Homework 3

Hengle Li

2/8/2020

```r
library(tidyverse)
library(rsample)
library(rcfss)
library(leaps)
library(yardstick)
library(caret)
library(tidymodels)
library(glmnet)
```

Training/test error for subset selection

```r
#Assuming the features of X are unknown, each X, is just a vector
#of length 20, containing 20 random numbers
X_generate <- function(x){
  set.seed(x)
  sample(0:100, 20, replace = TRUE)
}

X_list <- map(1:1000, X_generate)
```

```r
#beta is a known vector of length 20 with a number of 0s.
set.seed(1)
beta <- sample(1:20, 20, replace = TRUE)
#examining the numbers within beta
beta
```

```
##  [1]  4  7  1  2 11 14 18 19  1 10 14 10  7  9 15  5  9 14  5  5
```

```r
#assume there are 3 0s in beta, choose three random places
set.seed(2)
sample(1:20, 3, replace = FALSE)
```

```
## [1] 15  6 19
```

```r
#the 6th, 15th, and 19th will be 0
beta[c(6, 15, 19)] <- 0
#examine beta again
beta
```

```
##  [1]  4  7  1  2 11  0 18 19  1 10 14 10  7  9  0  5  9 14  0  5
```

```
#since there's no requirement for epsilon, just use a random vector
#of length 1000, one number for each Y
set.seed(66)
epsilon <- sample(0:100, 1000, replace = TRUE)
```

By the definition of Y, it's a vector of 20. Therefore, the entire dataset of Y's is what we are looking for.

```
#use a for loop to create a list of Y
out <- vector(mode = "list",
                length = 1000)
  for(i in seq_along(out)){
    out[[i]] <- sum(X_list[[i]] * beta + epsilon[[i]])}
```

```
#transform X_list to a proper dataset
df <- as.data.frame(X_list)
colnames(df) <- c(seq(1:1000))
df <- as.data.frame(t(df))
df <- cbind(df, Y = as.integer(out))
```

```
#split the dataset
df_split <- initial_split(df, 0.1)
df_train <- training(df_split)
df_test <- testing(df_split)
```

```
#create the model
#set size of subset to 20, because there are only 20 predictors
best_subset <- regsubsets(Y ~ .,
                          data = df_train,
                          nvmax = 20
                          )
```

