Dimension Reduction Methods in Linear Regression

Yifei Sun, Runze Cui

Contents

```r
library(ISLR)
library(pls)
library(caret)
library(tidymodels)
```

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)
```

```r
# training data
x <- model.matrix(Salary ~ ., training_data)[, -1]
y <- training_data$Salary

# test data
x2 <- model.matrix(Salary ~ .,testing_data)[, -1]
y2 <- testing_data$Salary
```

SVD

```r
# center and scale
x3 <- scale(x)

# SVD
x_svd <- svd(x3)
u <- x_svd$u
v <- x_svd$v
d <- diag(x_svd$d)
# corrplot::corrplot(t(u) %*% u, is.corr = FALSE)
# corrplot::corrplot(t(v) %*% v, is.corr = FALSE)
# corrplot::corrplot(v %*% t(v), is.corr = FALSE)
# corrplot::corrplot(d, is.corr = FALSE)

# definition
x4 <- u %*% d %*% t(v)
all.equal(x3, x4, check.attributes = FALSE)
```

```
## [1] TRUE
```

```r
# PCA
x_pca <- prcomp(x, scale. = TRUE)

all.equal(x_pca$rotation, v, check.attributes = FALSE)
```

```
## [1] TRUE
```

Principal components regression (PCR)

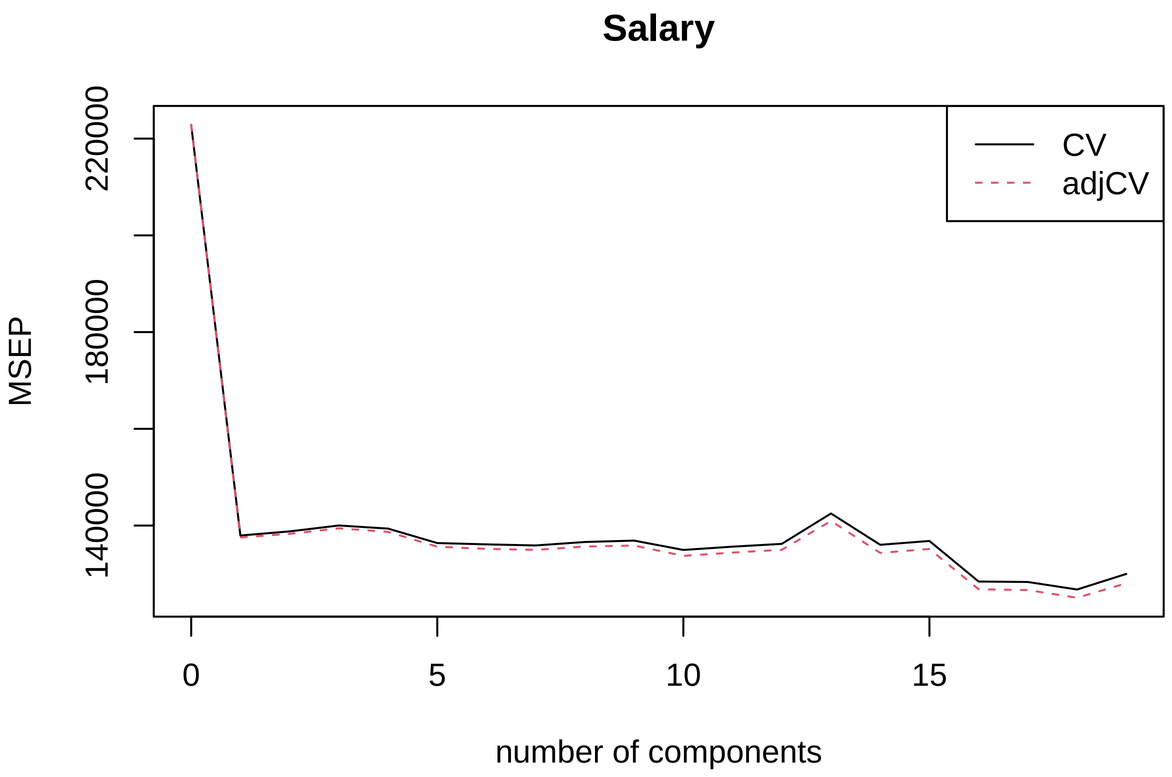
We fit the PCR model using the function `pcr()`.

