Problem Set 3

Pete Cuppernull

2/9/2020

Conceptual Exercises

Question 1. Generate Data

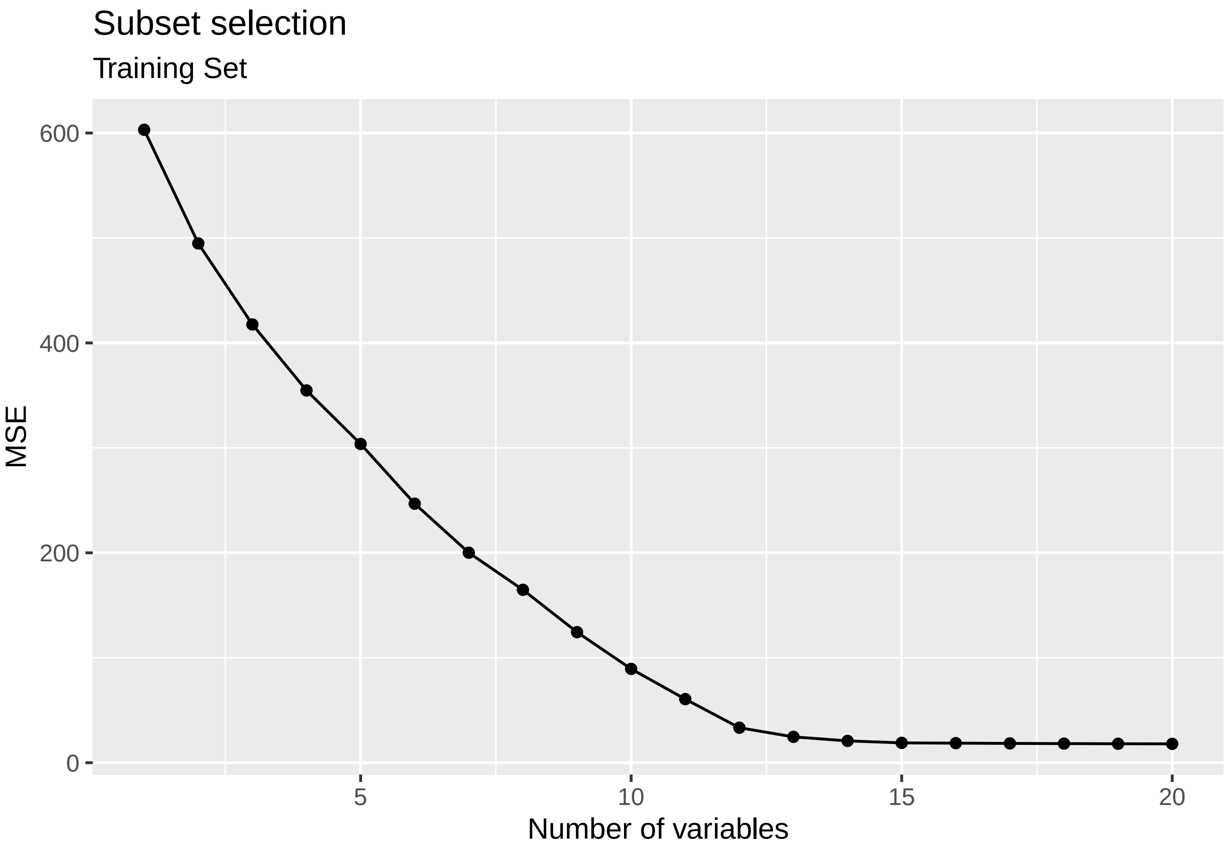
```
create_data <- function(){
  rnorm(1000, 3, 2)
}

data <- data.frame(replicate(20, create_data()))
betas <- sample(-500:500, 20, replace = TRUE)/100
betas[3] <- 0
betas[9] <- 0
betas[14] <- 0
betas[19] <- 0

error <- rnorm(1000, 0, 5)
data <- cbind(data, error)

data_scalar <- data %>%
  mutate(Y = X1*betas[1] + X2*betas[2] + X3*betas[3] +
             X4*betas[4] + X5*betas[5] + X6*betas[6] +
             X7*betas[7] + X8*betas[8] + X9*betas[9] +
             X10*betas[10] + X11*betas[11] + X12*betas[12] +
             X13*betas[13] + X14*betas[14] + X15*betas[15] +
             X16*betas[16] + X17*betas[17] + X18*betas[18] +
             X19*betas[19] + X20*betas[20] + error) %>%
  select(-error)
```

