Week 4 - Homework

STAT 420, Summer 2023, D. Unger

Student Name: Scott Girten
NetId: sgirten2

Directions

Students are encouraged to work together on homework. However, sharing, copying or providing any part of a homework solution or code is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible.

- Be sure to remove this section if you use this `.Rmd` file as a template.
- You may leave the questions in your final document.

Exercise 1 (Using `lm`)

For this exercise we will use the data stored in `nutrition-2018.csv`. It contains the nutritional values per serving size for a large variety of foods as calculated by the USDA in 2018. It is a cleaned version totaling 5956 observations and is current as of April 2018.

The variables in the dataset are:

- `ID`
- `Desc` - short description of food
- `Water` - in grams
- `Calories`
- `Protein` - in grams
- `Fat` - in grams
- `Carbs` - carbohydrates, in grams
- `Fiber` - in grams
- `Sugar` - in grams
- `Calcium` - in milligrams
- `Potassium` - in milligrams
- `Sodium` - in milligrams
- `VitaminC` - vitamin C, in milligrams
- `Chol` - cholesterol, in milligrams
- `Portion` - description of standard serving size used in analysis

(a) Fit the following multiple linear regression model in `R`. Use `Calories` as the response and `Fat`, `Sugar`, and `Sodium` as predictors.

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon_i.$$

Here,

- Y_i is `Calories`.
- x_{i1} is `Fat`.
- x_{i2} is `Sugar`.
- x_{i3} is `Sodium`.

Use an F-test to test the significance of the regression. Report the following:

- The null and alternative hypotheses
- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.01$
- A conclusion in the context of the problem

When reporting these, you should explicitly state them in your document, not assume that a reader will find and interpret them from a large block of `R` output.

Solution:

```r
library(tidyverse)

# Read in data
data = read_csv('nutrition-2018.csv')

# Null and Full model
nutrition_null = lm(Calories ~ 1, data = data)
nutrition_model = lm(Calories ~ Fat + Sugar + Sodium, data = data)

# ANOVA table
anov = anova(nutrition_null, nutrition_model)

# Variables for display in report
f_stat = anov[2,5]
p_val = anov[2,6]
```

Hypothesis Test

H_0 : All $\hat{\beta} = 0$

H_1 : At least one $\hat{\beta} \neq 0$

F-Statistic: 6590.9402239

p-value: 0

$\alpha = 0.01$

Decision: Reject the $Null$ hypothesis

The null hypothesis assumed that none of the three predictor variables of Fat, Sugar or Sodium had an explanatory relationship for the response variable Calories and as such would not be statistically different from the $Null$ model for predicting values of Calories. The ANOVA shows that at least one of the 3 predictors have a linear relationship with the response variable Calories and because the 3 predictors of Fat, Sugar and

Sodium have some linear relationship with Calories we reject the $Null$ hypothesis in favor of the alternative hypothesis.

(b) Output only the estimated regression coefficients. Interpret all $\hat{\beta}_j$ coefficients in the context of the problem.

Solution:

```
# Estimated regression coefficients
coef(nutrition_model)[2:4]
```

```
##         Fat       Sugar      Sodium
## 8.483289078 3.900517188 0.006165246
```

```
fat = coef(nutrition_model)[2]
sugar = coef(nutrition_model)[3]
sodium = coef(nutrition_model)[4]
```

$\hat{\beta}_1$ (Fat): Given a fixed level of Sugar and Sodium, every additional gram of Fat will increase Calories by approximately 8.483.

$\hat{\beta}_2$ (Sugar): Given a fixed level of Fat and Sodium, every additional gram of Sugar will increase Calories by approximately 3.901.

$\hat{\beta}_3$ (Sodium): Given a fixed level of Fat and Sugar, every additional milligram of Sodium will increase Calories by approximately 0.006.

(c) Use your model to predict the number of `Calories` in a Filet-O-Fish. According to McDonald's publicized nutrition facts, the Filet-O-Fish contains 18g of fat, 5g of sugar, and 580mg of sodium.

