Assignment 5: Data Visualization

Mazhar Bhuyan

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/EDA_Spring2025
```

```
library(cowplot)
```

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

```
#2
file_path_1 <- here("Data",
                    "Processed_KEY",
                    "NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv")
file_path_2 <- here("Data",
                    "Processed_KEY",
                    "NEON_NIWO_Litter_mass_trap_Processed.csv")

Nutrients <- read.csv(file_path_1,
                          stringsAsFactors = TRUE)

Litter <- read.csv(file_path_2,
                          stringsAsFactors = TRUE)
```

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_minimal(base_size = 14) +
  theme(
    plot.background = element_rect(fill = "white",
                                   color = NA),
    plot.margin = margin(10, 10, 10, 10, "pt"),
    plot.title = element_text(family = "Courier",
                              face = "bold",
                              color = "brown",
                              size = 14,
                              hjust = 0.5),  # Bold centered title
    axis.title = element_text(face = "bold", # Bold axis labels
```

```
                                size = 14),
    axis.text = element_text(family = "Courier",
                             color = "darkblue", # Blue axis text
                              size = 12),
    legend.background = element_rect(fill = "gray", # Light gray legend background
                                     color = NA),
    legend.key = element_rect(fill = "ivory",
                              color = "red"))
#,  # Black legend keys with gray border
    legend.position = "bottom"
#)
theme_set((my_theme))# Setting  the global 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()`).

```
#4
class(Nutrients$lakename)
```

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

```
#Remove Extreme Values
po4_limit <- quantile(Nutrients$po4, 0.95, na.rm = TRUE) #includes up to 95th percentile
tp_limit <- quantile(Nutrients$tp_ug, 0.95, na.rm = TRUE)

# scatter plot with regression lines
ggplot(Nutrients,
       aes(x = po4,
           y = tp_ug,
           color = lakename)) +
  geom_point(alpha = 1,
             size = 2,
             shape = 6) +  # Adding points with transparency
  geom_smooth(method = "lm",
              linetype = "solid",
              size = 0.75) +  # Adding linear regression lines
  scale_x_continuous(limits = c(0, po4_limit),
                     expand = expansion(mult = 0.02)) +  # Adjust x-axis
  scale_y_continuous(limits = c(0, tp_limit),
                     expand = expansion(mult = 0.02)) +  # Adjust y-axis
  labs(
    title = "Total Phosphorus vs. Phosphate in Peter & Paul Lakes",
    subtitle = "Best-fit lines applied for each lake",
    x = "Phosphate (po4)",
    y = "Total Phosphorus (tp_ug)",
```

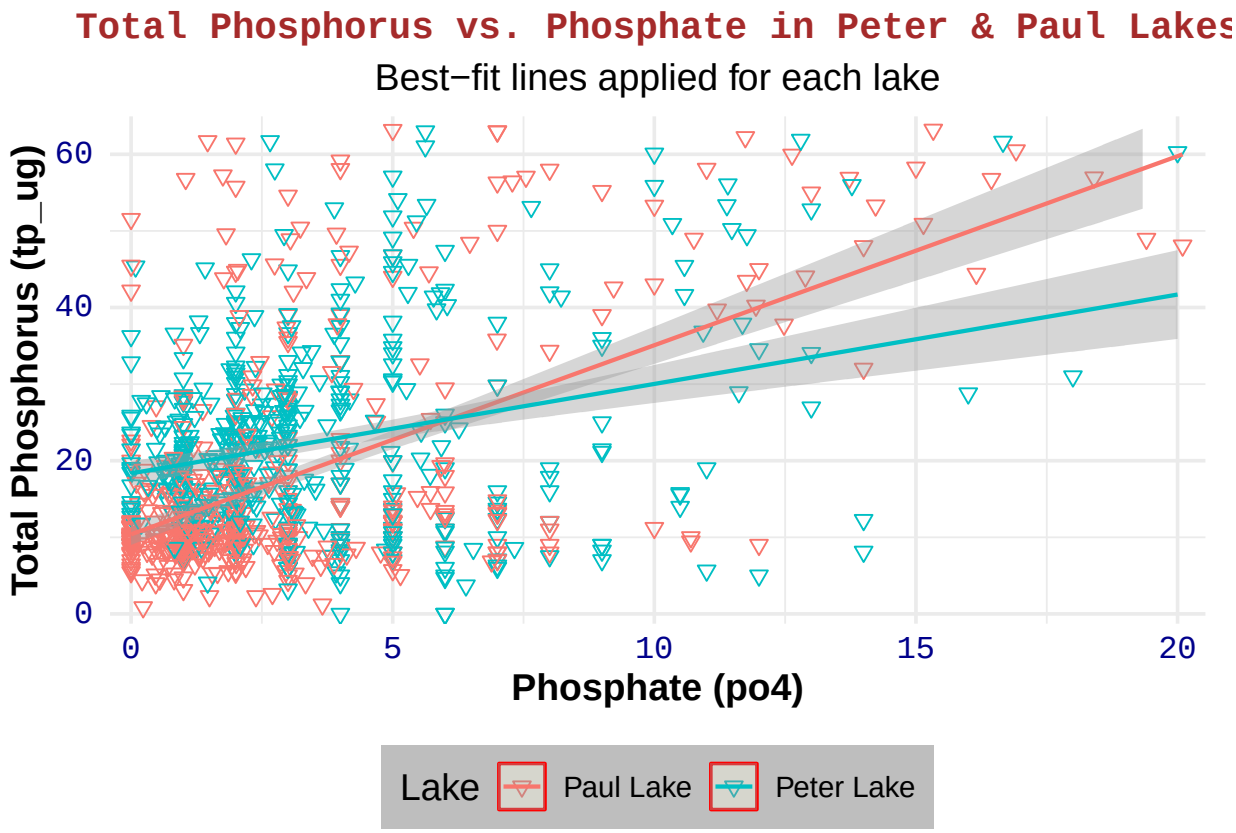
```
    color = "Lake"
  ) +
  my_theme + # Apply custom theme
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),  # Center title
    plot.subtitle = element_text(hjust = 0.5),
    legend.position = "bottom" # Center subtitle
  )
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

