Assignment 5: Data Visualization

Brock Keller

Fall 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     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## 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/EDE_Fall2024
```

```
library(cowplot)
```

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

```
library(ggplot2)
getwd()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

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

Peter_Paul_ChemPhys <- read.csv(
  here("./Data/Processed_KEY/NTL-LTER_Lake_ChemistryPhysics_PeterPaul_Processed.csv"),
  stringsAsFactors = TRUE
)

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

#2
class(Peter_Paul_nutrients$sampledate) #checking current format of date values
```

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

```
class(Peter_Paul_ChemPhys$sampledate)
```

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

```
class(Niwot_Ridge_Litter$collectDate)
```

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

```
#setting date format
Peter_Paul_nutrients$sampledate <- ymd(Peter_Paul_nutrients$sampledate)
Peter_Paul_ChemPhys$sampledate <- ymd(Peter_Paul_ChemPhys$sampledate)
Niwot_Ridge_Litter$collectDate <- ymd(Niwot_Ridge_Litter$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
my_theme <- theme_linedraw() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
        panel.grid.minor.y = element_line(size = 0), panel.grid.major.y = element_line(size = 0.08),
        panel.grid.major.x = element_line(linetype = "dotted", size = 0.25, color = "black"))
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
#remnants of playing around with this
#legend.position = "top", took this out of the default theme so it doesnt mess with cowplot
#panel.grid.major = element_line(linetype = "dotted") a bit ugly
#panel.grid.major.y = element_line(size = 0.1)
#panel.grid.major.x = element_line(size = 0.08)
```

Create graphs

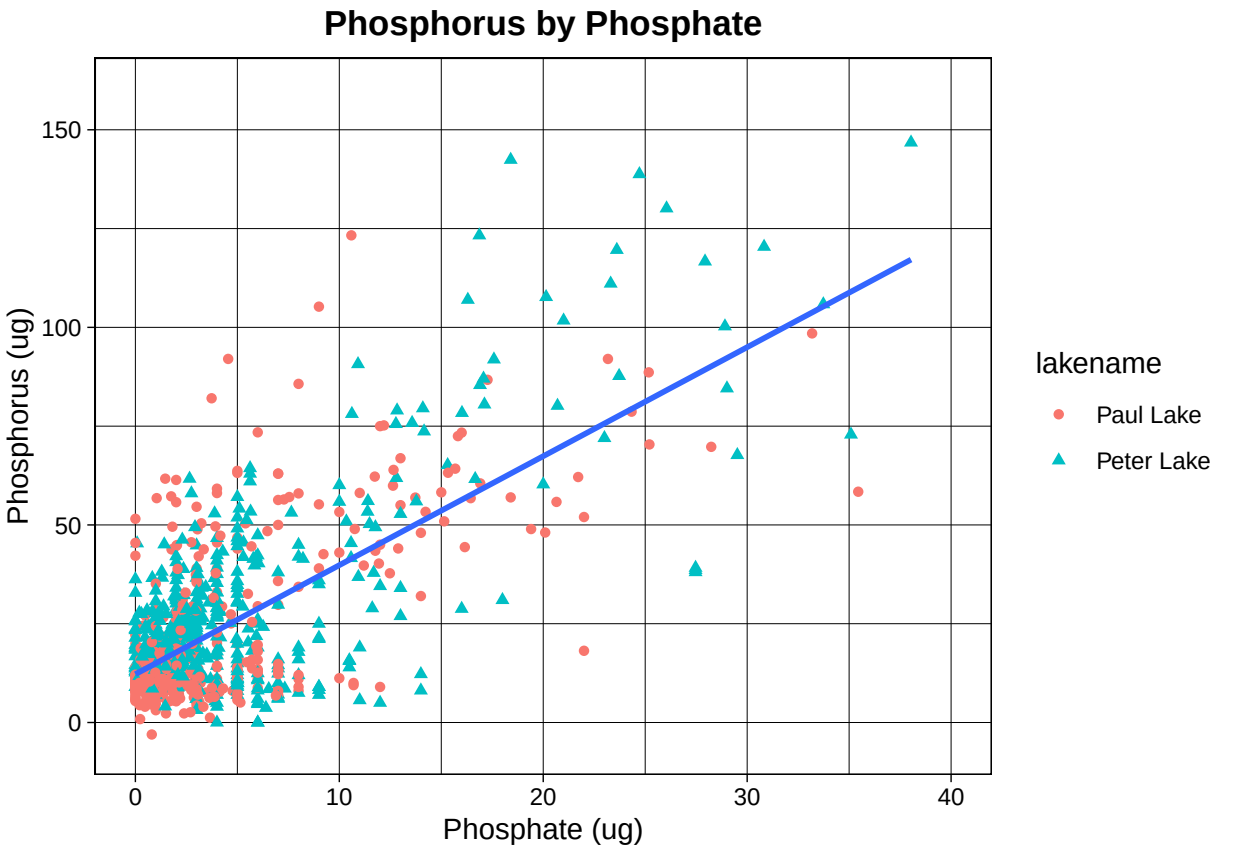
For numbers 4-7, create ggplot graphs and adjust aesthetics to follow best practices for data visualization. Ensure your theme, color palettes, axes, and additional aesthetics are edited accordingly.

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
theme_set(theme_linedraw())

PPL_phosphorus_phosphate <-
  ggplot(Peter_Paul_nutrients) +
  geom_point(aes(x = po4, y = tp_ug, color = lakename, shape = lakename)) +
  xlim(0, 40) +
  ylim(-5, 160) +
```

```
  geom_smooth(aes(x = po4, y= tp_ug), method = lm, se = FALSE) +
  labs(title = "Phosphorus by Phosphate", y = "Phosphorus (ug)", x = "Phosphate (ug)") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



print(PPL_phosphorus_phosphate)
```

