Lab 01 - Review of Statistical Concepts

PSC-103B

Marwin Carmo

2025-01-09

Central tendency: Mean, Median, and Mode

There are many measures of central tendency, but these 3 are the most common. They are used to describe a distribution of observations (e.g., all the grades on an exam) in one number that best represents that distribution. Let's see how this works.

First, let's create some variables! Suppose we asked a bunch of UC Davis students how many hours per week they spent watching Netflix, and how many hours they spent exercising during Winter break. We record their answers in two separate vectors:

```
netflix <- c(2, 6, 1, 7, 2, 4, 11, 40, 7, 0, 3, 4, 5, 2, 15)
exercise <- c(2, 2, 6, 2, 12, 45, 8, 3, 2, 6, 4, 0, 1, 3, 0)
```

How many observations are in each variable?

```
length(netflix)
```

```
## [1] 15
```

```
length(exercise)
```

```
## [1] 15
```

We have 15 people in our dataset now. Let's take a look at the average time each student spent on these activities:

```
mean(netflix, na.rm = TRUE) # use the argument na.rm = TRUE to ignore missing values
```

```
## [1] 7.266667
```

```
mean(exercise, na.rm = TRUE)
```

```
## [1] 6.4
```

Unsurprisingly, students exercised less than they watched netflix, on average. But is the mean a good representation of these data? Check out that person who looks like their job is watching Netflix. 40 hours a week? What about that athlete who exercised 45 hours per week over the break?

When we have outliers, sometimes the median is a better representation of the data we have. Remember, the median is the middle value of your data, after you have ordered it.

```
median(exercise, na.rm = TRUE)
```

```
## [1] 3
```

That's pretty different from what we had before!

Sometimes, we can't do arithmetic on the data we have. For example, if we had asked our 15 participants what their favorite flavor of ice cream was, we would not be able to describe that distribution using a mean or a median. That's when the Mode is useful: The mode is just the most frequent value. It's not used very often so R doesn't have a function for that. But you can use another very useful function to find the mode: `table()`

```
table(netflix)
```

```
## netflix
##  0  1  2  3  4  5  6  7 11 15 40
##  1  1  3  1  2  1  1  2  1  1  1
```

`table()` gives you the number of times each element shows up in an object. by looking at the results here, we can see that 2 shows up 3 times, so it's the mode of `netflix`.

To make it easier to see, you can use the sort function on the result of the table function to order it. Here's that for `exercise`. What's the mode?

```
sort(table(exercise))
```

```
## exercise
##  1  4  8 12 45  0  3  6  2
##  1  1  1  1  1  2  2  2  4
```

Spread: Variance and Standard Deviation

Imagine that I told you the mean number of hours students spent exercising each week over the break was 6.4 hours. You would know something about these students' exercise habits, but not a lot. Do all the students exercise about the same? Or do some students exercise a lot while others don't? These are the types of questions that measures of spread try to tackle: how are the observations spread out around the mean or median?

We're gonna look at two different kinds that are related: variance and standard deviation.

To get the variance:

1. Calculate the mean:

```
mean(exercise)
```

```
## [1] 6.4
```

2. Find the distance from each observation to the mean. R can do this for us pretty easily, because it is used to working with vectors. So when we do this:

```
exercise - mean(exercise)
```

```
##  [1] -4.4 -4.4 -0.4 -4.4  5.6 38.6  1.6 -3.4 -4.4 -0.4 -2.4 -6.4 -5.4 -3.4 -6.4
```

It will subtract 6.4 from each observation in `exercise`. Let's save that.

```
diffs <- exercise - mean(exercise)
```

3. Square the differences. Just as we did before, this is also very easy in R:

```
diffs_sq <- diffs^2
```

4. Sum everything and divide by N-1:

```
sum(diffs_sq)/14
```

```
## [1] 124.4
```

The variance of `exercise` is understandably a large number. Remember how that one athlete was very far from the mean? We can check our answers using `var()`

```
var(exercise)
```

```
## [1] 124.4
```

To get the standard deviation, we just get the square root of the variance:

```
ex_var <- var(exercise)
sqrt(ex_var)
```

```
## [1] 11.15347
```

Or, R has a `sd()` function:

```
sd(exercise)
```

```
## [1] 11.15347
```

We often prefer to use the standard deviation, because its units are the same units of our variable, unlike variance, which is units^2

Relationships between variables: correlation and covariance

Remember that in our imaginary example, we asked each person about both their exercise and netflix activity over the break. Therefore, each person gave us two bits of information. Let's organize our data into a dataframe to better keep track of it.

```
df <- data.frame(Netflix = netflix,
                 Exercise = exercise,
                 stringsAsFactors = FALSE)
```