Solution:

```
# Predict Filet-o-Fish
newdata = data.frame(Fat = 18, Sugar = 5, Sodium = 580)

# Prediction
my_prediction = predict(nutrition_model, newdata = newdata)
```

My nutrition model predicts the number of calories in a Filet-O-Fish to be 276.23

(d) Calculate the standard deviation, s_y, for the observed values in the Calories variable. Report the value of s_e from your multiple regression model. Interpret both estimates in the context of this problem.

Solution:

```
s_y = sd(data$Calories)
s_e = summary(nutrition_model)$sigma
```

s_y (168.050) the standard deviation of the observed values for the variable Calorie.
s_e (80.854) is the standard error of the residuals from the fitted model.

(e) Report the value of R^2 for the model. Interpret its meaning in the context of the problem.

Solution:

```r
r2 = summary(nutrition_model)$r.squared
```

76.86% of the variability of the response variable Calories can be explained by the predictor variables Fat, Sugar and Sodium.

(f) Calculate a 90% confidence interval for β_2. Give an interpretation of the interval in the context of the problem.

Solution:

```r
beta_ci = confint(nutrition_model, level = 0.90)
beta2_lwr = beta_ci[3,1]
beta2_upr = beta_ci[3,2]
```

At a 90% level of confidence, the mean value of β_2 (Sugar) is between 3.783051 and 4.0179834.

(g) Calculate a 95% confidence interval for β_0. Give an interpretation of the interval in the context of the problem.

Solution:

```r
beta_ci = confint(nutrition_model, level = 0.95)
beta0_lwr = beta_ci[1,1]
beta0_upr = beta_ci[1,2]
```

At a 95% level of confidence, the mean value of of the intercept of the regression model is between 97.6944297 and 103.2176836.

(h) Use a 99% confidence interval to estimate the mean Calorie content of a food with 15g of fat, 0g of sugar, and 260mg of sodium, which is true of a medium order of McDonald's french fries. Interpret the interval in context.

Solution:

```r
# Nutrition data for french fries
newdata = data.frame(Fat = 15, Sugar = 0, Sodium = 260)

french_fry = predict(nutrition_model, newdata = newdata, interval = 'confidence', level = 0.99)
```

At a 99% level of confidence, I estimate the mean Calorie content for a food with 15g of fat, 0g of sugar, and 260mg of sodium to be between 226.1657341 and 232.4509796).

(i) Use a 99% prediction interval to predict the Calorie content of a Crunchy Taco Supreme, which has 11g of fat, 2g of sugar, and 340mg of sodium according to Taco Bell's publicized nutrition information. Interpret the interval in context.

Solution:

```r
# Nutrition data for Crunchy Taco Supreme
newdata = data.frame(Fat = 11, Sugar = 2, Sodium = 340)

taco = predict(nutrition_model, newdata = newdata, interval = 'prediction', level = 0.99)
```

At a 99% level of confidence, I estimate the Calorie content of a Crunchy Supreme Taco to be between -4.6844814 and 412.0233906.

Exercise 2 (More `lm` for Multiple Regression)

For this exercise we will use the data stored in `goalies17.csv`. It contains career data for goaltenders in the National Hockey League during the first 100 years of the league from the 1917-1918 season to the 2016-2017 season. It holds the 750 individuals who played at least one game as goalie over this timeframe. The variables in the dataset are:

- `Player` - Player's Name (those followed by * are in the Hall of Fame as of 2017)
- `First` - First year with game recorded as goalie
- `Last` - Last year with game recorded as goalie
- `Active` - Number of seasons active in the NHL
- `GP` - Games Played
- `GS` - Games Started
- `W` - Wins
- `L` - Losses (in regulation)
- `TOL` - Ties and Overtime Losses
- `GA` - Goals Against
- `SA` - Shots Against
- `SV` - Saves
- `SV_PCT` - Save Percentage
- `GAA` - Goals Against Average
- `SO` - Shutouts
- `PIM` - Penalties in Minutes
- `MIN` - Minutes

