models_total_Candice

Candice Yu

2024-03-26

Contents

```r
library(caret)
library(earth)
library(tidyverse)
library(gridExtra)
```

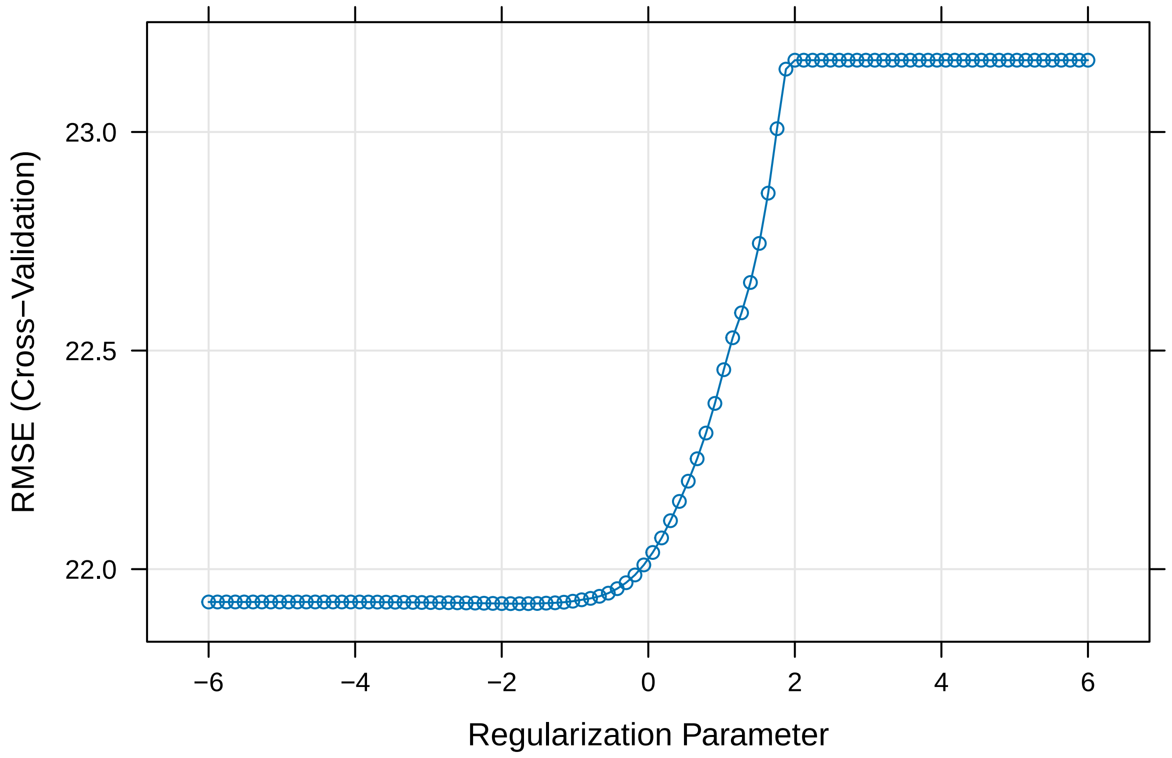
Load the training/test set & control method

```r
set.seed(2716)
# Load the training and test sets
train_data <- read.csv("./Data/train_data.csv")
test_data <- read.csv("./Data/test_data.csv")

# Load the control method
ctrl1 <- readRDS("./Data/train_control.rds")

# change variables to be factors again
train_data <- train_data %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         vaccine = as_factor(vaccine),
         severity = as_factor(severity))

test_data <- test_data %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         vaccine = as_factor(vaccine),
         severity = as_factor(severity))

# matrix of predictors
x <- train_data %>% select(-recovery_time)
y <- train_data$recovery_time

x_test <- test_data %>% select(-recovery_time)
y_test <- test_data$recovery_time
```