Here, the capitalized names are the variable names, and I'm assigning the objects `netflix` and `exercise` to those variables. You can look at `df` by clicking on it, using `View()`, typing it in the console, or using functions like `head()` and `tail()`.

```
df
```

```
##    Netflix Exercise
## 1        2        2
## 2        6        2
## 3        1        6
## 4        7        2
## 5        2       12
## 6        4       45
## 7       11        8
## 8       40        3
## 9        7        2
## 10       0        6
## 11       3        4
## 12       4        0
## 13       5        1
## 14       2        3
## 15      15        0
```

```
View(df)
head(df)
```

```
##   Netflix Exercise
```

```
## 1       2        2
## 2       6        2
## 3       1        6
## 4       7        2
## 5       2       12
## 6       4       45
```

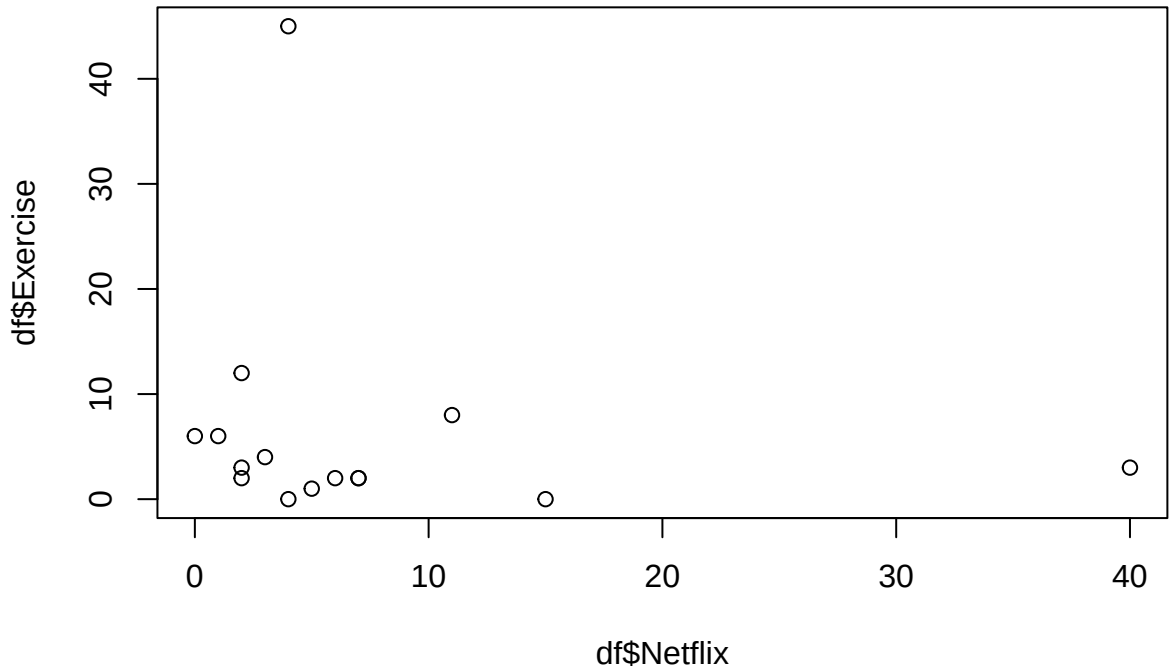
```
tail(df)
```

```
##    Netflix Exercise
## 10       0        6
## 11       3        4
## 12       4        0
## 13       5        1
## 14       2        3
## 15      15        0
```

It's also a good idea to plot your data. This helps you get a general idea for what your data looks like, and to see if there is anything weird going on (i.e., are there any large values in your dataset that could be outliers?)