Question 2. Split Data

```
split <- initial_split(data_scalar, prop = .1)
train <- training(split)
test <- testing(split)
```

Question 3. Best Subset Selection

```
#do best subset
best_sub <- regsubsets(Y ~ .,
                          data = train,
                          nvmax = 20
                          )

##Calculate MSE

train_matrix <- model.matrix(Y ~., data=train)

get_mse <- function(model){
        coefi = coef(best_sub, id=model)
        pred = train_matrix[,names(coefi)]%*%coefi
        train_mse = mean((train$Y-pred)^2)
        return(train_mse)
}

mse <- map_dbl(1:20, get_mse)

train_mses <- as.data.frame(cbind(1:20, mse))


training_mse_plot <- ggplot(train_mses, aes(V1, mse)) +
  geom_line() +
  geom_point() +
  #geom_vline(xintercept = which.min(data_clean_cv_train$.estimate), linetype = 2) +
  labs(title = "Subset selection",
       subtitle = "Training Set",
       x = "Number of variables",
       y = "MSE")

training_mse_plot
```



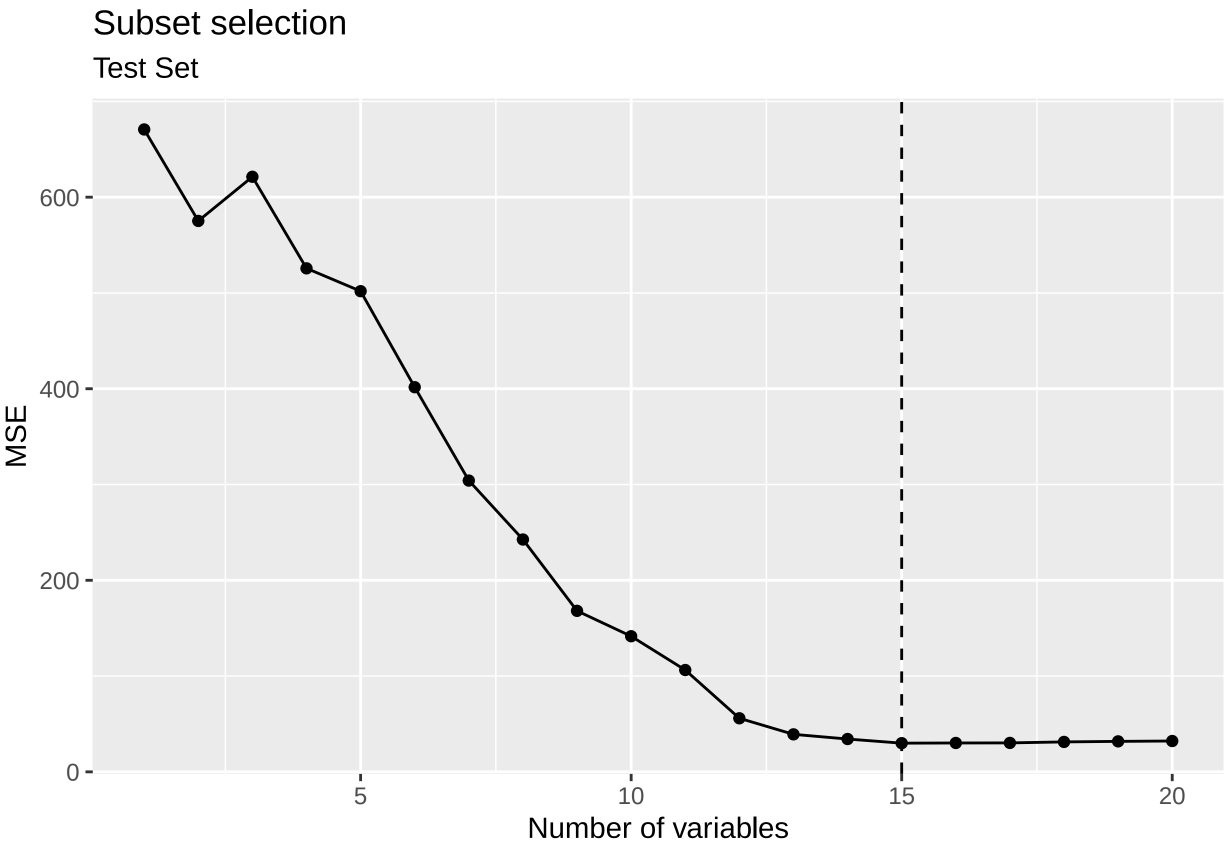
```
which.min(train_mses$mse)
```

```
## [1] 20
```

The model has the lowest MSE for the training set with 20 predictors.

