Assignment 5: Data Visualization

Jonathan Joyner

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.

```
#1
#import packages
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 errors
```

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

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

```
library(ggplot2)
#set path for processed data files
getwd()
```

```
## [1] "C:/Users/jbjoy/OneDrive/Documents/Grad School/Fall 2023/Environ 872/EDE_Fall2023"
```

```
setwd(
  "C:/Users/jbjoy/OneDrive/Documents/Grad School/Fall 2023/Environ 872/EDE_Fall2023/Data/Processed_KEY")
getwd()
```

```
## [1] "C:/Users/jbjoy/OneDrive/Documents/Grad School/Fall 2023/Environ 872/EDE_Fall2023/Data/Processed_
```

```
#read processed data files
PeterPaul<-
  read.csv(
    "NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv",
    stringsAsFactors = TRUE)
NIWOT<-
  read.csv(
    "NEON_NIWO_Litter_mass_trap_Processed.csv",
    stringsAsFactors = TRUE)
#2
#make sure date columns are read as date factor
PeterPaul$sampledate<-ymd(PeterPaul$sampledate)
NIWOT$collectDate<-ymd(NIWOT$collectDate)
```

Define your theme

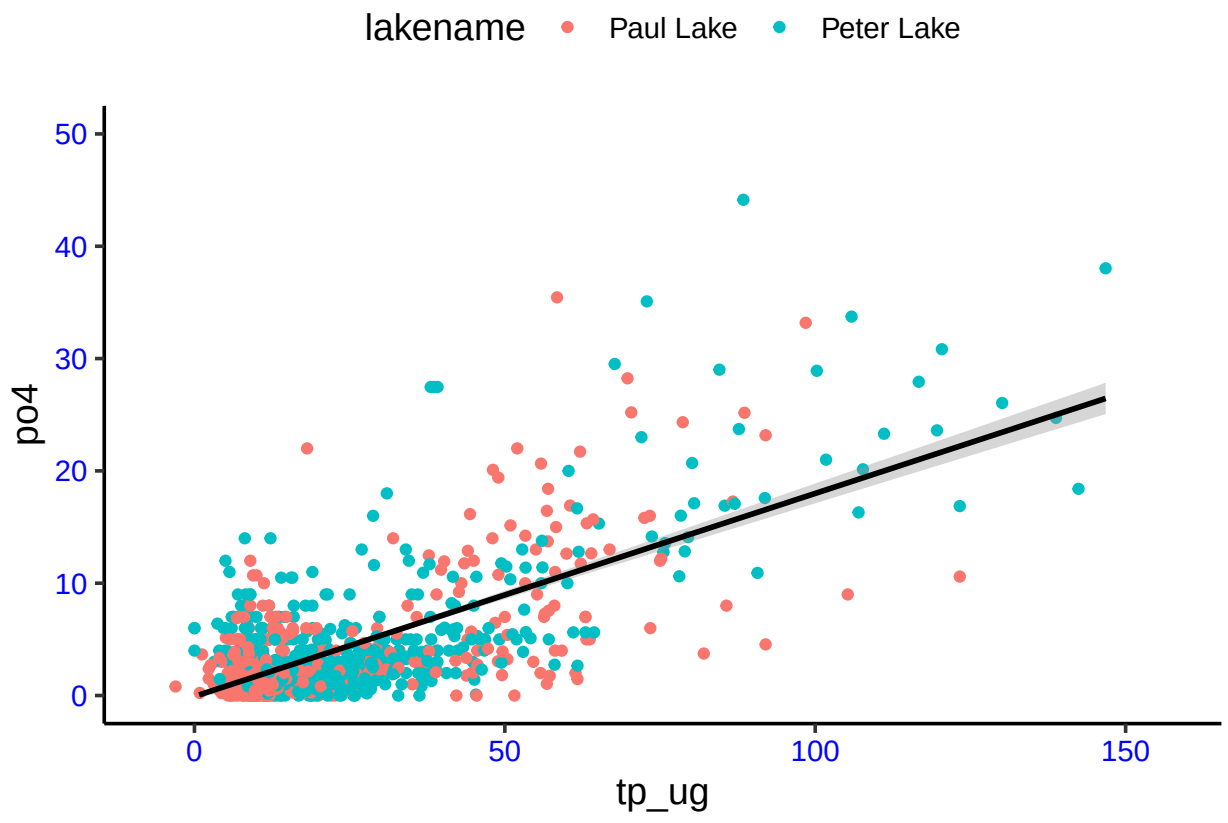
3. Build a theme and set it as your default theme. Customize the look of at least two of the following:

