Problem Set 3

Akira Masuda

2020/2/9

```
Course: MACS30100 Perspectives on Computational Modeling (Winter 2020)
Author: Akira Masuda (ID: alakira)
```

```r
library(knitr)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------------

## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3

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

```r
library(leaps)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 3.0-2
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(DT)
knitr::opts_chunk$set(echo=TRUE)
# options(width=1000)
rm(list=ls())
```

Conceptual Exercises

Training/test error for subset selection

1. Generating data

```
# Set seed
set.seed(110)
# Generate variables
x_raw = runif(20000, -1, 1)
x = matrix(x_raw, nrow=1000, ncol=20)
# Generate beta. Set 18th, 19th, and 20th beta to zero
beta = runif(20, 0, 10)
beta[18:20] = 0
# Error term
ep = rnorm(1000, 0, 0.25)
# Response vector
y = x %*% beta + ep
# Put them into dataframe
df = data.frame(x, y)
```

2. Split data

```
train = df[0:900,]
test = df[901:1000,]
```

3. Subset selection

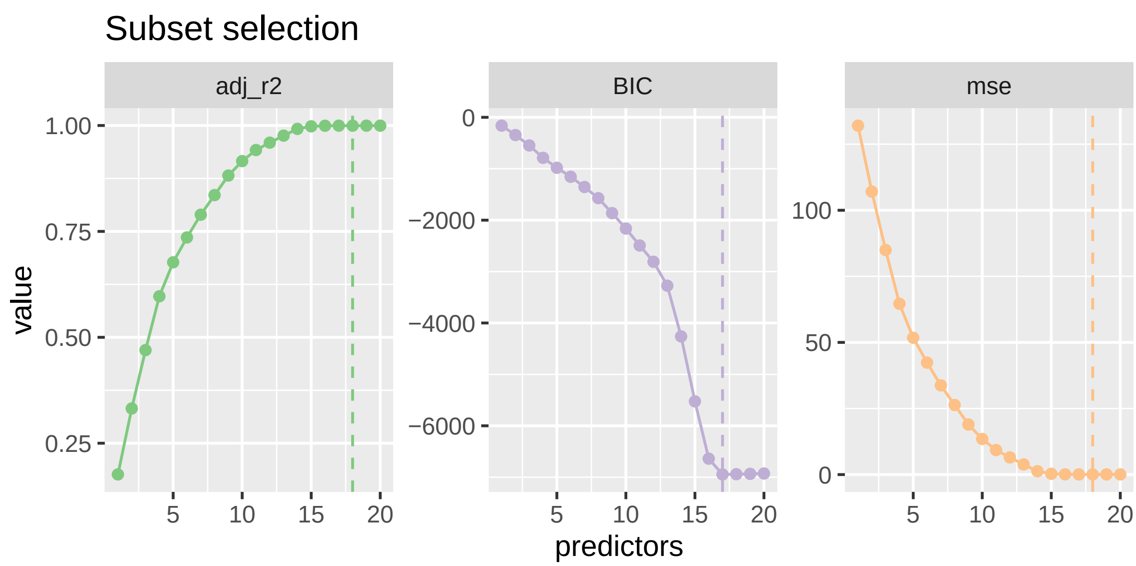
```
best_subset <- regsubsets(y ~ ., # regsubsets from "leaps" library
                          data = train,
                          nvmax = 20 # maximum size of subsets to examine
                          );
results <- summary(best_subset)
results$outmat # A logical matrix indicating which elements are in each model
```

