Intro to ML PS 2

Takahiro Minami

2/3/2020

```
df <- read.csv("nes2008.csv")
colnames(df)
```

```
## [1] "biden"  "female" "age"    "educ"   "dem"    "rep"
```

Question 1

I run the following regression.

$$biden_i = \beta_0 + \beta_1 age + \beta_2 female + \beta_3 educ + \beta_4 dem + \beta_5 rep + \epsilon_i$$

```
# run regression
lm_tra <- glm(biden ~ age+female+educ+dem+rep, data = df)
summary(lm_tra)
```

```
##
## Call:
## glm(formula = biden ~ age + female + educ + dem + rep, data = 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 ***
## age           0.04826    0.02825   1.708   0.0877 .
## female        4.10323    0.94823   4.327 1.59e-05 ***
## 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
```

Then, calculate MSE based on this result.

```
# predicted value
(full_mse <- augment(lm_tra, newdata = df) %>%
# MSE
mse(truth = biden, estimate = .fitted))
```

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

The MSE of the traditional linear regression is 395. This means that the average of the square of the deviation between the true biden feeling thermometer and the predicted biden feeling thermometer is 395. It's difficult to judge the model without other models I can compare to, but I think this MSE is relatively big given that the interval of biden feeling thermometer is zero to 100. It implies that this linear model has relatively large bias. In addition, I have to mention that I cannot evaluate the variance of the model. Because I used the whole sample as training set, I'm not sure how accurate this model is when I fit this model to a new data.

Question 2

step 1

```
# split the data into training and validation sets
set.seed(100)
df_split <- initial_split(data=df, prop = 0.5)
```

step 2

```
# fit the model to training set
lm_train <- glm(biden ~ age+female+educ+dem+rep,
                data = training(df_split))
summary(lm_train)
```

```
##
## Call:
## glm(formula = biden ~ age + female + educ + dem + rep, data = training(df_split))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -73.646  -11.255    1.134   12.960   52.995
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  57.24222    4.43081  12.919  < 2e-16 ***
## age           0.10089    0.04064   2.483  0.01322 *
## female        4.13074    1.33947   3.084  0.00211 **
## educ         -0.50178    0.27656  -1.814  0.06995 .
## dem          15.46403    1.51221  10.226  < 2e-16 ***
## rep         -14.26545    1.81898  -7.843 1.25e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 390.198)
##
##     Null deviance: 486087  on 903  degrees of freedom
## Residual deviance: 350398  on 898  degrees of freedom
## AIC: 7967.3
##
## Number of Fisher Scoring iterations: 2
```

step 3

```
# predicted value using test set
(holdout_mse <- augment(lm_tra, newdata = testing(df_split)) %>%
# MSE
mse(truth = biden, estimate = .fitted))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        401.
```

step 4

The MSE for the test set is 401, and I find that it is almost the same as the training MSE (395) in question 1. This implies that the estimated linear model has small variance, meaning the model can be fit to the new data as nicely as to the training set used for building the model. In other words, it suggests that I don't need to worry about over-fitting.

