Rahul Rangwani
Problem Set 2
February 3, 2020
Intro to Machine Learning

1. To run the required regression, I ran the following code:

```
lmMod <- lm(biden ~ female+age+educ+dem+rep, data=dat); summary(lmMod)
```

I got the following output:

```
Residuals:
    Min      1Q  Median      3Q     Max
-75.546 -11.295   1.018  12.776  53.977

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  58.81126    3.12444  18.823  < 2e-16 ***
female        4.10323    0.94823   4.327 1.59e-05 ***
age           0.04826    0.02825   1.708   0.0877 .
educ         -0.34533    0.19478  -1.773   0.0764 .
dem          15.42426    1.06803  14.442  < 2e-16 ***
rep         -15.84951    1.31136 -12.086  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 19.91 on 1801 degrees of freedom
Multiple R-squared:  0.2815,    Adjusted R-squared:  0.2795
F-statistic: 141.1 on 5 and 1801 DF,  p-value: < 2.2e-16
```

To get the MSE, I ran:

```
mse <- mean(lmMod$residuals^2)
```

To interpret the output of the regression, we look first at the coefficients. Here, we see that females are more warm towards Biden than men are, with a 4.1 sentiment point difference. With age, we do not see much differentiation in warmth, as the coefficient is very small at .048 points. It seems that more educated people are slightly colder towards Biden, as the coefficient is -.345 points. Democrats are, as expected, much warmer to Biden than Republicans, with the Democrat coefficient at 15.424 points and the Republican coefficient at -15.849. The MSE calculated was 395.27. The R^2 value we get is .2815, which is fairly low. The R^2 and high MSE value are indicative of not great fit.

2. For this question, to split the data and run a model on only the training data, I used the following code:

```
set.seed(44)
init_split <- initial_split(data = dat, prop = 0.5)
biden_train <- training(init_split)
biden_test <- testing(init_split)

lmMod_train <- lm(biden ~ female+age+educ+dem+rep, data=biden_train); summary(lmMod_train)

test_fitted <- predict(lmMod_train, newdata = biden_test)
mse_train <- mean((biden_test$biden - test_fitted)^2)
```

This was the regression output:

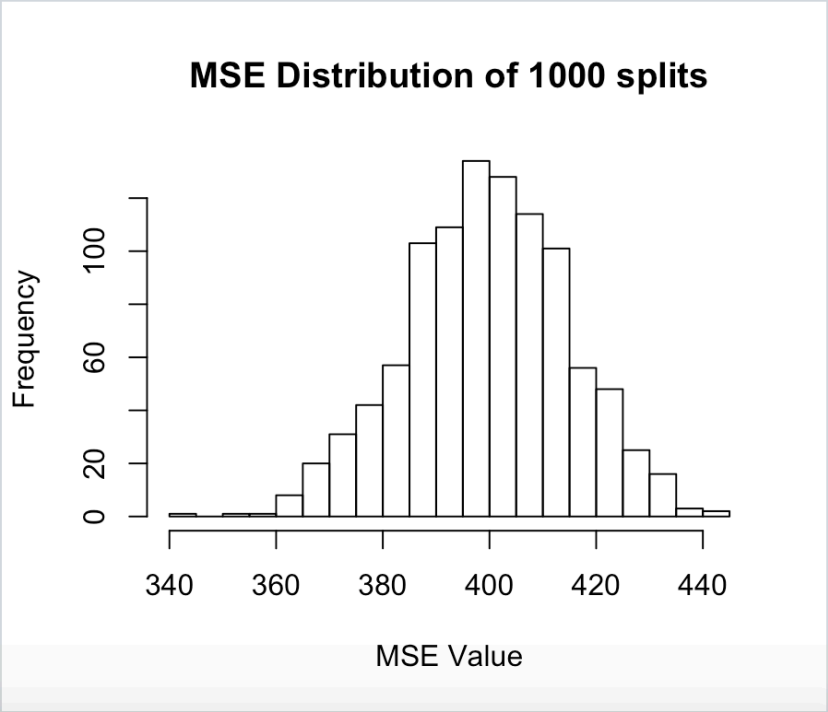
```
Residuals:
    Min      1Q  Median      3Q     Max
-75.331 -10.788   3.203  11.801  53.191

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  55.84275    4.51775  12.361   <2e-16 ***
female        2.82203    1.36711   2.064   0.0393 *
age           0.04277    0.04126   1.037   0.3001
educ         -0.08183    0.27954  -0.293   0.7698
dem          16.45029    1.54622  10.639   <2e-16 ***
rep         -15.45448    1.84499  -8.376   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.29 on 898 degrees of freedom
Multiple R-squared:  0.2815,    Adjusted R-squared:  0.2775
F-statistic: 70.35 on 5 and 898 DF,  p-value: < 2.2e-16
```

