Week 2 - Homework

STAT 420, Summer 2018, Connor Segneri - segneri3

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.

```
library(MASS)
```

(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`.

```
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.

```
cat_model$coefficients
```

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

$\hat{\beta}_0$, **-0.3566624** is supposed to represent the heart weight (g) of a cat when the weight (kg) is zero. Obviously the height and weight of a cat must be greater than zero, so any predictions made can be disregarded until they are made within the estimated range of the cat's body weight: `2, 3.9`.

β_1, 4.0340627, represents the relationship between the cat's height and body weight. Based on the data, the linear model estimates that with 1 kg in body weight gained by a cat, the expected heart weight of the cat increases by 1 g.

(c) Use your model to predict the heart weight of a cat that weights **2.7** kg. Do you feel confident in this prediction? Briefly explain.

```r
predict(cat_model, newdata=data.frame(Bwt = 2.7))
```

```
##        1
## 10.53531
```

I feel confident about this prediction, as it was made within the range of the estimation data. The range of the estimation data, the body weight (kg) of the cat, is `2, 3.9`.

(d) Use your model to predict the heart weight of a cat that weights **4.4** kg. Do you feel confident in this prediction? Briefly explain.

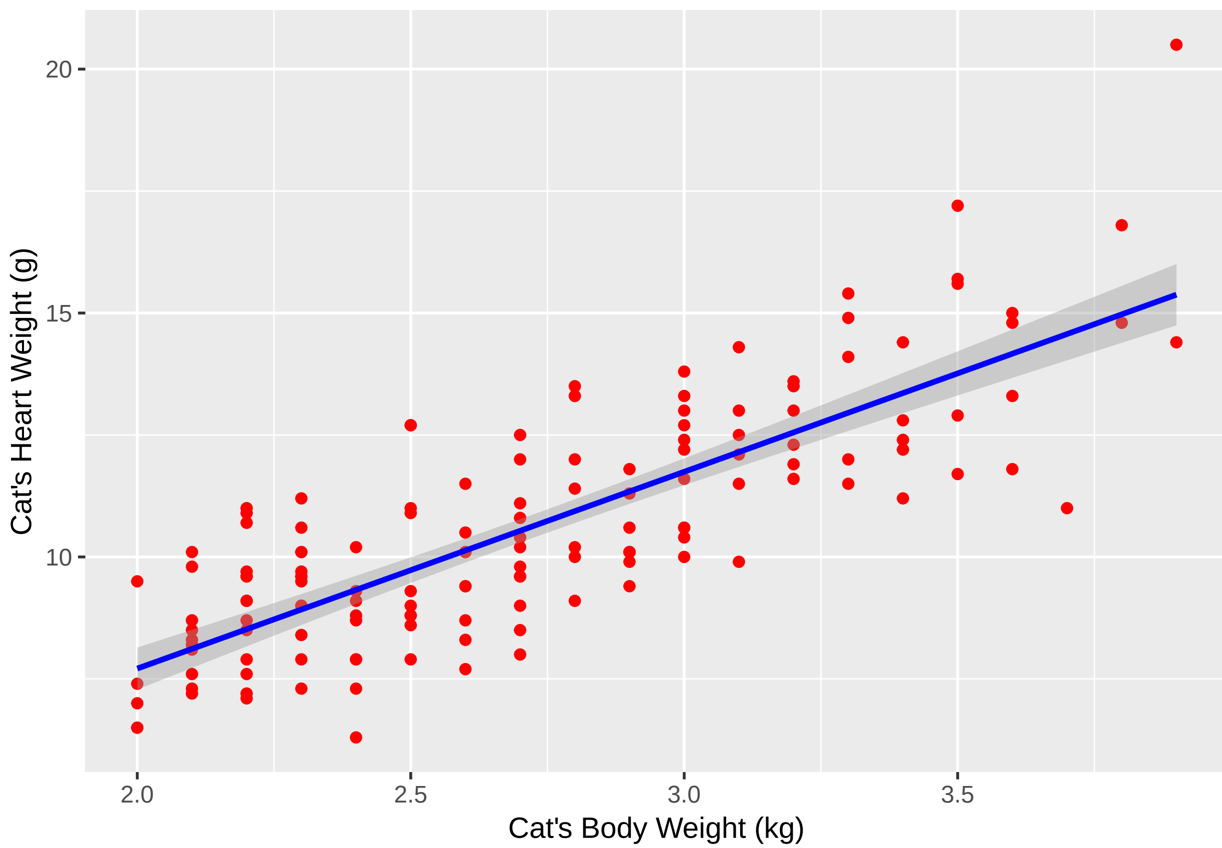
```r
predict(cat_model, newdata=data.frame(Bwt = 4.4))
```

