Week 2 - Homework

STAT 420, Summer 2022, Ilya Andreev, iandre3

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)
y = cats$Hwt
x = cats$Bwt
cat_model = lm(y ~ x)
summary(cat_model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## 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
## x             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.

The estimated regression coefficients are: $\hat{\beta}_0$ = -0.3566624, $\hat{\beta}_1$ = 4.0340627.

$\hat{\beta}_0$ is the estimated mean heart weight in grams when the body weight is 0 kg.

β_1 is the true mean increase in heart weight in grams when the body weight increases by 1 kg. We do not know the value of this parameter. We can only estimate it.

(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.

```
prediction = predict(cat_model, data.frame(x=3.1), se.fit=TRUE)
```

The prediction value of 12.1489319 is fairly reliable because the standard error of this estimate mean is only 0.1533667.

(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.

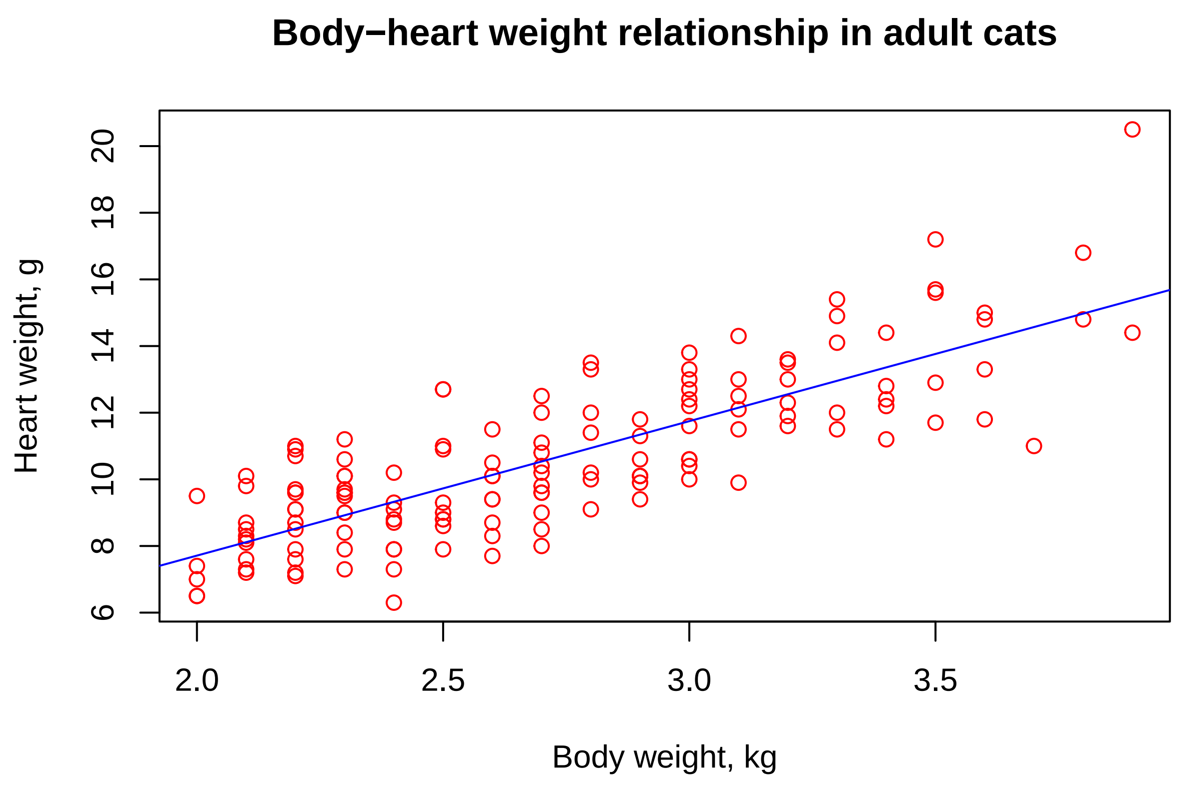
```
prediction = predict(cat_model, data.frame(x=1.5), se.fit=TRUE)
```

The prediction value of 5.6944316 is much less reliable because the standard error of this estimate mean is now 0.3292733, and in addition 1.5 kg is below the range of x we used as the input for our linear model.

