Dimension Reduction Methods in Linear Regression

Yifei Sun

Contents

```
library(ISLR)
library(pls)
library(caret)
```

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(2021)
trRows <- createDataPartition(Hitters$Salary,
                              p = .75,
                              list = F)

# training data
x <- model.matrix(Salary~.,Hitters)[trRows,-1]
y <- Hitters$Salary[trRows]

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

Principal components regression (PCR)

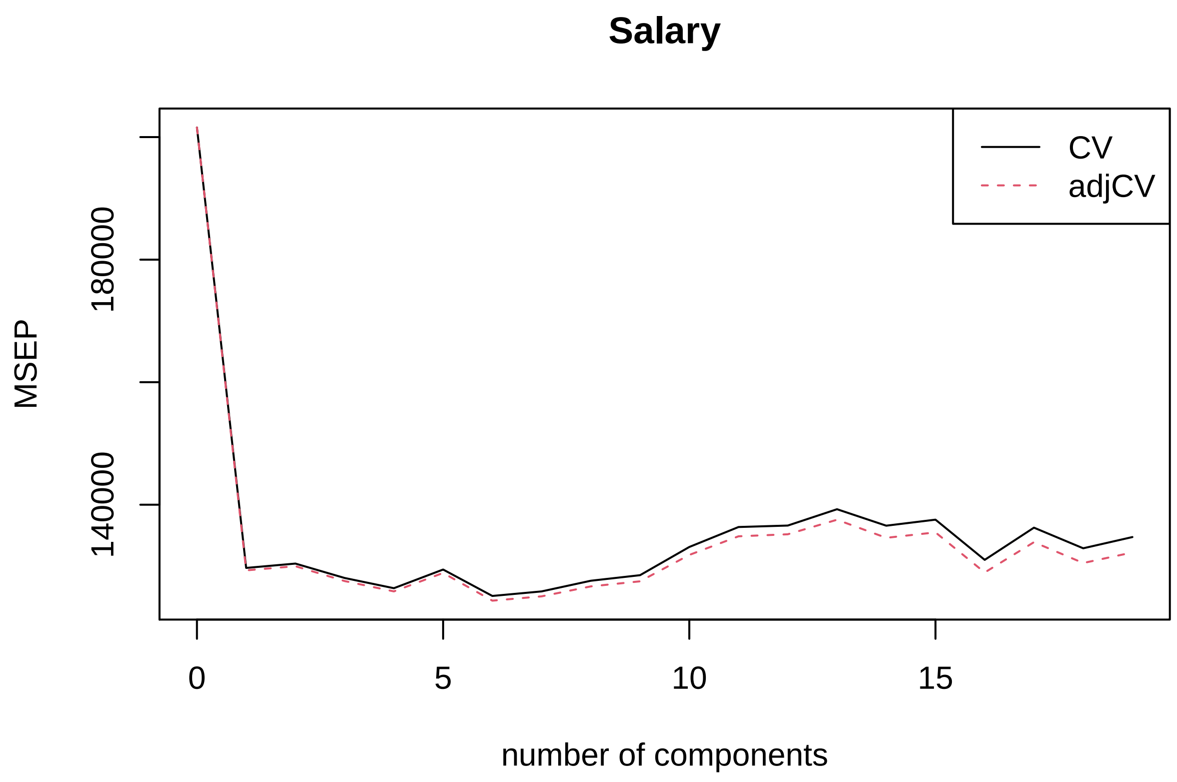
We fit the PCR model using the function `pcr()`.

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

summary(pcr.mod)
```

```
## Data:    X dimension: 200 19
##  Y dimension: 200 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             449    360.1    361.1    357.8    355.5    359.8    353.7
## adjCV          449    359.6    360.5    357.1    354.8    359.0    352.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       354.8    357.2    358.5     364.8     369.3     369.6     373.2
## adjCV    353.6    355.9    357.1     363.0     367.2     367.7     370.9
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        369.6     370.9     361.9     369.1     364.5     367.0
## adjCV     366.9     368.1     359.1     365.9     361.2     363.6
```

