Ariel OCallaghan Assignment 5: Data Visualization

Ariel O'Callaghan

OVERVIEW

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

Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.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 to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.

The completed exercise is due on Friday, Oct 14th @ 5:00pm.

Set up your session

1. Set up your session. Verify your working directory and load the tidyverse, lubridate, & 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_PeterPa` version) and the processed data file for the Niwot Ridge litter dataset (use the [`NEON_NIWO_Litter_mass_trap_Processe` version).

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

```
# 1
getwd()
```

```
## [1] "/home/guest/R/EDA-Fall2022"
```

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

# themes

# install.packages('viridis')
# install.packages('RColorBrewer')
# install.packages('colormap')

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(cowplot)
```

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

```
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(RColorBrewer)
library(colormap)

NTL.LTER.chem.nutrients <- read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Proce
    stringsAsFactors = TRUE)
NEON.litter.mass <- read.csv("./Data/Processed/NEON_NIWO_Litter_mass_trap_Processed.csv",
    stringsAsFactors = TRUE)
# 2

NEON.litter.mass$collectDate <- as.Date(NEON.litter.mass$collectDate,
    format = "%Y-%m-%d")
# factor(NEON.litter.mass$collectDate)
class(NEON.litter.mass$collectDate)
```

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

```
NTL.LTER.chem.nutrients$sampledate <- as.Date(NTL.LTER.chem.nutrients$sampledate,
    format = "%Y-%m-%d")
class(NTL.LTER.chem.nutrients$sampledate)
```

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

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"),
    legend.position = "top")

theme_set(mytheme)  #set theme for all subsequent plots
```

