Lab 2: Testing correlations

PSC 103B - Winter 2024

Read in Data

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

```
icecream <-
 ↪  read.csv("https://raw.githubusercontent.com/marwincarmo/phd/main/24-Winter/psc103b-wq2
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
```

Covariance and Correlation Review

Last week, 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?

Significance Testing for a Correlation

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")
```

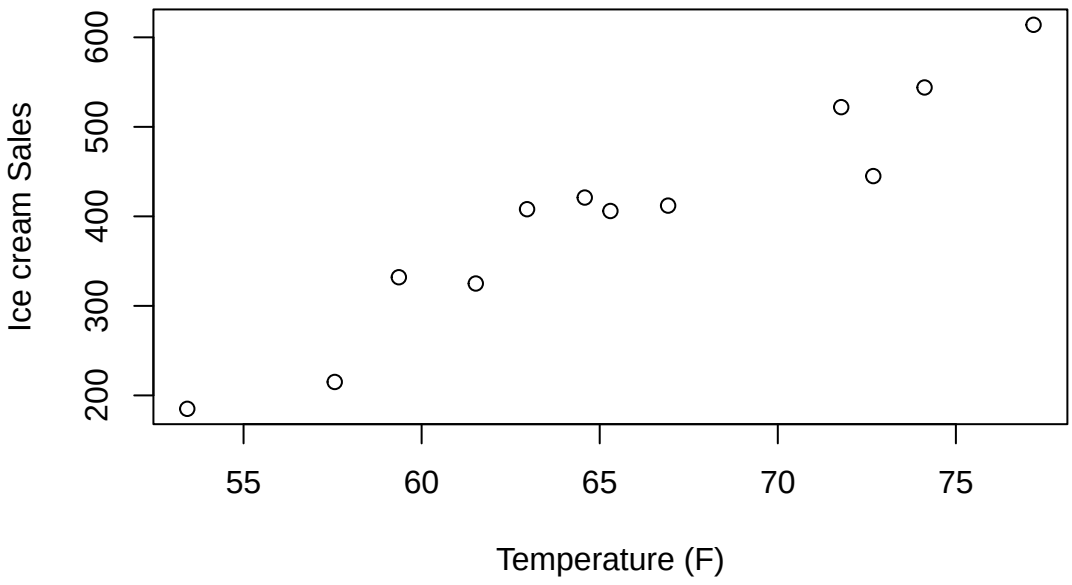


Figure 1: Scatterplot of Ice cream sales and temperature.

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

```
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.

```
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.

```
x <- seq(-5, 5, length = 100)
plot(x, dt(x, df = 10), col = "black", type = "l", xlab = "t-value",
 ↪  ylab = "Density", main = "t-distribution", lwd = 2)
x <- seq(2.5175, 5, length = 100)
z <- (dt(x, df = 10))
polygon(c(2.5175, x, 8), c(0, z, 0), col = rgb(1, 0, 0, 0.5))
x <- seq(-5, -2.5175, length = 100)
z <- (dt(x, df = 10))
polygon(c(-8, x, -2.5175), c(0, z, 0), col = rgb(1, 0, 0, 0.5))
```

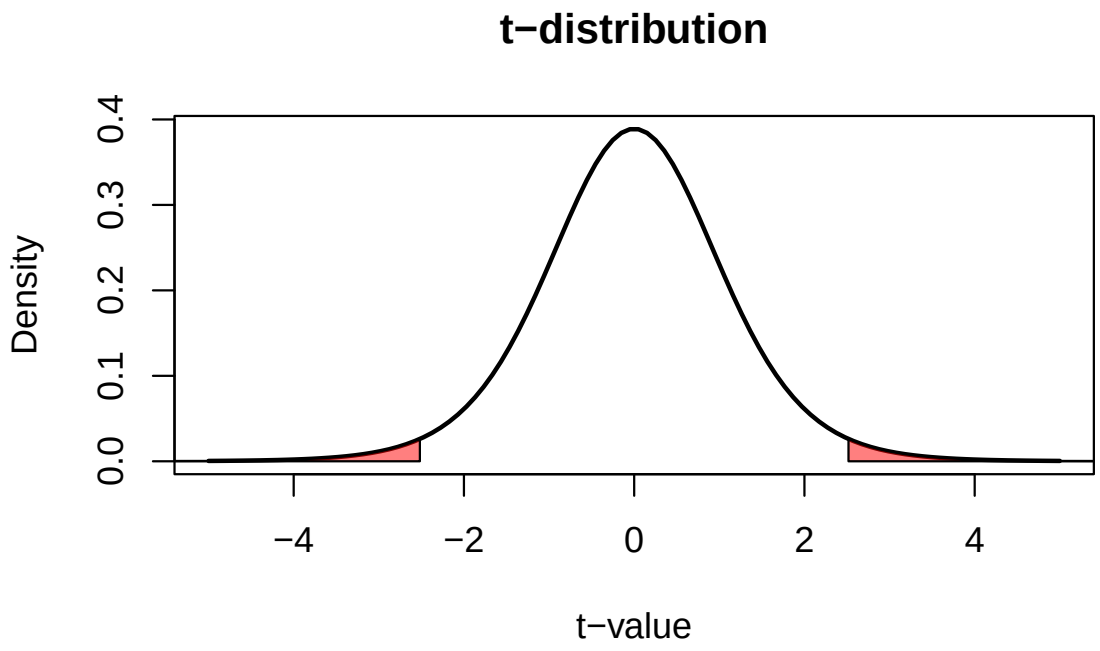


Figure 2: A t-distribution with 10 degrees of freedom.

What do we conclude?

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

