Week 3 - Homework

STAT 420, Summer 2020, D. Unger

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 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.

```r
library(MASS)
cat_model = lm(Hwt ~ Bwt, data = cats)
summary(cat_model)$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -0.3566624  0.6922770 -0.5152019 6.072131e-01
## Bwt          4.0340627  0.2502615 16.1193908 6.969045e-34
```

```r
summary(cat_model)$coefficients["Bwt", "t value"]
```

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

```r
summary(cat_model)$coefficients["Bwt", "Pr(>|t|)"]
```

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

H0: beta_one = 0

H1: beta_one != 0

The value of the test statistic is 16.11939

The p-value of the test is 6.969045e-34

Statistical Decision at alpha = 0.05 is Fail to Reject H0

Conclusion in the context of the problem: there exists a linear relationship between heart weight and body weight

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

```r
confint(cat_model, "Bwt", level = 0.95)
```

```
##        2.5 %   97.5 %
## Bwt 3.539343 4.528782
```

Confidence interval for β_1 is inbetween (3.539343, 4.528782).

We are 95% confident that given a 1 KG increase in the body weight, the avarage heart weight will be between 3.539343 and 4.528782. Since the interval doesn't have a zero value, it is not possible to have β_1 a zero value.

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

```r
confint(cat_model, "(Intercept)", level = 0.90)
```

```
##                   5 %      95 %
## (Intercept) -1.502834 0.7895096
```

Confidence interval for β_0 is inbetween (-1.502834, 0.7895096).

We are 90% confident that for a body weight of 0 KG, the average heart weight will be between -1.502834 and 0.7895096. Practically speaking it wouldn't make much sense.

(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?

```r
predict(cat_model, newdata = data.frame(Bwt = c(2.1, 2.8)), interval = c("confidence"), level = 0.90)
```

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

We are 90% confident that the mean heart weight for body weights 2.1 and 2.8 KG is in the intervals of (7.599225, 8.630513) and (10.618796, 11.258630) respectively

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

```r
predict(cat_model, newdata = data.frame(Bwt = c(2.8, 4.2)), interval = c("prediction"), level = 0.90)
```

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

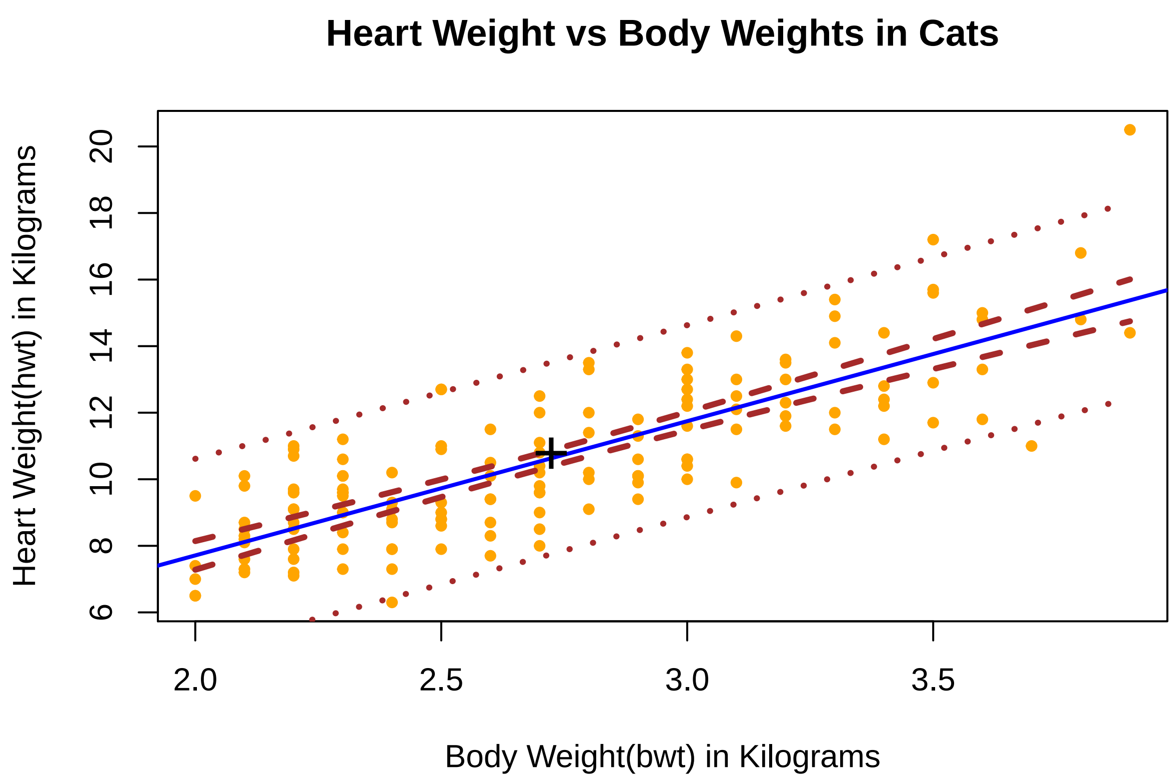
We are 90% confident that the new observation for body weights 2.8 and 4.2 KG is in the intervals of (5.688109, 10.54163) and (8.525541, 13.35189) respectively

