Assignment 3

Group 11

Chang Zhou 01983512, Qian Zhang 01939418, Yutong Zheng 01895402

2021/5/23

Question 1

```
data = readMat("olivettifaces.mat")
faces = data$faces
dim(faces)
```

```
## [1] 4096  400
```

```
NumFaces = length(faces[1,])
NumPixels = length(faces[,1])
```

```
NumFacesPCA = 400;  # Number of images we will use for PCA
set.seed(1)
index = sample(1:NumFaces, NumFacesPCA);
TrainImages = faces[,index]
```

```
library(plotrix)
n = 2
par(mfrow = c(n,n),      # 2x2 layout
    oma = c(0, 0, 0, 0), # controls rows for text at outer  margins
    mar = c(1, 1, 0, 0), # space for one row of text at ticks and to separate plots
    mgp = c(2, 1, 0),    # axis label at 2 rows distance, tick labels at 1 row
    xpd = NA)
for (i in 1:n^2){
  Face = matrix(TrainImages[,i],sqrt(NumPixels),byrow=FALSE)
  color2D.matplot(Face,axes=FALSE)
}
```

```r
pr.out <- prcomp(t(TrainImages), scale = TRUE)
names(pr.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```r
loading <- pr.out$rotation
scale <- pr.out$scale
center <- pr.out$center
x <- pr.out$x
```

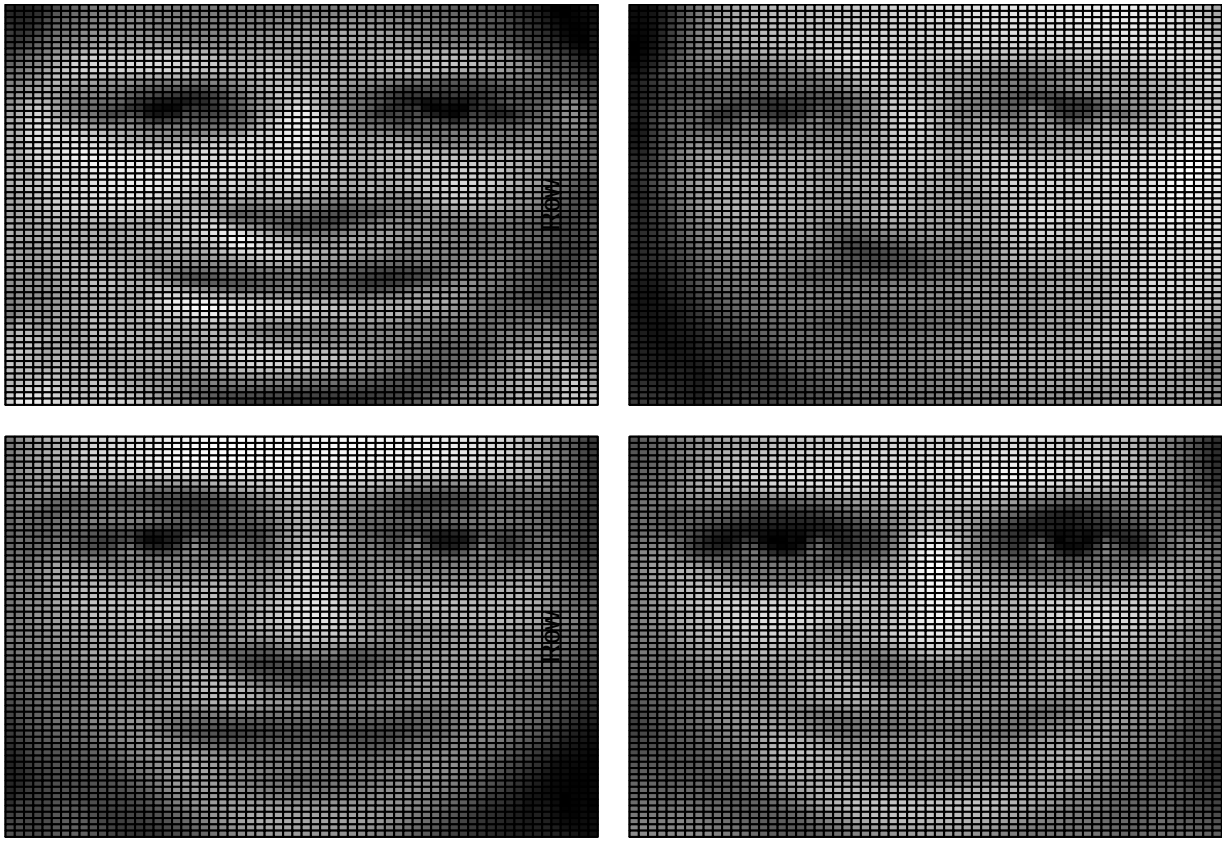
```r
pca_graph <- function(k) {
  cat('For k = ',k)
  ReconstructFace <- center + scale * loading[,1:k] %*% t(x[,1:k])
  dim(ReconstructFace)

  par(mfrow = c(n,n),      # 2x2 layout
      oma = c(0, 0, 0, 0), # controls rows for text at outer  margins
      mar = c(1, 1, 0, 0), # space for one row of text at ticks and to separate plots
      mgp = c(2, 1, 0),    # axis label at 2 rows distance, tick labels at 1 row
      xpd = NA)
  for (i in 1:n^2){
    Face = matrix(ReconstructFace[,i],sqrt(NumPixels),byrow=FALSE)
    color2D.matplot(Face,axes=FALSE)
  }
}
```

```r
pca_graph(3)
```

```
## For k =  3
```

```
pca_graph(10)
```

```
## For k =  10
```



```
pca_graph(25)
```

```
## For k =  25
```

```
pca_graph(50)
```

```
## For k =  50
```