Model Training: Linear models

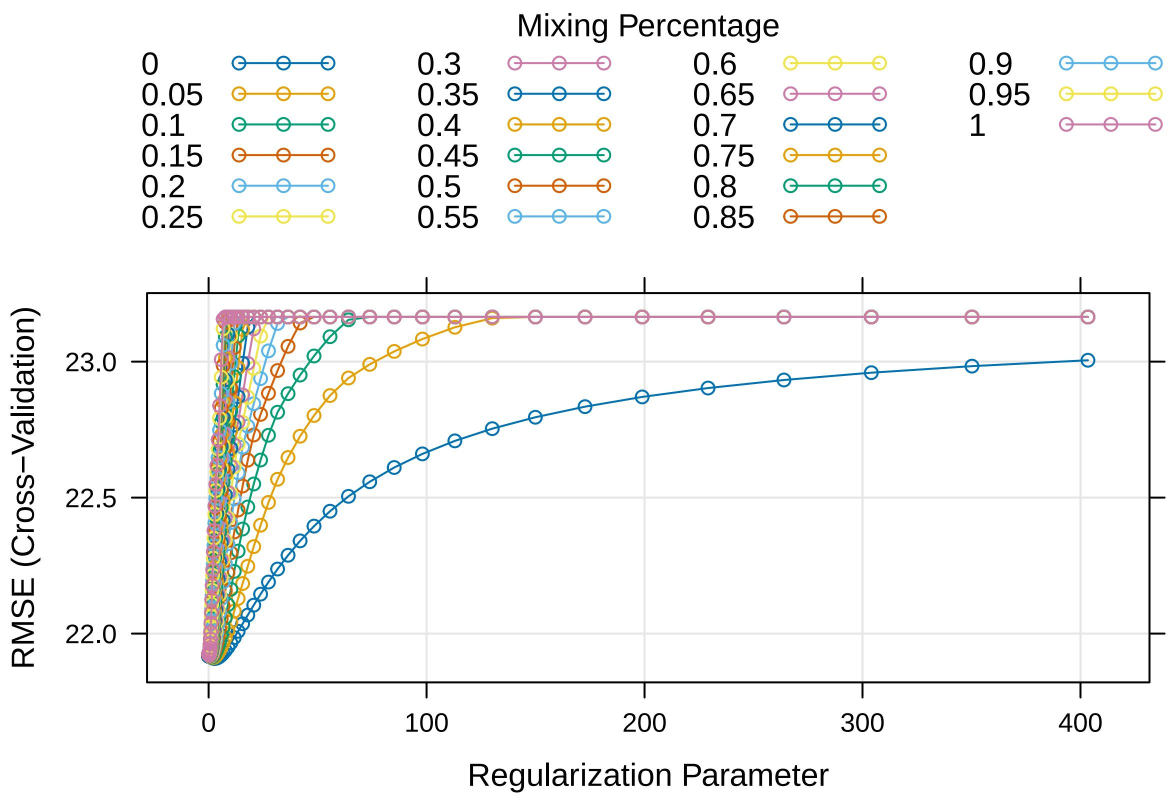
Lasso Regression Model

```r
set.seed(2716)
lasso_grid <- expand.grid(
  alpha = 1,
  lambda = exp(seq(-6, 6, length.out = 100))
)

lasso_fit <- train(x, y,
  method = "glmnet",
  tuneGrid = lasso_grid,
  trControl = ctrl1
)

plot(lasso_fit, xTrans = log)
```