```
##            X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15 X16 X17
## 1  ( 1 )  " " " " " " "*" " " " " " " " " " " " " " " " " " " " " " " " " " "
## 2  ( 1 )  " " " " "*" "*" " " " " " " " " " " " " " " " " " " " " " " " " " "
## 3  ( 1 )  " " " " "*" "*" " " " " " " " " " " " " " " " " " " " " "*" " " " "
## 4  ( 1 )  " " " " "*" "*" " " " " " " " " " " " " " " " " " " "*" "*" " " " "
## 5  ( 1 )  " " " " "*" "*" " " " " "*" " " " " " " " " " " " " "*" "*" " " " "
## 6  ( 1 )  " " " " "*" "*" " " " " "*" " " " " " " " " " " " " "*" "*" " " "*"
## 7  ( 1 )  " " "*" "*" "*" " " " " "*" " " " " " " " " " " " " "*" "*" " " "*"
## 8  ( 1 )  " " "*" "*" "*" " " " " "*" " " "*" " " " " " " " " "*" "*" " " "*"
## 9  ( 1 )  " " "*" "*" "*" " " "*" "*" " " "*" " " " " " " " " "*" "*" " " "*"
## 10  ( 1 ) " " "*" "*" "*" " " "*" "*" " " "*" " " " " "*" " " "*" "*" " " "*"
## 11  ( 1 ) " " "*" "*" "*" " " "*" "*" " " "*" "*" " " "*" " " "*" "*" " " "*"
## 12  ( 1 ) " " "*" "*" "*" " " "*" "*" " " "*" "*" " " "*" " " "*" "*" "*" "*"
## 13  ( 1 ) "*" "*" "*" "*" " " "*" "*" " " "*" "*" " " "*" "*" "*" "*" " " "*"
## 14  ( 1 ) "*" "*" "*" "*" " " "*" "*" " " "*" "*" " " "*" "*" "*" "*" "*" "*"
## 15  ( 1 ) "*" "*" "*" "*" " " "*" "*" " " "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 16  ( 1 ) "*" "*" "*" "*" "*" "*" "*" " " "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 17  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 18  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 19  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 20  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
##           X18 X19 X20
## 1  ( 1 )  " " " " " "
## 2  ( 1 )  " " " " " "
## 3  ( 1 )  " " " " " "
## 4  ( 1 )  " " " " " "
## 5  ( 1 )  " " " " " "
## 6  ( 1 )  " " " " " "
## 7  ( 1 )  " " " " " "
## 8  ( 1 )  " " " " " "
## 9  ( 1 )  " " " " " "
## 10  ( 1 ) " " " " " "
## 11  ( 1 ) " " " " " "
## 12  ( 1 ) " " " " " "
## 13  ( 1 ) " " " " " "
## 14  ( 1 ) " " " " " "
## 15  ( 1 ) " " " " " "
## 16  ( 1 ) " " " " " "
## 17  ( 1 ) " " " " " "
## 18  ( 1 ) "*" " " " "
## 19  ( 1 ) "*" " " "*"
## 20  ( 1 ) "*" "*" "*"
```

MSE is calculated by residual sum of squared divided by degrees of freedom.

```r
# best models per each information loss metric
results_best <- tibble(
  mse = which.min(results$rss / (900 - apply(results$which, 1, sum))),
  `adj_r2` = which.max(results$adjr2), # Adjusted r-squared
  BIC = which.min(results$bic), # Schwartz's information criterion
  # `c_p` = which.min(results$cp) # Mallows' Cp
) %>%
  gather(statistic, best)

