Assignment 5: Data Visualization

Candela Cerpa

Fall 2023

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, verify directory, read it data
library(tidyverse);library(lubridate);library(here);library(cowplot);library(ggthemes)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## 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
```

```
## here() starts at Z:/EDE_Fall2023
##
##
## Attaching package: 'cowplot'
##
##
## The following object is masked from 'package:lubridate':
##
##     stamp
##
##
##
## Attaching package: 'ggthemes'
##
##
## The following object is masked from 'package:cowplot':
##
##     theme_map
```

```
getwd()
```

```
## [1] "Z:/EDE_Fall2023"
```

```
NTL_LTER <-
  read.csv("./Data/Processed_KEY/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv",
                     stringsAsFactors = TRUE)
NEON_NIWO <-
  read.csv("./Data/Processed_KEY/NEON_NIWO_Litter_mass_trap_processed.csv",
                      stringsAsFactors = TRUE)

#2 check date formats and adapt
print(class(NTL_LTER$sampledate)) #read as factor
```

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

```
print(class(NEON_NIWO$collectDate)) #read as factor
```

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

```
NTL_LTER$sampledate <- ymd(NTL_LTER$sampledate) #change to date format
NEON_NIWO$collectDate <- ymd(NEON_NIWO$collectDate) #change to date format

print(class(NTL_LTER$sampledate)) #read as date
```

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

```
print(class(NEON_NIWO$collectDate)) #read as date
```

```
## [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

```
#3 Create a custom theme
my_theme <- theme_base() +
  theme(
    line = element_line(
      linewidth = 2
      ),
    plot.title = element_text(
      color = "darkgreen"
      ),
    axis.ticks = element_line(
      color = "darkgreen"
      ),
    plot.background = element_rect(
      color = "black",
      fill = "gray"
      ),
    complete = TRUE
    )
theme_set(my_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.

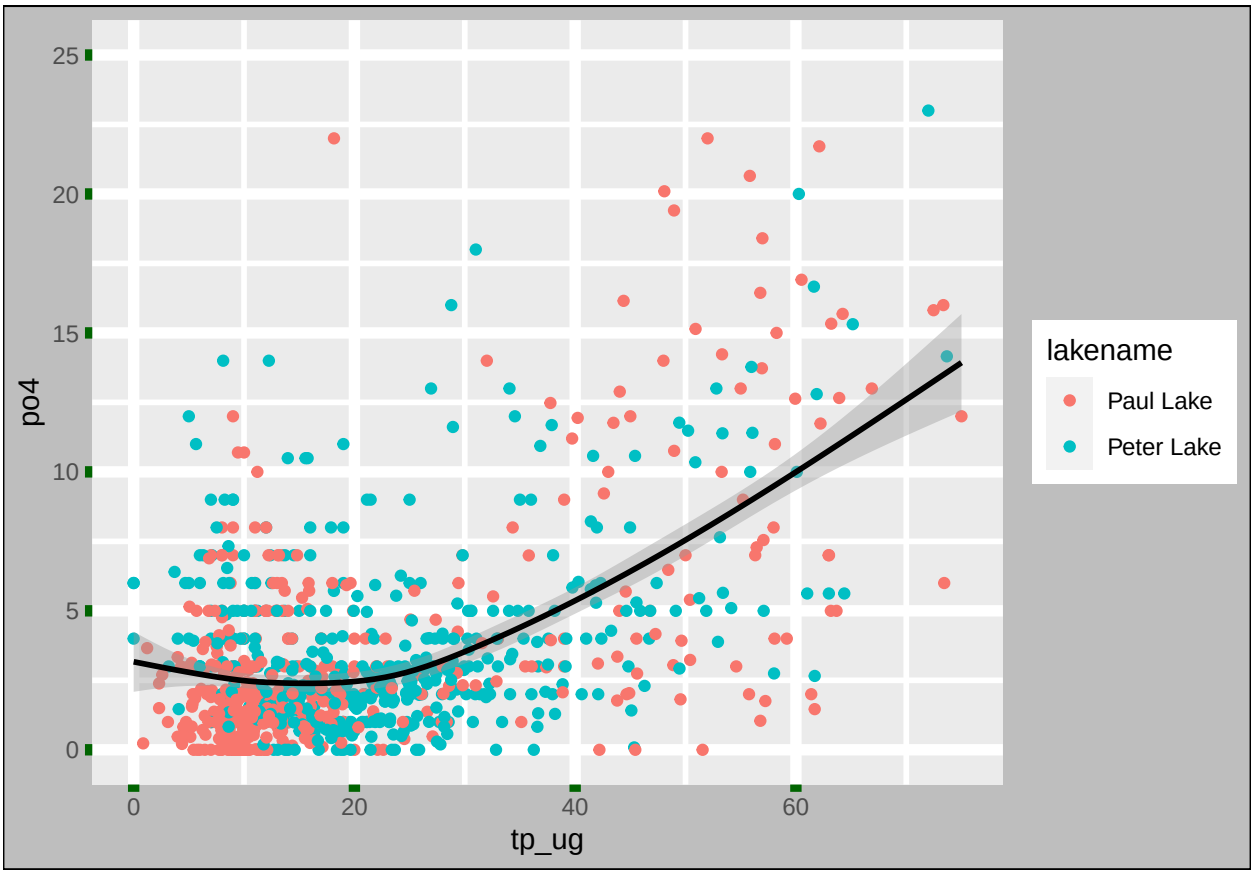
4. [NTL-LTER] Plot total phosphorus (`tp_ug`) by phosphate (`po4`), with separate aesthetics for Peter and Paul lakes. Add a line of best fit and color it black. Adjust your axes to hide extreme values (hint: change the limits using `xlim()` and/or `ylim()`).

