Homework3

Azadeh Glanpour

September 17, 2018

Problem 1: Mathematics of principal components

(a) Create a correlation matrix for "mtcars" data set and save it in "corMat"

```
data("mtcars")
corMat <- cor(mtcars)
```

(b) Compute the Eigenvalues and Eigenvector of "corMat"

```
cars.eigen<- eigen(corMat)
```

(c) Compute Principal Comonent Analysis

```
cars.PC<- prcomp(mtcars, scale. = T)
```

(d)

```
round(sum(abs(cars.PC$rotation) - abs(cars.eigen$vectors)), 2)
```

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

The results show that the rotation matrix of principal component is the eigen vecotor of the same data(Their result is zero).Correlation-based and covariance-based PCA will produce the exact same results (apart from a scalar multiplier) when the individual variances for each variable are all exactly equal to each other. When these individual variances are similar but not the same, both methods will produce similar results.

(e) Checks orthogonality

```
round(cars.PC$rotation[,1] %*% cars.PC$rotation[,2],2)
```

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

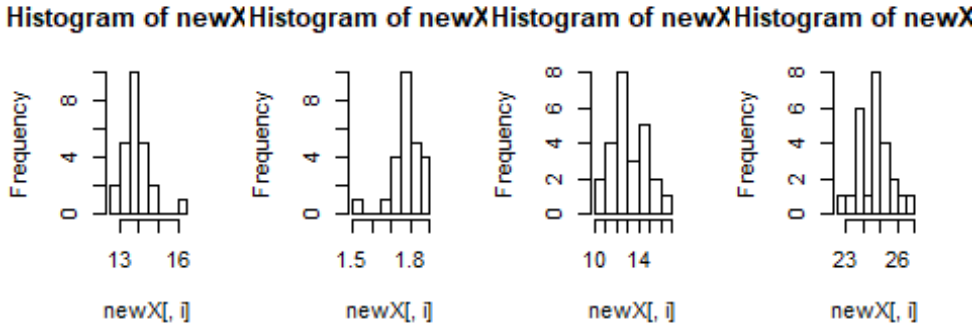
The results show that the PC1 and PC2 are orthogonal

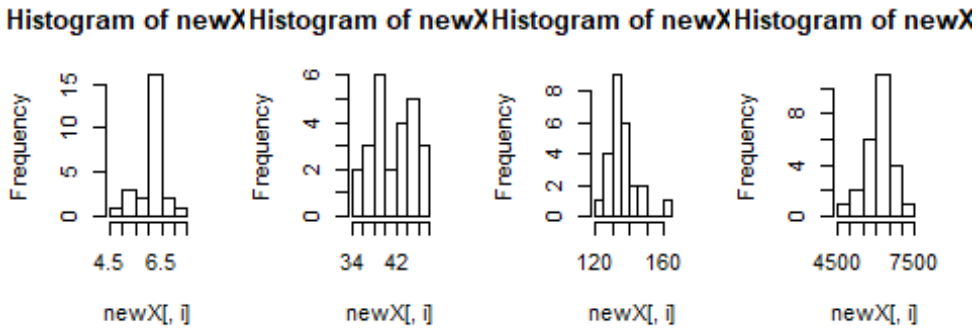
Problem 2: Principal Components for dimension reduction

```
library(HSAUR2)
```

```
## Loading required package: tools
```

```r
data("heptathlon")
par(mfrow=c(2,4))                # Divides the screen into four sections
temp <- apply(heptathlon[,1:8], 2, hist)  # Draws histograms
```






```r
par(mfrow=c(1,1))                # Resets the screen in normal
```

Since the number of records in this example is not much, it is difficult to say if the distribution is normal. But I guess the distribution is normal.

