Ridge Regression

Yifei Sun, Runze Cui

Contents

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

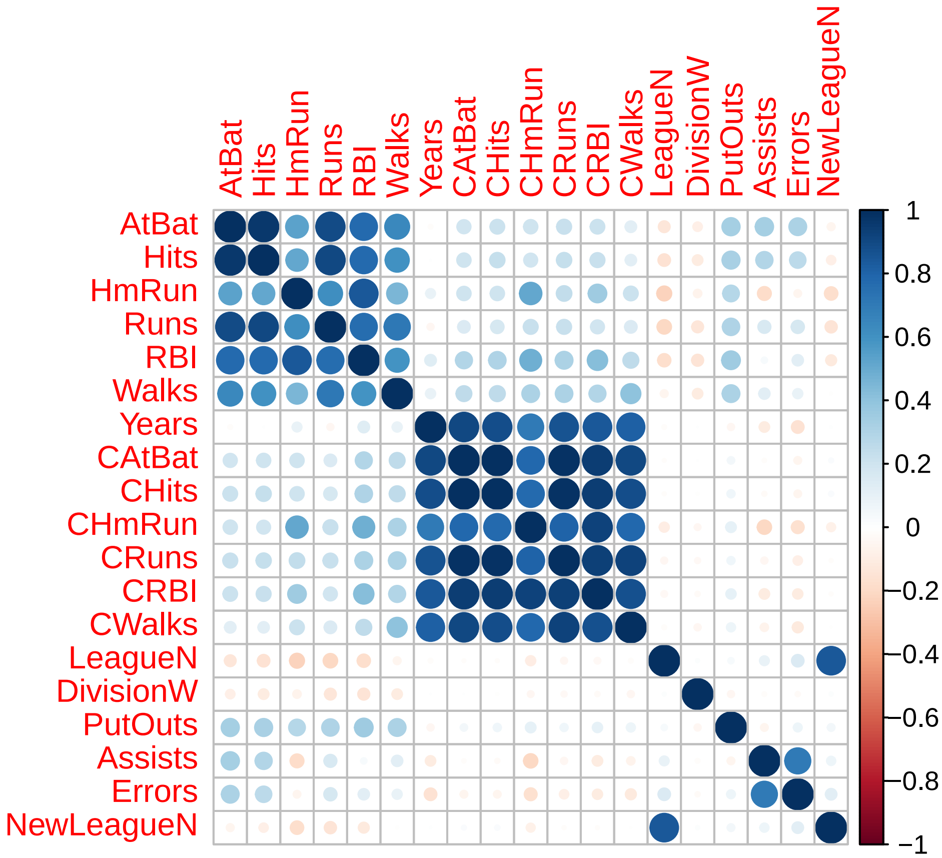
Predict a baseball player's salary on the basis of various statistics associated with performance in the previous year. Use `?Hitters` for more details.

```
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

```
# 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)))
```

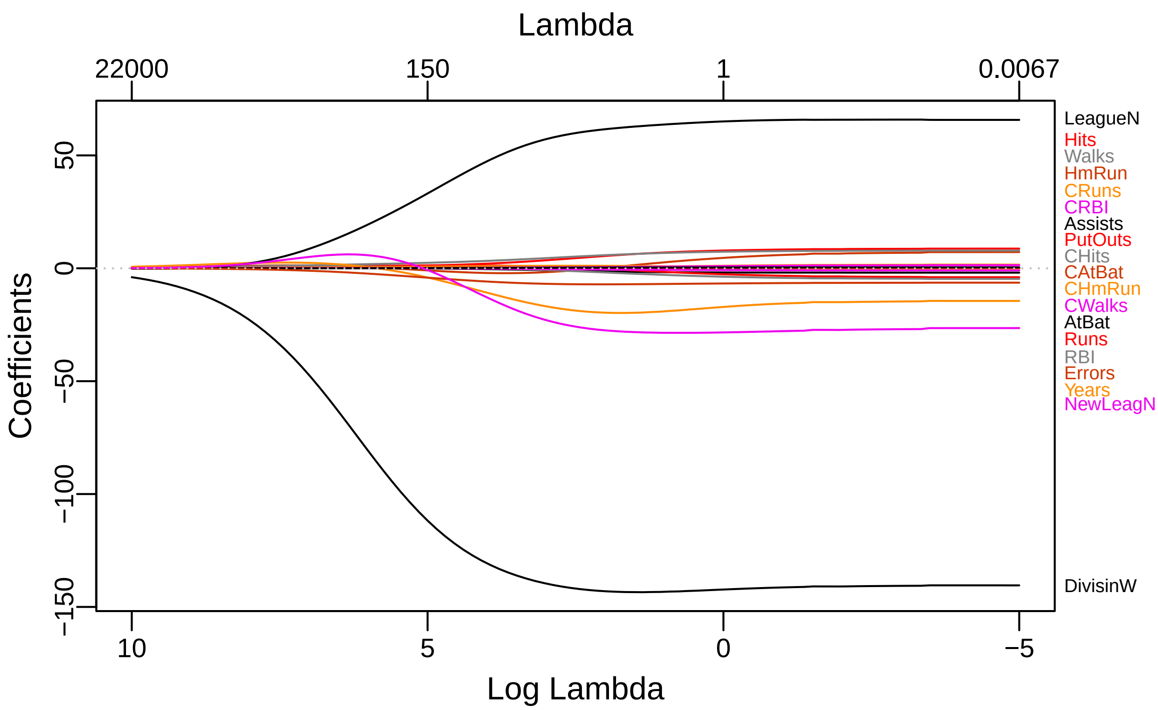
`coef(ridge.mod)` gives the coefficient matrix. Each column is the fit corresponding to one lambda value.

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

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

Trace plot

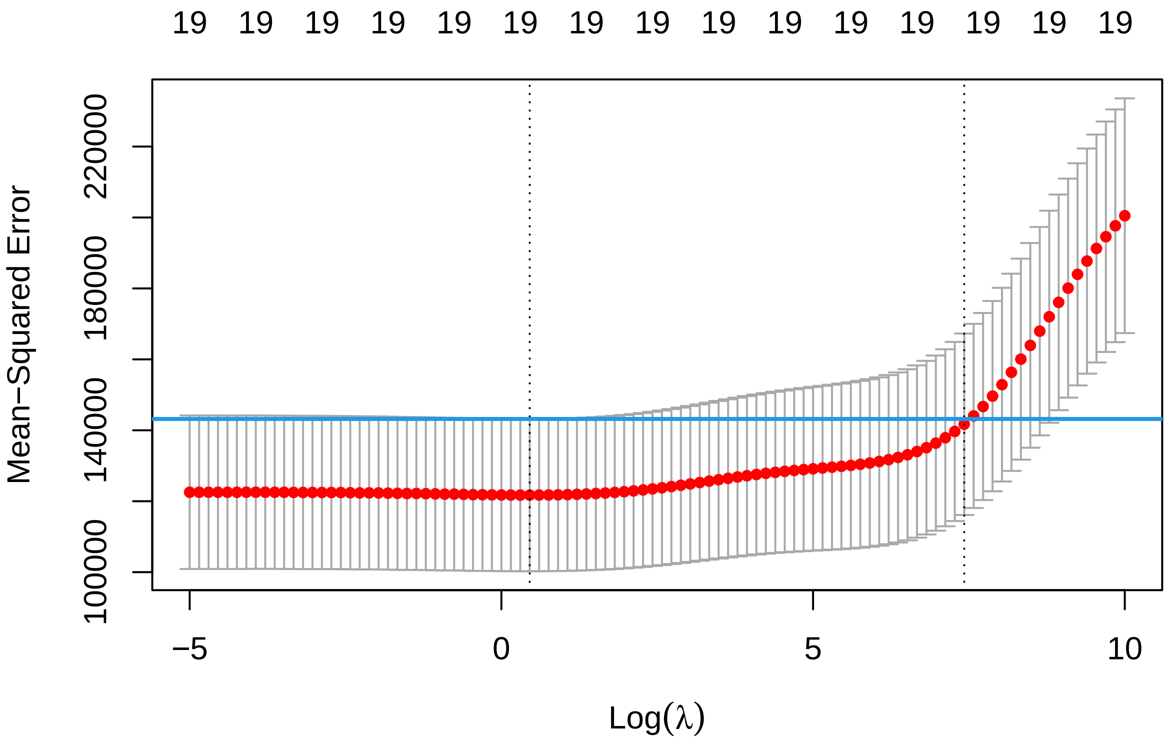
```
# plot(ridge.mod, xvar = "lambda", label = TRUE)
plot_glmnet(ridge.mod, xvar = "rlambda", label = 19)
```



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)
```