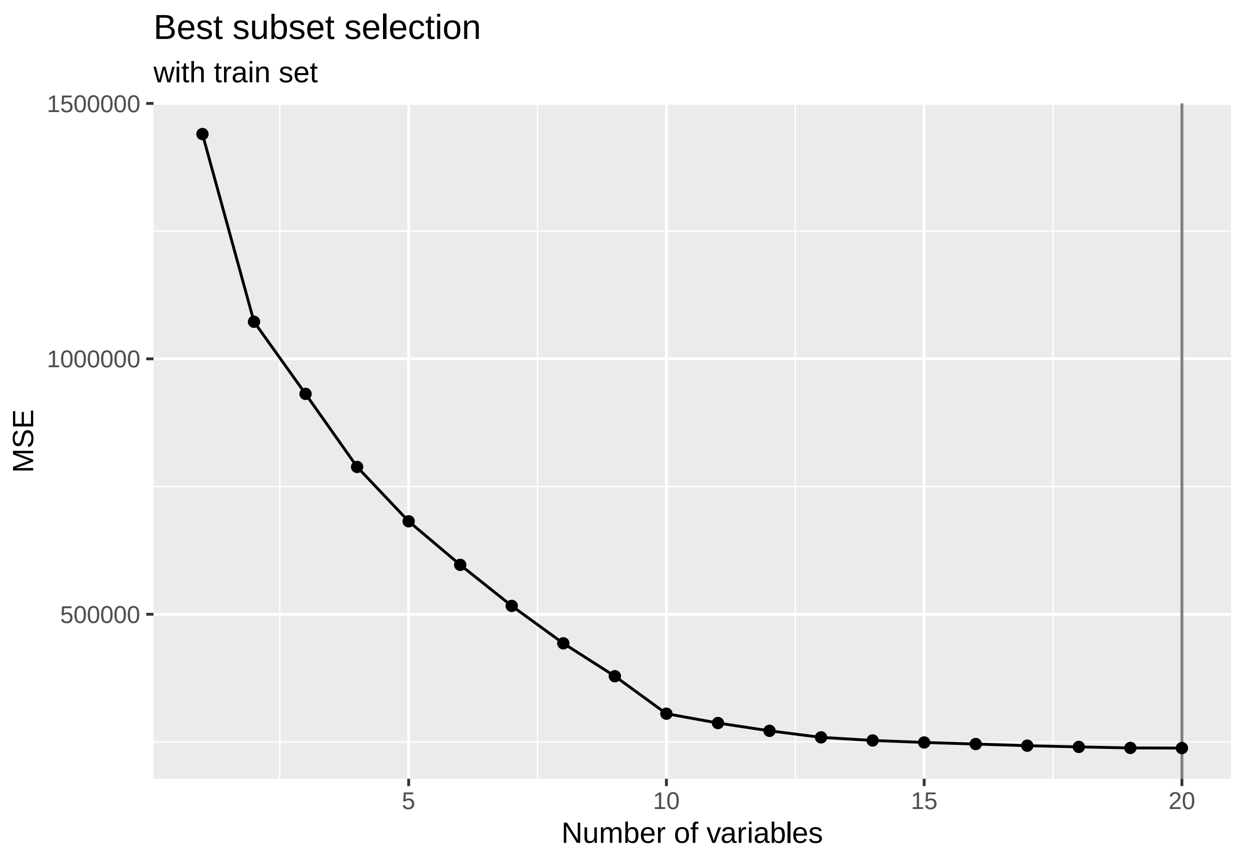
reasonings

- best_subset contains a list of models
- coefficients of models in best_subset can be extracted by `coef()`
- values of X can be extracted from the dataset by `model.matrix()` with the training set
- for each possible number of predictors, there are 100 results, because there are 100 observations in the training set

```
#construct functions for prediction
predict_best <- function(model, newdata, id){
  formulae <- as.formula(Y ~ .) #set formula for model.matrix()
  matrix_X <- model.matrix(formulae, newdata) #set model.matrix()
  variables <- names(coef(model, id = id)) #decide which variables to use
  #multiply X with coefficients to produce predictions
  as.vector(matrix_X[, variables] %*% coef(model, id = id), mode = "integer")
}
```

```r
#there will be 100 results for each possible n
result_train <- tibble(n = 1:20, pred = map(1:20, ~ predict_best(best_subset, df_train, .x))) %>%
  unnest(pred) %>%
  mutate(truth = rep(df_train$Y, 20))
#repeating the sequence of Y to match it with the results

#calculate MSE
train_mse <- result_train %>%
  group_by(n) %>%
  mse(truth = truth, estimate = pred)
```

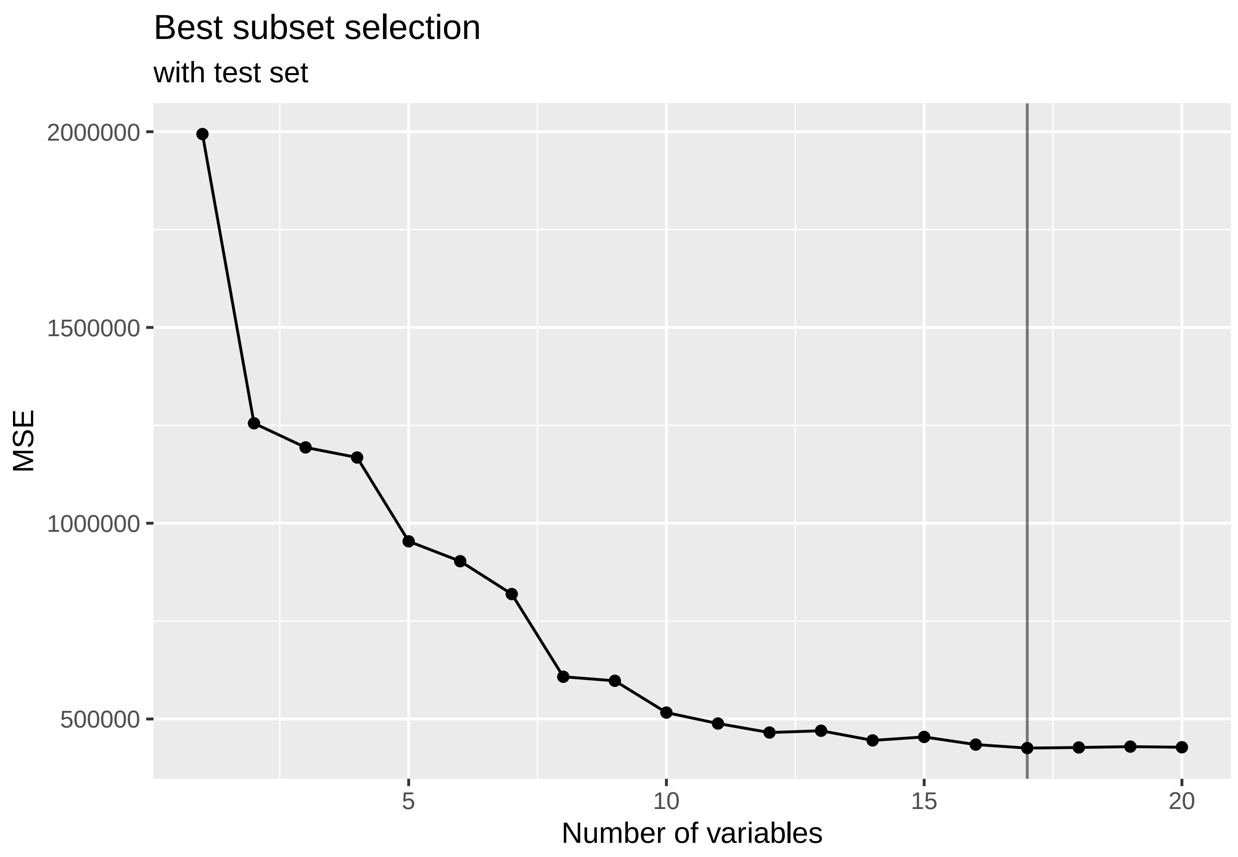
```r
train_mse %>%
  ggplot(aes(x = n, y = .estimate)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = which.min(train_mse$.estimate), alpha = .5) +
  labs(title = "Best subset selection",
       subtitle = "with train set",
       x = "Number of variables",
       y = "MSE")
```



