8106hw1

Ze Li zl2746

```r
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tibble       3.2.1
## v dplyr        1.1.3     v tidyr        1.3.0
## v infer        1.0.5     v tune         1.1.2
## v modeldata    1.3.0     v workflows    1.1.3
## v parsnip      1.1.1     v workflowsets 1.0.1
## v purrr        1.0.2     v yardstick    1.3.0
## v recipes      1.0.8

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()         masks scales::discard()
## x tidyr::expand()          masks Matrix::expand()
## x dplyr::filter()          masks stats::filter()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x tidyr::pack()            masks Matrix::pack()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## x tidyr::unpack()          masks Matrix::unpack()
## x recipes::update()        masks Matrix::update(), stats::update()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

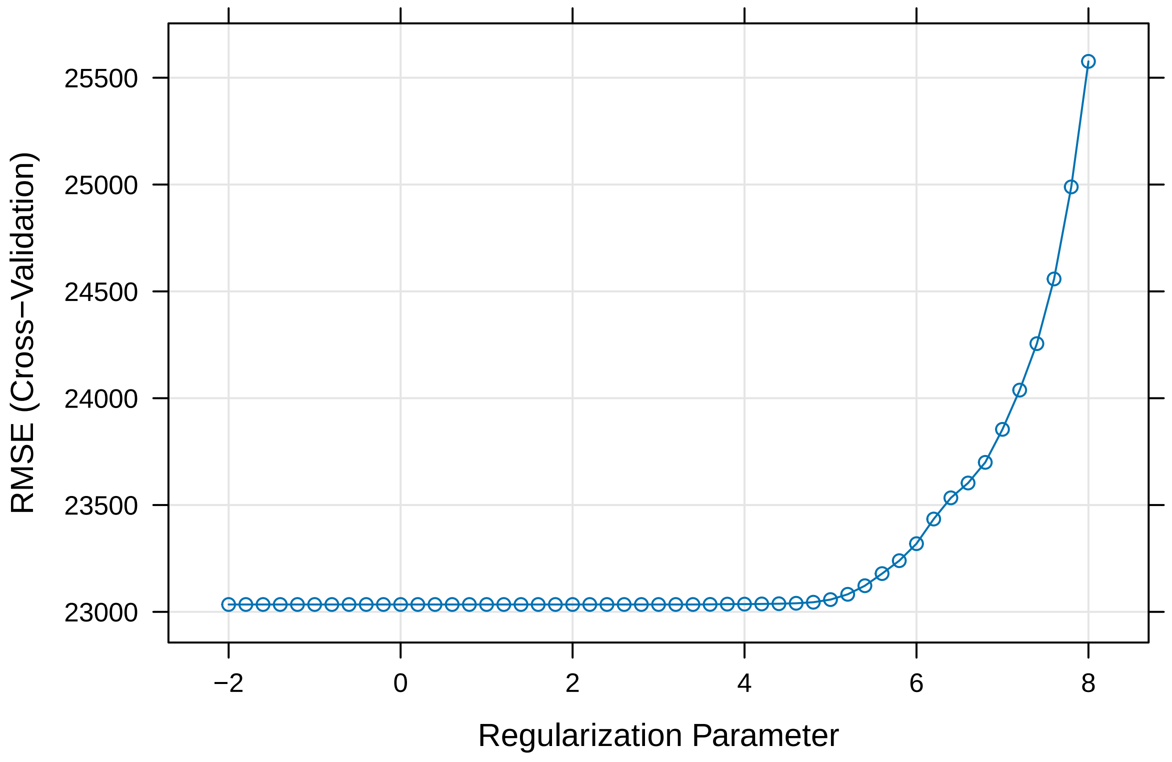
```r
library(ggplot2)
library(pls)
```

```
##
## 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
```

```r
train=read.csv("/Users/zeze/Library/Mobile Documents/com~apple~CloudDocs/2024/24S BIST P8106 DS II/hw1/
test=read.csv("/Users/zeze/Library/Mobile Documents/com~apple~CloudDocs/2024/24S BIST P8106 DS II/hw1/h
train = na.omit(train)
test = na.omit(test)

# train
# matrix of predictors (glmnet uses input matrix)
x <- model.matrix(Sale_Price ~ ., train)[,-1]
# vector of response
y <- train[, "Sale_Price"]
x_test <- model.matrix(Sale_Price ~ ., test)[,-1]
y_test <- test[, "Sale_Price"]
```

(a) Fit a lasso model on the training data. Report the selected tuning parameter and the test error.

```r
ctrl1 <- trainControl(method = "cv", number = 10)

set.seed(2024)
lasso.fit <- train(Sale_Price ~ .,
                   data = train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(8, -2, length = 51))),
                   trControl = ctrl1)

