Ridge Regression and Lasso

Yifei Sun

Contents

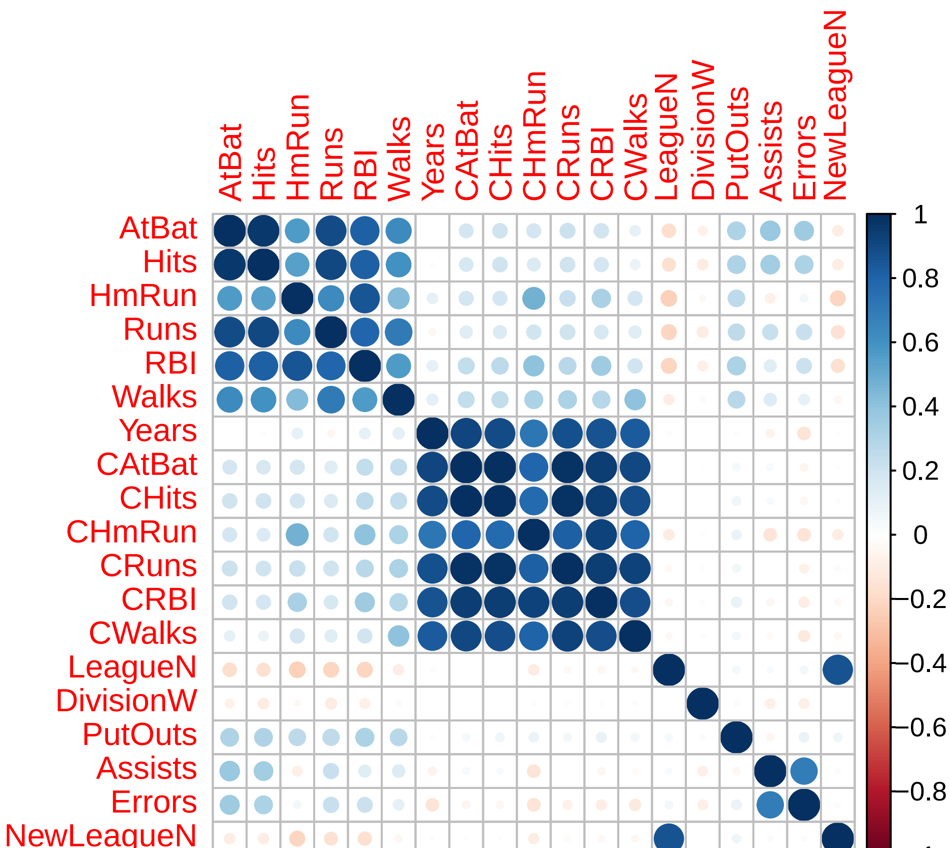
Using `glmnet` **3**
- Ridge . . . 3
- Lasso . . . 6

Using `caret` **8**
- Ridge . . . 8
- Lasso . . . 10
- Elastic net . . . 11
- Comparing different models . . . 12
- Prediction . . . 13

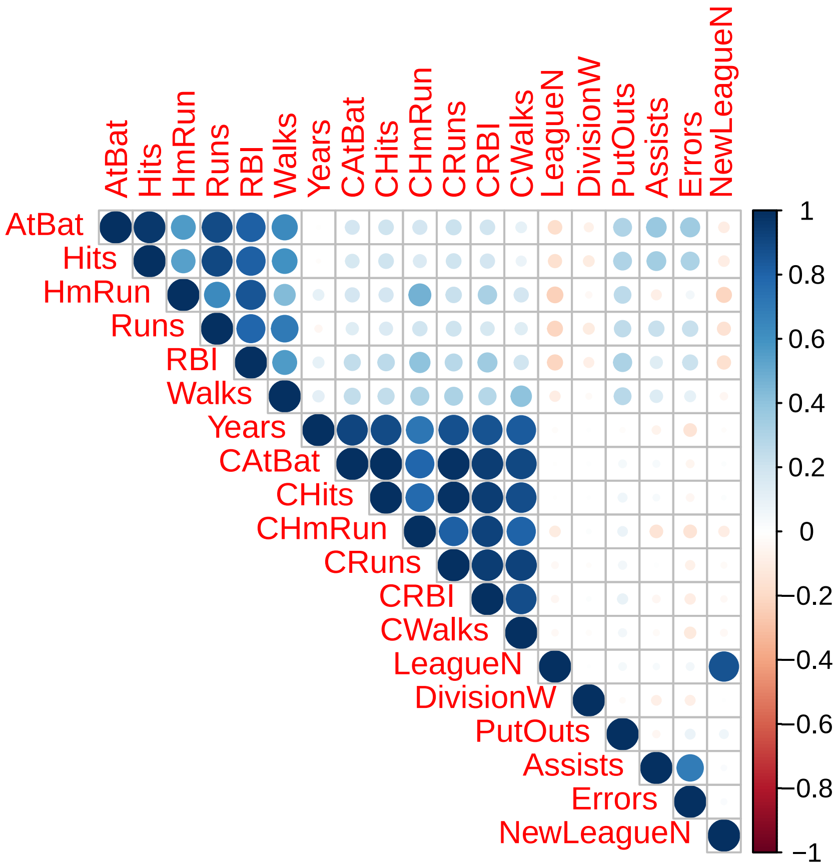
```r
library(ISLR)
library(glmnet)
library(caret)
library(corrplot)
library(plotmo)
```