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

```
plot(Hwt ~ Bwt, data = cats, xlab = "Body Weight(bwt) in Kilograms", ylab = "Heart Weight(hwt) in Kilog
     main = "Heart Weight vs Body Weights in Cats", pch = 20, cex = 1, col = "orange")
abline(cat_model, lwd = 2, col = "blue")

bwt_grid = seq(min(cats$Bwt), max(cats$Bwt), by = 0.01)
hwt_ci_band = predict(cat_model, newdata = data.frame(Bwt = bwt_grid), interval = "confidence", level =
hwt_pi_band = predict(cat_model, newdata = data.frame(Bwt = bwt_grid), interval = "prediction", level =

lines(bwt_grid, hwt_ci_band[, "lwr"], col = "brown", lwd = 3, lty = 2)
lines(bwt_grid, hwt_ci_band[, "upr"], col = "brown", lwd = 3, lty = 2)
lines(bwt_grid, hwt_pi_band[, "lwr"], col = "brown", lwd = 3, lty = 3)
lines(bwt_grid, hwt_pi_band[, "upr"], col = "brown", lwd = 3, lty = 3)
points(mean(cats$Bwt), mean(cats$Hwt), pch = "+", cex = 2)
```



(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.

```
beta_1 = coefficients(cat_model)[2]
stand_error = summary(cat_model)$coefficients["Bwt", "Std. Error"]
beta_1_hat_t = (beta_1 - 4) / stand_error
beta_1_hat_t
```

```
##       Bwt
## 0.1361084
```

```
p_value = 2 * pt(abs(beta_1_hat_t), df = length(resid(cat_model)) - 2, lower.tail = FALSE)
p_value
```

```
##       Bwt
## 0.8919283
```

The value of the test statistic is 0.1361084

The p-value of the test is 0.8919283.

At statistical decision at $\alpha = 0.05$ is "Fail to reject H0"

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.

```
ozone_model = lm(ozone ~ wind, data = Ozone)
summary(ozone_model)$coefficients["wind", "t value"]
```

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

```r
summary(ozone_model)$coefficients["wind", "Pr(>|t|)"]
```

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

H0: beta_one = 0

H1: beta_one != 0

The value of the test statistic is -0.2189811

The p-value of the test is 0.8267954

Statistical Decision at alpha = 0.01 is Fail to Reject H0

Conclusion in the context of the problem: there exists a NO 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.

```r
ozone_model = lm(ozone ~ temp, data = Ozone)
summary(ozone_model)$coefficients["temp", "t value"]
```

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

```r
summary(ozone_model)$coefficients["temp", "Pr(>|t|)"]
```

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

H0: beta_one = 0

H1: beta_one != 0

The value of the test statistic is 22.84896

The p-value of the test is 8.153764e-71

Statistical Decision at alpha = 0.01 is Reject H0

Conclusion in the context of the problem: there exists 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 #19830611
set.seed(birthday)
n = 50
x = seq(0, 10, length = n)

beta_0 = -5
beta_1 = 3.25
sigma = 4

num_samples = 2000
beta_0_hats = rep(0, num_samples)
beta_1_hats = rep(0, num_samples)

