Week 2 - Homework

Haixu Leng (netID: haixul2)

05/29/2021

Exercise 1 (Using `lm`)

For this exercise we will use the `cats` dataset from the `MASS` package. You should use `?cats` to learn about the background of this dataset.

(a) Suppose we would like to understand the size of a cat's heart based on the body weight of a cat. Fit a simple linear model in `R` that accomplishes this task. Store the results in a variable called `cat_model`. Output the result of calling `summary()` on `cat_model`.

Solution:

```
library(MASS)
cat_model = lm(Hwt ~ Bwt, data = cats)
summary(cat_model)
```

```
##
## Call:
## lm(formula = Hwt ~ Bwt, data = cats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5694 -0.9634 -0.0921  1.0426  5.1238
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.3567     0.6923  -0.515    0.607
## Bwt           4.0341     0.2503  16.119   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.452 on 142 degrees of freedom
## Multiple R-squared:  0.6466, Adjusted R-squared:  0.6441
## F-statistic: 259.8 on 1 and 142 DF,  p-value: < 2.2e-16
```

(b) Output only the estimated regression coefficients. Interpret $\hat{\beta}_0$ and β_1 in the *context of the problem.* Be aware that only one of those is an estimate.

Solution:

The intercept of -0.3566624 means that when the cat weighs 0 kg, its heart weighs -0.3566624 g. For the slope, it means that With the body weight of the cat increases by 1 kg, the heart weight of the cat increases by 4.0340627 g.

```
coefficients(cat_model)
```

```
## (Intercept)         Bwt
##  -0.3566624   4.0340627
```

(c) Use your model to predict the heart weight of a cat that weights **3.1** kg. Do you feel confident in this prediction? Briefly explain.

Solution:

```
x = 3.1
y_hat = coefficients(cat_model)[[1]] + x * coefficients(cat_model)[[2]]
```

The predicted heart weight is 12.1489319 g. **Yes**, I am confident with the prediction, because we have data with similar body weights. By looking at the scatter plot (section e), the data points around 3.1 kg body weight have a heart weight around 12 g. This confirms my prediction.

(d) Use your model to predict the heart weight of a cat that weights **1.5** kg. Do you feel confident in this prediction? Briefly explain.

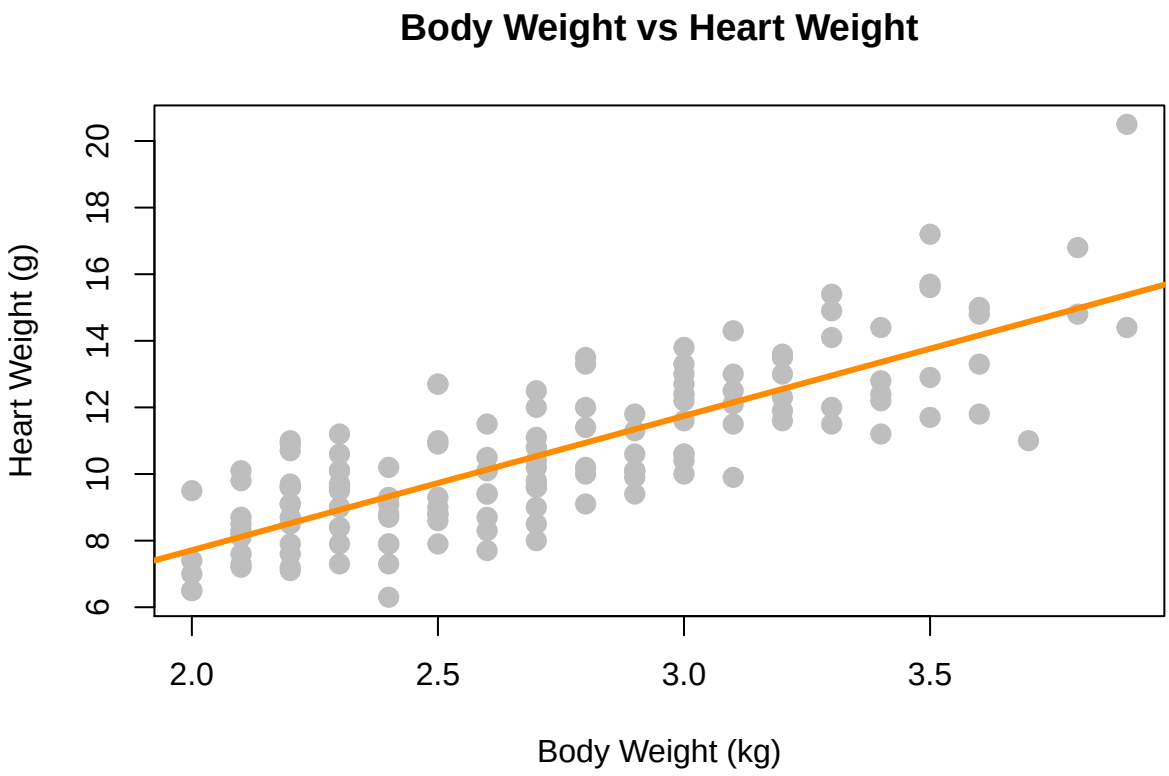
Solution:

```
x = 1.5
y_hat = coefficients(cat_model)[[1]] + x * coefficients(cat_model)[[2]]
```

