Week 4 - Homework

STAT 420, Summer 2019, Sushanta Panda

- Directions
  - Exercise 1 (Using `lm`)
  - Exercise 2 (More `lm` for Multiple Regression)
  - Exercise 3 (Regression without `lm`)
  - Exercise 4 (Regression for Prediction)
  - Exercise 5 (Simulating Multiple Regression)

Directions

- 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` (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`.

```
nut_data = read.csv("nutrition-2018.csv")
null_model = lm(Calories~1,data=nut_data)
full_model = lm(Calories~Fat+Sugar+Sodium,data=nut_data)
summary(full_model)
```

```
##
## Call:
## lm(formula = Calories ~ Fat + Sugar + Sodium, data = nut_data)
##
## 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
```

```
anova(null_model,full_model)
```

```
## Analysis of Variance Table
##
## Model 1: Calories ~ 1
## Model 2: Calories ~ Fat + Sugar + Sodium
##   Res.Df       RSS Df Sum of Sq      F    Pr(>F)
## 1   5955 168173911
## 2   5952  38910713  3 129263198 6590.9 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

- The null and alternative hypotheses
  - **H0**: All Parameter are zero, $\beta_1 = \beta_2 = \beta_3 = 0$, **Null Model:** $Y_i = \beta_0 + \epsilon$
  - **H1**: Atleast of the parameter is non zero, $\beta_1 \neq 0$ or $\beta_2 \neq 0$ or $\beta_3 \neq 0$, **Full Model:** $Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \epsilon$
- The value of the test statistic which is F Statistics is **6590.9402239**

- The p-value of the test (F Test) is 0 (The actual value is **2.2e-16**, however R is unable to print this value)
- A statistical decision at $\alpha = 0.01$: The **p value** for the F Test is **2.2e-16** which is very very small than that of alpha (0.01), hence we **reject the Null Hypothesis**
- A conclusion in the context of the problem : Since we **reject the null hypothesis**, the regression is significant, and atleast one of the predictor (Fat or Sugar or Sodium) is usefull

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

```
summary(full_model)$coefficient[,"Estimate"]
```

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

The estimated regression coefficient are β_0 **:** 100.4560567, β_1 **:** 8.4832891, β_2 **:** 3.9005172, β_3 **:** 0.0061652

Interpretation:

- β_0 **:** The mean calories is **100.4560567** when there is 0 g of Fat, 0 g of Sugar and 0 g of Sodium in the food item
- β_1 **:** The mean change of calories is **8.4832891**, when there is increase fat level by 1 g for a particular value of sugar and sodium in the food item
- β_2 **:** The mean change of calories is **3.9005172**, when there is increase sugar level by 1 g for a partifular value of fat and sodium in the food item
- β_3 **:** The mean change of calories is **0.0061652**, when there is increase sodium level by 1 g for a particular valie of Fat and Sugar in the food item

(c) Use your model to predict the number of `Calories` in a Big Mac. According to McDonald's publicized nutrition facts (https://www.mcdonalds.com/us/en-us/about-our-food/nutrition-calculator.html), the Big Mac contains 28g of fat, 9g of sugar, and 950mg of sodium.

```
newdata_bigmac = data.frame(Fat=c(28),Sugar=c(9),Sodium=c(950))
predict(full_model,newdata=newdata_bigmac)
```

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

Predicted **`Calories`** in a Big Mac is **378.9497894**

(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.

```
sd_y_hat = sd(nut_data$Calories)
se = summary(full_model)$sigma
```

Standard Deviation s_y: **168.0499661** and Standard Error s_e: **80.8543023**

Interpreting:

- s_y: The average distance between all the observed calories data from it's mean is **168.0499661** g
- s_e: The average distance between all observed calories data and the predicted calories from the model is **80.8543023** g

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

```
summary(full_model)$r.squared
```

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

The R_2 for the model is **0.7686281**

Intrepreting: The proportion of the calories variable which is explained by the linear relationship of the fat, sugar and sodium predictor varible is **0.7686281**

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

```
confint(full_model,interval="confident",level=0.95)["Sugar",]
```

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

Interpreting: We are 95% confident the mean change of Calories for an increase in Sugar of 1 g is in between **3.7605409** and **4.0404935**

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

```
confint(full_model,level=0.99)["(Intercept)",]
```

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

Interpreation We are 99% confident that mean Calories for 0g of Fat, 0g of Sugar and 0g of Sodium is in between **96.8262377** and **104.0858756**

(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.

