Assignment 5: Data Visualization

Nargis Taraki

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 error
```

```
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
```

```
here()
```

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

```
#here i read in my both datasets.

peter_paul_data <- read.csv(here("Data", "Processed_KEY", "NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_
                            stringsAsFactors = TRUE)

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


#2  Here I check the class (format) of the sampledate column in Peter and Paul Lakes data
class(peter_paul_data$sampledate)
```

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

```
# Check the class (format) of the collectDate column in Niwot Ridge litter data
class(niwot_litter_data$collectDate)
```

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

```
# Since it is "factor" I am Converting it to Date format
peter_paul_data$sampledate <- as.Date(as.character(peter_paul_data$sampledate))

# Here I converted the Niwot Ridge litter data to Date format
niwot_litter_data$collectDate <- as.Date(as.character(niwot_litter_data$collectDate))

# I run this codes to verify the conversion
class(peter_paul_data$sampledate)
```

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

```r
class(niwot_litter_data$collectDate)
```

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

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

```r
#3

library(ggplot2)

# Here I build a custom theme

custom_theme <- theme(
  plot.background = element_rect(fill = "lightblue", color = NA), # Customize plot background
  plot.title = element_text(face = "bold", size = 16, hjust = 0.5, color = "darkblue"), # Customize tit
  axis.title.x = element_text(face = "italic", size = 12), # Customize x-axis label
  axis.title.y = element_text(face = "italic", size = 12), # Customize y-axis label
  axis.text = element_text(color = "black"), # Customize axis text color
  panel.grid.major = element_line(color = "gray"), # Customize gridlines
  legend.position = "top", # Customize legend position
  legend.background = element_rect(fill = "white", color = "black") # Customize legend background
)

# I Set the custom theme as the default
theme_set(custom_theme)
```

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.

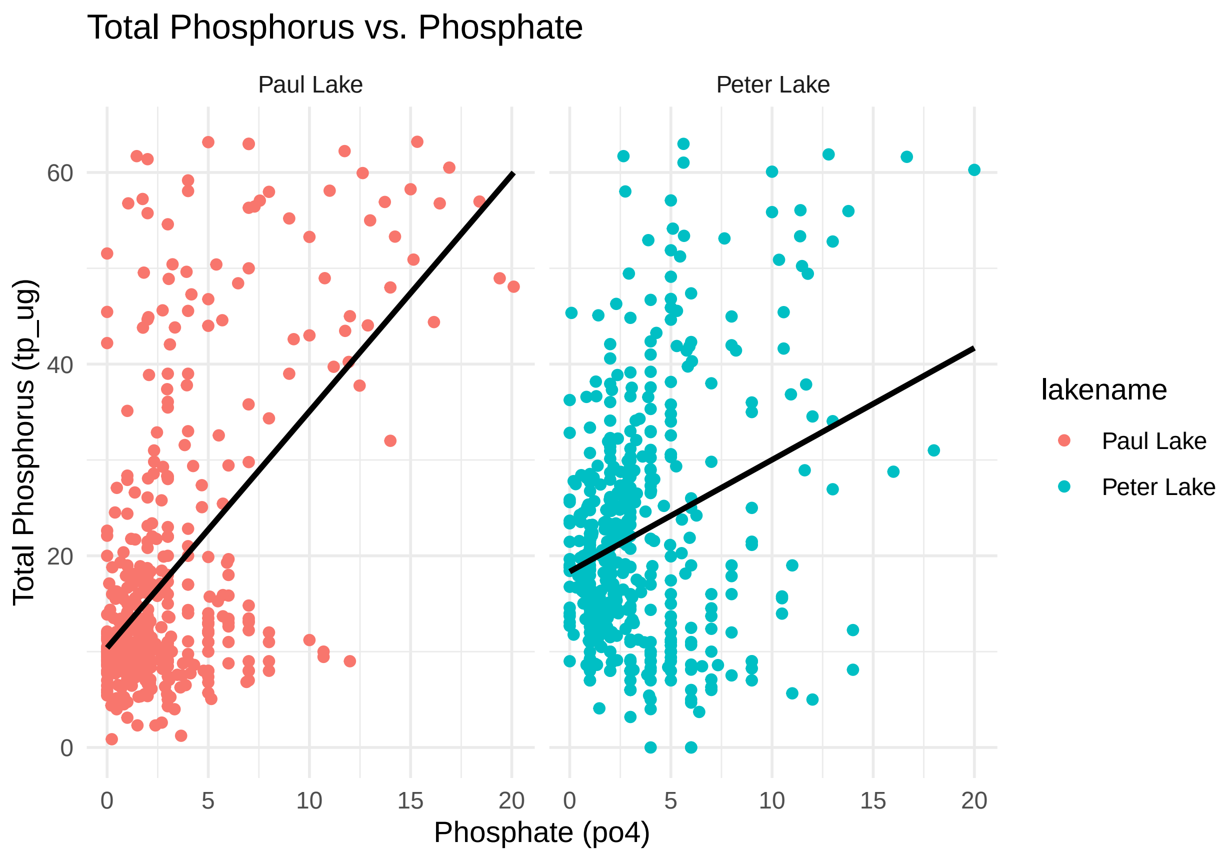
4. [NTL-LTER] Plot total phosphorus (`tp_ug`) by phosphate (`po4`), with separate aesthetics for Peter and Paul lakes. Add line(s) of best fit using the `lm` method. Adjust your axes to hide extreme values (hint: change the limits using `xlim()` and/or `ylim()`).