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

```
## 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.

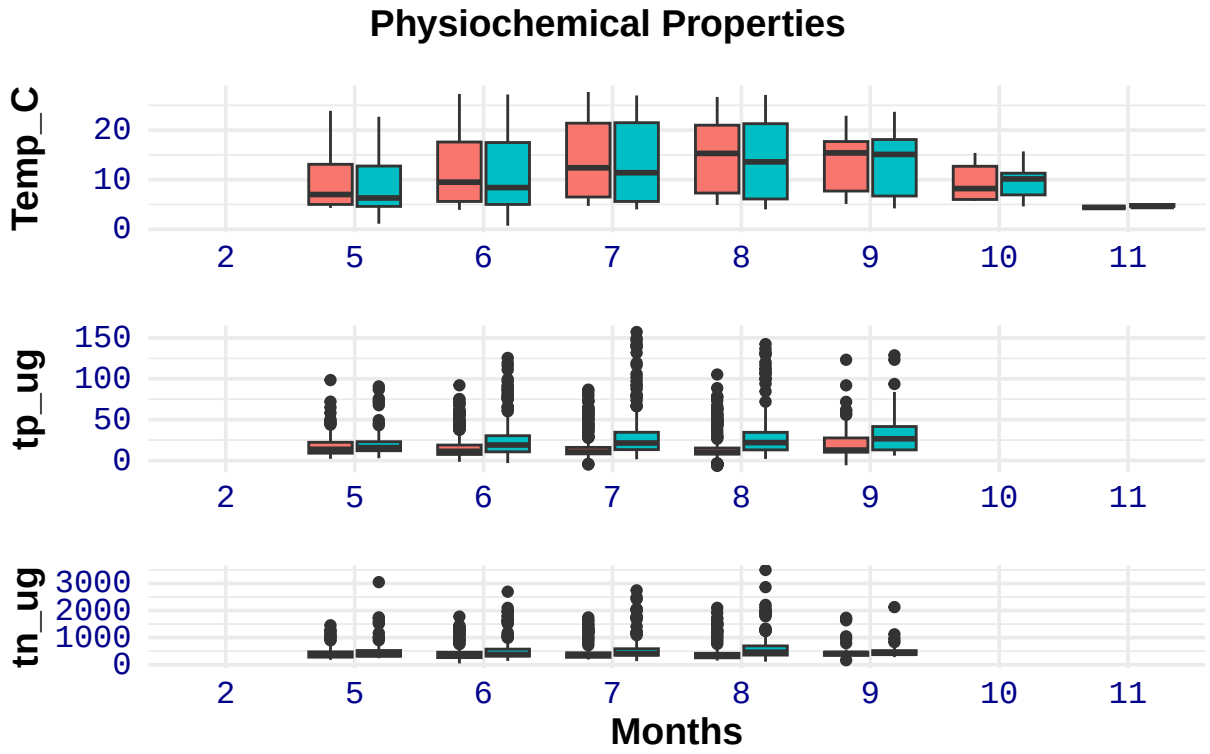
Tips: * Recall the discussion on factors in the lab section as it may be helpful here. * Setting an axis title in your theme to `element_blank()` removes the axis title (useful when multiple, aligned plots use the same axis values) * Setting a legend's position to "none" will remove the legend from a plot. * Individual plots can have different sizes when combined using `cowplot`.