```
#4 NTL-LTER tp_ug by po4. separate aesthetics for Peter & Paul Lakes
# add line of best fit. adjust axes to hide extreme values
o4_plot <- ggplot(NTL_LTER, (aes(x = tp_ug, y = po4, color = lakename))) +
  geom_point() +
  geom_smooth(color = "black") +
  xlim(0, 75) +
  ylim(0, 25)
print(o4_plot)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 21996 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 21996 rows containing missing values (`geom_point()`).
```

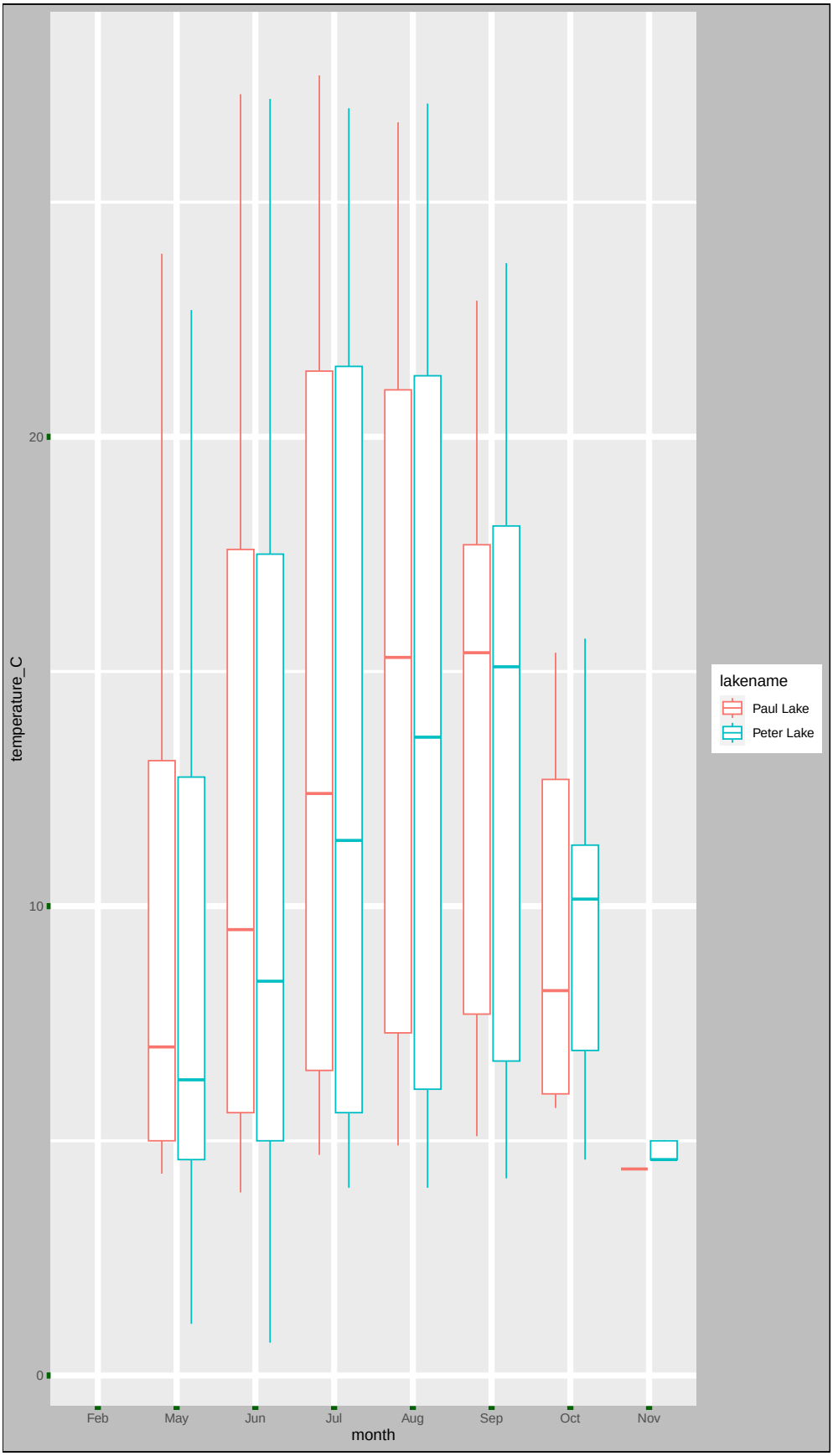


5. [NTL-LTER] Make three separate boxplots of (a) temperature, (b) TP, and (c) TN, with month