Ridge Regression and Lasso

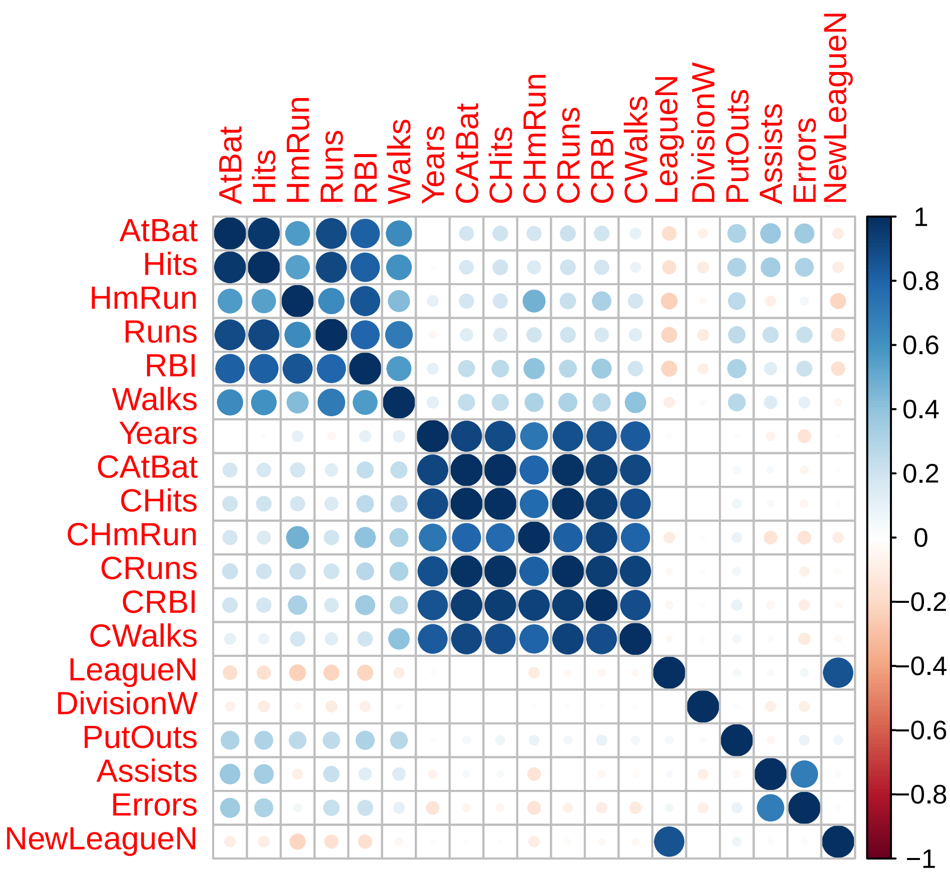
Yifei Sun

Contents

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

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")
```



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).

```
# 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.

