Week 4 - Homework

STAT 420, Summer 2020, D. Unger

Directions

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

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

Exercise 1 (Using `lm`)

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

The variables in the dataset are:

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

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

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

Here,

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

- x_{i3} is `Sodium`.

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

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

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

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

```
##
## 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
```

$Ho : \beta_1 = \beta_2 = \beta_3 = 0, H1 : atleastoneof \beta_j \neq 0$ Value of F statistic is 6591

p-value of the test is 0. Very very small value.

Satistical decision at $\alpha = 0.01$ is to Reject Ho

Conclusion in the context of the problem is that there EXISTS a linear relationship between Calories and atleast some of fat,sugar and sodium.

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

```
coef(nutrition_lm)
```

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

$\hat{\beta}_0 = 100.4561$ is the estimated calories value of a food when there is 0gm fat, 0gm sugar and 0mg Sodium

$\hat{\beta}_1 = 848.3289$ is the estimated change in mean Calories value of a food when there is an increase of 1gm fat with certain Sugar and Sodium content in the food

$\hat{\beta}_2 = 390.0517$ is the estimated change in mean Calories value of a food when there is an increase of 1gm sugar with certain fat and Sodium in the food.

$\hat{\beta}_3$ - 0.006165246 is the estimated change in mean Calories value of a food when there is an increase of 1mg of Sodium with certain fat and sugare in the food

(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 30g of fat, 9g of sugar, and 1010mg of sodium.

```
big_mac_query = data.frame(Fat = 30, Sugar = 9, Sodium = 1010)
predict(nutrition_lm, newdata = big_mac_query)
```

```
##        1
## 396.2863
```

Nuber of Calories is 396.2863

(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))
```

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

```
(s_e = summary(nutrition_lm)$sigma)
```

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

$s_y = 168.05$ gives us an estimate of the variations in the Calories value around its Mean Value.

$s_e = 80.8543$ gives us an estimate of the variations in the residuals of the model, especially how the observed Calorie value vares from it's fitted value by considering the predictors value.

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

```
#names(summary(nutrition_lm))
summary(nutrition_lm)$r.squared
```

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

$R^2 = 0.7686281$ explains that 76.8% of the observed variations in Calories can be explained by a linear relationsip with fat, sugar and sodium in the food.

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

```
confint(nutrition_lm, level = 0.90)[3,]
```

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

We are 90% confident that the mean Calorie for an increase of 1 gm of Sugar is in the range of 3.783051 and 4.01798 for certain values of fat and sodium

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

```
confint(nutrition_lm, level = 0.95)[1,]
```

```
##     2.5 %    97.5 %
##  97.69443 103.21768
```

We are 95% confidence that the true mean Calorie for food with 0gm of fat, 0gm of sugar and 0mg of Sodium is in the range of 97.69443 and 103.21768

(h) Use a 99% confidence interval to estimate the mean Calorie content of a food with 23g of fat, 0g of sugar, and 400mg of sodium, which is true of a large order of McDonald's french fries. Interpret the interval in context.

```
large_fries_query = data.frame(Fat = 23, Sugar = 0, Sodium = 400)
predict(nutrition_lm, newdata = large_fries_query, interval = "confidence", level = 0.99)
```

```
##        fit      lwr      upr
## 1 298.0378 294.3532 301.7224
```

We are 99% confident that the true mean calorie value for food with 23g of fat, 0g of sugare and 400mg of Sodium is in the range of 294.3532 and 301.7224

(i) Use a 99% prediction interval to predict the Calorie content of a Crunchwrap Supreme, which has 21g of fat, 6g of sugar, and 1200mg of sodium according to Taco Bell's publicized nutrition information. Interpret the interval in context.

