Stat 154 HW 06

Mokssh Surve

3/17/2020

```
library('matrixcalc')
library('ggplot2')
library('dplyr')
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library('reshape2')
library('tibble')
full_data <- read.csv("cars2004.csv")
full_data <- scale(read.csv("cars2004.csv")[,-1])
```

2) Coding PCR

```
my_pcr <- function(k, X, y) {

  data <- cbind(y,X)
  data.svd <- svd(data)

  S <- diag(nrow=k)
  diag(S) <- data.svd$d[1:k]

  U <- data.svd$u[, 1:k]

  #components
  pc <- U %*% S

  #loadings
  loadings <- data.svd$v[, 1:k]
```

```
  #coefficients
  coef <- matrix.inverse(t(pc) %*% pc) %*% t(pc) %*% y

  #xcoefficients
  xcoef <- loadings %*% matrix.inverse(S) %*% t(U[, 1:k]) %*% y

  #fitted
  y_hat <- pc %*% coef
  y_hat

}
```

3) PCR using data set cars2004.csv

```
full_data <- scale(read.csv("cars2004.csv")[,-1])
full_data_nameless <- as.data.frame(full_data)
full_data <- cbind(read.csv("cars2004.csv")[,1], full_data_nameless)

#price variable to be predicted
y <- full_data$price
#remaining 9 variables
X <- full_data[, -c(1,2)]

data <- X
data.svd <- svd(data)

k=9
S <- diag(nrow=k)
diag(S) <- data.svd$d[1:k]
U <- data.svd$u[, 1:k]
pc <- U %*% S
loadings <- data.svd$v[, 1:k]

#PC coefficients
pc_coef <- matrix.inverse(t(pc) %*% pc) %*% t(pc) %*% y

#X coefficients
x_coef <- loadings %*% matrix.inverse(S) %*% t(U[, 1:k]) %*% y

#Predicted Values
y_hat <- pc %*% pc_coef
#Residuals
res <- y_hat - y
#MSE of Residuals
mse_res <- mean(res^2)
```

3.1) Vector of PCR coefficients (i.e. b)

```
pc_coef
```

```
##             [,1]
##  [1,] -0.22607128
##  [2,]  0.38473210
##  [3,] -0.48798433
##  [4,] -0.08942945
##  [5,]  0.53237283
##  [6,] -0.28285634
##  [7,] -0.29057867
##  [8,] -0.38461967
##  [9,]  0.27675568
```

3.2) Vector of coefficients in terms of X-predictors (i.e. b∗)

```
x_coef
```

```
##             [,1]
##  [1,] -0.16822457
##  [2,]  0.19044147
##  [3,]  0.87744314
##  [4,] -0.06133804
##  [5,]  0.27993358
##  [6,]  0.35561173
##  [7,] -0.24968842
##  [8,]  0.02260203
##  [9,] -0.10830459
```

3.3) Compute residuals, and show the first 10 elements (first 10 residual values), as well as the mean squared error of residuals

```
res[1:10]
```

```
##  [1] -0.66812244 -0.78016925  0.56619191 -1.96778032  0.32310517  0.72575127
##  [7]  0.09382444 -0.02147918  0.34010671 -0.10457687
```

```
mse_res
```

```
## [1] 0.2543605
```