```r
#4 Here I Load the dplyr and ggplot2  libraries for data manipulation and data visualization
library(dplyr)
library(ggplot2)

# Here I create separate plots for Peter and Paul lakes
ggplot(peter_paul_data, aes(x = po4, y = tp_ug, color = lakename)) +
  geom_point() +  # Add points to the plot
  geom_smooth(method = "lm", formula = y ~ x, color = "black", se = FALSE) +  # Add line of best fit
  labs(
    title = "Total Phosphorus vs. Phosphate",
    x = "Phosphate (po4)",
    y = "Total Phosphorus (tp_ug)"
  ) +
  # Here I Set limits for the x and y-axis based on the 95th percentile of total phosphorus
  scale_y_continuous(limits = c(0, quantile(peter_paul_data$tp_ug, 0.95, na.rm = TRUE))) +
  scale_x_continuous(limits = c(0, quantile(peter_paul_data$po4, 0.95, na.rm = TRUE))) +
  theme_minimal() +
  facet_wrap(~ lakename, ncol = 2)  # I used the Facet-wrap command to Separate plots for Peter and Pau
```

```
## Warning: Removed 22012 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 22012 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`.

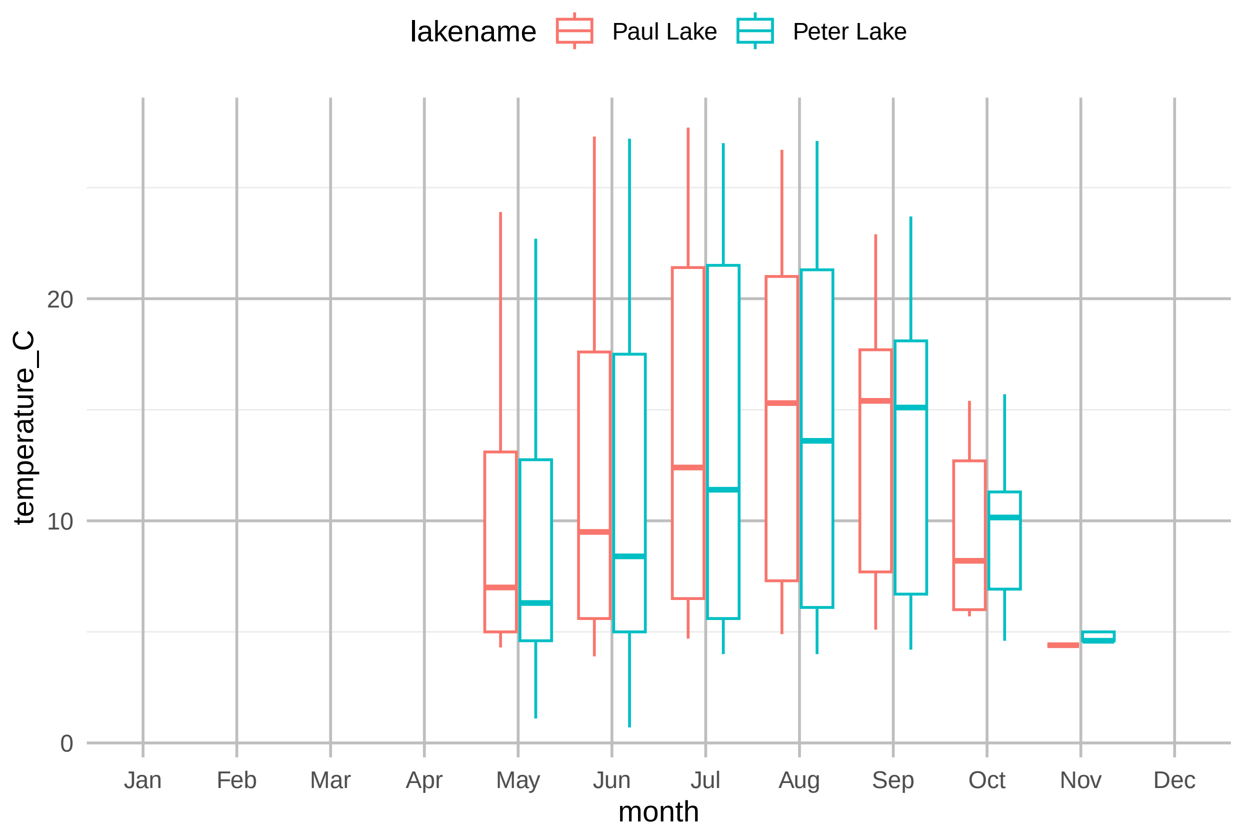
```
#5
# Here I update month from numeric value to character with correct ordering and labels
peter_paul_data$month <- factor(
  peter_paul_data$month,
  levels = c("2", "5", "6", "7", "8", "9", "10", "11")  # Only the months present in the data
)

# Then I converted numeric month to full month labels for consistency
peter_paul_data$month <- factor(
  peter_paul_data$month,
  levels = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"),
  labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
)

# Here I create  three different box plots for Temperature, TP (Total Phosphorus), and TN (Total Nitrog
PeterPaul_Chem_Temp <- ggplot(peter_paul_data, aes(x = month, y = temperature_C, color = lakename)) +
  geom_boxplot() +
  scale_x_discrete(drop = FALSE) +
  theme_minimal() +
  theme(legend.position = "top") +
  theme(panel.grid.major = element_line(color = "grey", linetype = "solid"))

PeterPaul_Chem_Temp
```