# visualization
plot(lasso.fit, xTrans = log)
```



```r
# tuning parameter
lasso.fit$bestTune
```

```
##    alpha  lambda
## 28     1 29.9641
```

```r
# coefficients in the final model
coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)              -4.891889e+06
## Gr_Liv_Area               6.576019e+01
## First_Flr_SF              7.854992e-01
## Second_Flr_SF             .
## Total_Bsmt_SF             3.533373e+01
## Low_Qual_Fin_SF          -4.132375e+01
## Wood_Deck_SF              1.179064e+01
## Open_Porch_SF             1.574960e+01
## Bsmt_Unf_SF              -2.089000e+01
## Mas_Vnr_Area              1.072139e+01
## Garage_Cars               4.139513e+03
## Garage_Area               8.020923e+00
## Year_Built                3.241105e+02
## TotRms_AbvGrd            -3.705645e+03
## Full_Bath                   -4.036604e+03
## Overall_QualAverage         -4.924548e+03
## Overall_QualBelow_Average  -1.260070e+04
## Overall_QualExcellent       7.446458e+04
## Overall_QualFair           -1.091887e+04
## Overall_QualGood            1.218755e+04
## Overall_QualVery_Excellent  1.337787e+05
## Overall_QualVery_Good       3.793984e+04
## Kitchen_QualFair           -2.556466e+04
## Kitchen_QualGood           -1.785058e+04
## Kitchen_QualTypical        -2.590666e+04
## Fireplaces                  1.087859e+04
## Fireplace_QuFair           -7.738271e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.978771e+03
## Fireplace_QuPoor           -5.709901e+03
## Fireplace_QuTypical        -7.015803e+03
## Exter_QualFair             -3.511041e+04
## Exter_QualGood             -1.674606e+04
## Exter_QualTypical          -2.116559e+04
## Lot_Frontage                1.007469e+02
## Lot_Area                    6.044410e-01
## Longitude                  -3.366154e+04
## Latitude                    5.661433e+04
## Misc_Val                    8.658075e-01
## Year_Sold                  -5.939770e+02
```

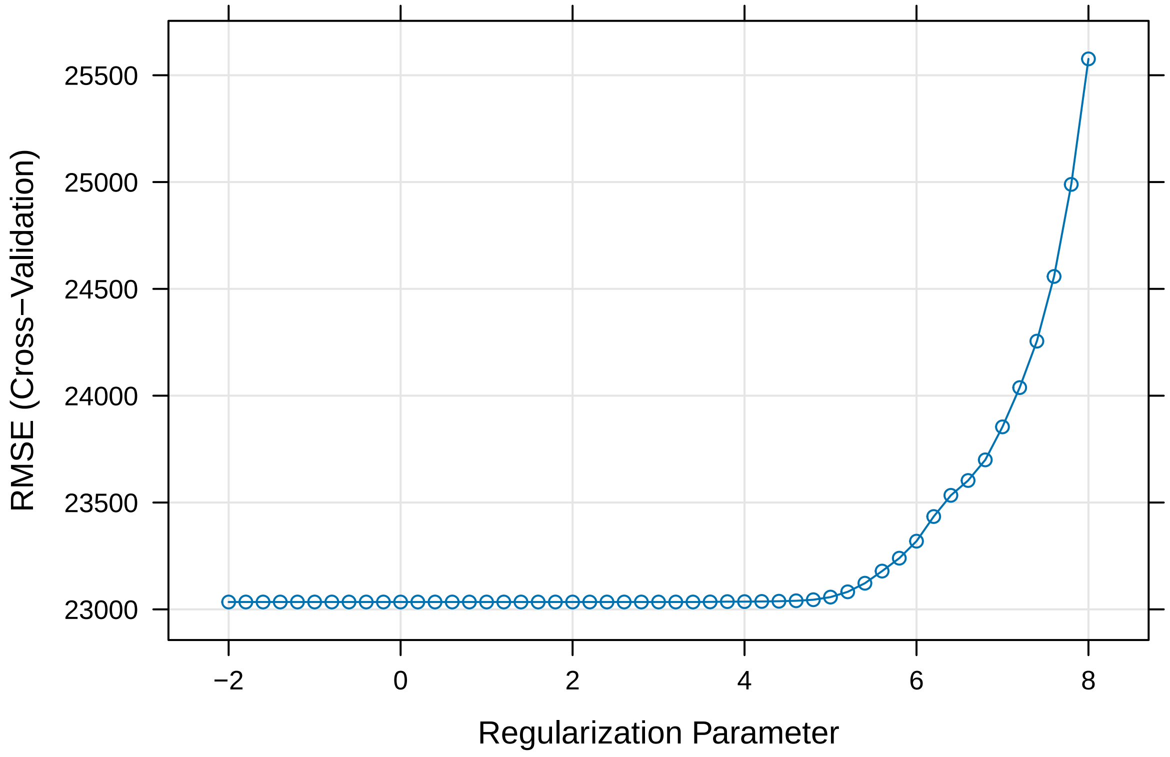
```r
# test error
lasso.predict <- predict(lasso.fit, newdata = test)
lasso.mse <- mean((y_test - lasso.predict)^2)
lasso.mse
```

```
## [1] 442720919
```

The selected tuning parameter lambda is 29.9641.

When the 1SE rule is applied, how many predictors are included in the model?

