Assignment 5: Data Visualization

Hanbin Lyu

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
#load packages
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
```

```r
library(lubridate)
library(here)
```

```
## here() starts at /home/guest/EDE_Fall2024
```

```r
library(cowplot)
```

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

```r
#check working directory
getwd()
```

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

```r
#read cvs files
lake = read.csv(
  "Data/Processed/Processed_KEY/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv",
  stringsAsFactors = TRUE)
litter = read.csv(
  "Data/Processed/Processed_KEY/NEON_NIWO_Litter_mass_trap_Processed.csv",
  stringsAsFactors = TRUE)

#2
# Check the structure of the data set, make sure dates are read as date format
str(lake)
```

```
## 'data.frame':    23008 obs. of  15 variables:
##  $ lakename       : Factor w/ 2 levels "Paul Lake","Peter Lake": 1 1 1 1 1 1 1 1 1 1 ...
##  $ year4          : int  1984 1984 1984 1984 1984 1984 1984 1984 1984 1984 ...
##  $ daynum         : int  148 148 148 148 148 148 148 148 148 148 ...
##  $ month          : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ sampledate     : Factor w/ 1103 levels "1984-05-27","1984-05-28",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ depth          : num  0 0.25 0.5 0.75 1 1.5 2 3 4 5 ...
##  $ temperature_C  : num  14.5 NA NA NA 14.5 NA 14.2 11 7 6.1 ...
##  $ dissolvedOxygen: num  9.5 NA NA NA 8.8 NA 8.6 11.5 11.9 2.5 ...
##  $ irradianceWater: num  1750 1550 1150 975 870 610 420 220 100 34 ...
##  $ irradianceDeck : num  1620 1620 1620 1620 1620 1620 1620 1620 1620 1620 ...
##  $ tn_ug          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ tp_ug          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ nh34           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ no23           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ po4            : num  NA NA NA NA NA NA NA NA NA NA ...
```

```r
str(litter)
```

```
## 'data.frame':    1692 obs. of  13 variables:
##  $ plotID          : Factor w/ 12 levels "NIWO_040","NIWO_041",..: 9 8 9 11 7 7 4 4 4 4 ...
##  $ trapID          : Factor w/ 15 levels "NIWO_040_139",..: 11 10 11 13 9 9 5 5 5 5 ...
##  $ collectDate     : Factor w/ 24 levels "2016-06-16","2016-07-14",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ functionalGroup : Factor w/ 8 levels "Flowers","Leaves",..: 6 5 8 6 4 2 2 6 7 8 ...
##  $ dryMass         : num  0 0.27 0.12 0 1.11 0 0 0 0.07 0.02 ...
##  $ qaDryMass       : Factor w/ 2 levels "N","Y": 1 1 1 1 2 1 1 1 1 1 ...
##  $ subplotID       : int  31 41 31 32 32 32 40 40 40 40 ...
##  $ decimalLatitude : num  40.1 40 40.1 40 40 ...
##  $ decimalLongitude: num  -106 -106 -106 -106 -106 ...
##  $ elevation       : num  3477 3413 3477 3373 3446 ...
##  $ nlcdClass       : Factor w/ 3 levels "evergreenForest",..: 3 1 3 1 3 3 2 2 2 2 ...
##  $ plotType        : Factor w/ 1 level "tower": 1 1 1 1 1 1 1 1 1 1 ...
##  $ geodeticDatum   : Factor w/ 1 level "WGS84": 1 1 1 1 1 1 1 1 1 1 ...
```

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)

#define my own theme
my_theme = theme(
    line = element_line(color = "darkseagreen4", linewidth = 2,
                        linetype = "solid", lineend = "round"),
    rect = element_rect(fill = "honeydew",color = "darkolivegreen4",
                        linewidth = 1, linetype = "solid"),
    text = element_text(family = "serif", face = "plain",
                        color = "darkolivegreen",size = 12, hjust = 0.5,
                        vjust = 0.5, angle = 0, lineheight = 1.5),

    # Modified inheritance structure of text element
    plot.title = element_text(family = "serif", face = "bold",
                              color = "darkolivegreen", size = 16, hjust = 0.5,
                              vjust = 0.5, angle = 0, lineheight = 1.5),
    axis.title.x = element_text(family = "serif", face = "plain",
                              color = "darkolivegreen", size = 12, hjust = 0.5,
                              vjust = 0.5, angle = 0, lineheight = 1.5),
    axis.title.y = element_text(family = "serif", face = "plain",
                              color = "darkolivegreen", size = 12, hjust = 0.5,
                              vjust = 0.5, angle = 0, lineheight = 1.5),
    axis.text = element_text(family = "serif", face = "plain",
                             color = "darkolivegreen", size = 12, hjust = 0.5,
                             vjust = 0.5, angle = 0, lineheight = 1.5),

    # Modified inheritance structure of line element
```

```
    axis.ticks = element_blank(),
    panel.grid.major =  element_line(color = "gray85", linewidth = 0.5,
                                     linetype = "solid", lineend = "square"),
    panel.grid.minor =  element_blank(),

    # Modified inheritance structure of rect element
    plot.background = element_rect(fill = "ivory"),
    panel.background = element_rect(fill = "lightyellow"),
    legend.key = element_rect(fill = "honeydew", linewidth = 0.5),

    # Modifiying legend.position
    legend.position = 'right',

    complete = TRUE
    )
```