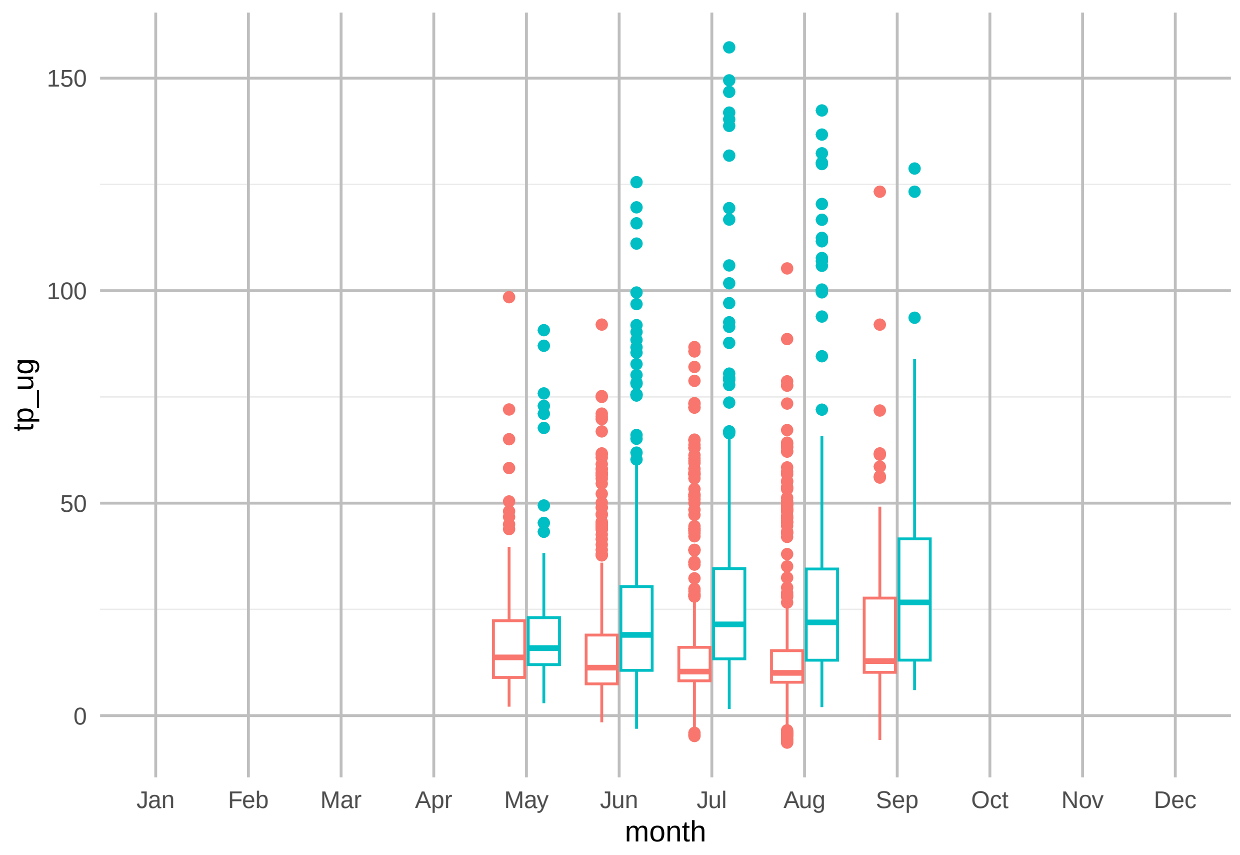
```
## Warning: Removed 3566 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



```
PeterPaul_Chem_TP <- ggplot(peter_paul_data, aes(x = month, y = tp_ug, color = lakename)) +
  geom_boxplot() +
  scale_x_discrete(drop = FALSE) +
  theme_minimal() +
  theme(legend.position = "none") +
  theme(panel.grid.major = element_line(color = "grey", linetype = "solid"))

