Data Science II - Homework 1

Bowen Xia (bx2232)

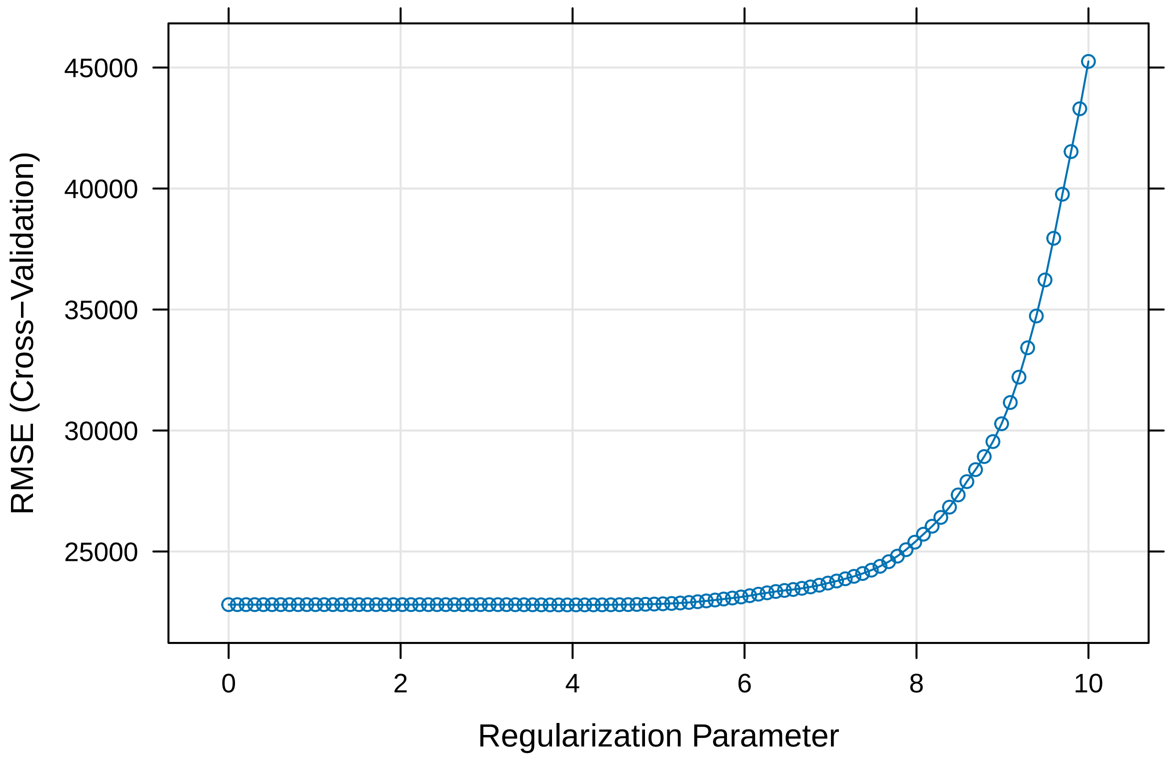
Contents

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

```r
train <- read.csv("housing_training.csv")
test  <- read.csv("housing_test.csv")

fac_vars <- c("Overall_Qual", "Kitchen_Qual", "Fireplace_Qu", "Exter_Qual")
train[fac_vars] <- lapply(train[fac_vars], as.factor)
test[fac_vars]  <- lapply(test[fac_vars],  as.factor)

y      <- train$Sale_Price
y.test <- test$Sale_Price

x      <- model.matrix(Sale_Price ~ ., train)[, -1]
x.test <- model.matrix(Sale_Price ~ ., test)[, -1]
```

(a) Lasso

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

set.seed(2)
lasso.fit <- train(Sale_Price ~ .,
                   data      = train,
                   method    = "glmnet",
                   tuneGrid  = expand.grid(alpha  = 1,
                                           lambda = exp(seq(10, 0, length = 100))),
                   trControl = ctrl1)

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



```r
lasso.fit$bestTune
```

```
##    alpha   lambda
## 39     1 46.45034
```