as the x axis and lake as a color aesthetic. Then, create a cowplot that combines the three graphs. Make sure that only one legend is present and that graph axes are aligned.

Tip: * Recall the discussion on factors in the previous section as it may be helpful here. * R has a built-in variable called `month.abb` that returns a list of months;see https://r-lang.com/month-abb-in-r-with-example

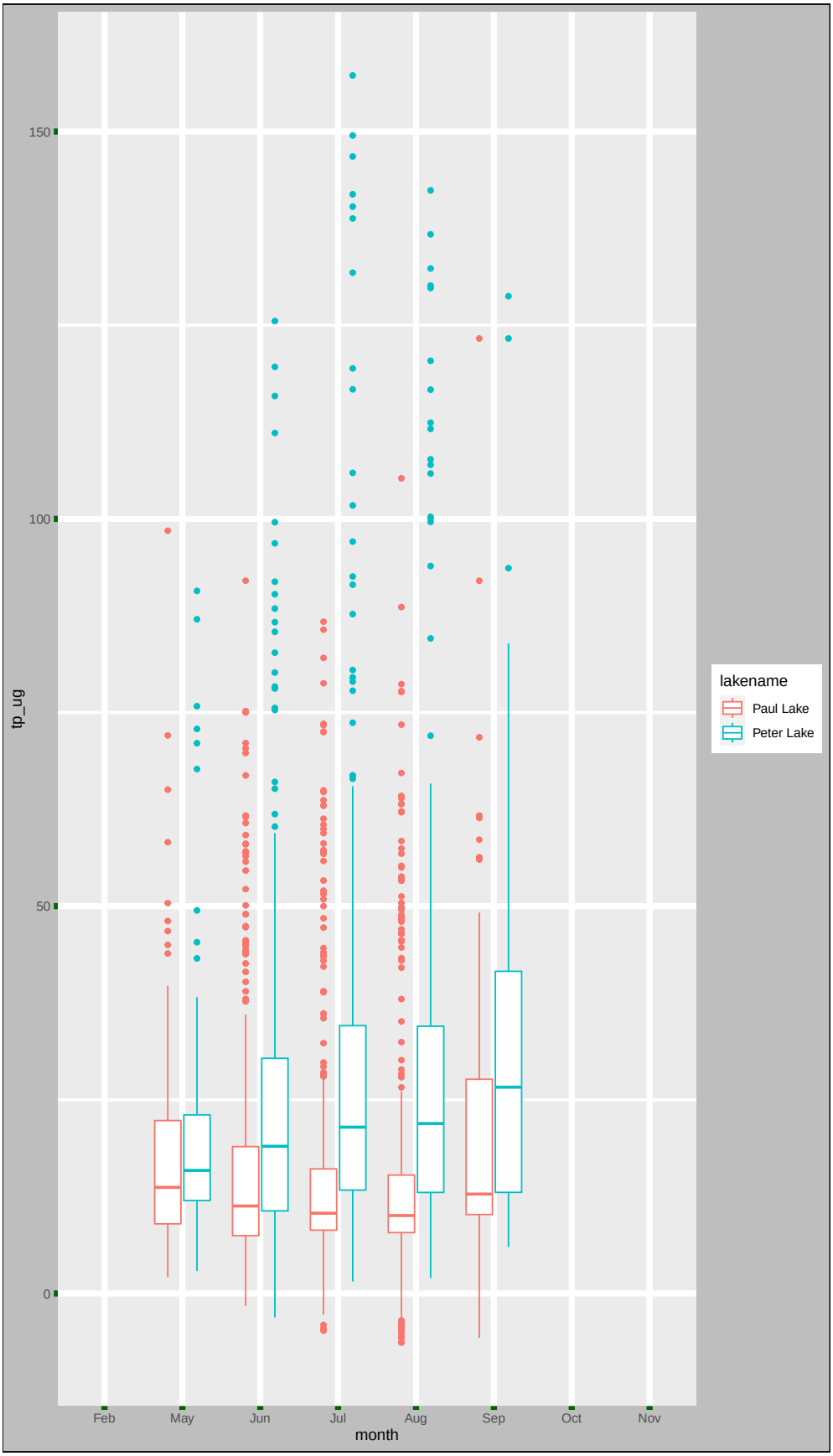
```
#5a boxplot for temp (month on x axis, lake as a color aesthetic)
NTL_LTER$month <- month.abb[NTL_LTER$month]
NTL_LTER$month <- factor(NTL_LTER$month, levels = month.abb)

