w01-hw-panda5

Sushanta Panda

5/28/2019

- Exercise 1 (Using lm)
- Exercise 2 (Writing Function)
- Exercise 3 (Simulating SLR)
- Exercise 4 (Be a Skeptic)
- Exercise 5 (Comparing Model)

Exercise 1 (Using lm)

a. Below print the **summary of the Linear fit model** of the cat's Heart Weight against the Cat's Brain's Weight

```
library(MASS)
data = MASS::cats
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
```

b. Below is to output the **estimated regression coefficient**

```
beta_0_hat = coef(cat_model)[1]
beta_1_hat = coef(cat_model)[2]
c(beta_0_hat,beta_1_hat)
```

```
## (Intercept)         Bwt
##  -0.3566624   4.0340627
```

- **4.0340627** is the estimated increase in mean cat's Heart Weight for an increase in cat's body weight of 1 Kg
- **-0.3566624** is the estimated cat's mean Heart Weight for Cat's body weight is 0 Kg

c. Predict the Cat's Heart Weight for the Cat's Body Weight = **2.7 Kg**

```
predict(cat_model,newdata=data.frame(Bwt=2.7))
```

```
##        1
## 10.53531
```

The **SLR Model cat_model** predicts the cat's hearth weight is:: **10.5353069** for the brain's weight as 2.7 kg. I feel **confident** about the prediction is because of the fact that, the given Cat's Brain Weight 2.7 Kg is with-in the range (Interpolation) of cat's Brain Weight **(i.e. TRUE)**, where the minimum Cat's Brain Weight which is 2 and maximum weight is 3.9

d. Predict the Cat's Heart Weight for the Cat's Body Weight = **4.4 Kg**

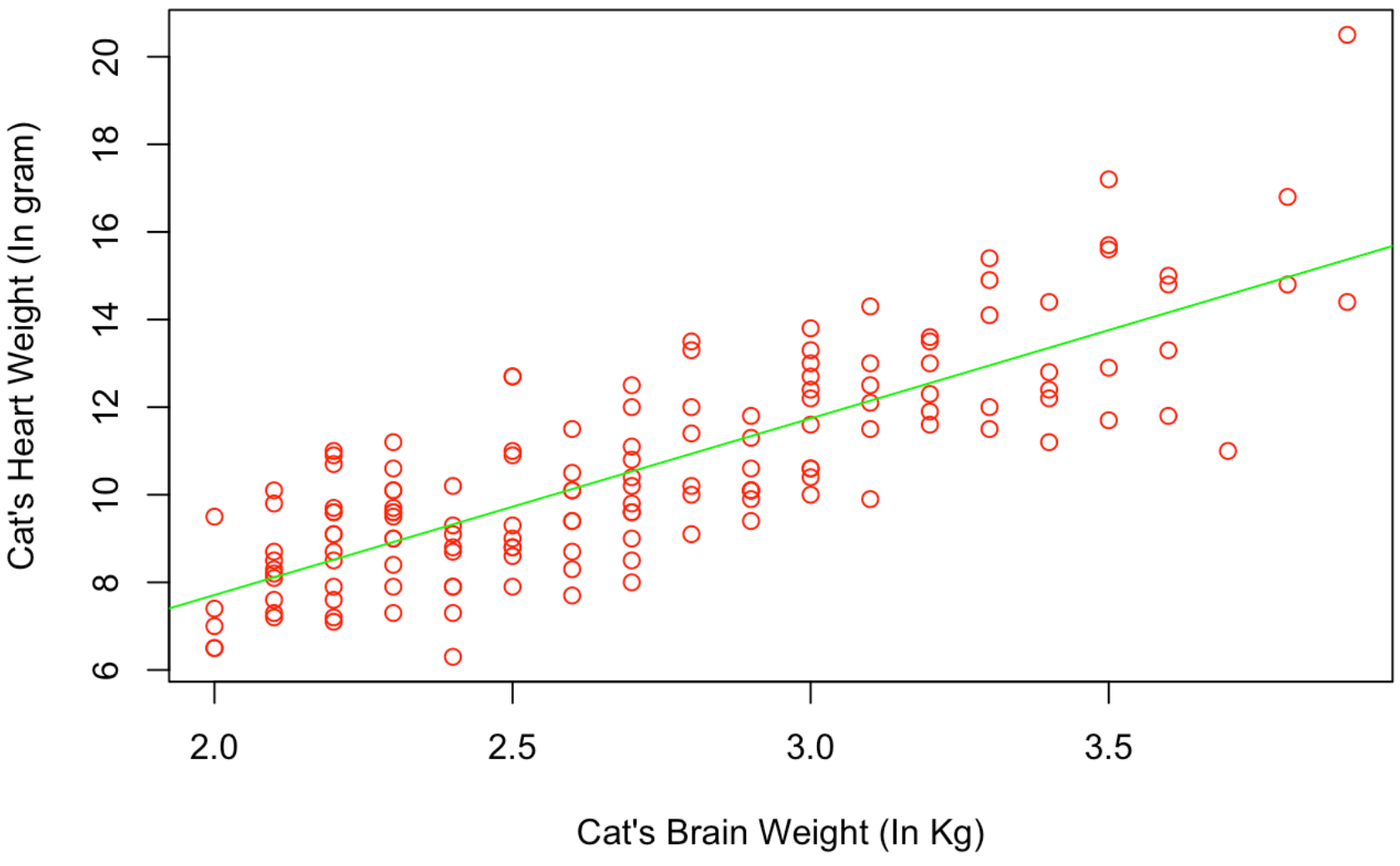
```
predict(cat_model,newdata=data.frame(Bwt=4.4))
```

```
##        1
## 17.39321
```