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

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

```
# 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.

```
# 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
```

```r
# 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
```

```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")
```

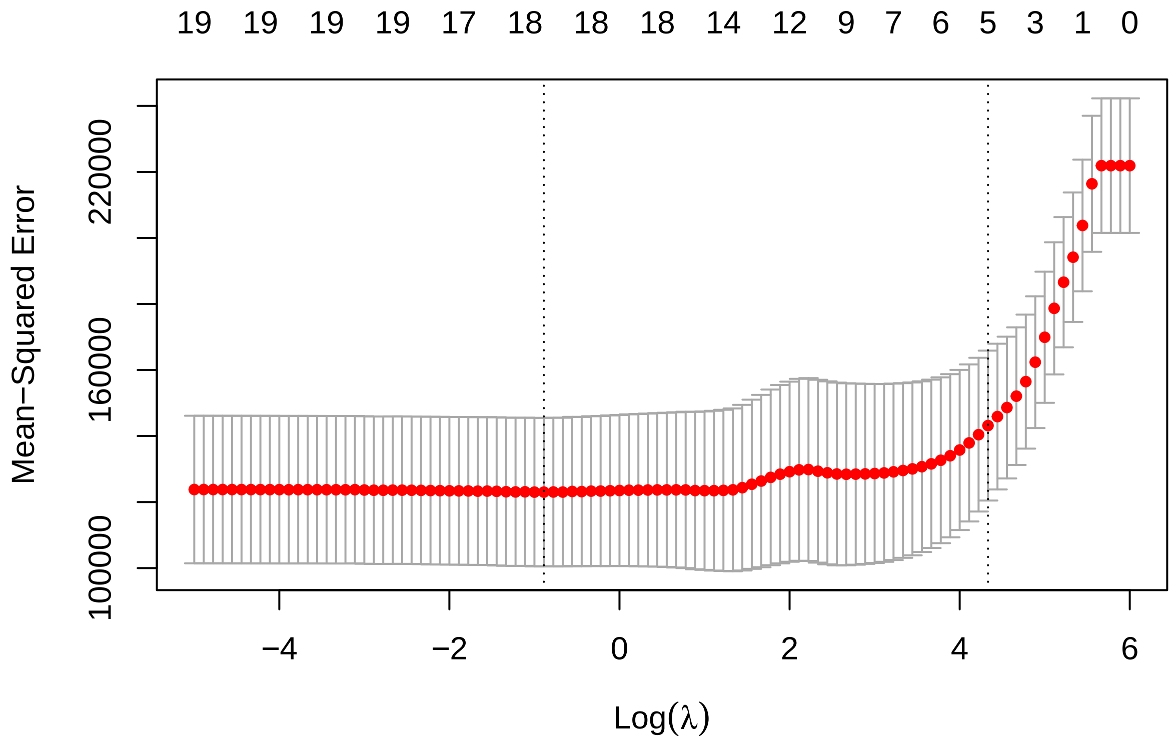
Lasso

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

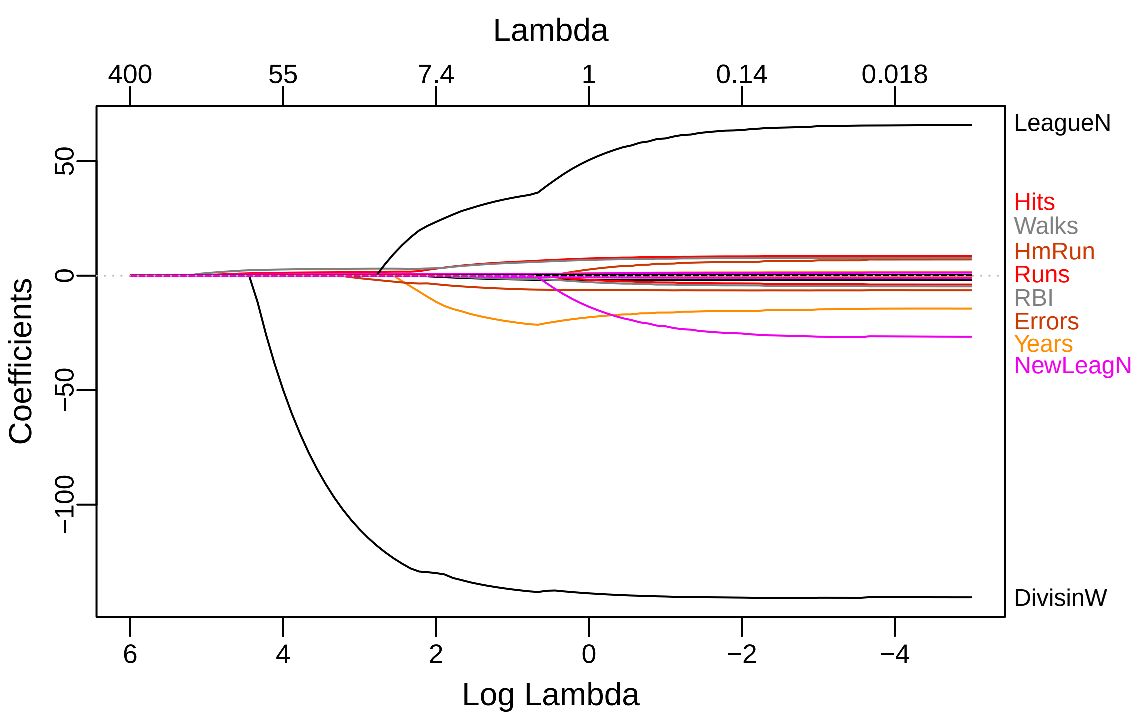
```r
cv.lasso <- cv.glmnet(x, y,
                      alpha = 1,
                      lambda = exp(seq(6, -5, length = 100)))