PeterPaul_Chem_TP
```

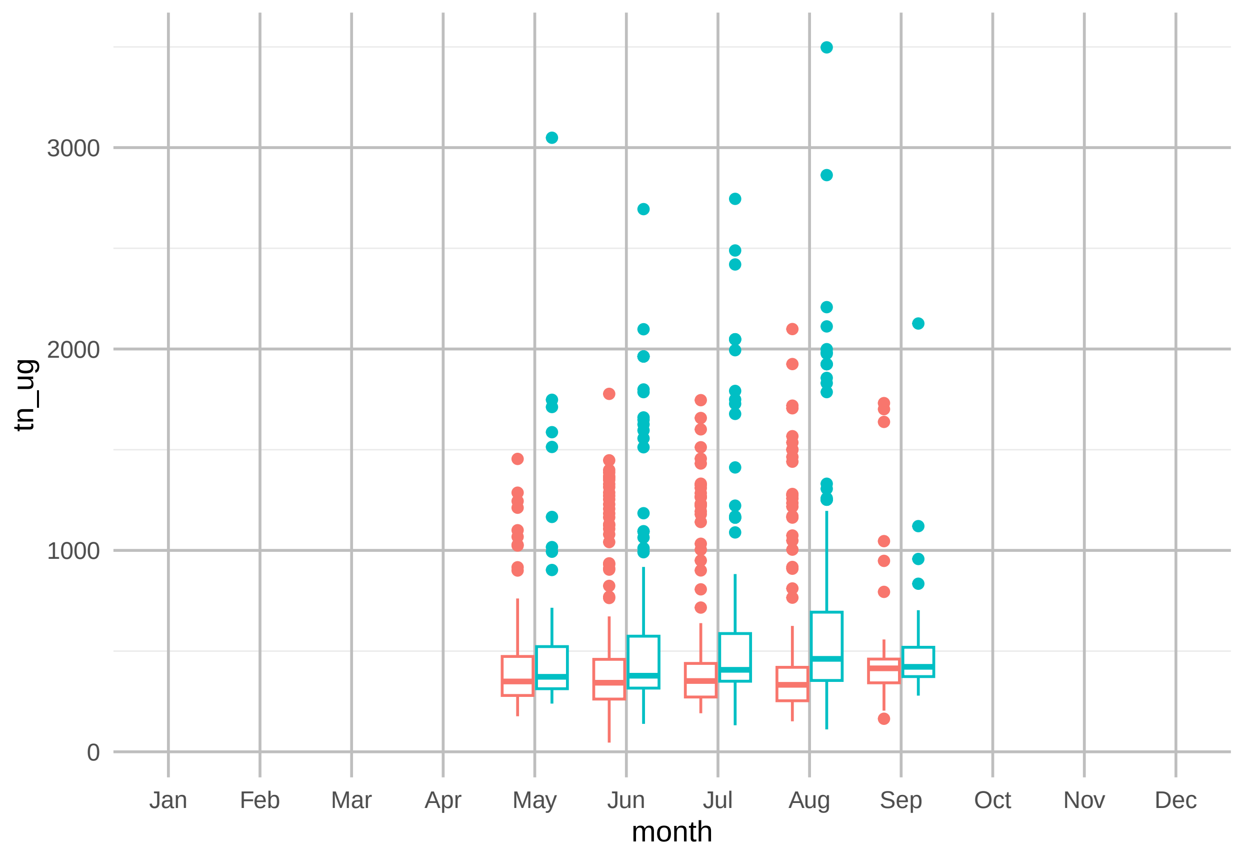
```
## Warning: Removed 20729 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```



```r
PeterPaul_Chem_TN <- ggplot(peter_paul_data, aes(x = month, y = tn_ug, color = lakename)) +
  geom_boxplot() +
  scale_x_discrete(drop = FALSE) +
  theme_minimal() +
  theme(legend.position = "none") +
  theme(panel.grid.major = element_line(color = "grey", linetype = "solid"))

PeterPaul_Chem_TN
```