(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(y ~ x,
     xlab="Body weight, kg",
     ylab="Heart weight, g",
     main="Body-heart weight relationship in adult cats",
     col="red")

abline(cat_model, 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`.

```
get_sd_est = function(fitted_vals, actual_vals, mle=FALSE) {
  e     = actual_vals - fitted_vals
  n     = length(e)
  if (mle) {
    sigma_hat = sqrt(sum(e^2) / n)
    sigma_hat
  } else {
    s_e  = sqrt(sum(e^2) / (n - 2))
    s_e
  }
}
```

(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.

```
s_e = get_sd_est(predict(cat_model), y, FALSE)
```

The least squares estimate of the standard deviation of the residuals, or the residual standard error, is 1.4523733, as measured in 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)**.

```
sigma_hat = get_sd_est(predict(cat_model), y, TRUE)
```

The maximum likelihood estimate of the standard deviation of the residuals is 1.4422522, as measured in grams. Unlike the value from **(b)**, this value is biased, meaning its expected value over many calculations is not necessarily is not equal to the true value.

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

```
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 = 19991124
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)

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)
}

data = sim_slr(x, 5, -3, sqrt(10.24))
```

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`.

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

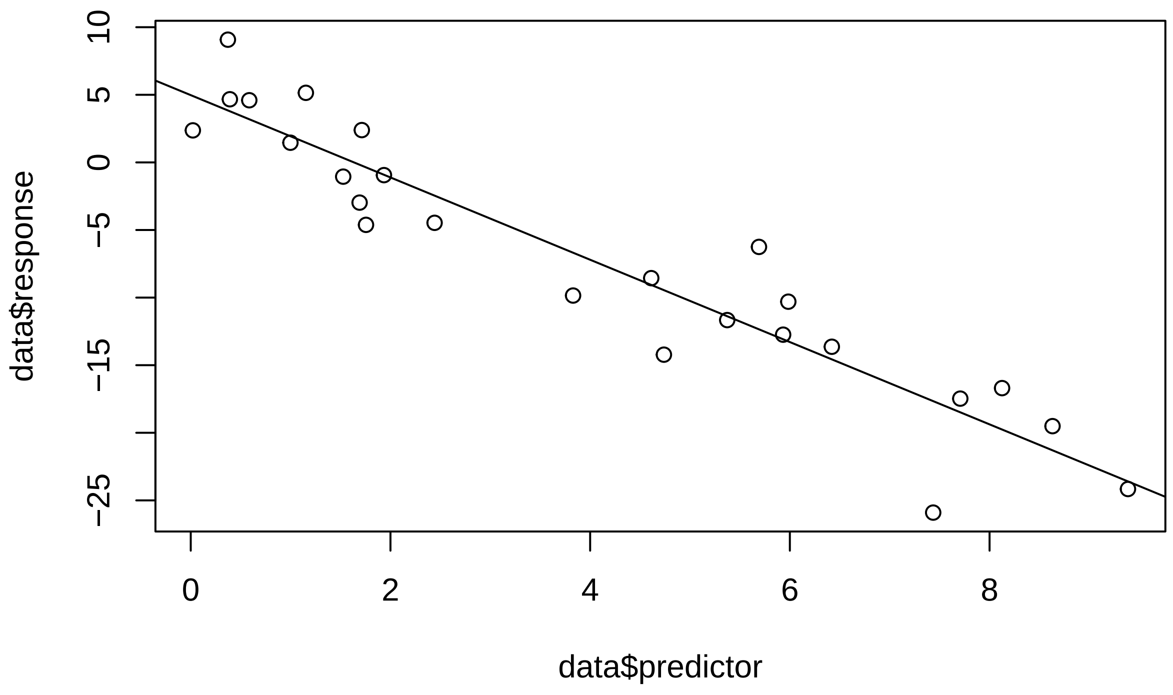
```
model = lm(data$response ~ data$predictor)
model$coefficients
```

```
##    (Intercept) data$predictor
##       4.975480      -3.044189
```

Evidently the coefficients are reasonably close to the true values, given that we had to fit based on only 25 samples.

