p8106_hw1

Hao Zheng

2/20/2022

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

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

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.2
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.2

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

## Loaded glmnet 4.1-3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

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

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.2

## corrplot 0.92 loaded
```

```
library(plotmo)
```

```
## Warning: package 'plotmo' was built under R version 4.1.2

## Loading required package: Formula

## Loading required package: plotrix

## Loading required package: TeachingDemos

## Warning: package 'TeachingDemos' was built under R version 4.1.2
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.1.2

##
## Attaching package: 'pls'

## The following object is masked from 'package:corrplot':
##
##     corrplot

## The following object is masked from 'package:caret':
##
##     R2

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

```
# Data import
train = read.csv("./data/housing_training.csv") %>% janitor::clean_names()
test = read.csv("./data/housing_test.csv") %>% janitor::clean_names()

train = na.omit(train)
test = na.omit(test)
```

Now, let's fit different models based on the dataset.

Linear model

```
set.seed(2022)

fit.lm = lm(sale_price ~ .,
            data = train,
            method = "lm",
            trControl = trainControl(method = "repeatedcv", number = 10))
```

```
## Warning in lm(sale_price ~ ., data = train, method = "lm", trControl =
## trainControl(method = "repeatedcv", : method = 'lm' is not supported. Using 'qr'
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'trControl' will be disregarded
```

```
summary(fit.lm)
```

```
##
## Call:
## lm(formula = sale_price ~ ., data = train, method = "lm", trControl = trainControl(method = "repeatec
##     number = 10))
##
## 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
```

```r
pred.lm <- predict(fit.lm, newdata = test)
```

```
## Warning in predict.lm(fit.lm, newdata = test): prediction from a rank-deficient
## fit may be misleading
```

```r
RMSE(pred.lm, test$sale_price)
```

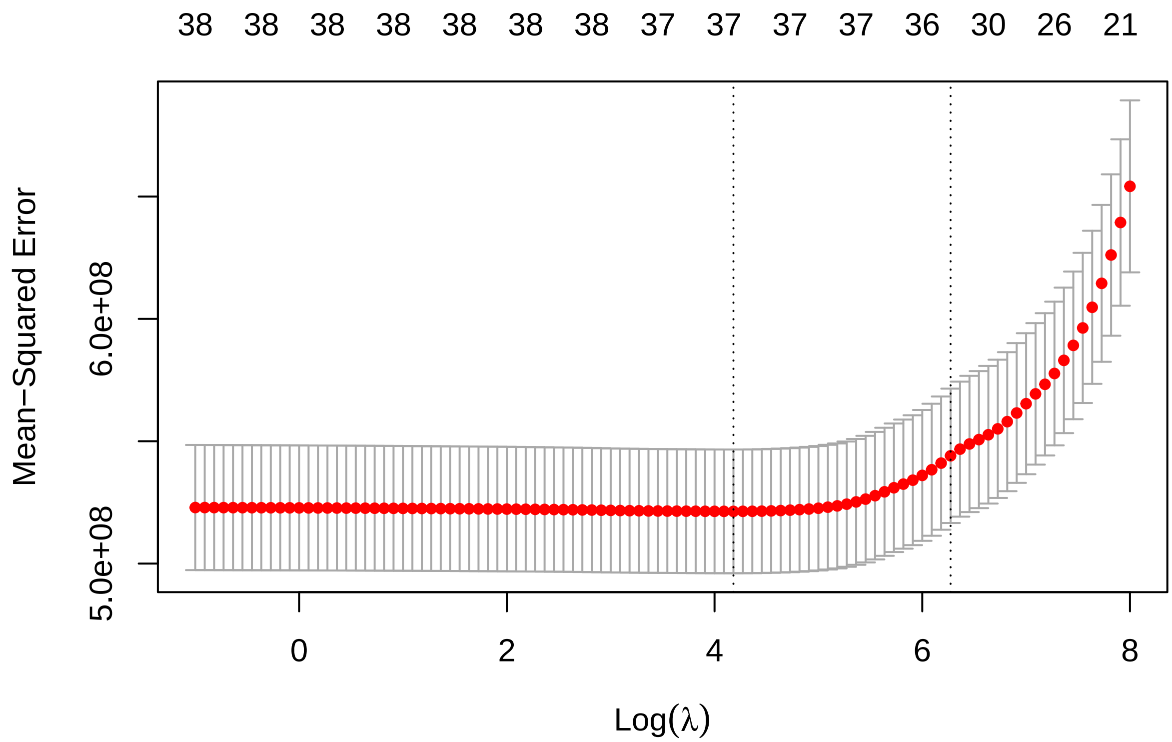
```
## [1] 21149.18
```

Potential advantage of the linear model is there may exist correlation which may disturb the model.