for(i in 1:num_samples) {
  eps = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + eps
  sim_model = lm(y ~ x)
  beta_0_hats[i] = coef(sim_model)[1]
  beta_1_hats[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

```
Sxx = sum((x - mean(x)) ^ 2)
var_beta_1_hat = (sigma ^ 2)/Sxx
var_beta_0_hat = (sigma ^ 2)*((1/n) + (mean(x)^2/Sxx))
data_summary = data.frame(value = c("E of known values", "mean of simulated vlaues", "SD of known vlaue
                          beta_0_hat = c(beta_0, mean(beta_0_hats), var_beta_0_hat, sd(beta_0_hats)),
                          beta_1_hat = c(beta_1, mean(beta_1_hats), var_beta_1_hat, sd(beta_1_hats))
                          )
library(knitr)
kable(data_summary)
```

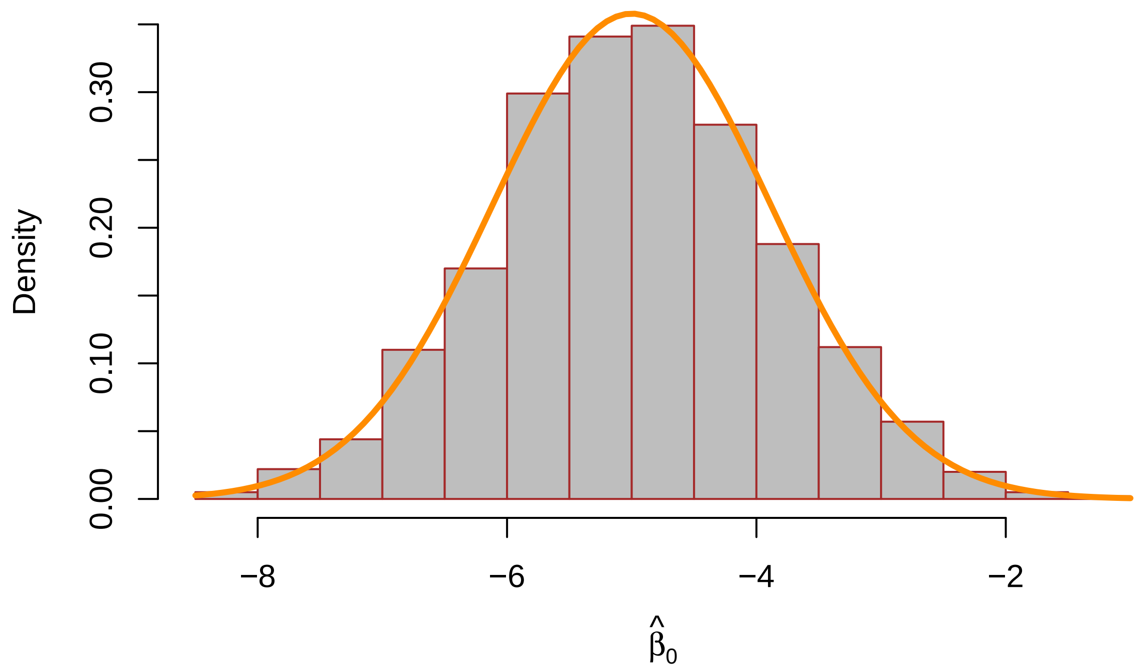
value	beta_0_hat	beta_1_hat
E of known values	-5.000000	3.2500000
mean of simulated vlaues	-4.981634	3.2431922

value	beta_0_hat	beta_1_hat
SD of known vlaues	1.242353	0.0368941
sd of simulated values	1.128547	0.1948865