cv.lasso$lambda.min
```

```
## [1] 0.4111123
```

```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"
##                 lambda.min
```

```
## (Intercept)  252.26294072
## AtBat         -1.90534710
## Hits           8.15158167
## HmRun          5.21665845
## Runs          -2.96842653
## RBI           -3.88269897
## Walks          7.42456593
## Years        -16.11287037
## CAtBat        -0.12910586
## CHits          .
## CHmRun         0.03941621
## CRuns          1.33679040
## CRBI           1.14477567
## CWalks        -0.84057637
## LeagueN       59.68175036
## DivisionW   -140.03617879
## PutOuts        0.23563138
## Assists        0.36514255
## Errors        -6.37340917
## NewLeagueN   -21.80787446
```

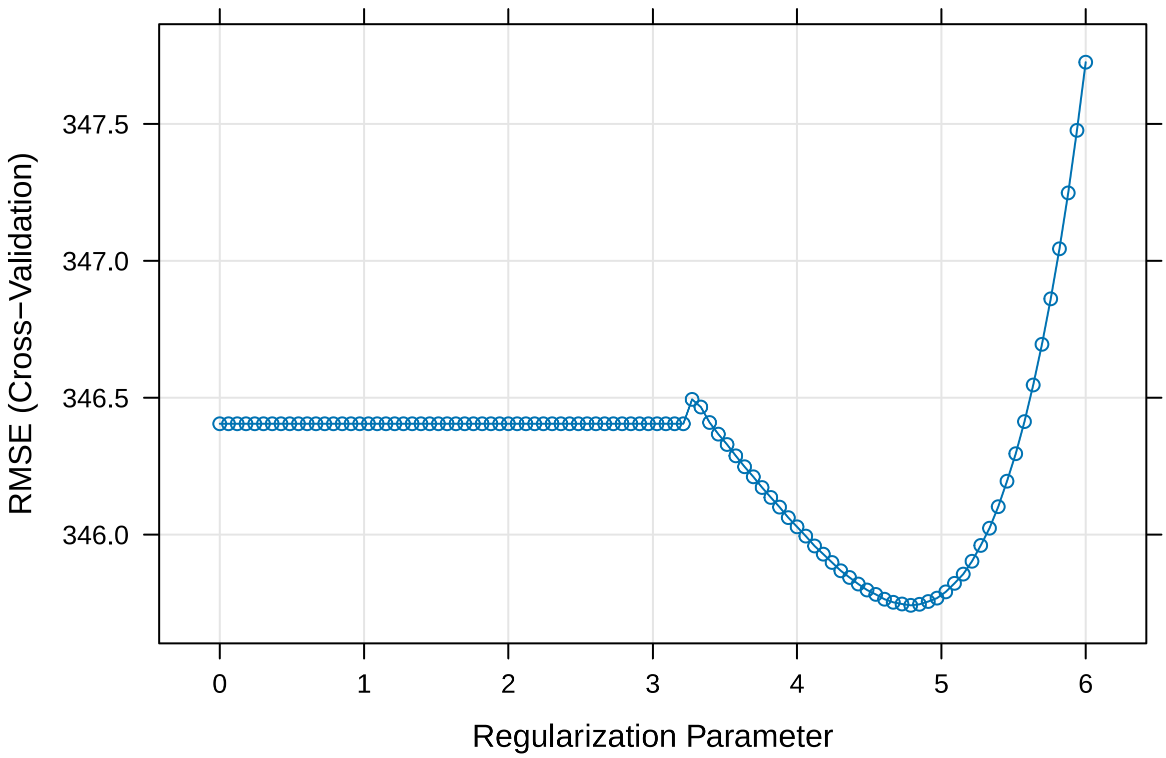
```
head(predict(cv.lasso, newx = model.matrix(Salary ~ ., testing_data)[,-1],
             s = "lambda.min", type = "response"))
```

```
##                  lambda.min
## -Bobby Bonilla     393.0056
## -Brian Downing     744.4067
## -Billy Hatcher     163.2661
## -Bill Schroeder    245.7250
## -Chris Bando       336.6702
## -Chili Davis       747.8962
```

Using caret

Ridge regression

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

set.seed(2)
ridge.fit <- train(Salary ~ . ,
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(6, 0, length=100))),
                   trControl = ctrl1)

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




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

