Gill_Sarah_ML_PS2

Sarah Gill

1/27/2020

```r
library(readr)
library(rcfss) #breaks tidy
```

```
##
## Attaching package: 'rcfss'

## The following objects are masked _by_ '.GlobalEnv':
##
##     mse, mse_vec
```

```r
library(boot)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.0.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.3.1
## v ggplot2 3.0.0     v forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.5.2

## Warning: package 'tidyr' was built under R version 3.5.2

## Warning: package 'purrr' was built under R version 3.5.2

## Warning: package 'dplyr' was built under R version 3.5.2

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(broom)
```

```
## Warning: package 'broom' was built under R version 3.5.2
```

```r
library(dplyr)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.5.2
```

```r
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 3.5.2

## For binary classification, the first factor level is assumed to be the event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.

##
## Attaching package: 'yardstick'

## The following object is masked from 'package:readr':
##
##     spec
```

```r
set.seed(1234)
```

1. 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.

```r
nes2008_df <- read_csv("nes2008.csv")
```

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

```r
regn_model <- glm(biden ~ female + age + educ + dem + rep, data = nes2008_df)

summary(regn_model)
```

```
##
## Call:
## glm(formula = biden ~ female + age + educ + dem + rep, data = nes2008_df)
##
## Deviance 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
##
## (Dispersion parameter for gaussian family taken to be 396.587)
##
##     Null deviance: 994144  on 1806  degrees of freedom
## Residual deviance: 714253  on 1801  degrees of freedom
## AIC: 15947
##
## Number of Fisher Scoring iterations: 2
```

```r
(mse <- augment(regn_model, newdata = nes2008_df) %>%
  mse(truth = biden, estimate = .fitted))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        395.
```

Present and discuss your results at a simple, high level.

mse = 395.27

This seems large, especially given the values that we are estimating (mean 62, variance 505). The mse, the average of the squared difference between estimates and the data points.

```r
max(nes2008_df$biden)
```

```
## [1] 100
```

```r
min(nes2008_df$biden)
```

```
## [1] 0
```

```r
var(nes2008_df$biden)
```

```
## [1] 550.4671
```

```r
mean(nes2008_df$biden)
```

```
## [1] 62.16381
```

2. Calculate the test MSE of the model using the simple holdout validation approach.

```r
#Split the sample set into a training set (50%) and a holdout set (50%). Be sure to set your seed prior
nes_split <- initial_split(data = nes2008_df,
                             prop = 0.5) #split the data in half
nes_train <- training(nes_split) #random subsample to train
nes_test <- testing(nes_split)
```

```r
#Fit the linear regression model using only the training observations.
nes_lm <- glm(biden~female+age+educ+dem+rep, data = nes_train) #fit model on training data

#(train_mse <- augment(nes_lm, newdata = nes_train) %>%
#  mse(truth = biden, estimate = .fitted))


#Calculate the MSE using only the test set observations.
(test_mse <- augment(nes_lm, newdata = nes_test) %>%
  mse(truth = biden, estimate = .fitted))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        389.
```

mse = 389.16 (recall mse from the simple general linear model was 395.27)

This is a slight improvement over the mse for the standard glm, however this is not expected and is likely because of the particular draw (see 3). Since we are modeling using only half of the data, then testing it on the other half we may expect a higher mse given that this estimate is generated from a smaller dataset, and unlike in 1 it is compared to different data than it was produced from.

