Week 2 - Homework

STAT 420, Summer 2019, D. Unger

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:

```r
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:

```r
coef(cat_model)
```

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

- The **estimated** mean cat heart weight (in grams) for a body weight of 0 kg is $\hat{\beta}_0 = -0.3566624$ grams. (See below. This is an example of extrapolation.)

- For each additional kg of body weight, the mean heart weight increases by β_1 grams.

(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.

Solution:

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

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

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

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

We predict a heart weight of 10.5353069 **grams.** Since 2.7 is inside the data range, we are interpolating, thus we are somewhat confident in our prediction. We will see later how to put bounds around this value that quantify our certainty.

(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.

Solution:

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

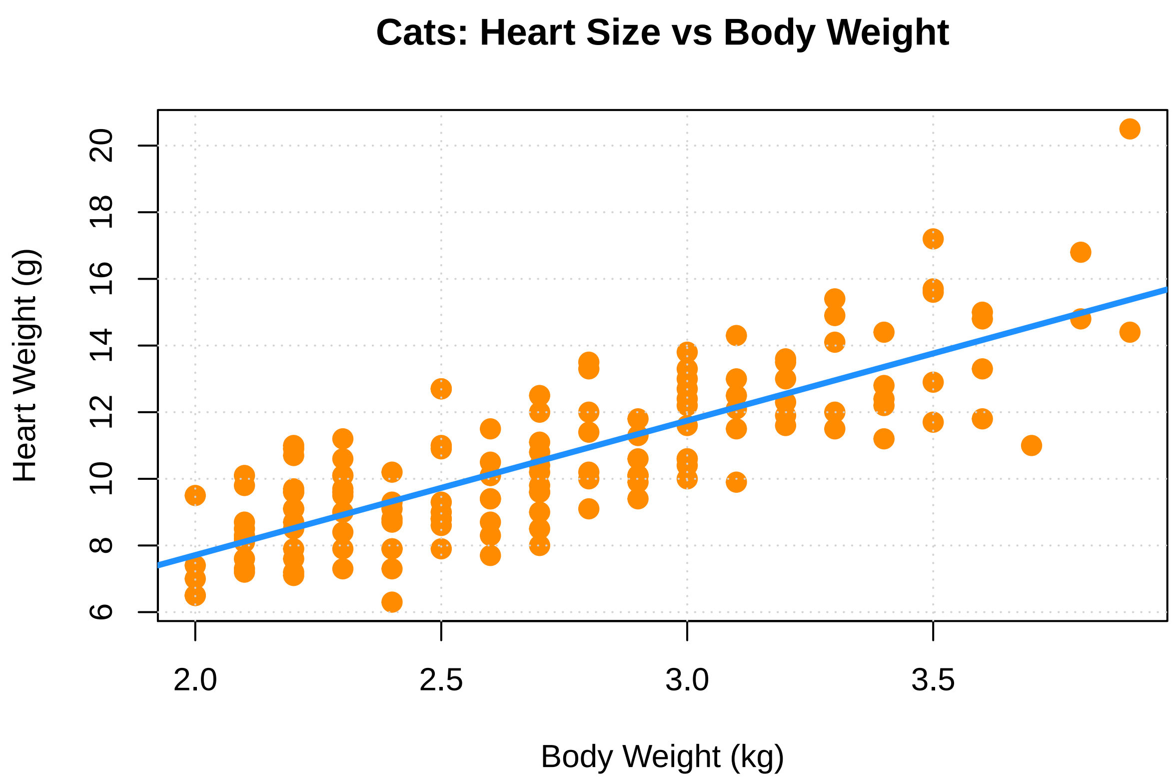
```
##        1
## 17.39321
```

We predict a heart weight of 17.3932134 **grams.** Since 4.4 is outside the data range, we are extrapolating, so we are **not** confident in our prediction. We will be able to put bounds on this, like the previous prediction; however, they will be rather meaningless.

(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.

Solution:

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



(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:

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

```
## [1] 0.6466209
```

This model has $R^2 = 0.6466$ which means that 64.66% of the observed variation in heart weight is explained by the linear relationship with body weight.

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_est = function(fitted_vals, actual_vals, mle = FALSE) {
  n = length(fitted_vals) - 2 * !mle
  sqrt(sum((actual_vals - fitted_vals) ^ 2) / n)
}
```