```
##    alpha   lambda
## 80     0 120.0465
```

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

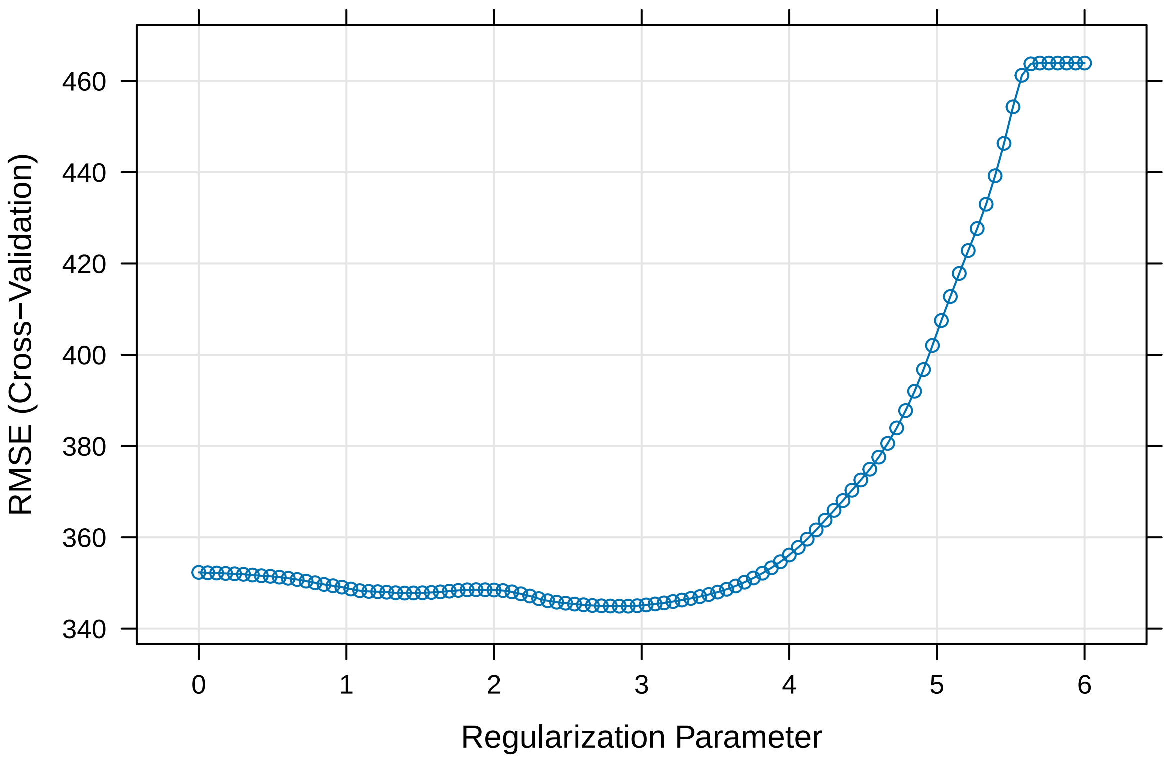
```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  6.658526e+01
## AtBat       -1.007003e-01
## Hits         1.352940e+00
## HmRun       -1.254633e+00
## Runs         1.260047e+00
## RBI          4.243512e-01
## Walks        2.583969e+00
## Years       -5.400280e+00
## CAtBat       9.054693e-03
## CHits        7.814029e-02
## CHmRun       7.760040e-01
## CRuns        1.528742e-01
## CRBI         2.209461e-01
## CWalks      -1.155740e-02
## LeagueN      3.626888e+01
## DivisionW   -1.168444e+02
## PutOuts      1.882210e-01
## Assists      8.157828e-02
## Errors      -4.552646e+00
## NewLeagueN  -3.106719e+00
```

Lasso

```
set.seed(2)
lasso.fit <- train(Salary ~ .,
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(6, 0, length = 100))),
                   trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

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




```
lasso.fit$bestTune
```

```
##    alpha   lambda
## 48     1 17.26161
```

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

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  4.658078e+01
## AtBat        .
## Hits         1.653581e+00
## HmRun        .
## Runs         .
## RBI          .
## Walks        3.065915e+00
```

```
## Years         .
## CAtBat        .
## CHits         .
## CHmRun        2.485882e-03
## CRuns         7.590546e-02
## CRBI          6.040837e-01
## CWalks        .
## LeagueN       .
## DivisionW   -1.157339e+02
## PutOuts       1.691929e-01
## Assists       .
## Errors      -1.626897e+00
## NewLeagueN    .
```

Elastic net

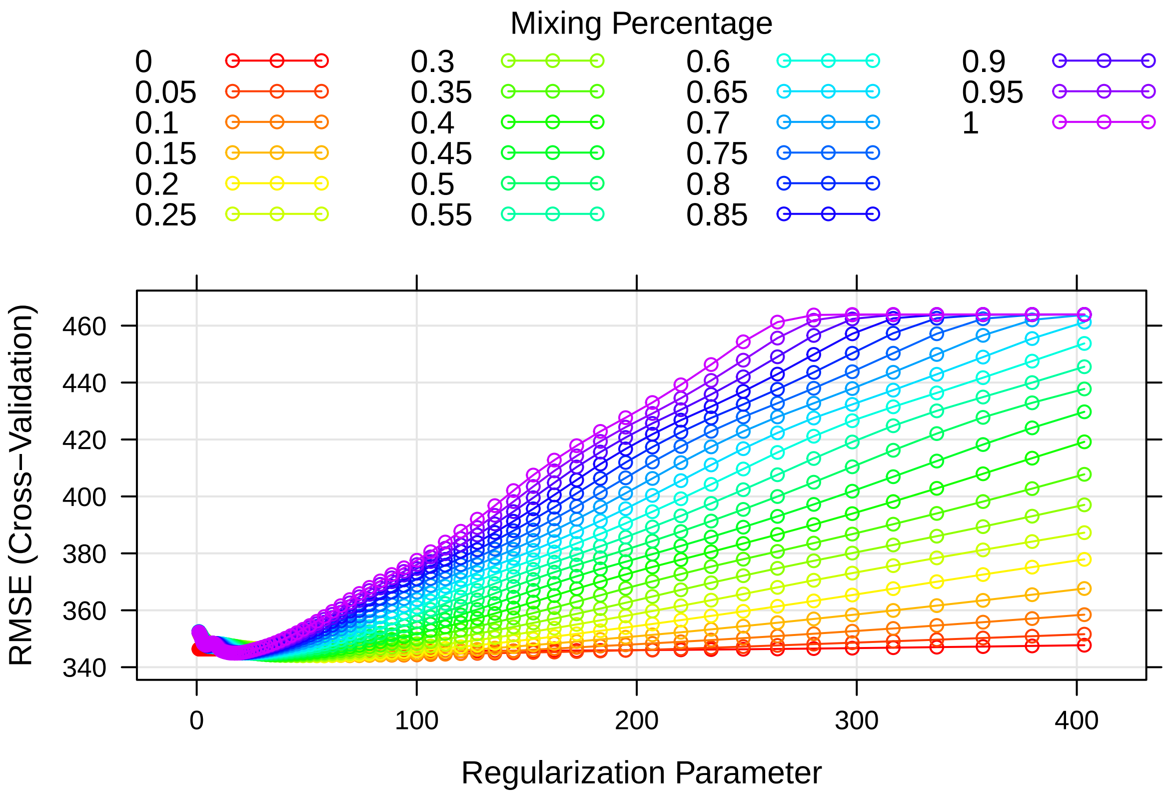
```
set.seed(2)
enet.fit <- train(Salary ~ .,
                  data = training_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(6, 0, length = 100))),
                  trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
enet.fit$bestTune
```