```
#5

# Ensure 'month' is a factor for proper ordering in the boxplots
Nutrients$month <- factor(Nutrients$month, levels = as.character(1:12))
class(Nutrients$month)
```

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

```
# Create Temperature Boxplot
temp_plot <- ggplot(Nutrients,
                    aes(x = month,
                        y = temperature_C,
                        fill = lakename)) +
  geom_boxplot() +
  labs(x = "",
       y = "Temp_C",
       fill = "Lake") +
  theme(legend.position = "none",
        axis.title.x = element_blank())
#print(temp_plot)

# Create Total Phosphorus (TP) Boxplot
tp_plot <- ggplot(Nutrients,
                  aes(x = month,
                      y = tp_ug,
                      fill = lakename)) +
  geom_boxplot() +
  labs(x = "",
       y = "tp_ug",
       fill = "Lake") +
  theme(legend.position = "none",
        axis.title.x = element_blank())  # Align x-axis and remove redundant title
#print(tp_plot)

# Create Total Nitrogen (TN) Boxplot
tn_plot <- ggplot(Nutrients,
                  aes(x = month,
                      y = tn_ug,
                      fill = lakename)) +
  geom_boxplot() +
  labs(x = "Months",
       y = "tn_ug",
       fill = "Lake") +
  theme(legend.position = "none")
```

```
#print(tn_plot)

# Combine plots using cowplot
final_plot <- plot_grid(temp_plot,
                        tp_plot,
                        tn_plot,
                        ncol = 1, align = "v",
                        rel_heights = c(1, 1, 1))  # Ensure equal sizes


