Lab 6: Regression with Dimension Reduction Methods PCR and PLSR

Prof. Gaston Sanchez

Stat 154, Fall 2017

```
read_chunk('lab06-pcr-pls-regression-chunks.R')
```

Introduction

In this lab, you are going to write R code to implement Principal Component Regression (PCR), as well as Partial Least Squares Regression (PLSR). You will also be using the data `Hitters` from the package `"ISLR"`. More specifically, you will sue `Salary` as the response variable, and the rest of the variables in `Hitters` as the predictors.

Data `Hitters`

The data set `Hitters` is part of the R package `"ISLR"`.

```
data(Hitters)
```

```
str(Hitters, vec.len = 1)
```

```
## 'data.frame':    322 obs. of  20 variables:
##  $ AtBat    : int  293 315 ...
##  $ Hits     : int  66 81 ...
##  $ HmRun    : int  1 7 ...
##  $ Runs     : int  30 24 ...
##  $ RBI      : int  29 38 ...
##  $ Walks    : int  14 39 ...
##  $ Years    : int  1 14 ...
##  $ CAtBat   : int  293 3449 ...
##  $ CHits    : int  66 835 ...
##  $ CHmRun   : int  1 69 ...
##  $ CRuns    : int  30 321 ...
##  $ CRBI     : int  29 414 ...
##  $ CWalks   : int  14 375 ...
##  $ League   : Factor w/ 2 levels "A","N": 1 2 ...
##  $ Division : Factor w/ 2 levels "E","W": 1 2 ...
##  $ PutOuts  : int  446 632 ...
##  $ Assists  : int  33 43 ...
##  $ Errors   : int  20 10 ...
##  $ Salary   : num  NA 475 ...
##  $ NewLeague: Factor w/ 2 levels "A","N": 1 2 ...
```

Principal Components Regression (PCR)

Principal Components Regression can be performed with the function `pcr()` which is part of the package `"pls"`. The code below computes PCR for the regression of `Salary` on the rest of 19 predictors.

```r
# principal component regression
pcr_fit <- pcr(Salary ~ ., data = Hitters, scale = TRUE, validation = "none")
names(pcr_fit)
```

```
##  [1] "coefficients"  "scores"         "loadings"       "Yloadings"
##  [5] "projection"    "Xmeans"         "Ymeans"         "fitted.values"
##  [9] "residuals"     "Xvar"           "Xtotvar"        "fit.time"
## [13] "na.action"     "ncomp"          "method"         "scale"
## [17] "call"          "terms"          "model"
```

1) Start with PCA

You are going write R code in order to replicate the results of `pcr()`. Follow the list of steps shown below:

- Remove observations from `Hitters` that have missing values in `Salary`

```r
# remove missing values
hitters <- na.omit(Hitters)
```

- Use `model.matrix()` to create a design matrix based on the formula `"Salary ~ ."`

```r
# model matrix
MM <- model.matrix(Salary ~ ., data = hitters)
```

- Note that the generated model matrix includes a constant column for the intercept term. Do not use this column.
- The model matrix (without constant column) will be the matrix of responses. Standardize the model matrix of responses; this will be $\mathbf{X}$

```r
# exclude 1st column of model matrix
X <- scale(MM[ ,-1])
```

- The variable `Salary` will be the response $\mathbf{y}$

```
y <- hitters$Salary
```

- Use `svd()` to get the Singular Value Decomposition of $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$

```
# SVD on X (without intercept term)
SVD <- svd(X)
U <- SVD$u
V <- SVD$v
D <- diag(SVD$d)
```