```
##     alpha   lambda
## 468   0.2 58.00946
```

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

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




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

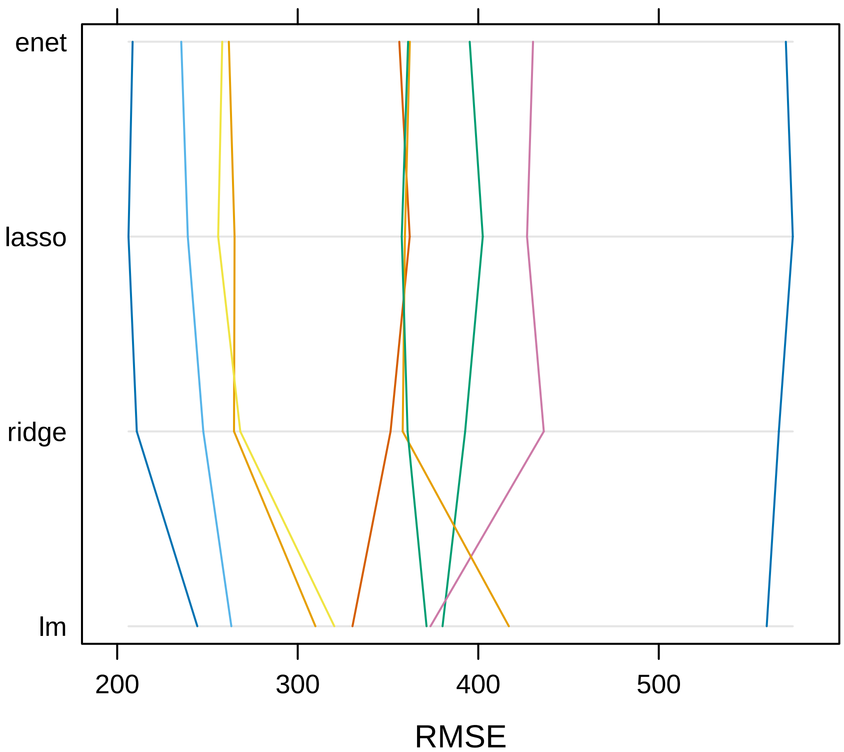
```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)   44.52822002
## AtBat          .
## Hits           1.52211332
## HmRun          .
## Runs           0.39390907
## RBI            .
## Walks          2.78161816
## Years          .
## CAtBat         .
## CHits          0.06181915
## CHmRun         0.67548343
## CRuns          0.14223025
## CRBI           0.25980591
## CWalks         .
## LeagueN       14.00730342
## DivisionW   -114.61716141
## PutOuts        0.17811859
## Assists        .
## Errors        -2.13644344
## NewLeagueN     .
```

Comparing different models

```
set.seed(2)
lm.fit <- train(Salary ~ .,
                data = training_data,
                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: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  164.9253 204.1572 249.3332 244.4124 270.8353 334.3207    0
## lasso 163.2809 205.4058 249.4258 246.1971 276.9839 337.3778    0
## ridge 164.3292 205.4469 248.4625 244.9825 269.6162 331.6979    0
## lm    183.9196 227.4910 266.1303 257.0003 283.0533 335.8049    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  208.5461 259.0966 358.7015 343.9540 386.9757 570.3850    0
## lasso 206.2545 258.2120 358.5009 344.9134 392.3931 574.2304    0
## ridge 210.8576 265.5782 354.7824 345.7418 384.7683 566.5050    0
## lm    244.3868 312.4299 350.8410 356.9864 378.5440 559.7796    0
##
## Rsquared
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet  1.845297e-04 0.4409921 0.5334431 0.5119499 0.6626711 0.7321554    0
## lasso 1.049530e-06 0.4423553 0.5419278 0.5104701 0.6684236 0.7264085    0
## ridge 7.444238e-04 0.4176904 0.5149058 0.5006795 0.6572723 0.7344016    0
## lm    2.382421e-02 0.3586180 0.5476438 0.4683147 0.5830379 0.6794523    0
```

```
parallelplot(resamp, metric = "RMSE")
```




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

Prediction

