Xu_Yilun_HW03

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

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

```
library(tibble)
library(dplyr)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.6.2
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.6.2
```

```
library(rcfss)
library(furrr)
```

```
## Warning: package 'furrr' was built under R version 3.6.2
```

```
## Loading required package: future
```

```
## Warning: package 'future' was built under R version 3.6.2
```

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 3.6.2
```

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

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

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

```
## Loaded glmnet 3.0-2
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

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

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.6.2
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 3.6.2
```

```
## For binary classification, the first factor level is assumed to be the event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.
```

```
##
## Attaching package: 'yardstick'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 3.6.2
```

```
set.seed(1234)
```

Conceptual Exercise 1

```
x <- matrix(rnorm(1000 * 20), ncol = 20)
error <- rnorm(1000)
coefs <- sample(seq(from = -5, to = 5), 20, replace = TRUE)
coefs[sample(seq(from = 1, to = 20), 5)] <- 0
y <- as.vector(x %*% coefs + error)
```

2

```
data <- as_tibble(x) %>%
  mutate(y = y)
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

```
data_split <- initial_split(data, prop = 0.1)
data_train <- training(data_split)
data_test <- testing(data_split)
```

3

```
predict.regsubsets <- function(object, newdata, id, ...){
  expression <- as.formula(object$call[[2]])
  matrix_here <- model.matrix(expression, newdata)
  coef_here <- coef(object, id = id)
  independent_vars <- names(coef_here)
  as.vector(matrix_here[, independent_vars] %*% coef_here)}

data_subsets <- regsubsets(y ~ ., data = data_train, nvmax = 20)

mse_train <- tibble(
  i = seq(from = 1, to = 20),
  preds = map(i, ~ predict(data_subsets, newdata = data_train, id = .x)),
  mse = map(preds, ~ tibble(
    facts = data_train$y, estimate = .x) %>%
      mse(facts, estimate))) %>%
  unnest(mse) %>%
  select(i, .estimate)

ggplot(mse_train, aes(i, .estimate)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of variables",
       y = "MSE for training data",
       title = 'Training set MSE and Model Size')+
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face= "bold"),
        axis.text=element_text(size=12,
                               face = "bold"),
        axis.title.x=element_text(size=14),
        axis.title.y=element_text(size=14))
```


```
filter(mse_train, .estimate == min(.estimate))
```

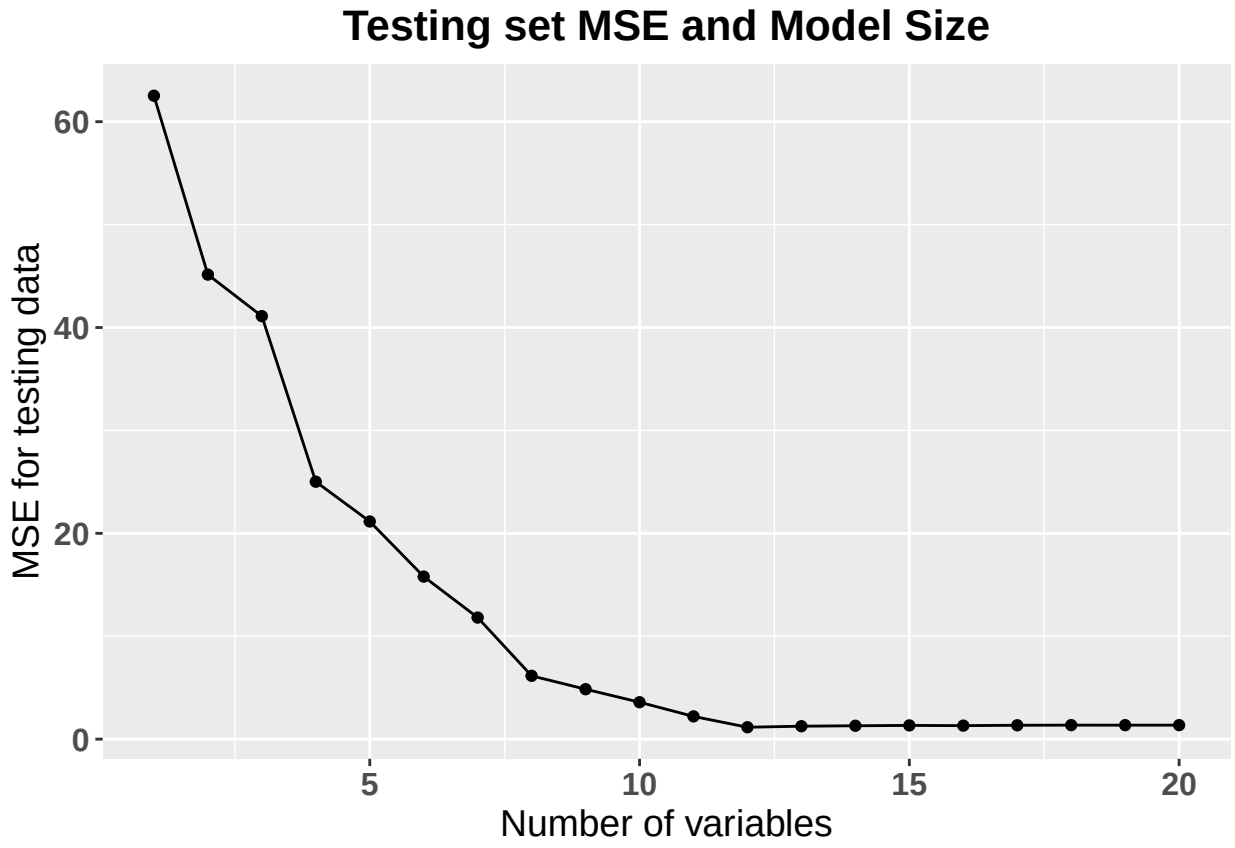
```
## # A tibble: 1 x 2
##       i .estimate
##   <int>     <dbl>
## 1    20     0.635
```

We can see that with the model size of 20, we achieve the minimum value.

4

```
data_subsets <- regsubsets(y ~ ., data = data_train, nvmax = 20)
mse_test <- tibble(
  i = seq(from = 1, to = 20),
  preds = map(i, ~ predict(data_subsets, newdata = data_test,
                           id = .x)),
  mse = map(preds, ~ tibble(
    facts = data_test$y, estimate = .x) %>%
      mse(facts, estimate))) %>%
  unnest(mse) %>%
  select(i, .estimate)

ggplot(mse_test, aes(i, .estimate)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of variables",
       y = "MSE for testing data",
       title = 'Testing set MSE and Model Size')+
```

```
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face= "bold"),
        axis.text=element_text(size=12,
                               face = "bold"),
        axis.title.x=element_text(size=14),
        axis.title.y=element_text(size=14))
```




```
filter(mse_test, .estimate == min(.estimate))
```

```
## # A tibble: 1 x 2
##       i .estimate
##   <int>     <dbl>
## 1    12      1.16
```

