PS2

Joanna Zhang

2/3/2020

1.

```
#load the dataset
nesdata <- read.csv("nes2008.csv")
#run the traditional model
nes_mod <- lm(biden ~ female + age + educ + rep + dem,data = nesdata)
knitr::kable(tidy(nes_mod))
```

term	estimate	std.error	statistic	p.value
(Intercept)	58.8112590	3.1244366	18.822996	0.0000000
female	4.1032301	0.9482286	4.327258	0.0000159
age	0.0482589	0.0282474	1.708438	0.0877274
educ	-0.3453348	0.1947796	-1.772952	0.0764057
rep	-15.8495061	1.3113624	-12.086290	0.0000000
dem	15.4242556	1.0680327	14.441745	0.0000000

```
#use the predicted volues to calculate mse
modmse <- augment(nes_mod, newdata = nesdata) %>%
  mse(truth = biden, estimate = .fitted)
knitr::kable(modmse)
```

.metric	.estimator	.estimate
mse	standard	395.2702

The regression results and the estimated MSE are in the tables above. We see that all the independent variables are significant at a 10% level. The model generates a MSE of 395.27, which means that on average, the squared distance between the actual values and the predicted values is 395.27. This implies that our model may not be very accurate in predicting values, since the feeling thermometer only ranges from 0 to 100, a squared error of about 400 is pretty large. The data set we used to train the model and the data set we used to test the model are the same, so this estimation of the MSE may not be accurate.

2.a

```
set.seed(123)
#split the data into to sets
nes_split <- initial_split(data = nesdata, prop = 0.5)
nes_train <- training(nes_split)
nes_test <- testing(nes_split)
```

2.b

```
nes_mod_train <- lm(biden ~ female + age + educ + rep + dem, data = nes_train)
tidy(nes_mod_train)
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  54.6       4.47      12.2   8.77e-32
## 2 female        2.75      1.35       2.04  4.15e- 2
## 3 age           0.0925    0.0402     2.30  2.16e- 2
## 4 educ         -0.160     0.277     -0.577 5.64e- 1
## 5 rep         -14.5       1.84      -7.87  1.01e-14
## 6 dem          15.1       1.53       9.83  9.81e-22
```

2.c

```
testmse <- augment(nes_mod_train, newdata = nes_test) %>%
  mse(truth = biden, estimate = .fitted)
knitr::kable(testmse)
```

.metric	.estimator	.estimate
mse	standard	392.381

2.d

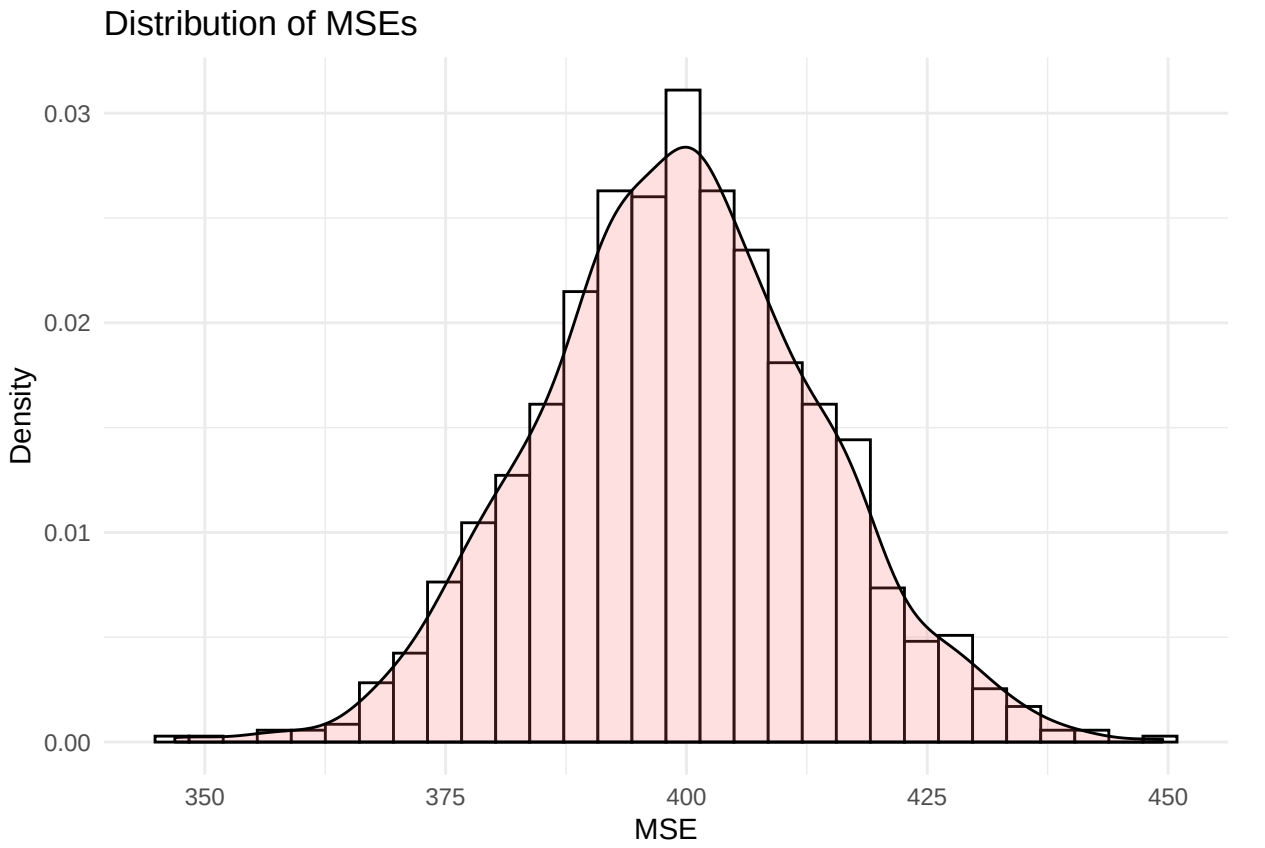
The MSE of the model using a simple holdout validation approach is 392.38. Compared to the MSE in question 1 (395), this MSE is slightly lower. This is unexpected because this time we fit the model on a smaller data set, and test the model on a different test set, while in question 1 we train and test the model on the same data set. The reason why we do not see an significant change in the MSE here is probability that we are randomly drawing half of the sample out. We still get a model that has about the same accuracy as in question 1. Note that since the process of generating the training set is random, we might get a very large or very small MSE if we change the split.