Lasso model

using glmnet

```r
set.seed(2022)

x_train = model.matrix(sale_price~., train)[,-1]
y_train = train$sale_price

cv.lasso <- cv.glmnet(
  x = x_train,
  y = y_train,
  alpha = 1,
  lambda = exp(seq(8, -1, length = 100))
)

plot(cv.lasso)
```




```r
# Look at the 1SE coefficient for lasso
predict(cv.lasso, s = "lambda.min",type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                lambda.min
## (Intercept)                 -4.822183e+06
## gr_liv_area                  6.537622e+01
## first_flr_sf                 8.030905e-01
## second_flr_sf                .
## total_bsmt_sf                3.541795e+01
## low_qual_fin_sf             -4.095780e+01
## wood_deck_sf                 1.163031e+01
## open_porch_sf                1.542701e+01
## bsmt_unf_sf                 -2.088622e+01
## mas_vnr_area                 1.089420e+01
## garage_cars                  4.087854e+03
## garage_area                  8.147291e+00
## year_built                   3.233856e+02
## tot_rms_abv_grd             -3.613282e+03
## full_bath                   -3.842700e+03
## overall_qualAverage         -4.850637e+03
## overall_qualBelow_Average   -1.245461e+04
## overall_qualExcellent        7.549162e+04
## overall_qualFair            -1.075078e+04
## overall_qualGood             1.212626e+04
## overall_qualVery_Excellent   1.357182e+05
```

```
## overall_qualVery_Good       3.789714e+04
## kitchen_qualFair           -2.489147e+04
## kitchen_qualGood           -1.724861e+04
## kitchen_qualTypical        -2.536149e+04
## fireplaces                  1.054325e+04
## fireplace_quFair           -7.659220e+03
## fireplace_quGood            .
## fireplace_quNo_Fireplace    1.441822e+03
## fireplace_quPoor           -5.636221e+03
## fireplace_quTypical        -7.006413e+03
## exter_qualFair             -3.328749e+04
## exter_qualGood             -1.504335e+04
## exter_qualTypical          -1.948004e+04
## lot_frontage                9.963599e+01
## lot_area                    6.043867e-01
## longitude                  -3.289000e+04
## latitude                    5.504015e+04
## misc_val                    8.282364e-01
## year_sold                  -5.600991e+02
```

When the 1SE rule is applied, we can see there are 35 predictors included in the model.

```r
x_test <- model.matrix(sale_price~., test)[ ,-1]
y_test <- test$sale_price

y_pred <- predict(cv.lasso, newx = x_test, s = "lambda.min", type = "response")
lasso_mse <- mean(RMSE(y_pred, y_test)^2); lasso_mse
```

```
## [1] 439963234
```

The test MSE for Lasso model (1SE) is 4.3996323×10^8.

