HW03

Minyoung Do

2/4/2020

Conceptual exercises

Training/test error for subset selection

(5 points) Generate a data set with $p = 20$ features, $n = 1000$ observations, and an associated quantitative response vector generated according to the model

$$Y = X\beta + \epsilon$$

where β has some elements that are exactly equal to zero.

```
# setting the seed
set.seed(420)

# function to create a data frame with random numbers betwwen 0 to 10
props <- function(ncol, nrow, var.names=NULL){
  if (ncol < 2) stop("ncol must be greater than 1")
  p <- function(n){
    y <- 0
    z <- sapply(seq_len(n-1), function(i) {
      x <- sample(seq(0, 10, by=.01), 1)
      y <<- y + x
      return(x)
    }
    )
  }
  DF <- data.frame(t(replicate(nrow, p(n=ncol))))
  if (!is.null(var.names)) colnames(DF) <- var.names
  return(DF)
}

# Data frame of variables and beta values
data <- props(ncol = 21, nrow = 1000)
# beta vector with random zero values
beta = runif(20, 0, 10)
beta[c(2, 7, 16, 19)] = 0
# adding noise to the data frame
error = rnorm(length(data), mean = 0, sd = 0.001)
# final data frame
finaldata <- data %>%
```

```
  mutate(Y = as.matrix(data) %*% beta + error)

head(finaldata)
```

(10 points) Split your data set into a training set containing 100 observations and a test set containing 900 observations.

```
# splitting data into training and test sets
split <- initial_split(finaldata, prop = .1)
train <- training(split)
test  <- testing(split)
```