5 The best model size is 12. The best model of the subset contains all the "good" independent variables.

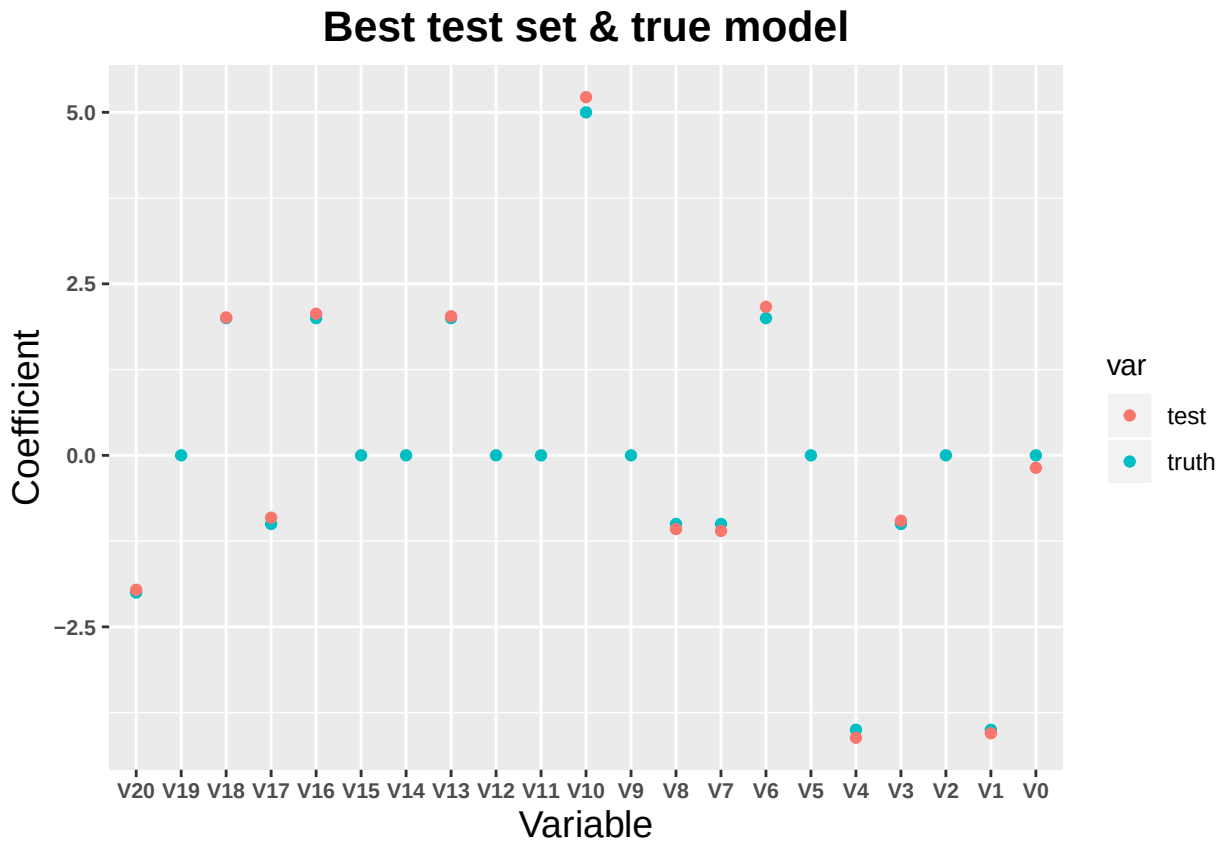
6

```
coef_best <- coef(data_subsets, id = which.min(mse_test$.estimate))
names(coef_best)[[1]] <- "V0"

tibble(
  beta = str_c("V", seq(from = 0, to = 20)),
  truth = c(0, coefs)) %>%
  left_join(enframe(coef_best, name = "beta",
                    value = "test")) %>%
  mutate(beta = factor(beta, levels = beta)) %>%
  gather(var, value, -beta) %>%
```

```
  ggplot(aes(fct_rev(beta), value, color = var)) +
  geom_point()+
  labs(x = "Variable",
       y = "Coefficient",
       title = 'Best test set & true model')+
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face= "bold"),
        axis.text=element_text(size=8,
                               face = "bold"),
        axis.title.x=element_text(size=14),
        axis.title.y=element_text(size=14))
```

```
## Joining, by = "beta"
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```



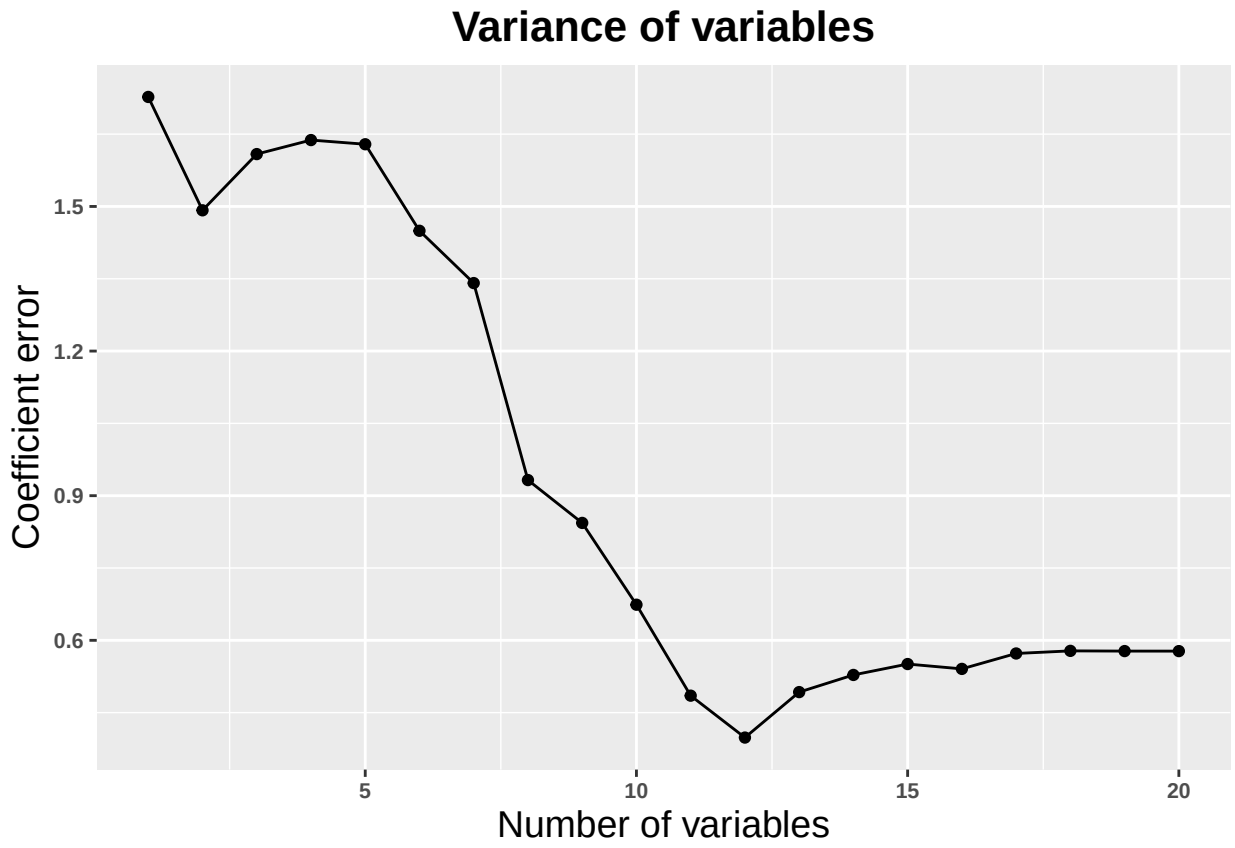
In the true model, more (and many) coefficients are equal to zero, and the variance among the coefficients are larger.

g

```
tibble(
  i = seq(from = 1, to = 20),
  coef = map(i, ~ coef(data_subsets, id = .x) %>%
               enframe(name = "beta", value = "estimate"))) %>%
  unnest(coef) %>%
  mutate(beta = str_replace(beta, "\\(Intercept\\)", "V0")) %>%
  left_join(tibble(beta = str_c("V", seq(from = 0, to = 20)),
```

```
                  truth = c(0, coefs))) %>%
  mutate(diff = (truth - estimate)^2) %>%
  group_by(i) %>%
  summarize(diff = sqrt(sum(diff))) %>%
  ggplot(aes(i, diff)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of variables", y = "Coefficient error",
       title = 'Variance of variables')+
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face= "bold"),
        axis.text=element_text(size=8, face = "bold"),
        axis.title.x=element_text(size=14),
        axis.title.y=element_text(size=14))
```

```
## Joining, by = "beta"
```



This plot looks similar to the test MSE plot.

Application exercises

1

```
g_train <- read_csv("C:/Users/mac/Desktop/hw03/gss_train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
```

