hw1

Johnstone Tcheou

2025-02-21

Contents

Set a seed to ensure reproducibility of results.

```
set.seed(81061)
library(ISLR)
library(glmnet)
library(caret)
library(tidymodels)
library(corrplot)
library(ggplot2)
library(plotmo)
library(ggrepel)
library(pls)
```

```
training <- read.csv("housing_training.csv")
testing <- read.csv("housing_test.csv")
```

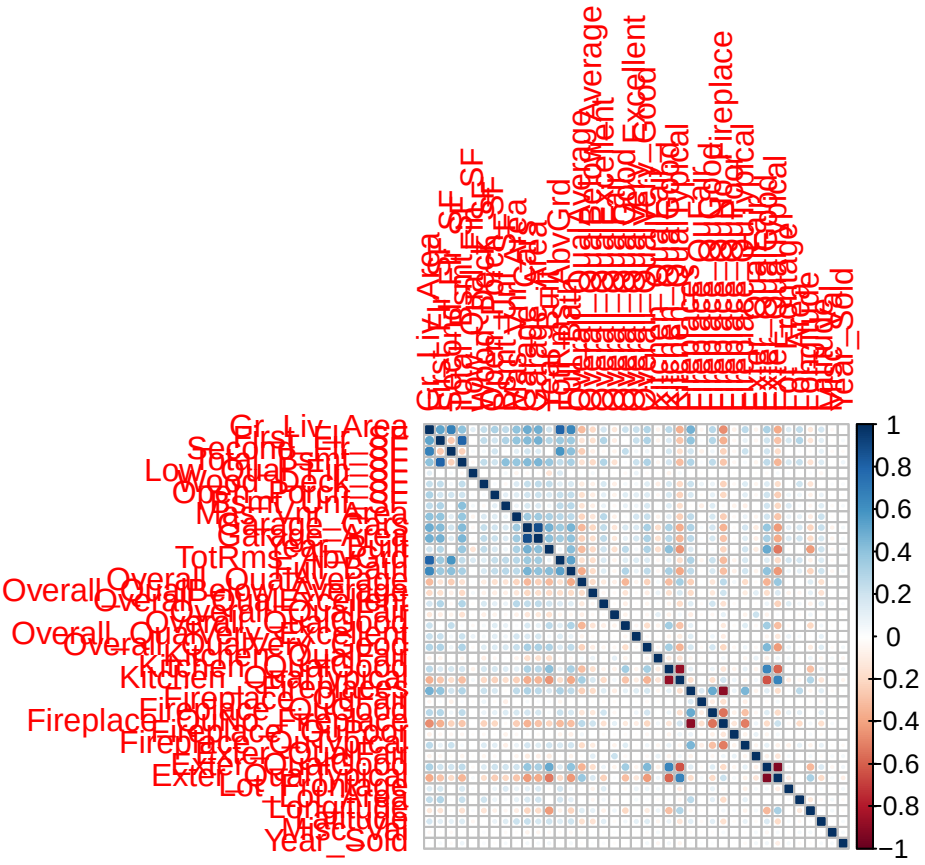
Question a

We will be using the `caret` metaengine to conduct a 10-fold cross-validation of a lasso regression on the training dataset. We will also need to configure the predictors as a matrix, the response variable as a vector, and the testing data as a matrix without the response variable.

```
predictors <- model.matrix(Sale_Price ~ ., training)[, -1]
response <- training[, "Sale_Price"]
test <- model.matrix(Sale_Price ~ ., testing)[, -1]
```

Before fitting a model, we should also check for correlations between predictors, which may cause problems with lasso regression.

```
corrplot::corrplot(
  cor(predictors),
  method = "circle",
  type = "full"
)
```



There are some predictors which are correlated with each other, such as `Total_Bsmt_SF` and `First_Flr_SF`, `Second_Flr_SF` and `Gr_Liv_Area`, `Kitchen_QualTypical` and `Kitchen_QualGood`, `Fireplaces` and `Fireplace_QuNo_FirePlace`.

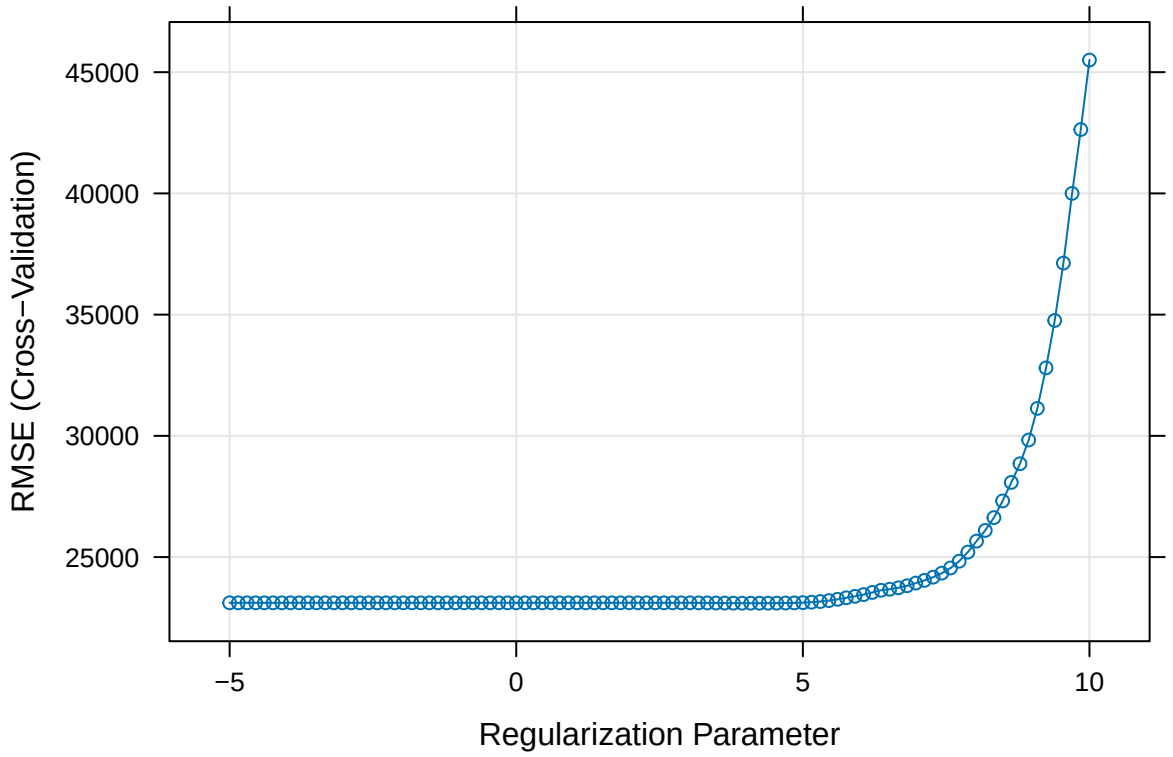
For 10-fold cross validation, we need to use `trainControl()` to specify control parameters, i.e. specifying the resampling method we are using and the number of folds - in this case, cross-validation and 10, respectively. The formula results are stored in `ctrl1`.

