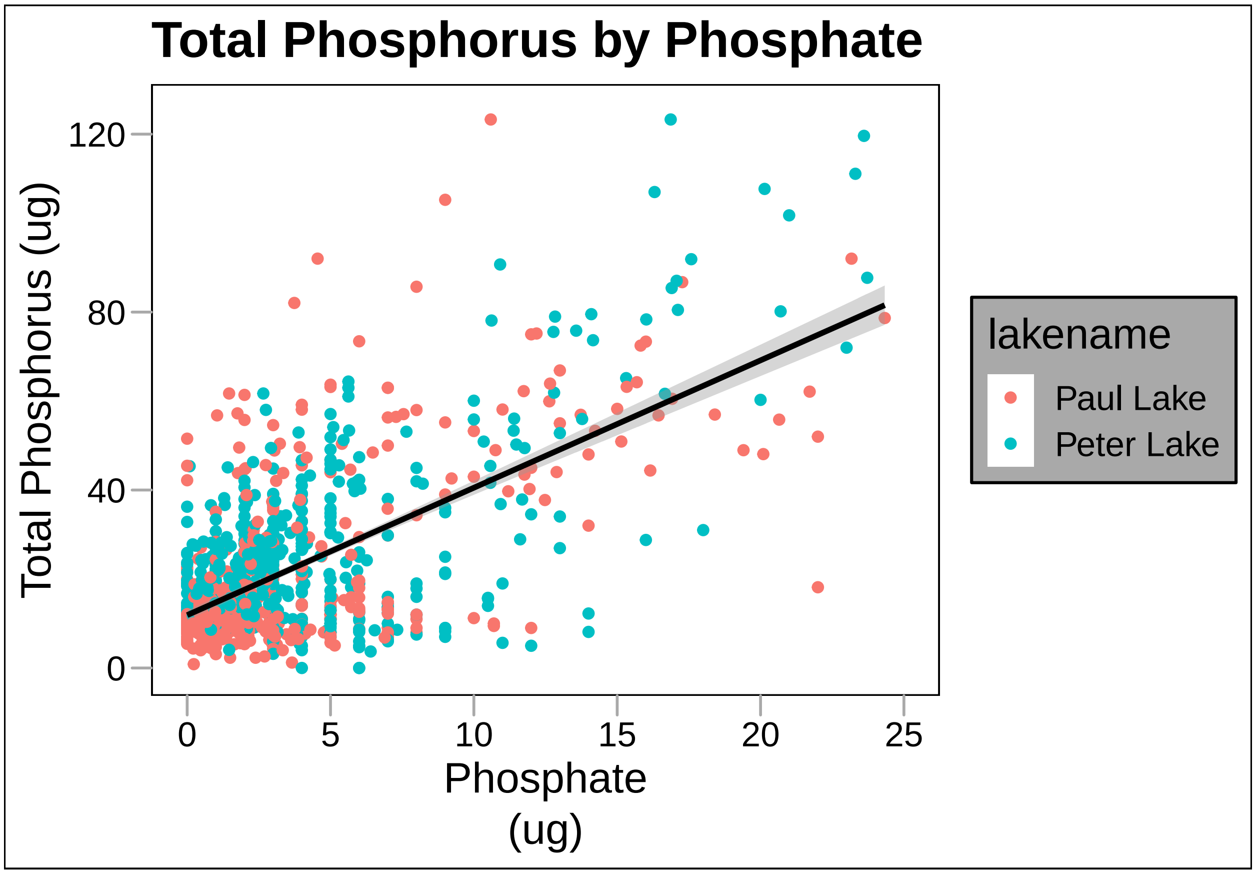
Create graphs

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

#Creation of graph with edits to title for aesthetics.

NTLplot <- ggplot(PeterPaul.NTL, aes(x = po4, y = tp_ug)) +
my_theme + geom_point(aes(color = lakename), na.rm = TRUE) +
xlim(0,25) + ylim(0, 125)+
geom_smooth(color = 'black', method=lm) + labs(x= "Phosphate
(ug)", y= "Total Phosphorus (ug)", title = "Total Phosphorus by Phosphate")
print(NTLplot)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 21968 rows containing non-finite values (‘stat_smooth()‘).
