Statistical Exploration Assignment 3

NWUDO CHIKAEZE FIDELIS JUNIOR / STUDENT NUMBER: 202290064

2023-04-03

Question 1

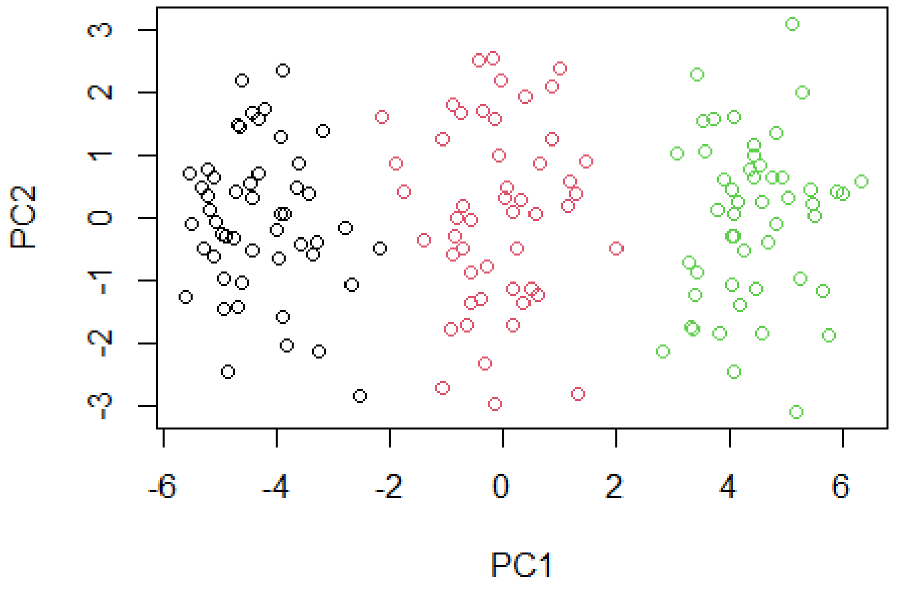
a) Generate a simulated data set with 50 observations in each of three classes (i.e. 150 observations total), and 50 variables. Be sure to add a mean shift to the observations in each class so that there are three distinct classes and choose covariance matrix of your choice.

(b) Perform PCA on the 150 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes.

c) Use diagnostic tools such as variance contribution plot and offer your comments.

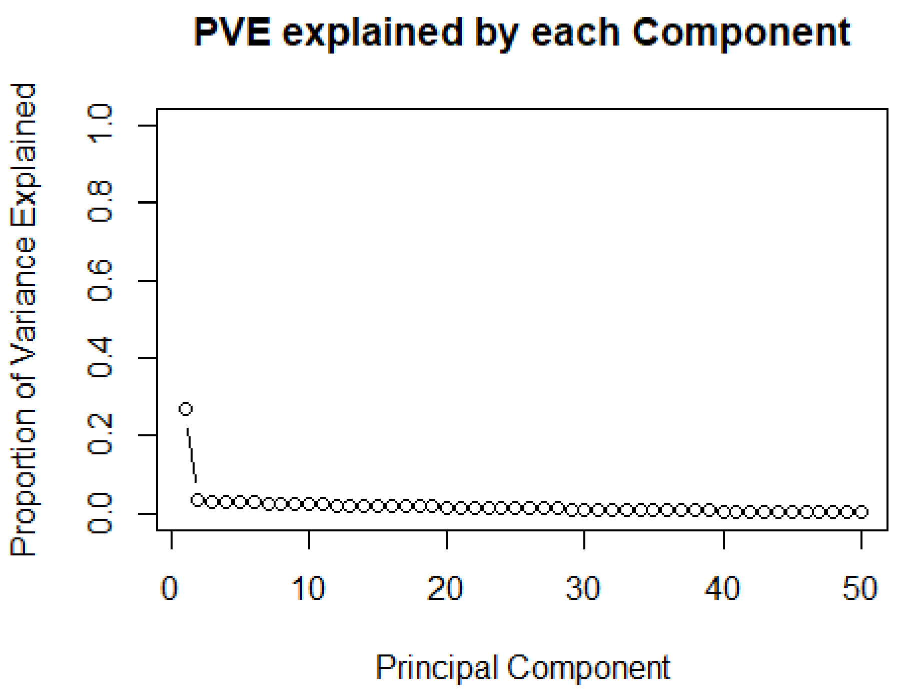
d) Also use loading plots & scatter plots of for first two PCs and interpret the results

```
# set seed for reproducibility
set.seed(123)

sim.data <- rbind(matrix(rnorm(50*50, mean = 0), nrow = 50),
matrix(rnorm(50*50, mean=0.7), nrow = 50),
matrix(rnorm(50*50, mean=1.4), nrow = 50))

# using prcomp automatically centers the data to mean 0 and standard
deviation 1
pr.out <- prcomp(sim.data, scale = TRUE)
pr.out$sdev

##  [1] 3.6645660 1.3134754 1.2777095 1.2495317 1.2257277 1.2090456 1.1815389
##  [8] 1.1573731 1.1247941 1.0922297 1.0724129 1.0504579 1.0281742 1.0134481
## [15] 1.0022929 0.9950544 0.9618867 0.9491197 0.9457791 0.9278890 0.9066939
## [22] 0.8751996 0.8593345 0.8543689 0.8317971 0.8112465 0.7977497 0.7915176
## [29] 0.7581247 0.7438617 0.7349149 0.6969727 0.6860205 0.6812307 0.6746564
## [36] 0.6558129 0.6439774 0.6201489 0.6108380 0.6010344 0.5895490 0.5594840
## [43] 0.5466670 0.5218285 0.4840195 0.4697280 0.4479767 0.4265977 0.4140485
## [50] 0.4028385

# pr.out$rotation

# Plotting first two principal components
X.pca <- pr.out$x
plot(X.pca[,1:2], col=c(rep(1,50), rep(2,50), rep(3,50)))
```