Predict a baseball player's salary on the basis of various statistics associated with performance in the previous year. Use `?Hitters` for more details.

```r
#Load data from ISLR package
data(Hitters)

# delete rows containing the missing data
Hitters <- na.omit(Hitters)
Hitters2 <- model.matrix(Salary ~ ., Hitters)[ ,-1]  #remove 1st column

set.seed(1)
trainRows <- createDataPartition(y = Hitters$Salary, p = 0.8, list = FALSE)

# matrix of predictors (glmnet uses input matrix)
x <- Hitters2[trainRows,]
# vector of response
y <- Hitters$Salary[trainRows]

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



```r
corrplot(cor(x), method = "circle", type = "upper")
```




```r
#when visualize general matrix, is.corr = FALSE
```

Using glmnet

Ridge

`alpha` is the elastic net mixing parameter. `alpha=1` is the lasso penalty, and `alpha=0` the ridge penalty. `glmnet()` function standardizes the independent variables by default (The coefficients are always returned on the original scale).

```r
# fit the ridge regression (alpha = 0) with a sequence of lambdas
ridge.mod <- glmnet(x = x, y = y,
                    standardize = TRUE,
                    alpha = 0,
                    lambda = exp(seq(10, -2, length = 100)))
```

`coef(ridge.mod)` gives the coefficient matrix. Each column is the fit corresponding to one lambda value.

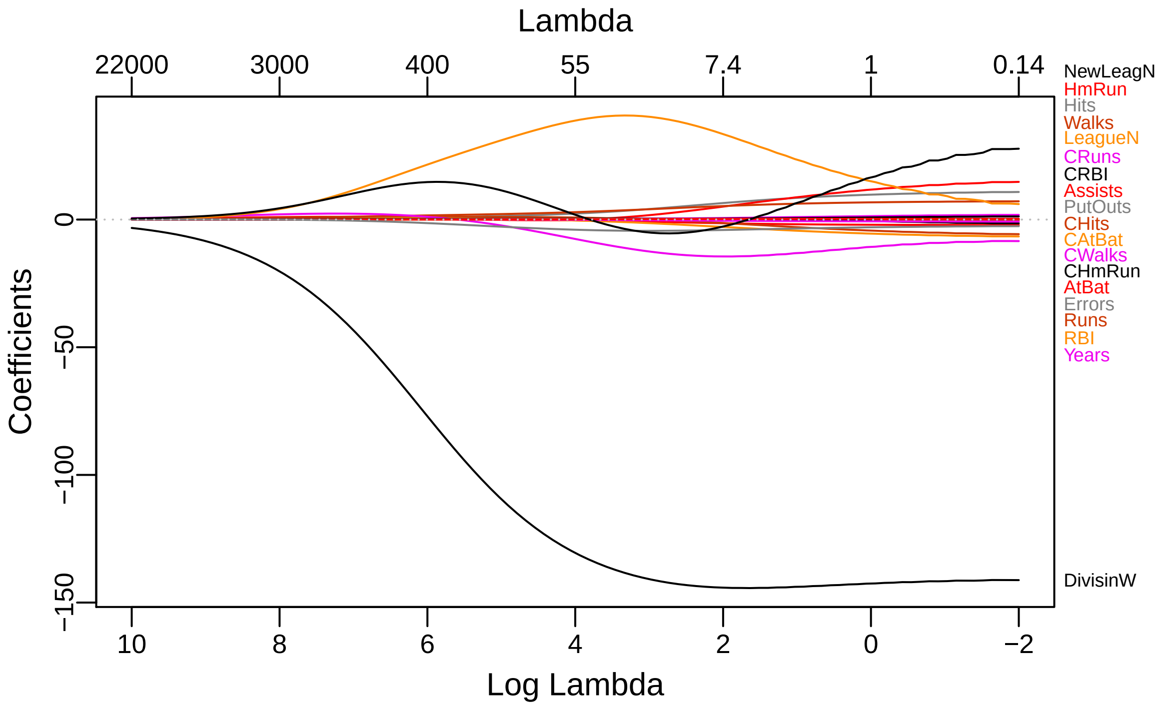
```r
mat.coef <- coef(ridge.mod)
dim(mat.coef)
```