We then pass this to the `train()` function, along with the desired model statement as a formula (`Sale_Price ~ .`), the training dataset, `alpha = 1` for lasso regression, and a corresponding lambda grid hopefully wide enough to capture the optimal lambda value.

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

lasso_caret <-
  train(
    Sale_Price ~.,
    data = training,
    method = "glmnet",
    tuneGrid = expand.grid(
      alpha = 1,
      lambda = exp(seq(10, -5, length = 100))
    ),
    trControl = ctrl1
  )
```

```
plot(lasso_caret, xTrans = log)
```




```
lasso_caret$bestTune
```

```
##    alpha   lambda
## 61     1 59.79423
```

```
coef(lasso_caret$finalModel, lasso_caret$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -4.840779e+06
## Gr_Liv_Area                 6.546330e+01
## First_Flr_SF                7.987905e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.540118e+01
## Low_Qual_Fin_SF            -4.102390e+01
## Wood_Deck_SF                1.166675e+01
## Open_Porch_SF               1.549835e+01
## Bsmt_Unf_SF                -2.088901e+01
## Mas_Vnr_Area                1.086359e+01
## Garage_Cars                 4.095060e+03
## Garage_Area                 8.138121e+00
## Year_Built                  3.234957e+02
```

```
## TotRms_AbvGrd              -3.635125e+03
## Full_Bath                  -3.880963e+03
## Overall_QualAverage        -4.868338e+03
## Overall_QualBelow_Average  -1.248694e+04
## Overall_QualExcellent       7.529368e+04
## Overall_QualFair           -1.078634e+04
## Overall_QualGood            1.213551e+04
## Overall_QualVery_Excellent  1.353248e+05
## Overall_QualVery_Good       3.790245e+04
## Kitchen_QualFair           -2.498642e+04
## Kitchen_QualGood           -1.733299e+04
## Kitchen_QualTypical        -2.543213e+04
## Fireplaces                  1.061166e+04
## Fireplace_QuFair           -7.682725e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.548227e+03
## Fireplace_QuPoor           -5.656288e+03
## Fireplace_QuTypical        -7.013339e+03
## Exter_QualFair             -3.368377e+04
## Exter_QualGood             -1.541581e+04
## Exter_QualTypical          -1.984825e+04
## Lot_Frontage                9.987240e+01
## Lot_Area                    6.043141e-01
## Longitude                  -3.308279e+04
## Latitude                    5.539552e+04
## Misc_Val                    8.359457e-01
## Year_Sold                  -5.671644e+02
```

Based on the CV RMSE, the optimal lambda that minimizes the CV RMSE is 59.7942254, which corresponds to 4.0909091 on the graph.

Excluding the intercept, there are 37 predictors included in this final model.

```
lasso_caret_pred <- predict(lasso_caret, newdata = testing)

