P8106-hw1

Renjie Wei rw2844

2/15/2022

Contents


```
library(ISLR)
library(glmnet)
library(caret)
library(corrplot)
library(plotmo)
library(tidyverse)
library(pls)
```

Create train and test data

```
# tried to do some manipulations on the data but it turned out to be redundant
train_data <-
  read.csv(file = "housing_training.csv") %>%
  janitor::clean_names()

test_data <- read.csv(file = "housing_test.csv") %>%
  janitor::clean_names()

train_data <- na.omit(train_data)
x_train <- model.matrix(sale_price ~., train_data)[,-1]
y_train <- train_data$sale_price

test_data <- na.omit(test_data)
x_test <- model.matrix(sale_price ~., test_data)[,-1]
y_test <- test_data$sale_price
#corrplot(cor(x_train), type = "full")
```

```r
# Using the default cross-validation as my train control method
myCtrl = trainControl(method = "cv")
```

Problem a

I fit a multiple linear regression using `lm.fit` fucntion.

```r
set.seed(2022)
lm.fit = train(
  x = x_train,
  y = y_train,
  method = "lm",
  trControl = myCtrl
)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##    Min      1Q Median     3Q    Max
## -89864 -12424    416  12143 140205
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -4.985e+06  3.035e+06  -1.642  0.10076
## gr_liv_area                 2.458e+01  1.393e+01   1.765  0.07778 .
## first_flr_sf                4.252e+01  1.409e+01   3.017  0.00260 **
## second_flr_sf               4.177e+01  1.379e+01   3.029  0.00250 **
## total_bsmt_sf               3.519e+01  2.744e+00  12.827  < 2e-16 ***
## low_qual_fin_sf                    NA         NA      NA       NA
## wood_deck_sf                1.202e+01  4.861e+00   2.474  0.01350 *
## open_porch_sf               1.618e+01  1.004e+01   1.611  0.10736
## bsmt_unf_sf                -2.087e+01  1.723e+00 -12.116  < 2e-16 ***
## mas_vnr_area                1.046e+01  4.229e+00   2.473  0.01353 *
## garage_cars                 4.229e+03  1.893e+03   2.234  0.02563 *
## garage_area                 7.769e+00  6.497e+00   1.196  0.23195
## year_built                  3.251e+02  3.130e+01  10.388  < 2e-16 ***
## tot_rms_abv_grd            -3.838e+03  6.922e+02  -5.545 3.51e-08 ***
## full_bath                  -4.341e+03  1.655e+03  -2.622  0.00883 **
## overall_qualAverage        -5.013e+03  1.735e+03  -2.890  0.00391 **
## overall_qualBelow_Average  -1.280e+04  2.677e+03  -4.782 1.92e-06 ***
## overall_qualExcellent       7.261e+04  5.381e+03  13.494  < 2e-16 ***
## overall_qualFair           -1.115e+04  5.240e+03  -2.127  0.03356 *
## overall_qualGood            1.226e+04  1.950e+03   6.287 4.30e-10 ***
## overall_qualVery_Excellent  1.304e+05  8.803e+03  14.810  < 2e-16 ***
## overall_qualVery_Good       3.798e+04  2.741e+03  13.852  < 2e-16 ***
## kitchen_qualFair           -2.663e+04  6.325e+03  -4.210 2.71e-05 ***
## kitchen_qualGood           -1.879e+04  4.100e+03  -4.582 5.01e-06 ***
## kitchen_qualTypical        -2.677e+04  4.281e+03  -6.252 5.37e-10 ***
## fireplaces                  1.138e+04  2.257e+03   5.043 5.18e-07 ***
```

```
## fireplace_quFair           -7.207e+03  6.823e+03  -1.056  0.29106
## fireplace_quGood            6.070e+02  5.833e+03   0.104  0.91713
## fireplace_quNo_Fireplace    3.394e+03  6.298e+03   0.539  0.59002
## fireplace_quPoor           -5.185e+03  7.399e+03  -0.701  0.48362
## fireplace_quTypical        -6.398e+03  5.897e+03  -1.085  0.27814
## exter_qualFair             -3.854e+04  8.383e+03  -4.598 4.66e-06 ***
## exter_qualGood             -1.994e+04  5.585e+03  -3.569  0.00037 ***
## exter_qualTypical          -2.436e+04  5.874e+03  -4.147 3.57e-05 ***
## lot_frontage                1.024e+02  1.905e+01   5.376 8.90e-08 ***
## lot_area                    6.042e-01  7.864e-02   7.683 2.91e-14 ***
## longitude                  -3.481e+04  2.537e+04  -1.372  0.17016
## latitude                    5.874e+04  3.483e+04   1.686  0.09193 .
## misc_val                    9.171e-01  1.003e+00   0.914  0.36071
## year_sold                  -6.455e+02  4.606e+02  -1.401  0.16132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22190 on 1401 degrees of freedom
## Multiple R-squared:  0.9116, Adjusted R-squared:  0.9092
## F-statistic: 380.3 on 38 and 1401 DF,  p-value: < 2.2e-16
```

```
lm.predict <- predict(lm.fit, newdata = x_test)
lm.mse = mean((y_test - lm.predict)^2)
```

