5: Lab - Data Visualization Basics

Environmental Data Analytics | John Fay and Luana Lima | Developed by Kateri Salk

Fall 2024

Objectives

1. Answer questions on M4/A4
2. Answer questions on M5 videos
3. Additional comments on videos - expression(), saving plots, factors
4. Perform simple data visualizations in the R package `ggplot`

Load packages, etc.

```
#Import basic libraries
library(tidyverse);library(lubridate);library(here)

#Library for creating ridge plots
library(ggridges)

#Enhanced color ramps
library(viridis)
library(RColorBrewer)
library(colormap)

#Add pre-set themes <--NEW
library(ggthemes)
```

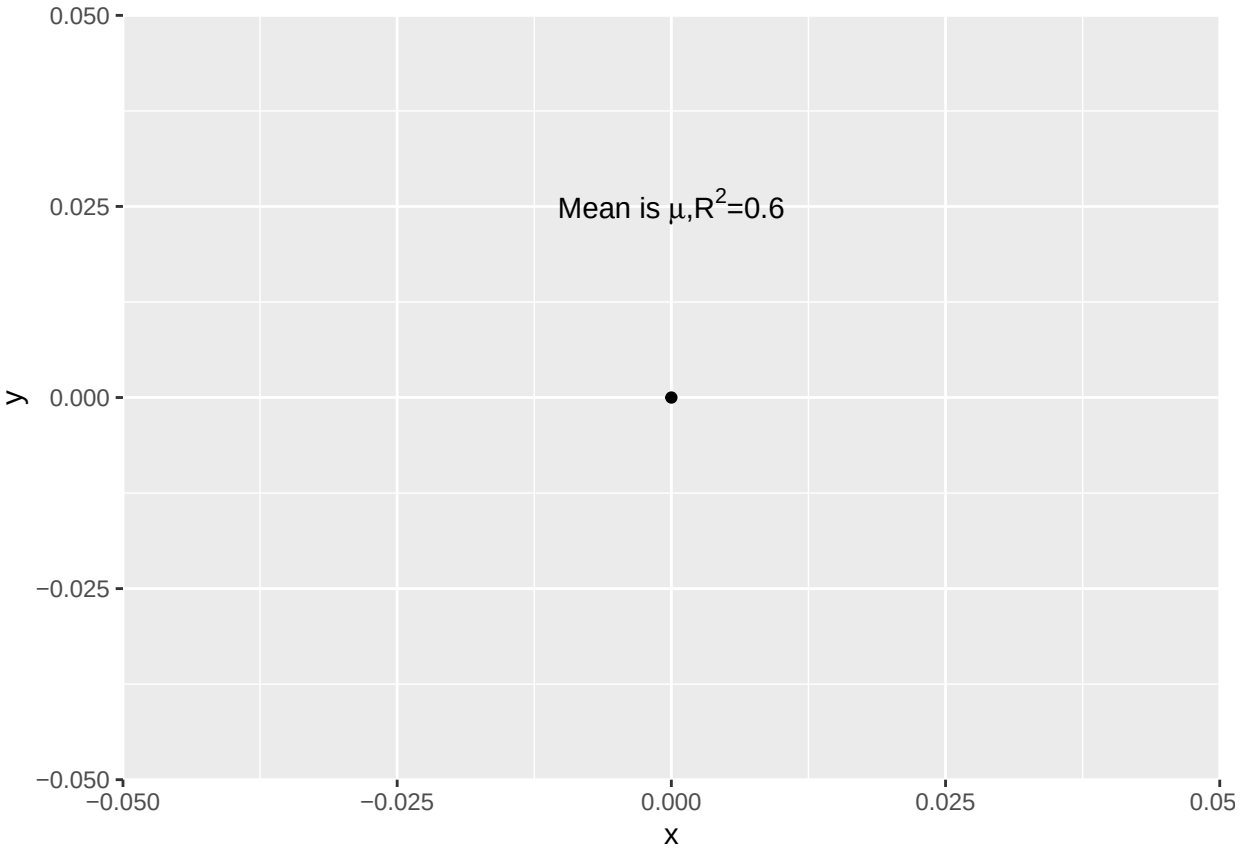
Expression function and Saving plots

The expression function can be used when you want to add mathematical symbols and subscripts mixed with regular letter to labels.

```
#Create a plot
Plot_text <-
ggplot(data=data.frame(x=0,y=0))+
  geom_point(aes(x=x,y=y)) +
  geom_text(
    x=0,
    y=0.025,
    label=expression(paste("Mean is ", mu,",", R^{2},'=0.6'))
    )
```

```r
#Print the plot
print(Plot_text)
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
```




```r
#Save the plot
ggsave(
  filename = "./Output/Plot_text.jpg",
  plot = Plot_text,
  height = 4,
  width = 6,
  units = "in",
  dpi = 300
  )
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
```

Import Datasets

Note the use of a string variable the points to the location of the data combined with the `here()` *package functionality. This enables us to quickly change where we get our data, i.e., by changing one line of code vs each* `read.csv()` *command.*

```r
#Assign a variable to the processed data folder location
processed_data = "./Data/Processed_KEY"

#Read in data
PeterPaul.chem.nutrients <- read.csv(
  here(processed_data,"NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv"),
  stringsAsFactors = TRUE)

EPAair <- read.csv(
  here(processed_data,"EPAair_O3_PM25_NC1819_Processed.csv"),
  stringsAsFactors = TRUE)

#Fix dates
PeterPaul.chem.nutrients$sampledate <- ymd(PeterPaul.chem.nutrients$sampledate)
EPAair$Date <- ymd(EPAair$Date)
```

A note on factors in plots...

Say we want to plot the number of records each month in the Peter/Paul lakes dataset. How might we do that? What challenges do we face?

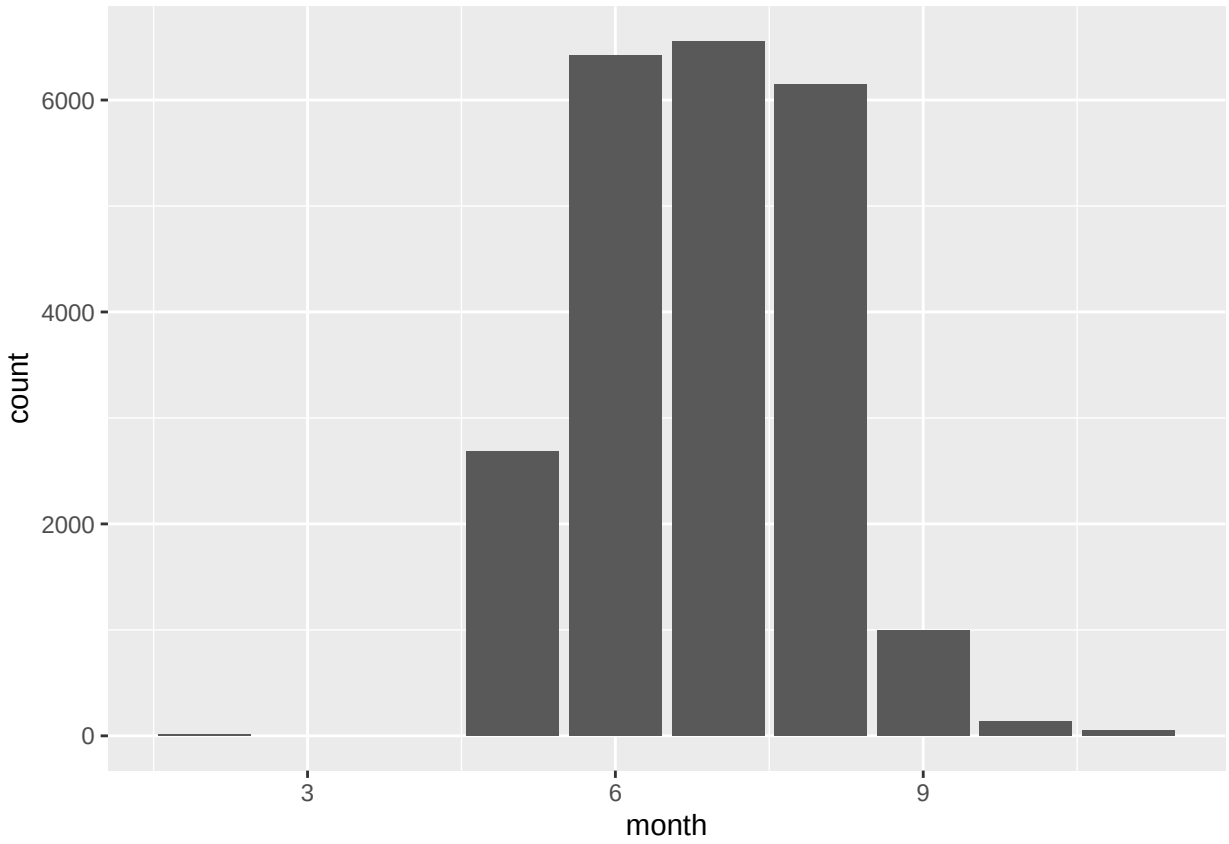
```r
#What kind of values are months?
class(PeterPaul.chem.nutrients$month)
```

```
## [1] "integer"
```