Questions 4 and 5. Plot the test MSE and determine the model with the minimum MSE.

```
test_matrix <- model.matrix(Y ~., data=test)

get_mse_test2 <- function(model){
        coefi = coef(best_sub, id=model)
        pred_test = test_matrix[,names(coefi)]%*%coefi
        test_mse = mean((test$Y-pred_test)^2)
        return(test_mse)
}

mse_test2 <- map_dbl(1:20, get_mse_test2)

test_mses2 <- as.data.frame(cbind(1:20, mse_test2))

ggplot(test_mses2, aes(V1, mse_test2)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = which.min(test_mses2$mse_test2), linetype = 2) +
  labs(title = "Subset selection",
       subtitle = "Test Set",
       x = "Number of variables",
       y = "MSE")
```



The model size with the minimum test set MSE has 15 predictors. The models with greater than 15 predictors have slightly higher MSEs. This is in line with our expected results – MSEs for the test set will not decrease monotonically because the models will begin to be overfit once a certain threshold of predictors is reached.

Question 6.

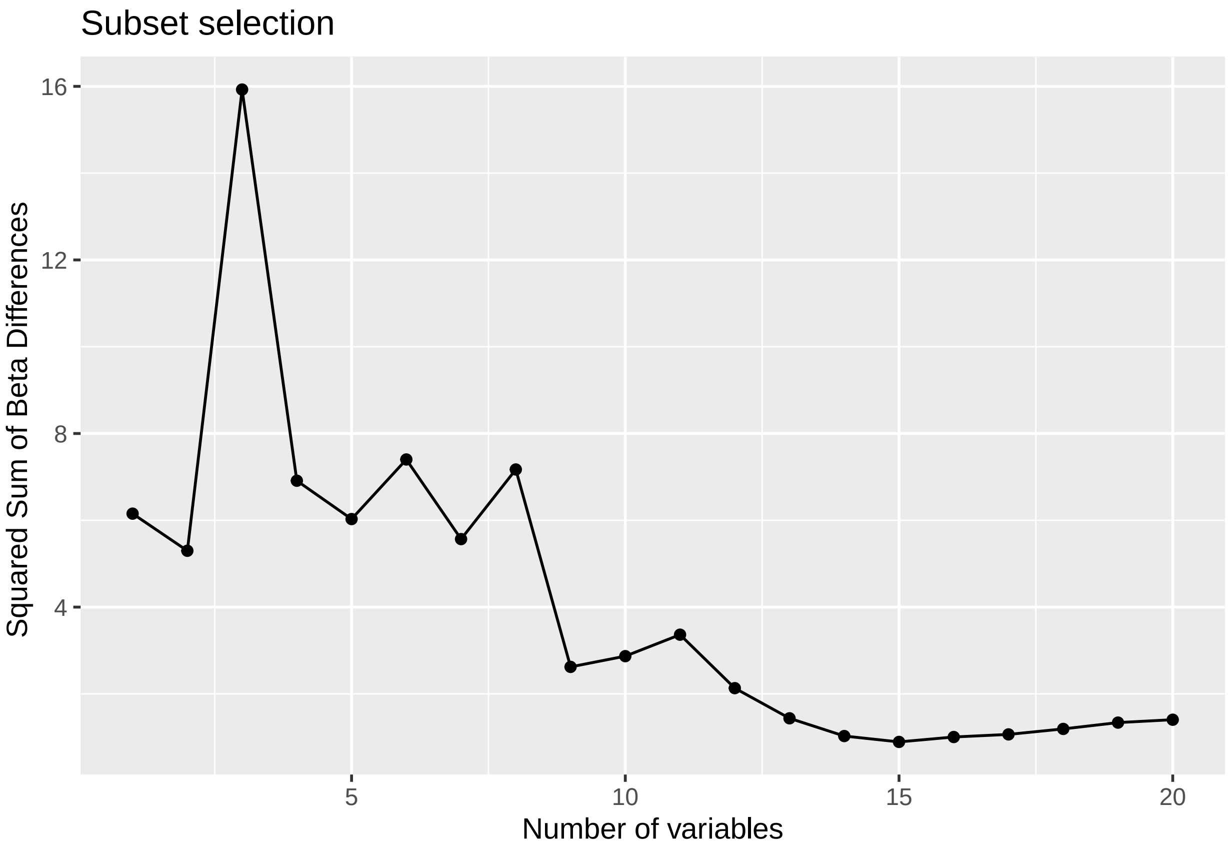
```
best_test_coefs <- rownames_to_column(as.data.frame(coef(best_sub, id=15))) %>%
  rename(best_test_coef = `coef(best_sub, id = 15)`)
beta_df <- cbind(paste("X", 1:20, sep = ""), as.data.frame(betas)) %>%
  rename(rowname = `paste("X", 1:20, sep = "")`,
         original_coef = betas)
coef_comparison <- left_join(best_test_coefs, beta_df) %>%
  na.omit()
print(coef_comparison)
```

```
##    rowname best_test_coef original_coef
## 2       X1     -0.7556753         -0.95
## 3       X4     -3.4123997         -3.49
## 4       X5      3.8784213          3.40
## 5       X6      4.3826966          4.01
## 6       X7      3.5055735          3.78
## 7       X8     -4.4519023         -4.34
## 8      X10      3.9193037          3.67
## 9      X11      1.7223539          1.92
## 10     X12     -2.6879245         -2.82
## 11     X13      3.7031473          3.59
## 12     X15      4.1387962          4.50
## 13     X16     -1.0381592         -1.00
## 14     X17     -3.0824912         -3.07
## 15     X18     -2.9983630         -2.74
## 16     X20     -4.5881831         -4.84
```