Potential disadvantages:

- First, collinearity will cause problem. Which means if there are strong correlations among predictors, the variance of coefficients tends to increase.
- Second, the linear regression methods are sensitive to outliers.
- Third, if the true relationships between X and Y are non-linear, the linear model cannot well performance.
- In this special case, we are including many non-informative variables, as we can see a lot of predictors with insignificant coefficients. Although we may got BLUE estimators, but all of them may perform poorly.

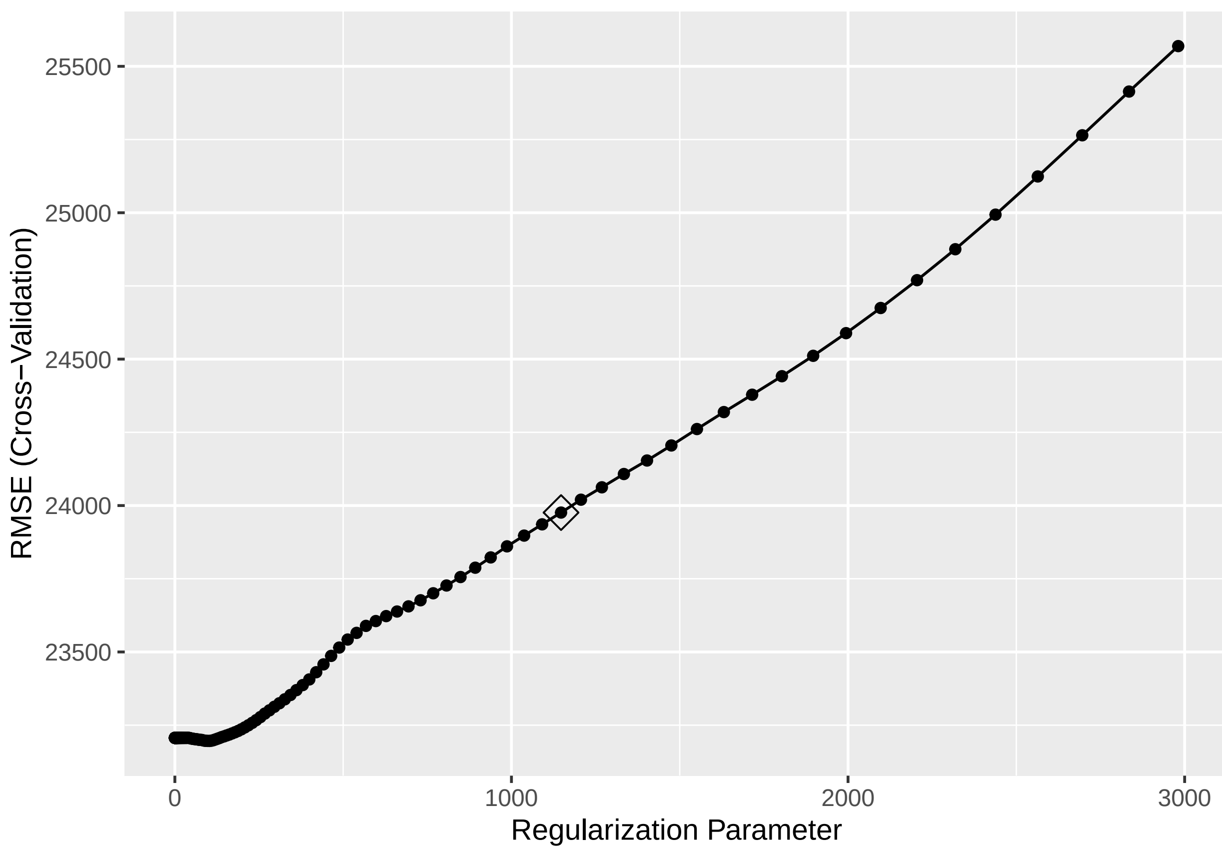
Problem b

I fit the lasso model using `caret`

```
set.seed(2022)
myCtrl_1se = trainControl(
  method = "cv",selectionFunction = "oneSE"
)
lasso.1se.fit <- train(
  x = x_train,
  y = y_train,
  method = "glmnet",
  tuneGrid = expand.grid(
    alpha = 1,
    lambda = exp(seq(8, -2, length = 200))
  ),
  trControl = myCtrl_1se
```

```
)
# this is for model comparsion
lasso.min.fit <- train(
  x = x_train,
  y = y_train,
  method = "glmnet",
  tuneGrid = expand.grid(
    alpha = 1,
    lambda = exp(seq(8, -2, length = 200))
  ),
  trControl = myCtrl
)

# visualization of the 1SE rule
ggplot(lasso.1se.fit, log = "x", highlight = T)
```




```
# coefficients matrix under 1SE rule
coef(lasso.1se.fit$finalModel, lasso.1se.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                 -8.069643e+05
## gr_liv_area                  5.403348e+01
## first_flr_sf                 1.280771e+00
## second_flr_sf                .
```

```
