Week 4 - Homework

STAT 420, Summer 2019, segneri3

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.

```r
library(readr)
nutrition = read_csv('nutrition-2018.csv')
```

```r
cal_model = lm(Calories ~ Fat + Sugar + Sodium, data = nutrition)
f = summary(cal_model)$fstatistic[1]
df1 = summary(cal_model)$fstatistic[2]
df2 = summary(cal_model)$fstatistic[3]
summary(cal_model)
```

```
##
## Call:
## lm(formula = Calories ~ Fat + Sugar + Sodium, data = nutrition)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -339.41  -64.82   -9.42   28.12  293.54
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.005e+02  1.409e+00  71.310  < 2e-16 ***
## Fat         8.483e+00  6.456e-02 131.394  < 2e-16 ***
## Sugar       3.901e+00  7.140e-02  54.627  < 2e-16 ***
## Sodium      6.165e-03  1.030e-03   5.983 2.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 80.85 on 5952 degrees of freedom
## Multiple R-squared:  0.7686, Adjusted R-squared:  0.7685
## F-statistic:  6591 on 3 and 5952 DF,  p-value: < 2.2e-16
```

The null hypothesis is $H_0 : \beta_1 = \beta_2 = ... = \beta_{p-1} = 0$. This describes a model where none of the predictors have a linear relationship with the response.

The alternative hypothesis is that at least one of the $beta_j$ from the null hypothesis is not zero. $H_1 :$ At least one of $\beta_j \neq 0, j = (1, 2, ..., (p-1)$

The value of the F-statistic is `6590.9402239`.

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

A statistical decision at $\alpha = 0.01$ would be to reject the null hypothesis. That is to say that the regression is significant, and at least one of the β_js is not zero.

A conclusion in the context of the problem would be that at least one of the predictors, `Fat`, `Sugar`, and `Sodium`, has a useful linear relationship with `Calories`.

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

```r
coef(cal_model)
```

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

The intercept is the Calorie value when the other values (`Fat`, `Sugar`, `Sodium`) are zero.

The `Fat` coefficient describes the change in Calories as the Fat content changes, assuming Sugar and Sodium are held constant.

The `Sugar` coefficient describes the change in Calories as the Sugar content changes, assuming Fat and Sodium are held constant.

The `Sodium` coefficent describes the change in Calories as the Sodium content changes, assuming Fat and Sugar are held constant.

(c) Use your model to predict the number of `Calories` in a Big Mac. According to McDonald's publicized nutrition facts, the Big Mac contains 28g of fat, 9g of sugar, and 950mg of sodium.

```
big_mac = data.frame(
  Fat = 28,
  Sugar = 9,
  Sodium = 950
)
predict(cal_model, newdata = big_mac)
```

```
##        1
## 378.9498
```

(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.

```
s_y = sd(nutrition$Calories)
s_e = summary(cal_model)$sigma
```

The standard deviation is `168.0499661`. This is the amount of variation in the Calories column in the nutrition data set.

The residual standard error is `80.8543023`. The lower this value is to zero, the better the fit is.

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

The R^2 value for the model is `0.7686281`. The closer this value is to one, the better the fit. This is a measure of how close the data are to the fitted regression. It shows how closely we can accurately fit the rows from the nutrition data set to their `Calorie` values, based on `Fat`, `Sugar`, and `Sodium`.

(f) Calculate a 95% confidence interval for β_2. Give an interpretation of the interval in the context of the problem.

```
confint(cal_model, level = .95)[3,]
```

```
##    2.5 %   97.5 %
## 3.760541 4.040494
```

Based on the data in the nutrition data set, the model is 95% confident that the mean of β_2 (the relationship between `Calories` and `Sugar`) is within the two values above.

(g) Calculate a 99% confidence interval for β_0. Give an interpretation of the interval in the context of the problem.

```
confint(cal_model, level = .99)[1,]
```

```
##     0.5 %    99.5 %
##  96.82624 104.08588
```

Based on the data in the nutrition data set, the model is 99% confident that the mean of β_0 (the value of `Calories` when the predictors are zero) is within the two values above.