It is not surprising that when n = 20, MSE reaches minimum. It's a feature of machine learning in dealing with the training set: more predictors, higher accuracy.

```r
#there will be 900 results for each possible n
result_test <- tibble(n = 1:20, pred = map(1:20, ~ predict_best(best_subset, df_test, .x))) %>%
  unnest(pred) %>%
  mutate(truth = rep(df_test$Y, 20))
#again, repeating the sequence of Y to match it with the results

#calculate MSE
test_mse <- result_test %>%
  group_by(n) %>%
  mse(truth = truth, estimate = pred)
```

```r
test_mse %>%
  ggplot(aes(x = n, y = .estimate)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = which.min(test_mse$.estimate), alpha = .5) +
  labs(title = "Best subset selection",
       subtitle = "with test set",
       x = "Number of variables",
       y = "MSE")
```



The graph indicates when 17 variables are included, MSE reaches minimum for the test set. It's noteworthy that 20 is no longer the best subset size here. On one hand, it's natural that when a feature is added, the MSE decreases due to the increased explanation power. On the other hand, one can see that for n equals 16 or above, the differences in MSE are quite small compared with those for n smaller than 16, i.e. the curve is flattened. Moreover, since there are 3 0s in beta, it's intuitively understandable that MSE reaches minimum at n=17: 20-3=17.

```r
#coefficients of the model when n is 17
coef(best_subset, 17)
```

```
## (Intercept)          V1          V2          V4          V5          V7
##  842.156469    5.892131    2.561686    2.036580   12.194403   17.724406
##          V8          V9         V10         V11         V12         V13
##   18.909786    4.244702    9.429593   11.411053   13.453363    9.668298
##         V14         V16         V17         V18         V19         V20
##    9.870179    5.382199    8.947430   11.574887    2.583881    2.329960
```

```r
#true value of the coefficients
beta
```

```
##  [1]  4  7  1  2 11  0 18 19  1 10 14 10  7  9  0  5  9 14  0  5
```

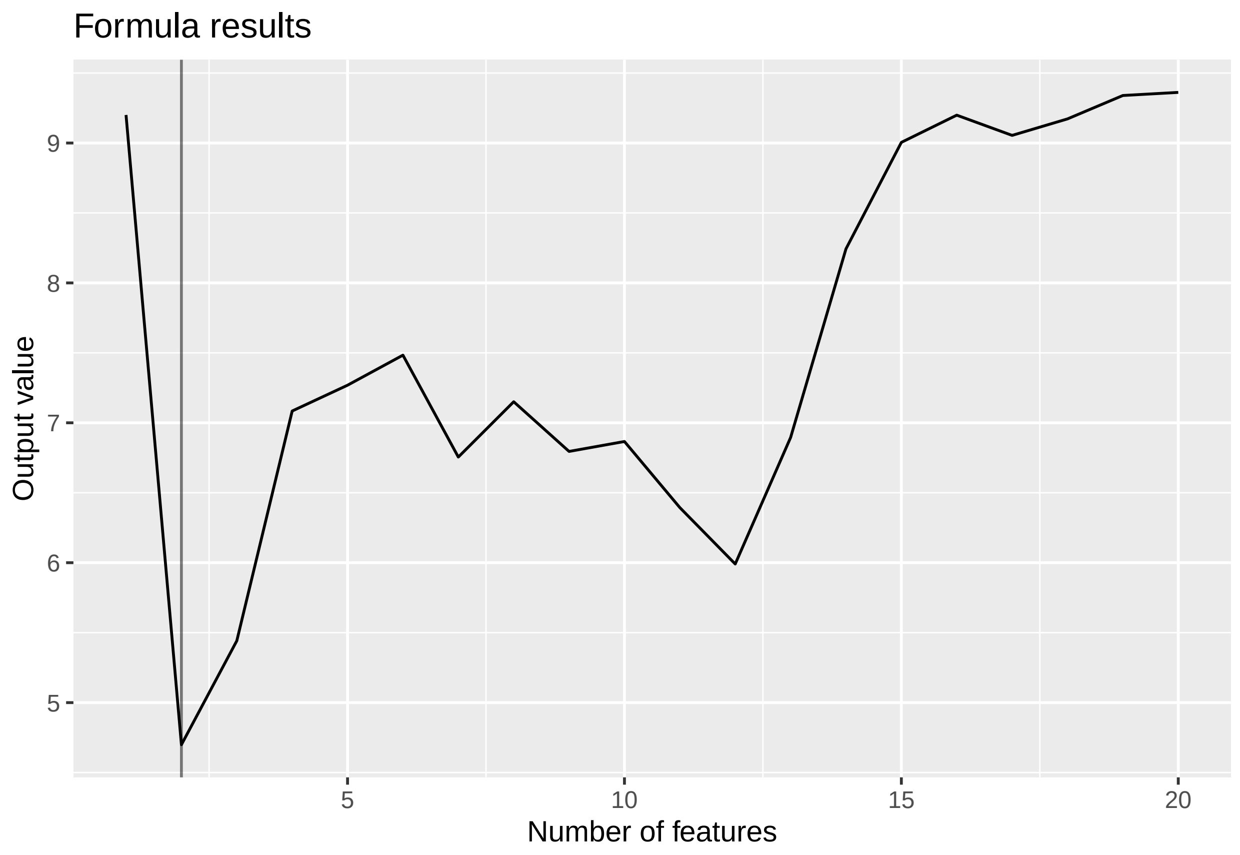
```r
#the mean value of epsilon
mean(epsilon)
```

```
## [1] 49.929
```