lasso_caret_testerror <- mean((lasso_caret_pred - testing[, "Sale_Price"])^2)
```

The test error for the selected lasso regression model is 4.4052006×10^8.

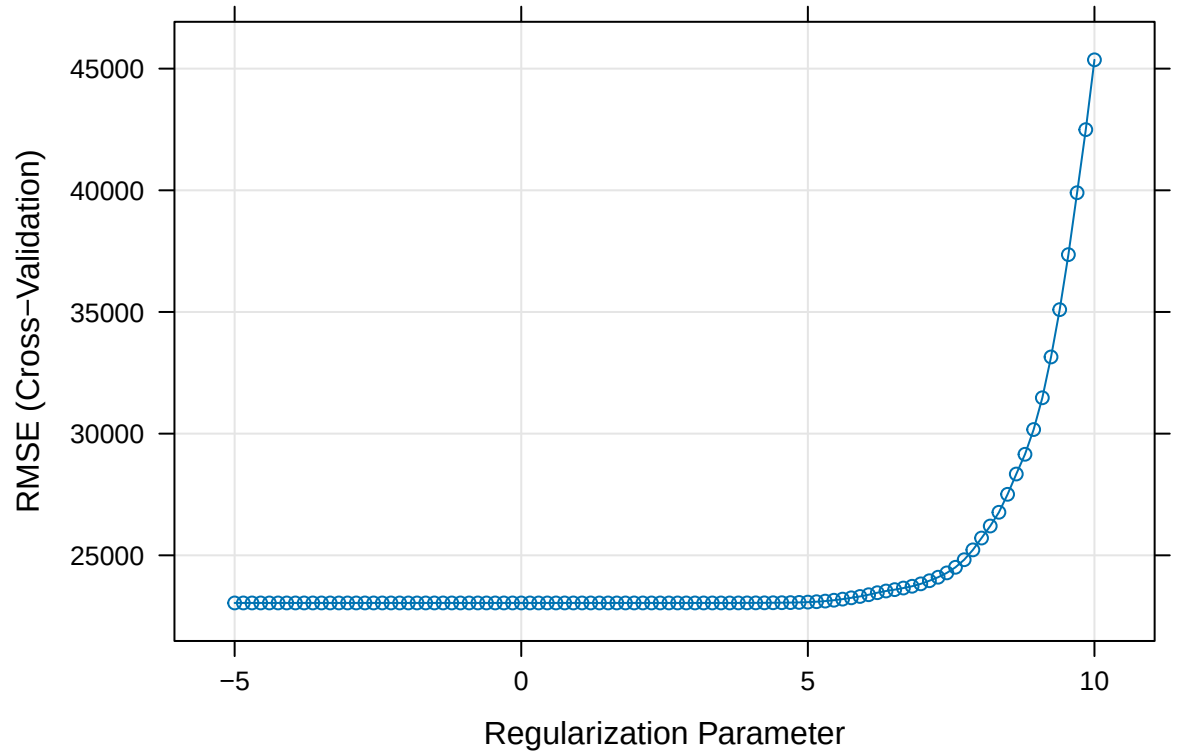
For `caret`, to get the lambda within the 1 SE rule, a different trainControl object needs to be initialized, with `oneSE` specified for the `selectionFunction` argument.

```
ctrl2 <- trainControl(method = "cv", number = 10, selectionFunction = "oneSE")

lasso_caret_1se <-
  train(
    Sale_Price ~.,
    data = training,
    method = "glmnet",
    tuneGrid = expand.grid(
      alpha = 1,
      lambda = exp(seq(10, -5, length = 100))
    ),
    trControl = ctrl2
```

```
  )

plot(lasso_caret_1se, xTrans = log)
```




```
lasso_caret_1se$bestTune
```

```
##    alpha   lambda
## 79     1 914.3211
```

```
coef(lasso_caret_1se$finalModel, lasso_caret_1se$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -1.833645e+06
## Gr_Liv_Area                 5.590387e+01
## First_Flr_SF                1.159582e+00
## Second_Flr_SF               .
## Total_Bsmt_SF               3.675634e+01
## Low_Qual_Fin_SF            -2.442870e+01
## Wood_Deck_SF                8.139740e+00
## Open_Porch_SF               7.319679e+00
## Bsmt_Unf_SF                -1.911105e+01
## Mas_Vnr_Area                1.434249e+01
## Garage_Cars                 3.083568e+03
```

```
## Garage_Area                 1.153597e+01
## Year_Built                  3.161492e+02
## TotRms_AbvGrd              -9.559177e+02
## Full_Bath                   .
## Overall_QualAverage        -2.939069e+03
## Overall_QualBelow_Average  -8.768967e+03
## Overall_QualExcellent       8.955958e+04
## Overall_QualFair           -5.666292e+03
## Overall_QualGood            9.534028e+03
## Overall_QualVery_Excellent  1.590825e+05
## Overall_QualVery_Good       3.571584e+04
## Kitchen_QualFair           -4.705230e+03
## Kitchen_QualGood            .
## Kitchen_QualTypical        -9.745366e+03
## Fireplaces                  6.594120e+03
## Fireplace_QuFair            .
## Fireplace_QuGood            4.550964e+03
## Fireplace_QuNo_Fireplace    .
## Fireplace_QuPoor            .
## Fireplace_QuTypical         .
## Exter_QualFair             -1.400666e+04
## Exter_QualGood              .
## Exter_QualTypical          -5.226267e+03
## Lot_Frontage                6.640048e+01
## Lot_Area                    5.492446e-01
## Longitude                  -7.796931e+03
## Latitude                    1.250104e+04
## Misc_Val                    .
## Year_Sold                   .
```

This generates a lambda of 914.3210959. In this model, there are 29 predictors included, excluding the intercept.

Question b

With elastic net, the `alpha` argument should be supplied a sequence of values from 0 to 1, being in between the extremes of 0 and 1, representing ridge and lasso regression.

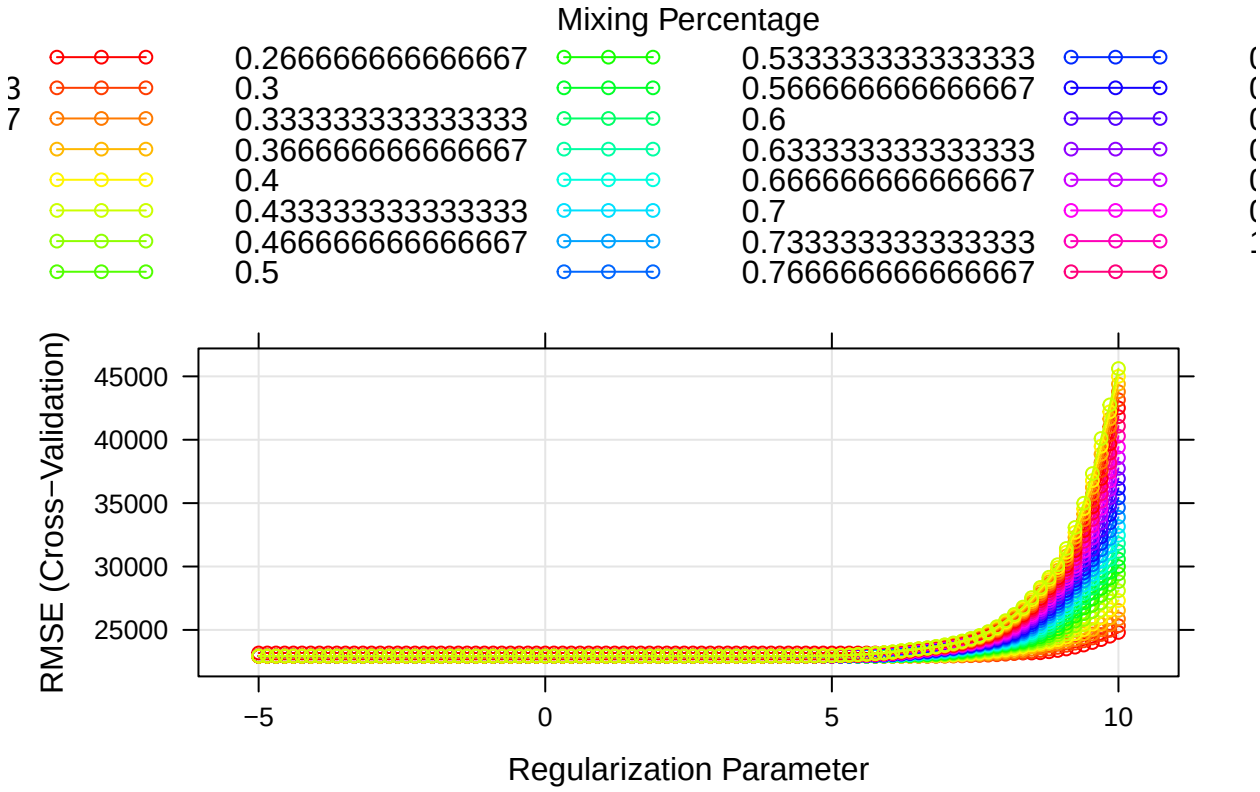
```
elastic_caret <-
  train(
    Sale_Price ~.,
    data = training,
    method = "glmnet",
    tuneGrid = expand.grid(
      alpha = seq(0, 1, length = 31),
      lambda = exp(seq(10, -5, length = 100))
    ),
    trControl = ctrl1
  )