```
## )
```

```
## See spec(...) for full column specifications.
```

```
g_test <- read_csv("C:/Users/mac/Desktop/hw03/gss_test.csv")
```

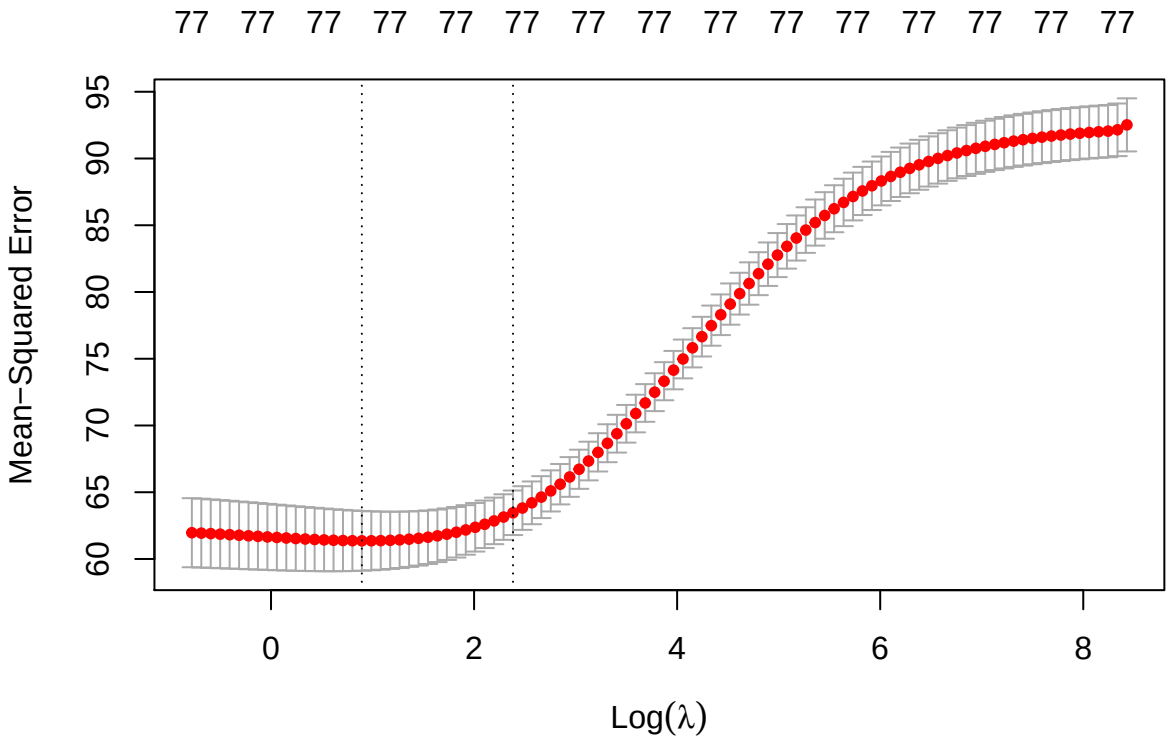
```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
## See spec(...) for full column specifications.
```

```
lsl <- lm(egalit_scale ~ ., data = g_train)
lsl_mse <- augment(lsl, newdata = g_test) %>%
    mse(truth = egalit_scale, estimate = .fitted)
lsl_mse
```

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

2

```
g_train_x <- model.matrix(egalit_scale ~ ., g_train)[, -1]
g_train_y <- g_train$egalit_scale
g_test_x <- model.matrix(egalit_scale ~ ., g_test)[, -1]
g_test_y <- g_test$egalit_scale

ridge <- cv.glmnet(x = g_train_x, y = g_train_y, alpha = 0)
plot(ridge)
```

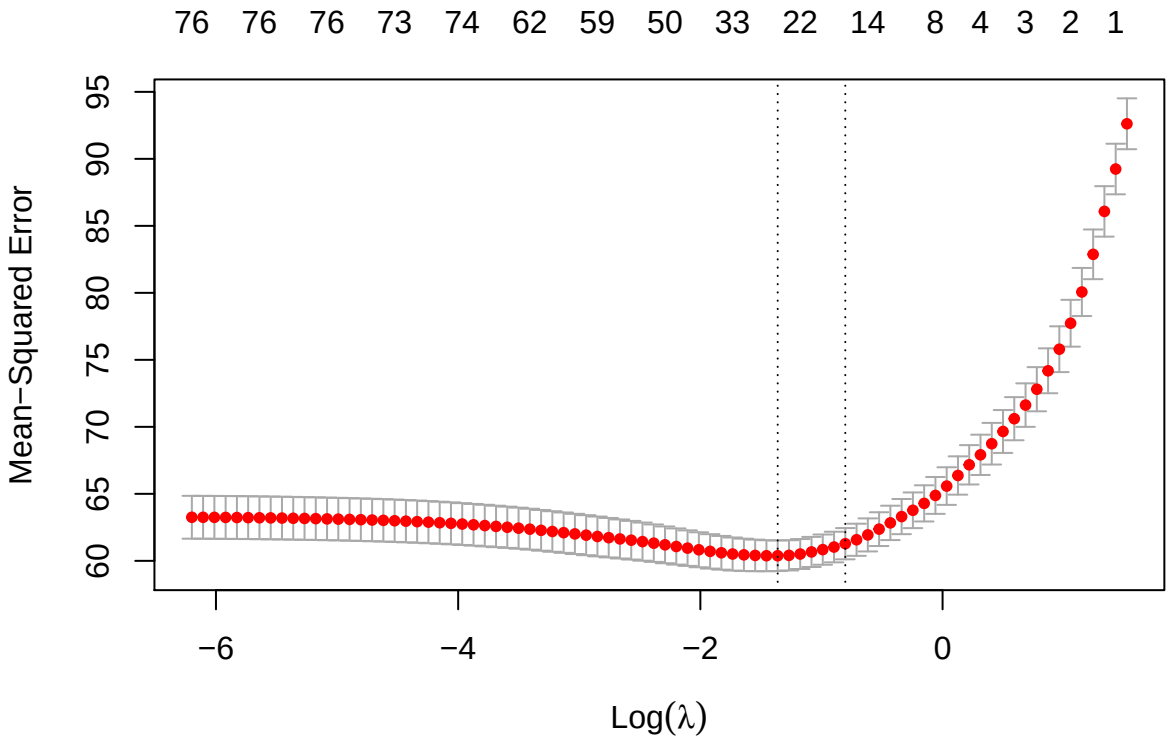



```
ridge_min <- glmnet(x = g_train_x,
                    y = g_train_y, alpha = 0)
ridge_mse <- g_test %>%
    mutate(.fitted = predict(ridge_min, newx = g_test_x,
                             s = ridge$lambda.1se)[,1]) %>%
    mse(truth = egalit_scale, estimate = .fitted)
ridge_mse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        61.9
```

3

```
lasso <- cv.glmnet(x = g_train_x,
                   y = g_train_y, alpha = 1)
plot(lasso)
```



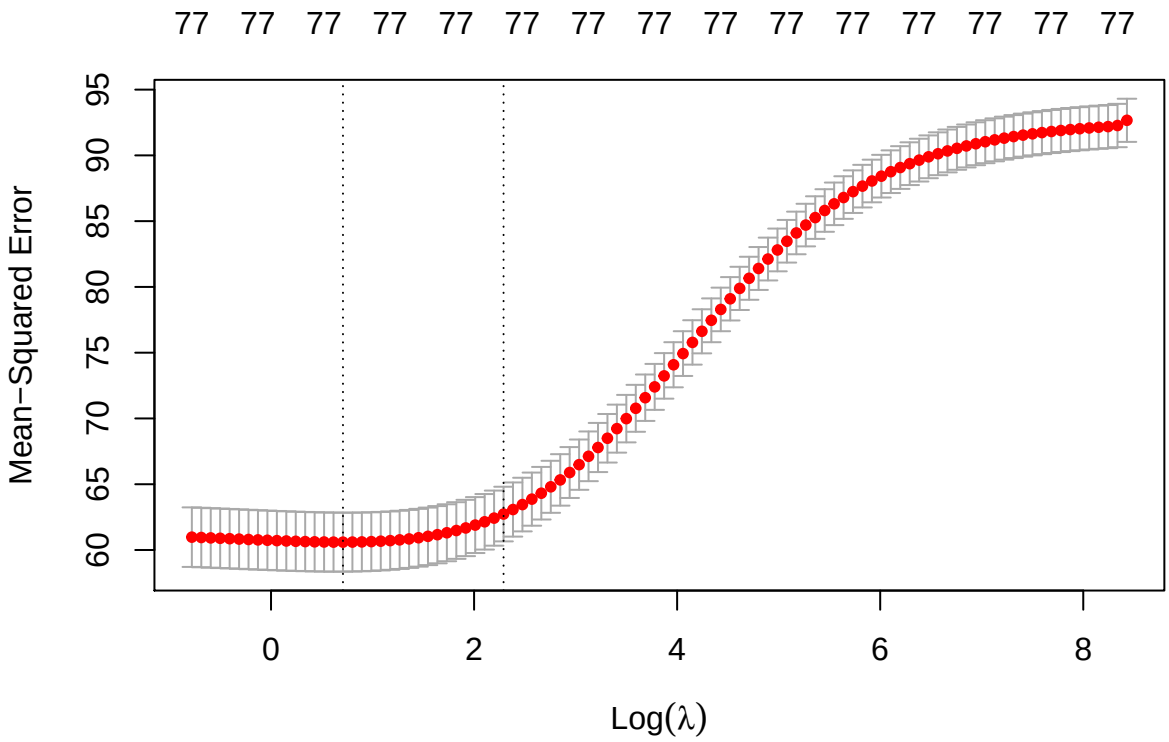

