MACS 33002: PSET 2

Deniz Turkcapar

1. (10 points) Estimate the MSE of the model using the traditional approach. That is, fit the linear regression model using the entire dataset and calculate the mean squared error for the entire dataset. Present and discuss your results at a simple, high level.

```
electData <- read_csv("nes2008.csv")
```

```
Parsed with column specification:
cols(
  biden = col_double(),
  female = col_double(),
  age = col_double(),
  educ = col_double(),
  dem = col_double(),
  rep = col_double()
)
```

```
reg <- lm(biden ~ female + age + educ + dem + rep, data=electData)
summary(reg)
```

```

Call:
lm(formula = biden ~ female + age + educ + dem + rep, data = electData)

Residuals:
   Min     1Q Median     3Q    Max 
-75.55 -11.29   1.02  12.78  53.98 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  58.8113     3.1244   18.82  < 2e-16 ***
female        4.1032     0.9482    4.33 0.000016 ***
age           0.0483     0.0282    1.71    0.088 .  
educ         -0.3453     0.1948   -1.77    0.076 .  
dem          15.4243     1.0680   14.44  < 2e-16 ***
rep         -15.8495     1.3114  -12.09  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.9 on 1801 degrees of freedom
Multiple R-squared:  0.282, Adjusted R-squared:  0.28 
F-statistic:  141 on 5 and 1801 DF,  p-value: <2e-16
```

```
mse <- MSE(y_pred = reg$fitted.values, y_true = electData$biden)
mse
```

```
[1] 395.3
```

```r
m <- mean((reg$fitted.values-electData$biden)^2)
m
```

```
[1] 395.3
```

Theoretically, a smaller MSE shows a better fit of the model to our data. However, a significantly small MSE can mean that the data is being overfit, which is a bad indication for a model. A significantly small MSE would mean that the model perfectly predicts the training data, but it does not have the flexibility to predict and other data. Therefore, we want a balance between underfitting (very high MSE for test/validation data) and overfitting (very low MSE for training data).

In our model above, we can see that the mean squared error is 395.3 when we use the whole data. When we look at the coefficients, we can see that they are significant on all the parameters. The coefficient for female shows that females tend to like Biden. The same can be observed for older people. There is a negative correlation between education and liking Biden, so more educated people tend to like Biden less. As we would expect, democrats tends to like Biden more than independents whereas republicans like him less than the independents do. The R-squared is 0.282, which means that the model is not explaining the variation in the data well. Therefore, it is likely that this model is missing features that could predict a person's feelings towards Biden. The mean squared error is 395.3, which shows that the representative residual is approximately 20 points away. This is concerning because 20 points is larger than our coefficients. Therefore, this model might not have good predictive power.

2. (30 points) Calculate the test MSE of the model using the simple holdout validation approach.

(5 points) Split the sample set into a training set (50%) and a holdout set (50%). Be sure to set your seed prior to this part of your code to guarantee reproducibility of results.

```r
set.seed(1234)

data_split <- initial_split(data = electData,
                            prop = 0.5)
data_train <- training(data_split)
data_test <- testing(data_split)
```

(5 points) Fit the linear regression model using only the training observations.

```r
reg_train <- lm(biden ~ female + age + educ + dem + rep, data=data_train)
sm <- summary(reg_train)
sm
```

```

Call:
lm(formula = biden ~ female + age + educ + dem + rep, data = data_train)

Residuals:
   Min     1Q Median     3Q    Max 
-71.60 -11.48   1.08  13.96  46.03 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  57.8664     4.4325   13.06  < 2e-16 ***
female        5.4727     1.3622    4.02 0.000064 ***
age           0.0398     0.0409    0.97     0.33    
educ         -0.3367     0.2783   -1.21     0.23    
dem          14.9923     1.5190    9.87  < 2e-16 ***
rep         -16.9624     1.8919   -8.97  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.2 on 898 degrees of freedom
Multiple R-squared:  0.284, Adjusted R-squared:  0.28 
F-statistic: 71.2 on 5 and 898 DF,  p-value: <2e-16
```

(10 points) Calculate the MSE using only the test set observations.

```r
mse_test <- mean((data_test$biden - predict.lm(reg_train, data_test)) ^ 2)
mse_test
```

```
[1] 389.2
```

(10 points) How does this value compare to the training MSE from question 1? Present numeric comparison and discuss a bit.

