Problem Set 2

Eric Karsten

February 3, 2020

1 Traditional Regression (10 points)

1.1 Problem Statement

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.

1.2 Solution

```r
# Loading Packages
library(tidyverse)
library(stargazer)
library(modelr)
library(rsample)

# Pretty table courtesy of
# Hlavac, Marek (2018).
# stargazer: Well-Formatted Regression and Summary Statistics Tables.
# R package version 5.2.2. https://CRAN.R-project.org/package=stargazer

# Loading Data
df <- read_csv("nes2008.csv")

# Specifying regression formula
f <- formula(biden ~ female + age + educ + dem + rep)


# Run simple regression
m1 <- lm(f, df)

# Plotting Regressions
stargazer(m1, omit.stat = c("f"),
          header = F,
          covariate.labels = c("Female", "Age", "Years of Education",
                               "Democrat", "Republican"),
          dep.var.caption = "Biden Feeling Thermometer",
          dep.var.labels.include = F,
          table.placement = "H",
          column.sep.width = "0pt",
          title = "Predicting Feelings Towards Biden")
```

Table 1: Predicting Feelings Towards Biden

	Biden Feeling Thermometer
Female	4.103***
	(0.948)
Age	0.048*
	(0.028)
Years of Education	−0.345*
	(0.195)
Democrat	15.424***
	(1.068)
Republican	−15.850***
	(1.311)
Constant	58.811***
	(3.124)
Observations	1,807
R^2	0.282
Adjusted R^2	0.280
Residual Std. Error	19.914 (df = 1801)
Note:	*p<0.1; **p<0.05; ***p<0.01

```r
mse(m1, df)
```

```
[1] 395.2702
```

We compute the mean squared error for the entire model to be 395.2701693. In general, we find coefficients that are significant on all of our parameters, women tend to like biden better, as do older people. Those who are more educated like him less, Democrats lke him more than independents and Republicans like him less then independents. This model is clearly missing many other dimensions that might predict a person's feeling towards biden. The Rˆ2 of the model is only 28% showing it is not explaining a lot of the variation in the data, and the mean squared error is about 400 which means the representative residual is about 20 points off. This is larger in magnitude than most of our coefficients, which is concerning if we wanted to use this for predictive reasons.

2 Simple Holdout Approach (30 Points)

2.1 Problem Statement

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.
- (5 points) Fit the linear regression model using only the training observations.
- (10 points) Calculate the MSE using only the test set observations.
- (10 points) How does this value compare to the training MSE from question 1? Present numeric comparison and discuss a bit.

2.2 Solution

```r
# function that does a split of the data and returns mse on the test
make_mse <- function(...) {
  split <- initial_split(df, prop = .5)
  train <- training(split)
  test <- testing(split)

  mod <- lm(f, train)
  return(mse(mod, test))
}

# Make one split and compute the MSE
set.seed(1984)
make_mse()
```

```
[1] 377.8091
```

This MSE (377.8091218) is smaller than the one from the prior fit of the model(395.2701693) where we fit it on all of the data. This is slightly unexpected because we might expect that fitting the model on less data and testing it on data it hadn't seen would lead to worse model performance, but we have to keep in mind that we are observing the outcome of a random process, so it is not surprising that for some seeds we will get smaller values and for others we will get larger values.

