C1M3_peer_reviewed

June 19, 2023

1 Module 3: Peer Reviewed Assignment

1.0.1 Outline:

The objectives for this assignment:

1. Learn how to read and interpret p-values for coefficients in R.
2. Apply Partial F-tests to compare different models.
3. Compute confidence intervals for model coefficients.
4. Understand model significance using the Overall F-test.
5. Observe the variability of coefficients using the simulated data.

General tips:

1. Read the questions carefully to understand what is being asked.
2. This work will be reviewed by another human, so make sure that you are clear and concise in what your explanations and answers.

```
[1]: # Load Required Packages
     library(ggplot2)
```

1.1 Problem 1: Individual t-tests

The dataset below measures the chewiness (mJ) of different berries along with their sugar equivalent and salt (NaCl) concentration. Let's use these data to create a model to finally understand chewiness.

Here are the variables: 1. `nacl`: salt concentration (NaCl) 2. `sugar`: sugar equivalent 3. `chewiness`: chewiness (mJ)

Dataset Source: I. Zouid, R. Siret, F. Jourjion, E. Mehinagic, L. Rolle (2013). "Impact of Grapes Heterogeneity According to Sugar Level on Both Physical and Mechanical Berries Properties and their Anthocyanins Extractability at Harvest," Journal of Texture Studies, Vol. 44, pp. 95-103.

1. (a) Simple linear regression (SLR) parameters In the below code, we load in the data and fit a SLR model to it, using `chewiness` as the response and `sugar` as the predictor. The summary of the model is printed. Let $\alpha = 0.05$.

Look at the results and answer the following questions: * What is the hypothesis test related to the p-value `2.95e-09`? Clearly state the null and alternative hypotheses and the decision made based on the p-value. * Does this mean the coefficient is statistically significant? * What does it mean for a coefficient to be statistically significant?

```
[2]: # Load the data
     chew.data = read.csv("berry_sugar_chewy.csv")

     chew.lmod = lm(chewiness~sugar, data=chew.data)
     summary(chew.lmod)
```

```
Call:
lm(formula = chewiness ~ sugar, data = chew.data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4557 -0.5604  0.1045  0.5249  1.9559

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.662878   0.756610  10.128  < 2e-16 ***
sugar       -0.022797   0.003453  -6.603 2.95e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9178 on 88 degrees of freedom
Multiple R-squared:  0.3313,Adjusted R-squared:  0.3237
F-statistic: 43.59 on 1 and 88 DF,  p-value: 2.951e-09
```

Null hypothesis:$H0:Y= 0+ \ ;.$

alternative hypothesis: H1:Some predictor is necessary.

at p-value $<= (=0.05)$ there is evidence against null hypothesis

Linear regression indicates parameter associated with sugar is statistically significant.

1. (b) MLR parameters   Now let's see if the second predictor/feature `nacl` is worth adding to the model. In the code below, we create a second linear model fitting `chewiness` as the response with `sugar` and `nacl` as predictors.

Look at the results and answer the following questions: * Which, if any, of the slope parameters are statistically significant? * Did the statistical significance of the parameter for `sugar` stay the same, when compared to 1 (a)? If the statistical signficance changed, explain why it changed. If it didn't change, explain why it didn't change.

```
[3]: chew.lmod.2 = lm(chewiness ~ ., data=chew.data)
     summary(chew.lmod.2)
```

2

```
Call:
lm(formula = chewiness ~ ., data = chew.data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.3820 -0.6333  0.1234  0.5231  1.9731

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.1107    13.6459  -0.521    0.604
nacl          0.6555     0.6045   1.084    0.281
sugar        -0.4223     0.3685  -1.146    0.255

Residual standard error: 0.9169 on 87 degrees of freedom
Multiple R-squared:  0.3402,Adjusted R-squared:  0.325
F-statistic: 22.43 on 2 and 87 DF,  p-value: 1.395e-08
```

the p-values for both the sugar and nacl coefficients are greater than 0.05, which means neither of them are statistically significant at the 5% significance level

the statistical significance of the parameter for sugar did not stay the same compare to 1-a

When second parameter added it did change.