(b)

```r
library(outliers)
temp<- apply(heptathlon[,1:8],2,grubbs.test)
temp <- apply(heptathlon[,1:8], 2, outlier)
heptathlon[heptathlon[,1:8] == outlier(heptathlon[,1:8]),]
```

```
##             hurdles highjump  shot run200m longjump javelin run800m score
## Launa (PNG)   16.42      1.5 11.78   26.16     4.88   46.38  163.43  4566
## NA               NA       NA    NA      NA       NA      NA      NA    NA
## NA.1             NA       NA    NA      NA       NA      NA      NA    NA
## NA.2             NA       NA    NA      NA       NA      NA      NA    NA
## NA.3             NA       NA    NA      NA       NA      NA      NA    NA
## NA.4             NA       NA    NA      NA       NA      NA      NA    NA
```

```r
heptathlon["Launa (PNG)",]
```

```
##             hurdles highjump  shot run200m longjump javelin run800m score
## Launa (PNG)   16.42      1.5 11.78   26.16     4.88   46.38  163.43  4566
```

```
heptathlon <- heptathlon[heptathlon$score != outlier(heptathlon$score),]
```

By using the Grubb's test and looking into the results of it we can undrestand that that **Launa (PNG)** definitly is the outliener. She is an ouliner at "*hurdles*", "*highjump*", "*run800m*", and "*score*". We can remove her record to have a cleaner data set.

(c)

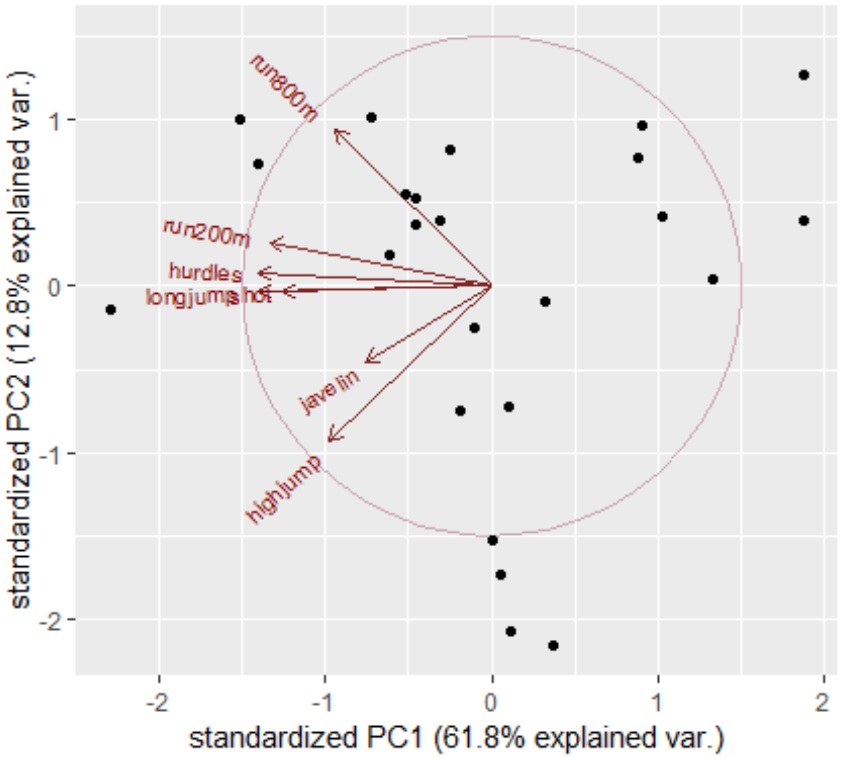
```
heptathlon$hurdles <- max(heptathlon$hurdles) - heptathlon$hurdles
heptathlon$run200m <- max(heptathlon$run200m) - heptathlon$run200m
heptathlon$run800m <- max(heptathlon$run800m) - heptathlon$run800m
```

(d)

```
Hpca <- prcomp(heptathlon[,1:7], scale. = T)
```

(e)