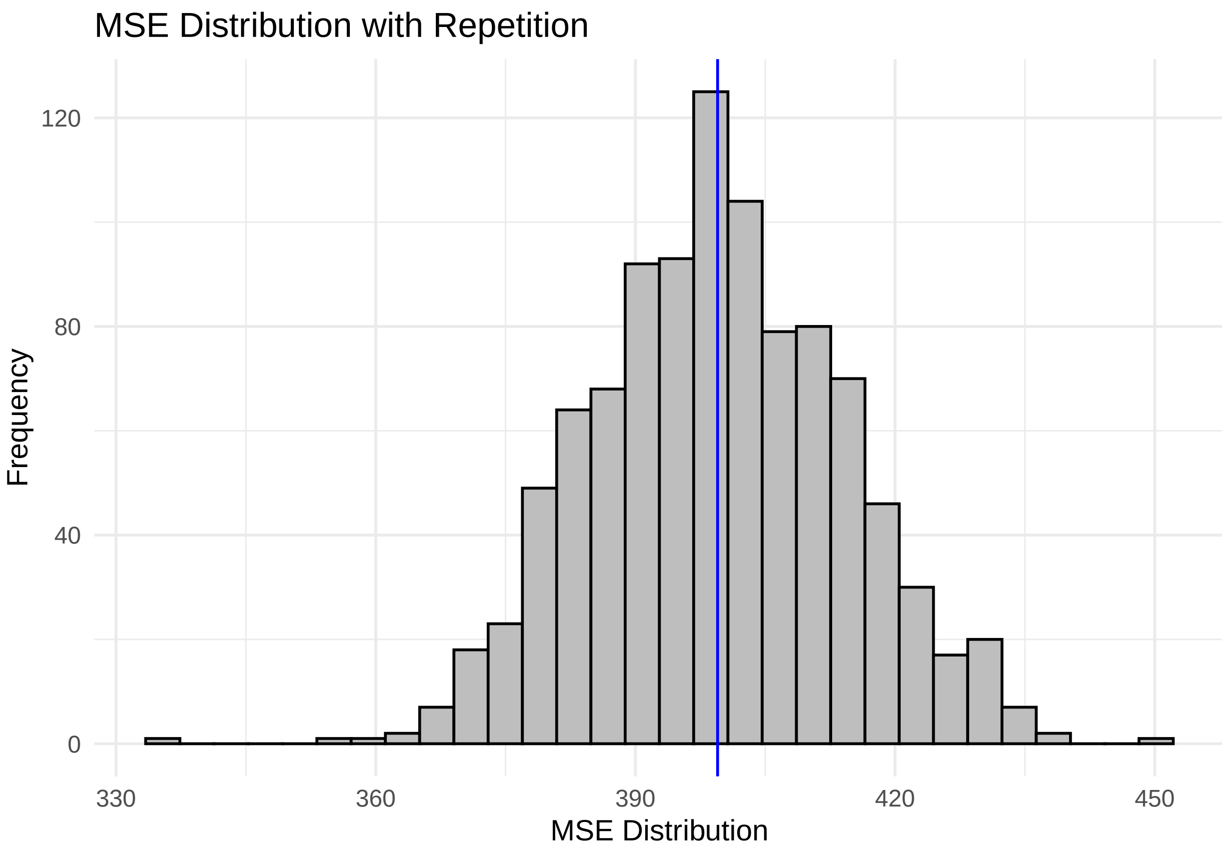
The MSE we've found in Question 1 (395.3) is higher than the one we've found in Question 2 (389.2). We can say that this was somehow not expected becausse fitting the model on less data and testing it on daata that it hasn't observed before would potentially lead to a worse model performance. However, we should keep in mind that this is a random process so we can expect that some seeds will grab smaller values and other seeds might grab larger values.