# extract and plot results
tibble(mse = results$rss / (900-apply(results$which, 1, sum)),
       # `c_p` = results$cp,
       `adj_r2` = results$adjr2,
       BIC = results$bic) %>%
  mutate(predictors = row_number()) %>%
  gather(statistic, value, -predictors) %>%
  ggplot(aes(predictors, value, color = statistic)) +
  geom_line() +
  geom_point() +
  geom_vline(data = results_best,
             aes(xintercept = best, color = statistic), linetype = 2) +
  facet_wrap(~ statistic, scales = "free") +
  scale_color_brewer(type = "qual", guide = FALSE) +
  ggtitle("Subset selection")
```



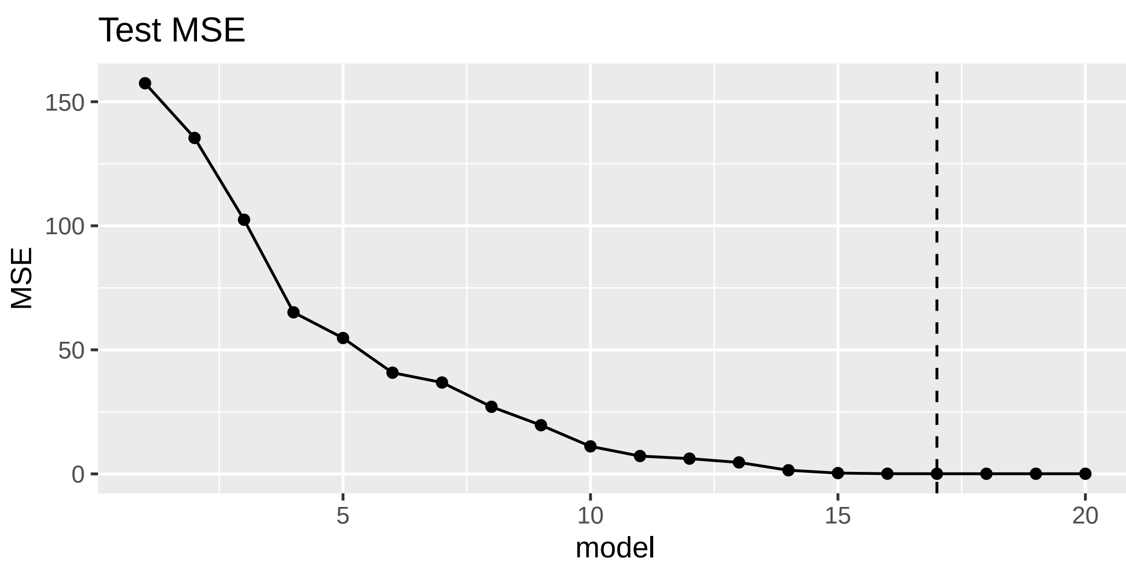
4. Plot the test set MSE

```r
# Define predict function typically for regsubsets (OOP!)
# This function is from http://www.science.smith.edu/~jcrouser/SDS293/labs/lab9-r.html
predict.regsubsets = function(object,newdata,id,...){
      form = as.formula(object$call[[2]]) # Extract the formula used when we called regsubsets()
      mat = model.matrix(form,newdata)    # Build the model matrix
      coefi = coef(object,id=id)          # Extract the coefficiants of the ith model
      xvars = names(coefi)                # Pull out the names of the predictors used in the ith model
      mat[,xvars]%*%coefi                 # Make predictions using matrix multiplication
}

val.errors = rep(NA,20)
for (i in 1:20){
  pred = predict(best_subset, test, id=i)
  val.errors[i] = mean((test$y-pred)^2)
}
val.errors = data.frame('model' = 1:20, 'MSE' = val.errors)
val.errors
```

model	MSE
1	157.4621526
2	135.4143456
3	102.4437412
4	65.1193782
5	54.7615075
6	40.7848469
7	36.8315032
8	27.0467086
9	19.6261491
10	11.0909825
11	7.1956494
12	6.1550549
13	4.6260797
14	1.4706086

model	MSE
15	0.3212926
16	0.0804768
17	0.0573670
18	0.0579004
19	0.0580335
20	0.0580330

```r
# Plot
best_MSE = val.errors$model[val.errors$MSE == min(val.errors$MSE)]
val.errors %>%
  ggplot(aes(x=model, y=MSE)) +
  geom_line() +
  geom_point() +
  geom_vline(aes(xintercept = best_MSE), linetype = 2) +
  scale_color_brewer(type = "qual", guide = FALSE) +
  ggtitle("Test MSE")
```



5.

From the MSE results above, the best model which has the minimum MSE is the model 17th, which contains all except three independent variables whose true beta is zero. This means that the subset selection properly identifies the true model.

6.

The estimated coefficients and the true intercept are the following:

```r
beta_hat = data.frame('Beta_hat'=c(coef(best_subset, 17), NA, NA, NA),
                      'sd'=c(sqrt(diag(vcov(best_subset, 17))), NA, NA, NA))
true_beta = data.frame('True_Beta' = c(mean(ep), beta))

rownames(true_beta) = c('Intercept', colnames(df[-21]))
cbind(beta_hat, true_beta)
```

	Beta_hat	sd	True_Beta
Intercept	-0.0018463	0.0085255	-0.0031689
X1	2.8757756	0.0147096	2.8598673
X2	5.2040917	0.0145610	5.2264123
X3	8.5817494	0.0147709	8.5620547
X4	9.6956283	0.0144018	9.6951894
X5	0.8273899	0.0150899	0.8315567
X6	5.1321769	0.0146162	5.1495462
X7	6.5319993	0.0145969	6.5341731
X8	0.2779805	0.0145105	0.2760716
X9	4.9916629	0.0146399	4.9921238
X10	3.2276018	0.0145564	3.2268088
X11	1.6728623	0.0147579	1.7016324
X12	3.7577918	0.0151946	3.7506407
X13	3.0491317	0.0149944	3.0517023
X14	8.6276771	0.0144434	8.6233271
X15	8.0467667	0.0147525	8.0309498
X16	2.6441161	0.0148271	2.6250869
X17	4.7617181	0.0142295	4.7565895
X18	NA	NA	0.0000000
X19	NA	NA	0.0000000
X20	NA	NA	0.0000000

As shown above table, the estimated coefficients are close to the true beta. Under parametric test, the distance betweent the two is closer than twice the standard deviation, meaning that the difference is not statistically significant with 5% level.

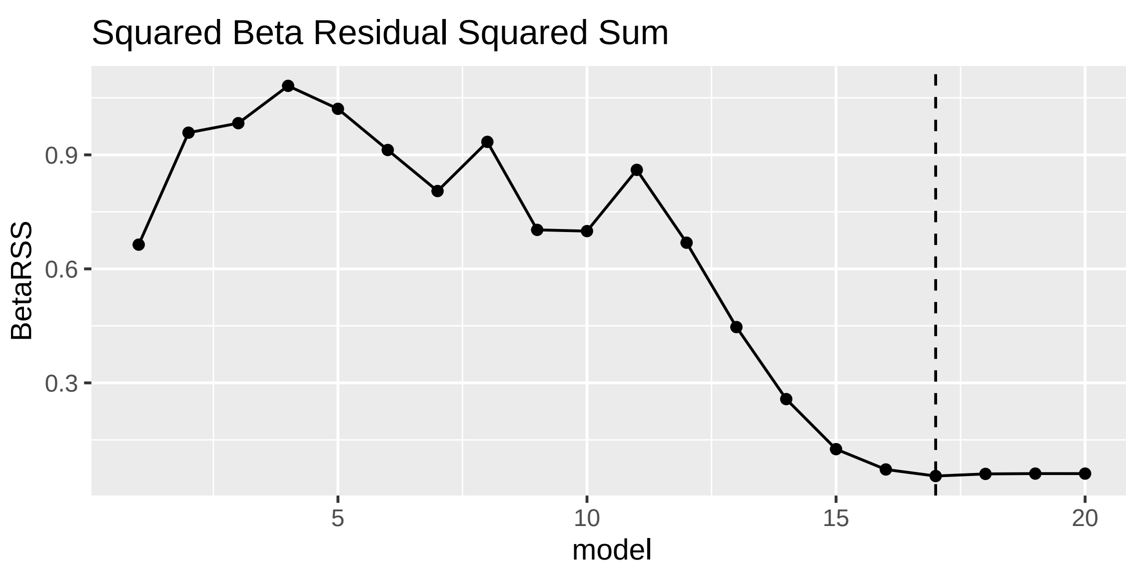
7.

```
val.betarss = rep(NA,20)
coln <- colnames(df[-21])
coefi <- coef(best_subset, id = 1)
for (i in 1:20) {
  coefi <- coef(best_subset, id = i)
  val.betarss[i] = sqrt(sum((beta[coln %in% names(coefi)]
                             - coefi[names(coefi) %in% coln])^2))
}
val.betarss = data.frame('model' = 1:20, 'BetaRSS' = val.betarss)
val.betarss
```

model	BetaRSS
1	0.6637318
2	0.9583835
3	0.9833700
4	1.0815494
5	1.0211163
6	0.9128218
7	0.8048314
8	0.9341899
9	0.7025852
10	0.6993811
11	0.8604987

model	BetaRSS
12	0.6687804
13	0.4468370
14	0.2572990
15	0.1255983
16	0.0721117
17	0.0548689
18	0.0604431
19	0.0612518
20	0.0612409