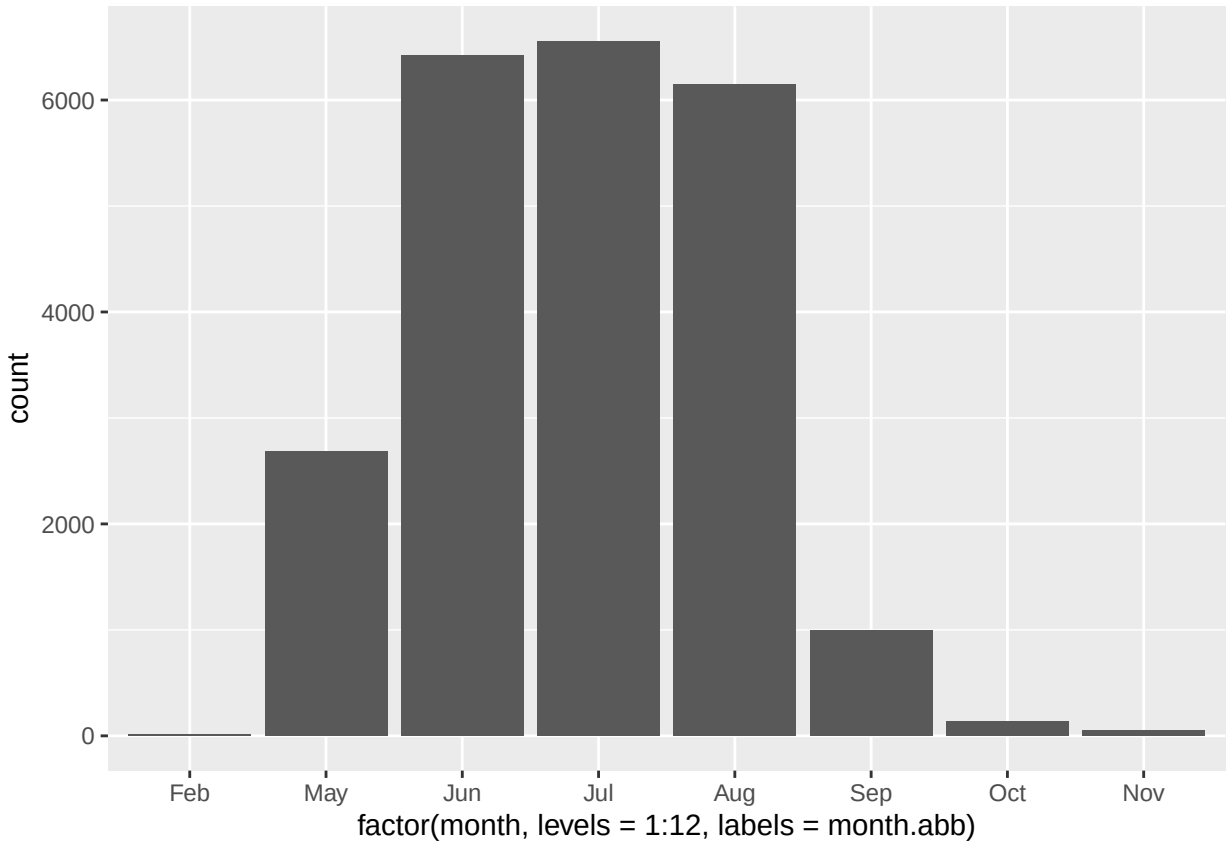
```r
#List unique values
unique(PeterPaul.chem.nutrients$month)
```

```
## [1]  5  6  7  8  9 10 11  2
```

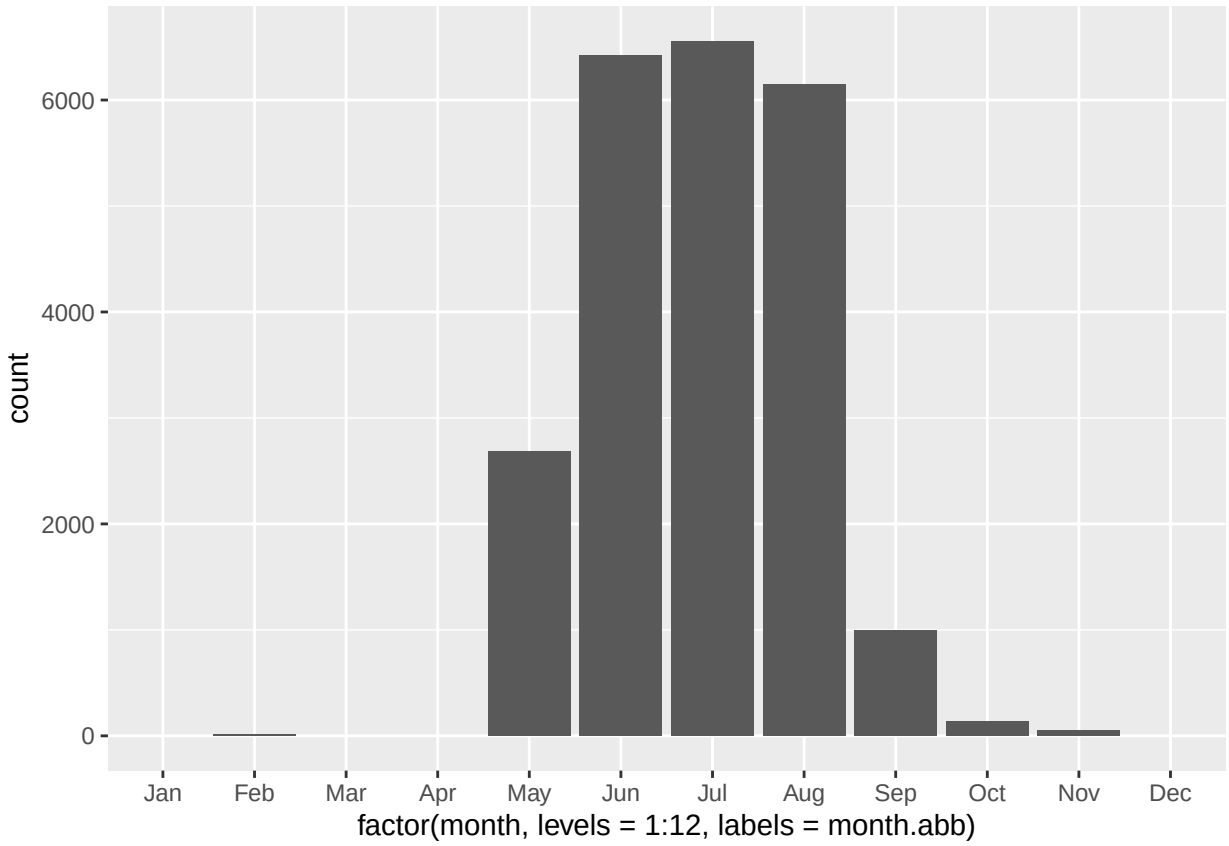
```r
#Plot how many Lake observations occur each month
PeterPaul.chem.nutrients %>%
  ggplot() +
  geom_bar(aes(x=month))
```