```
##        1
## 17.39321
```

I do not feel as condfident in this prediction. The relationship between the independent and dependent variables can change outside of the estimation range, which we previously established to be `2, 3.9` for a cat's body weight (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.

```r
library(ggplot2)

ggplot(data=cats, aes(x=Bwt,y=Hwt)) +
  geom_point(color = 'red') +
  labs(x = "Cat's Body Weight (kg)", y = "Cat's Heart Weight (g)") +
  geom_smooth(method="lm", col="blue")
```



(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)**.

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

```
## [1] 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`.

```r
get_sd_est = function(fitted_vals, actual_vals, mle=FALSE) {
  if (mle){
    sqrt((1/length(actual_vals)) * sum((actual_vals - fitted_vals)^2))
  } else {
    sqrt((1/(length(actual_vals)-2)) * sum((actual_vals - fitted_vals)^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.

```r
actual_values = cats$Hwt
fitted_values = predict(cat_model, newdata = cats)

get_sd_est(fitted_vals = fitted_values, actual_vals = actual_values)
```

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

This means that s_e, or the standard error of estimate is **1.452373**. This measures the variation in the actual values of the cat's heart weight to the computed values of the cat's heart weight using the linear model **cat_model**. The smaller the value is the better, if the standard error of estimate is zero, then the correlation is perfect.

(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)**.

```r
get_sd_est(fitted_vals = fitted_values, actual_vals = actual_values, mle = TRUE)
```

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

The estimated standard deviation, or $\hat{\sigma}$, is **1.442252**. The difference between this value and the standard error of estimate is minor. However, standard error of estimation is considered unbiased, since $E[s_e] = \sigma^2$, and the estimated standard deviation is considered biased because $\hat{\sigma} \neq \sigma^2$.

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

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

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

As expected, $E[s_e] = \sigma^2$.

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 = 19951015
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`.

```
n_obs = 25
beta_0 = 5
beta_1 = -3
sigma = sqrt(10.24)

sim_slr = function(x, beta_0 = 10, beta_1 = 5, sigma = 1) {
  n = length(x)
  epsilon = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + epsilon
  data.frame(predictor = x, response = y)
}

xy_predicted_pairs = sim_slr(x, beta_0=beta_0, beta_1=beta_1, sigma=sigma)
xy_predicted_pairs
```

```
##    predictor     response
## 1  2.2742908   0.76549469
## 2  2.5102764  -0.75829233
## 3  1.4152343   0.16623765
## 4  2.4855526  -2.70042609
## 5  9.6050454 -26.29040805
## 6  9.3487184 -27.14492106
## 7  3.0963052  -1.39510933
## 8  7.6120364 -23.90765300
## 9  0.1986710  11.08491420
## 10 6.0133098 -14.24597704
## 11 7.0420572 -11.05773776
## 12 8.8861357 -24.07050746
## 13 6.2577079 -15.49910844
## 14 2.4750132  -1.91129573
## 15 5.0603148 -15.26354468
```

