# Lab 2: Graphs

In this lab, you'll use pollution and weather data from the Fort McMurray area in Canada to visualize data in ggplot2. ggplot2 is a package in R that many researchers use to visualize their data in a clean way. (A package is like an add-on to R that expands what you can do)
By the end of this lab, you'll be able to install and load packages, understand ggplot2 syntax, and make graphs using ggplot2. You'll also use the following new functions:

```
install.packages()
library()
pivot_wider()
ggplot()
aes()
geom_col()
geom_boxplot()
geom_histogram()
geom_point()
geom_line()
ggtitle()
xlab()
ylab()
```

There are 7 question checkpoints in **bold**. Answer them directly in your R Markdown file, outside of the chunks.

## Section 1: ggplot and blank graphs

1. Open a new RMarkdown file.

2. Today we'll be using *ggplot2*, an R package that makes customizable graphs. In order to use a package, you'll first have to install it using the function `install.packages()`. Type `install.packages("ggplot2")` in your console and press enter.

   - You only need to install the package once, and it'll be available every time you open RStudio until the next time you update R.

3. For each new session of RStudio, you need to load in all the packages you want to use by running the `library()` function. Add `library("ggplot2")` to a chunk in your RMarkdown file and run it.

4. Let's start by making a box plot. Download the file `Wentworth_PAH_ER.csv`, move it into your working directory, and import it using `read.csv()` like you did last lab. Assign this to a variable called `data.box`, like in the example below. Look at the data; it's a version of the data from Lab 1, but with additional PAHs measured. You'll notice that it's in a long format with only 3 columns; this is how ggplot2 prefers to take data.

- Note: If you changed your working directory to import files last time, that won't work in R markdown since it resets the working directory at the beginning of each chunk. You can try setting the working directory in the same chunk that you import the file, but it's better to fix your filepath instead. Let me know if you need help with this.

```
data.box <- read.csv("Wentworth_PAH_ER.csv")
```

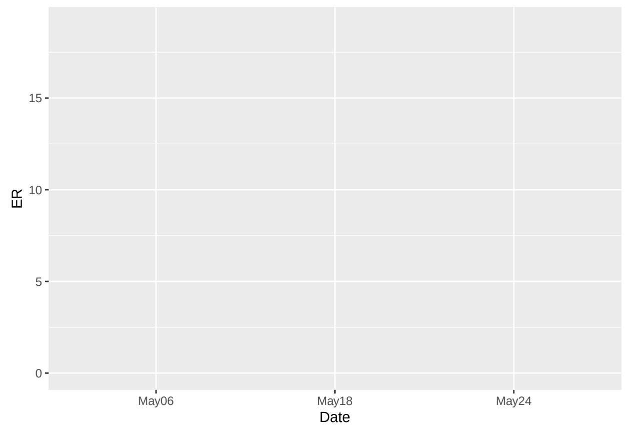
6. ggplot2 is part of a group of packages that use a modified syntax compared to base R (base R is R without any packages). Here's the basic syntax for ggplot2:

```
ggplot( data=DATAFRAME, aes(x=XCOLUMN, y=YCOLUMN) ) +
  #layers go here
```

- **ggplot()** is the main function to create a blank graph
- **data=** says that the following data frame contains the data to be graphed; optional as long as `DATAFRAME` is the first thing in `ggplot()`
- **DATAFRAME** should be replaced with the variable containing your data. This needs to be in a data frame format (so it should look like a table and appear in the "Data" section of the Environment in the top right panel)
- **aes()** defines the aesthetics of the graph
- **x=** and **y=** say that the following columns of `DATAFRAME` contain the x- and y- axis data, respectively. These should always go in the `aes()` function.
- **XCOLUMN** and **YCOLUMN** should be replaced with the name of the column of `DATAFRAME` that contains the x-axis data and the name of the column of `DATAFRAME` that contains the y-axis data, respectively. ggplot expects these to be columns, so you don't need to select for the columns like we were doing in the last lab. You can just put in the names.
- **+** goes at the end of a line *if you want to add another line* (a.k.a "layer") to that graph
- On the line after the `+`, you can add layers. We'll go over layers in a bit.