(h) Use a 90% confidence interval to estimate the mean Calorie content of a food with 24g of fat, 0g of sugar, and 350mg of sodium, which is true of a large order of McDonald's french fries. Interpret the interval in context.

```
french_fries = data.frame(
  Fat = 24,
  Sugar = 0,
  Sodium = 350
)

p = predict(cal_model, newdata = french_fries, interval = 'confidence', level = .90)
p
```

```
##        fit      lwr      upr
## 1 306.2128 303.8033 308.6224
```

The model's mean Calorie content of McDonald's french fries is `306.2128307`. The model is 90% confident that the mean Calorie content is within the lower and upper bounds shown above.

(i) Use a 90% prediction interval to predict the Calorie content of a Taco Bell Crunchwrap Supreme that has 21g of fat, 6g of sugar, and 1200mg of sodium. Interpret the interval in context.

```
crunch_wrap = data.frame(
  Fat = 21,
  Sugar = 6,
  Sodium = 1200
)

p = predict(cal_model, newdata = crunch_wrap, interval = 'prediction', level = .90)
p
```

```
##        fit      lwr      upr
## 1 309.4065 176.3678 442.4452
```

The model is 90% confident that the Calorie count of a Taco Bell Crunchwrap Supreme will be found within the lower and upper bound shown above.

Exercise 2 (More `lm` for Multiple Regression)

For this exercise we will use the data stored in `goalies.csv`. It contains career data for 462 players in the National Hockey League who played goaltender at some point up to and including the 2014-2015 season. The variables in the dataset are:

- `W` - Wins
- `GA` - Goals Against
- `SA` - Shots Against
- `SV` - Saves
- `SV_PCT` - Save Percentage
- `GAA` - Goals Against Average
- `SO` - Shutouts
- `MIN` - Minutes
- `PIM` - Penalties in 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

(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

```
goalies = read_csv('goalies.csv')

model1 = lm(W ~ GA + SV, data = goalies)
model2 = lm(W ~ GA + SV + SA + MIN + SO, data = goalies)
model3 = lm(W ~ GA+SA+SV+SV_PCT+GAA+SO+MIN+PIM, data = goalies)

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

```
## Analysis of Variance Table
##
## Model 1: W ~ GA + SV
## Model 2: W ~ GA + SV + SA + MIN + SO
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    459 294757
## 2    456  72899  3    221858 462.59 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis is $H_0 : Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$. Basically the null hypothesis is that the first model is useful, and the alternative hypothesis is that the second model is more useful.

The value of the F-statistic is `462.5934999`.

The p-value is `6.8082472\times 10^{-138}`.

A statistical decision at $\alpha = 0.05$ is that we reject the null hypothesis. The second model is more useful that the first.

The second model is the one I prefer.

(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

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

```
## Analysis of Variance Table
##
## Model 1: W ~ GA + SV + SA + MIN + SO
## Model 2: W ~ GA + SA + SV + SV_PCT + GAA + SO + MIN + PIM
##   Res.Df   RSS Df Sum of Sq     F   Pr(>F)
## 1    456 72899
## 2    453 70994  3    1905.1 4.052 0.007353 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis is that model 2 is more useful than model 3.

The F-statistic is `4.0519676`.

The p-value is `0.0073529`.

A statistical decision at $\alpha = 0.05$ is that we reject the null hypothesis. The third model is more useful that the second.

The third model is the one I prefer.

(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 $\alpha = 0.05$

```
test_sv = summary(model3)$coefficients['SV',]
test_sv
```

```
##      Estimate    Std. Error       t value      Pr(>|t|)
## -0.0582151227  0.0150904765 -3.8577391869  0.0001310371
```

The T-statistic is `-3.8577392`.

The p-value is `0.0001310371`.

A statistical decision at $\alpha = 0.05$ is to reject the null hypothesis. `Saves` are useful in the model.

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(
  rep(1, nrow(Ozone)),
  Ozone$wind,
  Ozone$humidity,
  Ozone$temp
)

y = Ozone$ozone

beta_hat_no_lm = solve(t(X) %*% X) %*% t(X) %*% y

beta_hat_no_lm
```

```
##              [,1]
## [1,] -16.38178539
## [2,]  -0.18594444
## [3,]   0.08340014
## [4,]   0.38984294
```

```
sum(beta_hat_no_lm ^ 2)
```

```
## [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)`.

```
oz_model = lm(ozone ~ wind + humidity + temp, data = Ozone)

beta_hat_lm = coef(oz_model)

beta_hat_lm
```

```
##  (Intercept)         wind     humidity         temp 
## -16.38178539  -0.18594444   0.08340014   0.38984294
```

```
sum(beta_hat_lm ^ 2)
```

```
## [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()`.

```
beta_hat_lm = unname(beta_hat_lm)

all.equal(beta_hat_lm, as.vector(beta_hat_no_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()`.