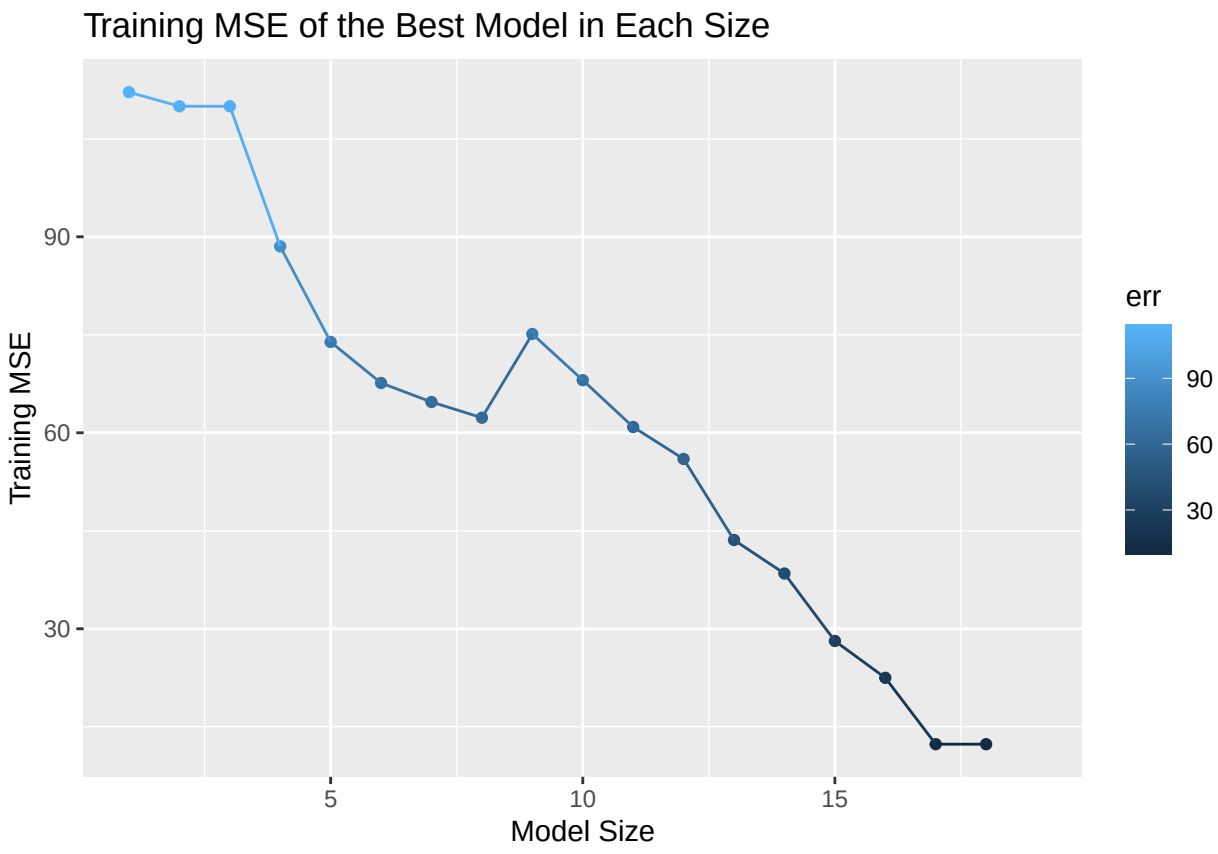
(10 points) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size. For which model size does the training set MSE take on its minimum value?

```
# running a best subset selection on the training set
best_subset <- regsubsets(Y ~ .,
                          data = train,
                          # setting the maximum size of subsets to examine
                          nvmax = 20
                          )
```

```
## Reordering variables and trying again:
```

```
results <- summary(best_subset)

# best models per each information loss metric
results_best <- tibble(
  `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)

# calculating MSE values across multiple models
# referenced this post: https://rpubs.com/davoodastaraky/subset
training.mat <- model.matrix(Y ~ ., data = train)

# creating a vector to fill out with for loop result
val.errors <- rep(NA, 19)

# for loop of calculating the MSE values for each model
for (i in 1:18){
        coefi = coef(best_subset, id=i)
```

```
        pred = training.mat[,names(coefi)]%*%coefi
        val.errors[i] = mean((train$Y-pred)^2)
}

# creating a data frame containing the MSE values with index of which model it belongs to
verr <- as.data.frame(val.errors);  names(verr) <- "err"
index <- c(1:nrow(verr))
verr <- cbind.data.frame(verr,index)

# plotting the MSE values across the best models of each size
verr %>%
  ggplot(aes(x = index, y = err)) +
  geom_point(aes(color = err)) +
  geom_line(aes(color = err)) +
  labs(x = "Model Size", y = "Training MSE", title = "Training MSE of the Best Model in Each Si
```




```
# to see which model size has the smallest training MSE.
which.min(val.errors)
```

```
## [1] 17
```

As can be seen in the plot, the best model with the lowest MSE has the size of 17. This is an