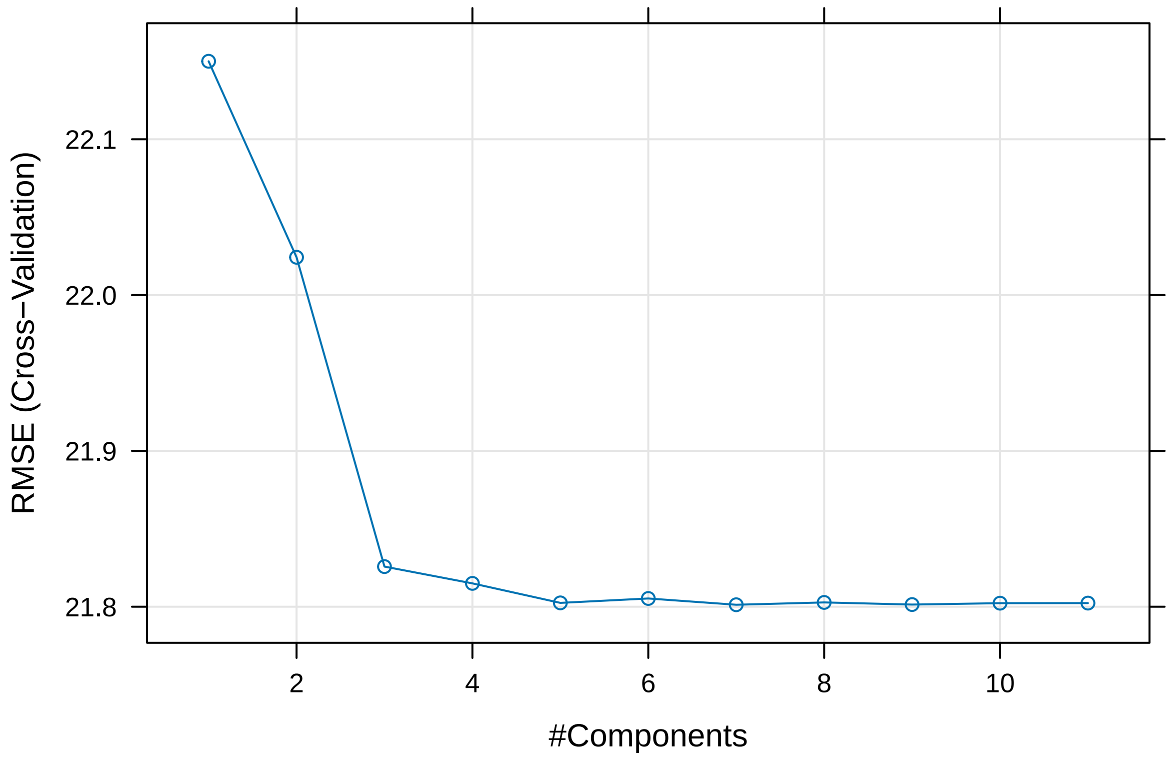
Elastic Net Model

```
set.seed(2716)
enet_grid <- expand.grid(
  alpha = seq(0, 1, length.out = 21),
  lambda = exp(seq(-8, 6, length.out = 100))
)

enet_fit <- train(x, y,
                  method = "glmnet",
                  tuneGrid = enet_grid,
                  trControl = ctrl1,
                  preProcess = c("center", "scale")
)
plot(enet_fit)
```



Partial Least Squares

```
set.seed(2716)
pls_fit <- train(x, y,
                 method = "pls",
                 tuneLength = 20,
                 trControl = ctrl1,
                 preProcess = c("center", "scale")
                 )
plot(pls_fit)
```



Evaluate the performance of linear models

```
lasso_pred <- predict(lasso_fit, newdata = x_test)
enet_pred <- predict(enet_fit, newdata = x_test)
pls_pred <- predict(pls_fit, newdata = x_test)

lasso_performance <- postResample(pred = lasso_pred, obs = test_data$recovery_time)
lasso_performance
```

```
##       RMSE   Rsquared        MAE
## 19.9896682  0.1042464 13.2470122
```

```
enet_performance <- postResample(pred = enet_pred, obs = test_data$recovery_time)
enet_performance
```

```
##       RMSE   Rsquared        MAE
## 19.9429346  0.1063538 13.1720658
```

```
pls_performance <- postResample(pred = pls_pred, obs = test_data$recovery_time)
pls_performance
```

```
##       RMSE   Rsquared        MAE
## 19.8802642  0.1166733 13.4239393
```

Model Training: Nonlinear Methods

The EDA plots show that the relationship between predictors and recovery time is likely non-linear, and there may be interactions between variables, especially considering the difference between study groups A and B.

Given the results from the EDA plots and the nature of the data, both generalized additive models (GAM) and multivariate adaptive regression splines (MARS) could be suitable choices for modeling. They both are capable of modeling complex, non-linear relationships in the data.

