Week 3 - Homework

STAT 420, Summer 2021, Haixu Leng (NetID: haixul2)

06/05/2021

Exercise 1 (Using `lm` for Inference)

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) Fit the following simple linear regression model in `R`. Use heart weight as the response and body weight as the predictor.

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Store the results in a variable called `cat_model`. Use a t test to test the significance of the regression. Report the following:

- The null and alternative hypotheses
- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.05$
- A conclusion in the context of the problem

When reporting these, you should explicitly state them in your document, not assume that a reader will find and interpret them from a large block of `R` output.

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
```

- Null hypothesis $H_0 : \beta_1 = 0, Y_i = \beta_0 + \epsilon_i$. Alternative hypothesis $H_1 : \beta_1 \neq 0, Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
- The t test value for H_0 is 16.1193908
- The p-value for the test is $6.9690446 \cdot 10^{-34}$.
- Since the p-value is much less than α, I can **reject** the null hypothesis (H_0).
- From the t test, we conclude that the null hypothesis can be rejected. So, I can say there is a **significant linear relationship** between body weight and heart weight of cats in the data.

(b) Calculate a 95% confidence interval for β_1. Give an interpretation of the interval in the context of the problem.

Solution:

```
confint(cat_model, level = 0.95)
```

```
##                 2.5 %   97.5 %
## (Intercept) -1.725163 1.011838
## Bwt          3.539343 4.528782
```

The confidence interval of β_1 is from 3.539343 to 4.5287824. **This means that I am 95% confident that for an increase of 1 kg in cat's body weight, the average increase in cat's heart weight is between 3.539343 g and 4.5287824 g**.

(c) Calculate a 90% confidence interval for β_0. Give an interpretation of the interval in the context of the problem.

Solution:

```
confint(cat_model, level = 0.90)[1,]
```

```
##        5 %       95 %
## -1.5028345  0.7895096
```

I am 90% confident that when the cat's body weight is 0 kg, the average cat's heart weight is between -1.5028345 g and 0.7895096 g. Of course, this does not make sense in the real world, because I don't expect to see cat with 0 kg body weight.

(d) Use a 90% confidence interval to estimate the mean heart weight for body weights of 2.1 and 2.8 kilograms. Which of the two intervals is wider? Why?

Solution:

```
new_bwt = data.frame(Bwt = c(2.1, 2.8))
(output = predict(cat_model, newdata = new_bwt, interval = c("confidence"), level = 0.9))
```

```
##         fit       lwr       upr
## 1  8.114869  7.787882  8.441856
## 2 10.938713 10.735843 11.141583
```

```r
# interval for 2.1 kg
output[1,3] - output[1,2]
```

```
## [1] 0.653974
```

```r
# interval for 2.8 kg
output[2,3] - output[2,2]
```

```
## [1] 0.4057402
```

```r
# group mean of the data
mean(cats$Bwt)
```

```
## [1] 2.723611
```

The interval for cat with a **2.1 kg body weight is wider**, because 2.1 kg further away from the mean (2.7236111 kg) of the data we have.

(e) Use a 90% prediction interval to predict the heart weight for body weights of 2.8 and 4.2 kilograms.

Solution:

```r
new_bwt = data.frame(Bwt = c(2.8, 4.2))
(output = predict(cat_model, newdata = new_bwt, interval = c("prediction"), level = 0.9))
```

