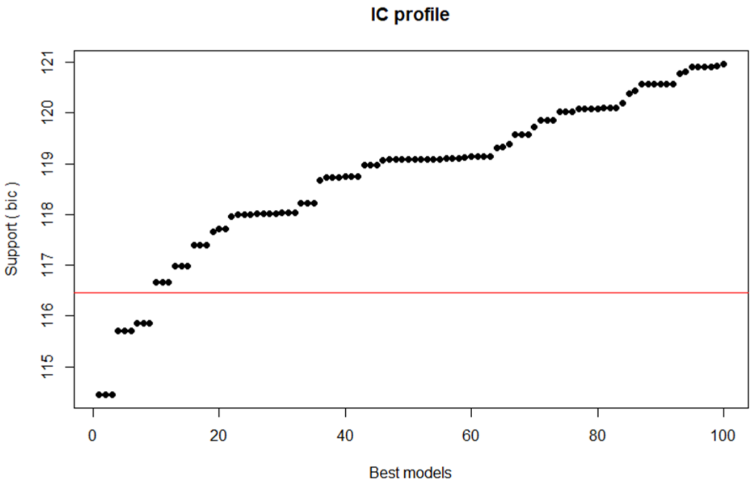
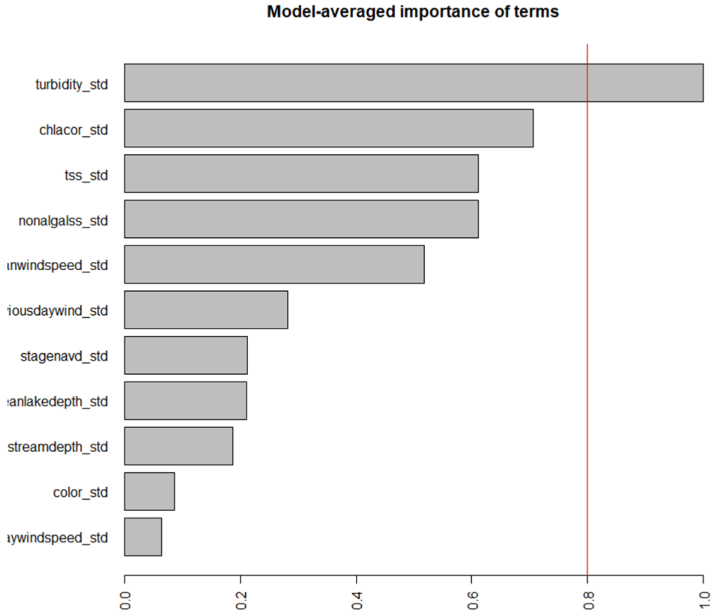
Advanced R:
Statistical Machine Learning

Dan Schmutz, MS
Chief Environmental Scientist

Zoom Workshop for SJRWMD
September 24, 2020





Regularized Regression: Ridge, Lasso, and Elastic Net

Regularization introduces some bias to reduce variance

- In statistical machine learning we frequently encounter datasets with multicollinearity: intercorrelated variables.
- Multicollinearity makes regression coefficients unstable: high variance between datasets.
- Normal regression seeks to minimize squared residuals
- Regularized regression adds either an L1 or L2 penalty to the regression coefficients (or a mixture of both).

GPI

Ridge (L2 regularization)

- As lambda increases, the coefficients are forced to be smaller, introducing bias, but decreasing variance

$$\text{minimize}\left(SSE + \lambda \sum_{j=1}^{p} \beta_j^2\right)$$

GPI

Lasso (L1 Regularlization)

- As lambda increases, the coefficients are forced to go to zero, effectively making the lasso a form of variable selection