elastic_caret$bestTune
```

```
##          alpha   lambda
## 176 0.03333333 580.3529
```

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

plot(elastic_caret, par.settings = myPar, xTrans = log)
```




```
coef(elastic_caret$finalModel, elastic_caret$bestTune$lambda)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                -5.144924e+06
## Gr_Liv_Area                 3.808779e+01
## First_Flr_SF                2.746150e+01
## Second_Flr_SF               2.624387e+01
## Total_Bsmt_SF               3.491390e+01
## Low_Qual_Fin_SF            -1.524123e+01
## Wood_Deck_SF                1.238235e+01
## Open_Porch_SF               1.698967e+01
## Bsmt_Unf_SF                -2.072841e+01
## Mas_Vnr_Area                1.161738e+01
## Garage_Cars                 4.060711e+03
## Garage_Area                 8.855567e+00
```

```
## Year_Built                   3.193289e+02
## TotRms_AbvGrd               -3.467301e+03
## Full_Bath                   -3.750131e+03
## Overall_QualAverage         -5.139922e+03
## Overall_QualBelow_Average   -1.275582e+04
## Overall_QualExcellent        7.555666e+04
## Overall_QualFair            -1.153906e+04
## Overall_QualGood             1.201138e+04
## Overall_QualVery_Excellent   1.359035e+05
## Overall_QualVery_Good        3.769049e+04
## Kitchen_QualFair            -2.394460e+04
## Kitchen_QualGood            -1.632514e+04
## Kitchen_QualTypical         -2.436249e+04
## Fireplaces                   1.095271e+04
## Fireplace_QuFair            -7.791909e+03
## Fireplace_QuGood             2.577608e+02
## Fireplace_QuNo_Fireplace     2.102623e+03
## Fireplace_QuPoor            -5.739219e+03
## Fireplace_QuTypical         -6.870557e+03
## Exter_QualFair              -3.342600e+04
## Exter_QualGood              -1.499088e+04
## Exter_QualTypical           -1.957084e+04
## Lot_Frontage                 1.005810e+02
## Lot_Area                     6.034191e-01
## Longitude                   -3.549894e+04
## Latitude                     5.828185e+04
## Misc_Val                     8.828488e-01
## Year_Sold                   -5.856503e+02
```

The lambda which minimizes the cross validation MSE is 580.3528982. This model has all 39 predictors (excluding the intercept).

```
elastic_caret_pred <- predict(elastic_caret, newdata = testing)

elastic_caret_testerror <- mean((elastic_caret_pred - testing[, "Sale_Price"])^2)
```

The test error from this optimal elastic net model is 4.3937318×10^8.

Unlike lasso, the 1SE rule is not applicable to elastic net regression. This is because elastic net has two different regularization parameters, alpha (the mixing of the two penalties), and lambda itself. The premise of the 1SE rule then becomes arbitrary when there are more than 1 regularization parameters that the 1SE can refer to.