- Compute principal components $\mathbf{Z}$ from the standardized model matrix $\mathbf{X}$ and the eogenvectors in $\mathbf{V}$

$$\mathbf{Z} = \mathbf{X}\mathbf{V}$$

```
# PCs
Z <- U %*% D
```

- Confirm that your principal components match those of `pcr_fit$scores`

```
# compare Z vs pcr-scores
head(cbind(pcr_fit$scores[,1], Z[,1]))
```

```
##                            [,1]         [,2]
## -Alan Ashby       -0.009630358 -0.009630358
## -Alvin Davis       0.410650757  0.410650757
## -Andre Dawson      3.460224766  3.460224766
## -Andres Galarraga -2.553449083 -2.553449083
## -Alfredo Griffin   1.025746581  1.025746581
## -Al Newman        -3.973081710 -3.973081710
```

```
head(cbind(pcr_fit$scores[,19], Z[,19]))
```

```
##                            [,1]         [,2]
## -Alan Ashby       -0.019213448 -0.019213448
## -Alvin Davis      -0.003388059 -0.003388059
## -Andre Dawson     -0.066374190 -0.066374190
## -Andres Galarraga -0.013555820 -0.013555820
## -Alfredo Griffin  -0.070900042 -0.070900042
## -Al Newman         0.025016886  0.025016886
```

2) PC Regression on the first component

- Use the first PC$\mathbf{z_1}$ to compute the regression of $\mathbf{y}$ on $\mathbf{z_1}$. That is, obtain the first PCR coefficient b_1 given by:

$$b_1 = (\mathbf{z_1}^\mathsf{T}\mathbf{z_1})^{-1}\mathbf{z_1}^\mathsf{T}\mathbf{y}$$

```
# regression with first component
pcreg_1 <- lm(y ~ Z[,1])
b1_pcr <- pcreg_1$coefficients
```

- Compute the vector of predicted values $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = b_1\mathbf{z_1}$$

- Compare your computed $\hat{\mathbf{y}}$ against `pcr_fit$fitted.values[ , ,1]`, which is the fitted response using PC1 provided by `pcr()`. Add the average of y to your predicted value before comparison.

```
# compare y-hat with pcr() output
head(cbind(pcreg_1$fitted.values, pcr_fit$fitted.values[,,1]))
```

```
##       [,1]     [,2]
## 1 534.8996 534.8996
## 2 579.6895 579.6895
## 3 904.6869 904.6869
## 4 263.8013 263.8013
## 5 645.2411 645.2411
## 6 112.5090 112.5090
```

3) PC Regression on all PCs

- Compute the vector of PCR-coefficients $\mathbf{b}_{pcr}$ by regressing $\mathbf{y}$ on all principal components $\mathbf{Z}$:

$$\mathbf{b}_{pcr} = (\mathbf{Z}^\mathsf{T}\mathbf{Z})^{-1}\mathbf{Z}^\mathsf{T}\mathbf{y}$$

- Compute the vector of predicted values $\hat{\mathbf{y}}$ using all PCs:

$$\hat{\mathbf{y}} = \mathbf{Z}(\mathbf{Z}^\mathsf{T}\mathbf{Z})^{-1}\mathbf{Z}^\mathsf{T}\mathbf{y}$$
$$\hat{\mathbf{y}} = \mathbf{Z}\mathbf{b}_{pcr}$$

- Compare your computed $\hat{\mathbf{y}}$ against `pcr_fit$fitted.values[ , ,19]` and confirm that you have the same results as `pcr()`. Add the average of y to your predicted value before comparison.

```
# comapre your coeffs vs those provided by pcr()
head(cbind(pcreg_all$fitted.values, pcr_fit$fitted.values[,,19]))
```

```
##          [,1]      [,2]
## 1  362.1361  362.1361
## 2  712.6952  712.6952
## 3 1171.3111 1171.3111
## 4  556.7875  556.7875
## 5  493.2515  493.2515
## 6  247.3852  247.3852
```

4) PCR coefficients in terms of the predictor variables

`pcr()` returns regression coefficients—in terms of the predictors—for all possible regressions: with one PC, two PCs, three PCs, and so on, until the regression that uses all 19 PCs.