$$\text{minimize}\left(SSE + \lambda\sum_{j=1}^{p}|\beta_j|\right)$$

GPI

Elastic Net Regularization

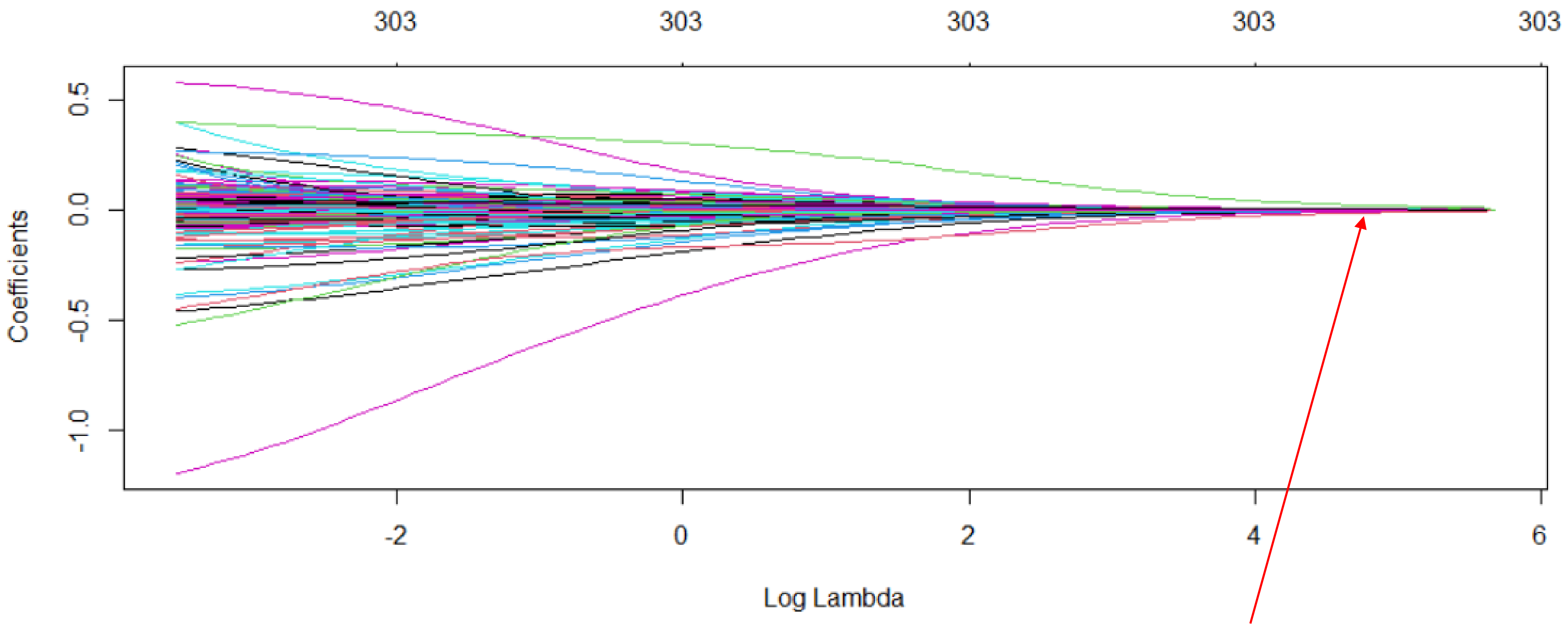
- Two hyper parameters, lambda you know, and alpha which controls the mixing ratio of L2 and L1

$$\text{minimize}\left(SSE + \lambda_1 \sum_{j=1}^{p} \beta_j^2 + \lambda_2 \sum_{j=1}^{p} |\beta_j|\right)$$

Ridge on ames data

```r
# Create training feature matrices
# we use model.matrix(...)[, -1] to discard the intercept
X <- model.matrix(Sale_Price ~ ., train_3)[, -1]
Xnotused<-model.matrix(Sale_Price ~.,train_3)

# transform y with log transformation
Y <- log(train_3$Sale_Price)

# Apply ridge regression to ames data
ridge <- glmnet(
  x = X,
  y = Y,
  alpha = 0
)

plot(ridge, xvar = "lambda")

```

Ridge on ames data



Higher lambda values crush the coefficients down to near but not reaching zero.

Ridge on ames data

- glmnet chose range of 100 lambdas to investigate
- Compare coefficients at the largest and smallest lambdas

```
> ridge$lambda %>% head()
[1] 291.4724 265.5788 241.9855 220.4882 200.9006 183.0532
> # small lambda results in large coefficients
> coef(ridge)[c("Latitude", "Overall_QualVery_Excellent"), 100]
                 Latitude Overall_QualVery_Excellent
                0.3950246                  0.1326520
> # large lambda results in small coefficients
> coef(ridge)[c("Latitude", "Overall_QualVery_Excellent"), 1]
                 Latitude Overall_QualVery_Excellent
             6.203488e-36               9.744773e-37
```

What is the optimal lamda for inference on out-of-sample data?

- Glmnet offers built-in crossvalidation (10-fold)

```r
# Apply CV ridge regression to ames data
ridge <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 0
)
```

