ds2\_midterm

xinran

4/2/2020

library(tidyverse)

## -- Attaching packages ------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ---------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(patchwork)  
library(splines)  
library(gam)

## Warning: package 'gam' was built under R version 3.6.3

## Loading required package: foreach

##   
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':  
##   
## accumulate, when

## Loaded gam 1.16.1

library(mgcv)

## Loading required package: nlme

##   
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':  
##   
## collapse

## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.

##   
## Attaching package: 'mgcv'

## The following objects are masked from 'package:gam':  
##   
## gam, gam.control, gam.fit, s

library(boot)

##   
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':  
##   
## melanoma

library(ggplot2)  
library(pdp)

## Warning: package 'pdp' was built under R version 3.6.3

##   
## Attaching package: 'pdp'

## The following object is masked from 'package:purrr':  
##   
## partial

library(summarytools)

## Warning: package 'summarytools' was built under R version 3.6.3

## Registered S3 method overwritten by 'pryr':  
## method from  
## print.bytes Rcpp

## For best results, restart R session and update pander using devtools:: or remotes::install\_github('rapporter/pander')

##   
## Attaching package: 'summarytools'

## The following object is masked from 'package:tibble':  
##   
## view

library(earth)

## Warning: package 'earth' was built under R version 3.6.3

## Loading required package: Formula

## Loading required package: plotmo

## Warning: package 'plotmo' was built under R version 3.6.2

## Loading required package: plotrix

## Loading required package: TeachingDemos

## Warning: package 'TeachingDemos' was built under R version 3.6.2

library(ModelMetrics)

## Warning: package 'ModelMetrics' was built under R version 3.6.2

##   
## Attaching package: 'ModelMetrics'

## The following objects are masked from 'package:caret':  
##   
## confusionMatrix, precision, recall, sensitivity, specificity

## The following object is masked from 'package:base':  
##   
## kappa

library(pls)

## Warning: package 'pls' was built under R version 3.6.2

##   
## Attaching package: 'pls'

## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

library(readxl)

Introduction

This dataset is from the Boston Standard Metropolitan Statistical Area (SMSA). Each observation in the Boston Housing dataset represents a town in Boston in the 1970s. There are 506 observations and 14 total variables initially. After excluding observations(n=74) with missing values, we have 452 observations. The dataset was split into a training dataset and a testing dataset.

medv is the outcome variable, meaning median value of owner-occupied homes in $1000’s. We want to predict the housing rent in Boston using the rest of the 13 predicting variables(proximity to the employment center, per capita crime rate by town, average number of rooms per dwelling and so on).

### import dataset and remove rows with missing values

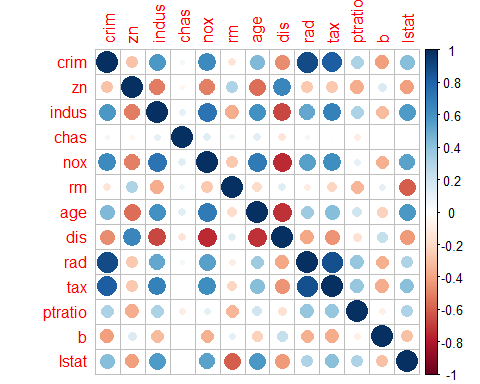
housing=read\_xlsx("./data/housing.xlsx")%>%  
 janitor::clean\_names()%>%  
 na.omit()

Exploratory analysis/visualization

In the correlation plot of the predicting variables, we found that some of them are highly correlated. Among positive correlations, crim(per capita crime rate by town) and rad(index of accessibility to radial highways), crim and tax(full-value property-tax rate per $10,000) are the 2 with the largest correlation coefficients. Among negative correlations, nox(nitric oxides concentration (parts per 10 million)) and dis(weighted distances to five Boston employment centres), age(proportion of owner-occupied units built prior to 1940) and dis are the 2 with the largest absolute values of correlation coefficients.