The predicted heart weight is 5.6944316 g. **No**, I am not confident with the prediction. Because the data that I am given only covers body weight from around 2 kg to 3.9 kg. Without enough data, I don't know whether we can extrapolate our model (prediction) to cats with body weights less than 2 kg.

(e) Create a scatterplot of the data and add the fitted regression line. Make sure your plot is well labeled and is somewhat visually appealing.

```
plot(Hwt ~ Bwt, data = cats,
     xlab = "Body Weight (kg)",
     ylab = "Heart Weight (g)",
     main = "Body Weight vs Heart Weight",
     pch = 20,
     cex = 2,
     col = "grey")
abline(cat_model, lwd = 3, col = "darkorange")
```



(f) Report the value of R^2 for the model. Do so directly. Do not simply copy and paste the value from the full output in the console after running `summary()` in part **(a)**.

Solution: $R^2 = \frac{SSReg}{SST} = \frac{\sum_{i=1}^{n}(\hat{y}-\bar{y})^2}{\sum_{i=1}^{n}(y_i-\hat{y})^2}$

```
x_i = cats$Bwt
y_i = cats$Hwt
y_mean = mean(y_i)
y_hat = coefficients(cat_model)[[1]] + x_i * coefficients(cat_model)[[2]]
SSReg = sum((y_hat - y_mean)^2)
SST = sum((y_i - y_mean)^2)
R_2 = SSReg / SST
```

The answer of R^2 is 0.6466209

Exercise 2 (Writing Functions)

This exercise is a continuation of Exercise 1.

(a) Write a function called `get_sd_est` that calculates an estimate of σ in one of two ways depending on input to the function. The function should take three arguments as input:

- `fitted_vals` - A vector of fitted values from a model
- `actual_vals` - A vector of the true values of the response
- `mle` - A logical (`TRUE` / `FALSE`) variable which defaults to `FALSE`

The function should return a single value:

- s_e if `mle` is set to `FALSE`.
- $\hat{\sigma}$ if `mle` is set to `TRUE`.

Solution:

```
get_sd_set = function(fitted_vals, actual_vals, mle){
  n = length(fitted_vals)
  SSReg = sum((fitted_vals - actual_vals)^2)
  # divided by n for mle, and n-2 for non-mle case
  return (ifelse(mle, sqrt(SSReg / n), sqrt(SSReg / (n -2))))
}
```

(b) Run the function `get_sd_est` on the residuals from the model in Exercise 1, with `mle` set to `FALSE`. Explain the resulting estimate in the context of the model.

Solution:

```
get_sd_set(fitted_vals = y_hat, actual_vals = y_i, mle = FALSE)
```

```
## [1] 1.452373
```

We assume that $y_i = \hat{y} + \epsilon$, where $\epsilon \sim N(0, \sigma)$. The residuals standard deviation σ means that the error/residual follows a normal distribution with a mean 0 and a standard deviation as σ. A larger σ indicates a worse prediction.

(c) Run the function `get_sd_est` on the residuals from the model in Exercise 1, with `mle` set to `TRUE`. Explain the resulting estimate in the context of the model. Note that we are trying to estimate the same parameter as in part **(b)**.

Solution:

```
get_sd_set(fitted_vals = y_hat, actual_vals = y_i, mle = TRUE)
```

```
## [1] 1.442252
```

We assume that $y_i = \hat{y} + \epsilon$, where $\epsilon \sim N(0, \sigma)$. The residuals standard deviation σ means that the error/residual follows a normal distribution with a mean 0 and a standard deviation as σ. The difference in MLE is that n is in the denominator instead of $n - 2$ in least square estimations. A larger σ indicates a worse prediction.