The **SLR Model cat_model** predicts the cat's hearth weight is **17.3932134** for the brain's weight as 4.4 kg. This doesn't feel confident about the prediction is because of the fact that, the given Cat's Brain Weight 4.4 Kg is outside the range (Extrapolation) of cat's Brain Weight **(i.e. FALSE)**, where the minimum Cat's Brain Weight which is 2 and maximum weight is 3.9

e. Scatter Plot of the Data, along with the fitted regression line

```
plot(Hwt~Bwt,
     data = cats,
     xlab = "Cat's Brain Weight (In Kg)",
     ylab = "Cat's Heart Weight (In gram)",
     main = "Cat's Heart's Weight Vs Brain's Weight",
     pch = 1,
     cex = 1,
     col = "red"
     )
abline(cat_model,lwd = 1, col = "green")
```



f. The **R2** value for the cat's model

```
SSTot = sum((cats$Hwt - mean(cats$Hwt))^2)
SSReg = sum((fitted(cat_model) - mean(cats$Hwt))^2)
R2 =  SSReg / SSTot
```

The R2 for the datamodel **cat_model** is calculated as **0.6466209**

Exercise 2 (Writing Function)

a. Create the function **get_sd_est**

```
get_sd_est = function (fitted_vals,actual_vals,mle=FALSE){
  if (mle){
    sigma_2 = sum((actual_vals - fitted_vals)^2)
    res_mean = sigma_2 / length(actual_vals)
  }
  else{
    se_2 = sum((actual_vals - fitted_vals)^2)
    res_mean = se_2 / (length(actual_vals) - 2)
  }
  sqrt(res_mean)
}
```

b. Output from the function **get_sd_est** with mle set to **FALSE** is :**1.4523733**. This tells us that our estimates for cat's heart weight are "typically" off by **1.4523733** gram.
c. Output from the function **get_sd_est** with mle set to **TRUE** is :**1.4422522**. This tells us that our estimates for cat's heart weight are "typically" off by **1.4422522** gram.
d. The **Sigma** which is **Residual Standard Error (RSE)** comes out from the **summary(cat_model)** which is **1.4523733** value is matching with the outcome from (b) where the value is : **1.4523733** where MLE is **False**