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

```
#3 adjust theme
d_theme<-theme_classic(base_size = 14) +
  theme(axis.text= element_text(color="blue"),
```

```
        legend.position="top",
        legend.justification ="center")
#set theme
theme_set(d_theme)

#4 Plot total by phosphate and set aesthetics with best fit line
#set vertical limit of 50
Plot4<-
  ggplot(PeterPaul,aes(x = tp_ug, y=po4,color=lakename))+
  geom_point()+
  geom_smooth(method=lm,color="black") +
  ylim(0,50)
print(Plot4)
```

```
## `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()`).

## Warning: Removed 2 rows containing missing values (`geom_smooth()`).
```

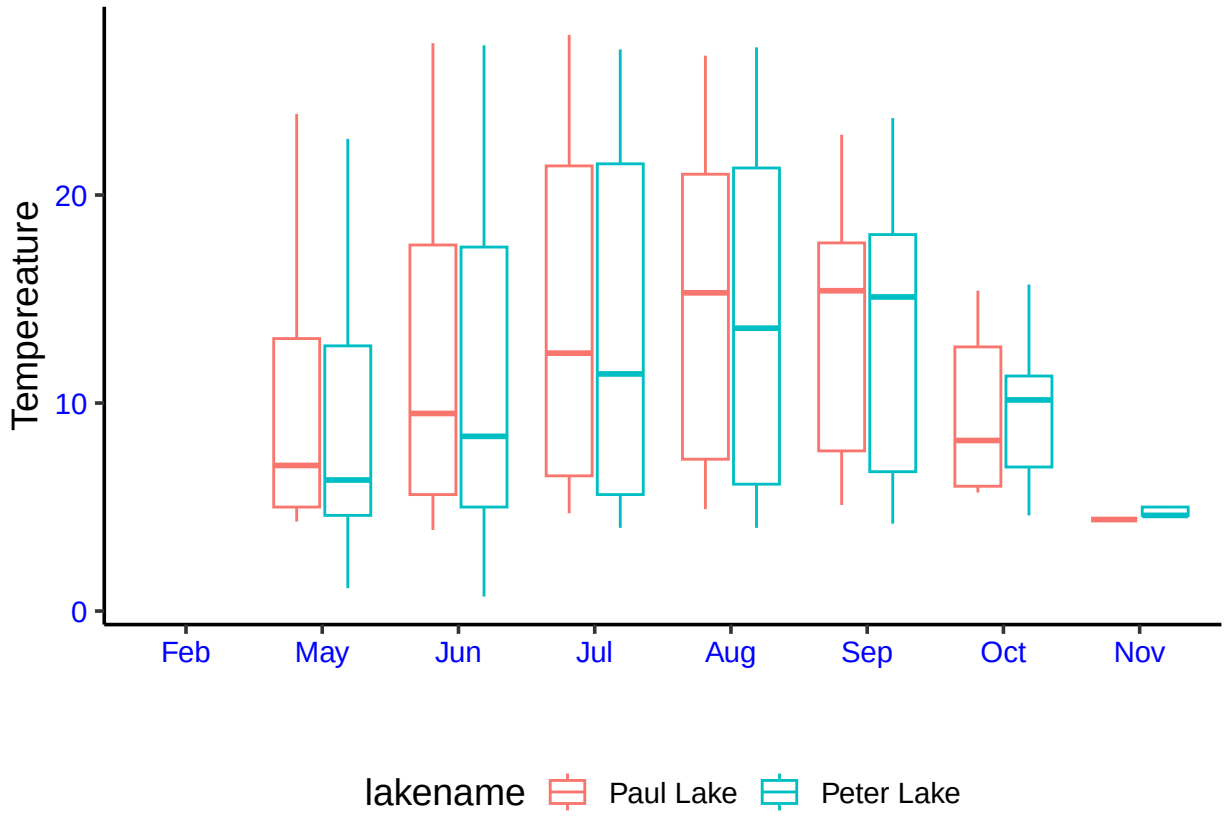


5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and (c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