```
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         40.20    60.14    71.10    78.94    84.21    88.82    92.35    95.07
## Salary    38.68    38.77    40.67    41.56    41.72    44.49    44.53    44.55
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.38     97.34     98.05     98.65     99.16     99.49     99.76
## Salary    44.86     45.21     45.34     45.37     46.04     48.71     49.01
##         16 comps  17 comps  18 comps  19 comps
## X          99.91     99.97     99.99    100.00
## Salary     51.87     51.87     53.35     53.46
```

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



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

```
## 6 comps
##       6
```

```r
predy2.pcr <- predict(pcr.mod, newdata = Hitters[-trRows,],
                      ncomp = ncomp.cv)
# test MSE
mean((y2 - predy2.pcr)^2)
```

```
## [1] 103756.1
```

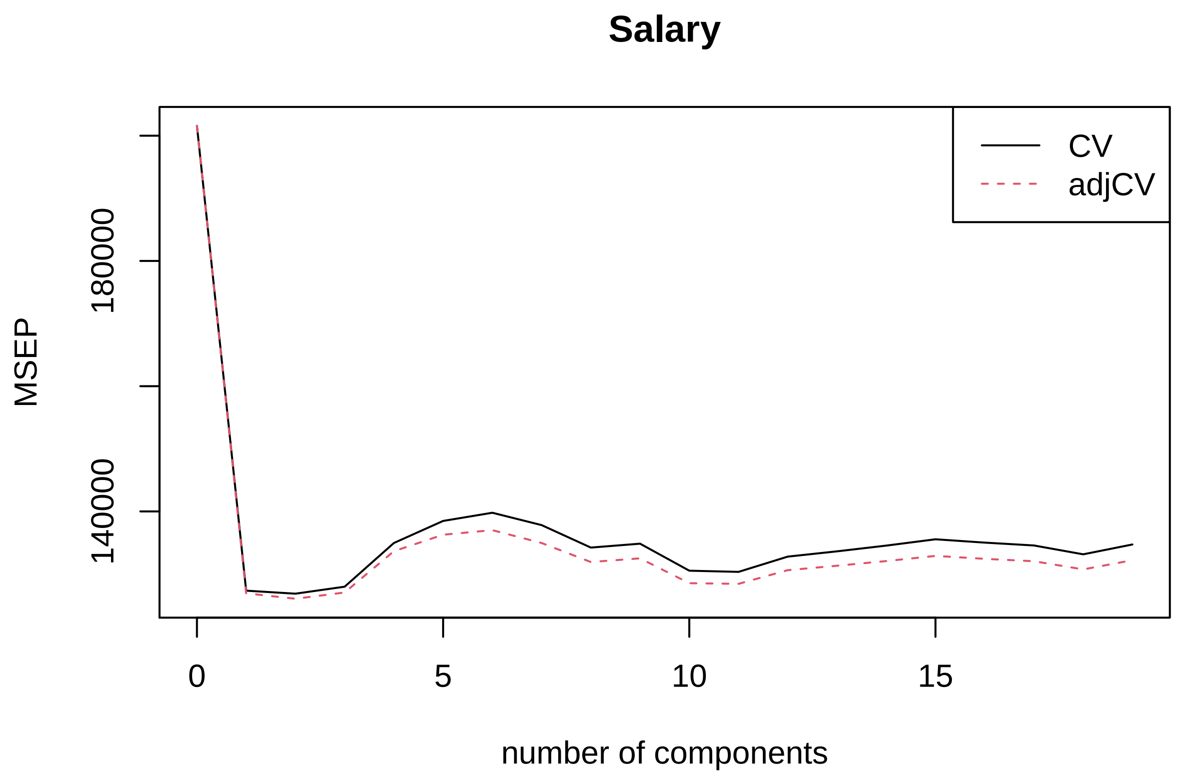
Partial least squares (PLS)

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

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

summary(pls.mod)
```

```
## Data:    X dimension: 200 19
##  Y dimension: 200 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             449    356.9    356.2    357.7    367.4    372.1    373.9
## adjCV          449    356.3    355.0    356.4    365.6    369.2    370.1
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       371.2    366.4    367.2     361.3     361.0     364.4     365.5
## adjCV    367.4    363.2    364.0     358.5     358.4     361.4     362.4
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        366.8     368.2     367.4     366.8     364.9     367.0
## adjCV     363.4     364.5     363.9     363.4     361.6     363.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         40.01    49.51    60.95    74.65    79.13    84.14    86.16    89.76
## Salary    40.75    44.71    45.91    46.68    49.00    50.39    51.39    51.76
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         92.77     94.43     96.71     97.78     98.31     98.64     99.06
## Salary    51.99     52.31     52.46     52.65     52.86     53.24     53.41
##         16 comps  17 comps  18 comps  19 comps
## X          99.43     99.93     99.95    100.00
## Salary     53.42     53.42     53.45     53.46
```