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

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                              s=46.45034
## (Intercept)               -4.872802e+06
## Gr_Liv_Area                6.564839e+01
## First_Flr_SF               7.903386e-01
## Second_Flr_SF              .
## Total_Bsmt_SF              3.535970e+01
## Low_Qual_Fin_SF           -4.121438e+01
## Wood_Deck_SF               1.174410e+01
## Open_Porch_SF              1.565541e+01
## Bsmt_Unf_SF               -2.089048e+01
## Mas_Vnr_Area               1.077520e+01
## Garage_Cars                4.122437e+03
## Garage_Area                8.065859e+00
## Year_Built                 3.238794e+02
## TotRms_AbvGrd             -3.679193e+03
## Full_Bath                 -3.977873e+03
## Overall_QualAverage       -4.903816e+03
## Overall_QualBelow_Average  -1.255837e+04
## Overall_QualExcellent       7.478188e+04
## Overall_QualFair           -1.086979e+04
## Overall_QualGood            1.216852e+04
## Overall_QualVery_Excellent  1.343708e+05
## Overall_QualVery_Good       3.792662e+04
## Kitchen_QualFair           -2.534577e+04
## Kitchen_QualGood           -1.765462e+04
## Kitchen_QualTypical        -2.572670e+04
## Fireplaces                  1.077660e+04
## Fireplace_QuFair           -7.718858e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.814710e+03
## Fireplace_QuPoor           -5.691040e+03
## Fireplace_QuTypical        -7.015719e+03
## Exter_QualFair             -3.456536e+04
## Exter_QualGood             -1.623783e+04
## Exter_QualTypical          -2.066184e+04
## Lot_Frontage                1.004202e+02
## Lot_Area                    6.044148e-01
## Longitude                  -3.344500e+04
## Latitude                    5.615697e+04
## Misc_Val                    8.546998e-01
## Year_Sold                  -5.839028e+02
```

```
# Test error
lasso.pred <- predict(lasso.fit, newdata = test)
mse.lasso  <- mean((lasso.pred - y.test)^2)
mse.lasso
```

```
## [1] 441875315
```

```
# 1SE rule
lasso.res        <- lasso.fit$results
threshold        <- min(lasso.res$RMSE) +
                    lasso.res$RMSESD[which.min(lasso.res$RMSE)] / sqrt(10)
lambda.1se.caret <- max(lasso.res$lambda[lasso.res$RMSE <= threshold])

coef.1se <- coef(lasso.fit$finalModel, s = lambda.1se.caret)
n.pred   <- sum(coef.1se != 0) - 1
cat("lambda (1SE):", round(lambda.1se.caret, 4), "\n")
```

```
## lambda (1SE): 580.3529
```

```
cat("Number of predictors (1SE):", n.pred, "\n")
```

```
## Number of predictors (1SE): 35
```

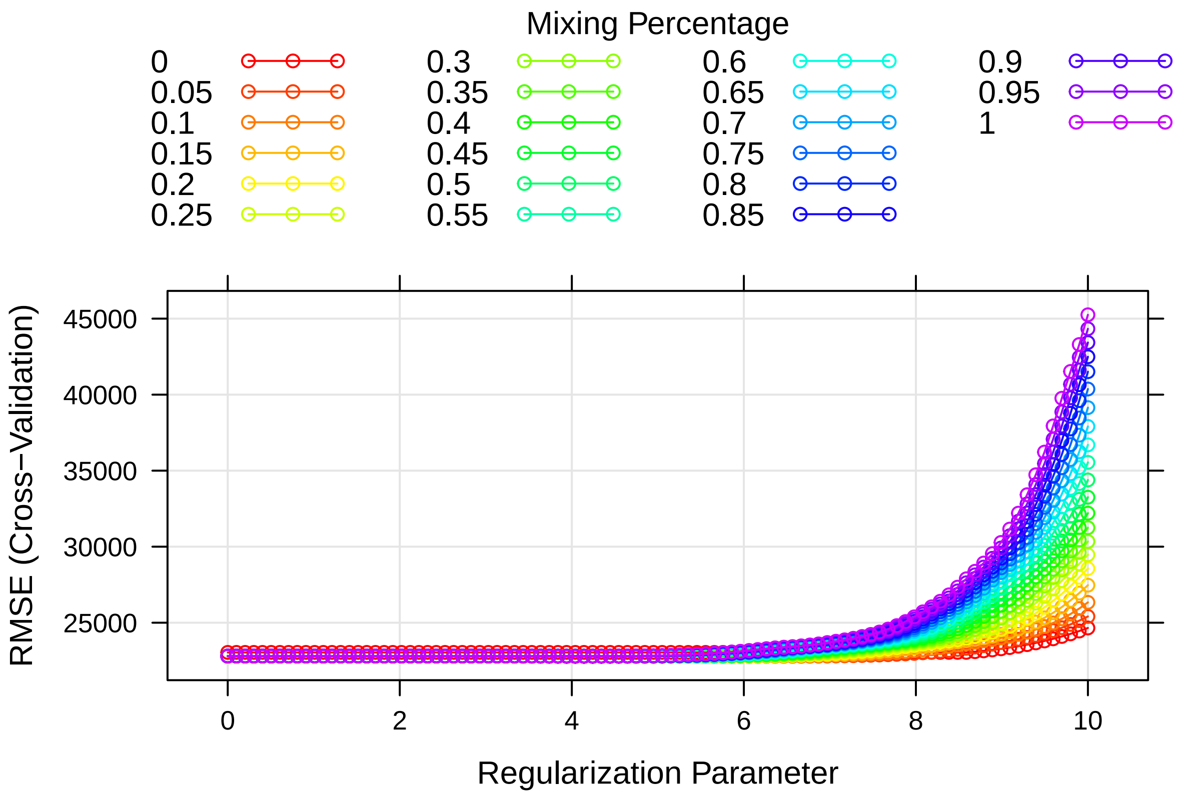
The selected tuning parameter (min CV) is $\lambda = 46.4503$, with a test MSE of 4.4187531×10^8. Under the 1SE rule ($\lambda = 580.3529$), **35 predictors** are included in the model.

(b) Elastic Net

