P8106 - HW1

Tucker Morgan, tlm2152

2/10/2022

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

house_trn <- read_csv(file = "./Homework 1/housing_training.csv") %>%
  janitor::clean_names() %>%
  na.omit()

house_tst <- read_csv(file = "./Homework 1/housing_test.csv") %>%
  janitor::clean_names() %>%
  na.omit()
```

In this analysis, we will predict the price of a house based on its characteristics. To begin, I have loaded in a training data set, `house_trn`, that consists of 1440 training observations and a test data set, `house_tst`, that contains 959 test observations. Each data set includes 26 variables: `sale_price` along with 25 predictors.

Part A

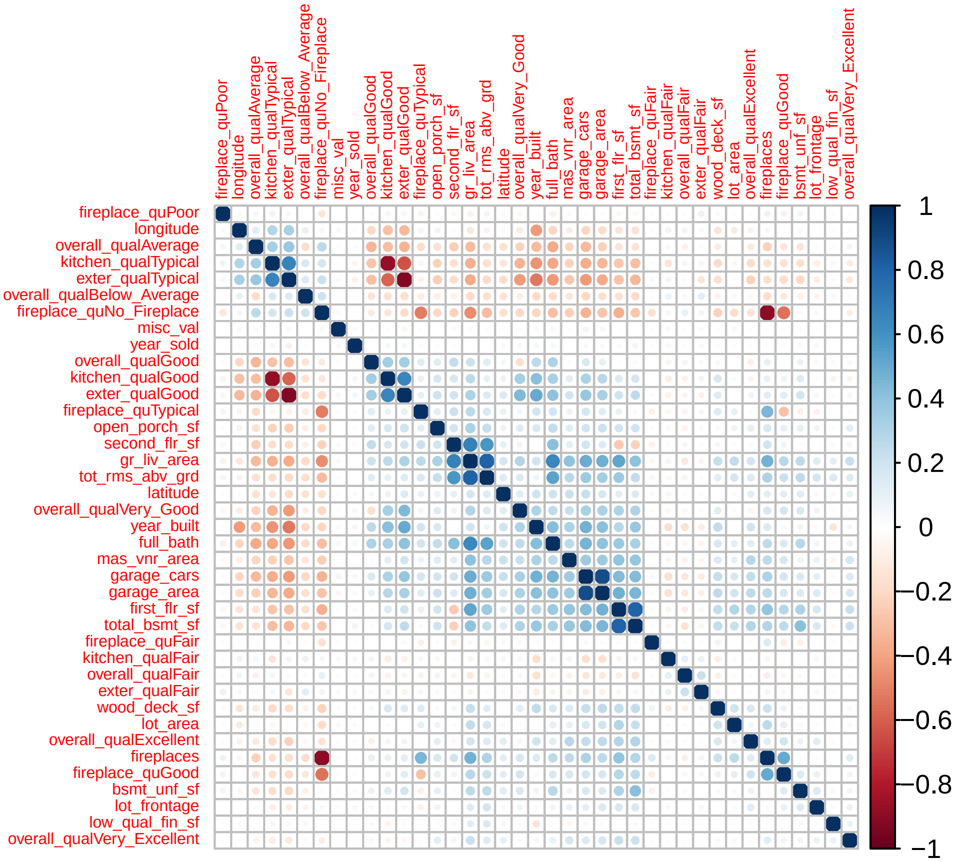
First, we will use the training data set to fit a linear model using least squares. We will use a cross-validation technique on the training data.

```
set.seed(100)
# setting up ten-fold CV, repeated five times
ctrl1 <- trainControl(method = "repeatedcv",
                      repeats = 5,
                      number = 10)

lm_fit <- train(sale_price ~ .,
                data = house_trn,
                method = "lm",
                trControl = ctrl1,
                preProcess = "scale")

mean(lm_fit$resample$RMSE) # cross-validation RMSE
```

```
## [1] 23039.1
```

This model is easy to fit and to interpret, but there are some issues that can arise when using least-squares linear regression. It is possible that some predictor terms truly share non-linear relationships with the response variable. For example in this exercise, square footage in a house might increase price dramatically from 1,500 to 2,000 ft^2, but the effect may be less pronounced when increasing from 3,000 to 3,500. Another issue that can arise in least-squares is that some predictors may be collinear, or have high correlation, with each other. We can check this with the correlation plot below.