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



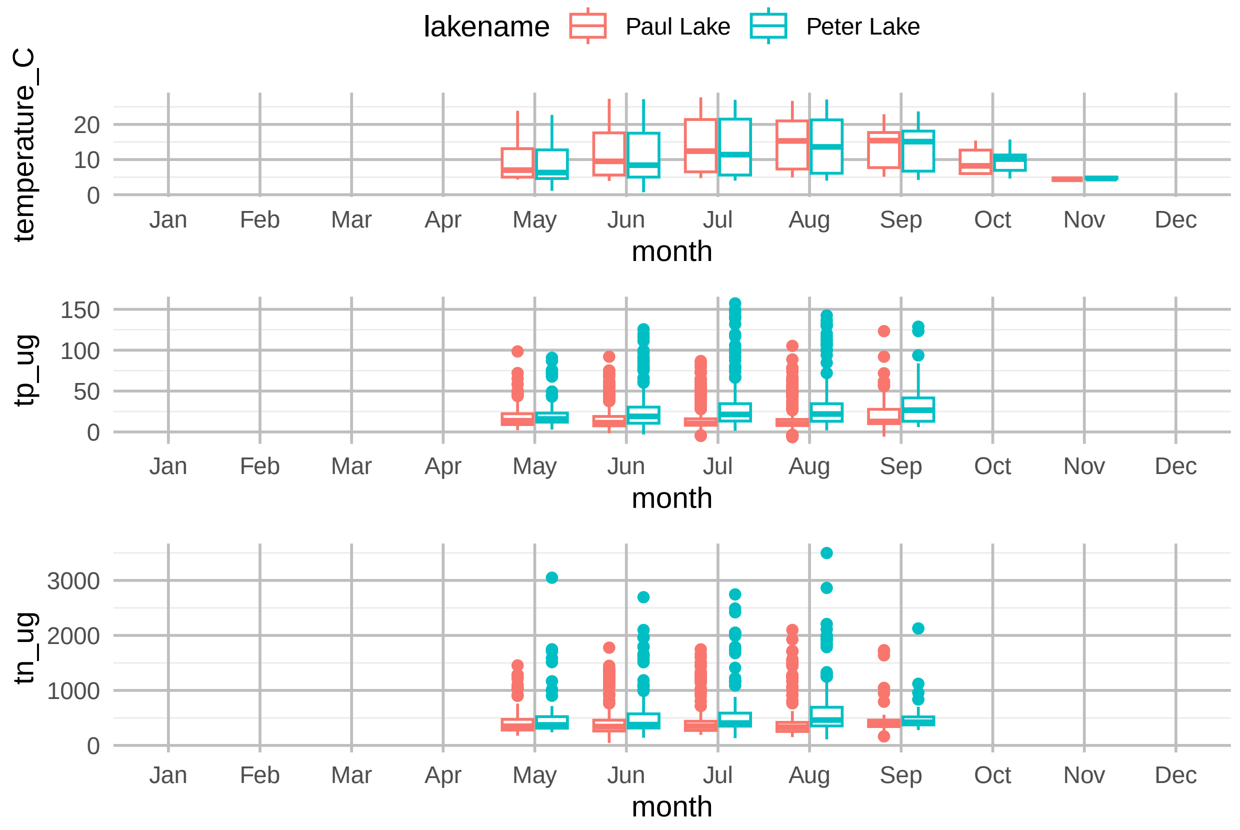
```r
# Here I plott all three boxplots together in one figure
combined_plot <- plot_grid(PeterPaul_Chem_Temp, PeterPaul_Chem_TP, PeterPaul_Chem_TN, nrow = 3, align =
```

```
## 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()‘).
```

```r
print(combined_plot)
```