```
## 16 9.2643617 -21.88985949
## 17 0.7721950   6.81281214
## 18 9.9277848 -23.93170619
## 19 8.6605695 -20.48473739
## 20 3.2194585  -2.93673970
## 21 9.7567962 -23.81214427
## 22 6.6705660 -14.62496419
## 23 3.9556773  -4.54954044
## 24 8.1895364 -21.95878316
## 25 0.4211773  -0.08963973
```

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

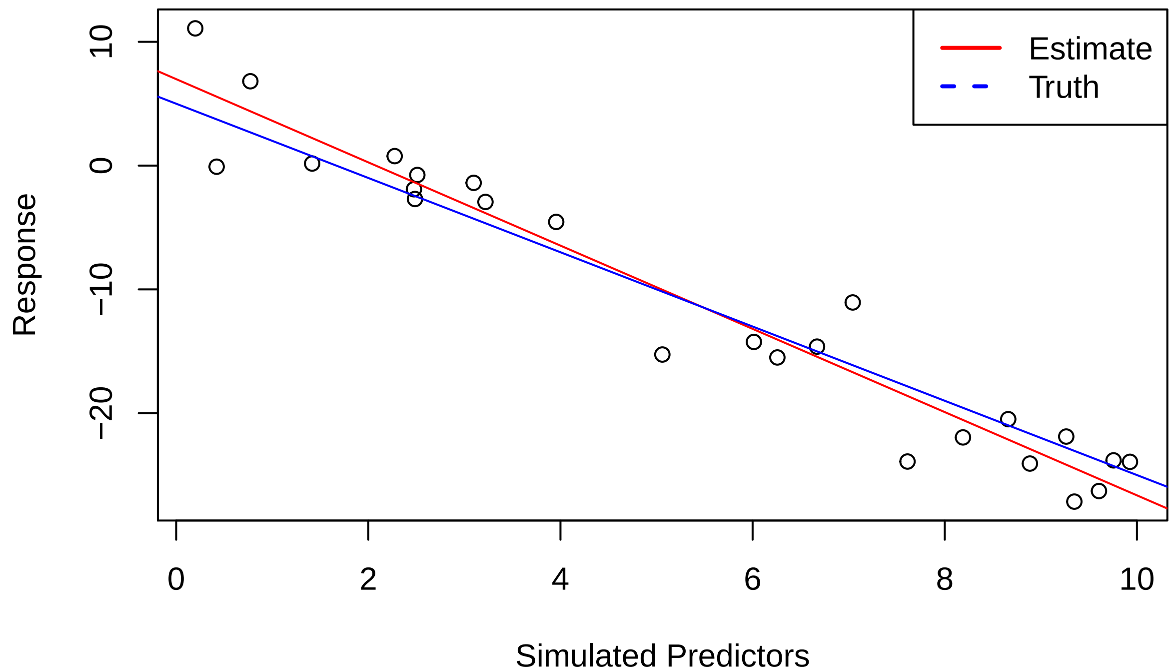
```r
model = lm(response ~ predictor, data=xy_predicted_pairs)

summary(model)
```

```
##
## Call:
## lm(formula = response ~ predictor, data = xy_predicted_pairs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6581 -1.4078  0.6974  2.0085  5.6351
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.9846     1.1374   6.141  2.9e-06 ***
## predictor    -3.3623     0.1803 -18.653  2.2e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.935 on 23 degrees of freedom
## Multiple R-squared:  0.938,  Adjusted R-squared:  0.9353
## F-statistic: 347.9 on 1 and 23 DF,  p-value: 2.199e-15
```

This is about what I expected, the only strange thing is how far off the y-intercept is from the original model. Original was `5` and the model gives `6.9846` for the intercept. The slope is only `0.3` off.

(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.

```r
plot(response ~ predictor, data=xy_predicted_pairs,
     xlab = "Simulated Predictors",
     ylab = "Response")
abline(model, col="red")
abline(beta_0, beta_1, col="blue")
legend("topright", c("Estimate", "Truth"), col = c("red", "blue"), lty = c(1,2), lwd = 2)
```



(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.