Question 3

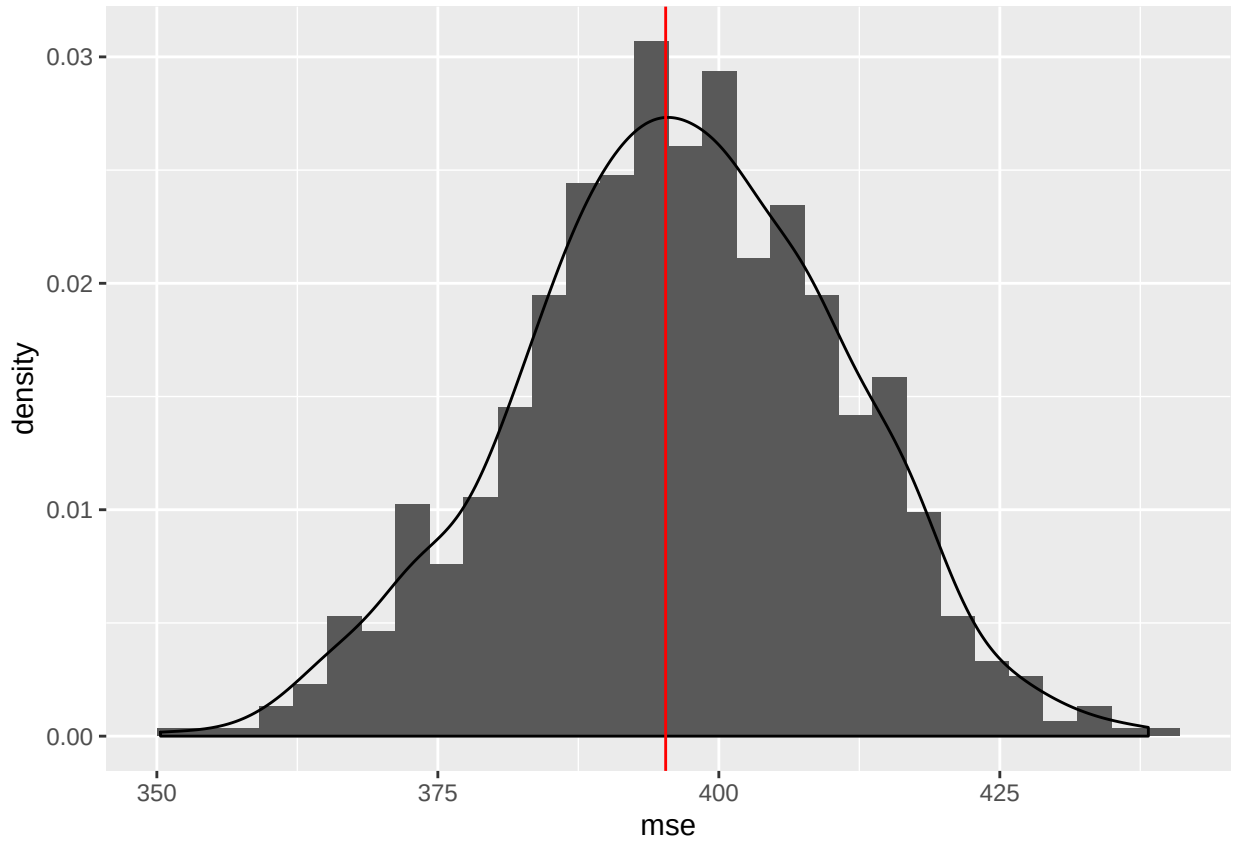
```
n_rep <- 1000 # number of replication
set.seed(200)
mse_rep<- c()

# repeat the process 1000 times
for (i in 1:n_rep) {
  # split the data into two
  df_split_rep <- initial_split(data=df, prop = 0.5)

  # fit the model to training set
  lm_train_rep <- glm(biden ~ age+female+educ+dem+rep,
                  data = training(df_split_rep))
  # calculate MSE for test set
  full_mse_rep <-
    augment(lm_tra, newdata = testing(df_split_rep)) %>%
    mse(truth = biden, estimate = .fitted)
  mse_rep[i] <- full_mse_rep$.estimate[1]
}
```

```
# draw histogram
mse_rep <- data.frame(mse = mse_rep)

ggplot(data=mse_rep, aes(x=mse))+
  geom_histogram(aes(y = ..density..))+
  geom_density()+
  geom_vline(aes(xintercept = full_mse$.estimate[1]),color="red")
```



The histogram shows the distribution of 1000 MSEs generated by the simulation. The red vertical line is the training MSE from question 1. I can find that the training MSE is placed at almost the center of the distribution. It shows that the concern of over-fitting is not big in this estimation. Plus, I can also find that the distribution of MSE places in relatively small interval. It implies that the variance of the model is relatively small.

Question 4

```
# define function
func_coefs <- function(splits, ...) {
  mod <- glm(..., data = analysis(splits))
  tidy(mod)
}

# bootstrap
set.seed(3000)
df_boot <- df %>%
  bootstraps(1000) %>%    # B=1000
  mutate(coef = map(splits, func_coefs,
                    as.formula(biden ~ age+female+educ+dem+rep)))
# get estimated parameter and s.e.
coef_boot <-
  df_boot %>%
  unnest(coef) %>%
  group_by(term) %>%
```

```
  summarize(.estimate = mean(estimate),
            .se = sd(estimate, na.rm = TRUE))
```

```
# merge the result of question 1 to bootstrap result
coef_full <- data.frame(coef(summary(lm_tra)))[,1:2]
coef_full$term <- row.names(coef_full)
comp_table <- left_join(coef_boot,coef_full, by = "term")
colnames(comp_table) <- c("term",
                          "Est_bootstrap",
                          "s.e._bootstrap",
                          "Est_original",
                          "s.e._original")
comp_table
```

```
## # A tibble: 6 x 5
##   term        Est_bootstrap s.e._bootstrap Est_original s.e._original
##   <chr>               <dbl>          <dbl>        <dbl>         <dbl>
## 1 (Intercept)       58.9            3.18        58.8           3.12
## 2 age                0.0464         0.0290       0.0483        0.0282
## 3 dem               15.4            1.04        15.4           1.07
## 4 educ              -0.347          0.199       -0.345         0.195
## 5 female             4.11           0.969        4.10          0.948
## 6 rep              -15.9            1.41       -15.8           1.31
```

The numeric outputs for bootstrap and original estimation are shown in the table. Comparing estimated parameters in both ways, there are no substantial differences. On the other hand, bootstrap has similar but slightly larger standard errors for intercept and coefficients on predictors (except for the one on *dem*). This is because bootstrap doesn't rely on the distributional assumptions including homoskedasticity of error terms while the original regression depends on them. In general, bootstrapping is very useful tool when we are not sure whether the distributional assumptions correctly stands or not because bootstrap is robust to these assumptions but the traditional regression may give biased standard errors if these assumptions are violated. The more severely these assumptions are violated, the larger divergence (impact) of bootstrap from traditional regression become. In this example, the deviation of standard errors between bootstrap and traditional regression are not large. This implies that the distribution assumptions in the traditional regression are relatively true.