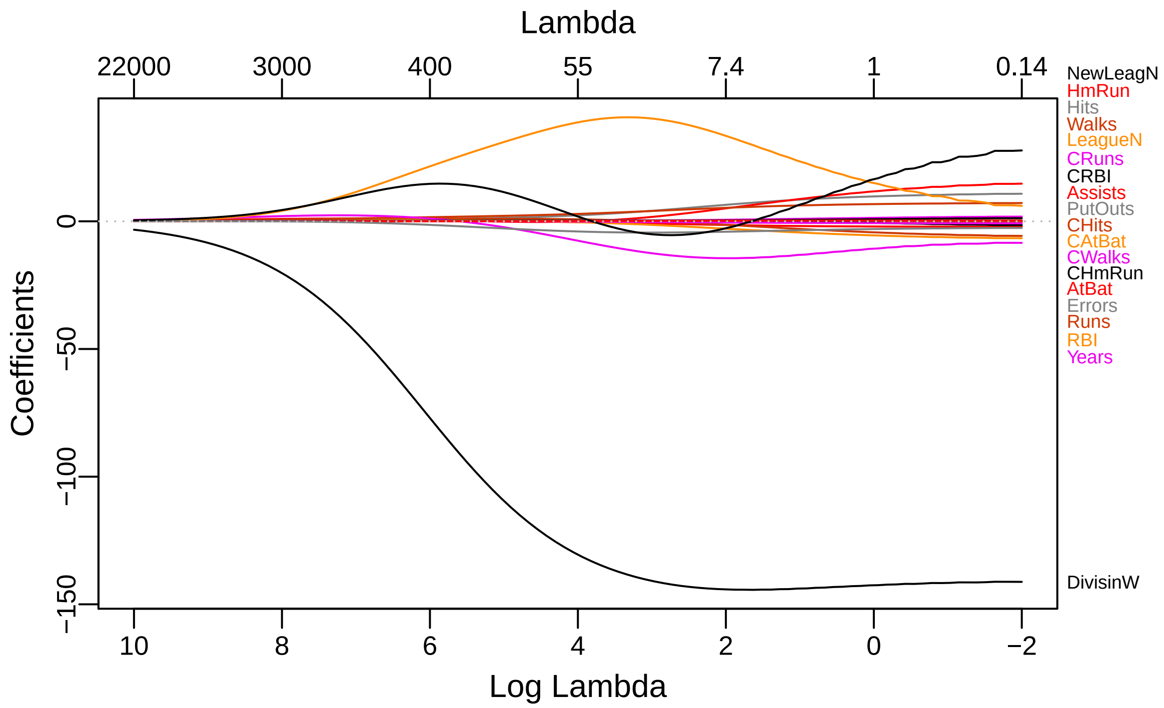
```
mat.coef <- coef(ridge.mod)
dim(mat.coef)
```

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

Trace plot

There are two functions for generating the trace plot.

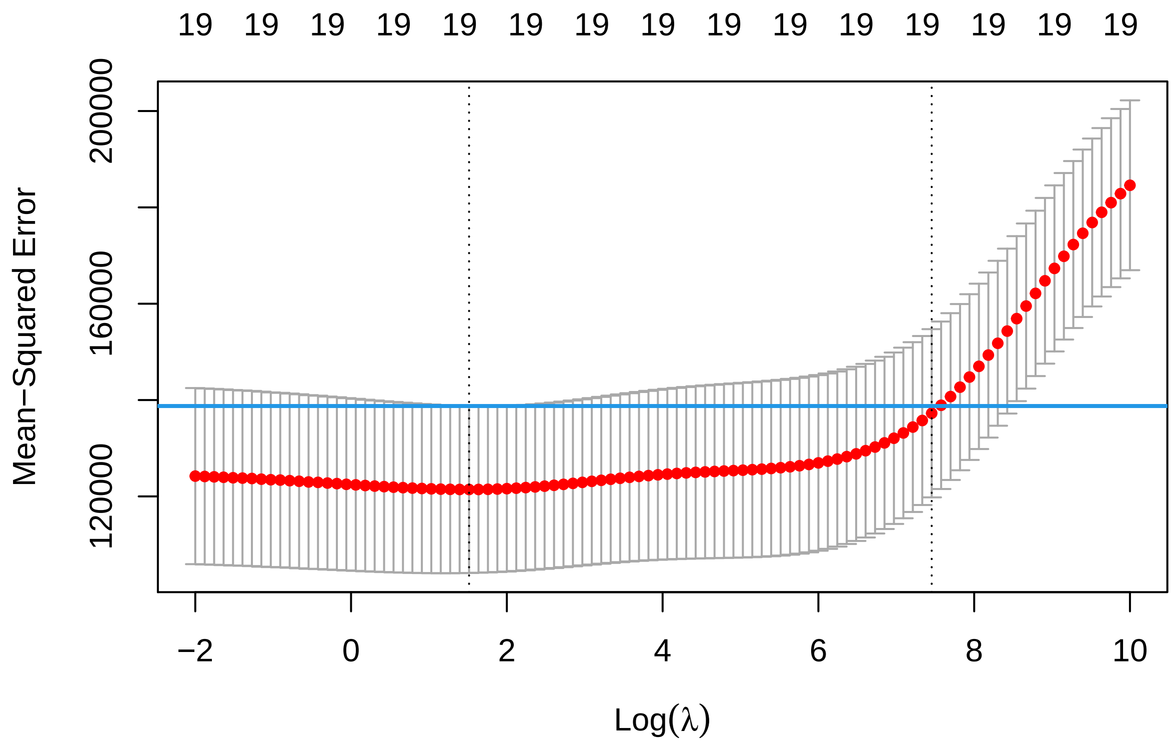
```
# plot(ridge.mod, xvar = "lambda", label = TRUE)
plot_glmnet(ridge.mod, xvar = "rlambda", label = 19)
```



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.

```
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")
```

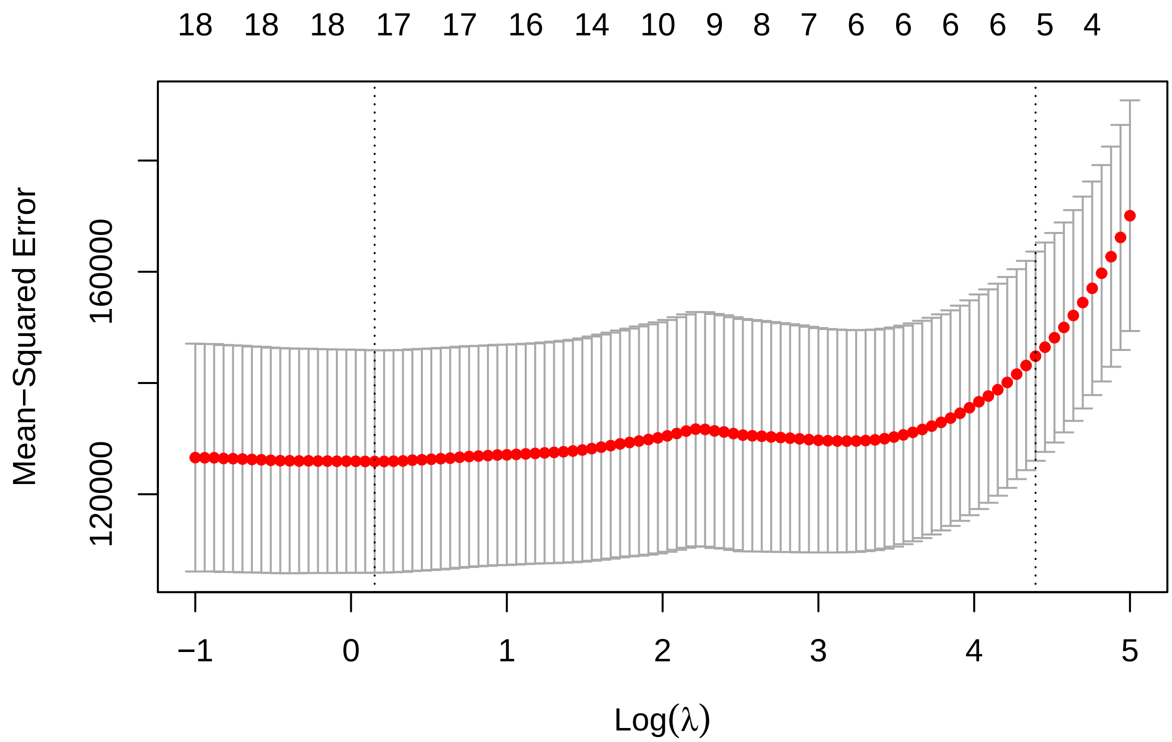
Lasso

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

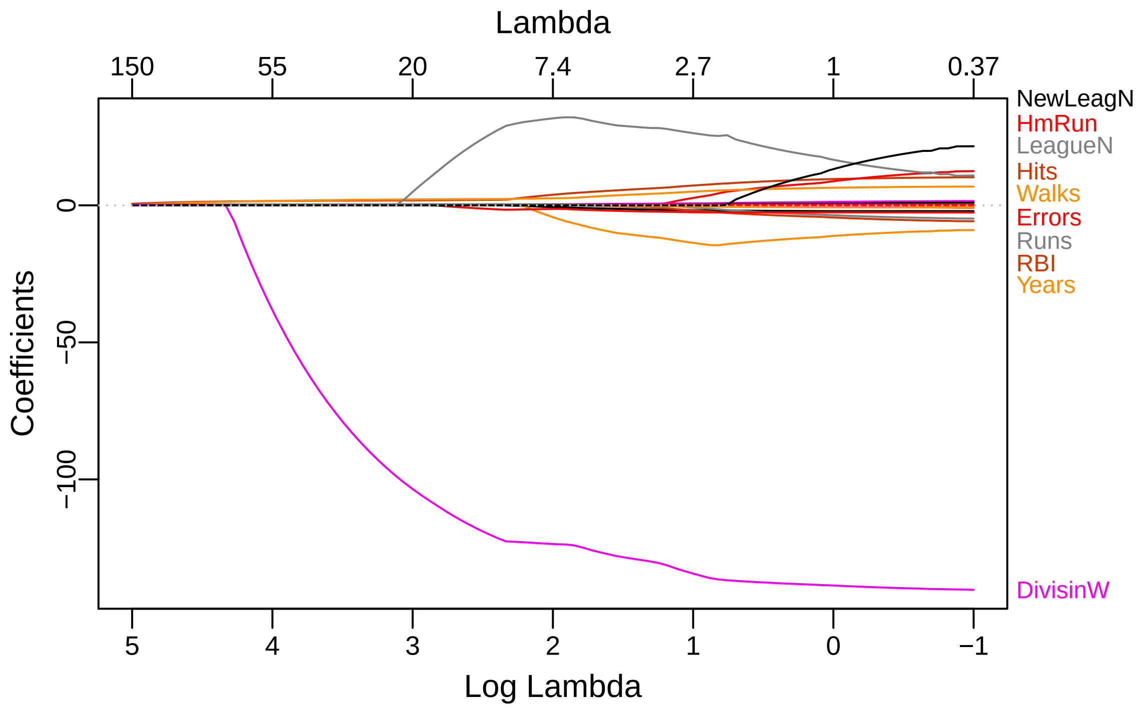
```r
cv.lasso <- cv.glmnet(x, y,
                      alpha = 1,
                      lambda = exp(seq(5, -1, length = 100)))

cv.lasso$lambda.min
```

```
## [1] 1.163596
```

```r
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)  195.84379666
## AtBat         -2.08071092
## Hits           9.38781075
## HmRun          7.95869370
## Runs          -3.27983552
## RBI           -4.10538819
## Walks          6.30149471
## Years        -11.72737414
## CAtBat        -0.07065959
## CHits          .
## CHmRun         .
## CRuns          1.20261591
## CRBI           0.55807973
## CWalks        -0.66361975
## LeagueN       18.15826533
## DivisionW   -138.38420328
## PutOuts        0.27861942
## Assists        0.24649382
## Errors        -2.53679564
## NewLeagueN    11.02829856
```

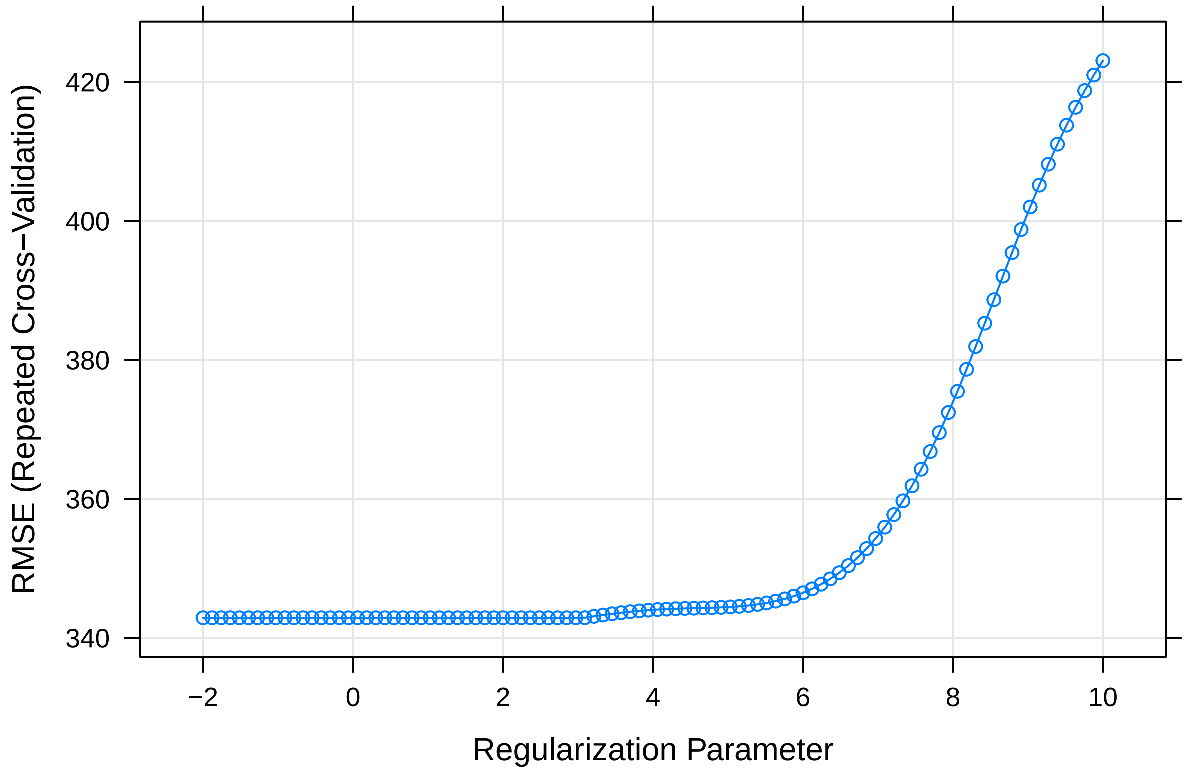
```r
head(predict(cv.lasso, newx = Hitters2[-trainRows,], s = "lambda.min", type = "response"))
```

```
##                            1
## -Alfredo Griffin  548.10104
## -Argenis Salazar   61.06747
## -Andres Thomas    124.23164
## -Alex Trevino     215.10299
## -Buddy Biancalana  96.94621
## -Bill Doran       682.40680
```

Using caret

Ridge

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

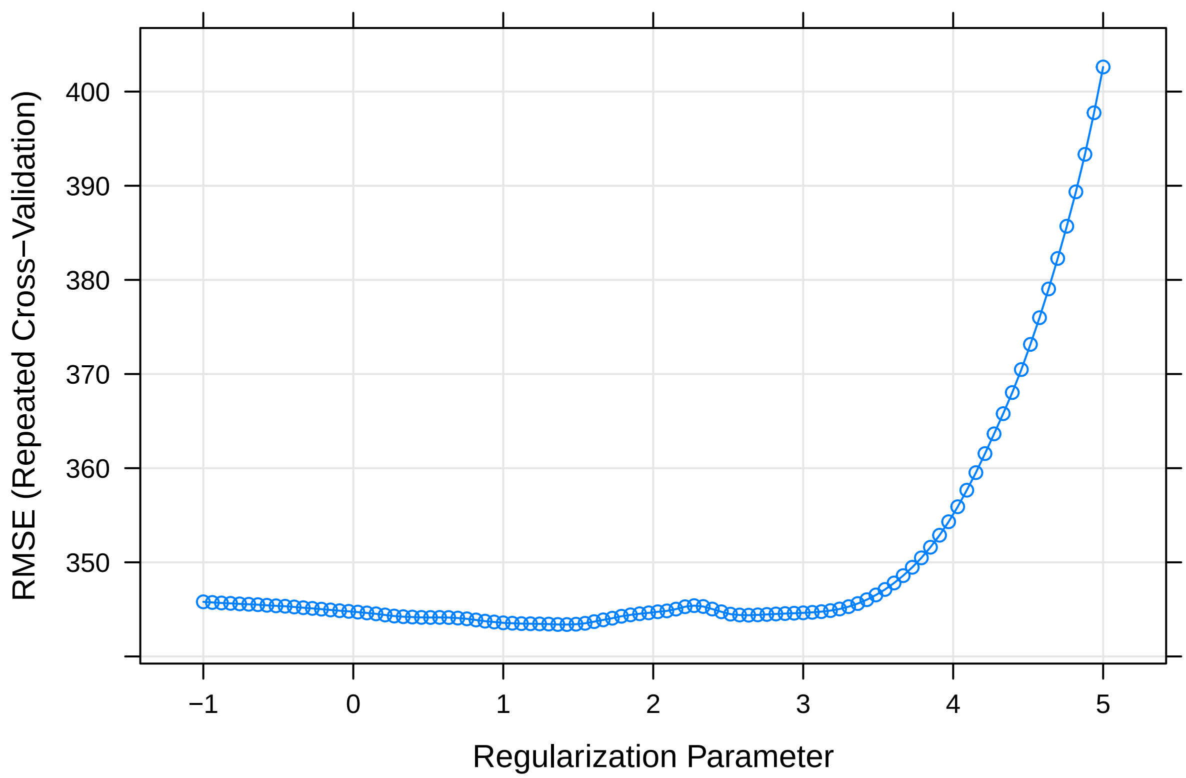
```r
# coefficients in the final model
coef(ridge.fit$finalModel, s = ridge.fit$bestTune$lambda)
```

```
## 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
```

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

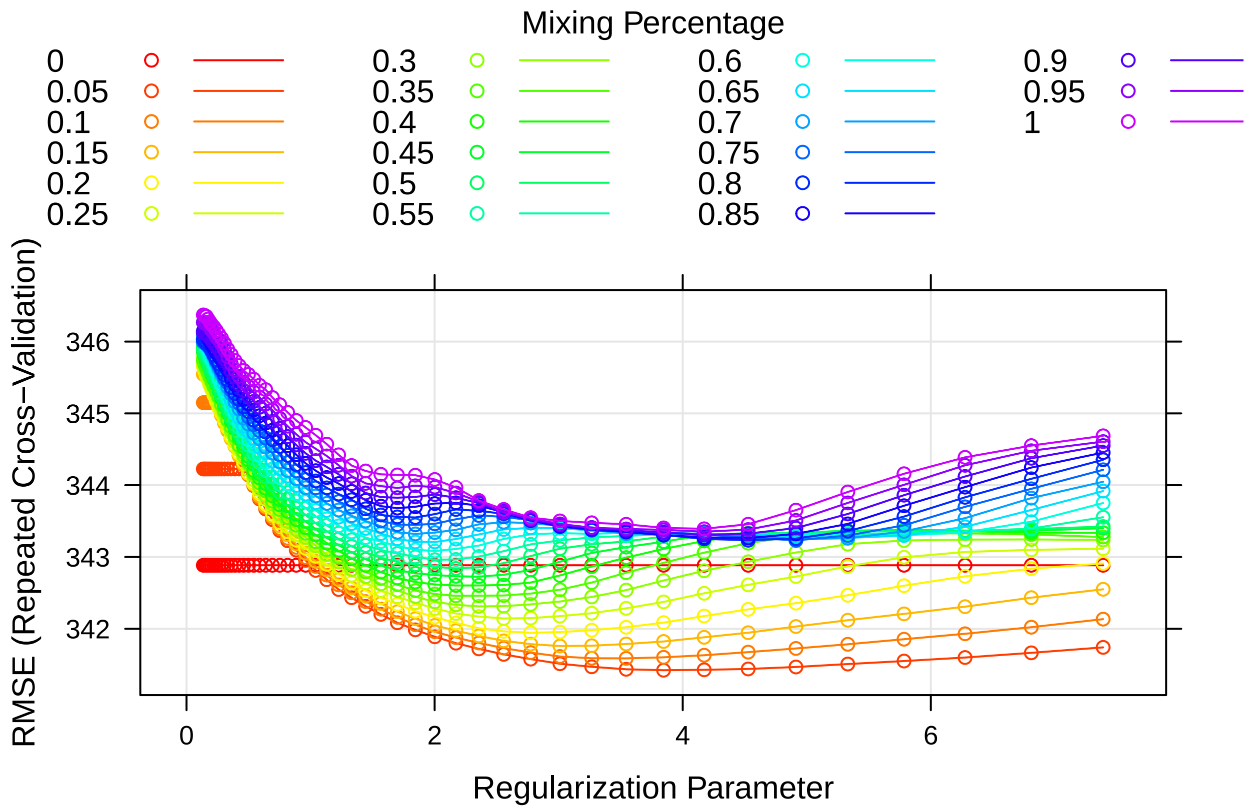
```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  130.91361512
## AtBat         -1.40216367
## Hits           5.82897162
## HmRun          .
## Runs           .
## RBI           -0.50242361
## Walks          3.95543093
## Years        -11.07854445
## CAtBat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

```r
set.seed(2)
enet.fit <- train(x, y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(2, -2, length = 50))),
                  trControl = ctrl1)
enet.fit$bestTune
```

```
##    alpha   lambda
## 92  0.05 3.845635
```

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

plot(enet.fit, par.settings = myPar)
```




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

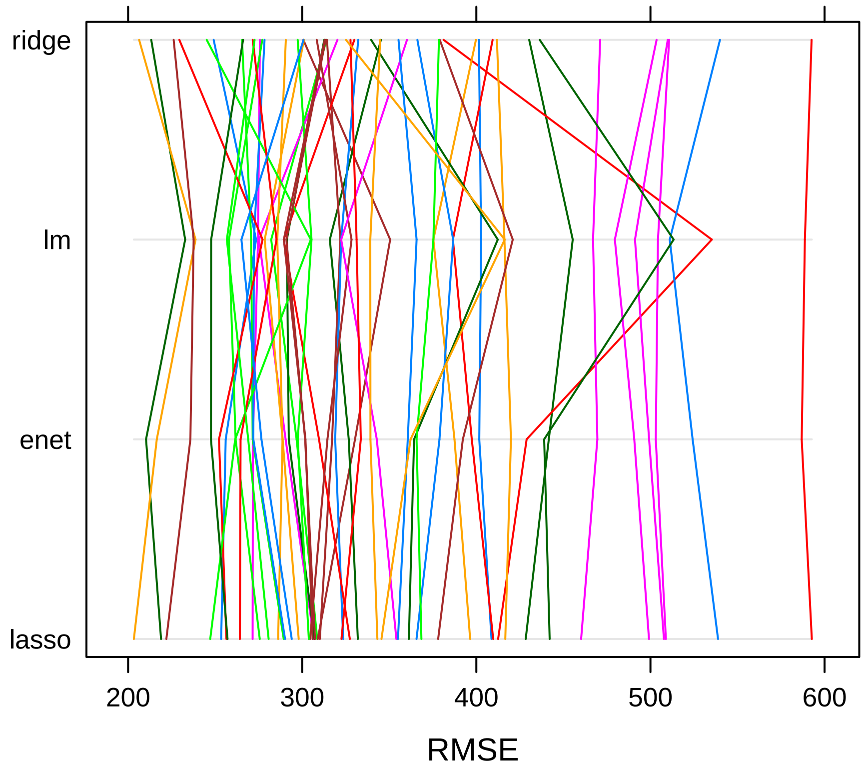
```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  189.46236012
## AtBat         -1.71757254
## Hits           7.81061753
## HmRun          7.26397879
## Runs          -2.31150169
## RBI           -3.76426931
## Walks          5.86936040
## Years        -13.88217929
## CAtBat        -0.06684253
## CHits          0.16110230
## CHmRun         0.09422352
## CRuns          0.83555621
## CRBI           0.49700124
## CWalks        -0.54386830
## LeagueN       27.12192310
## DivisionW   -143.58521322
## PutOuts        0.27715026
## Assists        0.25504021
## Errors        -3.62624892
## NewLeagueN     2.45529034
```

Comparing different models

```r
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))
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  168.2017 205.8303 242.6851 241.7662 262.8512 366.0989    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  210.3042 273.0975 318.0489 341.4230 391.0516 586.8653    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  0.04567264 0.3334405 0.4520697 0.4580264 0.5759741 0.7747336    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
parallelplot(resamp, metric = "RMSE")
```




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

Prediction

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

```
## [1] 86580.01
```