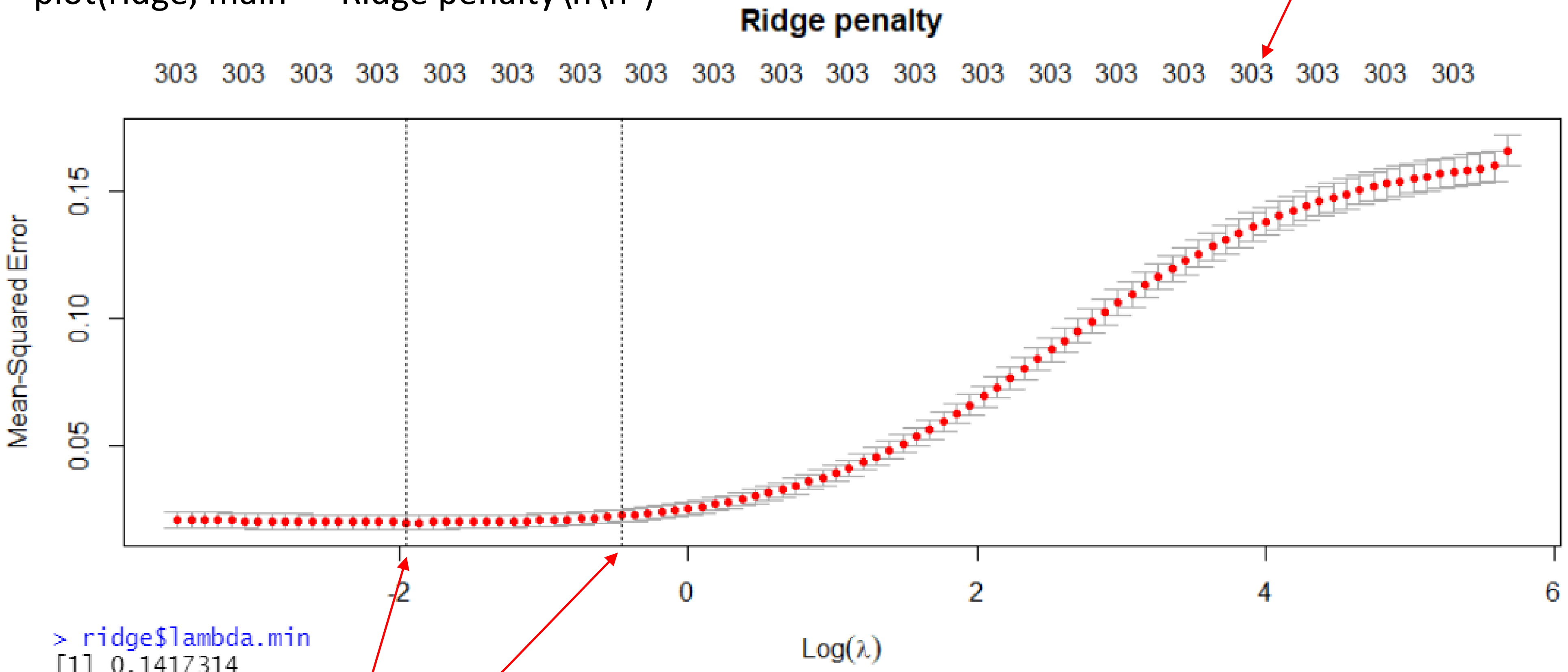
- Str(ridge)

```
  ..- attr(*, "class")= chr [1:2] "elnet" "glmnet"
 $ lambda.min: num 0.142
 $ lambda.1se: num 0.628
 - attr(*, "class")= chr "cv.glmnet"
```

crossvalidated results for ridge penalty

All 303 coefficients stay in model.

plot(ridge, main = "Ridge penalty\n\n")




```
> ridge$lambda.min
[1] 0.1417314
> log(ridge$lambda.min)
[1] -1.953821
> ridge$lambda.1se
[1] 0.6279583
> log(ridge$lambda.1se)
[1] -0.4652815
>
```

GPI

cv-selected ridge performance on train and test

- Looks much better than the glmulti-selected model (which only included numeric variables)

```
> # compute RMSE of transformed predicted
> RMSE(exp(predridge), exp(Y))
[1] 28870.87
> # can we run it on test?
> Xtest <- model.matrix(Sale_Price ~ ., test_3)[, -1]
> predridgetest <- predict(ridge, Xtest)
> Ytest <- log(test_3$Sale_Price)
> RMSE(exp(predridgetest), exp(Ytest))
[1] 43656.97
```

s Value(s) of the penalty parameter `lambda` at which predictions are required. Default is the value `s="lambda.1se"` stored on the CV `object`. Alternatively `s="lambda.min"` can be used. If `s` is numeric, it is taken as the value(s) of `lambda` to be used. (For historical reasons we use the symbol 's' rather than 'lambda' to reference this parameter)

Is lamda.min or lambda.1se better for inference on out-of-sample data?

- This is a question about parsimony. The lambda.min option refers to value of λ at the lowest CV error. The error at this value of λ is the average of the errors over the k folds and hence this estimate of the error is uncertain. The lambda.1se represents the value of λ in the search that was simpler than the best model (lambda.min), but which has error within 1 standard error of the best model. In other words, using the value of lambda.1se as the selected value for λ results in a model that is slightly simpler than the best model but which cannot be distinguished from the best model in terms of error given the uncertainty in the k-fold CV estimate of the error of the best model.

