Week 4 - Homework

STAT 420, Fall 2024, Banghao Chi

09/23/2024

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.

```
data = read.csv("nutrition-2018.csv")
model = lm(Calories ~ Fat + Sugar + Sodium, data = data)
f_test = summary(model)

f_statistic = f_test$fstatistic[1]
f_statistic
```

```
##   value
## 6590.94
```

```
p_value = pf(f_statistic, f_test$fstatistic[2], f_test$fstatistic[3], lower.tail = FALSE)
p_value
```

```
## value
##     0
```

- Null and Alternative Hypotheses:
- – Null Hypothesis: $\beta_1 = \beta_2 = \beta_3 = 0$, meaning all regression coefficients are zero.
- – Alternative Hypothesis: At least one $\beta_j \neq 0$ (At least one regression coefficient is not zero).
- Value of the test statistic: 6590.9402239.
- P-value of the test: 0
- Statistical decision at $\alpha = 0.01$:
- – Since the p-value we have is much smaller than $\alpha = 0.01$, we reject the null hypothesis.
- Conclusion in the context of the problem:
- – There is strong evidence to suggest that at least one of the predictor variables (Fat, Sugar, or Sodium) has a significant linear relationship with the response variable (Calories). This indicates that the regression model with these predictors is significant in explaining the variation in calorie content of foods.

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

```
coef(model)
```

```
##  (Intercept)          Fat        Sugar       Sodium
## 1.004561e+02 8.483289e+00 3.900517e+00 6.165246e-03
```

- For β_0: When fat, sugar, and sodium are all zero, the **mean** calorie content is 1.004561e+02 calories.
- For β_1: When at a specific level of sugar and sodium, for each additional gram of fat, the **mean** calorie content is expected to increase by 8.483289 calories.
- For β_2: When at a specific level of fat and sodium, for each additional gram of sugar, the **mean** calorie content is expected to increase by 3.900517 calories.
- For β_3: When at a specific level of fat and sugar constant, for each additional milligram of sodium, the **mean** calorie content is expected to increase by 6.165246e-03 calories.

(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.

```r
filet_o_fish = data.frame(Fat = 18, Sugar = 5, Sodium = 580)
predicted_calories = predict(model, newdata = filet_o_fish)
predicted_calories
```

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

According to McDonald's nutrition facts, a Filet-O-Fish contains 276.2336888 calories.

(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.

```r
sy <- sd(data$Calories)
sy
```

```
## [1] 168.05
```

```r
se <- summary(model)$sigma
se
```

```
## [1] 80.8543
```

Interpretation:

- For standard deviation: On average, the calorie content of foods deviates from the mean calorie content by about 168.0499661 calories.
- For residual standard error: On average, the calorie content of foods deviates from the mean calorie content by about 80.8543023 calories according to our regression model.

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

```r
r_squared = summary(model)$r.squared
r_squared
```

```
## [1] 0.7686281
```

0.7686281 of the variability in calorie content can be explained by our model using fat, sugar, and sodium as predictors.

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

```r
ci_sugar = confint(model, "Sugar", level = 0.90)
ci_sugar
```

```
##            5 %     95 %
## Sugar 3.783051 4.017983
```

We are 90% confident that the true mean value of the effect of sugar on calories lies between 3.783051 and 4.017983.

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

```r
ci_intercept <- confint(model, "(Intercept)", level = 0.95)
ci_intercept
```

```
##                2.5 %   97.5 %
## (Intercept) 97.69443 103.2177
```

We are 95% confident that the the calories of the food when all the predictors are at zero, lies between 97.69443 and 103.2177 calories.

(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.

```
new_data <- data.frame(Fat = 15, Sugar = 0, Sodium = 260)
ci_pred <- predict(model, newdata = new_data, interval = "confidence", level = 0.99)
ci_pred
```

```
##        fit      lwr     upr
## 1 229.3084 226.1657 232.451
```

We are 99% confident that the mean calorie content for foods with these nutritional characteristics (15g fat, 0g sugar, 260mg sodium) falls between 226.1657 and 232.451 calories.

