Week 3 - Homework

STAT 420, Summer 2018, Panda5

6/2/2019

- Exercise 1 (Using `lm` for Inference)
- Exercise 2 (More `lm` for Inference)
- Exercise 3 (Simulating Sampling Distributions)
- Exercise 4 (Simulating Confidence Intervals)
- Exercise 5 (Prediction Intervals "without" `predict`)

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:

```
library(MASS)
cat_model = lm(Hwt~Bwt,data=cats)
summary(cat_model)$coefficient[,"t value"]
```

```
## (Intercept)         Bwt
##  -0.5152019  16.1193908
```

```
summary(cat_model)$coefficient[,"Pr(>|t|)"]
```

```
##  (Intercept)          Bwt
## 6.072131e-01 6.969045e-34
```

```
alpha = 0.05
level = 1- alpha
confint(cat_model,level=level)
```

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

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

```
##       5 %      95 %
## 3.619716 4.448409
```

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

```
##      0.5 %    99.5 %
## -2.164125  1.450800
```

```
body_weight1 = data.frame(Bwt=c(2.1,2.8))
predict_newdata1=predict(cat_model,newdata=body_weight1,interval="confidence",leve
l=0.99)

body_weight2 = data.frame(Bwt=c(2.8,4.2))
predict_newdata2=predict(cat_model,newdata=body_weight2,interval="prediction",leve
l=0.99)

beta_0_hat = summary(cat_model)$coefficient[1,1]
beta_1_hat = summary(cat_model)$coefficient[2,1]

beta_0_hat_se = summary(cat_model)$coefficient[1,2]
beta_1_hat_se = summary(cat_model)$coefficient[2,2]

beta_0_hat_t = (beta_0_hat - 4) / beta_0_hat_se
beta_1_hat_t = (beta_1_hat - 4) / beta_1_hat_se

beta_0_hat_p = 2* pt(abs(beta_0_hat_t),nrow(cars)-2,lower.tail = FALSE)
beta_1_hat_p = 2* pt(abs(beta_1_hat_t),nrow(cars)-2,lower.tail = FALSE)
```

- The null and alternative hypotheses
    a. Null Hypothesis $H_0 : \beta_1 = 0$
    b. Alternate Hypothesis $H_1 : \beta_1 \neq 0$
- The value of the test statistic:

  t statistics for β_0 : **-0.5152019** and β_1 : **16.1193908**

- The p-value of the test :

  p value for β_0 :**0.6072131** and for β_1 : **6.969044610^{-34}**

- A statistical decision at $\alpha = 0.05$

  The P value for β_1 which is: **6.969044610^{-34}** < **0.5** (α). Since P is smaller than α, we **reject the null Hypothesis**

- A conclusion in the context of the problem:

  Since we **reject the null hypothesis**, hence β_1 is non zero, so there is clear indication of linear relationship exists between Cat's Body Weight and Cat's Heart Weight

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

90% confident interval for β_1, **lower bound:3.619716 upper bound : 4.4484094**

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

99% confidence interval for $_0 **lower bound:-1.5028345 upper bound : 0.7895096**

(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?

99% confidence interval of mean Body weight of **2.1** is :: **lower:7.5992252** upper: **8.6305133**

99% confidence interval of mean Body weight of **2.8** is :: **lower:10.6187959** upper: **11.2586303**

2.1 interval is **higher** than that of **2.8**, becase the range of **2.1** which is **1.0312881** > **0.6398344** of **2.8**

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

99% confidence interval of mean Body weight of **2.8** is :: **lower:7.1332472** upper: **14.7441791**

99% confidence interval of mean Body weight of **4.2** is :: **lower:12.6608829** upper: **20.5119189**

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

  test statistics for β_1 : **0.1361084**

- The p-value of the test:

  The p-value for β_1 :**0.8923048**

- A statistical decision at $\alpha = 0.05$

  Since the p-value of the test for β_1 is : **0.8923048** > 0.05 (=α), hence we **failed to reject the Hypothesis** of β_1 = 4