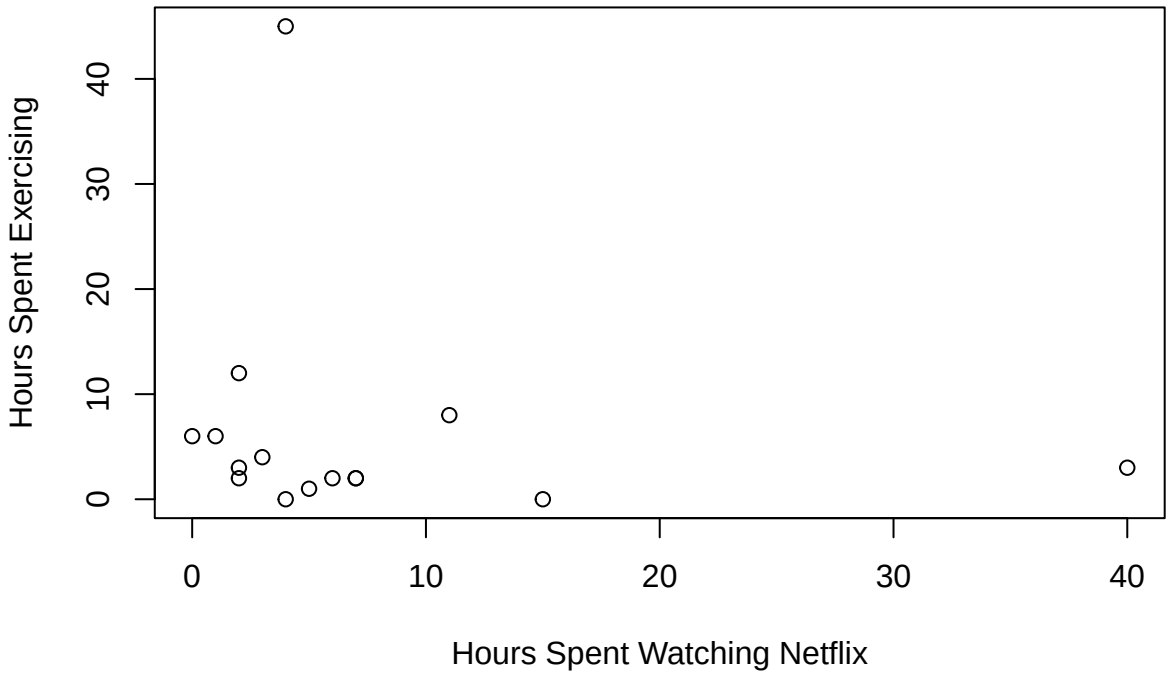
To make a scatterplot, we can use `plot()` function in R `plot()` works by plotting the first argument on the x-axis, and the second on the y-axis.

```
plot(df$Netflix, df$Exercise)
```



R just automatically sets the axis labels to whatever the input is. However, that's not very pretty, and depending on what you call your columns, it might not be very informative for other people to read. Luckily, we can change the axis labels, and even give our plot a title,

```
plot(df$Netflix, df$Exercise, xlab = "Hours Spent Watching Netflix", #changes the x-axis label
     ylab = "Hours Spent Exercising", # changes the y-axis label
     main = "Plot of Time Spent Watching Netflix vs. Exercising over Break") # gives your plot a title
```



This plot can also give us a general idea of the relation between our variables. For example, here it seems like the more time people spend watching Netflix, the less time they spend exercising. What if we wanted to quantify this relation? We can use the covariance and correlation!

Let's look at covariance first. The covariance between two variables is a measure of how the two variables change together. It only makes sense if there's some connection between the two variables. In this case, each row came from the same person.

The covariance is a bit like the variance, but instead of squared differences from the mean, we multiply these differences from the mean by each other. Let's use the variables in our new dataframe, `df`. Here are the steps:

1. Get the differences from the mean for each variable:

```
diff_nfx <- df$Netflix - mean(df$Netflix, na.rm = TRUE)
diff_ex <- df$Exercise - mean(df$Exercise, na.rm = TRUE)
# you can check this is right -- the differences will sum to 0
sum(diff_ex)
```

```
## [1] -3.552714e-15
```

2. Multiply them by each other (by row – R does this but you can check by hand, or compare the objects to see it)

```
mult_diffs <- diff_nfx * diff_ex
```

3. Sum all these multiplied differences

```
sum_diffs <- sum(mult_diffs, na.rm = TRUE)
```

4. Divide by N - 1

```
cov_NetEx <- sum_diffs/14
```

You can verify this yourself by using the `cov()` function

```r
cov_NetEx <- cov(df$Netflix, df$Exercise, use = "complete.obs")
cov_NetEx
```

```
## [1] -15.18571
```

For `cov()`, the `use = "complete.obs"` argument acts similarly to `na.rm = T` basically, it will only use data from people who gave an answer to both Netflix and exercise.

We see that the covariance is negative, indicating that the Netflix and Exercise variables are inversely related to each other: higher values for Netflix tend to go with lower values for Exercise, and vice versa!

But how strong is this association? We don't know, because covariances have arbitrary scales based on the scales of the original variables. We don't know how big they could get so we don't know if this value is large or small. That's not very helpful for us, so we need something we know the scale of: correlations.