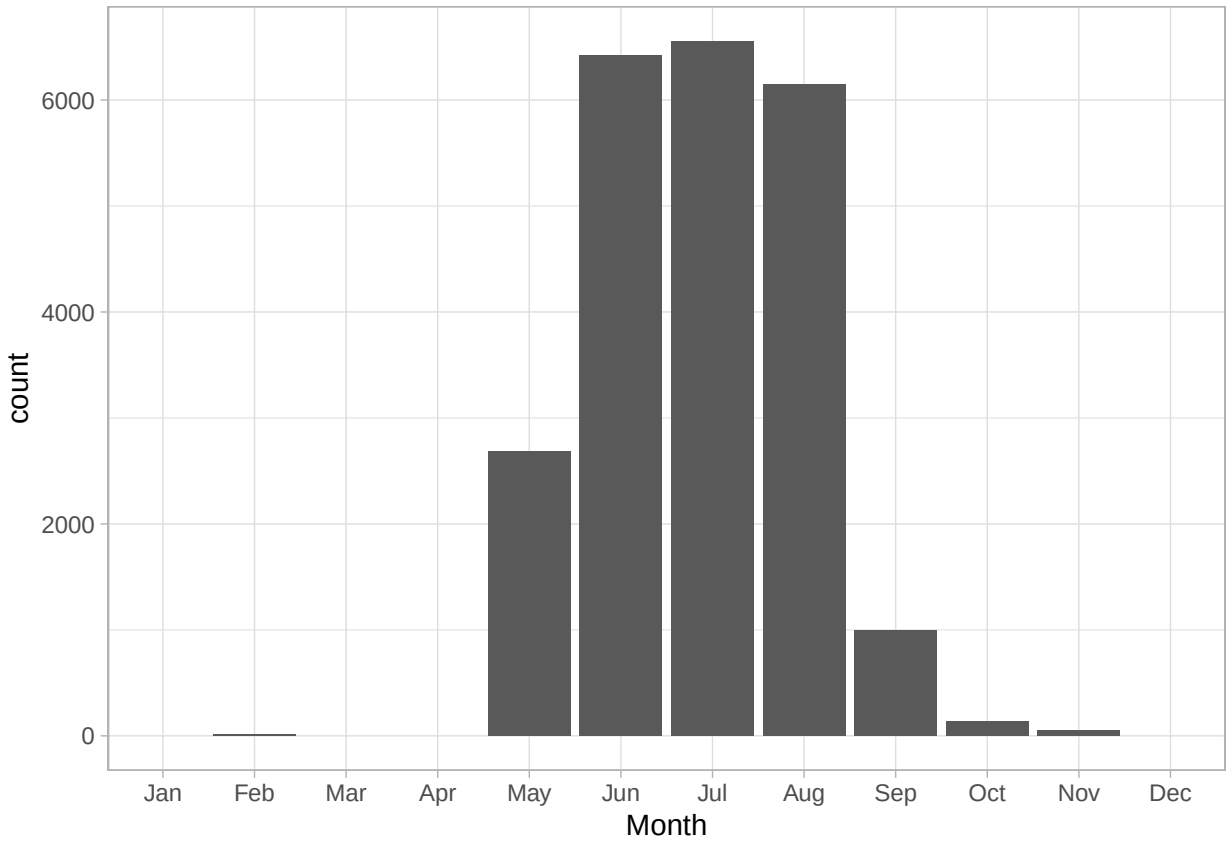

```
#Convert month to a factor -- with 12 levels, labelled with month names
#Commented just to create pdf
#factor(PeterPaul.chem.nutrients$month,
#       levels = 1:12,
#       labels = month.abb)

#Try again: Plot how many Lake observations occur each month
ggplot(
  PeterPaul.chem.nutrients,
  aes(x=factor(month,levels = 1:12,labels = month.abb)))+
  geom_bar()
```




```
#Alter to include missing months
ggplot(
  PeterPaul.chem.nutrients,
  aes(x=factor(month,levels = 1:12,labels = month.abb)))+
  scale_x_discrete(drop=F) +
  geom_bar()
```




```
#Tidy up the code
the_plot <- PeterPaul.chem.nutrients %>%
  ggplot(
    aes(x=factor(
      month,
      levels=1:12,
      labels=month.abb)
    )
  ) +
  geom_bar() +
  scale_x_discrete(
    name='Month',
    drop=FALSE)

#Show the plot, in the light theme
the_plot + theme_light()
```



More on themes

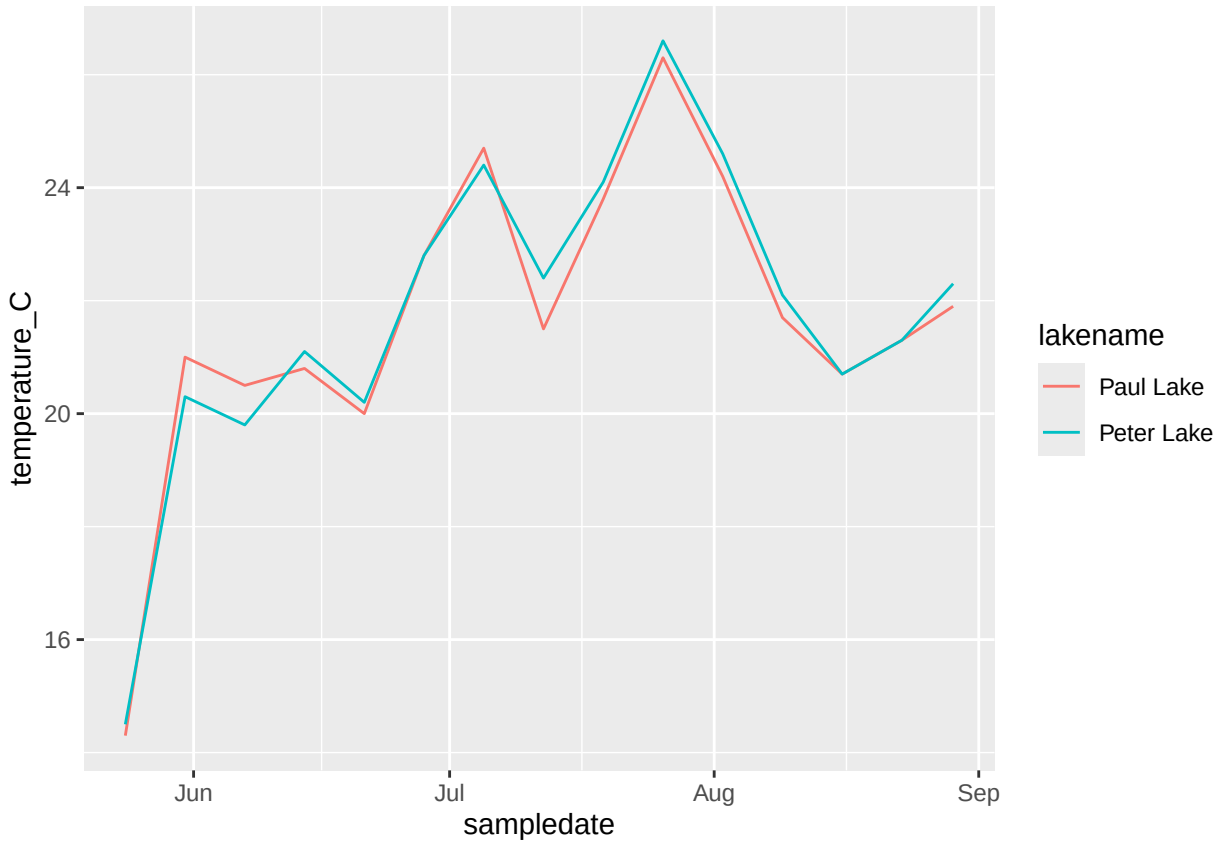
Themes are used to control the overall appearance of a plot, such as the color, size, font, and style of the various elements in the plot. Themes provide a quick and easy way to change the visual style of a plot, and they allow us to create custom visualizations that fit our specific needs and preferences.

