Stat 427/627 Statistical Machine Learning

In-class Lab 8: Principal Components and Partial Least Squares

Contents

Recall the `Auto` data set in the `ISLR2` package. This data frame has 392 observations on the following 9 variables.

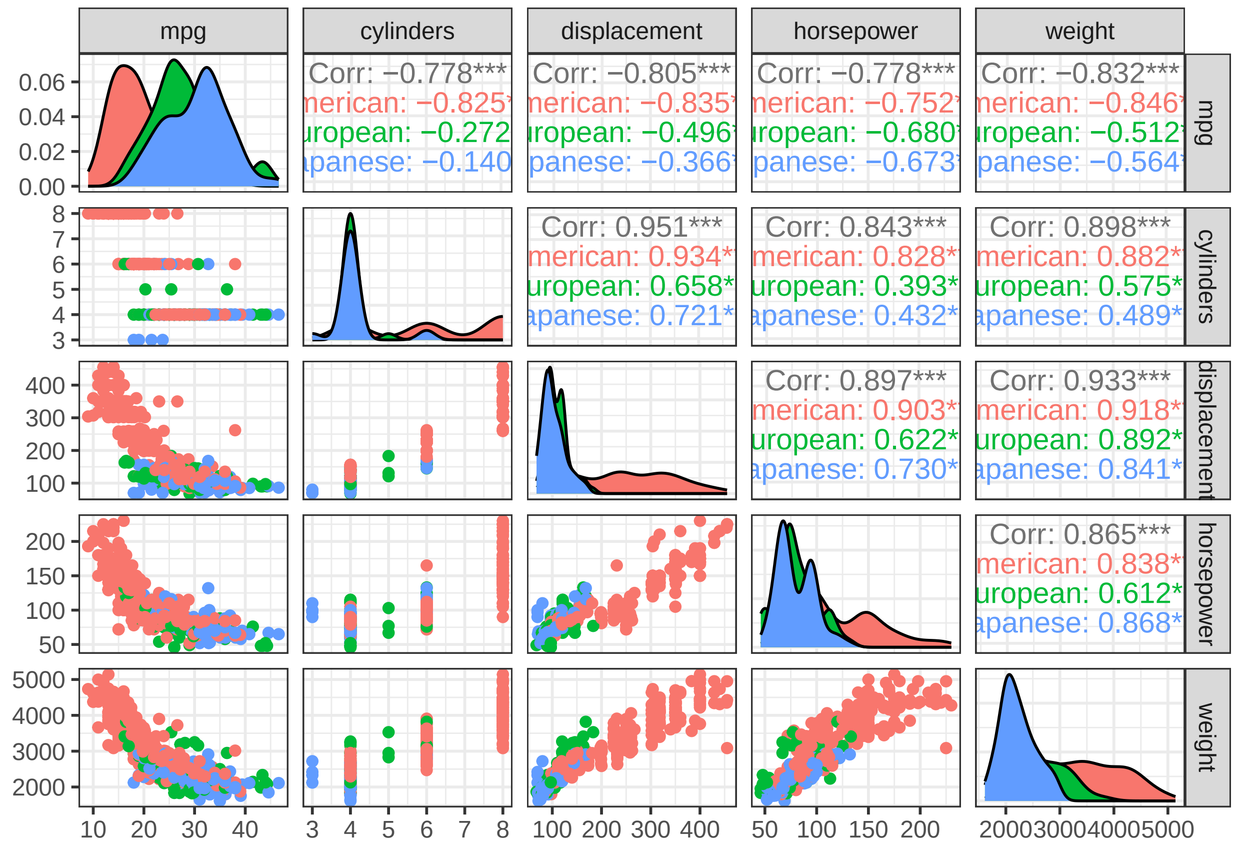
- `mpg`: miles per gallon
- `cylinders`: Number of cylinders between 4 and 8
- `displacement`: Engine displacement (cu. inches)
- `horsepower`: Engine horsepower
- `weight`: Vehicle weight (lbs.)
- `acceleration`: Time to accelerate from 0 to 60 mph (sec.)
- `year`: Model year (modulo 100)
- `origin`: Origin of car (1. American, 2. European, 3. Japanese)
- `name`: Vehicle name

```r
library(ISLR2)
colnames(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

```r
auto.data <- Auto
auto.data$country <- factor(auto.data$origin, labels = c("American", "European",
                                                         "Japanese"))
library("ggplot2")
library("GGally")

ggpairs(auto.data, columns = 1:5, ggplot2::aes(colour=country)) + theme_bw()
```



1 Concepts of principal component

If we use a linear regression model to predict `mpg`, we'll have high correlation amongst the predictors.