```
##        fit       lwr      upr
## 1 10.93871  8.525541 13.35189
## 2 16.58640 14.097100 19.07570
```

When the body weight is 2.8 kg, the interval is between 8.5255411 g and 13.3518851 g. When the body weight is 4.2 kg, the interval is between 14.0970999 g and 19.0757019 g.

(f) Create a scatterplot of the data. Add the regression line, 95% confidence bands, and 95% prediction bands.

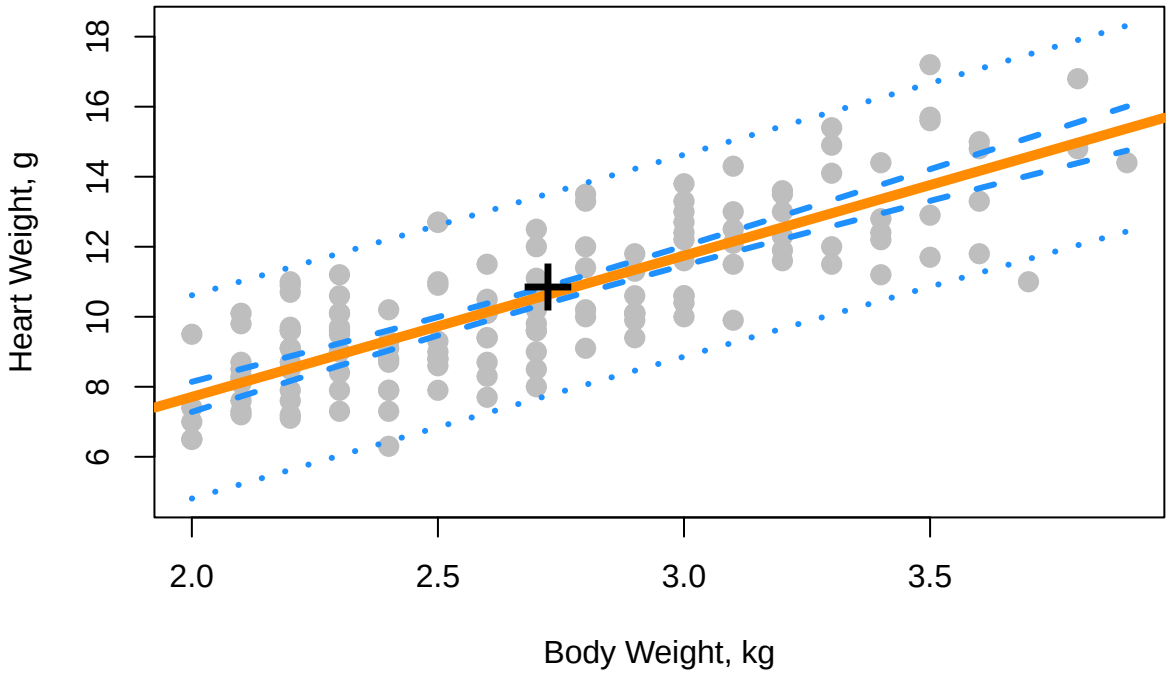
Solution:

```r
bwt_grid = seq(min(cats$Bwt), max(cats$Bwt), by = 0.01)
dist_ci_band = predict(cat_model,
                       newdata = data.frame(Bwt = bwt_grid),
                       interval = c("confidence"),
                       level = 0.95)

dist_pi_band = predict(cat_model,
                       newdata = data.frame(Bwt = bwt_grid),
                       interval = c("prediction"),
                       level = 0.95)

plot(Hwt ~ Bwt, data = cats,
     xlab = "Body Weight, kg",
     ylab = "Heart Weight, g",
     pch = 20,
```

```
     cex = 2,
     col = "grey",
     ylim = c(min(dist_pi_band), max(dist_pi_band)))
abline(cat_model, lwd = 5, col = "darkorange")
lines(bwt_grid, dist_ci_band[, "lwr"], col = "dodgerblue", lwd = 3, lty = 2)
lines(bwt_grid, dist_ci_band[, "upr"], col = "dodgerblue", lwd = 3, lty = 2)
lines(bwt_grid, dist_pi_band[, "lwr"], col = "dodgerblue", lwd = 3, lty = 3)
lines(bwt_grid, dist_pi_band[, "upr"], col = "dodgerblue", lwd = 3, lty = 3)
points(mean(cats$Bwt), mean(cats$Hwt), pch = "+", cex = 3)
```



(g) Use a t test to test:

- $H_0 : \beta_1 = 4$
- $H_1 : \beta_1 \neq 4$

Report the following:

- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.05$

When reporting these, you should explicitly state them in your document, not assume that a reader will find and interpret them from a large block of `R` output.

Solution:

```r
beta_h0 = 4
# display the summary of cat_model
#summary(cat_model)
estimate = summary(cat_model)$coefficients[2,1]
std_error = summary(cat_model)$coefficients[2,2]
# t value is
(t = (estimate - beta_h0) / std_error)
```