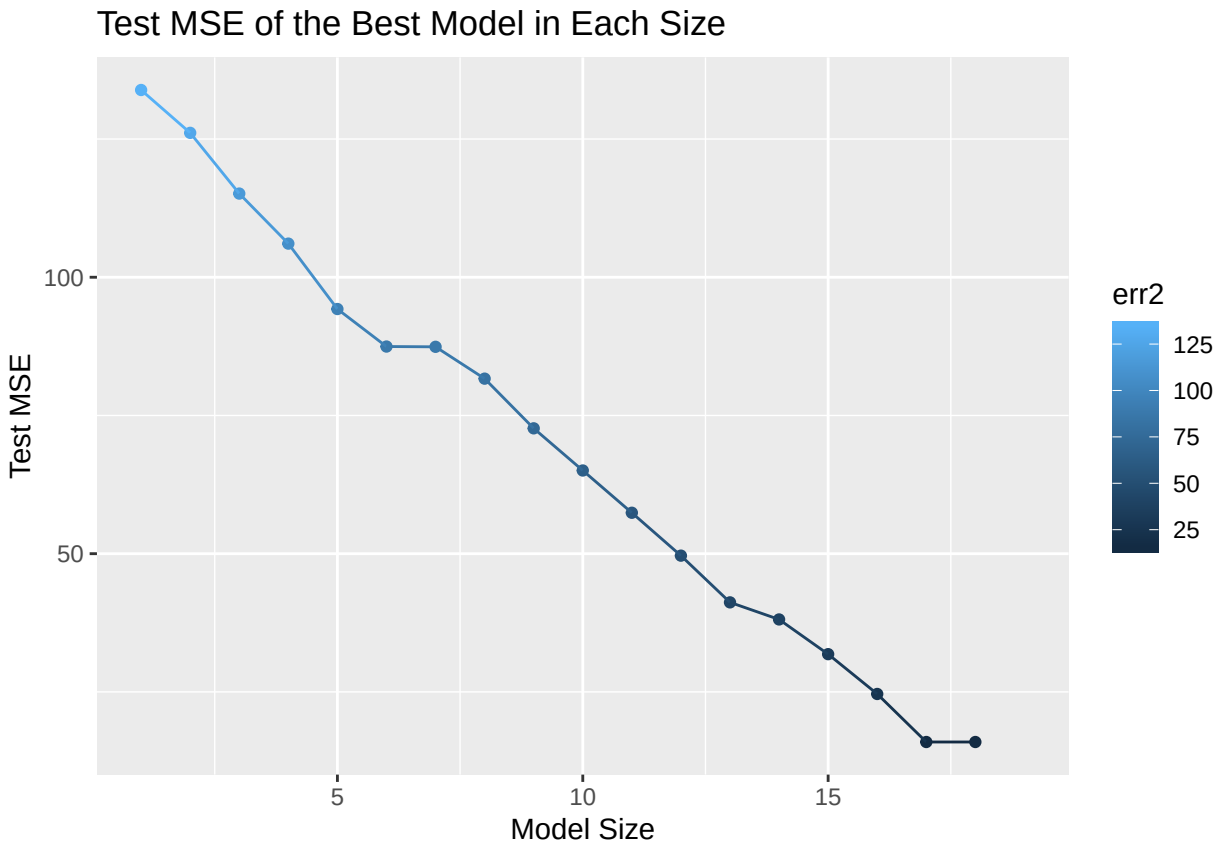
expected result as we learned that, in general, the more predictors, the higher prediction accuracy in the model.

(5 points) Plot the test set MSE associated with the best model of each size.

```
best_subset_2 <- regsubsets(Y ~ .,
                            data = test,
                            # setting the maximum size of subsets to examine
                            nvmax = 20
                            )
```

Reordering variables and trying again:

```
# plotting the test MSE
test.mat <- model.matrix(Y ~ ., data = test)

# creating a vector to fill out with for loop result
val.errors2 <- rep(NA, 19)

# for loop of calculating the MSE values for each model
for (i in 1:18){
        coefi = coef(best_subset_2, id=i)
        pred = test.mat[,names(coefi)]%*%coefi
        val.errors2[i] = mean((test$Y-pred)^2)
}

# creating a data frame containing the MSE values with index of which model it belongs to
verr2 <- as.data.frame(val.errors2); names(verr2) <- "err2"
index2 <- c(1:nrow(verr2))
verr2 <- cbind.data.frame(verr2,index2)

# plotting the MSE values across the best models of each size
verr2 %>%
  ggplot(aes(x = index2, y = err2)) +
  geom_point(aes(color = err2)) +
  geom_line(aes(color = err2)) +
  labs(x = "Model Size", y = "Test MSE", title = "Test MSE of the Best Model in Each Size")
```