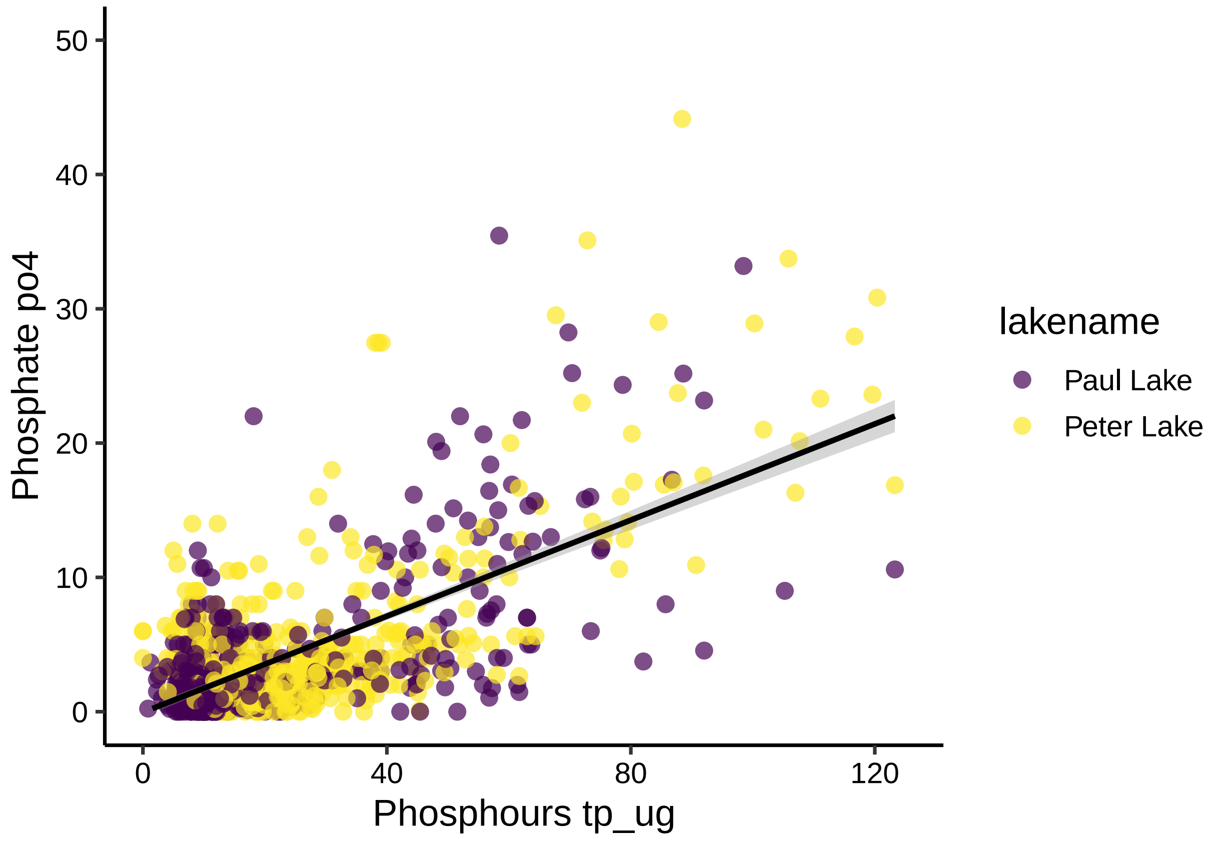
Create graphs

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 Display colors
# scales::show_col(colormap(colormap =
# colormaps$viridis, nshades = 16))
# scales::show_col(colormap(colormap =
# colormaps$inferno, nshades = 16))
# scales::show_col(colormap(colormap =
# colormaps$magma, nshades = 16))
# display.brewer.all(n = 9)

# scatterplot
phosphors.vs.phosphate <- ggplot(NTL.LTER.chem.nutrients,
    aes(x = tp_ug, y = po4, color = lakename)) +
    geom_point(alpha = 0.7, size = 2.5) + geom_smooth(method = lm,
    color = "black") + xlim(0, 125) + ylim(0,
    50) + scale_color_viridis_d() + theme(legend.position = "right") +
    labs(x = "Phosphours tp_ug", y = "Phosphate po4")
print(phosphors.vs.phosphate)
```

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

## Warning: Removed 21952 rows containing non-finite values (stat_smooth).

## Warning: Removed 21952 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_smooth).
```



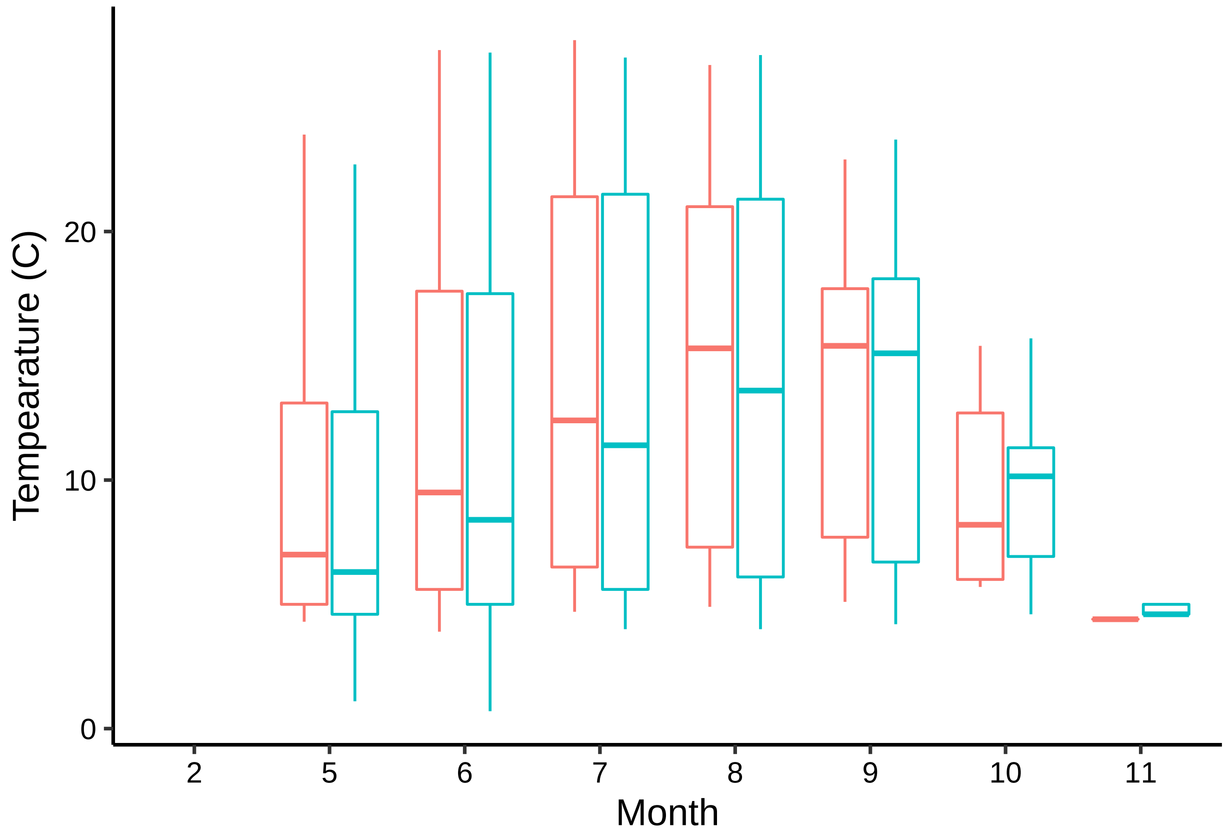
5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and

