Week 3 - Homework

STAT 420, Summer 2018, Unger

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
## -3.569 -0.963 -0.092  1.043  5.124
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.357      0.692   -0.52     0.61
## Bwt            4.034      0.250   16.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.45 on 142 degrees of freedom
## Multiple R-squared:  0.647,  Adjusted R-squared:  0.644
## F-statistic:  260 on 1 and 142 DF,  p-value: <2e-16
```

```
summary(cat_model)$coefficients["Bwt", "t value"] #test statistic
```

```
## [1] 16.12
```

```
summary(cat_model)$coefficients["Bwt", "Pr(>|t|)"] #p-value
```

```
## [1] 6.969e-34
```

- $H_0 : \beta_1 = 0$
- $H_1 : \beta_1 \neq 0$
- Test statistic: $t = 16.1194$
- P-value: 6.969×10^{-34}
- Decision: **Reject** H_0 at $\alpha = 0.05$.
- Conclusion: There is a linear relationship between heart weight and body weight.

(b) Calculate a 90% confidence interval for β_1. Give an interpretation of the interval in the context of the problem.

Solution:

```
confint(cat_model, "Bwt", level = 0.90)
```

```
##      5 %  95 %
## Bwt 3.62 4.448
```

A 90% confidence interval for β_1 is given by

$$(3.6197, 4.4484).$$

Notice that this interval does **not** contain 0, which suggests that 0 is not a plausible value for β_1. This notion matches our result from the previous hypothesis test.

Interpretation: We are 90% confident that given a 1-kilogram increase in body weight, the average increase in heart weight will be between 3.6197 and 4.4484 grams.

(c) Calculate a 99% confidence interval for β_0. Give an interpretation of the interval in the context of the problem.

Solution:

```
confint(cat_model, "(Intercept)", level = 0.99)
```

```
##               0.5 % 99.5 %
## (Intercept) -2.164  1.451
```

A 90% confidence interval for β_0 is given by

$$(-2.1641, 1.4508).$$

Interpretation: Mathematically, we are 99% confident that for a body weight of 0 kilograms, the average heart weight will be between -2.1641 and 1.4508 grams.

However, this confidence interval has no **practical** explanation because it is nonsense to consider the heart weight of a cat that weighs 0 kilograms.

(d) Use a 99% 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_body_weights = data.frame(Bwt = c(2.1, 2.8))
(heart_weight_ci = predict(cat_model, newdata = new_body_weights,
                           interval = c("confidence"), level = 0.99))
```

```
##      fit    lwr    upr
## 1  8.115  7.599  8.631
## 2 10.939 10.619 11.259
```

We are 99% confident that the mean heart weight for a body weight of 2.1 kilograms is in the interval

$$(7.5992, 8.6305).$$

We are 99% confident that the mean heart weight for a body weight of 2.8 kilograms is in the interval

$$(10.6188, 11.2586).$$

```
mean(cats$Bwt)
```

```
## [1] 2.724
```

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

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

The interval for a body weight of 2.1 kilograms is larger since it is further from the sample mean body weight. Also, note that both body weights fall within the range of observed body weights.

```
heart_weight_ci = unname(heart_weight_ci) # removes name information for future display
heart_weight_ci[, 2:3]
```

```
##        [,1]   [,2]
## [1,]  7.599  8.631
## [2,] 10.619 11.259
```

```
diff(heart_weight_ci[1, 2:3])
```

```
## [1] 1.031
```

```
diff(heart_weight_ci[2, 2:3])
```

```
## [1] 0.6398
```

```
diff(heart_weight_ci[1, 2:3]) < diff(heart_weight_ci[2, 2:3])
```

```
## [1] FALSE
```

(e) Use a 99% prediction interval to predict the heart weight for body weights of 2.8 and 4.2 kilograms.