```



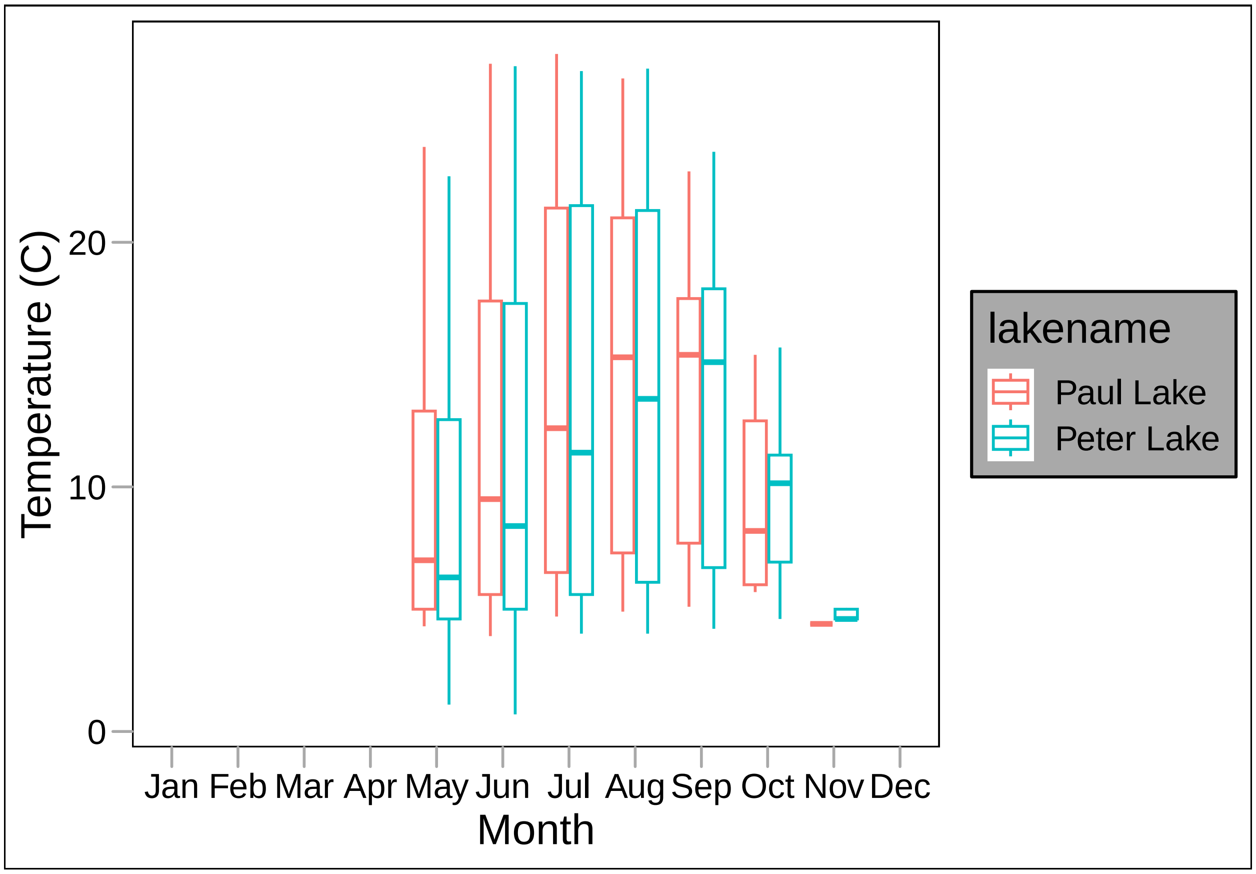
```
theme_set(my_theme)
```

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: * Recall the discussion on factors in the previous section as it may be helpful here. * R has a built-in variable called `month.abb` that returns a list of months;see https://r-lang.com/month-abb-in-r-with-example

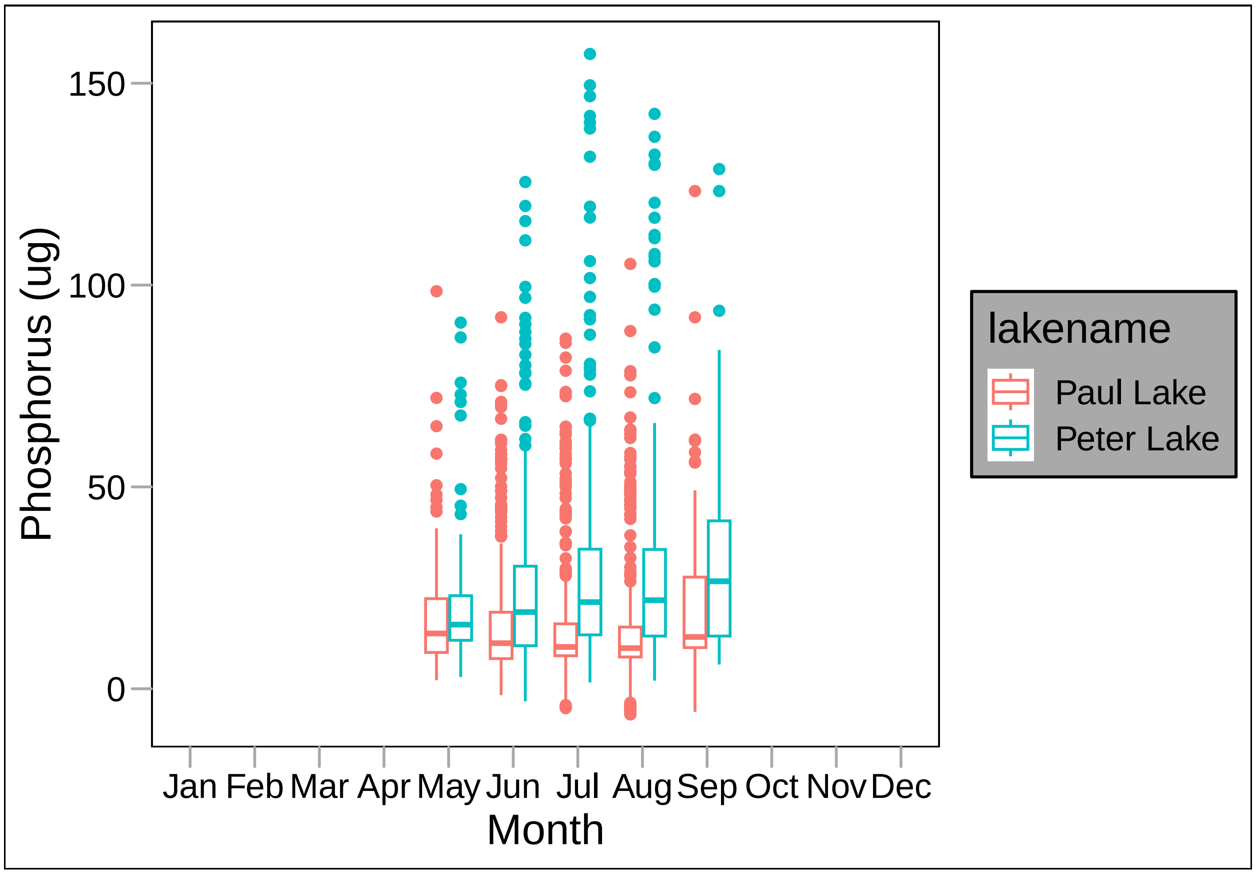