```
# Plot
best_betarss = val.betarss$model[val.betarss$BetaRSS == min(val.betarss$BetaRSS)]
val.betarss %>%
  ggplot(aes(x=model, y=BetaRSS)) +
  geom_line() +
  geom_point() +
  geom_vline(aes(xintercept = best_betarss), linetype = 2) +
  scale_color_brewer(type = "qual", guide = FALSE) +
  ggtitle("Squared Beta Residual Squared Sum")
```



The 17th model gets the minimum value, which is the same as we got with training set.

Application exercises

The General Social Survey

1. Least Squares Linear Model

```
gss_train <- read.csv('data/gss_train.csv')
gss_test <- read.csv('data/gss_test.csv')
```

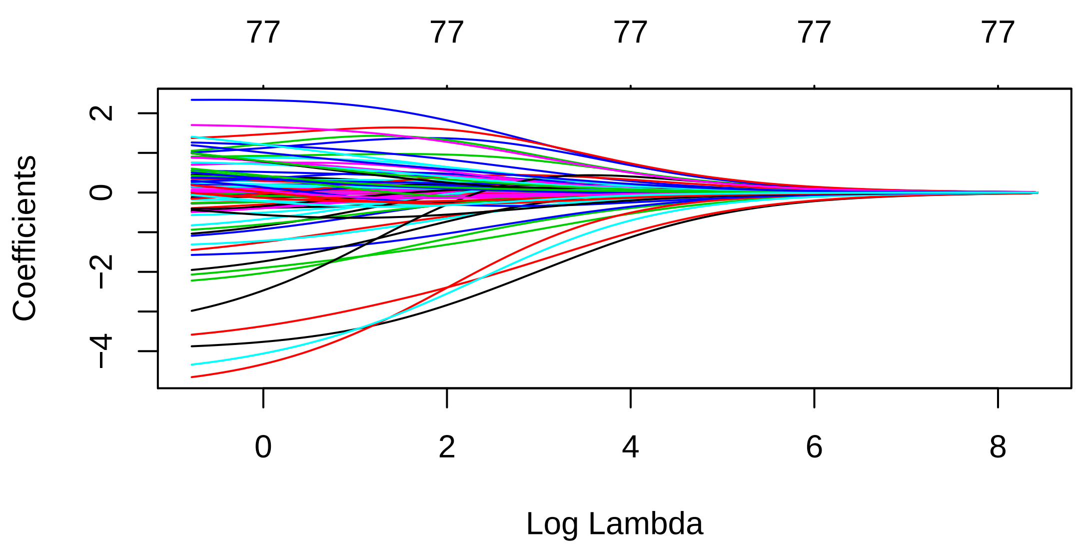
```
ls_model <- lm(egalit_scale ~ ., data=gss_train)
ls_pred <- predict(ls_model, gss_test)
ls_mse <- mean((ls_pred - gss_test$egalit_scale)^2)
ls_mse
```

```
## [1] 63.21363
```

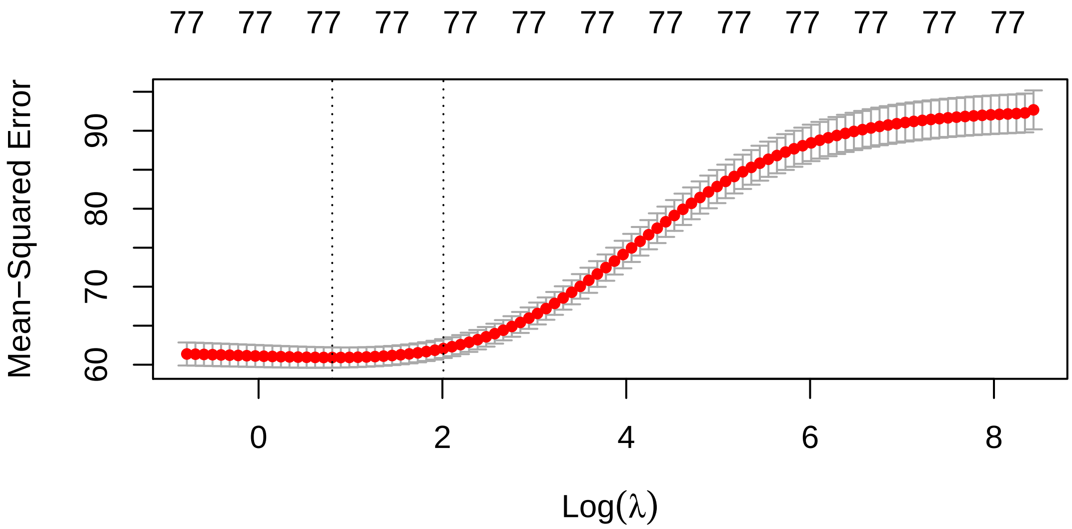
The test MSE is 63.21

2. Ridge Regression Model