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 Rename axis labels
#create boxplot by temperature
Plot5a<-
  ggplot(PeterPaul,
         aes(x=factor(month,levels=1:12,labels=month.abb),
             y = temperature_C,
             color=lakename))+
  geom_boxplot() +
  labs(y="Tempereature",x="") +
  theme(legend.position ="bottom")
print(Plot5a)
```

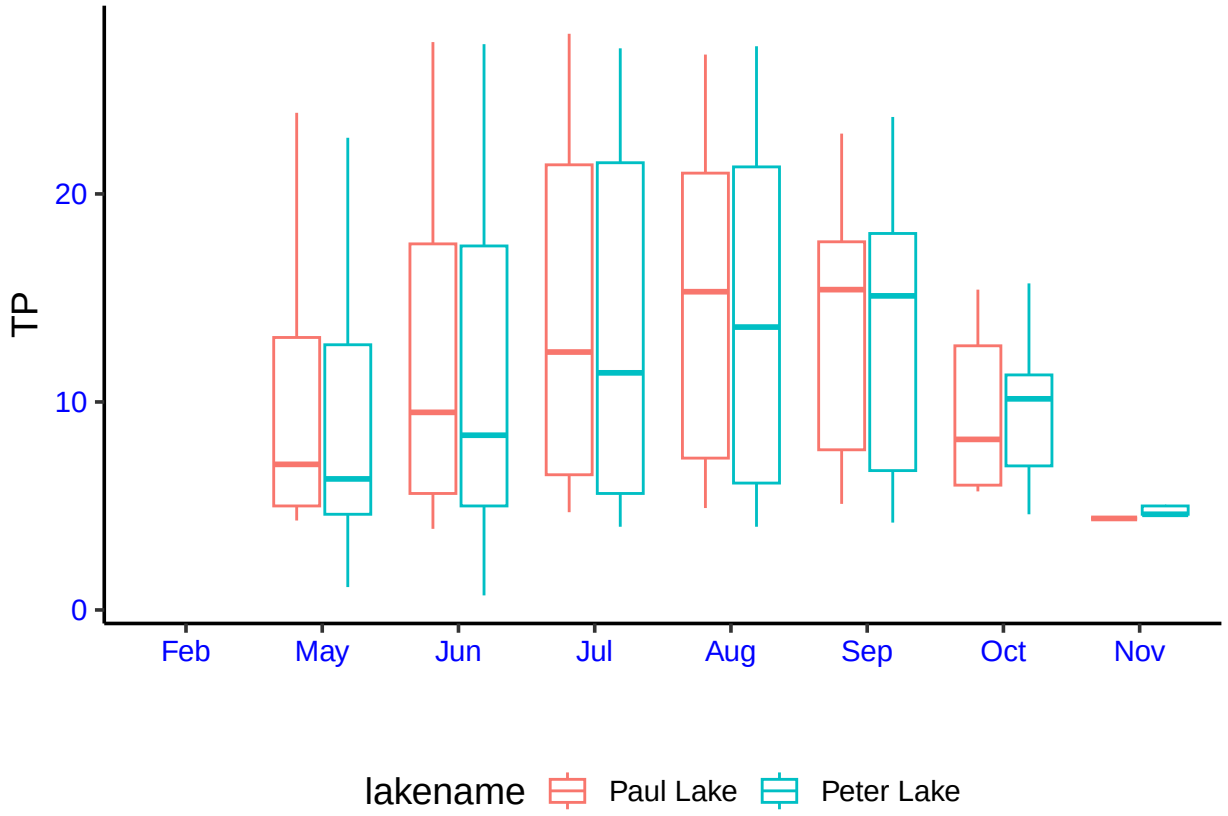
```
## Warning: Removed 3566 rows containing non-finite values (`stat_boxplot()`).
```