3.

```
set.seed(234)
#create an empty lists to store the MSEs
mse_thousand <- list()
#generate 1000 splits
for (i in 1:1000){
  nes_split_i <- initial_split(data = nesdata, prop = 0.5)
  nes_train_i <- training(nes_split_i)
  nes_test_i <- testing(nes_split_i)
  mod_i <- lm(biden ~ female + age + educ + rep + dem, data = nes_train_i)
  mse_i<-augment(mod_i, newdata = nes_test_i) %>%
    mse(truth = biden, estimate = .fitted)
  #append the MSE to the list that stores all MSEs
  mse_thousand<- c(mse_thousand, mse_i[[3]])
```

```
}
#Convert the list to a dataframe
msedf <- data.frame(matrix(mse_thousand))
names(msedf)[1] <- "MSE"
msedf$MSE <- as.numeric(msedf$MSE)

#Plot the distribution of the MSEs
msedf %>%
  ggplot(aes(x = MSE)) +
  geom_histogram(aes(y=..density..),color = "black", fill = "white")+
  geom_density(alpha=0.2, fill="#FF6666")+
  theme_minimal()+
  labs(title = "Distribution of MSEs", x = "MSE", y = "Density")
```



After calculating the MSE for 1000 different training sets, the distribution of MSE looks like a normal distribution with an average at about 400. Compared to the MSE generated by the model in question 1, the average MSE here is a bit larger. Since we are using 1000 samples here, this shows that the MSEs gradually converges to the real MSE of the model. If we apply this model to individuals in the population instead of the observations in the the data set, the MSE should be around 400. The iteration of sampling is useful because if we only had one split (like in question 2), we may get a misleading MSE that could be as small as 350, or as large as 450.

4.

```
knitr::kable(tidy(nes_mod), caption = "Regression results of traditional model")
```

Table 4: Regression results of traditional model

term	estimate	std.error	statistic	p.value
(Intercept)	58.8112590	3.1244366	18.822996	0.0000000
female	4.1032301	0.9482286	4.327258	0.0000159
age	0.0482589	0.0282474	1.708438	0.0877274
educ	-0.3453348	0.1947796	-1.772952	0.0764057
rep	-15.8495061	1.3113624	-12.086290	0.0000000
dem	15.4242556	1.0680327	14.441745	0.0000000

```
lm_coefs <- function(splits, ...) {
  ## use `analysis` or `as.data.frame` to get the analysis data
  mod <- lm(biden ~ female + age + educ + rep + dem, data = analysis(splits))
  tidy(mod)
}

nes_all <- nesdata %>%
  bootstraps(1000) %>%
  mutate(coef = map(splits, lm_coefs))

boot_est<- nes_all%>%
  unnest(coef) %>%
  group_by(term) %>%
  summarize(.estimate = mean(estimate),
            .se = sd(estimate, na.rm = TRUE))
knitr::kable(boot_est,caption = "Regression results of bootstrapped model")
```

Table 5: Regression results of bootstrapped model

term	.estimate	.se
(Intercept)	58.8293532	2.9324611
age	0.0479316	0.0289848
dem	15.3926008	1.0892455
educ	-0.3471750	0.1889529
female	4.1494624	0.9415222
rep	-15.8299308	1.3890588

After bootstrapping, the coefficients of female, dem, rep became larger, while the coefficients of age and education became smaller. Overall, the bootstrapped estimates are very close to the estimates in question 1. Some of the standard errors of age, dem, rep, and female are larger for the bootstrapped estimates.

Conceptually, the bootstrap process generates new data from the "population" by randomly sampling observations from the data set (with replacement) as a training set, and test the estimates on the observations that have not been selected. It allows us to simulate the original data set. By using the bootstrapping technique, we make distributional assumption on the data, and thus the estimates generated by bootstrapping is more robust, compared to traditional OLS estimates. Usually bootstrapping is better when our distributional assumptions are not met, or when our data set is too small, but note that bootstrapping usually generates larger standard errors.