(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: R has a build in variable called `month.abb` that returns a list of months; see https://r-lang.com/month-abb-in-r-with-example

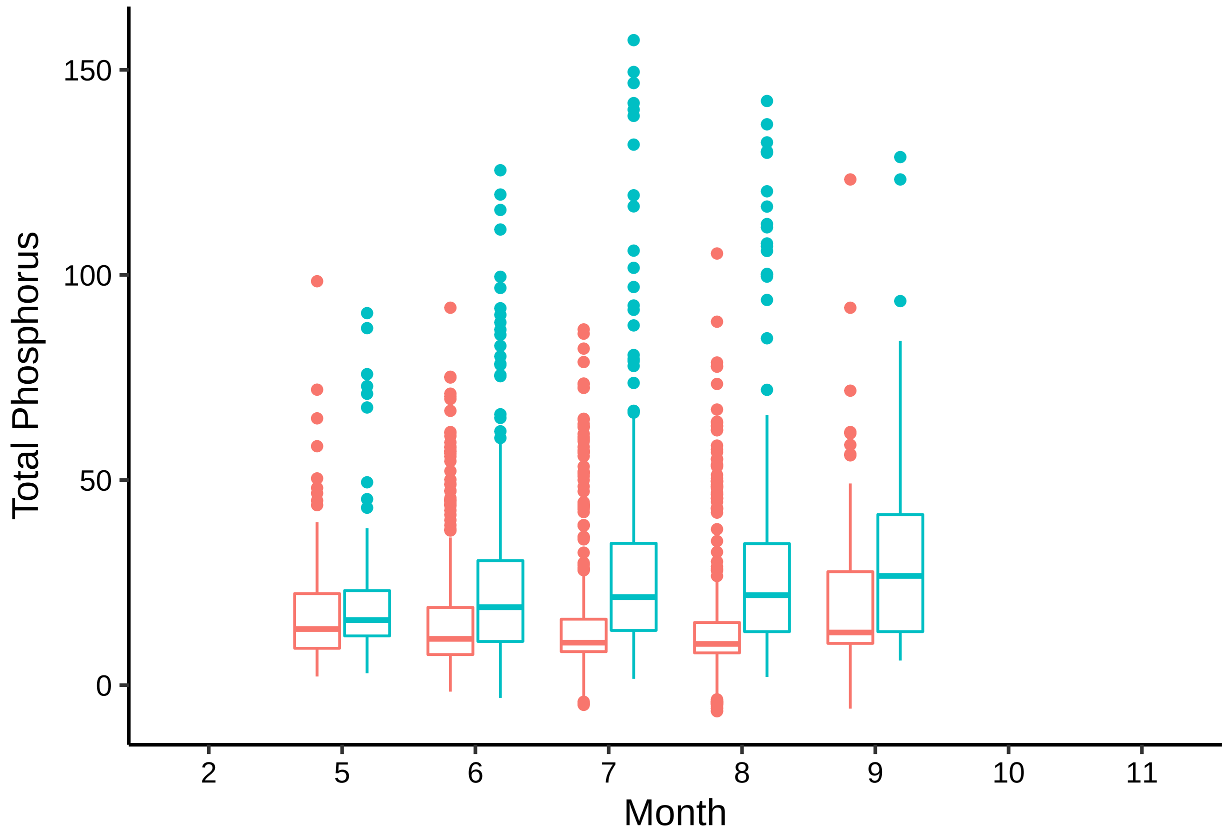
```
# 5

NTL.LTER.chem.nutrients$month <- as.factor(NTL.LTER.chem.nutrients$month)  #fix month so 1-12 show up

chem.nutrients.temp <- ggplot(NTL.LTER.chem.nutrients,
    aes(x = month, y = temperature_C)) + geom_boxplot(aes(color = lakename)) +
    xlab(expression("Month")) + ylab(expression("Tempearature (C)")) +
    labs(color = "Lake Name") + theme(legend.position = "none")
print(chem.nutrients.temp)
```