```r
set.seed(2024)
ctrl2 <- trainControl(method = "cv", number = 10, selectionFunction = "oneSE")
lasso.1se <- train(Sale_Price ~ .,
                   data = train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(8, -2, length = 51))),
                   trControl = ctrl2)
plot(lasso.1se, xTrans = log)
```



```
# tuning parameter
lasso.1se$bestTune
```

```
##    alpha  lambda
## 42     1 492.749
```

```
# coefficients in the final model
coef(lasso.1se$finalModel, lasso.1se$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                     s1
## (Intercept)                 -3.650675e+06
## Gr_Liv_Area                  5.998207e+01
## First_Flr_SF                 9.826894e-01
## Second_Flr_SF                .
## Total_Bsmt_SF                3.646813e+01
## Low_Qual_Fin_SF             -3.295822e+01
## Wood_Deck_SF                 9.554029e+00
## Open_Porch_SF                1.104636e+01
## Bsmt_Unf_SF                 -2.046797e+01
## Mas_Vnr_Area                 1.338002e+01
## Garage_Cars                  3.348448e+03
## Garage_Area                  1.006234e+01
## Year_Built                   3.128599e+02
## TotRms_AbvGrd               -2.258614e+03
## Full_Bath                   -8.177436e+02
## Overall_QualAverage         -3.767198e+03
## Overall_QualBelow_Average  -1.043870e+04
## Overall_QualExcellent       8.920869e+04
## Overall_QualFair           -8.227250e+03
## Overall_QualGood            1.085673e+04
## Overall_QualVery_Excellent  1.586352e+05
## Overall_QualVery_Good       3.719382e+04
## Kitchen_QualFair           -1.066519e+04
## Kitchen_QualGood           -4.286037e+03
## Kitchen_QualTypical        -1.336194e+04
## Fireplaces                  7.540979e+03
## Fireplace_QuFair           -2.375242e+03
## Fireplace_QuGood            3.111775e+03
## Fireplace_QuNo_Fireplace    .
## Fireplace_QuPoor           -1.431078e+02
## Fireplace_QuTypical        -2.918033e+03
## Exter_QualFair             -1.645683e+04
## Exter_QualGood              .
## Exter_QualTypical          -4.787335e+03
## Lot_Frontage                8.313807e+01
## Lot_Area                    5.866520e-01
## Longitude                  -1.973220e+04
## Latitude                    3.326790e+04
## Misc_Val                    1.504903e-01
## Year_Sold                  -7.950213e+01
```

```
num_1se <- sum(coef(lasso.1se$finalModel, lasso.1se$bestTune$lambda)[-1, ] != 0)
num_1se
```

```
## [1] 36
```

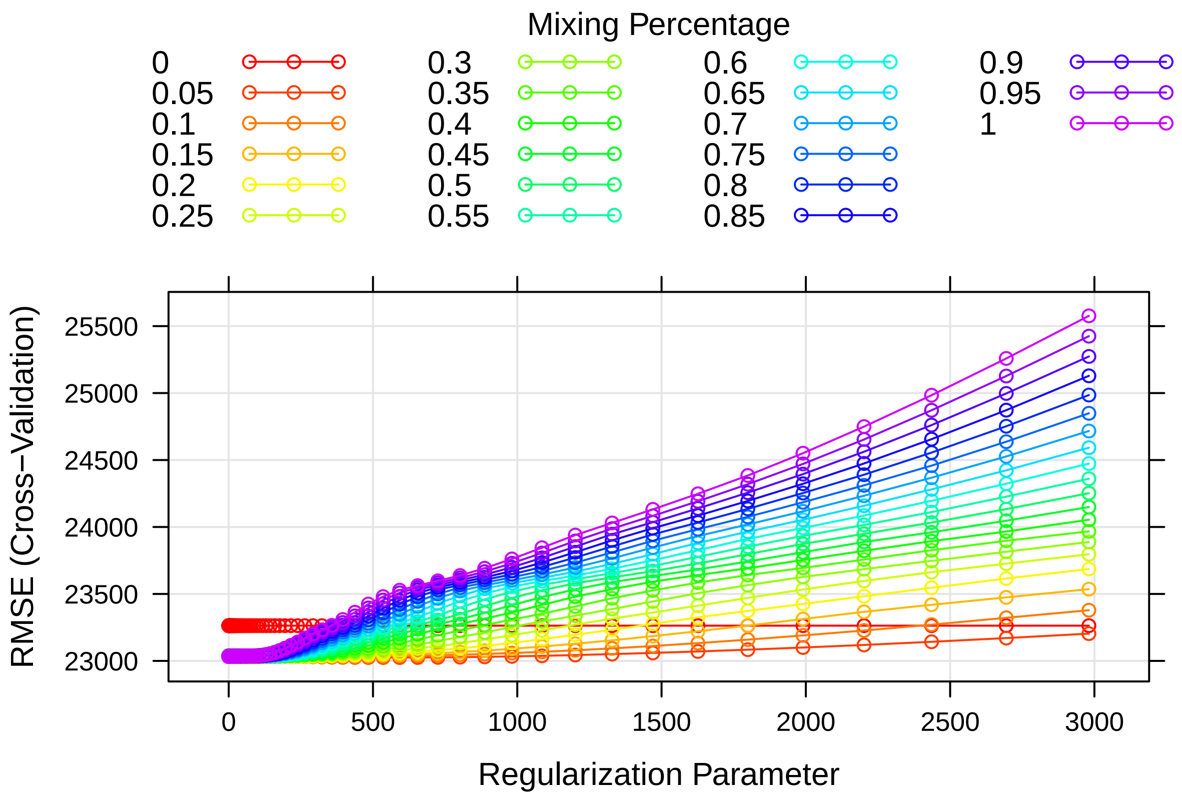
There are 36 nonzero coefficients when the 1SE rule is applied.

(b) Fit an elastic net model on the training data. Report the selected tuning parameters and the test error.