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

## Warning: Removed 21948 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 21948 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

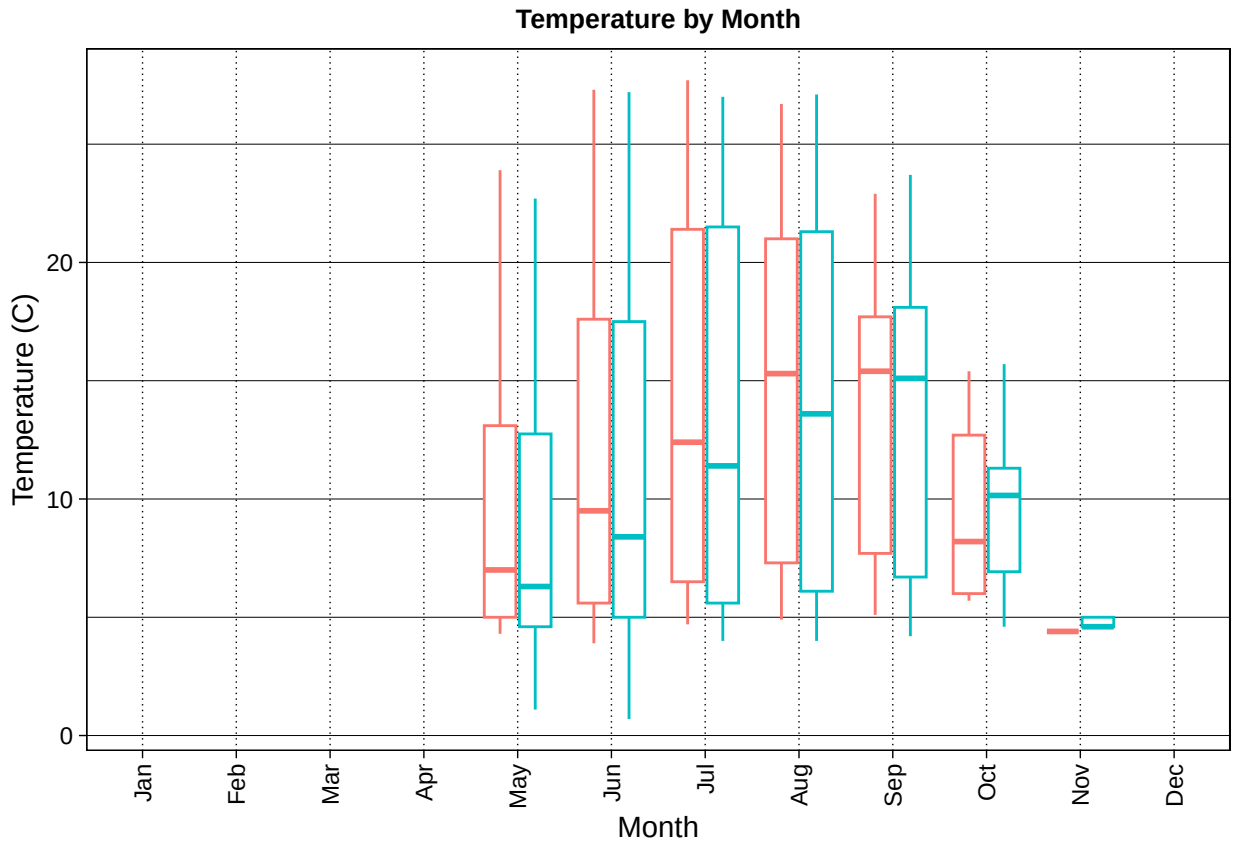


5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and (c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

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`.

```r
#5
#making my individual boxplots
theme_set(my_theme) #I added the dotted line in this theme because it makes it so much easier
#for me to see the pairs and what month they correspond to

PPL_temp_month <-
  Peter_Paul_nutrients %>%
  ggplot(aes(x = factor(month, levels = 1:12, labels = month.abb), y = temperature_C,
             color = lakename)) + #describing the aesthetics
  geom_boxplot() + #making the actual plot
  scale_x_discrete(name = 'Month', drop = FALSE) + #drop lets me keep months w/o data
  #scale_y_discrete(name = "Temperature (C)") +
  labs(title = "Temperature by Month", y = "Temperature (C)") +
  theme(legend.position = "none", plot.title = element_text(hjust = 0.5, face = "bold", size = 10))


print(PPL_temp_month)
```