```
beta_hat_1 = c()
for (i in 1:1500){
  xy_predicted_pairs = sim_slr(x, beta_0=beta_0, beta_1=beta_1, sigma=sigma)
  model = lm(response ~ predictor, data=xy_predicted_pairs)
  beta_hat_1[i] = coef(model)[2]
}
```

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

```r
mean(beta_hat_1)
```

```
## [1] -3.003974
```

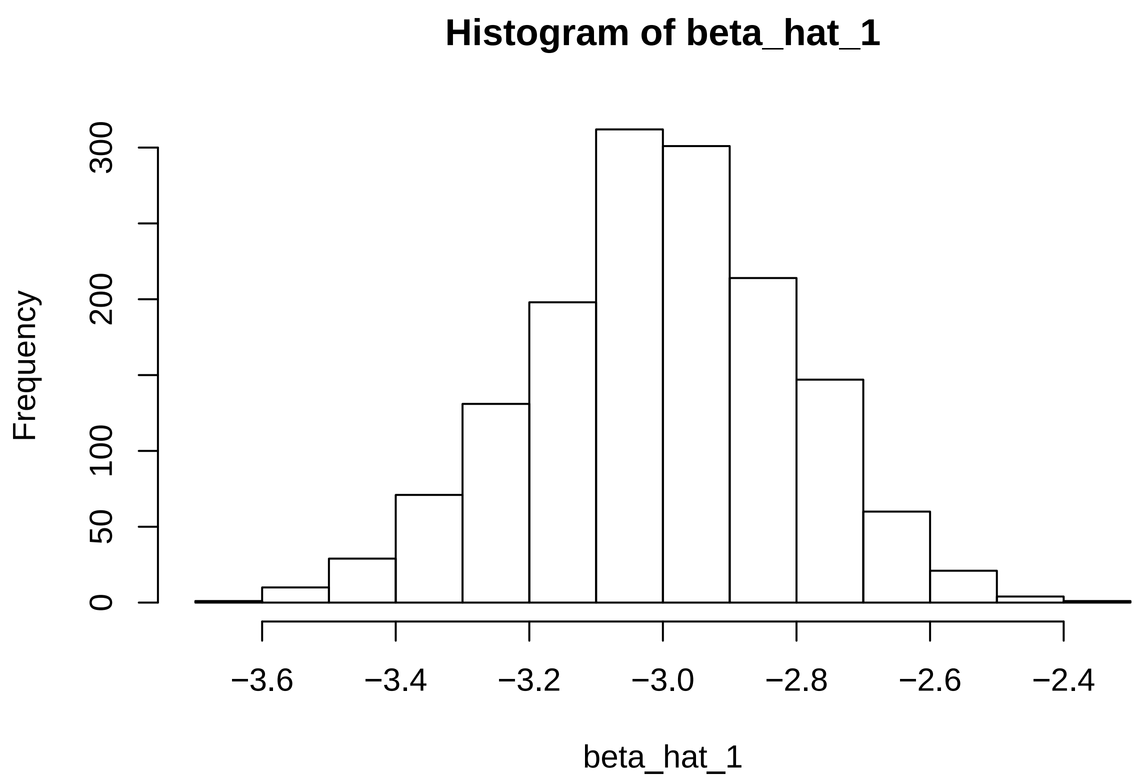
```r
sd(beta_hat_1)
```

```
## [1] 0.1964697
```

The mean of beta_hat_1 looks a lot like beta_1 for the original model, `-3`.

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