Solution:

```
new_body_weights = data.frame(Bwt = c(2.8, 4.2))
(heart_weight_pi = predict(cat_model, newdata = new_body_weights,
                           interval = c("prediction"), level = 0.99))
```

```
##     fit    lwr   upr
## 1 10.94  7.133 14.74
## 2 16.59 12.661 20.51
```

We are 99% confident that a *new* **observation** of heart weight for a body weight of 2.8 kilograms is in the interval

$$(7.1332, 14.7442).$$

We are 99% confident that a *new* **observation** of heart weight for a body weight of 4.2 kilograms is in the interval

$$(12.6609, 20.5119).$$

Note that the prediction interval for a body weight of 2.8 kilograms is wider than the confidence interval for the same body weight found in **(d)**.

```
heart_weight_pi = unname(heart_weight_pi)
diff(heart_weight_ci[2, 2:3]) < diff(heart_weight_pi[1, 2:3])
```

```
## [1] TRUE
```

(f) Create a scatterplot of the data. Add the regression line, 90% confidence bands, and 90% prediction bands.

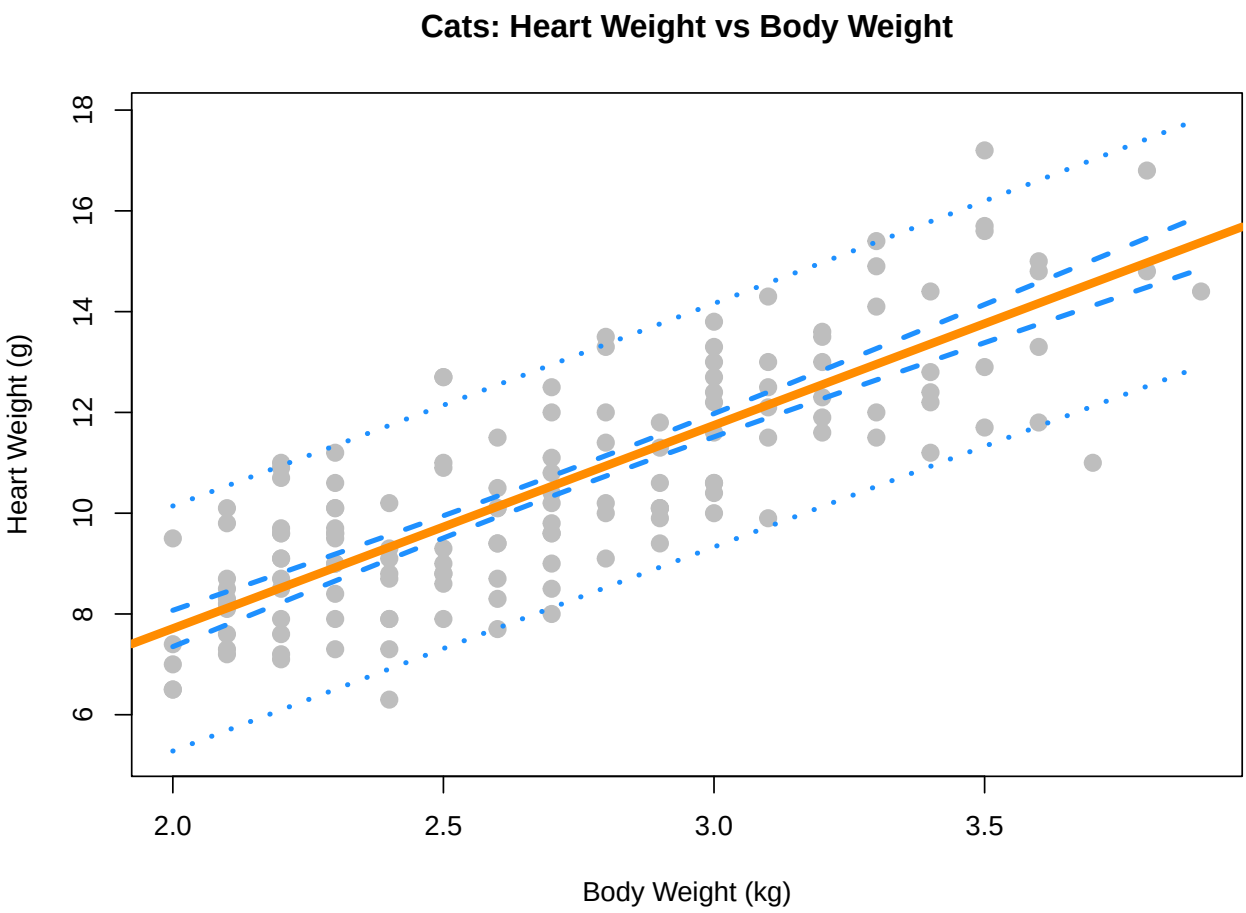
Solution:

```
bw_grid = seq(min(cats$Bwt), max(cats$Bwt), by = 0.01)

hw_ci_band = predict(cat_model, newdata = data.frame(Bwt = bw_grid),
                     interval = "confidence", level = 0.90)
hw_pi_band = predict(cat_model, newdata = data.frame(Bwt = bw_grid),
                     interval = "prediction", level = 0.90)

plot(Hwt ~ Bwt, data = cats,
     xlab = "Body Weight (kg)",
     ylab = "Heart Weight (g)",
     main = "Cats: Heart Weight vs Body Weight",
     pch  = 20,
     cex  = 2,
     col  = "grey",
     ylim = c(min(hw_pi_band), max(hw_pi_band)))

abline(cat_model, lwd = 5, col = "darkorange")
lines(bw_grid, hw_ci_band[,"lwr"], col = "dodgerblue", lwd = 3, lty = 2)
lines(bw_grid, hw_ci_band[,"upr"], col = "dodgerblue", lwd = 3, lty = 2)
lines(bw_grid, hw_pi_band[,"lwr"], col = "dodgerblue", lwd = 3, lty = 3)
lines(bw_grid, hw_pi_band[,"upr"], col = "dodgerblue", lwd = 3, lty = 3)
```