```
# creating model matrix of predictors
house_trn_pred <- model.matrix(sale_price ~ ., house_trn)[,-1]

corrplot(cor(house_trn_pred),
         method = "circle",
         type = "full",
         tl.cex = 0.5,
         order = "hclust")
```



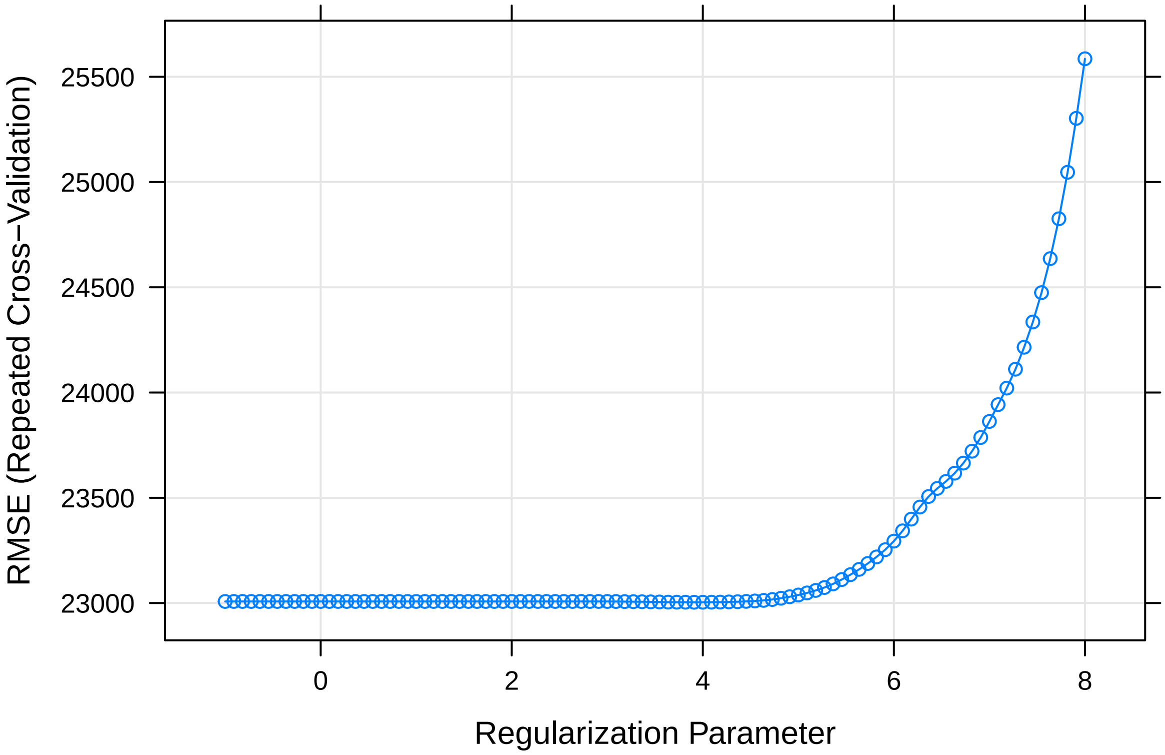
As we can see in the correlation plot, some variables like `kitchen_qual` and `exter_qual` or `first_flr_sf` and `total_bsmt_sf` seem to have high correlation to each other. It is difficult to separate the individual effects of these collinear variables. To get around this issue, we could drop one of the collinear variables or combine the two into one new predictor.

Part B

Next, we will fit a lasso model on the training data.