## total_bsmt_sf               3.670151e+01
## low_qual_fin_sf            -1.999235e+01
## wood_deck_sf                7.665041e+00
## open_porch_sf               5.801894e+00
## bsmt_unf_sf                -1.817782e+01
## mas_vnr_area                1.455491e+01
## garage_cars                 3.122120e+03
## garage_area                 1.206603e+01
## year_built                  3.227079e+02
## tot_rms_abv_grd            -2.398395e+02
## full_bath                   .
## overall_qualAverage        -2.614395e+03
## overall_qualBelow_Average  -7.967437e+03
## overall_qualExcellent       8.774444e+04
## overall_qualFair           -4.203659e+03
## overall_qualGood            8.727015e+03
## overall_qualVery_Excellent  1.563313e+05
## overall_qualVery_Good       3.463349e+04
## kitchen_qualFair           -3.414067e+03
## kitchen_qualGood            .
## kitchen_qualTypical        -9.917674e+03
## fireplaces                  6.952074e+03
## fireplace_quFair            .
## fireplace_quGood            4.080413e+03
## fireplace_quNo_Fireplace    .
## fireplace_quPoor            .
## fireplace_quTypical         .
## exter_qualFair             -1.292342e+04
## exter_qualGood              .
## exter_qualTypical          -5.597141e+03
## lot_frontage                5.684125e+01
## lot_area                    5.246052e-01
## longitude                  -1.258794e+03
## latitude                    2.299613e+03
## misc_val                    .
## year_sold                   .
```

```r
lasso.1se.fit$bestTune
```

```
##     alpha   lambda
## 181     1 1147.368
```

```r
lasso.1se.predict <- predict(lasso.1se.fit,newdata = x_test)
lasso.1se.mse <- mean((y_test - lasso.1se.predict)^2)
# for model comparsion only
lasso.min.predict <- predict(lasso.min.fit,newdata = x_test)
lasso.min.mse <- mean((y_test - lasso.min.predict)^2)
```