```
## [1]  20 100
```

Trace plot

There are two functions for generating the trace plot.

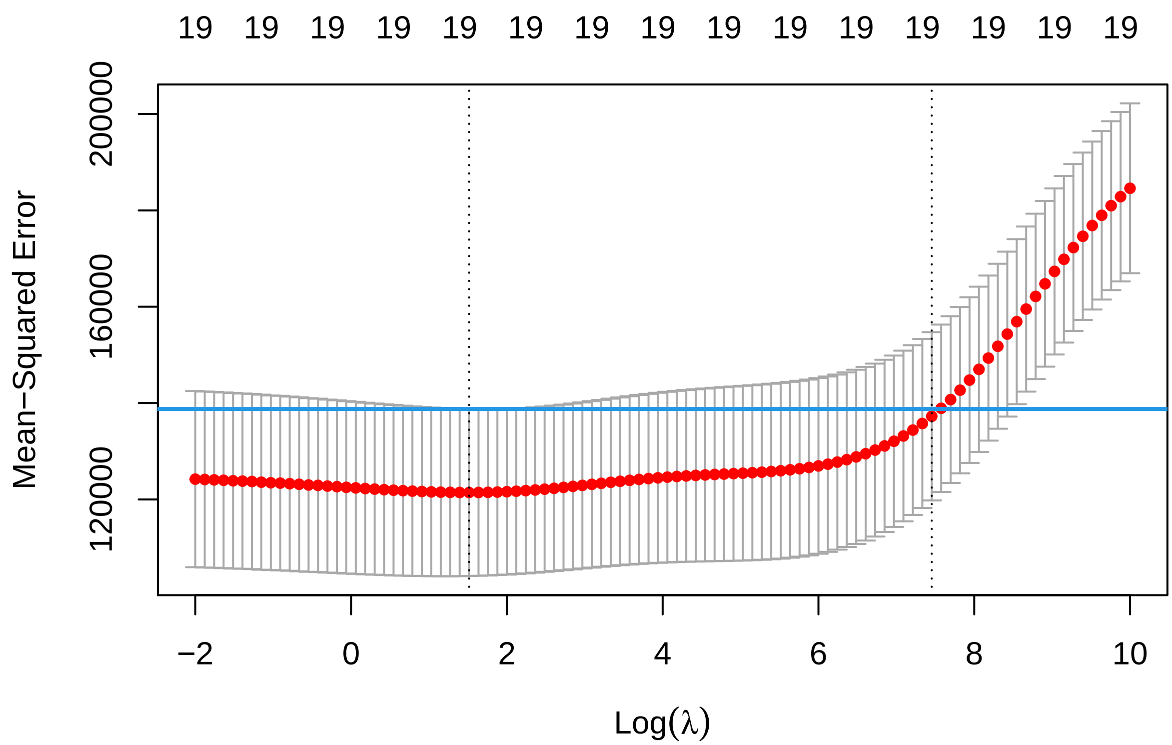
```r
# plot(ridge.mod, xvar = "lambda", label = TRUE)
plot_glmnet(ridge.mod, xvar = "rlambda", label = 19)
```



Cross-validation

We use cross-validation to determine the optimal value of `lambda`. The two vertical lines are the for minimal MSE and 1SE rule. The 1SE rule gives the most regularized model such that error is within one standard error of the minimum.

```r
set.seed(2)
cv.ridge <- cv.glmnet(x, y,
                      # type.measure = "mse",
                      alpha = 0,
                      lambda = exp(seq(10, -2, length = 100)))

# set.seed(2)
# cv.ridge <- cv.glmnet(x, y, alpha = 0, nlambda = 200)

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




```r
# min CV MSE
cv.ridge$lambda.min
```

```
## [1] 4.55011
```

```r
# the 1SE rule
cv.ridge$lambda.1se
```

```
## [1] 1727.698
```

Coefficients of the final model

Get the coefficients of the optimal model. `s` is value of the penalty parameter `lambda` at which predictions are required.

```r
# extract coefficients
predict(cv.ridge, s = cv.ridge$lambda.min, type = "coefficients")
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  187.47793981
## AtBat         -1.66099124
## Hits           7.55064892
## HmRun          6.90039547
## Runs          -2.15100868
## RBI           -3.63700287
## Walks          5.79807498
## Years        -14.09210967
## CAtBat        -0.06371461
## CHits          0.17916398
## CHmRun         0.18107631
## CRuns          0.78767779
## CRBI           0.46181854
```

```
## CWalks        -0.53218568
## LeagueN       28.60044968
## DivisionW   -144.24656509
## PutOuts        0.27817313
## Assists        0.25720737
## Errors        -3.79404046
## NewLeagueN     1.34412086
```

```r
# make prediction
head(predict(cv.ridge, newx = Hitters2[-trainRows,],
             s = "lambda.min", type = "response"))
```

```
##                            1
## -Alfredo Griffin   527.53239
## -Argenis Salazar    61.06837
## -Andres Thomas     109.27081
## -Alex Trevino      206.62951
## -Buddy Biancalana   67.48519
## -Bill Doran        679.44059
```

```r
# predict(cv.ridge, s = "lambda.min", type = "coefficients")
# predict(cv.ridge, s = "lambda.1se", type = "coefficients")
# predict(ridge.mod, s = cv.ridge$lambda.min, type = "coefficients")
```

