Week 3 - Homework

STAT 420, Fall 2024, Banghao Chi

09/14/2024

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.

```
library(MASS)
dataset = MASS::cats
?MASS::cats
```

(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:

```
cat_model = lm(Hwt ~ Bwt, data = dataset)
cat_model_summary = summary(cat_model)
t_stat = cat_model_summary$coefficients["Bwt", "t value"]
t_stat
```

```
## [1] 16.11939
```

```
p_val = cat_model_summary$coefficients["Bwt", "Pr(>|t|)"]
p_val
```

```
## [1] 6.969045e-34
```

- The null and alternative hypotheses:
  - The null hypotheses is $H_0 : \beta_1 = 0$, which means that there is no linear relationship between body weight and heart weight.

  - The alternative hypotheses is $H_0 : \beta_1 \neq 0$, which shows that there is a linear relationship between body weight and heart weight.

- The value of the test statistic is $t = 16.1193908$.
- The p-value of the test is $p = 6.9690446 \times 10^{-34}$.
- Since the p-value of the test is $6.9690446 \times 10^{-34} < \alpha = 0.05$, we reject the null hypothesis.
- The conclusion in the context of the problem: There is strong statistical evidence to suggest that there is a linear relationship between a cat's body weight and heart weight, since the extremely low p-value indicates that it is highly unlikely to observe such a strong relationship by chance if there were truly no relationship between these two variables in the population of cats.

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

Solution:

```
conf_int = confint(cat_model, parm = "Bwt", level = 0.95)
lwr = conf_int[1]
upr = conf_int[2]
```

We are 95% confident that the true change in **mean** heart weight of a cat in body weight per kg is between 3.539343 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:

```
conf_int = confint(cat_model, parm = "(Intercept)", level = 0.9)
lwr = conf_int[1]
upr = conf_int[2]
```

We are 90% confident that the true **mean** heart weight of a cat whose body weight is 0 kg is between -1.5028345 and 0.7895096 g.

(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:

```
conf_int = predict(cat_model,
                   newdata = data.frame(Bwt = c(2.1, 2.8)),
                   interval = c("confidence"), level = 0.9)
int_2_1 = conf_int[1, 3] - conf_int[1, 2]
int_2_1
```

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

```
int_2_8 = conf_int[2, 3] - conf_int[2, 2]
int_2_8
```

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

```
mean_bwt = mean(dataset$Bwt)
mean_bwt
```

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

The confidence interval of body weights of 2.1 kilograms is wider, since 2.8 is closer to the mean of the predictor which is 2.7236111 as compared to 2.1, which means that there are more samples around the

predictor value of 2.8 and therefore we are equally confident about 2.8 with smaller interval.

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

Solution:

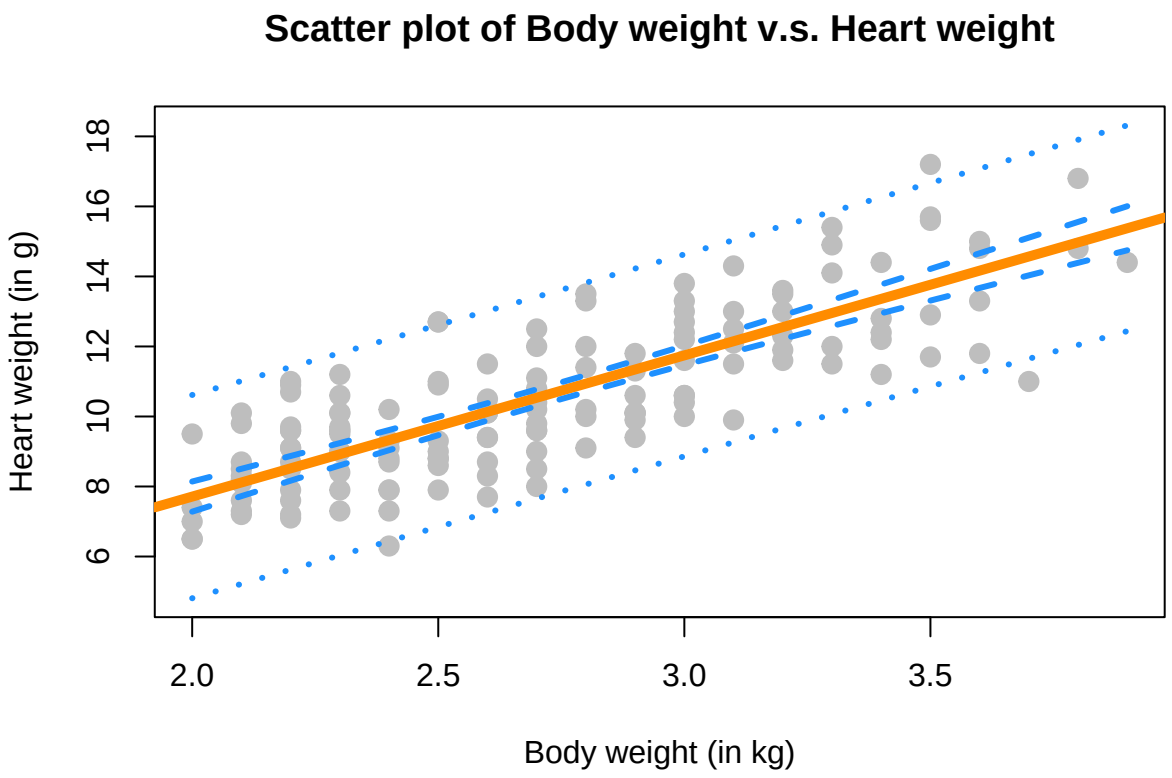
```
pred_int = predict(cat_model,
                   newdata = data.frame(Bwt = c(2.8, 4.2)),
                   interval = c("prediction"), level = 0.9)
pred_int
```