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



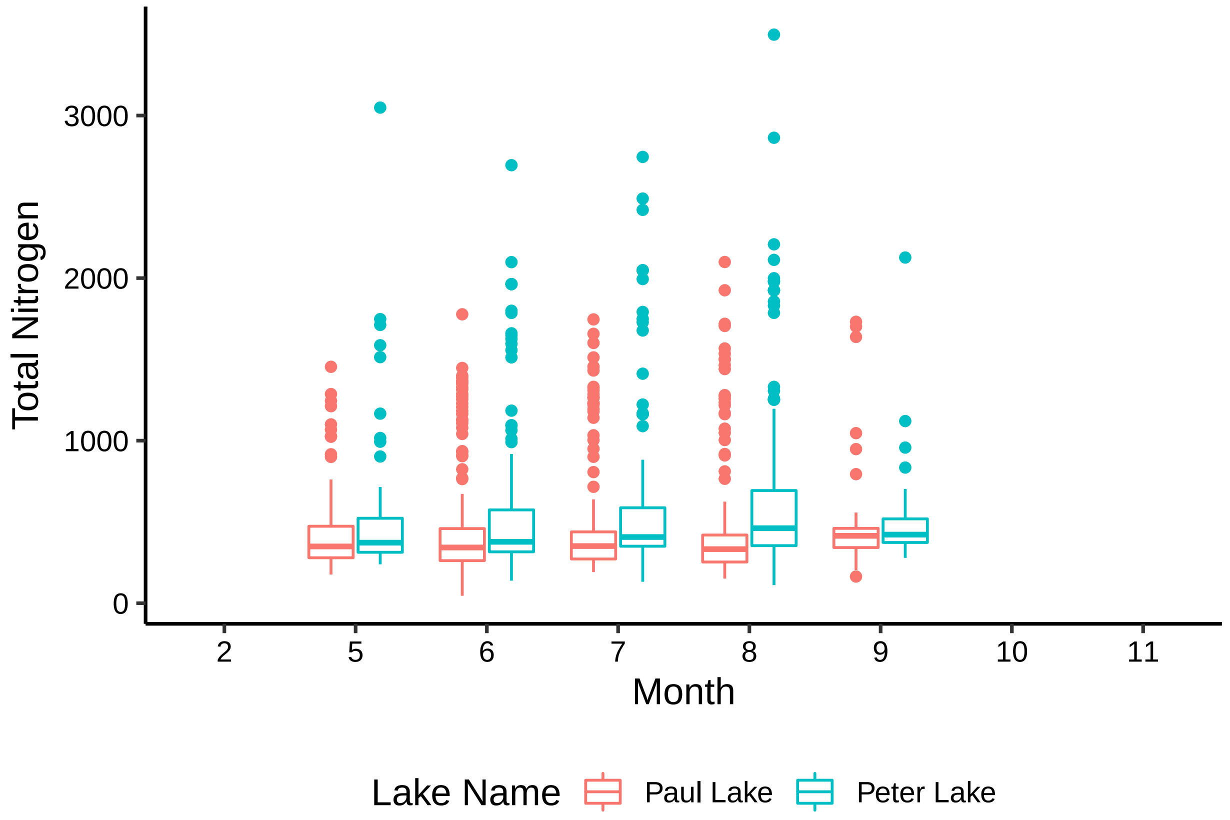
```r
chem.nutrients.TP <- ggplot(NTL.LTER.chem.nutrients,
    aes(x = month, y = tp_ug)) + geom_boxplot(aes(color = lakename)) +
    xlab(expression("Month")) + ylab(expression("Total Phosphorus")) +
    labs(color = "Lake Name") + theme(legend.position = "none")
print(chem.nutrients.TP)
```