Sugar and NaCl are highly correlated

1. (c) Model Selection   Determine which of the two models we should use. Explain how you arrived at your conclusion and write out the actual equation for your selected model.

```
[4]: # Your Code Here
     # Fit the first model
     chew.lmod.1 = lm(chewiness ~ sugar, data=chew.data)
     chew.lmod.2 = lm(chewiness ~ ., data=chew.data)
     anova(chew.lmod, chew.lmod.2)
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
A anova: 2 × 6	\<dbl>	\<dbl>	\<dbl>	\<dbl>	\<dbl>	\<dbl>
1	88	74.12640	NA	NA	NA	NA
2	87	73.13801	1	0.9883882	1.175719	0.2812249

based on f-test Linear Regression model is the choice

1. (d) Parameter Confidence Intervals   Compute 95% confidence intervals for each parameter in your selected model. Then, in words, state what these confidence intervals mean.

```
[5]:  # Your Code Here
      confint(chew.lmod)
```

		2.5 %	97.5 %
A matrix: 2×2 of type dbl	(Intercept)	6.15927388	9.16648152
	sugar	-0.02965862	-0.01593536

The 95% confidence intervals for the intercept is [6.159, 9.166] and for sugar is [-0.030, -0.016]. This means that if we repeated this experiment multiple times, constructing confidence intervals in the same way, we would expect the true value of the population slope coefficients to fall within these intervals 95% of the time.

2   Problem 2: Variability of Slope in SLR

In this exercise we'll look at the variability of slopes of simple linear regression models fitted to realizations of simulated data.

