Lecture 19 PLS Simulation

Load dataset

```
cars2004 <- read.csv('../data/cars2004.csv', stringsAsFactors = F)
cars2004 <- cars2004[, -1]
str(cars2004, vec.len = 1)
```

```
## 'data.frame':    385 obs. of  10 variables:
##  $ price    : int  43755 46100 ...
##  $ engine   : num  3.5 3.5 ...
##  $ cyl      : int  6 6 ...
##  $ hp       : int  225 225 ...
##  $ city_mpg: int  18 18 ...
##  $ hwy_mpg : int  24 24 ...
##  $ weight   : int  3880 3893 ...
##  $ wheel    : int  115 115 ...
##  $ length   : int  197 197 ...
##  $ width    : int  72 72 ...
```

Obtain the vector of weights

$\mathbf{w_1} = \mathbf{X_0^T y} \propto cov(\mathbf{X_0}, \mathbf{y})$

If normalizing $\mathbf{w_1}$, then $w_{1j} = \frac{cov(\mathbf{x_{0,j}}, \mathbf{y})}{\sum_{j=1}^{p} cov^2(\mathbf{x_{0,j}}, \mathbf{y})}$

```
# Centralized X and Y, but not scaled
X0 <- scale(cars2004[, -1], center = T, scale = F)
Y0 <- scale(cars2004[, 1], center = T, scale = F)
w1 <- t(X0) %*% Y0
scalar1 <- function(x) {x / sqrt(sum(x^2))}  # normalize
(w1 <- scalar1(w1))
```

```
##                  [,1]
## engine    0.001782118
## cyl       0.002857956
## hp        0.171985612
## city_mpg -0.007484109
## hwy_mpg  -0.007752089
## weight    0.984987298
## wheel     0.004225081
## length    0.008131684
## width     0.003089621
```

```
sum(scalar1(w1)^2)  # 1
```

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

```
unique(round(w1 / cov(X0, Y0)))  # prop to covariance matrix
```

```
##        [,1]
## engine    0
```

```r
# Alternative ways to calculate scaled W1
# w1 <- cov(X0, Y0) / sqrt(sum(cov(X0, Y0)^2))
# w1 <- scalar1(xt(X0) %*% Y0 / as.numeric(t(Y0) %*% Y0))
```

Obtain the first PLS component

$\mathbf{z_1} = \mathbf{X_0 w_1}$

Since $\mathbf{X} = \mathbf{Z}\mathbf{P}^\mathbf{T} + \mathbf{E}$, so $\mathbf{P}^\mathbf{T} = (\mathbf{Z}^\mathbf{T}\mathbf{Z})^{-1}\mathbf{Z}^\mathbf{T}\mathbf{X}$ and so $\mathbf{P_1} = \mathbf{X_0^T z_1}/\mathbf{z_1^T z_1}$

Since $\mathbf{y} = \mathbf{Z}\mathbf{d} + \mathbf{e}$, so $\mathbf{d} = (\mathbf{Z}^\mathbf{T}\mathbf{Z})^{-1}\mathbf{Z}^\mathbf{T}\mathbf{Y}$ and so $d_1 = \mathbf{Y_0^T z_1}/\mathbf{z_1^T z_1}$

```r
# Obtain the first PLS component
z1 <- X0 %*% w1
head(z1)
```

```
##             [,1]
## [1,]  344.24572
## [2,]  357.05055
## [3,]  913.48050
## [4,] -360.90753
## [5,] -745.89228
## [6,]   51.39841
```

```r
# Obtain a vector p1 of loadings
(p1 <- t(X0) %*% z1 / as.numeric(t(z1) %*% z1))
```

```
##                  [,1]
## engine    0.001176718
## cyl       0.001561745
## hp        0.064016991
## city_mpg -0.005536001
## hwy_mpg  -0.006343509
## weight    1.003819205
## wheel     0.007551534
## length    0.012276141
## width     0.003862309
```

```r
# p1 = (z1tz1)^{-1}(z1tX0)
p1t <- solve(t(z1) %*% z1) %*% t(z1) %*% X0

# Obtain regression coefficient d1
(d1 <- t(Y0) %*% z1 / as.numeric(t(z1) %*% z1))
```

```
##          [,1]
## [1,] 13.61137
```

```r
yhat <- z1 %*% d1
```