```
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, $r = 0.96$, $t(10) = 10.50$, $p < .001$.

Simple Linear Regression

In our discussion of covariance and correlation, we've only talked about whether the two variables are associated with each other – asking if there's an association between temperature and sales, which is the same as asking if there's an association between sales and temperature. None of this looks at whether one variable has an effect on the other variable.

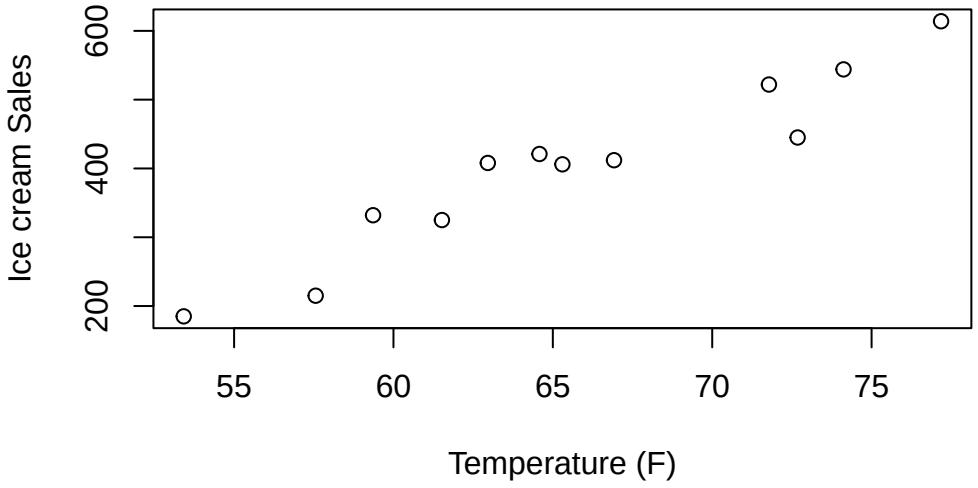
That's where linear regression comes in – now we're looking at the relation between a **dependent variable** and at least one **predictor variable**, so we've moved to the context of the predictor variable having an effect on the dependent variable.

Today's lab is going to focus on one predictor variable, which is called simple linear regression. Next week will be multiple regression, when we have at least 2 predictor variables.

For example, let us take our example of temperature and ice cream sales and see how temperature *affects* ice cream sales.

Let's plot our data again:

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



What is the equation for the regression line?

$$\hat{Y}_i = b_0 + b_1 X_{1i}$$

This gives us the predicted value of our outcome for a given value of X. b_0 represents the intercept, which is the value of the outcome when the predictor is 0. b_1 represents the slope – the expected amount of change in our outcome when our predictor increases by 1 unit.

