Lab 9. Correlation

# Introduction

Today you will be playing around with a very basic function and using it to look for relationships to help you strike it rich (or, well, sort of). You’ll walk through some fake examples to make sure you understand what a correlation coefficient is, then you’ll analyze some real data from a gold mining claim to see if you can figure out what element you could look for if you wanted to find gold.

# Learning Outcomes

By the end of today’s class you should be able to do the following in R:

* Use the cor() function to calculate Pearson’s correlation coefficient (r)
* Interpret both positive and negative r values
* Use the jitter() function to make fake data
* Use the plot() function to create a scatterplot
* Use the corrplot() function to create a correlation diagram
* Find gold???

# Part 1: Libraries

For todays lab, you will need a few different libraries: ggplot2 and a new package, the corrplot package. Install corrplot using install.packages(), then load both libraries.

# Part 2: Fake Data

Before you do the rest of the lab, let’s make sure you understand how to interpret the correlation coefficient (**R** or **r**).

## Pat 2.1 Make the Data

First, let’s make a random normal distribution (**x**), and we will make y just be the exact same thing as x.

x <- rnorm(n = 50, mean = 30, sd = 2)  
y <- x  
test <- data.frame(x = x, y = y) #data frame format for ggplot

This should have perfect correlation - after all, they are the exact same numbers. If you need to see them to believe me, make a scatterplot! Here are two chunks of code, using either ggplot or not.

plot(x, y)

ggplot(test, aes(x=x, y = y))+ geom\_point()

## Part 2.2 the cor() Function

Now, double check using the cor() function. It doesn’t matter whether you put x or y first!

cor(x,y) #for the values  
cor(test$x, test$y) #for the data frame version

Perfect correlation (1)! But an R of 1 doesn’t mean you have a 1:1 relationship where x and y are the same. You can also get this relationship with slightly different changes. Try comparing x and y using the cor() function on the following:

y <- x \* 50  
y <- x + 5000  
y <- x - 5000

### QUESTION 1: If you get a correlation coefficient of 1 or -1, that means:

1. Rejoice, for your science is perfect
2. Double check whether your x and y are the same thing with a different scale
3. Disregard your study results

## Part 2.3 Adjusting Correlation

So what would the correlation look like if change in y wasn’t perfectly correlated to change in x? Below are a few variations to substitute in for y. Run these, think about how they are or are not related to x, then answer the questions below.

y1 <- rnorm(n = 50, mean = 48, sd = 2)  
y2 <- x-3  
y3 <- jitter(x)  
y4 <- x \* -1

### QUESTION 2: Which of the above variables would have a correlation closest to zero?

1. x, y1
2. x, y2
3. x, y3
4. x, y4

### QUESTION 3: Which of the above variables would have a correlation closest to negative 1?

1. x, y1
2. x, y2
3. x, y3
4. x, y4

## Part 2.4: Visualizations

Sometimes it’s nice to see a bunch of correlations at once. We’ll use a simple example from the corrplot package, which makes these diagrams simple.

In order to make a correlation plot, you need to find correlation coefficients for a lot of categories all at once - creating a **correlation matrix** which is basically a contingency table of r values. You can use the cor() function itself to make the matrix! First, we’ll put all of our x and y categories from the previous question into a data frame:

all <- data.frame(x, y, y1, y2, y3, y4)

Then, use the cor() function to make a matrix:

M <- cor(all)

Now there are a few different options for graphics. You can use the *circle*, *color*, or *number* method. Try out each to see which you like the best:

corrplot(M, method = "circle")

corrplot(M, method = "color")

corrplot(M, method = "number")

### QUESTION 4: What color is used for r values where the two variables have almost no relationship?

1. Pale colors
2. Blue
3. Red

# Part 3: Assumptions

Though we’ve been using the cor() function already, we haven’t set any arguments. Like t.test(), it has quite a few! Use the ? indicator to figure out what the arguments are.

?cor

### QUESTION 5: According to the help documentation, which method of correlation is default if you don’t specify?

1. Pearson
2. Kendall
3. Spearman

The other argument accepted by cor is the **use** argument, which tells the function what to do if you have an NA value. In the test set, we don’t have any of those - so let’s take the 5th observation of our x value and turn that into an NA, like so:

x[5] <- NA

Now, run the cor() function on x and y to see what happens. Then, try the different argument options for **use** as listed in the help documentation.

### QUESTION 6: Which use option is the one chosen when you do not specify a use argument (aka - which one is the default)?

1. FALSE
2. everything
3. all.obs
4. pairwise.complete.obs

# Part 4: Real Data

Now that you’ve played around with some fake data, your task for this lab is pretty simple. You will be using a dataset from a mining site, where researchers were comparing the amount of different elements in some very old rocks to see if there was a relationship between gold (**Au** is the element acronym for gold) and different elements like Titanium, etc.

See, gold in nature can be diffuse or concentrated. If it’s diffuse, that means it isn’t easy (or sometimes possible) to see with the naked eye - but it can still potentially be mineable. The problem is you have to be able to *find* it, and if the gold is small and spreadout that might not be possible - unless there is a mineral that is usually found with it, because that mineral is made by the same mechanisms that make the gold.

So for example, if gold and pyrite are stongly correlated, that means that whenever you see pyrite crystals (which are often bigger and more obvious), that is a good place to also find gold. This is called a **proxy** - it’s something you can use to measure something else that is hard to measure, essentially.

So your task is to find the element in this dataset that has the best possible correlation with gold, as it’s possible you could use the mineral containing that element to mine gold more effectively.

Use the cor() function and/or the plot function to test the **Au.oz.ton** column against other columns in your dataset. You will be determining which of these elements is the best **proxy** for gold - that is, which has the strongest correlation with gold.

However, it’s worth noting some things:

1. If your correlation coefficient is *too* good, it might not be a different element…
2. Not all of the data in here is numeric continuous, which can behave… erratically.
3. There are NA values in many columns - use “complete.obs” to get the right answer

For the first and second issues, make sure to pay close attention to what you’re doing. You won’t get credit if you accidentally compared gold to itself, or to some sort of non-numeric data.

**NOTE:** if you want to try to use the corrplot() function, you may need to drop some columns - go back to lab 8 if you don’t remember the code for that!

### QUESTION 7: Do you want to find a postive or a negative correlation to answer this question?

1. A negative correlation
2. A positive correlation
3. It doesn’t matter, as long as it is close to 1 or -1
4. It doesn’t matter, as long as it is close to 0

### QUESTION 8: If you could only test for one element in this data sheet and could not test for gold itself, which element would you reccomend someone use to try to help them find gold? Use your R value to defend your answer.

### QUESTION 9: Is your element more abundant with gold, or very scarce around gold? Use your R value to defend your answer.