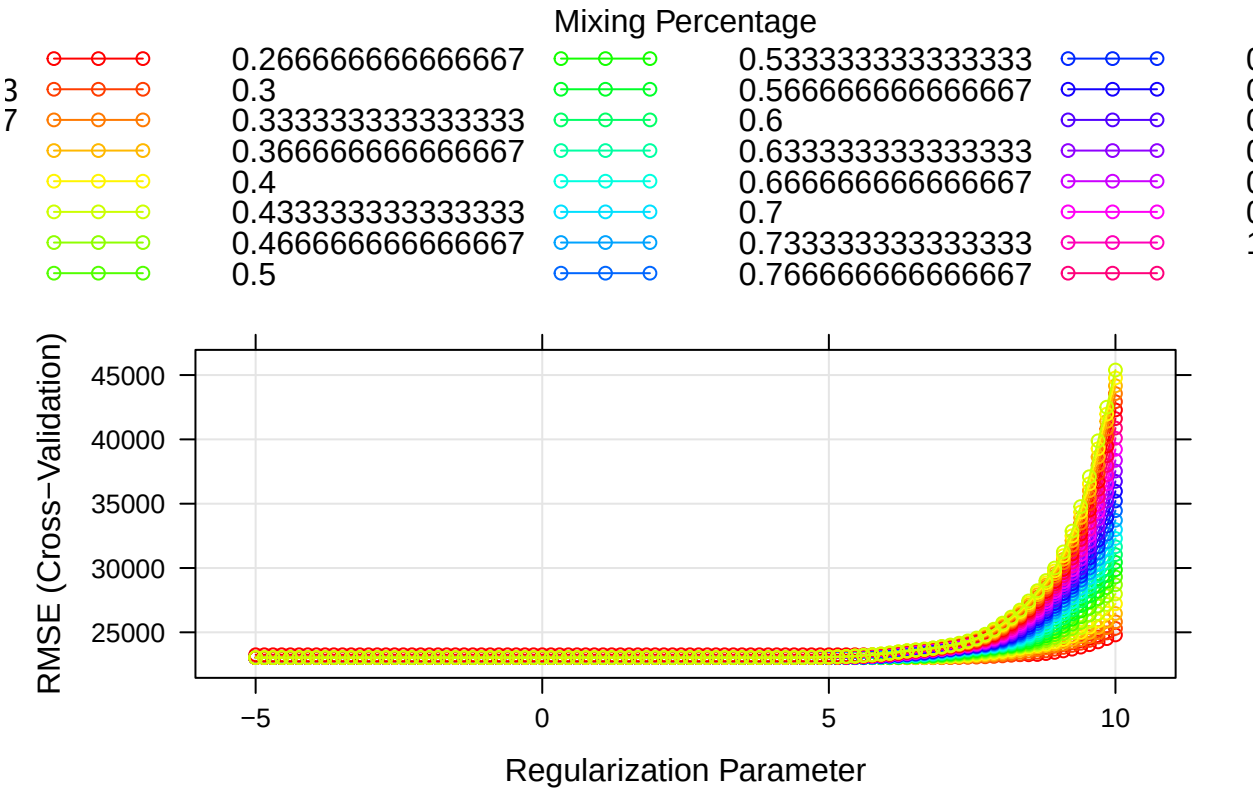
```
elastic_caret_1se <-
  train(
    Sale_Price ~.,
    data = training,
    method = "glmnet",
    tuneGrid = expand.grid(
      alpha = seq(0, 1, length = 31),
      lambda = exp(seq(10, -5, length = 100))
    ),
    trControl = ctrl2
  )
```

```
elastic_caret_1se$bestTune
```

```
##    alpha   lambda
## 93     0 7626.573
```

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

plot(elastic_caret_1se, par.settings = myPar, xTrans = log)
```




```
#coef(elastic_caret_1se$finalModel, elastic_caret_1se$bestTune$lambda)

elastic_caret_1se_pred <- predict(elastic_caret, newdata = testing)

elastic_caret_1se_testerror <- mean((elastic_caret_pred - testing[, "Sale_Price"])^2)
```

The test error when the 1SE rule is applied is 4.3937318×10^8.

Question c

```
pls_caret <-
  train(
    predictors, response,
    method = "pls",
    tuneGrid = data.frame(ncomp = 1:19),
    trControl = ctrl1,
    preProcess = c("center", "scale")
  )

pls_caret_pred <- predict(pls_caret, newdata = test)

pls_caret_testerror <- mean((pls_caret_pred - testing[, "Sale_Price"])^2)
```

The test error for the optimal partial least squares model is 4.4962272×10^8.

```
coef(pls_caret$finalModel, pls_caret$bestTune$ncomp)
```

```
## , , 12 comps
##
##                                .outcome
## Gr_Liv_Area                18756.88537
## First_Flr_SF               11050.66463
## Second_Flr_SF              11959.73073
## Total_Bsmt_SF              14259.35013
## Low_Qual_Fin_SF             -615.81992
## Wood_Deck_SF                1654.78686
## Open_Porch_SF               1147.52758
## Bsmt_Unf_SF                -8636.78765
## Mas_Vnr_Area                1718.35603
## Garage_Cars                 3539.98870
## Garage_Area                 1118.94737
## Year_Built                  9634.87380
## TotRms_AbvGrd              -6198.15321
## Full_Bath                  -2442.38900
## Overall_QualAverage        -2498.87545
## Overall_QualBelow_Average  -3475.02136
## Overall_QualExcellent      12344.65239
## Overall_QualFair           -1460.56548
## Overall_QualGood            4817.44603
## Overall_QualVery_Excellent 12624.88913
## Overall_QualVery_Good      11487.67494
## Kitchen_QualFair           -3416.40799
## Kitchen_QualGood           -9420.46352
## Kitchen_QualTypical       -13528.39030
## Fireplaces                  7648.59007
## Fireplace_QuFair           -1436.27750
## Fireplace_QuGood             -70.35994
## Fireplace_QuNo_Fireplace    1601.09601
## Fireplace_QuPoor            -806.93544
## Fireplace_QuTypical        -3043.58467
## Exter_QualFair             -3315.91744
## Exter_QualGood             -7309.97106
```