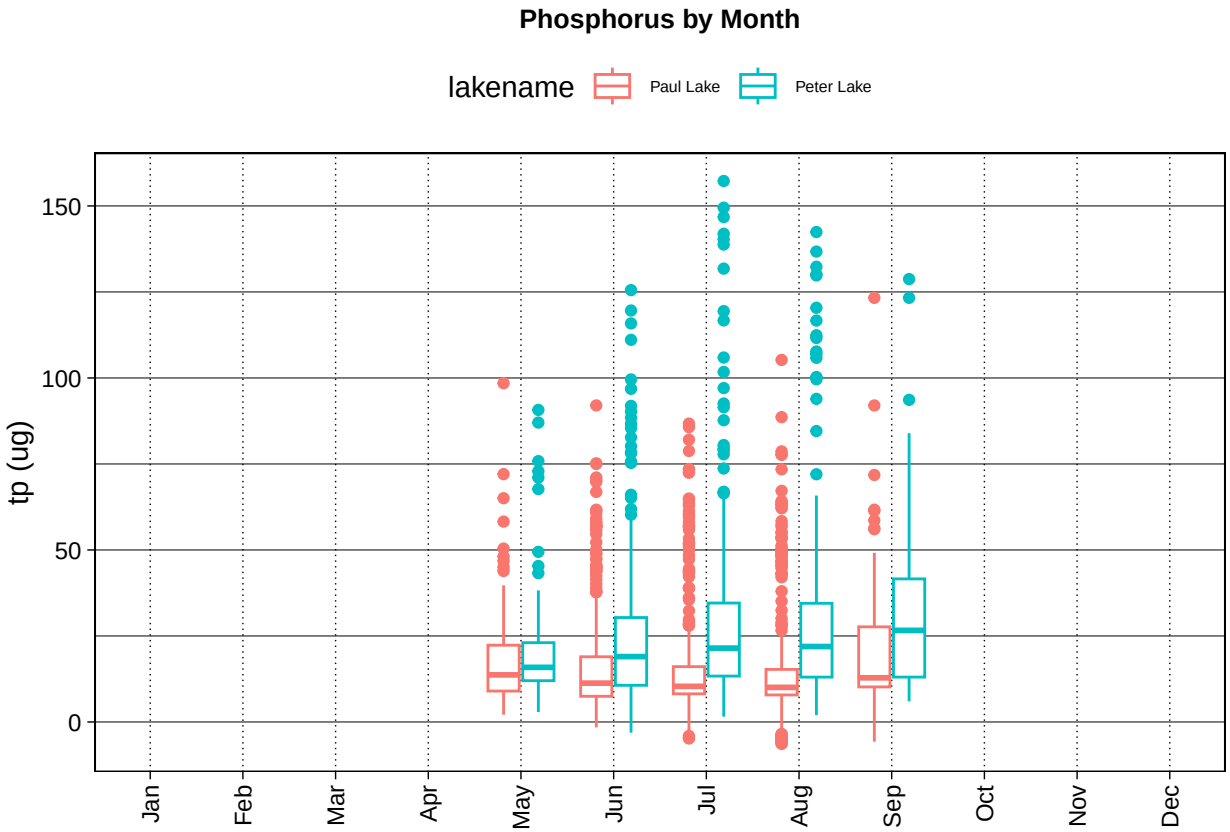
```
## Warning: Removed 3566 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```




```r
PPL_tp_month <-
  Peter_Paul_nutrients %>%
  ggplot(aes(x = factor(month, levels = 1:12, labels = month.abb), y = tp_ug, color = lakename)) +
  geom_boxplot() +
```

```
  scale_x_discrete(element_blank(), drop = FALSE) +
  #scale_y_discrete(name = "tp (ug)") +
  labs(title = "Phosphorus by Month", y = "tp (ug)") +
  theme(legend.position = "top", plot.title = element_text(hjust = 0.5, face = "bold", size = 10),
        legend.text = element_text(size = 6))