```
enet.pred <- predict(enet.fit, newdata = testing_data)
# test error
mean((enet.pred - testing_data[, "Salary"])^2)
```

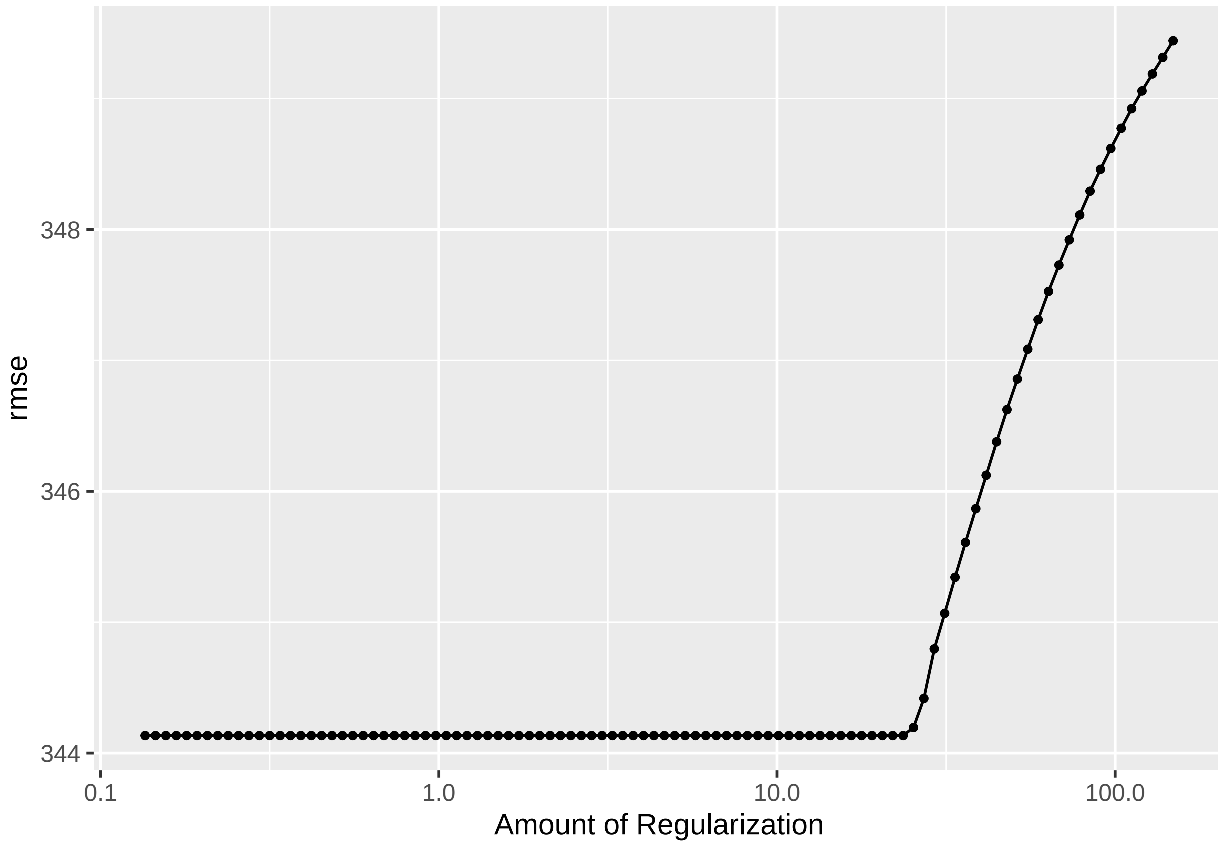
```
## [1] 71478.69
```

Using `tidymodels`

Ridge regression

```
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(-2, 5), trans = log_trans()))
ridge_grid <- grid_regular(ridge_grid_set, levels = 100)

# Set up the workflow
```

```r
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,
  control = control_resamples(extract = extract_fit_parsnip, save_pred = TRUE)
)

# 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))

# !!! see "why_is_ridge_CV_curve_flat.R"
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 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
```

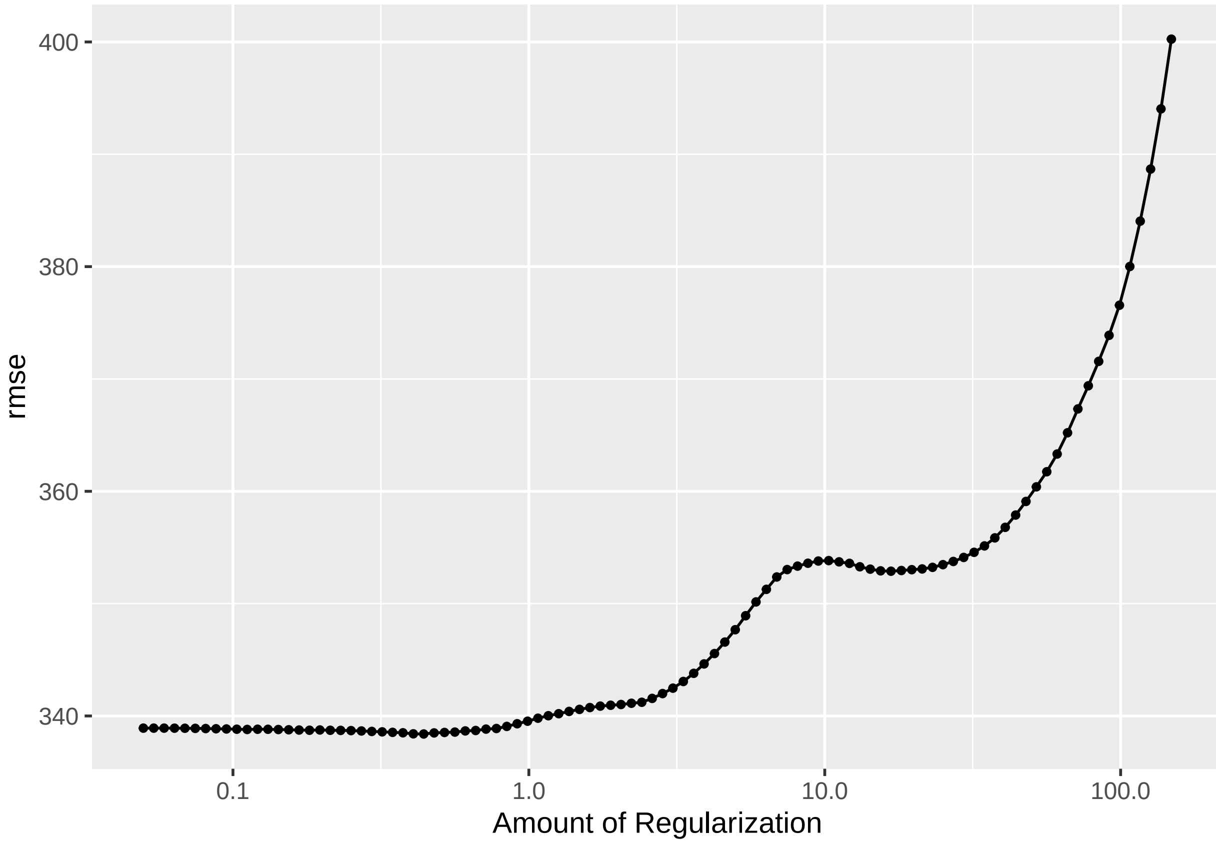
```r
# Update the model with the best lambda
final_ridge_spec <- ridge_spec %>%
  update(penalty = 1.3)