(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_est(fitted_vals = fitted(cat_model), actual_vals = cats$Hwt)
```

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

$$s_e = 1.4523733$$

Using the linear model (`cat_model`) from Exercise 1 and some cat's body weight, an estimate of a cat's heart weight will typically be off by about 1.4523733 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)**.

Solution:

```
get_sd_est(fitted_vals = fitted(cat_model), actual_vals = cats$Hwt, mle = TRUE)
```

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

$$\hat{\sigma} = 1.4422522$$

Using the linear model (`cat_model`) from Exercise 1 and some cat's body weight, an estimate of a cat's heart weight will typically be off by about 1.4422522 grams.

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

Solution:

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

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

We see that this value matches s_e.

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:

```
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 = 3.2)
```

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

Solution:

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

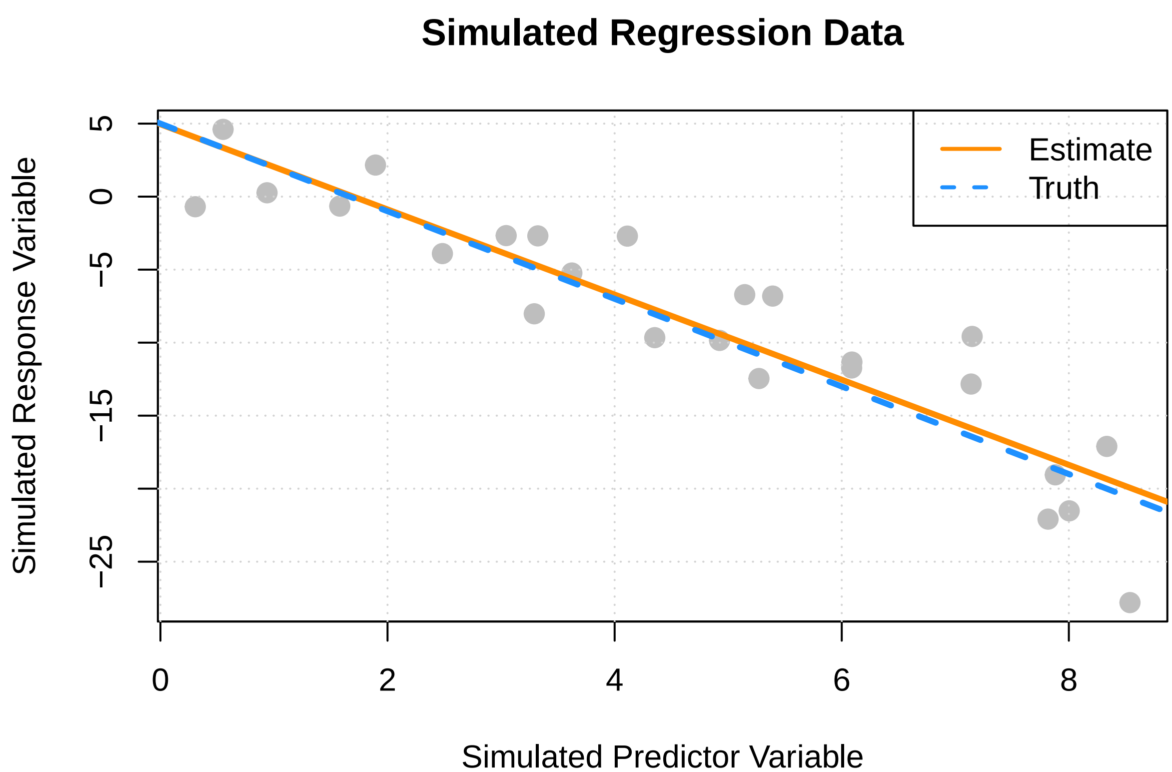
```
## (Intercept)   predictor
##    4.954205   -2.915525
```

Yes, they are somewhat close to the known parameters of the model, $\beta_0 = 5$ and $\beta_1 = -3$.

(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:

```r
plot(response ~ predictor, data = sim_data,
     xlab = "Simulated Predictor Variable",
     ylab = "Simulated Response Variable",
     main = "Simulated Regression Data",
     pch  = 20,
     cex  = 2,
     col  = "grey")
grid()
abline(sim_model, lwd = 3, lty = 1, col = "darkorange")
abline(5, -3, lwd = 3, lty = 2, col = "dodgerblue")
legend("topright", c("Estimate", "Truth"), lty = c(1, 2), lwd = 2,
       col = c("darkorange", "dodgerblue"))
```



(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:

```
beta_hat_1 = rep(0, 1500)
for (i in seq_along(beta_hat_1)) {
  sim_data = sim_slr(x = x, beta_0 = 5, beta_1 = -3, sigma = 3.2)
  model = lm(response ~ predictor, data = sim_data)
  beta_hat_1[i] = coef(model)[2]
}
```

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

Solution:

```
c(mean(beta_hat_1), sd(beta_hat_1))
```

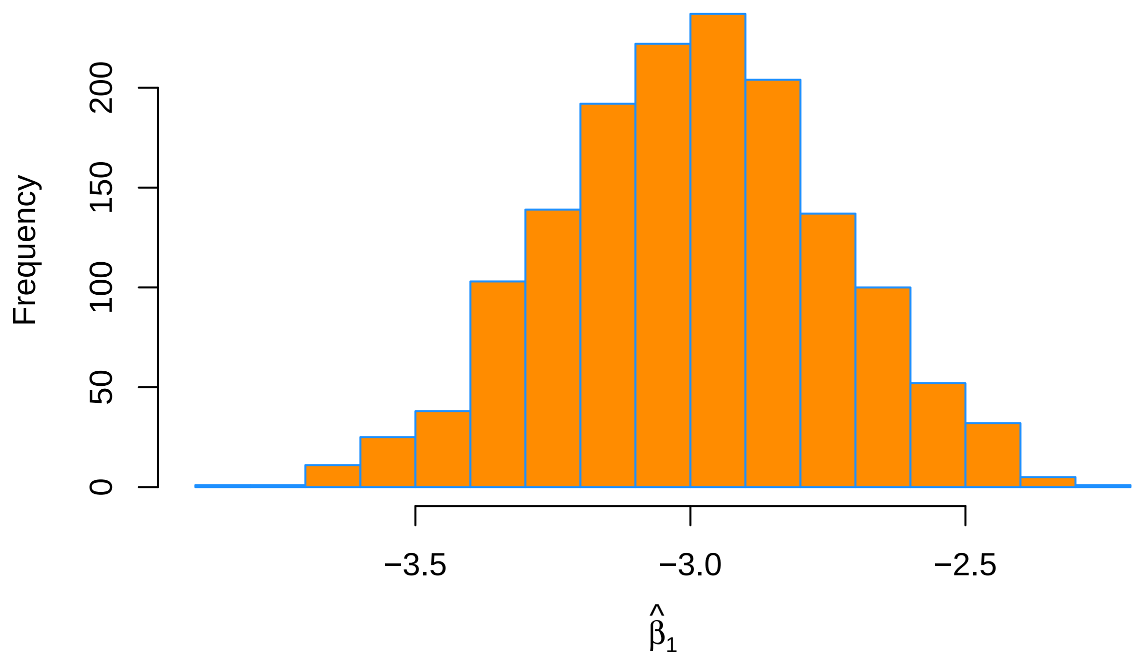
```
## [1] -2.9959160  0.2501589
```

The mean of `beta_hat_1` appears very similar to the parameter $\beta_1 = -3$.

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

Solution:

```
hist(beta_hat_1,
     main   = "",
     xlab   = expression(hat(beta)[1]),
     col    = "darkorange",
     border = "dodgerblue")
```



This histogram appears to be that of a normal distribution with a mean close to -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.

```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 $\beta_1 = 0$.