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
#check the distribution of data
summary(lake$po4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -0.233   1.000   2.324   5.919   5.000 373.836   21822
```

```
summary(lake$tp_ug)
```

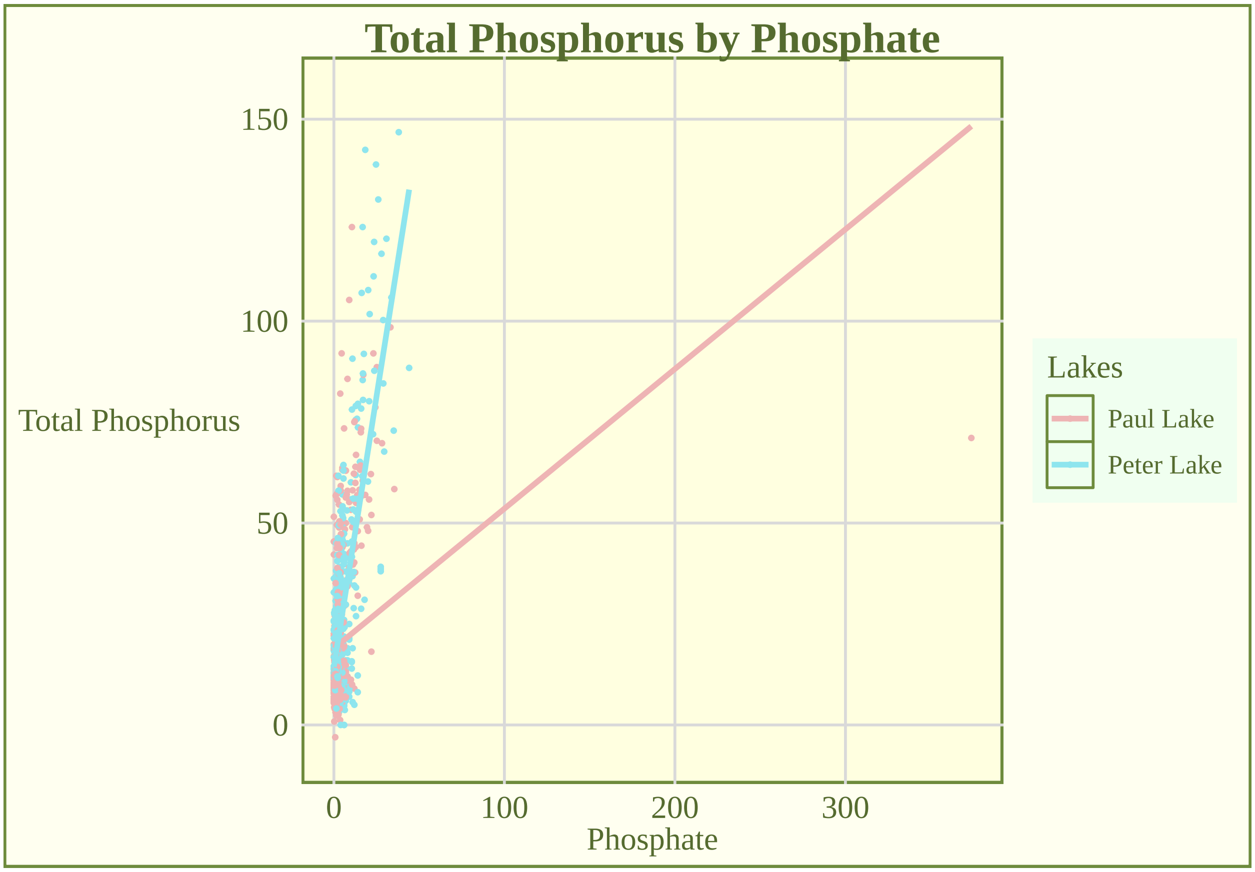
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -6.349   9.194  14.401  22.159  27.746 157.250   20729
```

```
#find the extreme value by looking at the plot
ggplot(lake, aes(x = po4, y = tp_ug, color = lakename,)) +
  #set x and y axis, separate lakes with different color
  geom_point(size = 0.5) +
  #add points for each lake
  geom_smooth(method = "lm", se = FALSE) +
  #add a best fit line
  labs(title = "Total Phosphorus by Phosphate",
       x = "Phosphate",
       y = "Total Phosphorus",
       color = "Lakes") +
  #add title, axis titles, and legend
  scale_color_manual(values = c("Peter Lake" = "cadetblue2", "Paul Lake" = "rosybrown2")) +
   my_theme
```