```
newdata_large_frenchfries = data.frame(Fat = c(24), Sugar = c(0), Sodium = c(350))
predict(full_model,newdata=newdata_large_frenchfries,interval="confidence",level=0
.90)
```

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

Interpretation: The mean response of Large McDonald's french fries (4g of fat, 0g of sugar, and 350mg of sodium,) is in between **303.8032608** and **308.6224006**

(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.

```
newdata_tachobell_Crunchwrap = data.frame(Fat = c(21), Sugar = c(6), Sodium = c(12
00))
predict(full_model,newdata=newdata_tachobell_Crunchwrap,interval="prediction",leve
l=0.90)
```

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

Interpretation: We are 90% predicted the new observation of Calories for the Taco Bell Crunchwrap Supreme (21g of fat, 6g of sugar, and 1200mg of sodium) is in between **176.3678466** and **442.4452051**

Exercise 2 (More `lm` for Multiple Regression)

For this exercise we will use the data stored in `goalies.csv` (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")
null_model = lm(W~GA+SV,data=goalies)
full_model = lm(W~GA+SV+SA+MIN+SO,data=goalies)
anova(null_model,full_model)
```

```
## 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
```

(a) Use an F-test to compares Models 1 and 2. Report the following:

- The null hypothesis
  - **H0**: All Parameter are zero, $\beta_3 = \beta_4 = \beta_5 = 0$, **Null Model:** $Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon$
- The value of the test statistic which is F Statistics is: **462.5934999**
- The p-value of the test (F Test) is 6.808247210^{-138}
- A statistical decision at $\alpha = 0.05$: Since the P Value of the F statsitics (from F Test) is extremly low **6.808247210^{-138}** < **0.05 (α)**, we **reject the Null Hypothesis**, means atleast one of the predictor "Shots Against" or "Minutes" or "Shutouts" has an impact and usefull on the response variable
- The model you prefer : Since we reject the null hypothesis, we prefer the full model, i.e. **Model 2**

(b) Use an F-test to compare Model 3 to your preferred model from part **(a)**. Report the following:

```
null_model = lm(W~GA+SV+SA+MIN+SO,data=goalies)
full_model = lm(W~.,data=goalies)
anova(null_model,full_model)
```

```
## 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
  - **H0**: All of the parameters are zero, $\beta_6 = \beta_7 = \beta_8 = 0$, **Null Model:** $Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \beta_5 x_{i5} + \epsilon$
- The value of the test statistic which is F Statistics is: **4.0519676**
- The p-value of the test statistics (F statostics) for the F Test is **0.0073529**
- A statistical decision at $\alpha = 0.05$: Since the P Value of the F statistics for the F Test is **0.0073529** < **0.05 (α)**, we **reject the null hypothesis**, means atleast one of the predictor "Save Percentage", "Goals Against Average" and "Penalities in Minutes" has significance and usefull on the response variable ("Wins")

- The model you prefer : Since we **reject** the null hypothesis, we prefer the **Model 3**

(c) Use a t-test to test $H_0 : \beta_{\text{SV}} = 0$ vs $H_1 : \beta_{\text{SV}} \neq 0$ for the model you preferred in part **(b)**. Report the following:

```
model = lm(W~.,data=goalies)
summary(model)$coefficient
```

```
##                Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)  5.26516186 1.681814e+01  0.3130644 7.543758e-01
## GA          -0.11328049 1.480846e-02 -7.6497158 1.220540e-13
## SA           0.05163855 1.355654e-02  3.8091256 1.586887e-04
## SV          -0.05821512 1.509048e-02 -3.8577392 1.310371e-04
## SV_PCT      -8.04751912 1.766002e+01 -0.4556915 6.488302e-01
## GAA         -0.04960055 4.821957e-01 -0.1028640 9.181165e-01
## SO           0.45993589 1.989567e-01  2.3117387 2.123986e-02
## MIN          0.01317900 9.503583e-04 13.8674046 1.068552e-36
## PIM          0.04684216 1.363733e-02  3.4348486 6.474527e-04
```

- The value of the test statistic(t statistics) for the t-test is : **-3.8577392**
- The p-value of the test statistics for the t test is: **1.310370710^{-4}**

The P value of test: **6.880740410^{-7}**

- A statistical decision at $\alpha = 0.05$ Since the P value is **6.880740410^{-7}** < of $\alpha = 0.05$, we **reject the null hypothesis**, means the coefficient of regression of “Saves” can’t be zero and has some usefullness with the response (“Wins”) variable

---

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$temp)
beta_hat_no_lm = solve(t(X) %*% X) %*% t(X) %*% Ozone$ozone
beta_hat_no_lm=as.vector(beta_hat_no_lm)
c(beta_hat_no_lm=beta_hat_no_lm,sum_beta_hat_no_lm=sum(beta_hat_no_lm ^ 2))
```

```
##    beta_hat_no_lm1    beta_hat_no_lm2    beta_hat_no_lm3
##       -16.38178539        -0.18594444         0.08340014
##    beta_hat_no_lm4 sum_beta_hat_no_lm
##         0.38984294       268.55640104
```

