# HW2

Nathan Krieger

2026-01-30

## Problem 1

### (a)

$\hat{\beta}x_i = \hat{\alpha}z_i$ $\hat{\beta} = c\hat{\alpha}$

$\hat{\alpha}_j = \frac{\hat{\beta}_j}{c}$

### (b)

### (c)

### (d)

I would say this is incorrect because the student forgot the $\epsilon$.

### (e)

False. You can make the training model whatever size you want.

### (f)

Unbiased means that if you did the experiment a large amount of times and calculate $\hat{\beta}$ for each of those times, the average of all the estimates would exactly equal $\beta$.

## Problem 2

```
#install.packages("ISLR2") #you only need to do this one time.
library(ISLR2) #you will need to do this every time you open a new R session.
```

### (a)

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

```r
set.seed(1)

n <- nrow(Boston)

train_index <- sample(1:n, floor(n / 2), replace = FALSE)


train_boston <- Boston[train_index, ]
test_boston <- Boston[-train_index, ]

m1 <- lm(crim ~ ., data = train_boston)

summary(m1)
```

```
##
## Call:
## lm(formula = crim ~ ., data = train_boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.574  -2.723  -0.566   1.351  57.279
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.665277  10.693890   1.745  0.08219 .
## zn            0.046105   0.028198   1.635  0.10336
## indus        -0.127032   0.116665  -1.089  0.27730
## chas         -0.916885   1.769171  -0.518  0.60476
## nox         -11.606805   7.924234  -1.465  0.14431
## rm            0.738859   0.913675   0.809  0.41951
## age          -0.010585   0.026291  -0.403  0.68761
## dis          -1.184115   0.427288  -2.771  0.00602 **
## rad           0.671788   0.130702   5.140  5.7e-07 ***
## tax          -0.004607   0.007552  -0.610  0.54237
## ptratio      -0.515160   0.284824  -1.809  0.07175 .
## lstat         0.296310   0.114591   2.586  0.01031 *
## medv         -0.249594   0.097588  -2.558  0.01115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.693 on 240 degrees of freedom
## Multiple R-squared:  0.5021, Adjusted R-squared:  0.4772
## F-statistic: 20.17 on 12 and 240 DF,  p-value: < 2.2e-16
```

```r
trainMSE <- sum(m1$residuals^2)/nrow(train_boston)

trainMSE
```

```
## [1] 42.49345
```

```r
test_predict <- predict(m1, newdata = test_boston)

test_MSE <- mean((test_predict - test_boston$crim)^2)

test_MSE
```

```
## [1] 41.19923
```

### (b)

```r
set.seed(1)

n <- nrow(Boston)

train_index <- sample(1:n, floor(n / 2), replace = FALSE)


train_boston <- Boston[train_index, ]
test_boston <- Boston[-train_index, ]

m1 <- lm(crim ~ zn + indus + nox + dis + rad + ptratio + medv, data = train_boston)

summary(m1)
```

```
##
## Call:
## lm(formula = crim ~ zn + indus + nox + dis + rad + ptratio +
##     medv, data = train_boston)
##
## Residuals:
##    Min      1Q Median      3Q     Max
## -9.145 -2.542 -0.645  1.175 57.028
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  29.29068    8.84292   3.312 0.001065 **
## zn            0.04940    0.02779   1.777 0.076733 .
## indus        -0.13410    0.10956  -1.224 0.222122
## nox         -12.25307    7.66941  -1.598 0.111408
## dis          -1.36389    0.39946  -3.414 0.000748 ***
## rad           0.63065    0.07015   8.990  < 2e-16 ***
## ptratio      -0.57530    0.28209  -2.039 0.042483 *
## medv         -0.35150    0.06458  -5.443 1.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.739 on 245 degrees of freedom
## Multiple R-squared:  0.4847, Adjusted R-squared:   0.47
## F-statistic: 32.93 on 7 and 245 DF,  p-value: < 2.2e-16
```

```r
trainMSE <- sum(m1$residuals^2)/nrow(train_boston)

trainMSE
```

```
## [1] 43.97466
```

```r
test_predict <- predict(m1, newdata = test_boston)

test_MSE <- mean((test_predict - test_boston$crim)^2)

test_MSE
```

```
## [1] 39.62763
```

The training MSE collected from part A was slightly smaller than the one from part B. The test MSE collected from part A was slightly larger than the one from part B.