# Display the final plot
title <- ggdraw() +
          draw_label("Physiochemical Properties", fontface='bold')
plot_grid(title,
          final_plot,
          nrow=3,
          ncol=1,
          rel_heights = c(0.1,1))
```



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

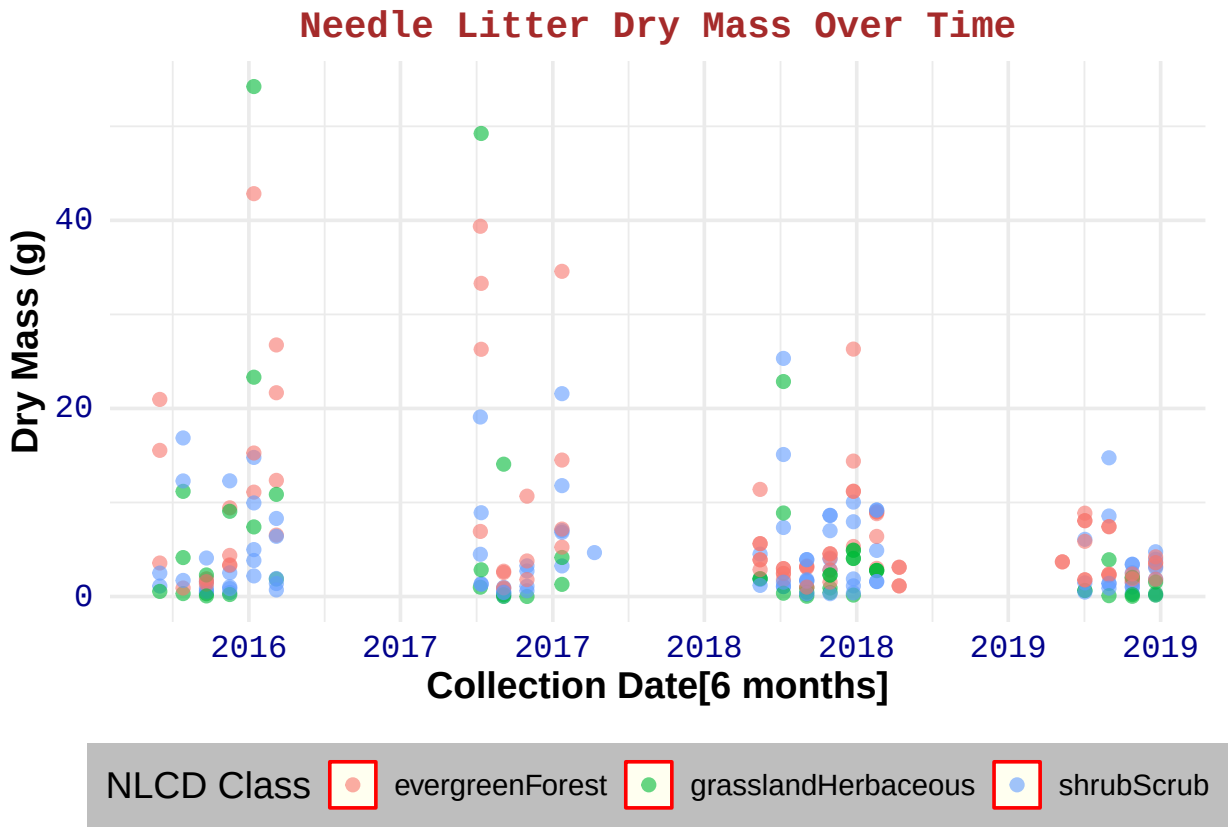
Answer: Temperature increases from May to August (spring to summer), peaking in July and August, and then declines from September to November (fall), with no major differences between lakes. Total phosphorus (TP) and total nitrogen (TN) remain relatively stable but show slight increases in late summer (August–September). Peter Lake has slightly higher TP and TN levels

than Paul Lake. Both nutrients exhibit high variability and outliers, likely due to runoff or biological activity, especially in spring and summer when external inputs may be more frequent.

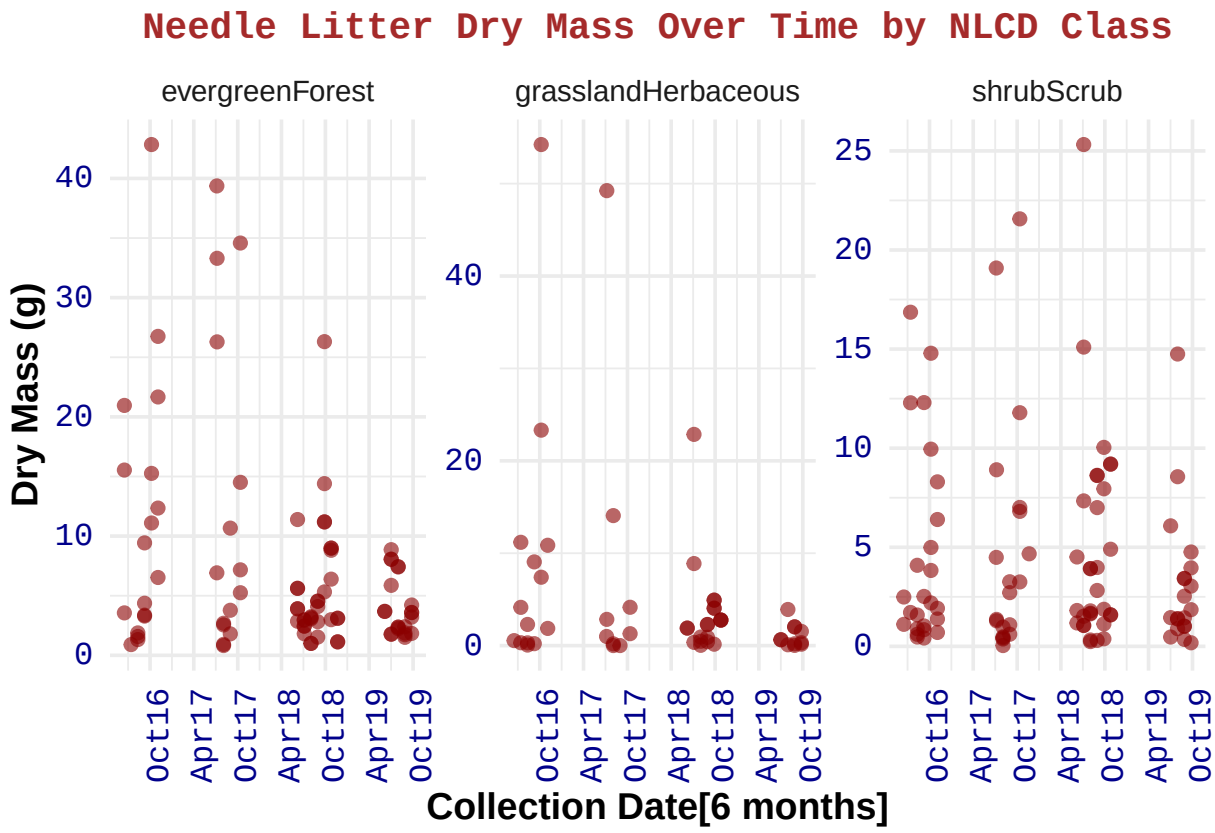
6. [Niwot Ridge] Plot a subset of the litter dataset by displaying only the "Needles" functional group. Plot the dry mass of needle litter by date and separate by NLCD class with a color aesthetic. (no need to adjust the name of each land use)

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