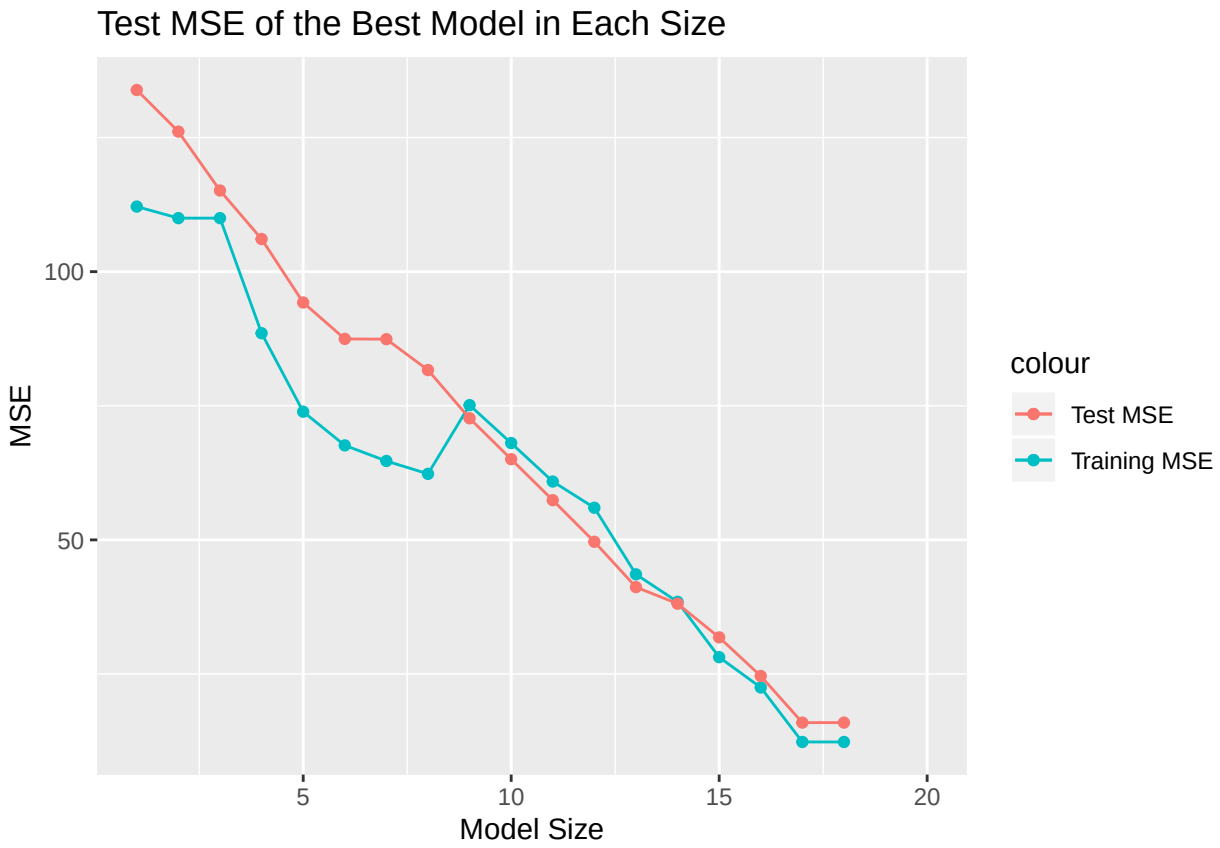
(5 points) For which model size does the test set MSE take on its minimum value? Comment on your results.

> If it takes on its minimum value for a model containing only an intercept or a model containing all of the features, then play around with the way that you generate the data previously until you create a data generating process in which the test set MSE is minimized for an intermediate model size.

```
test_min <- which.min(val.errors2)
test_min
```

```
## [1] 17
```

```
ggplot() +
  geom_point(data = verr, aes(x = index, y = err, color = "Training MSE")) +
  geom_line(data = verr, aes(x = index, y = err, color = "Training MSE")) +
  geom_point(data = verr2, aes(x = index2, y = err2, color = "Test MSE")) +
  geom_line(data = verr2, aes(x = index2, y = err2, color = "Test MSE")) +
  labs(x = "Model Size", y = "MSE", title = "Test MSE of the Best Model in Each Size")
```



It seems that the model with 17 coefficients, which is the same with the training MSE, has the lowest test MSE. It seems that the training models with less than 8 predictors tend to overfit the data, and the prediction accuracy of models becomes similar as the number of predictors increases.