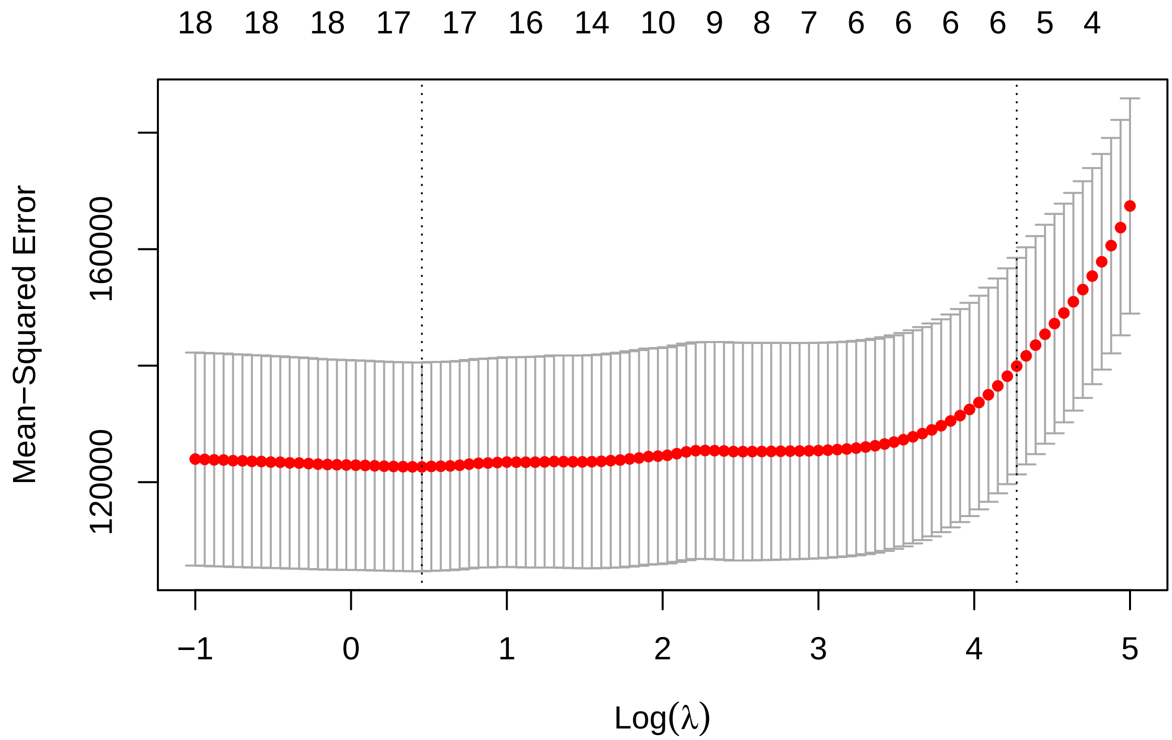
Lasso

The syntax is along the same line as ridge regression. Now we use `alpha = 1`.