(d) To check your work, output `summary(cat_model)$sigma`. It should match at least one of **(b)** or **(c)**.

Solution: The result is equal to the output where `mle = FALSE`.

```
summary(cat_model)$sigma
```

```
## [1] 1.452373
```

Exercise 3 (Simulating SLR)

Consider the model

$$Y_i = 5 + -3x_i + \epsilon_i$$

with

$$\epsilon_i \sim N(\mu = 0, \sigma^2 = 10.24)$$

where $\beta_0 = 5$ and $\beta_1 = -3$.

This exercise relies heavily on generating random observations. To make this reproducible we will set a seed for the randomization. Alter the following code to make `birthday` store your birthday in the format: `yyyymmdd`. For example, William Gosset, better known as *Student*, was born on June 13, 1876, so he would use:

```
birthday = 18760613
set.seed(birthday)
```

(a) Use `R` to simulate `n = 25` observations from the above model. For the remainder of this exercise, use the following "known" values of x.

```
x = runif(n = 25, 0, 10)
```

You may use the `sim_slr` function provided in the text. Store the data frame this function returns in a variable of your choice. Note that this function calls y `response` and x `predictor`.

Solution: Prepare x and y for this question.

```
birthday = 19900210
set.seed(birthday)
x = runif(n = 25, 0, 10)

sim_slr = function(x, beta_0 = 5, beta_1 = -3, sigma = sqrt(10.24)) {
  n = length(x)
  epsilon = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + epsilon
  data.frame(predictor = x, response = y)
}
```

(b) Fit a model to your simulated data. Report the estimated coefficients. Are they close to what you would expect? Briefly explain.

Solution:

```
y = sim_slr(x)
sim_model = lm(response ~ predictor, y)
coefficients(sim_model)
```

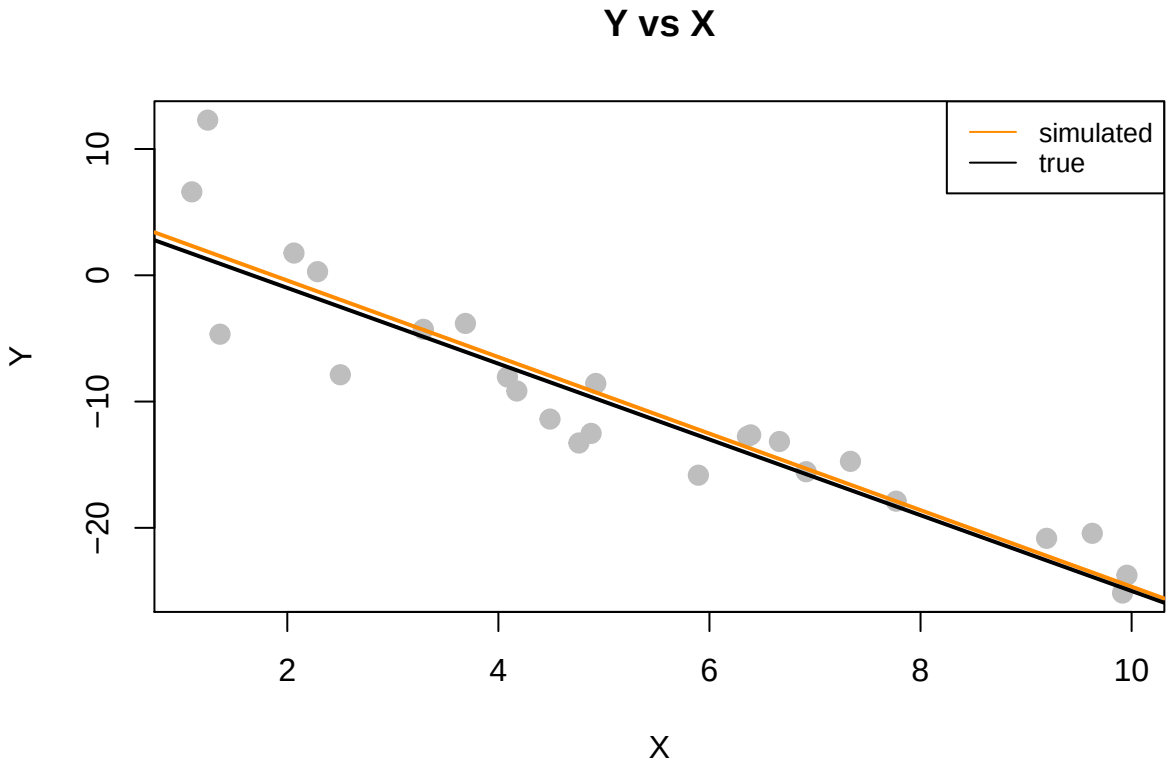
```
## (Intercept)   predictor
##    5.647334   -3.030475
```