(10 points) How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient sizes.

```
True <- coef(best_subset, id = 17) %>%
  as.data.frame() %>%
  rename(True = ".")

Model <-coef(best_subset_2, id = 17) %>%
  as.data.frame() %>%
  rename(Model = ".")

table_dif <- cbind(True, Model) %>%
  mutate(Difference = True - Model)
table_dif
```

```
##           True       Model  Difference
## 1   12.4646156   8.4025014  4.06211422
## 2    0.9180268   0.9934744 -0.07544762
## 3    1.0741501   1.1057644 -0.03161433
```

```
## 4    0.9213462   1.0341203 -0.11277410
## 5    1.0262152   1.0580922 -0.03187698
## 6    1.0464819   0.9517836  0.09469828
## 7    0.9666263   1.0279925 -0.06136620
## 8    1.1406789   1.0392666  0.10141231
## 9    0.7816627   1.0410389 -0.25937624
## 10   0.8617419   1.0170405 -0.15529856
## 11   0.7389148   1.0248100 -0.28589518
## 12   0.8208841   0.9885839 -0.16769987
## 13   1.0270959   0.9725917  0.05450421
## 14   1.1136714   1.0553399  0.05833142
## 15   0.9676285   1.0341261 -0.06649756
## 16   1.1353243   1.0081531  0.12717125
## 17   0.0000000   0.0000000  0.00000000
## 18 288.1168415 275.6380943 12.47874721
```

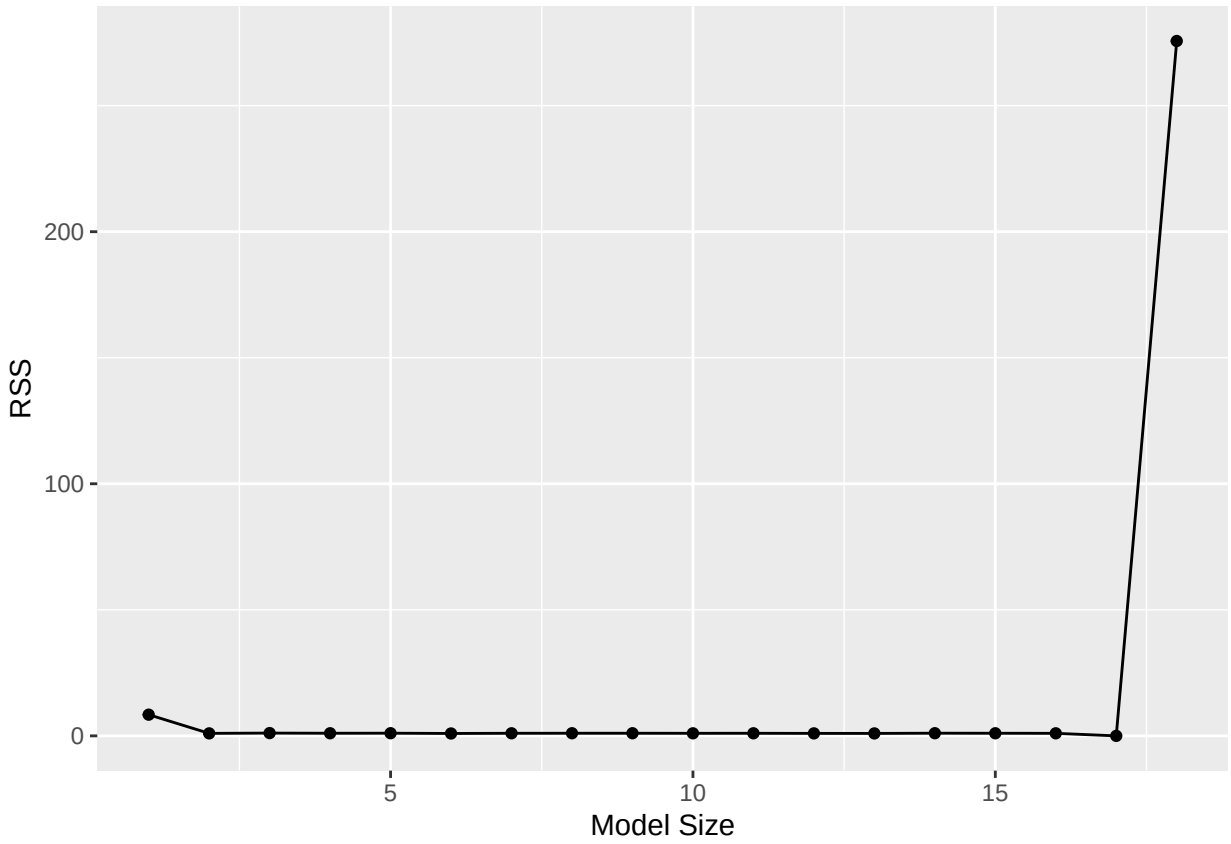
From the result, we can see that the difference between the coefficients of true model and best model are very similar; the differences are very small across the features. This indicates that our best model predicts the true data pretty well.

