Lab 2

Ming Zhong UNI:mz2692

September 24, 2018

Instructions

Before the lab is due, make sure that you upload a knitted HTML or pdf file to the canvas page (this should have a .html or .pdf extension). You also need to change the file name: replace "UNI" by your own uni.

Part (A): Simple Linear Regression Model

1) Import the **diamonds_small.csv** dataset into R and store in a dataframe called **diamonds**. Use the **lm()** command to regress **price** (response) on **carat** (predictor) and save this result as **lm0**. What are the coefficients of **lm0**? (Some of this problem is solved for you below.)

```
# You'll want to type your response to question A(1) here.  Your response should look like:
setwd("/Users/zhongming/Downloads")
diamonds <- read.csv("diamonds_small.csv", as.is = TRUE, header = TRUE)
rows       <- dim(diamonds)[1]
diamonds <- diamonds[sample(1:rows, 2000), ]
lm0<-lm(diamonds$price~diamonds$carat)
lm0
```

```
##
## Call:
## lm(formula = diamonds$price ~ diamonds$carat)
##
## Coefficients:
##    (Intercept)  diamonds$carat
##          -2204            7624
```

```
# Of course, your answer should not be commented out.
```

Recall from lecture that the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ that you just calculated with **lm()** are functions of the data values and are therefore themselves are random (they inherit variability from the data). If we were to recollect the diamonds data over and over again, the estimates would be different each time.

In this lab we'll use bootstrapping to answer the following questions:

1. **"How much does $\hat{\beta}_1$ vary from one replication of the experiment to the other?"**
2. **"What are all the values of β_1 that would have produced this data with high probability?"**

Part (B): How Does $\hat{\beta}_1$ Vary?

Strategy: we'll re-sample **(price, carat)** pairs in order to provide an estimate for how $\hat{\beta}_1$ varies across samples.

1) How many rows are in the **diamonds** dataset? Call this value **n**.

```
# You'll want to type your response to question B(1) here.  Your response should look like:
n <- nrow(diamonds)
n
```

```
## [1] 2000
```

2) We'll next use the **sample()** function to re-sample **n** rows of the **diamonds** dataset *with replacement*. The following code provides a single re-sample of the values $1, 2, \ldots, n$, or a single re-sample of the rows of the dataset.

```
resample1 <- sample(1:n, n, replace = TRUE)
# Remove the comment symbol in front of the above after n is assigned in B(1)
```

Now write a loop to calculate **B <- 1000** such re-samples and store them as rows of the matrix **resampled_values** which will have **B** rows and **n** columns.

```
# You'll want to type your response to question B(2) here.  Your response should look like:
 B <- 1000
 resampled_values <- matrix(NA, nrow = B, ncol = n)
 for (b in 1:B) {
   resampled_values[b,] <- sample(1:n,n,replace=T)
 }
```

3) Now we'll use each re-sampled dataset to provide a new estimate of $\hat{\beta}_1$. Write a line of code that uses **resample1** above to produce a resamples dataset of **(price, carat)** pairs. Using the re-sampled dataset, use **lm()** to produce new estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$. These values should be stored in a vector called **resample1_ests**.

```
# You'll want to type your response to question B(3) here.  Your response should look like:
 resample_data <- diamonds[resample1,]
 resample_data=data.frame(carat=unlist(resample_data[[1]]),price=unlist(resample_data[[2]]))
 resample1_ests <- lm(resample_data$price~resample_data$carat)
 resample1_ests
```

```
##
## Call:
## lm(formula = resample_data$price ~ resample_data$carat)
##
## Coefficients:
##         (Intercept)  resample_data$carat
##               -2310                 7820
```

4) Repeat the above call for each re-sampled dataset produced from the **resampled_values** matrix. We'll store the new coefficient estimates in a matrix **resampled_ests** with **B** rows and **2** columns. Again you'll want to write a loop, this time that iterates over the rows of **resampled_values**. (Note that if you are very clever this could be done using **apply()**.) Make sure to print **head(resample_ests)** at the end.

```
# You'll want to type your response to question B(4) here.  Your response should look like:
 resampled_ests <- matrix(NA, nrow = B, ncol = 2)
 names(resampled_ests) <- c("Intercept_Est", "Slope_Est")
 for (b in 1:B) {
   resampled_ests[b,] <- coefficients(lm(diamonds[resampled_values[b,],]$price~diamonds[resampled_value
 }
 head(resampled_ests)
```

```
##           [,1]     [,2]
## [1,] -2071.804 7429.441
```

```
## [2,] -2350.208 7892.800
## [3,] -2050.613 7372.194
## [4,] -2210.127 7622.836
## [5,] -2258.332 7673.258
## [6,] -2301.808 7806.000
```

Hint: You may want to use multiple lines of code within the for loop. One idea is to first use the rows corresponding to re-sample **b** provided in **resampled__values** to created a resampled dataset. Then use the new dataset to provide new estimates of the coefficients.

5) Recall from lecture that $(\hat{\beta}_1^{(b)})_{b=1}^B - \hat{\beta}_1$ approximates the sampling distribution of $\hat{\beta}_1 - \beta_1$ where β_1 is the population parameter, $\hat{\beta}_1$ is the estimate from out original dataset, and $(\hat{\beta}_1^{(b)})_{b=1}^B$ are the B bootstrap estimates.