The estimated intercept and slope are close to my expectation, because they are close to the values from the initial model.

(c) Plot the data you simulated in part **(a)**. Add the regression line from part **(b)** as well as the line for the true model. Hint: Keep all plotting commands in the same chunk.

Solution:

```
plot(response ~ predictor, data = y,
     xlab = "X",
     ylab = "Y",
     main = "Y vs X",
     pch = 20,
     cex = 2,
     col = "grey")
abline(sim_model, lwd = 2, col = "darkorange")
abline(a = 5, b = -3, lwd = 2, col = "black")
legend("topright", legend = c("simulated", "true"), col = c("darkorange", "black"), lty = 1:1, cex = 0.8
```



(d) Use `R` to repeat the process of simulating `n = 25` observations from the above model 1500 times. Each time fit a SLR model to the data and store the value of $\hat{\beta}_1$ in a variable called `beta_hat_1`. Some hints:

- Consider a `for` loop.
- Create `beta_hat_1` before writing the `for` loop. Make it a vector of length 1500 where each element is `0`.
- Inside the body of the `for` loop, simulate new y data each time. Use a variable to temporarily store this data together with the known x data as a data frame.

- After simulating the data, use `lm()` to fit a regression. Use a variable to temporarily store this output.
- Use the `coef()` function and `[]` to extract the correct estimated coefficient.
- Use `beta_hat_1[i]` to store in elements of `beta_hat_1`.
- See the notes on Distribution of a Sample Mean for some inspiration.

You can do this differently if you like. Use of these hints is not required.

Solution:

```
N = 1500
beta_hat_1 = rep(0, N)
for(i in 1:N){
  tmp = sim_slr(x)
  tmp_model = lm(response ~ predictor, tmp)
  beta_hat_1[i] = coefficients(tmp_model)[[2]]
}
```

(e) Report the mean and standard deviation of `beta_hat_1`. Do either of these look familiar?

Solution: The mean of fitted slope is very close to the true slope.

```
mean(beta_hat_1)
```

```
## [1] -3.002352
```

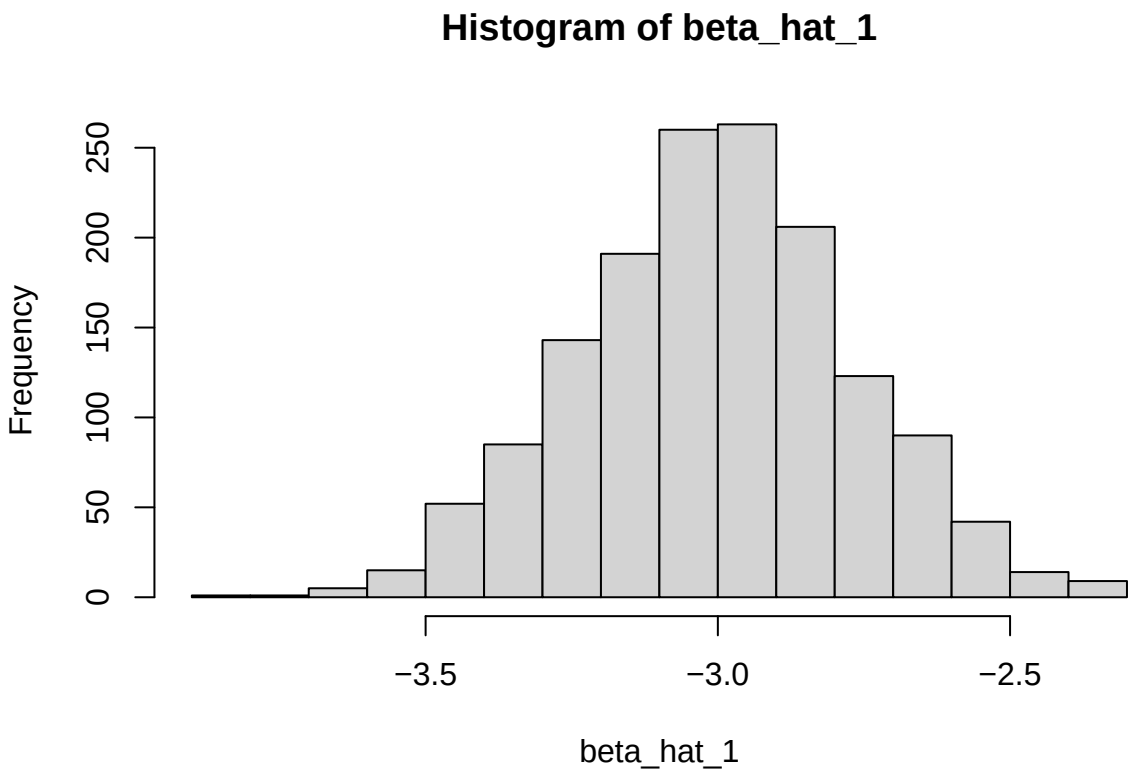
```
sd(beta_hat_1)
```

```
## [1] 0.235362
```