4) PCR regularizing effect

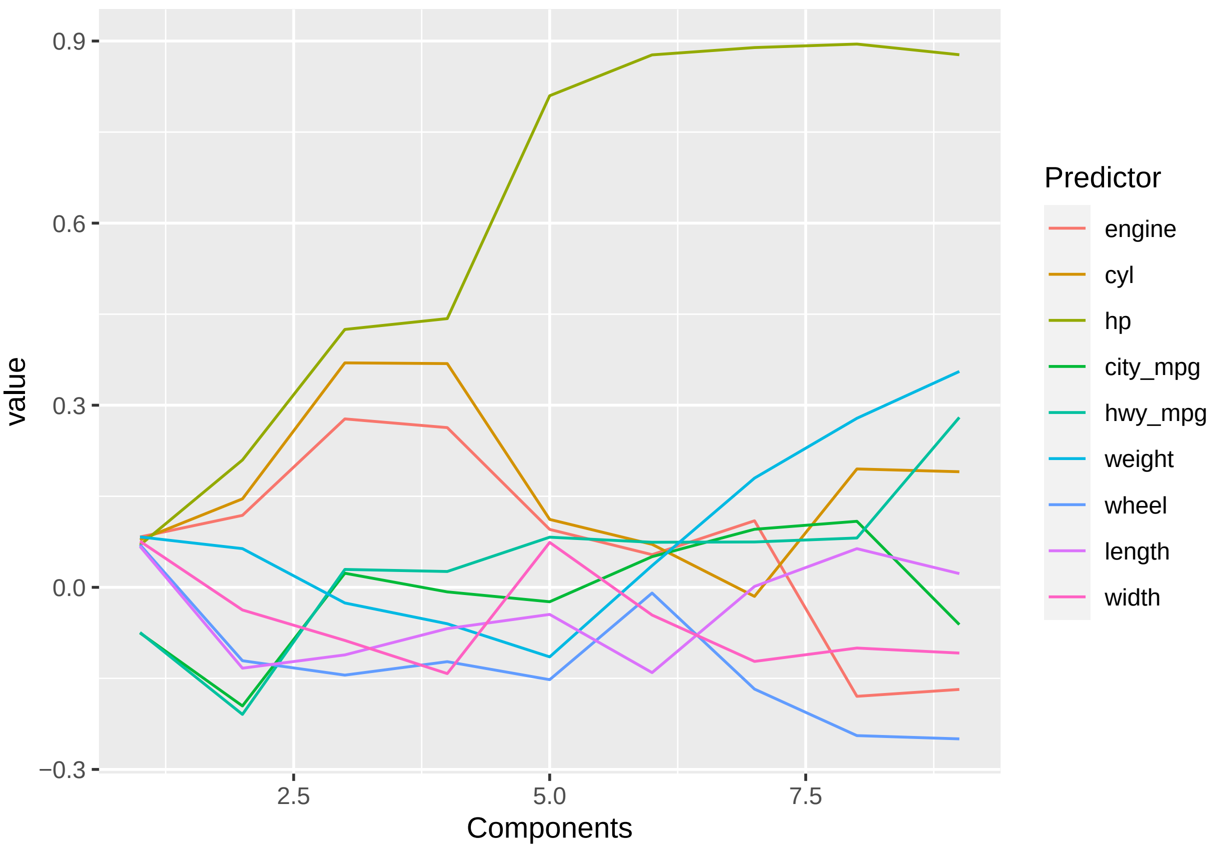
4.1)

```r
final_matrix <- matrix(0, nrow=9, ncol=9)

for (k in 1:9){

  if (k!=1){
    S <- diag(nrow=k)
    diag(S) <- data.svd$d[1:k]
  }
  else{
    S <- as.matrix(data.svd$d[1])
  }

  U <- as.matrix(data.svd$u[, 1:k])
  pc <- U %*% S
  loadings <- data.svd$v[, 1:k]

  #PC coefficients
  pc_coef <- matrix.inverse(t(pc) %*% pc) %*% t(pc) %*% y
  #X coefficients
  x_coef <- loadings %*% matrix.inverse(S) %*% t(U[, 1:k]) %*% y


  final_matrix[1:(length(x_coef)), k] <- x_coef

}

colnames(final_matrix) <- c('C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9')
rownames(final_matrix) <- c('engine', 'cyl', 'hp', 'city_mpg', 'hwy_mpg', 'weight', 'wheel', 'length',
final_matrix
```

```
##                  C1          C2          C3           C4          C5
## engine   0.08296078  0.11858508  0.27746810  0.263115893  0.09539968
## cyl      0.07805857  0.14570129  0.36976227  0.368572650  0.11188000
## hp       0.06961500  0.20975269  0.42485173  0.442715080  0.80989067
## city_mpg -0.07519778 -0.19541469  0.02321521 -0.007512768 -0.02377814
## hwy_mpg  -0.07443014 -0.20934517  0.02947971  0.026106008  0.08258079
## weight   0.08302231  0.06375174 -0.02578997 -0.060013790 -0.11459868
## wheel    0.07004472 -0.12093425 -0.14462390 -0.122487413 -0.15213017
## length   0.06734445 -0.13320854 -0.11132817 -0.068069878 -0.04465391
## width    0.07587851 -0.03732745 -0.08752653 -0.142143016  0.07401109
##                   C6           C7          C8          C9
## engine    0.053803020  0.109653843 -0.17957823 -0.16822457
## cyl       0.070825163 -0.014942256  0.19499259  0.19044147
## hp        0.877218133  0.889248936  0.89500854  0.87744314
## city_mpg  0.050509754  0.095635344  0.10875780 -0.06133804
## hwy_mpg   0.074256272  0.074731626  0.08127063  0.27993358
## weight    0.035734795  0.179902267  0.27847992  0.35561173
## wheel    -0.009468076 -0.167664846 -0.24431450 -0.24968842
## length   -0.140491594  0.001425069  0.06370463  0.02260203
## width    -0.045711353 -0.122055940 -0.10003648 -0.10830459
```