```
## Exter_QualTypical           -9510.00921
## Lot_Frontage                 3320.00348
## Lot_Area                     4977.45033
## Longitude                    -982.24036
## Latitude                     1139.65155
## Misc_Val                      523.60220
## Year_Sold                    -725.60818
```

This optimal PLR model also has 39 predictors, excluding the intercept.

Question d

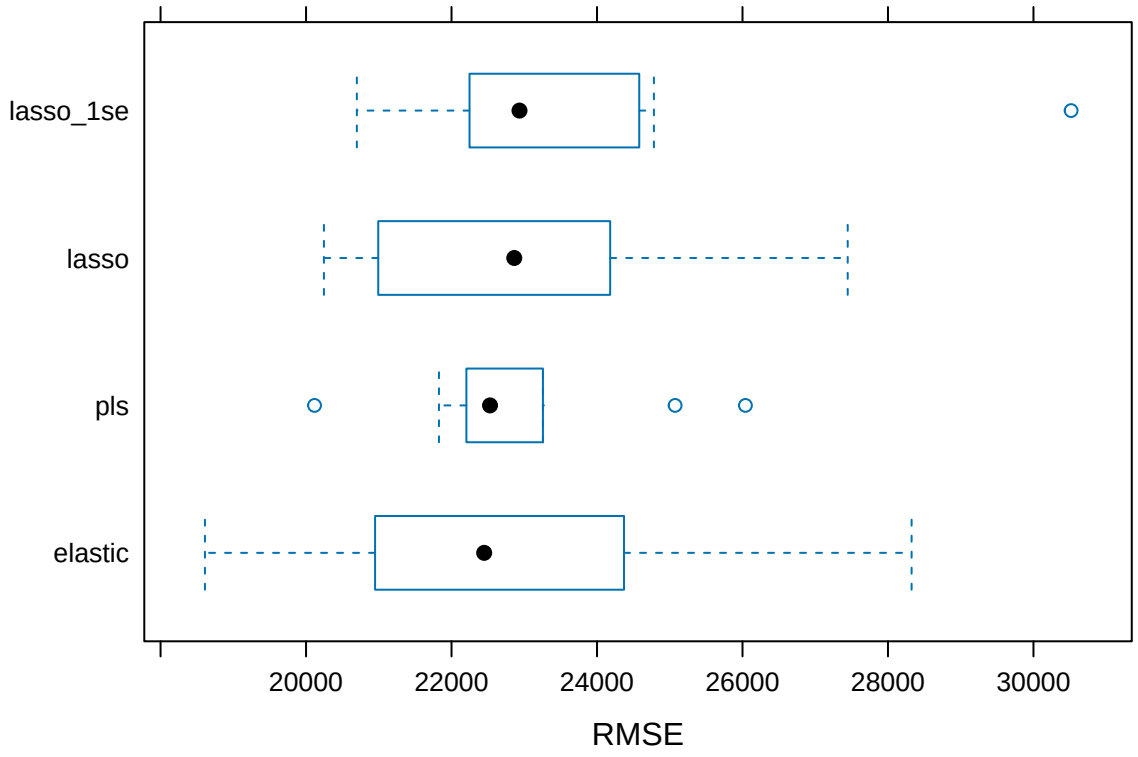
The best model to predict the response is the model with the lowest training error - which is lasso regression. Applied to the training data, the lasso model has a mean RMSE of 22709.16, while the lasso model with 1 SE rule has a mean RMSE of 23619.34, partial least squares has a mean RMSE of 22909.59, and elastic has a mean RMSE of 23001.77. Therefore, the best model is the lasso regression model.

```
resamp <- resamples(
  list(lasso = lasso_caret, lasso_1se = lasso_caret_1se, elastic = elastic_caret, pls = pls_caret)
)

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, lasso_1se, elastic, pls
## Number of resamples: 10
##
## MAE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso     15033.80 15597.48 16602.06 16703.33 17525.28 18919.79    0
## lasso_1se 15231.01 16378.38 16707.87 16857.31 16915.89 19747.02    0
## elastic   13900.98 16057.78 16439.93 16669.51 17507.72 19969.15    0
## pls       14996.23 16376.26 16694.34 16679.54 17102.06 18829.17    0
##
## RMSE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso     20245.70 21099.59 22862.38 23098.46 23993.08 27446.44    0
## lasso_1se 20698.32 22300.23 22934.29 23728.35 24540.70 30517.70    0
## elastic   18609.90 21213.61 22448.86 22921.75 24324.40 28324.74    0
## pls       20116.59 22214.73 22529.29 22903.31 23244.87 26040.55    0
##
## Rsquared
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso     0.8611835 0.8902872 0.9016153 0.9016068 0.9178027 0.9266175    0
## lasso_1se 0.8681562 0.8908306 0.8948502 0.8963668 0.9060775 0.9278425    0
## elastic   0.8537776 0.8958124 0.9063268 0.9038439 0.9193077 0.9339173    0
## pls       0.8791265 0.8933876 0.9050800 0.9020743 0.9128534 0.9187453    0
```