```r
set.seed(2)
enet.fit <- train(Sale_Price ~ .,
                  data      = train,
                  method    = "glmnet",
                  tuneGrid  = expand.grid(alpha  = seq(0, 1, length = 21),
                                          lambda = exp(seq(10, 0, length = 100))),
                  trControl = ctrl1)

enet.fit$bestTune
```

```
##      alpha   lambda
## 357   0.15 286.1642
```

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



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

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                 s=286.1642
## (Intercept)                  -4.967207e+06
## Gr_Liv_Area                   4.405619e+01
## First_Flr_SF                  2.189055e+01
## Second_Flr_SF                 2.083174e+01
## Total_Bsmt_SF                 3.516175e+01
## Low_Qual_Fin_SF              -2.036525e+01
## Wood_Deck_SF                  1.199785e+01
## Open_Porch_SF                 1.620836e+01
## Bsmt_Unf_SF                  -2.081728e+01
## Mas_Vnr_Area                  1.120355e+01
## Garage_Cars                   4.085804e+03
## Garage_Area                   8.461457e+00
## Year_Built                    3.216183e+02
## TotRms_AbvGrd                -3.548411e+03
## Full_Bath                    -3.809514e+03
## Overall_QualAverage          -4.997161e+03
## Overall_QualBelow_Average    -1.261057e+04
## Overall_QualExcellent         7.538235e+04
## Overall_QualFair             -1.113660e+04
## Overall_QualGood              1.207073e+04
## Overall_QualVery_Excellent    1.355886e+05
## Overall_QualVery_Good         3.778627e+04
## Kitchen_QualFair             -2.456740e+04
## Kitchen_QualGood             -1.695293e+04
## Kitchen_QualTypical          -2.501140e+04
## Fireplaces                    1.078265e+04
## Fireplace_QuFair             -7.834379e+03
## Fireplace_QuGood              .
## Fireplace_QuNo_Fireplace      1.714726e+03
## Fireplace_QuPoor             -5.785251e+03
## Fireplace_QuTypical          -7.051527e+03
## Exter_QualFair               -3.344347e+04
## Exter_QualGood               -1.509552e+04
## Exter_QualTypical            -1.961356e+04
## Lot_Frontage                  1.001034e+02
## Lot_Area                      6.036008e-01
## Longitude                    -3.405942e+04
## Latitude                      5.664338e+04
## Misc_Val                      8.552979e-01
## Year_Sold                    -5.744607e+02
```

```r
enet.pred <- predict(enet.fit, newdata = test)
mse.enet  <- mean((enet.pred - y.test)^2)
mse.enet
```