Write a function, called `sim_data()`, that returns a simulated sample of size $n = 20$ from the model $Y = 1 + 2.5X + \epsilon$ where $\epsilon \overset{iid}{\sim} N(0,1)$. We will then use this generative funciton to understand how fitted slopes can vary, even for the same underlying population.

```
[6]:  sim_data <- function(n=20, var=1, beta.0=1, beta.1=2.5){
          # BEGIN SOLUTION HERE
          x = seq(-1, 1, length.out = n); beta0 = 1; beta1 = 2.5; e = rnorm(n, 0,␣
      →sqrt(var))
          y = beta0 + beta1*x + e
          # END SOLUTION HERE
          data = data.frame(x=x, y=y)
          return(data)
      }
```

2. (a) Fit a slope   Execute the following code to generate 20 data points, fit a simple linear regression model and plot the results.

Just based on this plot, how well does our linear model fit the data?

```
[7]:  data = sim_data()
      lmod = lm(y~x, data=data)
      ggplot(aes(x=x, y=y), data=data) +
          geom_point() +
          geom_smooth(method="lm", formula=y~x, se=FALSE, color="#32CD32")
```
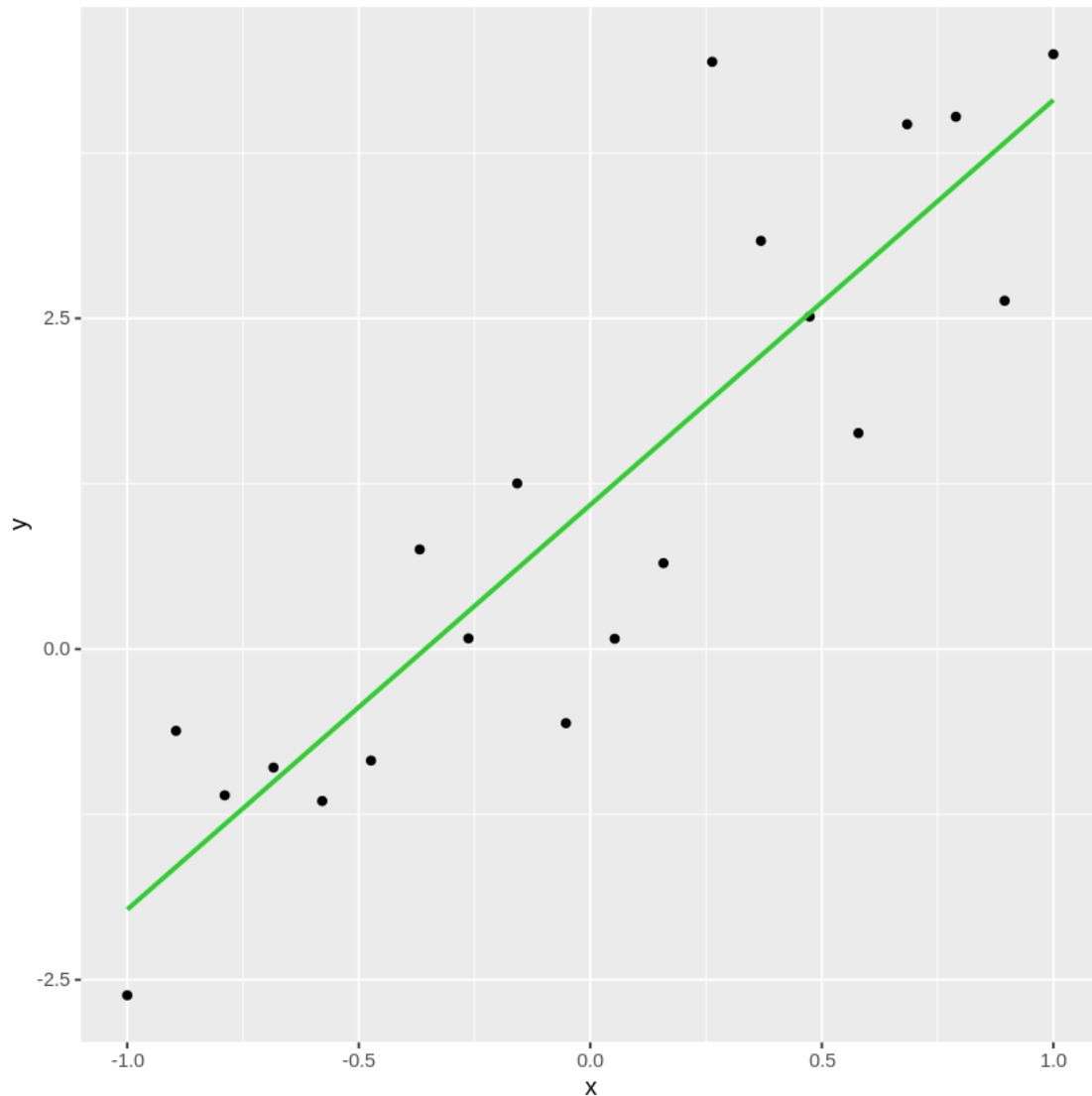
The model does fit well

2. (b) Do the slopes change?   Now we want to see how the slope of our line varies with different random samples of data. Call our data genaration funciton 50 times to gather 50 independent samples. Then we can fit a SLR model to each of those samples and plot the resulting slope. The function below performs this for us.

Experiment with different variances and report on what effect that has to the spread of the slopes.