Multivariate Adaptive Regression Spline (MARS)

Build the MARS model

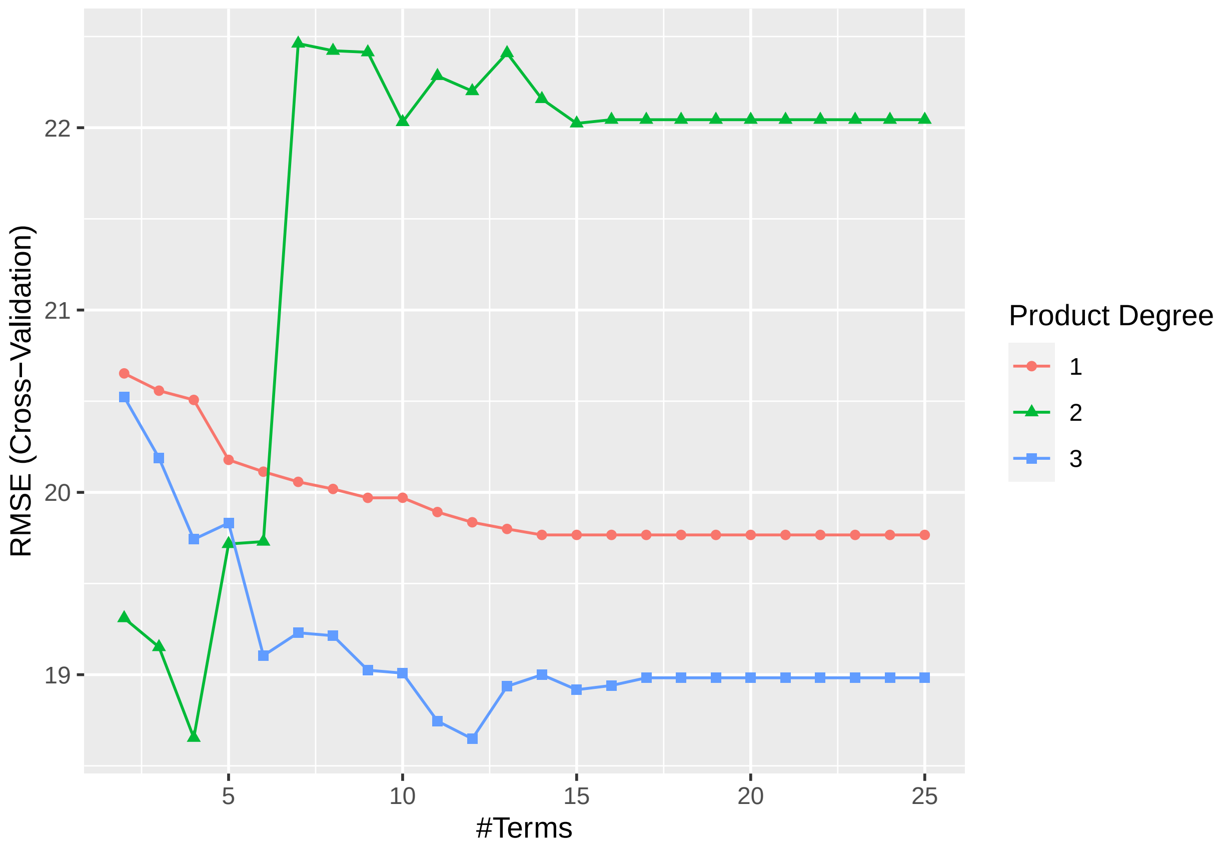
```r
# train the MARS model
mars_grid <- expand.grid(degree = 1:3, nprune = 2:25)

set.seed(2716) # set the same seed
mars_fit <- train(x, y,
                    method = "earth",
                    tuneGrid = mars_grid,
                    trControl = ctrl1)
```