```r
hist(beta_hat_1)
```



This histogram is bell-shaped. This indicates that the data is unimodal, meaning that the data has a single mode, identified by the 'peak' of the curve. The histogram looks this way because the data set is normally distributed.

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 = 19951015
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.

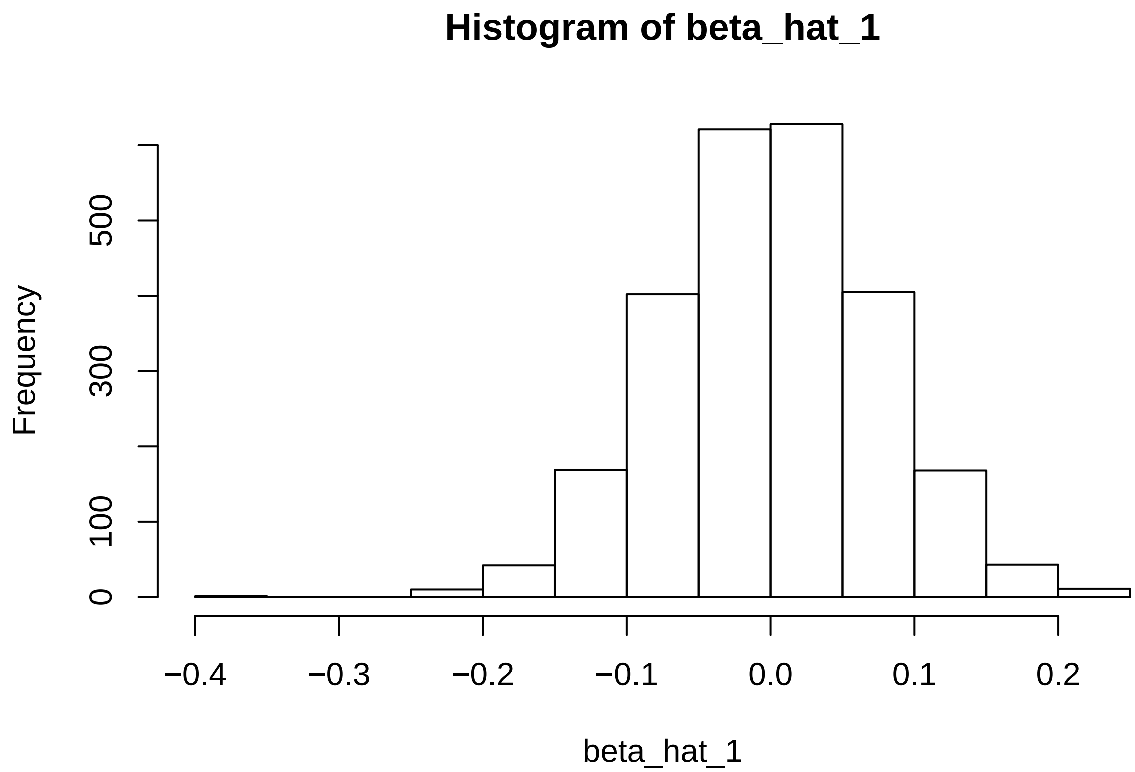
```
x = runif(n = 75, 0, 10)
```

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 $\beta_1 = 0$

```
n_obs = 75
beta_0 = 3
beta_1 = 0
sigma = sqrt(4)

beta_hat_1 = c()
for (i in 1:2500){
  xy_predicted_pairs = sim_slr(x, beta_0=beta_0, beta_1=beta_1, sigma=sigma)
  model = lm(response ~ predictor, data=xy_predicted_pairs)
  beta_hat_1[i] = coef(model)[2]
}
```

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