Exercise 3 (Simulating SLR)

a. Below to to generate n= 25 Observations for the model

```
birthday = 19770411
set.seed(birthday)

num_obs = 25
beta_0 = 5
beta_1 = -3

sim_slr=function(x, beta_0 = 5, beta_1 = -3, sigma){
  n = length(x)
  epsilon = rnorm(n, mean = 0, sd = sigma)
  y = beta_0 + beta_1 * x + epsilon
  data.frame (predictor = x, response = y)
}

x_vals = seq(from = 0, to = 25, length.out = num_obs)
sim_data = sim_slr(x=x_vals, beta_0 = 5, beta_1 = -3, sigma = sqrt(10.24))
sim_data
```

```
##    predictor    response
## 1   0.000000   4.9393477
## 2   1.041667   0.4981506
## 3   2.083333  -1.8049984
## 4   3.125000  -4.2763569
## 5   4.166667 -12.1535296
## 6   5.208333 -10.0307604
## 7   6.250000  -8.9396152
## 8   7.291667 -16.4525712
## 9   8.333333 -19.8212705
## 10  9.375000 -23.9122197
## 11 10.416667 -24.9540569
## 12 11.458333 -29.9961242
## 13 12.500000 -28.6644388
## 14 13.541667 -34.9897690
## 15 14.583333 -36.0777227
## 16 15.625000 -42.3774094
## 17 16.666667 -43.1264249
## 18 17.708333 -46.5952188
## 19 18.750000 -53.2543657
## 20 19.791667 -55.2366556
## 21 20.833333 -50.8176884
## 22 21.875000 -61.5743209
## 23 22.916667 -70.7395573
## 24 23.958333 -62.7789135
## 25 25.000000 -61.8887984
```

```
x = runif(n = 25, 0, 10)
```

b. Fit the model to the simulated data

```
sim_data_x = sim_slr(x=x, beta_0 = 5, beta_1 = -3, sigma = sqrt(10.24))
sim_fit_x = lm(response~predictor,data = sim_data_x)
sim_fit_x
```

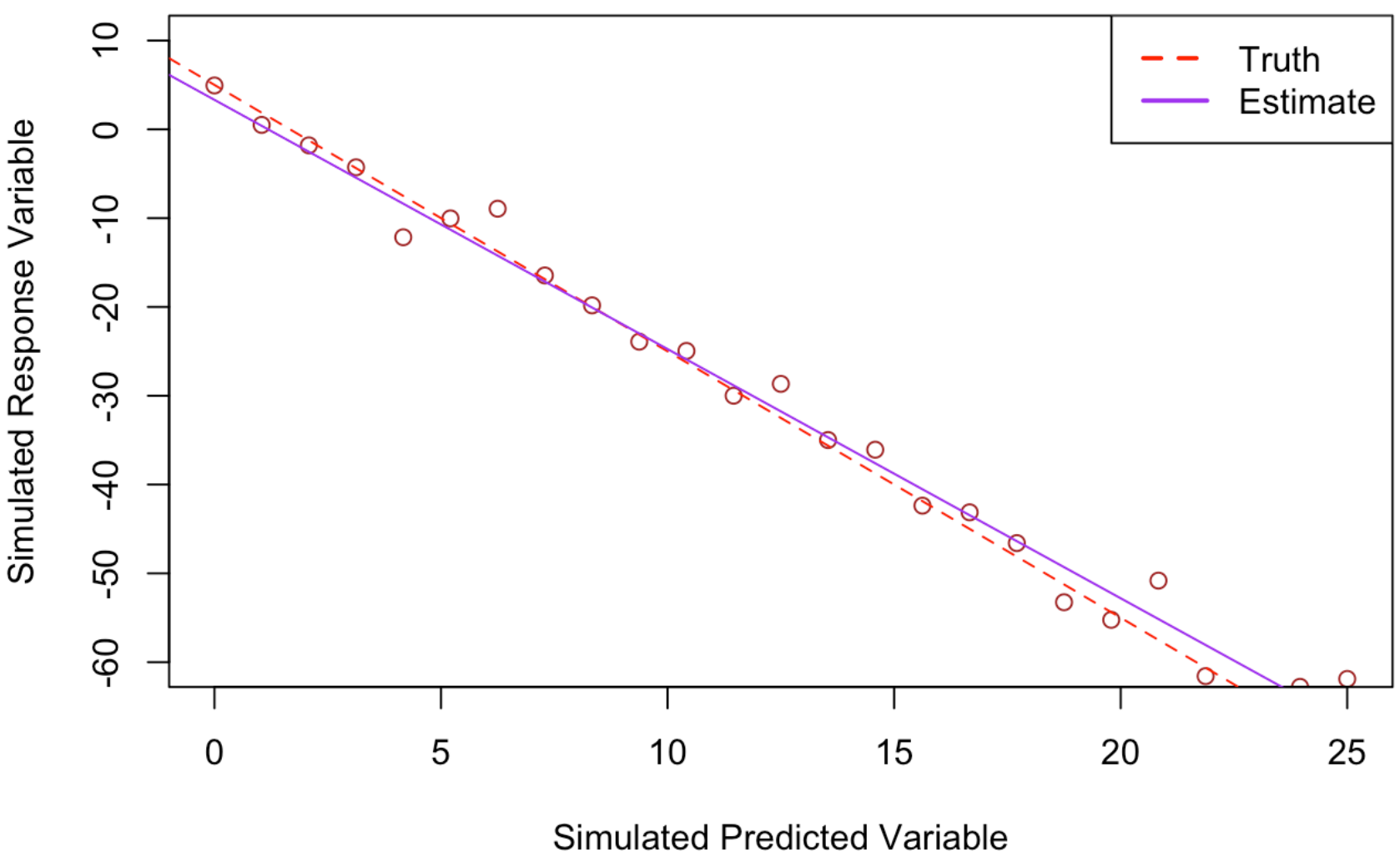
```
##
## Call:
## lm(formula = response ~ predictor, data = sim_data_x)
##
## Coefficients:
## (Intercept)    predictor
##       3.330       -2.807
```