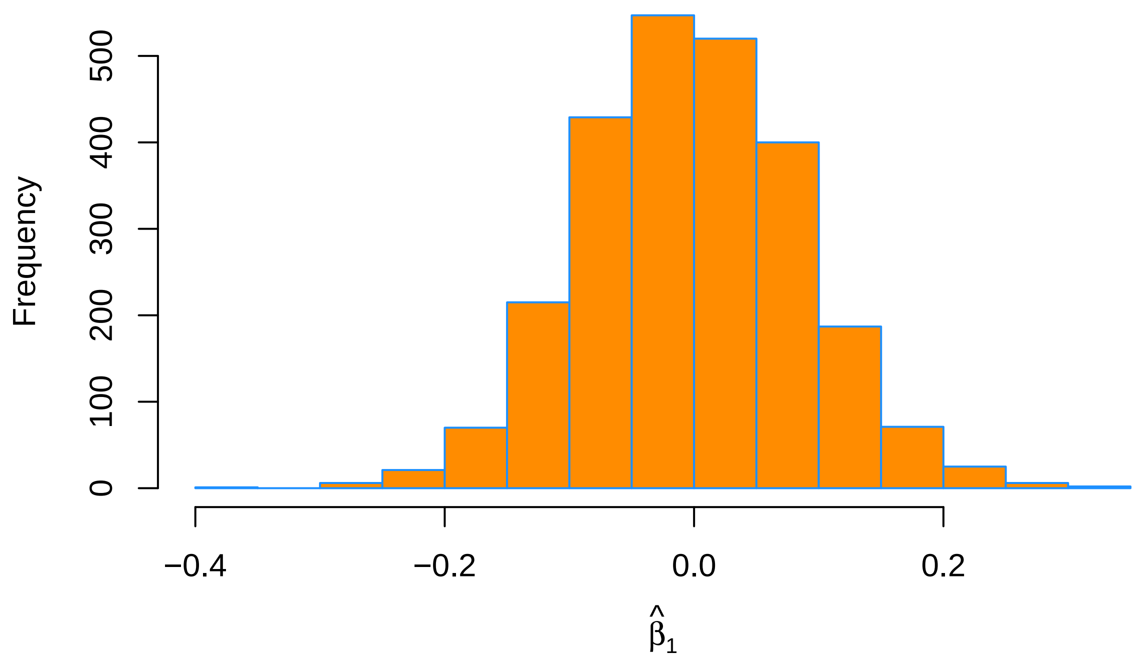
Solution:

```r
beta_hat_1 = rep(0, 2500)
for (i in seq_along(beta_hat_1)) {
  sim_data = sim_slr(x = x, beta_0 = 3, beta_1 = 0, sigma = 2)
  model = lm(response ~ predictor, data = sim_data)
  beta_hat_1[i] = coef(model)[2]
}
```

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

Solution:

```r
hist(beta_hat_1,
     main   = "",
     xlab   = expression(hat(beta)[1]),
     col    = "darkorange",
     border = "dodgerblue")
```



This histogram appears to be that of a normal distribution with a mean close to 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 β_1.

Solution:

```
# how the data was generated
set.seed(42)
x = runif(n = 75, 0, 10)
skeptic = sim_slr(x = x, beta_0 = 3.5, beta_1 = -0.25, sigma = 2)
write.csv(skeptic, file = "skeptic.csv", row.names = FALSE)
```