For this exercise we will consider three models, each with Wins as the response. The predictors for these models are:

- Model 1: Goals Against, Saves
- Model 2: Goals Against, Saves, Shots Against, Minutes, Shutouts
- Model 3: All Available

After reading in the data but prior to any modeling, you should clean the data set for this exercise by removing the following variables: `Player`, `GS`, `L`, `TOL`, `SV_PCT`, and `GAA`.

```r
# Clean data

# Read in data
goalies = read_csv('goalies17.csv')

# Remove columns
col_remove = c('Player', 'GS', 'L', 'TOL', 'SV_PCT', 'GAA')
goalies_cleaned = goalies %>%
  select(!all_of(col_remove))
```

(a) Use an F-test to compares Models 1 and 2. Report the following:

- The null hypothesis
- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.05$
- The model you prefer

Solution:

```r
model_1 = lm(W ~ GA + SV, data = goalies_cleaned)
model_2 = lm(W ~ GA + SV + SA + MIN + SO, data = goalies_cleaned)

anova_12 = anova(model_1, model_2)
```

$H_0 : \beta_{SV} = \beta_{SA} = \beta_{MIN} = \beta_{SO} = 0$

F-statistic: 496.3764519

p-value: $4.7666873 \times 10^{-149}$

Statistical Decision ($\alpha = 0.05$)**:** Reject the $Null$ hypothesis.

I would prefer model 2 since the $Null$ hypothesis was rejected and at least one of the additional variables in model 2 improves the performance of the model in estimating the response variable Wins.

(b) Use an F-test to compare Model 3 to your preferred model from part **(a)**. Report the following:

- The null hypothesis
- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.05$
- The model you prefer

Solution:

```r
model_3 = lm(W ~ ., data = goalies_cleaned)

anova_23 = anova(model_2, model_3)
```

$H_o : \beta_{First} = \beta_{Last} = \beta_{Active} = \beta_{GP} = \beta_{PIM} = 0$

F-statistic: 12.2831933

p-value: $3.0730074 \times 10^{-11}$

Statistical Decision ($\alpha = 0.05$)**:** Reject the $Null$ hypothesis.

I would prefer model 3 since the $Null$ hypothesis was rejected and at least one of the additional variables in model 3 improves the performance of the model in estimating the response variable Wins.

(c) Use a t-test to test $H_0 : \beta_{\texttt{SV}} = 0$ vs $H_1 : \beta_{\texttt{SV}} \neq 0$ for the model you preferred in part **(b)**. Report the following:

- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.05$

Solution:

```r
t_test = summary(model_3)
#t_test
t = coef(t_test)['SV', 't value']
p = coef(t_test)['SV', 'Pr(>|t|)']
```

t-statistic: -4.0770141

p-value: 5.3190411×10^{-5}

Statistical Decision ($\alpha = 0.05$): Reject the $Null$ hypothesis.

Exercise 3 (Regression without `lm`)

For this exercise we will once again use the `Ozone` data from the `mlbench` package. The goal of this exercise is to fit a model with `ozone` as the response and the remaining variables as predictors.

```
data(Ozone, package = "mlbench")
Ozone = Ozone[, c(4, 6, 7, 8)]
colnames(Ozone) = c("ozone", "wind", "humidity", "temp")
Ozone = Ozone[complete.cases(Ozone), ]
```

(a) Obtain the estimated regression coefficients **without** the use of `lm()` or any other built-in functions for regression. That is, you should use only matrix operations. Store the results in a vector `beta_hat_no_lm`. To ensure this is a vector, you may need to use `as.vector()`. Return this vector as well as the results of `sum(beta_hat_no_lm ^ 2)`.