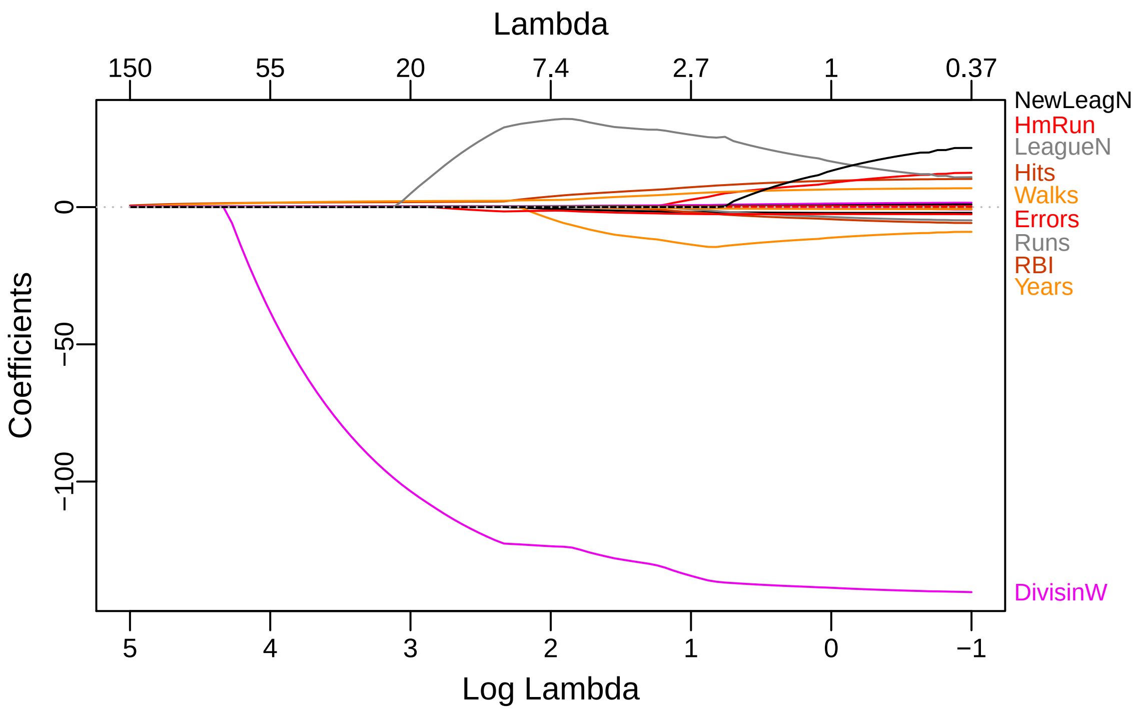
```r
set.seed(2)  #if compare the models, MUST set seed

cv.lasso <- cv.glmnet(x, y,
                      alpha = 1,
                      lambda = exp(seq(5, -1, length = 100)))

cv.lasso$lambda.min
```

```
## [1] 1.575457
```

```r
#CV plot
plot(cv.lasso)
```




```r
# cv.lasso$glmnet.fit is a fitted glmnet object using the full training data
# plot(cv.lasso$glmnet.fit, xvar = "lambda", label=TRUE)
plot_glmnet(cv.lasso$glmnet.fit)
```




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

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         1
```

```
## (Intercept)  190.1303902
## AtBat         -1.9991536
## Hits           8.8068531
## HmRun          6.7051224
## Runs          -2.6093156
## RBI           -3.5322303
## Walks          5.9778998
## Years        -12.7662402
## CAtBat        -0.0486237
## CHits          .
## CHmRun         .
## CRuns          1.0546063
## CRBI           0.5212946
## CWalks        -0.6059518
## LeagueN       21.0361217
## DivisionW   -137.7115296
## PutOuts        0.2781846
## Assists        0.2220624
## Errors        -2.5551464
## NewLeagueN     6.6744338
```

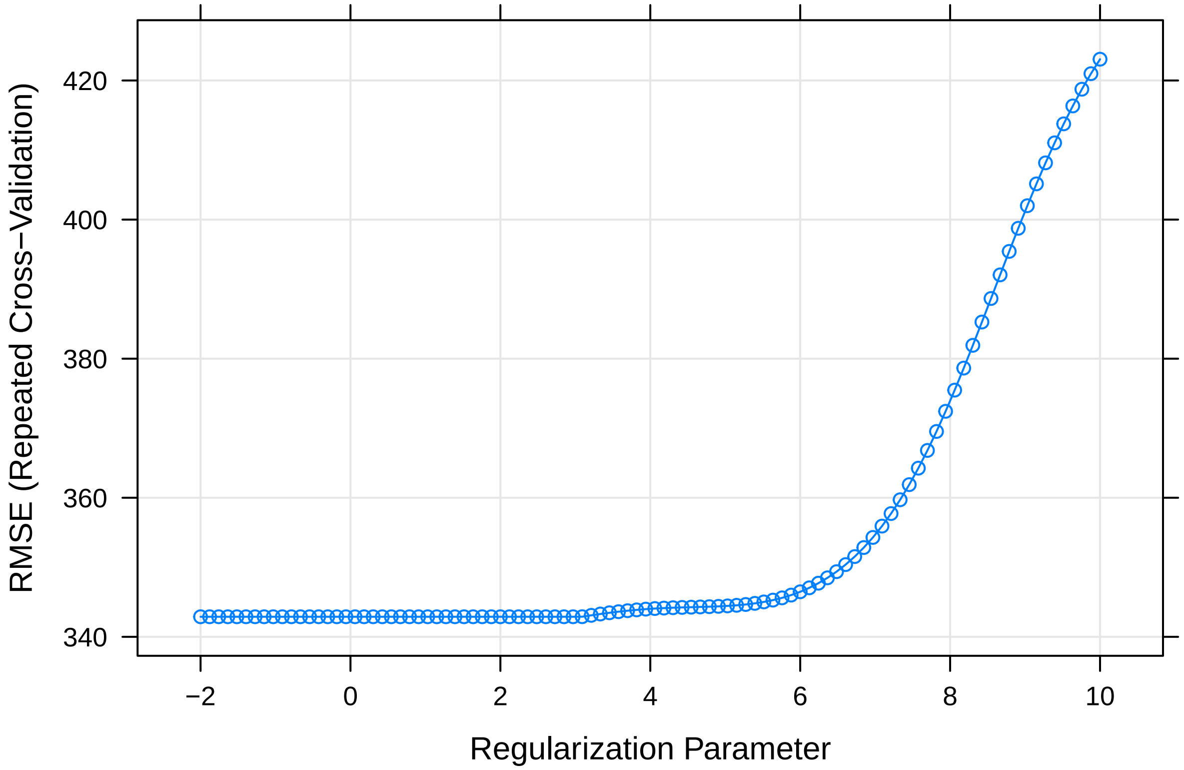
```r
head(predict(cv.lasso, newx = Hitters2[-trainRows,], s = "lambda.min", type = "response"))
```