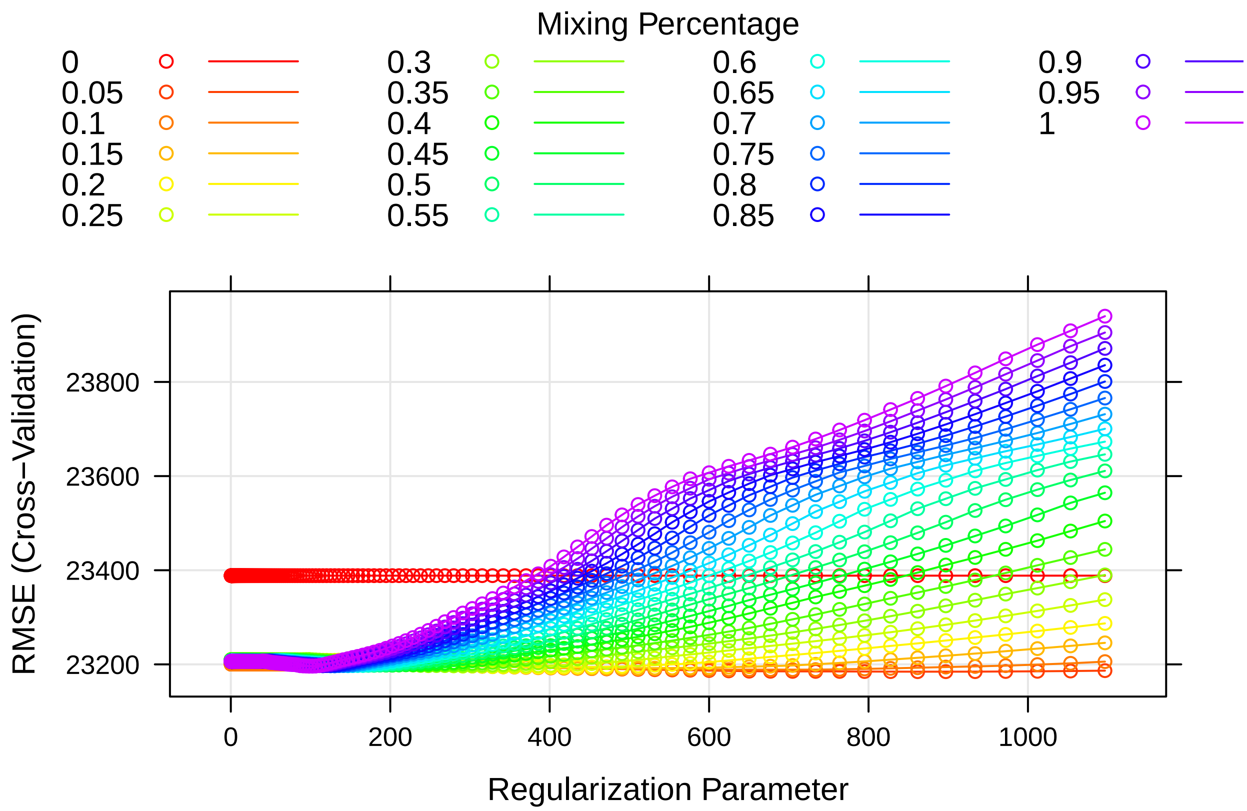
From the 40 X 1 sparse Matrix given by `coef(lasso.1se.fit$finalModel, lasso.1se.fit$bestTune$lambda)`, we can see that under the 1SE rule, 30 out of a total 40 predictors are included in the model.