Notice that, while the vast majority of the data points are within the prediction bands, very few points are within the confidence bands.

(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:

```
B = coefficients(cat_model)[2]
SE = summary(cat_model)$coefficients["Bwt", "Std. Error"]
TS = (B - 4) / SE
n = length(resid(cat_model))
p_val = 2 * pt(abs(TS), df = n - 2, lower.tail = FALSE)
```

- Test statistic: $t = 0.1361$
- P-value: 0.8919
- Decision: **Fail to reject** H_0 at $\alpha = 0.05$.

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.

```
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)$coefficients["wind", "t value"] # test statistic
```

```
## [1] -0.219
```

```
summary(ozone_wind_model)$coefficients["wind", "Pr(>|t|)"] # p-value
```

```
## [1] 0.8268
```

- $H_0 : \beta_1 = 0$, $Y_i = \beta_0 + \epsilon_i$
- $H_1 : \beta_1 \neq 0$, $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
- Test statistic: $t = -0.219$
- (Alternative) Test statistic: $F = 0.048$
- P-value: 0.8268
- Decision: **Fail to Reject** H_0 at $\alpha = 0.01$.
- Conclusion: There is **not** a linear relationship between ozone and wind speed.

(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)$coefficients["temp", "t value"] # test statistic
```

```
## [1] 22.85
```

```
summary(ozone_temp_model)$coefficients["temp", "Pr(>|t|)"] # p-value
```

```
## [1] 8.154e-71
```

- $H_0 : \beta_1 = 0$, $Y_i = \beta_0 + \epsilon_i$
- $H_1 : \beta_1 \neq 0$, $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
- Test statistic: $t = 22.849$
- (Alternative) Test statistic: $F = 522.0751$
- P-value: 8.1538×10^{-71}
- Decision: **Reject** H_0 at $\alpha = 0.01$.
- Conclusion: There is a linear relationship between ozone and temperature.

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.