We can observe that there are notable differences in the coeficients of the original betas and the model with 15 predictors, As expected, none of the predictors with true betas of 0 remained in the model with 15 predictors. These differences are due to the one non-zero beta predictor that is left out of the 15 predictor model and the error term that was originally used to generate Y.

Question 7.

```
beta_df2 <- as.data.frame(cbind(paste("X", 1:20, sep = ""), betas))

q7_function <- function(number){
  best_model_coefs <- rownames_to_column(as.data.frame(coef(best_sub, id=number)))

  names(best_model_coefs) <- c("rowname", "coef")

  select_betas <- left_join(best_model_coefs, beta_df2, by = c("rowname" = "V1")) %>%
                       na.omit()

  final <- select_betas %>%
              mutate(outcome = (as.numeric(as.character(betas)) - coef)^2) %>%
              summarize(sum_outcome = sum(outcome))

    final
}

q7_outcomes <- map_dfr(1:20, q7_function)

q7 <- as.data.frame(cbind(1:20, q7_outcomes))

ggplot(q7, aes(`1:20`, sum_outcome)) +
  geom_line() +
  geom_point() +
  labs(title = "Subset selection",
       x = "Number of variables",
       y = "Squared Sum of Beta Differences ")
```



We observe that models with less than 10 predictors have coeficients that differ the most from the true betas. Once 15 predictors is reached, the squared sum of the beta differences begins to increase, suggesting that the coefficients of the models with 15 predictors and above are not converging further on the true betas. The MSEs of the models on the test data also began to increase at this same threshold, similarly suggesting that the models were not obtaining more accurate coefficients beyond the 15 predictor threshold.

Application Exercises

Import Data

```
set.seed(1414)
gss_test <- read_csv("data/gss_test.csv")
gss_train <- read_csv("data/gss_train.csv")
```