print(PPL_tp_month)
```

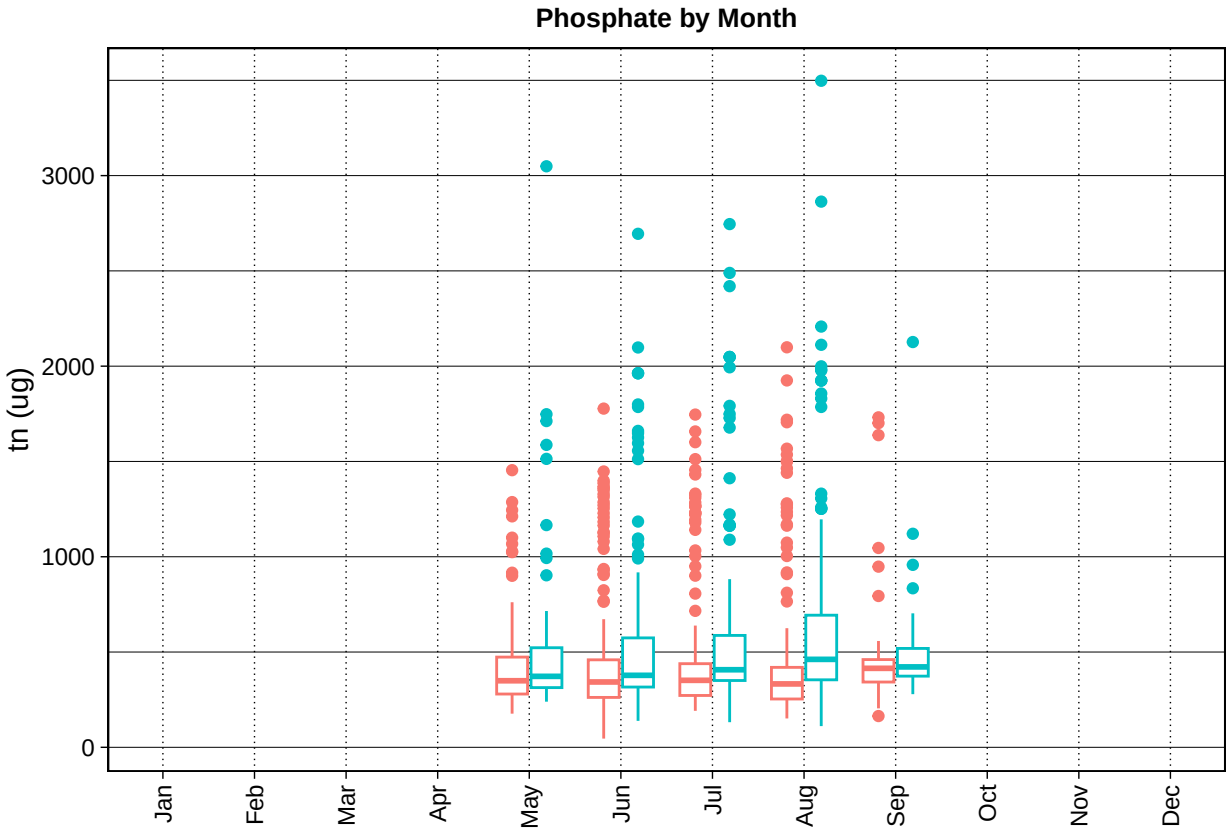
```
## Warning: Removed 20729 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```




```
PPL_tn_month <-
  Peter_Paul_nutrients %>%
  ggplot(aes(x = factor(month, levels = 1:12, labels = month.abb), y = tn_ug, color = lakename)) +
  geom_boxplot() +
  scale_x_discrete(element_blank(), drop = FALSE) +
  #scale_y_discrete(name = "tn (ug)") +
  labs(title = "Phosphate by Month", y = "tn (ug)") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 10), legend.position = "none")


print(PPL_tn_month)
```

```
## Warning: Removed 21583 rows containing non-finite outside the scale range
```

```
## (‘stat_boxplot()‘).
```



