# Week 2 - 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 `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`.

```
library(MASS, lib.loc = "/usr/lib/R/library")
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 $\beta_1$ in the *context of the problem*. Be aware that only one of those is an estimate.

```r
coef(cat_model)
```

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

beta_0 = -0.3566624 (this is an estimate for a body weight 0) beta_1 = 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.

```r
predict(cat_model, data.frame(Bwt = 3.1))
```

```
##        1
## 12.14893
```

```r
range(cats$Bwt)
```

```
## [1] 2.0 3.9
```

Since the weight of the cat is with-in the range of our dataset, we are confident in our 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.

```r
predict(cat_model, data.frame(Bwt = 1.5))
```

```
##        1
## 5.694432
```

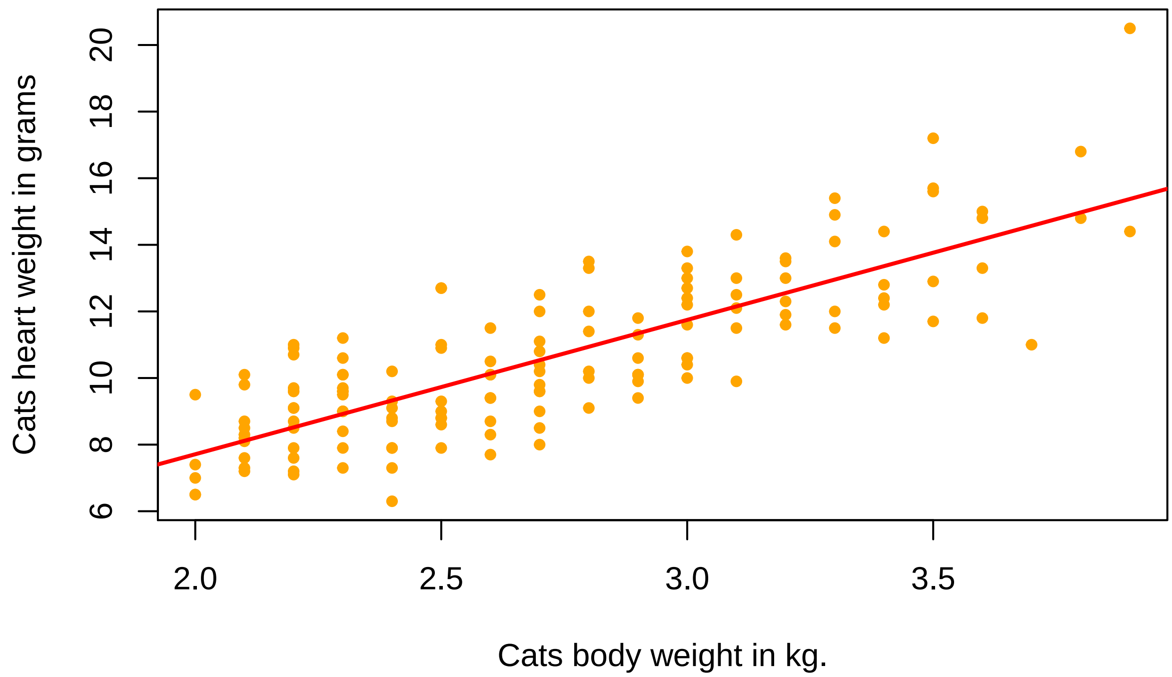
```r
range(cats$Bwt)
```

```
## [1] 2.0 3.9
```

Since the weight of the cat is NOT with-in the range of our dataset, we are NOT confident in our prediction.

**(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
plot(Hwt ~ Bwt, data = cats, xlab = "Cats body weight in kg.", ylab = "Cats heart weight in grams", pch
abline(cat_model, lwd = 2, col = "red")
```




**(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
```

R2 value 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 $\sigma$ 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`.

```
get_sd_est = function(fitted_vals, actual_vals, mle = FALSE) {
  e = actual_vals - fitted_vals
```

```r
  n = length(e)
  if (mle == TRUE) {
    s2_e = sum(e^2)/(n)
  } else {
    s2_e = sum(e^2)/(n-2)
  }
  return (s2_e^0.5)
}
```

**(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
fitted_values = cat_model$fitted.values
actual_vals = cats$Hwt
get_sd_est(fitted_values, actual_vals, FALSE)
```

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

In case of MLE is false, using the lm model we get a value of 1.452373 (Se). This means for a given cat's body weight, the estimated cat's heart weight could be off by about 1.452373 grams.