```
set.seed(2024)
enet.caret.fit <- train(Sale_Price ~ .,
                   data = train,
                   method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(8, -2, length = 100))),
                  trControl = ctrl1)
enet.caret.fit$bestTune
```

```
##     alpha   lambda
## 184  0.05 592.1964
```

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

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



```r
# coefficients in the final model
coef(enet.caret.fit$finalModel, enet.caret.fit$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)             -5.114274e+06
## Gr_Liv_Area              3.875866e+01
## First_Flr_SF             2.669170e+01
## Second_Flr_SF            2.543806e+01
## Total_Bsmt_SF            3.493817e+01
## Low_Qual_Fin_SF         -1.586520e+01
## Wood_Deck_SF             1.233064e+01
## Open_Porch_SF            1.689199e+01
## Bsmt_Unf_SF             -2.072377e+01
## Mas_Vnr_Area             1.167352e+01
## Garage_Cars              4.045106e+03
## Garage_Area              8.907812e+00
## Year_Built               3.190995e+02
## TotRms_AbvGrd           -3.434638e+03
## Full_Bath               -3.684017e+03
## Overall_QualAverage     -5.117400e+03
## Overall_QualBelow_Average  -1.270570e+04
## Overall_QualExcellent    7.585371e+04
## Overall_QualFair        -1.147647e+04
## Overall_QualGood         1.197632e+04
## Overall_QualVery_Excellent  1.364418e+05
## Overall_QualVery_Good       3.764880e+04
## Kitchen_QualFair            -2.364800e+04
## Kitchen_QualGood            -1.607125e+04
## Kitchen_QualTypical         -2.412269e+04
## Fireplaces                   1.082077e+04
## Fireplace_QuFair            -7.859362e+03
## Fireplace_QuGood             1.483160e+02
## Fireplace_QuNo_Fireplace     1.803069e+03
## Fireplace_QuPoor            -5.805479e+03
## Fireplace_QuTypical         -6.962902e+03
## Exter_QualFair              -3.291145e+04
## Exter_QualGood              -1.450876e+04
## Exter_QualTypical           -1.910816e+04
## Lot_Frontage                 1.001339e+02
## Lot_Area                     6.031761e-01
## Longitude                   -3.515975e+04
## Latitude                     5.774130e+04
## Misc_Val                     8.675661e-01
## Year_Sold                   -5.738409e+02
```

```
# test error
enet.caret.predict <- predict(enet.caret.fit, newdata = test)
enet.caret.mse <- mean((y_test - enet.caret.predict)^2)
enet.caret.mse
```

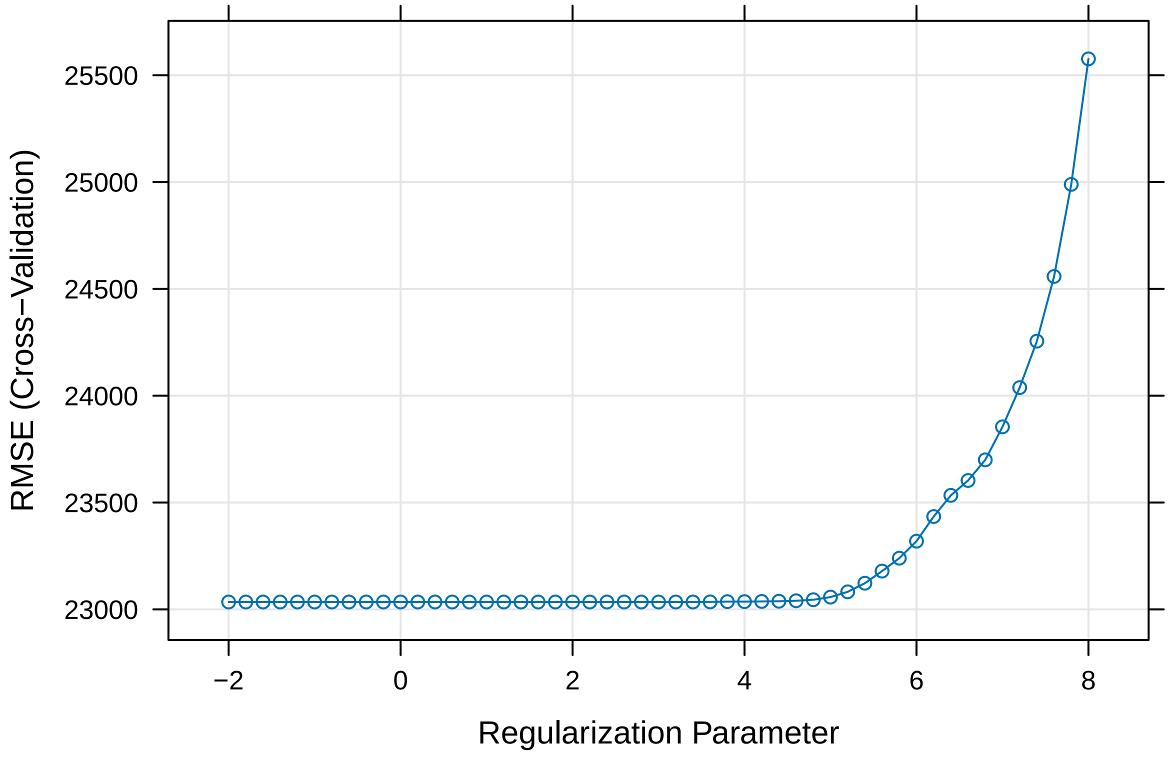
```
## [1] 438490038
```

The best parameter is when alpha = 0.05 and lambda = 592.196428.

Is it possible to apply the 1SE rule to select the tuning parameters for elastic net?

If the 1SE rule is applicable, implement it to select the tuning parameters. If not, explain why