Problem c

Same as (b), I fitted the elastic-net model using `caret`. The `tuneGrid` is designed by the following form after some tries on searching the ideal tuning parameter intervals.

```
set.seed(2022)
enet.fit <- train(
  x = x_train,
  y = y_train,
  method = "glmnet",
  tuneGrid = expand.grid(
    alpha = seq(0, 1, length = 21),
    lambda = exp(seq(7, -1, length=200))
  ),
  trControl = myCtrl
)

myCol<- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
                    superpose.line = list(col = myCol))
plot(enet.fit, par.settings = myPar)
```




```
enet.fit$bestTune$alpha
```

```
## [1] 0.05
```

```r
enet.fit$bestTune$lambda
```

```
## [1] 896.9454
```

```r
enet.predict <- predict(enet.fit,newdata = x_test)
enet.mse <- mean((y_test - enet.predict)^2)
```

Based on the fitted model, the selected tuning parameters are α=0.05 and λ=896.9453859. And the test error of this model is 4.3528674×10^8.

Problem c

Since there are some issue when fitting a partial least square model using `caret`, I switched to the `pls` package to fit my PLS model.

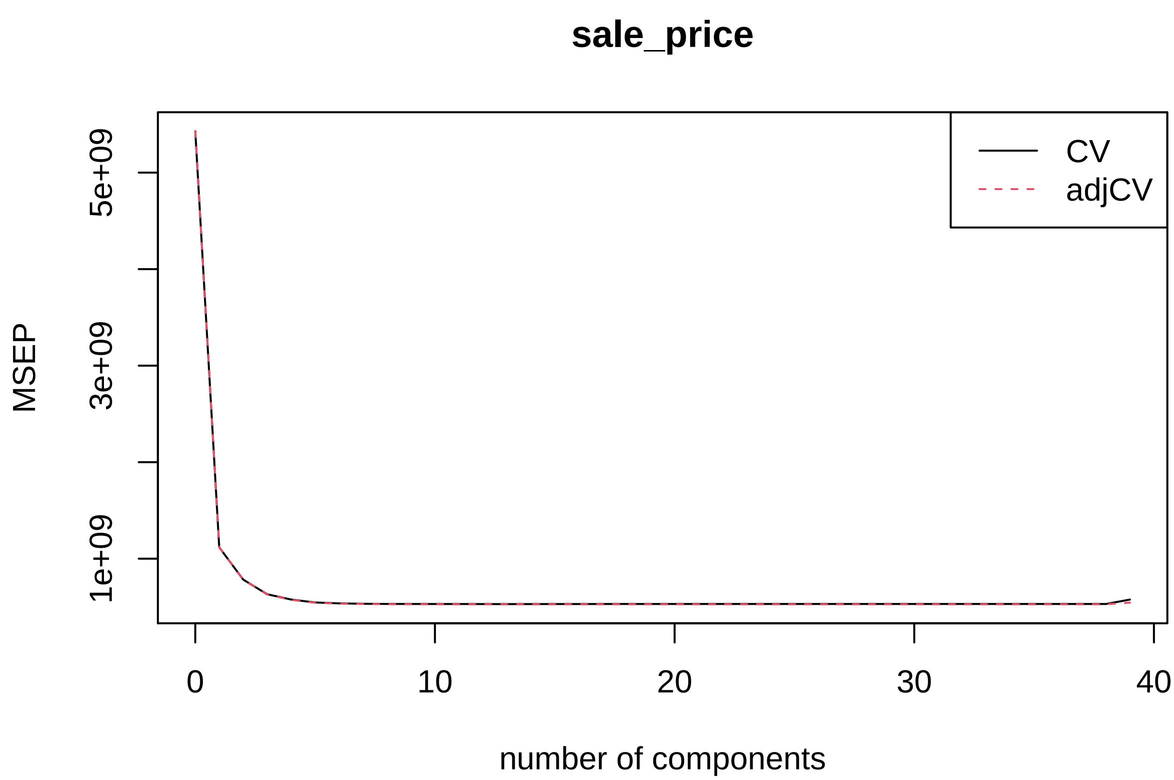
```r
set.seed(2022)
pls.fit <- plsr(sale_price~.,
                data = train_data,
                scale = TRUE,
                validation = "CV")

summary(pls.fit)
```

```
## Data:    X dimension: 1440 39
##  Y dimension: 1440 1
## Fit method: kernelpls
## Number of components considered: 39
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           73685    33432    27986    25125    24011    23369    23171
## adjCV        73685    33426    27949    25054    23942    23303    23113
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       23078    23036    23033     23027     23027     23014     23009
## adjCV    23022    22982    22977     22969     22967     22955     22950
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        23013     23020     23021     23025     23030     23027     23032
## adjCV     22954     22960     22961     22965     22969     22966     22971
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        23032     23032     23032     23032     23032     23032     23034