```
set.seed(100)
# creating another control for the 1se rule lasso
```

```
ctrl2 <- trainControl(method = "repeatedcv",
                      repeats = 5,
                      number = 10,
                      selectionFunction = "oneSE")

# creating vector of response variable
house_trn_price <- house_trn$sale_price

lasso_fit <- train(x = house_trn_pred,
                   y = house_trn_price,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(8, -1, length = 100))),
                   trControl = ctrl2,
                   preProcess = c("center", "scale"))

plot(lasso_fit, xTrans = log)
```



```
lasso_fit$bestTune # 1se lambda value
```

```
##    alpha   lambda
## 76     1 336.3599
```

```r
# creating model matrix of predictors
house_tst_mat <- model.matrix(sale_price ~ ., house_tst)[,-1]

# creating predictions using lasso model
lasso_tst_predict <- predict(lasso_fit, newdata = house_tst_mat)

# calculating test RMSE
postResample(pred = lasso_tst_predict, obs = house_tst$sale_price) %>%
  knitr::kable()
```

	x
RMSE	2.055714e+04
Rsquared	9.028134e-01
MAE	1.505168e+04

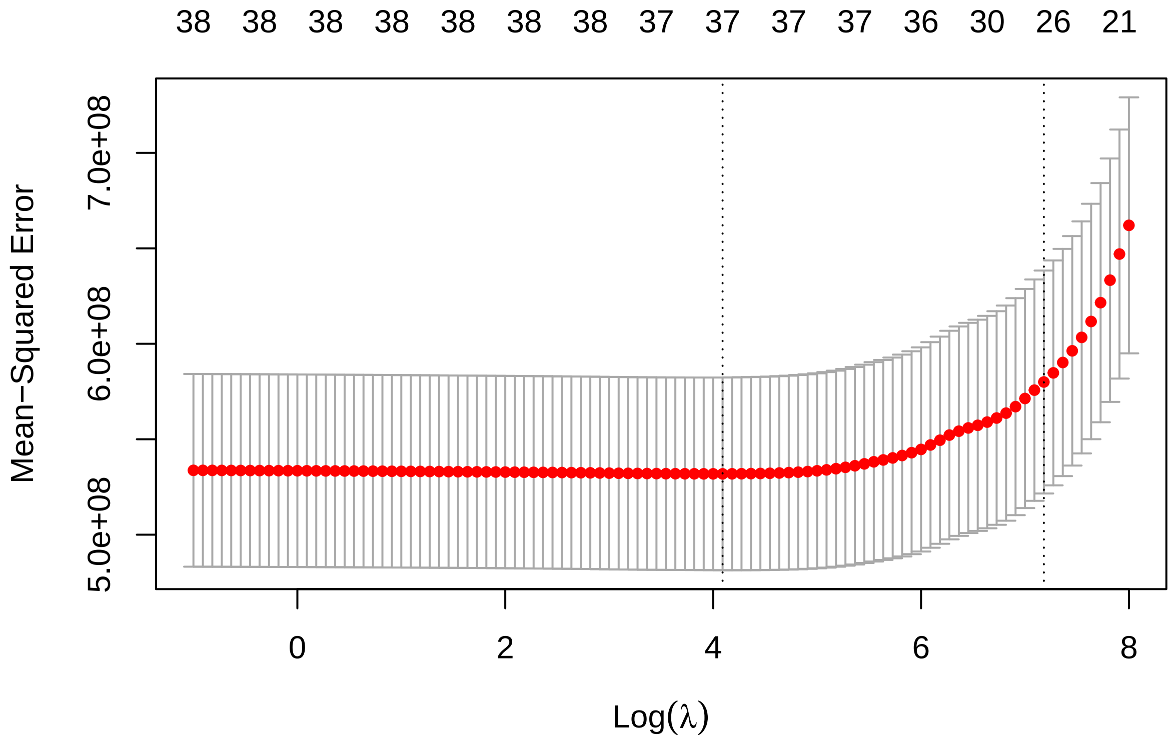
The test RMSE noted above is around 20,500 with a tuning parameter (lambda) value of 336. Below are the 37 coefficients in the model with this tuning parameter.