```
lasso_min <- glmnet(x = g_train_x,
                    y = g_train_y, alpha = 1)
lasso_mse <- g_test %>%
    mutate(.fitted = predict(lasso_min, newx = g_test_x,
                             s = lasso$lambda.1se)[,1]) %>%
    mse(truth = egalit_scale, estimate = .fitted)
lasso_mse
```

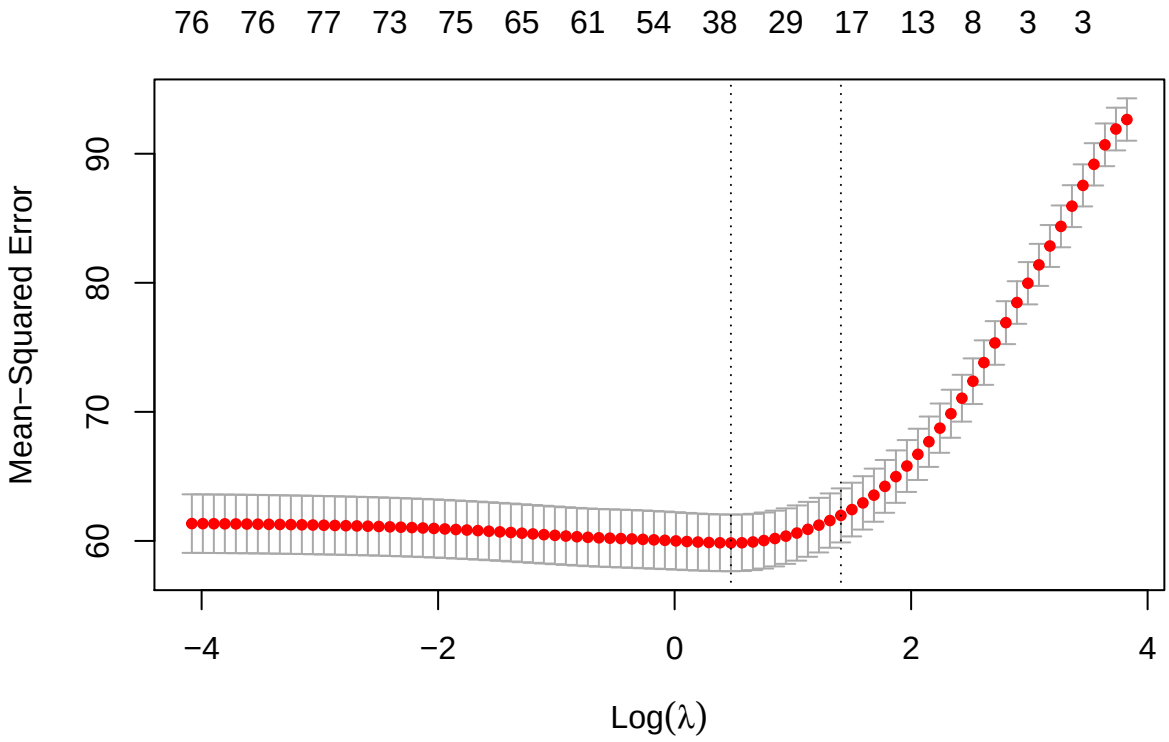
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        61.7
```