```
bwplot(resamp, metric = "RMSE")
```



Question e

We will be using `glmnet` to fit a lasso model on the training data.

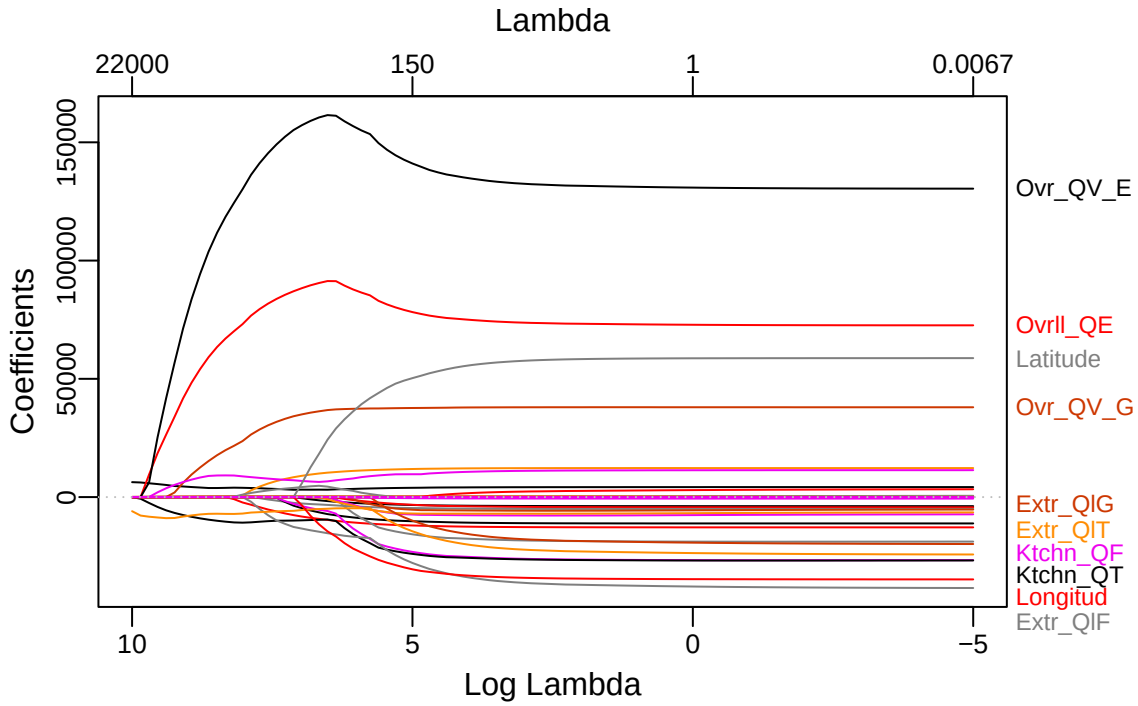
Fitting the lasso model with `glmnet` requires `alpha` to be 1 for lasso regression, and passing a sequence of `lambda` values to hopefully capture the optimal lambda. The lambda values must be in descending order, hence the initial sequence value of 10 and the terminal sequence value of -20.

```
lasso_glmnet <-
  glmnet(
    predictors,
    response,
    alpha = 1,
    lambda = exp(seq(10, -5, length = 100))
)

mat.coef <- coef(lasso_glmnet)
dim(mat.coef)
```

```
## [1]  40 100
```

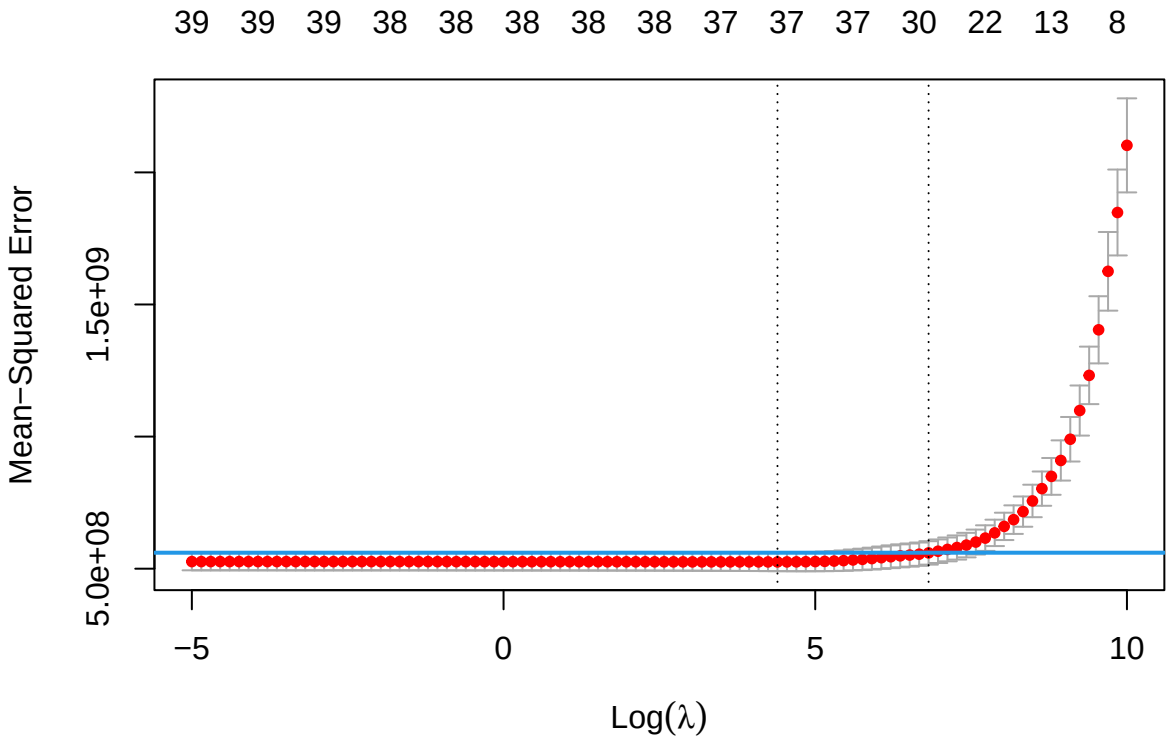
```
plot_glmnet(lasso_glmnet)
```



Use 10-fold cross validation to determine optimal lambda and regression parameters.

