# Lab 1: Descriptive Statistics

This lab will teach you how to find descriptive statistics in R. By the end of the lab, you should know how to select a column from a dataset, declare variables, and use the following functions:

```
read.csv()
mean()
summary()
var()
sd()
```

There will also be 4 question checkpoints in **bold**. Please answer these in your preferred type of text file (.doc, .pdf, .txt, comments in .R, etc.) and submit them along with your saved script.

## Section 1: Importing data

1. Before starting, let's talk about variables. In R, you can create something called a variable which will hold a value or data for you using the `<-` symbols. You can then use the variable in place of the value or data itself in your code. It's like a variable in math, but you can use it for more than just math as we'll see in a moment.
    - Variable names can be whatever you like as long as they start with a letter and only contain letters, numbers, periods, and underscores (no spaces!). The syntax to assign a variable is: `variable.name <- whatever.youre.assigning`.
    - As an example, if you wanted to add together 4+5, you could write the following and the last line would give you the same result as 4+5. The result of the code is shown by the `> [1] 9` at the bottom.
    - Note that the `#` symbol in the code itself creates a comment; everything after it on the same line (shown in brown) won't run as code, so you can type notes into your script without it messing up your code.

```
X <- 4 # assigns the value 4 to the variable X
Y <- 5 # assigns the value 5 to the variable Y
X + Y
```

```
> [1] 9
```

2. Try assigning some variables in your console and observe how they appear in the environment panel at the top right. Then, try calling the variables by entering them in the console.

3. Now, open a new R script file.

    *From here onward, make sure you're typing everything into your script and not the console, so that you can save and save often! Remember to run your script periodically to make sure it works. Grading is based on what happens when it is run from top to bottom in a clean environment.*

4. Today we'll be using data from Wentworth, et al. (2021). The data was collected during the Fort McMurray Horse River Wildfire, which burned from May to July 2016 and resulted in an evacuation off Fort McMurray on May 3. These data were specifically measured at AMS 7, a sampling site in Fort McMurray itself, within the burn area of the wildfire.

5. Download the dataset from the course website (entitled Wentworth_et_al_data.csv) and move it to your working directory. You can check your working directory by going to the file tab of the bottom right panel in RStudio, clicking on the "More" dropdown, and choosing "Go to working directory". This should be the folder that contains your R script and project files.

6. Now we need to import the file into R. Use `read.csv("FILENAMEHERE")` but replace FILENAMEHERE with the *exact* name of the file you're importing, including the file extension and keeping the quotations marks intact. The filename should end in .csv. + Because the file is in our working directory, we don't need to specify where the file is on our computer. If it were somewhere else, we would need to specify the exact folder it's in.

7. Assign the file to a variable name. Here's what my final file import code looks like:

```
all.data <- read.csv("Wentworth_et_al_data.csv")
```

As you can see, I'm reading in the file `Wentworth_et_al_data.csv` and assigning the result to the variable `all.data`. Now, every time I type `all.data` R will know that I'm talking about this dataset; I'll use this variable name moving forward. If you ever want to check what's in a variable, you can type its name in the console, use `View()`, or click on it in the Environment tab of the top right panel (some of these strategies will only work for certain types of variables, which we'll go over next week. If one option doesn't work, try something else.).

## Section 2: Getting to know our data

8. Let's take a look at our data using the function `View()` . Type `View(data.df)` into the console and press enter to see PAH enhancement ratios from AMS 7.
    - Note that R is a case-sensitive language, so we need to make sure that `View` has a capital V. Your variable name should also match by case (capital and lowercase letters).
    - We use `View()` in the console because it brings up the object in a new tab, so we don't want it to run every time we run the script itself. Anything you want to save goes into the script itself, but everything else can be run from the console.

9. Familiarize yourself with the data. This is always the first step of any data analysis.
    - Each column represents a different polycyclic aromatic hydrocarbon, or PAH. BaP is benzo(a)pyrene, Chrysene is simply chrysene, and DahP is Dibenzo(a,h)pyrene.
    - Each row represents the day on which the sample was taken. These are May 6, May 18, and May 24.
    - The values are the enhancement ratios for each PAH, in the units (ng mˆ-3 ppmˆ-1 CO). Put simply, enhancement ratios represent the concentration of each PAH in the smoke plume that was measured, while also accounting for the concentration of the smoke itself. Therefore, higher values indicate a higher concentration of that PAH in smoke.

10. Let's take a closer look at the column `Chrysene` . One way to access a column is to use the format `object$column` . For example, to access the `BaP` column it would be `all.data$BaP` .