We can calculate estimates of these values using their formulas:

$$b_1 = \frac{\text{Cov}(X, Y)}{\text{Var}(X)}$$
$$b_0 = \bar{Y} - b_1 \bar{X}$$

```
b1_estimate <- cov(icecream$Temperature,
 ↪  icecream$Sales)/var(icecream$Temperature)
b0_estimate <- mean(icecream$Sales) - b1_estimate *
 ↪  mean(icecream$Temperature)

b1_estimate
```

```
[1] 16.71548
```

```
b0_estimate
```

```
[1] -694.3695
```

Let's write out our regression line:

$$Sales_i = -694.37 + 16.72 \times Temperature_i$$

Now we can also add this regression line to our plot:

```
plot(icecream$Temperature, icecream$Sales, xlab = "Temperature (F)",
     ylab = "Ice cream Sales")
abline(a = b0_estimate, b = b1_estimate, col = "red")
# abline() is a function to add a straight line to a plot
```

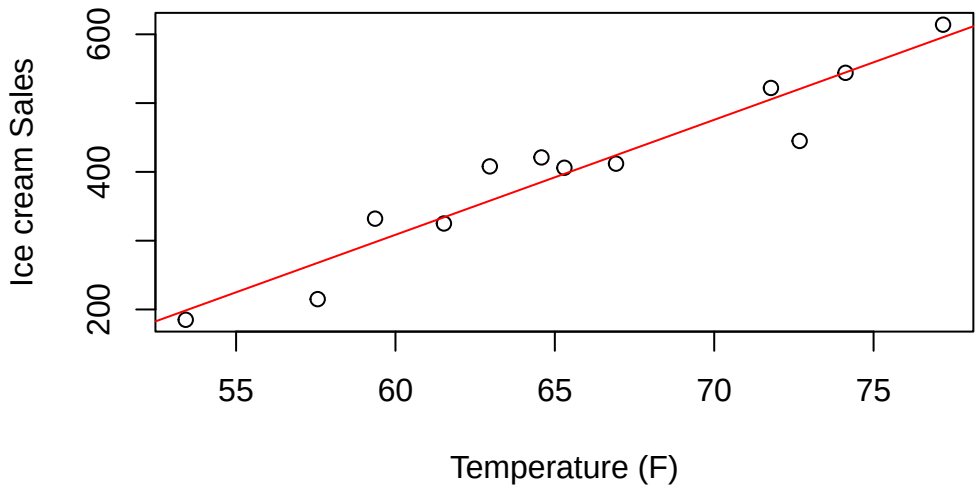


Figure 3: Simple linear regression of Ice cream sales predicted by temperature.

Of course, rather than calculating the values by hand, we can simply do it in R.

We use the `lm()` function (lm stands for "linear model") And the function arguments are as follows: `lm(dependent_variable ~ independent_variable, data = data_name)`. This might look familiar - it's pretty similar to how we conducted our t-tests in R!

```
simple_regression <- lm(Sales ~ Temperature, data = icecream)
```

Let's take a look at the output using the `summary()` function:

```
summary(simple_regression)
```

```
Call:
lm(formula = Sales ~ Temperature, data = icecream)

Residuals:
    Min      1Q  Median      3Q     Max
-75.512 -12.566   4.133  22.236  49.963

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -694.369    105.048   -6.61 6.00e-05 ***
Temperature   16.715      1.592   10.50 1.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 38.13 on 10 degrees of freedom
Multiple R-squared:  0.9168,    Adjusted R-squared:  0.9085
F-statistic: 110.2 on 1 and 10 DF,  p-value: 1.016e-06
```

You'll notice that we can get estimates for b_0 (`(Intercept)`) and for b_1 (`Temperature`). Do those match what we calculated before? How do we interpret these values?

Our intercept is the expected value of the dependent variable when the independent variable is 0. This means that when the temperature is 0 degrees Fahrenheit we expect to make -694 dollars in ice cream sales.

Note that this is a sort of ridiculous value - it never gets to 0 degrees Fahrenheit in Davis, so do we care about expected ice cream sales at that temperature? This is why centering your predictor is sometimes handy to improve interpretation of the intercept!