7. Let's see what a blank graph looks like. Copy and paste the code below into a new chunk, and then run it. A graph should show up without any data on it. Notice that there's no **+** at the end of the line because there are no lines after it. Only include **+** at the end of a line when you want to add another line to the graph. + Notice how the syntax is used here. Take note of the data variable, and the column names listed as x and y within `aes()`. These column names come from the variable you defined as `data`, which is `data.box` in this case.

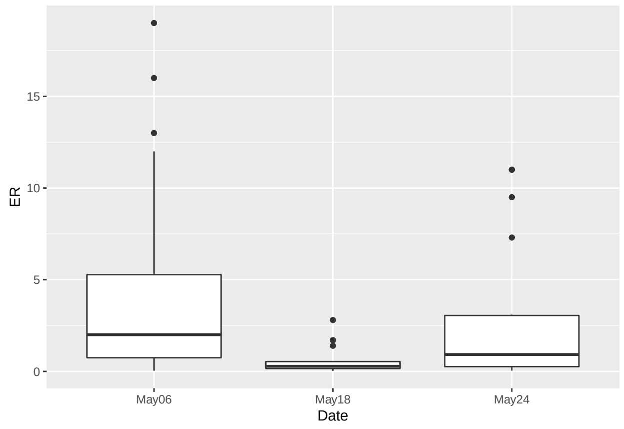
```
ggplot( data=data.box, aes( x=Date, y=ER ) )
```



## Section 2: Box and bar plots for descriptive data

8. In order to actually graph the data onto our plot, we need to add a line that specifies what kind of plot we want. This is called a *layer*. To make a box plot, we'll use `geom_boxplot()`, as shown below. Add the box plot layer to your blank graph and run it. (Note the `+` at the end of the first line!)

```
ggplot( data = data.box, aes( x=Date, y=ER ) ) +
  geom_boxplot( )
```



9. Congrats, there's your first graph in R! Let's try a bar plot next. Whereas box plots use all of the data at once to determine quartiles and outliers, bar plots only use a single value for each bar. Therefore, instead of inputting all of the data into the bar plot, we need to only input the specific values we want to graph (in this case, the means). We know how to do that from our first lab, but if you look at the data you imported before as `data.box`, you'll notice that the format is a little strange.
   - The data is currently in what's known as a "long" format. This is the format you need when using many of ggplot's features. However, it's not very easy to find the means in the format. So, let's use a packaged called `tidyr` to get the "wide" format.
   - Start by installing the `tidyr` package using `install.packages("tidyr")` in the console. Then, add `library("tidyr")` to a chunk to load it in. Remember: you only need to install a package the first time you use it, but then you need to load it in each time you create a script or markdown that uses it.

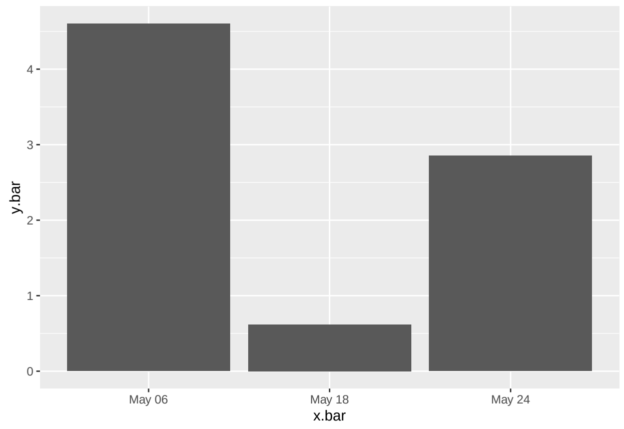
10. Now we need a function called `pivot_wider()`. Inspect and copy the code below.
    - `data.box` is the original data that we want to reformat.
    - `values_from = ER` tells R which column of `data.box` contains the values.
    - `names_from = Date` tells R which column of `data.box` contains the names of the new columns we want.
    - Therefore, this code should result in a new variable (`wide.data`) that has a date for each column, and ER values as the cells in each column.
    - After running the code, inspect `wide.data` to confirm.

```
wide.data <- pivot_wider(data.box, values_from = ER, names_from = Date)
```