MARS Model Summary

```r
# Model summary
summary(mars_fit)
```

```
## Call: earth(x=data.frame[2402,12], y=c(29,34,41,50,3...), keepxy=TRUE,
##             degree=3, nprune=12)
##
##                                 coefficients
## (Intercept)                       -8.0457447
## gender1                           -3.3771125
## vaccine1                          -5.6486741
## h(1-smoking)                      -3.3000670
## h(bmi-23.8)                        7.4964488
## h(31-bmi)                          7.2296297
## h(bmi-23.8) * severity1            1.3291655
## h(bmi-31) * studyB               -18.3936203
## smoking * h(bmi-31) * studyB      12.0251817
## h(age-62) * h(bmi-31) * studyB     4.6574999
## h(bmi-31) * h(112-LDL) * studyB    1.3157487
## h(bmi-31) * h(LDL-87) * studyB     0.8267105
##
## Selected 12 of 18 terms, and 8 of 12 predictors (nprune=12)
## Termination condition: Reached nk 25
## Importance: bmi, studyB, LDL, age, vaccine1, smoking, severity1, gender1, ...
## Number of terms at each degree of interaction: 1 5 2 4
## GCV 290.5539    RSS 681447.1    GRSq 0.4803533    RSq 0.4921888
```

```r
ggplot(mars_fit)
```



```r
mars_fit$bestTune
```

```
##    nprune degree
## 59     12      3
```

```r
coef(mars_fit$finalModel)
```

```
##                     (Intercept)                        h(31-bmi) 
##                      -8.0457447                        7.2296297 
##             h(bmi-31) * studyB  h(age-62) * h(bmi-31) * studyB 
##                     -18.3936203                        4.6574999 
##                     h(bmi-23.8) h(bmi-31) * h(112-LDL) * studyB 
##                       7.4964488                        1.3157487 
##                        vaccine1    smoking * h(bmi-31) * studyB 
##                      -5.6486741                       12.0251817 
##        h(bmi-23.8) * severity1  h(bmi-31) * h(LDL-87) * studyB 
##                       1.3291655                        0.8267105 
##                    h(1-smoking)                          gender1 
##                      -3.3000670                       -3.3771125
```

MARS Model Description:

The MARS model is a flexible regression method capable of uncovering complex nonlinear relationships between the dependent variable (recovery_time) and a set of independent variables. It does this by fitting piecewise linear regressions, which can adapt to various data shapes. This is particularly useful for modeling the recovery time from COVID-19 since the relationship between predictors and recovery time could be highly nonlinear and interaction-heavy.

Assumptions:

- The relationships between predictors and the response can be captured using piecewise linear functions.
- Interactions between variables can be important and are modeled by products of basis functions.
- There is no assumption of a parametric form of the relationship between predictors and the response.

Final Model Selection:

- The optimal hyperparameters were degree (degree of interaction) = 3 and nprune (number of terms) = 16.
- The selected model terms involve interactions between patient characteristics, their biometrics, the specific study group they belong to, and some non-linear transformations of these variables.

Evaluate performance on the test set

```
# Evaluate its performance on the test set:
predictions <- predict(mars_fit, newdata = test_data)
postResample(pred = predictions, obs = test_data$recovery_time)
```

```
##       RMSE   Rsquared        MAE
## 17.7117226  0.3132324 11.9569422
```

The results from evaluating the MARS model on the test set provide three key metrics:

1. **Root Mean Squared Error (RMSE):** RMSE measures the average magnitude of the prediction error. It represents the square root of the average squared differences between the predicted and actual values. An RMSE of 19.629 suggests that, on average, the model's predictions of the recovery time are about 19.629 days off from the actual recovery times.

2. **R-squared (R^2):** R^2 is a statistical measure that represents the proportion of the variance for the dependent variable that's explained by the independent variables in the model. In your case, the R^2 value is 0.2177, which means approximately 21.77% of the variance in the recovery time is explained by the model. This is a relatively low value, indicating that there is a lot of variability in the recovery time that is not captured by the model.

3. **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted values and the actual values, providing a linear score that reflects the average error magnitude without considering its direction. An MAE of 12.409 suggests that the model's predictions are, on average, 12.409 days different from the actual recovery time.