```r
coef(lasso_fit$finalModel, lasso_fit$finalModel$lambdaOpt)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                 177568.50208
## gr_liv_area                  29933.71078
## first_flr_sf                   340.07337
## second_flr_sf                     .
## total_bsmt_sf                14953.52327
## low_qual_fin_sf              -1638.11571
## wood_deck_sf                  1384.95299
## open_porch_sf                  812.40144
## bsmt_unf_sf                  -8588.13614
## mas_vnr_area                  2128.83391
## garage_cars                   2600.55063
## garage_area                   1914.98269
## year_built                    9298.06465
## tot_rms_abv_grd              -4158.59861
## full_bath                    -1004.14952
## overall_qualAverage          -1911.25254
## overall_qualBelow_Average    -2888.30522
## overall_qualExcellent        14424.51828
## overall_qualFair             -1125.87374
## overall_qualGood              4605.83789
## overall_qualVery_Excellent  14572.57365
## overall_qualVery_Good        11429.00005
## kitchen_qualFair             -2161.32044
## kitchen_qualGood             -4955.23750
## kitchen_qualTypical          -9349.85198
## fireplaces                    5650.99399
## fireplace_quFair              -816.09155
## fireplace_quGood               637.50900
## fireplace_quNo_Fireplace          .
## fireplace_quPoor             -342.98853
## fireplace_quTypical         -2070.01681
## exter_qualFair              -1756.91793
## exter_qualGood                -14.90519
## exter_qualTypical           -2315.11851
## lot_frontage                 2900.05089
## lot_area                     4941.05083
## longitude                    -650.69049
## latitude                      747.72297
## misc_val                      252.56840
## year_sold                    -297.12827
```

Below is a similar process using `glmnet` instead of `caret`.

```
# creating model matrix of predictors for glmnet
house_trn_pred <- model.matrix(sale_price ~ ., house_trn)[,-1]
# creating vector of response variable
house_trn_price <- house_trn$sale_price

cv_lasso <- cv.glmnet(x = house_trn_pred,
                      y = house_trn_price,
                      alpha = 1,
                      lambda = exp(seq(8, -1, length = 100)))
plot(cv_lasso)
```



```r
cv_lasso$lambda.1se
```

```
## [1] 1315.298
```

```r
# creating model matrix of test predictors
house_tst_pred <- model.matrix(sale_price ~ ., house_tst)[,-1]
# creating a vector of test set observations
house_tst_price <- house_tst$sale_price

test_perf <- assess.glmnet(cv_lasso,
                           newx = house_tst_pred,
                           newy = house_tst_price,
                           s = "lambda.1se")
```

The test RMSE for the lasso model is approximately 2.079e+04 with the “1se” lambda value of 1315.3.