```
#create boxplot by TP
Plot5b<-
  ggplot(PeterPaul,
         aes(x = factor(month,levels=1:12,labels=month.abb),
             y = temperature_C,
             color=lakename))+
  geom_boxplot() +
  labs(y="TP",x="") +
```

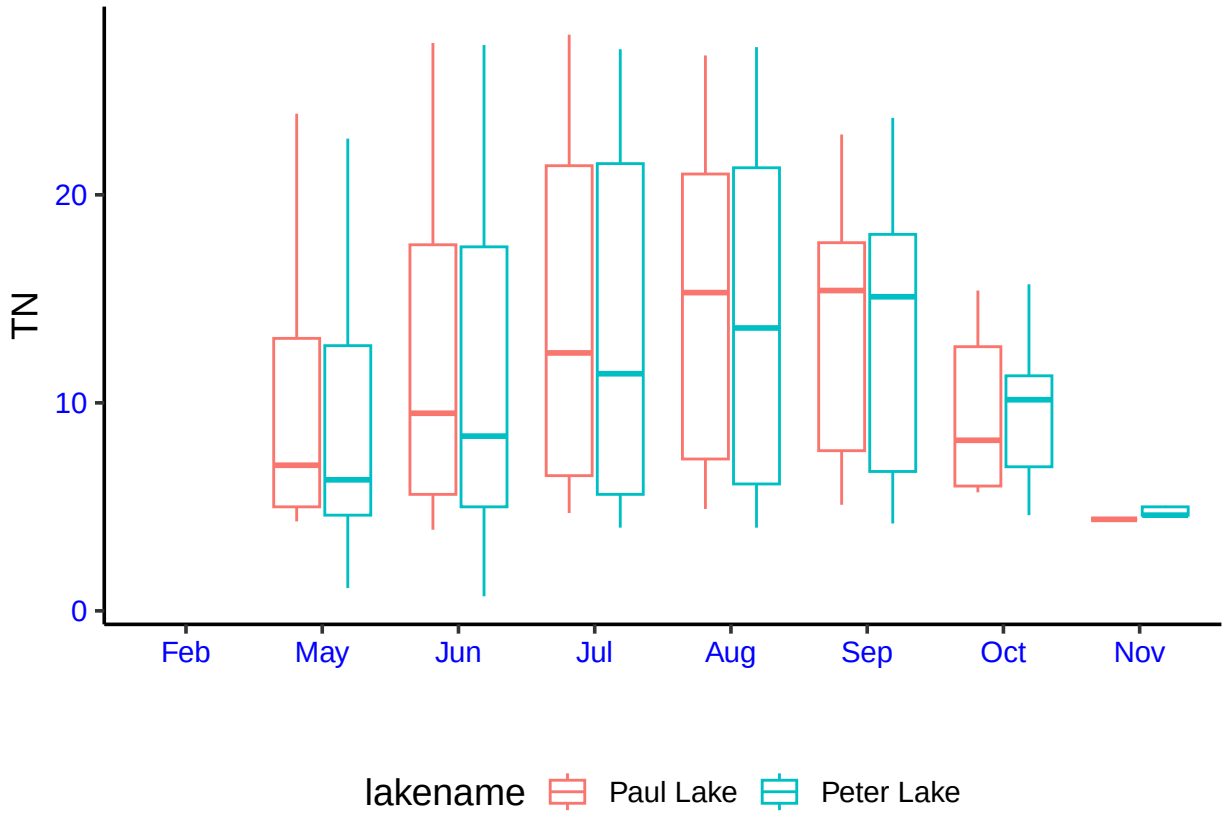
```
  theme(legend.position ="bottom")
print(Plot5b)
```

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