```
mpg.lmF <- lm(mpg ~ .-origin-name, data=auto.data)
mpg.lmF
```

```
##
## Call:
## lm(formula = mpg ~ . - origin - name, data = auto.data)
##
## Coefficients:
##     (Intercept)        cylinders     displacement       horsepower
##       -17.95460         -0.48971          0.02398         -0.01818
##          weight     acceleration             year  countryEuropean
##        -0.00671          0.07910          0.77703          2.63000
## countryJapanese
##         2.85323
```

```
car::vif(mpg.lmF)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## cylinders    10.737771  1        3.276854
## displacement 22.937950  1        4.789358
## horsepower    9.957265  1        3.155513
## weight       11.074349  1        3.327814
## acceleration  2.625906  1        1.620465
## year           1.301373  1        1.140777
## country        2.096060  2        1.203236
```

We will focus on the predictors now.

```
auto.X <- model.matrix(mpg.lmF)[, -1]  # Design matrix
```

Function `prcomp()` conducts the Principal Component Analysis.

```
# Caution: the following line of code is not ideal. It is to illustrate an issue.
auto.pc <- prcomp(auto.X)
summary(auto.pc)
```

```
## Importance of components:
##                             PC1      PC2      PC3     PC4   PC5    PC6    PC7
## Standard deviation     855.6585 38.90971 16.16207 3.31353 1.697 0.5249 0.4167
## Proportion of Variance   0.9976  0.00206  0.00036 0.00001 0.000 0.0000 0.0000
## Cumulative Proportion    0.9976  0.99962  0.99998 1.00000 1.000 1.0000 1.0000
##                           PC8
## Standard deviation     0.2446
## Proportion of Variance 0.0000
## Cumulative Proportion  1.0000
```

```
auto.pc
```

```
## Standard deviations (1, .., p=8):
## [1] 855.6585163  38.9097121  16.1620689   3.3135262   1.6966834   0.5249057
## [7]   0.4167494   0.2446327
##
## Rotation (n x k) = (8 x 8):
##                           PC1           PC2          PC3          PC4
## cylinders       -0.0017926225 -0.0133245279  0.007294275  0.001414710
## displacement    -0.1143412856 -0.9457785881  0.303312504 -0.009143349
## horsepower      -0.0389670412 -0.2982553337 -0.948761071 -0.043076559
## weight          -0.9926735354  0.1207516411  0.002454212  0.001480458
## acceleration     0.0013528348  0.0348264293  0.077006895  0.059516278
## year             0.0013368415  0.0238516081  0.042819254 -0.996935229
## countryEuropean  0.0001308250  0.0024889942 -0.002857670  0.022100094
## countryJapanese  0.0002103564  0.0003765828 -0.004796684 -0.012089823
##                           PC5           PC6           PC7           PC8
## cylinders       -1.719368e-02  0.9911554803  0.1211162208  4.909265e-02
## displacement     1.059355e-02 -0.0146594359 -0.0006512752 -4.394368e-03
## horsepower       8.646402e-02  0.0038232742  0.0034425206  4.435100e-03
## weight          -3.152970e-03 -0.0002093216 -0.0003053766 -5.729471e-06
## acceleration     9.944974e-01  0.0168319859  0.0012233398  1.799780e-03
## year             5.549653e-02 -0.0001647840  0.0240346554 -7.643176e-03
## countryEuropean  9.052576e-05 -0.0483462982  0.6888706846 -7.229226e-01
## countryJapanese  1.150938e-03  0.1214929883 -0.7142804151 -6.891098e-01
```

We see that the 1st principal component contains a huge portion of the total variation of X variables, and it is dominated by variable “weight”. Looking at the data, we see that weight simply has the largest values.

For this reason, the variables are usually standardized first (subtract each variable by its mean, then divide each centered-variable by its standard deviation). This is done using `scale=TRUE` argument.