# 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)  5.861653e+01
## AtBat       -5.354971e-02
## Hits         1.221857e+00
## HmRun       -9.467771e-01
## Runs         1.242374e+00
## RBI          5.036152e-01
## Walks        2.433494e+00
## Years       -4.094684e+00
## CAtBat       9.741795e-03
## CHits        7.282548e-02
## CHmRun       7.251615e-01
## CRuns        1.437921e-01
## CRBI         2.033823e-01
## CWalks       1.200667e-02
## LeagueN      3.316672e+01
## DivisionW   -1.118010e+02
## PutOuts      1.801343e-01
## Assists      6.702114e-02
## Errors      -4.160429e+00
## NewLeagueN  -7.919486e-01
```

Lasso

```r
lasso_spec <- linear_reg(penalty = tune(), mixture = 1) %>% # mixture = 1 for lasso regression
  set_engine("glmnet") %>%
  set_mode("regression")

# lasso_spec %>% extract_parameter_dials("penalty")

lasso_grid_set <- parameters(penalty(range = c(-3, 5), trans = log_trans()))
lasso_grid <- grid_regular(lasso_grid_set, levels = 100)

lasso_workflow <- workflow() %>%
  add_model(lasso_spec) %>%
  add_formula(Salary ~ .)

lasso_tune <- tune_grid(
  lasso_workflow,
  resamples = cv_folds,
  grid = lasso_grid
```

```
)

autoplot(lasso_tune, metric = "rmse")
```



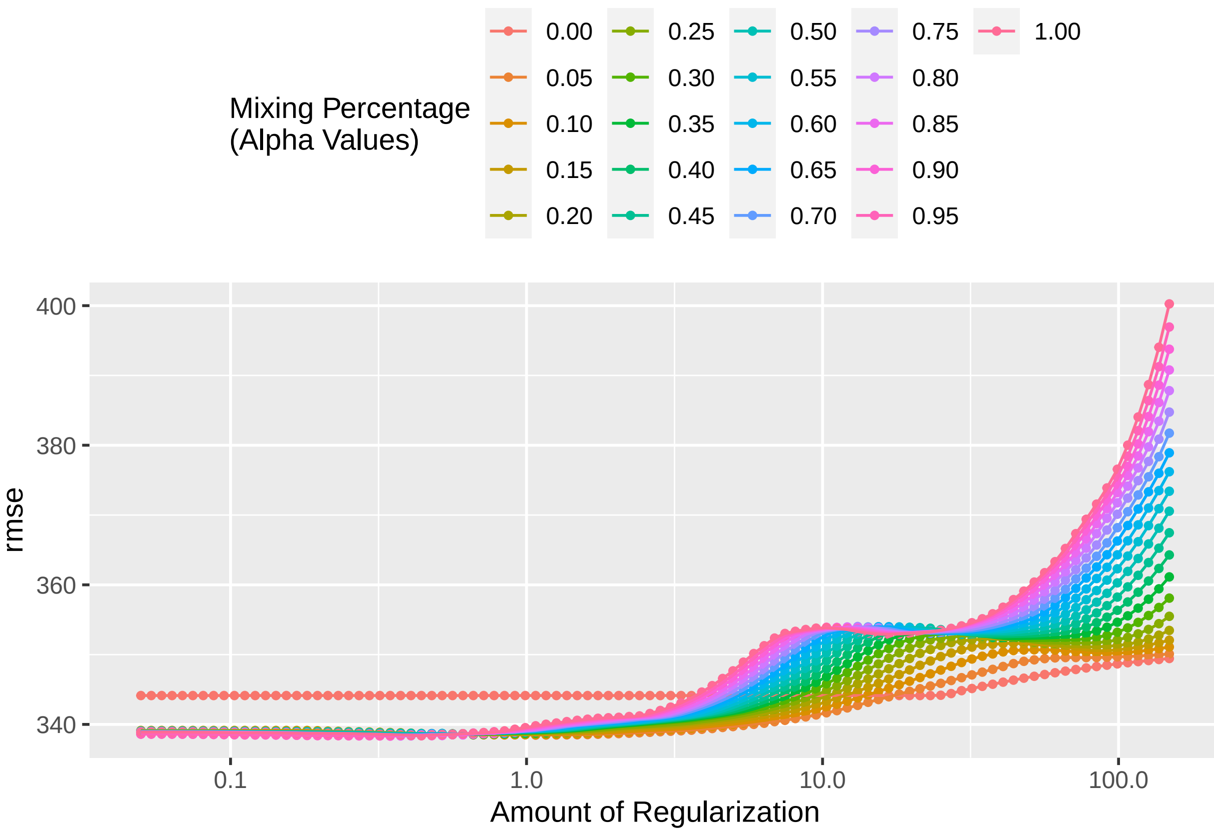

```
lasso_best <- select_best(lasso_tune, metric = "rmse")

final_lasso_spec <- lasso_spec %>%
  update(penalty = lasso_best$penalty)

lasso_fit <- fit(final_lasso_spec, formula = Salary ~ ., data = training_data)

lasso_model <- extract_fit_engine(lasso_fit)
coef(lasso_model, s = lasso_best$penalty)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  251.98015600
## AtBat         -1.90706009
## Hits           8.13049335
## HmRun          5.03747929
## Runs          -2.91855520
## RBI           -3.82012546
## Walks          7.40413119
## Years        -16.18698675
## CAtBat        -0.12649708
## CHits          .
## CHmRun         0.07876078
```