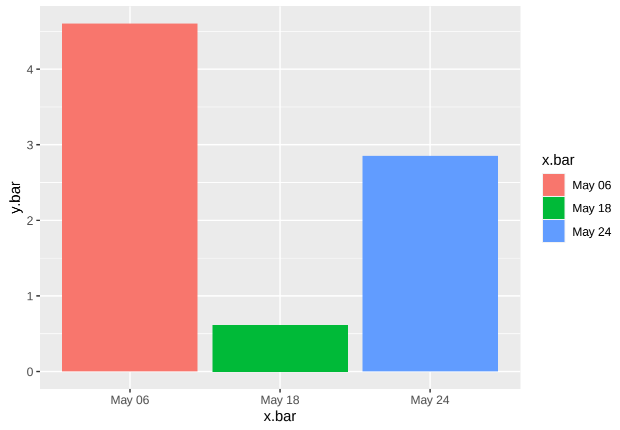
11. With this format, you should be able to get the means of each row using the same method as in Lab 1. Assign the mean of May 6 to the variable `May06`, the mean of May 18 to `May18`, and the mean of May 24 to `May24`. Then, combine them all using the code below.

```r
x.bar <- c("May 06", "May 18", "May 24")
y.bar <- c( May06, May18, May24 )
data.bar <- data.frame(x.bar, y.bar)
```

12. `data.bar` should now contain the means for each of the three dates. This data is now ready to be input into a bar graph. Using the same format as that of the box plot, try creating the bar graph yourself this time. `x.bar` should go on the x axis, `y.bar` should go on the y axis, and the data should all come from the `data.bar` variable.
    - This time, instead of using `geom_boxplot()`, use `geom_col()`. This new function is how you make a bar plot layer.
    - **Q1. What is an advantage of using a box plot over a bar plot? What is an advantage of using a bar plot over a box plot?**



13. Try making the graph look a little nicer now. You can do so by adding color.
    - You can actually add attributes such as `aes()` to layers besides `ggplot()` too. The rule is that anything in that first `ggplot()` line will be inherited by all of its layers below, but if you only want a specific layer to have an attribute, you can specify it in that specific layer and none of the other layers will inherit it.
    - Within the parentheses of the `geom_col()` layer, add an `aes()` function that contains `fill=x.bar`, like so: `geom_col( aes(fill=x.bar) )`. This will fill each bar with a different color according to the `x.bar` column.
    - This aesthetic is specifically listed under the `geom_col()` layer, so only the bars of the bar plot are affected by it. Since you only have one layer, this won't matter much, but it's good to know in the future.



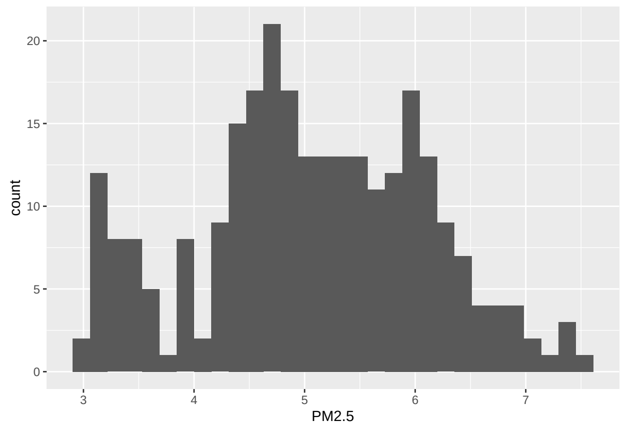
## Section 3: Histograms and scatter plots for distributions