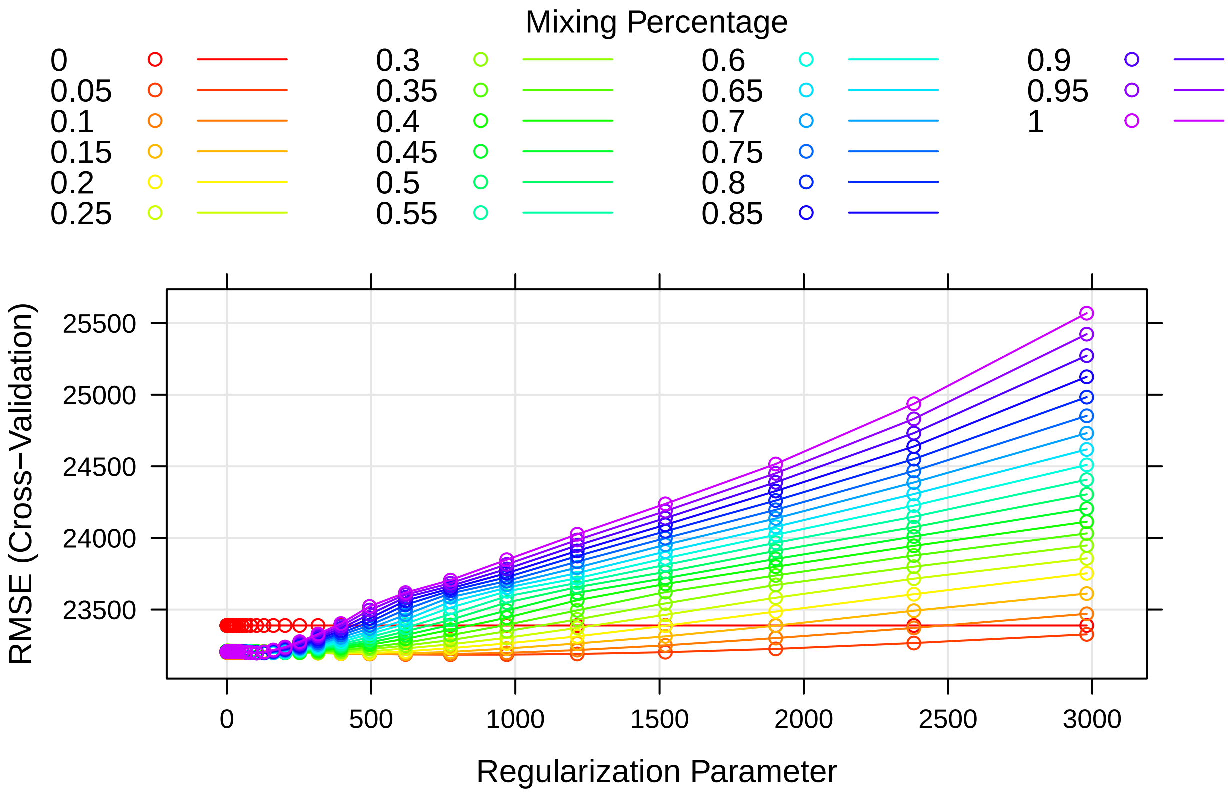
Elastic Net model

```r
set.seed(2022)

enet.fit = train(x_train, y_train,
                 method = "glmnet",
                 tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                        lambda = exp(seq(8, -3, length = 50))),
                 trControl = trainControl(method = "cv", number = 10))

enet.fit$bestTune
```

```
##    alpha   lambda
## 95  0.05 970.2473
```

The selected tuning parameter lambda = 970.247332, alpha = 0.05. Then we visualize the elastic net result.

```r
# Visualization
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(enet.fit, par.settings = myPar)
```




```r
coef(enet.fit$finalModel, enet.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)           -5.172951e+06
## gr_liv_area            3.812434e+01
## first_flr_sf           2.637357e+01
## second_flr_sf          2.480921e+01
## total_bsmt_sf          3.473579e+01
## low_qual_fin_sf       -1.604315e+01
## wood_deck_sf           1.252592e+01
## open_porch_sf          1.727662e+01
## bsmt_unf_sf           -2.057836e+01
## mas_vnr_area           1.235856e+01
## garage_cars            3.958861e+03
## garage_area            9.528576e+00
## year_built             3.152748e+02
## tot_rms_abv_grd       -3.197583e+03
```

```
## full_bath                     -3.313259e+03
## overall_qualAverage           -5.160203e+03
## overall_qualBelow_Average     -1.263204e+04
## overall_qualExcellent          7.695123e+04
## overall_qualFair              -1.165139e+04
## overall_qualGood               1.174884e+04
## overall_qualVery_Excellent     1.386447e+05
## overall_qualVery_Good          3.733755e+04
## kitchen_qualFair              -2.220863e+04
## kitchen_qualGood              -1.475407e+04
## kitchen_qualTypical           -2.285583e+04
## fireplaces                     1.053606e+04
## fireplace_quFair              -8.000449e+03
## fireplace_quGood               6.186069e+01
## fireplace_quNo_Fireplace       1.117703e+03
## fireplace_quPoor              -5.936463e+03
## fireplace_quTypical           -7.046712e+03
## exter_qualFair                -3.077995e+04
## exter_qualGood                -1.239462e+04
## exter_qualTypical             -1.717937e+04
## lot_frontage                   9.877631e+01
## lot_area                       6.017072e-01
## longitude                     -3.534682e+04
## latitude                       5.708440e+04
## misc_val                       8.336667e-01
## year_sold                     -5.374119e+02
```

Next, we calculate the test MSE on our test dataset.

```
# Elastic net test MSE
enet_pred <- predict(enet.fit, newdata = x_test)
enet_mse <- mean(RMSE(enet_pred, y_test)^2); enet_mse
```

```
## [1] 434592441
```

The test error for Elastics net model is 4.3459244×10^8.