(f) Plot a histogram of `beta_hat_1`. Comment on the shape of this histogram.

Solution: The histogram of fitted β_1 looks like a normal distribution. It has a mean of -3.0023519, which is very close to the true value of -3. The variance of β_1 is 0.235362.

```
hist(beta_hat_1)
```



Exercise 4 (Be a Skeptic)

Consider the model

$$Y_i = 3 + 0 \cdot x_i + \epsilon_i$$

with

$$\epsilon_i \sim N(\mu = 0, \sigma^2 = 4)$$

where $\beta_0 = 3$ and $\beta_1 = 0$.

Before answering the following parts, set a seed value equal to **your** birthday, as was done in the previous exercise.

```
birthday = 18760613
set.seed(birthday)
```

(a) Use `R` to repeat the process of simulating `n = 75` observations from the above model 2500 times. For the remainder of this exercise, use the following "known" values of x.

```
x = runif(n = 75, 0, 10)
```

Each time fit a SLR model to the data and store the value of $\hat{\beta}_1$ in a variable called `beta_hat_1`. You may use the `sim_slr` function provided in the text. Hint: Yes $\beta_1 = 0$.

Solution:

```
birthday = 19900210
set.seed(birthday)
x = runif(n = 75, 0, 10)

# Adjust parameters for the new simulation
N = 2500
beta_hat_1 = rep(0, N)
for(i in 1:N){
  tmp = sim_slr(x, beta_0 = 3, beta_1 = 0, sigma = 2)
  tmp_model = lm(response ~ predictor, tmp)
  beta_hat_1[i] = coefficients(tmp_model)[[2]]
}
```