**(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
fitted_values = cat_model$fitted.values
actual_vals = cats$Hwt
get_sd_est(fitted_values, actual_vals, TRUE)
```

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

In case of MLE is true, using the lm model we get a value of 1.442252. This means for a given cat's body weight, the estimated cat's heart weight could be off by about 1.442252 grams.

**(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
```

We notice that this value matches with **(b)** "Se".

---

## 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 #19830611
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`.

```
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)
}

sim_data = sim_slr(x = x, beta_0 = 5, beta_1 = -3, sigma = sqrt(10.24))
```

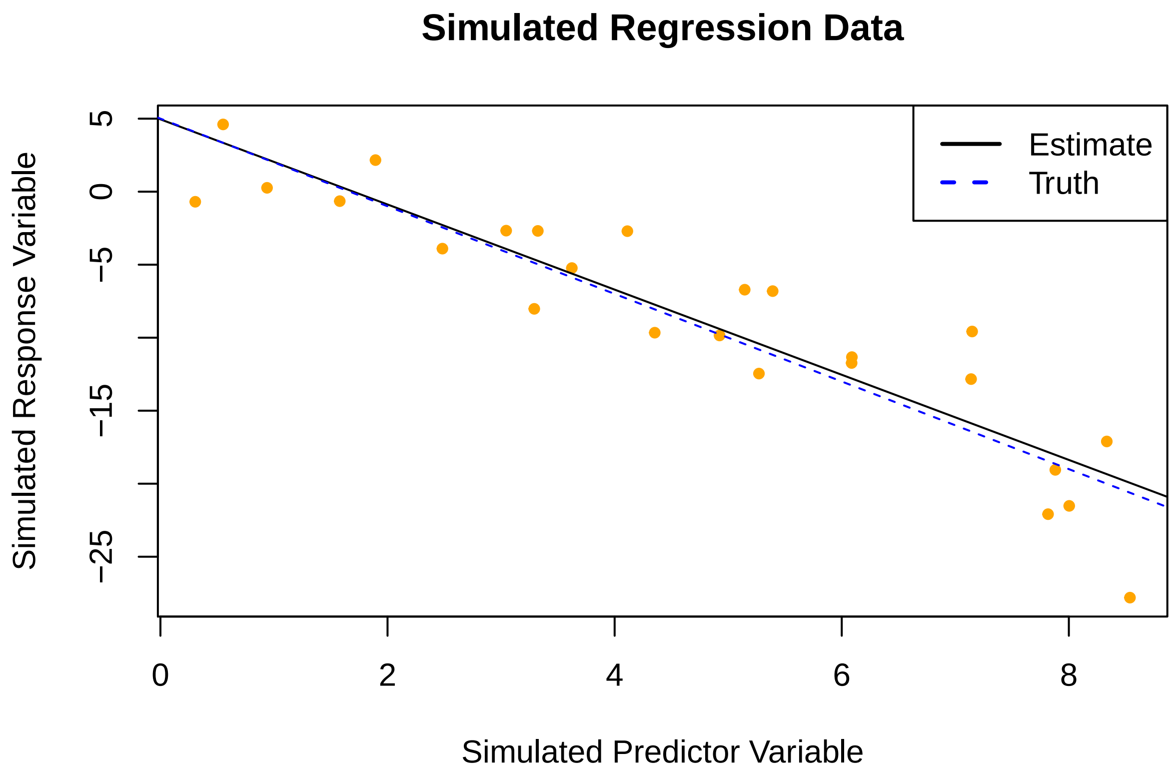
**(b)** Fit a model to your simulated data. Report the estimated coefficients. Are they close to what you would expect? Briefly explain.

```
sim_fit = lm(response ~ predictor, data = sim_data)
coef(sim_fit)
```

```
## (Intercept)   predictor
##    4.954205   -2.915525
```

