Week 4 - Homework

STAT 420, Summer 2022, Ilya Andreev, iandre3

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.

```
nutrition = read.csv("nutrition-2018.csv")
x1 = nutrition$Fat
x2 = nutrition$Sugar
x3 = nutrition$Sodium
y = nutrition$Calories
nutrition_model = lm(y ~ x1 + x2 + x3)
sum = summary(nutrition_model)
```

The null hypothesis: $\beta_1 = \beta_2 = \beta_3 = 0$. The alternative hypothesis: at least one of $\beta_1, \beta_2, \beta_3 \neq 0$. The value of the test statistic: 6590.9402239.

The p-value of the test is under 2.2e-16.

At $\alpha = 0.01$, the statistical decision is to reject the null hypothesis.

We conclude that at least one of the three predictors has a linear relationship with the response.

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

```
coef(nutrition_model)
```

```
##  (Intercept)           x1           x2           x3
## 1.004561e+02 8.483289e+00 3.900517e+00 6.165246e-03
```

The model interpretation:

- The caloric value with no fat, sugar, or sodium is 100.4560567.
- For a fixed amount of sugar and sodium, an increase in fat by a gram leads to an increase in calories by 8.4832891.
- For a fixed amount of fat and sodium, an increase in sugar by a gram leads to an increase in calories by 3.9005172.
- For a fixed amount of fat and sugar, an increase in sodium by a milligram leads to an increase in calories by 0.0061652.

(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.

```
newdata = data.frame(list(18, 5, 580))
colnames(newdata) = c("x1", "x2", "x3")
predict(nutrition_model, newdata=newdata)
```

```
##        1
## 276.2337
```

(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.

The standard deviation s_y of observed values is 168.0499661.

The s_e from the multiple regression model is 80.8543023. The difference in the value is explained by the fact that s_e is an estimate that uses a smaller number of degrees of freedom.

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

The value of R^2 of the model is 0.7686281. This the share of variability in the response that is explained by the predictors included in the model.

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

We can say with 90% confidence that the value of β_2 lies 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.

We can say with 95% confidence that the value of β_0 lies between 98.1385369 and 102.7735765.

(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.

```
newdata = data.frame(list(15, 0, 260))
colnames(newdata) = c("x1", "x2", "x3")
res = predict(nutrition_model, interval="confidence", level=0.99, newdata=newdata)
```

We can say with 99% confidence that the mean Calorie content of a food with 15g of fat, 0g of sugar, and 260mg of sodium lies between 226.1657341 and 232.4509796. Our point estimate of the mean is 229.3083568.

(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.

```
newdata = data.frame(list(11, 2, 340))
colnames(newdata) = c("x1", "x2", "x3")
res = predict(nutrition_model, interval="prediction", level=0.99, newdata=newdata)
```

We can say with 99% confidence that the Calorie content of a food with 11g of fat, 2g of sugar, and 340mg of sodium lies between -4.6844814 and 412.0233906. Our point estimate of the mean is 203.6694546.

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`.

```
goalie_data = read.csv("goalies17.csv")
goalie_data = goalie_data[ , !(names(goalie_data) %in% c("Player", "GS", "L", "TOL", "SV_PCT", "GAA"))]
head(goalie_data)
```

```
##   First Last Active GP W GA SA SV SO PIM MIN
## 1  2010 2011      1  1 0  5 12  7  0   0  40
## 2  2006 2007      1  1 0  1  3  2  0   0   2
## 3  2008 2009      1  1 0  3  9  6  0   0  20
## 4  2015 2016      1  1 0  1  3  2  0   0  18
## 5  2000 2001      1  1 0  3 10  7  0   0  25
## 6  2011 2012      1  1 1  3 10  7  0   0  41
```

```
model1 = lm(goalie_data$W ~ goalie_data$GA + goalie_data$SV, goalie_data)
model2 = lm(goalie_data$W ~ goalie_data$GA + goalie_data$SV + goalie_data$SA + goalie_data$MIN + goalie
model3 = lm(goalie_data$W ~ ., goalie_data)
```

(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

```
a = anova(model1, model2)
a
```

```
## Analysis of Variance Table
##
## Model 1: goalie_data$W ~ goalie_data$GA + goalie_data$SV
## Model 2: goalie_data$W ~ goalie_data$GA + goalie_data$SV + goalie_data$SA +
##     goalie_data$MIN + goalie_data$SO
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    500 310758
## 2    497  77763  3    232996 496.38 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Null hypothesis: $\beta_3 = \beta_4 = \beta_5 = 0$.

The value of the test statistic: 496.3764519.

The p-value of the test is under 2.2e-16.

The statistical decision at $\alpha = 0.05$ is to reject the null hypothesis

The preferred model is the full model, i.e. model 2.

(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