Themes control the following elements

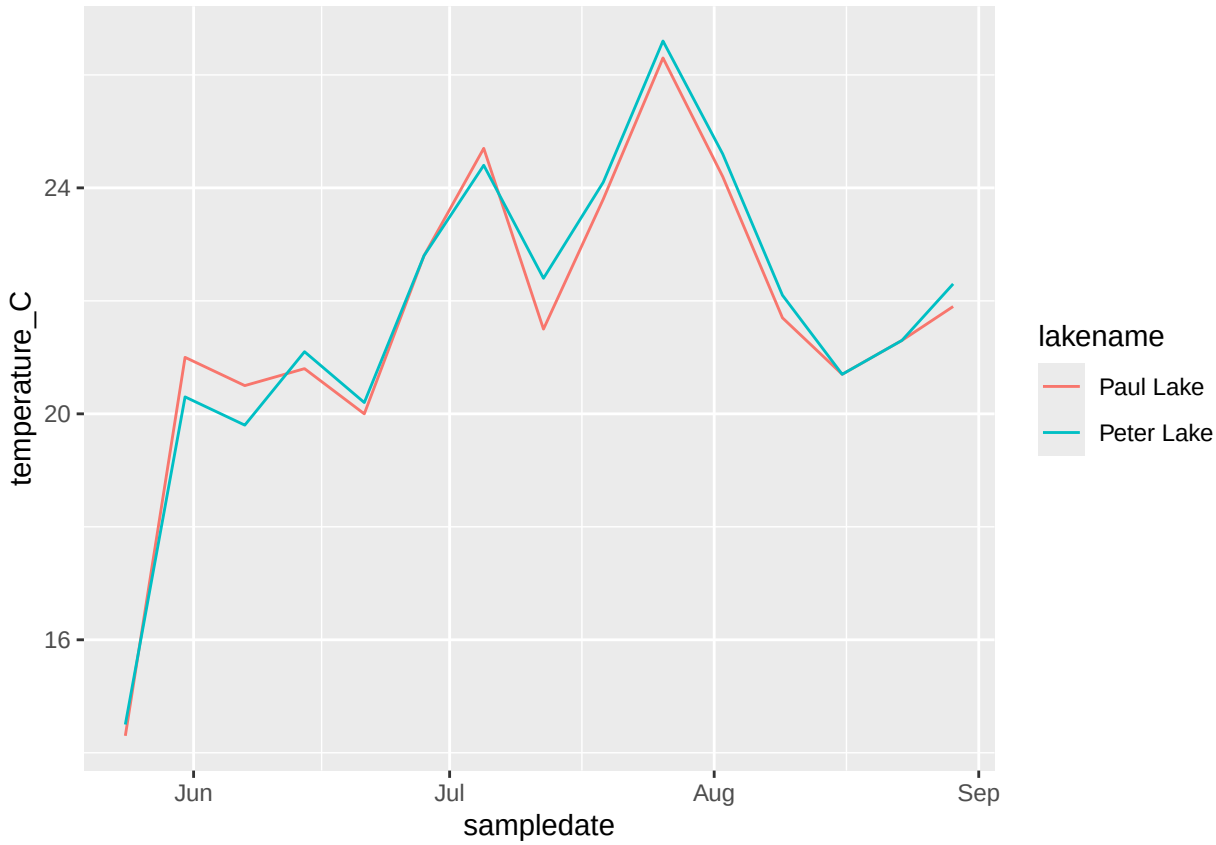
- *Plot background*: The background color or fill pattern of the plot area.
- *Plot title*: The size, font, and position of the plot title.
- *Axis labels*: The font, size, and position of the x-axis and y-axis labels.
- `_Axis ticks and grid lines_: The color, size, and position of the tick marks and grid lines on the`
- *Legend*: The font, size, and position of the legend.

Resource: - https://mran.microsoft.com/snapshot/2017-02-04/web/packages/ggthemes/vignettes/ggthemes.html - https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/ - https://ggplot2.tidyverse.org/reference/theme.html

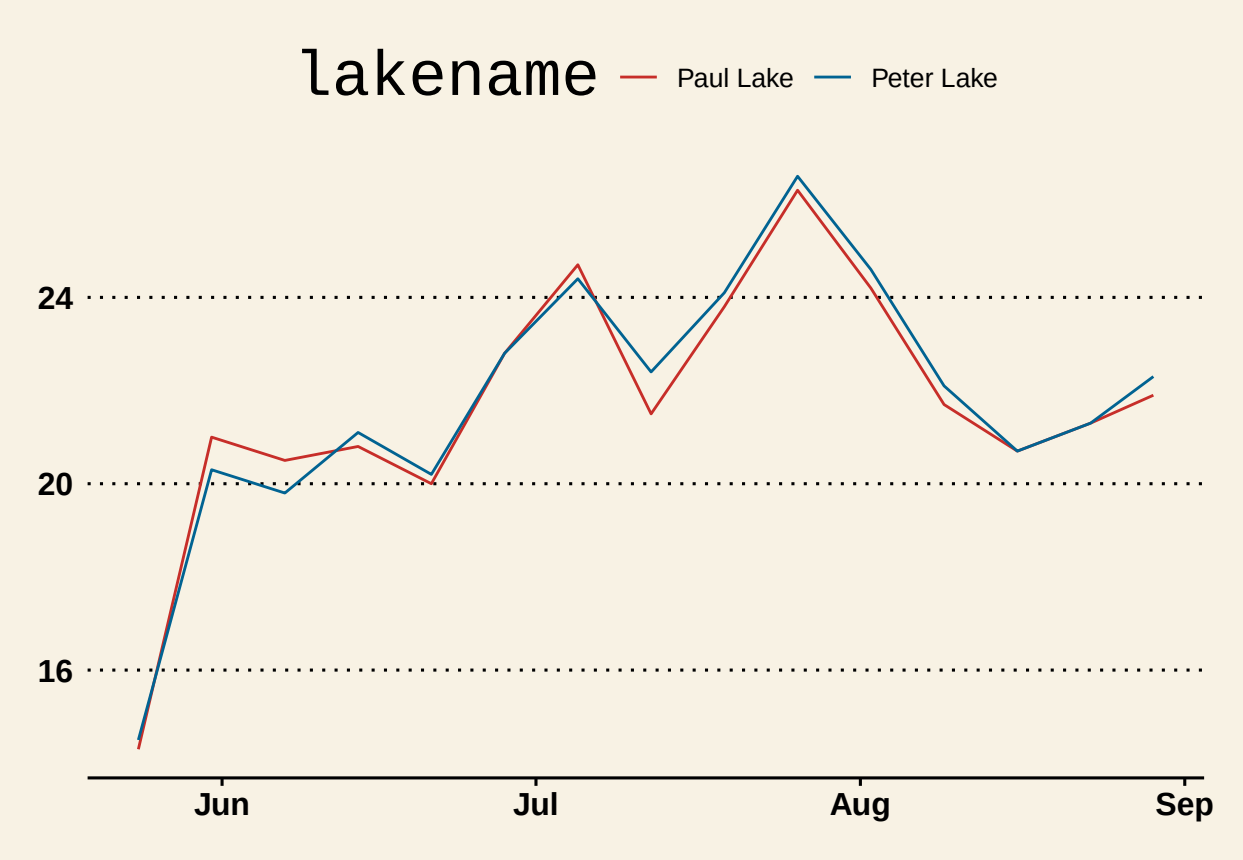
```
#Create a base plot to play with
surface_temp_1999_plot <- PeterPaul.chem.nutrients %>%
  filter(depth == 0 & year4 == 1999) %>%
  ggplot(
    mapping = aes(
      x=sampledate,
      y=temperature_C,
      color=lakename)
    ) +
  geom_line()

#Show the basic plot
surface_temp_1999_plot
```