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



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

```
## 2 comps
##       2
```

```
predy2.pls <- predict(pls.mod, newdata = Hitters[-trRows,],
                      ncomp = ncomp.cv)
# test MSE
mean((y2 - predy2.pls)^2)
```

```
## [1] 104418.8
```

PCR and PLS using `caret`

PCR

```
ctrl1 <- trainControl(method = "cv",
                      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
```

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

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

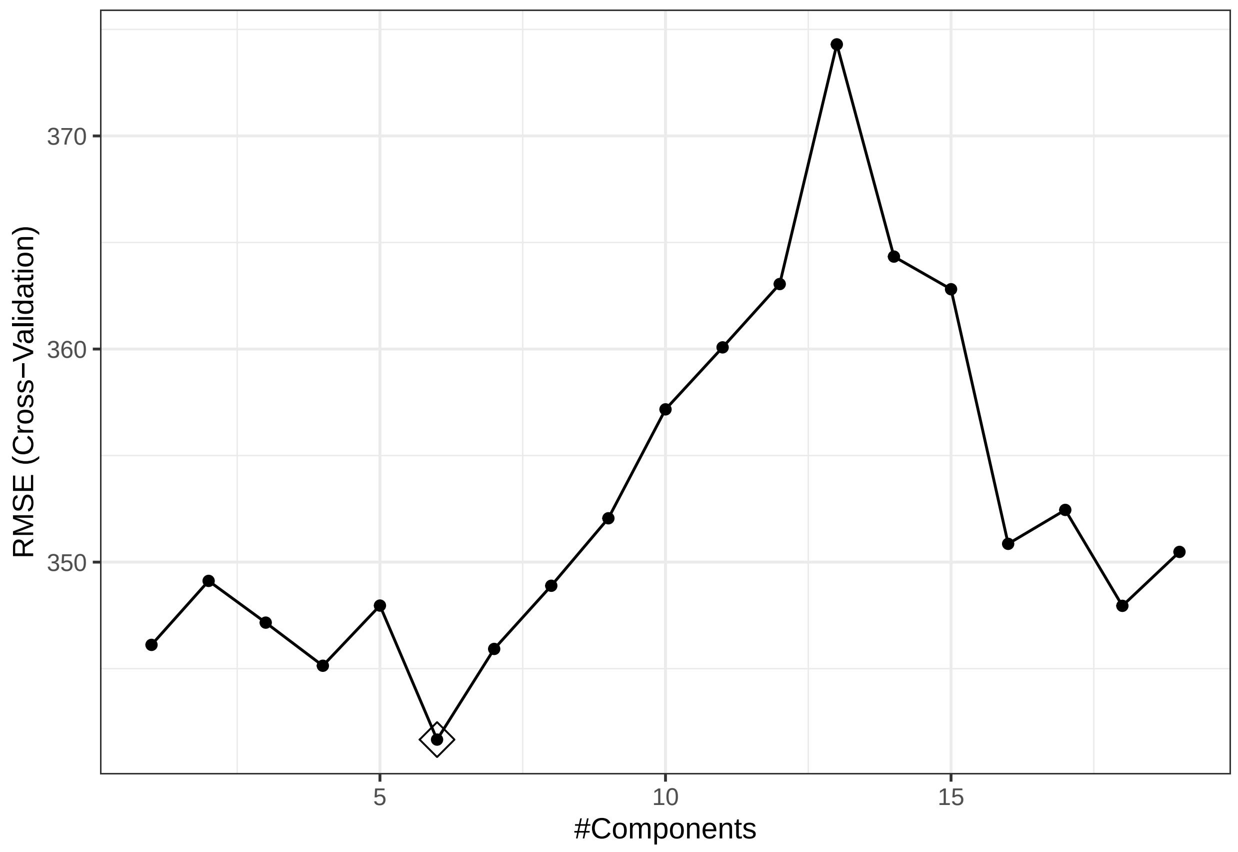
```
# Two ways for standardizing predictors

# 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] 103756.1
```

```
# 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.fit, newdata = x2)
mean((y2 - predy2.pcr3)^2)
```

```
## [1] 103756.1
```