#I know the order above is weird; it wouldn't work if I tried to both
#turn into a factor and put them in order and I ran out of time!

o5a_plot <- ggplot(NTL_LTER) +
  geom_boxplot(aes(x= month, y = temperature_C, color = lakename))
print(o5a_plot)
```

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


temperature_C
20
10
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
lakename
Paul Lake
Peter Lake

```r
#5b boxplot for TP (month on x axis, lake as a color aesthetic)
o5b_plot <- ggplot(NTL_LTER) +
  geom_boxplot(aes(x= month, y = tp_ug, color = lakename)
               )
print(o5b_plot)
```

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


150
100
50
0
tp_ug
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
lakename
Paul Lake
Peter Lake

```r
#5c boxplot for TN (month on x axis, lake as a color aesthetic)
o5c_plot <- ggplot(NTL_LTER) +
  geom_boxplot(aes(x= month, y = tn_ug, color = lakename))
print(o5c_plot)
```

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


3000
2000
1000
0
tn_ug
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
lakename
Paul Lake
Peter Lake

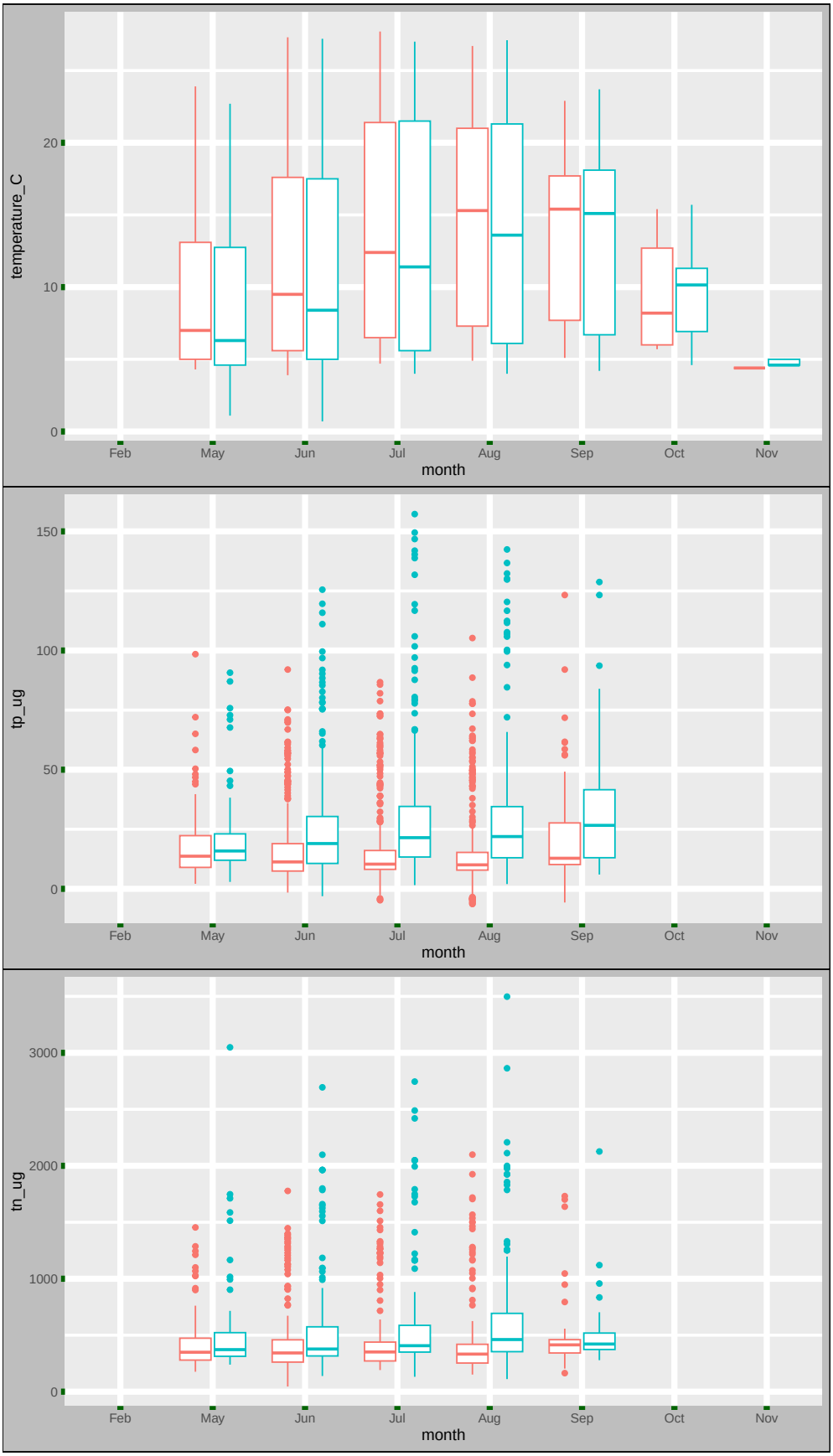
```r
#5d cowplot that combines the 3