```
auto.pcs <- prcomp(auto.X, scale=TRUE)
summary(auto.pcs)
```

```
## Importance of components:
```

```
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.133 1.1356 0.9497 0.80460 0.62678 0.35401 0.24235
## Proportion of Variance 0.569 0.1612 0.1127 0.08092 0.04911 0.01567 0.00734
## Cumulative Proportion  0.569 0.7302 0.8429 0.92383 0.97294 0.98861 0.99595
##                            PC8
## Standard deviation     0.18005
## Proportion of Variance 0.00405
## Cumulative Proportion  1.00000
```

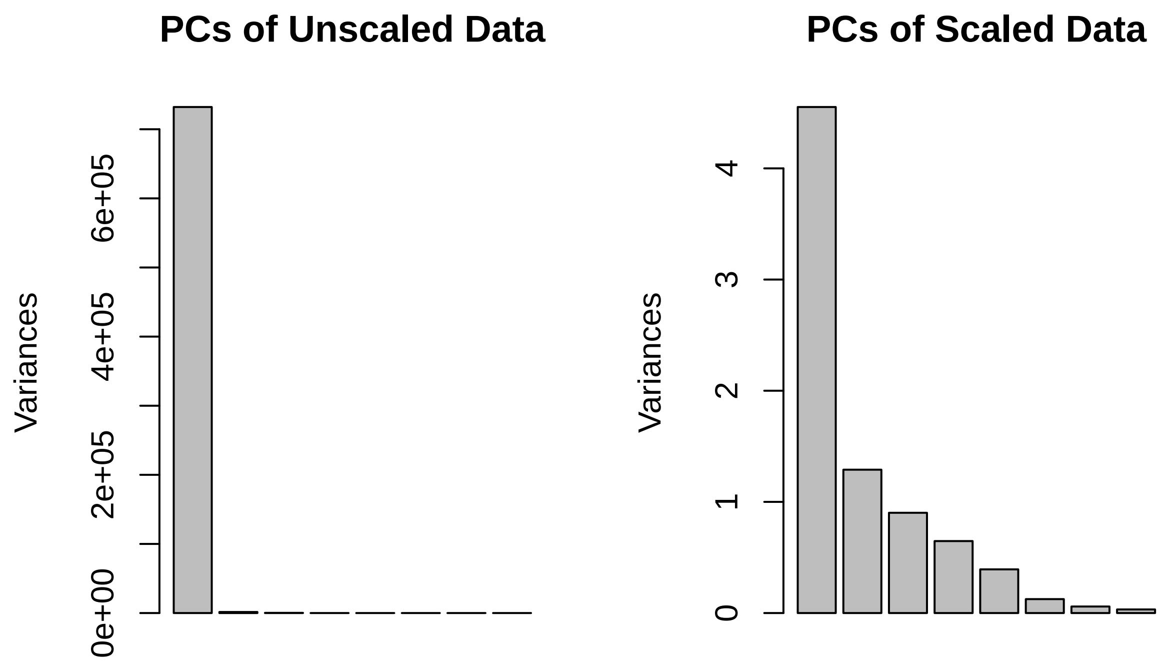
```
round(cor(auto.pcs$x), 4)
```

```
##     PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8
## PC1   1   0   0   0   0   0   0   0
## PC2   0   1   0   0   0   0   0   0
## PC3   0   0   1   0   0   0   0   0
## PC4   0   0   0   1   0   0   0   0
## PC5   0   0   0   0   1   0   0   0
## PC6   0   0   0   0   0   1   0   0
## PC7   0   0   0   0   0   0   1   0
## PC8   0   0   0   0   0   0   0   1
```