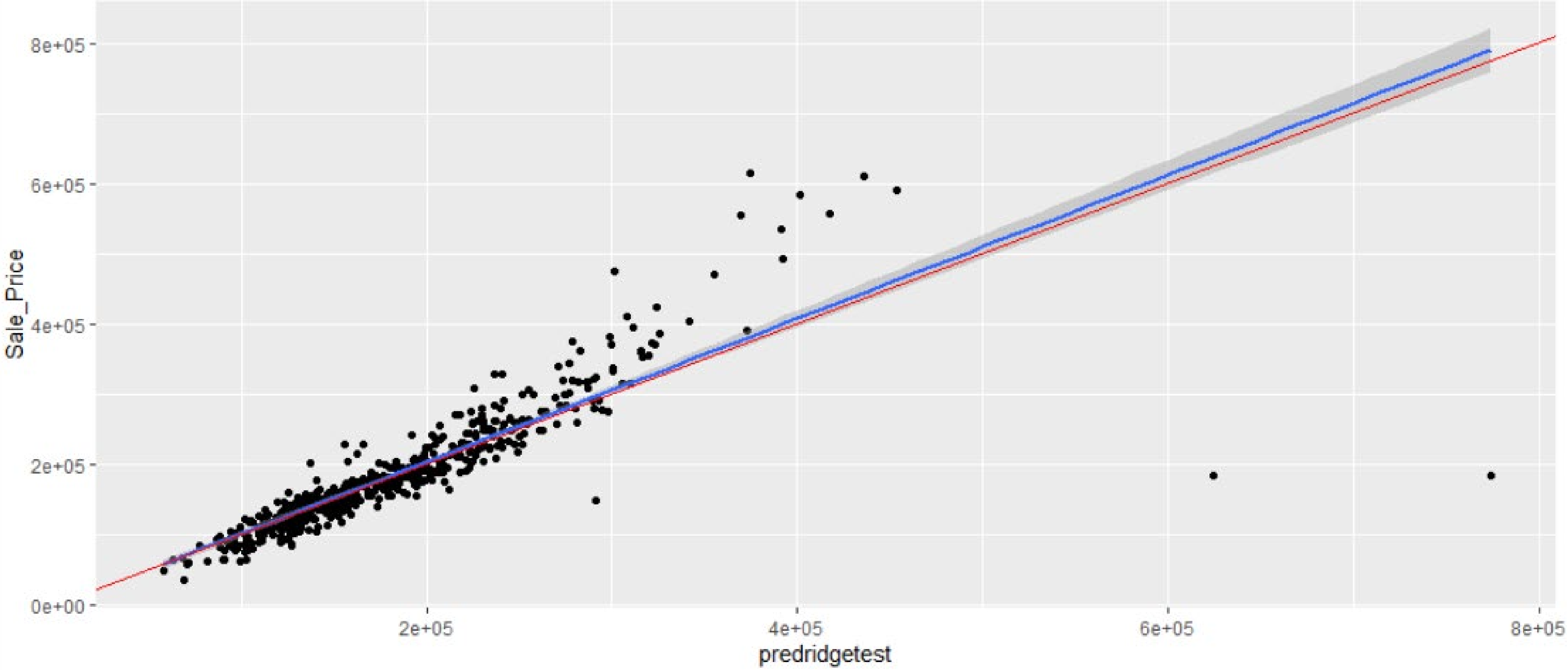
- The choice is yours:

- The best model that may be too complex of slightly overfitted: lambda.min
- The simplest model that has comparable error to the best model given the uncertainty: lambda.1se

https://stats.stackexchange.com/questions/70249/feature-selection-model-with-glmnet-on-methylation-data-pn