(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(data$response ~ data$predictor)
abline(model)
```



(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.

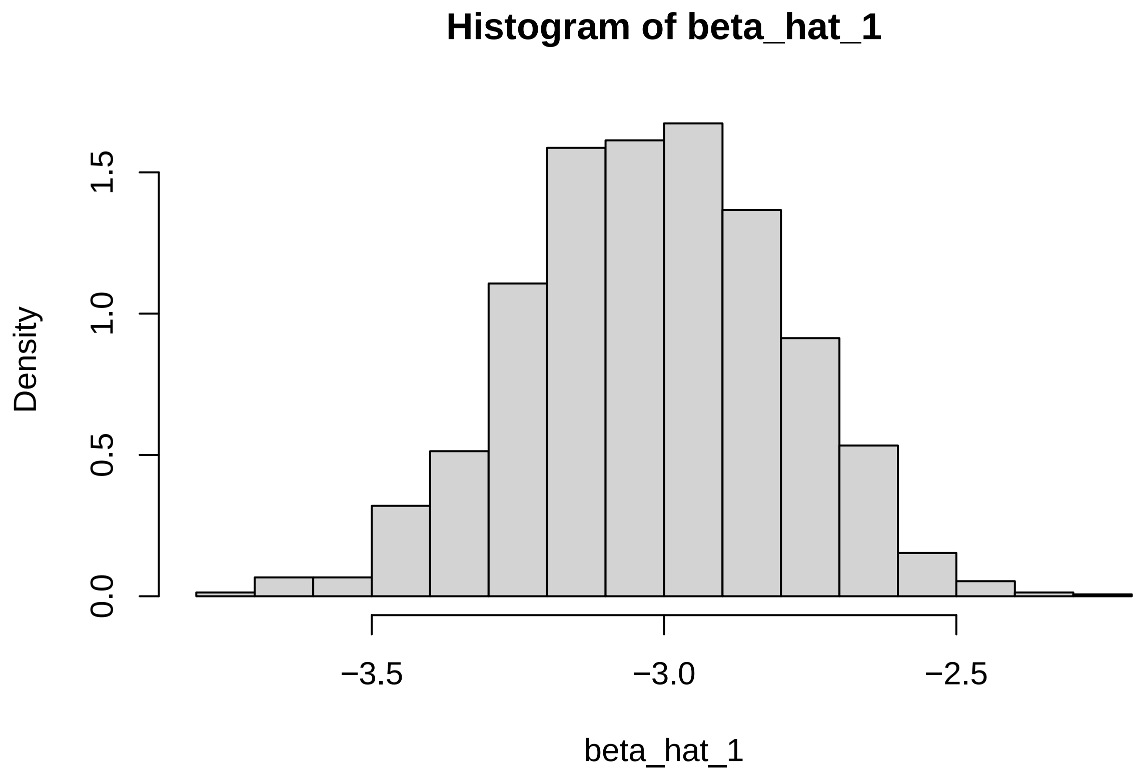
```r
beta_hat_1 = rep(0, 1500)
for (i in 1:1500) {
  data = sim_slr(x, 5, -3, sqrt(10.24))
  model = lm(data$response ~ data$predictor)
  beta_hat_1[i] = model$coefficients[2]
}
```

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

The mean of $\hat{\beta}_1$ is -3.01762. The standard deviation of $\hat{\beta}_1$ is 0.2223697. This shows that the estimator is indeed unbiased and its expected value is the true value of the parameter.

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

```r
hist(beta_hat_1, prob=TRUE)
```



The histogram clearly exhibits a normal distribution centered around -3, as expected.

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.

```r
birthday = 19991124
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.