```
#5

TempPlot <- PeterPaul.NTL %>% ggplot(aes(y = temperature_C, x = factor(month, levels = 1:12,
labels=month.abb ))) +
scale_x_discrete(name="Month", drop =FALSE)+ geom_boxplot(aes(color = lakename)) + labs(x= "Month",
y= "Temperature (C)")
print(TempPlot)
```

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



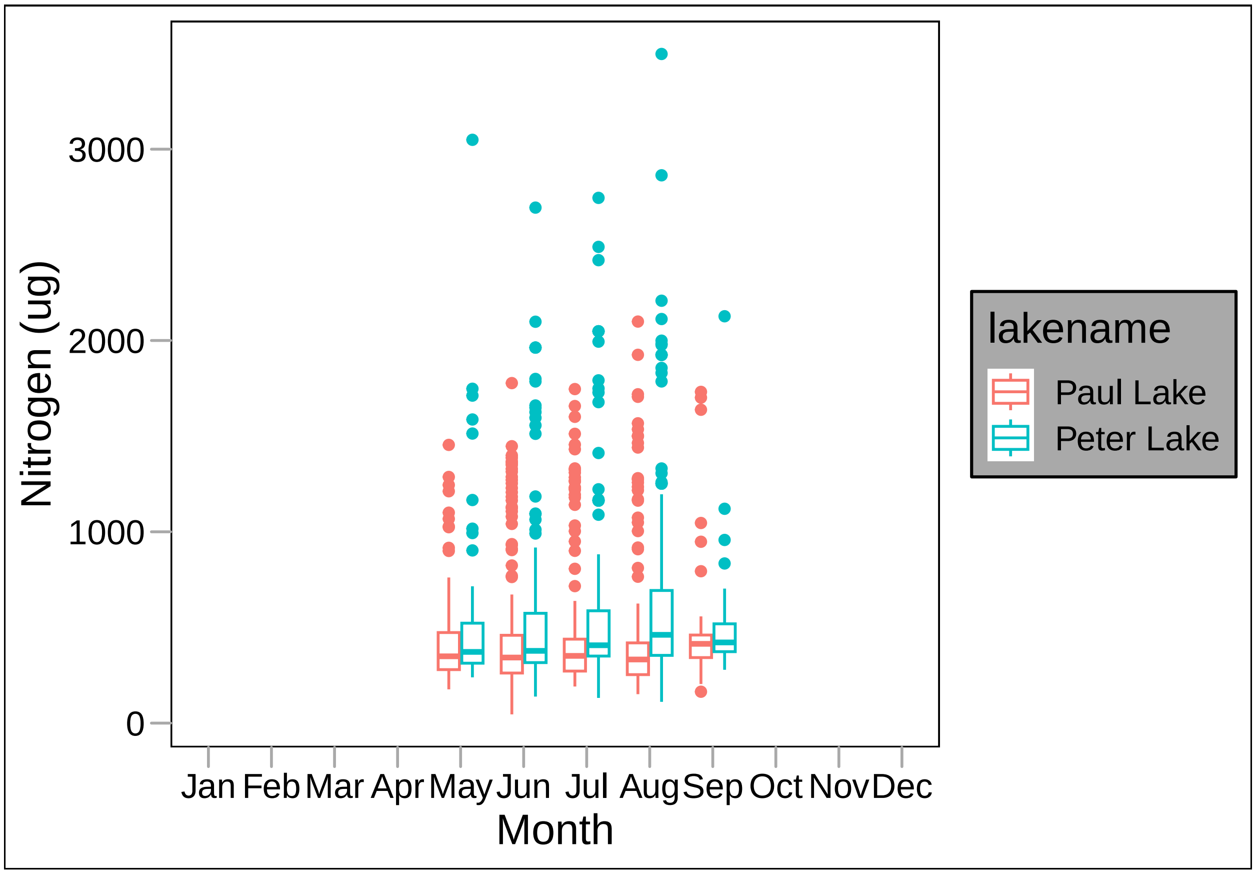
```