```
## [1] 0.1361084
```

```r
# find p-value
(p = pt(t, length(cats$Bwt) - 2))
```

```
## [1] 0.5540358
```

The t test for the $H_0 : \beta_1 = 4$ is:

$$t = \frac{\hat{\beta}_1 - \beta_{H0}}{SE[\hat{\beta}_1]}$$

where the values are calculated from the `summary(cat_model)`. So, t is 0.1361084. Based on this t value, p-value is calculated to be 0.5540358, and it is greater than α. So I fail to reject the hypothesis of H_0.

Exercise 2 (More `lm` for Inference)

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 re-perform the data cleaning done in the previous homework.

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

(a) Fit the following simple linear regression model in `R`. Use the ozone measurement as the response and wind speed as the predictor.

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Store the results in a variable called `ozone_wind_model`. Use a t test to test the significance of the regression. Report the following:

- The null and alternative hypotheses
- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.01$
- A conclusion in the context of the problem

When reporting these, you should explicitly state them in your document, not assume that a reader will find and interpret them from a large block of `R` output.

Solution:

```
ozone_wind_model = lm(ozone ~ wind, data = Ozone)
summary(ozone_wind_model)
```

```
##
## Call:
## lm(formula = ozone ~ wind, data = Ozone)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.730  -6.652  -1.752   4.687  26.359
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.8636     1.0856  10.928   <2e-16 ***
## wind         -0.0445     0.2032  -0.219    0.827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.985 on 342 degrees of freedom
## Multiple R-squared:  0.0001402,  Adjusted R-squared:  -0.002783
## F-statistic: 0.04795 on 1 and 342 DF,  p-value: 0.8268
```

- The null hypothesis is $H_0 : \beta_1 = 0$, and the alternative hypothesis is $H_1 : \beta_1 \neq 0$
- The value of the test statistic of β_1 is -0.2189811.
- The p-value of the test is 0.8267954.
- The p-value is much larger than α (0.01), so I fail to reject the null hypothesis
- I fail to reject the null hypothesis where $\beta_1 = 0$. So, there is no significant linear relationship between wind and ozone in the test data.

(b) Fit the following simple linear regression model in `R`. Use the ozone measurement as the response and temperature as the predictor.

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Store the results in a variable called `ozone_temp_model`. Use a t test to test the significance of the regression. Report the following:

- The null and alternative hypotheses
- The value of the test statistic
- The p-value of the test
- A statistical decision at $\alpha = 0.01$
- A conclusion in the context of the problem

When reporting these, you should explicitly state them in your document, not assume that a reader will find and interpret them from a large block of `R` output.

Solution:

```
ozone_temp_model = lm(ozone ~ temp, data = Ozone)
summary(ozone_temp_model)
```

```
##
## Call:
## lm(formula = ozone ~ temp, data = Ozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.7630  -3.7495  -0.1849   3.1099  15.1118
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.88480    1.19229  -12.48   <2e-16 ***
## temp          0.42968    0.01881   22.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.024 on 342 degrees of freedom
## Multiple R-squared:  0.6042, Adjusted R-squared:  0.603
## F-statistic: 522.1 on 1 and 342 DF,  p-value: < 2.2e-16
```

- The null hypothesis is $H_0 : \beta_1 = 0$, and the alternative hypothesis is $H_1 : \beta_1 \neq 0$
- The value of the test statistic of β_1 is 22.848962.
- The p-value of the test is $8.1537636 \cdot 10^{-71}$.
- The p-value is less than α (0.01), so I can reject the null hypothesis
- I can reject the null hypothesis where $\beta_1 = 0$. So, there is a significant linear relationship between temperature and ozone in the test data.

Exercise 3 (Simulating Sampling Distributions)

For this exercise we will simulate data from the following model:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Where $\epsilon_i \sim N(0, \sigma^2)$. Also, the parameters are known to be:

- $\beta_0 = -5$
- $\beta_1 = 3.25$
- $\sigma^2 = 16$

We will use samples of size $n = 50$.

(a) Simulate this model 2000 times. Each time use `lm()` to fit a simple linear regression model, then store the value of $\hat{\beta}_0$ and $\hat{\beta}_1$. Set a seed using **your** birthday before performing the simulation. Note, we are simulating the x values once, and then they remain fixed for the remainder of the exercise.