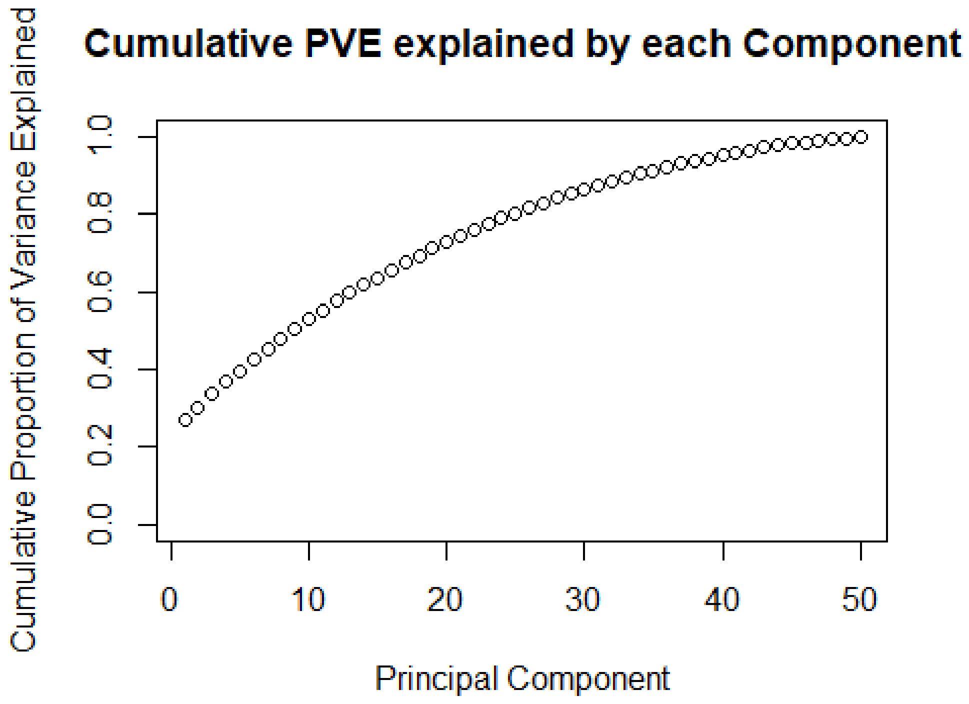


```
#legend("bottom", legend = c("class1", "class2", "class3"), col = 1:3, pch =
16, xpd = TRUE)

pr.var <- (pr.out$sdev)^2
pve <- pr.var / sum(pr.var)
summary(pr.out)

## Importance of components:
##                             PC1    PC2     PC3     PC4     PC5     PC6
PC7
## Standard deviation     3.6646 1.3135 1.27771 1.24953 1.22573 1.20905
1.18154
## Proportion of Variance 0.2686 0.0345 0.03265 0.03123 0.03005 0.02924
0.02792
## Cumulative Proportion  0.2686 0.3031 0.33574 0.36696 0.39701 0.42625
0.45417
##                             PC8    PC9    PC10   PC11    PC12    PC13
PC14
## Standard deviation     1.15737 1.1248 1.09223 1.0724 1.05046 1.02817
1.01345
## Proportion of Variance 0.02679 0.0253 0.02386 0.0230 0.02207 0.02114
0.02054
## Cumulative Proportion  0.48096 0.5063 0.53012 0.5531 0.57519 0.59633
0.61688
##                            PC15   PC16   PC17    PC18    PC19    PC20
PC21
## Standard deviation     1.00229 0.9951 0.9619 0.94912 0.94578 0.92789
```