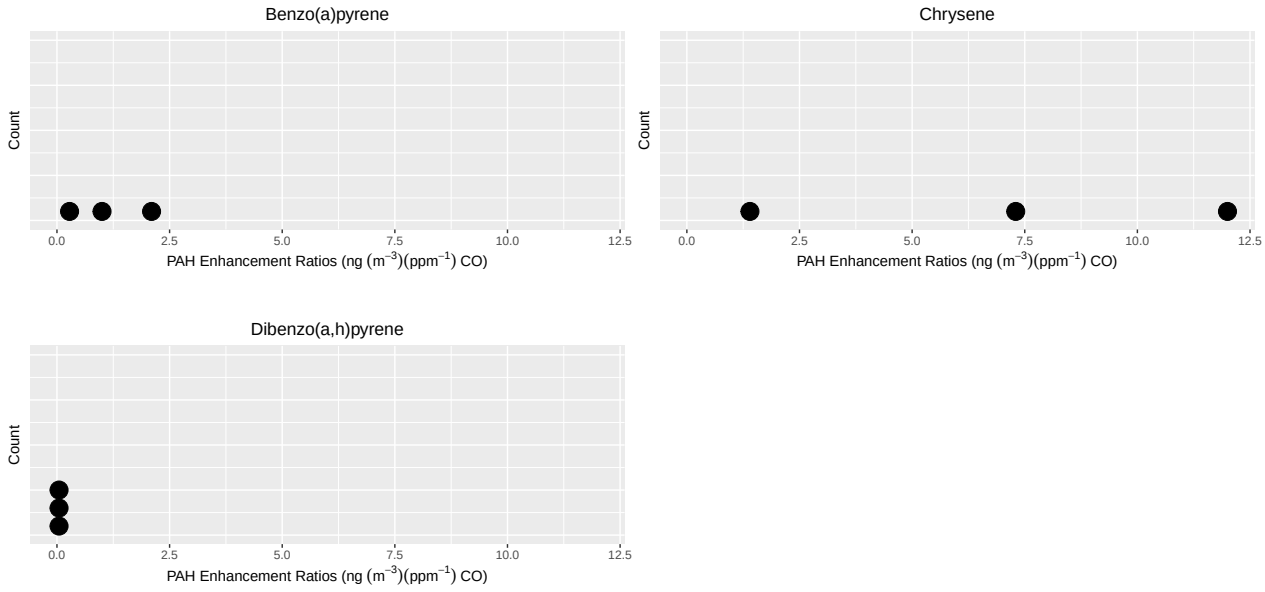
- A second way to access a column is with the format `object[ row, column ]` , but without specifying a row, so `object[ , column ]` . However, in this case, you can use numbers to specify the rows and columns. If you're using the row/column names instead of numbers, you need to surround the names in quotes. So, BaP would be `all.data[ , "BaP" ]` or `all.data[ , 1 ]` since it's the first column .
- A third way is to use double square brackets, like so: `all.data[["BaP"]]` . Again, you can use column numbers instead of column names if you leave out the quotes, as in `all.data[[ 1 ]]` .
- The fourth way is very similar to the third, but it only used one set of brackets, like so: `all.data["BaP]` or `all.data[[ 1 ]]`. Otherwise, the usage is the same, but sometimes a certain method won't work for what you're trying to do so you'll have to switch to another one.
- **Q1. What are four ways in which we can access the `Chrysene` column? Write the exact code we would need for the `Chrysene` column.**

11. Using one of the methods in step 3, assign the BaP data to the variable `BaP` on one line, assign the data from chrysene to `Chrysene` on the next line, and assign the data from DahP to `DahP` on the next line after that. Once you run this code, your three variables should appear in the environment tab in the top right panel. You should also be able to type the variable name into the console and have it return the values for that variable.

    - For BaP, this could look like `BaP <- all.data$BaP`. Do the rest on your own.
    - Now, anytime you want to refer to data from either of those three columns, you can simply use the variable name instead of retyping step 3.
    - Remember that variables are case sensitive, so it matters which letters are capitalized! You also have to type the variable name exactly how it was created in order to use a variable, so watch out for typos.
    - These variables will stay exactly the same unless you change them by assigning something new to them, even if the original data changes. Essentially, if you want the variable to update, you have to re-run the code that you used to create the variable.

12. Below are dot plots for each of the three PAHs we're looking at.

    - **Q2. Without making any calculations or using R, which PAH do you think has the greatest standard deviation, based on the dot plots? Which do you think has the smallest standard deviation?**



## Section 3: Calculating descriptive statistics

13. Let's start by analyzing chrysene data with the `mean()` function. If we wanted the mean of BaP, we would type `mean(BaP)` into our script (notice how we're using the variable we created). Do the same, but for chrysene.

14. Use the `summary()` function on all three PAHs to get quick summary statistics. You will need to run each summary on a separate line.

15. Find the variance and standard deviation using `var()` and `sd()` respectively, for all three PAHs.

    - **Q3. Do the results match your prediction from Q2? Why do the results make sense?**

    - **Q4. Taking any of the statistics you calculated above, what is one rough, preliminary conclusion you could draw about the three PAHs? You only need to write 1-2 sentences here.**

16. Feel free to play around some more with the data! Once you're done, submit both your final script and your answers to the questions in **bold**. (Make sure you've answered Q1-Q4)

**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