Week 4 - Homework

STAT 420, Summer 2019, D. Unger

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:

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

```
##
## 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
```

- $H_0 : \beta_1 = \beta_2 = \beta_3 = 0$
- H_1 : At least one of $\beta_j \neq 0, j = 1, 2, 3$
- Test statistic: $F = 6590.9402239$
- P-value: 0. (although, not actually 0, but very, very small)
- Decision: **Reject** H_0 at $\alpha = 0.01$.
- Conclusion: There is a linear relationship between Calories and at least some of fat, sugar, and sodium.

Note that we used the broom package to obtain some results directly. (See the `.Rmd` file for details.) This is a useful package for "cleaning" some of the default `R` output.

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

Solution:

```
coef(nutrition_mod)
```

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

- $\hat{\beta}_0 = 100.4560567$ is the estimated Calories of a food with 0g fat, 0g sugar, and 0mg sodium.
- $\hat{\beta}_1 = 8.4832891$ is the estimated change in mean Calories for an increase of 1g fat for a food with a certain sugar and sodium content.
- $\hat{\beta}_2 = 3.9005172$ is the estimated change in mean Calories for an increase of 1g sugar for a food with a certain fat and sodium content.
- $\hat{\beta}_3 = 0.0061652$ is the estimated change in mean Calories for an increase of 1g sodium for a food with a certain fat and sugar content.

(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.

Solution:

```
big_mac = data.frame(Fat = 28, Sugar = 25, Sodium = 950)
predict(nutrition_mod, newdata = big_mac)
```

```
##        1
## 441.3581
```

(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(nutrition$Calories))
```

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

```
(s_e = glance(nutrition_mod)$sigma)
```

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

- $s_y = 168.0499661$ gives us an estimate of the variability of Calories, specifically how the observed Calorie data varies about its mean. (We could think of this as an estimate for how the observations vary in the model $Y_i = \beta_0 + \epsilon_i$.) This estimate does not use the predictors in any way.
- $s_e = 80.8543023$ gives us an estimate of the variability of the residuals of the model, specifically how the observed Calorie data varies about the fitted regression. This estimate does take into account the predictors.

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

Solution:

```
glance(nutrition_mod)$r.squared
```

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

$R^2 = 0.7686$ tells us that 76.86% of the observed variation in Calories is explained by a linear relationship with 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.

Solution:

```
confint(nutrition_mod, level = 0.95)
```

```
##                    2.5 %       97.5 %
## (Intercept) 97.694429707 1.032177e+02
## Fat          8.356720924 8.609857e+00
## Sugar        3.760540854 4.040494e+00
## Sodium       0.004145254 8.185238e-03
```

```
confint(nutrition_mod, level = 0.95)[3,]
```

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

We are 95% confident that the true change in mean Calories for an increase of 1g sugar is in this interval for particular values of fat and sodium.

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

Solution:

```
confint(nutrition_mod, level = 0.99)[1,]
```

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

We are 99% confident that the true mean Calories for a food with 0g of fat, sugar, and sodium is in this interval.

(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.

Solution:

```
large_fries = data.frame(Fat = 24, Sugar = 0, Sodium = 350)
predict(nutrition_mod, newdata = large_fries, interval = "confidence", level = 0.90)
```

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

We are 90% confident that the true **mean** Calories for foods with 24g of fat, 0g of sugar, and 350g of sodium is in this interval.

(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.

Solution:

```
crunchwrap = data.frame(Fat = 21, Sugar = 6, Sodium = 1200)
predict(nutrition_mod, newdata = crunchwrap, interval = "prediction", level = 0.90)
```

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

We are 90% confident that the Calories for a food item with 21g of fat, 6g of sugar, and 1200mg of sodium is in this interval.

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

```
# read in data
goalies_cleaned = read.csv("goalies.csv")