GPI

Obs vs. exp plot for ames test_3

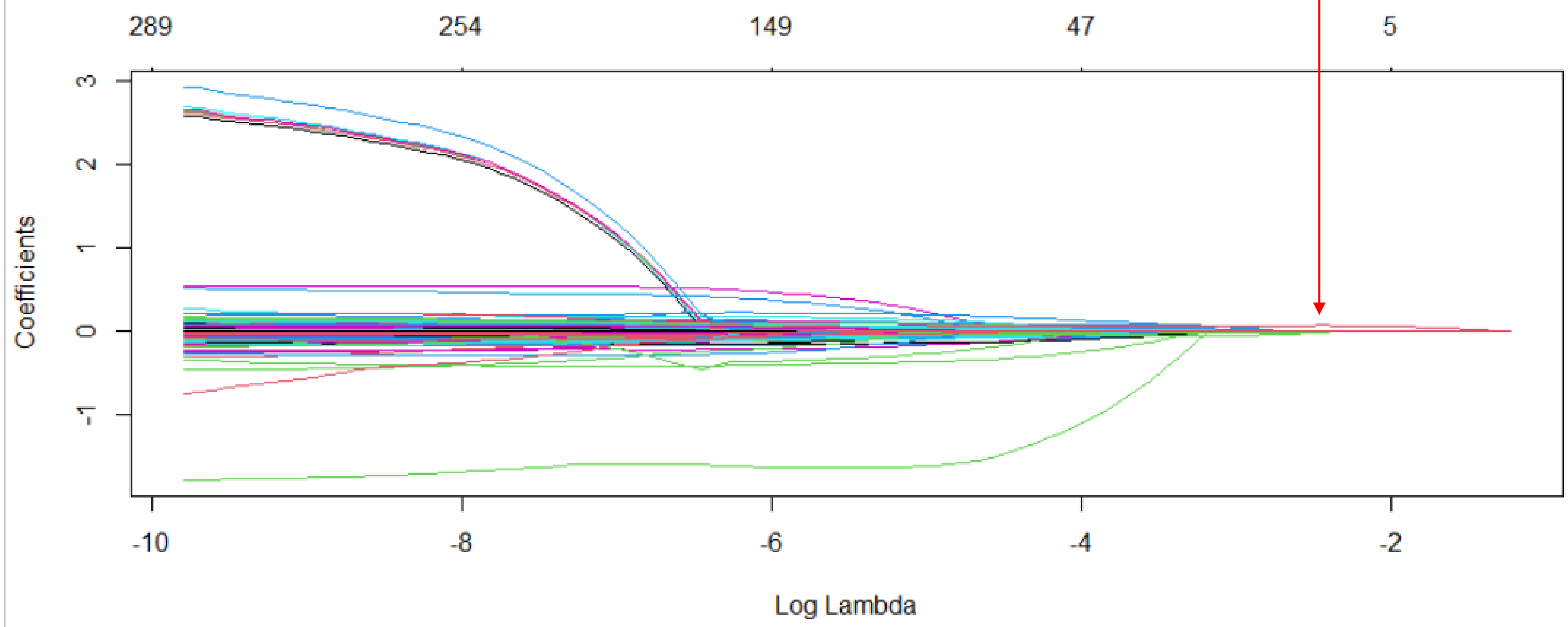



```
> cor(test_3aug$predridgetest,test_3aug$Sale_Price)^2
          [,1]
1 0.7264753
> res_predridgetest<-test_3aug$Sale_Price-test_3aug$predridgetest
> (mean((res_predridgetest)^2))^0.5
[1] 43656.97
```

Lasso on ames data

```
#lasso on ames data

lasso <- glmnet(
  x = X,
  y = Y,
  alpha = 1
)

plot(lasso, xvar = "lambda")
```

Higher lambda values squeeze coefficients down to zero, creating a variable selection situation.



GPI

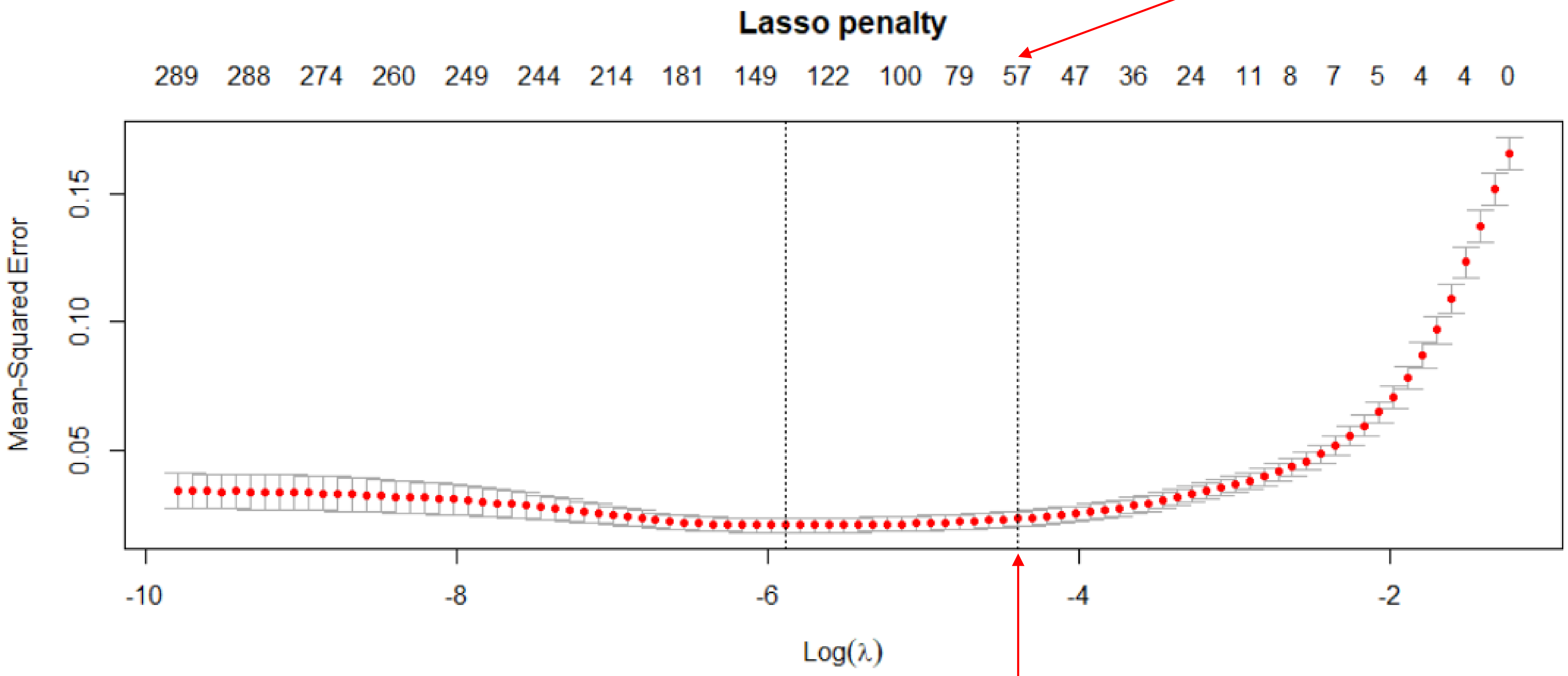
Lasso on ames data

```r
#lasso on ames data

# Apply CV lasso regression to Ames data
lasso <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 1
)
plot(lasso, main = "Lasso penalty\n\n")
```