3. (30 points) Repeat the simple validation set approach from the previous question 1000 times, using 1000 different splits of the observations into a training set and a test/validation set. Visualize your results as a sampling distribution (hint: think histogram or density plots). Comment on the results obtained.

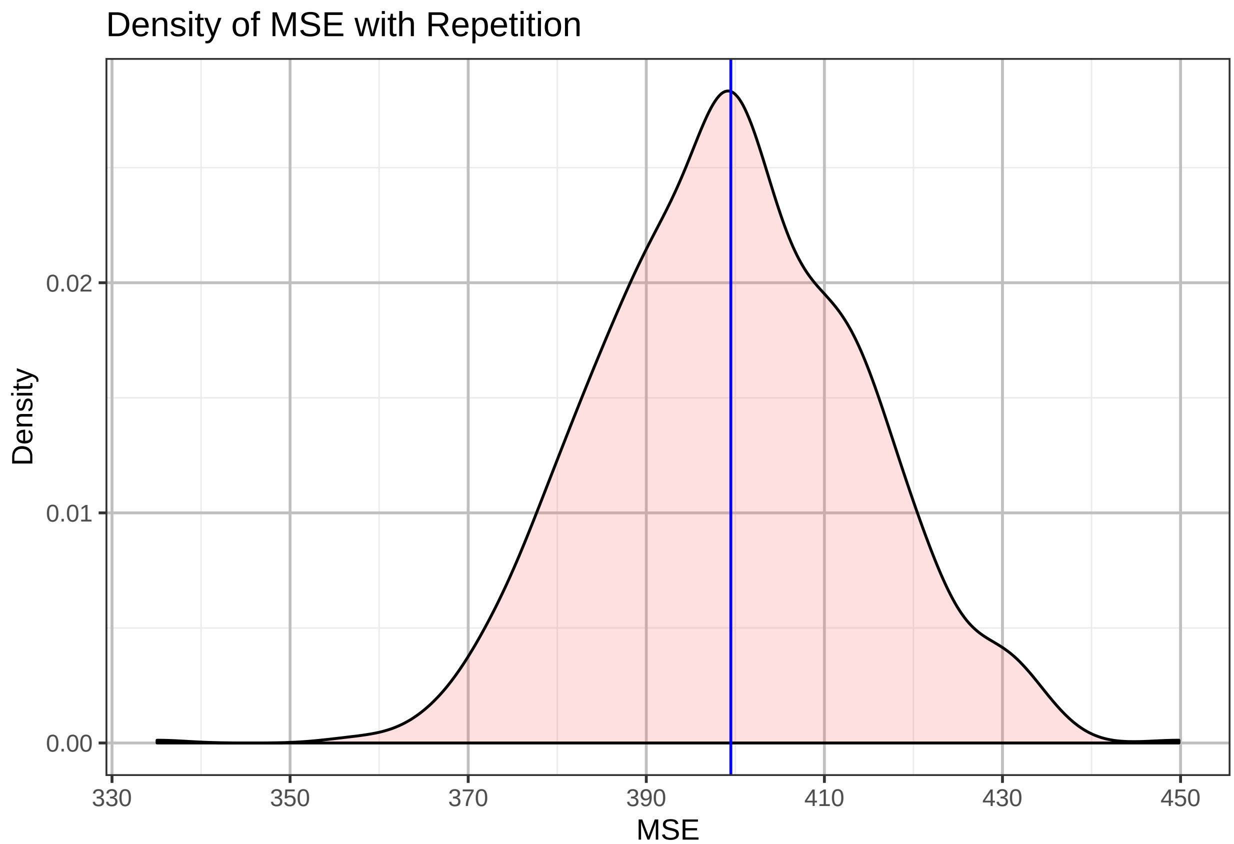
```r
results <- numeric(1000)

for (i in 1:1000) {
  split_1000 <- initial_split(data = electData, prop = 0.5)
  train_1000 <- training(split_1000)
  test_1000 <-  testing(split_1000)
  model_1000 <- lm(biden ~ female + age + educ + dem + rep, data = train_1000)
  mse_test_1000 <- mean((test_1000$biden - predict.lm(model_1000, test_1000)) ^ 2)
  results[i] <- mse_test_1000
}
res_df <- as.data.frame(results)
```

```
ggplot(res_df) + geom_histogram(aes(results), boundary = 1, color="black", fill="gray")+
  theme_minimal() +
  geom_vline(xintercept = mean(results), color = "blue") +
  labs(x = "MSE Distribution", y = "Frequency") +
  ggtitle("MSE Distribution with Repetition")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(res_df, aes(results)) +
geom_density(alpha=.2, fill="#FF6666") +
geom_vline(xintercept = mean(results), color = "blue") + scale_x_continuous(breaks = seq(330, 450,
labs(x = "MSE", y = "Density",
       title = "Density of MSE with Repetition")
```