3 Repeated Simple Validation (30 points)

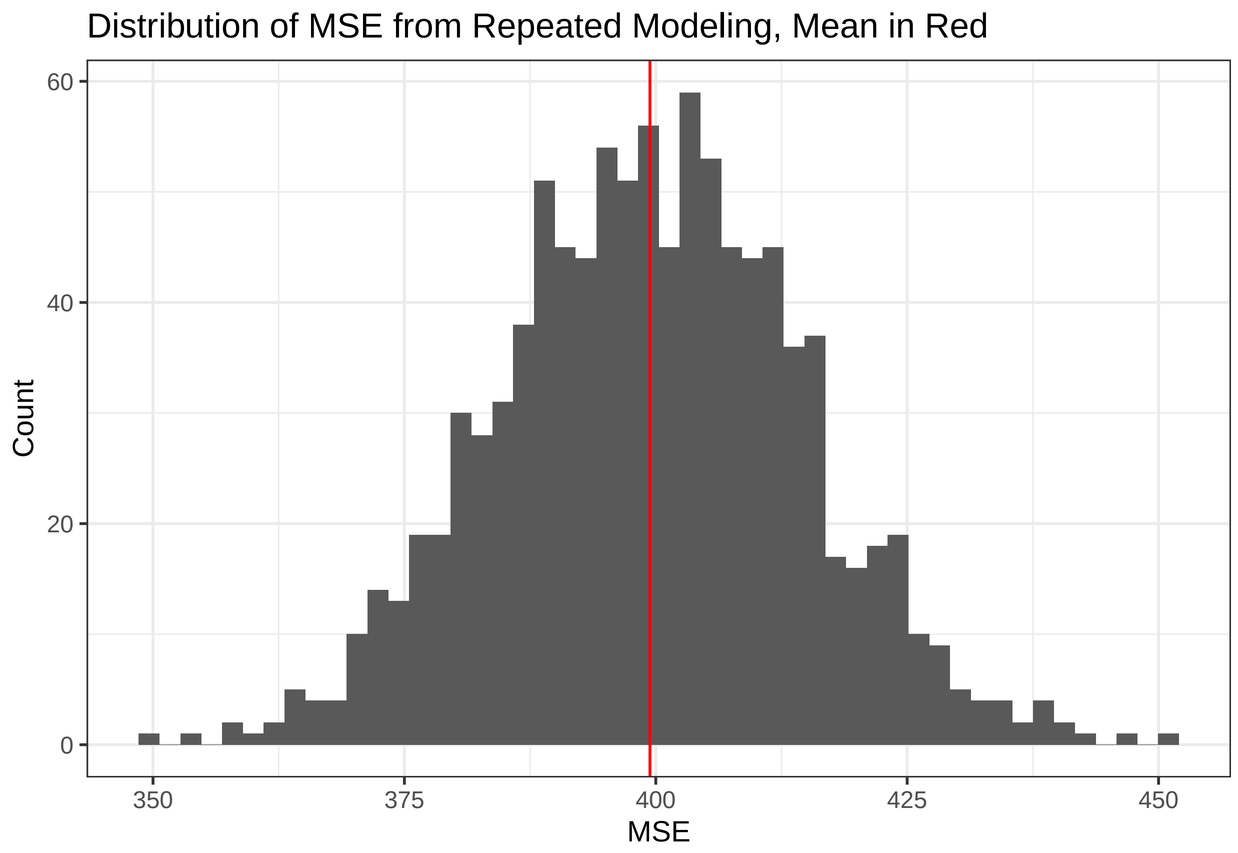
3.1 Problem Statement

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.

3.2 Solution

```r
set.seed(1776)
mse_tab <- tibble(
  MSE =  map(1:1000, make_mse) %>% unlist()
)

mse_tab %>%
  ggplot(aes(x = MSE)) +
  geom_histogram(bins = 50) +
  theme_bw() +
  geom_vline(xintercept = mean(mse_tab$MSE), color = "red") +
  labs(y = "Count",
       title = "Distribution of MSE from Repeated Modeling, Mean in Red")
```



We see from the above that as we take more and more samples of our MSE, we see a convergence in the distribution towards a normal with a mean of just under 400, the same as the MSE of our full dataset. We do see however that there is a fair bit of variation. We can compute the standard deviation of this distribution to be 'r sd(mse_tab$MSE), which shows how dramatic the differences in different fits of the model can be and highlights the importance of iteration in converging on optimal results when calibrating models. We wouldn't want to be misled by a realization in one of the tails of this distribution.

4 Discussion (30 Points)

4.1 Problem Statement

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.

4.2 Solution

```r
# function to return coefficients when passed a dataset
coeff <- function(dat_split) {
  mod <- lm(f, analysis(dat_split))
  tidy(mod) %>% select(term, estimate)
}

lm_df <-
  summary(m1)$coefficients %>%
  data.frame() %>%
  mutate(term = names(m1$coefficients)) %>%
  select(term, Lm_Estimate = Estimate, Lm_Sd = Std..Error)

boot_df <- df %>%
  bootstraps(1000) %>%
  { map_df(.$splits, coeff) } %>%
  group_by(term) %>%
  summarize(Boot_Estimate = mean(estimate),
            Boot_Sd = sd(estimate, na.rm = TRUE)) %>%
  left_join(lm_df)

knitr::kable(boot_df, digits = 3)
```

term	Boot_Estimate	Boot_Sd	Lm_Estimate	Lm_Sd
(Intercept)	58.747	3.101	58.811	3.124
age	0.048	0.030	0.048	0.028
dem	15.475	1.056	15.424	1.068
educ	-0.338	0.191	-0.345	0.195
female	4.061	0.959	4.103	0.948
rep	-15.855	1.329	-15.850	1.311

As we can see from the regression table above (which is not as nice as the first stargazer one, I know), the bootstrap produces similar results for many coefficients and standard deviations as the linear model. In fact, the coefficients produced by the two methods are statistically indistibguishable given the standard errors computed by either method. This is useful because in many settings, we don't have another method of computing standard errors on our estimates, so we can employ the bootstrap to get robust standard errors.