```
##                             1
## -Alfredo Griffin  555.34750
## -Argenis Salazar   59.73438
## -Andres Thomas    115.07960
## -Alex Trevino     220.78767
## -Buddy Biancalana  86.47488
## -Bill Doran       678.23355
```

Using caret

Ridge

```r
ctrl1 <- trainControl(method = "repeatedcv", number = 10, repeats = 5) #repeat 10fold CV 5 times

# you can try other options

set.seed(2)
ridge.fit <- train(x, y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(10, -2, length=100))),
                   # preProc = c("center", "scale"),
                   trControl = ctrl1)

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




```r
ridge.fit$bestTune
```

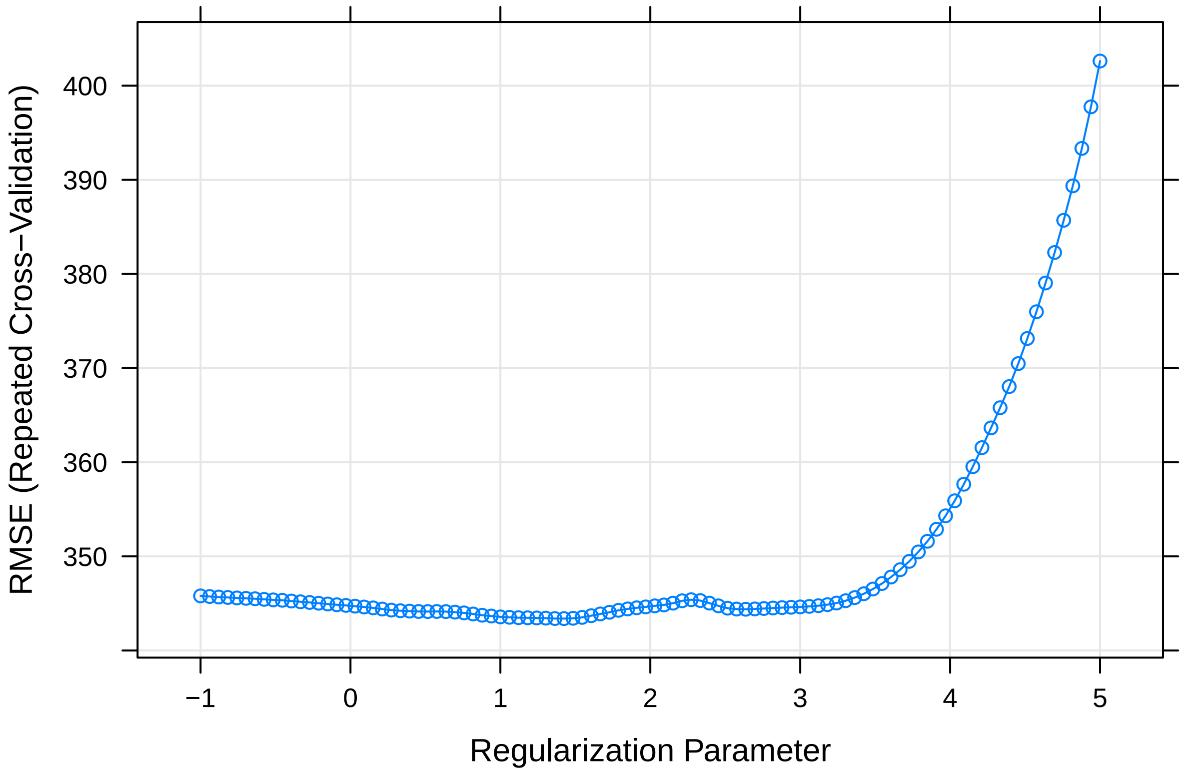
```
##    alpha   lambda
## 42     0 19.48601
```

```r
# extract coefficients in the final model
coef(ridge.fit$finalModel, s = ridge.fit$bestTune$lambda)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  116.30034503
## AtBat         -0.75547486
## Hits           3.80544418
## HmRun          1.35084857
## Runs           0.19577377
## RBI           -1.20746819
## Walks          3.86063643
## Years        -11.91127105
## CAtBat        -0.00542965
## CHits          0.14543241
## CHmRun         0.39763107
## CRuns          0.34162393
## CRBI           0.25842524
## CWalks        -0.23931269
## LeagueN       40.63457163
## DivisionW   -139.83145367
## PutOuts        0.27532833
## Assists        0.17761899
## Errors        -4.39495131
## NewLeagueN    -4.59177054
```