```
[8]: gen_slopes <- function(num.slopes=50, var=1, num.samples=20){
        g = ggplot()
        # Repeat the sample for the number of slopes
        for(ii in 1:num.slopes){
```

```
        # Generate a random sampling of data
        data = sim_data(n=num.samples, var=var)
        # Add the slope of the best fit linear model to the plot
        g = g + stat_smooth(aes(x=x, y=y), data=data, method="lm", geom="line",
                            se=FALSE, alpha=0.4, color="#CFB87C", size=1)
    }
    return(g)
}
```

[11]: `gen_slopes()`

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
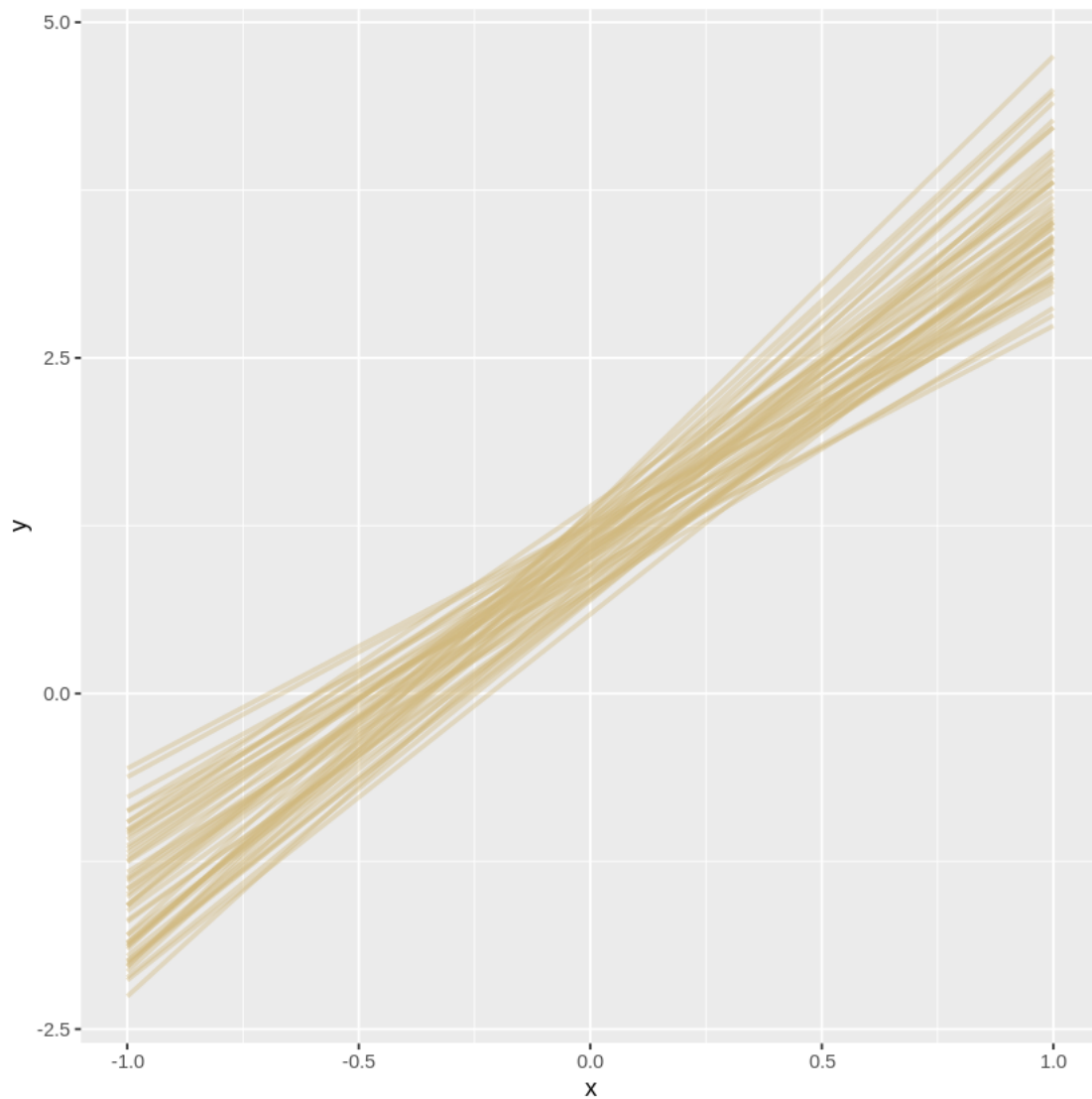
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```

```
`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'

`geom_smooth()` using formula 'y ~ x'
```

variability in slopes changes and increases.

2. (c) Distributions of Slopes   As we see above, the slopes are somewhat random. That means that they follow some sort of distribution, which we can try to discern. The code below computes `num_samples` independent realizations of the model data, computes the SLR model, and generates a histogram of the resulting slopes.

Again, experiment with different variances for the simulated data and record what you notice. What do you notice about the shapes of the resulting histograms?

```
[12]: hist_slopes <- function(num.slopes=500, var=1, num.samples=20){
          slopes = rep(0, num.slopes)
          # For num.slopes, compute a SLR model slope
          for(i in 1:num.slopes){
              # Simulate the desired data
              data = sim_data(var=var, n=num.samples)
              # Fit an SLR model to the data
              lmod = lm(y~x, data=data)
              # Add the slopes to the vector of slopes
              slopes[i] = lmod$coef[2]
          }
          # Plot a histogram of the resulting slopes
          g = ggplot() + aes(slopes) + geom_histogram(color="black", fill="#CFB87C")
          return(g)
      }
```