#extract a legend from one of the plots, then make it into an object
o5_legend <- get_legend(o5a_plot)
```

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

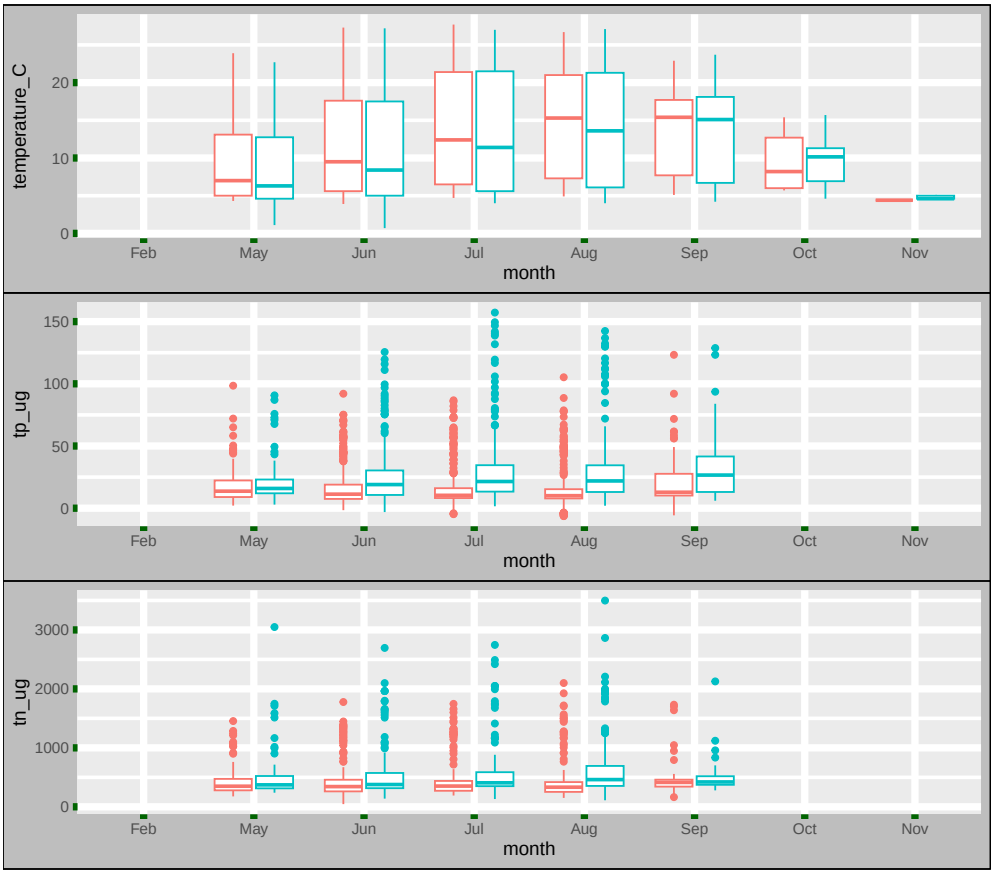
```r
#add the plots without legends to a cowplot
o5_plots <- plot_grid(
  o5a_plot + theme(legend.position = "none"),
  o5b_plot + theme(legend.position = "none"),
  o5c_plot + theme(legend.position = "none"),
  ncol = 1, align = "v", axis = "month")
```

```
## 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()‘).
```

```r
print(o5_plots)
```


temperature_C
20
10
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
tp_ug
150
100
50
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
tn_ug
3000
2000
1000
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month