Obtain the second PLS component

```r
# Deflate each xj w.r.t. z1
X1 <- X0 - z1 %*% t(p1)

# Deflate y0 w.r.t. z1
```

```r
Y1 <- Y0 - z1 %*% d1

# Obtain the vector of weights
(w2 <- scalar1(t(X1) %*% Y1))
```

```
##                  [,1]
## engine    0.005515374
## cyl       0.011808873
## hp        0.983627243
## city_mpg -0.017747864
## hwy_mpg  -0.012832587
## weight   -0.171564444
## wheel    -0.030305000
## length   -0.037757269
## width    -0.007039429
```

```r
# Obtain the second PLS component
z2 <- X1 %*% w2
head(z2)
```

```
##            [,1]
## [1,] -12.053948
## [2,] -12.878753
## [3,]  -7.619465
## [4,] 100.553595
## [5,]  33.927682
## [6,]  52.930801
```

```r
# Obtain a vector p2 of loadings
(p2 <- t(X1) %*% z2 / as.numeric(t(z2) %*% z2))
```

```
##                  [,1]
## engine    0.006139666
## cyl       0.011322638
## hp        0.984752947
## city_mpg -0.024940504
## hwy_mpg  -0.019595807
## weight   -0.172152033
## wheel    -0.014980542
## length   -0.013459092
## width    -0.001586278
```

```r
# p2 = (z2tz2)^{-1}(z2tX1)
p2t <- solve(t(z2) %*% z2) %*% t(z2) %*% X1

# Obtain regression coefficient d2
(d2 <- t(Y1) %*% z2 / as.numeric(t(z2) %*% z2))
```

```
##          [,1]
## [1,] 247.0772
```

```r
yhat <- z1 %*% d1 + z2 %*% d2
```

Obtain the third PLS component

```
# Deflate each xj w.r.t. z2
X2 <- X1 - z2 %*% t(p2)

# Deflate y0 w.r.t. z2
Y2 <- Y1 - z2 %*% d2

# Obtain the vector of weights
(w3 <- scalar1(t(X2) %*% Y2))
```

```
##                  [,1]
## engine   -0.02020490
## cyl       0.01573673
## hp       -0.03643282
## city_mpg  0.23278601
## hwy_mpg   0.21888803
## weight    0.01901701
## wheel    -0.49596803
## length   -0.78639773
## width    -0.17648838
```

```
# Obtain the second PLS component
z3 <- X2 %*% w3
head(z3)
```

```
##            [,1]
## [1,] -8.8605669
## [2,] -8.6388306
## [3,] 10.0986288
## [4,]  3.9017210
## [5,]  2.7514665
## [6,] -0.8229918
```

```
# Obtain a vector p3 of loadings
(p3 <- t(X2) %*% z3 / as.numeric(t(z3) %*% z3))
```

```
##                  [,1]
## engine   -0.01501603
## cyl      -0.01775333
## hp       -0.04767647
## city_mpg  0.03786978
## hwy_mpg  -0.05070461
## weight    0.01857051
## wheel    -0.38912886
## length   -1.00271271
## width    -0.10562988
```

```
# Obtain regression coefficient d3
(d3 <- t(Y2) %*% z3 / as.numeric(t(z3) %*% z3))
```

```
##          [,1]
## [1,] 238.3535
```

```
yhat <- z1 %*% d1 + z2 %*% d2 + z3 %*% d3
```

Propoerties

```r
t(z1) %*% z2 < 1e-7
```

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

```r
t(w1) %*% p1
```

```
##      [,1]
## [1,]    1
```

```r
sum(t(w1) %*% t(X2) > 1e-7)
```

```
## [1] 0
```