Solution:

```
# Response variable
y = Ozone$ozone

# Predictor variables and intercept
X = Ozone %>%
  select(-ozone) %>%
  mutate(intercept = 1) %>%
  relocate(intercept)

# Convert to matrix
X = as.matrix(X)

beta_hat_no_lm = solve(t(X) %*% X) %*% t(X) %*% y
beta_hat_no_lm = as.vector(beta_hat_no_lm)
```

$\hat{\beta}$ coefficients: -16.3817854, -0.1859444, 0.0834001, 0.3898429

Sum of squared $\hat{\beta}$ vector: 268.556401

(b) Obtain the estimated regression coefficients **with** the use of `lm()`. Store the results in a vector `beta_hat_lm`. To ensure this is a vector, you may need to use `as.vector()`. Return this vector as well as the results of `sum(beta_hat_lm ^ 2)`.

Solution:

```
beta_hat_lm = lm(ozone ~ ., data = Ozone)
beta_hat_lm = as.vector(coef(beta_hat_lm))
```

$\hat{\beta}$ coefficients: -16.3817854, -0.1859444, 0.0834001, 0.3898429

Sum of squared $\hat{\beta}$ vector: 268.556401

(c) Use the `all.equal()` function to verify that the results are the same. You may need to remove the names of one of the vectors. The `as.vector()` function will do this as a side effect, or you can directly use `unname()`.

Solution:

```
all.equal(beta_hat_no_lm, beta_hat_lm)
```

```
## [1] TRUE
```

(d) Calculate s_e without the use of `lm()`. That is, continue with your results from **(a)** and perform additional matrix operations to obtain the result. Output this result. Also, verify that this result is the same as the result obtained from `lm()`.

Solution:

```
# Variables needed for matrix calculation of s_e
n = nrow(X)
p = length(beta_hat_no_lm)

y_hat = X %*% beta_hat_no_lm

e = y - y_hat

# Calculate s_e
se_no_lm = sqrt((t(e) %*% e) / (n - p))

# s_e using lm()
se_lm = lm(ozone ~ ., data = Ozone)
se_lm = summary(se_lm)$sigma
```

s_e without using `lm()`: 4.8061147
s_e using `lm()`: 4.8061147

(e) Calculate R^2 without the use of `lm()`. That is, continue with your results from **(a)** and **(d)**, and perform additional operations to obtain the result. Output this result. Also, verify that this result is the same as the result obtained from `lm()`.

Solution:

```
# Components of R-squared calculation
y_bar = mean(Ozone$ozone)

ss_reg = sum(e^2)
sst = sum((y - y_bar)^2)

# Calculate r-squared without lm()
r2_no_lm = 1 - (ss_reg / sst)

# R-squared using lm()
r2_lm = lm(ozone ~ ., data = Ozone)
r2_lm = summary(r2_lm)$r.squared
```

R^2 without using `lm()`: 0.6398887
R^2 using `lm()`: 0.6398887

Exercise 4 (Regression for Prediction)

For this exercise use the `Auto` dataset from the `ISLR` package. Use `?Auto` to learn about the dataset. The goal of this exercise is to find a model that is useful for **predicting** the response `mpg`. We remove the `name` variable as it is not useful for this analysis. (Also, this is an easier to load version of data from the textbook.)

```
# load required package, remove "name" variable
library(ISLR)
Auto = subset(Auto, select = -c(name))
dim(Auto)
```

```
## [1] 392   8
```

When evaluating a model for prediction, we often look at RMSE. However, if we both fit the model with all the data as well as evaluate RMSE using all the data, we're essentially cheating. We'd like to use RMSE as a measure of how well the model will predict on *unseen* data. If you haven't already noticed, the way we had been using RMSE resulted in RMSE decreasing as models became larger.

To correct for this, we will only use a portion of the data to fit the model, and then we will use leftover data to evaluate the model. We will call these datasets **train** (for fitting) and **test** (for evaluating). The definition of RMSE will stay the same

$$\text{RMSE}(\text{model, data}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where

- y_i are the actual values of the response for the given data.
- $\hat{y}_i$ are the predicted values using the fitted model and the predictors from the data.