(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.

```
new_data <- data.frame(Fat = 11, Sugar = 2, Sodium = 340)
pi_pred <- predict(model, newdata = new_data, interval = "prediction", level = 0.99)
pi_pred
```

```
##        fit       lwr      upr
## 1 203.6695 -4.684481 412.0234
```

We are 99% confident that the actual calorie content of a Crunchy Taco Supreme falls between -4.684481 and 412.0234 calories.

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`.

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

(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

```
model1 = lm(W ~ GA + SV, data = goalies)
model2 = lm(W ~ GA + SV + SA + MIN + SO, data = goalies)

f_test = anova(model1, model2)

f_statistic =  f_test["F"][2, ]
f_statistic
```

```
## [1] 496.3765
```

```
p_value =  f_test["Pr(>F)"][2, ]
p_value
```

```
## [1] 4.766687e-149
```

- The null hypothesis: Model 1 is adequate.
- The value of the test statistic is: 496.3764519.
- The p-value of the test is: $4.7666873 \times 10^{-149}$
- A statistical decision at $\alpha = 0.05$: Since $p < \alpha = 0.05$, we reject the null hypothesis.
- The statistical decision at $\alpha = 0.05$ shows that the smaller model is not enough. Therefore, I prefer the Model 2, the larger one.

(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

```
model3 = lm(W ~ ., data = goalies)

f_test = anova(model2, model3)

f_statistic =  f_test["F"][2, ]
f_statistic
```

```
## [1] 12.28319
```

```
p_value =  f_test["Pr(>F)"][2, ]
p_value
```

```
## [1] 3.073007e-11
```

- The null hypothesis: Model 2 is adequate.
- The value of the test statistic is: 12.2831933.
- The p-value of the test is: $3.0730074 \times 10^{-11}$
- A statistical decision at $\alpha = 0.05$: Since $p < \alpha = 0.05$, we reject the null hypothesis.
- The statistical decision at $\alpha = 0.05$ shows that the smaller model is not enough. Therefore, I prefer the Model 3, the larger one.

(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 $\alpha = 0.05$

```
model_summary <- summary(model3)

t_statistic <- model_summary$coefficients["SV", "t value"]
t_statistic
```

```
## [1] -4.077014
```

```
p_value <- model_summary$coefficients["SV", "Pr(>|t|)"]
p_value
```

```
## [1] 5.319041e-05
```

- The value of the test statistic is: -4.0770141
- The p-value of the test is: 5.3190411×10^{-5}
- A statistical decision at $\alpha = 0.05$ is: Since $p < \alpha = 0.05$, we 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 = cbind(1, Ozone[, "wind"], Ozone[, "humidity"], Ozone[, "temp"])
y = Ozone$ozone

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

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

```
sum_squared_coefficients = sum(beta_hat_no_lm ^ 2)
sum_squared_coefficients
```

```
## [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)`.

```
model_lm = lm(ozone ~ wind + humidity + temp, data = Ozone)
beta_hat_lm = as.vector(coef(model_lm))

sum_squared_coefficients_lm = sum(beta_hat_lm ^ 2)
sum_squared_coefficients_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(sum_squared_coefficients, sum_squared_coefficients_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 = X %*% beta_hat_no_lm

residuals = y - y_hat

n = nrow(X)
p = ncol(X)
df = n - p

s_e_no_lm = sqrt(sum(residuals^2) / df)

model_lm = lm(ozone ~ wind + humidity + temp, data = Ozone)
s_e_lm = summary(model_lm)$sigma

all.equal(s_e_no_lm, s_e_lm)
```

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

(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()`.

```
y_mean = mean(y)
TSS = sum((y - y_mean)^2)

RSS = sum(residuals^2)

R_squared_no_lm = 1 - (RSS / TSS)

model_lm = lm(ozone ~ wind + humidity + temp, data = Ozone)
R_squared_lm = summary(model_lm)$r.squared

all.equal(R_squared_no_lm, R_squared_lm)
```

```
## [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)
```

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.