4.2) Path graph of coefficients

```
final_df <- as.data.frame(t(final_matrix))
final_df$Components <- 1:9

melted_df <- melt(final_df ,  id.vars = 'Components', variable.name = 'Predictor')
ggplot(melted_df, aes(Components,value)) + geom_line(aes(colour = Predictor))
```



**************************** COMMENT ON PATTERN ****************************

The X-predictors refer to the coefficients associated with the original predictors. Thus, the coefficient value on the Y axis corresponding to the number of components gives us some information on how prevalent the original predictors are in the PCs being used - since after all the PCs are linear combinations of the original predictors => there is a chained relationshp between the original predictors and the coefficients in the PCR. As for the most notable patterns, the coefficients associated with the **hp** and **weight** characteristics of the car **increase** the most as the complexity i.e. the number of PCs used increased increases.
On the other hand, the values associated with the coefficients of **wheel** and **engine** characteristics **decrease** drastically as the complexity (no. of principal components) increases.

Part II) Partial Least Squares Regression (PLSR)

5) Coding PLSR

```
my_plsr <- function(X, y, k){

  r <- matrix.rank(X)
  X_0 <- X
  y_0 <- y
  Z <- NULL
  W <- NULL
  V <-  NULL


  for (i in 1:k){

    w_h <- (t(X_0) %*% y_0) / (t(y_0) %*% y_0)
    w_h <- w_h / (sqrt(sum(w_h^2)))
    z_h <- (X_0 %*% w_h) / (t(w_h) %*% w_h)
    v_h <- (t(X_0) %*% z_h) / (t(z_h) %*% z_h)
    X_0 <- X_0 - (z_h %*% t(v_h))
    b_h <- (t(y_0) %*% z_h) / (t(z_h) %*% z_h)
    y_0 <- y_0 - (b_h %*% z_h)

    Z <- cbind(Z, z_h)
    W <- cbind(W, w_h)
    V <- cbind(V, v_h)

  }

  W_star <- W %*% matrix.inverse(t(V) %*% W)
  b_star <- b_h %*% W_star

}
```

6) PLSR using data set cars2004.csv

```
full_data <- scale(read.csv("cars2004.csv")[,-1])
full_data_nameless <- as.data.frame(full_data)
full_data <- cbind(read.csv("cars2004.csv")[,1], full_data_nameless)

#price variable to be predicted
y <- as.matrix(full_data$price)
#remaining 9 variables
X <- as.matrix(full_data[, -c(1,2)])

  r <- 9
  X_0 <- X
  y_0 <- y
```

```r
  Z <- NULL
  W <- NULL
  V <-  NULL
  b <- NULL


  for (i in 1:r){

    w_h <- as.matrix((t(X_0) %*% y_0) / as.numeric((t(y_0) %*% y_0)))
    w_h <- as.matrix(w_h / (sqrt(sum(w_h^2))))
    z_h <- as.matrix((X_0 %*% w_h) / as.numeric((t(w_h) %*% w_h)))
    v_h <- (t(X_0) %*% z_h) / as.numeric((t(z_h) %*% z_h))
    X_0 <- X_0 - (z_h %*% t(v_h))
    b_h <- as.numeric((t(y_0) %*% z_h) / as.numeric((t(z_h) %*% z_h)))
    y_0 <- y_0 - (b_h * z_h)

    Z <- cbind(Z, z_h)
    W <- cbind(W, w_h)
    V <- cbind(V, v_h)
    b <- append(b, b_h)

  }

  W_star <- W %*% matrix.inverse(t(V) %*% W)
  b_star <- W_star %*% b

  y_hat <- Z %*% b
  res <- y_hat - y
  mse_res <- mean(res^2)
```