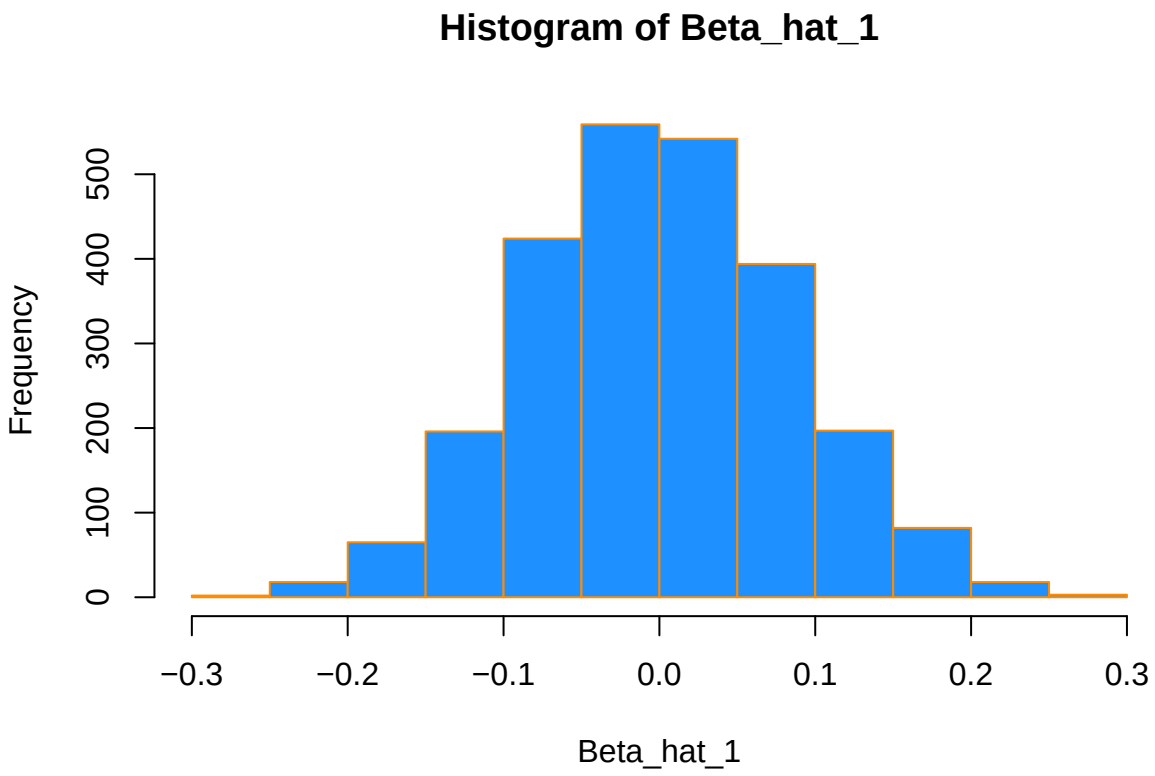
(b) Plot a histogram of `beta_hat_1`. Comment on the shape of this histogram.

Solution: The histogram of `beta_hat_1` looks like a normal distribution with a mean of 0.000429 and a standard deviation of 0.0839194.

```
summary(beta_hat_1)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.264532 -0.057722 -0.001004  0.000429  0.057934  0.273690
```

```
hist(beta_hat_1,
     xlab   = "Beta_hat_1",
     main   = "Histogram of Beta_hat_1",
     col    = "dodgerblue",
     border = "darkorange")
```



(c) Import the data in skeptic.csv and fit a SLR model. The variable names in `skeptic.csv` follow the same convention as those returned by `sim_slr()`. Extract the fitted coefficient for β_1.

Solution

```
skeptic = read.csv("skeptic.csv")
skeptic_model = lm(response ~ predictor, skeptic)
coefficients(skeptic_model)
```

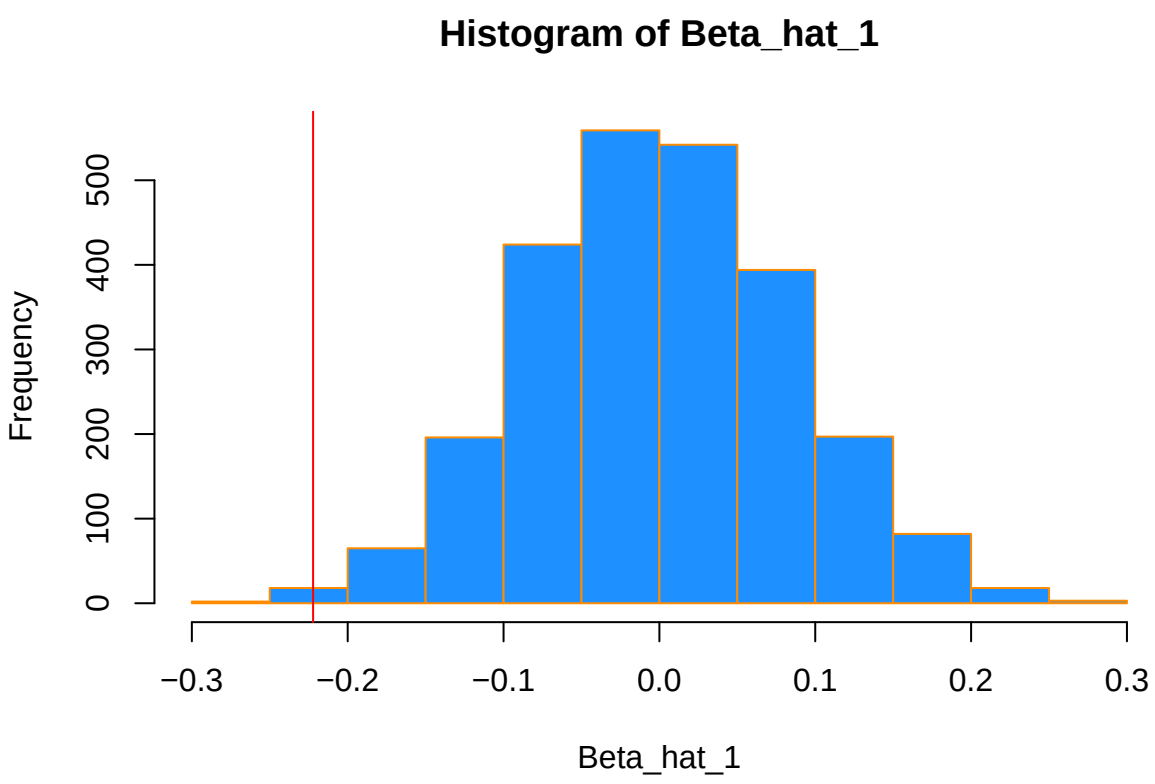
```
## (Intercept)   predictor
##   3.1504230  -0.2221927
```