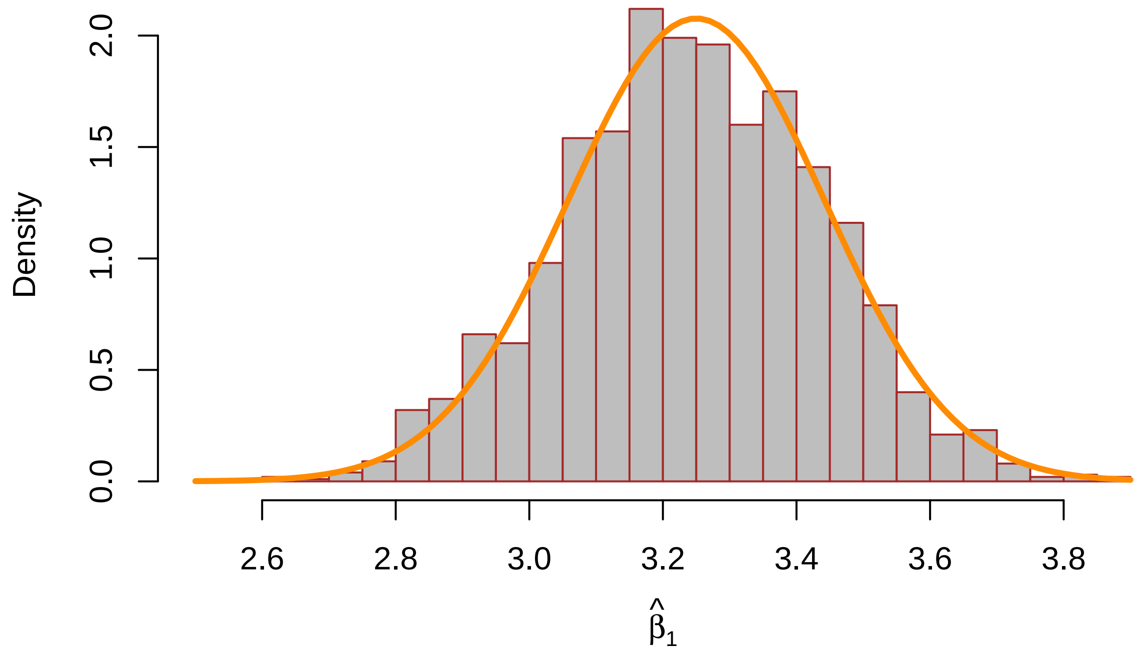
(c) Plot two histograms side-by-side:

- 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 $\hat{\beta}_1$.

```
hist(beta_0_hats, prob = TRUE, breaks = 25, col = "grey", border = "brown", xlab = expression(hat(beta)
curve(dnorm(x, mean = beta_0, sd = sqrt(var_beta_0_hat)), col = "darkorange", add = TRUE, lwd = 3)
```




```
hist(beta_1_hats, prob = TRUE, breaks = 25, col = "grey", border = "brown", xlab = expression(hat(beta)
curve(dnorm(x, mean = beta_1, sd = sqrt(var_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.

```
birthday = 18760613
set.seed(birthday)
n = 25
x = seq(0, 2.5, length = n)
```

```
Sxx = sum((x - mean(x)) ^ 2)
beta_0 = 5
beta_1 = 2
sigma = 3

num_samples = 2500
beta_hat_1 = rep(0, num_samples)
s_e = rep(0, num_samples)
for (i in 1:num_samples) {
  y = beta_0 + beta_1 * x + rnorm(n, 0, sigma)
  sim_model = lm(y ~ x)
  beta_1_hats[i] = coef(sim_model)[2]
  s_e[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.

```
#confint(sim_model, parm = "(Intercept)", level = 0.95)
alpha = 0.05
crit = -qt(alpha / 2, df = n - 2)
lower_95 = beta_1_hats - crit * s_e / sqrt(Sxx)
upper_95 = beta_1_hats + crit * s_e / sqrt(Sxx)
```

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

```
mean(lower_95 < beta_1 & beta_1 < upper_95)
```

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

(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 $\alpha = 0.05$?

```
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`.

```
alpha = 0.01
crit = -qt(alpha / 2, df = n - 2)
lower_99 = beta_1_hats - crit * s_e / sqrt(Sxx)
upper_99 = beta_1_hats + crit * s_e / sqrt(Sxx)
```

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

```
mean(lower_99 < beta_1 & beta_1 < upper_99)
```

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

(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 $\alpha = 0.01$?

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

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

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

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

  t_95 = abs(qt(0.05/2, df = length(resid(model)) - 2))
  interval_95 = predict(model, newdata, interval = "prediction")
  lwr_95 = interval_95[2]
  upr_95 = interval_95[3]
  margin =(upr_95 - lwr_95)/2
  se = margin/t_95

  alpha = 1 - level
  t = abs(qt(alpha/2, df = length(resid(model)) - 2))

  c(est = interval_95[1], lwr = interval_95[1] - t*se , upr = interval_95[1] + t*se)
}
```

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

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

(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)
```