```
lasso_glmnet_cv <-
  cv.glmnet(
    predictors,
    response,
    alpha = 1,
    lambda = exp(seq(10, -5, length = 100))
  )

plot(lasso_glmnet_cv)
abline(h = (lasso_glmnet_cv$cvm + lasso_glmnet_cv$cvsd)[which.min(lasso_glmnet_cv$cvm)], col = 4, lwd =
```




```
lasso_glmnet_cv$lambda.1se
```

```
## [1] 914.3211
```

```
lasso_glmnet_cv$cvm[which.min(lasso_glmnet_cv$cvm)]
```

```
## [1] 525595138
```

The lambda value, our tuning parameter, that minimizes MSE is given by 80.9587199, and the smallest lambda value within 1 SE of the MSE is given by 914.3210959.

To get the parameters of the model which minimizes MSE, we need to pass the corresponding lambda, `lasso_glmnet_cv$lambda_min`, to the `s` argument of `predict()`.

```
predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.min, type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                -4.776647e+06
## Gr_Liv_Area                 6.523458e+01
## First_Flr_SF                8.099936e-01
## Second_Flr_SF               .
## Total_Bsmt_SF               3.546165e+01
## Low_Qual_Fin_SF            -4.077091e+01
```

```
## Wood_Deck_SF                1.153538e+01
## Open_Porch_SF               1.525018e+01
## Bsmt_Unf_SF                -2.086593e+01
## Mas_Vnr_Area                1.096835e+01
## Garage_Cars                 4.062059e+03
## Garage_Area                 8.239439e+00
## Year_Built                  3.234063e+02
## TotRms_AbvGrd              -3.571120e+03
## Full_Bath                  -3.756191e+03
## Overall_QualAverage        -4.805719e+03
## Overall_QualBelow_Average  -1.235997e+04
## Overall_QualExcellent       7.572879e+04
## Overall_QualFair           -1.061939e+04
## Overall_QualGood            1.205545e+04
## Overall_QualVery_Excellent  1.363974e+05
## Overall_QualVery_Good       3.779305e+04
## Kitchen_QualFair           -2.489531e+04
## Kitchen_QualGood           -1.727310e+04
## Kitchen_QualTypical        -2.542113e+04
## Fireplaces                  1.034447e+04
## Fireplace_QuFair           -7.615630e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.122172e+03
## Fireplace_QuPoor           -5.601195e+03
## Fireplace_QuTypical        -7.002598e+03
## Exter_QualFair             -3.223191e+04
## Exter_QualGood             -1.405814e+04
## Exter_QualTypical          -1.850266e+04
## Lot_Frontage                9.899308e+01
## Lot_Area                    6.038976e-01
## Longitude                  -3.245461e+04
## Latitude                    5.402129e+04
## Misc_Val                    8.088360e-01
## Year_Sold                  -5.415878e+02
```

```
dim(predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.min, type = "coefficients"))
```

```
## [1] 40  1
```

```
predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.1se, type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                -1.844218e+06
## Gr_Liv_Area                 5.594300e+01
## First_Flr_SF                1.185739e+00
## Second_Flr_SF               .
## Total_Bsmt_SF               3.677880e+01
## Low_Qual_Fin_SF            -2.446653e+01
## Wood_Deck_SF                8.166610e+00
## Open_Porch_SF               7.369809e+00
## Bsmt_Unf_SF                -1.912732e+01
## Mas_Vnr_Area                1.432827e+01
```

```
## Garage_Cars                 3.126503e+03
## Garage_Area                 1.139961e+01
## Year_Built                  3.158255e+02
## TotRms_AbvGrd              -9.650770e+02
## Full_Bath                   .
## Overall_QualAverage        -2.941056e+03
## Overall_QualBelow_Average  -8.770246e+03
## Overall_QualExcellent       8.952784e+04
## Overall_QualFair           -5.663642e+03
## Overall_QualGood            9.527042e+03
## Overall_QualVery_Excellent  1.590395e+05
## Overall_QualVery_Good       3.569317e+04
## Kitchen_QualFair           -4.657608e+03
## Kitchen_QualGood            .
## Kitchen_QualTypical        -9.707437e+03
## Fireplaces                  6.576694e+03
## Fireplace_QuFair            .
## Fireplace_QuGood            4.549987e+03
## Fireplace_QuNo_Fireplace    .
## Fireplace_QuPoor            .
## Fireplace_QuTypical         .
## Exter_QualFair             -1.404558e+04
## Exter_QualGood              .
## Exter_QualTypical          -5.250572e+03
## Lot_Frontage                6.641687e+01
## Lot_Area                    5.488127e-01
## Longitude                  -7.897670e+03
## Latitude                    1.254227e+04
## Misc_Val                    .
## Year_Sold                   .
```

```
dim(predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.1se, type = "coefficients"))
```

```
## [1] 40  1
```

With the lambda that minimizes MSE, there are 37 predictors excluding the intercept. With the smallest lambda within 1 SE, there are likewise 29 predictors. Both of these number of predictors corroborate with the results from `caret`.

For the optimal lambda values, both the `glmnet` (80.9587199) and `caret` (59.7942254) implementations get the same numbers. However, the lambdas within the 1 SE rule are also the different (914.3210959 for `glmnet` and 914.3210959 for `caret`). The differences in this may be in the implementation of getting the 1 SE of the regularization parameter, since `caret` requires fitting a new model altogether using the prespecified control parameters with the 1 SE rule, while the lambda within 1 SE is a subobject of the model outputted by `glmnet`.

```
lasso_glmnet_cv_pred <- predict(lasso_glmnet_cv, s = lasso_glmnet_cv$lambda.min, newx = test, type = "r

lasso_glmnet_cv_testerror <- mean((lasso_glmnet_cv_pred - testing[, "Sale_Price"])^2)
```

After fitting the lasso regression model to the testing dataset, the test error is 4.3879387×10^8.