We can observe that as we take more samples and calculate the mean squared error, we can see thaat the distribution converges to a notmal with a mean approximately 400, which is very close to the MSE calculated using the full dataset. There are still some variations due to the random sampling that we do. We can calculate the spread of the distribution:

```
df <- as.data.frame(results)
summary(df)
```

```
    results
 Min.   :335
 1st Qu.:389
 Median :399
 Mean   :400
 3rd Qu.:410
 Max.   :450
```

As observed, the max and min show a difference in fits of the model. The histogram that we've plotted above also shows the importance of iteration to converge to an optimal result in order to have a calibrated model. By just doing sampling once, we could have been misled if we happened to get a value around the tail of this distribution.

4. (30 points) Compare the estimated parameters and standard errors from the original model in question 1 (the model estimated using all of the available data) to parameters and standard errors estimated using the bootstrap (B = 1000). Comparison should include, at a minimum, both numeric output as well as discussion on differences, similarities, etc. Talk also about the conceptual use and impact of bootstrapping.

```
library(ISLR)
lm_df <-
  summary(reg)$coefficients %>%
  data.frame() %>%
  mutate(term = names(reg$coefficients)) %>%
  select(term, Lm_Estimate = Estimate, Lm_Sd = Std..Error)
lm_df
```

```
         term Lm_Estimate   Lm_Sd
1 (Intercept)    58.81126 3.12444
2      female     4.10323 0.94823
3         age     0.04826 0.02825
4        educ    -0.34533 0.19478
5         dem    15.42426 1.06803
6         rep   -15.84951 1.31136
```

```
lm_coefs <- function(splits, ...) {

  mod <- lm(..., data = analysis(splits))
  tidy(mod)
}

auto_boot <- electData %>%
  bootstraps(1000) %>%
  mutate(coef = map(splits, lm_coefs, as.formula(biden ~ female + age + educ + dem + rep)))

auto_boot %>%
  unnest(coef) %>%
  group_by(term) %>%
  summarize(Boot_Estimate = mean(estimate),
            Boot_Sd= sd(estimate, na.rm = TRUE))
```

```
# A tibble: 6 x 3
  term        Boot_Estimate Boot_Sd
  <chr>               <dbl>   <dbl>
1 (Intercept)       58.7     3.07
2 age                0.0491  0.0284
3 dem               15.4     1.06
4 educ              -0.344   0.197
5 female             4.13    0.960
6 rep              -15.8     1.41
```

Bootstrapping relies on random sampling with replacement to assign measures of accuracy to sample estimates. We know that using a bootstraped estimator is useful because the bootstrap estimator will be more accurate especially when the distributional assumptions are not met. Therefore, it is better to use bootstrapping when we only have access to a smaller sample and/or we cannot make distributional assumptions about the data.

Looking at our results above, we can observe that the bootstrap results are similar to the results we got for the original non-bootstrapped model. The standard deviations are also relatively close to each other. When we take the standard errors into account, the coefficients are even the same as the original model that we've found in Q1. This is a very good sign because we would not have many other ways to compute standard errors on the estimates, so we know that we can use bootstrapping to get robust standard errors.