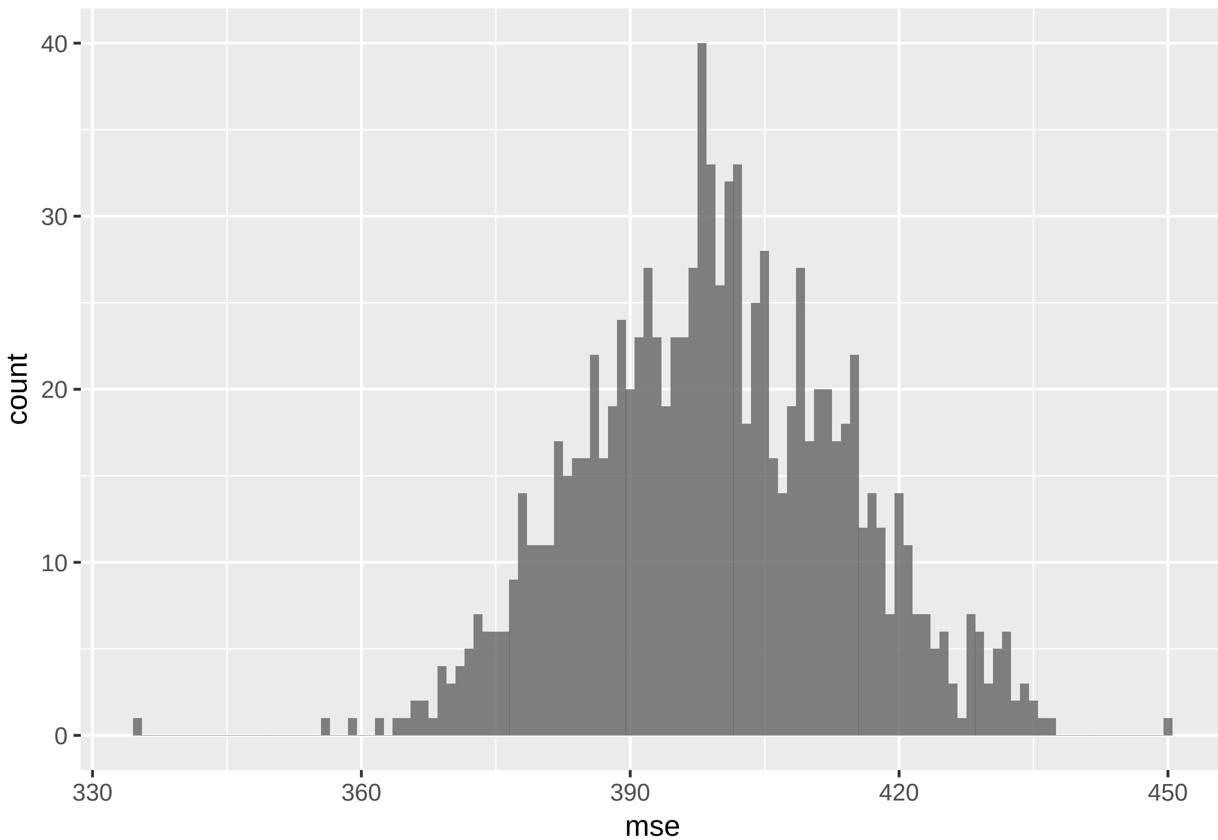
3. Repeat the simple validation set approach from the previous question 1000 times, using 1000 different splits of the observations into a training set and a test/validation set.

```r
x <- 1
mse_list <- c()

repeat{
  nes_split <- initial_split(data = nes2008_df,
                             prop = 0.5) #split the data in half
  nes_train <- training(nes_split) #random subsample to train
  nes_test <- testing(nes_split)

  #Fit the linear regression model using only the training observations.
  nes_lm <- glm(biden~female+age+educ+dem+rep, data = nes_train) #fit model on training   data
  mse <- augment(nes_lm, newdata = nes_test) %>%
    mse(truth = biden, estimate = .fitted)%>%
    select(.estimate)%>%
    as.numeric()
  mse_list <- append(mse_list,mse)
  x= x+1

  if (x == 1000){
    break
  }
}
 #source https://www.datamentor.io/r-programming/repeat-loop/
mean(mse_list)
```

```
## [1] 399.4956
```

Visualize your results as a sampling distribution (hint: think histogram or density plots). Comment on the results obtained.

```
data <- data.frame(mse = mse_list)

ggplot(data, aes(x = mse))+
  geom_histogram(binwidth = 1, alpha = 0.75)
```



When we iteratively split the data, run a regression on each split and extract the mse we can see that the mse estimates fall in a roughly normal distribution, centered on the mean 398.37.

Note that this mean is similar to the mse from the simple linear regression: 395.27

4. 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.

```
#broom::tidy()

regn_model <- lm(biden ~ female + age + educ + dem + rep, data = nes2008_df)
tidy(regn_model)
```

```
## # A tibble: 6 x 5
```

```
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  58.8       3.12       18.8  2.69e-72
## 2 female        4.10      0.948       4.33 1.59e- 5
## 3 age           0.0483    0.0282      1.71 8.77e- 2
## 4 educ         -0.345     0.195      -1.77 7.64e- 2
## 5 dem          15.4       1.07       14.4  8.14e-45
## 6 rep         -15.8       1.31      -12.1  2.16e-32
```

```r
# 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 <- nes2008_df %>%
  bootstraps(1000) %>%
  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))
```

```
## # A tibble: 6 x 3
##   term        .estimate    .se
##   <chr>           <dbl>  <dbl>
## 1 (Intercept)   58.8    3.14
## 2 age            0.0491 0.0284
## 3 dem           15.4    1.06
## 4 educ          -0.345  0.202
## 5 female         4.13   0.964
## 6 rep          -15.8    1.42
```

Bootstrap: Linear Regression: Estimates: Estimates:

(Intercept) 58.7844692 (Intercept) 58.81125899 age 0.0484685 age 0.04825892
dem 15.4786044 dem 15.42425563
educ -0.3448924 educ -0.34533479 female 4.0856644 female 4.10323009 rep -15.8100846 rep -15.84950614

The bootstrap and linear regression estimates are very similar, often only different at the tens or hundredths place.

Bootstrap standard errors are larger though (e.g. the SE on the coefficient estimate for age is 0.0282474 for the liner mode and 0.02897384 for the bootstrap). However, some SEs are actually larger in the linear model than the bootstrap (e.g. females is 0.9482286 in the linear model and 0.94451110 in the bootstrap). The larger standard errors in bootstrap makes sense because we are not assuming a distribution. This is useful if we are not prepared to make an assumption about the distribution of our population.