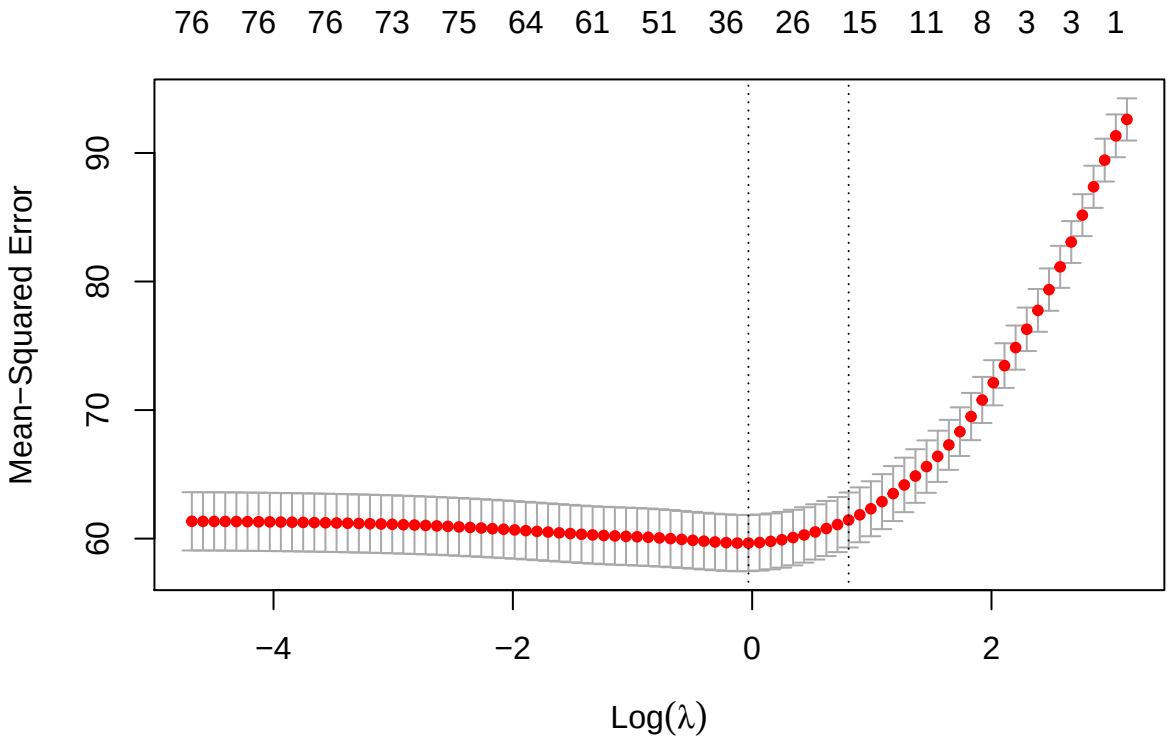
4

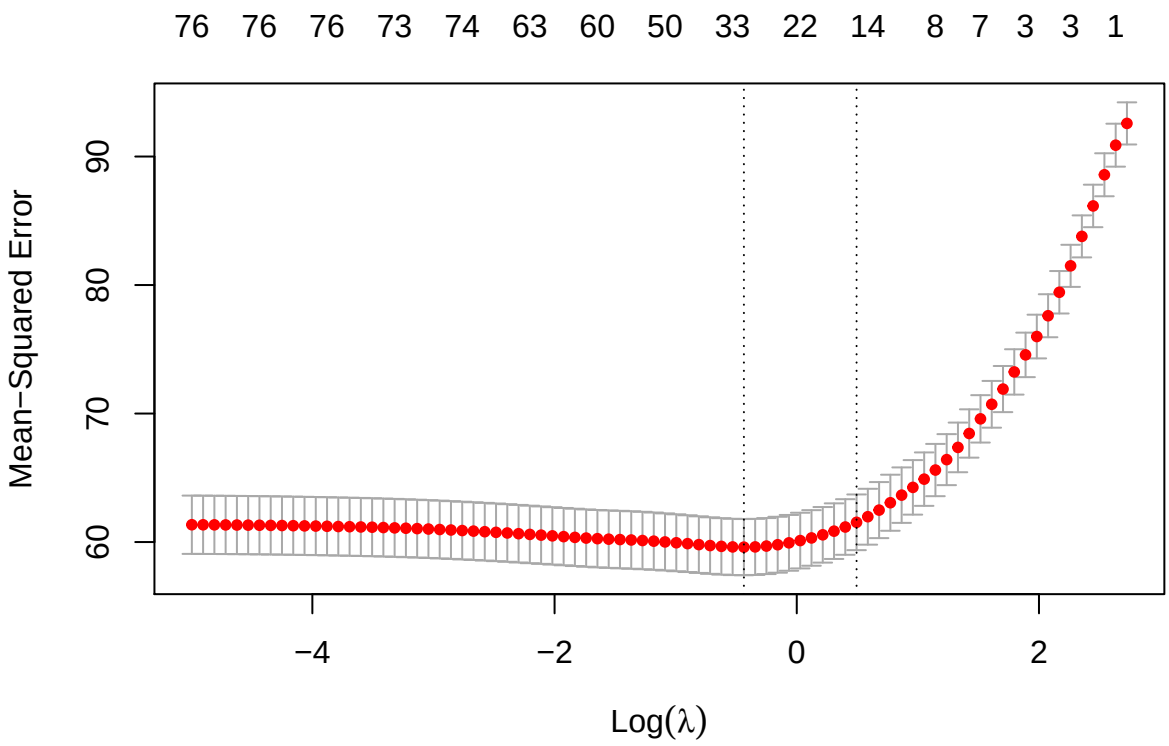
```
fold <- sample(1:10, size = length(g_train_y), replace = TRUE)

t_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(t_grid$alpha)) {
  e_n <- cv.glmnet(g_train_x, g_train_y, alpha =
                     t_grid$alpha[i], foldid = fold)
  plot(e_n)
  t_grid$lambda_min[i] <- e_n$lambda.min
  t_grid$mse_min[i] <- e_n$cvm[e_n$lambda == e_n$lambda.min]
```

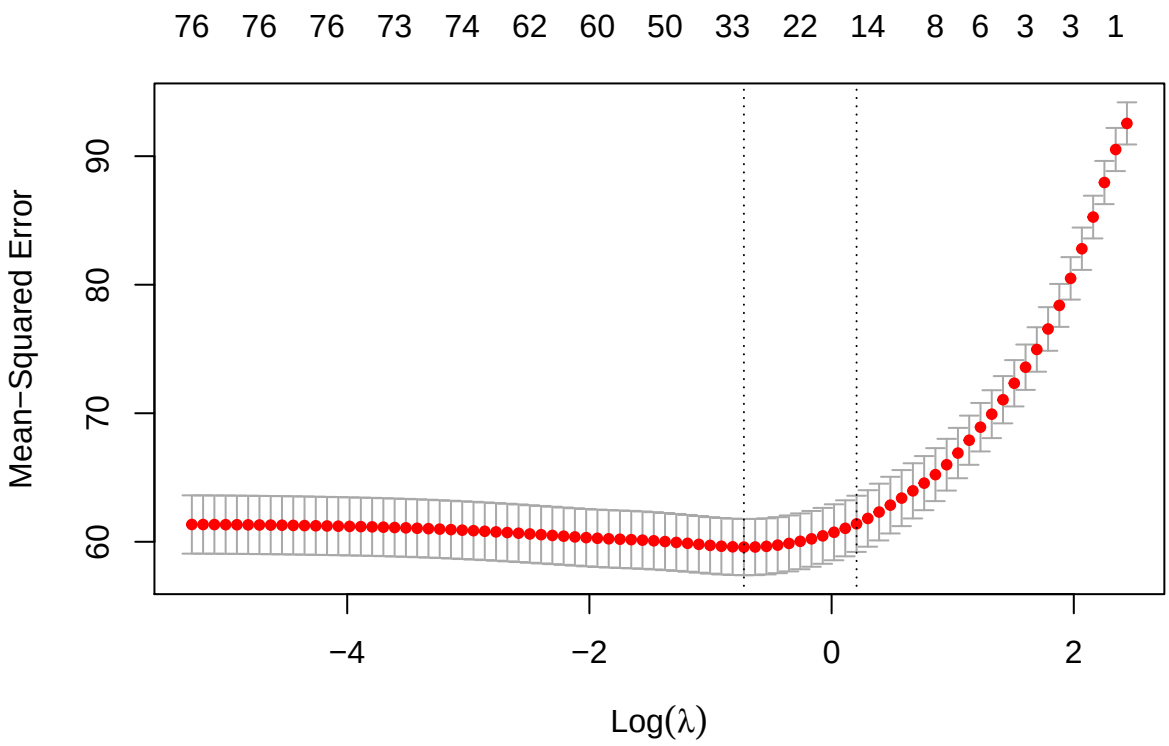
```
  t_grid$mse_1se[i] <- e_n$cvm[e_n$lambda == e_n$lambda.1se]
  t_grid$lambda_1se[i] <- e_n$lambda.1se}
```