Question 2

a.

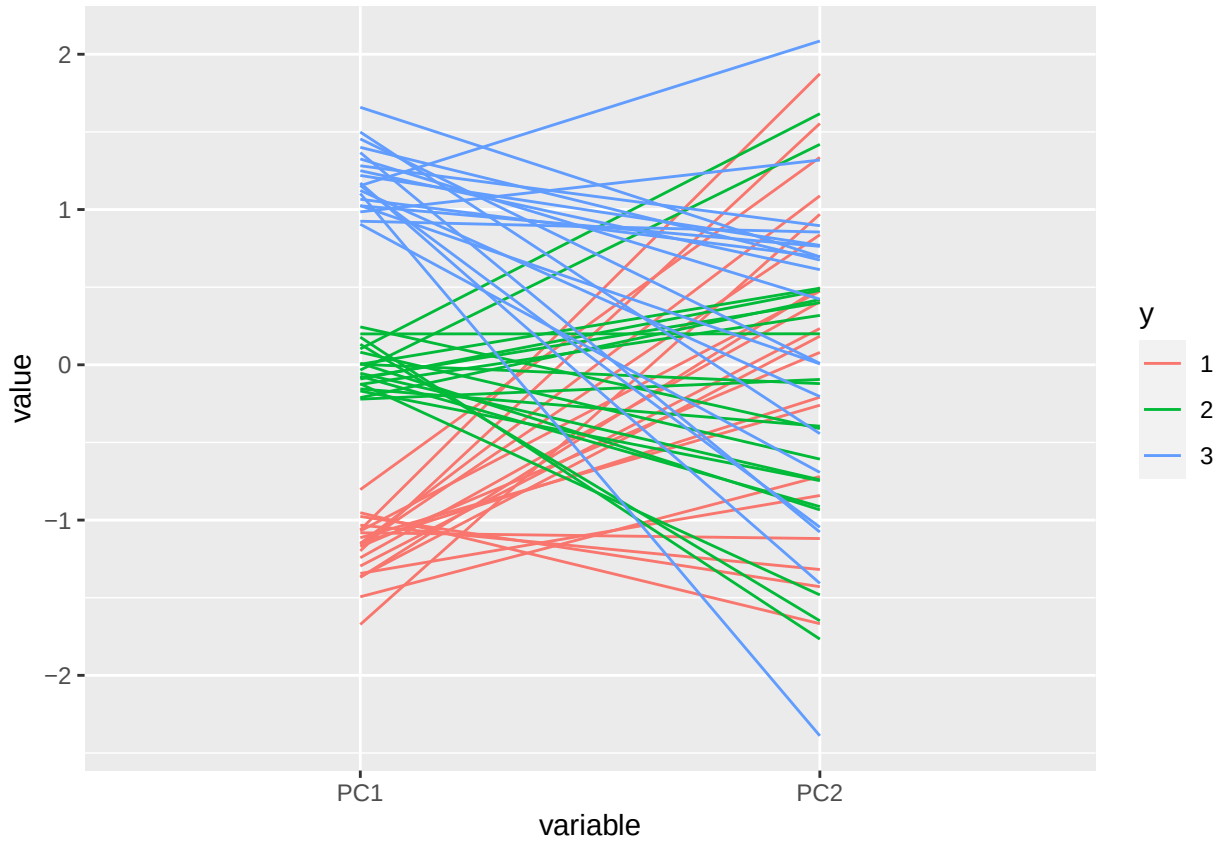
```
y <- c(rep(1, 20),rep(2, 20),rep(3, 20))
#1-20 were divided into group 1, 21-40 into group 2, and the rest into group 3
set.seed(1)
x <- rbind(matrix(rnorm(20*50, mean = 0), nrow = 20),
           matrix(rnorm(20*50, mean = 1), nrow = 20),
           matrix(rnorm(20*50, mean = 2), nrow = 20))
# Three groups of data were generated, with the average of 0, 1 and 2
all <- cbind(y,x)
```