```
auto.pcs$rotation # Extract the loading (i.e., transformation)
```

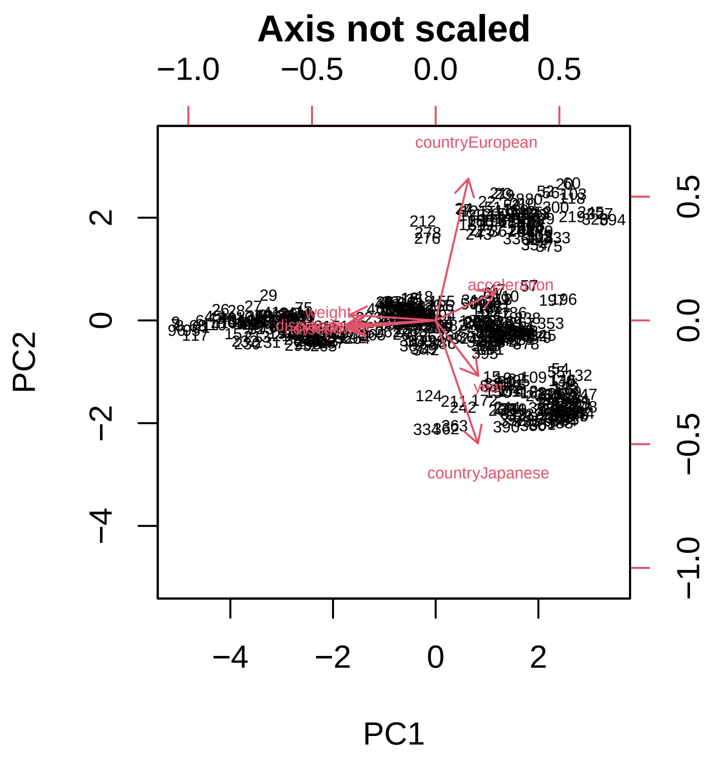
```
##                         PC1         PC2        PC3         PC4         PC5
## cylinders       -0.4418880 -0.03183661  0.1513759  0.08112451 -0.17557152
## displacement    -0.4568839 -0.02468263  0.1334604  0.05932358 -0.09729606
## horsepower      -0.4427613 -0.04007041 -0.1065626 -0.15129036 -0.20758380
## weight          -0.4343646  0.02796626  0.2382679  0.10749689 -0.31123269
## acceleration     0.3038014  0.14422734  0.4574875  0.70453506 -0.30582654
## year             0.2153593 -0.27979875  0.7122813 -0.58611507 -0.13985800
## countryEuropean  0.1648074  0.71513286 -0.1544922 -0.33648172 -0.55320193
## countryJapanese  0.2131795 -0.62087485 -0.3881227  0.04831876 -0.63295244
##                          PC6          PC7         PC8
## cylinders       -0.714889530  0.219309935  0.42922755
## displacement    -0.157462297  0.046158763 -0.85611061
## horsepower       0.571133678  0.610447413  0.16180753
## weight           0.318542871 -0.705651981  0.21433158
## acceleration     0.127257355  0.266640825 -0.02042327
## year             0.007589381  0.067111659 -0.01832329
## countryEuropean -0.114101557 -0.006045755 -0.07244053
## countryJapanese -0.084877251 -0.059530689 -0.06868871
```