```
set.seed(2024)
enet.caret.1se <- train(Sale_Price ~ .,
                   data = train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(8, -2, length = 51))),
                   trControl = ctrl2)
plot(enet.caret.1se, xTrans = log)
```



```r
# tuning parameter
enet.caret.1se$bestTune
```

```
##    alpha  lambda
## 42     1 492.749
```

```r
# coefficients in the final model
coef(enet.caret.1se$finalModel, enet.caret.1se$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)                -3.650675e+06
## Gr_Liv_Area                 5.998207e+01
## First_Flr_SF                9.826894e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.646813e+01
## Low_Qual_Fin_SF            -3.295822e+01
## Wood_Deck_SF                9.554029e+00
## Open_Porch_SF               1.104636e+01
## Bsmt_Unf_SF                -2.046797e+01
## Mas_Vnr_Area                1.338002e+01
## Garage_Cars                 3.348448e+03
## Garage_Area                 1.006234e+01
## Year_Built                  3.128599e+02
## TotRms_AbvGrd              -2.258614e+03
## Full_Bath                   -8.177436e+02
## Overall_QualAverage         -3.767198e+03
## Overall_QualBelow_Average  -1.043870e+04
## Overall_QualExcellent        8.920869e+04
## Overall_QualFair            -8.227250e+03
## Overall_QualGood             1.085673e+04
## Overall_QualVery_Excellent  1.586352e+05
## Overall_QualVery_Good        3.719382e+04
## Kitchen_QualFair            -1.066519e+04
## Kitchen_QualGood            -4.286037e+03
## Kitchen_QualTypical         -1.336194e+04
## Fireplaces                   7.540979e+03
## Fireplace_QuFair            -2.375242e+03
## Fireplace_QuGood             3.111775e+03
## Fireplace_QuNo_Fireplace     .
## Fireplace_QuPoor            -1.431078e+02
## Fireplace_QuTypical         -2.918033e+03
## Exter_QualFair              -1.645683e+04
## Exter_QualGood               .
## Exter_QualTypical           -4.787335e+03
## Lot_Frontage                 8.313807e+01
## Lot_Area                     5.866520e-01
## Longitude                   -1.973220e+04
## Latitude                     3.326790e+04
## Misc_Val                     1.504903e-01
## Year_Sold                   -7.950213e+01
```

```r
num_1se_enet_caret <- sum(coef(enet.caret.1se$finalModel, enet.caret.1se$bestTune$lambda)[-1, ] != 0)
num_1se_enet_caret
```

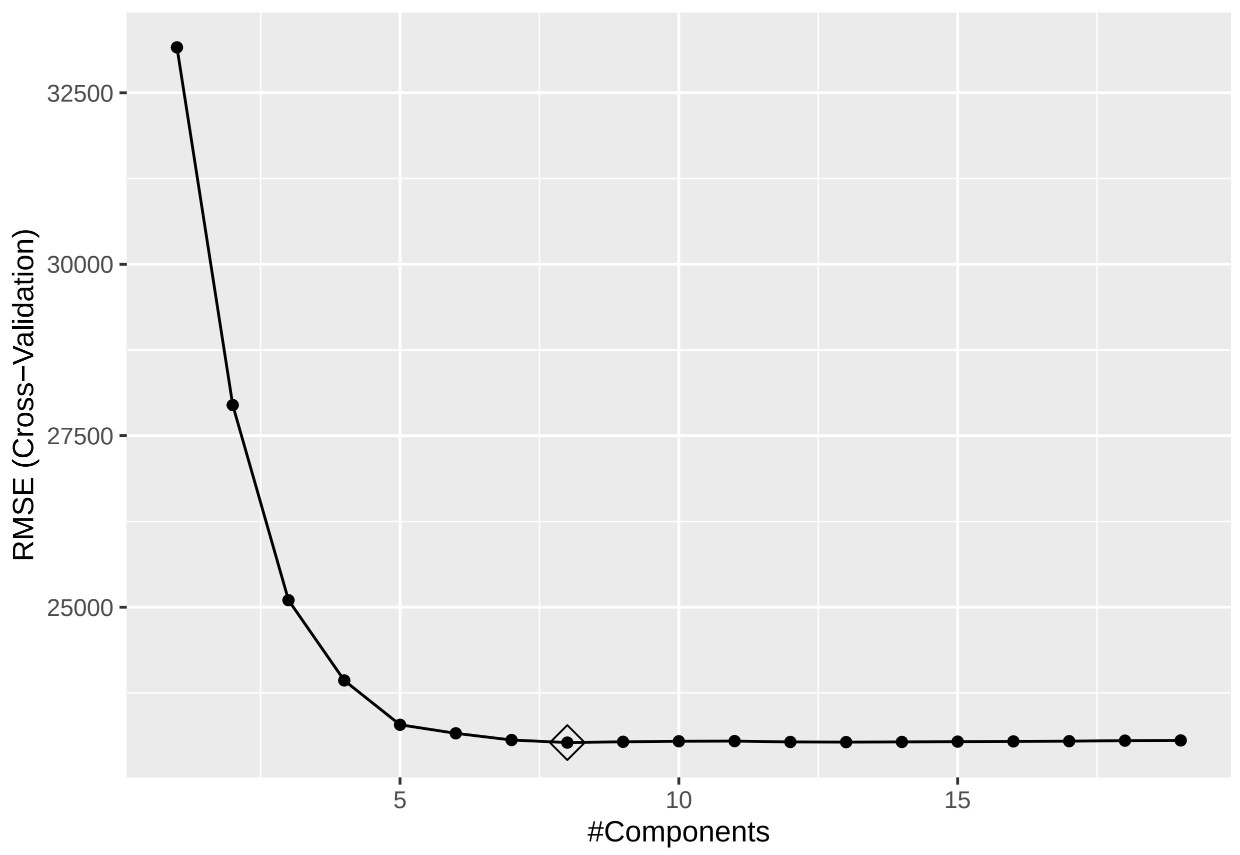
```
## [1] 36
```

There are 36 nonzero coefficients when the 1SE rule is applied.

(c) Fit a partial least squares model on the training data and report the test error.

How many components are included in your model?