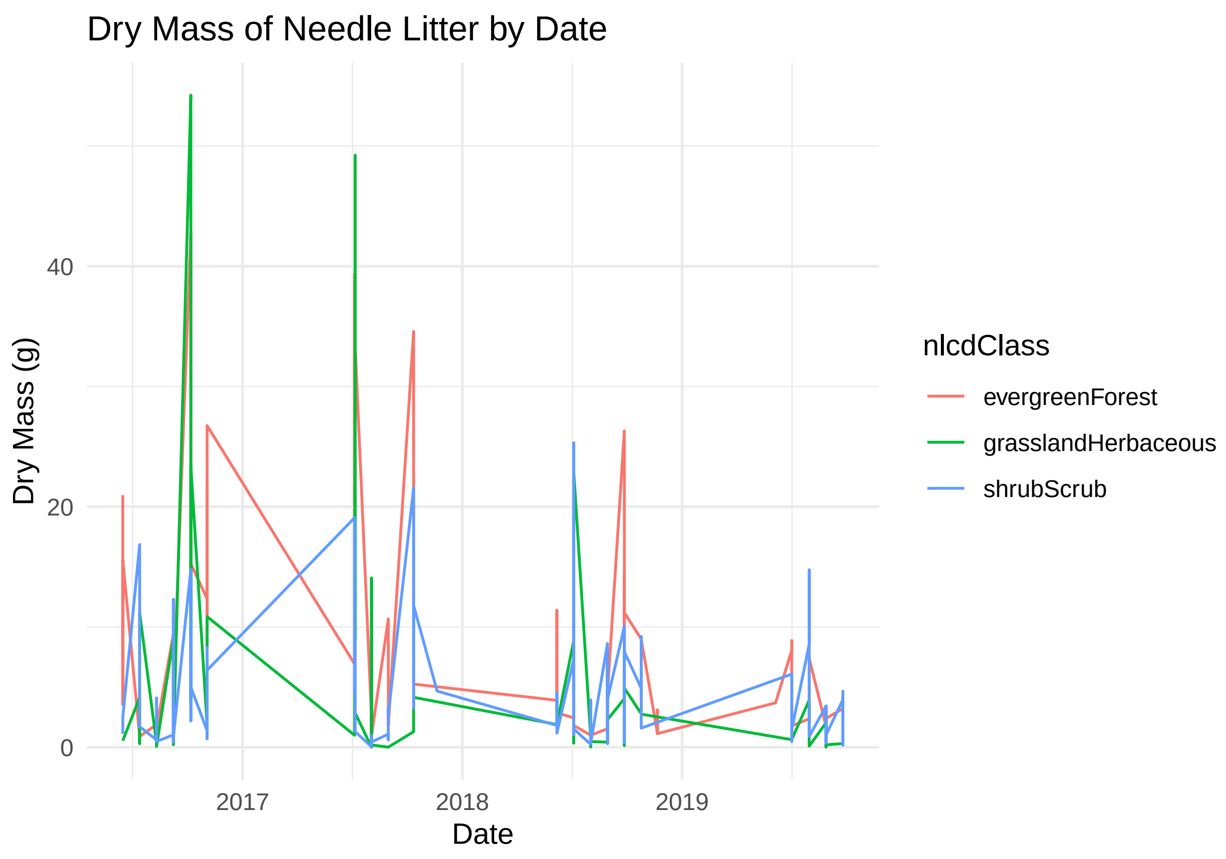
Question: What do you observe about the variables of interest over seasons and between lakes?

> Answer: The data shows that both Paul Lake and Peter Lake have similar temperature patterns, with temperatures rising in the summer and peaking around 20°C from June to August. However, Peter Lake consistently has higher levels of nutrients, particularly total phosphorus and total nitrogen, compared to Paul Lake. These differences are most noticeable during the warmer months (May to September), where Peter Lake's nutrient levels are significantly higher, especially in the summer.

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.