Lasso

```
set.seed(2)
lasso.fit <- train(x, y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(5, -1, length=100))),
                   trControl = ctrl1)
plot(lasso.fit, xTrans = log)
```




```
lasso.fit$bestTune
```

```
##    alpha   lambda
## 41     1 4.154709
```

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

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  130.91361512
## AtBat         -1.40216367
## Hits           5.82897162
## HmRun          .
## Runs           .
## RBI           -0.50242361
## Walks          3.95543093
## Years        -11.07854445
## CAtBat         .
## CHits          .
## CHmRun         0.05318777
## CRuns          0.57987143
```

```
## CRBI           0.36023577
## CWalks        -0.26891300
## LeagueN       28.44226984
## DivisionW   -129.10416611
## PutOuts        0.27339148
## Assists        0.10246507
## Errors        -2.14742985
## NewLeagueN     .
```

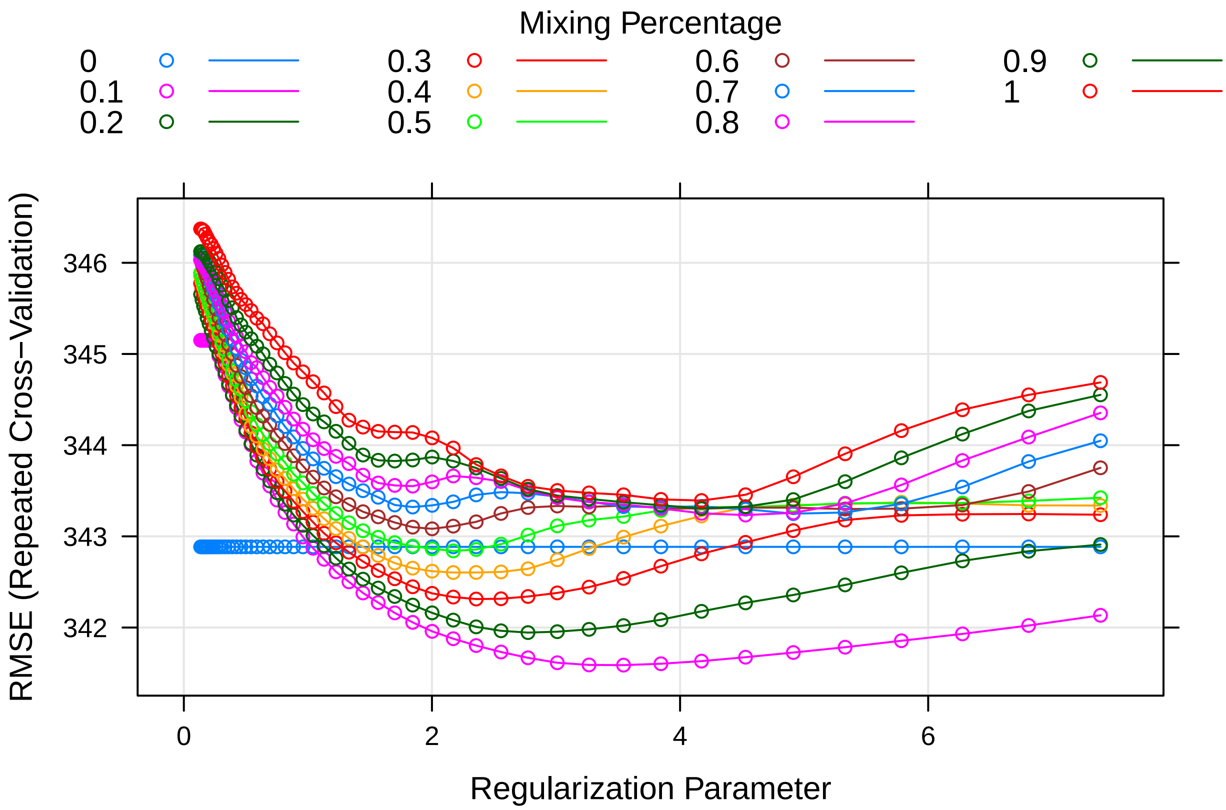
Elastic net

For elastic net, lambda is somewhere between 0 and 1 => need to use expand.grid