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

## Warning: Removed 21946 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 21946 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

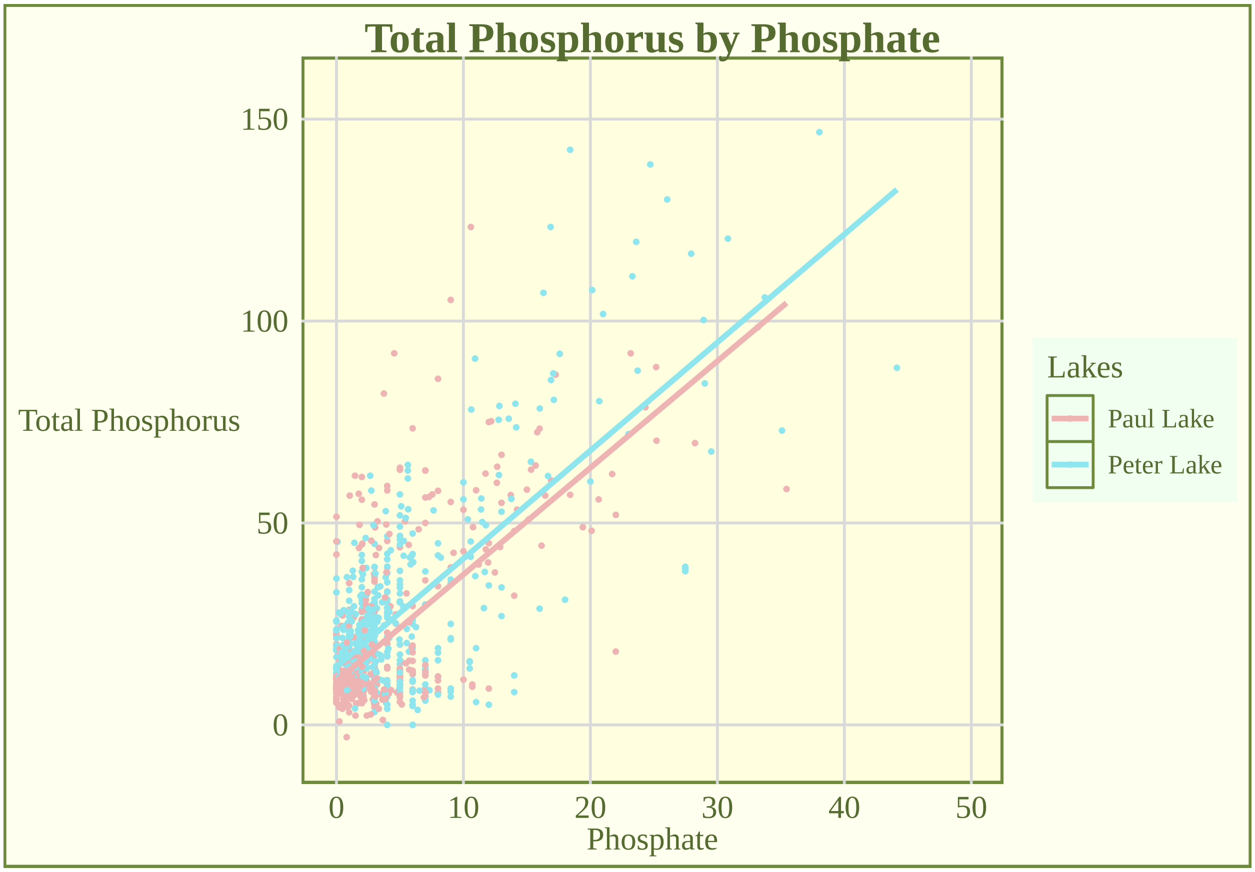



```r
#get ride of extreme value
ggplot(lake, aes(x = po4, y = tp_ug, color = lakename,)) +
  #set x and y axis, separate lakes with different color
  geom_point(size = 0.5) +
  #add points for each lake
  geom_smooth(method = "lm", se = FALSE) +
  #add a best fit line
  xlim(-0.233, 50) +
  #adjust x-axis limits to hide extreme values
  labs(title = "Total Phosphorus by Phosphate",
       x = "Phosphate",
       y = "Total Phosphorus",
       color = "Lakes") +
  #add title, axis titles, and legend
  scale_color_manual(values = c("Peter Lake" = "cadetblue2", "Paul Lake" = "rosybrown2")) +
   my_theme
```

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

## Warning: Removed 21947 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 21947 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