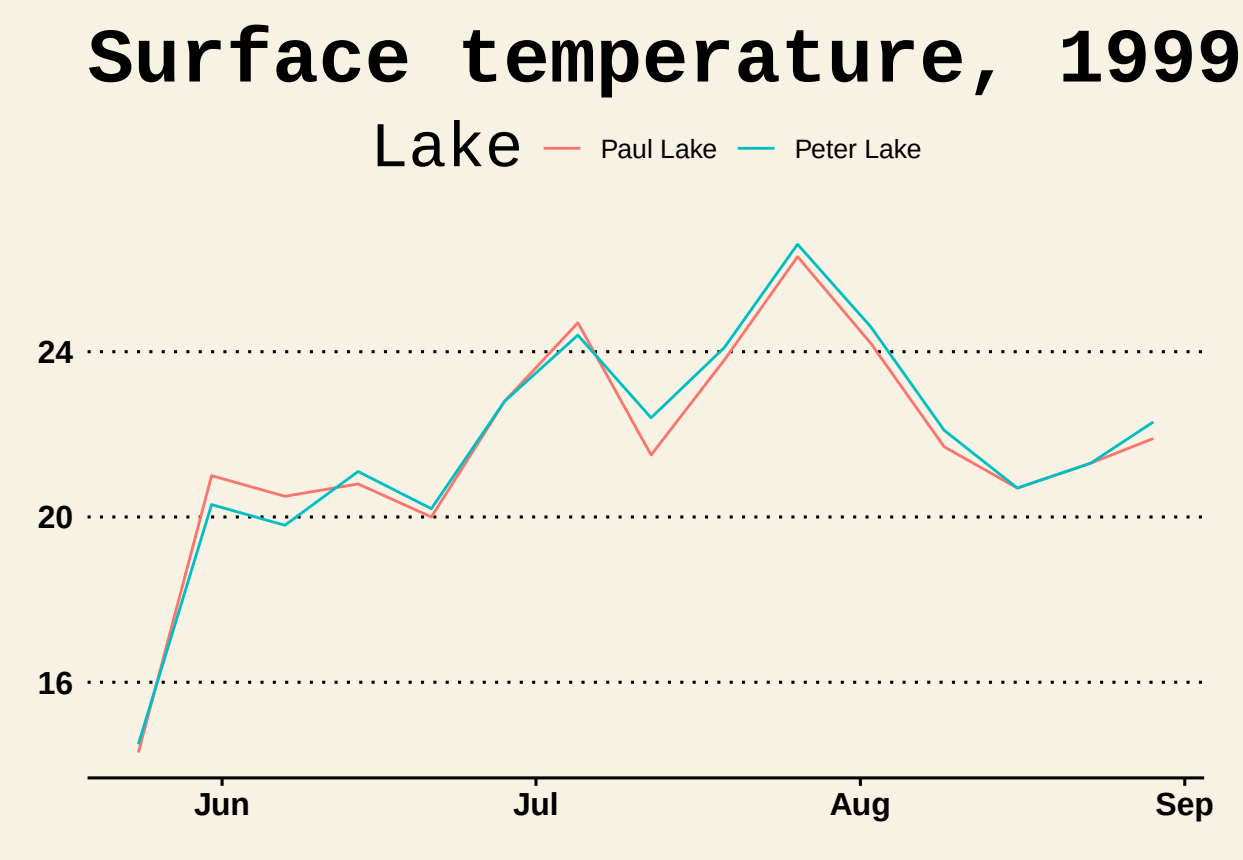

```
#Apply a theme
surface_temp_1999_plot +
  theme_gray()
```




```
#Apply a ggtheme
surface_temp_1999_plot +
  theme_wsj() +
  scale_color_wsj()
```




```
#Add a title
surface_temp_1999_plot +
  ggtitle("Surface temperature, 1999") +
  labs(color='Lake') +
  theme_wsj()
```




```
#Create a custom theme
my_theme <- theme_base() +
  theme(
    line = element_line(
      color='red',
      linewidth =2
    ),
    legend.background = element_rect(
      color='grey',
      fill = 'orange'
    ),
    legend.title = element_text(
      color='blue'
    )

  )

#TIP: Expand the theme in the Environments
```

```
my_theme <- theme_base()  +
  theme(
    #line =              element_line(),
    #rect =              element_rect(),
    #text =              element_text(),

    # Modified inheritance structure of text element
```

```
    #plot.title =        element_text(),
    #axis.title.x =      element_blank(),
    #axis.title.y =      element_blank(),
    #axis.text =         element_text(),

    # Modified inheritance structure of line element
    #axis.ticks =        element_line(),
    #panel.grid.major =  element_line(),
    #panel.grid.minor =  element_blank(),

    # Modified inheritance structure of rect element
    #plot.background =   element_rect(),
    #panel.background =  element_rect(),
    #legend.key =        element_rect(),

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

    #complete = TRUE
    )
```

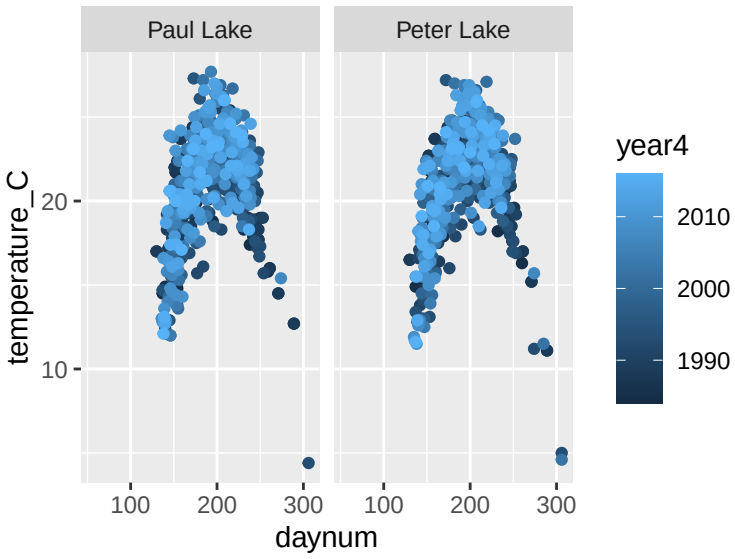
Scales

Scales in ggplot are used to map data values to aesthetic properties of plot elements, such as position, size, color, and shape. In addition to the basic types of scales, such as continuous and discrete scales, ggplot also provides date and time scales, and guide scales, which allow us to format and label the x-axis with appropriate date or time values, and to add legends and other visual guides to the plot. With a little bit of practice and experimentation, you can use ggplot to create beautiful and informative data visualizations with customized scales.