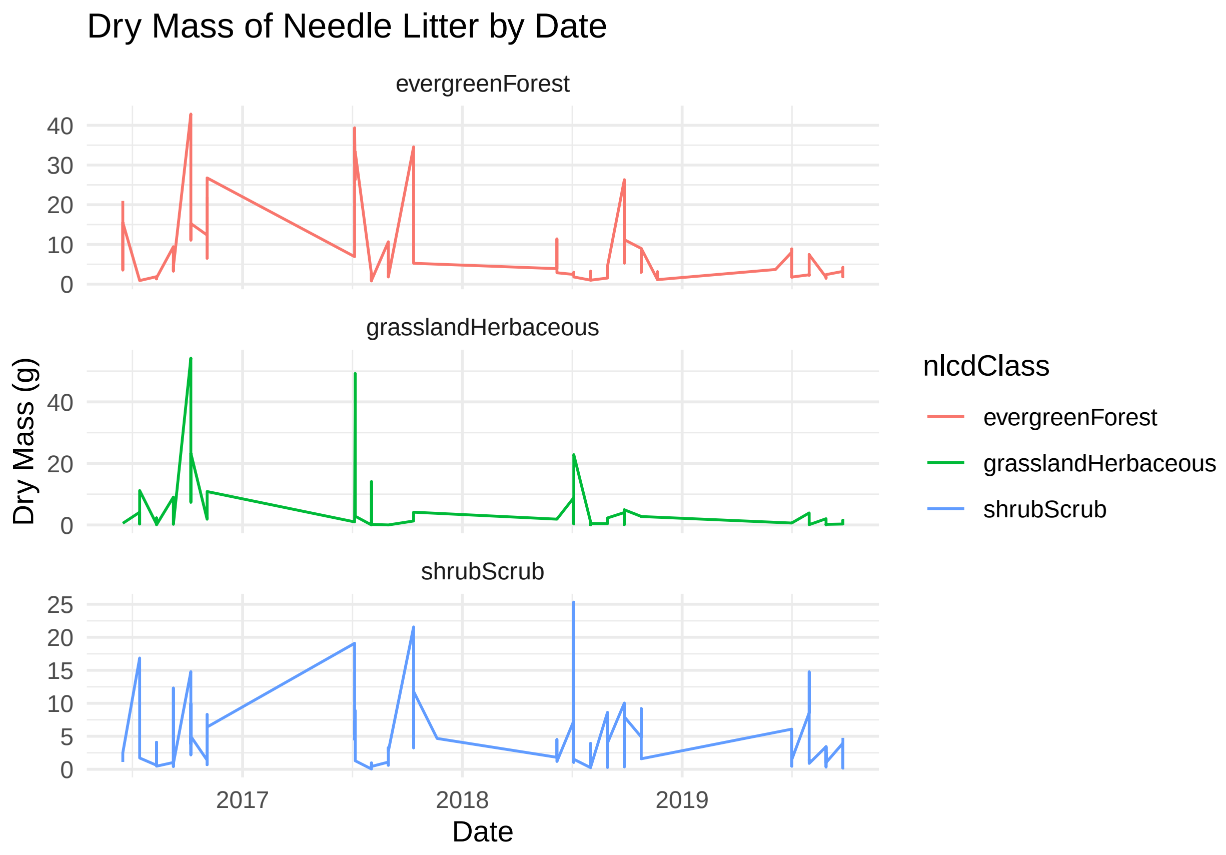
```r
#6 Here I filtered the dataset to include only the "Needles" functional group
needles_subset <- niwot_litter_data %>%
  filter(functionalGroup == "Needles")

# Then I create the plot with dry mass of needle litter by date, separated by NLCD class using color
ggplot(needles_subset, aes(x = collectDate, y = dryMass, color = nlcdClass)) +
  geom_line() +
  labs(
    title = "Dry Mass of Needle Litter by Date",
    x = "Date",
    y = "Dry Mass (g)"
  ) +
  theme_minimal()
```



```
#7 Here I Filter the dataset to include only the "Needles" functional group
needles_subset <- niwot_litter_data %>%
  filter(functionalGroup == "Needles")

# Here I create the plot with facets
ggplot(needles_subset, aes(x = collectDate, y = dryMass)) +
  geom_line(aes(color = nlcdClass)) +  # Use color to differentiate NLCD classes
  labs(
    title = "Dry Mass of Needle Litter by Date",
    x = "Date",
    y = "Dry Mass (g)"
  ) +
  facet_wrap(~ nlcdClass, scales = "free_y", ncol = 1) +  # then Separate it by NLCD class into facets
  theme_minimal()
```



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

Answer: Visually, second plot (No. 7) is usually more effective. The color-coded plot (No. 6) shows all the data at once, making it easier to see trends quickly. However, if there are many NLCD classes, the lines can get crowded and hard to read. On the other hand, Plot No. 7, which uses separate sections for each class, makes it clearer and easier to understand the trends without overlapping lines. This way, each class can be viewed individually, helping to highlight differences in dry mass over time. While Plot No. 6 is faster for quick comparisons, Plot No. 7 is better for a clear and detailed view of the data.