```r
set.seed(2024)
pls.fit <- train(x, y,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:19),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))

ggplot(pls.fit, highlight = TRUE)
```



```
pls.fit$bestTune
```

```
##   ncomp
## 8     8
```

```
# test error
pls.predict <- predict(pls.fit, newdata = x_test)
pls.mse = mean((y_test - pls.predict)^2)
pls.mse
```

```
## [1] 440217938
```

The the test error of pls model is 4.4021794×10^8.

There are 8 components are included in my pls model.

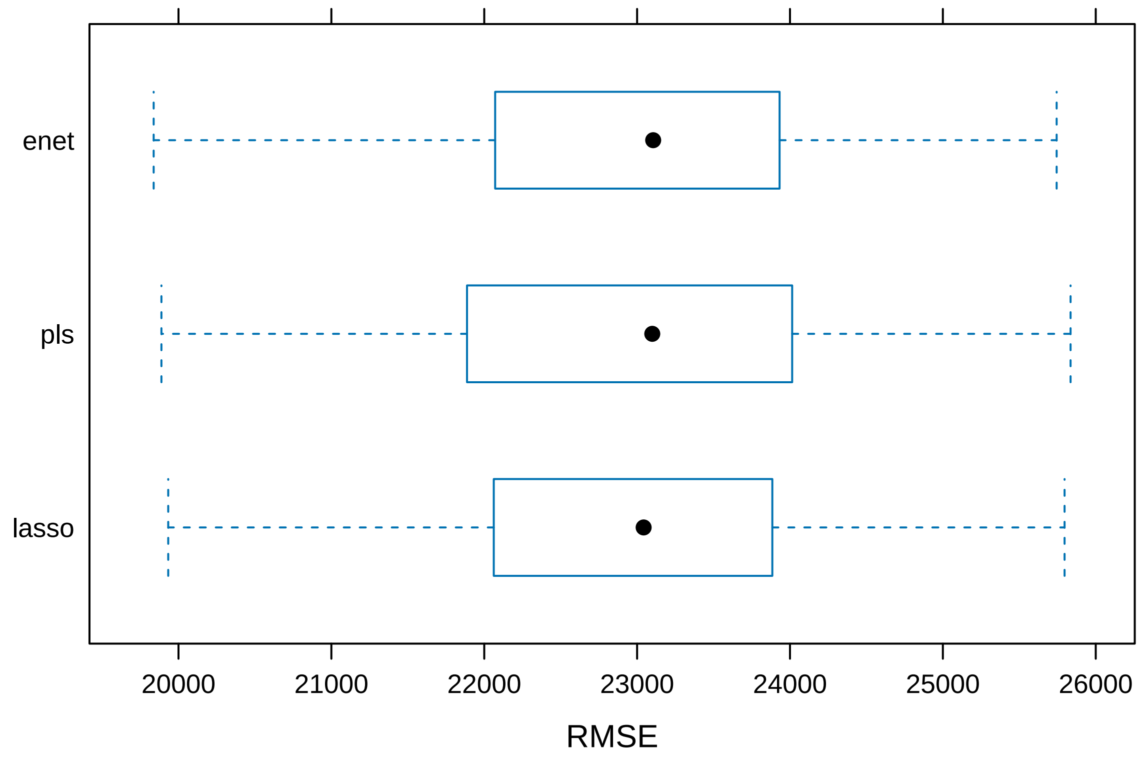
(d) Choose the best model for predicting the response and explain your choice.