```r
#combine plots and legend
o5_plots_legend <- plot_grid(
  o5_plots,
  o5_legend,
  ncol = 1
  )
print(o5_plots_legend)
```


temperature_C
20
10
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
tp_ug
150
100
50
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month
tn_ug
3000
2000
1000
0
Feb
May
Jun
Jul
Aug
Sep
Oct
Nov
month


lakename
Paul Lake
Peter Lake

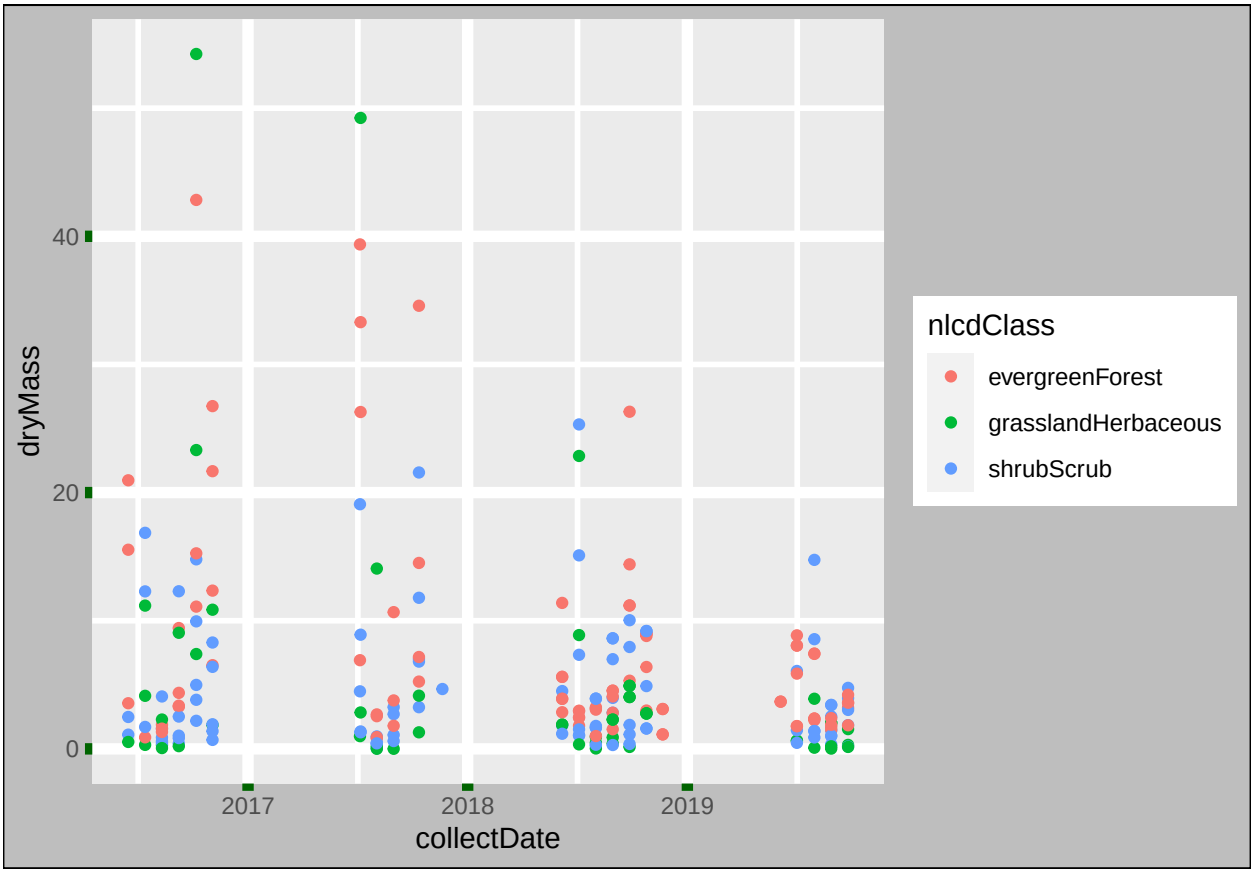
Question: What do you observe about the variables of interest over seasons and between lakes?

> Answer: Temperature-wise, Peter Lake is cooler than Paul Lake most of the year until October. Peter Lake has higher mean tp and tn, but not by much; both lakes have a lot of outliers.

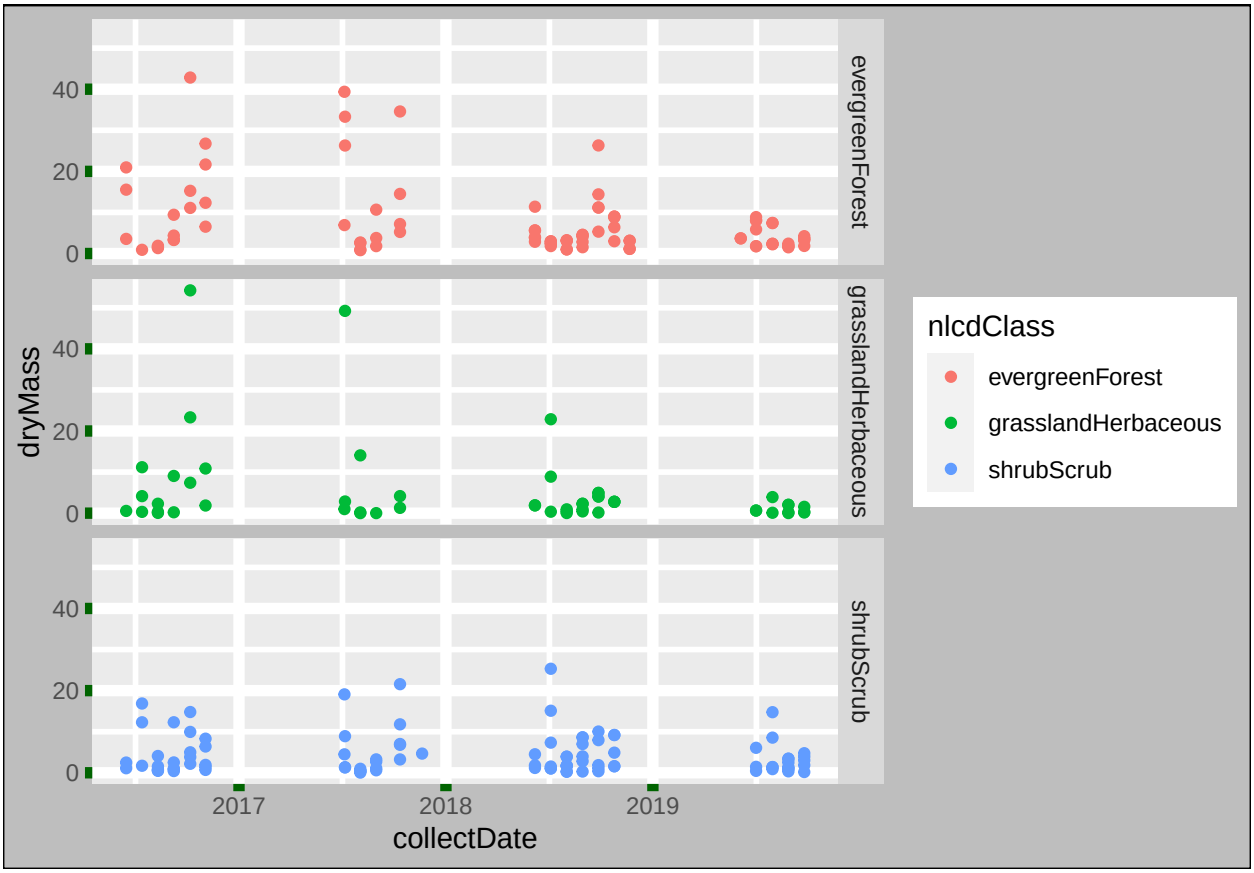
6. [Niwot Ridge] Plot a subset of the litter dataset by displaying only the "Needles" functional group. Plot the dry mass of needle litter by date and separate by NLCD class with a color aesthetic. (no need to adjust the name of each land use)

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

```
#6 plot only Needles; dry mass by date, separate by NLCD class with color.
Needles_df <- filter(NEON_NIWO, functionalGroup == "Needles")
Needles_plot <- ggplot(Needles_df) +
  geom_point(aes(x = collectDate, y = dryMass, color = nlcdClass))
print(Needles_plot)
```




```
#7 plot 6 with each NLCD classes in its own plot
Needles_plot_faceted <- ggplot(Needles_df) +
  geom_point(aes(x = collectDate, y = dryMass, color = nlcdClass)) +
  facet_grid(vars(nlcdClass))
print(Needles_plot_faceted)
```



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

Answer: The second plot (faceted) allows for a clearer view of the trend by class while still allowing for easy comparisons based on year and quantity.