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



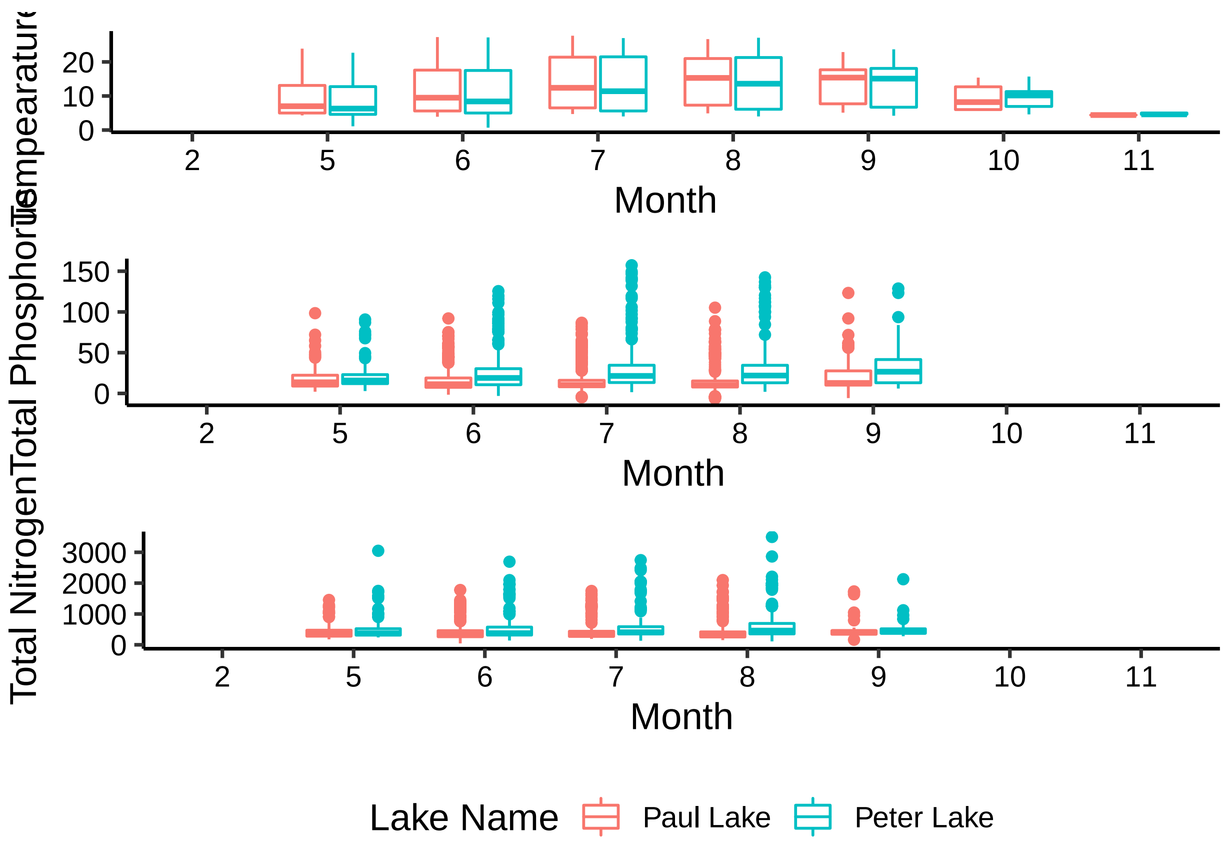
```
chem.nutrients.TN <- ggplot(NTL.LTER.chem.nutrients,
    aes(x = month, y = tn_ug)) + geom_boxplot(aes(color = lakename)) +
    xlab(expression("Month")) + ylab(expression("Total Nitrogen")) +
    labs(color = "Lake Name") + theme(legend.position = "bottom")
print(chem.nutrients.TN)
```

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



```
plot_grid(chem.nutrients.temp, chem.nutrients.TP,
    chem.nutrients.TN, nrow = 3, align = "h",
    rel_heights = c(2.5, 3, 4))
```

```
## 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.
```



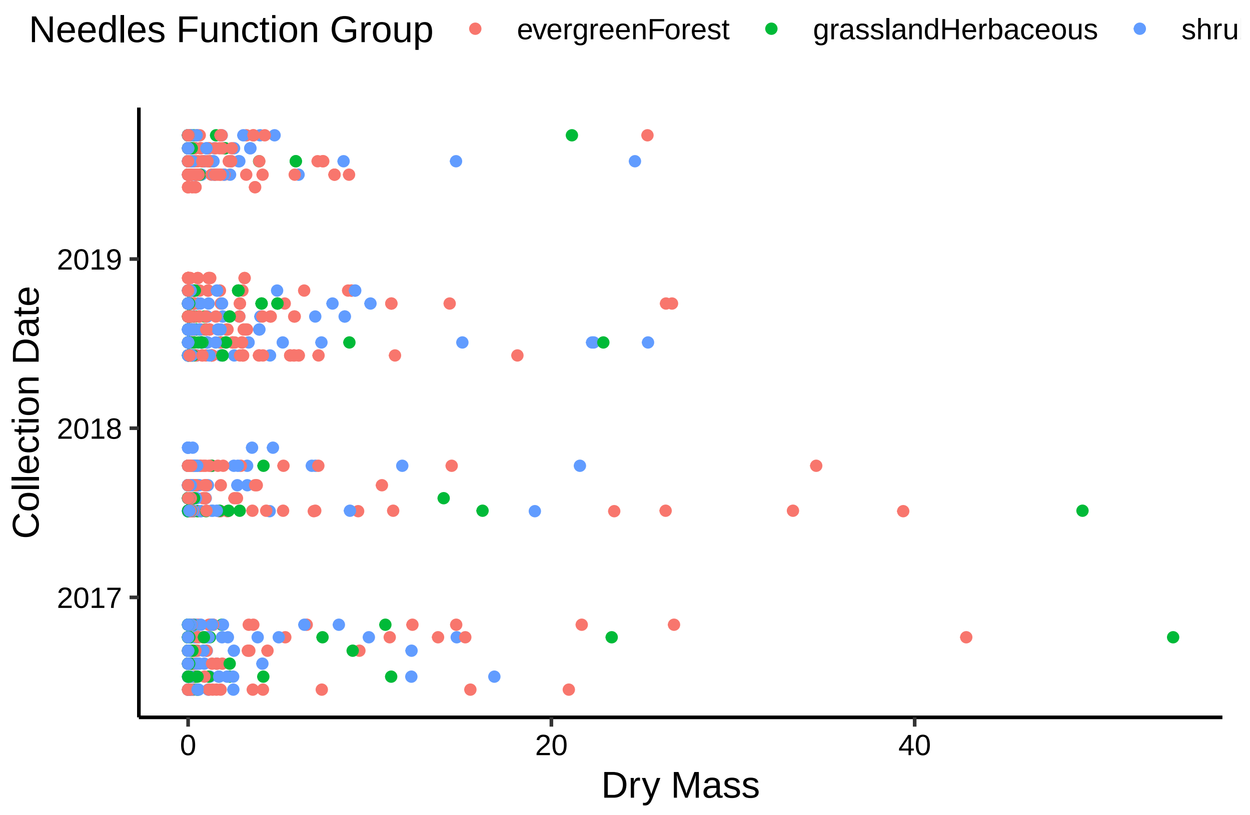
```
#+element_text(size = NULL)

# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN, nrow
# = 3, align = 'v', rel_heights =
# c(2.5,3,4))+ theme(axis.title.y = 'right')

# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN, nrow
# = 3, align = 'v', rel_heights =
# c(2.5,3,4))+ theme(axis.text.y =
# element_text(angle = 45))

# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN,
# align = 'v', rel_heights = c(8,8,10))

# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN,
# align = 'h')

# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN, nrow
# = 1, axis = 'I')

# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN,
# aling = 'v')
# plot_grid(chem.nutrients.temp,
# chem.nutrients.TP, chem.nutrients.TN, axis
# = 'tblr')
```

