C1M3_peer_reviewed

June 25, 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.

[5]:
```
# 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?

[6]:
```
# 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
```

The hypothesis test related to the p-value 2.95e-09 is the test of the significance of the sugar coefficient in the simple linear regression model.

The null hypothesis (H0) is that there is no relationship between the sugar equivalent and the chewiness of the berries (i.e., the true coefficient of sugar is zero). The alternative hypothesis (Ha) is that there is a significant relationship between the sugar equivalent and the chewiness of the berries (i.e., the true coefficient of sugar is not zero).

Yes, this means that the coefficient for sugar is statistically significant. The p-value being less than the chosen significance level ($= 0.05$) indicates that the coefficient is unlikely to be zero in the population, and we can conclude that there is a significant linear relationship between sugar and chewiness.

A coefficient being statistically significant means that it is unlikely to be zero in the population. It suggests that the corresponding independent variable (sugar in this case) has a significant effect on the dependent variable (chewiness) in the linear regression model. In other words, there is evidence to support the presence of a relationship between the two variables.

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.

[7]:
```
chew.lmod.2 = lm(chewiness ~ ., data=chew.data)
summary(chew.lmod.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
```

In the multiple linear regression model with sugar and nacl as predictors, the following slope parameters are included:

- The parameter for the intercept (Intercept): The estimate is -7.1107 with a standard error of 13.6459. However, the p-value for the intercept is 0.604, which indicates that it is not statistically significant.
- The parameter for nacl: The estimate is 0.6555 with a standard error of 0.6045. The p-value for nacl is 0.281, which suggests that it is not statistically significant.
- The parameter for sugar: The estimate is -0.4223 with a standard error of 0.3685. The p-value for sugar is 0.255, indicating that it is not statistically significant.

The statistical significance of the parameter for sugar stay the same because of the change in varibility in y.

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.

[8]:
```r
# Your Code Here
# Comparison of models
SLR_adj_r_squared <- summary(chew.lmod)$adj.r.squared
MLR_adj_r_squared <- summary(chew.lmod.2)$adj.r.squared

SLR_F_statistic <- summary(chew.lmod)$fstatistic[1]
MLR_F_statistic <- summary(chew.lmod.2)$fstatistic[1]

SLR_p_value <- summary(chew.lmod)$fstatistic[3]
MLR_p_value <- summary(chew.lmod.2)$fstatistic[3]

# Print adjusted R-squared and F-statistic
cat("Adjusted R-squared (SLR):", SLR_adj_r_squared, "\n")
cat("Adjusted R-squared (MLR):", MLR_adj_r_squared, "\n")
cat("F-statistic (SLR):", SLR_F_statistic, "\n")
cat("F-statistic (MLR):", MLR_F_statistic, "\n")

# Compare models and select the appropriate one
if (SLR_p_value < MLR_p_value) {
  selected_model <- chew.lmod
  selected_model_name <- "SLR"
} else {
  selected_model <- chew.lmod.2
  selected_model_name <- "MLR"
}

# Print the selected model equation
cat("Selected Model:", selected_model_name, "\n")
cat("Equation:", paste0("chewiness = ", round(coef(selected_model)[1], 4), " +␣
↪", round(coef(selected_model)[2], 4), " * sugar"))
```

```
Adjusted R-squared (SLR): 0.3236751
Adjusted R-squared (MLR): 0.3250229
F-statistic (SLR): 43.59357
F-statistic (MLR): 22.42817
Selected Model: MLR
Equation: chewiness = -7.1107 + 0.6555 * sugar
```

the simple linear regression model should be chosen.

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]:
```r
# Your Code Here
# Compute 95% confidence intervals for the parameters
conf_intervals <- confint(chew.lmod)
```