```r
set.seed(22)
auto_trn_idx = sample(1:nrow(Auto), 290)
auto_trn = Auto[auto_trn_idx, ]
auto_tst = Auto[-auto_trn_idx, ]

rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

model1 <- lm(mpg ~ ., data = auto_trn)
train_rmse1 <- rmse(auto_trn$mpg, predict(model1, auto_trn))
test_rmse1 <- rmse(auto_tst$mpg, predict(model1, auto_tst))

model2 <- lm(mpg ~ displacement, data = auto_trn)
train_rmse2 <- rmse(auto_trn$mpg, predict(model2, auto_trn))
test_rmse2 <- rmse(auto_tst$mpg, predict(model2, auto_tst))

model3 <- lm(mpg ~ weight + horsepower, data = auto_trn)
train_rmse3 <- rmse(auto_trn$mpg, predict(model3, auto_trn))
test_rmse3 <- rmse(auto_tst$mpg, predict(model3, auto_tst))

model4 <- lm(mpg ~ weight + year + origin, data = auto_trn)
train_rmse4 <- rmse(auto_trn$mpg, predict(model4, auto_trn))
test_rmse4 <- rmse(auto_tst$mpg, predict(model4, auto_tst))

model5 <- lm(mpg ~ weight + horsepower + year + origin, data = auto_trn)
train_rmse5 <- rmse(auto_trn$mpg, predict(model5, auto_trn))
test_rmse5 <- rmse(auto_tst$mpg, predict(model5, auto_tst))

library(knitr)

results <- data.frame(
  Model = c("1. All predictors",
            "2. Only displacement",
            "3. Weight and horsepower",
            "4. Weight, year, and origin",
            "5. Weight, horsepower, year, and origin"),
  Train_RMSE = c(train_rmse1, train_rmse2, train_rmse3, train_rmse4, train_rmse5),
  Test_RMSE = c(test_rmse1, test_rmse2, test_rmse3, test_rmse4, test_rmse5)
)

kable(results, format = "markdown",
      col.names = c("Model", "Train RMSE", "Test RMSE"),
      align = c("l", "r", "r"))
```

Model	Train RMSE	Test RMSE
1. All predictors	3.412868	2.968136
2. Only displacement	4.739880	4.300485
3. Weight and horsepower	4.355525	3.855136
4. Weight, year, and origin	3.460605	2.945094
5. Weight, horsepower, year, and origin	3.441924	3.011585

Since the Test RMSE for the fifth model, which set weight, horsepower, year, and origin as predictors, is the least, this means that it is the best model within these five.

The reason

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(420)
sample_size = 40
```

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

```
X = cbind(x0, x1, x2, x3, x4, x5)

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

y = rep(0, sample_size)

sim_data = data.frame(y = y, x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = x5)

sum_diag_C = sum(diag(C))
sum_diag_C
```

```
## [1] 0.1734533
```

```
sim_data[5, ]
```

```
##   y        x1       x2       x3       x4        x5
## 5 0 0.7959582 3.338119 1.335327 1.830513 0.6028196
```

(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)
```

(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`

```
beta0 = 2
beta1 = -0.75
beta2 = 1.6
beta3 = 0
beta4 = 0
beta5 = 2
sigma_squared = 25

for(i in 1:2500) {
  epsilon = rnorm(sample_size, mean = 0, sd = sqrt(sigma_squared))

  sim_data$y = beta0 + beta1*sim_data$x1 + beta2*sim_data$x2 + beta3*sim_data$x3 + beta4*sim_data$x4 + b

  model = lm(y ~ x1 + x2 + x3 + x4 + x5, data = sim_data)

  beta_hat_1[i] = coef(model)["x1"]
  beta_3_pval[i] = summary(model)$coefficients["x3", "Pr(>|t|)"]
  beta_5_pval[i] = summary(model)$coefficients["x5", "Pr(>|t|)"]
}
```

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

The true distribution of $\hat{\beta}_1$ follows a normal distribution. In multiple linear regression, the vector of estimated coefficients $\hat{\beta}$ follows a multivariate normal distribution, with the distribution of a single element inside estimated coefficients vectors is also normal. For the estimated coefficients vectors, we have:

$$\hat{\beta} \sim N\left(\beta, \sigma^2 (X^\top X)^{-1}\right)$$

where $\hat{\beta}_1$ within the estimated coefficients vectors follows:

$$\hat{\beta}_1 \sim N\left(\beta_1, \sigma^2 c_{22}\right)$$

where c_{22} is computed from X.

(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?