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

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

Solution:

```
body_grid = seq(min(dataset$Bwt), max(dataset$Bwt), by = 0.01)

dist_ci_band = predict(cat_model,
                       newdata = data.frame(Bwt = body_grid),
                       interval = c("confidence"), level = 0.95)

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

plot(Hwt ~ Bwt, data = dataset,
     main = "Scatter plot of Body weight v.s. Heart weight",
     xlab = "Body weight (in kg)",
     ylab = "Heart weight (in 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(body_grid, dist_ci_band[, "lwr"], col = "dodgerblue", lwd = 3, lty = 2)
lines(body_grid, dist_ci_band[, "upr"], col = "dodgerblue", lwd = 3, lty = 2)
lines(body_grid, dist_pi_band[, "lwr"], col = "dodgerblue", lwd = 3, lty = 3)
lines(body_grid, dist_pi_band[, "upr"], col = "dodgerblue", lwd = 3, lty = 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:

```
EST = cat_model_summary$coefficients["Bwt", "Estimate"]
SE = cat_model_summary$coefficients["Bwt", "Std. Error"]
HYP = 4
t_stat = (EST - HYP) / SE
t_stat
```

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

```
p_val = 2 * pt(abs(t_stat), df = nrow(dataset) - 2, lower.tail = FALSE)
p_val
```

```
## [1] 0.8919283
```

- The value of the test statistic is $t = 0.1361084$.
- The p-value of the test is $p = 0.8919283$.
- The statistical decision at $\alpha = 0.05$ is that: we fail to reject the null hypothesis, since our p-value $p = 0.8919283 > \alpha = 0.05$.

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), ]
str(Ozone)
```

```
## 'data.frame':    344 obs. of  4 variables:
##  $ ozone   : num  3 5 5 6 4 4 6 7 4 6 ...
##  $ wind    : num  4 3 3 4 6 3 3 3 8 3 ...
##  $ humidity: num  28 37 51 69 19 25 73 59 27 44 ...
##  $ temp    : num  40 45 54 35 45 55 41 44 54 51 ...
```

(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)
ozone_wind_model_summary = summary(ozone_wind_model)
t_stat = ozone_wind_model_summary$coefficients["wind", "t value"]
t_stat
```

```
## [1] -0.2189811
```

```
p_val = ozone_wind_model_summary$coefficients["wind", "Pr(>|t|)"]
p_val
```

```
## [1] 0.8267954
```

- The null and alternative hypotheses:
  - The null hypotheses is $H_0 : \beta_1 = 0$, which means that there is no linear relationship between ozone measurement and wind speed.
  - The alternative hypotheses is $H_0 : \beta_1 \neq 0$, which shows that there is a linear relationship between ozone measurement and wind speed.

- The value of the test statistic is $t = -0.2189811$.
- The p-value of the test is $p = 0.8267954$.
- Since the p-value of the test is $0.8267954 > \alpha = 0.01$, we fail to reject the null hypothesis.
- The conclusion in the context of the problem: There is no strong statistical evidence to suggest that there is linear relationship between ozone measurement and wind speed, since the big p-value indicates that it is highly likely to observe such a strong relationship by chance if there were truly no relationship between these two variables in the samples of the dataset.

(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)
ozone_temp_model_summary = summary(ozone_temp_model)
t_stat = ozone_temp_model_summary$coefficients["temp", "t value"]
t_stat
```

```
## [1] 22.84896
```

```
p_val = ozone_temp_model_summary$coefficients["temp", "Pr(>|t|)"]
p_val
```

```
## [1] 8.153764e-71
```