Value of **beta_hat_no_lm: -16.3817854, -0.1859444, 0.0834001, 0.3898429**

Value of **sum(beta_hat_no_lm ^ 2): 268.556401**

(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_model = lm(ozone~.,data=Ozone)
beta_hat_lm = summary(Ozone_lm_model)$coefficient[,"Estimate"]
beta_hat_lm=as.vector(beta_hat_lm)
c(beta_hat_lm=beta_hat_lm,sum_beta_hat_lm=sum(beta_hat_lm ^ 2))
```

```
##    beta_hat_lm1    beta_hat_lm2    beta_hat_lm3    beta_hat_lm4
##    -16.38178539     -0.18594444      0.08340014      0.38984294
## sum_beta_hat_lm
##    268.55640104
```

Value of **beta_hat_lm: -16.3817854, -0.1859444, 0.0834001, 0.3898429**

Value of **sum(beta_hat_lm ^ 2): 268.556401**

(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
```

After compare via the all.equal between **beta_hat_no_lm & beta_hat_lm**, the results coming as: `all.equal(beta_hat_no_lm,beta_hat_lm)`

(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
s_e_2 = sum((y_hat - Ozone$ozone) ^ 2) / (nrow(Ozone)-4)
s_e = sqrt(s_e_2)
s_e
```

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

s_e without the use of `lm()` **4.8061147**

(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()`.

```
SSReg = sum((y_hat - mean(Ozone$ozone))^2)
SSTot = sum((Ozone$ozone - mean(Ozone$ozone))^2)
R_2 = SSReg/SSTot
R_2
```

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

The R^2 without the use of `lm()` **0.6398887**

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.

```
n_train = length(auto_trn_idx)
n_test = nrow(Auto) - length(auto_trn_idx)

model1 = lm(mpg~.,data=Auto[auto_trn_idx,])
model2 = lm(mpg~displacement,data=Auto[auto_trn_idx,])
model3 = lm(mpg~cylinders+horsepower,data=Auto[auto_trn_idx,])
model4 = lm(mpg~weight+acceleration,data=Auto[auto_trn_idx,])
model5 = lm(mpg~cylinders+horsepower+weight+acceleration,data=Auto[auto_trn_idx,])
```

```
train_rmse_model1 = sum((Auto[auto_trn_idx,]$mpg - predict(model1))^2) / n_train
train_rmse_model2 = sum((Auto[auto_trn_idx,]$mpg - predict(model2))^2) / n_train
train_rmse_model3 = sum((Auto[auto_trn_idx,]$mpg - predict(model3))^2) / n_train
train_rmse_model4 = sum((Auto[auto_trn_idx,]$mpg - predict(model4))^2) / n_train
train_rmse_model5 = sum((Auto[auto_trn_idx,]$mpg - predict(model5))^2) / n_train

newdata_model1 = subset(Auto[-auto_trn_idx,],select=c("cylinders","displacement","
horsepower","weight","acceleration","year","origin"))
newdata_model2 = subset(Auto[-auto_trn_idx,],select=c("displacement"))
newdata_model3 = subset(Auto[-auto_trn_idx,],select=c("cylinders","horsepower"))
newdata_model4 = subset(Auto[-auto_trn_idx,],select=c("weight","acceleration"))
newdata_model5 = subset(Auto[-auto_trn_idx,],select=c("cylinders","horsepower","we
ight","acceleration"))

test_rmse_model1 = sum((Auto[-auto_trn_idx,]$mpg - predict(model1,newdata=newdata_
model1))^2) / n_test
test_rmse_model2 = sum((Auto[-auto_trn_idx,]$mpg - predict(model2,newdata=newdata_
model2))^2) / n_test
test_rmse_model3 = sum((Auto[-auto_trn_idx,]$mpg - predict(model3,newdata=newdata_
model3))^2) / n_test
test_rmse_model4 = sum((Auto[-auto_trn_idx,]$mpg - predict(model4,newdata=newdata_
model4))^2) / n_test
test_rmse_model5 = sum((Auto[-auto_trn_idx,]$mpg - predict(model5,newdata=newdata_
model5))^2) / n_test
```