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), ]

ozone_wind_model = lm(ozone~wind,data=Ozone)
wind_beta_0_hat = summary(ozone_wind_model)$coefficient[1,1]
wind_beta_1_hat = summary(ozone_wind_model)$coefficient[2,1]
wind_beta_0_hat_se = summary(ozone_wind_model)$coefficient[1,2]
wind_beta_1_hat_se = summary(ozone_wind_model)$coefficient[2,2]
wind_beta_0_hat_t = summary(ozone_wind_model)$coefficient[1,3]
wind_beta_1_hat_t = summary(ozone_wind_model)$coefficient[2,3]

ozone_temp_model = lm(ozone~temp,data=Ozone)
temp_beta_0_hat = summary(ozone_temp_model)$coefficient[1,1]
temp_beta_1_hat = summary(ozone_temp_model)$coefficient[2,1]
temp_beta_0_hat_se = summary(ozone_temp_model)$coefficient[1,2]
temp_beta_1_hat_se = summary(ozone_temp_model)$coefficient[2,2]
temp_beta_0_hat_t = summary(ozone_temp_model)$coefficient[1,3]
temp_beta_1_hat_t = summary(ozone_temp_model)$coefficient[2,3]
```

(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
  a. Null Hypothesis $H_0 : \beta_1 = 0$
  b. Alternate Hypothesis $H_1 : \beta_1 \neq 0$
- The value of the test statistic

  The t statistics for β_0 : **10.9278455** and β_1 : **-0.2189811**

- The p-value of the test

  The p value for β_0 :**4.818462110^{-24}** and for β_1 : **0.8267954**

- A statistical decision at $\alpha = 0.01$

  The P value for β_1 which is: **0.8267954** > **0.01** (α). Since P is larger than α, we **failed to reject the null Hypothesis**

- A conclusion in the context of the problem

  Since we **failed to reject the null hypothesis**, β_1 would be 0, hence there is **no linear relationship** exists between Ozone's 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

  a. Null Hypothesis $H_0 : \beta_1 = 0$
  b. Alternate Hypothesis $H_1 : \beta_1 \neq 0$
- The value of the test statistic

  The t statistics for β_0 : **-12.4842524** and β_1 : **22.848962**

- The p-value of the test

  The p value for β_0:**9.947454510^{-30}** and for β_1: **8.153763610^{-71}**

- A statistical decision at $\alpha = 0.01$

  The P value for β_1 which is: **8.153763610^{-71}** < **0.01** (α). Since P is larger than α, we **reject the null Hypothesis**

- A conclusion in the context of the problem

  Since we **reject the null hypothesis**, $\beta_1 \neq$ 0, hence there is **certainly clearn linear relationship** exists between Ozone's 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.

```
beta_0 = -5
beta_1 = 3.25
sigma_2 = 16
sigma = sqrt(sigma_2)

birthday = 19770411
set.seed(birthday)
n = 50
x = seq(0, 10, length = n)

sim_data = function(x,beta0,beta1,n){
  epsilon = rnorm(n,mean=0,sd=sigma)
  y = beta0 + beta1 * x + epsilon
  data.frame(predictor = x, response = y)
}

loop_times = 2000

beta_0_hat = rep(0,loop_times)
beta_1_hat = rep(0,loop_times)

for(i in 1:loop_times){
  epsilon = rnorm(n,mean=0,sd=sigma)
  y = beta_0 + beta_1 * x + epsilon
  data = sim_data(x,beta_0,beta_1,n)
  data_model = lm(response~predictor,data=data)
  beta_0_hat[i] = summary(data_model)$coefficient[1,1]
  beta_1_hat[i] = summary(data_model)$coefficient[2,1]
}
Sxx = sum((x - mean(x))^2)
sd_beta_1_hat = sqrt(sigma_2 / Sxx)
sd_beta_0_hat = sqrt(sigma_2 * ((1/n) + (mean(x)^2/Sxx)))
```

(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