```r
set.seed(2)
enet.fit <- train(x, y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 11),
                                         lambda = exp(seq(2, -2, length = 50))),
                  trControl = ctrl1)
enet.fit$bestTune
```

```
##    alpha   lambda
## 91   0.1 3.544178
```

```r
plot(enet.fit)
```




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

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                          1
```

```
## (Intercept)  189.40183300
## AtBat         -1.74356695
## Hits           7.91109071
## HmRun          7.26915235
## Runs          -2.34816631
## RBI           -3.75754990
## Walks          5.87536936
## Years        -13.71886801
## CAtBat        -0.06617559
## CHits          0.14398058
## CHmRun         0.05610600
## CRuns          0.86094283
## CRBI           0.51034122
## CWalks        -0.54822240
## LeagueN       26.45193163
## DivisionW   -142.94761134
## PutOuts        0.27691667
## Assists        0.25137406
## Errors        -3.51244002
## NewLeagueN     2.93610504
```

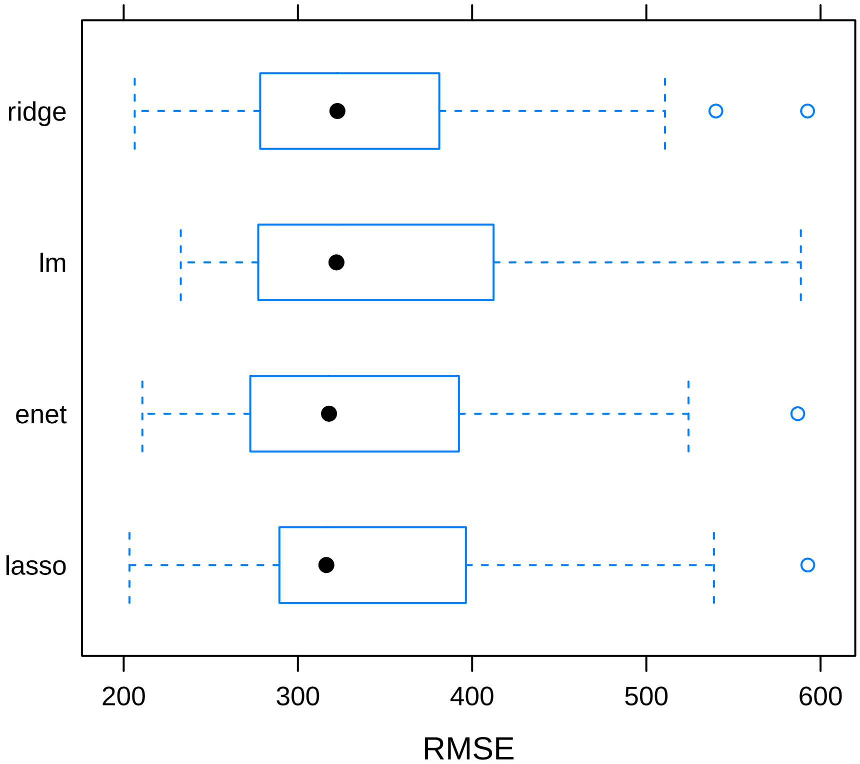
Comparing different models

```
set.seed(2)
lm.fit <- train(x, y,
                method = "lm",
                trControl = ctrl1)

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

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: enet, lasso, ridge, lm
## Number of resamples: 50
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  168.6714 205.4745 242.5138 241.7830 263.2851 366.0139    0
## lasso 168.9988 208.8255 235.2644 240.7921 268.7084 347.1911    0
## ridge 161.2851 207.0816 239.2387 240.6205 266.5522 355.9162    0
## lm    184.3250 208.2682 241.4425 246.8083 268.6806 366.6814    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  210.7348 273.6172 317.8660 341.5879 391.1444 586.9171    0
## lasso 203.3181 289.5702 316.3301 343.3909 391.8513 592.6729    0
## ridge 206.3196 281.4159 322.6818 342.8847 380.6289 592.5404    0
## lm    232.7716 277.5456 322.1400 348.3648 409.9136 588.6902    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
## enet  0.04529733 0.3328151 0.4522642 0.4577072 0.5761208 0.7715195    0
## lasso 0.03140716 0.3192695 0.4691541 0.4526682 0.5613431 0.7646081    0
## ridge 0.02952308 0.3283170 0.4830048 0.4518820 0.5511450 0.7835714    0
## lm    0.05372656 0.2972000 0.4413350 0.4475750 0.6017544 0.7596438    0
```

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



Prediction

```r
enet.pred <- predict(enet.fit, newdata = Hitters2[-trainRows,])
# test error
mean((enet.pred - Hitters$Salary[-trainRows])^2)
```

```
## [1] 86710.95
```