Lab 2

Shreya Rao and sr3843

May 13th, 2021

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). No need to upload the .Rmd file.

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.)

```
diamonds <- read.csv("diamonds_small.csv", as.is = TRUE, header = TRUE)
rows       <- dim(diamonds)[1]
diamonds <- diamonds[sample(1:rows, 2000), ]

lm0 <- lm(price~carat, data=diamonds)
lm0$coefficients
```

```
## (Intercept)       carat
##   -2202.350    7676.059
```

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**.

```
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)
head(resample1)
```

```
## [1]  279  784 1975  274  360  565
```

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.

```
B <- 1000
resampled_values <- matrix(NA, nrow = B, ncol = n)
for (b in 1:B) {
  resampled_values[b,] <- sample(1:n, n, replace = TRUE)
}
```

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**.

```
resampled_data <- diamonds[resample1,]
resample1_ests <- (lm(price~carat, data=resampled_data))$coefficients
resample1_ests
```

```
## (Intercept)       carat
##   -1958.570    7314.552
```

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.

```
resampled_ests <- matrix(NA, nrow = B, ncol = 2)
#resampled_ests2 <- matrix(NA, nrow = B, ncol = 2)
names(resampled_ests) <- c("Intercept_Est", "Slope_Est")

for (b in 1:B) {
  temp_rows <- sample(1:n, n, replace = TRUE)
  temp_diamonds <- diamonds[temp_rows,]
  temp_lm <- lm(price~carat, data=temp_diamonds)
  resampled_ests[b,] <- temp_lm$coefficients
```

```
  #resampled_ests2[b,] <- coefficients(lm(price~carat, data = diamonds[sample(1:n, n, replace=TRUE),]))
}
head(resampled_ests)
```

```
##           [,1]     [,2]
## [1,] -1971.401 7314.313
## [2,] -2307.121 7846.074
## [3,] -1979.134 7353.864
## [4,] -2312.838 7841.392
## [5,] -2260.003 7761.362
## [6,] -2236.296 7713.385
```

```
#head(resampled_ests2)
```

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.

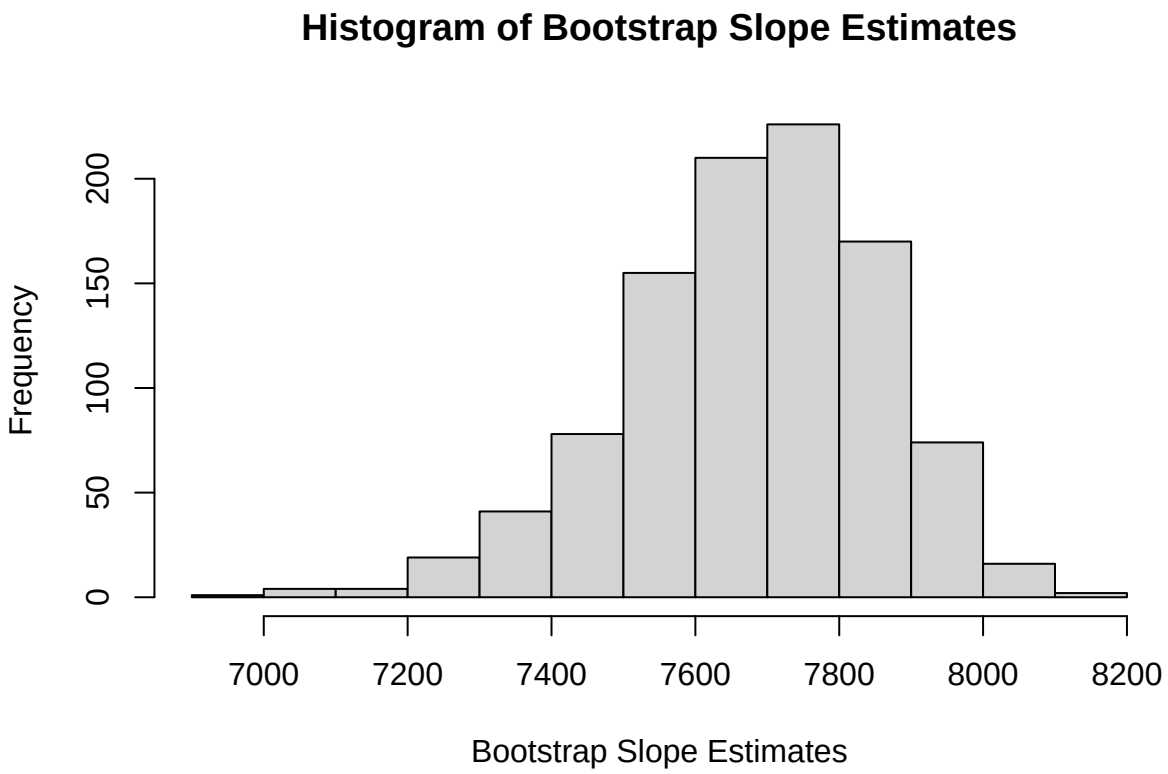
Make a vector **diff_estimates** that holds the differences between the original estimate of $\hat{\beta}_1$ from **lm0** and the bootstrap estimates. It should have length **B**.

```
#lm0$coefficients[2]
#head(resampled_ests[,2])
diff_estimates <- resampled_ests[,2] - lm0$coefficients[2]
head(diff_estimates)
```

```
## [1] -361.74568  170.01529 -322.19448  165.33287   85.30283   37.32559
```

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

```
hist(resampled_ests[,2], xlab="Bootstrap Slope Estimates", main="Histogram of Bootstrap Slope Estimates
```



7) Calculate the standard deviation of the bootstrap estimates.

```
#Intercept sd
sd(resampled_ests[,1])
```

```
## [1] 124.8019
```

```
#Slope sd
sd(resampled_ests[,2])
```

```
## [1] 176.6167
```

```
apply(data.frame(resampled_ests), 2, sd)
```

```
##       X1       X2
## 124.8019 176.6167
```

```
#data.frame(resampled_ests)
```

Part (C): Bootstrap Confidence Intervals

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.

```
Cl <- quantile(diff_estimates, 0.025) + lm0$coefficients[2]
Cu <- quantile(diff_estimates, 0.975) + lm0$coefficients[2]
int_1 <- c(Cl, Cu)
int_1
```

```
##     2.5%    97.5%
## 7290.344 7985.111
```

```
#alternate formula
#Cl_a <- 2*coefficients(lm0)[2] - quantile(resampled_ests[,2], 0.975)
#Cu_a <- 2*coefficients(lm0)[2] - quantile(resampled_ests[,2], 0.025)
#int_a <- c(Cl_a, Cu_a)
#int_a
```

(2) Instead of 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).

```
u_percentile <- quantile(resampled_ests[, 2], 0.975)
l_percentile <- quantile(resampled_ests[, 2], 0.025)
int_2 <- c(l_percentile, u_percentile)
int_2
```

```
##     2.5%    97.5%
## 7290.344 7985.111
```