**(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.

```
plot(response ~ predictor, data = sim_data, xlab = "Simulated Predictor Variable", ylab = "Simulated Re
     main = "Simulated Regression Data", pch = 20, cex = 1, col = "orange")
abline(sim_fit, lwd = 1, lty = 1, col = "black")
beta_0 = 5
beta_1 = -3
abline(beta_0, beta_1, lwd = 1, lty = 2, col = "blue")
legend("topright", c("Estimate", "Truth"), lty = c(1,2), lwd = 2, col = c("black", "blue"))
```




**(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:

```
beta_hat_1 = rep(0, 1500)
for (i in 1:1500) {
  sim_data = sim_slr(x, beta_0 = 5, beta_1 = -3, sigma = sqrt(10.24))
  sim_model = lm(response ~ predictor, data = sim_data)
  beta_hat_1[i] = coef(sim_model)[2]
}
```

- 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.

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

```
mean(beta_hat_1)
```

```
## [1] -2.995916
```

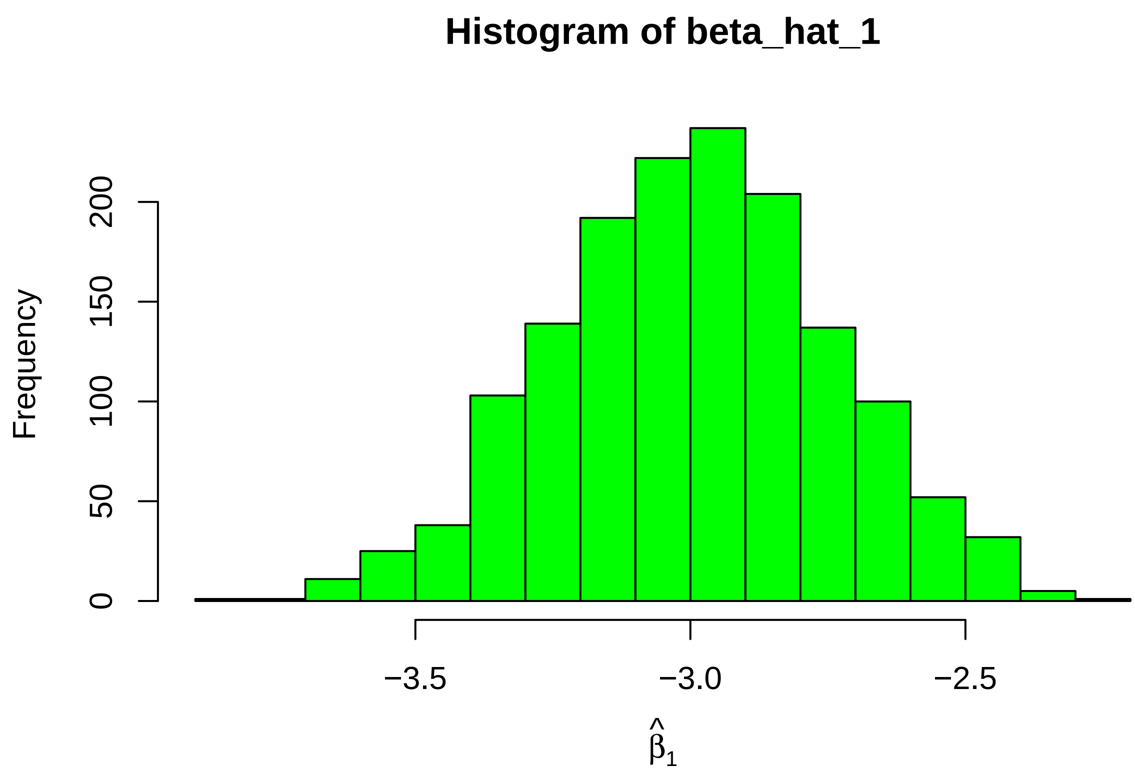
```r
sd(beta_hat_1)
```

```
## [1] 0.2501589
```

Value of beta_hat_1 and mean of beta_hat_1 are pretty close at value -3.

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

```r
hist(beta_hat_1, main = "Histogram of beta_hat_1", xlab = expression(hat(beta)[1]), col = "green")
```




Histogram looks like a normal distribution with a mean of -3.