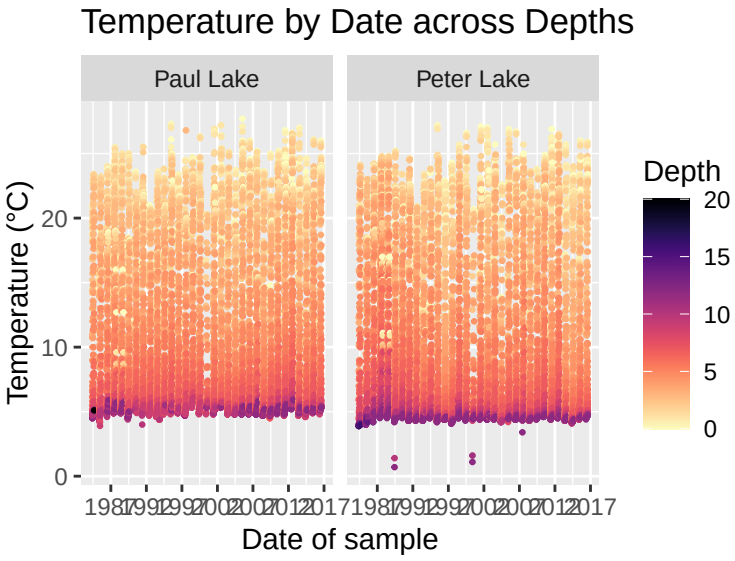
Plotting continuous variables over time: Scatterplot and Line Plot

Exercise: build your own scatterplots of PeterPaul.chem.nutrients

```
# 1.
# Plot surface temperatures by day of  year.
# Color your points by year, and facet by lake in two rows.
# Change the ylab name
PeterPaul.chem.nutrients %>%
  filter(depth==0) %>%
  ggplot(aes(x=daynum,y=temperature_C,color=year4)) +
  geom_point()+
  facet_wrap(
    facets = vars(lakename),
    nrow=1,ncol=2)
```




```
#2.
# Plot temperature by date. Color your points by depth.
# Change the size of your point to 0.5
# Change the color palette to magma and play with direction (+- 1), which one makes more sense?
# Change x axis to include marker/labels every 5 years
PeterPaul.chem.nutrients %>%
  ggplot(aes(
    x=sampledate,
    y=temperature_C,
    color=depth)
    ) +
  geom_point(size=0.5)+
  facet_wrap(
    facets = vars(lakename),
    nrow=1,ncol=2
    ) +
  scale_color_viridis(
    option = "magma",
    direction = -1
    ) +
  scale_x_date(
    date_breaks = "5 years",
    date_labels='%Y') +
  labs(
    title = "Temperature by Date across Depths",
    y = "Temperature (°C)",
    x = "Date of sample",
    color = "Depth"
   )
```

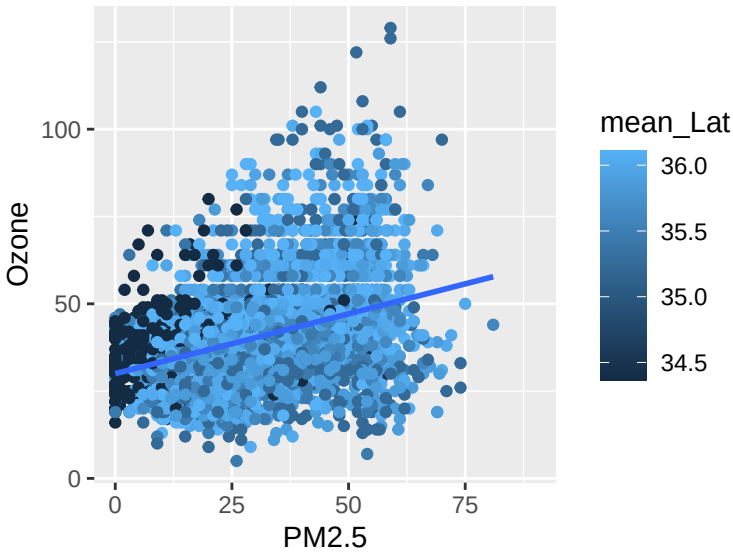


Plotting the relationship between two continuous variables: Scatterplot

Exercise: Plot relationships between air quality measurements

```
# 3.
# Plot AQI values for ozone by PM2.5, colored by latitude
# Make the points 50 % transparent
# Add a line of best fit for the linear regression of these variables.
EPAair %>%
  ggplot(aes(
    x=PM2.5,
    y=Ozone,
    color=mean_Lat
  ),
  alpha=0.5
  ) +
  geom_point() +
  geom_smooth(
    method = lm,
    se=FALSE
    )
```

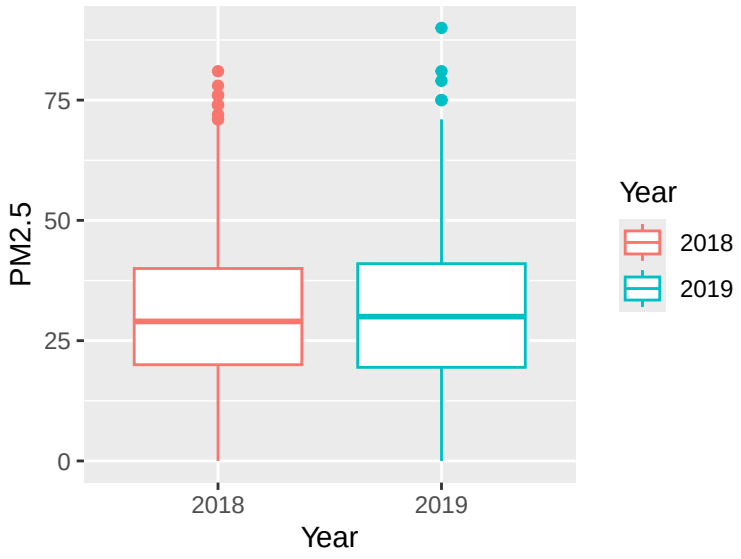
```
## `geom_smooth()` using formula = 'y ~ x'
```