(10 points) Create a plot displaying

$$\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$$

for a range of values of r, where $\hat{\beta}_j^r$ is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the test MSE plot?

```
td <- table_dif %>%
  mutate(Difference = sqrt((True - Difference)^2)) %>%
  ggplot(aes(y = Difference, x = seq(1, length(Difference)))) +
  geom_point() +
  geom_line() +
  labs(y = "RSS", x = "Model Size")
td
```



This graph shows the RSS trend of differences between $\hat{\beta}$ and β across different model sizes, i.e., different number of predictors. There are only two zero RSS values in this plot. The model that only contains the intercept and the model with 18 predictors. The graph looks different from the test MSE plot, fluctuating up and down.

Application exercises

In this problem set, you are going to predict individual feelings towards egalitarianism. Specifically, egalit_scale is an additive index constructed from a series of questions designed to measure how egalitarian individuals are – that is, the extent to which they think economic opportunities should be distributed more equally in society. The feature ranges from 1 (low egalitarianism) to 35 (high egalitarianism).

Your task is to construct a series of statistical/machine learning models to accurately predict an individual's egalitarianism using model selection and regularization methods. Use all the available predictors for each model unless otherwise specified.

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

(10 points) Fit a least squares linear model on the training set, and report the test MSE.

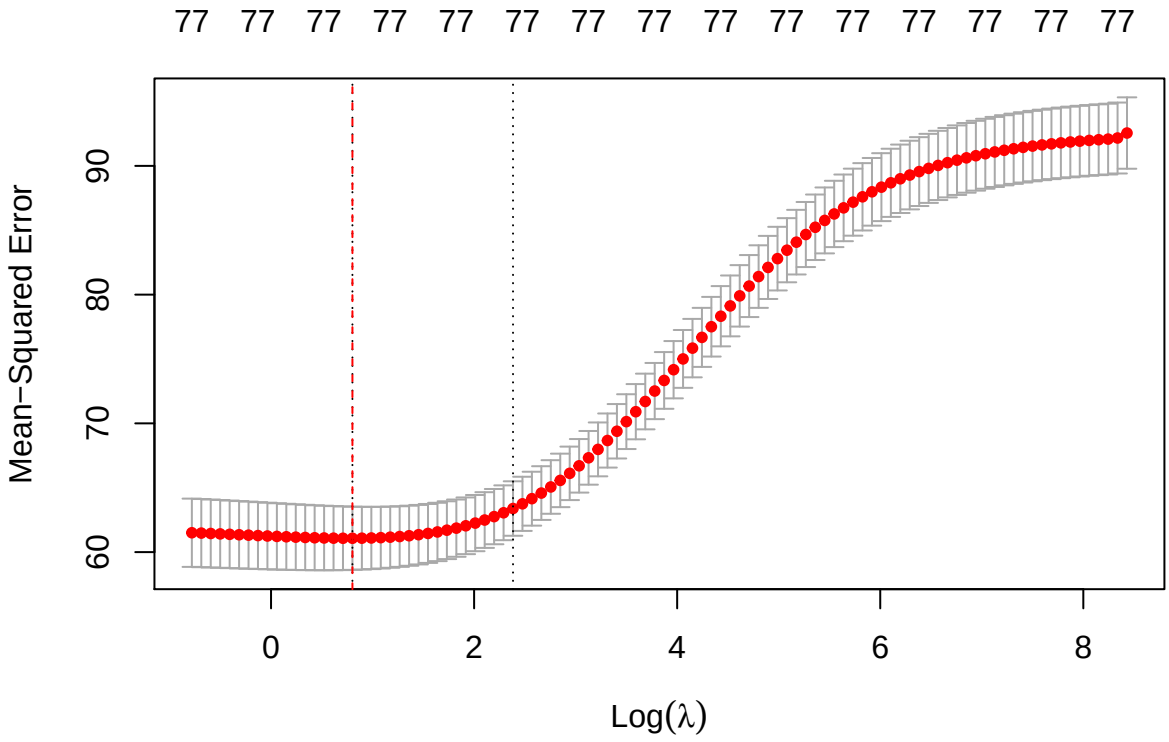
```r
lm_train <- lm(egalit_scale ~ ., data = gss_train)
pred_train <- predict(lm_train, gss_test)
Metrics::mse(gss_test$egalit_scale, pred_train)
```