b.

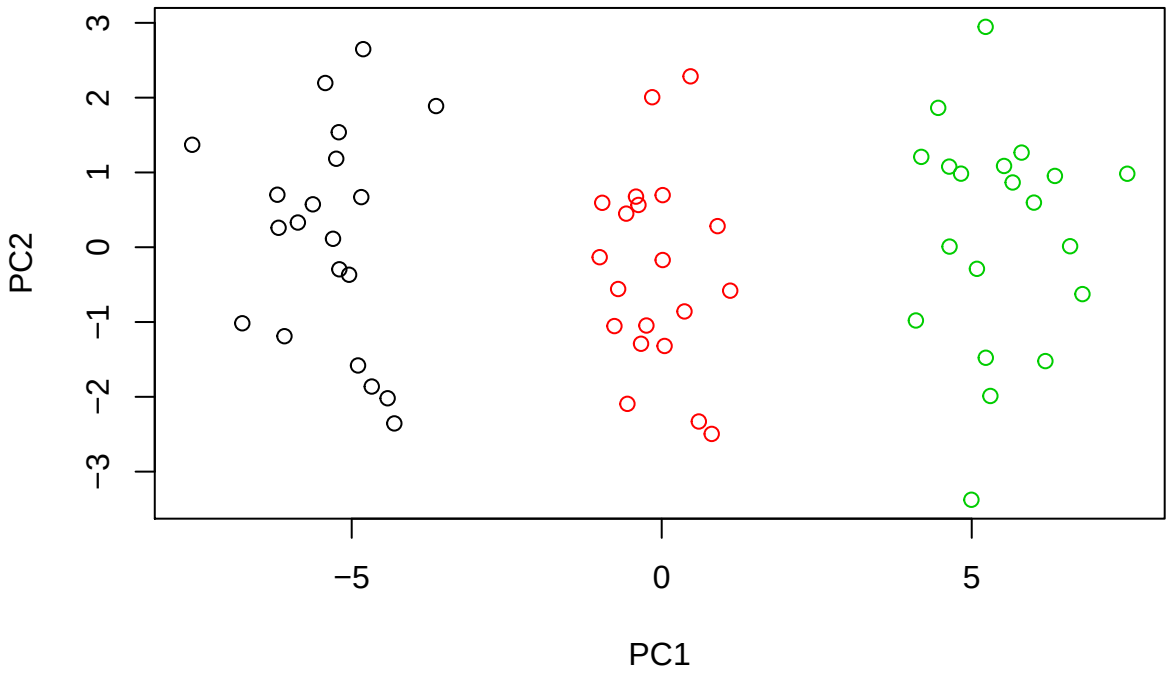
```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
pca <- prcomp(x, scale=TRUE)
pc_label <- data.frame(pca$x, y=as.factor(y))
# parallel coordinates
ggparcoord(data=pc_label, columns=1:2, groupColumn=ncol(pc_label))
```




```
# scatter plot
plot(pc_label[1:2],col=y)
```



c.

```
res = kmeans(x, centers = 3)
table (res$cluster,y)
```

```
##    y
##      1  2  3
##   1  0 20  0
##   2  0  0 20
##   3 20  0  0
```

The results of K-means clustering are perfectly matched with original class labels. $\rho(\mathbf{Z}) = \sum_{i \neq j}(\mathbf{Y}_{ij}(1 - \mathbf{Z}_{ij}) + (1 - \mathbf{Y}_{ij})\mathbf{Z}_{ij}) = 0$.

d.

```
res2 = kmeans(x, centers = 2)
table (res2$cluster,y)
```

```
##    y
##      1  2  3
##   1  0 20 20
##   2 20  0  0
```

The group with the variables' values in the middle (mean=1) will be clustered into one of the other two groups.