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

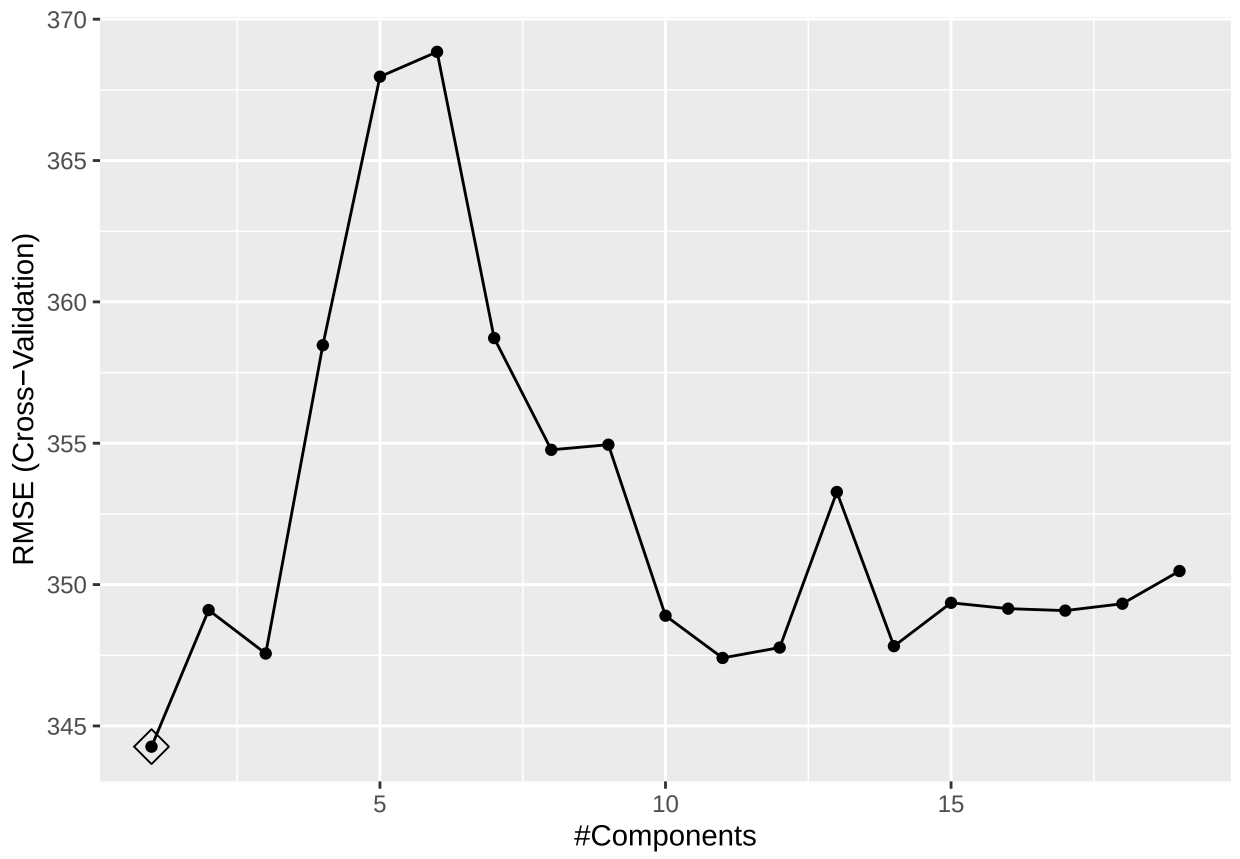


PLS

```
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] 107400.5
```

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



Here are some old codes on ridge, lasso and ordinary least squares.