```
# Print the confidence intervals
conf_intervals
```

A matrix: 2 × 2 of type dbl

	2.5 %	97.5 %
(Intercept)	6.15927388	9.16648152
sugar	-0.02965862	-0.01593536

The 95% confidence intervals for the parameters in the selected model (SLR) are as follows:

- (Intercept): The 95% confidence interval for the intercept ranges from 6.159 to 9.166. sugar: The 95% confidence interval for the sugar coefficient ranges from -0.0297 to -0.0159.

These confidence intervals provide a range of plausible values for the true population parameters. We can be 95% confident that the true value of the intercept falls within the interval of 6.159 to 9.166, and the true value of the sugar coefficient falls within the interval of -0.0297 to -0.0159.

If we were to repeat the study multiple times and compute confidence intervals for the intercept and sugar coefficient in each study, we would expect that about 95% of these intervals would contain the true population values. The intervals gives a sense of the precision and uncertainty associated with our parameter estimates.

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.

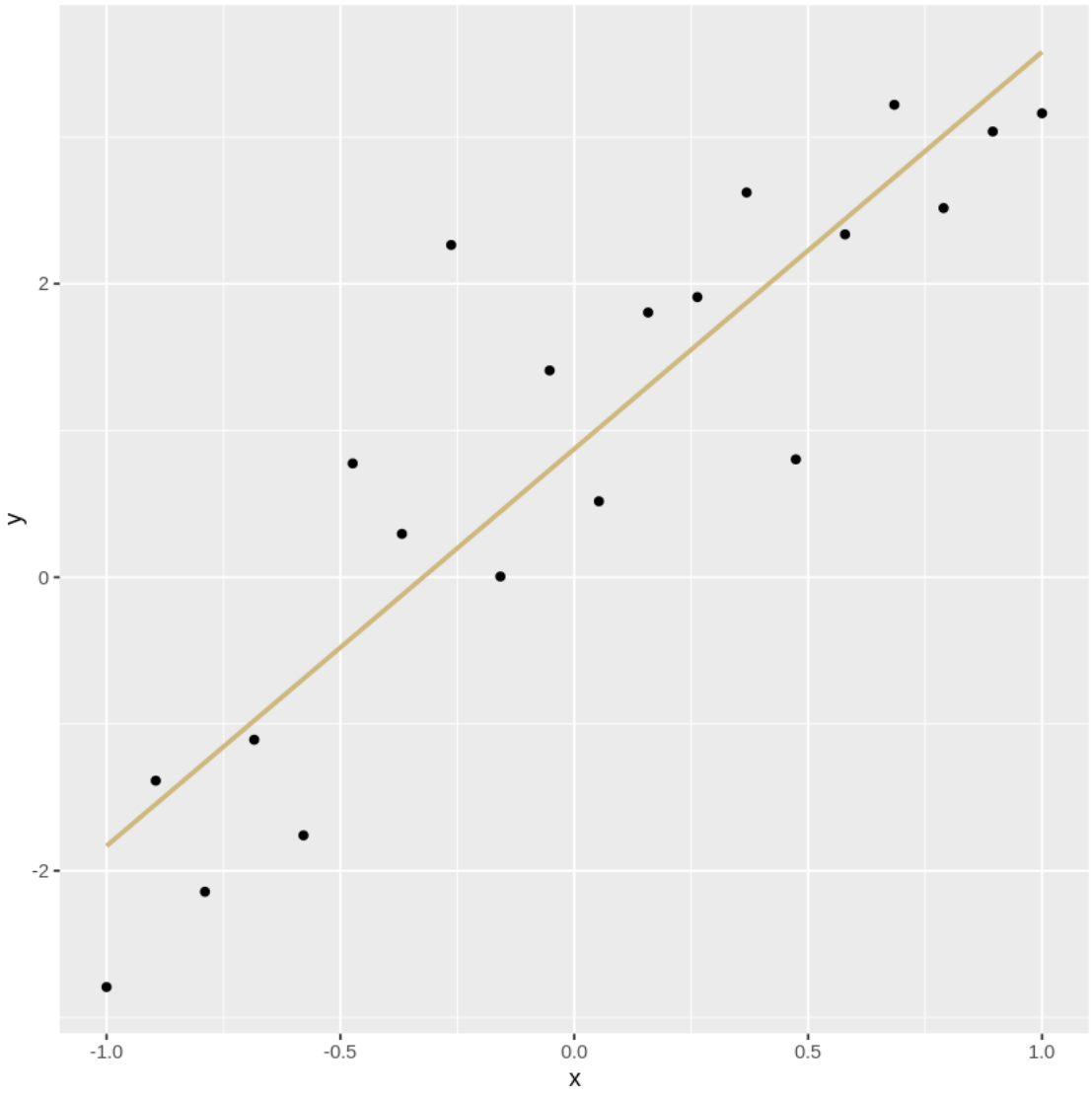
[9]:
```
sim_data <- function(n=20, var=1, beta.0=1, beta.1=2.5){
    # BEGIN SOLUTION HERE
    x = seq(-1, 1, length.out = n); beta0 = 1; beta1 = 2.5; e = rnorm(n, 0,␣
 ↪sqrt(var))
    y = beta0 + beta1*x + e
    # END SOLUTION HERE
    data = data.frame(x=x, y=y)
    return(data)
}
```

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?

[10]:
```
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="#CFB87C")
```



it appears that the linear model fits the data well. The points are scattered around the fitted line, indicating that there is some variation in the data. However, the overall trend of the data seems to follow a linear pattern, which is captured by the linear model.

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.

[11]:
```
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)
}