Consider the PC regression on the first PC $\mathbf{z_1}$. The PCR-coefficient is:

$$b_1 = (\mathbf{z_1}^\mathsf{T}\mathbf{z_1})^{-1}\mathbf{z_1}^\mathsf{T}\mathbf{y}$$

and the fitted $\hat{\mathbf{y}}$ is:

$$\hat{\mathbf{y}} = b_1\mathbf{z_1}$$

You can re-write the regression of PC1 in terms of the response variables as:

$$\begin{aligned}\hat{\mathbf{y}} &= b_1\mathbf{z_1} \\ &= b_1\mathbf{X}\mathbf{v_1} \\ &= \mathbf{X}(b_1\mathbf{v_1}) \\ &= \mathbf{X}\mathbf{b_1^*}\end{aligned}$$

where:

- $\mathbf{v_1}$ is the loading associated to the first PC, that is, the first column of $\mathbf{V}$
- $\mathbf{b_1^*}$ is a vector of regression coefficients in terms of the predictors

In general, the PC regression coefficients can be expressed in terms of the predictors as:

$$\mathbf{b_k^*} = \mathbf{V_k}\mathbf{D_k^{-1}}\mathbf{U_k^\mathsf{T}}\mathbf{y}$$

where the index k indicates matrices associated to the first k components. More specifically, V_k is a matrix of the first k columns of V, U_k is a matrix of the first k columns of U, and D_k is a $k \times k$ diagonal matrix.

Your turn:

- Take your previously computed coefficient b_1 and calculate the associated vector of coefficients $\mathbf{b_1^*} = b_1 \mathbf{v_1}$. Confirm that your vector $\mathbf{b_1^*}$ matches that of `pcr_fit$coefficients[ , , 1]`

```
# PCR coeffs with PC1, in terms of predictors
pcreg_1$coefficients[-1] * V[,1]
```

```
##  [1] 21.13207878 20.87321071 21.77988064 21.13705999 25.06279956
##  [6] 22.26529508 30.11445915 35.21789413 35.24760132 33.99408860
## [11] 36.04328244 36.27081015 33.76212997 -5.80503669 -2.74157997
## [16]  8.28029613 -0.08969488 -0.83758395 -4.46643991
```

```
# compare with output from pcr()
pcr_fit$coefficients[,,1]
```

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks
## 21.13207878 20.87321071 21.77988064 21.13705999 25.06279956 22.26529508
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI
## 30.11445915 35.21789413 35.24760132 33.99408860 36.04328244 36.27081015
##      CWalks     LeagueN   DivisionW     PutOuts     Assists      Errors
## 33.76212997 -5.80503669 -2.74157997  8.28029613 -0.08969488 -0.83758395
##   NewLeagueN
## -4.46643991
```

- Do the same for all possible sets of PCs, and verify your coefficients against the output of `pcr_fit$coefficients`.

```
coeffs <- matrix(0, nrow = ncol(X), ncol = ncol(Z))
for (k in 1:ncol(Z)) {
  pcreg_k <- lm(y ~ Z[ ,1:k])
  if (k == 1) {
    coeffs[ ,k] <- pcreg_k$coefficients[-1] * V[,1]
  } else {
    coeffs[ ,k] <- V[,1:k] %*% pcreg_k$coefficients[-1]
  }
}
rownames(coeffs) <- colnames(X)
colnames(coeffs) <- paste0('Z', 1:ncol(Z))