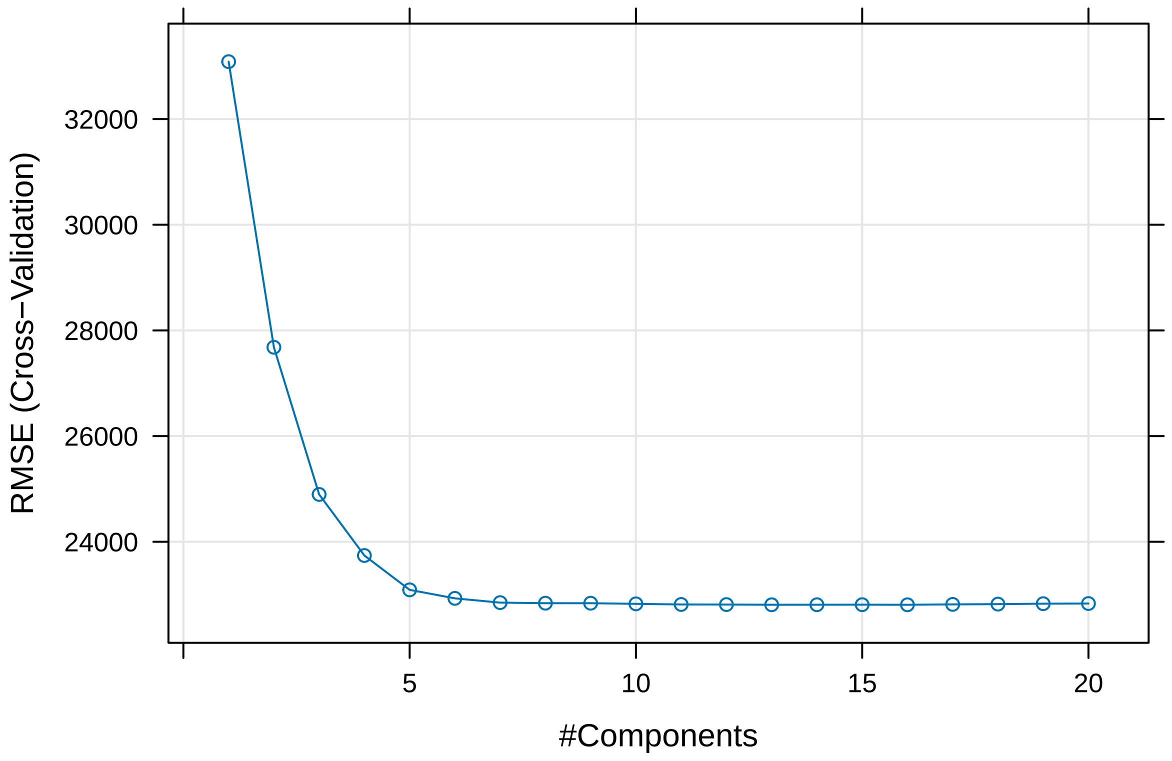
```
## [1] 439998442
```

The selected tuning parameters are $\alpha = 0.15$ and $\lambda = 286.1642$, with a test MSE of 4.3999844×10^8.

The 1SE rule is **not applicable** to elastic. Elastic net has two tuning parameters (α, λ).

(c) Partial Least Squares

```
set.seed(2)
pls.fit <- train(Sale_Price ~ .,
                 data       = train,
                 method     = "pls",
                 tuneGrid   = data.frame(ncomp = 1:20),
                 trControl  = ctrl1,
                 preProcess = c("center", "scale"))

plot(pls.fit)
```



```
pls.fit$bestTune
```

```
##    ncomp
## 16    16
```

```
pls.pred <- predict(pls.fit, newdata = test)
mse.pls  <- mean((pls.pred - y.test)^2)
mse.pls
```

```
## [1] 446775692
```