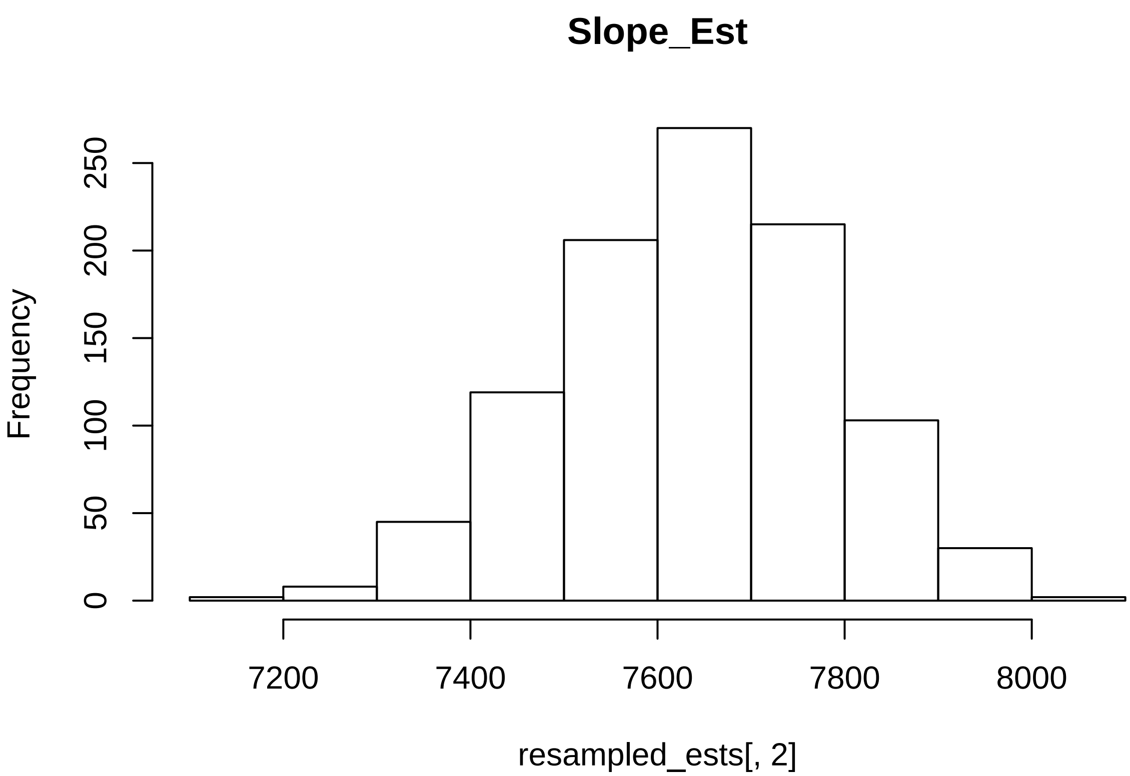
Make a vector **diff__estimates** that holds the differences between the original estimate of $\hat{\beta}_1$ from **lm0** and the bootstrap estimates. It should have length **B**.

```
# You'll want to type your response to question B(5) here.  Your response should look like:
 diff_estimates <-resampled_ests[,2]-coefficients(lm0)[2]
 length(diff_estimates)
```

```
## [1] 1000
```

6) Plot a histogram of the bootstrap estimates of $\hat{\beta}_1$ (they're in the 'Slope_Est' column). Label the x-axis appropriately.

```
# You'll want to type your response to question B(6) here.
hist(resampled_ests[,2],main="Slope_Est")
```



7) Calculate the standard deviation of the bootstrap estimates.

```
# You'll want to type your response to question B(7) here.
sd(resampled_ests[,2])
```

```
## [1] 143.9599
```

Part (C): Bootstrap Confidence Intervals

Note: This section is optional. If you get the chance to do it during lab, great, but it's not necessary that this part is completed when you turn in the lab.

Finally we'd like to approximate confidence intervals for the regression coefficients. Recall that a confidence interval is a random interval which contains the truth with high probability (the confidence level). If the confidence interval for β_1 is C, and the confidence level is $1 - \alpha$, then we want

$$Pr(\beta_1 \in C) = 1 - \alpha$$

no matter what the true value of β_1.

We estimate the confidence interval from the bootstrap estimates by finding a range of $(\hat{\beta}_1^{(b)})_{b=1}^B - \hat{\beta}_1$ which holds $1 - alpha$ percent of the values. In our case, let $\alpha = 0.05$, so we estimate a confidence interval with level 0.95.

(1) Let **Cu** and **Cl** be the upper and lower limits of the confidence interval. Use the **quantile()** function to find the 0.025 and 0.975 quantiles of the vector **diff_estimates** calculated in B(5). Then **Cu** is the sum of the original estimate of $\hat{\beta}_1$ from **lm0** with the upper quantile and **Cl** is the sum of the original estimate of $\hat{\beta}_1$ from **lm0** with the lower quantile.

```
# You'll want to type your response to question C(1) here.  Your response should look like:
 Cl <- quantile(diff_estimates,probs=0.025)+coefficients(lm0)[2]
 Cu <- quantile(diff_estimates,probs=0.975)+coefficients(lm0)[2]
 int <- c(Cl, Cu)
 int
```

```
##     2.5%    97.5%
## 7343.093 7907.861
```

(2) Instead if traditional bootstrap intervals, construct **percentile** based bootstrap intervals. Use the **quantile()** function to find the 0.025 and 0.975 quantiles of the vector **resampled_ests[, "Slope_Est"]** calculated in B(4).

```
quantile(resampled_ests[,2],0.025)
```

```
##     2.5%
## 7343.093
```

```
quantile(resampled_ests[,2],0.975)
```

```
##    97.5%
## 7907.861
```