Modified Weights

```r
w1_star <- w1 %*% solve(t(p1) %*% w1)
w2_star <- w2 %*% solve(t(p2) %*% w2)
z1_alt <- X0 %*% w1_star
z2_alt <- X1 %*% w2_star
head(cbind(z1, z1_alt, z2, z2_alt))
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,]  344.24572  344.24572 -12.053948 -12.053948
## [2,]  357.05055  357.05055 -12.878753 -12.878753
## [3,]  913.48050  913.48050  -7.619465  -7.619465
## [4,] -360.90753 -360.90753 100.553595 100.553595
## [5,] -745.89228 -745.89228  33.927682  33.927682
## [6,]   51.39841   51.39841  52.930801  52.930801
```

Decomposition

```r
betaOLS <- solve(t(X0) %*% X0) %*% t(X0) %*% Y0
# Suppose to show betaOLS = sum(d %*% w_star)
```

Gasoline Data

Preliminary Analysis

```r
gasoline <- read.table("../data/gasoline.txt", header = TRUE)
dim(gasoline)
```

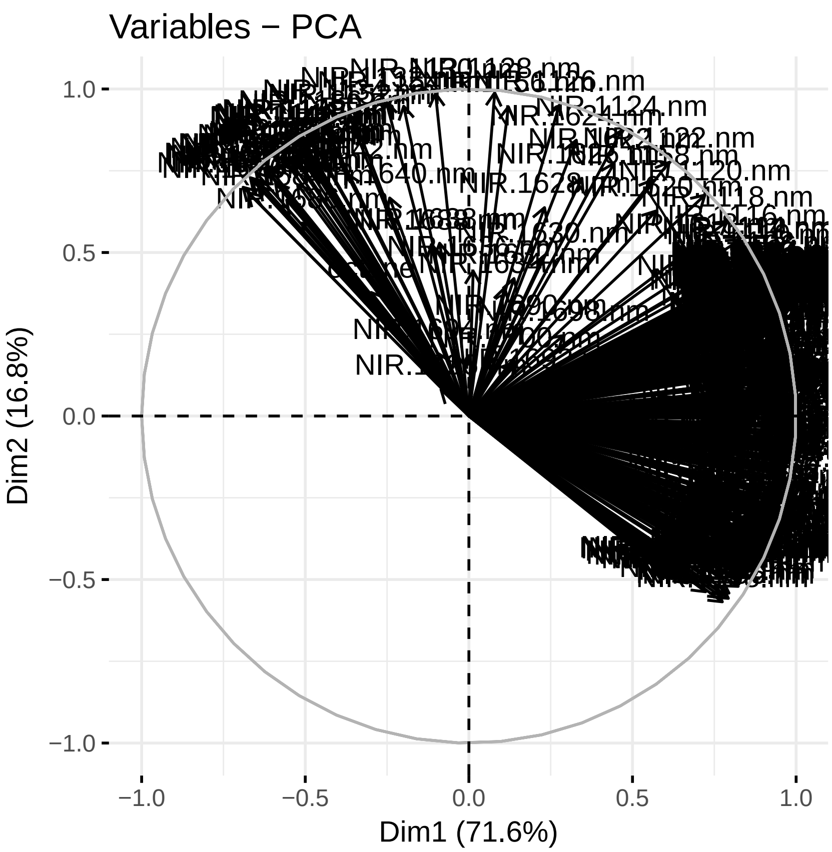
```
## [1]  60 402
```

```r
# Circle of correlations
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
library("FactoMineR")
gasoline.pca <- PCA(gasoline, ncp = NCOL(gasoline), graph = FALSE)
# scores <- cars2004.pca$ind$coord
fviz_pca_var(gasoline.pca, col.var = "black")
```




```r
octane <- gasoline[, 1]  # response
NIR <- gasoline[, 2 : ncol(gasoline)]  # predictors

# training and test sets
train <- 1:50
test <- 51:60

corrs <- cor(NIR, octane)
summary(corrs)
```

```
##        V1
##  Min.   :-0.90362
##  1st Qu.:-0.38877
##  Median :-0.19437
##  Mean   :-0.18578
##  3rd Qu.:-0.05055
##  Max.   : 0.56396
```