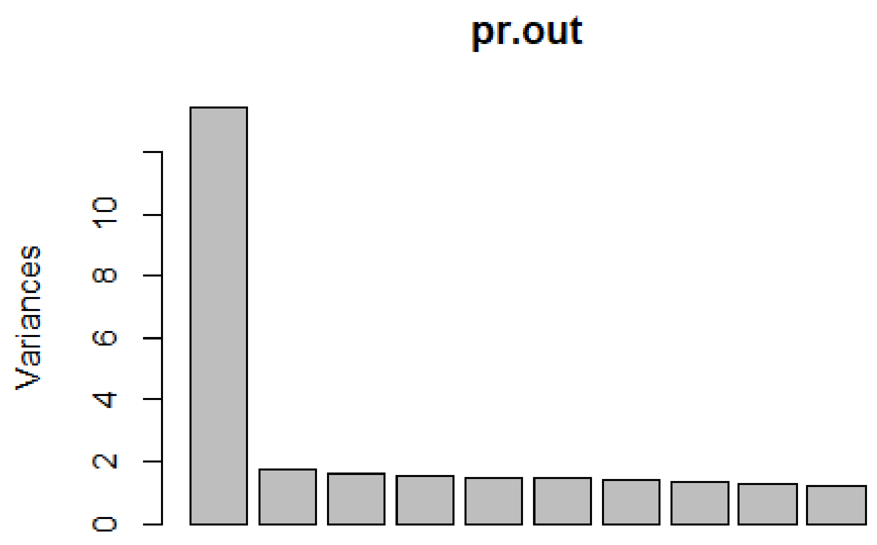
```
0.90669
## Proportion of Variance 0.02009 0.0198 0.0185 0.01802 0.01789 0.01722
0.01644
## Cumulative Proportion  0.63697 0.6568 0.6753 0.69329 0.71118 0.72840
0.74484
##                           PC22    PC23   PC24    PC25    PC26    PC27
PC28
## Standard deviation     0.87520 0.85933 0.8544 0.83180 0.81125 0.79775
0.79152
## Proportion of Variance 0.01532 0.01477 0.0146 0.01384 0.01316 0.01273
0.01253
## Cumulative Proportion  0.76016 0.77493 0.7895 0.80337 0.81653 0.82926
0.84179
##                          PC29    PC30   PC31    PC32    PC33    PC34
PC35
## Standard deviation     0.7581 0.74386 0.7349 0.69697 0.68602 0.68123
0.6747
## Proportion of Variance 0.0115 0.01107 0.0108 0.00972 0.00941 0.00928
0.0091
## Cumulative Proportion  0.8533 0.86435 0.8751 0.88487 0.89428 0.90356
0.9127
##                          PC36    PC37    PC38    PC39    PC40    PC41
PC42
## Standard deviation     0.6558 0.64398 0.62015 0.61084 0.60103 0.58955
0.55948
## Proportion of Variance 0.0086 0.00829 0.00769 0.00746 0.00722 0.00695
0.00626
## Cumulative Proportion  0.9213 0.92956 0.93725 0.94471 0.95194 0.95889
0.96515
##                           PC43    PC44    PC45    PC46    PC47    PC48
PC49
## Standard deviation     0.54667 0.52183 0.48402 0.46973 0.44798 0.42660
0.41405
## Proportion of Variance 0.00598 0.00545 0.00469 0.00441 0.00401 0.00364
0.00343
## Cumulative Proportion  0.97113 0.97657 0.98126 0.98567 0.98969 0.99333
0.99675
##                           PC50
## Standard deviation     0.40284
## Proportion of Variance 0.00325
## Cumulative Proportion  1.00000

plot(pve, xlab="Principal Component", ylab="Proportion of Variance
Explained", main="PVE explained by each Component",ylim=c(0,1), type='b')
```



```
plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative Proportion of
Variance Explained", main="Cumulative PVE explained by each Component",
ylim=c(0,1), type='b')
```

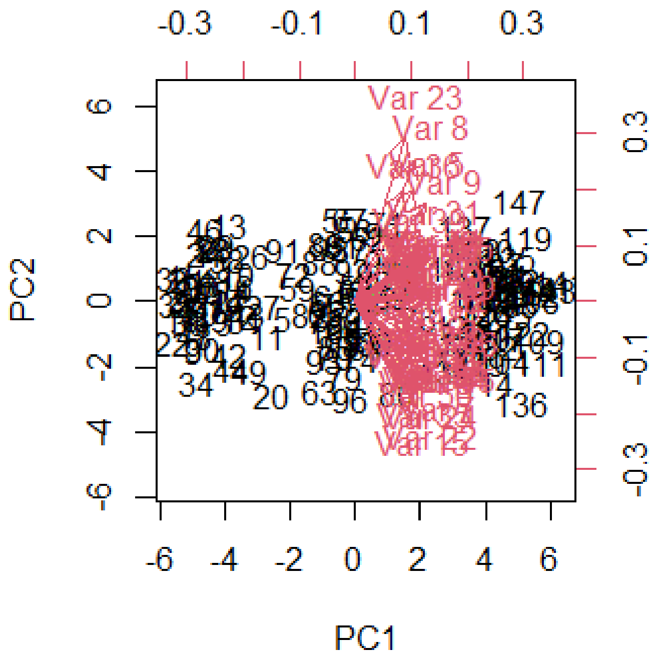


```r
plot(pr.out)
```

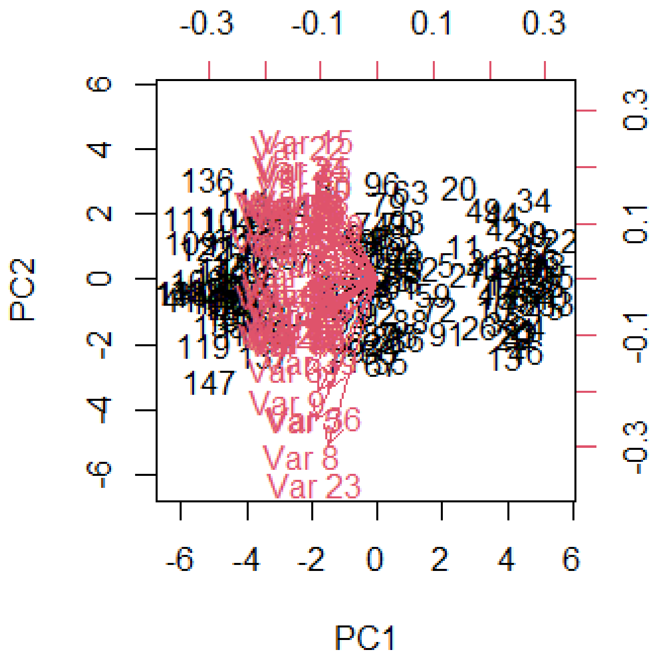


- In this case, PC1 explains the largest proportion of the variation in the data about 0.26, followed by PC2 and PC3 and so on. The cumulative proportion of the first 5 PCs is around 30%, and the cumulative proportion of the first 10 PCs is around 55%. This means that using the first 10 PCs, we can explain more than half of the variation in the original data. The first 34PCs explain around 90% of the variation of the data.

```r
# Loading plot for the two PCs
biplot(pr.out, scale=0)
```



```r
# mirror image
pr.out$rotation <- - pr.out$rotation
pr.out$x <- - pr.out$x
biplot(pr.out, scale=0)
```



```
pr.out$x <- pr.out$x
```

- The figure above plots the first two principal components of these data and the loading vectors in a single biplot. The dark numbers represent the scores for the first two principal components. The red arrows indicate the first two principal component loading vectors. The loadings represent the correlation between the variables and the components. From the figure above, we can see that most of the variables are clustered closely together, indicating positive correlation.

Question 2.