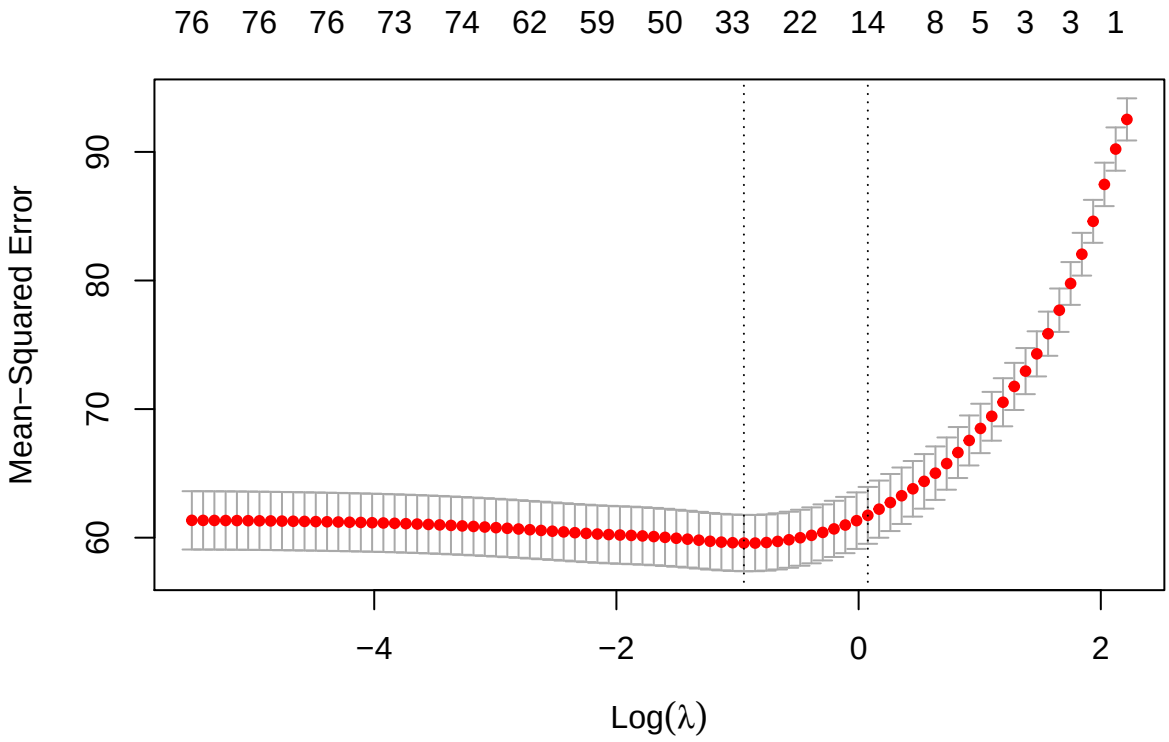


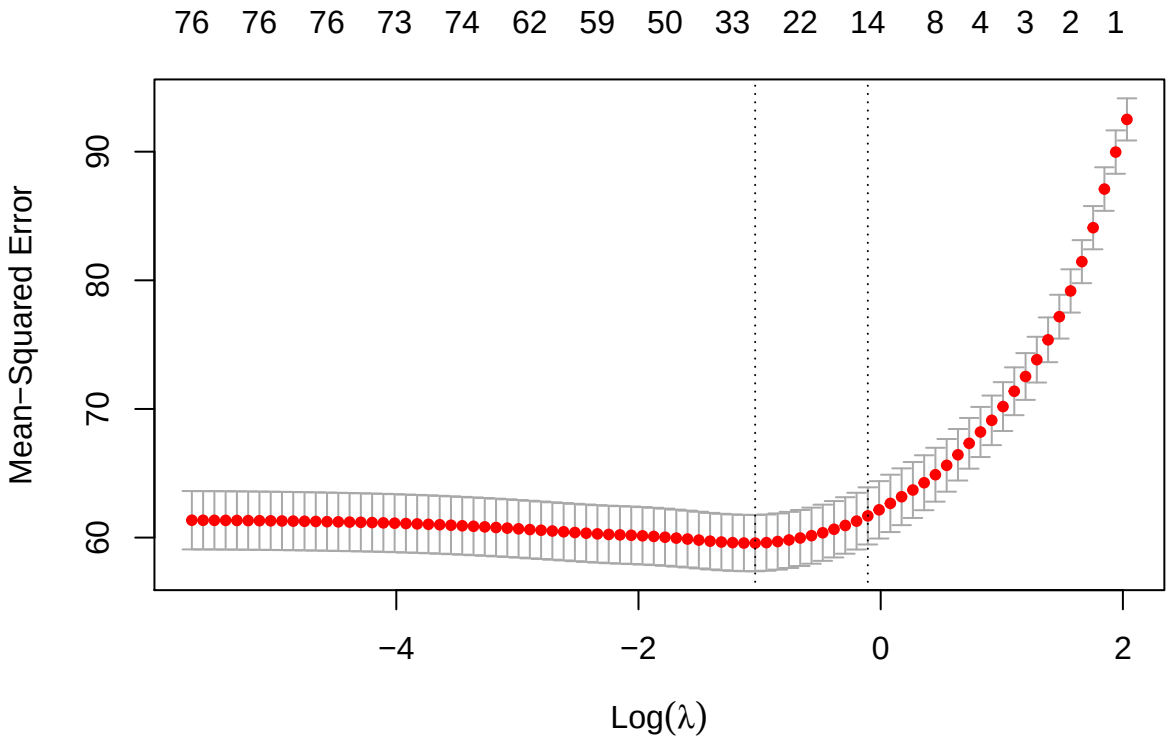

76 76 77 73 75 65 61 54 38 29 17 13 8 3 3
Mean−Squared Error
60
70
80
90
−4
−2
0
2
4
Log(λ)

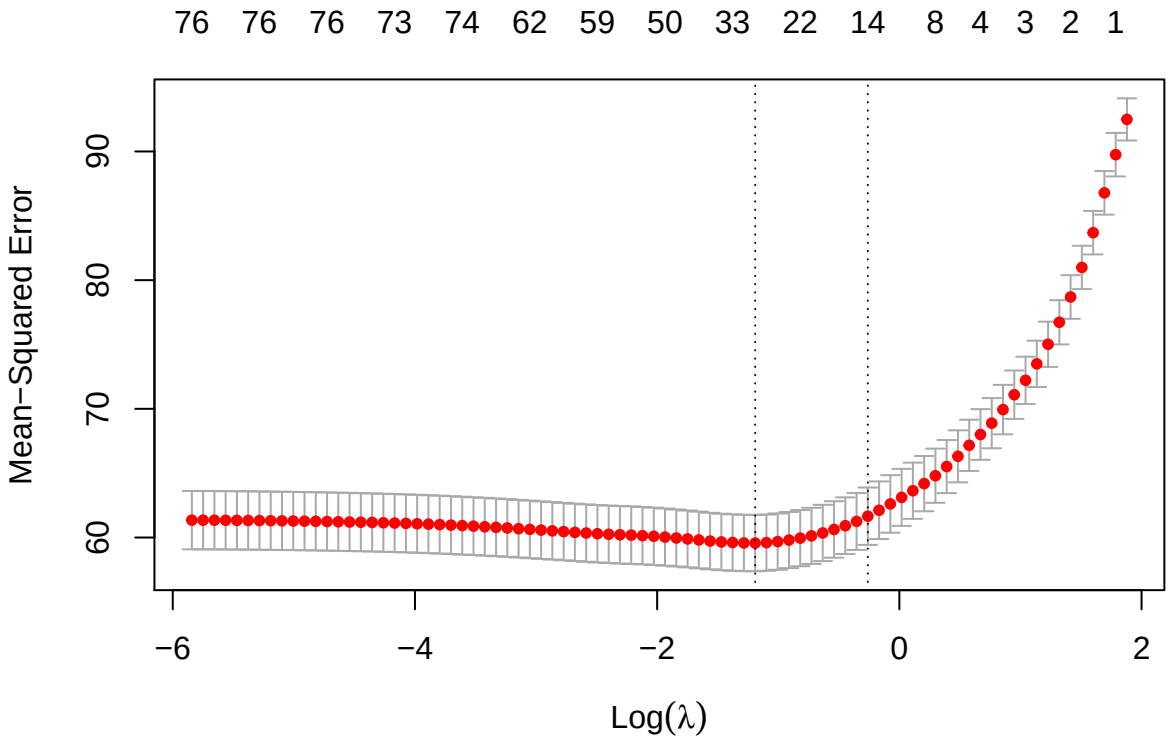

76 76 76 73 75 64 61 51 36 26 15 11 8 3 3 1
Mean−Squared Error
60
70
80
90
−4
−2
0
2
Log(λ)

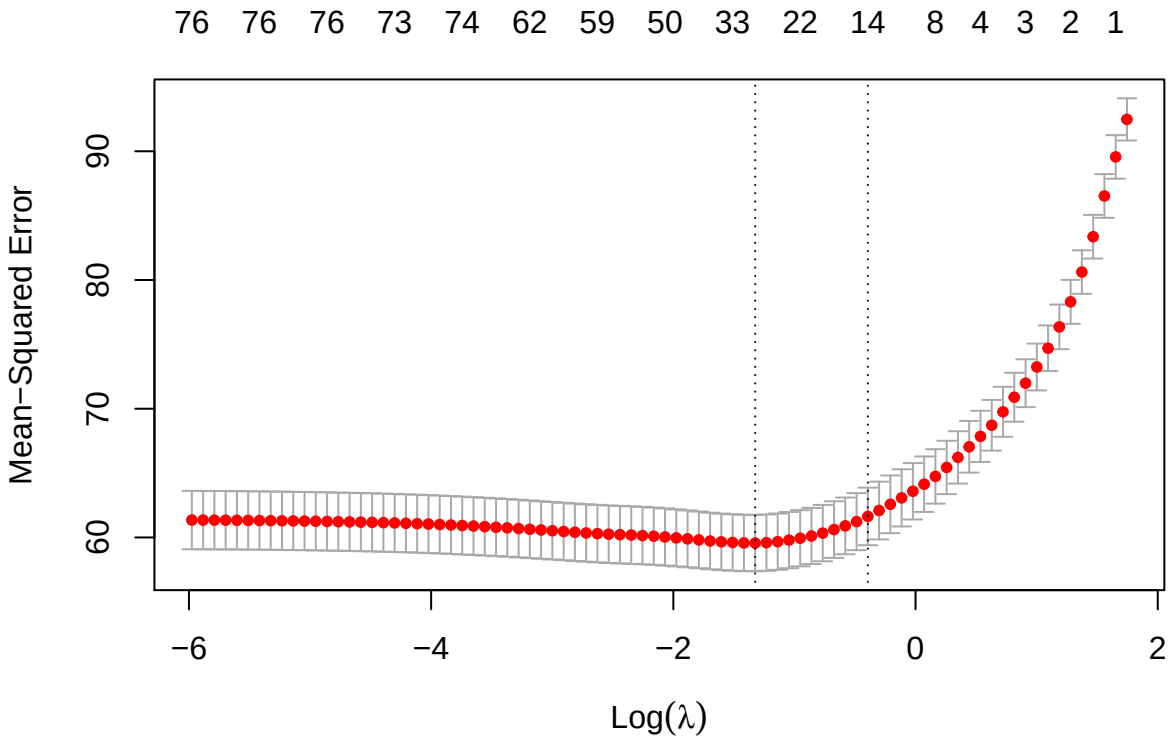

76 76 76 73 74 63 60 50 33 22 14 8 7 3 3 1
Mean−Squared Error
60
70
80
90
−4
−2
0
2
Log(λ)