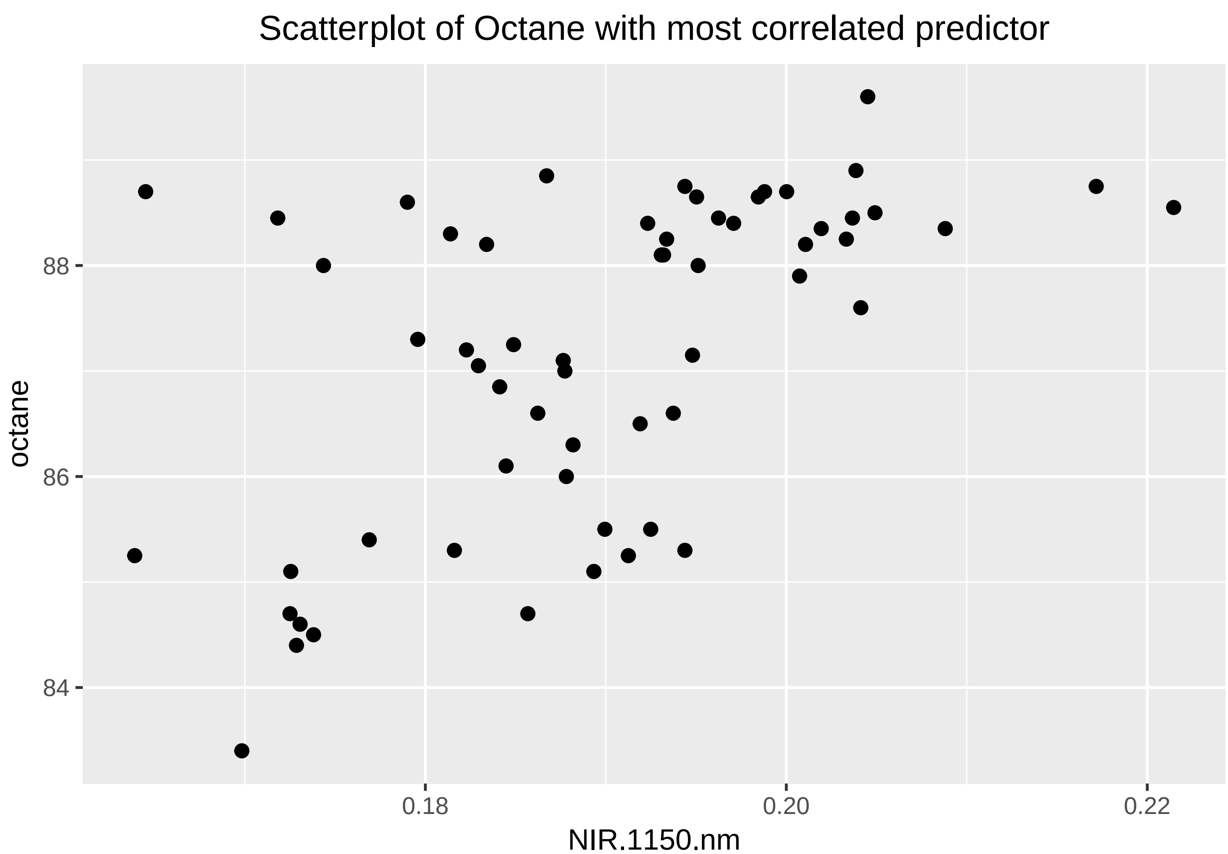
```r
which.max(corrs)
```

```
## [1] 126
```

```r
corrs[which.max(corrs)]
```

```
## [1] 0.5639595
```

```r
ggplot(gasoline, aes(x=NIR.1150.nm, y=octane)) + geom_point(size=2) +
  labs(title = "Scatterplot of Octane with most correlated predictor") +
  theme(plot.title = element_text(hjust = 0.5))
```



OLS Regression analysis

```r
# OLS regression attempt
gas_train <- gasoline[train, ]
gas_test <- gasoline[test, ]
reg <- lm(octane ~ ., data = gas_train)
# print(summary(reg))
```