The estimated coefficient is **beta_hat_0**: **3.3303019** and **beta_hat_1** is **-2.8068654**

The predicted response though close, however not identical matching with the simulation data. If we consider the difference between the true value (simulation) versus the prediction value with square the value is comings as 181.4522859, which seems to be not identical match

c. **Scatter plot** of the data, including the True Line Vs Estimated Line

```
plot(response~predictor,
     data=sim_data,
     xlab = "Simulated Predicted Variable",
     ylab = "Simulated Response Variable",
     pch = 1,
     cex = 1,
     col = "brown",
     ylim = c(-60,10))

abline(beta_0,beta_1,lty=2,lwd=1,col="red")
abline(sim_fit_x,lty=1,lwd=1,col="purple")

legend("topright",c("Truth","Estimate"),lty=c(2,1),lwd=2,col=c("red","purple
"))
```



d. Looping to get the **1500** beta_hat_1 after the model fitting

```
beta_hat_1 = rep(0,1500)
for (k in 1:1500){
  temp_sim_data = sim_slr(x=x, beta_0 = 5, beta_1 = -3, sigma = sqrt(10.24))
  temp_sim_slr_fit = lm(response~predictor,data=temp_sim_data)
  beta_hat_1[k] = coef(temp_sim_slr_fit)[2]
}
```

e. Mean and standard deviation of the beta_hat_1

```
mean(beta_hat_1)
```

```
## [1] -3.000044
```

```
sd(beta_hat_1)
```