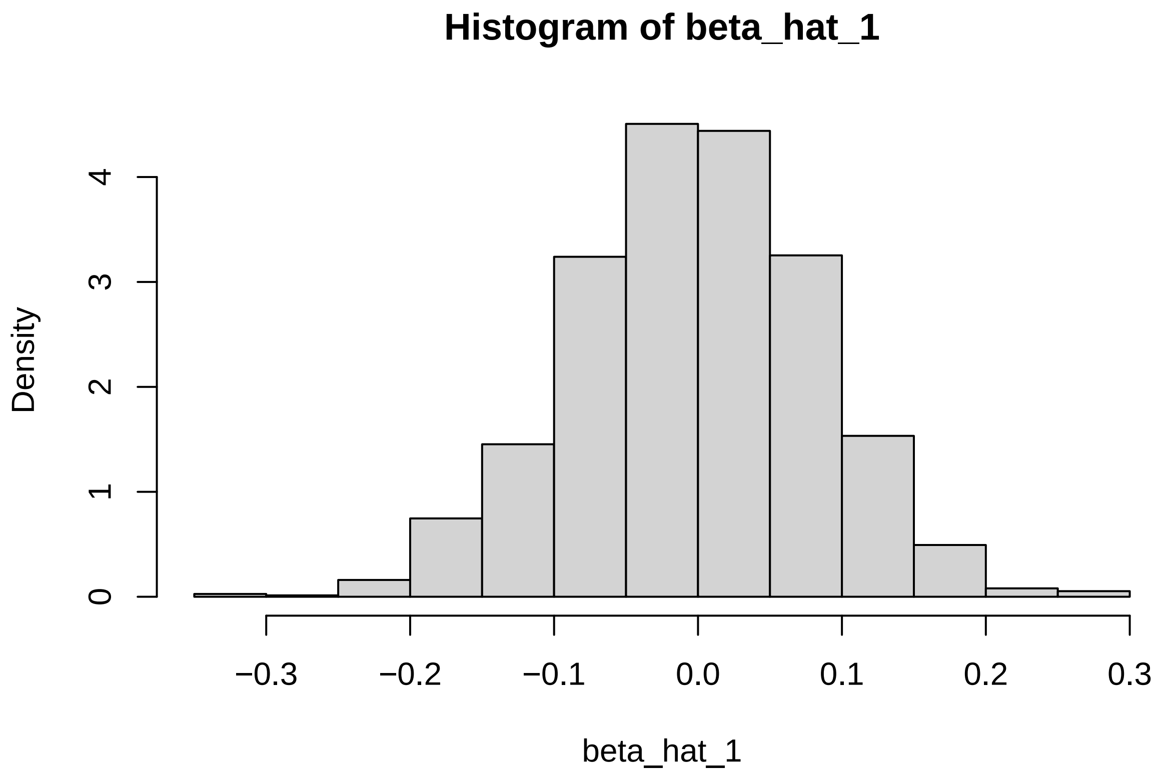
```r
x = runif(n = 75, 0, 10)
```

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 $\beta_1 = 0$.

```r
beta_hat_1 = rep(0, 1500)
for (i in 1:1500) {
  data = sim_slr(x, 3, 0, 2)
  model = lm(data$response ~ data$predictor)
  beta_hat_1[i] = model$coefficients[2]
}
```

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

```r
hist(beta_hat_1, prob=TRUE)
```



The histogram is again centered around 0, the true value of the parameter.