However, we will now evaluate it on both the **train** set and the **test** set separately. So each model you fit will have a **train** RMSE and a **test** RMSE. When calculating **test** RMSE, the predicted values will be found by predicting the response using the **test** data with the model fit using the **train** data. ***Test** data should never be used to fit a model.*

- Train RMSE: Model fit with *train* data. Evaluate on **train** data.
- Test RMSE: Model fit with *train* data. Evaluate on **test** data.

Set a seed of `22`, and then split the `Auto` data into two datasets, one called `auto_trn` and one called `auto_tst`. The `auto_trn` data frame should contain 290 randomly chosen observations. The `auto_tst` data will contain the remaining observations. Hint: consider the following code:

```
set.seed(22)
auto_trn_idx = sample(1:nrow(Auto), 290)
```

Fit a total of five models using the training data.

- One must use all possible predictors.
- One must use only `displacement` as a predictor.
- The remaining three you can pick to be anything you like. One of these should be the *best* of the five for predicting the response.

For each model report the **train** and **test** RMSE. Arrange your results in a well-formatted markdown table. Argue that one of your models is the best for predicting the response.

solution:

```
library(kableExtra)

# Split data into test and train
auto_trn = Auto %>% filter(row_number() %in% auto_trn_idx)
auto_tst = Auto %>% filter(!row_number() %in% auto_trn_idx)


# Function to calculate RMSE
rmse = function(n, y, y_hat){
  output = sqrt(sum((y - y_hat)^2) / n)
  return(output)
}

# Create variables to hold data for calculating RMSE
n_trn = length(auto_trn$mpg)
n_tst = length(auto_tst$mpg)

y_trn = auto_trn$mpg
y_tst = auto_tst$mpg


# All predictors
all_trn = lm(mpg ~ ., data = auto_trn)
all_tst = predict(all_trn, newdata = auto_tst)

rmse_all_trn = rmse(n_trn, y_trn, all_trn$fitted.values)
rmse_all_tst = rmse(n_tst, y_tst, all_tst)


# Only Displacement as predictor
disp_trn = lm(mpg ~ displacement, data = auto_trn)
disp_tst = predict(disp_trn, newdata = auto_tst)

rmse_disp_trn = rmse(n_trn, y_trn, disp_trn$fitted.values)
rmse_disp_tst = rmse(n_tst, y_tst, disp_tst)


# Weight, year and origin
mod_3_trn = lm(mpg ~ weight + year + origin, data = auto_trn)
mod_3_tst = predict(mod_3_trn, newdata = auto_tst)

rmse_3_trn = rmse(n_trn, y_trn, mod_3_trn$fitted.values)
rmse_3_tst = rmse(n_tst, y_tst, mod_3_tst)
```

Predictors	Training RMSE	Testing RMSE
All	3.4129	2.9681
Displacement	4.7399	4.3005
Weight, Year and Origin	3.4606	2.9451
Weight, Year, Origin and Displacement	3.4580	2.9286
Weight, Year and Displacement	3.5339	3.0619

```
# Weight, year, origin and displacement
mod_4_trn = lm(mpg ~ weight + year + origin + displacement, data = auto_trn)
mod_4_tst = predict(mod_4_trn, newdata = auto_tst)

rmse_4_trn = rmse(n_trn, y_trn, mod_4_trn$fitted.values)
rmse_4_tst = rmse(n_tst, y_tst, mod_4_tst)


# Weight, year and displacement
mod_5_trn = lm(mpg ~ weight  + year + displacement, data = auto_trn)
mod_5_tst = predict(mod_5_trn, newdata = auto_tst)

rmse_5_trn = rmse(n_trn, y_trn, mod_5_trn$fitted.values)
rmse_5_tst = rmse(n_tst, y_tst, mod_5_tst)


# Dataframe for table output
display_table = tibble(Predictors = c('All', 'Displacement', 'Weight, Year and Origin', 'Weight, Year, (
                                      'Weight, Year and Displacement'),
                       `Training RMSE` = c(rmse_all_trn, rmse_disp_trn, rmse_3_trn, rmse_4_trn, rmse_5_
                       `Testing RMSE` = c(rmse_all_tst, rmse_disp_tst, rmse_3_tst, rmse_4_tst, rmse_5_ts
  mutate(`Training RMSE` = scales::number(`Training RMSE`, 0.0001),
         `Testing RMSE` = scales::number(`Testing RMSE`, 0.0001))


display_table %>%
  kbl() %>%
  kable_styling()
```