```
a = anova(model2, model3)
a
```

```
## Analysis of Variance Table
##
## Model 1: goalie_data$W ~ goalie_data$GA + goalie_data$SV + goalie_data$SA +
##     goalie_data$MIN + goalie_data$SO
## Model 2: goalie_data$W ~ First + Last + Active + GP + GA + SA + SV + SO +
##     PIM + MIN
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    497 77763
## 2    492 69133  5    8629.8 12.283 3.073e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Null hypothesis: the coefficients for `First`, `Last`, `Active`, `GP`, `PIM` are 0.

The value of the test statistic: 12.2831933.

The p-value of the test is 3.073e-11.

The statistical decision at $\alpha = 0.05$ is to reject the null hypothesis.

The preferred model is the full model, i.e. model 3.

(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 $\alpha = 0.05$

```
summary(model3)$coefficients
```

```
##                Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) 22.75771538 1.242128e+02  0.1832156 8.547043e-01
## First        1.07263612 5.468259e-01  1.9615680 5.037627e-02
## Last        -1.08328270 5.470809e-01 -1.9801144 4.824737e-02
## Active       0.57344962 8.024510e-01  0.7146226 4.751812e-01
## GP          -0.44852640 1.200002e-01 -3.7377124 2.075063e-04
## GA          -0.10166978 1.413883e-02 -7.1908226 2.410486e-12
## SA           0.05102737 1.249799e-02  4.0828458 5.191544e-05
## SV          -0.05668788 1.390426e-02 -4.0770141 5.319041e-05
## SO           0.38073227 1.868746e-01  2.0373682 4.214774e-02
## PIM          0.04172541 1.287684e-02  3.2403460 1.274679e-03
## MIN          0.02044002 2.040587e-03 10.0167353 1.285060e-21
```

The value of the test statistic is -4.0770141.

The p-value of the test is 5.3190411×10^{-5}.

The statistical decision at $\alpha = 0.05$ is to 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)`.

```
x = data.matrix(Ozone[ , (names(Ozone) %in% c("wind", "humidity", "temp"))])
x = cbind(Bias=1, x)
y = data.matrix(Ozone["ozone"])
x_t = t(x)
beta_hat_no_lm = solve(x_t %*% x) %*% x_t %*% y
beta_hat_no_lm = as.vector(beta_hat_no_lm)
sum_sq_beta_hat_no_lm = sum(beta_hat_no_lm ^ 2)

beta_hat_no_lm
```

```
## [1] -16.38178539  -0.18594444   0.08340014   0.38984294
```

```
sum_sq_beta_hat_no_lm
```

```
## [1] 268.5564
```

(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)`.

```
m = lm(Ozone$ozone ~ Ozone$wind + Ozone$humidity + Ozone$temp)
beta_hat_lm = as.vector(summary(m)$coefficients[, 1])
sum_sq_beta_hat_lm = sum(beta_hat_lm ^ 2)

beta_hat_lm
```

```
## [1] -16.38178539  -0.18594444   0.08340014   0.38984294
```

```
sum_sq_beta_hat_lm
```

```
## [1] 268.5564
```

(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()`.

```
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()`.

```
y_hat = beta_hat_no_lm[1] + beta_hat_no_lm[2] * Ozone$wind + beta_hat_no_lm[3] * Ozone$humidity + beta_
e = as.vector(y - y_hat)
s_e = as.numeric(sqrt((t(e) %*% e) / (length(y) - (length(beta_hat_no_lm) - 1))))

s_e
```

```
## [1] 4.799062
```

```
sigma(m)
```

```
## [1] 4.806115
```

```
all.equal(s_e, sigma(m))
```

```
## [1] "Mean relative difference: 0.001469509"
```

(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()`.