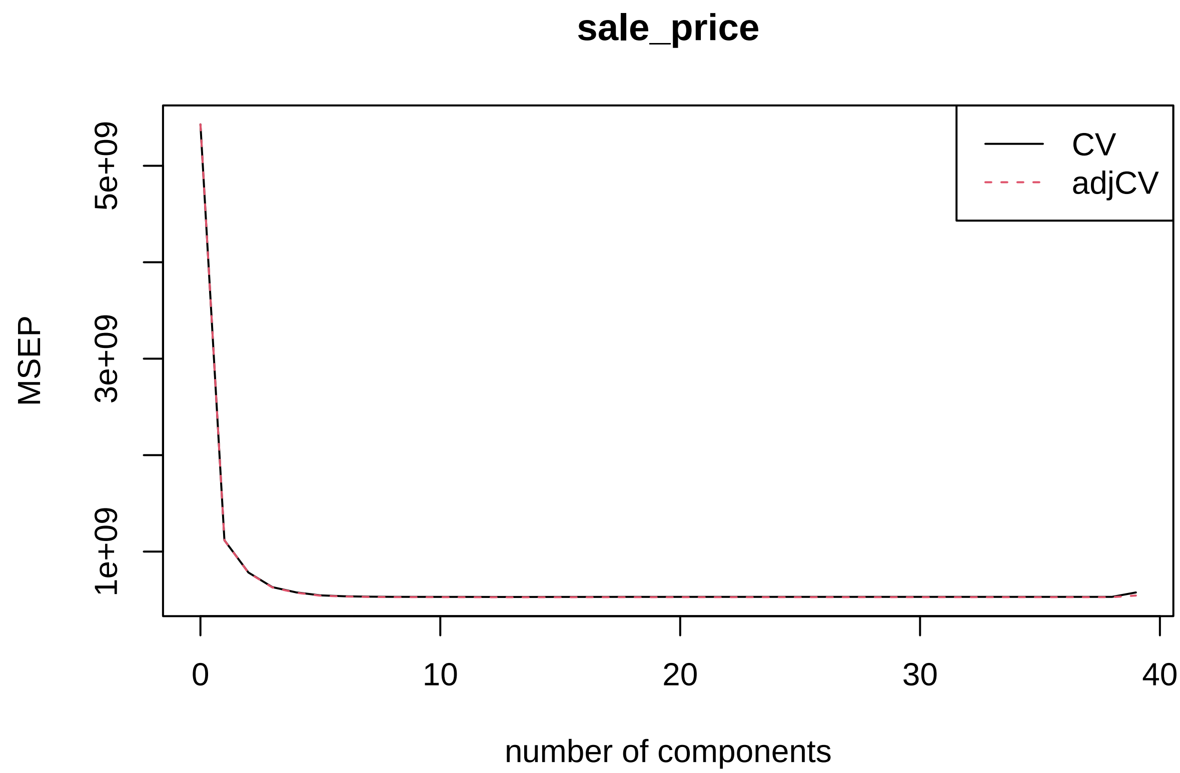
Partial Least Square model

```
set.seed(2022)

pls.fit <- plsr(sale_price~.,
                data = train,
                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")
```




```
# Calculate the number of component in the model
cv.mse <- RMSEP(pls.fit)
ncomp.cv <- which.min(cv.mse$val[1,,]) - 1; ncomp.cv
```

```
## 13 comps
##       13
```

```
pls_pred <- predict(pls.fit, newdata = x_test, ncomp = ncomp.cv)
pls_mse <- mean(RMSE(y_test, pls_pred)^2); pls_mse
```

```
## [1] 448737340
```

There are 13 component in pls model, and the test error (MSE) is 4.4873734×10^8.

Comparing different models