## adjCV     22971     22971     22971     22971     22971     22971     22973
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV        23034     23034     23034     23034     23034     23034     23034
## adjCV     22973     22973     22973     22973     22973     22973     22973
##        35 comps  36 comps  37 comps  38 comps  39 comps
## CV        23034     23034     23034     23034     24005
## adjCV     22973     22973     22973     22973     23330
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
```

```
## X               20.02    25.93    29.67    33.59    37.01    40.03    42.49
## sale_price      79.73    86.35    89.36    90.37    90.87    90.99    91.06
##              8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X               45.53    47.97     50.15     52.01     53.69     55.35     56.86
## sale_price      91.08    91.10     91.13     91.15     91.15     91.16     91.16
##              15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## X                58.64     60.01     62.18     63.87     65.26     67.10
## sale_price       91.16     91.16     91.16     91.16     91.16     91.16
##              21 comps  22 comps  23 comps  24 comps  25 comps  26 comps
## X                68.44     70.12     71.72     73.35     75.20     77.27
## sale_price       91.16     91.16     91.16     91.16     91.16     91.16
##              27 comps  28 comps  29 comps  30 comps  31 comps  32 comps
## X                78.97     80.10     81.83     83.55     84.39     86.34
## sale_price       91.16     91.16     91.16     91.16     91.16     91.16
##              33 comps  34 comps  35 comps  36 comps  37 comps  38 comps
## X                88.63     90.79     92.79     95.45     97.49    100.00
## sale_price       91.16     91.16     91.16     91.16     91.16     91.16
##              39 comps
## X               100.67
## sale_price       91.16
```

```
validationplot(pls.fit, val.type="MSEP", legendpos = "topright")
```




```
cv.mse <- RMSEP(pls.fit)
ncomp.cv <- which.min(cv.mse$val[1,,])-1
```

```
pls.predict <- predict(pls.fit, newdata = test_data, ncomp = ncomp.cv)
pls.mse = mean((y_test - pls.predict)^2)
```