```
gss_train_x <- model.matrix(egalit_scale ~ ., gss_train)[, -1]
gss_train_y <- gss_train$egalit_scale
gss_test_x <- model.matrix(egalit_scale ~ ., gss_test)[, -1]
gss_test_y <- gss_test$egalit_scale

ridge_model <- glmnet(x = gss_train_x,
                      y = gss_train_y,
                      alpha = 0,
                      nfolds = 10
                      )
plot(ridge_model, xvar = "lambda")
```




```
ridge_cv <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0,
  nfolds = 10
)
plot(ridge_cv)
```



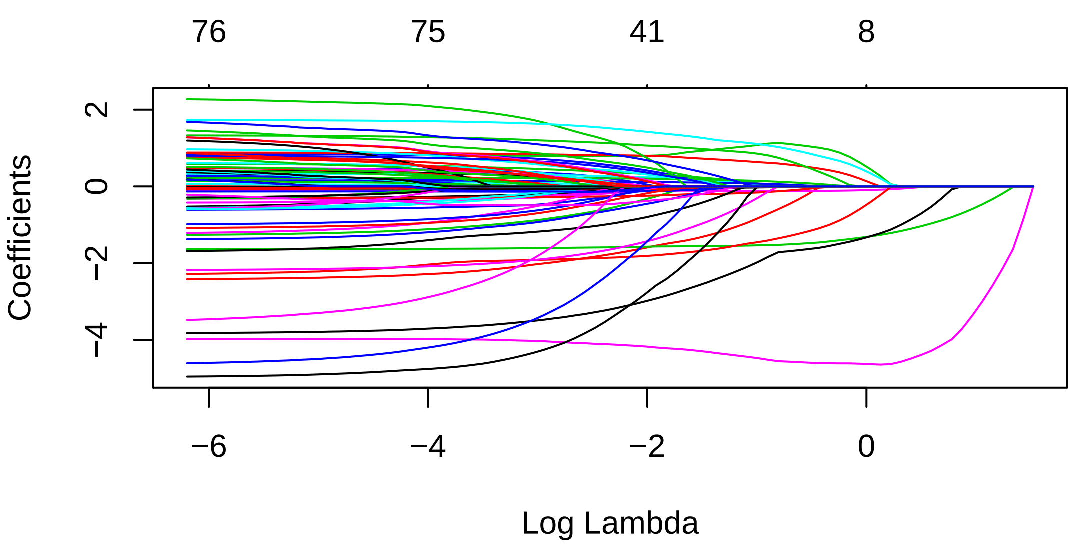

```r
ridge_best <- ridge_cv$lambda.min
ridge_pred <- predict(ridge_model, s = ridge_best, newx = gss_test_x)
ridge_mse <- mean((ridge_pred - gss_test_y)^2)
ridge_mse
```