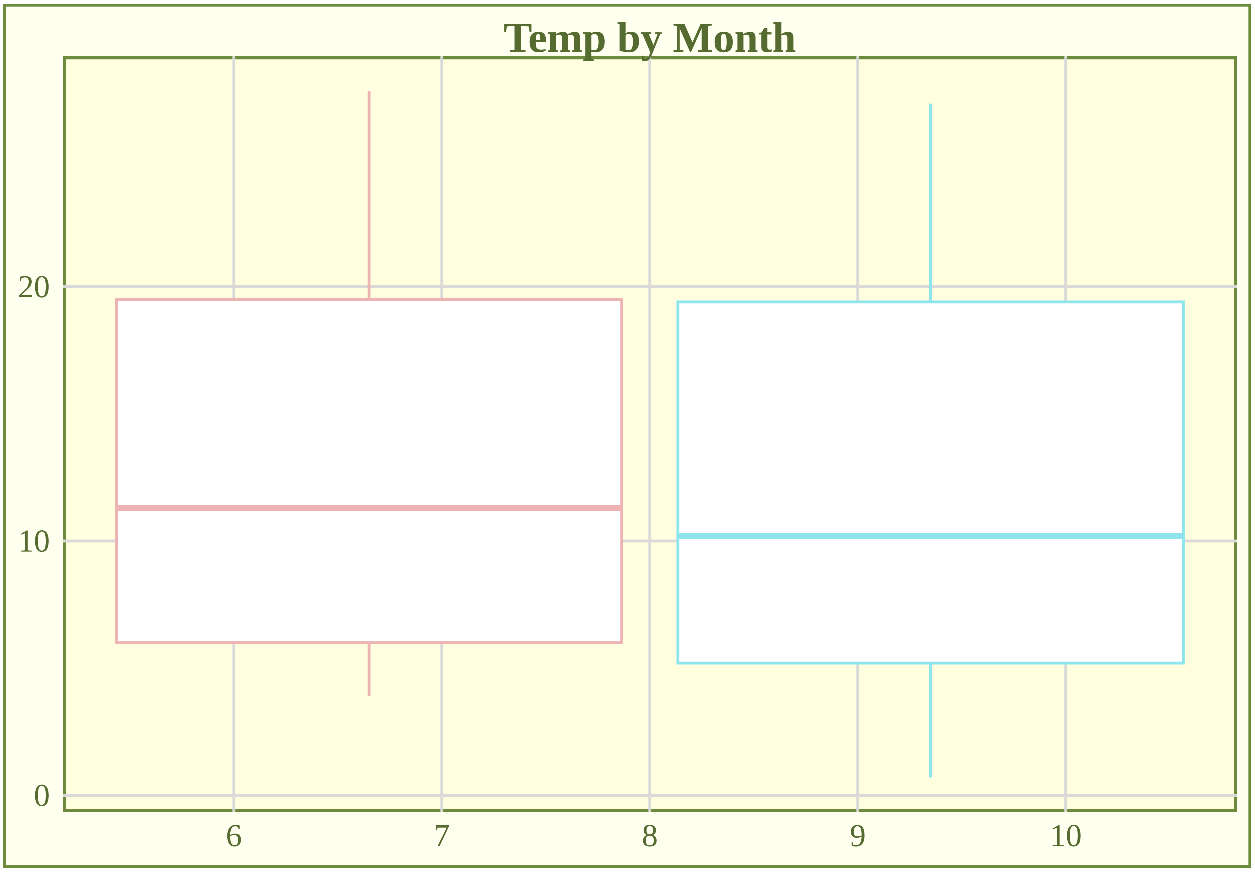


5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and (c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

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`.