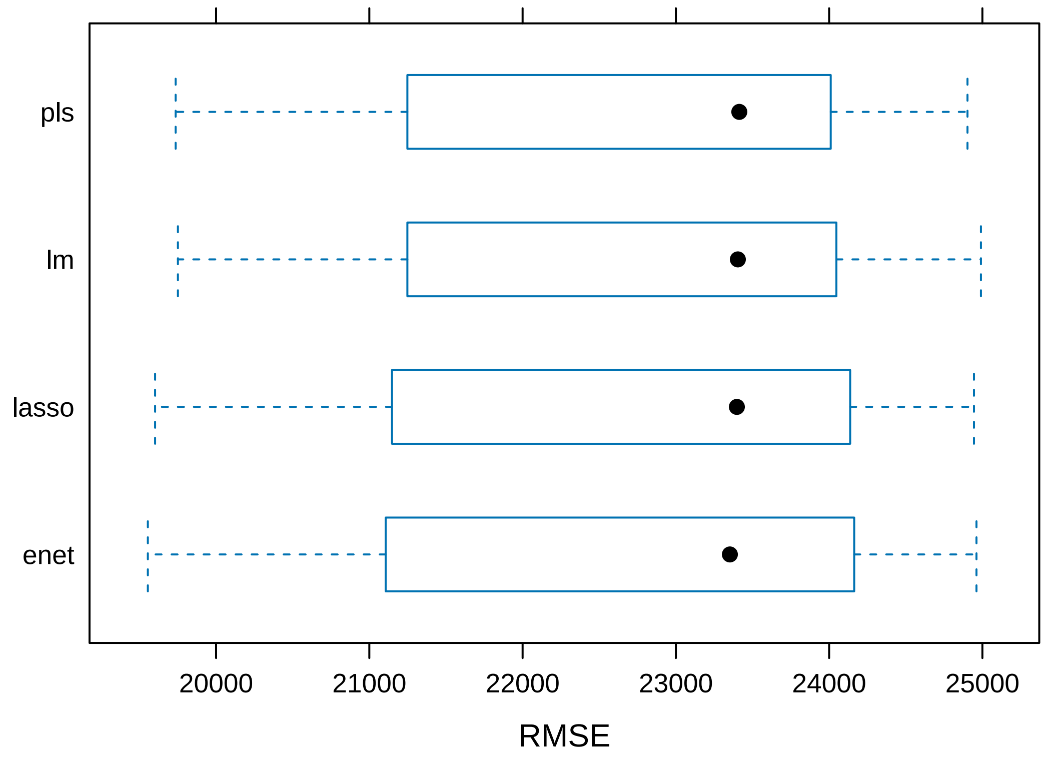
The PLS model includes **16 components**, with a test MSE of 4.4677569×10^8.

(d) Model Comparison

```r
set.seed(2)
lm.fit <- train(Sale_Price ~ .,
                data      = train,
                method    = "lm",
                trControl = ctrl1)

resamp <- resamples(list(lasso = lasso.fit,
                         enet  = enet.fit,
                         pls   = pls.fit,
                         lm    = lm.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, enet, pls, lm
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 14526.50 15737.08 16601.22 16598.73 17548.67 18443.98    0
## enet  14478.62 15705.19 16576.39 16578.81 17526.31 18436.99    0
## pls   14585.07 15807.81 16639.83 16629.39 17570.18 18442.95    0
## lm    14586.71 15828.63 16641.19 16639.99 17568.06 18439.70    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 19602.12 21578.44 23397.99 22793.99 24058.09 24945.08    0
## enet  19554.72 21551.89 23352.35 22792.47 24093.38 24961.37    0
## pls   19736.20 21657.28 23413.90 22807.42 23936.71 24902.72    0
## lm    19750.89 21655.50 23404.48 22839.27 23982.43 24990.50    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.8736637 0.8881870 0.9058714 0.9039043 0.9195235 0.9265603    0
## enet  0.8739298 0.8878946 0.9061908 0.9039321 0.9193882 0.9264909    0
## pls   0.8730979 0.8894494 0.9055091 0.9038628 0.9194429 0.9257294    0
## lm    0.8733064 0.8891481 0.9052586 0.9036505 0.9195059 0.9248318    0
```

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



```r
data.frame(
  Model    = c("Lasso", "Elastic Net", "PLS"),
  Test_MSE = round(c(mse.lasso, mse.enet, mse.pls), 2)
) |> knitr::kable()
```