# fit requested models
fit_1 = lm(W ~ GA + SV, data = goalies_cleaned)
fit_2 = lm(W ~ GA + SV + SA + MIN + SO, data = goalies_cleaned)
fit_3 = lm(W ~ ., data = goalies_cleaned)
```

(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:

```
anova(fit_1, fit_2)
```

```
## 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
```

- $H_0 : \beta_{\texttt{SA}} = \beta_{\texttt{MIN}} = \beta_{\texttt{SO}} = 0$
- Test statistic: $F = 462.5934999$
- P-value: $6.8082472 \times 10^{-138}$
- Decision: **Reject** H_0 at $\alpha = 0.05$.
- Model Preference: The larger model, 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

Solution:

```
anova(fit_2, fit_3)
```

```
## 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
```

- $H_0 : \beta_{\texttt{SV PCT}} = \beta_{\texttt{GAA}} = \beta_{\texttt{PIM}} = 0$
- Test statistic: $F = 4.0519676$
- P-value: 0.0073529
- Decision: **Reject** H_0 at $\alpha = 0.05$.
- Model Preference: The larger model, 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 $\alpha = 0.05$

Solution:

```r
summary(fit_3)
```

```
##
## Call:
## lm(formula = W ~ ., data = goalies_cleaned)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.204  -3.126   0.935   2.835  64.078
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.2651619 16.8181423   0.313 0.754376
## GA          -0.1132805  0.0148085  -7.650 1.22e-13 ***
## SA           0.0516385  0.0135565   3.809 0.000159 ***
## SV          -0.0582151  0.0150905  -3.858 0.000131 ***
## SV_PCT      -8.0475191 17.6600154  -0.456 0.648830
## GAA         -0.0496006  0.4821957  -0.103 0.918116
## SO           0.4599359  0.1989567   2.312 0.021240 *
## MIN          0.0131790  0.0009504  13.867  < 2e-16 ***
## PIM          0.0468422  0.0136373   3.435 0.000647 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.52 on 453 degrees of freedom
## Multiple R-squared:  0.9858, Adjusted R-squared:  0.9856
## F-statistic:  3938 on 8 and 453 DF,  p-value: < 2.2e-16
```

- Test statistic: $t = 3.8091256$
- P-value: 1.5868871×10^{-4}
- Decision: **Reject** H_0 at $\alpha = 0.05$.

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.

```r
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:

```r
# setup response vector and design matrix
y = Ozone$ozone
n = length(y)
X = cbind(rep(1, n), as.matrix(subset(Ozone, select = -c(ozone))))

# solve for estimated coefficients
beta_hat_no_lm = solve(crossprod(X, X)) %*% t(X) %*% y

# coerce results to vector
beta_hat_no_lm = as.vector(beta_hat_no_lm)

# return requested results
beta_hat_no_lm
```

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

```r
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)`.

Solution:

```r
# fit model with lm() and extract coefficients
fit = lm(ozone ~ ., data = Ozone)
beta_hat_lm = as.vector(coef(fit))

# return requested results
beta_hat_lm
```

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

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

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

Notice the sum is the same, but we will complete one more step to verify equality.

(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:

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

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

Note that `all.equal()` allows for some minor differences. It only looks for "near equality." If we investigate further, we'd notice that in reality none of the estimates are the same.

```
beta_hat_no_lm == beta_hat_lm
```

```
## [1] FALSE FALSE FALSE FALSE
```

But why is this? While we do know the analytic solution

$$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top y,$$

it is not what `R` is using when calculating the regression coefficients using `lm()`. In reality, `R` is using a QR decomposition for speed and stability of the needed matrix operations. This creates very small numerical differences in the results.

(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:

```
# obtain the fitted values
y_hat = crossprod(t(X), beta_hat_no_lm)

# calculate s_e
s_e = sqrt(crossprod(y - y_hat, y - y_hat) / (length(y) - length(beta_hat_no_lm)))
s_e
```

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

```
# check that result matches lm()
all.equal(as.vector(s_e), summary(fit)$sigma)
```

```
## [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()`.