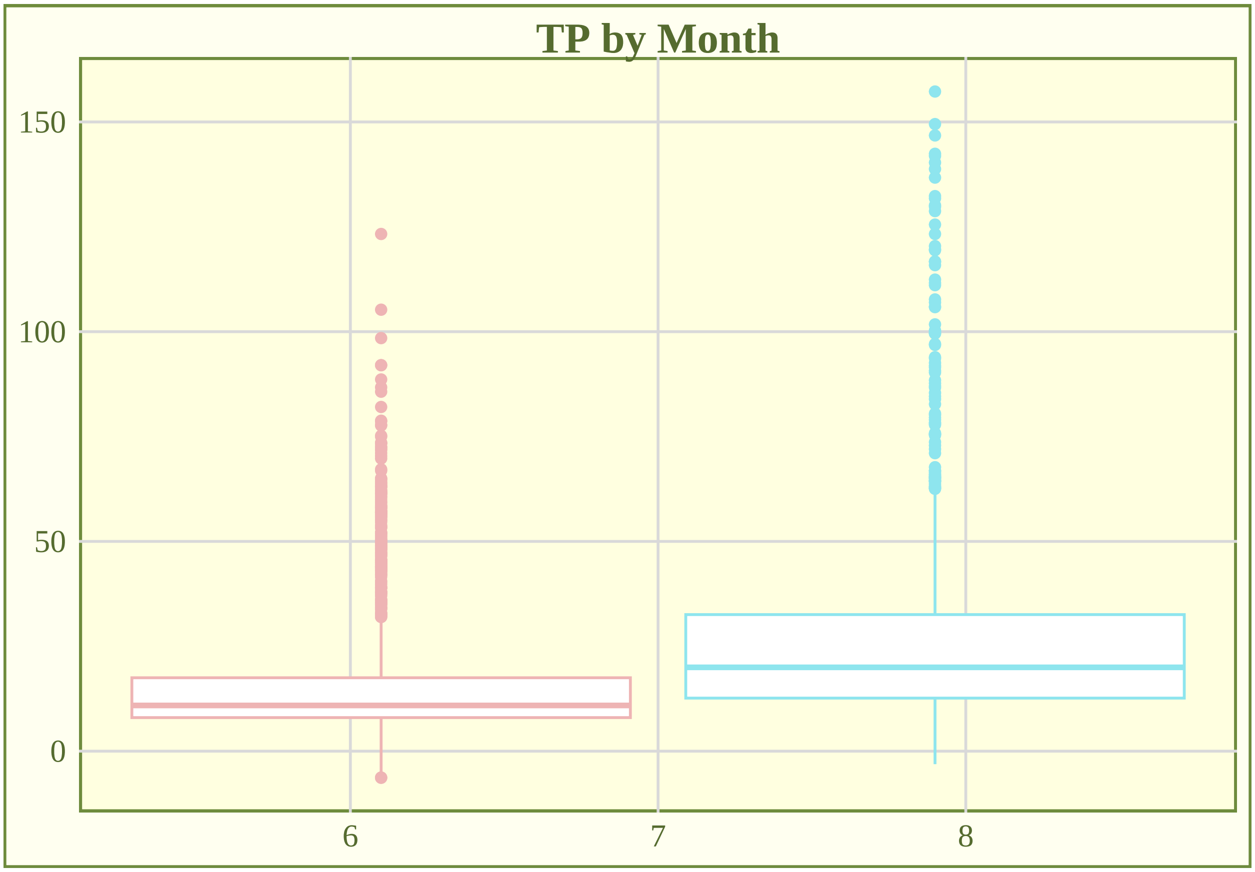
```r
#5
#make box plot of temperature by month
temperature = ggplot(lake, aes(x = month, y = temperature_C, color = lakename)) +
  geom_boxplot() +
  labs(title = "Temp by Month", x = "Months", y = "Temperature", color = "Lakes") +
  scale_color_manual(values = c("Peter Lake" = "cadetblue2", "Paul Lake" = "rosybrown2")) +
  my_theme +
  theme(axis.title = element_blank(), legend.position = "none")
temperature
```

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



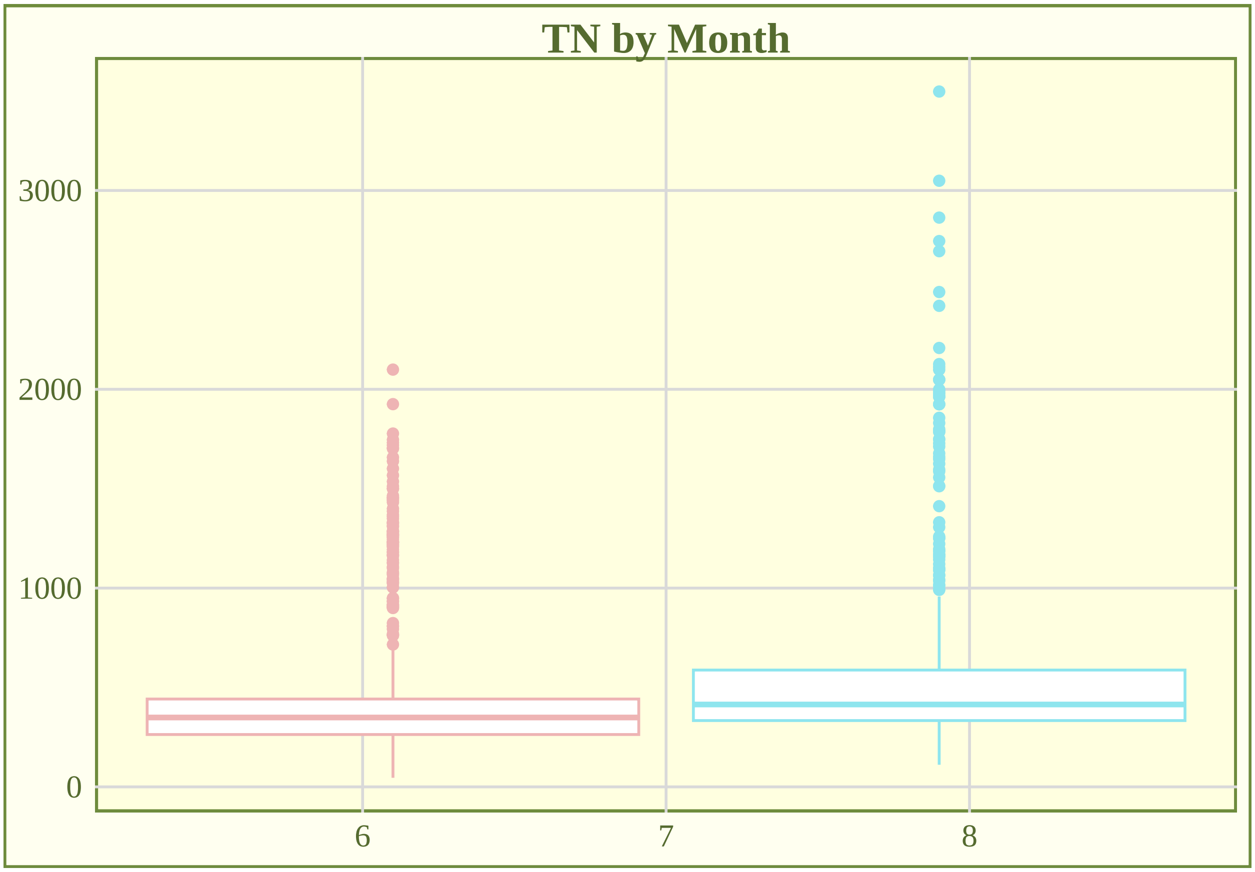

```r
#make box plot of TP by month
TP = ggplot(lake, aes(x = month, y = tp_ug, color = lakename)) +
  geom_boxplot() +
  labs(title = "TP by Month", x = "Months", y = "TP", color = "Lakes") +
  scale_color_manual(values = c("Peter Lake" = "cadetblue2", "Paul Lake" = "rosybrown2")) +
  my_theme +
  theme(axis.title = element_blank(), legend.position = "none")
TP
```