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

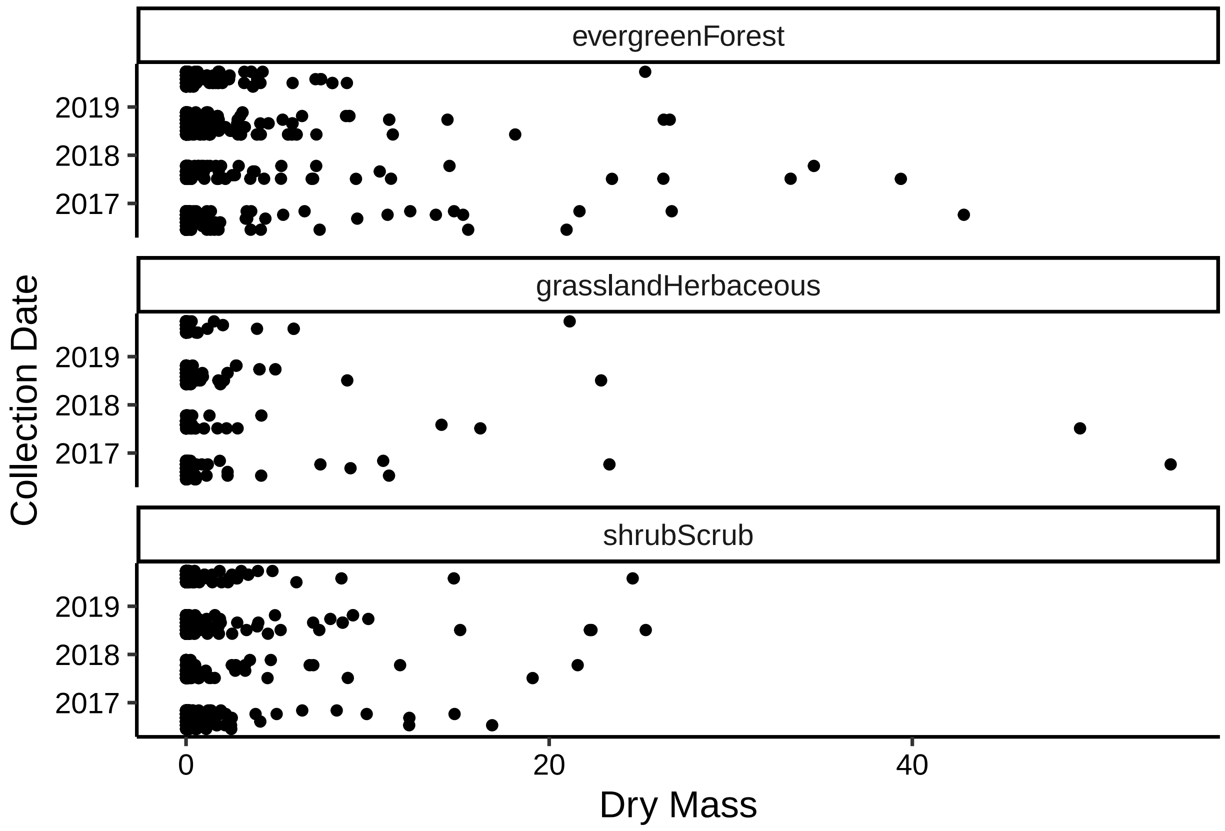
> Answer: The temperatue increases during the summer months and is colder in the December. Temperature in Peter and Paul lake follow similar trends with the greatest different in means for the month of November. Total Phosphorus is lesss in Paul lake then Peter lake with it slowly increasing throught the year. Total Nitrogen doesis higher in Peter lake then Paul 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
NEON.litter.mass.function.group <- filter(NEON.litter.mass,
    functionalGroup %in% c("Needles"))

Neon.litter.needles <- ggplot(NEON.litter.mass,
    aes(x = dryMass, y = collectDate)) + geom_point(aes(color = nlcdClass)) +
    xlab(expression("Dry Mass")) + ylab(expression("Collection Date")) +
    labs(color = "Needles Function Group")
print(Neon.litter.needles)
```



```
# 7

Neon.litter.needles.facet <- ggplot(NEON.litter.mass,
    aes(x = dryMass, y = collectDate)) + geom_point() +
    xlab(expression("Dry Mass")) + ylab(expression("Collection Date")) +
    labs(color = "Needles Function Group") + facet_wrap(vars(nlcdClass),
    nrow = 3)
print(Neon.litter.needles.facet)
```



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

> Answer: Graph 7 splits out the data for each needles functional group and easier to see. The points on graph 6 overlap. Graph 7 parces out the data more and is easier to compare.