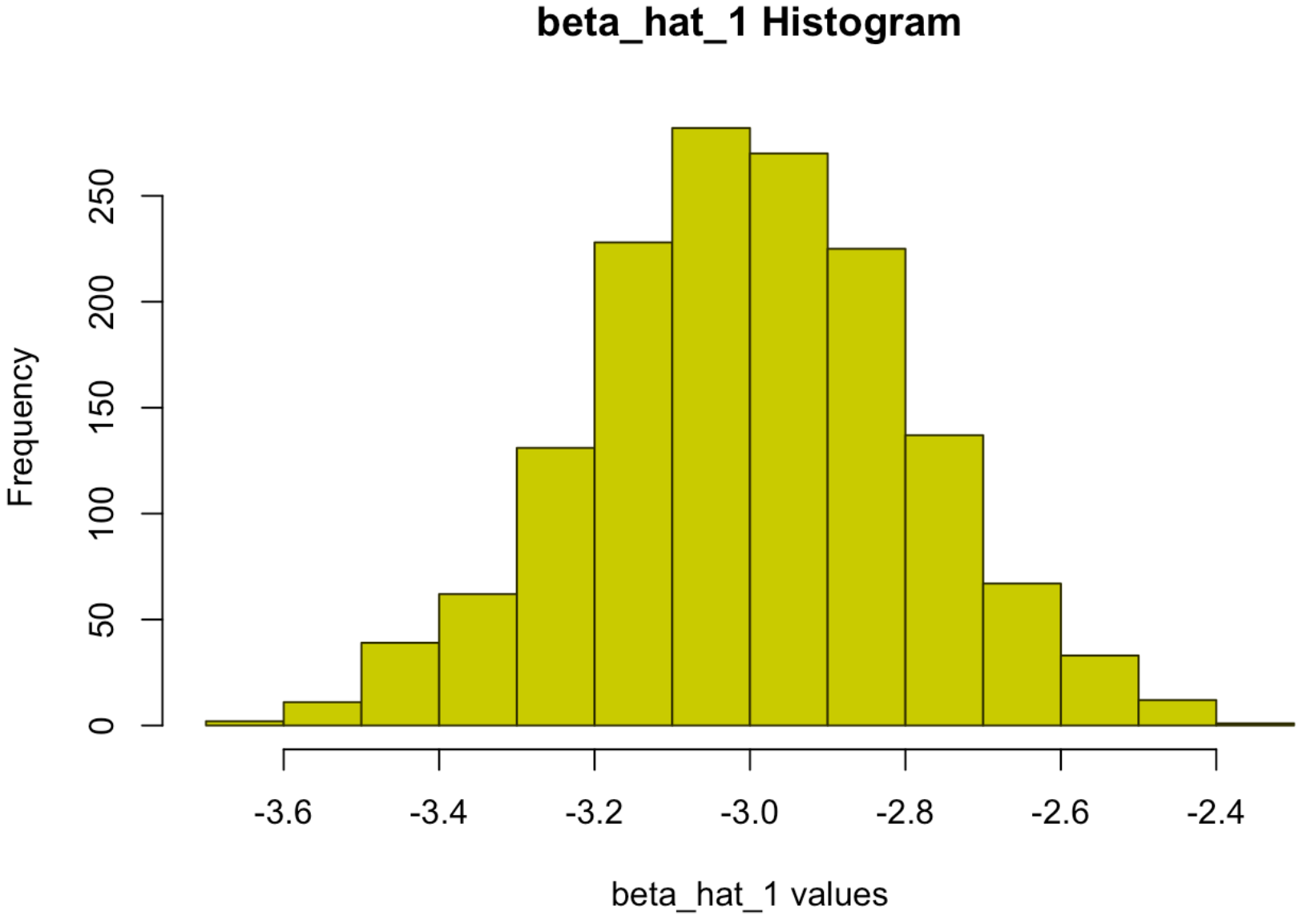
```
## [1] 0.209789
```

Mean is: **-3.0000441** and Standard Deviation (SD) is: **0.209789**

The mean which is **-3.0000441** seems to be somehow closing to the **beta_hat_1** which is calculated from the **point (b)** where the value of the **beta_hat_1** is **-2.8068654**

f. Histogram of the **beta_hat_1**

```
hist(beta_hat_1,
     xlab = "beta_hat_1 values",
     main = "beta_hat_1 Histogram",
     col = "#CCCC00",
     border = "#333300"
)
```



The histogram is normal distributed, where the mean of the distribution is **-3.0000441** and standard deviation (SD): **0.209789** after simulating the data over 1500 times, with everytime it generates different response value.