### visualization

x = model.matrix(medv ~., housing)[, -1]  
corrplot::corrplot(cor(x))



### split the data into a training set and a test set.

data(housing)

## Warning in data(housing): data set 'housing' not found

housing <- na.omit(housing)  
  
set.seed(1)  
trRows <- createDataPartition(housing$medv,  
 p = .75,  
 list = F)  
  
# training data  
x <- model.matrix(medv~.,housing)[trRows,-1]  
y <- housing$medv[trRows]  
train\_data = housing[trRows,]  
ctrl1 <- trainControl(method = "repeatedcv", number = 10, repeats = 5)  
  
# test data  
x2 <- model.matrix(medv~.,housing)[-trRows,-1]  
y2 <- housing$medv[-trRows]  
test\_data = housing[-trRows,]

### set a seed

set.seed(1)

### Fit a linear model

# train with 10-fold CV  
lm\_fit\_tr <- train(x, y, method = "lm",trControl = ctrl1,metric = 'RMSE')  
summary(lm\_fit\_tr)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -18.7120 -2.3366 -0.7135 1.7447 27.3597   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 23.006810 6.183571 3.721 0.000234 \*\*\*  
## crim -0.163388 0.254938 -0.641 0.522041   
## zn 0.040413 0.015942 2.535 0.011709 \*   
## indus 0.021499 0.068203 0.315 0.752789   
## chas 2.905853 0.933486 3.113 0.002016 \*\*   
## nox -11.226260 4.394365 -2.555 0.011081 \*   
## rm 4.705918 0.515137 9.135 < 2e-16 \*\*\*  
## age -0.004840 0.015335 -0.316 0.752491   
## dis -1.393414 0.236904 -5.882 1.00e-08 \*\*\*  
## rad 0.232182 0.100306 2.315 0.021246 \*   
## tax -0.009805 0.004405 -2.226 0.026716 \*   
## ptratio -0.806821 0.148535 -5.432 1.09e-07 \*\*\*  
## b 0.013668 0.004162 3.284 0.001133 \*\*   
## lstat -0.551001 0.063818 -8.634 2.64e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.575 on 327 degrees of freedom  
## Multiple R-squared: 0.7438, Adjusted R-squared: 0.7336   
## F-statistic: 73.03 on 13 and 327 DF, p-value: < 2.2e-16

# calculate RMSE using the test data  
pred\_lm\_tr = predict(lm\_fit\_tr, test\_data)  
lm\_rmse\_test = sqrt(mean((pred\_lm\_tr - test\_data$medv)^2));  
lm\_rmse\_test

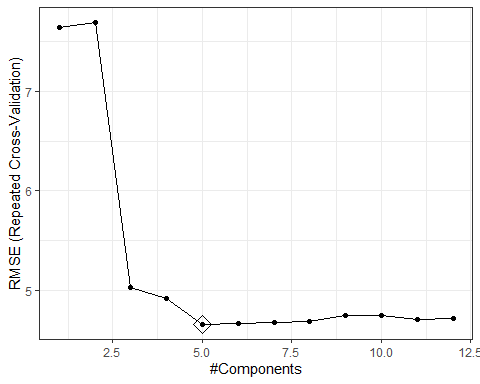
## [1] 4.544709

### fit a PCR model

# fit the model by centering the data  
pcr.fit <- train(x, y,  
 method = "pcr",  
 tuneLength = length(train\_data) - 1,  
 trControl = ctrl1,  
 scale = TRUE)  
  
# select tunning parameter  
pcr.fit$bestTune

## ncomp  
## 5 5

#predicted values based on test dataset  
predy2.pcr <- predict(pcr.fit$finalModel, newdata = x2,   
 ncomp = pcr.fit$bestTune$ncomp)  
  
# test MSE  
pcr\_rmse\_test=rmse(y2, predy2.pcr)  
  
#Validation plot showing ncomp=5 have the lowest cross-validation error  
ggplot(pcr.fit, highlight = TRUE) + theme\_bw()

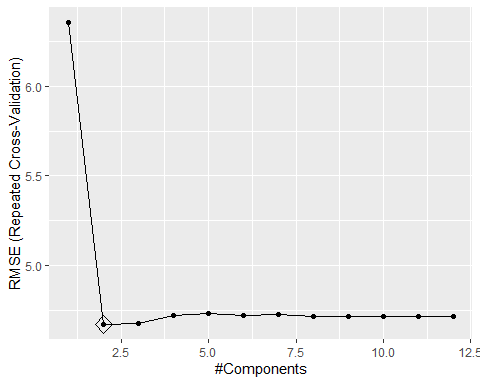
 The tuning parameter selected by cross validation is 5.

### fit a PLS model

pls.fit <- train(x, y,  
 method = "pls",  
 tuneLength = length(train\_data) - 1,  
 trControl = ctrl1,  
 preProc = c("center", "scale"))  
  
# select tuning parameter  
pls.fit$bestTune

## ncomp  
## 2 2

#predicted values based on test dataset  
predy2.pls<- predict(pls.fit, newdata = x2,ncomp = pls.fit$bestTune$ncomp)  
  
#test RMSE  
pls\_rmse\_test=rmse(y2, predy2.pls)  
  
#Validation plot showing ncomp=5 have the lowest cross-validation error  
ggplot(pls.fit, highlight = TRUE)



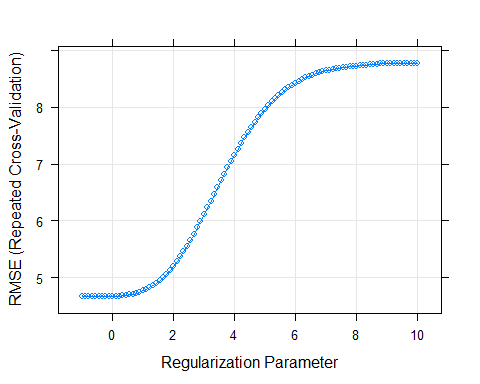
The tuning parameter selected by cross validation is 2.

### fit a ridge model

ridge.fit <- train(x, y,  
 method = "glmnet",  
 tuneGrid = expand.grid(alpha = 0,   
 lambda = exp(seq(-1, 10, length=100))),  
 trControl = ctrl1)

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

#predicted values based on test dataset  
predy2.ridge <- predict(ridge.fit, newdata = x2)  
  
#test MSE  
ridge\_rmse\_test=rmse(y2, predy2.ridge)  
  
# plot the RMSE on tuning parameters  
plot(ridge.fit, xTrans = function(x) log(x))



# select the tuning parameter  
ridge.fit$bestTune

## alpha lambda  
## 5 0 0.5737534

#coefficient matrix  
coef(ridge.fit$finalModel,ridge.fit$bestTune$lambda)

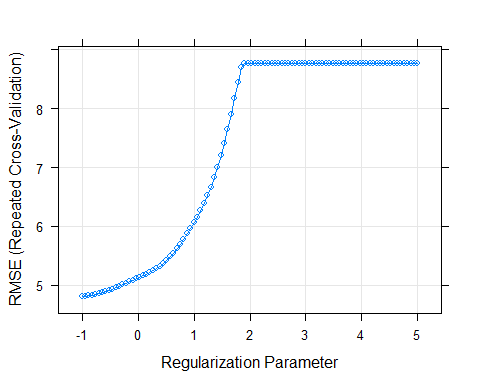
## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 18.348062409  
## crim -0.062263534  
## zn 0.027639254  
## indus -0.020702335  
## chas 2.982917327  
## nox -7.711182458  
## rm 4.743606323  
## age -0.006666669  
## dis -1.098566931  
## rad 0.114571145  
## tax -0.005434965  
## ptratio -0.753346715  
## b 0.013248394  
## lstat -0.518475873

### fit a lasso model

lasso.fit <- train(x, y,  
 method = "glmnet",  
 tuneGrid = expand.grid(alpha = 1,   
 lambda = exp(seq(-1, 5, length=100))),  
 # preProc = c("center", "scale"),  
 trControl = ctrl1)

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

#predicted values based on test dataset  
predy2.lasso <- predict(lasso.fit, newdata = x2)  
  
# test MSE  
lasso\_rmse\_test=rmse(y2, predy2.lasso)  
  
# plot the RMSE on tuning parameters  
plot(lasso.fit, xTrans = function(x) log(x))



# select the tuning parameter  
lasso.fit$bestTune

## alpha lambda  
## 1 1 0.3678794

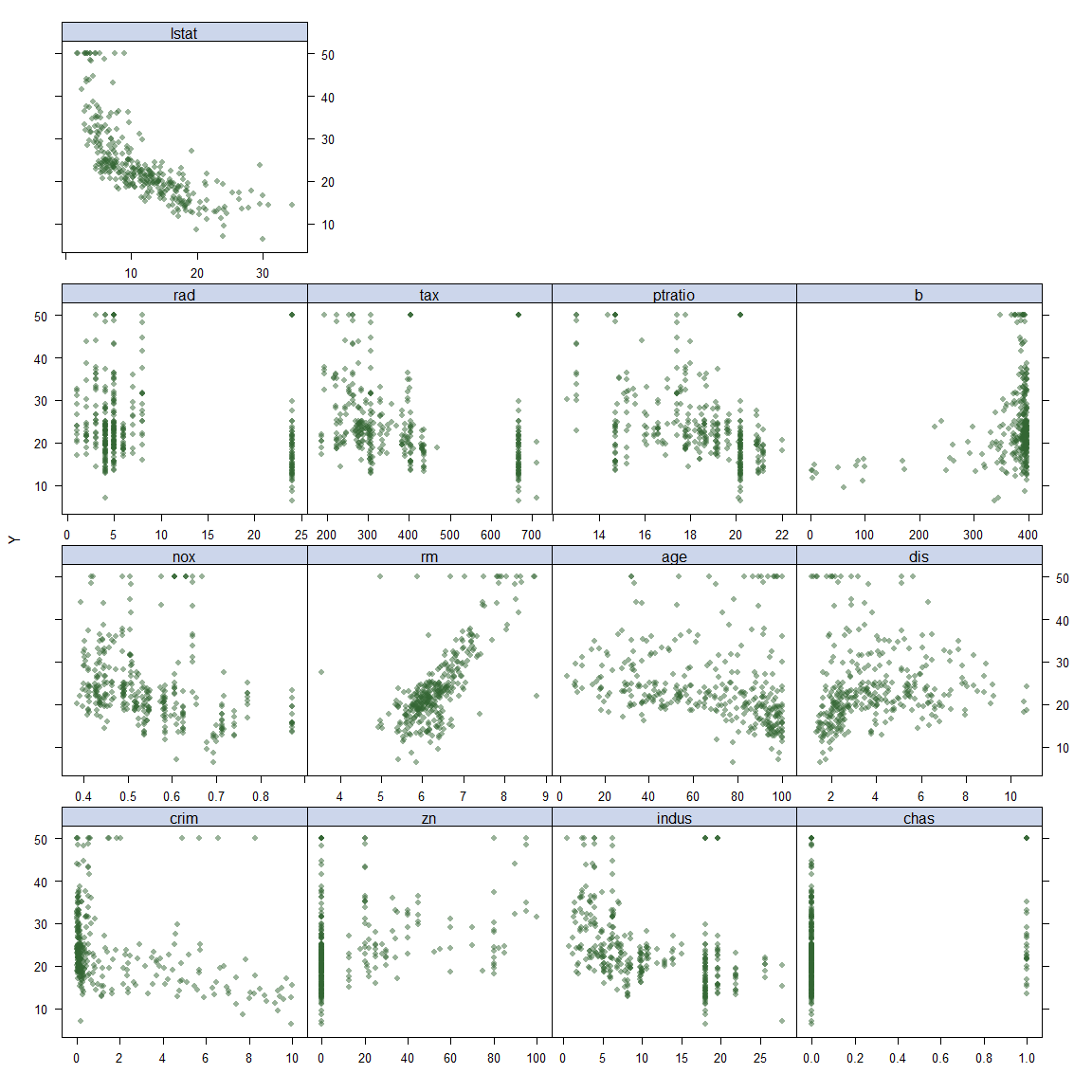
#coefficient matrix  
coef(lasso.fit$finalModel,lasso.fit$bestTune$lambda)

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 7.9885650302  
## crim .   
## zn .   
## indus .   
## chas 1.9698540565  
## nox .   
## rm 5.0742399121  
## age .   
## dis -0.3572482843  
## rad .   
## tax -0.0006839664  
## ptratio -0.6675146605  
## b 0.0099924446  
## lstat -0.5564773710

## non-linear models

### visualization

# set theme  
theme1 <- trellis.par.get()  
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)   
theme1$plot.symbol$pch <- 16  
theme1$plot.line$col <- rgb(.8, .1, .1, 1)   
theme1$plot.line$lwd <- 2  
theme1$strip.background$col <- rgb(.0, .2, .6, .2)  
trellis.par.set(theme1)  
  
# use scatterplot to visualize the relationship between medv and the rest of the variables  
featurePlot(x, y, plot = "scatter", labels = c("","Y"), type = c("p"), layout = c(4, 4))



We plot each of the 13 predicting variables using scatterplot. Variable lstat has a potentially nonlinear trend.

### Generalized Additive Model (GAM)

gam.fit <- train(x, y,  
 method = "gam",  
 tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE,FALSE)),  
 trControl = ctrl1)

## Warning: model fit failed for Fold03.Rep2: method=GCV.Cp, select= TRUE Error in magic(G$y, G$X, msp, G$S, G$off, L = G$L, lsp0 = G$lsp0, G$rank, :   
## magic, the gcv/ubre optimizer, failed to converge after 400 iterations.

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

gam.fit$bestTune

## select method  
## 1 FALSE GCV.Cp

gam.fit$finalModel

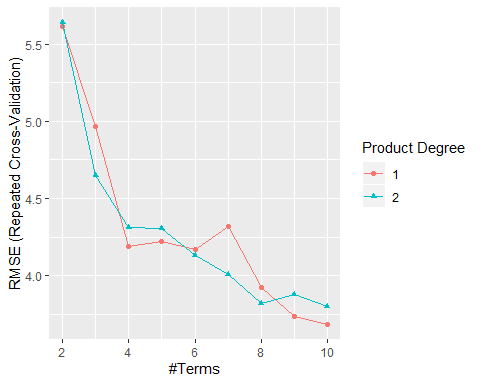
##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## .outcome ~ chas + rad + s(zn) + s(ptratio) + s(tax) + s(indus) +   
## s(nox) + s(b) + s(age) + s(dis) + s(rm) + s(lstat) + s(crim)  
##   
## Estimated degrees of freedom:  
## 1.00 1.79 6.57 4.94 8.46 1.00 1.62   
## 8.92 6.23 7.57 1.00 total = 52.08   
##   
## GCV score: 10.25373