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




```r
#make box plot of TN by month
TN = ggplot(lake, aes(x = month, y = tn_ug, color = lakename)) +
  geom_boxplot() +
  labs(title = "TN by Month", x = "Months", y = "TN", color = "Lakes") +
  scale_color_manual(values = c("Peter Lake" = "cadetblue2", "Paul Lake" = "rosybrown2")) +
  my_theme +
  theme(axis.title = element_blank(), legend.position = "none")
TN
```

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



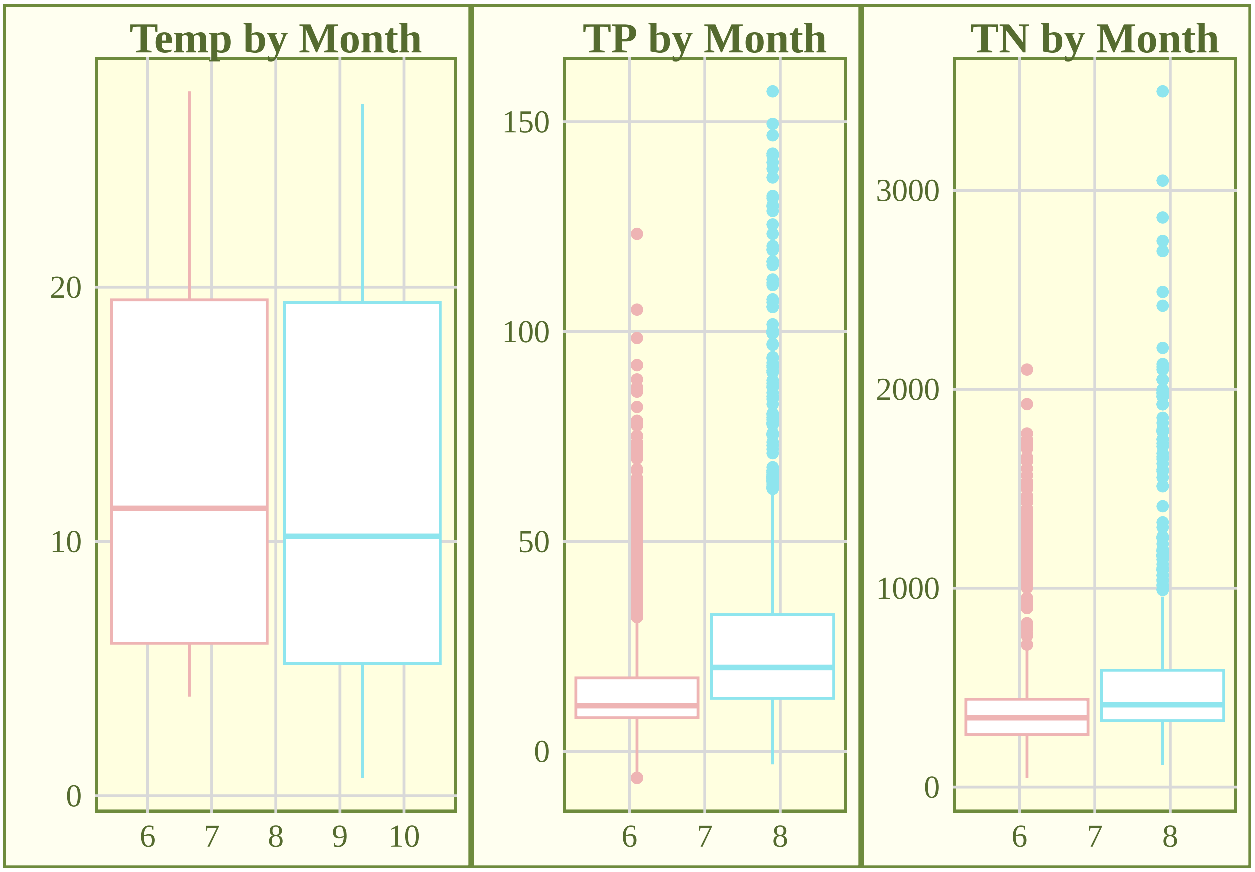

```r
#combine 3 plots
cowplot = plot_grid(temperature, TP, TN, ncol = 3, align = 'v', rel_widths = c(1.2, 1, 1))
```

```
## 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
cowplot
```