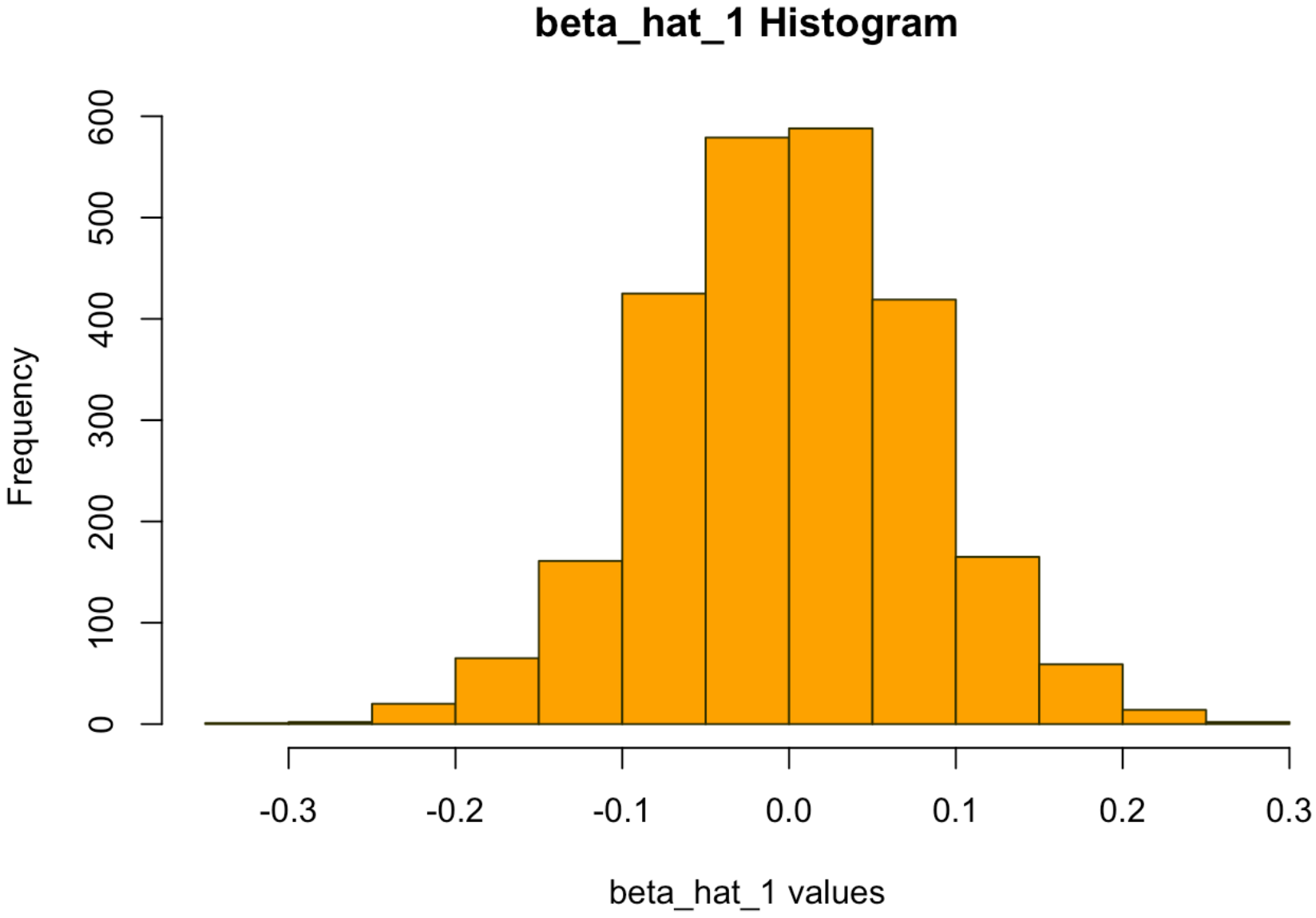
Exercise 4 (Be a Skeptic)