14. For the next two graphs, let's use air quality data from a recent day with no fire: December 1, 2021. This data is taken from the same database used in the paper we're reading, the WBEA, from the site AMS 7. Download and import `WBEA_PM_O3.csv` from Canvas, then assign it to a variable (The O in the file name is a letter, not a zero). I'll name my variable `WBEA.data` and refer to it as such for the rest of the lab. As usual, take a look at the new data so you know what you're working with.

```
WBEA.data <- read.csv("WBEA_PM_O3.csv")
```

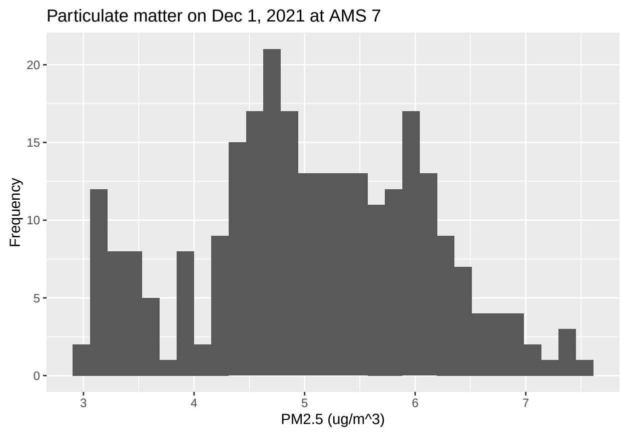
15. Let's investigate the particulate matter levels that Fort McMurray might see on a normal December day, using a histogram. Use `WBEA.data` and put the column `PM2.5` on the x-axis. Histograms show the counts along the y-axis, so we won't define a y value. The layer to use is `geom_histogram()`. (You might get a warning about bin size, but the default size is fine here so don't worry about it)

    - **Q2. What are some uses you can think of for histograms?**



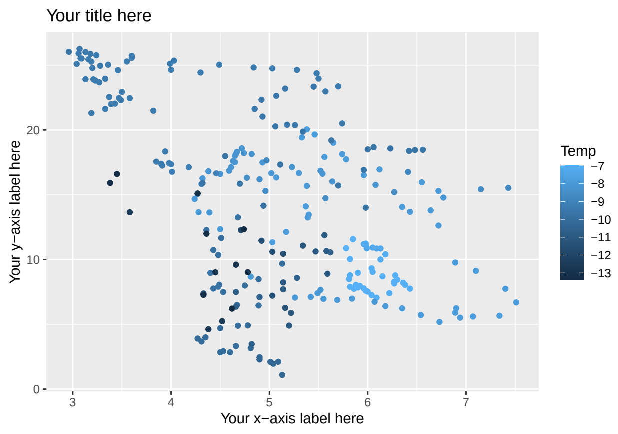
16. Let's try customizing this histogram a little more. Add the code below to your histogram to add a title, x-axis label, and y-axis label. Don't forget the + at the end of the preceding line.

```
ggtitle("Particulate matter on Dec 1, 2021 at AMS 7") + # plot title
xlab("PM2.5 (ug/m^3)") +                                # x-axis label
ylab("Frequency")                                       # y-axis label
```



17. A scatter plot will show every data point that we have. Using the `WBEA.data`, create a scatter plot that shows PM2.5 (ug/m^3) vs O3 (ppb) levels. `geom_point()` is the layer for scatter plots, but we can add temperature data by changing the color. This time, the color controls points instead of filling in bars like on the bar plot, so use `color=Temp` instead of `fill`. Remember that you need to define the color within `aes()`. Finally, add an appropriate title, x-axis label, and y-axis label.