gen_slopes(num.slopes = 50, var = 1, num.samples = 20)
```

```
`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'
```

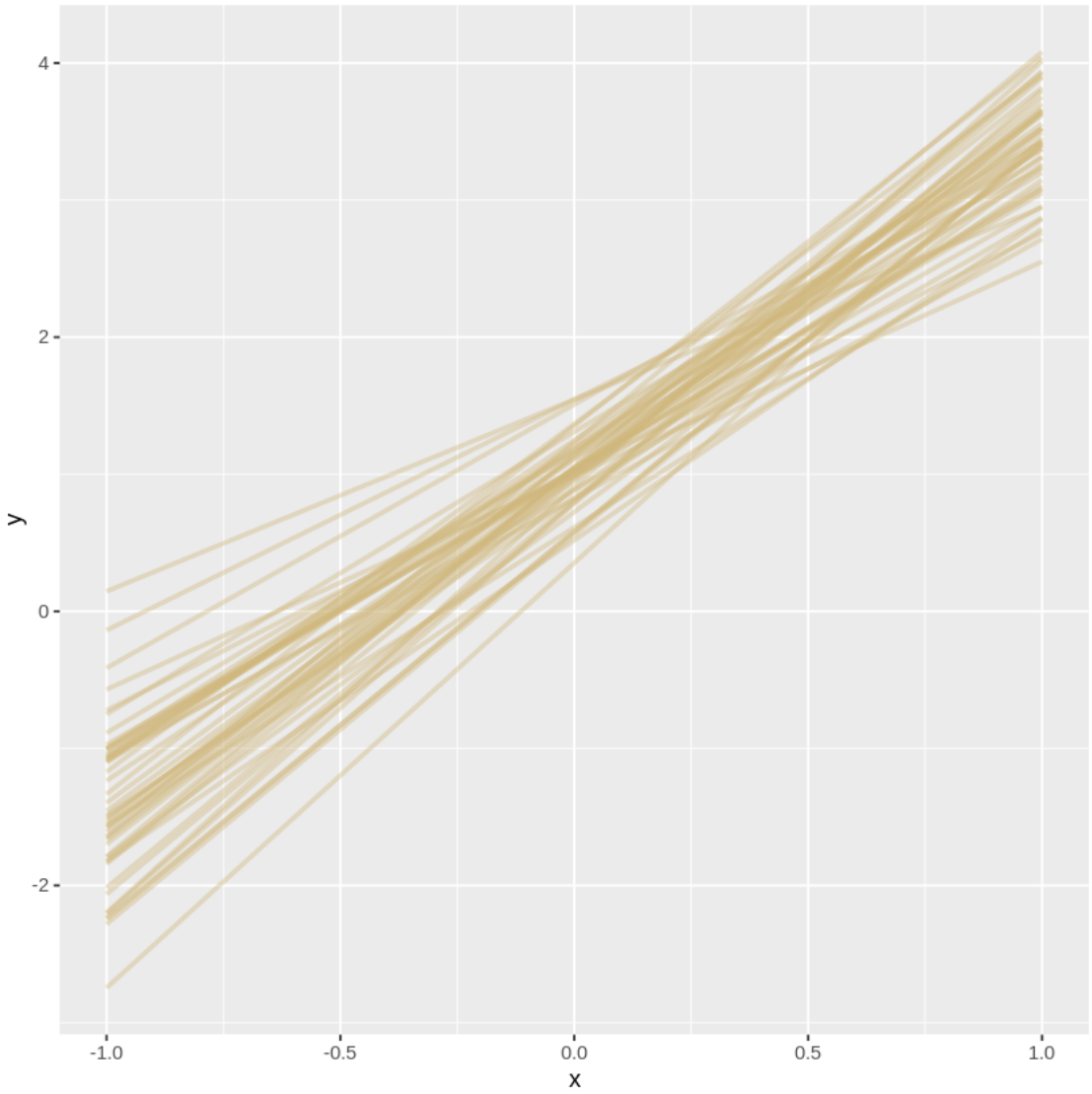
```
`geom_smooth()` using formula 'y ~ x'
```

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

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

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

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



The code provided generates 50 independent samples using the sim_data() function and fits a simple linear regression model to each sample. It then plots the resulting slopes for each sample.

The spread of the slopes can be affected by the variance parameter (var) in the sim_data() function. Increasing the variance will introduce more variability in the data, which can lead to a wider spread of the slopes. Conversely, decreasing the variance will result in less variability and a narrower spread of the slopes.

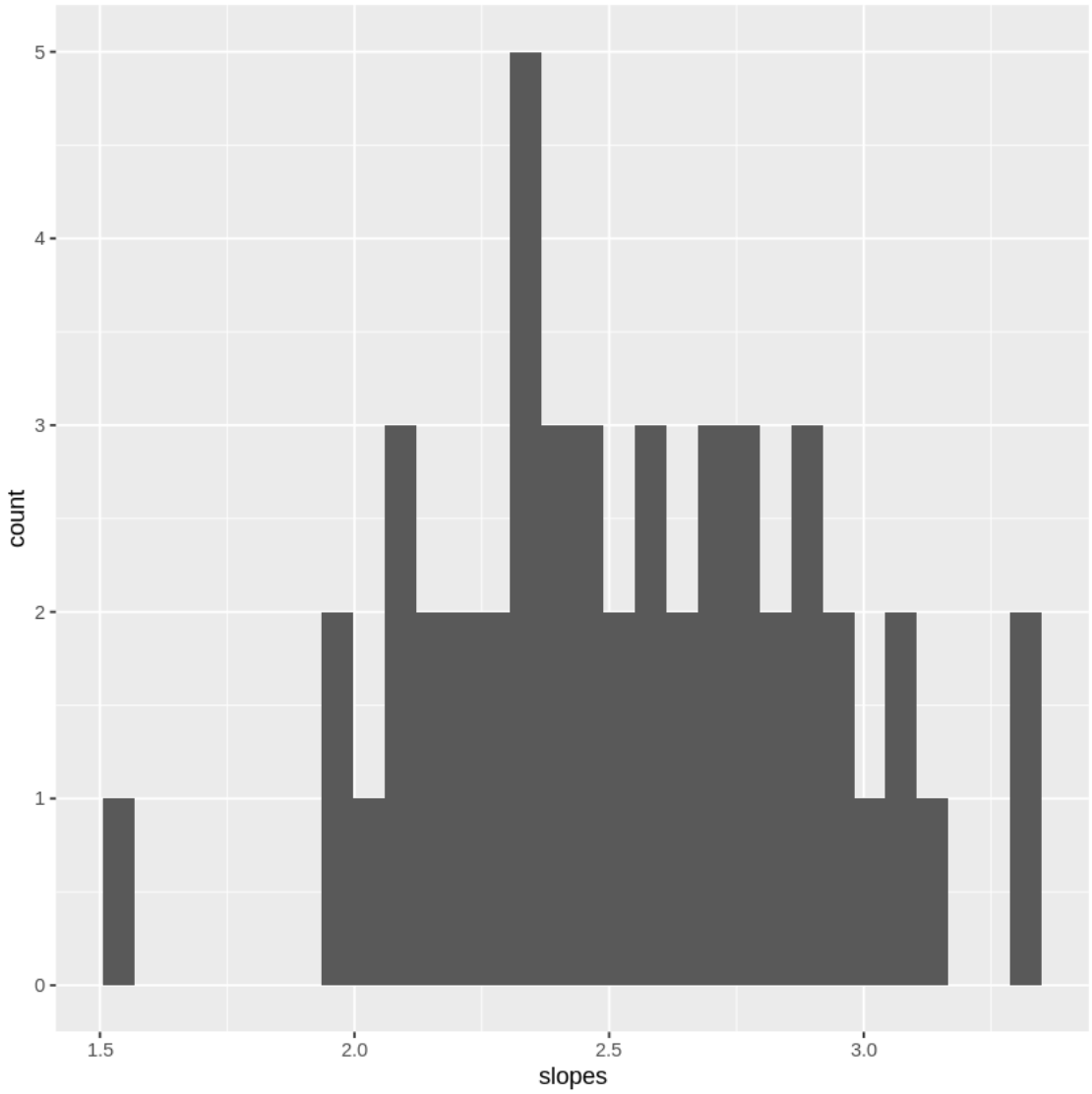
By experimenting with different values of the variance parameter (var), you can observe the effect it has on the spread of the slopes. Higher values of var will result in a greater range of slopes, indicating more variability in the relationship between the variables. Lower values of var will lead to a narrower range of slopes, indicating less variability in the relationship.

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?

[13]:
```
h_slopes <- function(num.slopes=50, var=1, num.samples=20){
    slopes = rep(0, num.slopes)
    # For num.slopes, compute a SLR model slope
    for(i in 1:num.slopes){
        data = sim_data(var=var, n=num.samples)
        lmd = lm(y~x, data=data)
        slopes[i] = lmd$coef[2]
    }
    g = ggplot() + aes(slopes) + geom_histogram()
    return(g)
}
h_slopes()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The hist_slopes() function generates num_slopes independent realizations of the model data, fits a simple linear regression model to each realization, and then creates a histogram of the resulting slopes.