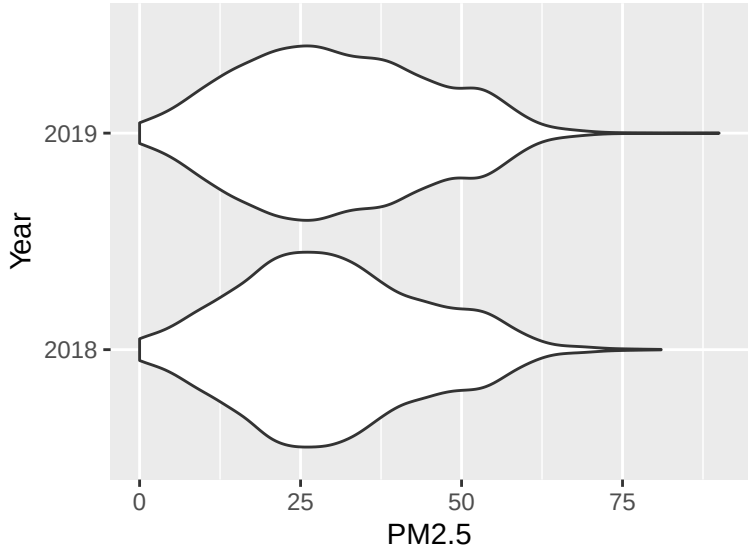
Plotting continuous vs. categorical variables

Exercise: Plot distributions of AQI values for EPAair.

```
# 4.
# Create several types of plots depicting PM2.5, divided by year.
# Choose which plot displays the data best and justify your choice.
EPAair %>%
  ggplot(aes(
    x=factor(Year),
    y=PM2.5,
    color=factor(Year)
    )
  ) +
  geom_boxplot()+
  labs(
    x='Year',
    color='Year')
```