```r
set.seed(2)
pcr.mod <- pcr(Salary ~ .,
               data = training_data,
               scale = TRUE, # scale = FALSE by default
               validation = "CV")

summary(pcr.mod)
```

```
## Data:    X dimension: 210 19
##  Y dimension: 210 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           472.1    371.4    372.6    374.2    373.3    369.3    368.9
## adjCV        472.1    370.9    371.9    373.4    372.4    368.3    367.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       368.6    369.6    370.0     367.4     368.3     369.1     377.5
## adjCV    367.4    368.3    368.6     365.7     366.6     367.4     375.4
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        368.8     369.9     358.4     358.2     356.1     360.6
## adjCV     366.5     367.6     356.1     355.9     353.7     357.9
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         37.90    59.83    71.09    79.07    84.27    88.44    92.14    94.77
## Salary    40.56    41.42    41.56    42.62    44.55    45.57    46.06    46.06
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.12     97.10     97.89     98.60     99.12     99.47     99.74
## Salary    46.40     48.56     48.60     48.61     48.66     51.26     51.33
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     54.55     55.02     55.93     56.02
```

```r
validationplot(pcr.mod, val.type = "MSEP", legendpos = "topright")
```



```
cv.mse <- RMSEP(pcr.mod)
ncomp.cv <- which.min(cv.mse$val[1,,]) - 1
ncomp.cv
```

```
## 18 comps
##       18
```

```
predy2.pcr <- predict(pcr.mod, newdata = testing_data,
                      ncomp = ncomp.cv)
# test MSE
mean((y2 - predy2.pcr)^2)
```

```
## [1] 90894.84
```

Partial least squares (PLS)

We fit the PLS model using the function `plsr()`.

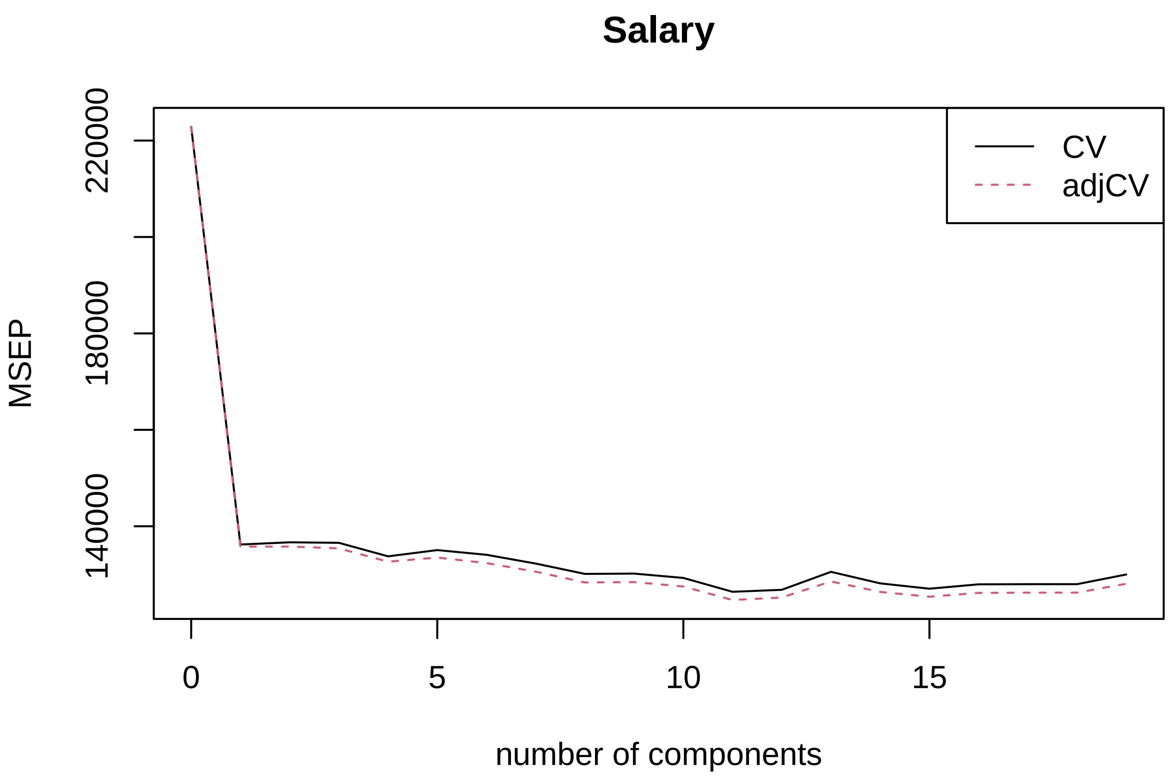
```
set.seed(2)
pls.mod <- plsr(Salary~.,
                data = training_data,
                scale = TRUE,
                validation = "CV")

summary(pls.mod)
```

```
## Data:    X dimension: 210 19
##  Y dimension: 210 1
## Fit method: kernelpls
## Number of components considered: 19
##
```

```
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           472.1    369.1    369.7    369.6    365.7    367.5    366.2
## adjCV        472.1    368.5    368.5    368.0    364.2    365.4    363.8
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       363.7    360.7    360.8     359.6     355.5     356.1     361.3
## adjCV    361.4    358.3    358.4     357.1     353.1     353.9     358.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        358.0     356.4     357.7     357.8     357.8     360.6
## adjCV     355.5     354.1     355.2     355.3     355.3     357.9
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         37.70    50.45    65.18    71.93    77.45    84.28    88.12    89.70
## Salary    42.77    46.29    47.94    49.63    51.55    52.45    53.26    54.57
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         92.76     94.43     95.31     97.55     98.02     98.40     98.92
## Salary    54.80     55.11     55.44     55.50     55.68     55.88     55.92
##         16 comps  17 comps  18 comps  19 comps
## X          99.13     99.64     99.99    100.00
## Salary     55.97     55.98     55.99     56.02
```

```
validationplot(pls.mod, val.type = "MSEP", legendpos = "topright")
```



```
cv.mse <- RMSEP(pls.mod)
ncomp.cv <- which.min(cv.mse$val[1,,]) - 1
ncomp.cv
```

```
## 11 comps
##       11
```

```r
predy2.pls <- predict(pls.mod, newdata = testing_data,
                      ncomp = ncomp.cv)
# test MSE
mean((y2 - predy2.pls)^2)
```

```
## [1] 92933.01
```

PCR and PLS using caret

PCR

```r
ctrl1 <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 5,
                      selectionFunction = "best") # "oneSE" for the 1SE rule

# show information about the model
modelLookup("pcr")
```

```
##   model parameter       label forReg forClass probModel
## 1   pcr     ncomp #Components   TRUE    FALSE     FALSE
```

```r
modelLookup("pls")
```

```
##   model parameter       label forReg forClass probModel
## 1   pls     ncomp #Components   TRUE     TRUE      TRUE
```

```r
# Two ways for standardizing predictors

# train(..., preProc = c("center", "scale"))
set.seed(2)
pcr.fit <- train(x, y,
                 method = "pcr",
                 tuneGrid  = data.frame(ncomp = 1:19),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))

predy2.pcr2 <- predict(pcr.fit, newdata = x2)
mean((y2 - predy2.pcr2)^2)
```

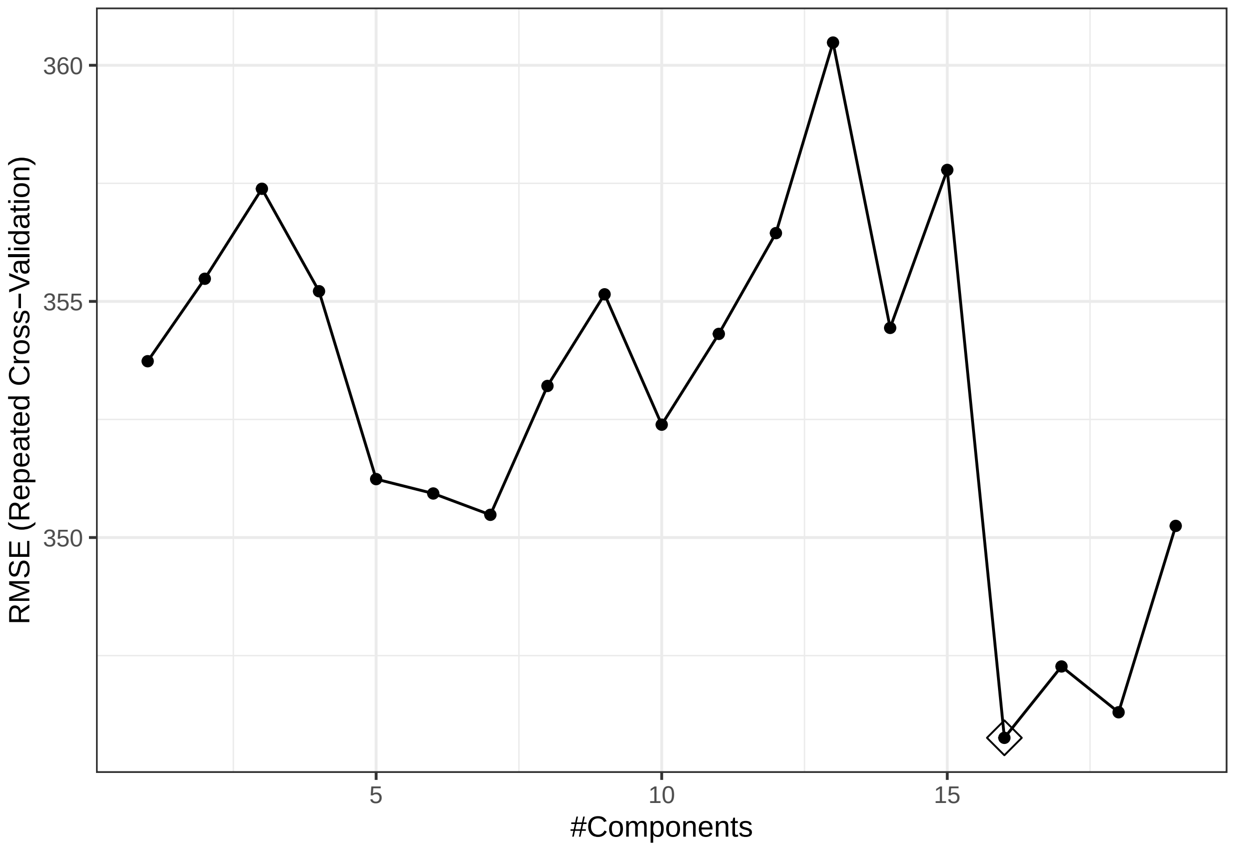
```
## [1] 93174.43
```

```r
# pcr(..., scale = TRUE)
set.seed(2)
pcr.fit2 <- train(x, y,
                  method = "pcr",
                  tuneGrid = data.frame(ncomp = 1:19),
                  trControl = ctrl1,
                  scale = TRUE)

predy2.pcr3 <- predict(pcr.fit2, newdata = x2)
mean((y2 - predy2.pcr3)^2)
```

```
## [1] 93174.43
```

```
ggplot(pcr.fit, highlight = TRUE) + theme_bw()
```

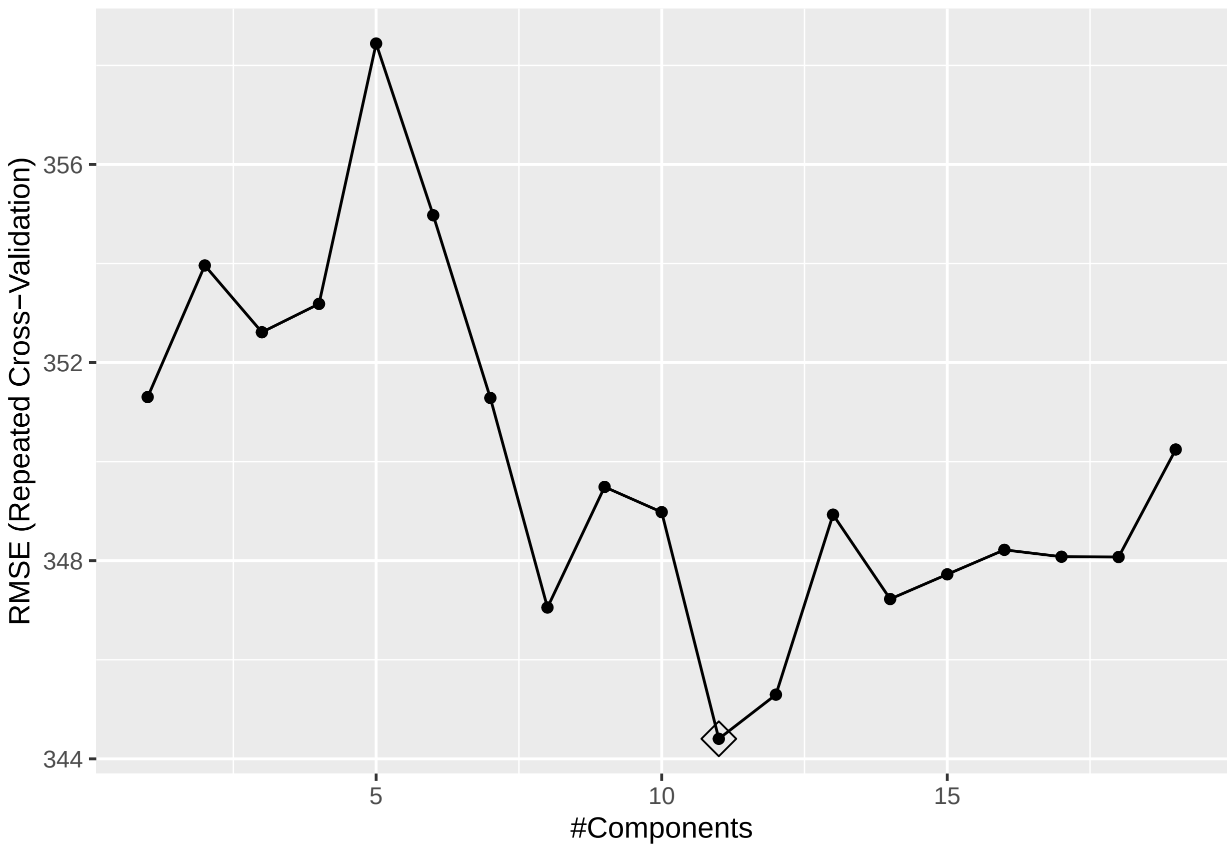


PLS

```
set.seed(2)
pls.fit <- train(x, y,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:19),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))
predy2.pls2 <- predict(pls.fit, newdata = x2)
mean((y2 - predy2.pls2)^2)
```

```
## [1] 92933.01
```

```
ggplot(pls.fit, highlight = TRUE)
```



Here are some old code on elastic net.