```
## CRuns           1.32715099
## CRBI            1.12699656
## CWalks         -0.83880880
## LeagueN        59.19338905
## DivisionW    -139.90200495
## PutOuts         0.23578289
## Assists         0.36344738
## Errors         -6.36103522
## NewLeagueN    -21.29990967
```

Elastic net

```
enet_spec <- linear_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet") %>%
  set_mode("regression")

# enet_spec %>% extract_parameter_dials("mixture")
# enet_spec %>% extract_parameter_dials("mixture")

enet_grid_set <- parameters(penalty(range = c(-3, 5), trans = log_trans()),
                            mixture(range = c(0, 1)))
enet_grid <- grid_regular(enet_grid_set, levels = c(100, 21))



enet_workflow <- workflow() %>%
  add_model(enet_spec) %>%
  add_formula(Salary ~ .)

enet_tune <- tune_grid(
  enet_workflow,
  resamples = cv_folds,
  grid = enet_grid
)

autoplot(enet_tune, metric = "rmse") +
  theme(legend.position = "top") +
  labs(color = "Mixing Percentage\n(Alpha Values)")
```



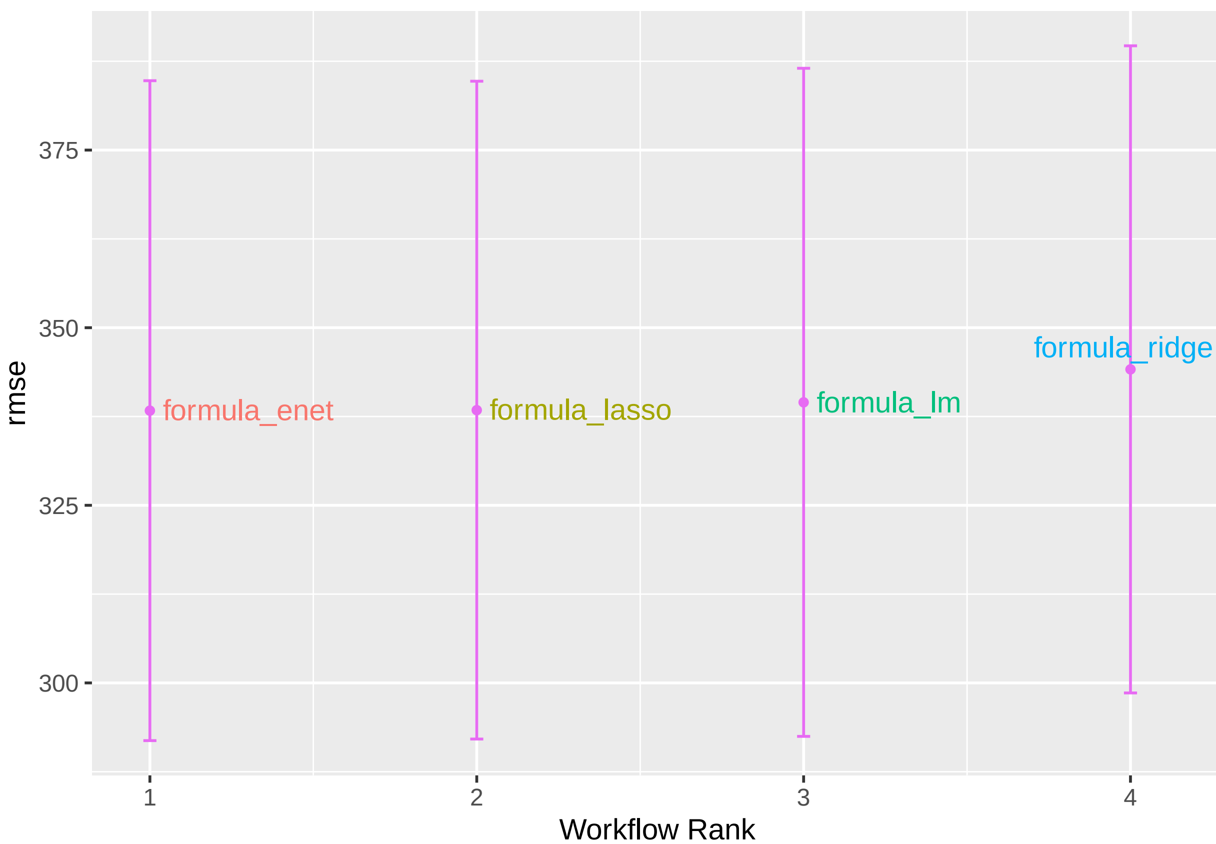

```r
enet_best <- select_best(enet_tune, metric = "rmse")

final_enet_spec <- enet_spec %>%
  update(penalty = enet_best$penalty, mixture = enet_best$mixture)

enet_fit <- fit(final_enet_spec, formula = Salary ~ ., data = training_data)

# Get coefficients
enet_model <- extract_fit_engine(enet_fit)
coef(enet_model, s = enet_best$penalty)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  253.08491855
## AtBat         -1.91195009
## Hits           8.23438240
## HmRun          5.44229593
## Runs          -3.10812552
## RBI           -3.99113557
## Walks          7.49790883
## Years        -15.89822178
## CAtBat        -0.13399311
## CHits          .
## CHmRun         0.03588247
## CRuns          1.36943136
## CRBI           1.15439630
## CWalks        -0.85379441
## LeagueN       61.00679148
```

```
## DivisionW   -140.22111587
## PutOuts        0.23628935
## Assists        0.37009397
## Errors        -6.39448993
## NewLeagueN   -22.98418529
```

Comparing different models

```
lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

model_compare <- workflow_set(preproc = list(Salary ~ .),
                              models = list(lm = lm_spec,
                                            lasso = final_lasso_spec,
                                            ridge = final_ridge_spec,
                                            enet = final_enet_spec)) %>%
  workflow_map(resamples = cv_folds)

autoplot(model_compare, metric = "rmse") +
  geom_text_repel(aes(label = wflow_id, color = wflow_id),
                  nudge_x = 1/8, nudge_y = 1/100) +
  theme(legend.position = "none")
```



Prediction

```r
enet_pred <- predict(enet_fit, new_data = testing_data)

# Calculate test RMSE
sqrt(mean((enet_pred[[1]] - testing_data$Salary)^2))
```

```
## [1] 299.8814
```