### (c)

I expected part b to have a larger MSE because it is being trained on less variables so it's not likely to be as good as part a that was trained on every variable.

### (d)

## Problem 3

### (a)

$\beta_0 = 2$ $\beta_1 = 3$ $\beta_2 = 5$

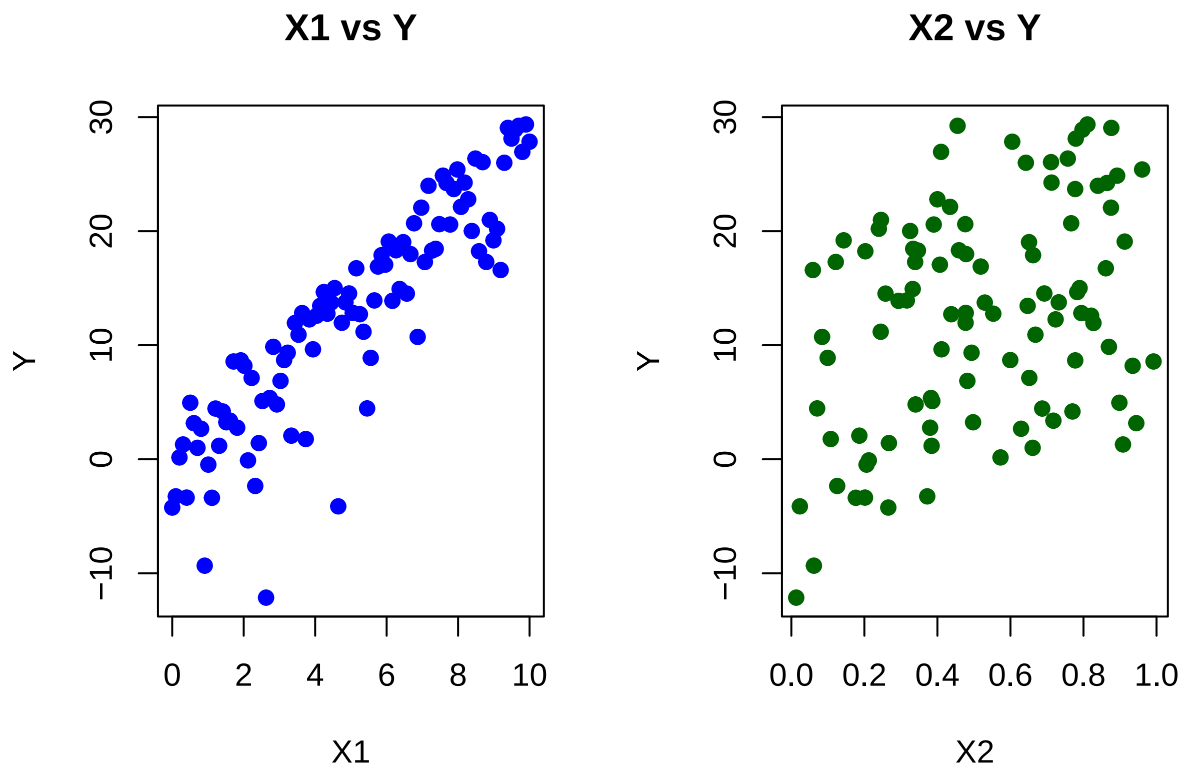
### (b)

```r
set.seed(1) # For reproducibility
X1 = seq(0, 10, length.out = 100)
X2 = runif(100)
epsilon = rnorm(100, mean = 0, sd = 1)

Y = 2 + 3*X1 + 5*log(X2) + epsilon
Y
```

```
##   [1]  -4.23243317  -3.25163817   0.16155277   1.29831740  -3.36017219
##   [6]   4.95979463   3.16639013   1.00554051   2.67674841  -9.32815238
##  [11]  -0.46809191  -3.37647122   4.44916434   1.18318278   4.19129737
##  [16]   3.24545027   3.38444034   8.57643582   2.77034153   8.67147697
##  [21]   8.19849529  -0.09879487   7.13633701  -2.33945369   1.42069144
##  [26]   5.10909193 -12.13061519   5.37632558   9.86110241   4.80943921
##  [31]   6.87401554   8.70101323   9.34431309   2.07223625  11.94948013
##  [36]  10.92516778  12.82034189   1.77720644  12.26835426   9.64280773
##  [41]  12.59220416  13.45553046  14.66413305  12.76885860  13.74312926
##  [46]  15.01216225  -4.12701656  11.97037589  13.76311067  14.53952062
##  [51]  12.83644531  16.74957378  12.72008067  11.18201057   4.46102116
##  [56]   8.89426498  13.93063644  16.90011981  17.89753266  17.06416690
##  [61]  19.09030520  13.89557628  18.32735160  14.93305155  19.03933549
##  [66]  14.53050849  17.99498458  20.69307892  10.73023226  22.06593930
##  [71]  17.29846356  23.98309105  18.30688716  18.45521373  20.61605275
##  [76]  24.86960493  24.22779022  20.58752240  23.69519179  25.41423139
##  [81]  22.13662294  24.26178623  22.79845699  20.01888537  26.36971877
##  [86]  18.24079376  26.05506809  17.30404313  20.99204629  19.19887827
##  [91]  20.21505958  16.59568737  26.00022507  29.05787821  28.11965954
##  [96]  28.90449384  29.24380140  26.95436970  29.36243300  27.84620901
```