```
rs <- resamples(list(lasso = lasso.fit, enet = enet.caret.fit, pls = pls.fit))
summary(rs)
```

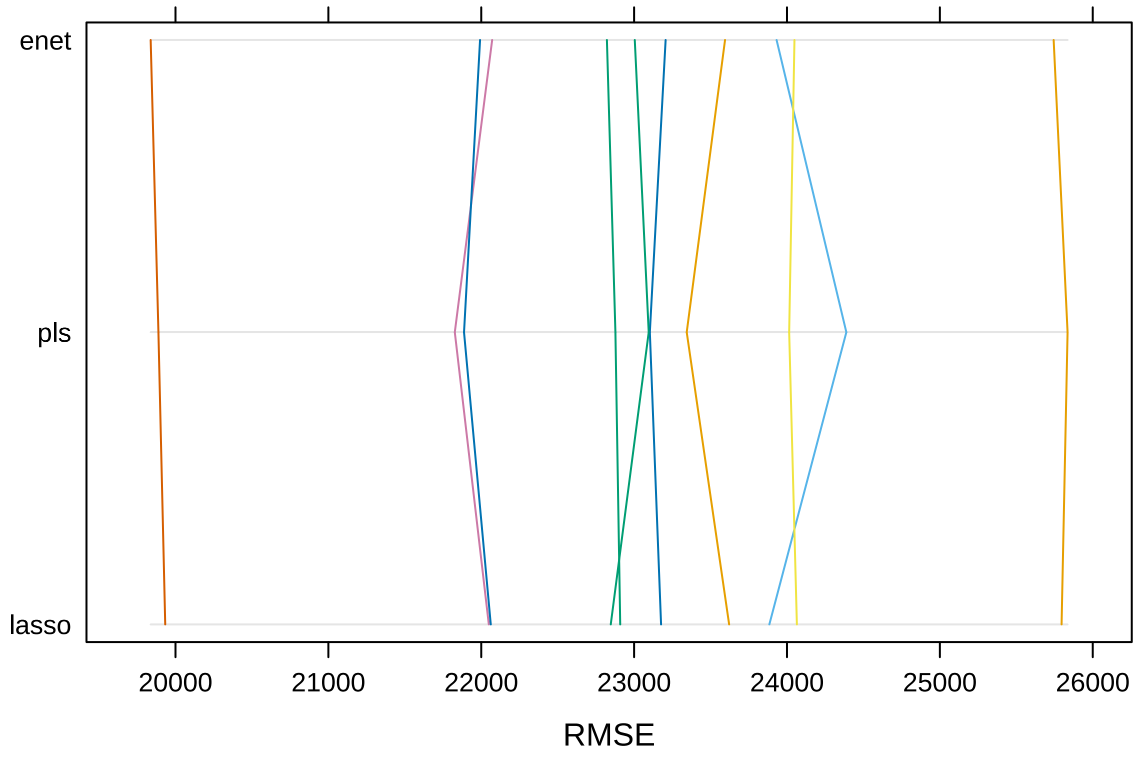
```
##
## Call:
## summary.resamples(object = rs)
```

```
##
## Models: lasso, enet, pls
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 14225.06 16179.99 16580.96 16714.49 17206.37 20650.73    0
## enet  14146.21 16186.76 16512.61 16663.56 17197.45 20574.68    0
## pls   14210.11 16150.19 16467.89 16679.74 17112.41 20673.72    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 19932.90 22258.50 23042.64 23034.42 23818.78 25796.20    0
## enet  19837.53 22259.04 23105.14 23025.33 23847.63 25744.40    0
## pls   19888.40 22135.02 23099.22 23026.11 23846.79 25835.45    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.8678709 0.8844116 0.9046978 0.9021219 0.9195353 0.9276290    0
## enet  0.8677922 0.8846764 0.9048287 0.9022137 0.9188753 0.9278572    0
## pls   0.8689165 0.8835208 0.9055200 0.9021858 0.9183595 0.9292953    0
```

```r
bwplot(rs, metric = "RMSE")
```



```r
parallelplot(rs, metric = "RMSE")
```



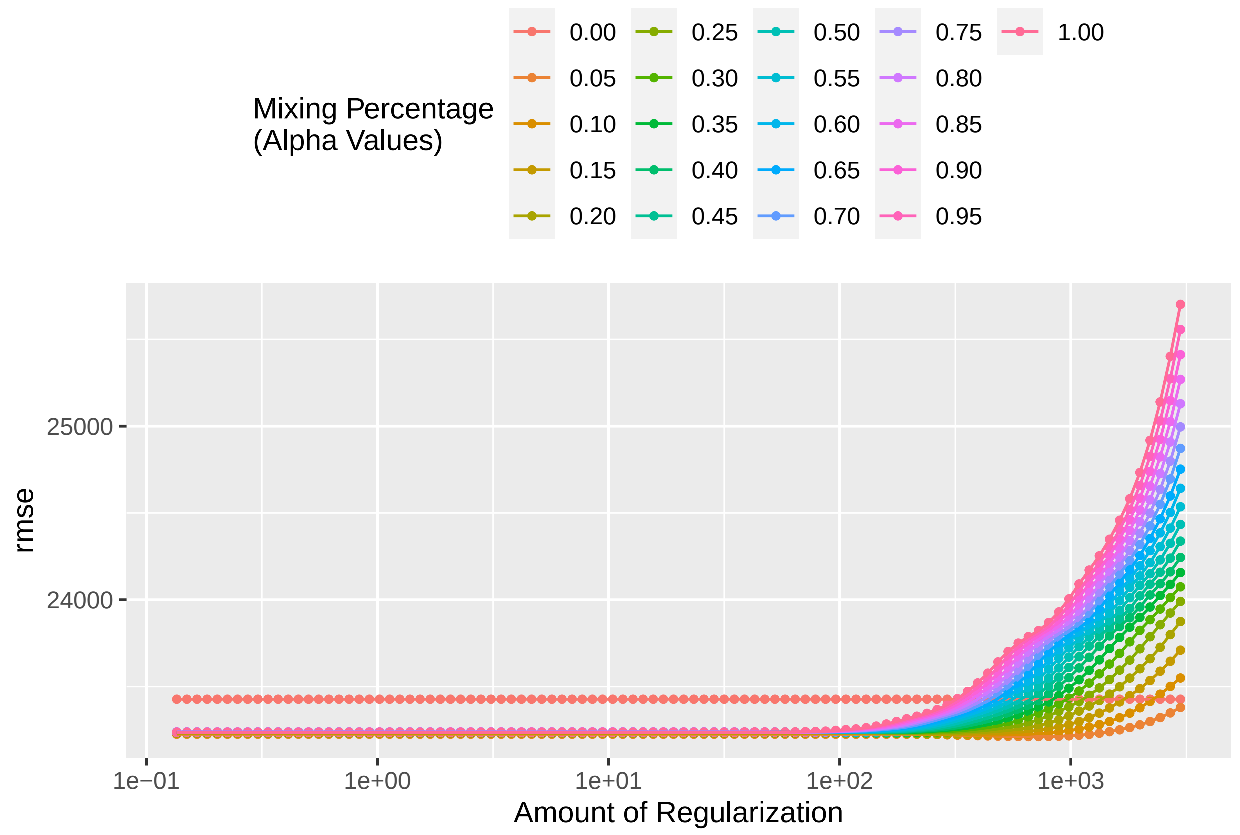
The elastic net model since the test error has smaller MAE and RMSE.

However, partial least squares model has higher Rsquared, but the difference is very slight.

Overall, I will choose elastic net model.

(e) If "caret" was used for the elastic net in (b), retrain this model with "tidymodels", and vice versa.