```
birthday = 18760613
set.seed(birthday)
n = 50
x = seq(0, 10, length = n)
```

Solution:

```
beta_0    = -5
beta_1    = 3.25
sigma     = 4
true_line = beta_0 + beta_1 * x

num_sim   = 2000
beta_hats = matrix(0, num_sim, 2)
for (i in 1:num_sim) {
  y = true_line + rnorm(n, mean = 0, sd = sigma)
  beta_hats[i, ] = coef(lm(y ~ x))
}

beta_0_hats = beta_hats[, 1]
beta_1_hats = beta_hats[, 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

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

Value	$\hat{\beta}_0$	$\hat{\beta}_1$
E[]	-5	3.25
`mean()`	-4.9816	3.2432
SD[]	1.1146	0.1921
`sd()`	1.1285	0.1949

(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 $\hat{\beta}_1$.

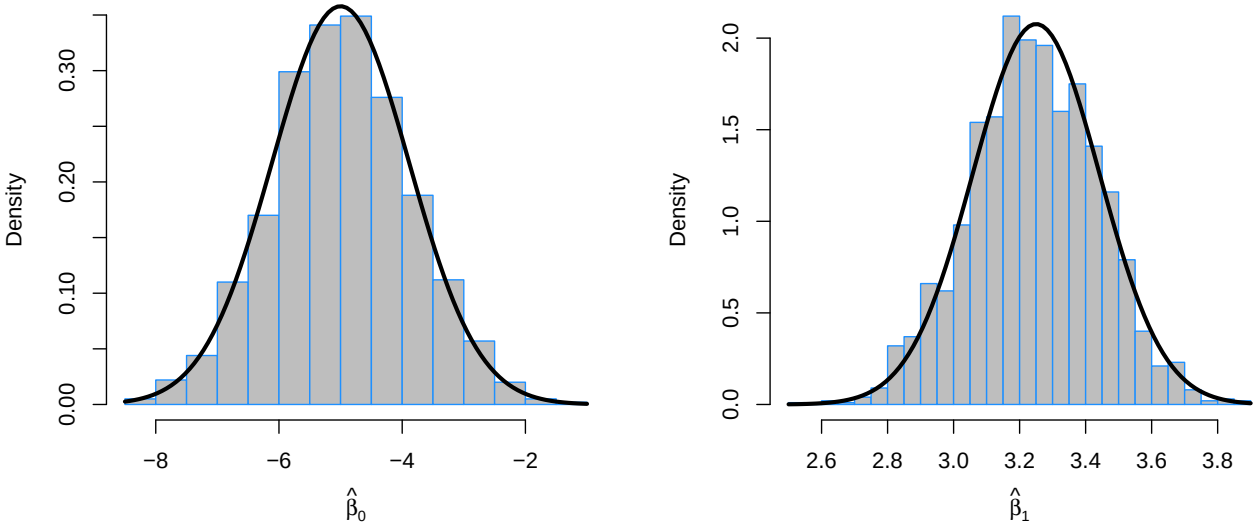
Solution:

```
# setup plotting
par(mfrow = c(1, 2))

# create plot for beta-0
hist(beta_0_hats, breaks = 25, col = "grey", border = "dodgerblue",
     prob = TRUE, xlab = expression(hat(beta)[0]), main = "")
e_beta_0_hat  = beta_0
sd_beta_0_hat = sigma * sqrt(1 / n + mean(x) ^ 2 / Sxx)
```

```r
curve(dnorm(x, mean = e_beta_0_hat, sd = sd_beta_0_hat), add = TRUE, lwd = 3)

# create plot for beta-1
hist(beta_1_hats, breaks = 25, col = "grey", border = "dodgerblue",
     prob = TRUE, xlab = expression(hat(beta)[1]), main = "")
e_beta_1_hat  = beta_1
sd_beta_1_hat = sigma / sqrt(Sxx)
curve(dnorm(x, mean = e_beta_1_hat, sd = sd_beta_1_hat), 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 = 18760613
set.seed(birthday)
n = 25
x = seq(0, 2.5, length = n)
```