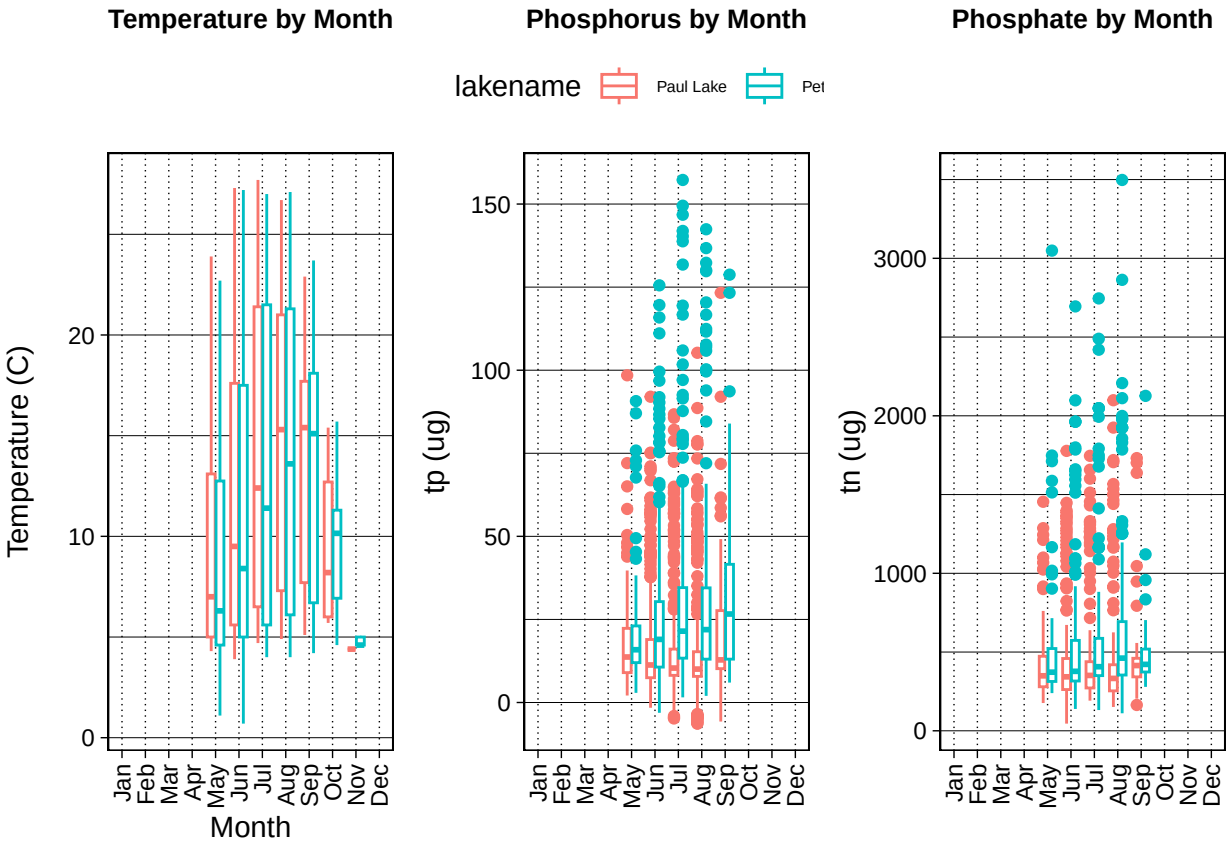

```
together <-
  plot_grid(PPL_temp_month, PPL_tp_month, PPL_tn_month, nrow = 1, align = "hv")
```

```
## Warning: Removed 3566 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```

```
## Warning: Removed 20729 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```

```
## Warning: Removed 21583 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```