```r
predict(cv_lasso, s = "lambda.1se", type = "coefficients")
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                               lambda.1se
## (Intercept)                -5.999732e+05
## gr_liv_area                 5.347386e+01
## first_flr_sf                1.416800e+00
## second_flr_sf               .           
## total_bsmt_sf               3.656018e+01
## low_qual_fin_sf            -1.639094e+01
## wood_deck_sf                7.324484e+00
## open_porch_sf               4.820064e+00
## bsmt_unf_sf                -1.745468e+01
## mas_vnr_area                1.463122e+01
## garage_cars                 3.306303e+03
## garage_area                 1.200709e+01
## year_built                  3.262336e+02
## tot_rms_abv_grd             .           
## full_bath                   .           
## overall_qualAverage        -2.297241e+03
## overall_qualBelow_Average  -7.361731e+03
## overall_qualExcellent       8.641791e+04
## overall_qualFair           -3.028855e+03
## overall_qualGood            8.101184e+03
## overall_qualVery_Excellent  1.541066e+05
## overall_qualVery_Good       3.374993e+04
## kitchen_qualFair           -2.424008e+03
## kitchen_qualGood            .           
## kitchen_qualTypical        -9.965604e+03
## fireplaces                  7.152757e+03
## fireplace_quFair            .           
## fireplace_quGood            3.785450e+03
## fireplace_quNo_Fireplace    .           
## fireplace_quPoor            .           
## fireplace_quTypical         .           
## exter_qualFair             -1.202492e+04
## exter_qualGood                .
## exter_qualTypical            -5.792050e+03
## lot_frontage                  5.082190e+01
## lot_area                      5.054348e-01
## longitude                     .
## latitude                      .
## misc_val                      .
## year_sold                     .
```

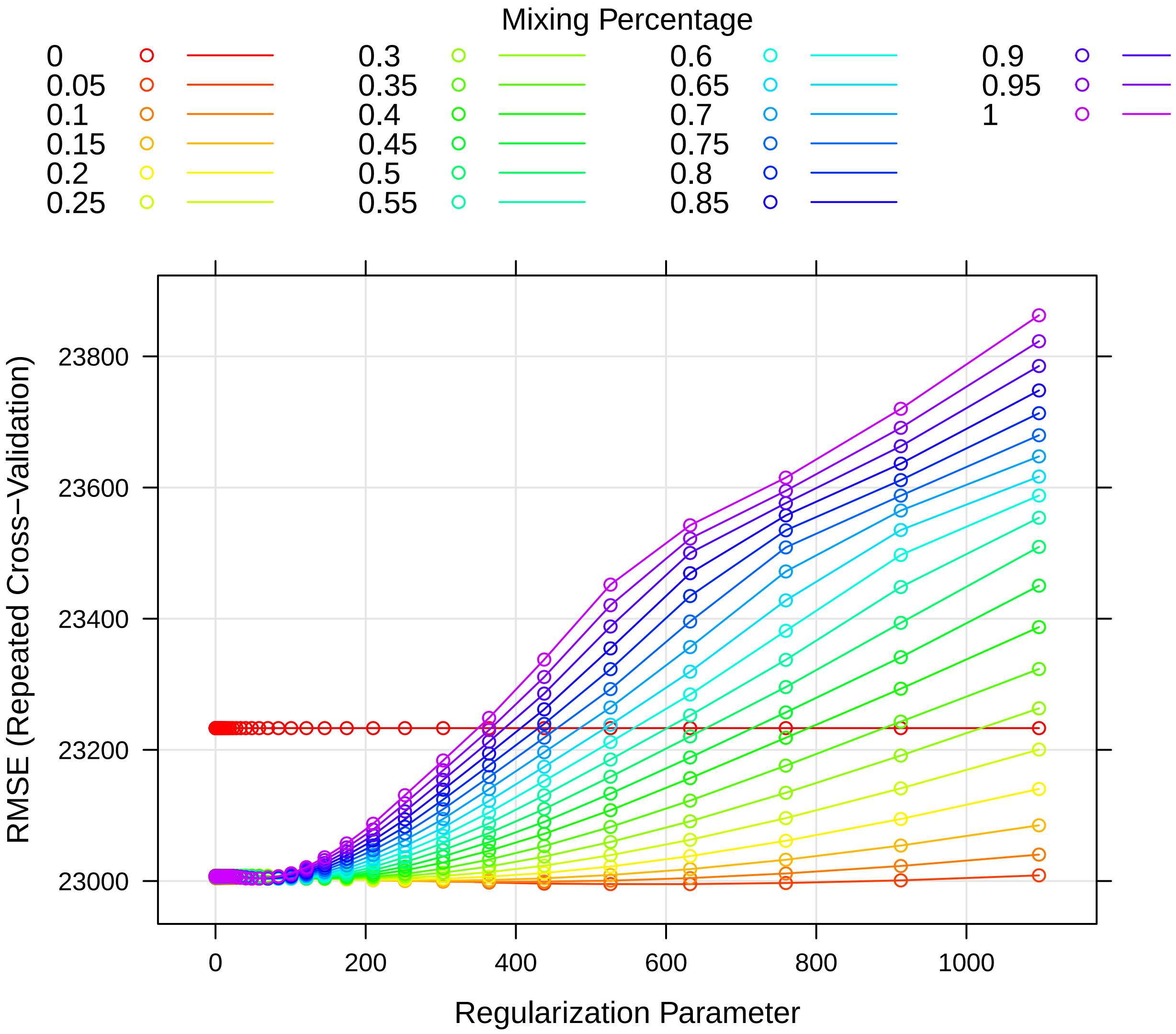
There are 26 coefficients and the intercept in the lasso model when the “1se” rule is used. This lambda value can be seen in the plot above, shown by the right-most dashed line.

Part C

Now, we will fit an elastic net model on the training data set.