a. Simulate for **n = 75 observations**, where the model needs to be repeated **2500 times**.

```
birthday = 19770411
set.seed(birthday)

x = runif(n = 75, 0, 10)

num_obs = 75
beta_0 = 3
beta_1 = 0
sigma = sqrt(4)

beta_hat_1 = rep(0,2500)
for (k in 1:2500){
  temp_sim_data = sim_slr(x=x, beta_0 = beta_0, beta_1 = beta_1, sigma = sig
ma)
  temp_sim_slr_fit = lm(response~predictor,data=temp_sim_data)
  beta_hat_1[k] = coef(temp_sim_slr_fit)[2]
}
```

b. Create the Histogram

```
hist(beta_hat_1,
     xlab = "beta_hat_1 values",
     main = "beta_hat_1 Histogram",
     col = "orange",
     border = "#333300"
)
```



The histogram is normal distributed, where the mean of the distribution is **-4.346183910^{-4}** after simulating the data over 1500 times, with everytime it generates different response value.

c. Import the skeptic data and calculate the coefficient for beta_1

```
library(readr)
skeptic = read_csv("skeptic.csv")
```

```
## Parsed with column specification:
## cols(
##   predictor = col_double(),
##   response = col_double()
## )
```

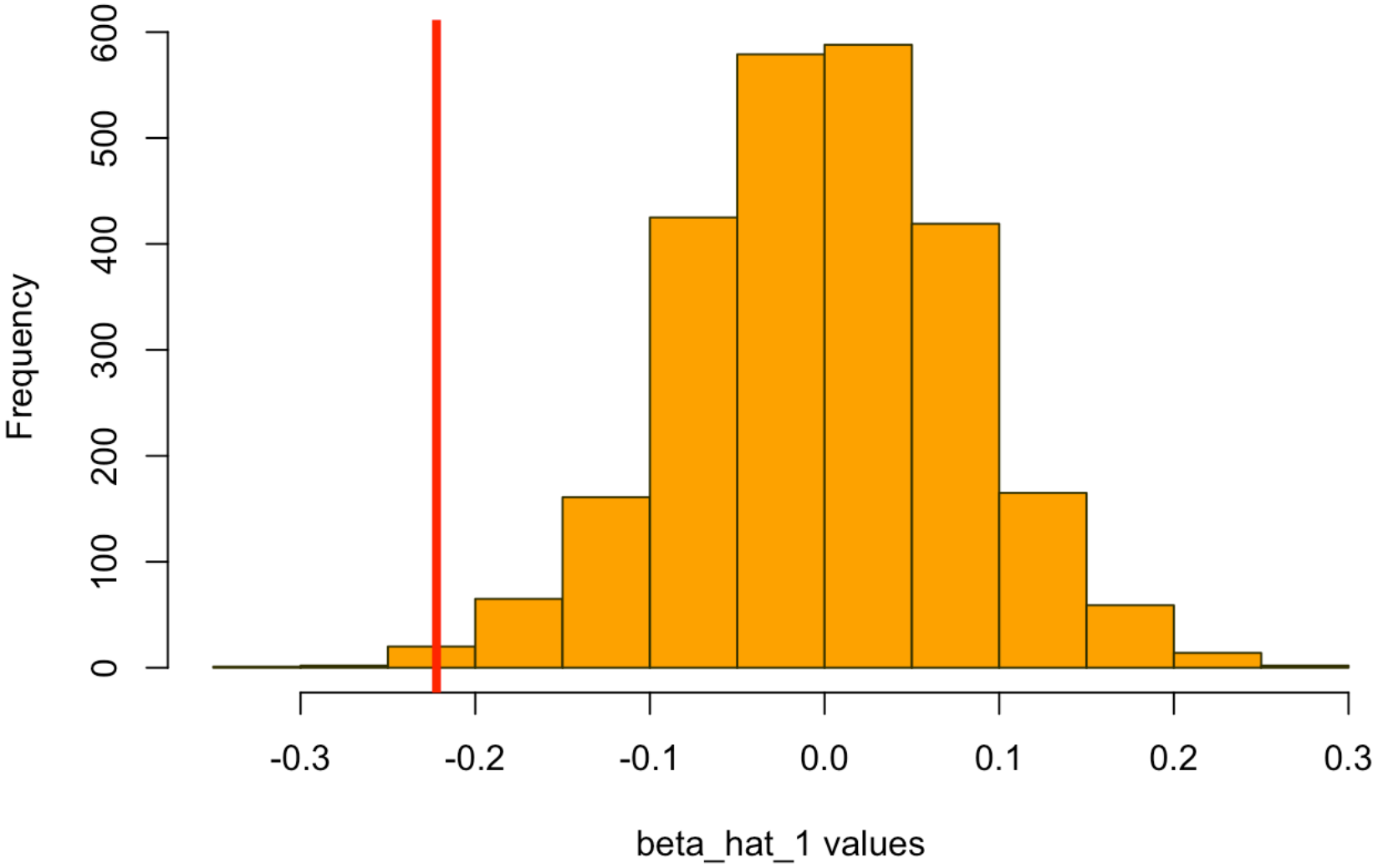
```
skeptic_fit = lm(response~predictor,data=skeptic)
skeptic_fit
```

```
##
## Call:
## lm(formula = response ~ predictor, data = skeptic)
##
## Coefficients:
## (Intercept)    predictor
##      3.1504      -0.2222
```