   - **Q3. Describe an example scenario in which it would be most appropriate to visualize your data with a scatter plot. What would go on each axis?**
   - **Q4. What is one hypothesis that this scatter plot would support?**

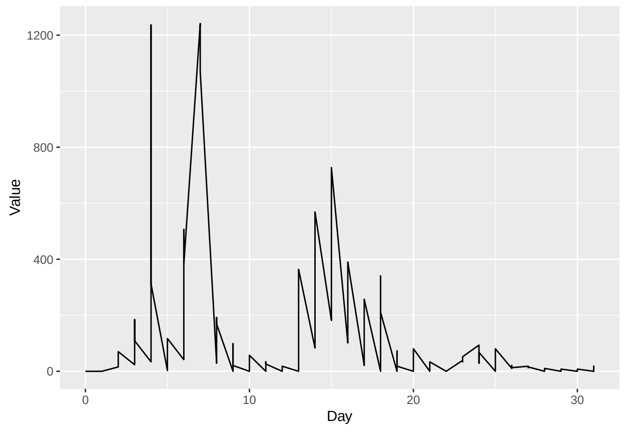


## Section 4: Line plot with particulate matter data

18. For this section, we're going to use approximations of the PM2.5 data from figure 2a-c of the paper to compare particulate matter levels of three different sites in May 2016. Download and import `Wentworth_fig2.csv` from Canvas. I'll assign the dataset to the variable `data.line` and refer to it as such for this lab.

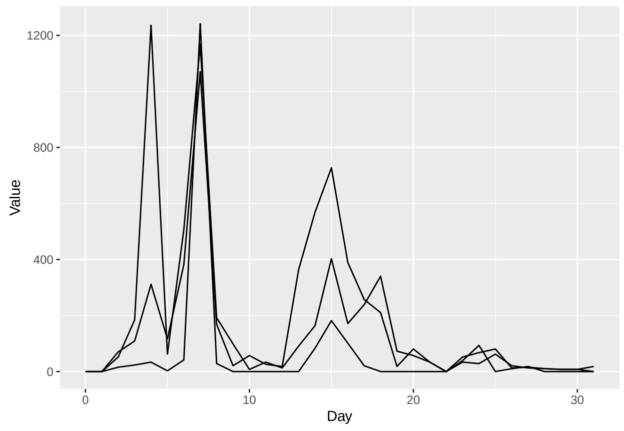
   - As always, take a look at the dataset before you do anything with it. Notice that it's already in a long format, which ggplot2 likes.
   - Day 0 means April 30, day 1 means May 1, day 2 means May 2, and so on.

19. In a new chunk, start by creating a ggplot from `data.line` with `Day` on the x-axis and `Value` on the y-axis. Add a line plot layer with `geom_line()`, and as always don't forget the `+` on the preceding line.



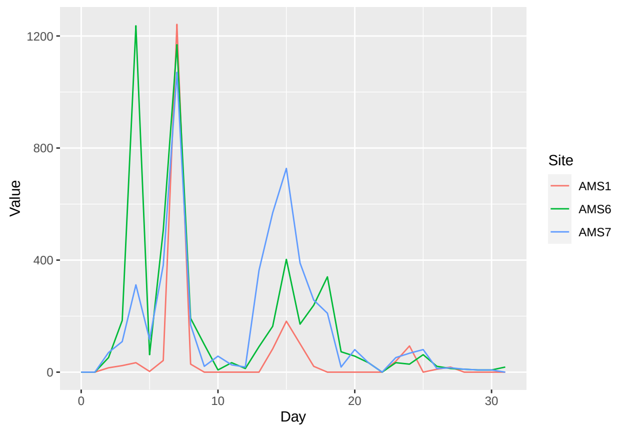
20. When you run the chunk with your line plot, you should get the strange result seen above. If you take a look at `data.line`, you'll see that the data frame actually contains data for three different sites: AMS1, AMS6, and AMS7. It would make sense to plot each site as a separate line, but we haven't told R to do that, so it's trying to combine them all in one.

    - Adding `aes( group=Site )` inside the `geom_line()` layer's parentheses will tell R how to group the data into different lines. Once you do this and run it, your new plot should have three separate lines. Much better!
    - We're adding this aesthetic to the line plot layer specifically because the `group` attribute is specific to this layer type.



21. Unfortunately we still don't know which line goes to which site. You can differentiate them by color like we've been doing by using `color=Site` in the `aes()` parentheses of the first `ggplot()` line. Just like how we've already separated `x` and `y` with commas, you need to separate every new attribute with commas too, i.e. `aes( x = Day, y = Value, color = Site )`.