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

(10 points) Fit a ridge regression model on the training set, with λ chosen by 10-fold cross-validation. Report the test MSE.

```r
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
```

```r
# ridge regression on the train data
gss_ridge <- glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0,
  nfolds = 10
)

# Apply CV Ridge regression to train data
gss_ridge_cv <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0,
  nfolds = 10
)

# minimum lambda value
gss_ridge_min <- gss_ridge_cv$lambda.min

# plotting the cross validation ridge regression with the minimum value in dashed line
plot(gss_ridge_cv, xvar = "lambda")
abline(v = log(gss_ridge_min), col = "red", lty = "dashed")
```



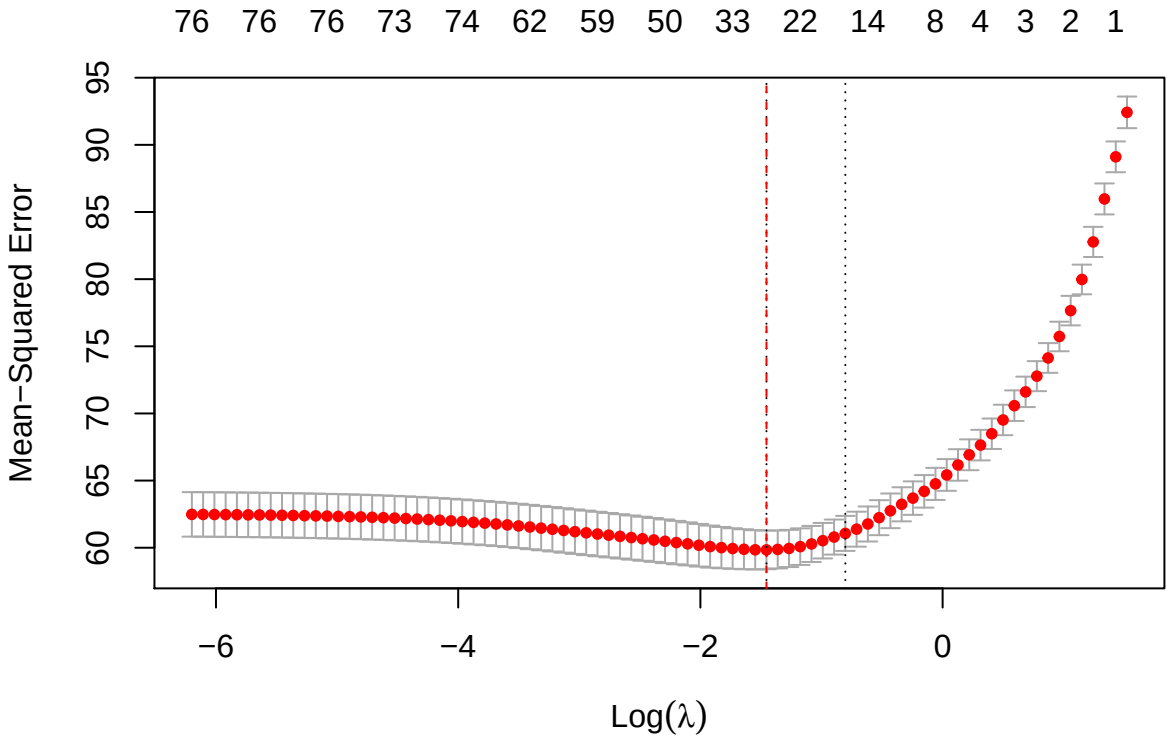

```
# calculating MSE
pred_ridge <- predict(gss_ridge, s = gss_ridge_min, newx = gss_test_x)
Metrics::mse(gss_test$egalit_scale, pred_ridge)
```