GPI

Lasso on ames data

Note coefficients being forced to 0 with L1 regularization, 57 left in model at 1SE rule




```
> lasso$lambda.1se
[1] 0.01352895
> #[1] 0.6279583
> log(lasso$lambda.1se)
[1] -4.302923
> #[1] -0.4652815
```

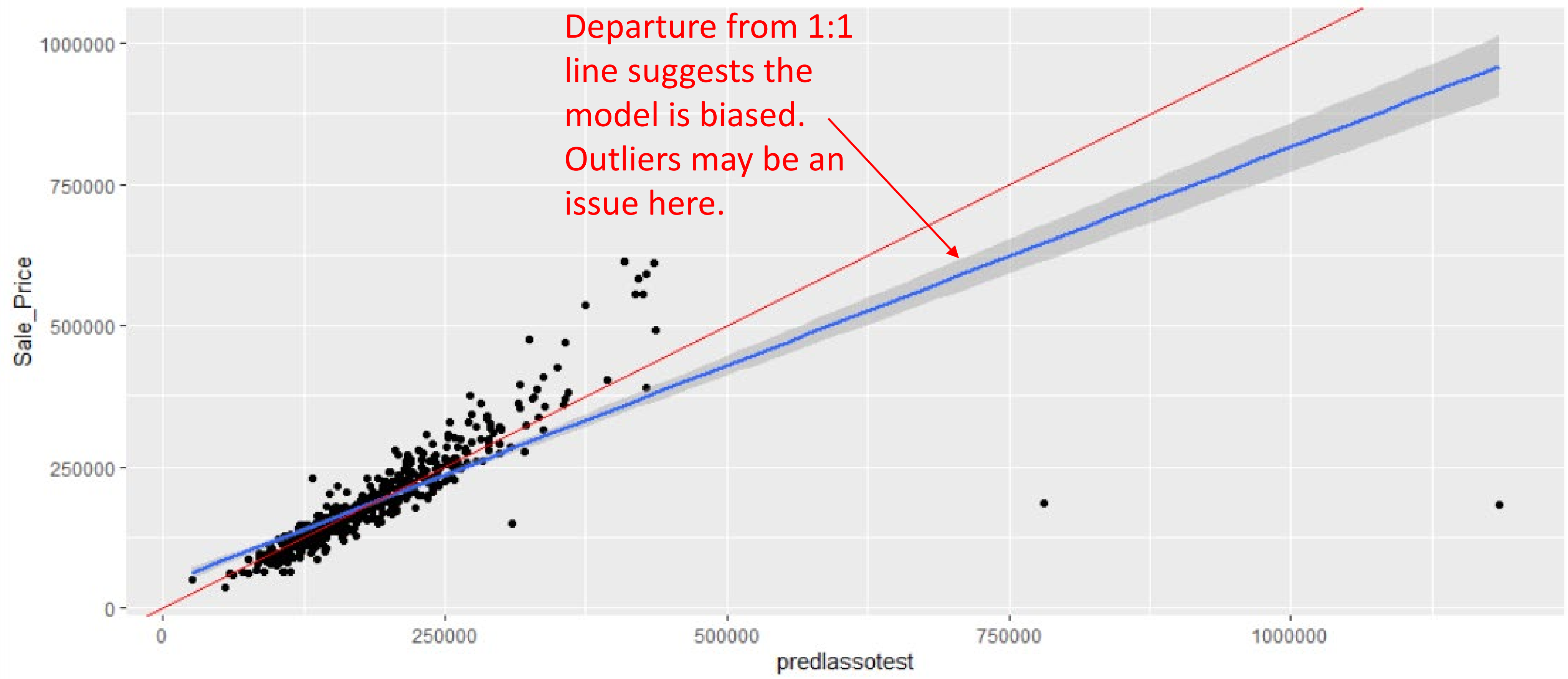
cv-selected lasso performance on train and test