Comparing the coefficients of the model and the values in beta, we can see that the model leaves out V3, V6, V15, while the true null coefficients are V6, V15, V19, i.e. it misses only one coefficient. On the other hand, some coefficients are very close to the actual value of beta, including V4, V5, V7, V8, V10 etc. It should also be noted that the value of the intercept is very different from the mean value of epsilon, which shows the difference between high-dimensional space and the general understanding of 2d space.

```r
#transform beta into a dataset for processing
beta_frame <- as.data.frame(t(beta))

#function to calculate through the formula
fancy <- function(x){
  set <- as.data.frame(t(coef(best_subset, x)))
  set <- set[-c(1)]
  process <- beta_frame %>%
    dplyr::select(colnames(set))
  sqrt(sum((process[1,] - set[1,])^2))
}

#calculate the numbers
output <- map(1:20, fancy)
results_form <- tibble(n = 1:20, output = as.numeric(output))
```

```r
results_form %>%
  ggplot(aes(x = n, y = output)) +
  geom_line() +
  geom_vline(xintercept = which.min(results_form$output), alpha = .5) +
  labs(title = "Formula results",
       x = "Number of features",
       y = "Output value")
```



The graph shows when n = 2, the stat reaches its minimum. If we judge by this graph, the best subset selection should be when n is 2. However, when n is 2, there are only 2 features included in the model. By the formula, it's bound to be smaller than any other n: when n is 1, the model coefficient will be much different from its counterpart in beta for greater accuracy; when n is greater than 1, the more coefficients are included, the greater the stat will be in general. The intuitive of this stats is contradictory to that of best subset selection. Therefore, the stat is not a reliable way to tell which model best fits the datasets. Its accuracy is far worse than calculating MSE for the test set. If calculating test set MSE shows 17 is the best subset size, this stat shows 17 is a very poor size in performance.

Application

```
gss_test <- read_csv("data/gss_test.csv")
gss_train <- read_csv("data/gss_train.csv")
```

```
X <- gss_train %>%
  select(-egalit_scale)
Y <- gss_train$egalit_scale

X_cv <- model.matrix(egalit_scale ~ ., data = gss_train)[, -1]

X_test <- model.matrix(egalit_scale ~ ., data = gss_test)[, -1]
```