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

(10 points) Fit a lasso regression on the training set, with λ chosen by 10-fold cross-validation. Report the test MSE, along with the number of non-zero coefficient estimates.

```
# ridge regression on the train data
gss_lasso <- glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 1,
  nfolds = 10
)

# Apply CV Ridge regression to train data
gss_lasso_cv <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 1,
  nfolds = 10
)
```

```r
# minimum lambda value
gss_lasso_min <- gss_lasso_cv$lambda.min

# plotting the cross validation ridge regression with the minimum value in dashed line
plot(gss_lasso_cv, xvar = "lambda")
abline(v = log(gss_lasso_min), col = "red", lty = "dashed")
```




```r
# calculating MSE
pred_lasso <- predict(gss_lasso, s = gss_lasso_min, newx = gss_test_x)

Metrics::mse(gss_test$egalit_scale, pred_lasso)
```

```
## [1] 61.27006
```

```r
coef_lasso <- coef.glmnet(gss_lasso_cv, s = "lambda.min", newx = gss_test$egalit_scale)
```

Number of Coefficients: 32

(10 points) Fit an elastic net regression model on the training set, with α and λ chosen by 10-fold cross-validation. That is, estimate models with $\alpha = 0, 0.1, 0.2, \ldots, 1$ using the same values for λ across each model. Select the combination of α and λ with the lowest cross-validation MSE. For that combination, report the test MSE along with the number of non-zero coefficient estimates.

```r
# Now, more efficient grid search for varying alpha
fold_id <- sample(1:10, size = length(gss_train_y), replace = TRUE) # maintain the same folds

# 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
)

for(i in seq_along(tuning_grid$alpha)) {
  # fit CV model for each alpha value
  fit <- cv.glmnet(gss_train_x,
                   gss_train_y,
                   alpha = tuning_grid$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
}

# minimum MSE
min(tuning_grid[,2])
```

```
## [1] 59.76789
```

```r
# combination of alpha = 1 & lambda = 0.2127845
coef_en <- coef(fit, alpha = 1)
```

Number of Coefficients: 14

(5 points) Comment on the results obtained. How accurately can we predict an individual's egalitarianism? Is there much difference among the test errors resulting from these approaches?

As reported above, all three regularization methods have a lower MSE value than the least squares linear model; i.e., the prediction accuracy improves in regularized models. As expected, the elastic net regresion model has the lowest error in prediction compared to ridge and LASSO models. Also, the elastic net regression model has a smaller number of coefficients, which is 14, while LASSO has 32 coefficients in the model. This indicates that the elastic net model is not only more accurate, but also it is a simpler model. This result confirms why elastic net is always preferred over ridge or LASSO regression because it solves the limitations of both methods.