summary(gam.fit)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## .outcome ~ chas + rad + s(zn) + s(ptratio) + s(tax) + s(indus) +   
## s(nox) + s(b) + s(age) + s(dis) + s(rm) + s(lstat) + s(crim)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 22.4072 1.2223 18.331 <2e-16 \*\*\*  
## chas 0.9677 0.6824 1.418 0.157   
## rad 0.1579 0.1487 1.062 0.289   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(zn) 1.000 1.000 0.439 0.5082   
## s(ptratio) 1.789 2.234 14.158 6.39e-07 \*\*\*  
## s(tax) 6.571 7.568 6.376 4.27e-07 \*\*\*  
## s(indus) 4.937 5.886 2.745 0.0087 \*\*   
## s(nox) 8.455 8.867 12.410 < 2e-16 \*\*\*  
## s(b) 1.000 1.000 1.949 0.1637   
## s(age) 1.618 2.024 1.520 0.2307   
## s(dis) 8.917 8.995 8.489 1.72e-11 \*\*\*  
## s(rm) 6.226 7.401 28.373 < 2e-16 \*\*\*  
## s(lstat) 7.569 8.491 17.255 < 2e-16 \*\*\*  
## s(crim) 1.000 1.000 0.964 0.3270   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.889 Deviance explained = 90.6%  
## GCV = 10.254 Scale est. = 8.6877 n = 341

### Multivariate Adaptive Regression Splines (MARS)

mars\_grid <- expand.grid(degree = 1:2,   
 nprune = 2:10)  
  
mars.fit <- train(x, y,  
 method = "earth",  
 tuneGrid = mars\_grid,  
 trControl = ctrl1)  
  
ggplot(mars.fit)



mars.fit$bestTune

## nprune degree  
## 9 10 1

coef(mars.fit$finalModel)

## (Intercept) h(lstat-5.98) h(5.98-lstat) h(6.406-rm)   
## -31.7573869 -0.5184470 2.3382330 6.1704560   
## h(ptratio-14.7) h(dis-2.4982) h(2.4982-dis) h(dis-1.5916)   
## -0.6526660 -58.7092436 53.4324769 58.2613221   
## h(233-tax) h(rm-5.565)   
## 0.1211654 7.9949138

resamp=resamples(list(lm = lm\_fit\_tr,  
 ridge = ridge.fit,  
 lasso = lasso.fit,  
 pcr = pcr.fit,  
 gam=gam.fit,  
 mars = mars.fit))  
summary(resamp)

##   
## Call:  
## summary.resamples(object = resamp)  
##   
## Models: lm, ridge, lasso, pcr, gam, mars   
## Number of resamples: 50   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lm 2.447263 2.855957 3.242914 3.221541 3.506380 4.626068 0  
## ridge 2.253028 2.711008 3.049061 3.154675 3.564865 4.163023 0  
## lasso 1.923688 2.794503 3.132394 3.217061 3.799784 4.419945 0  
## pcr 1.880511 2.764123 2.988365 3.159902 3.517870 4.565237 0  
## gam 1.792398 2.207461 2.424941 2.546061 2.838919 3.773005 0  
## mars 1.975552 2.353750 2.534475 2.640772 2.983622 3.654135 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lm 3.214209 3.798174 4.489662 4.660772 5.191315 7.244682 0  
## ridge 2.980752 3.798796 4.420559 4.659751 5.397514 7.260478 0  
## lasso 2.569293 3.859025 4.307281 4.798862 5.663987 8.024007 0  
## pcr 2.323103 3.724235 4.089730 4.655230 5.402729 7.824329 0  
## gam 2.391489 2.922155 3.283364 3.650850 3.859911 6.716735 0  
## mars 2.401421 3.020158 3.491032 3.683709 4.186218 5.714297 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lm 0.3923012 0.6546880 0.7606691 0.7272179 0.8206781 0.8838696 0  
## ridge 0.2637023 0.6618358 0.7626795 0.7250595 0.8131584 0.9160635 0  
## lasso 0.1705321 0.6351326 0.7786312 0.7107266 0.8337174 0.9046594 0  
## pcr 0.3099444 0.6807363 0.7757521 0.7322454 0.8276220 0.9053964 0  
## gam 0.3802733 0.8202381 0.8646726 0.8285167 0.9083103 0.9277636 0  
## mars 0.5363283 0.7931733 0.8509195 0.8317611 0.8879964 0.9566800 0

bwplot(resamp, metric = "RMSE")