```
n = nrow(Ozone)
p = 3
y_hat = X %*% beta_hat_no_lm
e = y - y_hat
sigma_no_lm = sqrt(t(e) %*% e / (n - p))
sigma_no_lm
```

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

```
sigma_lm = summary(oz_model)$sigma

all.equal(sigma_lm, as.vector(sigma_no_lm))
```

```
## [1] "Mean relative difference: 0.001467352"
```

(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_bar = mean(y)
SSR = sum((y_hat - y_bar) ^ 2)
SSTO = sum((y - y_bar) ^ 2)
```

```r
r_sqrd_no_lm = SSR/SSTO
r_sqrd_no_lm
```

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

```r
r_sqrd_lm = summary(oz_model)$r.squared

all.equal(r_sqrd_lm, as.vector(r_sqrd_no_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.)

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

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 `1`, 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 292 randomly chosen observations. The `auto_tst` data will contain the remaining observations. Hint: consider the following code:

```r
set.seed(1)
auto_trn_idx = sample(1:nrow(Auto), 292)

auto_trn = Auto[auto_trn_idx,]
auto_tst = Auto[-auto_trn_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.

```r
model1 = lm(mpg ~ cylinders+displacement+horsepower+
              weight+acceleration+year+origin, data = auto_trn)
model2 = lm(mpg ~ displacement, data = auto_trn)
model3 = lm(mpg ~ year, data = auto_trn)
model4 = lm(mpg ~ weight, data = auto_trn)
model5 = lm(mpg ~ weight + year + origin + displacement, data = auto_trn)
```

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
library(knitr)
library(kableExtra)

rnames = c(
  "All_Predictors",
  "Displacement",
  "Year",
  "Weight",
  "Weight+Year+Origin+Displacement"
)

train_RMSE = c(
  sqrt(mean(model1$residuals^2)),
  sqrt(mean(model2$residuals^2)),
  sqrt(mean(model3$residuals^2)),
  sqrt(mean(model4$residuals^2)),
  sqrt(mean(model5$residuals^2))
)

get_test_RMSE = function(test_data, model){
  y_i = test_data$mpg
  y_i_hat = predict(model, newdata = test_data)

  RMSE = sqrt((1/nrow(test_data) * sum((y_i - y_i_hat) ^ 2)))
  RMSE
}

test_RMSE = c(
  get_test_RMSE(auto_tst, model1),
  get_test_RMSE(auto_tst, model2),
  get_test_RMSE(auto_tst, model3),
  get_test_RMSE(auto_tst, model4),
  get_test_RMSE(auto_tst, model5)
)

results = data.frame(
  train_RMSE = train_RMSE,
  test_RMSE = test_RMSE
)
```

```r
row.names(results) = rnames

results %>%
  kable() %>%
  kable_styling()
```

	train_RMSE	test_RMSE
All_Predictors	3.302024	3.292404
Displacement	4.640660	4.573109
Year	6.307960	6.488993
Weight	4.314112	4.345098
Weight+Year+Origin+Displacement	3.346480	3.269263

My fifth model, the one with Weight, Year, Origin, and Displacement as the predictors is the best model. After looking through the summary information on the first model with every predictor, the p-values for each individual predictor indicated that these four were particularly useful. The first model is still a closer fit, but not by much.

The r-squared for the first model is `0.8212614`, and the r-squared for my fifth model is `0.8164161`.

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:

- $\beta_0 = 2$
- $\beta_1 = -0.75$
- $\beta_2 = 1.5$
- $\beta_3 = 0$
- $\beta_4 = 0$
- $\beta_5 = 2$
- $\sigma^2 = 25$

We will use samples of size `n = 42`.

We will verify the distribution of $\hat{\beta}_2$ 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 = 42
n = sample_size

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

The sum of the diagonal of `C` is `0.1792443`.

The 5th row of sim_data is:

```
sim_data[5,]
```

```
##   y x0        x1        x2        x3        x4         x5
## 5 0  1 0.7959582 0.4283331 0.6079313 0.4994189 -0.9018014
```

(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`.