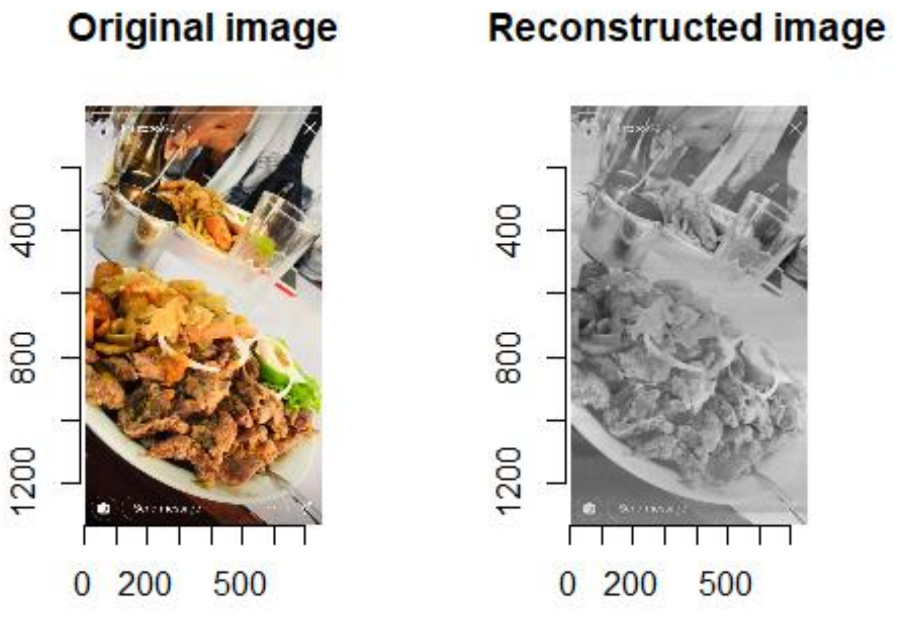
a) Choose any photo of your choice and convert it into the numerical data. Perform a PCA and reconstruct back the data based on first 100 PCA. Compare the file sizes of original photo and new photo based on PCA. Comment on the quality of photo based on PCA.

b) For matrix completion, the codes used in the book ”Statistical Learning using R” used the svd() function in R. Instead we can use the prcomp(). write a function using prcomp() for matrix completion.

ii) Use the generated data from Q.1(a). Consider only first 10 variables, so that your new data is of size 150 x 10. Randomly select 5 observations from this new data set and assume that these 5 observations are missing. Use Matrix Completion algorithm to estimate missing values using PCA. Offer your comments on the performance of the missing value estimation using PCA.

```
library(jpeg)
library(imager)
```