```
par(mfrow=c(1, 2))
plot(auto.pc, main="PCs of Unscaled Data")
screeplot(auto.pcs, main="PCs of Scaled Data")
```

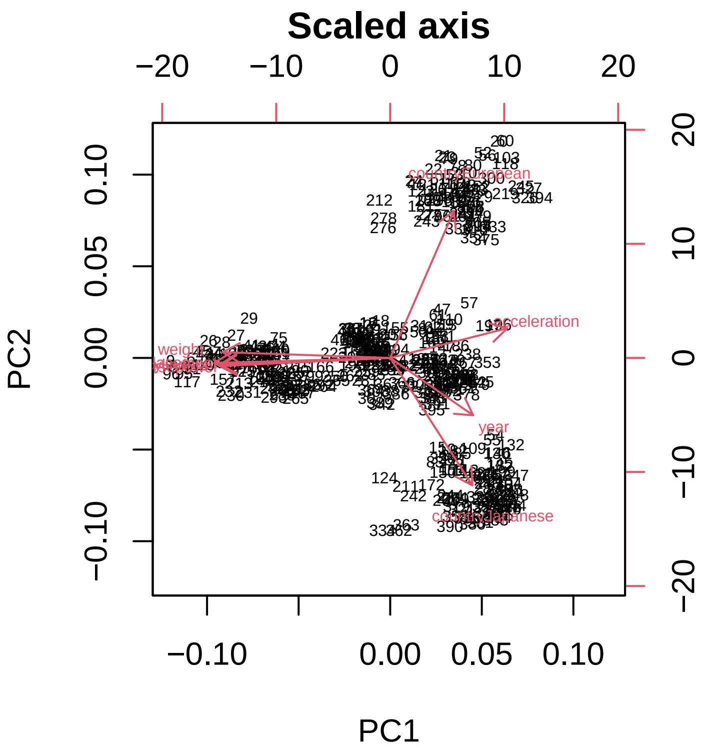


A biplot shows the first 2 PCs scores (bottom, left) and their loadings (top, right).

```r
biplot(auto.pcs, scale=0, cex=0.5, main="Axis not scaled") # Loadings are plotted as is.
```



```r
biplot(auto.pcs, scale=1, cex=0.5, mai="Scaled axis") # Loadings are scaled.
```



For observations in the data frame, their PC scores are saved. PC scores for new observations can be computed using `predict()` function.

```
auto.pcs$x[c(1:3), ]
```

```
##         PC1        PC2       PC3          PC4       PC5        PC6         PC7
## 1 -2.631256 0.17038496 -1.033093  0.327091974 0.4948202 -0.7335556 -0.07630174
## 2 -3.373310 0.10389029 -1.105045  0.110134107 0.2522577 -0.2311124  0.29240679
## 3 -2.984687 0.09245312 -1.259333 -0.009259703 0.5125006 -0.5249749  0.20558186
##           PC8
## 1  0.07875985
## 2 -0.07450685
## 3  0.06308631
```

```
newx <- data.frame(cylinders = 6, displacement = 250, horsepower = 88,
                   weight = 3021, acceleration = 17, year =73,
                   countryEuropean = 0, countryJapanese = 0)
predict(auto.pcs, newdata = newx)
```

```
##             PC1       PC2       PC3      PC4       PC5        PC6         PC7
## [1,] -0.4081881 0.2825017 0.1067126 1.103147 0.4889963 -0.3771085 -0.08551976
##             PC8
## [1,] -0.3085454
```

2 Principal Components regression (PCR)

It will be more convenient to use `pcr()` function from package `pls` to use PCs in regression analysis. It puts 2 analysis into 1 function:

- First, conduct PCA on the predictors and save the PC scores for all observations.
- Next, conduct linear regression use the PCs as the predictors in the model.

```
# install.packages("pls")
library(pls)
```

2.1 Fit PCR

Be sure to include `scale=TRUE` argument.

```
auto.pcr <- pcr(mpg ~ .-name-origin, data=auto.data, scale=T)
summary(auto.pcr)
```