```
num_sims = 2500

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

(c) Simulate 2500 samples of size `n = 42` 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`

```
beta_0 = 2
beta_1 = -0.75
```

```r
beta_2 = 1.5
beta_3 = 0
beta_4 = 0
beta_5 = 2
sigma = 5

for (i in 1:num_sims) {
  eps = rnorm(n, mean = 0 , sd = sigma)
  sim_data$y = beta_0*x0 + beta_1*x1 + beta_2*x2 + beta_3*x3 + beta_4*x4 + beta_5*x5 + eps
  fit = lm(y ~ x1+x2+x3+x4+x5, data = sim_data)

  beta_hat_1[i] = coef(fit)[2]
  beta_3_pval[i] = summary(fit)$coefficients[4,4]
  beta_5_pval[i] = summary(fit)$coefficients[6,4]
}
```

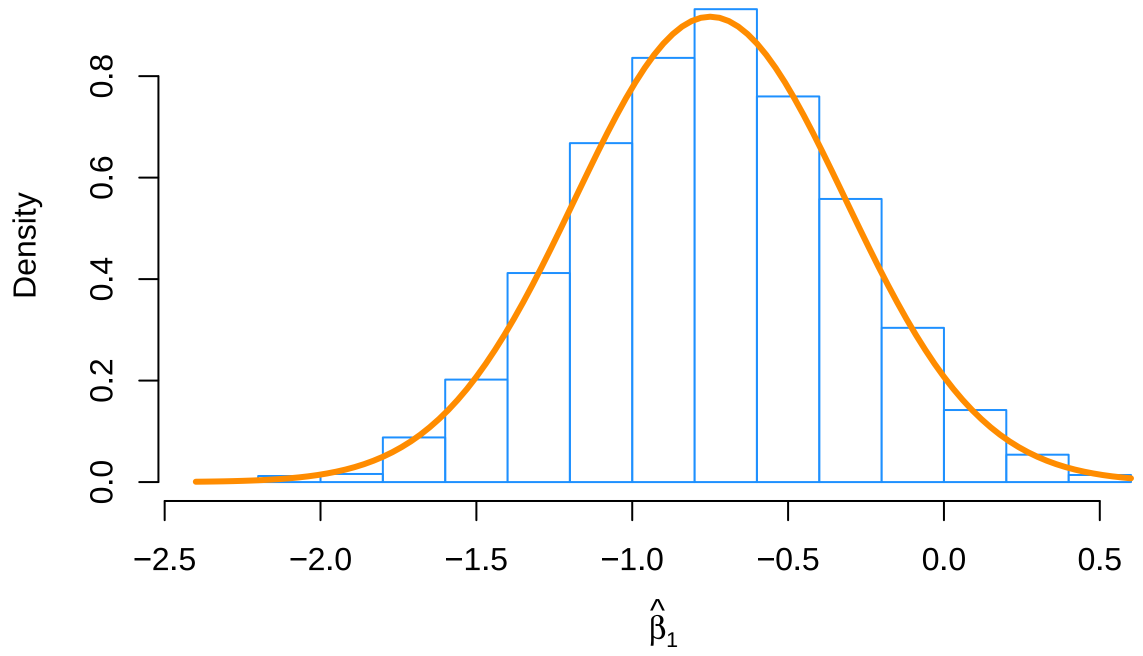
(d) Based on the known values of X, what is the true distribution of $\hat{\beta}_1$?

The true distribution is a normal distribution with mean `-0.75` and standard deviation `0.434983`.

(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?

The mean of `beta_hat_1` is `-0.7461209`, and the variance of `beta_hat_1` is `0.1853515`. They are close to what we would expect, as can be seen by the histogram below. The curve matches the histogram.

```r
hist(beta_hat_1, prob = TRUE, breaks = 20,
     xlab = expression(hat(beta)[1]), main = "", border = "dodgerblue")
curve(dnorm(x, mean = beta_1, sd = sqrt(sigma ^ 2 * C[1 + 1, 1 + 1])),
      col = "darkorange", add = TRUE, lwd = 3)
```



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

The proportion of p-values stored in `beta_3_pval` that are less than 0.10 is `0.096`. Considering that `beta_3` is `0`, I expected this value to be this low, as it indicates there is not a useful relationship with the third predictor.

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

The proportion of p-values stored in `beta_5_pval` that are less than 0.01 is `0.7956`. This indicates that in most of the simulations, the fifth predictor was useful. I expected this based on the value of `beta_5`, which is `2`.