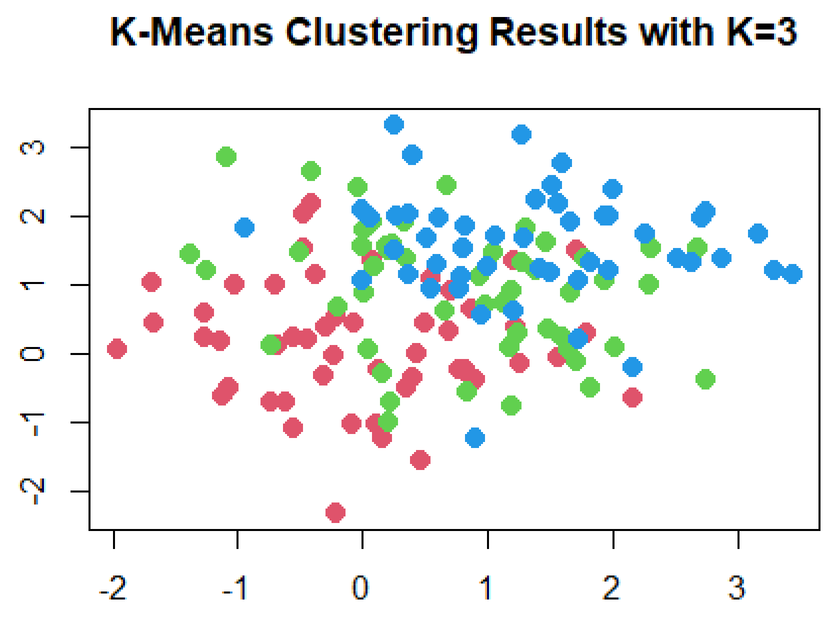
```
## Loading required package: magrittr

##
## Attaching package: 'imager'

## The following object is masked from 'package:magrittr':
##
##     add

## The following objects are masked from 'package:stats':
##
##     convolve, spectrum

## The following object is masked from 'package:graphics':
##
##     frame

## The following object is masked from 'package:base':
##
##     save.image

# Loading the image data
img <- load.image("IMG_0655.png")

# Convert to grayscale
gray_img <- grayscale(img)

# convert to matrix
mat_img <- as.matrix(gray_img)

# perform PCA
pca <- prcomp(mat_img, scale. = TRUE)


# Based on the 100 PC components we can approximate the image by multiplying
the PC scores based on the first 100  components with the transpose of the
100 eigen vectors matrix.
X.approx <- pca$x[,1:100] %*% t(pca$rotation[,1:100])
recon_img <- as.cimg(X.approx, dim(gray_img))

par(mfrow=c(1,2))
plot(img, main="Original image")
plot(recon_img, main="Reconstructed image")
```



```
# rank represents the number of principle components to be used for
approximating
# maxiter specifies the maximum number of iterations that the algorithm will
perform before stopping
# tol sets the tolerance level for convergence of the algorithm.

iterative_pca_mc <- function(X, rank, maxiter=1000, tol=1e-4) {

  # Step 1: Initialize missing values with mean of their respective columns
  Y <- X
  for (j in 1:ncol(Y)) {
    col_mean <- mean(Y[,j], na.rm=TRUE)
    Y[is.na(Y[,j]),j] <- col_mean
  }

  # Step 2: Perform PCA on the initialized matrix
  pca <- prcomp(Y, scale= TRUE)

  # Step 3: Iterate until convergence or maximum iterations reached
  for (i in 1:maxiter) {

    # Compute the low-rank approximation of Y
    Y_approx <- pca$x[, 1:rank] %*% t(pca$rotation[, 1:rank])

    # Replace missing values in Y with those in the low-rank approximation
    Y[is.na(Y)] <- Y_approx[is.na(Y)]
    # Update the PCA decomposition of Y
    pca <- prcomp(Y, center=TRUE, scale=FALSE)

    # Check for convergence
    if (sum((Y - Y_approx)^2, na.rm=TRUE) < tol) {
      break
    }
  }

  # Return the completed matrix
  return(Y)
}

# selecting the first 10 variable from the data
sim.new <- sim.data[,1:10]

# Select 5 random values from sim.new data
missing_values <- sample(sim.new, 5)

# Set the selected values to NA
sim.new[sim.new %in% missing_values] <- NA

# call my function to perform matrix completion
matrix_completed.data <- iterative_pca_mc(sim.new, 10, 1000, 1e-4)

pr.out1 <- prcomp(matrix_completed.data, scale. = TRUE)
pr.out1

## Standard deviations (1, .., p=10):
##  [1] 1.7813024 1.0101420 0.9840402 0.9534919 0.9149566 0.8694097 0.8514549
##  [8] 0.7860630 0.7499698 0.6563146
##
## Rotation (n x k) = (10 x 10):
##             PC1         PC2           PC3           PC4         PC5
##  [1,] 0.2775746 -0.59113465  3.828109e-01 -0.0446115253  0.04329729
##  [2,] 0.3125305  0.03686960 -4.716736e-01 -0.1416498462  0.48376107
##  [3,] 0.2886856  0.28293998  4.609870e-01 -0.3830048616  0.30680052
##  [4,] 0.3901518  0.10701537 -2.725465e-01  0.0006388144  0.08557812
##  [5,] 0.3129585  0.14488002  1.780640e-01  0.4024476238 -0.11144782
##  [6,] 0.2618896 -0.14227570 -2.985060e-01 -0.3402182009 -0.74279256
##  [7,] 0.3154149 -0.22694123 -5.381374e-05 -0.4618159304  0.07481754
##  [8,] 0.3511368 -0.05677016  3.609139e-01  0.2931902933 -0.16642968
##  [9,] 0.3658369 -0.15835523 -2.965005e-01  0.5024839623  0.10648989
## [10,] 0.2580111  0.66080178  7.214667e-02 -0.0426010880 -0.23206120
##                PC6         PC7         PC8         PC9        PC10
##  [1,]  0.175843589 -0.01598279 -0.47146108 -0.38675495  0.1416750
##  [2,] -0.004690189 -0.30714564 -0.45996814  0.26040424 -0.2250740
##  [3,] -0.246511516 -0.35836051  0.26615900  0.01628203  0.3506870
##  [4,]  0.166247857 -0.01414449  0.41284137 -0.68178347 -0.3026856
##  [5,] -0.729960882  0.15557873 -0.20330662 -0.06821550 -0.2682176
##  [6,] -0.177995452 -0.32450646 -0.01171029  0.05653995  0.1193971
##  [7,] -0.058830712  0.69502740  0.18909331  0.29945886 -0.1348460
##  [8,]  0.409139954 -0.25411586  0.20905316  0.43764679 -0.4029312
##  [9,]  0.029723298  0.10677601  0.20335310  0.16080787  0.6401937
## [10,]  0.379980866  0.29704465 -0.40322755 -0.05095908  0.1951631
```

Question 3. For this problem, use the data in Q1(a). a) Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same. (b) Perform K-means clustering with K = 2. Describe your results. (c) Now perform K-means clustering with K = 4, and describe your results. (d) Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 150x2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results. (g) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in Q1? Explain.