TPPlot <- PeterPaul.NTL %>% ggplot(aes(y = tp_ug, x = factor(month, levels = 1:12,
labels= month.abb
))) + scale_x_discrete (name="Month", drop =FALSE)+ geom_boxplot(aes(color = lakename))+
labs(x= "Month", y= "Phosphorus (ug)")
print(TPPlot)
```

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



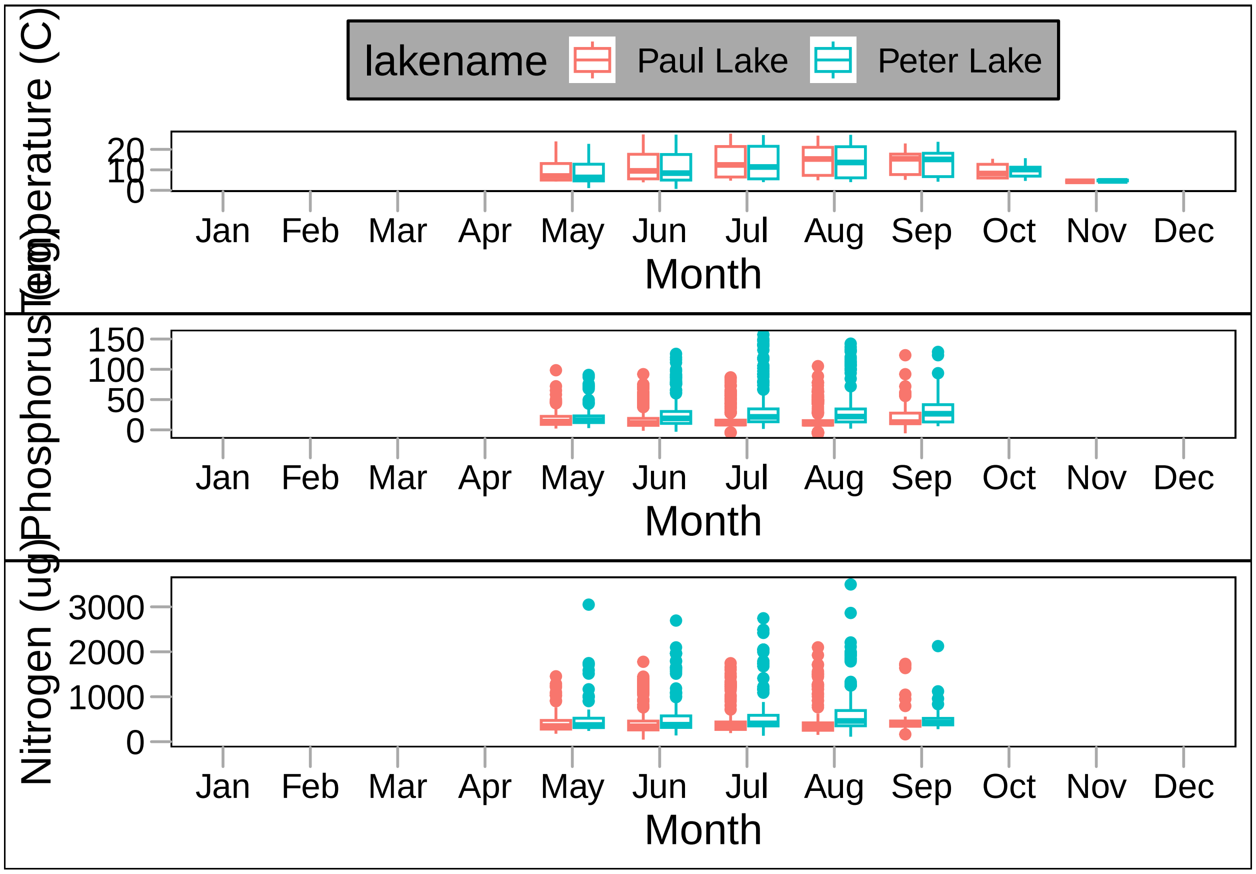
```r
TNPlot <-PeterPaul.NTL %>% ggplot(aes(y = tn_ug, x = factor(month, levels = 1:12, labels=month.abb
))) + scale_x_discrete(name="Month", drop =FALSE)+ geom_boxplot(aes(color = lakename))+
labs(x= "Month", y= "Nitrogen (ug)")
print(TNPlot)
```

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



```
plot_grid(TempPlot + theme(legend.position = "top"), #legend included at the top for aligned axis.