```
crunchwrapsupreme_query = data.frame(Fat = 21, Sugar = 6, Sodium = 1200)
predict(nutrition_lm, newdata = crunchwrapsupreme_query, interval = "prediction", level = .99)
```

```
##        fit      lwr      upr
## 1 309.4065 101.0345 517.7786
```

We are 99% confident that the true mean calorie value for food with 21g of fat, 6g of sugar and 1200mg of sodium is in the range of 101.0345 and 517.7786

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

```
goalies = read.csv("goalies.csv")
glm_1 = lm(W ~ GA + SV, data = goalies)
glm_2 = lm(W ~ GA + SV + SA + MIN + SO, data = goalies)
glm_3 = lm(W ~ ., data = goalies)
```

(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

```
anova(glm_1, glm_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
```

$Ho : \beta_{\texttt{SA}} = \beta_{\texttt{SV}} = \beta_{\texttt{SO}} = 0$

The value of the Test Statistics is F = 462.59

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

A statistical decision at $\alpha = 0.05$ is Reject Ho

Model Preference: 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

```
anova(glm_2, glm_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
```

$Ho : \beta_{\texttt{SVPCT}} = \beta_{\texttt{GAA}} = \beta_{\texttt{PIM}} = 0$

The value of the Test Statistics is F = 4.052

The p-value of the test is 0.007353

A statistical decision at $\alpha = 0.05$ is Reject Ho

Model Preference: 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(glm_3)
```

```
##
## Call:
## lm(formula = W ~ ., data = goalies)
##
## 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
```

The value of the test statistic is -3.858

The p-value of the test is 0.000131

A statistical decision at $\alpha = 0.05$ is Reject Ho

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)`.

```
n = nrow(Ozone)
X = cbind(rep(1,n), as.matrix(subset(Ozone, select = c("wind", "humidity", "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(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)`.

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

```
## [1] -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()`.

```
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()`.

```
#summary(ozone_lm)$sigma

y_hat = X %*% solve(t(X) %*% X) %*% t(X) %*% y
e = y - y_hat
(s_e = sqrt(t(e) %*% e/ (n - length(beta_hat_no_lm)))[1,])
```

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

```
all.equal(summary(ozone_lm)$sigma, s_e)
```

```
## [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()`.

```
sse = sum(t(y-y_hat) %*% (y-y_hat))
sst = sum(t(y-mean(y)) %*% (y-mean(y)))

(r2 = 1 - sse/sst)
```

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

```
all.equal(r2, summary(ozone_lm)$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)) #Error in eval(substitute(select), nl, parent.frame()) : object
#summary(Auto)
```

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 `11`, 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(11)
auto_trn_idx = sample(1:nrow(Auto), 292)
names(Auto)
```

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

set.seed(11)
auto_trn_idx = sample(1:nrow(Auto), 292)
auto_trn = Auto[auto_trn_idx, ]
auto_tst = Auto[-auto_trn_idx, ]

auto_1 = lm(mpg ~ ., data = auto_trn)
auto_2 = lm(mpg ~ displacement, data = auto_trn)
auto_3 = lm(mpg ~ cylinders + displacement + weight + year + origin, data = auto_trn)
auto_4 = lm(mpg ~ cylinders + displacement, data = auto_trn)
auto_5 = lm(mpg ~ displacement + horsepower + weight, data = auto_trn)

trn_res = c(rmse(auto_trn$mpg, predict(auto_1, auto_trn)),
            rmse(auto_trn$mpg, predict(auto_2, auto_trn)),
            rmse(auto_trn$mpg, predict(auto_3, auto_trn)),
            rmse(auto_trn$mpg, predict(auto_4, auto_trn)),
            rmse(auto_trn$mpg, predict(auto_5, auto_trn)))
```

```
## Warning in predict.lm(auto_1, auto_trn): prediction from a rank-deficient fit
## may be misleading
```

```r
#tst_res = c(rmse(auto_tst$mpg, predict(auto_1, auto_tst)),
#            rmse(auto_tst$mpg, predict(auto_2, auto_tst)),
#            rmse(auto_tst$mpg, predict(auto_3, auto_tst)),
#            rmse(auto_tst$mpg, predict(auto_4, auto_tst)),
#            rmse(auto_tst$mpg, predict(auto_5, auto_tst)))

#test_summary = data.frame(Model = c("auto_1", "auto_2", "auto_3", "auto_4", "auto_5"),
#                          Trained_RMSE =  trn_res,
#                          Tested_RMSE = tst_res)
#colnames(test_summary) = c("Train Model", "Train RMSE", "Test RMSE")

#library(knitr)
#kable(res_summary)
```