The fitted β_1 is -0.2221927

(d) Re-plot the histogram from **(b)**. Now add a vertical red line at the value of $\hat{\beta}_1$ in part **(c)**. To do so, you'll need to use `abline(v = c, col = "red")` where `c` is your value.

Solution:

```
hist(beta_hat_1,
     xlab   = "Beta_hat_1",
     main   = "Histogram of Beta_hat_1",
     col    = "dodgerblue",
     border = "darkorange")
abline(v = coefficients(skeptic_model)[[2]], col = "red")
```



(e) Your value of $\hat{\beta}_1$ in **(c)** should be negative. What proportion of the `beta_hat_1` values is smaller than your $\hat{\beta}_1$? Return this proportion, as well as this proportion multiplied by `2`.

Solution:

```
c = coefficients(skeptic_model)[[2]]
proportion = length(beta_hat_1[beta_hat_1 < c]) / length(beta_hat_1)
```

The proportion is 0.0028, and twice of it is 0.0056.

(f) Based on your histogram and part **(e)**, do you think the `skeptic.csv` data could have been generated by the model given above? Briefly explain.

Solution: No, I think the probability that `skeptic.csv` data have been generated by the model given above is very **low**. From my simulation, only about 0.56% of the cases are equal or more extreme than `skeptic.csv` data. So, it is more likely that `skeptic.csv` data is not generated by the model given above.

Exercise 5 (Comparing Models)

For this exercise we will use the `Ozone` dataset from the `mlbench` package. You should use `?Ozone` to learn about the background of this dataset. You may need to install the `mlbench` package. If you do so, do not include code to install the package in your `R` Markdown document.

For simplicity, we will perform some data cleaning before proceeding.

```
data(Ozone, package = "mlbench")
Ozone = Ozone[, c(4, 6, 7, 8)]
colnames(Ozone) = c("ozone", "wind", "humidity", "temp")
Ozone = Ozone[complete.cases(Ozone), ]
```

We have:

- Loaded the data from the package
- Subset the data to relevant variables
  - This is not really necessary (or perhaps a good idea) but it makes the next step easier
- Given variables useful names
- Removed any observation with missing values
  - This should be given much more thought in practice

For this exercise we will define the "Root Mean Square Error" of a model as

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

(a) Fit three SLR models, each with "ozone" as the response. For the predictor, use "wind speed," "humidity percentage," and "temperature" respectively. For each, calculate RMSE and R^2. Arrange the results in a markdown table, with a row for each model. Suggestion: Create a data frame that stores the results, then investigate the `kable()` function from the `knitr` package.