```
set.seed(100)
enet_fit <- train(x = house_trn_pred,
                  y = house_trn_price,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(7, -2, length = 50))),
                  trControl = ctrl1)
```

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

plot(enet_fit, par.settings = myPar)
```



```
# coefficients of the elastic net model
coef(enet_fit$finalModel, enet_fit$finalModel$lambdaOpt)
```

```
## 40 x 1 sparse Matrix of class "dgCMatrix"
##                                     s1
## (Intercept)              -5.121373e+06
## gr_liv_area               3.869889e+01
## first_flr_sf              2.666395e+01
## second_flr_sf             2.538014e+01
## total_bsmt_sf             3.492069e+01
## low_qual_fin_sf          -1.586739e+01
## wood_deck_sf              1.235126e+01
## open_porch_sf             1.693644e+01
## bsmt_unf_sf              -2.070820e+01
## mas_vnr_area              1.174573e+01
## garage_cars               4.039665e+03
## garage_area                 8.967288e+00
## year_built                  3.187168e+02
## tot_rms_abv_grd            -3.412076e+03
## full_bath                  -3.648167e+03
## overall_qualAverage        -5.119758e+03
## overall_qualBelow_Average  -1.269567e+04
## overall_qualExcellent       7.596050e+04
## overall_qualFair           -1.148865e+04
## overall_qualGood            1.194995e+04
## overall_qualVery_Excellent  1.366564e+05
## overall_qualVery_Good       3.761025e+04
## kitchen_qualFair           -2.348985e+04
## kitchen_qualGood           -1.592850e+04
## kitchen_qualTypical        -2.398308e+04
## fireplaces                  1.078709e+04
## fireplace_quFair           -7.865488e+03
## fireplace_quGood            1.470124e+02
## fireplace_quNo_Fireplace    1.736309e+03
## fireplace_quPoor           -5.811255e+03
## fireplace_quTypical        -6.964621e+03
## exter_qualFair             -3.269973e+04
## exter_qualGood             -1.429848e+04
## exter_qualTypical          -1.891697e+04
## lot_frontage                9.998943e+01
## lot_area                    6.029439e-01
## longitude                  -3.518902e+04
## latitude                    5.767178e+04
## misc_val                    8.641384e-01
## year_sold                  -5.700070e+02
```

The selected tuning parameters and test RMSE can be seen in the tables below, corresponding to the minimum value in the plot above.