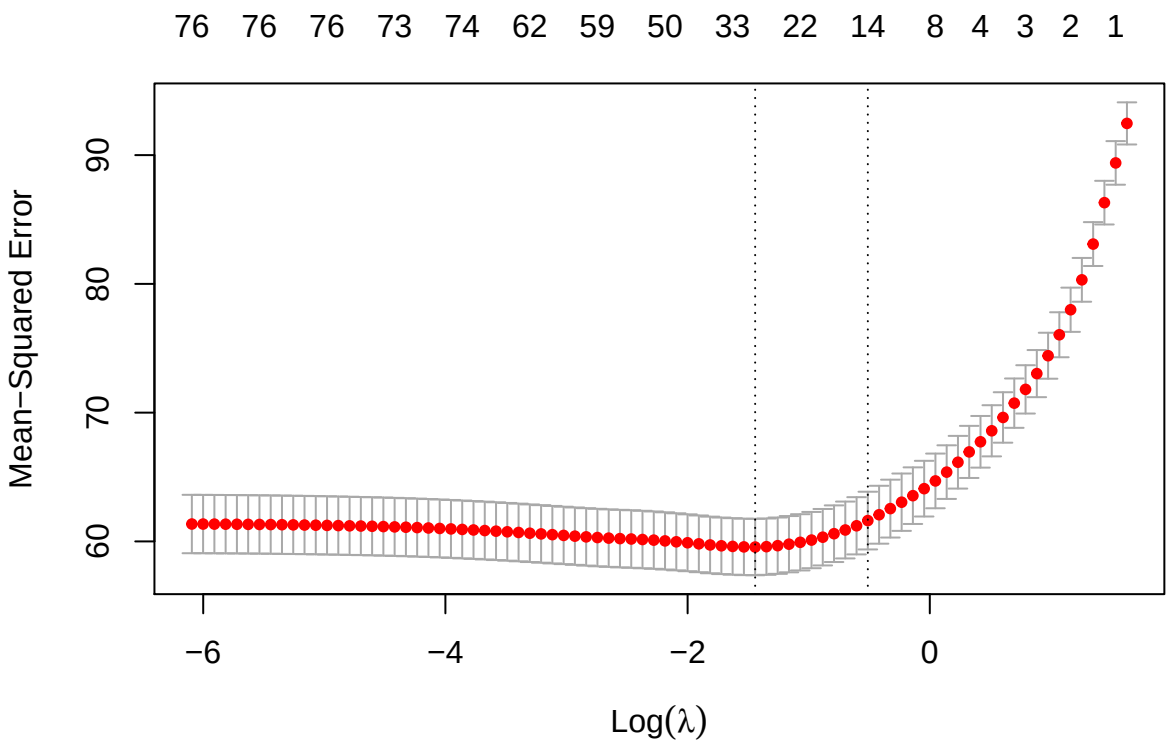

76 76 76 73 74 62 60 50 33 22 14 8 6 3 3 1
Mean−Squared Error
60
70
80
90
−4
−2
0
2
Log(λ)

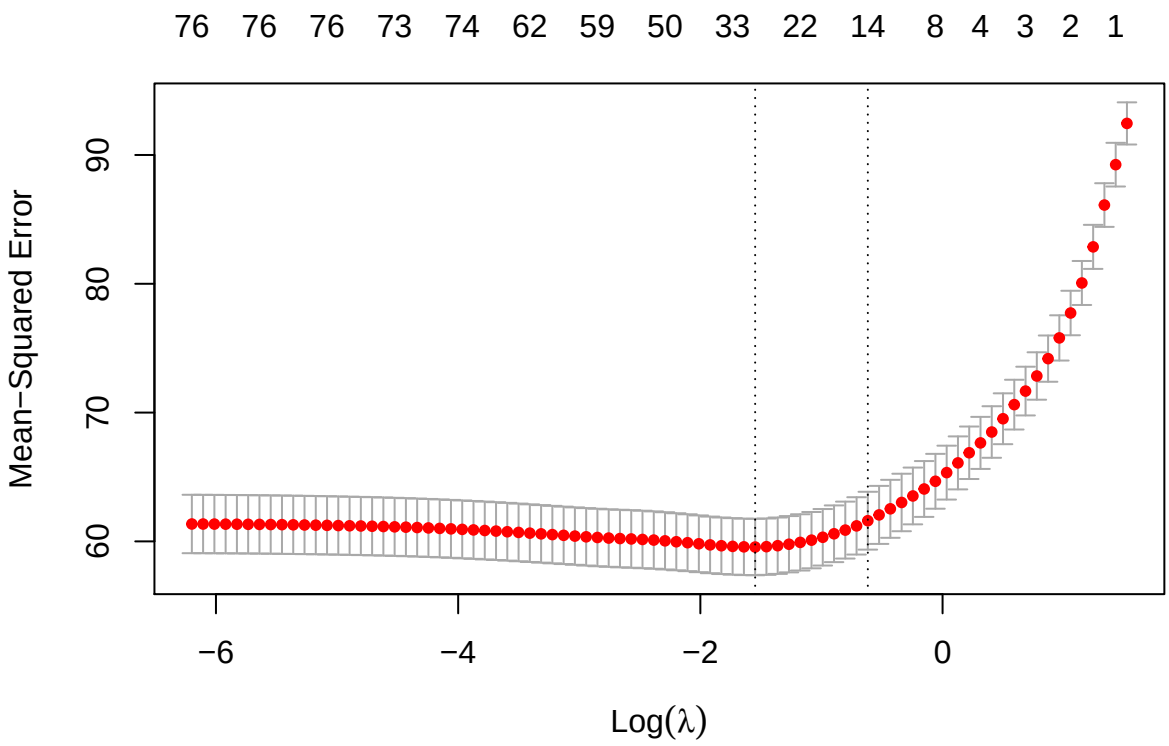

76 76 76 73 74 62 59 50 33 22 14 8 5 3 3 1
Mean−Squared Error
60
70
80
90
−4
−2
0
2
Log(λ)


76 76 76 73 74 62 59 50 33 22 14 8 4 3 2 1
Mean−Squared Error
60
70
80
90
−4
−2
0
2
Log(λ)


76 76 76 73 74 62 59 50 33 22 14 8 4 3 2 1
Mean−Squared Error
60
70
80
90
−6
−4
−2
0
2
Log(λ)


76 76 76 73 74 62 59 50 33 22 14 8 4 3 2 1
Mean−Squared Error
60
70
80
90
−6
−4
−2
0
2
Log(λ)