```
hist(beta_hat_1)
```



This histogram is left-skewed. The mean of the data is closer to the right than either the median of the mode.

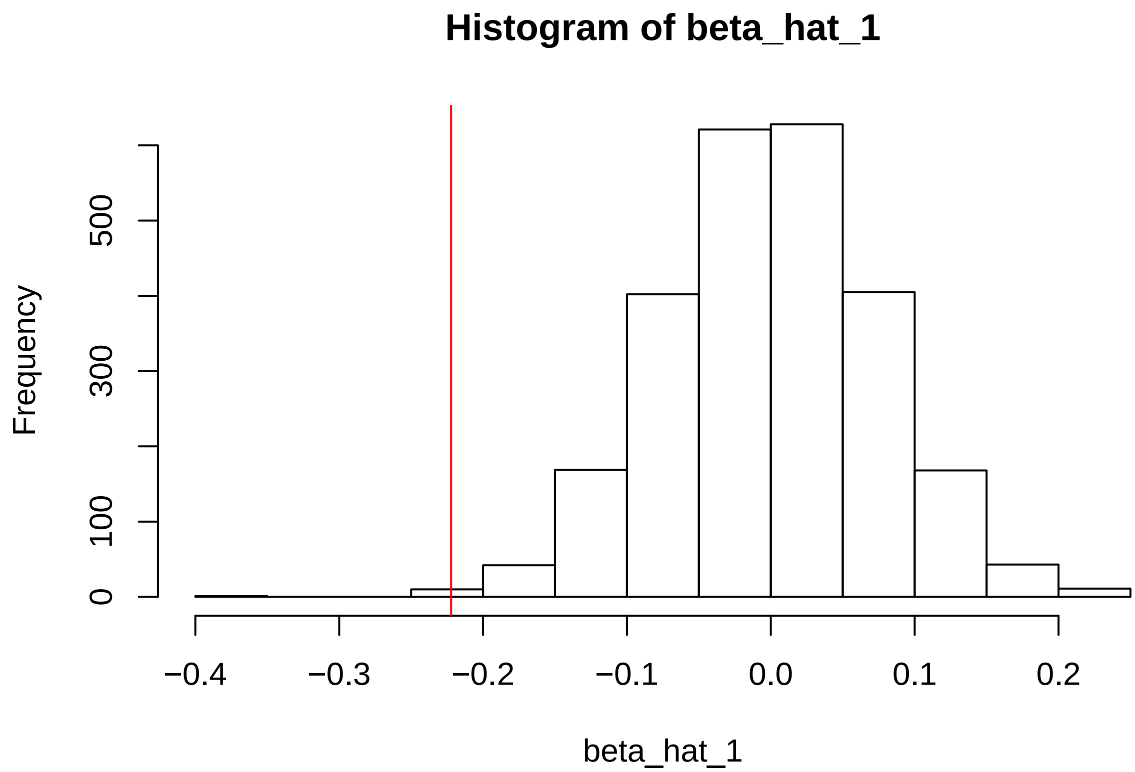
(c) Import the data in skeptic.csv and fit a SLR model. The variable names in `skeptic.csv` follow the same convention as those returned by `sim_slr()`. Extract the fitted coefficient for β_1.

```
library(readr)
skeptics = read_csv('skeptic.csv')

skeptics_model = lm(response ~ predictor, data=skeptics)

beta_1 = coef(skeptics_model)[2]
beta_1
```

```
##  predictor
## -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.

```
hist(beta_hat_1)
abline(v=beta_1, 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`.

```
prop_smaller = length(beta_hat_1[beta_hat_1 < beta_1]) / length(beta_hat_1)
prop_smaller
```

```
## [1] 8e-04
```

```
prop_smaller * 2
```

```
## [1] 0.0016
```

(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.

No I don't thnk the `skeptic.csv` data could have been generated by the model given above. The slope of `skeptics model`, $\hat{\beta}_1$, generated from the skeptics data is far from where the bulk of the generated beta_1 values are in `beta_hat_1`. Looking at the histogram tells us that much. An incredibly low proportion of of the `beta_hat_1` values are smaller than $\hat{\beta}_1$. If the skeptics data was generated by the original model, then I would expect the proportion calculated in **e** to be closer to `0.5`. It's still possible that the skeptics data was generated by the original model, but it is improbable.

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.

```r
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.

```r
library(kableExtra)

wind_model = lm(ozone ~ wind, data=Ozone)
humidity_model = lm(ozone ~ humidity, data=Ozone)
temp_model = lm(ozone ~ temp, data=Ozone)

find_rmse = function(actual, predicted){
  sqrt((1/length(actual)) * sum((actual - predicted)^2))
}

df = data.frame(
  name = c("Wind Model", "Humidity Model", "Temperature Model"),
  rmse = c(
    find_rmse(actual = Ozone$ozone, predicted = predict(wind_model, newdata = Ozone)),
    find_rmse(actual = Ozone$ozone, predicted = predict(humidity_model, newdata = Ozone)),
    find_rmse(actual = Ozone$ozone, predicted = predict(temp_model, newdata = Ozone))
  ),
  r_squared = c(
```

```r
    summary(wind_model)$r.squared,
    summary(humidity_model)$r.squared,
    summary(temp_model)$r.squared
  )
)

kable(df) %>%
  kable_styling()
```

name	rmse	r_squared
Wind Model	7.961695	0.0001402
Humidity Model	7.147822	0.1941105
Temperature Model	5.009257	0.6042011

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

The predictor that is most helpful for predicting ozone readings is `temperature`. This predictor has the lowest rmse value, and an r_squared value closest to 1.

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.

(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.

(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?

(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)**.