Correlations can only range between -1 and 1, so they're easier to interpret. We'll standardize the covariance to get a correlation. Standardizing in this case means dividing by the variables' standard deviations. We already know how to get that:

```r
sd_N <- sd(df$Netflix, na.rm = T)
sd_E <- sd(df$Exercise, na.rm = T)
```

We divide the covariance by the product of the variances to get a correlation:

```r
cov_NetEx/(sd_N*sd_E)
```

```
## [1] -0.1377893
```

And we can check this using the `cor()` function:

```r
cor(df$Netflix, df$Exercise, use = "complete.obs")
```

```
## [1] -0.1377893
```

The answers match! Now, since the correlation ranges between -1 and 1, we can say something about the strength of this relation. This value is close to 0, so based on some rules of thumb we can say there is a weak negative relation between watching Netflix and exercise. Of course, what is considered strong vs. weak can depend on the area of research you're in.

Significance Testing for a Correlation

Now, we're going to use a fake dataset looking at the relation between temperature (in Fahrenheit), ice cream sales, and pool accidents.

```r
icecream <- read.csv("https://raw.githubusercontent.com/marwincarmo/phd/main/24-Winter/psc103b-wq24/lab
icecream
```

```
##    Temperature Sales      Pool
## 1        57.56   215 1.6101279
## 2        61.52   325 0.0000997
## 3        53.42   185 0.1348124
## 4        59.36   332 3.2882103
## 5        65.30   406 1.9550195
## 6        71.78   522 0.9825883
## 7        66.92   412 1.9631966
## 8        77.18   614 1.1486187
## 9        74.12   544 0.5999595
## 10       64.58   421 0.4459096
## 11       72.68   445 2.0538231
```

```
## 12       62.96   408 1.7382948
```

Previously, we talked about how to calculate the covariance and correlation between two variables – recall that you can do this using the `cov()` and `cor()` functions in R.

However, we're often working with more than 2 variables, and are interested in getting a quick summary of their pairwise relationships at once, rather than calculating each covariance or correlation individually.

Luckily, we can do this using the exact same functions we used last week! The only difference is that instead of inputting our data as `cov(x = , y = )` or `cor(x = , y = )`, we have just one data argument, which is our entire data frame. According to `cor()` documentation, `x` can be a numeric vector, matrix or data frame. By specifying only `x`, R understands that we want the covariance between all columns in the data frame.

Let's do this for the covariance first:

```
cov(x = icecream, use = "complete.obs")
```

```
##              Temperature        Sales       Pool
## Temperature  52.12939091   871.367727 0.05836522
## Sales       871.36772727 15886.810606 1.28329681
## Pool          0.05836522     1.283297 0.91458359
```

The output of this is called a variance-covariance matrix and it describes (as the name suggests) the variances and covariances of the variables in your dataset.

The diagonal elements of the variance-covariance matrix are the variances of the variables. Why? Because the way that a variable relates to itself is just its spread; you can use the covariance equation to see how it becomes the variance formula when the variable is the same.

The off-diagonal elements are the covariances between each pair of variables. Notice that this matrix is symmetric. The covariance between, say, temperature and ice cream sales is the same as the covariance of ice cream sales and temperature.

How can we interpret these values? For example, the covariance between temperature and sales is 871.37 – what does this tell us about their relation?

We can only tell that the relation between the two is **positive** – that is, the higher the temperature, the more ice cream is sold. But is this relation really strong? That's not something we can get from the covariance, and is why we turn to correlation instead!

To get the correlation matrix, we just use the `cor()` function.

```
cor(icecream, use = "complete.obs")
```

```
##             Temperature      Sales       Pool
## Temperature  1.00000000 0.95750662 0.00845281
## Sales        0.95750662 1.00000000 0.01064626
## Pool         0.00845281 0.01064626 1.00000000
```

Now let's look at these values – what are they telling us about the relations between our variables? Which relations appear to be strong? Which ones appear to be weak?

So even though we were able to calculate our correlations, we aren't really able to tell whether any of these values are representing significant relations – significant meaning that the correlation coefficient is different from 0.

Since a correlation coefficient of 0 means that there is no significant linear relation between the variables, let's test if there is a significant correlation between ice cream sales and temperature.