Based on the above results, auto_4 model is the best model for predicting, since it has lowest Test_RMSE value of 3.566250 with "cylinders + displacement + weight + year + origin" predictors.

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
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`.

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

for (i in 1: num_sims) {
  sim_data$y = 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)
  sim_model = lm(y ~ ., data = sim_data)

  beta_hat_1[i] = coef(sim_model)[2]
  beta_3_pval[i] = summary(sim_model)$coefficients["x3", "Pr(>|t|)"]
  beta_5_pval[i] = summary(sim_model)$coefficients["x5", "Pr(>|t|)"]
}
```

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

(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?

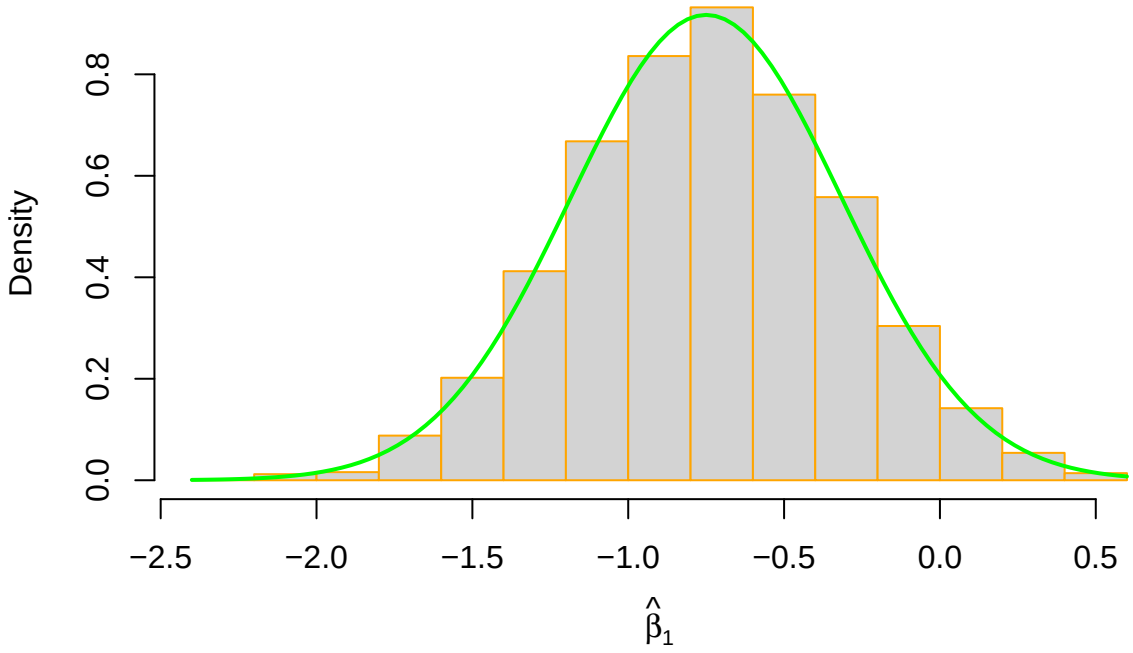
```
mean(beta_hat_1)
```

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

```
var(beta_hat_1)
```

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

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



The results are close to our expectations.

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?

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

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

Yes. This matches our expectaions, since $\beta_3 \neq 0$ meaning that 9.6% of the p-values are significant at alpha = 0.10

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

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

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

Yes. This matches our expectations, since $\beta_3 \neq 0$.