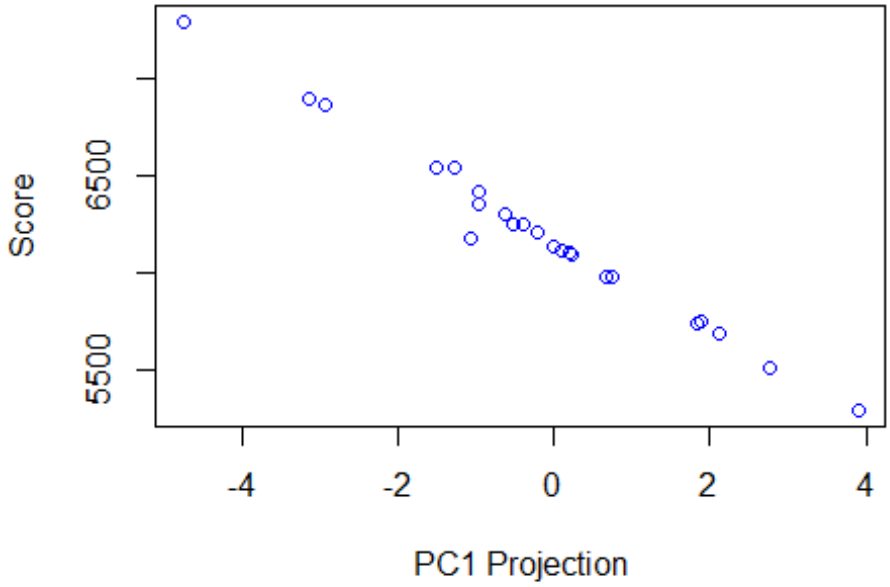
```
#install.packages("devtools")
library(devtools)
#install_github( "vqv/ggbiplot",force=T)
library(ggplot2)
library(plyr)
library(scales)
library(grid)
library(ggbiplot)
ggbiplot(Hpca,circle = T)
```



The biplot shows that (run200,hurdles,long jump,shot) are highly correlated to pc1 in negative way ,javelin is also negatively correalted to pc1. While run800 and highjump have marjnal behaviare, we could also say highjump is negatively related to pc1 and run800 is positively related to pc2.

(f)

```
plot(x=Hpca$x[,1],y=heptathlon$score,
     xlab = "PC1 Projection",
     ylab = "Score",
     col="blue")
```



The figure shows that PC1 is a very desriptive variable and we can condense all the original attributes into one variable and still keep the variance for analyzing purposes. The PC1 variable which is the reduced dimension of all the original variables, has a negative relationship with the score variable. In the other words, we can reduce 7 dimensions into a single dimension without loosing much information. Since the original variables are correlated, as could be expected, it is possible to condense all of them into a simple variable. #Problem 3: Face Recognition-well,sort of

```
classdigits <- read.csv("classDigits.csv")
class7test <- read.csv("class7Test.csv")

# Copies the pixel data in a matrix
digits.data = as.matrix(classdigits[,2:785])

# Runs PC Analusyis and stores the results
```

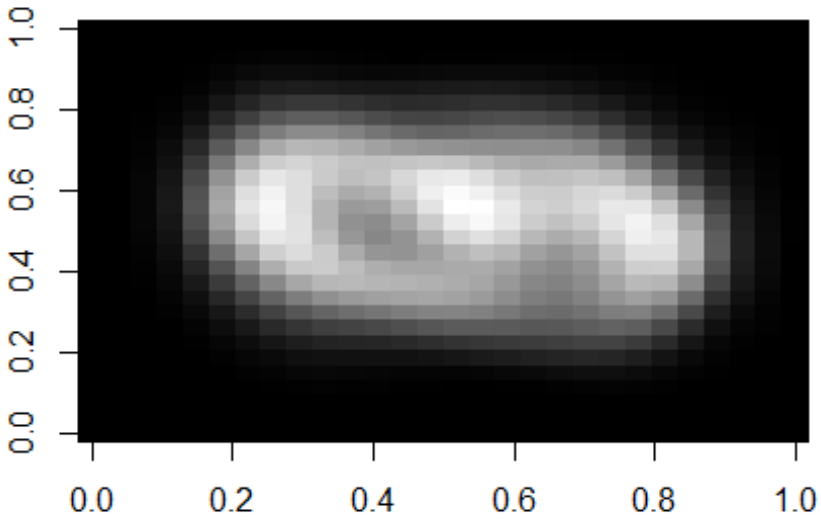
```
digits.PCA = prcomp(digits.data)

# Stores the eigen vectors in a separated variable
digits.Eigen.Vector = as.matrix(digits.PCA$rotation)

# Shows a small part of the eigen vectors
digits.Eigen.Vector[1:4, 1:4]
```

```
##                   PC1           PC2           PC3           PC4
## pixel0  2.219274e-20 -5.732181e-19 6.287447e-20 -1.759315e-19
## pixel1  2.081668e-17  1.110223e-16 2.081668e-17  8.326673e-17
## pixel2 -1.942890e-16  0.000000e+00 4.857226e-17 -4.163336e-17
## pixel3 -1.387779e-16  1.110223e-16 4.336809e-17 -1.110223e-16
```