- The null and alternative hypotheses:
  - The null hypotheses is $H_0 : \beta_1 = 0$, which means that there is no linear relationship between ozone measurement and temperature.
  - The alternative hypotheses is $H_0 : \beta_1 \neq 0$, which shows that there is a linear relationship between ozone measurement and temperature.
- The value of the test statistic is $t = 22.848962$.
- The p-value of the test is $p = 8.1537636 \times 10^{-71}$.
- Since the p-value of the test is $8.1537636 \times 10^{-71} < \alpha = 0.01$, we reject the null hypothesis.
- The conclusion in the context of the problem: There is strong statistical evidence to suggest that there is linear relationship between ozone measurement and temperature, since the extremely low p-value indicates that it is highly unlikely to observe such a strong relationship by chance if there were truly no relationship between these two variables in the samples of the dataset.

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.

Solution:

```
birthday = 20021023
set.seed(birthday)
n = 50
x = seq(0, 10, length = n)

sim_slr = function(x, beta_0, beta_1, sigma) {
  n = length(x)
  epsilon = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + epsilon
  data.frame(predictor = x, response = y)
}

iter = 2000
beta_1s = rep(0, iter)
beta_0s = rep(0, iter)

for (i in 1:iter) {
  sim_data = sim_slr(x, beta_0 = -5, beta_1 = 3.25, sigma = sqrt(16))
  sim_model = lm(response ~ predictor, data = sim_data)
  beta_0s[i] = coef(sim_model)[1]
  beta_1s[i] = coef(sim_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:

```
library(knitr)

true_beta_0 = -5
true_beta_1 = 3.25
```

```r
Sxx = sum((x - mean(x))^2)
sd_beta_1_hat = 4 / sqrt(Sxx)
sd_beta_0_hat = 4 * sqrt((1 / n + mean(x) ^ 2 / Sxx))

results = data.frame(
  name = c("True expected value",
           "Mean of the simulated values",
           "True standard deviation",
           "Standard deviation of the simulated values"),
  beta_0_ = c(true_beta_0,
              mean(beta_0s),
              sd_beta_0_hat,
              sd(beta_0s)),
  beta_1_ = c(true_beta_1,
              mean(beta_1s),
              sd_beta_1_hat,
              sd(beta_1s))
)

kable(results, format = "markdown", col.names = c("Name", "beta_0", "beta_1"))
```

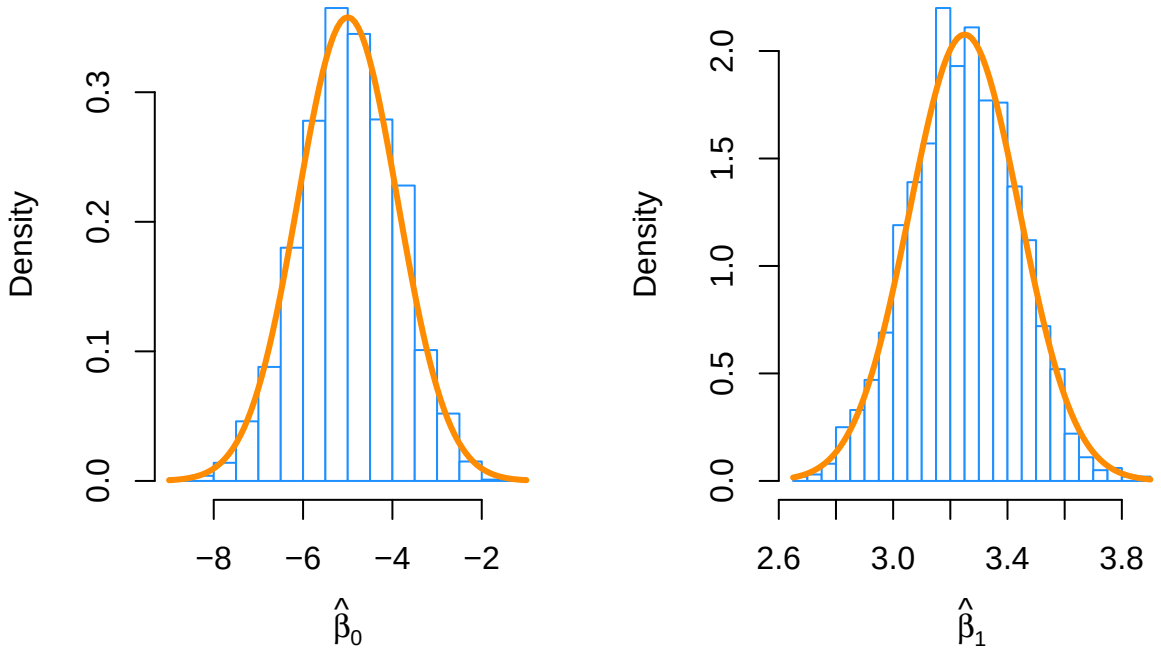
Name	beta_0	beta_1
True expected value	-5.000000	3.2500000
Mean of the simulated values	-4.960986	3.2452972
True standard deviation	1.114609	0.1920784
Standard deviation of the simulated values	1.081619	0.1879081

(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
par(mfrow = c(1, 2))

hist(beta_0s, prob = TRUE, breaks = 20,
     xlab = expression(hat(beta)[0]), main = "", col = "white",
     border = "dodgerblue")
curve(dnorm(x, mean = true_beta_0, sd = sd_beta_0_hat), col = "darkorange", add = TRUE, lwd = 3)

hist(beta_1s, prob = TRUE, breaks = 20,
     xlab = expression(hat(beta)[1]), main = "", col = "white",
     border = "dodgerblue")
curve(dnorm(x, mean = true_beta_1, sd = sd_beta_1_hat),
      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.

Solution:

```
birthday = 20021023
set.seed(birthday)
n = 25
x = seq(0, 2.5, length = n)

iter = 2500
beta_1s = rep(0, iter)
s_es = rep(0, iter)

for (i in 1:iter) {
  sim_data = sim_slr(x, beta_0 = 5, beta_1 = 2, sigma = sqrt(9))
```

```
  sim_model = lm(response ~ predictor, data = sim_data)
  beta_1s[i] = coef(sim_model)[2]
  s_es[i] = summary(sim_model)$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:

```
Sxx = sum((x - mean(x))^2)
crt = qt(0.025, df = n - 2, lower.tail = FALSE)
lower_95 = rep(0, iter)
upper_95 = rep(0, iter)

for (i in 1:iter) {
  margin = crt * s_es[i] / sqrt(Sxx)
  lower_95[i] = beta_1s[i] - margin
  upper_95[i] = beta_1s[i] + margin
}
```

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

Solution:

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

```
## [1] 0.9524
```

0.9524 of these intervals contains the true value of β_1.

(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:

```
reject_h0 = mean(lower_95 > 0 | upper_95 < 0)
reject_h0
```

```
## [1] 0.6676
```

0.6676 of the simulations would reject the test $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ at $\alpha = 0.05$.

(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:

```
Sxx = sum((x - mean(x))^2)
crt = qt(0.005, df = n - 2, lower.tail = FALSE)
lower_99 = rep(0, iter)
upper_99 = rep(0, iter)

for (i in 1:iter) {
  margin = crt * s_es[i] / sqrt(Sxx)
  lower_99[i] = beta_1s[i] - margin
```

```
  upper_99[i] = beta_1s[i] + margin
}
```

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

Solution:

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

```
## [1] 0.9884
```

0.9884 of these intervals contains the true value of β_1.

(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:

```
reject_h0 = mean(lower_99 > 0 | upper_99 < 0)
reject_h0
```

```
## [1] 0.3868
```

0.3868 of the simulations would reject the test $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ at $\alpha = 0.01$.

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:

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

  newdata = newdata[1, 1]

  x_bar = mean(model$model[, 2])
  Sxx = sum((model$model[, 2] - x_bar)^2)
  s_e = summary(model)$sigma
  n = nrow(model$model)
  crt = qt((1 - level) / 2, df = n - 2, lower.tail = FALSE)

  margin = crt * s_e * sqrt(1 + 1/n + (newdata - x_bar) ^ 2 / Sxx)

  c(
    estimate = estimate,
    lower = estimate - margin,
    upper = estimate + margin
    )
}
```

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

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

```
## estimate.1    lower.1    upper.1
##   15.77959   12.83018   18.72900
```

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

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

```
## estimate.1    lower.1    upper.1
##   12.95574   10.53099   15.38050
```