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

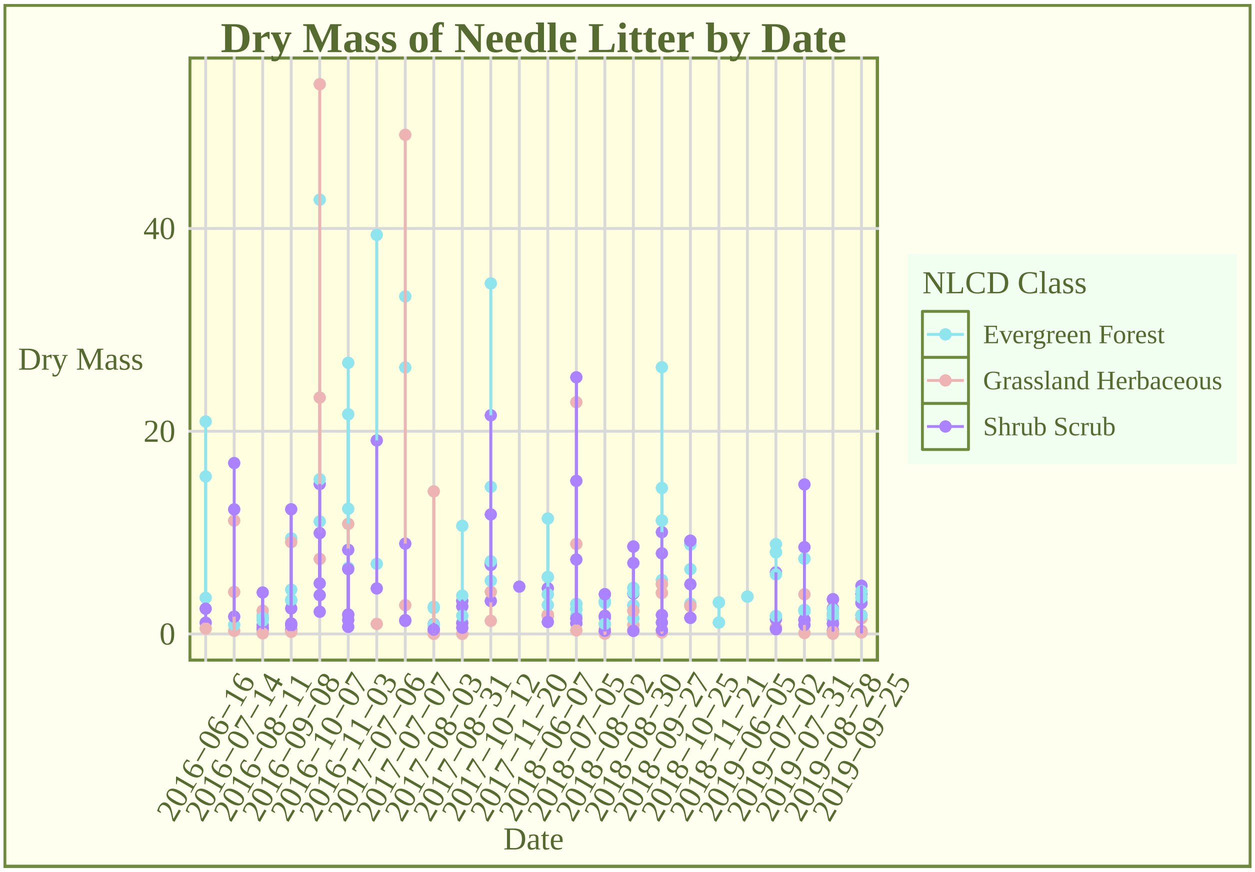
> Answer: peter lake has higher TP and TN values overall.

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.

