models_total_Candice

Candice Yu

2024-03-26

Contents

Load the training/test set & control method **2**

Model Training: Linear models **2**
Lasso Regression Model 2
Elastic Net Model 3
Partial Least Squares 4
Evaluate the performance of linear models 5

Model Training: Nonlinear Methods **5**
Multivariate Adaptive Regression Spline (MARS) 6
GAM model 8
Random Forest 9
Model Comparison 10

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

Load the training/test set & control method

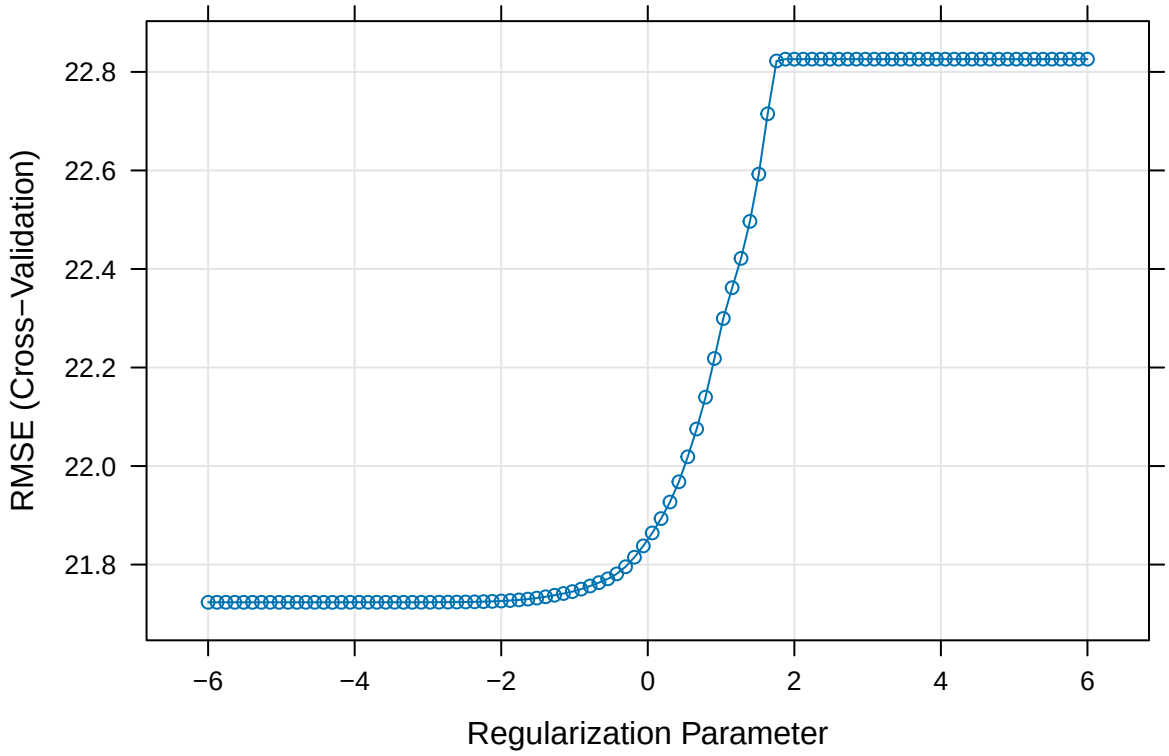
```r
set.seed(2716)
# Load the training and test sets
train_data <- read.csv("./Data/train_data.csv")
test_data <- read.csv("./Data/test_data.csv")

# 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

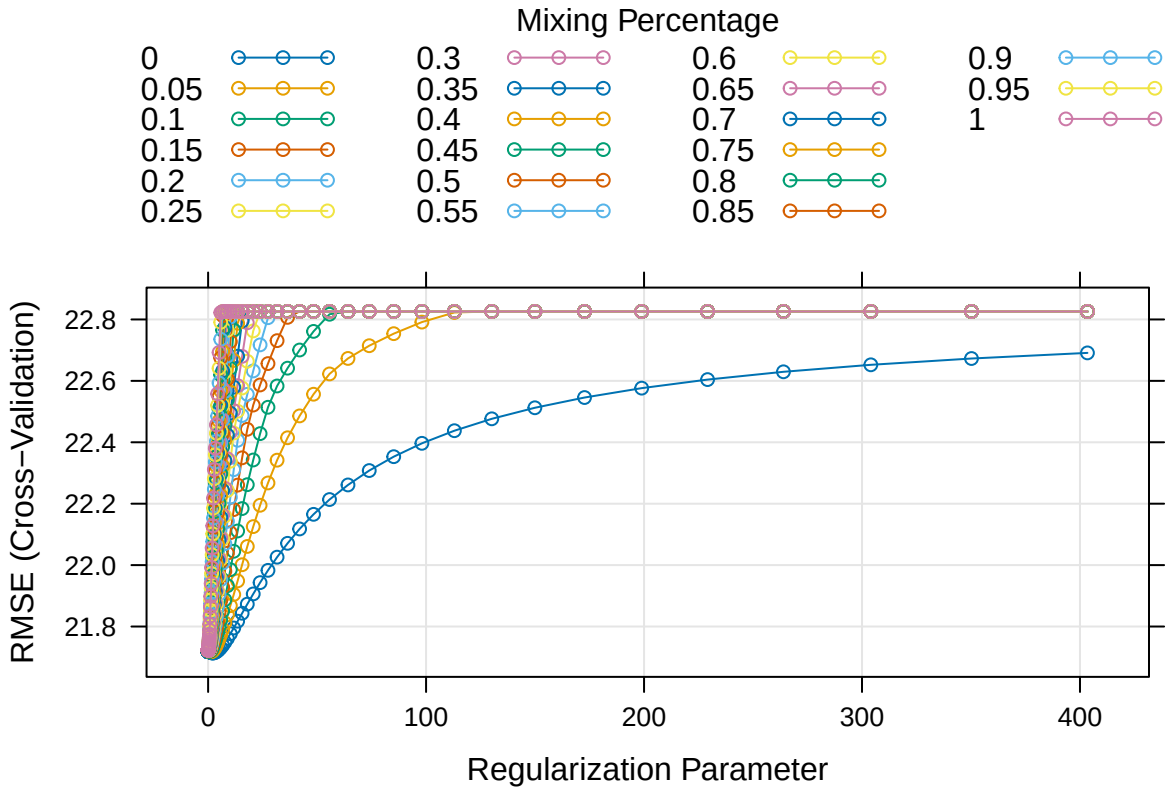
```r
set.seed(2716)
lasso_grid <- expand.grid(
  alpha = 1,
  lambda = exp(seq(-6, 6, length.out = 100))
)