e.

```
res3 = kmeans(x, centers = 4)
table (res3$cluster,y)
```

```
##    y
##      1  2  3
##   1 20  0  0
##   2  0  0 12
##   3  0 20  0
##   4  0  0  8
```

One of the three groups will be divided into two clusters.

f.

```
res4 = kmeans(pc_label[1:2], centers = 3)
table (res4$cluster,y)
```

```
##    y
##      1  2  3
##   1  0  0 20
##   2  0 20  0
##   3 20  0  0
```

Same as (c), the clustering is perfect, which conforms to our intuition since in (b), we've seen that data points in 3 classes can be separated clearly using only the first two principal component score vectors.

g.

```
res5 = kmeans(scale(x), centers = 3)
table (res5$cluster,y)
```

```
##    y
##      1  2  3
##   1  0  0  9
##   2 20 18  0
##   3  0  2 11
```

Same as (c), the clustering is perfect. We can observe from (b) that 2 principal component score vectors obtained from the scaled X can separate 3 classes perfectly, therefore scaling each variable to perform K-means will not differ a lot.

Question3

(a)

```
df <- read_csv("Efron94_MissingData-Bootstrap.csv")

# Replace "?" with NaN and convert columns into dbl type
df$A[df$A=='?'] <- NaN
df$E[df$E=='?'] <- NaN
df$A <- as.double(as.character(df$A))
df$E <- as.double(as.character(df$E))

# Build a function using two-way linear model to estimate the missing data
impute.fn = function(data,index){
  df_missing <- data.matrix(data[index,])
  x_bar <- mean(df_missing, na.rm = T) # avg of all scores
  ai <- apply(df_missing, 2, mean, na.rm = T) # effects from different exams
  bj <- apply(df_missing, 1, mean, na.rm = T) # effects from different students
  for (i in seq(1:22)){
    if (is.nan(df_missing[,1][i])){
      # Estimate scores of student i in exam A
      df_missing[,1][i] <- x_bar + ai[1] + bj[i]
    }
  }
  for (i in seq(1:22)){
    if (is.nan(df_missing[,5][i])){
      # estimate scores of student i in exam E
      df_missing[,5][i] <- x_bar + ai[5] + bj[i]
    }
  }
  # Calculate the covariance matrix
  cov <- var(df_missing)
  # Return the maximum eigenvalue
  return(eigen(cov)$values[1])
}
```

(b)

```
df_original <- df[,2:6]
theta_hat <- impute.fn(df_original, 1:22)
cat("Imputed value of theta hat is", theta_hat,".")
```

```
## Imputed value of theta hat is 2775.86 .
```

(c)

```
# Use the boot function in boot library to perform Boostrap analysis
BootStrap_Data = boot(df_original,impute.fn,R=2000)
head(BootStrap_Data$t) # theta_b for b=1,..,2000
```

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

```
## [1,] 3368.156
## [2,] 2427.992
## [3,] 3264.824
## [4,] 2579.516
## [5,] 2909.623
## [6,] 3041.719
```

(d)

```
BootStrap_Data
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df_original, statistic = impute.fn, R = 2000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*  2775.86 325.036     447.338
```

```
q = quantile(BootStrap_Data$t,c(.025,.975))
cat("Approx 95% C.I. for theta is
    [",c(2*theta_hat-q[2],  2*theta_hat-q[1]),"] \n")
```

```
## Approx 95% C.I. for theta is
##     [ 1443.018 3196.166 ]
```

The bias is 292.0907 approximately and the approximate 95% confidence interval for θ is [1540.91, 3192.293].

(e) We should consider adding some noise to (1), which means the model may become $o_{ij} \approx \nu + \alpha_i + \beta_j + e_{ij}$, where e_{ij} are i.i.d. $N(0, \sigma^2)$ errors. Suppose we remain the original model, and apply it to the missing data-set, the scores actually don't conform to the model. For example, consider student 1 & 2, $o_{1j} \approx \nu + \alpha_1 + \beta_j$ and $o_{2j} \approx \nu + \alpha_2 + \beta_j$ are their scores in specific exam j. Therefore, $o_{2j} - o_{1j} = \alpha_2 - \alpha_1$, which means the difference of their scores should be constant no matter what exam. However, we can observe that in exam B, $o_{2B} - o_{1B} = -2$ while in exam C, $o_{2C} - o_{1C} = 7$. Hence, it would be better to add a noise term in (1) to account for the randomness of students' scores.