TPPlot + theme(legend.position="none"), TNPlot + theme(legend.position="none"),
nrow=3, align= 'v', #align = v aligned
#vertical axis plot when placing in the legend at the top.
rel_heights = c(1.25,1))
```

```
## 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()‘).
```

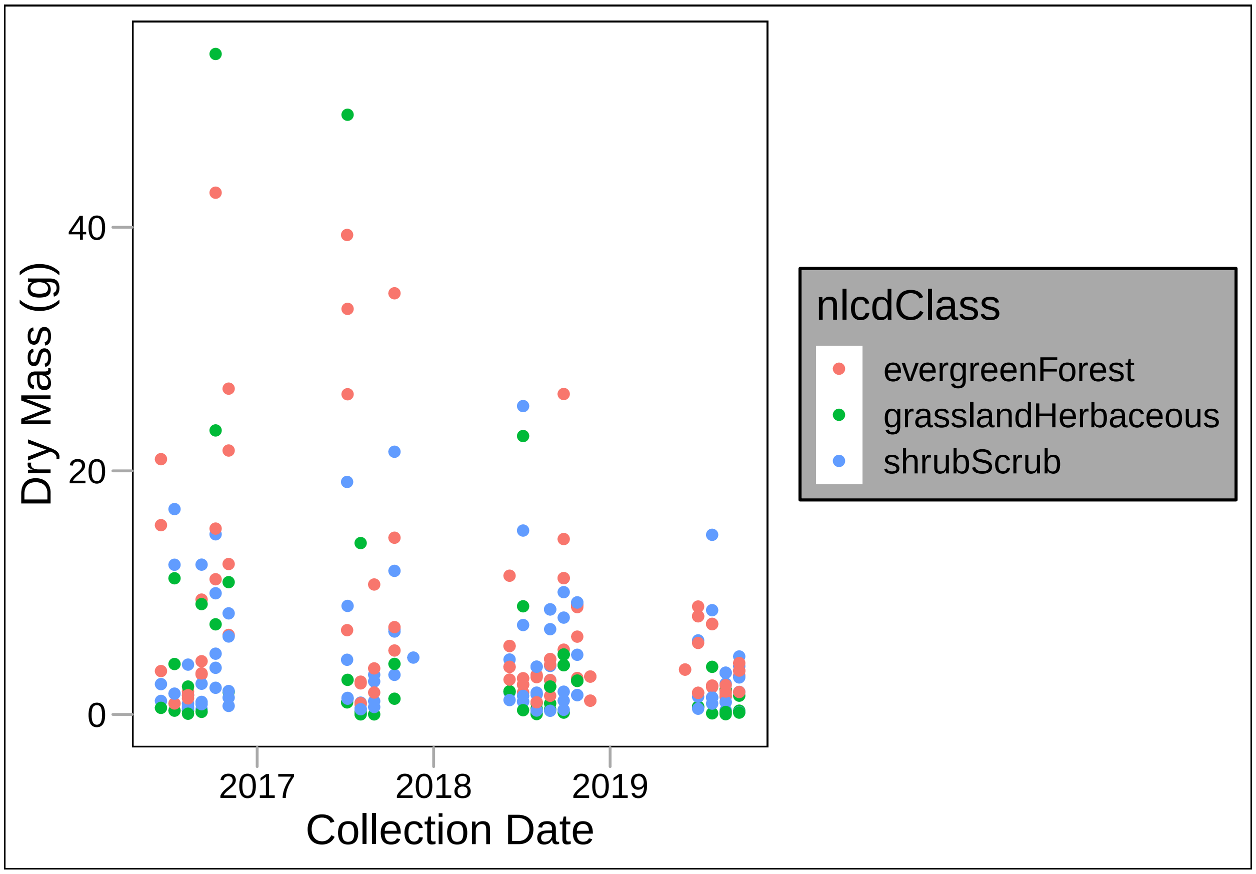


Question: What do you observe about the variables of interest over seasons and between lakes?