Solution:

```
# some intermediate calculations
sse = sum(crossprod(y - y_hat, y - y_hat))
sst = sum(crossprod(y - mean(y), y - mean(y)))

# calculate r2
r2 = 1 - sse / sst
r2
```

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

```
# check that result matches lm()
all.equal(as.vector(r2), summary(fit)$r.squared)
```

```
## [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))
```

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

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

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:

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

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:

```r
# split the data into train and test sets
set.seed(1)
auto_trn_idx = sample(1:nrow(Auto), 292) # randomly chosen observations for training
auto_trn = Auto[auto_trn_idx, ]
auto_tst = Auto[-auto_trn_idx, ]

# fit the five models
mod_1 = lm(mpg ~ displacement, data = auto_trn)
mod_2 = lm(mpg ~ acceleration + displacement + weight, data = auto_trn)
mod_3 = lm(mpg ~ acceleration + cylinders + displacement + weight + year,
           data = auto_trn)
mod_4 = lm(mpg ~ acceleration + cylinders + displacement + weight + year +
           origin, data = auto_trn)
mod_5 = lm(mpg ~ ., data = auto_trn)

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

# calculate all train errors
train_error = c(
  rmse(auto_trn$mpg, predict(mod_1, auto_trn)),
  rmse(auto_trn$mpg, predict(mod_2, auto_trn)),
  rmse(auto_trn$mpg, predict(mod_3, auto_trn)),
  rmse(auto_trn$mpg, predict(mod_4, auto_trn)),
  rmse(auto_trn$mpg, predict(mod_5, auto_trn))
)

# calculate all test errors
test_error = c(
  rmse(auto_tst$mpg, predict(mod_1, auto_tst)),
  rmse(auto_tst$mpg, predict(mod_2, auto_tst)),
  rmse(auto_tst$mpg, predict(mod_3, auto_tst)),
  rmse(auto_tst$mpg, predict(mod_4, auto_tst)),
  rmse(auto_tst$mpg, predict(mod_5, auto_tst))
)

auto_models = c("`mod_1`", "`mod_2`", "`mod_3`", "`mod_4`", "`mod_5`")
auto_results = data.frame(auto_models, train_error, test_error)
colnames(auto_results) = c("Model", "Train RMSE", "Test RMSE")
knitr::kable(auto_results)
```

Model	Train RMSE	Test RMSE
mod_1	4.675831	4.493195
mod_2	4.336099	4.048654
mod_3	3.406973	3.460830
mod_4	3.328084	3.289951
mod_5	3.322208	3.288330

Based on these results, we believe `mod_4` is the best model for predicting since it achieves the lowest **test** RMSE, **3.2899509**.

- `acceleration + cylinders + displacement + weight + year + origin`

First, note that the models chosen happen to be nested, but this is not necessary. However, it does illustrate that the train RMSE decreases as the size of the model increases.

Also note that the predictors for `mod_4` were chosen in a somewhat ad-hoc manner. Initial consideration was given to predictors from the full model that were significant. This is not a guaranteed method, but is a decent starting point when guessing and checking.

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
```

Solution:

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

```
sum(diag(C))
```

```
## [1] 0.1792443
```

```
sim_data[5, ]
```

```
##   y        x1        x2        x3        x4         x5
## 5 0 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`.

Solution:

```
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`

Solution:

```
beta_0 = 2
beta_1 = -0.75
beta_2 = 1.5
beta_3 = 0
beta_4 = 0
beta_5 = 2
```

```
sigma  = 5

library(broom)

for (i in 1:num_sims) {

  # simulate y data
  sim_data$y = with(sim_data,
                    beta_0 * x0 + beta_1 * x1 + beta_2 * x2 +
                    beta_3 * x3 + beta_4 * x4 + beta_5 * x5 +
                    rnorm(n = sample_size, mean = 0 , sd = sigma))