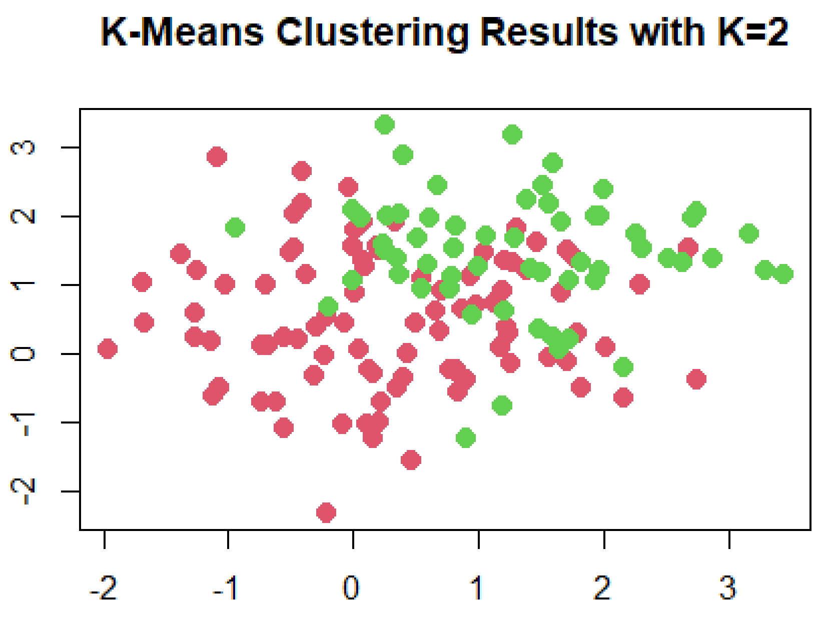
```
#K-means clustering with K = 3
km.out3 <- kmeans(sim.data, 3, nstart = 20)
km.out3$tot.withinss

## [1] 7232.31

plot(sim.data, col=(km.out3$cluster+1), main="K-Means Clustering Results with
K=3", xlab = "", ylab = "", pch=20, cex=2)
```



```
true_class = c(rep(1,50), rep(2,50), rep(3,50))
table(true_class, km.out3$cluster)

##
## true_class  1  2  3
##          1 50  0  0
##          2  1 49  0
##          3  0  1 49
```

- The clustering result are almost perfectly clustered except for one observation in both class 2 & 3 that are classified to to a different cluster.

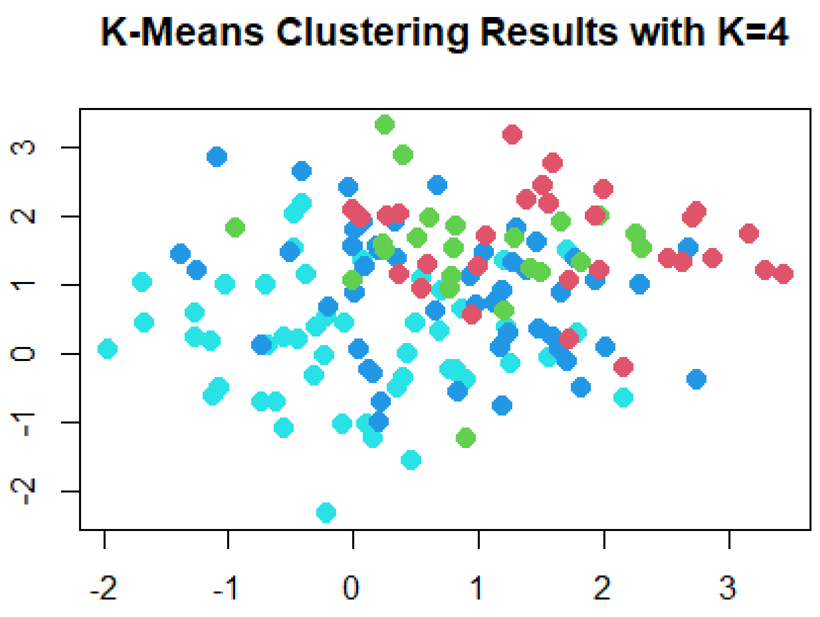
```
# K-means clustering with K = 2
km.out2 <- kmeans(sim.data, 2, nstart = 20)
km.out2$tot.withinss

## [1] 7788.048

plot(sim.data, col=(km.out2$cluster+1), main="K-Means Clustering Results with
K=2", xlab = "", ylab = "", pch=20, cex=2)
```



```
table(true_class, km.out2$cluster)

##
## true_class  1  2
##          1 50  0
##          2 41  9
##          3  0 50
```

- The clustering result is almost perfect as well except for nine observations in class 2 that have been classified to the to another cluster. The extreme classes are classified correctly.