Our slope is the expected change in the dependent variable for a 1-unit change in the independent variable So as the temperature increases by 1 degree Fahrenheit, we expect ice cream sales to increase by $16.72.

Hypothesis Testing for the Slope and R-Squared

You'll notice that in addition to the estimates of the intercept and slope, we also have some other information In particular, we have a standard error, test statistic, and p-value.

These are for testing the null hypothesis that the parameter is equal to 0 versus the alternative that it's not equal to 0.

Although we get this test for the intercept as well that's typically not very interesting to us. Instead, we're more interested in the test for the slope Because that tells us whether the predictor is useful in predicting the outcome; otherwise, the slope is no different from 0 and there is no linear relation, no useful relation between X and Y that we can use to predict Y.

The hypotheses for this test are:

- H_0: $b_1 = 0$
- H_1: $b_1 \neq 0$

The t-statistic in this test is distributed as a t-distribution with N - 2 df. What is the df for our example?

Based on our `summary()` output, do we reject or fail to reject the null hypothesis? What does that tell us?

We could also create a confidence interval around our estimate of the slope using the information provided. Recall that the formula for a CI is:

$$\text{CI}_{95\%} = \hat{b_1} \pm t_{crit} \times \text{SE}$$

In a simple linear regression, we can get the standard error for the slope as:

$$\text{SE}_{b_1} = \frac{s_\varepsilon}{\sqrt{SS_X}}$$

where,

$$s_\varepsilon = \sqrt{\frac{1}{N-2}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2},$$
$$SS_X = \sum_{i=1}^{N}(X_i - \bar{X})^2$$

As R code:

```
sd_error <- sqrt( (1/(sample_size - 2)) * sum((icecream$Sales -
 ↪  predict(simple_regression))^2) )
ss_x <- sum((icecream$Temperature - mean(icecream$Temperature))^2)
se_b1 <- sd_error/sqrt(ss_x)
```

Of course, we don't need to do all these calculations by hand. The standard error for the slope is given in the output when we call the `summary()` function on our model. Take a look at it again and try to find it!

To get the 95%CI we have to estimate $\pm\ t_{crit} \times$ SE where t_{crit} is the critical value that cuts off the upper 2.5% of a t-distribution with N - 2 df:

```
t_crit <- qt(.025, nrow(icecream) - 2, lower.tail = FALSE)

lower_limit <- b1_estimate - t_crit * 1.592
upper_limit <- b1_estimate + t_crit * 1.592

lower_limit
```

```
[1] 13.16828
```

```
upper_limit
```

```
[1] 20.26268
```

We can also obtain the 95% CI using the `confint()` function:

```
confint(object = simple_regression, level = 0.95)
```

```
                2.5 %     97.5 %
(Intercept) -928.4318 -460.30714
Temperature   13.1679   20.26305
```

R-Squared

One other piece of useful information that you get from the `summary()` output is R-squared, also called the coefficient of determination. R-squared is the proportion of total variability

in our outcome that is explained by our predictor(s) This is calculated as sum of squares regression divided by total sum of squares: SS_{reg}/SS_{total} **or** $1 - (SS_{resid}/SS_{total})$

You could manually calculate SS_{resid} and SS_{total}, but I'm not going to bother with that You can see the R^2 value in the `summary()` output, and that value is 0.9168. This means that almost 92% of the variation in ice cream sales is explained by temperature.

You'll notice that there is also an adjusted R^2 This is more important in the context of multiple regression because as you add more predictors to the model, your R^2 will typically increase even if the added predictors are just explaining noise; therefore, adjusted R^2 corrects for the number of predictors in the model.

Why do we care about R^2? Because it may be the case that we have a significant predictor, but it doesn't actually explain a whole lot of the total variance So that's why in our write-up of the results, we'd want to report both the regression coefficients and R^2.

Here's what that write-up could look like:

We ran a simple linear regression to determine whether temperature predicts ice cream sales. The effect of Temperature is statistically significant and positive ($b = 16.72$, 95% CI [13.17, 20.26], $t(10) = 10.50$, $p < .001$). Temperature explained 91.68% of the total variability in ice cream sales.

Your Turn

Run a simple linear regression predicting the number of pool accidents from the temperature. Interpret the intercept and slope. Is the slope significant?