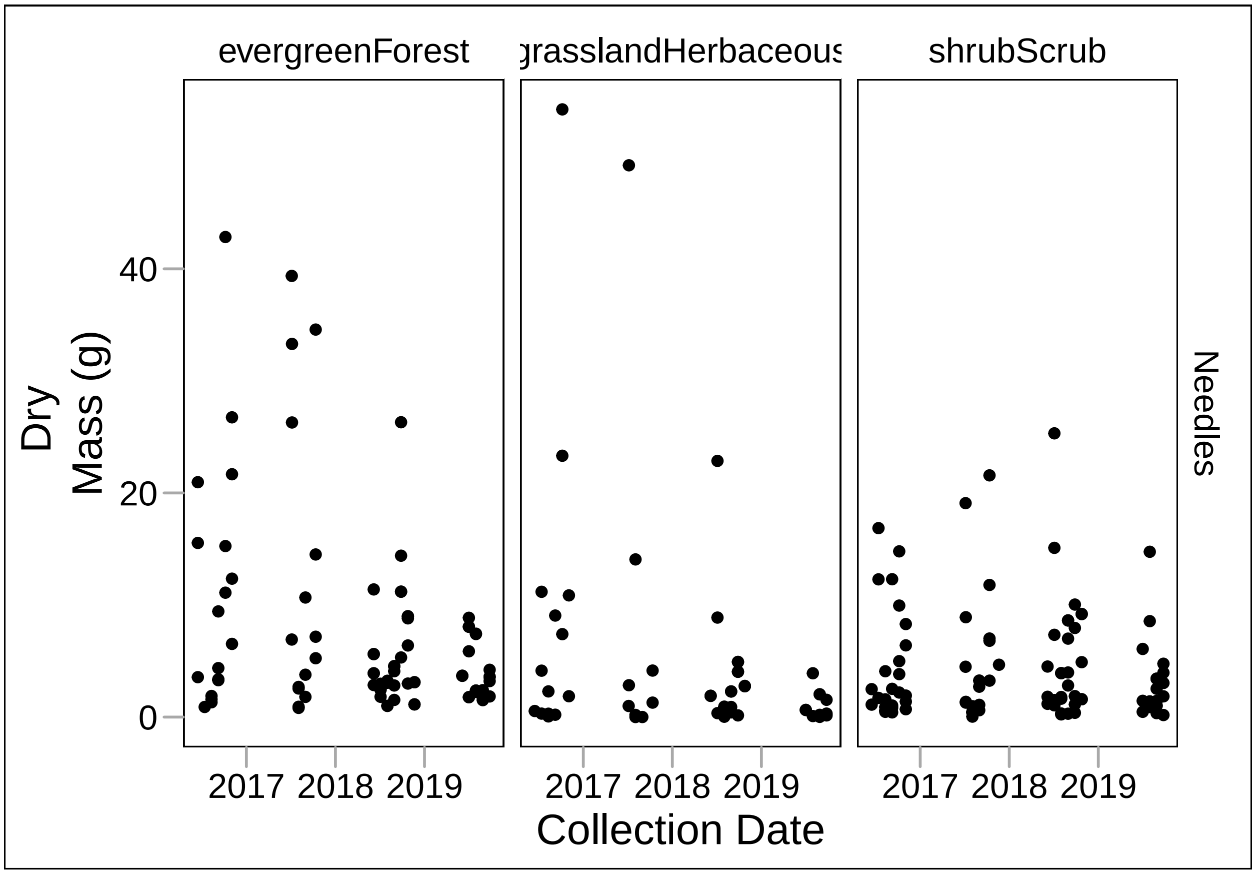
> Answer: Each lake is more prominent over warmer/summer months (May-Septemeber). There is also not much variance for each lake across variables and the months.

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.

```r
#6

Needles_Group <- ggplot(subset(NR.Litter, functionalGroup == "Needles"), aes(x = collectDate,
y = dryMass))+ geom_point(aes(color = nlcdClass))+ labs(x= "Collection Date", y= "Dry Mass (g)")
print(Needles_Group)
```



```
#7

Needles_Facet <- ggplot(subset(NR.Litter, functionalGroup == "Needles"), aes(x = collectDate,
y = dryMass))+ geom_point() + facet_wrap(vars(functionalGroup), nrow = 3) + labs(x= "Collection Date",
y= "Dry
Mass (g)") + facet_grid( functionalGroup ~ nlcdClass) #plotting Needles by facets of nlcd classes.
print(Needles_Facet)
```



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

Answer:While plot 6 is in color, values are overlapping. Because of this, I think plot 7 that includes facets helps the viewer easily compare each class of data.