```r
library(knitr)
prediction_report = data.frame(
  ModelName = c("Model 1","Model 2","Model 3","Model 4","Model 5"),
  ParameterName = c("All","displacement","cylinders+horsepower","weight+accelerati
on","cylinders+horsepower+weight+acceleration"),
  Train_Rmse = c(train_rmse_model1,train_rmse_model2,train_rmse_model3,train_rmse_
model4,train_rmse_model5),
  Test_Rmse = c(test_rmse_model1,test_rmse_model2,test_rmse_model3,test_rmse_model
4,test_rmse_model5)
)

kable(prediction_report, format = "pandoc",padding = 2,caption = "Summary of Predi
ction Results")
```

Summary of Prediction Results

ModelName	ParameterName	Train_Rmse	Test_Rmse
Model 1	All	10.90336	10.83992
Model 2	displacement	21.53572	20.91332
Model 3	cylinders+horsepower	20.83070	20.97244
Model 4	weight+acceleration	18.20859	18.36692
Model 5	cylinders+horsepower+weight+acceleration	17.70513	17.94857

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

beta_0 = 2
beta_1 = -0.75
beta_2 = 1.5
beta_3 = 0
beta_4 = 0
beta_5 = 2
sigma_2 = 25

x0 = rep(1,length=sample_size)
x1 = rnorm(sample_size,mean=0,sd=2)
x2 = sample(seq(0,4,length=sample_size))
x3 = rnorm(sample_size,mean=0,sd=1)
x4 = sample(seq(-2,2,length=sample_size))
x5 = rnorm(sample_size,mean=0,sd=2)
X = cbind(x0,x1,x2,x3,x4,x5)
C = solve(t(X) %*% X)
y = rep(0,length=sample_size)
sim_data = data.frame(predictor = c(X),response=y)
sim_data = data.frame(X,y)

sum_diag = 0
for(j in 1:6){
  sum_diag = sum_diag + C[j,j]
}
```

Sum of the diagonal of C is **0.1881355**

Sum of 5th row of sim_data **2.1362345**

(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 = 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`

```
for(l in 1:2500){
  epsilon = rnorm(sample_size,mean=0,sd=sqrt(sigma_2))
  beta_hat = rbind(beta_0,beta_1,beta_2,beta_3,beta_4,beta_5)
  sim_data[,"y"] = X %*% beta_hat + epsilon
  sim_model = lm(y~x1+x2+x3+x4+x5,data=sim_data)
  beta_hat_1[l] = summary(sim_model)$coefficient[2,1]
  beta_3_pval[l] = summary(sim_model)$coefficient[4,4]
  beta_5_pval[l] = summary(sim_model)$coefficient[6,4]
}
```

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

```
var_beta_1_hat = sqrt(sigma_2 * C[1+1,1+1])
```

The true distribution of β_1 : **mean**: **-0.75** and **varience**: **0.429738**