Here, we can see that the coefficients are mostly the same. A big change comes with the female coefficient, which in this split, decreased to 2.82 from 4.10. Another relatively large change was with the educ variable, where the coefficient increased from -.34 to -.08. These changes may have simply come from the split of the data. The MSE value was 383.29, which was not too far off the MSE from the first regression, which was 395.27. This value is slightly lower, which may indicate better prediction, but this difference is very small. The R^2 value here is .2815, matching exactly the value from the initial regression. These results show us that the fit of the first regression and the split regression are very similar, and present a consistent model.

3. To run 1000 trials, I used a simple for loop and stored the MSE value in a list:

```
mse_list <- list()

for (i in 1:1000){
  #creating splits
  set.seed(i)
  init_split <- initial_split(data = dat, prop = 0.5)
  biden_train <- training(init_split)
  biden_test <- testing(init_split)

  #running model and calculating MSE
  lmMod_train_iter <- lm(biden ~ female+age+educ+dem+rep, data=biden_train)
  test_fitted <- predict(lmMod_train_iter, newdata = biden_test)
  mse_train <- mean((biden_test$biden - test_fitted)^2)

  #saving MSE values
  mse_list <- c(mse_list,mse_train)
}

hist(as.numeric(mse_list), breaks=30, xlab="MSE", main="MSE Distribution of 1000 splits")
```

I then generated a histogram to see the distribution of the MSE values:



From this histogram, we can see that after running the model over 1000 randomized splits, the MSE value appears normally distributed around 400. What this tells us is that our original model does not appear to be overfit, as the MSE value is very close to the center of this distribution. We can also see here that when we fit the model on only a training set of the data, it performs very slightly worse. Despite this, it still makes sense to generally use a training and test set when tuning the model, as it is useful to have a way to assess the model's accuracy with new data.

4. To use the bootstrap, I ran the following code:

```
b = 1000

dat <- as_tibble(dat)

lmMod_tib <- lm(biden ~ female+age+educ+dem+rep, data = dat)
tidy(lmMod_tib)

# bootstrapped estimates of the parameter estimates and standard errors
lm_coefs <- function(splits, ...) {
  ## use `analysis` or `as.data.frame` to get the analysis data
  mod <- lm(..., data = analysis(splits))
  tidy(mod)
}

biden_boot <- dat %>%
  bootstraps(b) %>%
  mutate(coef = map(splits, lm_coefs, as.formula(biden ~ female+age+educ+dem+rep)))
biden_boot %>%
  unnest(coef) %>%
  group_by(term) %>%
  summarize(.estimate = mean(estimate),
            .se = sd(estimate, na.rm = TRUE))
```

The output was as follows:

```
# A tibble: 6 x 3
  term        .estimate    .se
  <chr>           <dbl>  <dbl>
1 (Intercept)   58.9     3.12
2 age            0.0477 0.0302
3 dem           15.3     1.09
4 educ          -0.349   0.192
5 female         4.11    0.947
6 rep          -15.9     1.41
```

From the original model, we don't see much difference in the bootstrap model. Our coefficient estimates are all very close to the original estimates. For age, dem, and rep, the standard errors are very slightly larger, with the parameters having slightly smaller standard errors. Because bootstrapping does not need assumptions based on the distribution, it is more robust than a standard model. The biggest and most important impact of using bootstrapping is that it allows us to draw conclusions about the whole population given only a sample through resampling. Using this allows us to generate estimators for the standard error and confidence intervals for the distribution in question. An important benefit of bootstrapping is that due to the way the standard error is calculated, the population mean is contained within the 95% confidence interval.