PLS Regression analysis

```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked _by_ '.GlobalEnv':
##
```

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

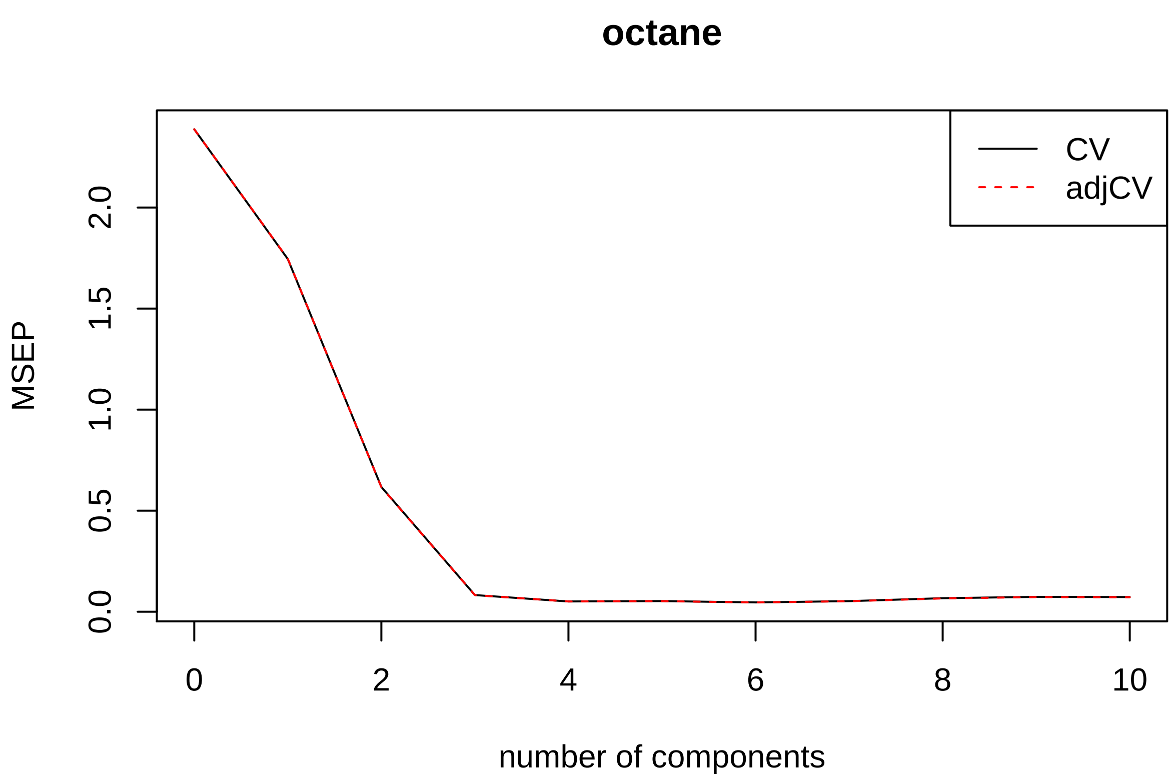
```r
set.seed(1)
pls1 <- plsr(octane ~ ., ncomp = 10, data = gasoline, subset = train,
             scale = TRUE, validation = "LOO")
pls1
```

```
## Partial least squares regression , fitted with the kernel algorithm.
## Cross-validated using 50 leave-one-out segments.
## Call:
## plsr(formula = octane ~ ., ncomp = 10, data = gasoline, subset = train,     scale = TRUE, validation
```

```r
summary(pls1)
```

```
## Data:    X dimension: 50 401
##  Y dimension: 50 1
## Fit method: kernelpls
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 50 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.545    1.321   0.7857   0.2869   0.2254   0.2295   0.2145
## adjCV        1.545    1.322   0.7848   0.2866   0.2251   0.2287   0.2141
##        7 comps  8 comps  9 comps  10 comps
## CV      0.2287   0.2586   0.2710    0.2695
## adjCV   0.2279   0.2567   0.2692    0.2676
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         64.31    85.24    95.79    97.22    97.59    98.19    98.61
## octane    31.59    79.29    97.13    98.49    98.91    99.01    99.10
##         8 comps  9 comps  10 comps
## X         98.74    99.10     99.25
## octane    99.37    99.46     99.57
```

```r
plot(MSEP(pls1), legendpos = "topright")
```