(b)

The variable $center is mean value for the data set. After Reshaping the mean data into a 28x28 matrix results the Figure 1. It is saved as file named meanDigits.jpeg.

```
digit.mean <- as.matrix(digits.PCA$center)
digitMatrix <- matrix(digit.mean,ncol=28,nrow = 28,byrow=T)
image(digitMatrix,
      col = grey(seq(0, 1, length = 256)))
```




```
library(jpeg)
writeJPEG(digitMatrix,
          target = "meanDigit.jpg")
```

(c)

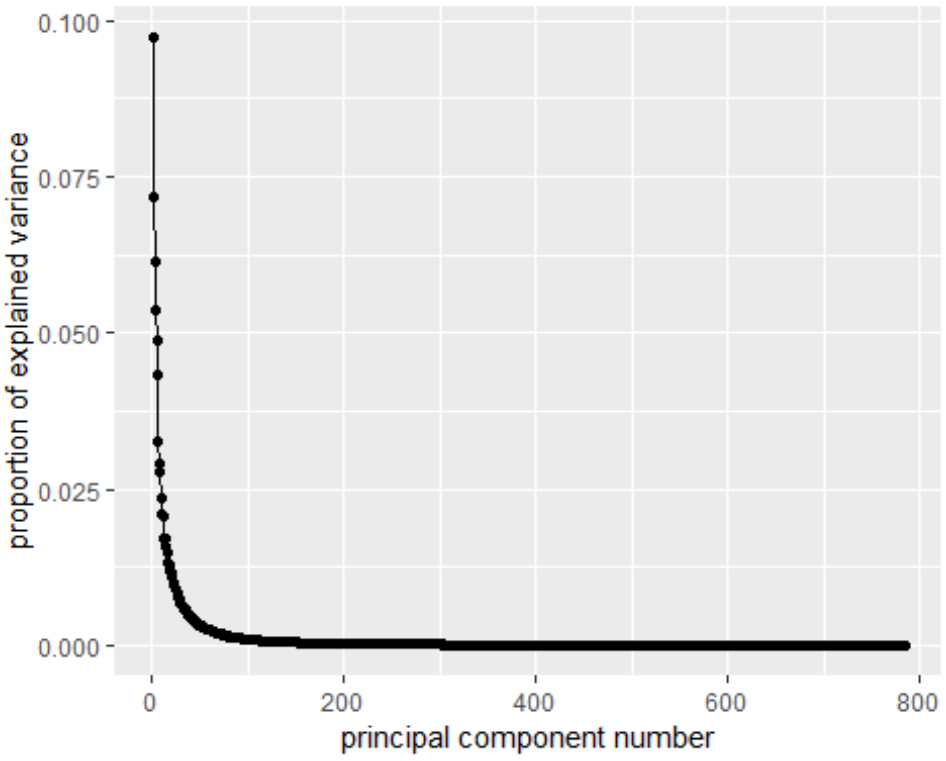
A loop has been constructed to reconstruct images #15 and #100 with lower dimensions and saves them in the separate files. The following R code was used for this part.

```
X.mean = t(digit.mean)
for (img in c(15, 100)){                              # Outer loop for images
  for (k in c(5, 20, 100)){                           # Inner loop for dimensions
    X = digits.data[img,]                             # Pick the image data
    E = digits.Eigen.Vector[,1:k]                     # Pick the eigen vectors
    weight = digits.PCA$x[img, 1:k]                   # Calulate the weight values
    new.image = X.mean + weight %*% t(E)              # Reconstruct the image
    new.image = matrix(new.image,
             nrow = 28, ncol=28, byrow = T)  # Converts data into a matrix to
show
    image.name =
      do.call("paste0", list("image", img, "-", k, ".jpg")) # Make proper
file name
    jpeg(image.name, width = 2800, height = 2800, res = 600)  # Open a jpeg
output
    image(new.image, col = grey(seq(0, 1, length = 256))) # Write the image
into a jpeg output
    dev.off()                                         # Save the jpeg file
  }
}
```