The model using Weight, Year, Origin and Displacement (row 4 in the table) would be the preferred model for predicting MPG since the testing RMSE is the lowest of the 5 models.

Exercise 5 (Simulating Multiple Regression)

For this exercise we will simulate data from the following model:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \beta_5 x_{i5} + \epsilon_i$$

Where $\epsilon_i \sim N(0, \sigma^2)$. Also, the parameters are known to be:

- $\beta_0 = 2$

- $\beta_1 = -0.75$
- $\beta_2 = 1.6$
- $\beta_3 = 0$
- $\beta_4 = 0$
- $\beta_5 = 2$
- $\sigma^2 = 25$

We will use samples of size `n = 40`.

We will verify the distribution of $\hat{\beta}_1$ as well as investigate some hypothesis tests.

(a) We will first generate the X matrix and data frame that will be used throughout the exercise. Create the following nine variables:

- `x0`: a vector of length `n` that contains all `1`
- `x1`: a vector of length `n` that is randomly drawn from a normal distribution with a mean of `0` and a standard deviation of `2`
- `x2`: a vector of length `n` that is randomly drawn from a uniform distribution between `0` and `4`
- `x3`: a vector of length `n` that is randomly drawn from a normal distribution with a mean of `0` and a standard deviation of `1`
- `x4`: a vector of length `n` that is randomly drawn from a uniform distribution between `-2` and `2`
- `x5`: a vector of length `n` that is randomly drawn from a normal distribution with a mean of `0` and a standard deviation of `2`
- `X`: a matrix that contains `x0`, `x1`, `x2`, `x3`, `x4`, and `x5` as its columns
- `C`: the C matrix that is defined as $(X^\top X)^{-1}$
- `y`: a vector of length `n` that contains all `0`
- `sim_data`: a data frame that stores `y` and the **five** *predictor* variables. `y` is currently a placeholder that we will update during the simulation.

Report the sum of the diagonal of `C` as well as the 5th row of `sim_data`. For this exercise we will use the seed `420`. Generate the above variables in the order listed after running the code below to set a seed.

Solution:

```
set.seed(400)
#sample_size = 40
n = 40

# Vectors
x0 = rep(1, n)
x1 = rnorm(n, mean = 0, sd = 2)
x2 = runif(n, min = 0, max = 4)
x3 = rnorm(n, mean = 0, sd = 1)
x4 = runif(n, min = -2, max = 2)
x5 = rnorm(n, mean = 0, sd = 2)

# X matrix
X = matrix(c(x0, x1, x2, x3, x4, x5), nrow = n, ncol = 6)

# C matrix
C = solve(t(X) %*% X)

# Empty vector y
y = rep(0, n)
```

```r
# Dataframe for simulated data
sim_data = data.frame(y, x1, x2, x3, x4, x5)

# Reporting
C_diag_sum = sum(diag(C))
sim_data_row5 = sim_data %>%
  filter(row_number() == 5)
```

Sum of the diagonal of Matrix C: 0.1763287

Fifth row of `sim_data` dataframe: 0, -1.20367714263776, 3.3941135359928, -0.0920876558277171, 1.74332613591105, -0.780832901648995

(b) Create three vectors of length `2500` that will store results from the simulation in part **(c)**. Call them `beta_hat_1`, `beta_3_pval`, and `beta_5_pval`.