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

```
)

plot(lasso_fit, xTrans = log)
```



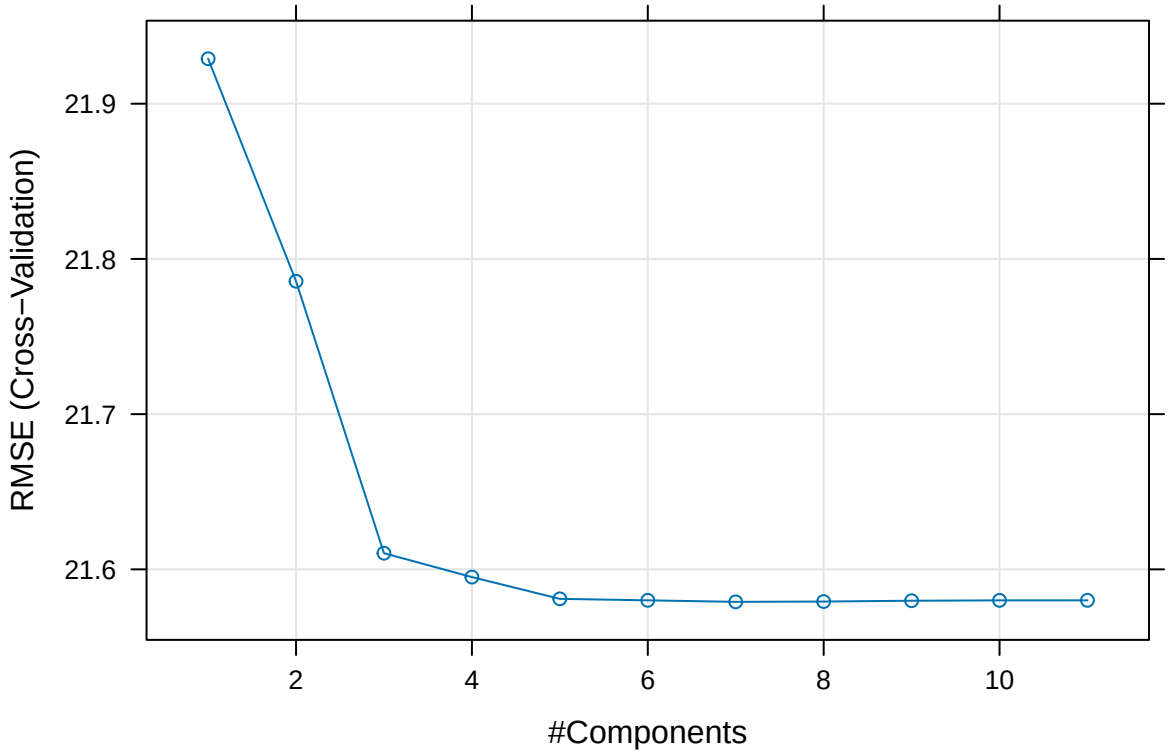
Elastic Net Model

```
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
## 21.6711297  0.1592601 12.5830955
```