```
## Data:    X dimension: 392 8
##  Y dimension: 392 1
## Fit method: svdpc
## Number of components considered: 8
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       56.9    73.02    84.29    92.38    97.29    98.86    99.59   100.00
## mpg     71.8    73.64    73.96    79.25    79.25    80.22    81.55    82.42
```

- What is the value of R^2 if we use:
  - The first PC for the regression?
  - The first 2 PCs for the regression?
  - The first 5 PCs for the regression?
  - All PCs (8 in this case) for thr regression?

2.2 Cross-validation

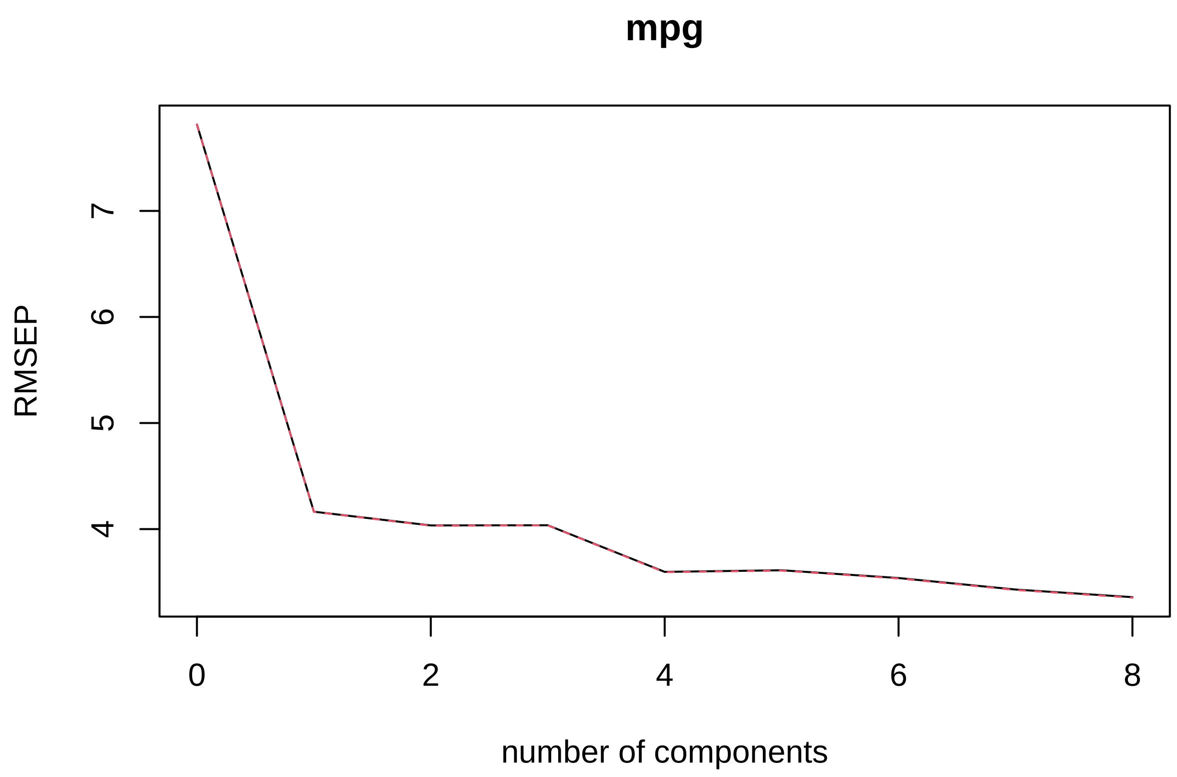
`pcr()` function has be built-in option for K-fold (default $K = 10$) and LOOCV. Use argument `validation = "CV"` or `validation = "LOO"`.

```
auto.pcrCV <- pcr(mpg ~ .-name-origin, data=auto.data, scale=T, validation="CV")
summary(auto.pcrCV)
```

```
## Data:    X dimension: 392 8
##  Y dimension: 392 1
## Fit method: svdpc
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           7.815    4.164    4.034    4.036    3.597    3.612    3.538
## adjCV        7.815    4.163    4.032    4.035    3.594    3.608    3.535
##        7 comps  8 comps
## CV       3.430    3.358
## adjCV    3.426    3.353
##
```