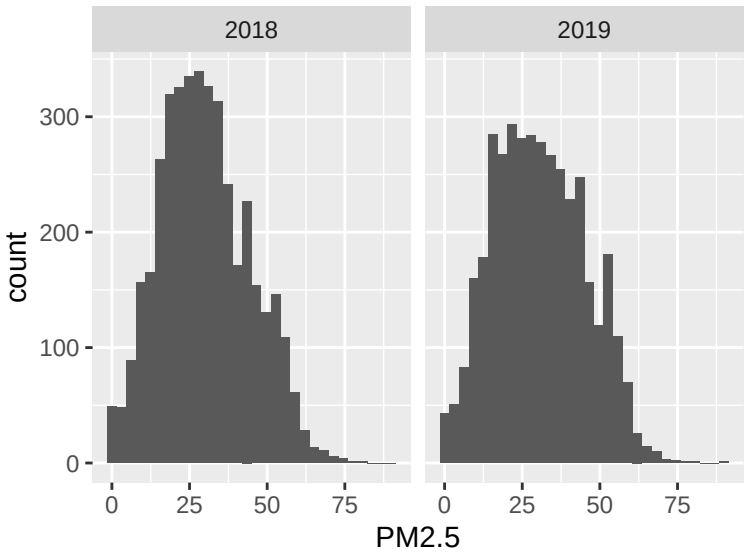



```
ggplot(EPAair,aes(x=PM2.5,y=as.factor(Year)))+
  #geom_boxplot() +
  geom_violin() +
  #geom_dotplot(size=0.2) +
  #geom_density_ridges(aes(fill=as.factor(Year)))+
  ylab('Year') +
  labs(fill='Year')
```




```
#Faceted histograms
ggplot(EPAair,aes(x=PM2.5)) +
  geom_histogram()+
  facet_wrap('Year')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

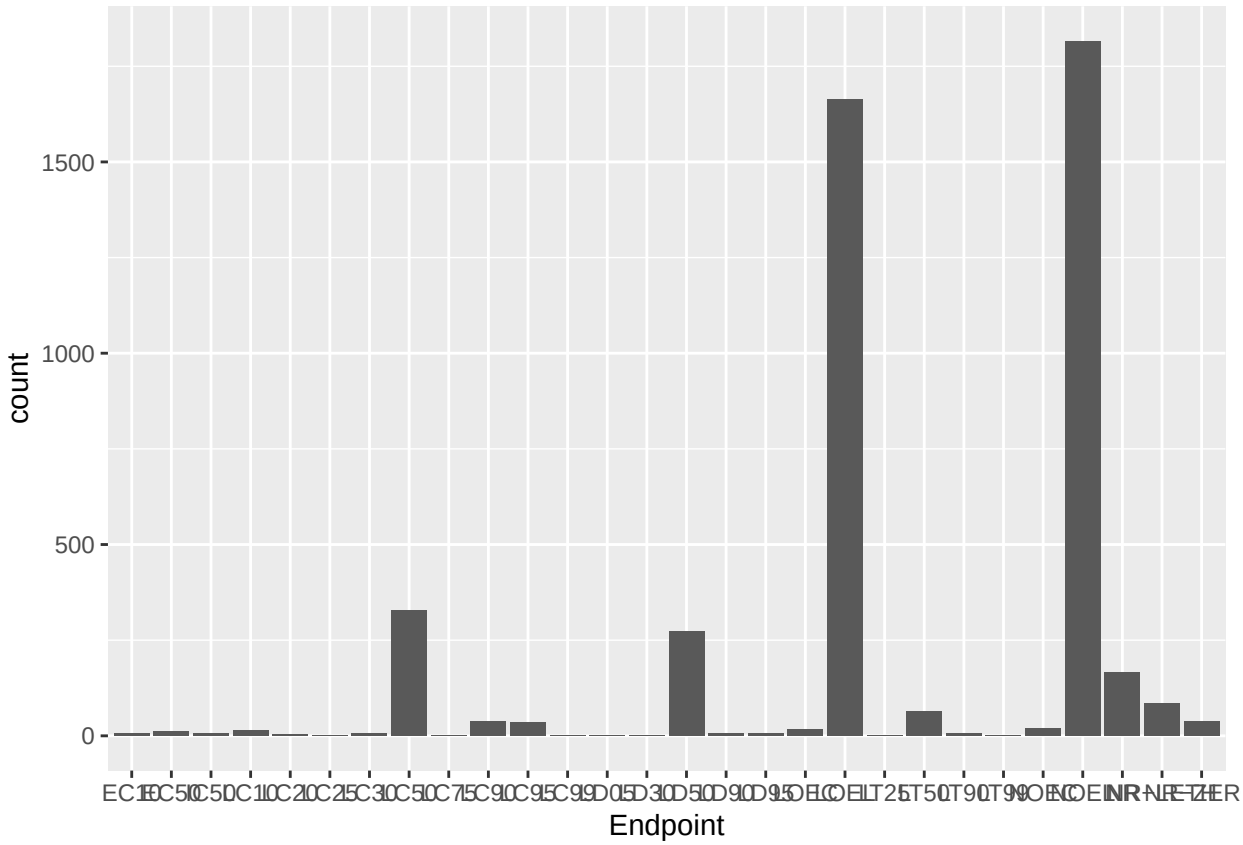


Visualization challenge

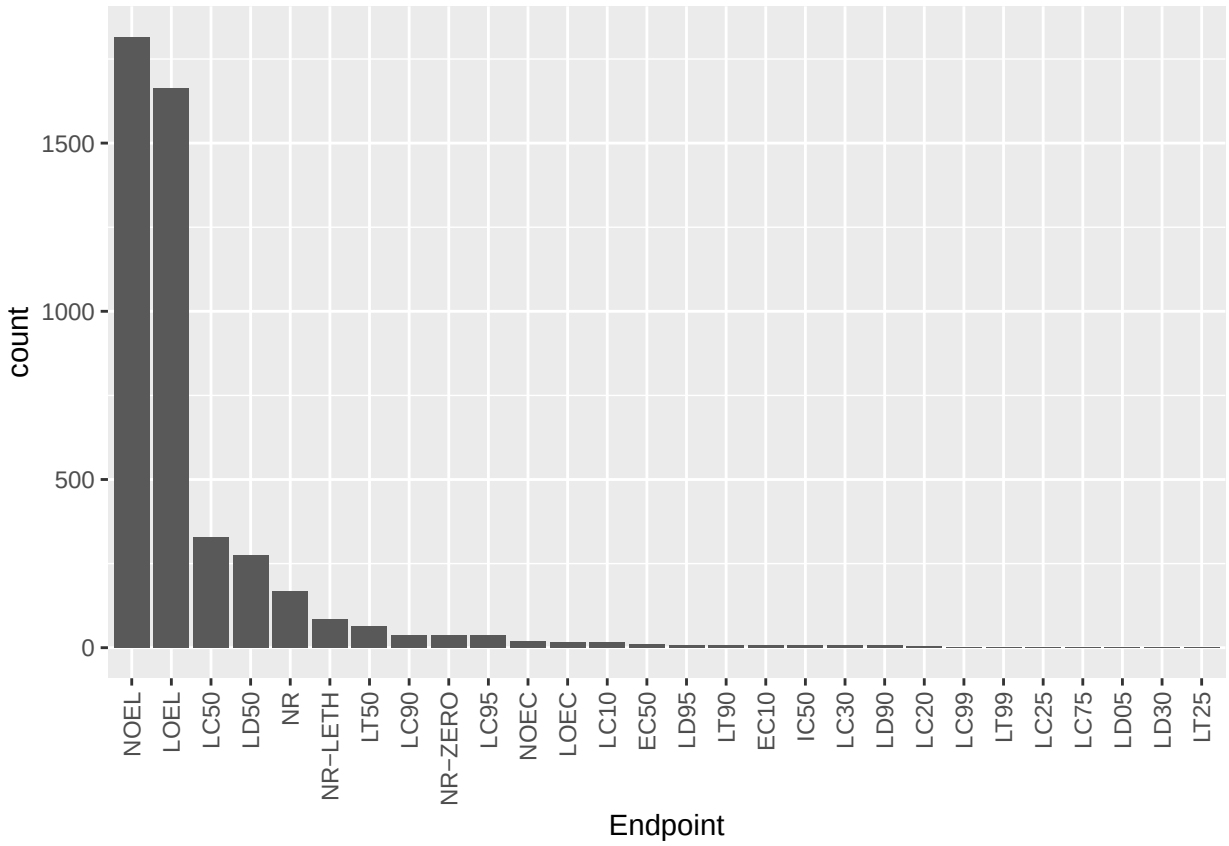
The following graph displays the counts of specific endpoints measured in neonicotinoid ecotoxicology studies. The way it is visualized, however, is not effective. Make the following coding changes to improve the graph:

1. Change the ordering of the "Endpoint" factor (function: `reorder`) so that the highest counts are listed first (hint: FUN = length)
2. Plot the barplot with the reordered factor levels. Add this line of code to make the bars show up left to right: `scale_x_discrete(limits = rev(levels(Neonics$Endpoint)))`
3. Adjust the x axis labels so they appear at a 45 degree angle.
4. Change the color and/or border on the bars. Should you have a consistent color across all bars, or a different color for each bar?

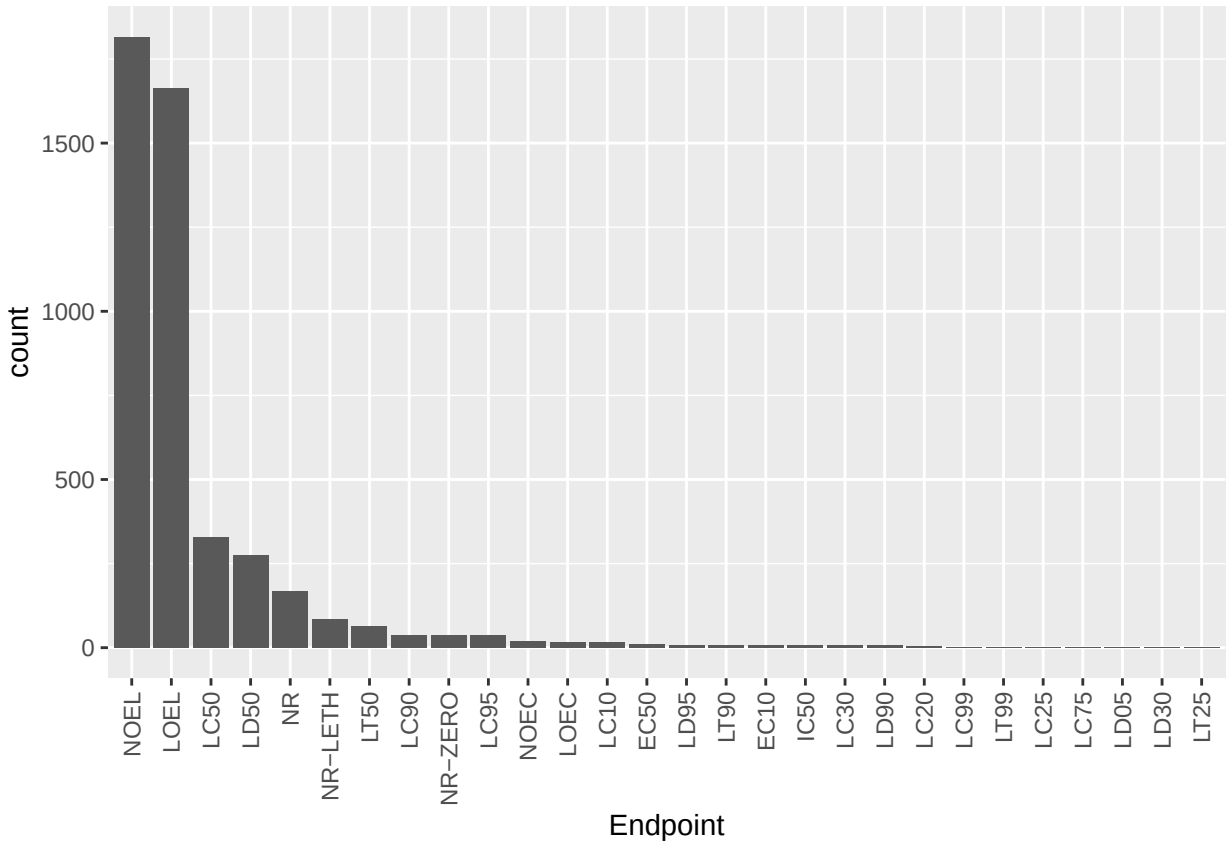
```
Neonics <- read.csv(
  here("./Data/Raw/ECOTOX_Neonicotinoids_Insects_raw.csv"),
  stringsAsFactors = TRUE)

ggplot(Neonics) +
  geom_bar(aes(x = Endpoint))
```




```
#Using reorder
ggplot(Neonics,aes(x=reorder(Endpoint, Endpoint, function(x) 1-length(x)))) +
  geom_bar() +
  labs(x='Endpoint') +
  theme(    axis.text.x = element_text(
      angle = 90,
       vjust = 0.5,
       hjust=1)
  )
```




```
#Using forcats
ggplot(Neonics,aes(x = fct_infreq(Endpoint))) +
  geom_bar(stat='count') +  labs(x='Endpoint') +
  theme(
    axis.text.x = element_text(
      angle = 90,
       vjust = 0.5,
       hjust=1)
  )
```


count
1500
1000
500
0
NOEL
LOEL
LC50
LD50
NR
NR−LETH
LT50
LC90
NR−ZERO
LC95
NOEC
LOEC
LC10
EC50
LD95
LT90
EC10
IC50
LC30
LD90
LC20
LC99
LT99
LC25
LC75
LD05
LD30
LT25
Endpoint