### (c)

```r
par(mfrow=c(1,2))
plot(X1, Y, main="X1 vs Y", col="blue", pch=19)
plot(X2, Y, main="X2 vs Y", col="darkgreen", pch=19)
```



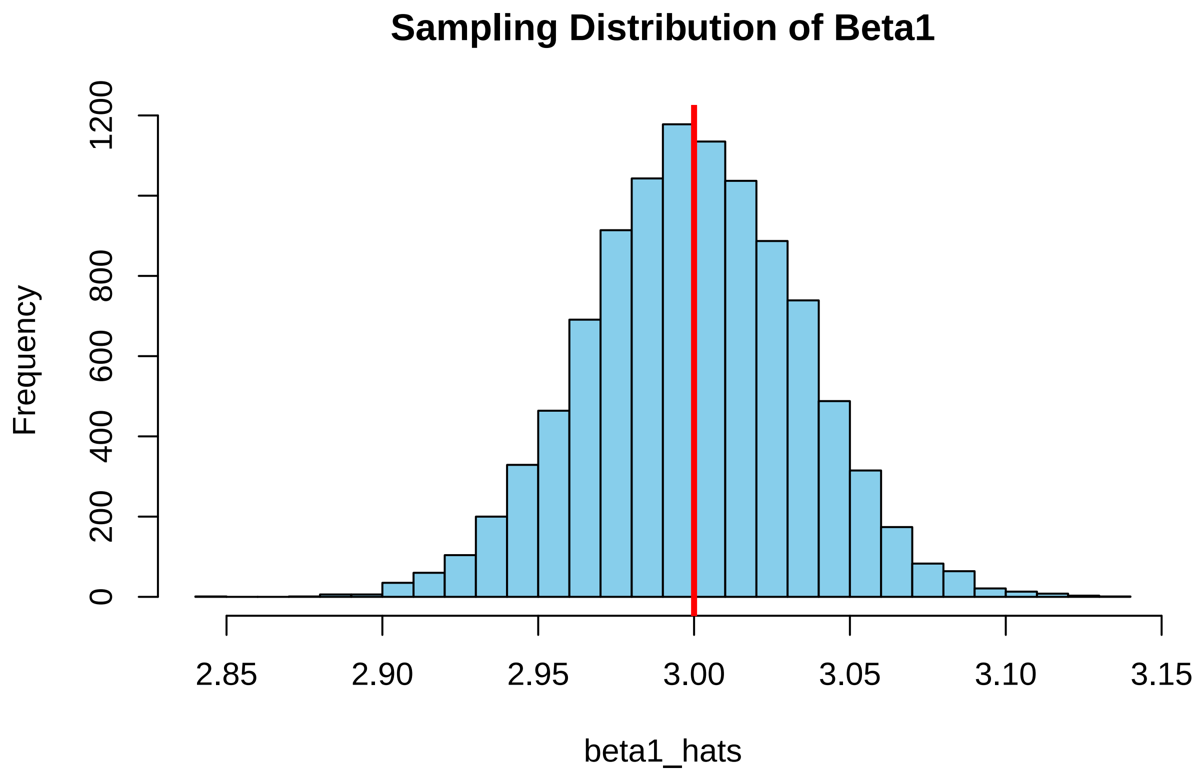

**(d)**

```r
n_sim <- 10000
beta1_hats <- rep(NA, n_sim)

for(i in 1:n_sim) {
  # We generate new noise each time to simulate a new sample
  eps_sim <- rnorm(100, 0, 1)
  Y_sim <- 2 + 3*X1 + 5*log(X2) + eps_sim

  # Fit the model (using log(X2) since that is the true relationship)
  fit <- lm(Y_sim ~ X1 + log(X2))
  beta1_hats[i] <- coef(fit)[2]
}

mean(beta1_hats)
```

```
## [1] 2.999792
```