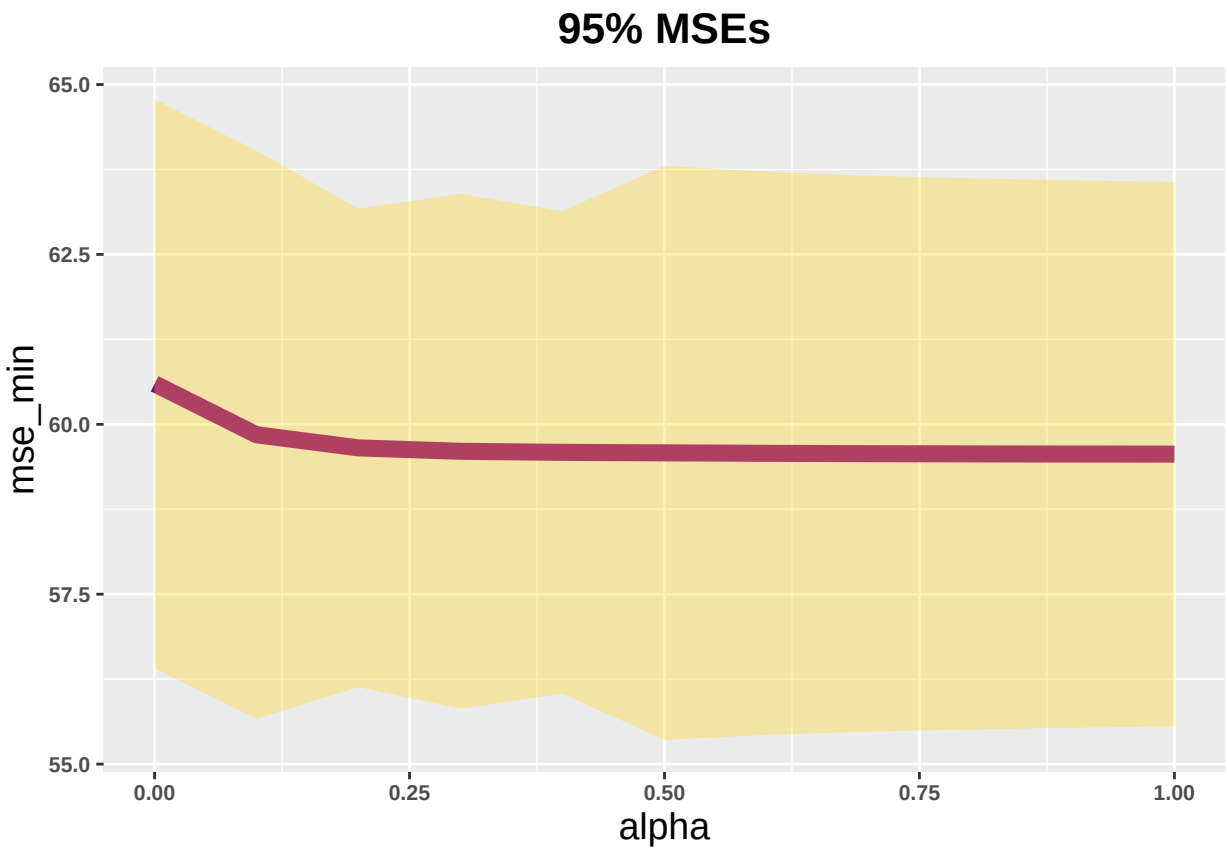

76 76 76 73 74 62 59 50 33 22 14 8 4 3 2 1
Mean−Squared Error
60
70
80
90
−6
−4
−2
0
Log(λ)




```
#alpha and lambda
filter(t_grid, mse_min == min(mse_min))
```

```
## # A tibble: 1 x 5
##   alpha mse_min mse_1se lambda_min lambda_1se
##   <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
## 1     1    59.6    61.6      0.213      0.539
```

```
t_grid %>% mutate(error_here = mse_1se - mse_min) %>%
  ggplot(aes(alpha, mse_min)) +
  geom_line(size = 3, color = 'darkmagenta') +
  geom_ribbon(aes(ymax = mse_min + 1.96 * error_here,
                  ymin = mse_min - 1.96 * error_here),
              alpha = 0.3, fill = 'gold') +
  ggtitle("95% MSEs") +
  theme(plot.title = element_text(hjust = 0.5,
                                  size = 16,face= "bold"),
        axis.text=element_text(size=8, face = "bold"),
        axis.title.x=element_text(size=14),
        axis.title.y=element_text(size=14))
```



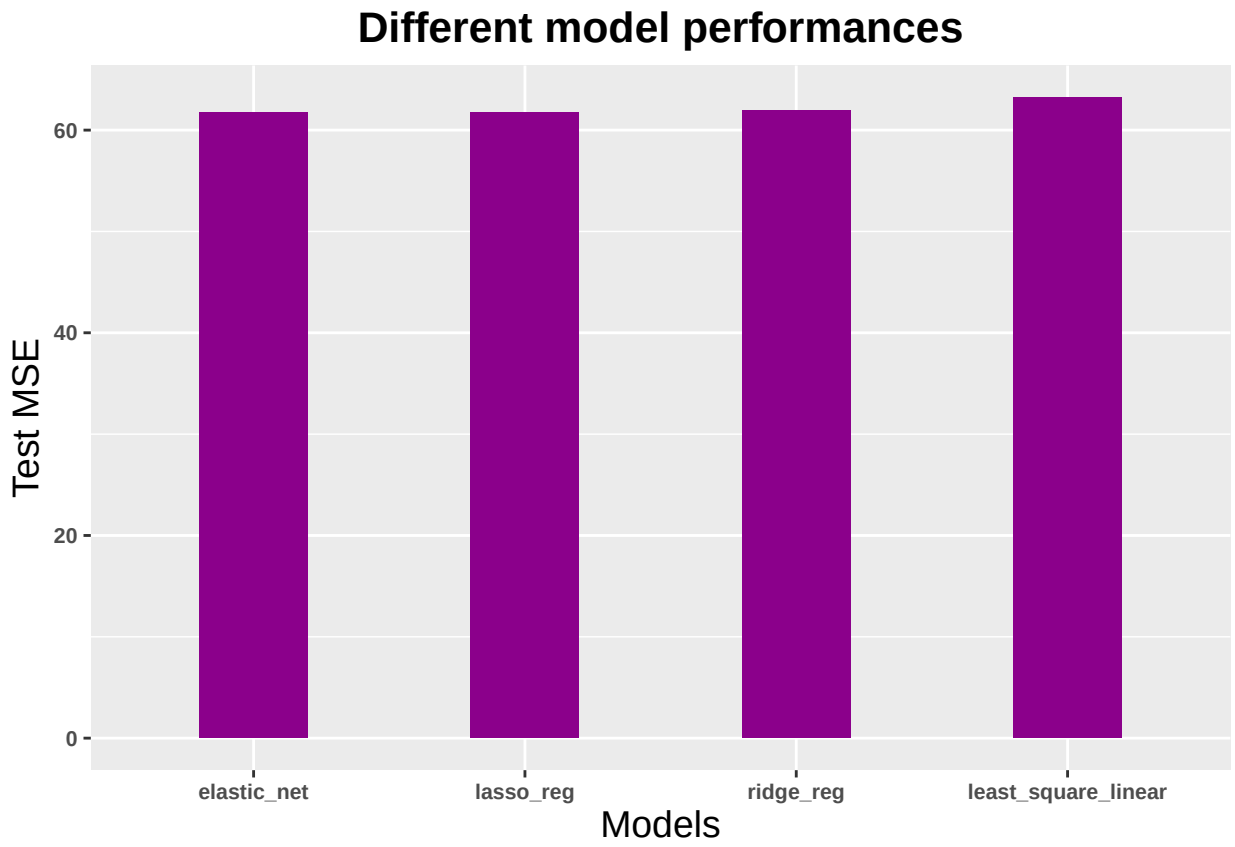

```
en_mse <- g_test %>%
  mutate(preds = predict(lasso_min, newx = g_test_x,
                         s = lasso$lambda.1se)[,1]) %>%
    mse(truth = egalit_scale, estimate = preds)

#MSE
en_mse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        61.7
```

5 According to the figures seperatively of different methods, we can find that the interval for the MSEs are relatively small, which means that these models can be used to analyze this dataset with relatively high accuracy.

```
list(least_square_linear = lsl_mse, ridge_reg = ridge_mse,
     lasso_reg = lasso_mse,
     elastic_net = en_mse) %>%
  bind_rows(.id = "model") %>%
  ggplot(aes(fct_reorder(model, .estimate), .estimate)) +
  geom_col(fill='darkmagenta', width =0.4) +
  labs(title = "Different model performances",
       x = "Models", y = "Test MSE") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face= "bold"),
```

```
        axis.text=element_text(size=8, face = "bold"),
        axis.title.x=element_text(size=14),
        axis.title.y=element_text(size=14))
```



According to this figure, we can see that for this dataset, these 4 methods can yields very similar test MSEs, which means that there is not very much different among the test errors from these approaches.