Solution:

```r
len = 2500

beta_hat_1 = rep(0, len)
beta_3_pval = rep(0, len)
beta_5_pval= rep(0, len)
```

(c) Simulate 2500 samples of size `n = 40` from the model above. Each time update the `y` value of `sim_data`. Then use `lm()` to fit a multiple regression model. Each time store:

- The value of $\hat{\beta}_1$ in `beta_hat_1`
- The p-value for the two-sided test of $\beta_3 = 0$ in `beta_3_pval`
- The p-value for the two-sided test of $\beta_5 = 0$ in `beta_5_pval`

Solution:

```r
# Parameters for model and simulation
num_sims = 2500

beta_1 = -0.75
beta_2 = 1.6
beta_3 = 0
beta_4 = 0
beta_5 = 2
sigma = sqrt(25)

sim_results = tibble(beta_hat_1, beta_3_pval, beta_5_pval)


for(i in 1:num_sims){
  # Noise
  eps = rnorm(n, mean = 0, sd = sigma)

  # Calculate y from simulated data and noise
  sim_data = sim_data %>%
    mutate(y = x1 * beta_1 + x2 * beta_2 + x3 * beta_3 + x4 * beta_4 + x5 * beta_5 + eps)
```

```
  # Fit model from simulated data
  fit = lm(y ~ x1 + x2 + x3 + x4 + x5, data = sim_data)

  # Get results from model
  bh_1 = summary(fit)$coefficients['x1','Estimate']
  b3_pval = summary(fit)$coefficients['x3','Pr(>|t|)']
  b5_pval = summary(fit)$coefficients['x5','Pr(>|t|)']

  # Store results in dataframe
  sim_results$beta_hat_1[i] = bh_1
  sim_results$beta_3_pval[i] = b3_pval
  sim_results$beta_5_pval[i] = b5_pval

}
```

(d) Based on the known values of X, what is the true distribution of $\hat{\beta}_1$?

Solution:

```
# Calculate standard deviation of beta_hat_1 from C matrix
var_beta_hat_1 = (25 * C[2,2])
```

The true distribution of $\hat{\beta}_1$ is a mean of -.75 and a standard deviation of 0.2230

(e) Calculate the mean and variance of `beta_hat_1`. Are they close to what we would expect? Plot a histogram of `beta_hat_1`. Add a curve for the true distribution of $\hat{\beta}_1$. Does the curve seem to match the histogram?

Solution:

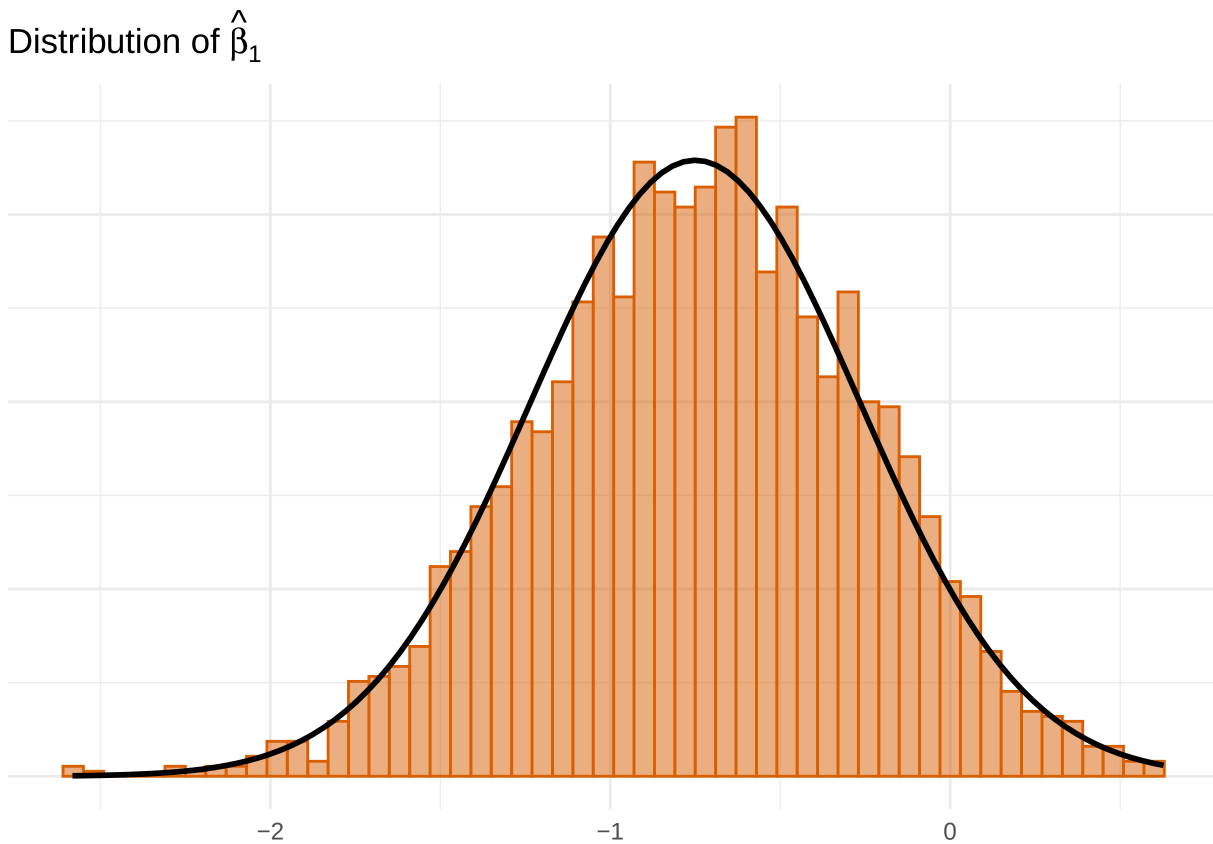
```
bh_1_mean = mean(sim_results$beta_hat_1)
bh_1_var = sd(sim_results$beta_hat_1)^2
```

The estimated mean (-0.7359) and variance (0.2353) for $\hat{\beta}_1$ are very close to the known values of β_1.

```
library(latex2exp)

hist_beta_1 = ggplot(sim_results, aes(x = beta_hat_1)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.06, fill = "#d95f02", color = "#d95f02", alpha = 0.5
  geom_function(fun = dnorm, args = list(mean = -0.75, sd = sqrt(bh_1_var)), size = 1) +
  theme_minimal() +
  theme(axis.text.y = element_blank()) +
  labs(x = NULL,
       y = NULL,
       title = TeX(r'(Distribution of $\hat{\beta}_1$)'))

hist_beta_1
```



The curve for the known distribution for β_1 does seem to match the estimate of $\hat{\beta}_1$ from the simulation.

(f) What proportion of the p-values stored in `beta_3_pval` is less than 0.10? Is this what you would expect?

Solution:

```
b3_pval_prop = sim_results %>%
  filter(beta_3_pval < 0.10) %>%
  summarise(prop = length(beta_3_pval) / 2500)
```

The proportion of `beta_3_pval` less than 0.10 is 0.1012. Yes, I would expect β_3 p-value to have a relatively low proportion of significant p-values since the true value of β_3 is 0 and the coefficient should not affect the model since the value is 0.

(g) What proportion of the p-values stored in `beta_5_pval` is less than 0.01? Is this what you would expect?

Solution:

```
b5_pval_prop = sim_results %>%
  filter(beta_5_pval < 0.01) %>%
  summarise(prop = length(beta_5_pval) / 2500)
```

The proportion of `beta_5_pval` less than 0.01 is 0.8564. Yes, I would expect a large proportion of p-values for β_5 to be significant since the true value of β_5 is 2 and the coefficient should affect the performance of the model.