```
mean_beta_hat_1 = mean(beta_hat_1)
mean_beta_hat_1
```

```
## [1] -0.7418858
```

```
var_beta_hat_1 = var(beta_hat_1)
var_beta_hat_1
```

```
## [1] 0.1754524
```

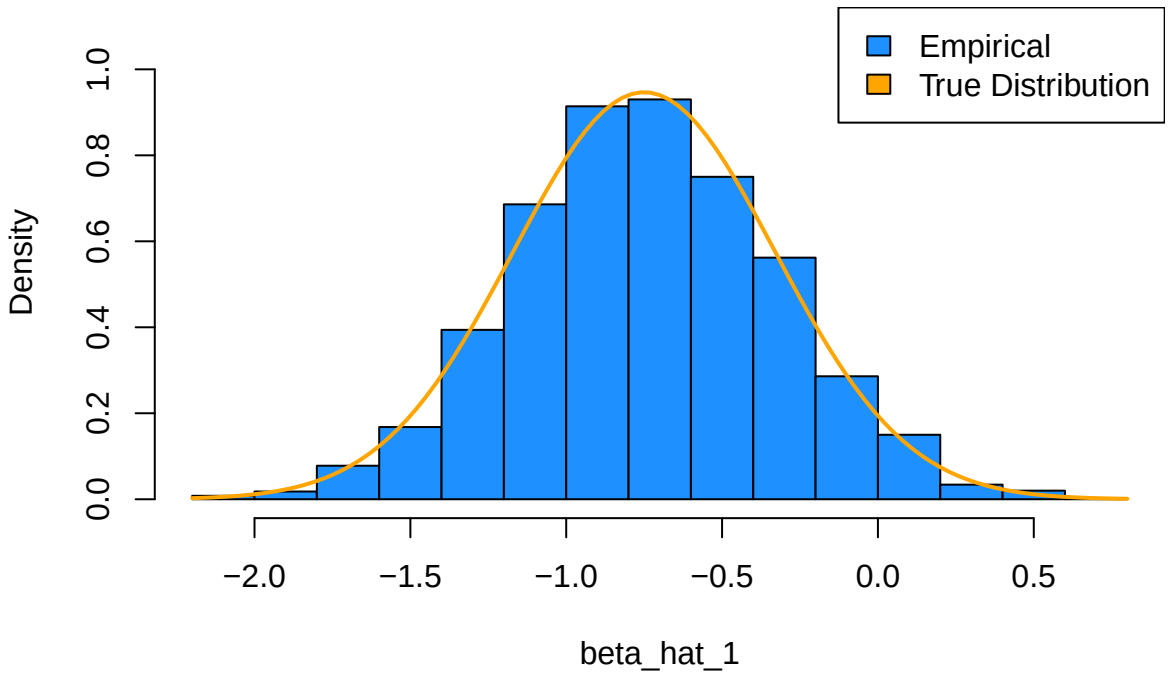
```
true_mean = -0.75
true_mean
```

```
## [1] -0.75
```

```
true_var = 25 * C[2,2]
true_var
```

```
## [1] 0.1775346
```

```
hist(beta_hat_1, prob = TRUE, main = "Distribution of beta_hat_1",
     xlab = "beta_hat_1", ylim = c(0, 1.1), col = "dodgerblue")
curve(dnorm(x, mean = true_mean, sd = sqrt(true_var)),
      col = "orange", lwd = 2, add = TRUE)
legend("topright", legend = c("Empirical", "True Distribution"),
       fill = c("dodgerblue", "orange"))
```



Yes, they are close to what we would expect, and the curve seems to match the histogram!

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

```
proportion_beta_3 = mean(beta_3_pval < 0.10)
proportion_beta_3
```

```
## [1] 0.1012
```

Yes, this is what I would expect, since the low proportion shows that the β_3 predictor may not be significant to the response in the model with other predictors has linear relationship with the response, which corresponds the its real value 0, indicating no linear relationship between it and the response.

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

```
proportion_beta_5 = mean(beta_5_pval < 0.01)
proportion_beta_5
```

```
## [1] 0.7404
```

Yes, this is what I would expect, since the high proportion shows that the β_3 predictor is significant to the response in the model with other predictors has linear relationship with the response, which corresponds its real value 2, indicating linear relationship between it and the response.