```r
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.
```

```r
# myCol <- rainbow(25)
# myPar <- list(superpose.symbol = list(col = myCol),
#               superpose.line = list(col = myCol))
# plot(enet.fit, xTrans = log, par.settings = myPar)
```

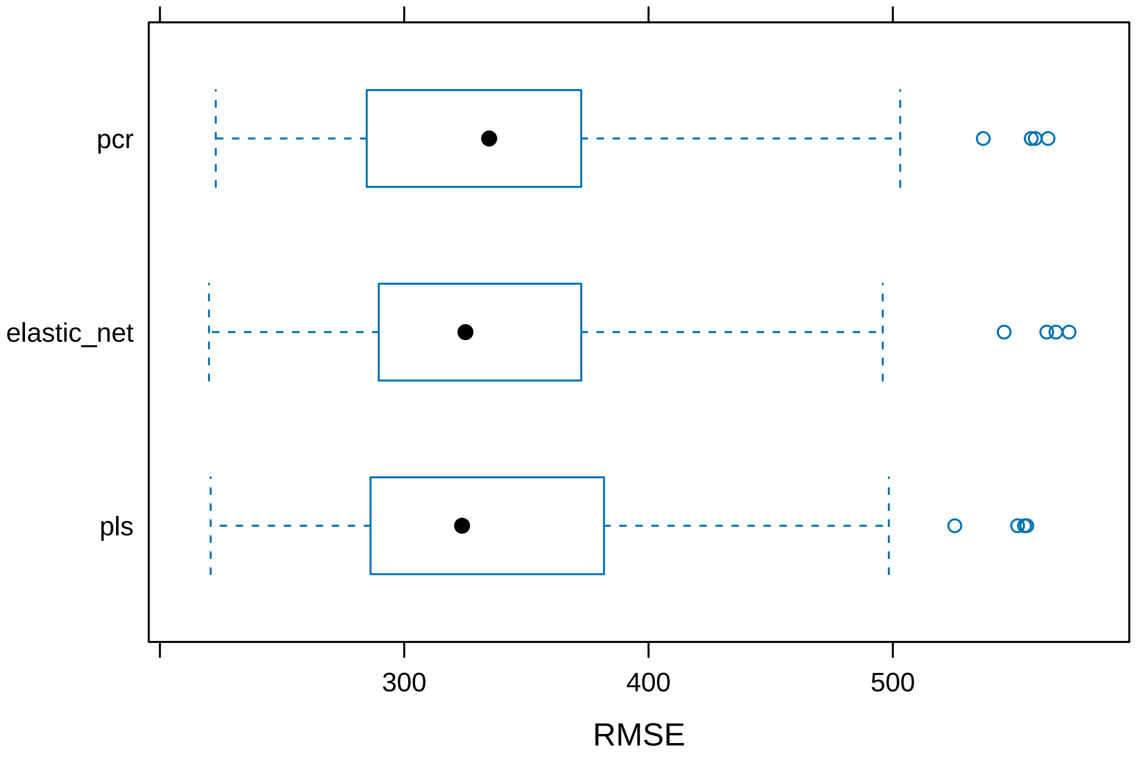
Comparing the models based on resampling results.

```r
resamp <- resamples(list(elastic_net = enet.fit,
                         pcr = pcr.fit,
                         pls = pls.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
```

```
## Models: elastic_net, pcr, pls
## Number of resamples: 50
##
## MAE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## elastic_net 163.3124 215.5571 240.4248 248.2230 275.3263 373.5337    0
## pcr         171.4586 222.0582 243.1119 253.4684 275.8308 379.9711    0
## pls         164.1584 217.9899 241.4971 251.3777 275.9601 379.1324    0
##
## RMSE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## elastic_net 220.0769 289.7156 325.0586 344.2359 371.1565 572.1447    0
## pcr         222.8634 286.5283 334.7599 345.7586 372.2937 563.5358    0
## pls         220.7667 286.4764 323.6777 344.4050 380.5525 554.8279    0
##
## Rsquared
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## elastic_net 0.01505160 0.3869746 0.4944062 0.4811950 0.6202539 0.8622079    0
## pcr         0.02684558 0.3883547 0.4925494 0.4750800 0.6026677 0.8347328    0
## pls         0.02822971 0.3507958 0.5037413 0.4775314 0.6066276 0.8513646    0
```

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


pcr
elastic_net
pls
RMSE