Lab 2 Activity

In this lab activity we are going to use the `attitude` dataset, which is already loaded into R.

1. Run `help("attitude")` to get some info on the data and the meaning of each variable. You can also run `View(attitude)` to open the data in the data viewer window.

2. plot `rating` on the y-axis and `complaints` on the x-axis. If you were to run a regression, do you think the slope (b_1) will be positive or negative? Why?

3. Run a linear regression with `complaints` predicting `rating`.

- What do you conclude about the relation between rating and complaints?
- How much would we expect `rating` increase to be on average if `complaints` increased by 3 units?

4. Run a *standardized* regression with `complaints` predicting `rating`. What changes do you see in the `summary()` output?

5. What is the predicted value of `rating` in standardized units when `complaints` is 1 standard deviation below average?

Tricky questions

Predictions based on regression represent the *mean* expected value of Y given some value of X. Then if the formula to standardize any variable (Var) is:

$$\text{Var}_{\text{std}} = \frac{\text{Var} - \text{Var}_{\text{mean}}}{\text{Var}_{\text{SD}}}$$

Try answering the following questions:

- can you convert the value of `rating` from question **5** back into unstandardized units? (HINT: you will need to use the mean and standard deviation of the original `rating` variable)
- How do you get the same value using the unstandardized regression equation? (HINT: you need to use the mean and standard deviation of the `complaints` variable)

Some R Practice: Functions

Functions are the foundation of R and many other programming languages. A function works in **3 steps**:

1. The function take in some objects and possibly instructions (input).
2. The function runs some R code that performs operations based on the given objects and instructions.
3. The function returns the result of the R operations in step 2 (output).

Here is how you define a function in R:

```
# you need to name you function as you wold a normal R object
# the function() part is used to define the function arguments (the inputs that the function accepts)

new_function <- function(input){

  # run some code that does something to the input


  # the function needs to return some output
  # you specify what is returned by using return()

  return(output)
}
```

So, for example, we can create a function that calculates the mean of any given vector:

```
# we specify `vector` as the only function argument

mean_function <- function(vector){

  mean <- sum(vector)/length(vector)

  return(mean)
}


# now we can call our function and input a vector

x <- c(1, 2, 4, 5, 6)

mean_function(vector = x)
```

```
## [1] 3.6
```

We can see that this is the exact same result as the `mean()` function in R:

```
mean(x)
```

```
## [1] 3.6
```

Note that you can call a function argument whatever you like as long as you are consistent:

```
mean_function_2 <- function(ice_cream){

  name_doesnotmatter <- sum(ice_cream)/length(ice_cream)

  return(name_doesnotmatter)
}

# will produce the same exact output
mean_function_2(x)
```

```
## [1] 3.6
```

Why use functions? Usually you have some code that you run multiple times where you only change a few things. Functions help you not having to copy and paste code over an over, as well as making code more readable. Of course, there are MANY more benefits to being familiar with functions and using functions.

Tasks:

- can you create a function that computes the standard deviation of a vector? The function that you create must only use the `sum()`, `mean()`, `sqrt()`, and `length()` functions. To test your function, try it on both `c(1,5,66,7,4,3,2)` and `c(17,34,23,2)` as inputs. Then, check that you get the same result as:

```
x1 <- c(1,5,66,7,4,3,2)
sd(x1)
```

```
## [1] 23.64217
```

```
x2 <- c(17,34,23,2)
sd(x2)
```

```
## [1] 13.34166
```

The formula for the sample standard deviation of a variable X is:

$$SD_X = \sqrt{\frac{\sum(X - \bar{X})^2}{N-1}},$$