```r
birthday = 19900210
set.seed(birthday)
n = 50
x = seq(0, 10, length = n)
```

Solution:

```r
N = 2000

# function to simulate data
sim_slr = function(x, beta_0 = -5, beta_1 = 3.25, sigma = sqrt(16)) {
  n = length(x)
  epsilon = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + epsilon
  data.frame(predictor = x, response = y)
}

# an array to store lm outputs
beta_0_array = rep(0, N)
beta_1_array = rep(0, N)

for(i in 1:N){
  tmp = sim_slr(x)
  tmp_model = lm(response ~ predictor, tmp)
  #tmp_model
  #coefficients(tmp_model)
  beta_0_array[i] = coefficients(tmp_model)[[1]]
  beta_1_array[i] = coefficients(tmp_model)[[2]]
}
```

(b) Create a table that summarizes the results of the simulations. The table should have two columns, one for $\hat{\beta}_0$ and one for $\hat{\beta}_1$. The table should have four rows:

- A row for the true expected value given the known values of x
- A row for the mean of the simulated values
- A row for the true standard deviation given the known values of x
- A row for the standard deviation of the simulated values

Solution:

```r
# Build a data frame to store the table
models = data.frame(c("True expected value", "Simulated mean", "True standard deviation", "Simulated sta
colnames(models) = c("", "beta_0", "beta_1")

# fill in known expected values
models$beta_0[[1]] = -5
models$beta_1[[1]] = 3.25

# fill in the calculated mean values
models$beta_0[[2]] = mean(beta_0_array)
models$beta_1[[2]] = mean(beta_1_array)
```

```r
# fill in the true standard deviation
sigma = 4
Sxx = sum((x - mean(x))^2)
models$beta_0[[3]] = sigma * sqrt(1 / length(x) + mean(x)^2 / Sxx)
models$beta_1[[3]] = sigma / sqrt(Sxx)

# fill in the simulated standard deviation
models$beta_0[[4]] = sd(beta_0_array)
models$beta_1[[4]] = sd(beta_1_array)

knitr::kable(models)
```

	beta_0	beta_1
True expected value	-5.000000	3.2500000
Simulated mean	-4.986058	3.2453853
True standard deviation	1.114609	0.1920784
Simulated standard deviation	1.114372	0.1909282

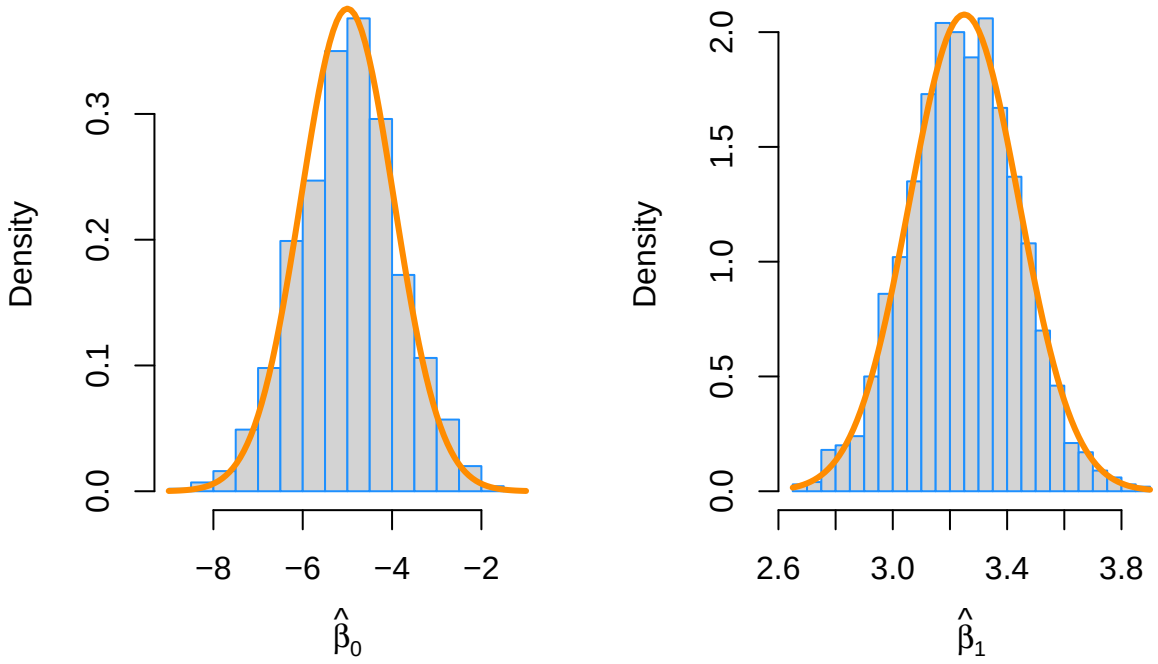
(c) Plot two histograms side-by-side:

- A histogram of your simulated values for $\hat{\beta}_0$. Add the normal curve for the true sampling distribution of $\hat{\beta}_0$.
- A histogram of your simulated values for $\hat{\beta}_1$. Add the normal curve for the true sampling distribution of $\hat{\beta}_1$.

Solution:

```r
# plot two histograms side by side
par(mfrow = c(1,2))
hist(beta_0_array, probability = TRUE, breaks = 20,
     xlab = expression(hat(beta)[0]), main = "", border = "dodgerblue")
curve(dnorm(x, mean = -5, sd = sigma * sqrt(1 / length(x) + mean(x)^2 / Sxx)),
      col = "darkorange", add = TRUE, lwd = 3)

hist(beta_1_array, probability = TRUE, breaks = 20,
     xlab = expression(hat(beta)[1]), main = "", border = "dodgerblue")
curve(dnorm(x, mean = 3.25, sd = sigma / sqrt(Sxx)),
      col = "darkorange", add = TRUE, lwd = 3)
```



Exercise 4 (Simulating Confidence Intervals)

For this exercise we will simulate data from the following model:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Where $\epsilon_i \sim N(0, \sigma^2)$. Also, the parameters are known to be:

- $\beta_0 = 5$
- $\beta_1 = 2$
- $\sigma^2 = 9$

We will use samples of size $n = 25$.

Our goal here is to use simulation to verify that the confidence intervals really do have their stated confidence level. Do **not** use the `confint()` function for this entire exercise.

(a) Simulate this model 2500 times. Each time use `lm()` to fit a simple linear regression model, then store the value of $\hat{\beta}_1$ and s_e. Set a seed using **your** birthday before performing the simulation. Note, we are simulating the x values once, and then they remain fixed for the remainder of the exercise.

```r
birthday = 19900210
set.seed(birthday)
n = 25
x = seq(0, 2.5, length = n)
```

Solution:

```r
N = 2500

# function to simulate data
sim_slr = function(x, beta_0 = 5, beta_1 = 2, sigma = sqrt(9)) {
  n = length(x)
  epsilon = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + epsilon
  data.frame(predictor = x, response = y)
}

# initialize two vectors to store beta_1_hat and se
beta_1_hat = rep(0, N)
SE = rep(0, N)

# run simulations
y_real = 5 + 2 * x
for(i in 1:N){
  tmp = sim_slr(x)
  tmp_model = lm(response ~ predictor, tmp)
  beta_1_hat[i] = coefficients(tmp_model)[[2]]
  SE[i] = sqrt(sum((y_real - tmp$response)^2) / (length(x) - 2))
}
```

The calculated values are stored in `beta_1_hat` and `SE`.

(b) For each of the $\hat{\beta}_1$ that you simulated, calculate a 95% confidence interval. Store the lower limits in a vector `lower_95` and the upper limits in a vector `upper_95`. Some hints:

- You will need to use `qt()` to calculate the critical value, which will be the same for each interval.
- Remember that `x` is fixed, so S_{xx} will be the same for each interval.
- You could, but do not need to write a `for` loop. Remember vectorized operations.

Solution:

```r
# calculate Sxx
Sxx = sum((x - mean(x))^2)

# initialize two vectors
lower_95 = rep(0, N)
upper_95 = rep(0, N)

# calculate the intervals of 95% confidence
alpha = 0.05
# get critical values of a t distribution
lower_crit = qt(alpha/2, length(x) -2, lower.tail = TRUE)
upper_crit = qt(alpha/2, length(x) -2, lower.tail = FALSE)
```

```
# calculate the standard error
STDERROR = SE / sqrt(Sxx)
lower_95 = beta_1_hat + lower_crit * STDERROR
upper_95 = beta_1_hat + upper_crit * STDERROR
```

The calculation of the lower and upper 95% confidence interval is based on foluma:

$$\hat{\beta}_1 \pm t_{\alpha/2,n-2} \cdot \frac{s_e}{S_{xx}}$$

s_e is calculated from the previous question, and S_{xx} is calculated accordingly.

(c) What proportion of these intervals contains the true value of β_1?

Solution:

If a simulated interval contains the true value, then we will have:

$$upper_{95\%} \geq \beta_1 \geq lower_{95\%}$$

```
# a boolean array where upper_95 > beta_true
beta_1_true = 2
less_than_upper = (upper_95 >= beta_1_true)
greater_than_lower = (lower_95 <= beta_1_true)

# logical AND the two boolean vectors, so both conditions are met
N_within_interval = sum("&"(less_than_upper, greater_than_lower))

# proportion is
N_within_interval / N
```

```
## [1] 0.9708
```

As shown above, the proportion that the intervals contain the true value of β_1 is **0.9708** with a 95% confidence.

(d) Based on these intervals, what proportion of the simulations would reject the test $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ at $\alpha = 0.05$?

Solution: To reject the null hypothesis, we need:

$$upper_{95\%} < 0$$

or

$$lower_{95\%} > 0$$

```
# a boolean array where upper_95 < 0 or lower_95 > 0
beta_1_true = 0
less_than_upper = (upper_95 < beta_1_true)
greater_than_lower = (lower_95 > beta_1_true)

# logical AND the two boolean vectors, so both conditions are met
N_within_interval = sum("|"(less_than_upper, greater_than_lower))

# proportion is
N_within_interval / N
```

```
## [1] 0.6456
```

From the calculation, we see that the proportion where the lower 95% end is larger than 0 is 0.6456. In these simulations, the null hypothesis can be rejected since $\beta_1 = 0$ is not within the 95% interval.

(e) For each of the $\hat{\beta}_1$ that you simulated, calculate a 99% confidence interval. Store the lower limits in a vector `lower_99` and the upper limits in a vector `upper_99`.

Solution: In this question, I will repeat the process for previous questions, by replacing 95% with 99%.

```
# calculate Sxx
Sxx = sum((x - mean(x))^2)

# initialize two vectors
lower_99 = rep(0, N)
upper_99 = rep(0, N)

# calculate the intervals of 99% confidence
alpha = 0.01
# get critical values of a t distribution
lower_crit = qt(alpha/2, length(x) -2, lower.tail = TRUE)
upper_crit = qt(alpha/2, length(x) -2, lower.tail = FALSE)
# calculate the standard error
STDERROR = SE / sqrt(Sxx)
lower_99 = beta_1_hat + lower_crit * STDERROR
upper_99 = beta_1_hat + upper_crit * STDERROR
```

(f) What proportion of these intervals contains the true value of β_1?

Solution:

If a simulated interval contains the true value, then we will have:

$$upper_{99\%} \geq \beta_1 \geq lower_{99\%}$$

```
# a boolean array where upper_99 > beta_true
beta_1_true = 2
less_than_upper = (upper_99 >= beta_1_true)
greater_than_lower = (lower_99 <= beta_1_true)

# logical AND the two boolean vectors, so both conditions are met
N_within_interval = sum("&"(less_than_upper, greater_than_lower))