---

### 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 $x$.

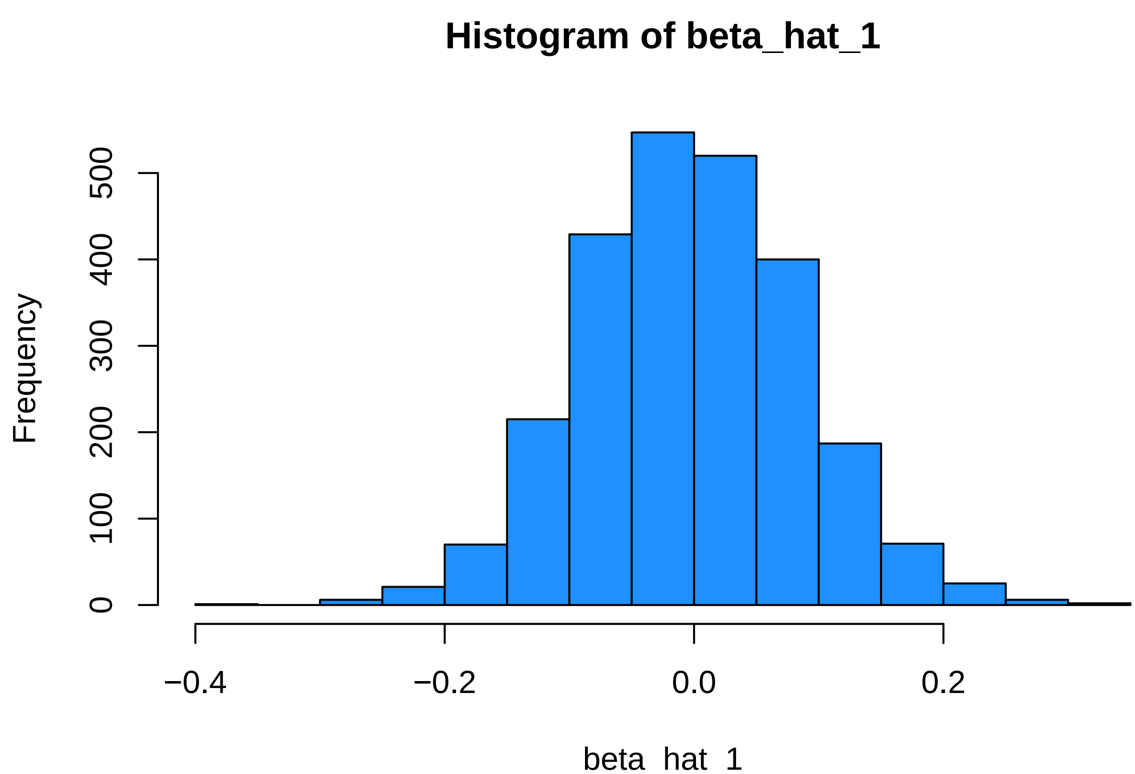
```
x = runif(n = 75, 0, 10)
```

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 $\beta_1 = 0$.

```
beta_0 = 3
beta_1 = 0
sigma = sqrt(4)
beta_hat_1 = rep(0, 2500)
for (i in 1:2500) {
  sim_data = sim_slr(x, beta_0, beta_1, sigma)
  sim_model = lm(response ~ predictor, data = sim_data)
  beta_hat_1[i] = coef(sim_model)[2]
}
```

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

```
hist(beta_hat_1, main = "Histogram of beta_hat_1", col = "dodgerblue")
```