```
r_squared_expected = summary(m)$r.squared
r_squared_manual = 1 - sum(e ^ 2) / sum((y - mean(y)) ^ 2)
r_squared_manual
```

```
## [1] 0.6398887
```

```
r_squared_expected
```

```
## [1] 0.6398887
```

```
all.equal(r_squared_expected, r_squared_manual)
```

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

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)
auto_test_idx = setdiff(1:nrow(Auto), auto_trn_idx)
train_data = Auto[auto_trn_idx,]
test_data = Auto[auto_test_idx,]
```

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.

```
model_one = lm(mpg ~ ., train_data)
model_two = lm(mpg ~ displacement, train_data)
model_three = lm(mpg ~ year + weight + cylinders + displacement + horsepower + weight, train_data)
model_four = lm(mpg ~ year + weight + cylinders + displacement + acceleration + origin, train_data)
model_five = lm(mpg ~ cylinders + horsepower + acceleration, train_data)

calculate_test_rmse = function(model) {
  y_hat = predict(model, newdata=test_data)
  e = y_hat - test_data$mpg
  rmse = sqrt(mean(e ^ 2))
  rmse
}

train_rmse_one = sqrt(mean(residuals(model_one)^2))
train_rmse_two = sqrt(mean(residuals(model_two)^2))
train_rmse_three = sqrt(mean(residuals(model_three)^2))
train_rmse_four = sqrt(mean(residuals(model_four)^2))
train_rmse_five = sqrt(mean(residuals(model_five)^2))

res = cbind(
        c(
          train_rmse_one,
          train_rmse_two,
          train_rmse_three,
          train_rmse_four,
          train_rmse_five
        ),
        c(
          calculate_test_rmse(model_one),
```

```
          calculate_test_rmse(model_two),
          calculate_test_rmse(model_three),
          calculate_test_rmse(model_four),
          calculate_test_rmse(model_five)
        )
      )
colnames(res) = c('Train RMSE', 'Test RMSE')
rownames(res) = c('Model 1', 'Model 2', 'Model 3', 'Model 4', 'Model 5')
knitr::kable(res)
```

	Train RMSE	Test RMSE
Model 1	3.412868	2.968136
Model 2	4.739880	4.300485
Model 3	3.520207	3.104366
Model 4	3.425229	2.942623
Model 5	4.642395	4.050398

Empirically, model 4 performs the best on the test data set. This model uses all available predictors except horsepower. The fact that model 4 performs worse than model 1 on the train data set is inconsequential, since the train data set is meant for model fitting, not evaluation.

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.

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

```
n = 40
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 = cbind(x0, x1, x2, x3, x4, x5)
C = solve(t(x) %*% x)
y = rep(0, n)
sim_data = data.frame(y, x1, x2, x3, x4, x5)
head(sim_data)
```

```
##   y        x1        x2          x3         x4         x5
## 1 0 -2.073098 3.7959235 -1.01627809 -0.2905468 -0.0379368
## 2 0  1.230567 0.4225633  0.44983312  1.5408233  0.3904568
## 3 0  2.945865 1.5902661  0.88275370  1.8234378 -0.5156721
## 4 0 -1.365375 0.7021491  0.01193124 -1.6156026 -3.3439114
## 5 0 -1.203677 3.3941135 -0.09208766  1.7433261 -0.7808329
## 6 0 -2.705219 0.7646769 -0.86439389  1.7151162  0.1283220
```

The sum of the diagonal of `C` is 0.1763287 and the 5th row of `sim_data` is 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`.

```
beta_hat_1 = rep(0, 2500)
beta_3_pval = rep(0, 2500)
beta_5_pval = rep(0, 2500)
head(sim_data)
```

```
##   y        x1        x2          x3         x4         x5
## 1 0 -2.073098 3.7959235 -1.01627809 -0.2905468 -0.0379368
## 2 0  1.230567 0.4225633  0.44983312  1.5408233  0.3904568
## 3 0  2.945865 1.5902661  0.88275370  1.8234378 -0.5156721
## 4 0 -1.365375 0.7021491  0.01193124 -1.6156026 -3.3439114
## 5 0 -1.203677 3.3941135 -0.09208766  1.7433261 -0.7808329
## 6 0 -2.705219 0.7646769 -0.86439389  1.7151162  0.1283220
```

(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:

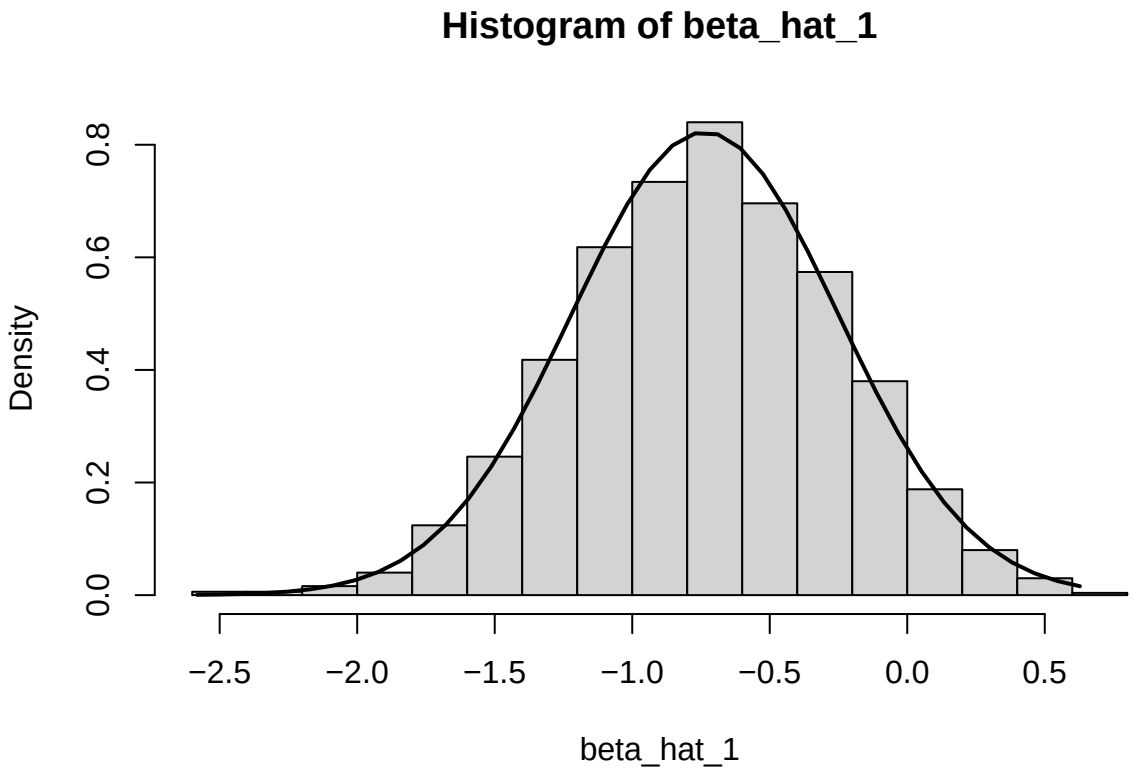
- The value of $\hat{\beta}_1$ in `beta_hat_1`
- The p-value for the two-sided test of $\beta_3 = 0$ in `beta_3_pval`
- The p-value for the two-sided test of $\beta_5 = 0$ in `beta_5_pval`

```
for (i in 1:2500) {
  y = 2 - 0.75 * x1 + 1.6 * x2 + 0 * x3 + 0 * x4 + 2 * x5 + rnorm(40, mean = 0, sd = 5)
  sim_data[, 1] = y
  m = lm(y ~ x1 + x2 + x3 + x4 + x5, sim_data)
  beta_hat_1[i] = coef(m)[2]
  beta_3_pval[i] = summary(m)$coefficients[4, 4]
  beta_5_pval[i] = summary(m)$coefficients[6, 4]
}
```

(d) Based on the known values of X, what is the true distribution of $\hat{\beta}_1$? The true distribution of $\hat{\beta}_1$ is $N(\beta, 0.2230073)$

(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?

```
hist(beta_hat_1, freq=FALSE)
xf = seq(min(beta_hat_1), max(beta_hat_1), length=40)
yf = dnorm(xf, mean=mean(beta_hat_1), sd=sd(beta_hat_1))
lines(xf, yf, col="black", lwd=2)
```



The mean is -0.7359474. The variance is 0.2352889. The curve for the true distribution matches the empirical histogram. Same goes for the mean and variance of the true distribution and the observed mean and variance.

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

This is a surprisingly large proportion given that the true value of the coefficient is known to be zero. Still, in 10.12% of the samples we find enough evidence to assume that the coefficient is not zero.

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

```
sum(beta_5_pval < 0.01) / length(beta_5_pval)
```

```
## [1] 0.8564
```

We know that the true value of the coefficient is 2, and it is somewhat surprising that the proportion of p-values stored in `beta_5_pval` that is less than 0.01 is 0.1012. This means that in 89.88% of the cases we would fail to reject the null hypothesis that assume that the value of the coefficient is 0.