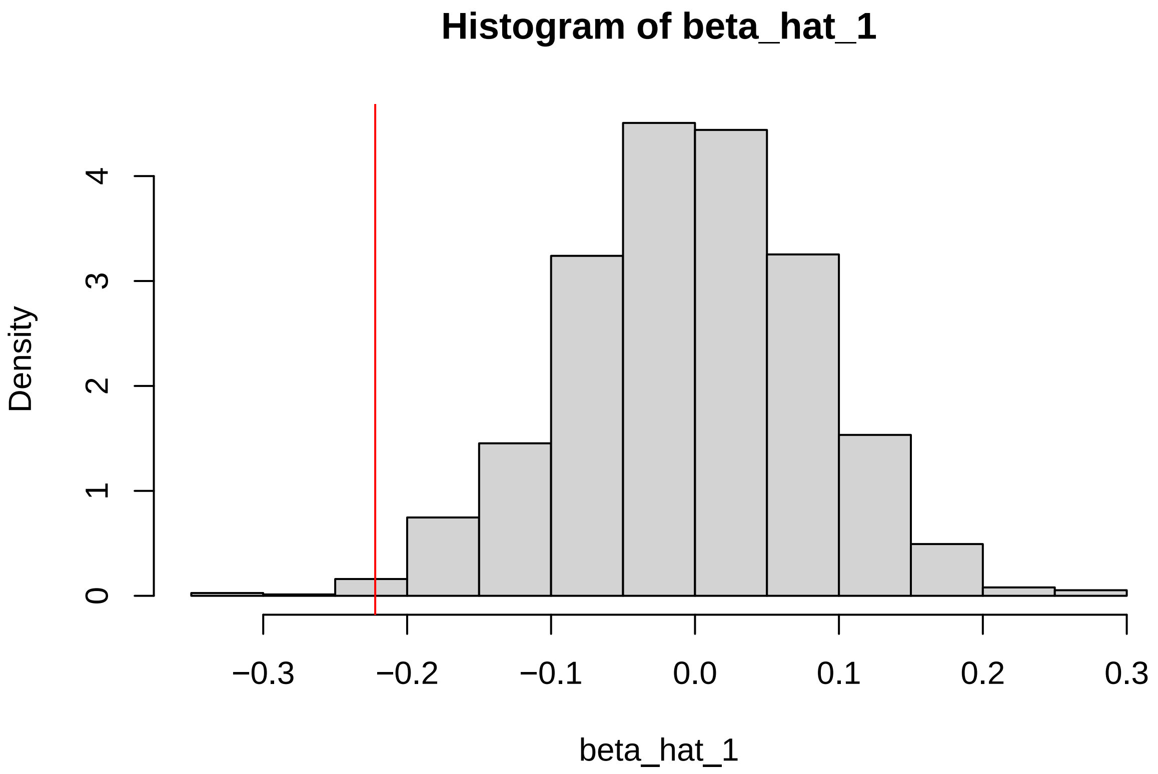
(c) Import the data in `skeptic.csv` and fit a SLR model. The variable names in `skeptic.csv` follow the same convention as those returned by `sim_slr()`. Extract the fitted coefficient for β_1.

```
data = read.csv(file = 'skeptic.csv')
beta_1_hat = lm(data$response ~ data$predictor)$coefficients[2]
```

The fitted coefficient for 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.

```
hist(beta_hat_1, prob=TRUE)
abline(v=beta_1_hat, 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.

```
proportion = sum(beta_hat_1 < beta_1_hat) / length(beta_hat_1)
proportion_times_two = proportion * 2
```

The proportion is 0.006; when multiplied by 2 it is 0.012.

(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.

If we were to state our null hypothesis as that $\hat{\beta}_1$ is equal to 0 and obtain the value of -0.2221927, we would only have a 0.6% chance of obtaining a value at least as extreme as -0.2221927. This is much lower than the common p-value of 5%, meaning that an observation like this would make us reject the null hypothesis. This is evidence that the data in skeptic.csv was likely not generated from 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.

```
wind = lm(Ozone$ozone ~ Ozone$wind)
humidity = lm(Ozone$ozone ~ Ozone$humidity)
temperature = lm(Ozone$ozone ~ Ozone$temp)

rmse = function(y, y_hat) {
  sqrt(sum((y - y_hat) ^ 2) / length(y))
}

mat.data  = c(
  rmse(Ozone$ozone, predict(wind)),
  summary(wind)$r.squared,
  rmse(Ozone$ozone, predict(humidity)),
  summary(humidity)$r.squared,
  rmse(Ozone$ozone, predict(temperature)),
  summary(temperature)$r.squared)

df = data.frame(matrix(mat.data, nrow=3, ncol=2, byrow=TRUE))
colnames(df) = c("RMSE", "R squared")
rownames(df) = c("Wind speed", "Humidity percentage", "Temperature")
knitr::kable(df)
```

	RMSE	R squared
Wind speed	7.961695	0.0001402
Humidity percentage	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.

Temperature is the best predictor of Ozone. It has the smallest standard deviation of residuals and explains the highest proportion of variation in Ozone measurements.