Solution:

```
beta_0    = 5
beta_1    = 2
sigma     = 3
true_line = beta_0 + beta_1 * x

num_sim    = 2500
beta_hat_1 = rep(0, num_sim)
s_e        = rep(0, num_sim)

for (i in 1:num_sim) {
  y             = true_line + rnorm(n, 0, sigma)
  beta_hat_1[i] = coef(lm(y ~ x))[2]
  s_e[i]        = summary((lm(y ~ x)))$sigma
}
```

(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:

Recall,

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

```
alpha = 0.05
t_crit_95 = -qt(alpha / 2, df = n - 2)
Sxx = sum((x - mean(x)) ^ 2)

lower_95 = beta_hat_1 - t_crit_95 * s_e / sqrt(Sxx)
upper_95 = beta_hat_1 + t_crit_95 * s_e / sqrt(Sxx)
```

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

Solution:

```
mean(lower_95 < 2 & 2 < upper_95)
```

```
## [1] 0.9448
```

Unsurprisingly, the result is near 95%. (If we increased the number of simulations, this value should move closer to 0.95.)

(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:

```
1 - mean(lower_95 < 0 & 0 < upper_95)
```

```
## [1] 0.6576
```

(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:

Recall,

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

```
alpha = 0.01
t_crit_99 = -qt(alpha / 2, df = n - 2)
Sxx = sum((x - mean(x)) ^ 2)

lower_99 = beta_hat_1 - t_crit_99 * s_e / sqrt(Sxx)
upper_99 = beta_hat_1 + t_crit_99 * s_e / sqrt(Sxx)
```

Note that we could have stored confidence intervals directly when performing the simulation, using `confint()`. However, then we would not have been so easily able to modify the confidence level. We would have needed to perform the simulation again.

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

Solution:

```
mean(lower_99 < 2 & 2 < upper_99)
```

```
## [1] 0.994
```

Unsurprisingly, the result is near 99%.

(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:

```
1 - mean(lower_99 < 0 & 0 < upper_99)
```

```
## [1] 0.3952
```

Note that if β_1 was actually 0, then by random chance around alpha = 0.01 of the simulations would reject the null hypothesis. But this proportion is much larger than 0.01 and cannot be explained away as random chance. In fact, β_1 is not 0. (But we already knew that because we set it as 2.)

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:

```
calc_pred_int = function(model, newdata, level = 0.95) {

  # calculations for specified level
  n = length(resid(model))
  alpha = 1 - level
  t = abs(qt(alpha / 2, df = n - 2))

  # deconstructing 95% CI
  t_95 = abs(qt(0.05 / 2, df = n - 2))
  int_95 = predict(model, newdata, interval = "prediction")
  lower_95 = int_95[2]
  upper_95 = int_95[3]
  margin_95 = (int_95[3] - int_95[2]) / 2

  # results needed from 95% CI
  est = int_95[1]
  SE = margin_95 / t_95

  # constructing requested interval
  c(estimate = est,
    lower = est - t * SE,
    upper = est + t * SE)
}
```

In practice we would never write a function like this, but here it helps us to better understand the construction of the interval.

(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
##    15.78    12.83    18.73
```

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

```
newcat_2 = data.frame(Bwt = 3.3)
calc_pred_int(cat_model, newcat_2, level = 0.99)
```

Solution:

```
newcat_2 = data.frame(Bwt = 3.3)
calc_pred_int(cat_model, newcat_2, level = 0.99)
```

```
## estimate    lower    upper
##   12.956    9.132   16.779
```