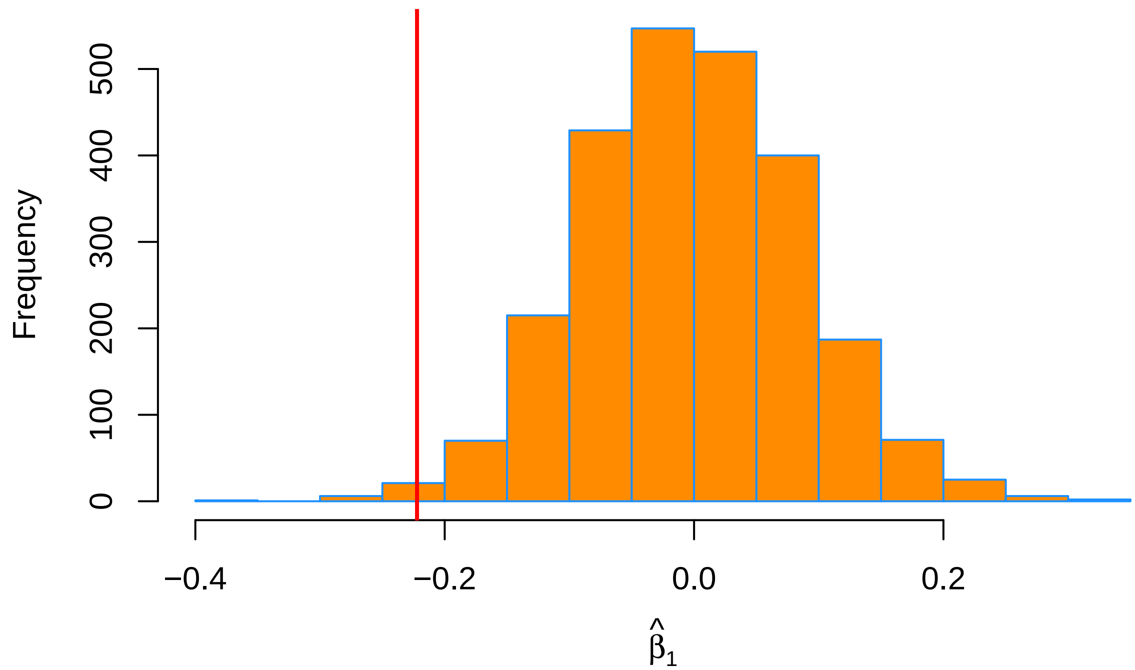
```
skeptic = read.csv("skeptic.csv")
skeptic_model = lm(response ~ predictor, data = skeptic)
coef(skeptic_model)[2]
```

```
##  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.

Solution:

```
hist(beta_hat_1,
     main   = "",
     xlab   = expression(hat(beta)[1]),
     col    = "darkorange",
     border = "dodgerblue")
abline(v = coef(skeptic_model)[2], col = "red", lwd = 2)
```



(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:

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

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

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

Solution:

```
range(beta_hat_1)
```

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

While it is certainly possible, since -0.2221927 is within the range of $\hat{\beta}_1$ values that we simulated, it is one of the more extreme values, with only 0.52% of the simulated values being smaller. So, possible? Yes. Probable? Not exactly.

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.

Solution:

```r
# fit models
oz_mod_wnd = lm(ozone ~ wind, data = Ozone)
oz_mod_hum = lm(ozone ~ humidity, data = Ozone)
oz_mod_tmp = lm(ozone ~ temp, data = Ozone)
```

```r
# helper functions

get_rmse = function(model) {
  sqrt(mean(resid(model) ^ 2))
}

get_r2 = function(model) {
  summary(model)$r.squared
}
```

```r
# create list of model
mod_list = list(oz_mod_wnd, oz_mod_hum, oz_mod_tmp)

# obtain results
results = data.frame(
  Predictor = c("`wind`", "`humidity`", "`temp`"),
  RMSE = sapply(mod_list, get_rmse),
  R2 = sapply(mod_list, get_r2)
)

# create markdown table of results
knitr::kable(results)
```

Predictor	RMSE	R2
`wind`	7.961695	0.0001402
`humidity`	7.147822	0.1941105
`temp`	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:

The predictor `temp` since the model that uses `temp` obtains the lowest RMSE, which is essentially the square root of the average of the errors of the model. So, lower is better.

We note that this strategy of simply comparing RMSEs is somewhat flawed. It only works here because we are comparing three SLR models. When comparing more complicated models, we will need to be more careful. (We will use hold-out data to better detect "overfitting.")

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:

```
get_beta_no_int = function(x, y) {
  sum(x * y) / sum(x ^ 2)
}
```

(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:

To minimize the function

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

we take the first derivative with respect to β

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

Then solving

$$\frac{\partial f}{\partial \beta} = 0$$

for β gives

$$\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}.$$

Lastly, we verify that

$$\frac{\partial^2 f}{\partial \beta^2} = 2\sum_{i=1}^{n} x_i^2 > 0.$$

Thus $\hat{\beta}$ does minimize $f(\beta)$ therefore it is the least-squares estimator of β.

(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:

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

```
## [1] 3.907113
```

$$\hat{\beta} = 3.9071133$$

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

```r
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:

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

```
##
## Call:
## lm(formula = Hwt ~ 0 + Bwt, data = cats)
##
## Coefficients:
##   Bwt
## 3.907
```