```
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       56.9    73.02    84.29    92.38    97.29    98.86    99.59   100.00
## mpg     71.8    73.64    73.96    79.25    79.25    80.22    81.55    82.42
```

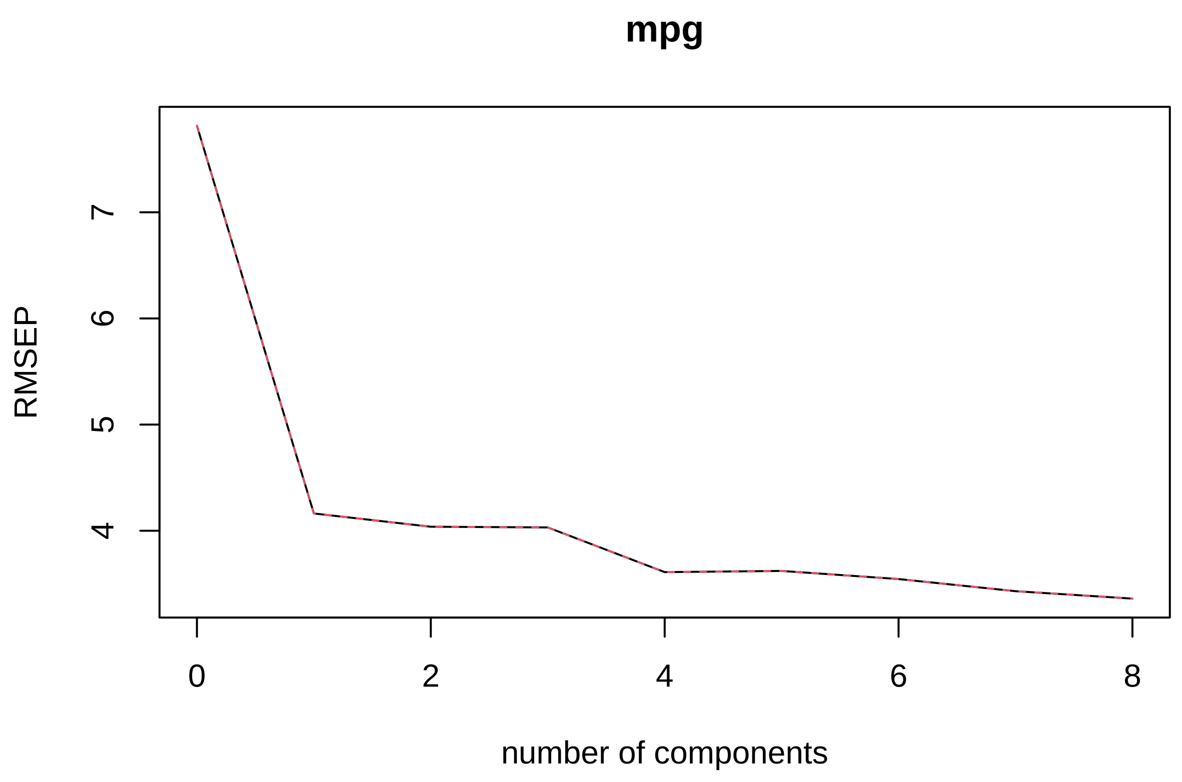
```
validationplot(auto.pcrCV)
```



```
auto.pcrLOO <- pcr(mpg ~ .-name-origin, data=auto.data, scale=T, validation="LOO")
summary(auto.pcrLOO)
```

```
## Data:    X dimension: 392 8
##  Y dimension: 392 1
## Fit method: svdpc
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 392 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           7.815    4.163    4.038    4.031     3.61    3.621    3.545
## adjCV        7.815    4.163    4.038    4.031     3.61    3.621    3.545
##        7 comps  8 comps
## CV       3.431    3.361
## adjCV    3.431    3.360
##
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       56.9    73.02    84.29    92.38    97.29    98.86    99.59   100.00
## mpg     71.8     73.64     73.96     79.25     79.25     80.22     81.55     82.42
```

```r
validationplot(auto.pcrLOO)
```



2.3 Predict the response for new data.

`predict()` function works for `pcr()` output object. Use argument `ncomp =` to declare how many PC's to use in the model.

```r
predict(auto.pcr, newdata=auto.data[c(1, 2), ], ncomp=4) # If 4 PCs are used.
```