```
#create boxplot by TN
Plot5c<-
  ggplot(PeterPaul,
         aes(x = factor(month,levels=1:12,labels=month.abb),
             y = temperature_C,
             color=lakename))+
  geom_boxplot() +
  labs(y="TN",x="") +
  theme(legend.position ="bottom")
print(Plot5c)
```

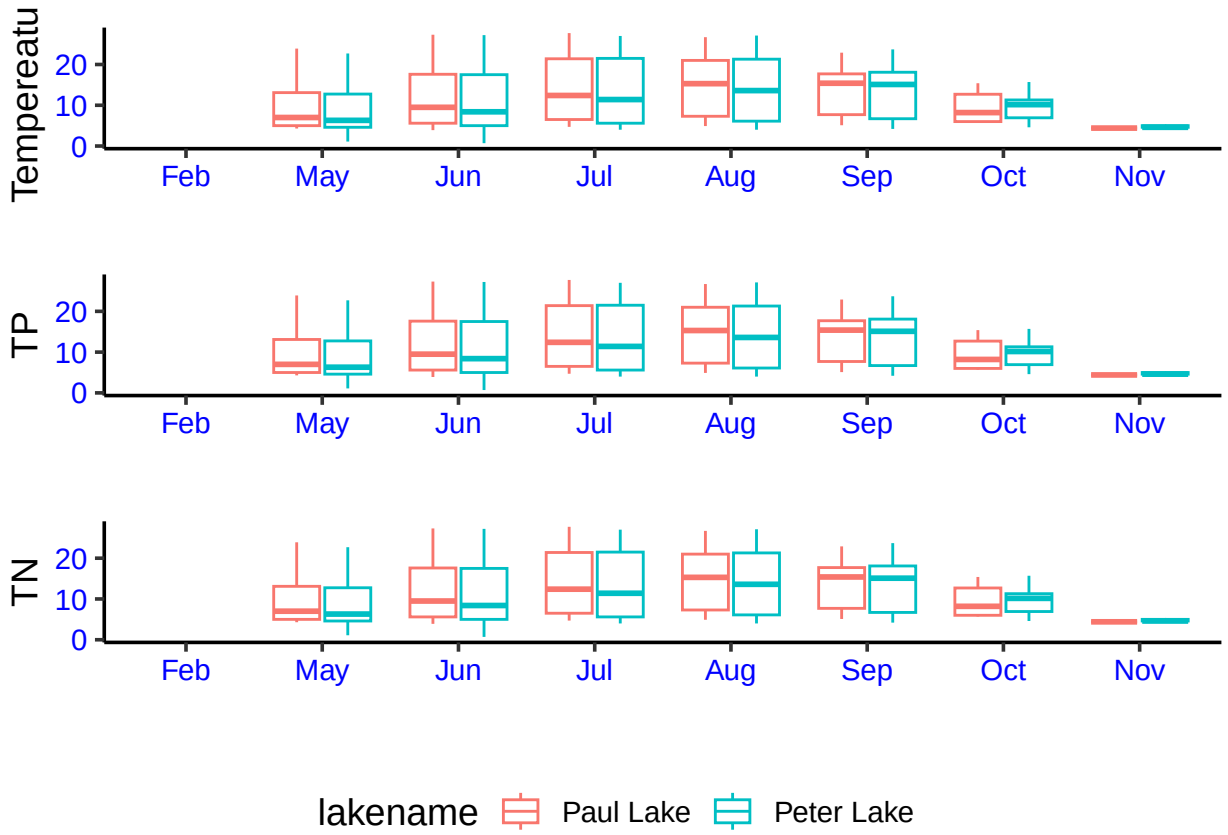
```
## Warning: Removed 3566 rows containing non-finite values (‘stat_boxplot()‘).
```




```
#Combine 3 boxplots, removing legends and adding back modified legend
Plot5<-
  plot_grid(Plot5a + theme(legend.position = "none"),
            Plot5b + theme(legend.position = "none"),
            Plot5c + theme(legend.position = "none"),
  get_legend(Plot5a),
  ncol=1,
  align="vh",
  axis="b",
  rel_heights=c(1,1,1,0.5))
```

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