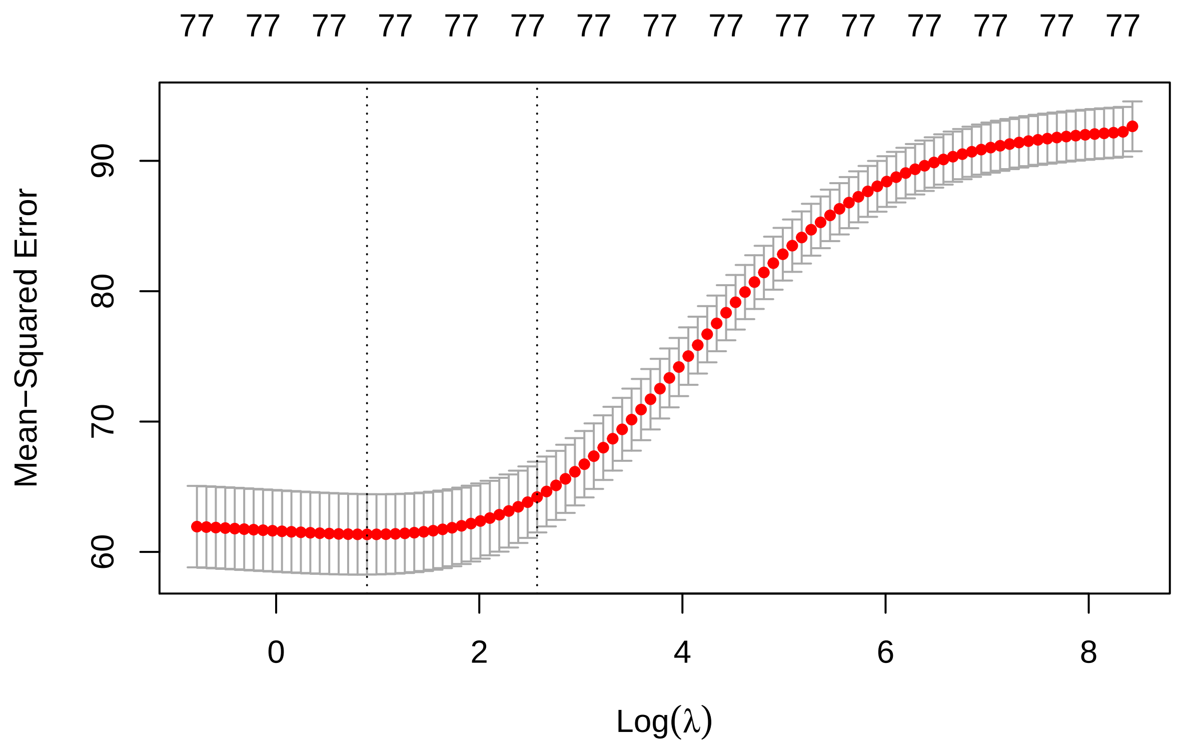
least squars linear

```r
#train function
lm_gss <- train(
  x = X,
  y = Y,
  method = "lm"
)
#predict
lm_result <- tibble(truth = gss_test$egalit_scale, pred = predict(lm_gss, newdata = gss_test))

#calculate MSE
lm_result %>%
  mse(truth = truth, estimate = pred)
```

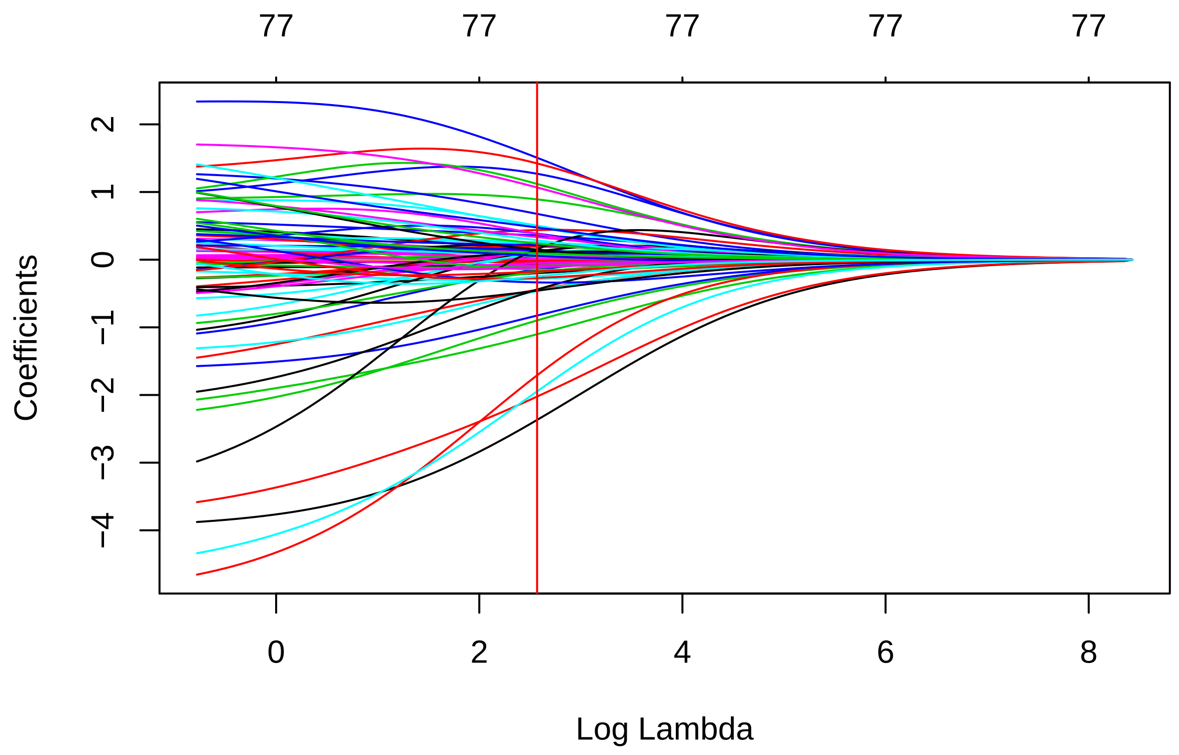
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        63.2
```

Ridge regression