```
## , , 4 comps
##
##        mpg
## 1 13.92093
## 2 12.13321
```

3 Partial Least Squares regression (PLS)

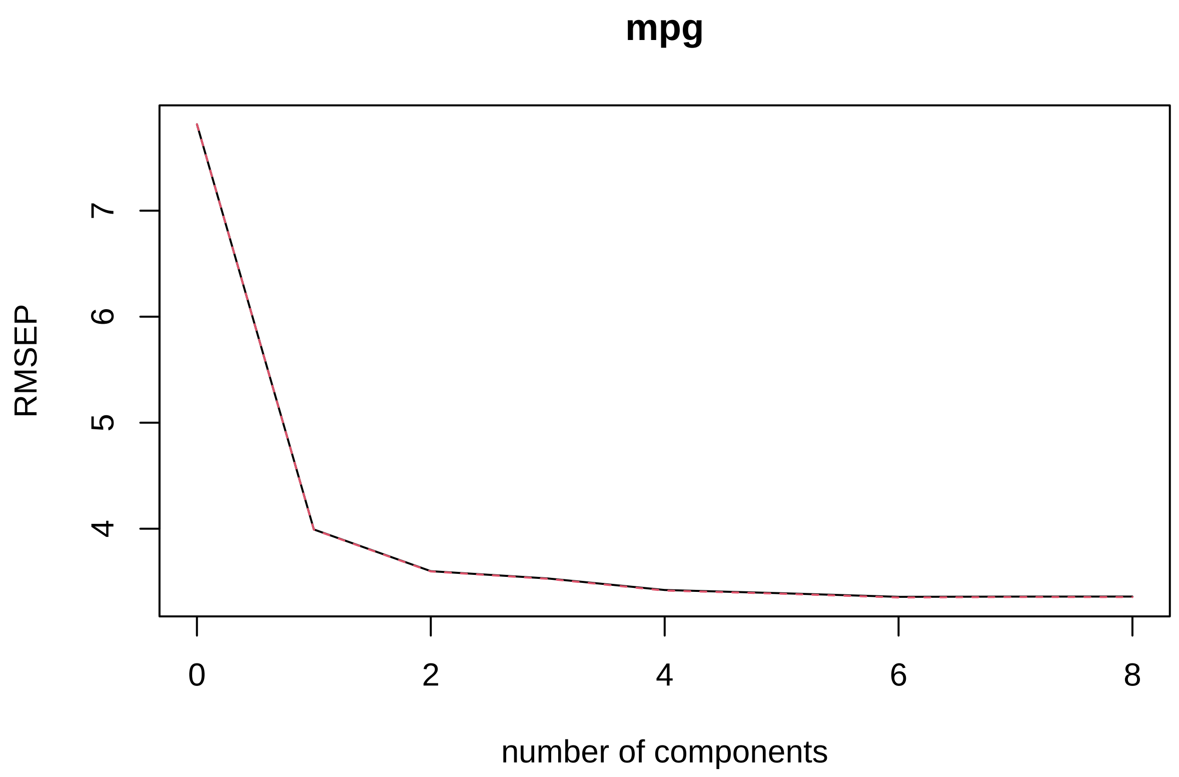
Use function `plsr()`. It is similar to `pcr()`

```r
auto.plsCV <- plsr(mpg ~ .-name-origin, data=auto.data, scale=T, validation="CV")
summary(auto.plsCV)
```

```
## Data:    X dimension: 392 8
##  Y dimension: 392 1
## Fit method: kernelpls
## Number of components considered: 8
```

```
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           7.815    3.993    3.600    3.531    3.422    3.391    3.357
## adjCV        7.815    3.991    3.597    3.527    3.416    3.387    3.352
##        7 comps  8 comps
## CV       3.360    3.360
## adjCV    3.356    3.355
##
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      56.73    68.84    80.75    84.08    93.48    94.88    99.33   100.00
## mpg    74.32    79.37    80.29    81.71    82.00    82.35    82.38    82.42
```

```r
validationplot(auto.plsCV)
```



```r
names(auto.plsCV)
```

```
##  [1] "coefficients"    "scores"          "loadings"        "loading.weights"
##  [5] "Yscores"         "Yloadings"       "projection"      "Xmeans"
##  [9] "Ymeans"          "fitted.values"   "residuals"       "Xvar"
## [13] "Xtotvar"         "fit.time"        "ncomp"           "method"
## [17] "center"          "scale"           "validation"      "call"
## [21] "terms"           "model"
```

```r
predict(auto.plsCV, newdata=auto.data[c(1, 2), ], ncomp=4) # If 4 PCs are used.
```

```
## , , 4 comps
##
##         mpg
## 1 15.92016
## 2 13.97132
```