```r
set.seed(2024)
cv_folds <- vfold_cv(train, v = 10)

enet_spec <- linear_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet") %>%
  set_mode("regression")

# enet_spec %>% extract_parameter_dials("mixture")

enet_grid_set <- parameters(penalty(range = c(-2, 8), trans = log_trans()),
                            mixture(range = c(0, 1)))
enet_grid <- grid_regular(enet_grid_set, levels = c(100, 21))



enet_workflow <- workflow() %>%
  add_model(enet_spec) %>%
  add_formula(Sale_Price ~ .)

enet_tune <- tune_grid(
  enet_workflow,
  resamples = cv_folds,
  grid = enet_grid
)

autoplot(enet_tune, metric = "rmse") +
  theme(legend.position = "top") +
  labs(color = "Mixing Percentage\n(Alpha Values)")
```



```
enet_best <- select_best(enet_tune, metric = "rmse")

final_enet_spec <- enet_spec %>%
  update(penalty = enet_best$penalty, mixture = enet_best$mixture)

enet_fit <- fit(final_enet_spec, formula = Sale_Price ~ ., data = train)

# Get coefficients
enet_model <- extract_fit_engine(enet_fit)
coef(enet_model, s = enet_best$penalty)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -5.127779e+06
## Gr_Liv_Area                 3.866594e+01
## First_Flr_SF                2.664915e+01
## Second_Flr_SF               2.534443e+01
## Total_Bsmt_SF               3.491118e+01
## Low_Qual_Fin_SF            -1.588249e+01
## Wood_Deck_SF                1.236454e+01
## Open_Porch_SF               1.696177e+01
## Bsmt_Unf_SF                -2.070083e+01
## Mas_Vnr_Area                1.178732e+01
## Garage_Cars                 4.034888e+03
## Garage_Area                 9.006057e+00
## Year_Built                  3.184553e+02
## TotRms_AbvGrd              -3.398798e+03
## Full_Bath                  -3.629278e+03
## Overall_QualAverage        -5.122976e+03
## Overall_QualBelow_Average  -1.269084e+04
## Overall_QualExcellent       7.603516e+04
## Overall_QualFair           -1.149678e+04
## Overall_QualGood            1.193541e+04
## Overall_QualVery_Excellent  1.368041e+05
## Overall_QualVery_Good       3.758880e+04
## Kitchen_QualFair           -2.339282e+04
## Kitchen_QualGood           -1.584152e+04
## Kitchen_QualTypical        -2.389744e+04
## Fireplaces                  1.076339e+04
## Fireplace_QuFair           -7.874412e+03
## Fireplace_QuGood            1.415502e+02
## Fireplace_QuNo_Fireplace    1.688718e+03
## Fireplace_QuPoor           -5.819483e+03
## Fireplace_QuTypical        -6.970545e+03
## Exter_QualFair             -3.255621e+04
## Exter_QualGood             -1.415554e+04
## Exter_QualTypical          -1.878567e+04
## Lot_Frontage                9.990104e+01
## Lot_Area                    6.028135e-01
## Longitude                  -3.522036e+04
## Latitude                    5.764727e+04
## Misc_Val                    8.620465e-01
## Year_Sold                  -5.676133e+02
```

```r
num_enet <- sum(coef(enet_model, s = enet_best$penalty)[-1, ] != 0)
num_enet
```

```
## [1] 39
```

There are 39 nonzero coefficients when the 1SE rule is applied, which is 3 more than the caret model.

Compare the selected tuning parameters between the two software approaches.

Should there be discrepancies in the chosen parameters, discuss potential reasons for these differences.

```
enet_best
```

```
## # A tibble: 1 x 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1    655.    0.05 Preprocessor1_Model0185
```

```
enet.caret.fit$bestTune
```

```
##     alpha   lambda
## 184  0.05 592.1964
```

The tuning parameter of tidymodel are 655.139693128863, 0.05, Preprocessor1_Model0185, while that of caret model are 0.05, 592.196427961576.

The alpha is the same, but the lambda have a slightly difference. It may because that `caret` and `tidymodels` may implement the Elastic Net algorithm slightly differently under the hood, leading to discrepancies in the optimal parameters found. Different preprocessing steps could lead to models with different parameters being selected as optimal.