# compare agains pcr()
cbind(coeffs[ ,1:3], pcr_fit$coefficients[,,1:3])
```

```
##                     Z1        Z2        Z3     1 comps   2 comps   3 comps
## AtBat      21.13207878 29.438966 31.596172 21.13207878 29.438966 31.596172
## Hits       20.87321071 29.039128 30.841116 20.87321071 29.039128 30.841116
## HmRun      21.77988064 26.912608 21.650526 21.77988064 26.912608 21.650526
## Runs       21.13705999 29.312723 28.894882 21.13705999 29.312723 28.894882
## RBI        25.06279956 31.870731 30.091792 25.06279956 31.870731 30.091792
## Walks      22.26529508 27.235049 28.345853 22.26529508 27.235049 28.345853
## Years      30.11445915 24.434849 25.276570 30.11445915 24.434849 25.276570
## CAtBat     35.21789413 31.042550 33.076799 35.21789413 31.042550 33.076799
## CHits      35.24760132 31.288812 33.388213 35.24760132 31.288812 33.388213
## CHmRun     33.99408860 31.260422 29.160504 33.99408860 31.260422 29.160504
## CRuns      36.04328244 32.314419 33.604363 36.04328244 32.314419 33.604363
## CRBI       36.27081015 32.632509 32.997441 36.27081015 32.632509 32.997441
## CWalks     33.76212997 29.599532 30.624769 33.76212997 29.599532 30.624769
## LeagueN    -5.80503669 -7.865898  5.466047 -5.80503669 -7.865898  5.466047
## DivisionW  -2.74157997 -3.535498 -3.929845 -2.74157997 -3.535498 -3.929845
## PutOuts     8.28029613 11.651168 12.899211  8.28029613 11.651168 12.899211
## Assists    -0.08969488  3.560723 13.245133 -0.08969488  3.560723 13.245133
## Errors     -0.83758395  3.507803 12.826601 -0.83758395  3.507803 12.826601
## NewLeagueN -4.46643991 -6.145712  7.109718 -4.46643991 -6.145712  7.109718
```

The lab continues on the next page.

Partial Least Squares Regression

Below are the steps of the PLSR algorithm (in its "classic" version). Assume that the predictors in $\mathbf{X}$ and the response $\mathbf{y}$ are standardized: mean $= 0$, variance 1.

```
Set $\mathbf{X_0} = \mathbf{X}$ and $\mathbf{y_0} = \mathbf{y}$
for $h = 1, 2, \dots, r$ do
    $\mathbf{w_h} = \mathbf{X_{h-1}^{\mathsf{T}} y_{h-1}}$
    normalize weights: $\|\mathbf{w_h}\| = 1$
    $\mathbf{z_h} = \mathbf{X_{h-1} w_h / w_h^{\mathsf{T}} w_h}$
    $\mathbf{p_h} = \mathbf{X_{h-1}^{\mathsf{T}} z_h / z_h^{\mathsf{T}} z_h}$
    $\mathbf{X_h} = \mathbf{X_{h-1}} - \mathbf{z_h p_h^{\mathsf{T}}}$
    $b_h = \mathbf{y_{h-1}^{\mathsf{T}} z_h / z_h^{\mathsf{T}} z_h}$
    $\mathbf{y_h} = \mathbf{y_{h-1}} - b_h \mathbf{z_h}$
end for
```

where r is the rank of $\mathbf{X}$

Your mission is to write R code that carries out PLS regression according to the steps shown above. Your code should contain the following objects:

- `components`: matrix of PLS components $\mathbf{Z}$
- `weights`: matrix of weights $\mathbf{W}$
- `loadings`: matrix of loadings $\mathbf{P}$
- `coefficients`: vector of regression coefficients $\mathbf{b}$
- `fitted`: matrix of fitted (predicted) values $\hat{\mathbf{Y}}$

The first steps are the same as with PCR:

- Remove observations from `Hitters` that have missing values in `Salary`
- Use `model.matrix()` to create a design matrix based on the formula `"Salary ~ ."`
- Note that the generated model matrix includes a constant column for the intercept term. Do not use this column.
- The model matrix (without constant column) will be the matrix of responses.
- Standardize the model matrix of responses; this will be $\mathbf{X}$
- The response `Salary` will be $\mathbf{y}$

Check your first PLS component

- Calculate $\mathbf{w_1}, \mathbf{z_1}$, and $\mathbf{p_1}$
- Compare your results with `pls_fit$loading.weights[,1]`, `pls_fit$scores[,1]`, `pls_fit$loadings[,1]`,
- Compare the first fitted $\hat{\mathbf{y}}$, i.e. regressing $\mathbf{y}$ on the first PLS component $\mathbf{z_1}$, and compare it with `pls_fit$fitted.values[␣,1]`