(d)

There are several ways to choose pricipal components which are the dimension of the target space inder to solve the problem in less complex space. One of them is to plot the PCA and find the elbow. Other way is to consider a cut off number like 90% which is a criterion to keep a certain amount of the variances. Here the graphical way illustrated in Figure 2, but the numerical method was considered to choose proper k. To maintain 90% of the variance, 87 components have been selected.

```
#install.packages("devtools")
library(devtools)
#install_github( "vqv/ggbiplot",force=T)
library(ggplot2)
library(plyr)
library(scales)
library(grid)
library(ggbiplot)
print(ggscreeplot(digits.PCA))
```




```
s = summary(digits.PCA)$importance[3,]
names(s[s>.8][1])

## [1] "PC43"

names(s[s>.85][1])

## [1] "PC59"

names(s[s>.9][1])

## [1] "PC87"

k = 87                              # Choose a k from above (I chose from 90%)
X = as.matrix(class7test[,3:786])                              # Rest Data set
E = digits.Eigen.Vector[,1:k]                                 # Eigen vectors
X.mean = matrix(rep(digit.mean, 7),
                nrow = 7, ncol = 784, byrow = T)      # Mean digit from digit
space
X.diff = X - X.mean                                        # Deviation form mean
digit
test.projection = X.diff %*% E                            # Map test data into new
space
training.projection = digits.PCA$x[ ,1:k]    # Map training data into the new
sapce
projection.corr = cov(training.projection)        # Calculates the covariance

mahala.mean = rep(0,7)                      # Empty matrix
```

```
for(i in 1:7){
  mahala.mean[i] = mean(mahalanobis(
    training.projection, test.projection[i,], projection.corr))
}
mahala.mean                                          # Displays the results

## [1] 145.7205 162.6667 162.6691 136.8206 189.2969 213.8493 509.3984
```

To calculate the Mahalanobis distance for each point in test data set, first, the differennce between the point and mean digit space were calculated. Then the points have been projected into the new space. All the digit of digit space also projected into the new space. The Mahalanobis distances were calculated and the mean value of the distances for each point of the test data set reported.

(e)

In this section a two loops are applied to determine the minumum dimension required to recognize the digit in the test data set. The outer loop counts the image number. It starts from image number 4 to image number 6. The inner loop count the target dimension. It start with k = 1, then finds the Mahalanobis distance of the image to the all images in the digit space. The lable of the closet point is examined if it is same as our test lable, we found the correct answer, but if they are not same the loop runs for next value of the k until find the right answer.

```
for (img in 4:6){
  k = 1
  repeat {
    X = as.matrix(class7test[,3:786])
    E = digits.Eigen.Vector[,1:k]
    X.mean = matrix(rep(t(as.matrix(digit.mean)), 7),
                    nrow = 7, ncol = 784, byrow = T)
    X.diff = X - X.mean
    test.projection = X.diff %*% E
    training.projection = as.matrix(digits.PCA$x[ ,1:k] )
    projection.corr = cov(training.projection)

    a = which.min(mahalanobis(
      training.projection, test.projection[img,], projection.corr))
    predict.lable = classdigits[a, 1]
    test.lable = class7test[img, 2]

    if (predict.lable == test.lable || k > 784) break
    k = k + 1
  }
  print (k)
}
```

```
## [1] 4
## [1] 11
## [1] 2
```