Histogram looks like a normal distribution with a mean of 0.0

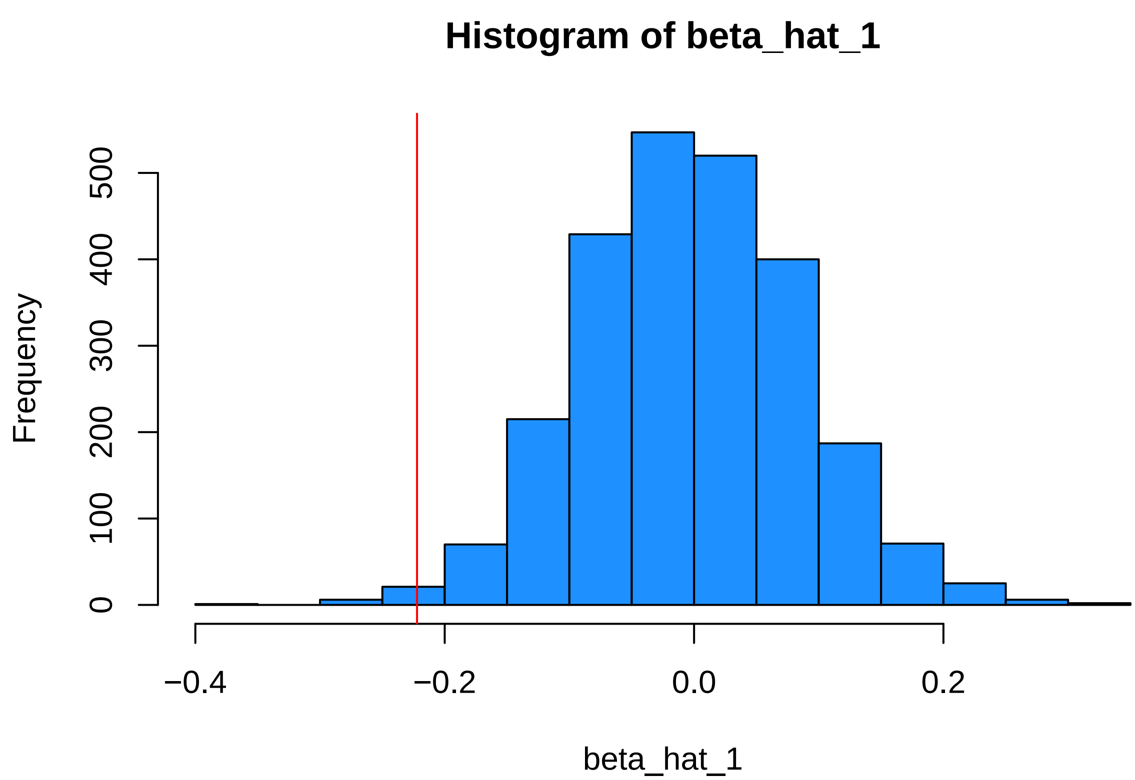
**(c)** Import the data in skeptic.csv and fit a SLR model. The variable names in `skeptic.csv` follow the same convention as those returned by `sim_slr()`. Extract the fitted coefficient for $\beta_1$.

```
skeptic_data = read.csv("skeptic.csv")
skeptic_model = lm(response ~ predictor, data = skeptic_data)
coef(skeptic_model)[2]
```

```
##  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, main = "Histogram of beta_hat_1", col = "dodgerblue")
abline(v = coef(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.

```
mean(beta_hat_1 < coef(skeptic_model)[2])
```

```
## [1] 0.0052
```

```
mean(beta_hat_1 < coef(skeptic_model)[2]) * 2
```

```
## [1] 0.0104
```

**(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.

```r
range(beta_hat_1)
```

```
## [1] -0.3596241  0.3022105
```

---

The beta_1 value from the skeptic histgram value is -0.2221927. This is well with-in the range of beta_hat_1. It is possible to generate.

## 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
ozone_windspeed = lm(ozone ~ wind, data = Ozone)
rmse_wind = sqrt(sum(ozone_windspeed$residuals^2) / length(ozone_windspeed$residuals))
r2_wind = summary(ozone_windspeed)$r.squared

ozone_humidity = lm(ozone ~ humidity, data = Ozone)
rmse_humidity = sqrt(sum(ozone_humidity$residuals^2) / length(ozone_humidity$residuals))
r2_humidity = summary(ozone_humidity)$r.squared

ozone_temperature = lm(ozone ~ temp, data = Ozone)
rmse_temp = sqrt(sum(ozone_temperature$residuals^2) / length(ozone_temperature$residuals))
r2_temp = summary(ozone_temperature)$r.squared

ozone_summary = data.frame(predictor = c("Wind", "Humidity", "Temperature") ,
                           rmse_ozone = c(rmse_wind, rmse_humidity, rmse_temp),
```

```r
                        r2_ozone = c(r2_wind, r2_humidity, r2_temp))

library(knitr)
kable(ozone_summary)
```

| predictor | rmse_ozone | r2_ozone |
|---|---:|---:|
| 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.

ozone prediction with reference temperature yeilds better results, as it has 60% of the observed variability in estimating the ozone.

---