```
[86]: # hist_slopes()
```

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
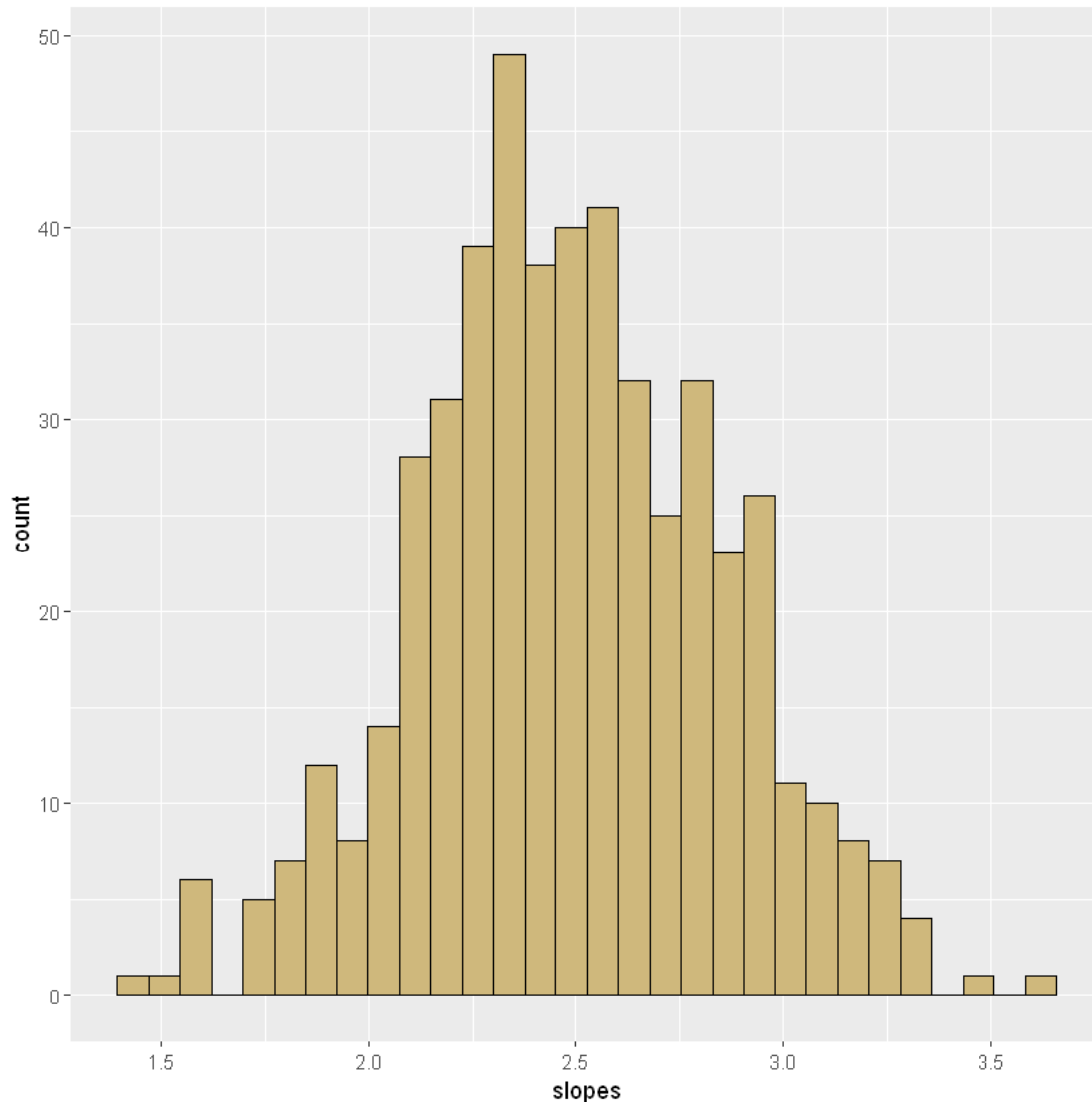
We got a bell chape curve. The width increases as 2 increases.

2. (d) Confidence Intervals of Slopes   What does that all mean? It means that when we fit a linear regression model, our parameter *estimates* will not be equal to the true parameters. Instead, the estimates will vary from sample to sample, and form a distribution. This is true for any linear regression model with any data - not just simulated data - as long as we assume that there is a large population that we can resample the response from (at fixed predictor values). Also note that we only demonstrated this fact with the slope estimate, but the same principle is true for the intercept, or if we had several slope parameters.

This simulation shows that there is a chance for a linear regression model to have a slope that is very different from the true slope. But with a large sample size, n, or small error variance, σ^2, the distribution will become narrower. Confidence intervals can help us understand this variability.

The procedure that generates confidence intervals for our model parameters has a high probability of covering the true parameter. And, the higher n is, for a fixed σ^2, or the smaller σ^2 is, for a fixed n, the narrower the confidence interval will be!

Draw a single sample of size $n = 20$ from `sim_data()` with variance $\sigma^2 = 1$. Use your sample to compute a 95% confidence interval for the slope. Does the known slope for the model (which we can recall is 2.5) fall inside your confidence interval? How does the value of σ^2 affect the CI width?

[14]:
```
# Your code here
df = sim_data(var = 1)
lm_sim = lm(y ~ x, data = df)
confint(lm_sim)
```

		2.5 %	97.5 %
A matrix: 2 × 2 of type dbl	(Intercept)	0.620125	1.408259
	x	1.425761	2.724219

The slope is in the 95% confidence interval. CI increase with 2

[ ]: