Ridge Regression

Yifei Sun, Runze Cui

Contents

```r
library(ISLR)
library(glmnet)
library(caret)
library(tidymodels)
library(corrplot)
library(ggplot2)
library(plotmo)
```

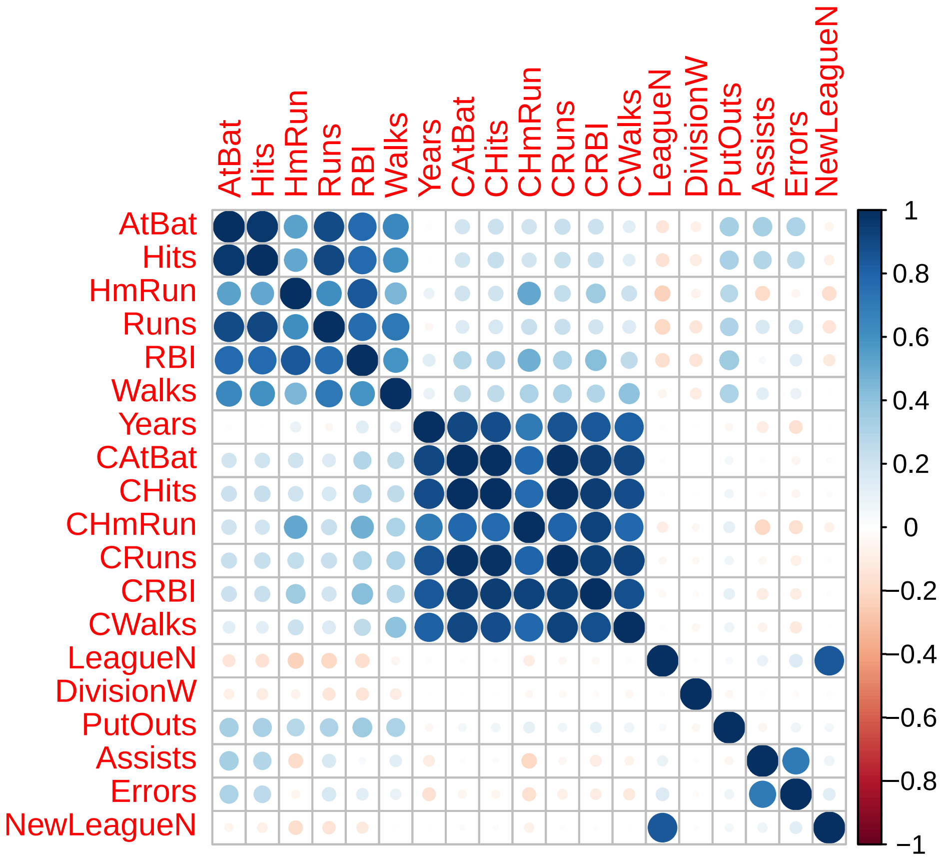
Predict a baseball player's salary on the basis of various statistics associated with performance in the previous year. Use `?Hitters` for more details.

```r
data(Hitters)
Hitters <- na.omit(Hitters)

set.seed(2222)
data_split <- initial_split(Hitters, prop = 0.8)

# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)
```

Using glmnet

Ridge regression

```r
# matrix of predictors (glmnet uses input matrix)
x <- model.matrix(Salary ~ ., training_data)[,-1]
# vector of response
y <- training_data[, "Salary"]

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



`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, -5, length = 100)))
```

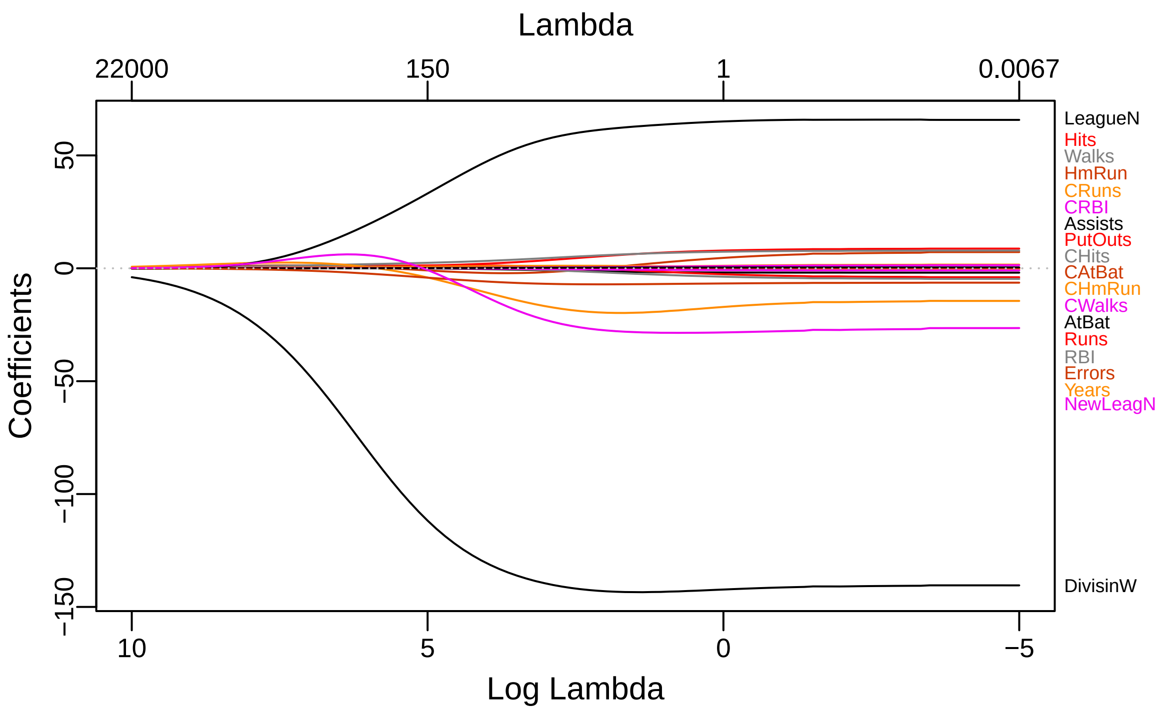
`coef(ridge.mod)` gives the coefficient matrix. Each column is the fit corresponding to one lambda value.

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

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

Trace plot

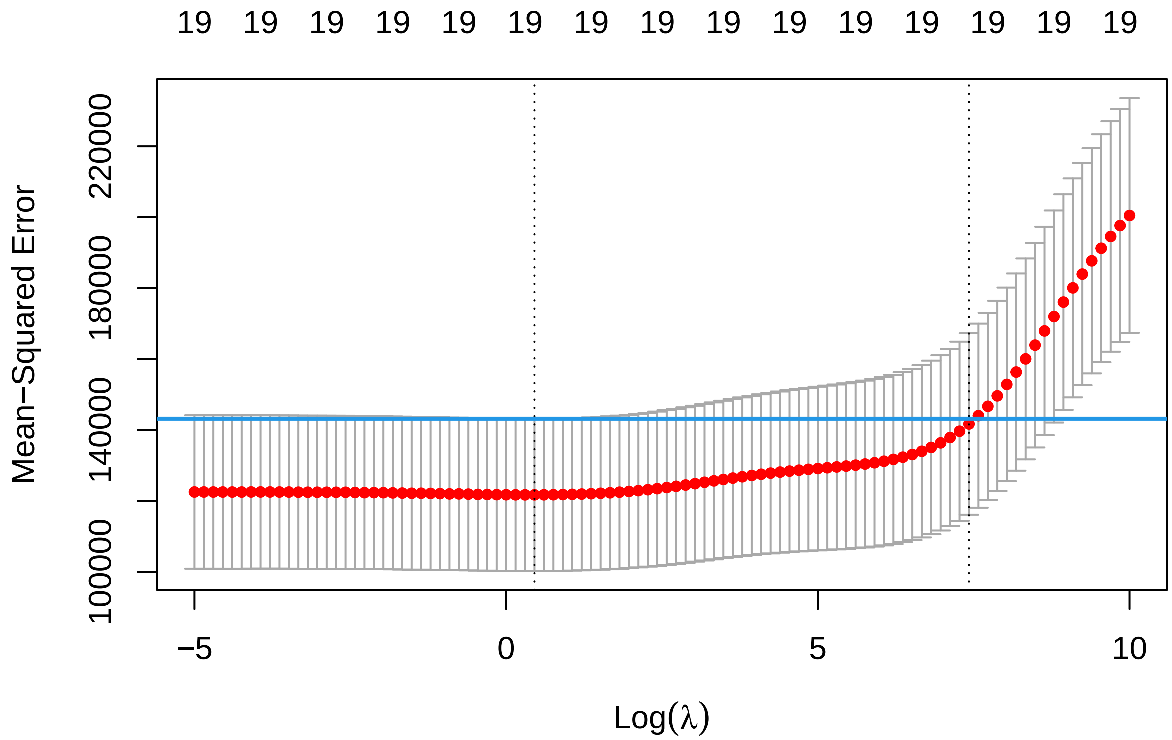
```
# plot(ridge.mod, xvar = "lambda", label = TRUE)
plot_glmnet(ridge.mod, xvar = "rlambda", label = 19)
```



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,
                      alpha = 0,
                      lambda = exp(seq(10, -5, 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] 1.575457
```

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

```
## [1] 1676.129
```

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"
##                        s1
## (Intercept)  252.8878597
## AtBat         -1.8235418
## Hits           7.5414783
## HmRun          3.7944825
## Runs          -2.2100103
## RBI           -3.3985935
## Walks          7.1356483
## Years        -18.0404714
## CAtBat        -0.1074182
## CHits          0.1561575
## CHmRun         0.6828518
## CRuns          0.9920259
## CRBI           0.8501477
```

```
## CWalks         -0.7601740
## LeagueN        64.5057331
## DivisionW    -142.8063145
## PutOuts         0.2377037
## Assists         0.3642801
## Errors         -6.8190642
## NewLeagueN    -28.5620199
```

```
# make prediction
head(predict(cv.ridge, newx = model.matrix(Salary ~ ., testing_data)[,-1],
             s = "lambda.min", type = "response"))
```

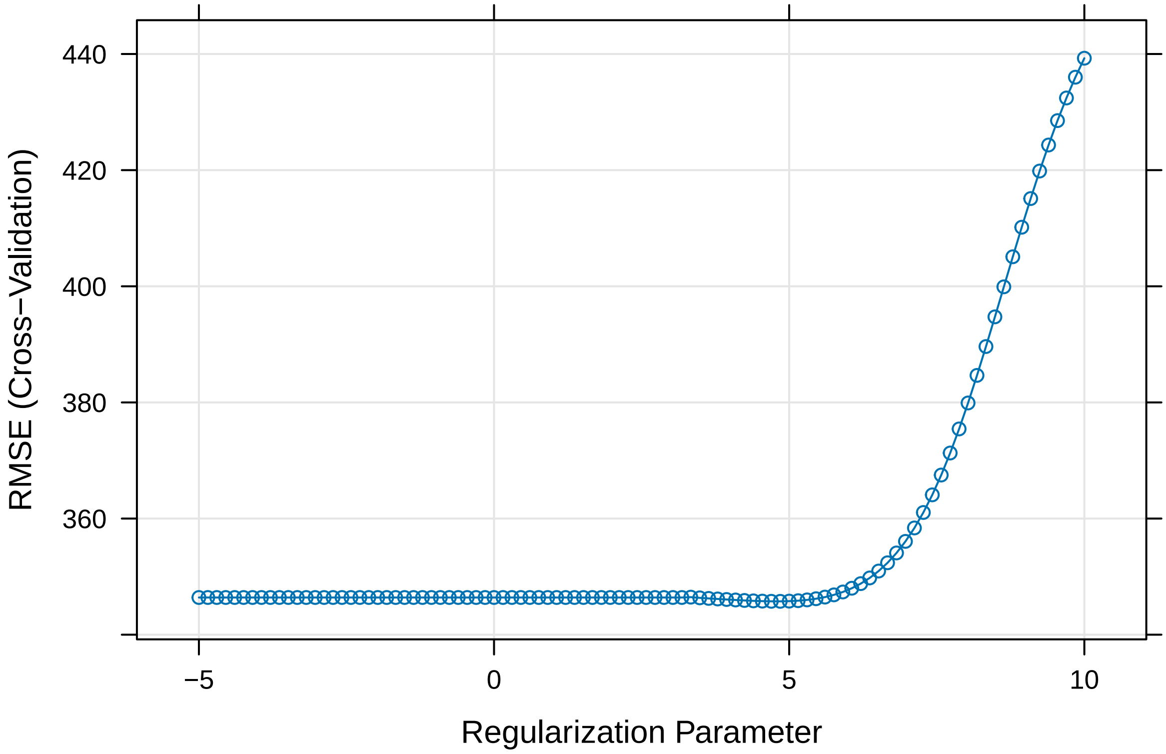
```
##                  lambda.min
## -Bobby Bonilla     392.2585
## -Brian Downing     751.8110
## -Billy Hatcher     167.3150
## -Bill Schroeder    260.5651
## -Chris Bando       338.3910
## -Chili Davis       747.5390
```

```
# 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")
```

Using caret

Ridge regression

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


# set.seed(2)
# ridge.fit <- train(x, y,
#                    method = "glmnet",
#                    tuneGrid = expand.grid(alpha = 0,
#                                           lambda = exp(seq(10, -5, length=100))),
#                    trControl = ctrl1)


set.seed(2)
ridge.fit <- train(Salary ~ . ,
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(10, -5, length=100))),
                   trControl = ctrl1)

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




```
ridge.fit$bestTune
```

```
##    alpha  lambda
## 66     0 127.547
```

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

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  6.419272e+01
## AtBat       -8.668269e-02
## Hits         1.313982e+00
## HmRun       -1.169621e+00
## Runs         1.255705e+00
## RBI          4.476937e-01
## Walks        2.540621e+00
## Years       -5.025347e+00
## CAtBat       9.274057e-03
## CHits        7.662685e-02
## CHmRun       7.618388e-01
## CRuns        1.501816e-01
## CRBI         2.157345e-01
## CWalks      -4.639071e-03
## LeagueN      3.538937e+01
## DivisionW   -1.154709e+02
## PutOuts      1.860033e-01
## Assists      7.735345e-02
## Errors      -4.442285e+00
## NewLeagueN  -2.432437e+00
```

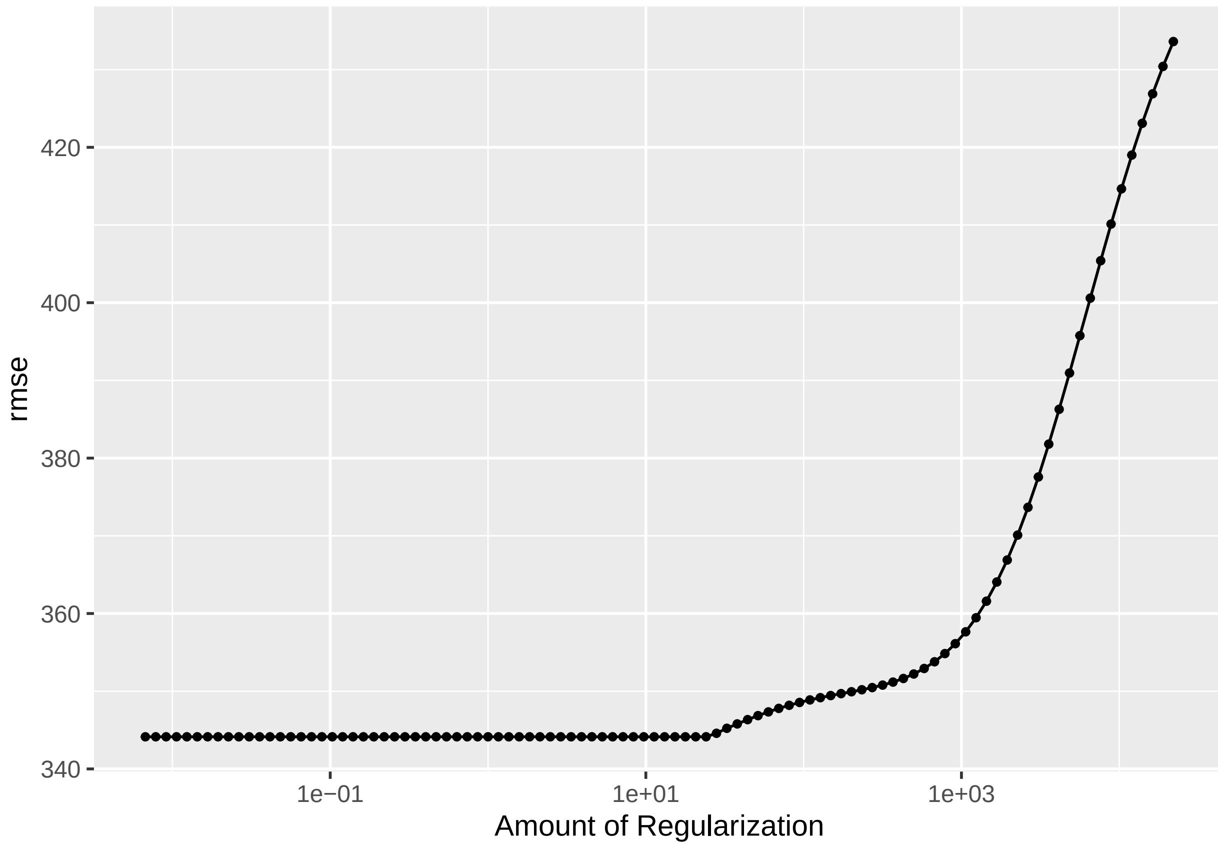
```r
# ridge.pred <- predict(ridge.fit, newdata = model.matrix(Salary ~ ., testing_data)[,-1])
ridge.pred <- predict(ridge.fit, newdata = testing_data)

# test error
mean((ridge.pred - testing_data[, "Salary"])^2)
```

```
## [1] 72518.15
```

Using tidymodels

Ridge regression

```r
# Setup the resampling method
set.seed(2)
cv_folds <- vfold_cv(training_data, v = 10)

# Model specification for ridge regression
ridge_spec <- linear_reg(penalty = tune(), mixture = 0) %>% # mixture = 0 for ridge regression
  set_engine("glmnet") %>%
  set_mode("regression")

# ridge_spec %>% extract_parameter_dials("penalty")

# Grid of tuning Parameters
ridge_grid_set <- parameters(penalty(range = c(-5, 10), trans = log_trans()))
ridge_grid <- grid_regular(ridge_grid_set, levels = 100)

# Set up the workflow
ridge_workflow <- workflow() %>%
  add_model(ridge_spec) %>%
  add_formula(Salary ~ .)

# Tune the model
ridge_tune <- tune_grid(
  ridge_workflow,
  resamples = cv_folds,
  grid = ridge_grid
)

# CV plot
autoplot(ridge_tune, metric = "rmse")
```




```r
# Select tuning parameters based on 1SE rule
ridge_1SE <- select_by_one_std_err(ridge_tune, metric = "rmse", desc(penalty))

# !!!
ridge_best <- select_best(ridge_tune, metric = "rmse")
cv_rmse <- ridge_tune %>% collect_metrics() %>% filter(.metric == "rmse")
cv_rmse_mean <- cv_rmse$mean
which(cv_rmse_mean == min(cv_rmse_mean))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## [51] 51 52 53 54
```

```r
# Update the model with the best lambda
final_ridge_spec <- ridge_spec %>%
  update(penalty = ridge_1SE$penalty)

# Fit your final model to the train data
ridge_fit <- fit(final_ridge_spec, formula = Salary ~ ., data = training_data)

# Get coefficients
ridge_model <- extract_fit_engine(ridge_fit)
coef(ridge_model, s = ridge_1SE$penalty)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  1.989361e+02
## AtBat        9.320758e-02
```

```
## Hits          4.080310e-01
## HmRun         1.221729e+00
## Runs          6.724653e-01
## RBI           6.554699e-01
## Walks         9.400550e-01
## Years         2.473188e+00
## CAtBat        8.649168e-03
## CHits         3.346335e-02
## CHmRun        2.833809e-01
## CRuns         6.797791e-02
## CRBI          7.590009e-02
## CWalks        7.106475e-02
## LeagueN       3.490156e+00
## DivisionW    -2.852986e+01
## PutOuts       5.466209e-02
## Assists       9.524413e-04
## Errors       -5.443143e-01
## NewLeagueN    2.802890e+00
```

```r
# prediction
ridge_pred <- predict(ridge_fit, new_data = testing_data)

# test MSE
test_error <- mean((testing_data$Salary - ridge_pred$.pred)^2)
test_error
```

```
## [1] 71450.61
```