Interpretation

- The **RMSE** of 19.629 days is relatively high, depending on the context of the recovery times' range. If the typical recovery time is on the order of a few days, this is a substantial error. However, if recovery times are generally several weeks, the error may be more acceptable.

- The **R-squared** value of 0.2177 is not very high, suggesting that there might be other factors not included in the model that affect the recovery time. It also indicates that the relationship between the predictors and the recovery time has a significant amount of unexplained variability.

- The **MAE** gives us an indication that, despite the direction of the errors, the model's predictions are off by about two weeks on average. MAE is less sensitive to outliers than RMSE, so this value suggests that the model has a consistent average error across the test dataset.

GAM model

Build the GAM model

```
set.seed(2716) # set the same seed
gam_fit <- train(x = x, y = y,
                 method = "gam",
                 trControl = ctrl1)
```

Display the summary of the final model

```
gam_model_final <- gam_fit$finalModel
summary(gam_model_final)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender + hypertension + diabetes + vaccine + severity +
##     study + smoking + race + s(age) + s(SBP) + s(LDL) + s(bmi)
##
## Parametric coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      43.0689     1.1322  38.039  < 2e-16 ***
## gender1          -3.6317     0.7936  -4.576 4.97e-06 ***
## hypertension1     3.2473     0.7998   4.060 5.06e-05 ***
## diabetes1        -1.2603     1.1111  -1.134 0.256797
## vaccine1         -6.3587     0.8101  -7.849 6.26e-15 ***
## severity1         8.1497     1.2720   6.407 1.78e-10 ***
## studyB            4.6462     0.8468   5.487 4.52e-08 ***
## smoking           1.9839     0.5779   3.433 0.000607 ***
## race             -0.1192     0.3699  -0.322 0.747330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df       F p-value
## s(age) 3.908e-07      9   0.000   0.510
## s(SBP) 1.737e-06      9   0.000   0.395
## s(LDL) 3.093e-01      9   0.049   0.231
## s(bmi) 8.115e+00      9 109.190  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.327   Deviance explained = 33.2%
## GCV = 378.61  Scale est. = 375.87    n = 2402
```

Evaluate the GAM model's performance

```
test_predictions <- predict(gam_model_final, x_test)
postResample(pred = test_predictions, obs = test_data$recovery_time)
```

```
##       RMSE   Rsquared        MAE
## 18.5198607  0.2440446 12.7977052
```

Random Forest

Build the rf model

```
set.seed(2716)
# Parameters for Random Forest training
tunegrid <- expand.grid(mtry = 1:5)

# build the rf model
rf_fit <- train(
   x = x, y = y,
   method = "rf",
   trControl = ctrl1,
   tuneGrid = tunegrid
 )
rf_model_final <- rf_fit$finalModel
```

Evaluate the rf model's performance

```
# Calculate and print the RMSE for training and test datasets
rf_predictions <- predict(rf_model_final, x_test)
postResample(pred = rf_predictions, obs = test_data$recovery_time)
```

```
##       RMSE   Rsquared        MAE
## 18.0162887  0.2713562 12.1402869
```