  # fit model to simulated data
  fit = lm(y ~ ., data = sim_data)

  # extract the three desired values
  beta_hat_1[i]  = coef(fit)[2]
  beta_3_pval[i] = tidy(fit)[4, ]$p.value
  beta_5_pval[i] = tidy(fit)[6, ]$p.value
}
```

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

Solution:

$$\hat{\beta}_1 \sim N\left(\beta_1, \sigma^2 C_{11}\right)$$

$$\hat{\beta}_1 \sim N\left(\mu = -0.75, \sigma^2 = 25 \times 0.0075684 = 0.1892102\right).$$

$$\hat{\beta}_1 \sim N\left(\mu = -0.75, \sigma^2 = 0.1892102\right).$$

(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:

```
mean(beta_hat_1)
```

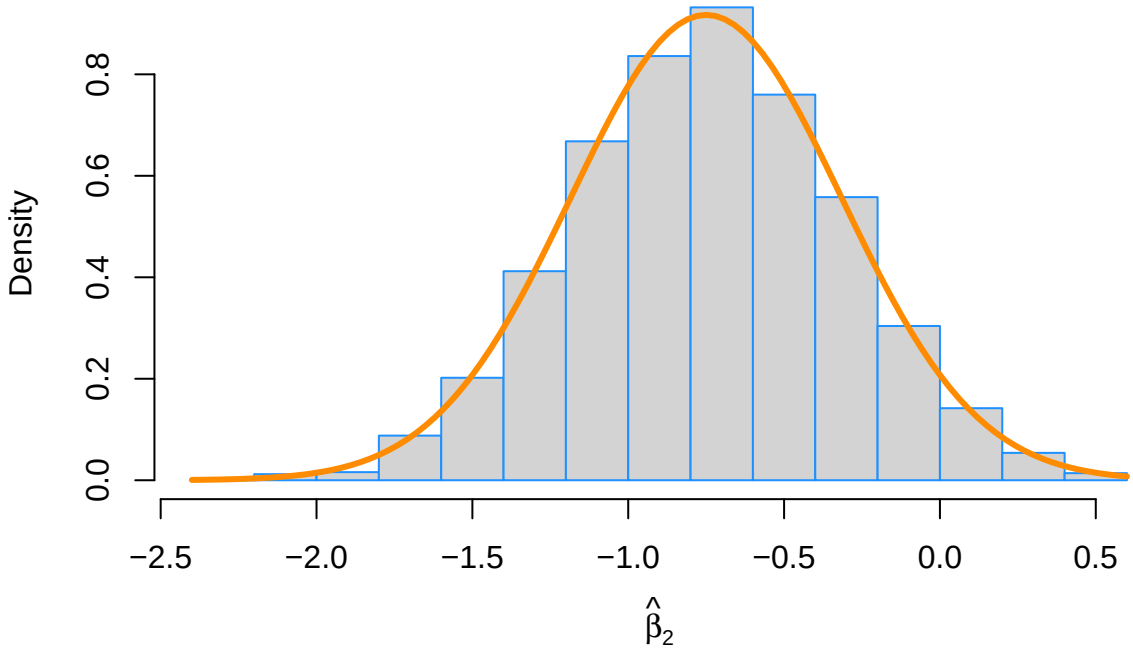
```
## [1] -0.7461209
```

```
var(beta_hat_1)
```

```
## [1] 0.1853515
```

The empirical results match what we would expect.

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



The true curve matches the histogram of simulated values well.

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

Solution:

```
mean(beta_3_pval < 0.10)
```

```
## [1] 0.096
```

Since $\beta_3 = 0$, we expect roughly 10% of the p-values to be significant at $\alpha = 0.10$ **by chance**, so this roughly matches our expectation.

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

Solution:

```
mean(beta_5_pval < 0.01)
```

```
## [1] 0.7956
```

Since $\beta_5 \neq 0$, we hope many of the p-values are significant at $\alpha = 0.01$, so this roughly matches our expectation.