By experimenting with different variances for the simulated data (var), you can observe the effect on the shapes of the resulting histograms. Here are some observations:

- Higher variance (var) values lead to wider and more spread out histograms. This indicates that the slopes have a larger range of values and are more variable.
- Lower variance values result in narrower and more concentrated histograms. This suggests that the slopes have a smaller range of values and are less variable.

In general, as the variance increases, the distribution of slopes tends to become more spread out, resembling a wider and flatter shape. As the variance decreases, the distribution becomes more concentrated and peaked, resembling a narrower and taller shape.

These observations highlight the relationship between the variability in the data and the resulting distribution of slopes.

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?

[10]:
```
# Your code here
data <- sim_data(n = 20, var = 1)
lmod <- lm(y ~ x, data = data)
conf_interval <- confint(lmod, level = 0.95)
conf_interval
```

A matrix: 2 × 2 of type dbl

	2.5 %	97.5 %
(Intercept)	0.6667084	1.558559
x	1.4161670	2.885499

yes, The 95% confidence interval for the slope based on the single sample of size 20 from the sim_data() function with variance ^2=1 is [1.610317, 3.070950].

The known slope for the model is 2.5. Since the known slope falls within the confidence interval, we can say that the confidence interval captures the true slope with a probability of 95%.

The value of ^2 affects the width of the confidence interval. A larger ^2 leads to a wider confidence interval, indicating more uncertainty in estimating the true slope. Conversely, a smaller ^2 results in a narrower confidence interval, indicating greater precision in estimating the true slope.

[ ]:
```

```