First, let's plot our data – we want to make sure that using the correlation to quantify the relationship between the variables is suitable. We wouldn't want to use correlation to measure what looks like a non-linear relation, because correlation and covariance can only capture linear relations. We also don't want to

necessarily use correlation if there are strong outliers in our data, as those outliers can change the value of our correlation coefficient.

```
plot(icecream$Temperature, icecream$Sales, xlab = "Temperature (F)", ylab = "Ice cream Sales")
```

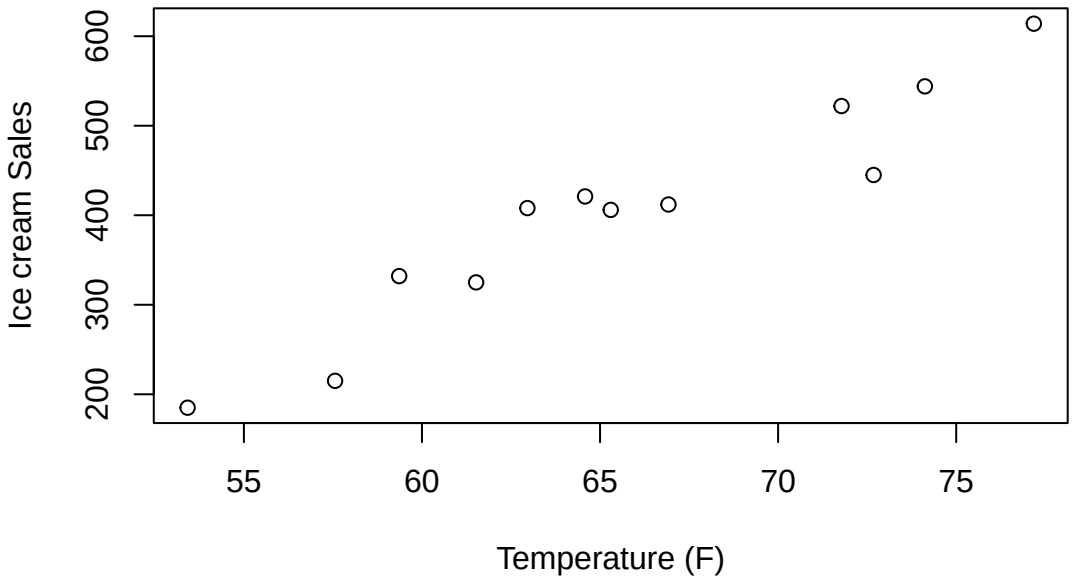


Figure 1: Scatterplot of Ice cream sales and temperature.

Does this relation look linear?

Yeah it does! So we are justified in using the correlation and testing its significance.

Let's proceed with the test.

Our null and alternative hypotheses are:

- H_0: $\rho = 0$
- H_1: $\rho \neq 0$

Notice that this is a **two-sided test** – either there is a relation or there isn't a relation, but we're not specifying which direction we expect that relation to be in.

In order to calculate our test statistic, we just need two things: the sample estimate of the correlation and the sample size. Recall that the formula to obtain the test statistic for the correlation estimate is given as

$$t_{N-2} = \frac{r_{xy}}{\sqrt{(1 - r_{xy}^2)/(N-2)}}$$

Now, let's get those things!

```
cor_estimate <- cor(icecream$Temperature, icecream$Sales)

sample_size <- nrow(icecream)
```

The denominator of our test statistic is an estimate of the standard error of the correlation

Let's calculate that separately since it's a bit long

```
cor_se <- sqrt((1 - cor_estimate^2) / (sample_size - 2))
```

Now our test statistic

```r
t_stat <- cor_estimate / cor_se
```

So we now have our test statistic This statistic is distributed as a t-distribution with df = N - 2

What is the df for our test? 10 (12 - 2)

If we wanted to see whether we reject or fail to reject our null hypothesis, we could calculate the critical value of our distribution and see if our value is larger or calculate the p-value.

I'll calculate the p-value.

```r
2 * pt(t_stat, df = sample_size - 2, lower.tail = FALSE)
```

```
## [1] 1.015893e-06
```

Notice that I'm multiplying my p-value by 2 because we're doing a two-sided test.

What do we conclude?

And here is how we could do it in one line of R code:

```r
cor.test(icecream$Temperature, icecream$Sales, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  icecream$Temperature and icecream$Sales
## t = 10.499, df = 10, p-value = 1.016e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8515370 0.9883148
## sample estimates:
##       cor
## 0.9575066
```

Since there are a few different ways to calculate correlation coefficients, we had to specify `method = "pearson"` because that is the method we've learned in class.

Here is an example of how we could write up our results:

> We examined the relation between temperature and ice cream sales in a sample of 12 days using a Pearson's correlation coefficient test. There was a positive, statistically significant, and large between temperature and ice cream sales, $r = 0.96$, $t(10) = 10.50$, $p < .001$.