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

```
##       RMSE   Rsquared        MAE
## 21.7323312  0.1589219 12.5660278
```

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

```
##       RMSE   Rsquared        MAE
## 21.6161627  0.1611026 12.8106190
```

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

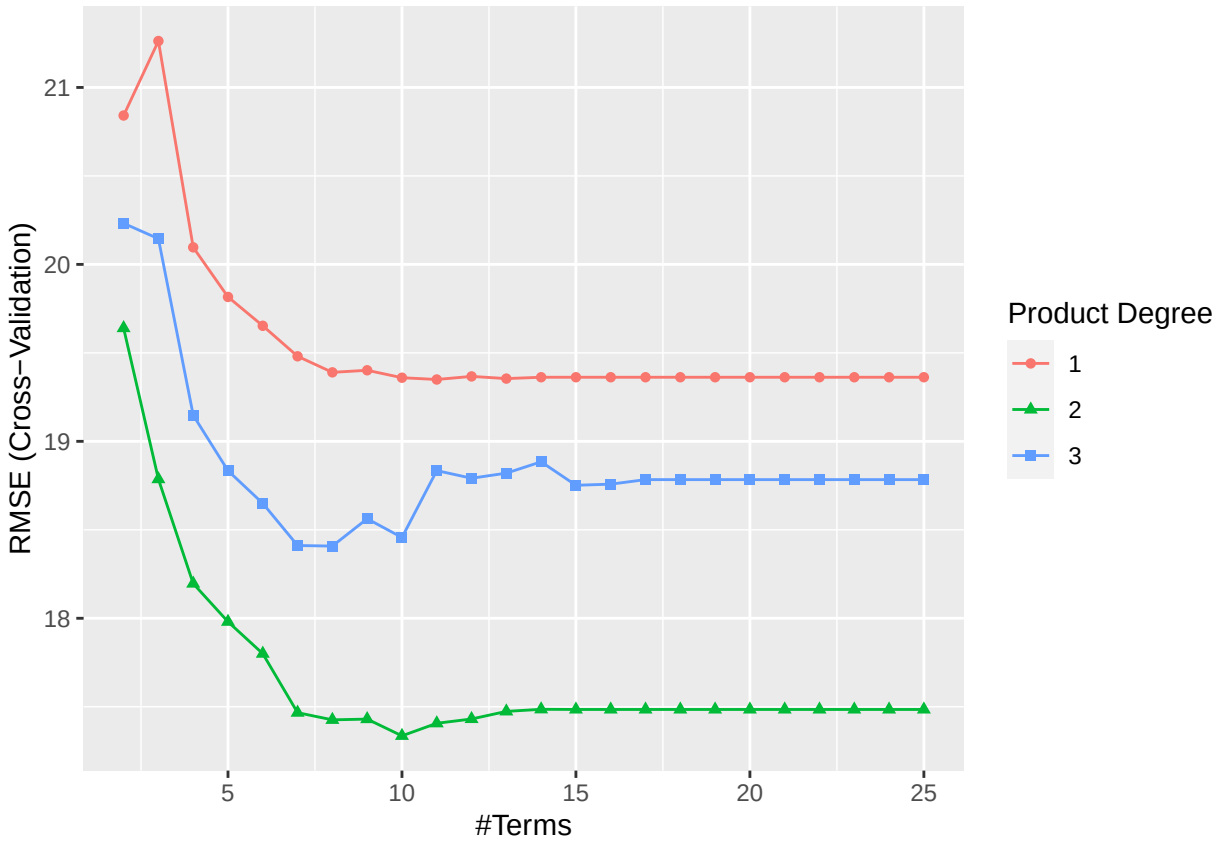
```
# train the MARS model
mars_grid <- expand.grid(degree = 1:3, nprune = 2:25)

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

MARS Model Summary

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