The model result shows that there are 13 components included in this model.

Problem e

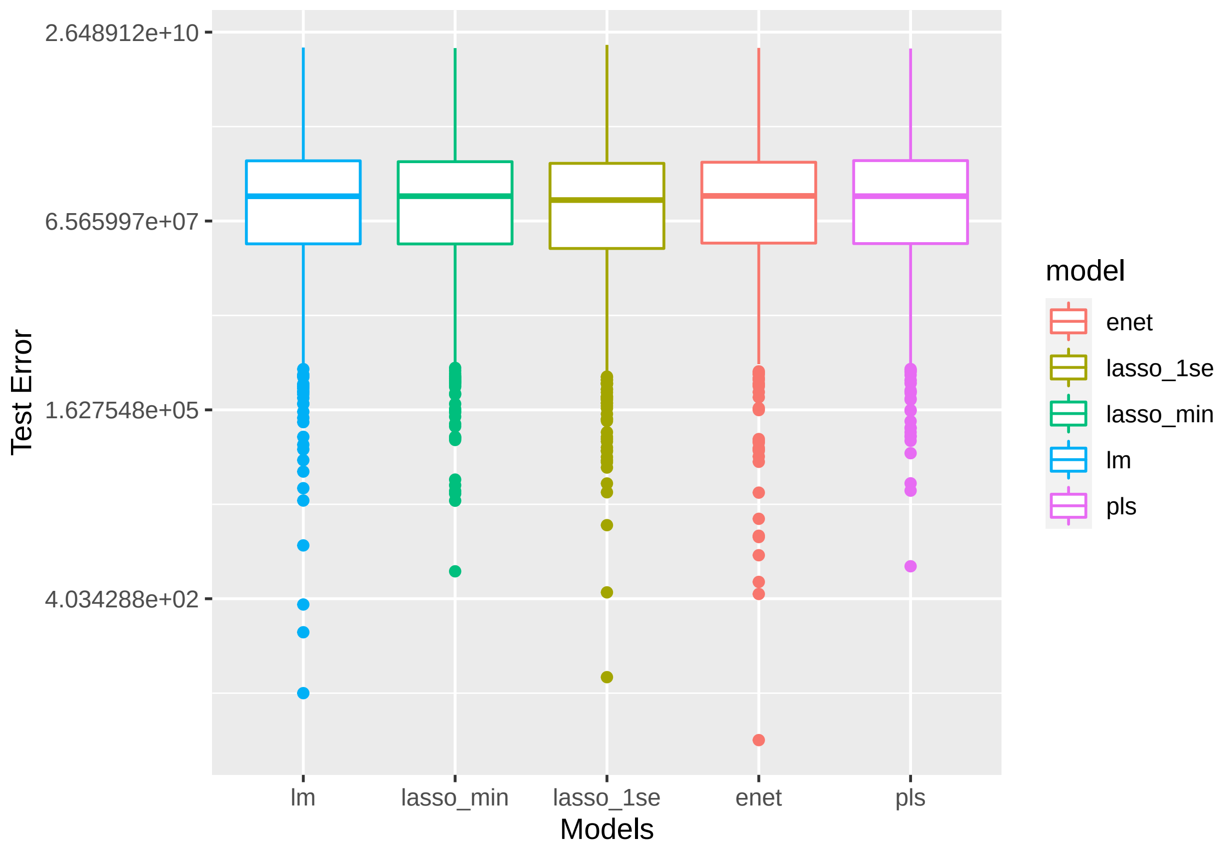
Since I use the same seed with `set.seed(2022)` and using the same resampling method in estimating test errors in each model, which is the cross-validation, I just summarized these models' performance on the test data using MSE.

From the box plot, we can see that the `lasso_1se` model has the lowest mean test error.

```
lm_test_err = (lm.predict-y_test)^2
test_len = length(lm_test_err)
lasso_min_test_err = (lasso.min.predict-y_test)^2
#length(lasso_min_test_err)
lasso_1se_test_err = (lasso.1se.predict-y_test)^2
#length(lasso_1se_test_err)
enet_test_err = (enet.predict-y_test)^2
#length(enet_test_err)
pls_test_err = (pls.predict-y_test)^2
#length(pls_test_err)


summary_tab <-
  tibble(
    error = c(lm_test_err, lasso_min_test_err, lasso_1se_test_err, enet_test_err, pls_test_err),
    model = c(rep("lm", test_len),rep("lasso_min",test_len),rep("lasso_1se",test_len),rep("enet",test_le
  )

ggplot(data = summary_tab,aes(x = factor(model, level = c("lm", "lasso_min", "lasso_1se", "enet", "pls")
  geom_boxplot()+
  scale_y_continuous(trans = "log")+
  xlab("Models")+
  ylab("Test Error")
```



And I summarized some statistics of the test errors of the different models. Same as the plot above, the `lasso_1se` has the lowest test MSE.

```
summary_tab %>%
  group_by(model) %>%
  summarize(
    Min = min(error),
    Q_25 = quantile(error,probs = 0.25),
    Median = median(error),
    MSE = mean(error),
    Q_75 = quantile(error, probs = 0.75),
    Max = max(error)
  ) %>%
  knitr::kable()
```

model	Min	Q_25	Median	MSE	Q_75	Max
enet	4.522061	32537227	145668366	435286735	424471277	16060886842
lasso_1se	33.403000	27451162	127681781	427207463	409968391	17678935336
lasso_min	963.929764	31694006	144575401	441167159	431961394	15969512194
lm	20.101168	31772015	144147390	447287652	444513622	16246868321
pls	1131.057696	32035823	144529209	448737340	447444835	15748741707

As a conclusion, considering the test MSE, I would like to choose the LASSO model using the 1SE rule for predicting the response.