```
#I know this looks cramped on the PDF but I spent forever making it look gorgeous in R Studio and I
#can't figure out how to change the display on the knit more than this!
print(together)
```



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

> Answer: As expected, each month sees a negligable difference in water temperature between the lakes. Data on the specific nutrient composition is limited to a few summer months, but from what is available it seems that Peter Lake has slighter higher concentrations of phosphorus and phosphate across the board despite Paul Lake having more frequent high outliers. Also interestingly, the mean phosphorus concentration in Peter Lake increases throughout the summer while mean concentration progressively decreases in Paul Lake until September.

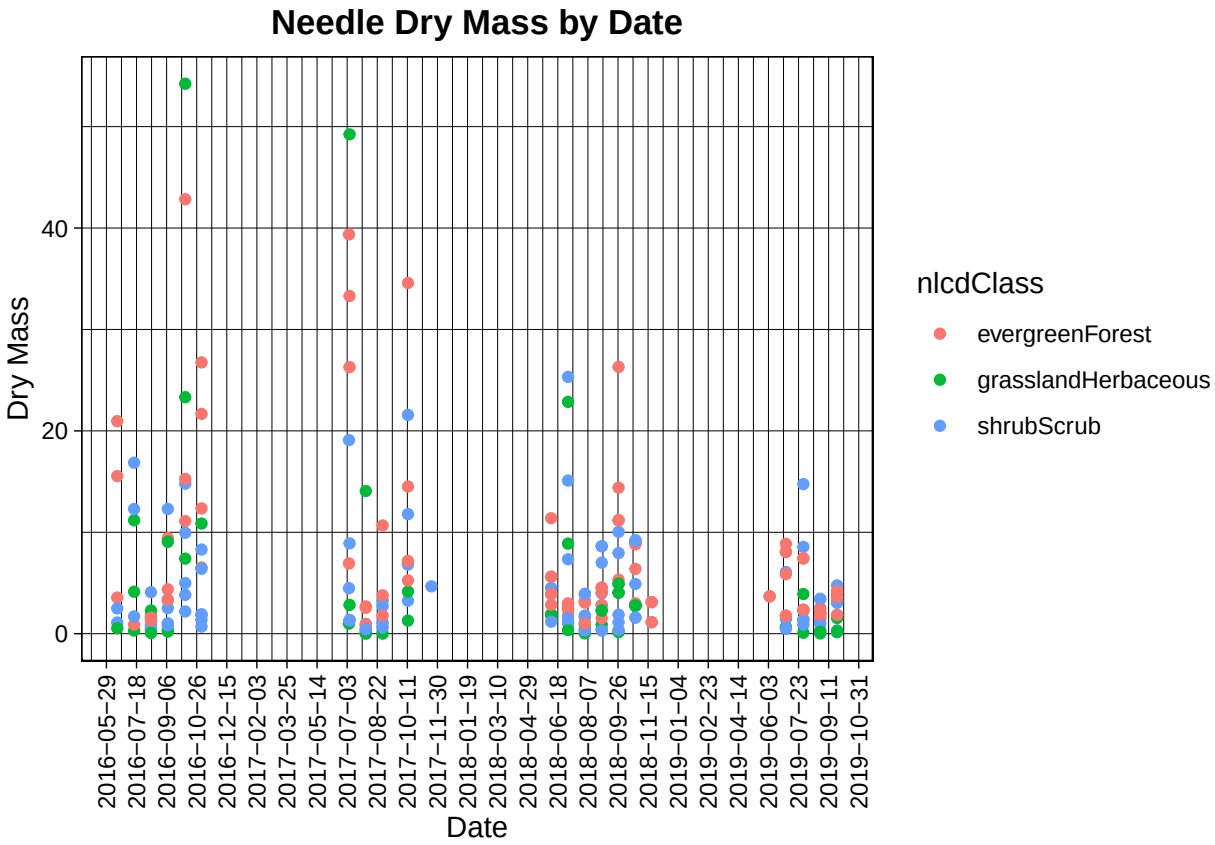
6. [Niwot Ridge] Plot a subset of the litter dataset by displaying only the "Needles" functional group. Plot the dry mass of needle litter by date and separate by NLCD class with a color aesthetic. (no need to adjust the name of each land use)

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

```
#6
theme_set(theme_linedraw())

subset <- Niwot_Ridge_Litter %>%
  filter(functionalGroup == "Needles")