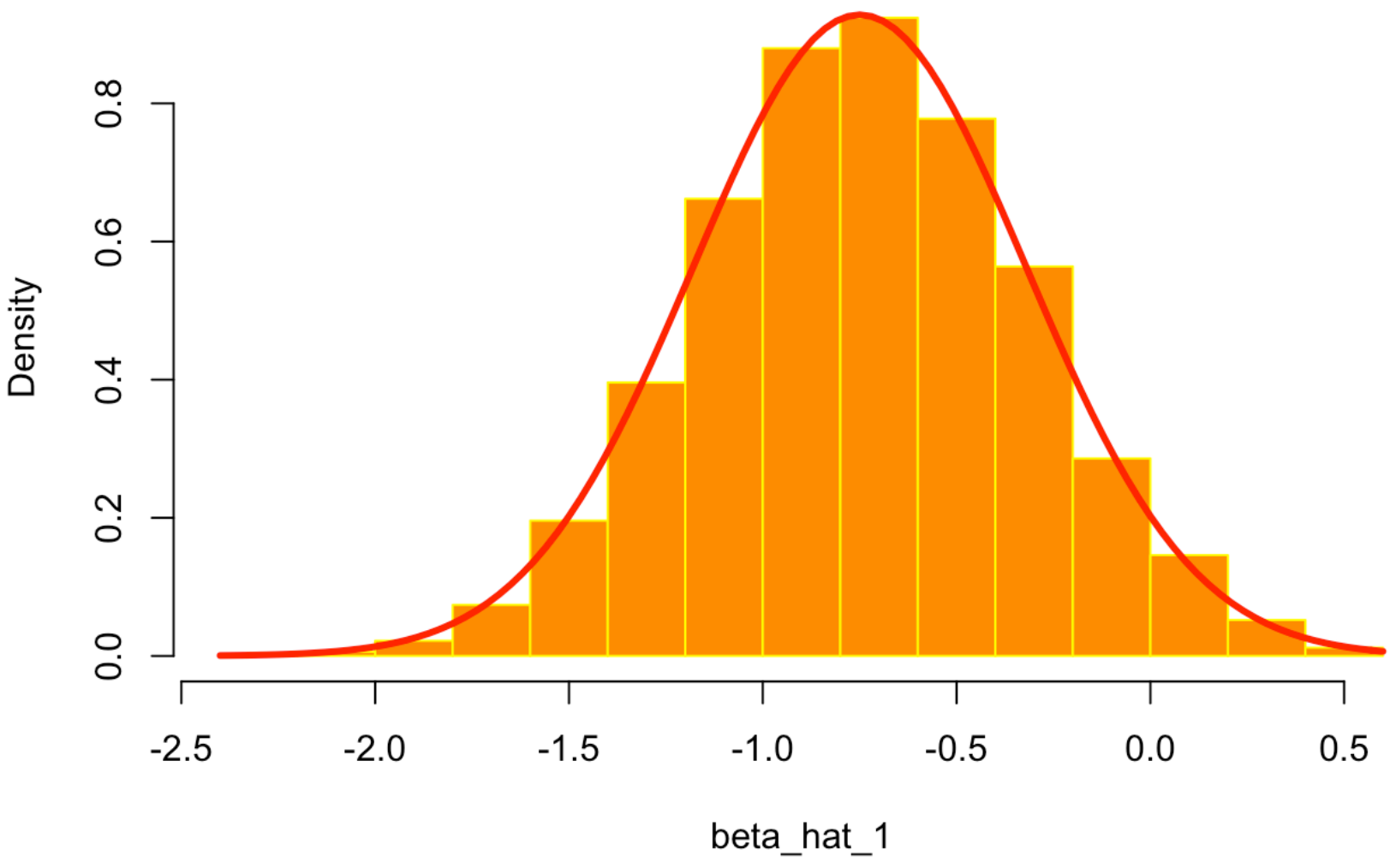
(e) Calculate the mean and variance of `beta_hat_1`. Are they close to what we would expect? Plot a histogram of `beta_hat_1`. Add a curve for the true distribution of $\hat{\beta}_1$. Does the curve seem to match the histogram?

```
mean_beta_hat_1 = mean(beta_hat_1)
var_beta_hat_1 = var(beta_hat_1)
```

The mean of **`beta_hat_1`** is **-0.7445587** which is close to the true **`beta_1`** which is **-0.75** and would expect

The variance of **`beta_hat_1`** is **0.1808** somehow not very close to the true variance of **`beta_1`** which is **0.1846747**

```
hist(beta_hat_1,
     col="darkorange",
     border="yellow",
     breaks=12,
     probability = TRUE
)
curve(dnorm(x,mean=beta_1,sd=sqrt(sigma_2*C[2,2])),add = TRUE,lwd=3,col="red")
```



The True curve is **matching** with 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] 0.094
```

0.094 proportion of the p-values stored in the `beta_3_pval` is less than 0.10

We are **expecting** this value, because the true value of β_3 is **0**, so in order to expect $\beta_3 = 0$ (true value), we must **failed to reject null hypothesis** (i.e $H0 : \beta_3 = 0$), where the p value must be > 0.10 (α). Since the **proportion of the beta_3_pval is lower below 0.1** (which is 0.094) or would be **higher over 0.1** , we can **failed to reject the null hypothesis**, which states $\beta_3 = 0$

(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] 0.792
```

0.792 proportion of the p-values stored in the `beta_5_pval` is less than 0.01

We are **expecting** this value, because the true value of β_5 is **2**, so in order to expect β_5 = 2 (true value), we must **reject null hypothesis** (i.e $H0 : \beta_5$ = 0), where the p value must be < 0.01 (α). Since the **proportion of the beta_5_pval is higher below 0.01** (which is 0.792), we can **reject the null hypothesis**, which states $\beta_5 \neq 0$