```r
#cv.glmnet does 10-fold CV
ridge_10fold <- cv.glmnet(
  x = X_cv,
  y = Y,
  alpha = 0
)

plot(ridge_10fold)
```



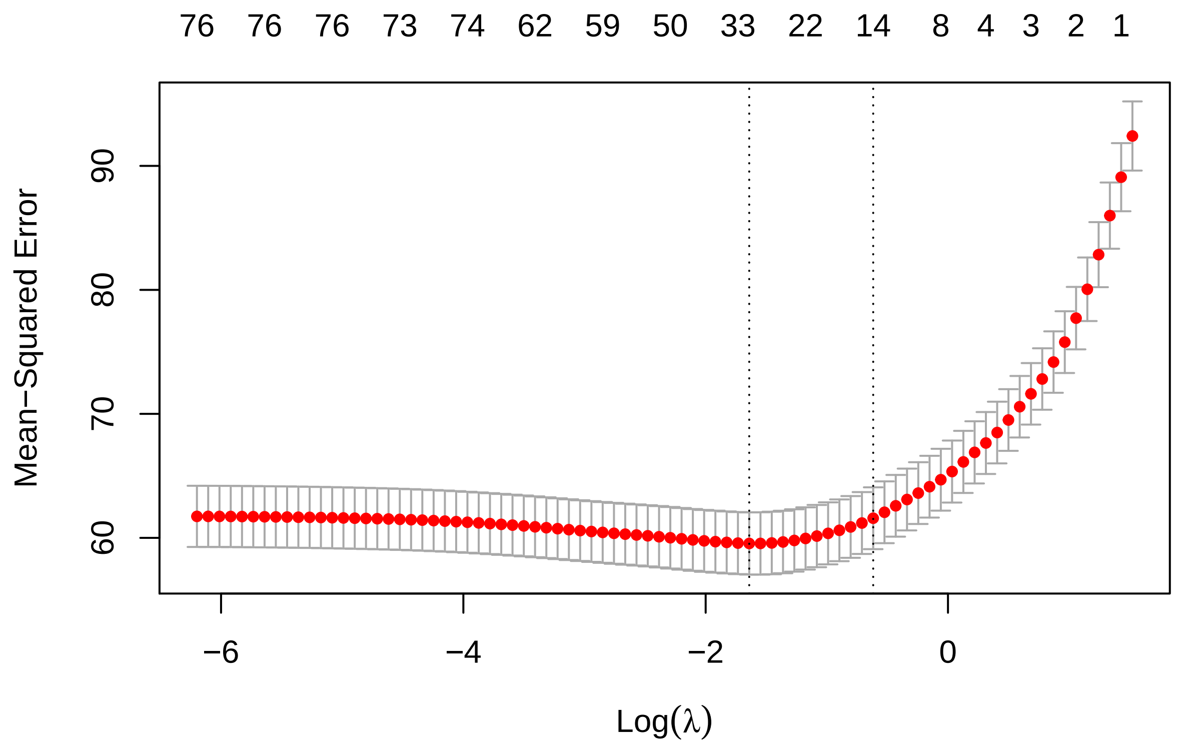
```r
#use the lambda from the above 10-fold CV
ridge_tuned <- glmnet(
  x = X_cv,
  y = Y,
  alpha = 0
)

#speculate the value of lambda
plot(ridge_tuned, xvar = "lambda")
abline(v = log(ridge_10fold$lambda.1se), col = "red")
```



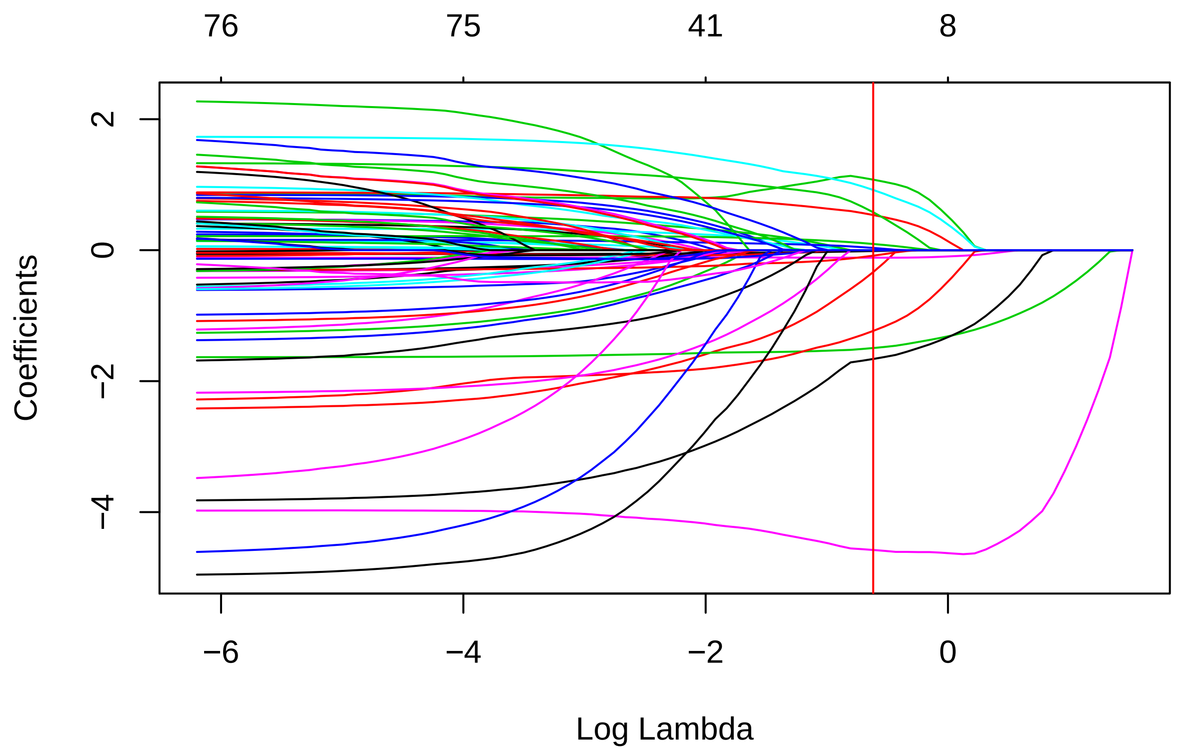
```r
#make predictions
ridge_result <- tibble(truth = gss_test$egalit_scale,
                       pred = predict(ridge_tuned,
                                      s = ridge_10fold$lambda.1se,
                                      newx = X_test)) %>%
  mutate(pred = as.numeric(pred))