```
library(knitr)
beta0_beta_1_summary_report = data.frame(
  attributes = c("True Exepected Value","Mean Simulated Value","True Standard Devi
ation","Standard Deviation of Simulated Value"),
  beta_0 = c(beta_0,mean(beta_0_hat),sd_beta_0_hat,sd(beta_0_hat)),
  beta_1 = c(beta_1,mean(beta_1_hat),sd_beta_1_hat,sd(beta_1_hat))
)

kable(beta0_beta_1_summary_report, format = "pandoc",padding = 2,caption = "Summar
izes the results of the Simulations")
```

Summarizes the results of the Simulations

attributes	beta_0	beta_1
True Exepected Value	-5.000000	3.2500000
Mean Simulated Value	-4.989433	3.2473950
True Standard Deviation	1.114609	0.1920784
Standard Deviation of Simulated Value	1.114203	0.1897273

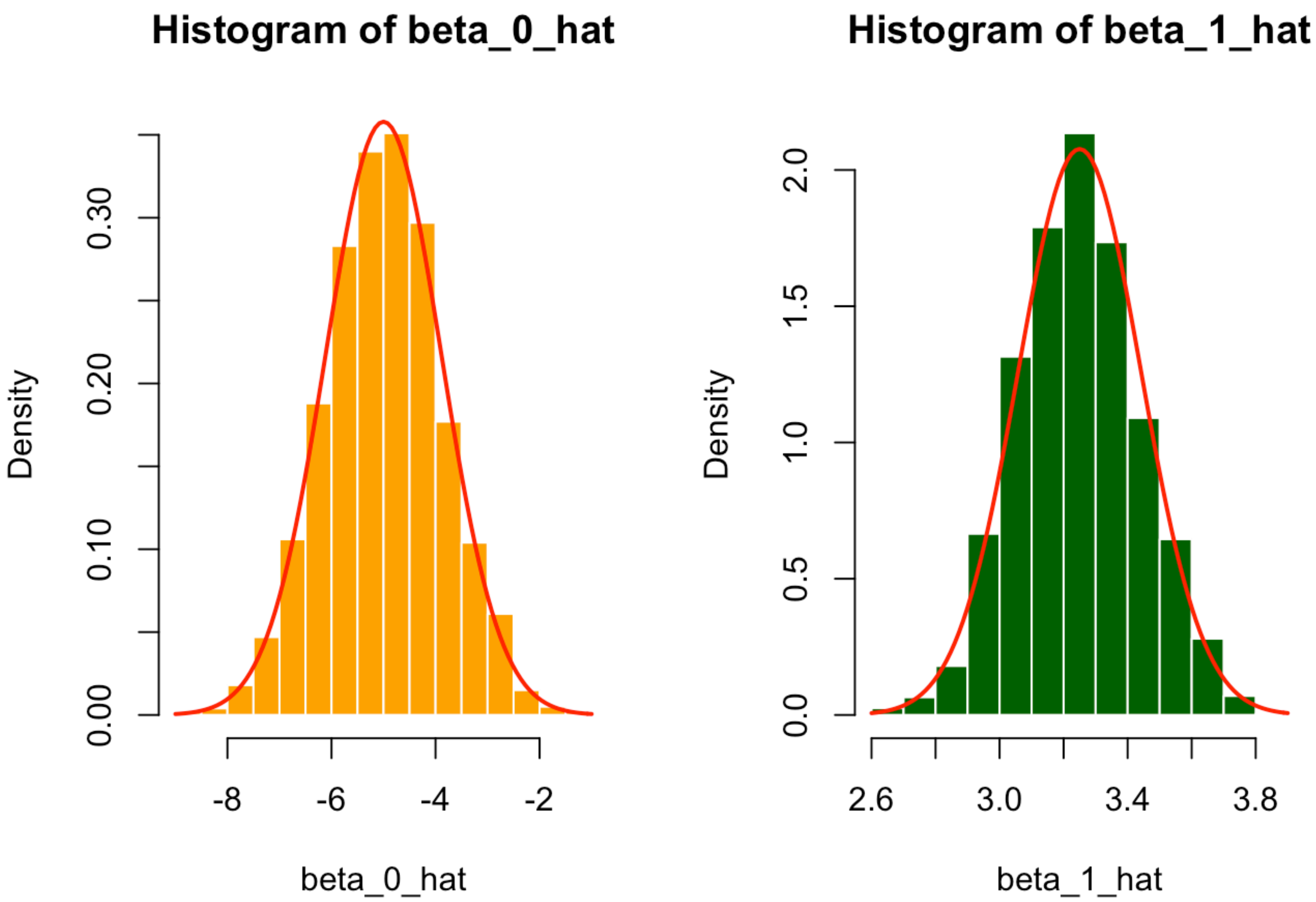
(c) Plot two histograms side-by-side:

- 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 $\hat{\beta}_1$.

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

hist(beta_0_hat,
     probability = TRUE,
     col="orange",
     border="white"
     )
curve(dnorm(x,mean=beta_0,sd=sd_beta_0_hat),add = TRUE,lwd=2,col="red")

hist(beta_1_hat,
     probability = TRUE,
     col="darkgreen",
     border="white")
curve(dnorm(x,mean=beta_1,sd=sd_beta_1_hat),add = TRUE,lwd=2,col="red")
```