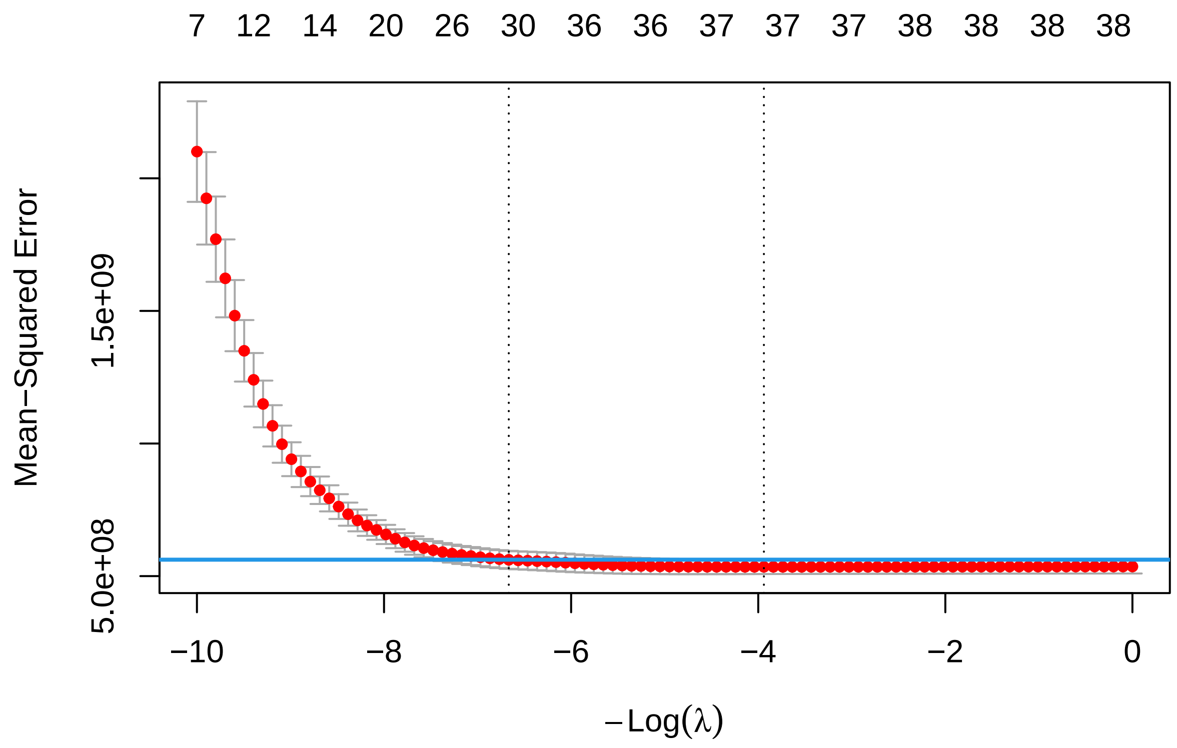
Model	Test_MSE
Lasso	441875315
Elastic Net	439998442
PLS	446775692

The **elastic net** is the best model. It has the lowest test error while providing flexibility over Lasso through the additional α parameter.

(e) Lasso: `caret` vs `glmnet`

```r
set.seed(2)
cv.lasso <- cv.glmnet(x, y,
                      alpha  = 1,
                      lambda = exp(seq(10, 0, length = 100)))

plot(cv.lasso)
```

```
abline(h = (cv.lasso$cvm + cv.lasso$cvsd)[which.min(cv.lasso$cvm)],
       col = 4, lwd = 2)
```