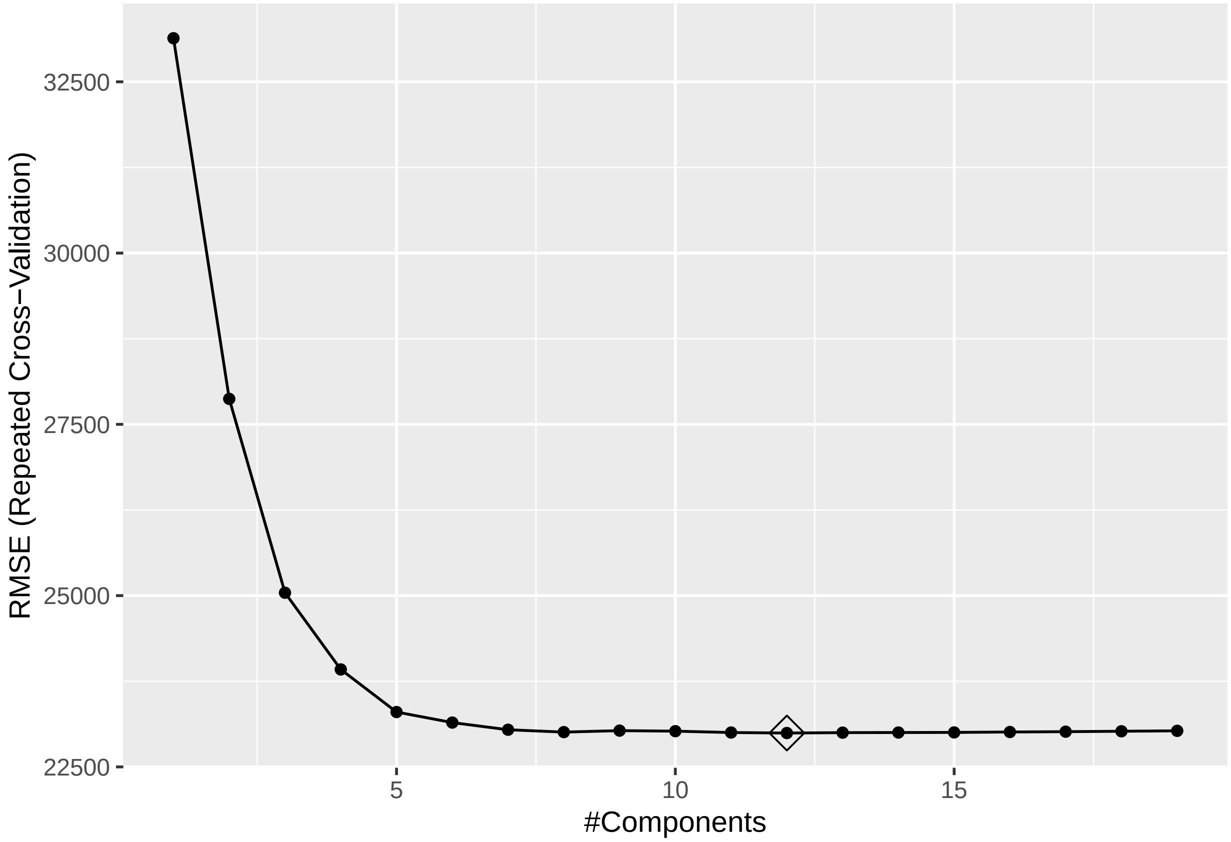
	alpha	lambda
97	0.05	632.057

	x
RMSE	2.093031e+04
Rsquared	8.997484e-01
MAE	1.531767e+04

Part D

Next, we will fit a partial least squares model on the training data.

```
set.seed(100)
pls_fit <- train(x = house_trn_pred,
                 y = house_trn_price,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:19),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))

ggplot(pls_fit, highlight = TRUE)
```



Based on the plot above, the model with the minimum RMSE has 12 components.

```
# creating predictions using the pls model
pls_tst_predict <- predict(pls_fit, newdata = house_tst_mat)

#calculating RMSE
postResample(pred = pls_tst_predict, obs = house_tst$sale_price) %>%
  knitr::kable()
```

	x
RMSE	2.120431e+04
Rsquared	8.975121e-01
MAE	1.555218e+04

Part E

We can assess the various models using the `resamples` function. This function analyzes a set of resampling results from the various models on a common data set. Although we found the test error in the preceding sections, it is typically best practice to train models on training data only and compare performance based on cross-validation RMSE rather than test RMSE. This being the case, and our goal being prediction, we will want to choose the model with the lowest mean RMSE from the following output.

```
resamp <- resamples(list(linear = lm_fit,
                         lasso = lasso_fit,
                         enet = enet_fit,
                         pls = pls_fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: linear, lasso, enet, pls
## Number of resamples: 50
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA’s
## linear 14455.54 16038.97 16691.50 16725.97 17321.33 19270.47    0
## lasso  14453.01 15912.78 16740.02 16650.44 17089.73 19434.43    0
## enet   14321.89 15930.15 16627.12 16630.65 17210.82 19227.87    0
## pls    14485.54 16034.32 16716.82 16717.60 17304.11 19234.29    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA’s
## linear 20063.41 21887.41 22767.91 23039.10 24113.02 29258.68    0
## lasso  20195.30 22169.30 22934.73 23218.92 24007.31 30597.39    0
## enet   20032.18 21930.94 22693.09 22995.32 24000.56 29570.99    0
## pls    20069.96 21865.43 22749.12 22994.27 23947.79 29154.53    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA’s
## linear 0.8407624 0.8956820 0.9028378 0.9036628 0.9138771 0.9294587    0
## lasso  0.8406124 0.8943710 0.9027326 0.9021598 0.9130797 0.9324043    0
## enet   0.8415879 0.8957525 0.9032866 0.9039992 0.9141009 0.9326765    0
## pls    0.8402265 0.8956822 0.9028180 0.9039302 0.9147384 0.9322010    0
```

The elastic net and partial least squares models have very similar RMSE values. If `enet_fit` contained fewer coefficients, we may want to select this model based on the principle of parsimony. However, as can be seen in section (c), the `enet_fit` model maintains all of the possible predictors. For this reason, I would choose the `pls_fit` model fot predicting the response because it has the lowest mean RMSE.