6.1) Vector of PLSR coefficients (i.e. b)

```r
b_h
```

```
## [1] 0.01100328
```

6.2) Vector of coefficients in terms of X-predictors (i.e. b∗)

```r
b_star
```

```
##                 [,1]
## engine   -0.16822457
## cyl       0.19044147
## hp        0.87744314
## city_mpg -0.06133804
## hwy_mpg   0.27993358
## weight    0.35561173
## wheel    -0.24968842
## length    0.02260203
## width    -0.10830459
```

6.3) Compute residuals, and show the first 10 elements (first 10 residual values), as well as the mean squared error of residuals

```
res[1:10]
```

```
##  [1] -0.66812244 -0.78016925  0.56619191 -1.96778032  0.32310517  0.72575127
##  [7]  0.09382444 -0.02147918  0.34010671 -0.10457687
```

```
mse_res
```

```
## [1] 0.2543605
```

7) PLSR regularizing effect

7.1) Regression Coefficients in terms of X-predictors

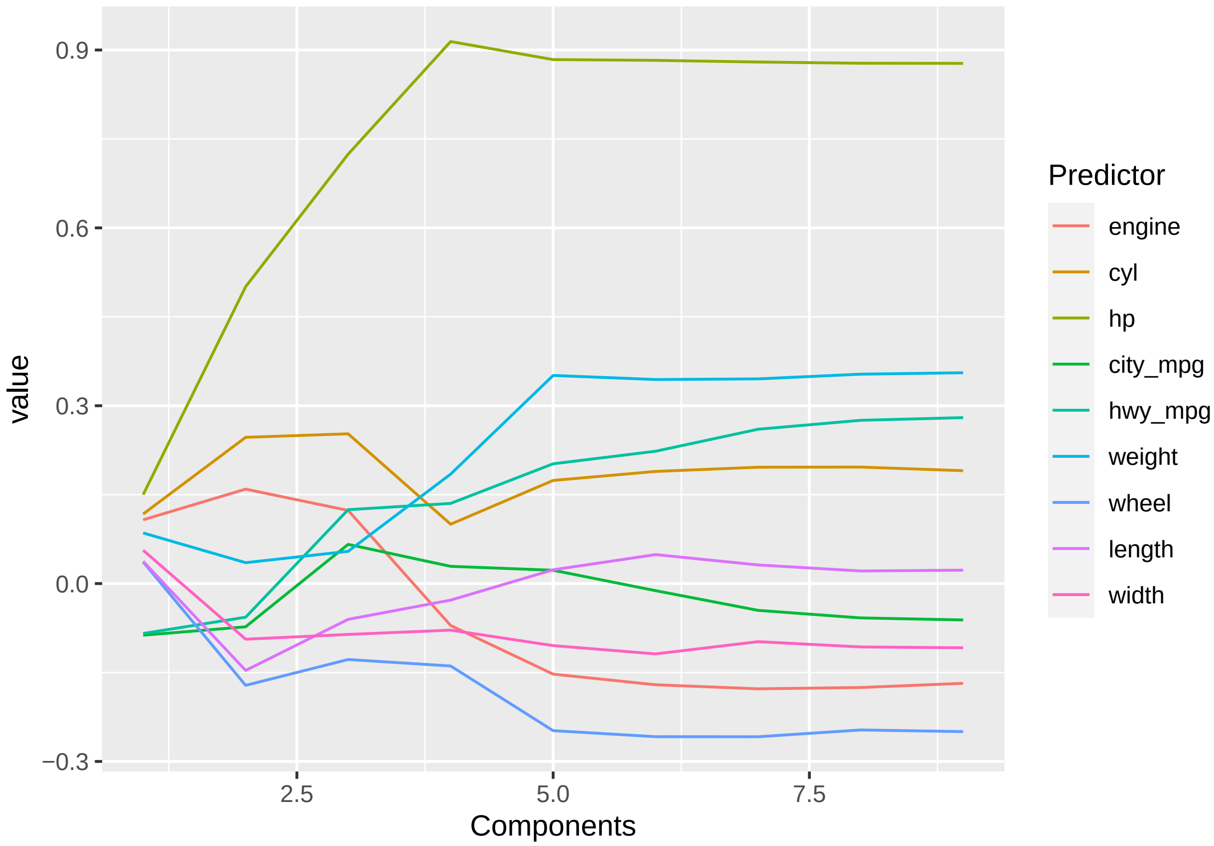
```
for (k in 1:9){

  X_0 <- X
  y_0 <- y
  Z <- NULL
  W <- NULL
  V <-  NULL
  b <- NULL

  for (i in 1:k){

    w_h <- as.matrix((t(X_0) %*% y_0) / as.numeric((t(y_0) %*% y_0)))
    w_h <- as.matrix(w_h / (sqrt(sum(w_h^2))))
    z_h <- as.matrix((X_0 %*% w_h) / as.numeric((t(w_h) %*% w_h)))
    v_h <- (t(X_0) %*% z_h) / as.numeric((t(z_h) %*% z_h))
    X_0 <- X_0 - (z_h %*% t(v_h))
    b_h <- as.numeric((t(y_0) %*% z_h) / as.numeric((t(z_h) %*% z_h)))
    y_0 <- y_0 - (b_h * z_h)

    Z <- cbind(Z, z_h)
    W <- cbind(W, w_h)
    V <- cbind(V, v_h)
    b <- append(b, b_h)

  }

  W_star <- W %*% matrix.inverse(t(V) %*% W)
  b_star <- (W_star) %*% (b)

  final_matrix[1:(length(x_coef)), k] <- b_star

}
```

```r
colnames(final_matrix) <- c('C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9')
rownames(final_matrix) <- c('engine', 'cyl', 'hp', 'city_mpg', 'hwy_mpg', 'weight', 'wheel', 'length',
final_matrix
```

```