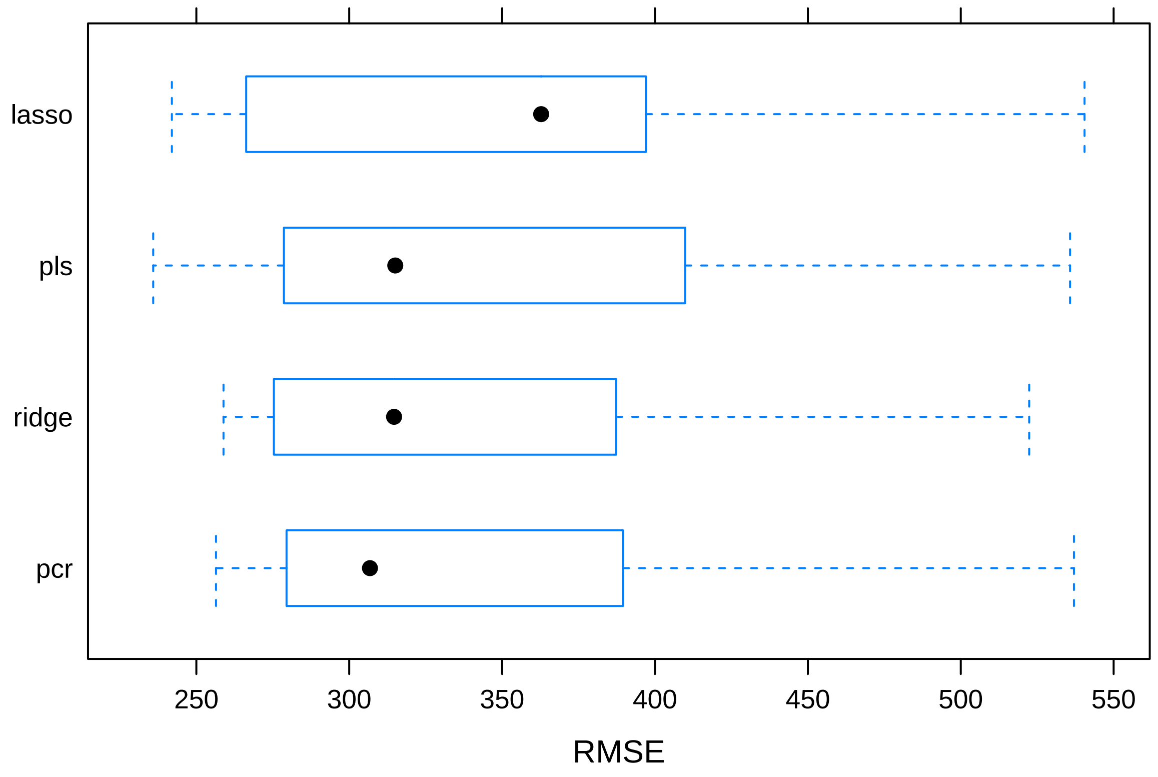
```
set.seed(2)
ridge.fit <- train(x, y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(-1, 10, length=100))),
                   trControl = ctrl1)
predy2.ridge <- predict(ridge.fit, newdata = x2)


set.seed(2)
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)
predy2.lasso <- predict(lasso.fit, newdata = x2)
```

Comparing the models based on resampling results.

```
resamp <- resamples(list(lasso = lasso.fit,
                         ridge = ridge.fit,
                         pcr = pcr.fit,
                          pls = pls.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, ridge, pcr, pls
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 167.5977 208.5325 254.8819 241.2160 274.4900 299.1552    0
## ridge 175.1284 207.5289 230.4954 235.8461 255.7910 300.1394    0
## pcr   172.5466 200.4329 220.4705 230.5195 262.9214 297.0131    0
## pls   177.6785 207.5252 219.5236 232.5475 261.8803 301.0774    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 241.9932 267.6539 362.7664 346.0944 389.3080 540.4914    0
## ridge 258.8813 278.3543 314.6673 344.7360 387.0793 522.3878    0
## pcr   256.4354 280.3404 306.7773 341.6713 388.7912 537.0166    0
## pls   235.8924 281.6246 315.0572 344.2670 400.4012 535.7458    0
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.04942920 0.2492379 0.4905530 0.4514729 0.6339149 0.7392623    0
## ridge 0.03183006 0.3390105 0.4625176 0.4518270 0.6157224 0.7482311    0
## pcr   0.03822160 0.3153999 0.4606597 0.4538285 0.6161780 0.7650883    0
## pls   0.02315483 0.4227511 0.5119100 0.4562174 0.5966268 0.6990407    0
```

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


lasso
pls
ridge
pcr
250
300
350
400
450
500
550
RMSE