```
## [1] 61.03781
```

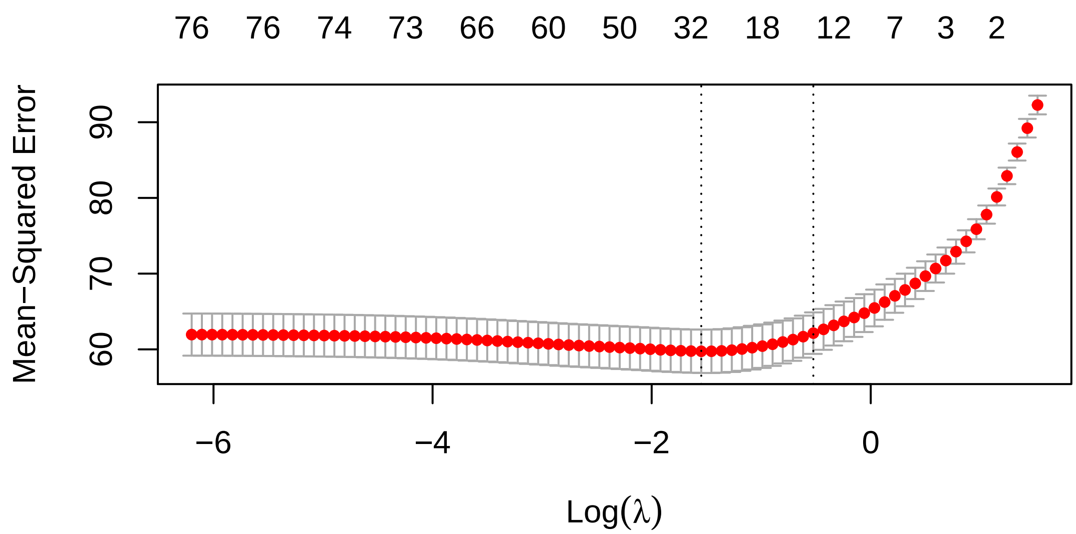
The test MSE is 61.038

3. Lasso Regression

```r
lasso_model <- glmnet(x = gss_train_x, y = gss_train_y, alpha = 1, nfolds = 10)
plot(lasso_model, xvar = "lambda")
```



```r
lasso_cv <- cv.glmnet(x = gss_train_x, y = gss_train_y, alpha = 1, nfolds = 10)
plot(lasso_cv)
```




```r
lasso_best <- lasso_cv$lambda.min
lasso_cv$nzero[lasso_cv$lambda == lasso_best]
```

```
## s33
##  29
```

```r
lasso_pred <- predict(lasso_model, s = lasso_best, newx = gss_test_x)
lasso_mse <- mean((lasso_pred - gss_test_y)^2)
lasso_mse
```

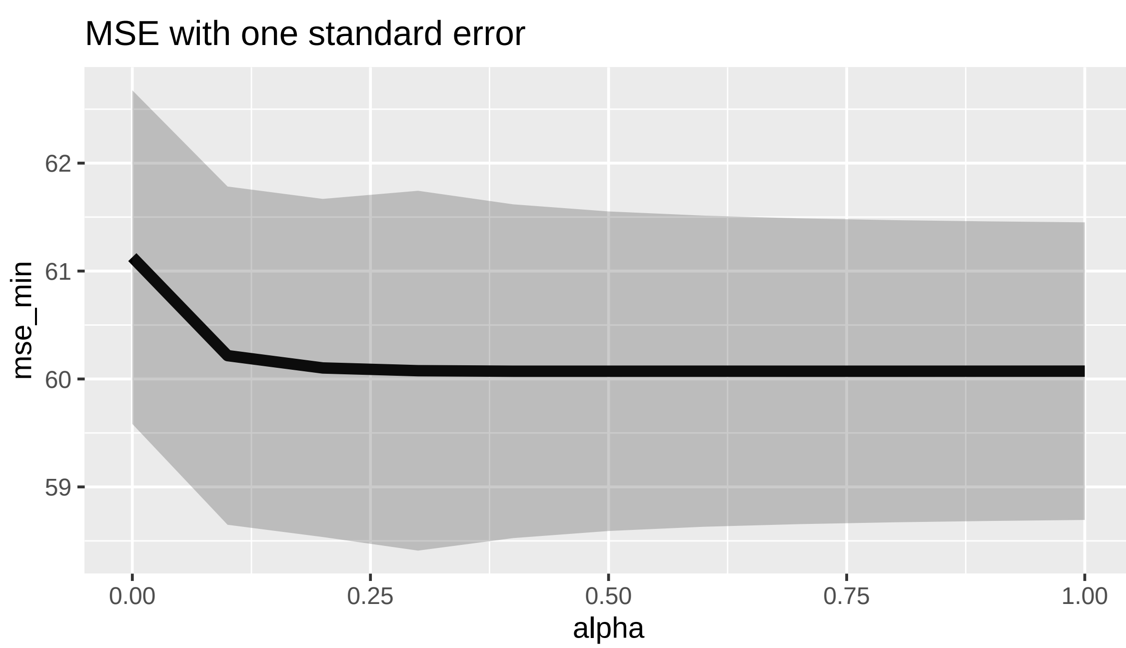
```
## [1] 61.22341
```

The test MSE is 61.22 and the number of non-zero coefficients are 29.

4. Elastic Net Regression Model