#calculate MSE
ridge_result %>%
  mse(truth = truth, estimate = pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        62.6
```

```r
#cv.glmnet does 10-fold CV
lasso_10fold <- cv.glmnet(
  x = X_cv,
  y = Y,
  alpha = 1
)

plot(lasso_10fold)
```



```r
#use the lambda from the above 10-fold CV
lasso_tuned <- glmnet(
  x = X_cv,
  y = Y,
  alpha = 1
)

#speculate the value of lambda
plot(lasso_tuned, xvar = "lambda")
abline(v = log(lasso_10fold$lambda.1se), col = "red")
```



```
#make predictions
lasso_result <- tibble(truth = gss_test$egalit_scale,
                       pred = predict(ridge_tuned,
                                      s = lasso_10fold$lambda.1se,
                                      newx = X_test)) %>%
  mutate(pred = as.numeric(pred))

#calculate MSE
lasso_result %>%
  mse(truth = truth, estimate = pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        62.3
```

```
#randomize which fold among the 10 to use for training
folds <- sample(1:10, size = length(Y), replace = TRUE)

#create a grid for entries of lambda and MSE
#using lambda.1se here as in the previous two models
tuning <- tibble(alpha = seq(0, 1, by = .1),
                 mse_1se = NA,
                 lambda_1se = NA
                 )
```

```r
#fill in the grid with values
#for each alpha, there is a corresponding lambda
for(i in seq_along(tuning$alpha)){
  filling <- cv.glmnet(x = X_cv,
                  y = Y,
                  alpha = tuning$alpha[i],
                  foldid = folds)
  tuning$mse_1se[i] <- filling$cvm[filling$lambda == filling$lambda.1se]
  tuning$lambda_1se[i] <- filling$lambda.1se
}

#create another grid for the calculated lambda and possible alpha
testing <- expand.grid(alpha = seq(0, 1, by = 0.1),
                       lambda_1se = tuning$lambda_1se)
```

```r
#a grid for all models and their testing MSE
models_enet <- tibble(alpha = testing$alpha,
                      lambda_1se = testing$lambda_1se,
                      MSE = NA)

#use a function to make predictions
very_fancy <- function(al, bi){
  #reproduce the model with corresponding alpha
  dumb <- glmnet(
    x = X_cv,
    y = Y,
    alpha = al
    )
  mid_filler <- tibble(truth = gss_test$egalit_scale,
                  pred = predict(dumb,
                                 s = bi,
                                 newx = X_test)) %>%
    mutate(truth = as.numeric(truth)) %>%
    mutate(pred = as.numeric(pred))
  #extract the MSE from the resulting form
  MSE_filler <- mid_filler %>%
    mse(truth = truth,
        estimate = pred)
  MSE_filler$.estimate
}
```

```r
for(i in seq_along(models_enet$alpha)) {
  models_enet$MSE[i] <- very_fancy(models_enet$alpha[i],
                                   models_enet$lambda_1se[i])
}

models_enet
```

```
## # A tibble: 121 x 3
##    alpha lambda_1se   MSE
##    <dbl>      <dbl> <dbl>
##  1   0         11.9  62.2
## 2   0.1       11.9  70.3
## 3   0.2       11.9  77.8
## 4   0.3       11.9  85.0
## 5   0.4       11.9  90.5
## 6   0.5       11.9  90.5
## 7   0.6       11.9  90.5
## 8   0.7       11.9  90.5
## 9   0.8       11.9  90.5
## 10  0.9       11.9  90.5
## # ... with 111 more rows
```