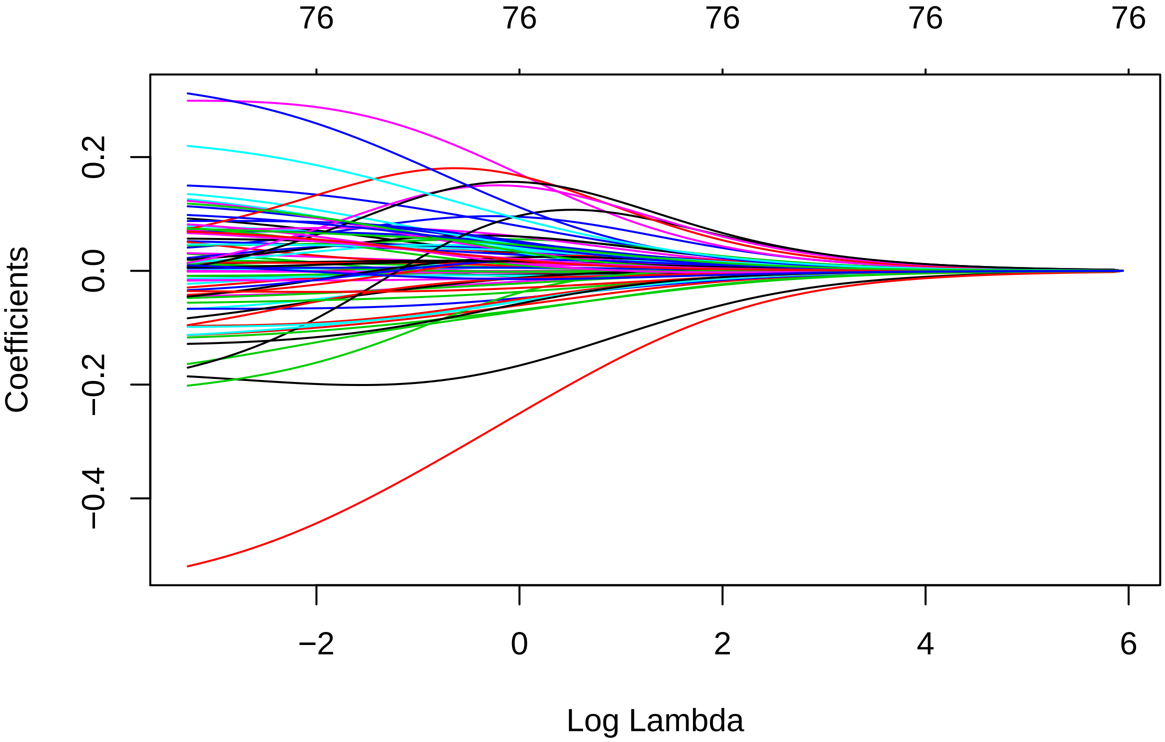
Question 1. LS Linear Model

```
lm_gss <- lm(egalit_scale ~ ., data = gss_train)

lm_gss_test <- predict(lm_gss,
                newdata = gss_test)

lm_mse <- mse(gss_test, gss_test$egalit_scale, lm_gss_test)$.estimate
```

The MSE of the test set for the least squares model is 63.2136296.

Question 2. Ridge Regression

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

gss_train_x <- model.matrix(egalit_scale ~ ., gss_test)[, -1]
gss_train_y <- log(gss_test$egalit_scale)

gss_ridge <- glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0
)