```r
# maintain the same folds across all models
fold_id <- sample(1:10, size = length(gss_train_y), replace = TRUE)

# search across a range of alphas
tuning_grid <- tibble::tibble(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,
  mse_1se    = NA,
  lambda_min = NA,
  lambda_1se = NA,
  non_zero = NA,
  test_mse = NA
)

for(i in seq_along(tuning_grid$alpha)) {
  # fit CV model for each alpha value
```

```r
  fit <- cv.glmnet(gss_train_x,
                   gss_train_y,
                   alpha = tuning_grid$alpha[i],
                   foldid = fold_id)

  # extract MSE and lambda values
  tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda == fit$lambda.min]
  tuning_grid$mse_1se[i]    <- fit$cvm[fit$lambda == fit$lambda.1se]
  tuning_grid$lambda_min[i] <- fit$lambda.min
  tuning_grid$lambda_1se[i] <- fit$lambda.1se
  tuning_grid$non_zero[i] <- fit$nzero[fit$lambda == fit$lambda.min]

  ela_best <- fit$lambda.min
  ela_pred <- predict(fit, s = ela_best, newx = gss_test_x)
  ela_mse <- mean((ela_pred - gss_test_y)^2)
  tuning_grid$test_mse[i] <- ela_mse
}

tuning_grid %>%
  mutate(se = mse_1se - mse_min) %>%
  ggplot(aes(alpha, mse_min)) +
  geom_line(size = 2) +
  geom_ribbon(aes(ymax = mse_min + se, ymin = mse_min - se), alpha = .25) +
  ggtitle("MSE with one standard error")
```




```r
tuning_grid %>%
  filter(mse_min == min(mse_min))
```

alpha	mse_min	mse_1se	lambda_min	lambda_1se	non_zero	test_mse
0.7	60.07186	61.48857	0.3039779	0.7022284	29	61.18543

```r
ela_mse <- tuning_grid %>%
  filter(mse_min == min(mse_min)) %>%
  select(test_mse) %>%
  pull
ela_mse
```

```
## [1] 61.18543
```

From the results above, the lowest cross-validation MSE is under the combination of $\alpha = 0.7$ where non-zero coefficients are 29. The test MSE on that model is 61.19, which is in the middle of the results of Ridge and Lasso.

5. Comments

To capture the accuracy, I calculate R squared for each model:

$$R^2 = 1 - \frac{MSE}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

```r
ega_bar <- mean(gss_test_y)
ega_var <- mean((ega_bar - gss_test_y)^2)
final_results <- data.frame('model'=c('LS Linear', 'Ridge', 'Lasso', 'Elastic_Net'),
                            'MSE' = c(ls_mse, ridge_mse, lasso_mse, ela_mse))
final_results <- final_results %>%
  mutate(R2 = 1- MSE/ega_var)
final_results
```

model	MSE	R2
LS Linear	63.21363	0.3004585
Ridge	61.03781	0.3245369
Lasso	61.22341	0.3224829
Elastic_Net	61.18543	0.3229033

From the results above, the best model is Ridge regression model. However, the MSEs and R squares are close to each other, so model does not matter very much. The R squared indicates that the model explains the dependent variable (egalitarianism) about 30%, which is low for prediction.