```
#find the combination where MSE is smallest
models_enet[which.min(models_enet$MSE),]
```

```
## # A tibble: 1 x 3
##   alpha lambda_1se   MSE
##   <dbl>      <dbl> <dbl>
## 1     0       4.48  60.7
```

```
best_enet <- glmnet(
  x = X_cv,
    y = Y,
    alpha = 0
)
coef(best_enet, s = 4.47891)
```

```
## 78 x 1 sparse Matrix of class "dgCMatrix"
##                                         1
## (Intercept)                  27.812527177
## age                          -0.029085324
## attend                       -0.031341874
## authoritarianism              0.003866456
## black                         1.364676936
## born                          0.289876900
## childs                        0.194967153
## colath                        0.266453153
## colrac                        0.156851491
## colcom                        0.043574755
## colmil                       -0.438590430
## colhomo                       0.750249170
## colmslm                       0.104384205
## con_govt                     -0.065899396
## evangelical                  -0.133158895
## grass                        -1.405777809
## happy                         0.426998495
## hispanic_2                    0.307646909
## homosex                       0.100523730
## income06                     -0.096928562
## mode                          0.103578549
## owngun                        0.972169886
## polviews                     -1.198183408
## pornlaw2                     -0.192456599
## pray                          0.069395900
## pres08                          -3.182909149
## reborn_r                         0.020441434
## science_quiz                    -0.115688048
## sex                              0.956155286
## sibs                             0.126992674
## social_connect                   0.020416834
## south                           -0.295715271
## teensex                         -0.045812010
## tolerance                       -0.146522863
## tvhours                          0.180091491
## vetyears                        -0.295169695
## wordsum                         -0.060414625
## degree_HS                        0.185338612
## degree_Junior.Coll              -0.759780199
## degree_Bachelor.deg             -1.478496223
## degree_Graduate.deg              0.158210060
## marital_Widowed                 -0.827780054
## marital_Divorced                -0.073622325
## marital_Separated               -0.241458720
## marital_Never.married            0.302591881
## news_FEW.TIMES.A.WEEK            0.155015226
## news_ONCE.A.WEEK                 0.250035811
## news_LESS.THAN.ONCE.WK           0.113685398
## news_NEVER                       0.523135870
## partyid_3_Ind                   -1.014429245
## partyid_3_Rep                   -2.683959252
## relig_CATHOLIC                  -0.418442609
## relig_JEWISH                     0.502864793
## relig_NONE                      -0.192799020
## relig_OTHER                      0.654155907
## relig_BUDDHISM                  -0.018351662
## relig_HINDUISM                  -3.010624953
## relig_OTHER.EASTERN              1.419629233
## relig_MOSLEM.ISLAM               2.043560515
## relig_ORTHODOX.CHRISTIAN        -3.040467945
## relig_CHRISTIAN                 -0.147421158
## relig_NATIVE.AMERICAN           -0.850438470
## relig_INTER.NONDENOMINATIONAL  1.640825181
## social_cons3_Mod                -0.018429052
## social_cons3_Conserv            -0.219388032
## spend3_Mod                       0.498563410
## spend3_Liberal                   1.426027733
## zodiac_TAURUS                    0.369941216
## zodiac_GEMINI                   -0.228641288
## zodiac_CANCER                    0.430359483
## zodiac_LEO                       0.146563036
## zodiac_VIRGO                     0.787060925
## zodiac_LIBRA                    -0.138296162
## zodiac_SCORPIO                  -0.614916305
## zodiac_SAGITTARIUS              -0.258068072
## zodiac_CAPRICORN                 0.175495915
## zodiac_AQUARIUS                  0.683237980
## zodiac_PISCES                   -0.354799065
```

- Judging by the results, if alpha can be 0, the combination when MSE reaches its minimum is alpha = 0, lambda_1se = 4.47891 (If alpha cannot be 0, then the smallest-MSE combination will be alpha = 0.1, lambda = 1.2288997.)
- When alpha = 0, lambda_1se = 4.47891, the MSE is 60.73831. As the function shows, none of the coefficient estimates are 0, and therefore there are 78 non-zero coefficient estimates.

discussion

- Looking at the resulting MSEs, it appears that elastic net approach can produce the most accurate regression model, next is ridge, then lasso, and last least squares.
- All of the MSEs are greater than 60. By definition of MSE, the average difference between estimated attitudes and actual attitudes is at least 7.79.
- Moreover, the differences among the final MSEs from the 4 approaches are not very large, unless we consider the disgarded combinations of alpha and lambda in the elastic net approach. Therefore, in terms of accuracy, the approaches are not very different, though elastic net is much better than least square, it's not substantially better than lasso or ridge.
- In terms of overall accuracy in predicting egalitarianism, none of these four approaches is very accurate. Depending on the definition of the egalitarian scale, one could be predicted to be low-egalitarian while he's in fact high-egalitarian, due to errors in the predictions. (That is, 18 is a boundary to decide whether a person is high or low egalitarian, and the respondent happens to score close to this boundary.)