Model Comparison

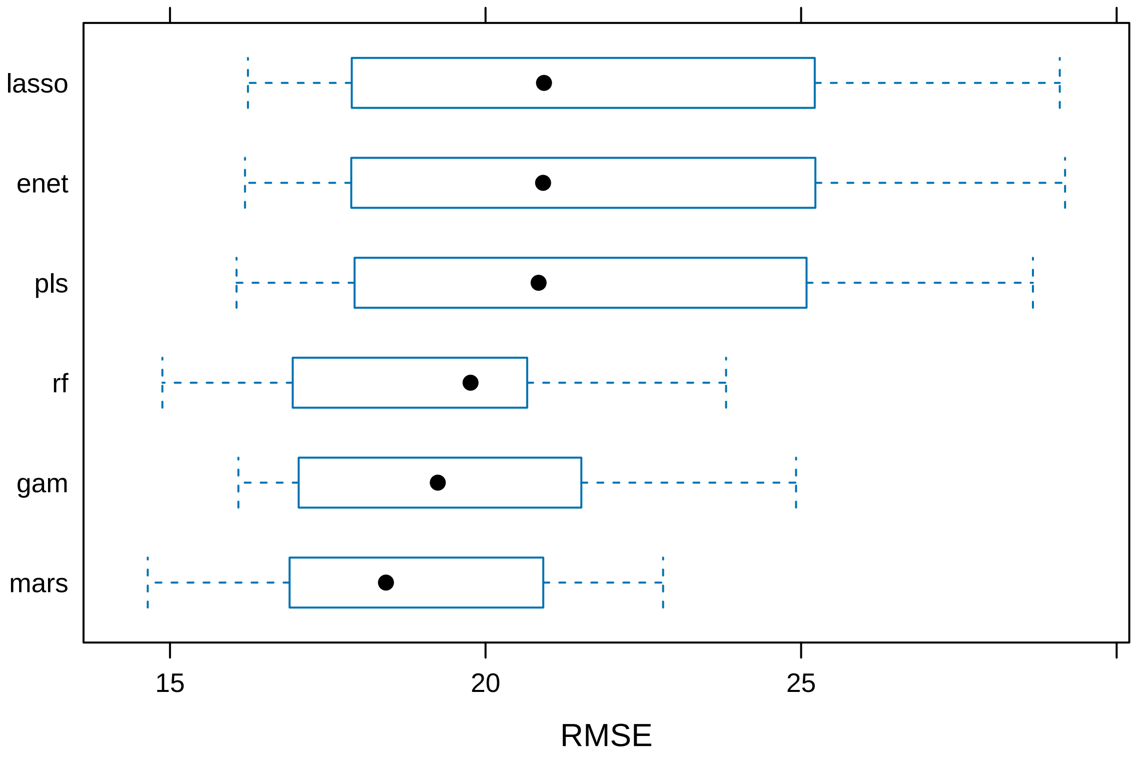
```
set.seed(2716)
resamp =
  resamples(list(lasso = lasso_fit,
                 gam = gam_fit,
                 enet = enet_fit,
                 pls = pls_fit,
                 mars = mars_fit,
                 rf = rf_fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, gam, enet, pls, mars, rf
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 12.19465 12.62816 12.94743 13.27089 13.62969 15.90977    0
## gam   11.57691 11.73191 12.43424 12.60549 12.98565 14.49104    0
## enet  12.15571 12.59841 12.82872 13.19966 13.55387 15.84683    0
## pls   12.08375 12.90874 13.16163 13.41886 13.78774 15.92179    0
## mars  10.89088 11.75502 11.98479 12.05695 12.47158 13.64296    0
## rf    10.91565 11.54185 11.98962 12.10608 12.30243 13.82018    0
##
```

```
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 16.23518 18.23494 20.92608 21.92098 25.14086 29.09870    0
## gam   16.08372 17.27037 19.24321 19.59045 21.38481 24.92036    0
## enet  16.18870 18.17792 20.91208 21.90863 25.15381 29.18282    0
## pls   16.05432 18.24747 20.84099 21.80127 24.95942 28.67325    0
## mars  14.64730 17.01084 18.42199 18.64837 20.45147 22.81323    0
## rf    14.87951 17.00871 19.76240 18.86915 20.45101 23.81148    0
##
## Rsquared
##             Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.04833821 0.09323473 0.1281056 0.1129636 0.1339402 0.1422558    0
## gam   0.13502675 0.19667929 0.2719358 0.2929828 0.3944992 0.4462167    0
## enet  0.05393326 0.09334982 0.1278820 0.1133779 0.1349355 0.1408792    0
## pls   0.05980481 0.10981191 0.1254004 0.1239830 0.1460552 0.1665966    0
## mars  0.13026481 0.24311171 0.3585692 0.3627122 0.5139298 0.5734756    0
## rf    0.20475307 0.22395088 0.3016613 0.3482261 0.4918171 0.5299956    0
```

Using bw-plot to compare their RMSE

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


lasso
enet
pls
rf
gam
mars
RMSE