- Not as good as ridge for this problem

```
> # evaluation on train_3 lasso
> predlasso <- predict(lasso, X)
> # compute RMSE of transformed predicted
> RMSE(exp(predlasso), exp(Y))
[1] 33823.32
> predlassotest <- predict(lasso, Xtest)
> RMSE(exp(predlassotest), exp(Ytest))
[1] 56674.53
```

GPI

Obs vs. exp plot for lasso on test_3




```
> cor(test_3aug$predlassotest,test_3aug$Sale_Price)^2
       [,1]
1 0.5856687
> res_predlassotest<-test_3aug$Sale_Price-test_3aug$predlassotest
> (mean((res_predlassotest)^2))^0.5
[1] 56674.53
```

GPI

elastic net on ames data requires tuning of both lambda and alpha hyperparmeters

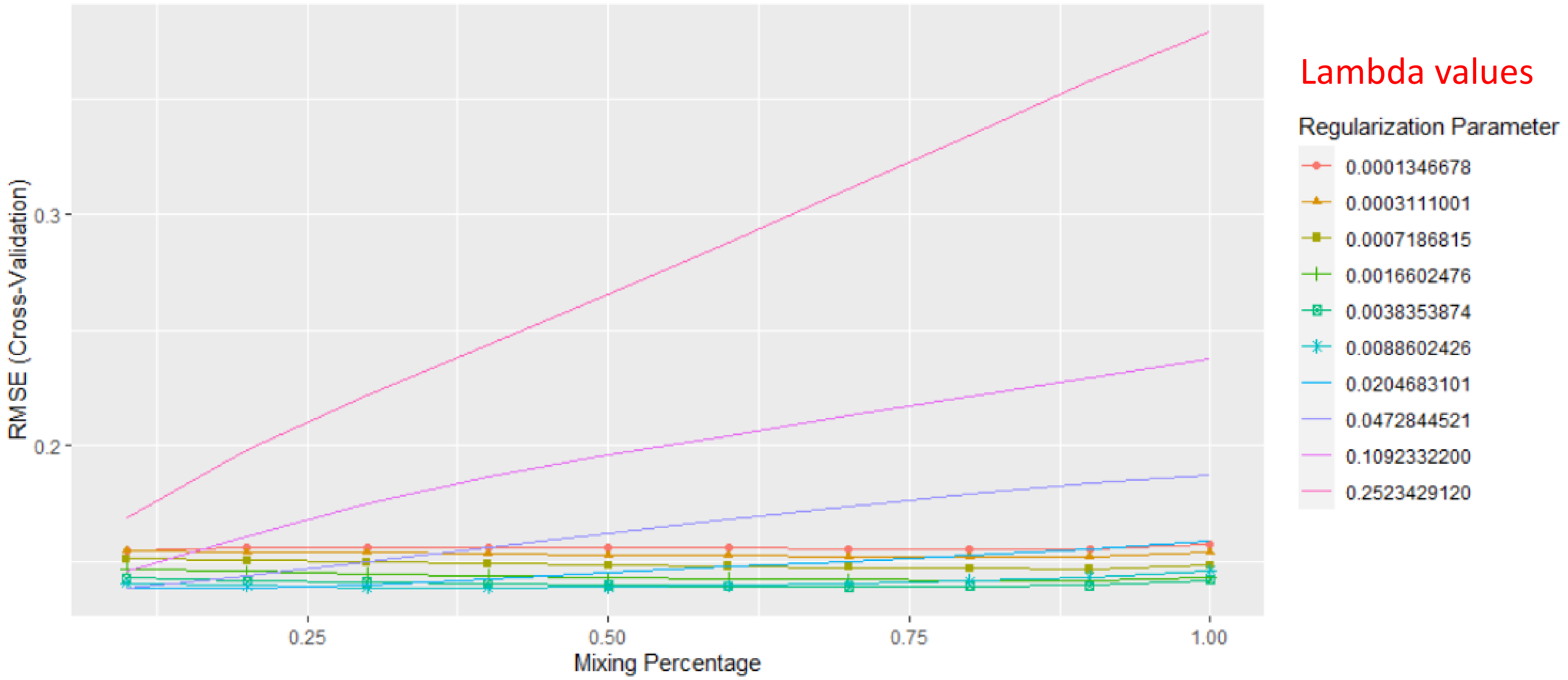
- Grid search implemented using caret package

```
# elastic net model
# for reproducibility
set.seed(42)

# grid search across
cv_glmnet <- train(
  x = X,
  y = Y,
  method = "glmnet",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)

# model with lowest RMSE
cv_glmnet$bestTune
##   alpha     lambda
## 8   0.1 0.04728445

# results for model with lowest RMSE
cv_glmnet$results %>%
  filter(alpha == cv_glmnet$bestTune$alpha, lambda == cv_glmnet$bestTune$lambda
##   alpha     lambda      RMSE  Rsquared        MAE    RMSESD RsquaredSD
## 1   0.1 0.04728445 0.1382785 0.8852799 0.08427956 0.0273119 0.04179193
## MAESD
## 1 0.005330001
```