##                  C1          C2          C3          C4          C5
## engine    0.10757142  0.15931567  0.12343454 -0.07066327 -0.15270983
## cyl       0.11736837  0.24670786  0.25264591  0.10014959  0.17395619
## hp        0.14995267  0.50082086  0.72429900  0.91428220  0.88385312
## city_mpg -0.08705847 -0.07289907  0.06611491  0.02910301  0.02249980
## hwy_mpg  -0.08419220 -0.05667268  0.12461975  0.13523319  0.20203475
## weight    0.08538581  0.03525618  0.05440547  0.18460903  0.35108233
## wheel     0.03650589 -0.17169924 -0.12806075 -0.13894195 -0.24805230
## length    0.03760384 -0.14648862 -0.06035851 -0.02789218  0.02323934
## width     0.05623287 -0.09382342 -0.08575829 -0.07850738 -0.10463570
##                  C6          C7          C8          C9
## engine   -0.17070122 -0.17749965 -0.17534118 -0.16822457
## cyl       0.18918559  0.19629578  0.19651153  0.19044147
## hp        0.88254428  0.87974169  0.87770127  0.87744314
## city_mpg -0.01196936 -0.04514402 -0.05789964 -0.06133804
## hwy_mpg   0.22324132  0.26034437  0.27540225  0.27993358
## weight    0.34417455  0.34526636  0.35322273  0.35561173
## wheel    -0.25821063 -0.25836657 -0.24686609 -0.24968842
## length    0.04887873  0.03142671  0.02127910  0.02260203
## width    -0.11860960 -0.09796353 -0.10685958 -0.10830459
```

7.2) Path graph of coefficients

```r
final_df <- as.data.frame(t(final_matrix))
final_df$Components <- 1:9

melted_df <- melt(final_df ,  id.vars = 'Components', variable.name = 'Predictor')
ggplot(melted_df, aes(Components,value)) + geom_line(aes(colour = Predictor))
```



**************************** COMMENT ON PATTERN ****************************

It is notable to see that the trends in PLSR associated with the coefficients of the original predictor variables are **almost identical** to those from PCR. This is expected since the basic premise and principles on which these regression methods are based on are the same - error minimisation and OLS.