class(subset$collectDate) #making sure these are stored as dates
```

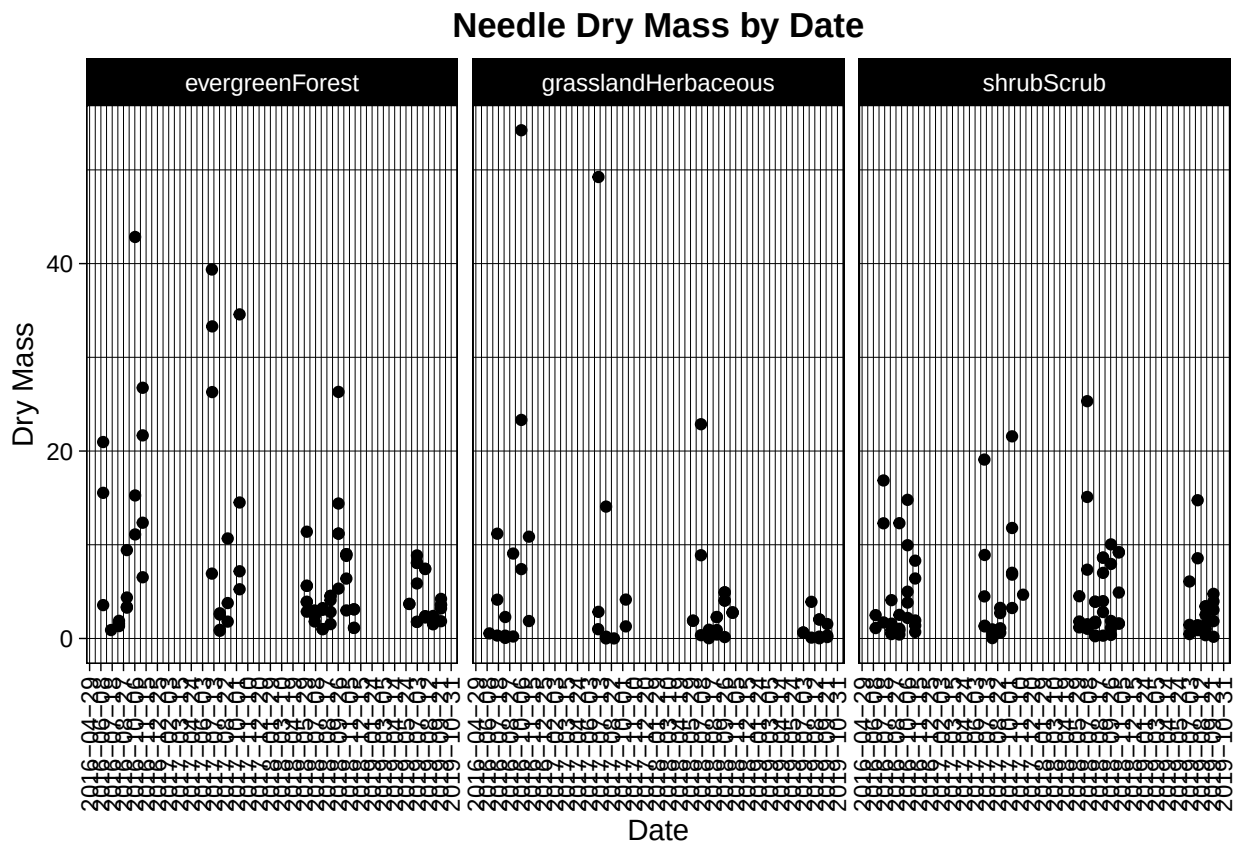
```
## [1] "Date"
```

```r
subset$collectDate <- ymd(subset$collectDate) #setting date format


Needle_mass_by_date <-
  subset %>%
  ggplot(aes(x = collectDate, y = dryMass, color = nlcdClass)) +
  geom_point() +
  scale_x_date(breaks = "50 days") + #adding axis break so the data is visible
  labs(title = "Needle Dry Mass by Date", x = "Date", y = "Dry Mass") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
        plot.title = element_text(hjust = 0.5, face = "bold"))


print(Needle_mass_by_date)
```




```r
#7
Needle_mass_by_date_faceted <-
  subset %>%
  ggplot(aes(x = collectDate, y = dryMass)) +
  geom_point() +
  facet_wrap(facets = vars(nlcdClass), ncol = 3, nrow = 1) +
  scale_x_date(breaks = "40 days") + #adding axis break so the data is visible
  labs(title = "Needle Dry Mass by Date", x = "Date", y = "Dry Mass") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
        plot.title = element_text(hjust = 0.5, face = "bold"))
```

```
print(Needle_mass_by_date_faceted)
```



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

> Answer: I think the faceted plot is significantly more effective at actually telling a story with the data. Even though the points in plot 6 are classified by color, there is just too much noise and it is hard to discern any real trend between or within categories. Having the three plots side by side makes it so much easier to see what the general trends are in each class and also allows you to compare side by side with the others.