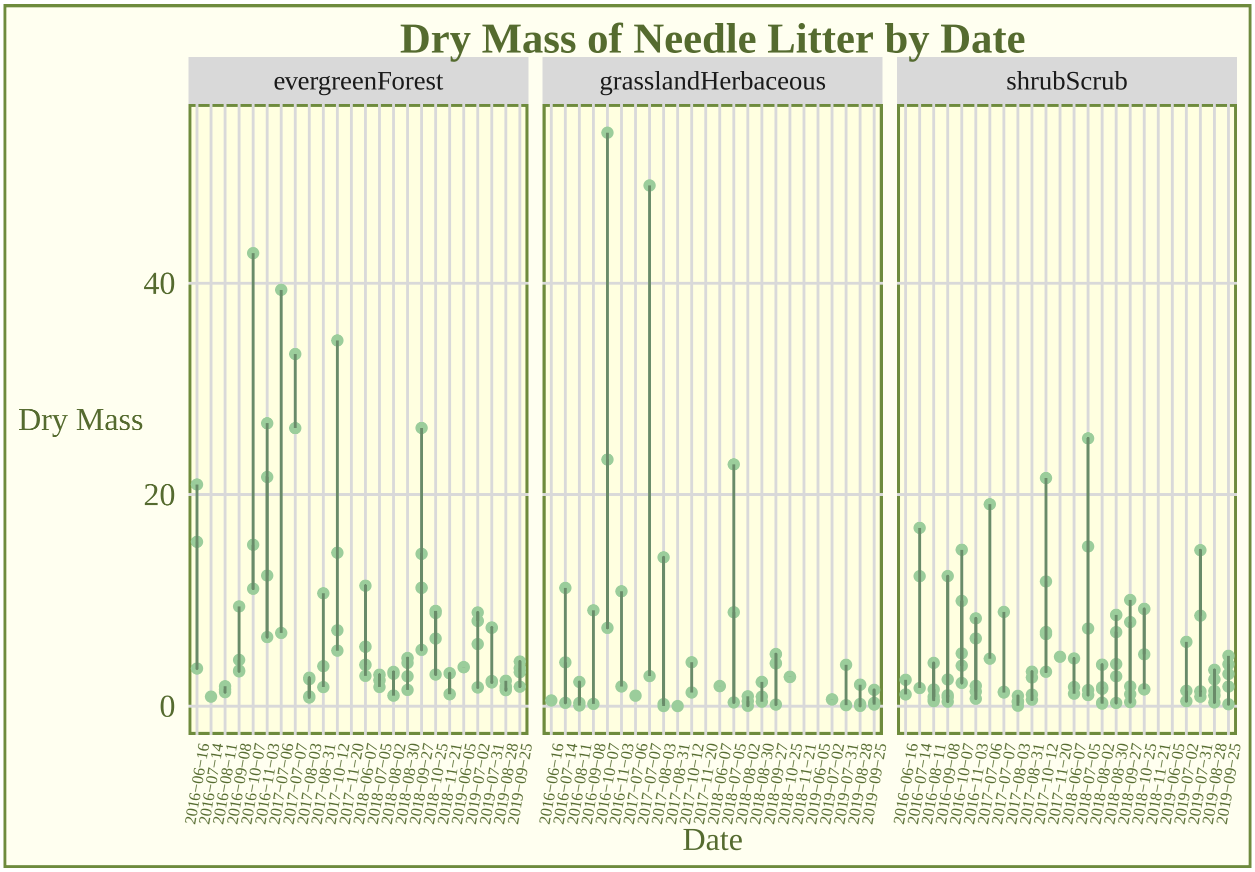
```
#6
#filter needles functional group
needles = litter %>%
  filter(functionalGroup == "Needles")

#plot dry mass of needles by date with NLCD class as color
ggplot(needles, aes(x = collectDate, y = dryMass, color = nlcdClass)) +
  geom_point() +
  geom_line() +
  labs(title = "Dry Mass of Needle Litter by Date",
    x = "Date",
    y = "Dry Mass",
    color = "NLCD Class") +
  scale_color_manual(
    values = c("evergreenForest" = "cadetblue2", "grasslandHerbaceous" = "rosybrown2", "shrubScrub" = "r
```

```
    labels = c("evergreenForest" = "Evergreen Forest", "grasslandHerbaceous" = "Grassland Herbaceous",
  my_theme +
  theme(axis.text.x = element_text(angle = 60))
```




```
#7
#NLCD classes separated into three facets
ggplot(needles, aes(x = collectDate, y = dryMass)) +
  geom_point(color = "darkseagreen3") +
  geom_line(color = "darkseagreen4") +
  facet_wrap(facets = vars(nlcdClass)) +
  labs(title = "Dry Mass of Needle Litter by Date",
    x = "Date",
    y = "Dry Mass") +
  my_theme +
  theme(axis.text.x = element_text(angle = 80, size = 6))
```



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

> Answer: I think facet plot is more effcient because each NLCD class has its own facet, making it easier to focus on individual patterns without clutter. In the plot 6, points are overlapped so it is difficult to distinguish between the different classes, even with distinct colors.