```
## Call: earth(x=data.frame[2402,12], y=c(44,29,40,31,5...), keepxy=TRUE,
##             degree=2, nprune=10)
##
##                       coefficients
## (Intercept)             10.008474
## gender1                 -3.241867
## hypertension1            3.723705
## vaccine1                -5.722648
## h(1-smoking)            -3.115184
## h(bmi-25)                6.228884
## h(30.5-bmi)              5.517656
## severity1 * studyB      16.689471
## h(bmi-30.5) * studyB    13.947100
## h(bmi-35.3) * studyB   214.977746
##
## Selected 10 of 16 terms, and 7 of 12 predictors (nprune=10)
## Termination condition: Reached nk 25
## Importance: bmi, studyB, severity1, vaccine1, hypertension1, gender1, ...
## Number of terms at each degree of interaction: 1 6 3
## GCV 291.0637    RSS 685521.9    GRSq 0.4526657    RSq 0.4628759
```

```
ggplot(mars_fit)
```




```
mars_fit$bestTune
```

```
##    nprune degree
## 33     10      2
```

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

```
##          (Intercept)          h(30.5-bmi) h(bmi-30.5) * studyB
##            10.008474             5.517656            13.947100
##            h(bmi-25) h(bmi-35.3) * studyB             vaccine1
##             6.228884           214.977746            -5.722648
##        hypertension1              gender1         h(1-smoking)
##             3.723705            -3.241867            -3.115184
##   severity1 * studyB
##            16.689471
```

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
## 36.4666940  0.3475349 13.0216390
```

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)    42.8076     1.1065  38.688  < 2e-16 ***
## gender1        -3.3114     0.7785  -4.254 2.18e-05 ***
## hypertension1   3.7325     0.7797   4.787 1.80e-06 ***
## diabetes1      -2.1123     1.0899  -1.938  0.05274 .
## vaccine1       -6.1878     0.7919  -7.814 8.22e-15 ***
## severity1       8.5067     1.2453   6.831 1.07e-11 ***
## studyB          4.8893     0.8274   5.910 3.92e-09 ***
## smoking         1.7482     0.5733   3.049  0.00232 **
## race           -0.1552     0.3692  -0.420  0.67427
## ---
## 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) 8.954e-07      9   0.000   0.733
## s(SBP) 3.476e-07      9   0.000   0.436
## s(LDL) 1.210e-01      9   0.015   0.287
## s(bmi) 8.924e+00      9 104.228  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.321   Deviance explained = 32.6%
## GCV = 363.47  Scale est. = 360.74    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
## 21.5742112  0.3553758 13.3806433
```

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
## 18.4158221  0.4098351 12.1458134
```

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 11.61557 13.07002 13.39915 13.38478 13.90405 14.52028    0
## gam   11.40892 12.50309 12.79853 12.67674 12.91602 13.59344    0
## enet  11.53360 13.00630 13.31441 13.30771 13.83171 14.44532    0
## pls   11.77544 13.10149 13.49789 13.46198 13.87717 14.71427    0
## mars  11.20474 11.67112 11.89548 11.83433 12.04223 12.25247    0
## rf    11.13365 11.74141 12.31137 12.14820 12.46796 12.87401    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 16.81778 20.27198 21.53662 21.72343 22.69250 27.76260    0
## gam   16.04036 18.14591 20.19595 19.42976 20.46708 22.23657    0
## enet  16.78687 20.24242 21.51955 21.71395 22.73373 27.76476    0
## pls   16.80247 20.23810 21.29022 21.57906 22.52061 27.47453    0
## mars  15.75106 16.91733 17.36787 17.33552 17.95541 18.96887    0
```

```
## rf    15.80939 18.34123 18.98318 18.91953 19.73957 22.38047    0
##
## Rsquared
##             Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.04406941 0.07134251 0.1143569 0.1023988 0.1262758 0.1725520    0
## gam   0.17020989 0.22165750 0.2856678 0.3045001 0.4006389 0.4813416    0
## enet  0.04422923 0.07102989 0.1137048 0.1026903 0.1267357 0.1741316    0
## pls   0.04632791 0.07682653 0.1271983 0.1149609 0.1418805 0.1833651    0
## mars  0.15122880 0.29360907 0.3963813 0.4036020 0.5143793 0.6386991    0
## rf    0.16504454 0.25627740 0.2895327 0.3314130 0.4385469 0.5013773    0
```

Using bw-plot to compare their RMSE

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