**(e)**

```r
hist(beta1_hats, breaks=30, col="skyblue", main="Sampling Distribution of Beta1")
abline(v = 3, col = "red", lwd = 3) # The true Beta1
```



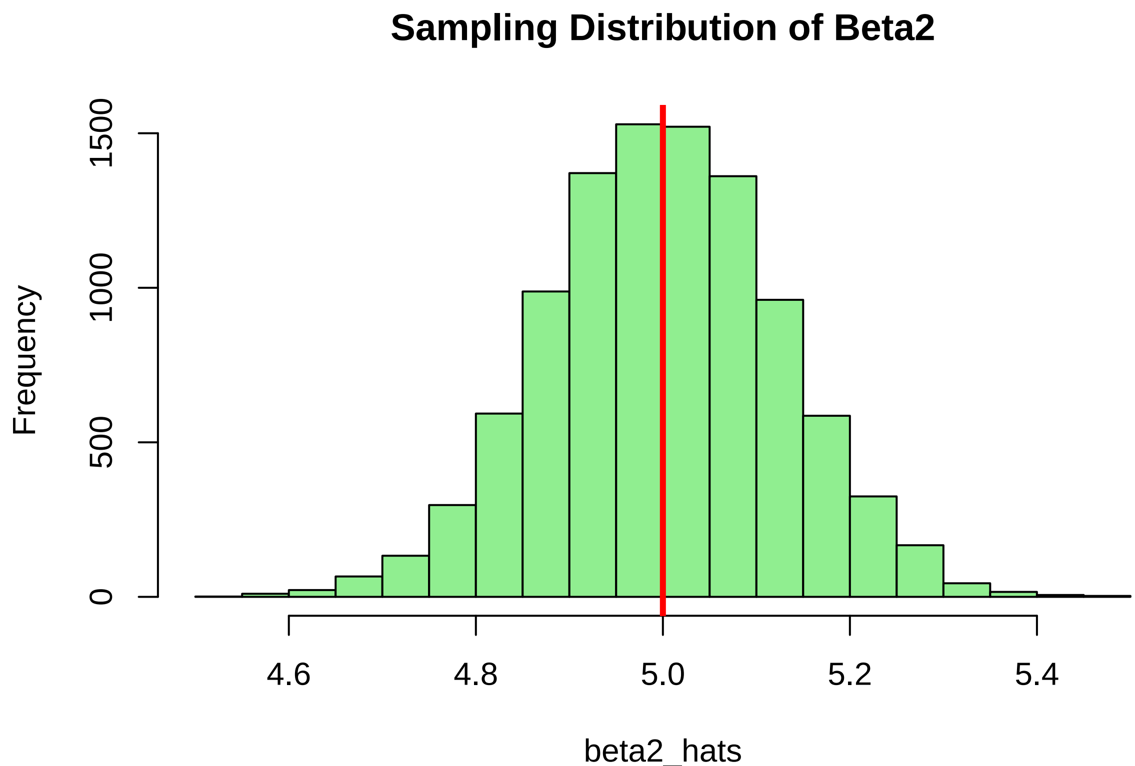

## (f)

```
beta2_hats <- rep(NA, n_sim)

for(i in 1:n_sim) {
  eps_sim <- rnorm(100, 0, 1)
  Y_sim <- 2 + 3*X1 + 5*log(X2) + eps_sim
  fit <- lm(Y_sim ~ X1 + log(X2))
  beta2_hats[i] <- coef(fit)[3]
}

mean(beta2_hats)
```

```
## [1] 4.999968
```

### (g)

```
hist(beta2_hats, breaks=30, col="lightgreen", main="Sampling Distribution of Beta2")
abline(v = 5, col = "red", lwd = 3) # The true Beta2
```


Sampling Distribution of Beta2
Frequency
0
500
1000
1500
4.6
4.8
5.0
5.2
5.4
beta2_hats