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.

```
beta_0 = 5
beta_1 = 2
sigma_2 = 9
sigma = sqrt(sigma_2)
```

```
birthday = 19770411
set.seed(birthday)
n = 25
x = seq(0, 2.5, length = n)


sim_data = function(x,beta0,beta1,n){
  epsilon = rnorm(n,mean=0,sd=sigma)
  y = beta0 + beta1 * x + epsilon
  data.frame(predictor = x, response = y)
}

loop_times = 2500

se_sim = rep(0,loop_times)
beta_1_hat = rep(0,loop_times)

for(i in 1:loop_times){
  epsilon = rnorm(n,mean=0,sd=sigma)
  data = sim_data(x,beta_0,beta_1,n)
  data_model = lm(response~predictor,data=data)
  se = sqrt((sum((predict(data_model) - data$response)^2))/(length(data$response)
- 2))

  beta_1_hat[i] = summary(data_model)$coefficient[2,1]
  se_sim[i] = se
}

alpha = (1-0.95)
alpha_2 = alpha/2
crit_95 = qt(1-alpha_2,df = nrow(data) - 2)

alpha = (1-0.99)
alpha_2 = alpha/2
crit_99 = qt(1-alpha_2,df = nrow(data) - 2)

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

lower_95 = beta_1_hat - crit_95 * se_sim/sqrt(Sxx)
upper_95 = beta_1_hat + crit_95 * se_sim/sqrt(Sxx)

lower_99 = beta_1_hat - crit_99 * se_sim/sqrt(Sxx)
upper_99 = beta_1_hat + crit_99 * se_sim/sqrt(Sxx)

beta_hat_1_t = (beta_1_hat - 0) / (se_sim/sqrt(Sxx))
beta_hat_1_p = 2* pt(beta_hat_1_t,df=length(beta_hat_1_t) - 2, lower.tail = FALSE)

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

```
## [1] 0.948
```

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

```
## [1] 0.9896
```

```
mean(beta_hat_1_p < 0.05)
```

```
## [1] 0.7072
```

```
mean(beta_hat_1_p < 0.01)
```

```
## [1] 0.486
```

(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.

`lower_95` and `upper_95` is created in the above R chunk and Length of `lower_95` : **2500** and length of `upper_95` : **2500**

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

0.948 is the proportion of these intervals (95% CI) 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 $\alpha = 0.05$?

0.7072 proportion of 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`.

Length of `lower_99` 2500 and length of upper_99 2500

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

0.9896 is the proportion of these intervals (99% CI) 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 $\alpha = 0.01$?

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

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

```
calc_pred_int = function (model, newdata, level = c(0.95)) {
  #Derive the Critical Value/Y-hat and Lower Bound for the default CI - 95%
  crit_95 = qt(1-(1 - 0.95) / 2,length(predict(cat_model)) - 2)
  y_hat = predict(model,newdata=newdata)
  lwr = predict(cat_model,newdata=newdata,interval="prediction")[2]

  #Calculate the function
  X_FIND = (y_hat - lwr) / crit_95

  #Derive the Critical Value for the default CI - 95%
  crit = qt(1-(1 - level) / 2,length(predict(cat_model)) - 2)
  lower = (y_hat - crit * X_FIND)
  upper = (y_hat + crit * X_FIND)

  results <- c(y_hat,lower,upper)
  names(results) <- c("estimate","lower","upper")
  results
}
```

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

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

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

(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)
```

```
##  estimate     lower     upper
## 12.955744  9.132013 16.779476
```