Solution:

```
# define a function for RMSE
rmse = function(y_i, y_hat){
  return (sqrt(sum((y_i - y_hat)^2) / length(y_i)))
}

# define a function R^2
r2 = function(y_i, y_hat){
  y_m = mean(y_i)
  SST = sum((y_i - y_m)^2)
  SSReg = sum((y_hat - y_m)^2)
  return (SSReg / SST)
}

# initialize a dataframe
models = data.frame(c("wind", "humidity", "temperature"), rep(0, 3), rep(0,3))
colnames(models) = c("Predictor", "RMSE", "R_squared")

# wind speed
wind_model = lm(ozone ~ wind, data = Ozone)
#summary(wind_model)
x = Ozone$wind
y_i = Ozone$ozone
y_hat = coefficients(wind_model)[[1]] + x * coefficients(wind_model)[[2]]
models$RMSE[1] = rmse(y_i = y_i, y_hat = y_hat)
```

```
models$R_squared[1] = r2(y_i = y_i, y_hat = y_hat)

# wind speed
humidity_model = lm(ozone ~ humidity, data = Ozone)
#summary(humidity_model)
x = Ozone$humidity
y_i = Ozone$ozone
y_hat = coefficients(humidity_model)[[1]] + x * coefficients(humidity_model)[[2]]
models$RMSE[2] = rmse(y_i = y_i, y_hat = y_hat)
models$R_squared[2] = r2(y_i = y_i, y_hat = y_hat)

# wind speed
temp_model = lm(ozone ~ temp, data = Ozone)
#summary(temp_model)
x = Ozone$temp
y_i = Ozone$ozone
y_hat = coefficients(temp_model)[[1]] + x * coefficients(temp_model)[[2]]
models$RMSE[3] = rmse(y_i = y_i, y_hat = y_hat)
models$R_squared[3] = r2(y_i = y_i, y_hat = y_hat)

# build the table
knitr::kable(models)
```

Predictor	RMSE	R_squared
wind	7.961695	0.0001402
humidity	7.147822	0.1941105
temperature	5.009257	0.6042011

(b) Based on the results, which of the three predictors used is most helpful for predicting ozone readings? Briefly explain.

Solution: Temperature is the most helpful predictor for predicting ozone readings, because it has the largest R^2 value and the smallest $RMSE$ value. The largest R^2 means that the model using temperature as a predictor can explain most of the data. Smallest $RMSE$ value also indicates that the model using temperature as a predictor has the least residuals.

Exercise 00 (SLR without Intercept)

This exercise will *not* be graded and is simply provided for your information. No credit will be given for the completion of this exercise. Give it a try now, and be sure to read the solutions later.

Sometimes it can be reasonable to assume that β_0 should be 0. That is, the line should pass through the point $(0, 0)$. For example, if a car is traveling 0 miles per hour, its stopping distance should be 0! (Unlike what we saw in the book.)

We can simply define a model without an intercept,

$$Y_i = \beta x_i + \epsilon_i.$$

(a) In the **Least Squares Approach** section of the text you saw the calculus behind the derivation of the regression estimates, and then we performed the calculation for the `cars` dataset using `R`. Here you need to do, but not show, the derivation for the slope only model. You should then use that derivation of $\hat{\beta}$ to write a function that performs the calculation for the estimate you derived.

In summary, use the method of least squares to derive an estimate for β using data points (x_i, y_i) for $i = 1, 2, \ldots n$. Simply put, find the value of β to minimize the function

$$f(\beta) = \sum_{i=1}^{n}(y_i - \beta x_i)^2.$$

Then, write a function `get_beta_no_int` that takes input:

- `x` - A predictor variable
- `y` - A response variable

The function should then output the $\hat{\beta}$ you derived for a given set of data.

Solution:

```
# function to estimate beta
get_beta_no_int = function(x, y){
  return (sum(y) / sum(x))
}
```

(b) Write your derivation in your `.Rmd` file using TeX. Or write your derivation by hand, scan or photograph your work, and insert it into the `.Rmd` as an image. See the RMarkdown documentation for working with images.

Solution:

$\frac{\partial f}{\partial \beta} = \sum_{i=1}^{n}(-2x_i(y_i - \beta x_i))$

$f(\beta)$ is minimized when the above expression equals zero, and because x_i cannot always be zero, so

$\sum_{i=1}^{n}(y_i - \beta x_i) = 0$

$\beta = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i}$

(c) Test your function on the `cats` data using body weight as `x` and heart weight as `y`. What is the estimate for β for this data?

Solution:

```
beta = get_beta_no_int(x = cats$Bwt, y = cats$Hwt)
```

β is 3.9031107.

(d) Check your work in `R`. The following syntax can be used to fit a model without an intercept:

```
lm(response ~ 0 + predictor, data = dataset)
```

Use this to fit a model to the `cat` data without an intercept. Output the coefficient of the fitted model. It should match your answer to **(c)**.

Solution:

```
new_model = lm(Hwt ~ 0 + Bwt, data = cats)
coefficients(new_model)
```

```
##      Bwt
## 3.907113
```

The value of β from using `lm()` is 3.9071133, it is almost identical to the manually calculated β.