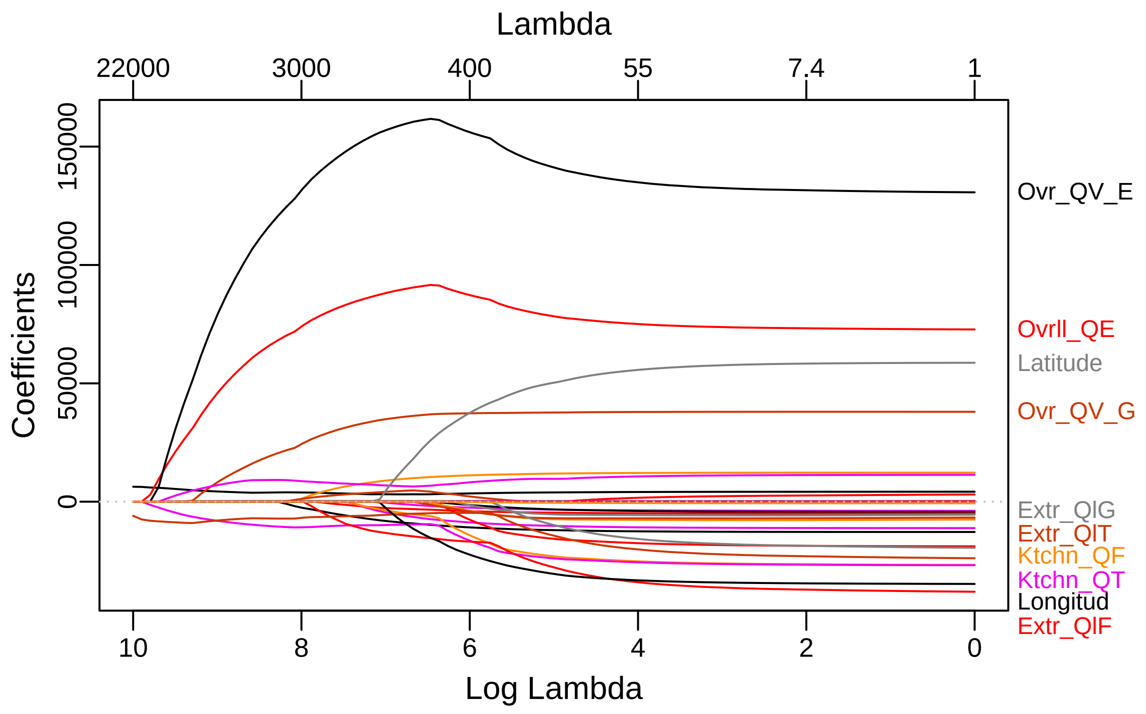
```
cv.lasso$lambda.min
```

```
## [1] 51.38745
```

```
cv.lasso$lambda.1se
```

```
## [1] 785.772
```

```
plot_glmnet(cv.lasso$glmnet.fit)
```



```r
predict(cv.lasso, s = "lambda.min", type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                lambda.min
## (Intercept)                 -4.858436e+06
## Gr_Liv_Area                  6.558518e+01
## First_Flr_SF                 7.943481e-01
## Second_Flr_SF                .
## Total_Bsmt_SF                3.537392e+01
## Low_Qual_Fin_SF             -4.115127e+01
## Wood_Deck_SF                 1.171129e+01
## Open_Porch_SF                1.559229e+01
## Bsmt_Unf_SF                 -2.088779e+01
## Mas_Vnr_Area                 1.080038e+01
## Garage_Cars                  4.107153e+03
## Garage_Area                  8.108573e+00
## Year_Built                   3.238466e+02
## TotRms_AbvGrd               -3.662693e+03
## Full_Bath                   -3.944089e+03
## Overall_QualAverage         -4.891844e+03
## Overall_QualBelow_Average   -1.253195e+04
## Overall_QualExcellent        7.492544e+04
## Overall_QualFair            -1.083724e+04
## Overall_QualGood             1.214989e+04
## Overall_QualVery_Excellent   1.346747e+05
## Overall_QualVery_Good       3.790534e+04
## Kitchen_QualFair           -2.526799e+04
## Kitchen_QualGood           -1.758893e+04
## Kitchen_QualTypical        -2.567062e+04
## Fireplaces                  1.071297e+04
## Fireplace_QuFair           -7.704585e+03
## Fireplace_QuGood            .
## Fireplace_QuNo_Fireplace    1.712992e+03
## Fireplace_QuPoor           -5.678637e+03
## Fireplace_QuTypical        -7.014304e+03
## Exter_QualFair             -3.423120e+04
## Exter_QualGood             -1.592280e+04
## Exter_QualTypical          -2.035264e+04
## Lot_Frontage                1.002152e+02
## Lot_Area                    6.044007e-01
## Longitude                  -3.329199e+04
## Latitude                    5.585746e+04
## Misc_Val                    8.479681e-01
## Year_Sold                  -5.777901e+02
```

```r
pred.glmnet <- as.numeric(predict(cv.lasso, newx = x.test, s = "lambda.min"))
mse.glmnet  <- mean((pred.glmnet - y.test)^2)

data.frame(
  Package  = c("caret", "glmnet"),
  Lambda   = round(c(lasso.fit$bestTune$lambda, cv.lasso$lambda.min), 4),
  Test_MSE = round(c(mse.lasso, mse.glmnet), 2)
) |> knitr::kable()
```

Package	Lambda	Test_MSE
caret	46.4503	441875315
glmnet	51.3874	441403812

`caret` searches only the user-supplied grid; While `glmnet` generates its own data-driven sequence starting from $\lambda_{\max}$. Despite these difference, both approaches yield similar test errors