    - This will also work if you put it in the `geom_line()` layer, but we're going to add another layer soon that will also need color so it's better if you define the color in the `ggplot()` line and allow both of the layers below it to inherit that feature.

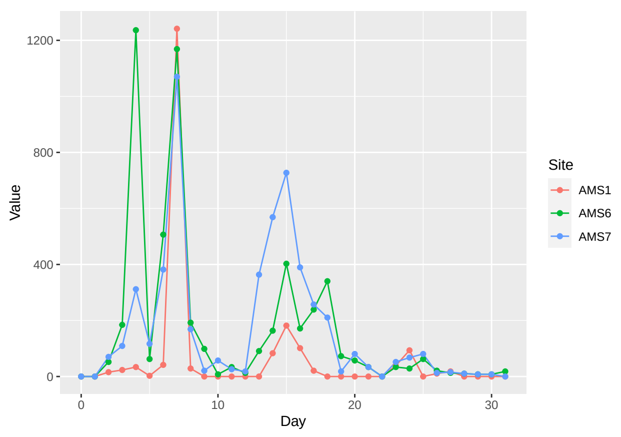


22. Another handy thing you can do with graphs is assign them to variables. Assign your line plot to `PM.plot` by adding `PM.plot <-` to the beginning of the first line, right before ggplot. It should go on the same line as ggplot, and all the layers after it will be saved with it. To make sure you've done this correctly, run it and look for it in your object environment on the top right. It should be called `PM.plot` and be shown as a "`List of 9`".

- NOTE: When you assign a plot to a variable and run the code, the plot itself won't be output. If you want to output the plot, you can just type the variable name now that it's assigned to the plot.

23. Sometimes, line plots look better with points clearly indicated on the lines. We can do this by adding a new layer in the same way we made the scatter plot: `geom_point()`. Since we define the color aesthetic in the parent ggplot line, this layer will inherit that value and the colors should correspond to the lines too. + The good thing about layers is that you can add several on top of each other. In this case, we're basically drawing both a line plot and a scatter plot on top of the same grid, so it looks like a line plot with points. + We already have a variable for the plot, so we'll use that and add the new layer on top of it. Note that there should still be a `+` after the variable to indicate that the next line is a layer, like so:

```
PM.plot +
  geom_point()
```



24. As a final step, give the line graph a title and x and y labels. Use the earlier graphs as a guide on how to do that. The question checkpoints below focus on interpreting the figures, which is a little subjective without specific stats, but do your best!
    - **Q5. You'll notice that this combination of data isn't included as one graph anywhere in the paper. What research question is this line graph answering that the figures in the paper don't answer?**
    - **Q6. What do you think is interesting about this graph, in terms of the data? This is a place where you can just write down some thoughts on what you see and how you would interpret the graph. Just a couple sentences is enough.**
    - **Q7. Compare this line plot to the histogram from before. The line plot shows data from May 2016, during a wildfire. The histogram shows data from December 1, 2021, which was not during a wildfire. Taken together, what do you think that these two figures say about PM2.5 levels?**

25. Feel free to play around some more with the graphs! If you're interested, you can look up ggplot2 to find more ways you can customize plots, but these are the basics. When you're done, make sure you've answered Q1-Q7 outside of the chunks in your R Markdown file, then knit your file to PDF or HTML and submit it. Make sure you're submitting a knitted PDF or HTML report or you'll lose points.

## Reference

Wentworth, G.R., Aklilu, Y., Landis, M.S., Hsu, Y.-M. **Impacts of a large boreal wildfire on ground level atmospheric concentrations of PAHs, VOCs and ozone.** Atmos. Environ., 178 (2018), pp. 19-

30, 10.1016/j.atmosenv.2018.01.013

WBEA. **Historical Monitoring Data.** Wood Buffalo Environmental Association (2016) Available from: http://wbea.org/network-and-data/historical-monitoring-data, Accessed Jan 2022