```
#6
Litter$collectDate = ymd(Litter$collectDate)
needles_subset <- Litter %>%
  filter(functionalGroup == "Needles")

ggplot(needles_subset, aes(x = collectDate, y = dryMass, color = nlcdClass)) +
  geom_point(alpha = 0.6, size = 2) +  # Scatter plot with transparency
  scale_x_date(date_breaks = "6 months", date_labels = "%Y")+
  labs(title = "Needle Litter Dry Mass Over Time",
       x = "Collection Date[6 months]",
       y = "Dry Mass (g)",
       color = "NLCD Class") +
  my_theme +
  theme(legend.position = "bottom")+
    theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



```r
#7
ggplot(needles_subset, aes(x = collectDate, y = dryMass)) +
  geom_point(alpha = 0.6, size = 2, color = "darkred") +  # Uniform color for clarity
  scale_x_date(date_breaks = "6 months", date_labels = "%b%y") +  # Improve date formatting
  labs(title = "Needle Litter Dry Mass Over Time by NLCD Class",
       x = "Collection Date[6 months]",
       y = "Dry Mass (g)") +
  facet_wrap(~nlcdClass, scales = "free_y") +  # Create facets by NLCD class
  my_theme +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.text.x = element_text(angle = 90, hjust = 1)
  )
```



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

Answer: Plot 7 is more effective because it reduces overlap by separating NLCD classes into facets, making individual trends clearer. It also allows independent y-axis scaling, which is beneficial since different land-use classes may have varying dry mass distributions. In contrast, Plot 6 forces all data into the same y-axis scale, which can obscure differences. While Plot 6 is useful for seeing an overall trend, Plot 7 is better for comparing trends across land-use types separately.