```
print(Plot5)
```



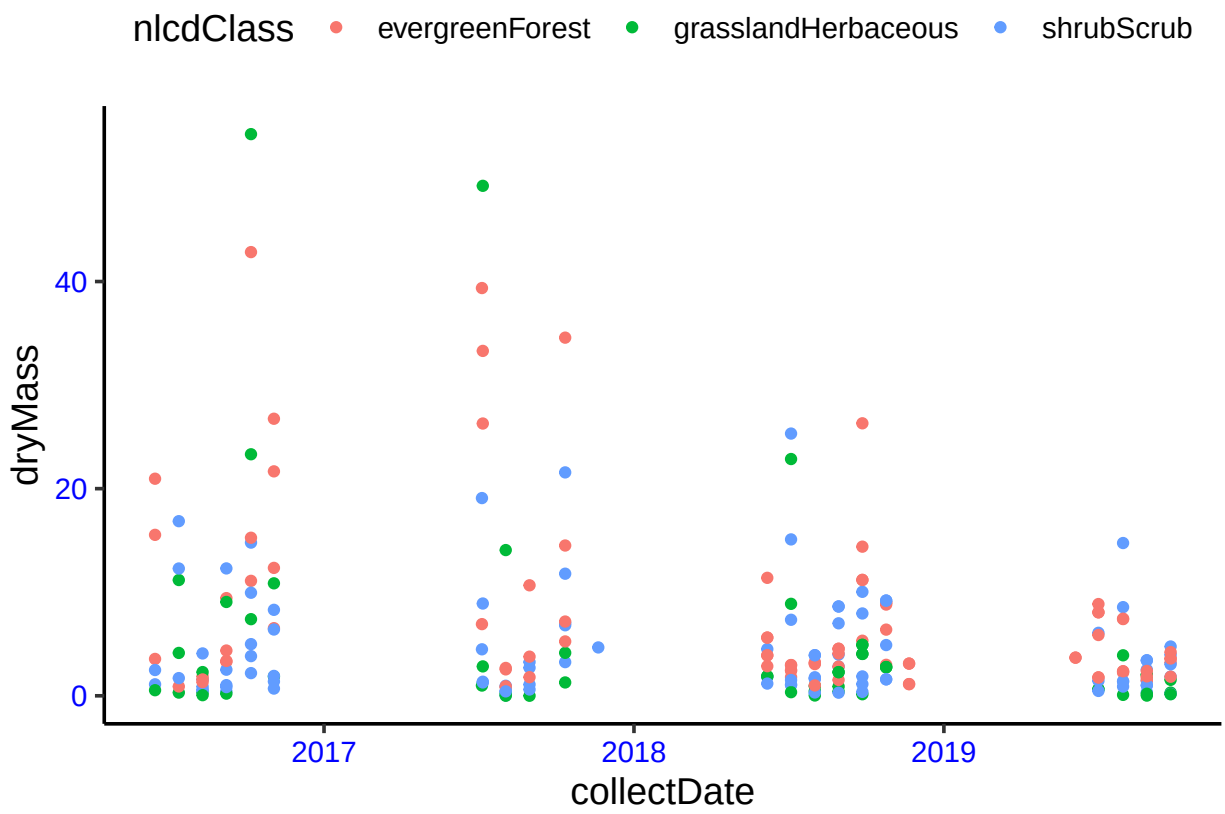
Question: What do you observe about the variables of interest over seasons and between lakes?

Answer: The levels of of TP and TN correlate strongly with seasonal temperature, rising in the summer and falling in winter.

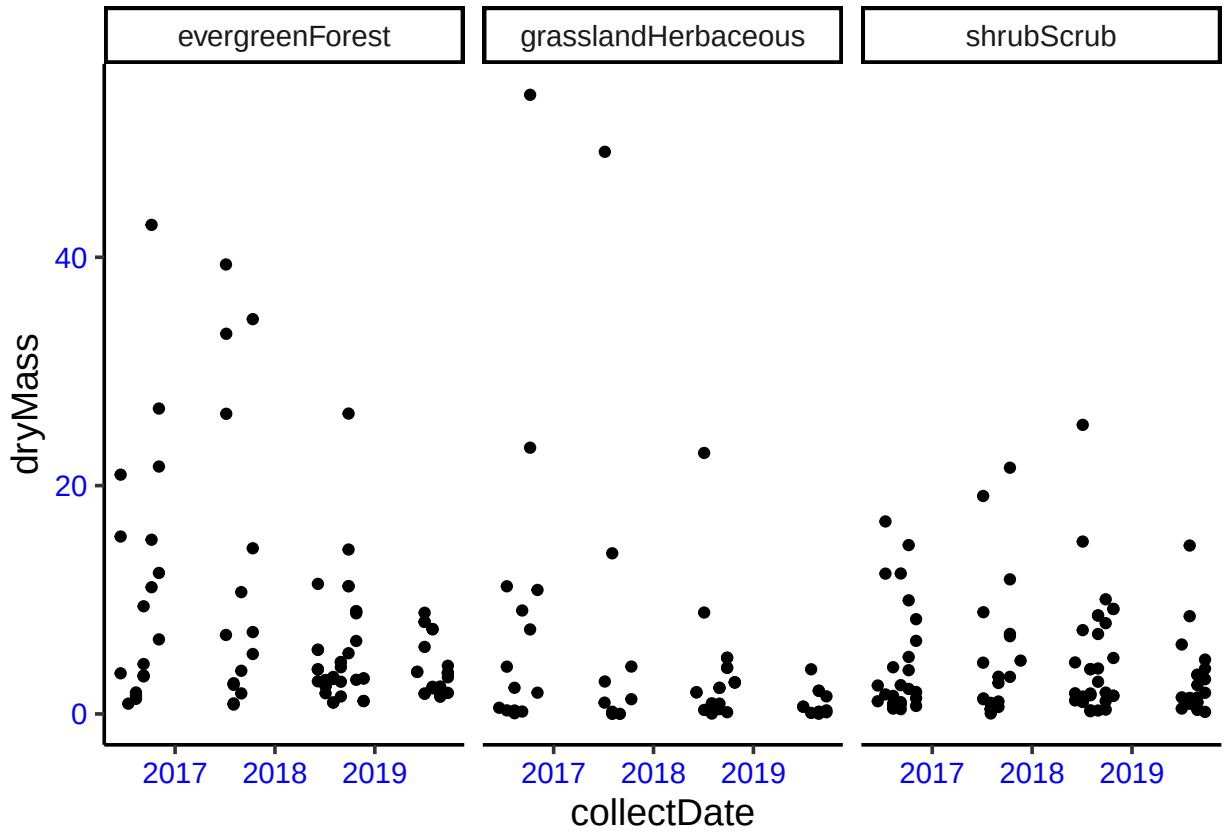
6. [Niwot Ridge] Plot a subset of the litter dataset by displaying only the "Needles" functional group. Plot the dry mass of needle litter by date and separate by NLCD class with a color aesthetic. (no need to adjust the name of each land use)

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

```
#6
#Filter NIWOT data for needles
NIWOTNeedles<-
  NIWOT%>%
  filter(functionalGroup=="Needles")
#Plot filtered data by date, mass, and class
Plot6F<-
  ggplot(NIWOTNeedles,
         aes(x = collectDate,
             y = dryMass,
             color=nlcdClass))+
  geom_point()
print(Plot6F)
```




```
#7
Plot7 <-
  ggplot(NIWOTNeedles,
         aes(x = collectDate,
             y = dryMass)) +
  geom_point() +
  facet_wrap(vars(nlcdClass))
print(Plot7)
```



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

Answer: I believe 7 is more effective because viewing the data by isolated class is easier to understand than trying to color code data. I this case, the color coded data in 6 ends up being both crowded and obscured.