```r
pls_fit <- plsr(Salary ~ ., data = Hitters, scale = TRUE, validation = "none")

pls_regression <- function(X, y) {
  # Assume X is standardized.

  n <- nrow(X)
  p <- ncol(X)
  r <- qr(X)$rank
  Z <- matrix(0, nrow=n, ncol=r)
  W <- matrix(NA, nrow=p, ncol=r)
  P <- matrix(NA, nrow=p, ncol=r)
  b <- rep(0, r)
  Yhat <- matrix(mean(y), nrow=n, ncol=r)

  for (h in 1:r) {
    W[, h] <- t(X) %*% y
    W[, h] <- W[, h] / sqrt(sum(W[, h]^2))
    Z[, h] <- X %*% W[, h]
    P[, h] <- t(X) %*% Z[, h] / sum(Z[, h]^2)
    X <- X - Z[, h] %*% t(P[, h])
    b[h] <- sum(y * Z[, h]) / sum(Z[, h]^2)
    y <- y - b[h] * Z[, h]
    Yhat[, h] <- Yhat [, h] + Z[, 1:h, drop=FALSE] %*% b[1:h]
  }

  list(components=Z, weights=W, loadings=P,
       coefficients=b, fitted=Yhat)
}

my_pls <- pls_regression(X, y)

# Check if the loading weights match.
sapply(1:qr(X)$rank, function(j) {
  all(abs(my_pls$weights[, j] - pls_fit$loading.weights[, j]) < 1e-6)
})
```

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

```r
head(cbind(my_pls$weights[, 1], pls_fit$loading.weights[, 1]))
```

```
##            [,1]      [,2]
## AtBat 0.2256137 0.2256137
## Hits  0.2507049 0.2507049
## HmRun 0.1960424 0.1960424
## Runs  0.2399514 0.2399514
## RBI   0.2568671 0.2568671
## Walks 0.2536725 0.2536725
```

```
# Check if the scores match.
sapply(1:qr(X)$rank, function(j) {
  all(abs(my_pls$components[, j] - pls_fit$scores[, j]) < 1e-6)
})
```

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

```
head(cbind(my_pls$components[, 1], pls_fit$scores[, 1]))
```

```
##                         [,1]       [,2]
## -Alan Ashby       -0.1090169 -0.1090169
## -Alvin Davis       0.6670947  0.6670947
## -Andre Dawson      3.4717021  3.4717021
## -Andres Galarraga -2.1298594 -2.1298594
## -Alfredo Griffin   0.9770842  0.9770842
## -Al Newman        -4.0036686 -4.0036686
```

```
# Check if the loadings match.
sapply(1:qr(X)$rank, function(j) {
  all(abs(my_pls$loadings[, j] - pls_fit$loadings[, j]) < 1e-6)
})
```

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

```
head(cbind(my_pls$loadings[, 1], pls_fit$loadings[, 1]))
```

```
##            [,1]      [,2]
## AtBat 0.2256185 0.2256185
## Hits  0.2231972 0.2231972
## HmRun 0.2179161 0.2179161
## Runs  0.2249696 0.2249696
## RBI   0.2566359 0.2566359
## Walks 0.2292001 0.2292001
```

```
# Check if the fitted values match.
sapply(1:qr(X)$rank, function(j) {
  all(abs(my_pls$fitted[, j] - as.vector(pls_fit$fitted.values[,,j])) < 1e-6)
})
```

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

```r
head(cbind(my_pls$fitted[, 1], as.vector(pls_fit$fitted.values[,,1])))
```

```
##           [,1]      [,2]
## [1,] 523.82552 523.82552
## [2,] 609.97025 609.97025
## [3,] 921.26845 921.26845
## [4,] 299.52153 299.52153
## [5,] 644.37762 644.37762
## [6,]  91.53754  91.53754
```