The fitted Coeficient for beta_1 is: **-0.2221927**

d. Re-Plot the Histogram from b and plot the abline

```
hist(beta_hat_1,
     xlab = "beta_hat_1 values",
     main = "beta_hat_1 Histogram",
     col = "orange",
     border = "#333300"
)
abline(v=coef(skeptic_fit)[2],col="red",lwd=4)
```



e. The **beta_1_hat** value from **(c)** is **-0.2221927** which is **negative**.

The proportion of beta_hat_1 values is smaller than **-0.2221927** is :**0.0027038**. After multiply with 2 the value is coming as : :**0.0054076**

f. Based on the data, it seems the **skeptic.csv** data cann't be generated from the given model in **Exercise 4**. As the estimated **beta_1** after the fitted linear regression from the skeptic data which is:**-0.2221927** is far away from the mean of the beta_hat_1 histogram, where the mean is **-4.346183910^{-4}** which is nearly **zero**. Though the estimated intercept which is :**3.150423** is closer to the **beta_0** of the model which is **3**

Exercise 5 (Comparing Model)

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

ozone_model1 = lm(ozone~wind,data=Ozone)
ozone_model2 = lm(ozone~humidity,data=Ozone)
ozone_model3 = lm(ozone~temp,data=Ozone)

ozone_model1_rmse = sqrt((sum((Ozone$ozone - predict(ozone_model1))^2))/leng
th(Ozone$ozone))
ozone_model2_rmse = sqrt((sum((Ozone$ozone - predict(ozone_model2))^2))/leng
th(Ozone$ozone))
ozone_model3_rmse = sqrt((sum((Ozone$ozone - predict(ozone_model3))^2))/leng
th(Ozone$ozone))


ozone_model1_r2 = (summary(ozone_model1))$r.squared
ozone_model2_r2 = (summary(ozone_model2))$r.squared
ozone_model3_r2 = (summary(ozone_model3))$r.squared

model_compare = data.frame(
  x = c("Model 1","Model 2","Model 3"),
  rmse = c(ozone_model1_rmse,ozone_model2_rmse,ozone_model3_rmse),
  r2 = c(ozone_model1_r2,ozone_model2_r2,ozone_model3_r2)
)

kable(model_compare, format = "pandoc",padding = 2,caption = "Model Comparis
ion Table Between Model 1 Vs Model 2 Vs Model 3")
```

Model Comparision Table Between Model 1 Vs Model 2 Vs Model 3

x	rmse	r2
Model 1	7.961695	0.0001402
Model 2	7.147822	0.1941105
Model 3	5.009257	0.6042011

Based on the above Table againist the 3 models, the **Model 3** seems to be most helpful. The reason to choose the **Model 3** is because the RMSE (Error between the True Vs the Predicted) is small among all the 3 models and **R squared value (Coefficient of determination)** (which is the propotion of the observed variation for the response) is highest among the 3 models.