GPI

elastic net tuning results

- ggplot(cv_glmnet)



Lambda values

Alpha values

Elastic net performance on train and test

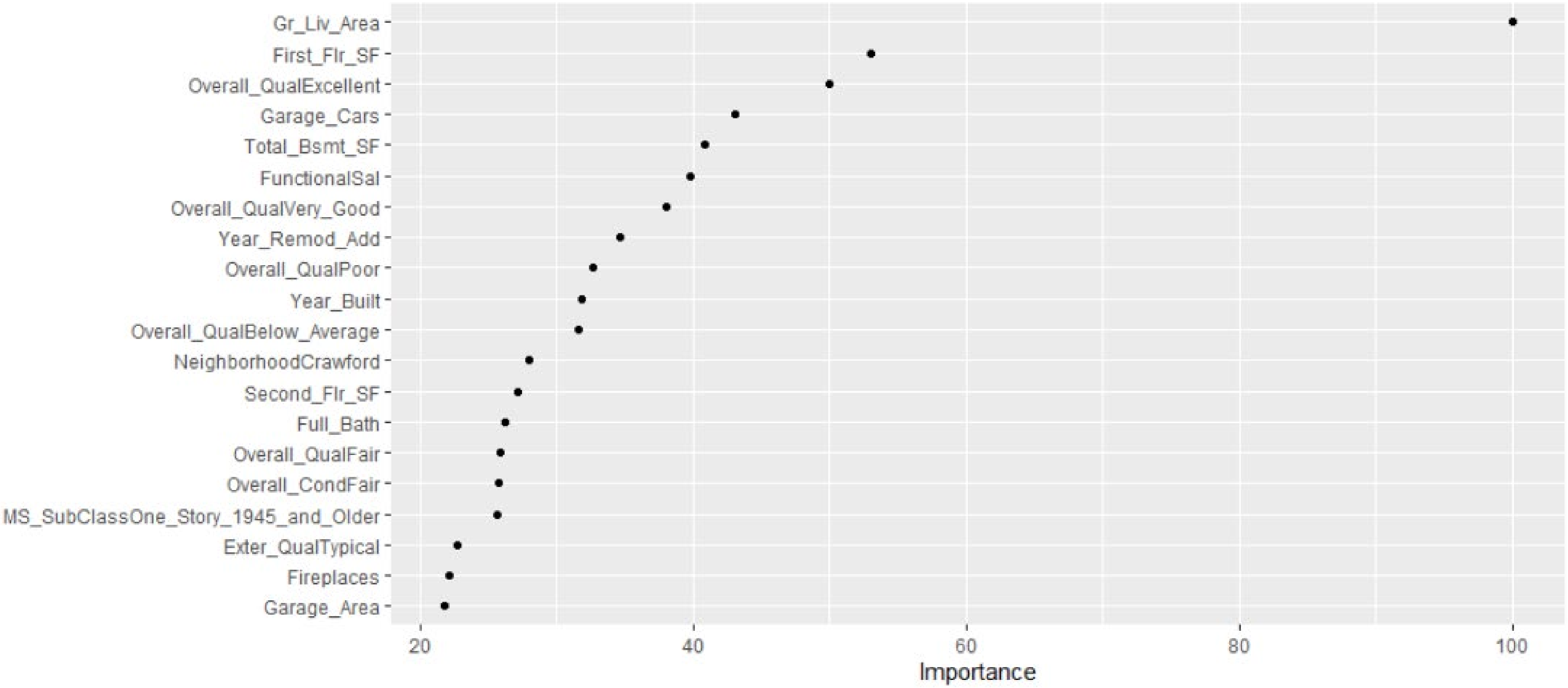
- Better than ridge on training set but worse on test!

```
> RMSE(exp(predelastic), exp(Y))
[1] 25329.31
> predelastictest <- predict(cv_glmnet, Xtest)
> RMSE(exp(predelastictest), exp(Ytest))
[1] 50741.77
> test_3aug$predelastictest<-exp(predelastictest)
> ggplot(test_3aug,aes(x=predelastictest,y=Sale_Price))+
+   geom_point()+stat_smooth(method=lm)+geom_abline(slope=1, intercept=0
`geom_smooth()` using formula 'y ~ x'
> cor(test_3aug$predelastictest,test_3aug$Sale_Price)^2
[1] 0.6622576
> res_predelastictest<-test_3aug$Sale_Price-test_3aug$predelastictest
> (mean((res_predelastictest)^2))^0.5
[1] 50741.77
>
```

GPI

Varianble importance for elastic net (top 20 variables)

- vip(cv_glmnet, num_features = 20, geom = "point")



GPI