# proportion is
N_within_interval / N
```

```
## [1] 0.9964
```

As shown above, the proportion that the intervals contain the true value of β_1 is **0.9964** with a 99% confidence.

(g) Based on these intervals, what proportion of the simulations would reject the test $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ at $\alpha = 0.01$?

Solution: To reject the null hypothesis, we need:

$$upper_{99\%} < 0$$

or

$$lower_{99\%} > 0$$

```
# a boolean array where upper_99 < 0 or lower_99 > 0
beta_1_true = 0
less_than_upper = (upper_99 < beta_1_true)
greater_than_lower = (lower_99 > beta_1_true)

# logical AND the two boolean vectors, so both conditions are met
N_within_interval = sum("|"(less_than_upper, greater_than_lower))

# proportion is
N_within_interval / N
```

```
## [1] 0.37
```

From the calculation, we see that the proportion where the lower 99% end is larger than 0 is 0.37. In these simulations, the null hypothesis can be rejected since $\beta_1 = 0$ is not within the 99% interval. ***

Exercise 5 (Prediction Intervals "without" `predict`)

Write a function named `calc_pred_int` that performs calculates prediction intervals:

$$\hat{y}(x) \pm t_{\alpha/2, n-2} \cdot s_e \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{S_{xx}}}.$$

for the linear model

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i.$$

(a) Write this function. You may use the `predict()` function, but you may **not** supply a value for the `level` argument of `predict()`. (You can certainly use `predict()` any way you would like in order to check your work.)

The function should take three inputs:

- `model`, a model object that is the result of fitting the SLR model with `lm()`
- `newdata`, a data frame with a single observation (row)
  - This data frame will need to have a variable (column) with the same name as the data used to fit `model`.
- `level`, the level (0.90, 0.95, etc) for the interval with a default value of `0.95`

The function should return a named vector with three elements:

- `estimate`, the midpoint of the interval
- `lower`, the lower bound of the interval

- `upper`, the upper bound of the interval

Solution:

```
# write the function named calc_pred
calc_pred_int = function(model, newdata, level = 0.95){
  # calculate the parameters
  beta_0 = coefficients(model)[[1]]
  beta_1 = coefficients(model)[[2]]
  n = length(model$residuals)
  t_crit = qt((1 - level) / 2, n - 2, lower.tail = FALSE)
  y_i = model$model[,1]
  x_i = model$model[,2]
  y_hat = beta_0 + x_i * beta_1
  Sxx = sum((x_i - mean(x_i))^2)
  se = sqrt(sum((y_i - y_hat)^2) / (n - 2))
  x = newdata[[1]]
  standard_error = se * sqrt(1 + 1 / n + (x - mean(x_i))^2 / Sxx)

  # estimate
  estimate = beta_0 + beta_1 * x

  # lower
  lower = estimate - t_crit * standard_error

  # upper
  upper = estimate + t_crit * standard_error

  output = data.frame(c(estimate), c(lower), c(upper))
  colnames(output) = c("estimate", "lower", "upper")
  return(output)
}
```

(b) After writing the function, run this code:

```
newcat_1 = data.frame(Bwt = 4.0)
calc_pred_int(cat_model, newcat_1)
```

Solution:

```
newcat_1 = data.frame(Bwt = 4.0)
calc_pred_int(cat_model, newcat_1)
```

```
##   estimate    lower  upper
## 1 15.77959 12.83018 18.729
```

```
# result from predict as a comparison
predict(cat_model, newcat_1, interval = c("prediction"), level = 0.95)
```

```
##        fit      lwr    upr
## 1 15.77959 12.83018 18.729
```

The result from `predict()` is **identical** to the home-made function.

(c) After writing the function, run this code:

```
newcat_2 = data.frame(Bwt = 3.3)
calc_pred_int(cat_model, newcat_2, level = 0.90)
```

Solution:

```
newcat_2 = data.frame(Bwt = 3.3)
calc_pred_int(cat_model, newcat_2, level = 0.90)
```

```
##   estimate    lower   upper
## 1 12.95574 10.53099 15.3805
```

```
# result from predict as a comparison
predict(cat_model, newcat_2, interval = c("prediction"), level = 0.90)
```

```
##        fit      lwr     upr
## 1 12.95574 10.53099 15.3805
```

The result from `predict()` is **identical** to the home-made function.