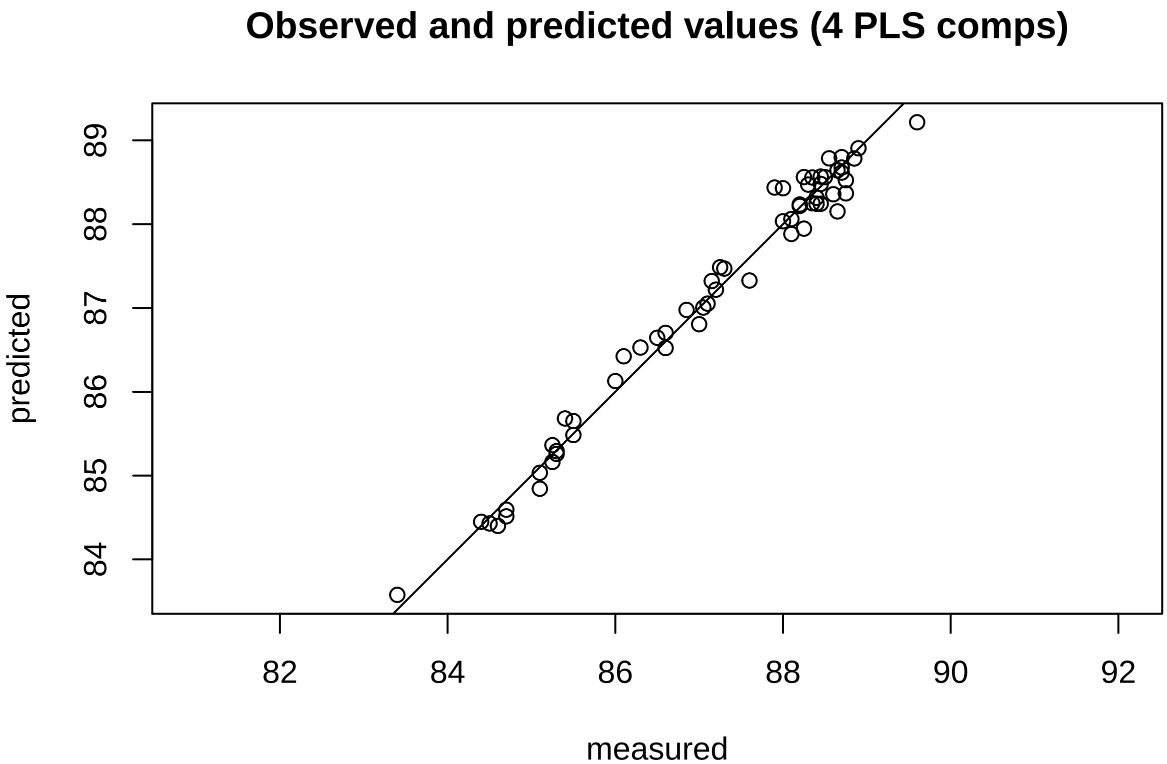



```
# Test MSEs
mse_test <- MSEP(pls1, newdata = gas_test)
# RMSEP(pls1, newdata = gas_test)

pls_fit <- plsr(octane ~ ., ncomp = 4, data = gasoline, scale = T)
summary(pls_fit)
```

```
## Data:    X dimension: 60 401
##  Y dimension: 60 1
## Fit method: kernelpls
## Number of components considered: 4
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps
## X         64.97    83.51    93.72    96.33
## octane    30.54    79.79    97.73    98.27
```

```
plot(pls_fit, ncomp = 4, asp = 1, line = TRUE,
     main = "Observed and predicted values (4 PLS comps)")
```


Observed and predicted values (4 PLS comps)
predicted
measured
84
85
86
87
88
89
82
84
86
88
90
92