plot(gss_ridge, xvar = "lambda")
```

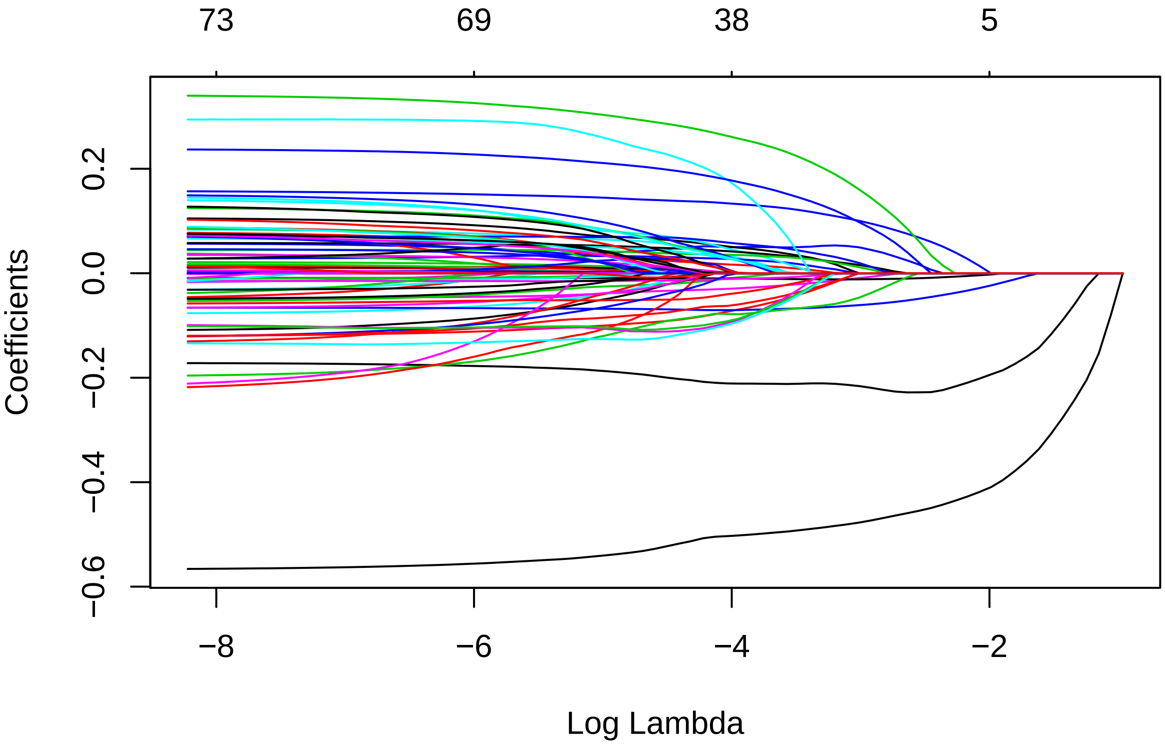


```
gss_ridge_cv <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0
)
lambda_min_ridge <- which(gss_ridge_cv$lambda == gss_ridge_cv$lambda.min)
ridge_mse <- gss_ridge_cv$cvm[lambda_min_ridge]
```

The MSE of the test set for the Ridge regression model is 0.4960419.

Question 3. Lasso Regression

```r
gss_lasso <- glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 1
)

plot(gss_lasso, xvar = "lambda")
```



```r
gss_lasso_cv <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 1
)

gss_lasso_cv
```

```
##
## Call:  cv.glmnet(x = gss_train_x, y = gss_train_y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Measure      SE Nonzero
## min 0.04481  0.4826 0.05652      18
## 1se 0.15020  0.5378 0.06016       4
```

```r
lambda_min_lasso <- which(gss_lasso_cv$lambda == gss_lasso_cv$lambda.min)
lasso_mse <- gss_lasso_cv$cvm[lambda_min_lasso]
```

The MSE of the test set for the Lasso regression model is 0.4825839 and there are 18 non-zero coefficients.

Question 4. Elastic Net

```
lasso    <- glmnet(gss_train_x, gss_train_y, alpha = 1.0)
elastic1 <- glmnet(gss_train_x, gss_train_y, alpha = 0.25)
elastic2 <- glmnet(gss_train_x, gss_train_y, alpha = 0.75)
ridge    <- glmnet(gss_train_x, gss_train_y, alpha = 0.0)

fold_id <- sample(1:10, size = length(gss_train_y), replace = TRUE)

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
}


elastic_mse <- min(tuning_grid$mse_min)

tuning_grid
```

```
## # A tibble: 11 x 5
##    alpha mse_min mse_1se lambda_min lambda_1se
##    <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
##  1   0     0.499   0.554     0.748       4.38
##  2   0.1   0.484   0.533     0.309       0.943
##  3   0.2   0.481   0.529     0.186       0.568
##  4   0.3   0.480   0.530     0.136       0.416
##  5   0.4   0.479   0.526     0.102       0.312
##  6   0.5   0.479   0.523     0.0896      0.249
##  7   0.6   0.479   0.530     0.0747      0.228
##  8   0.7   0.479   0.529     0.0640      0.196
##  9   0.8   0.479   0.528     0.0560      0.171
## 10   0.9   0.479   0.527     0.0498      0.152
## 11   1     0.479   0.527     0.0448      0.137
```

```
##Non zero coefs
cv.glmnet <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = tuning_grid[which(tuning_grid[,2]==min(tuning_grid[,2])), 1]
)

cv.glmnet
```

```
##
## Call:  cv.glmnet(x = gss_train_x, y = gss_train_y, alpha = tuning_grid[which(tuning_grid[,      2] ==
##
## Measure: Mean-Squared Error
##
##      Lambda Measure      SE Nonzero
## min 0.05315  0.4871 0.03515      22
## 1se 0.16231  0.5183 0.03672       5
```

The combination of α and λ that produce the lowest cross validation MSE are 0.7 and 0.0531498, respectively. The test MSE is 0.4793359 and there are 22 nonzero coefficients.

Question 5. Reflection.

The elastic net approach provides a slightly lower MSE than both ridge and lasso regression, at 0.4793359 compared to 0.4960419 and 0.4825839, respectively. Considering that egalitarianism is measured on a 35 point scale, this would imply that model predictions are in expectation about 0.7 points removed from the true value – I would argue that this is a relatively accurate estimate, which could potentially even be improved further with the inclusion of further relevant predictor variables.