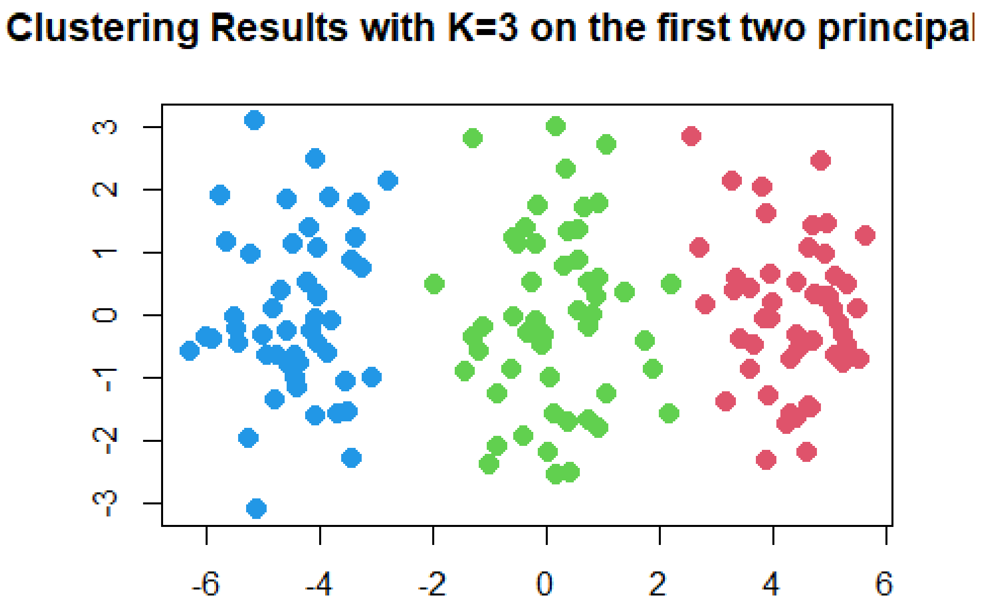
```
# K-means clustering with K = 4
km.out4 <- kmeans(sim.data, 4, nstart = 20)
km.out4$tot.withinss

## [1] 7077.42

plot(sim.data, col=(km.out4$cluster+1), main="K-Means Clustering Results with
K=4", xlab = "", ylab = "", pch=20, cex=2)
```



```
table(true_class, km.out4$cluster)

##
## true_class  1  2  3  4
##          1  0  0  1 49
##          2  0  1 48  1
##          3 29 21  0  0
```

The clustering result indicate that 1 of the observation in class 1 have been classified in another cluster. The observations in class 2 have been split into 3 clusters (48 observations in one cluster and the other two observations split in two different clusters). Finally, class three observations are split into 2 clusters.

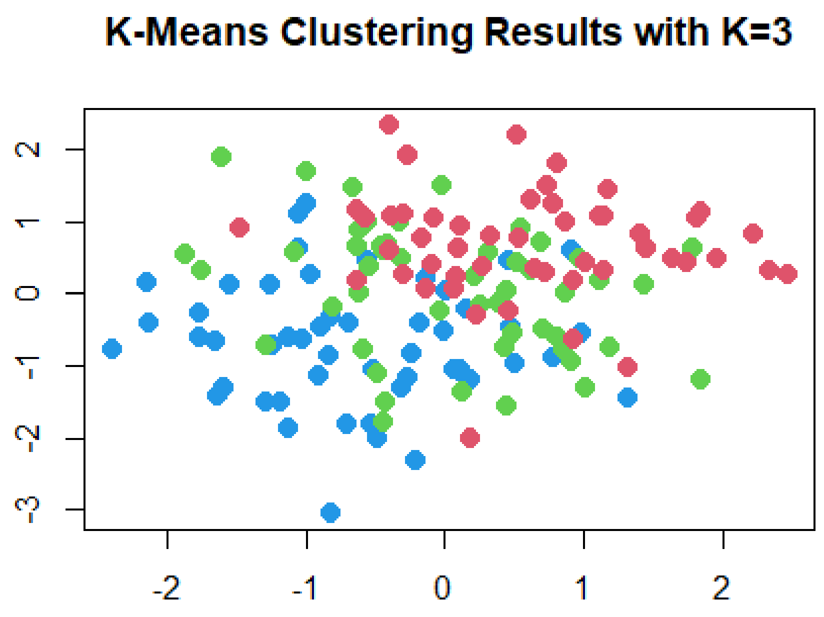
```
# K-means clustering with K = 3 on the first two principal score vectors.
m.comp <- pr.out$x[,1:2]
km.out <- kmeans(m.comp, 3, nstart = 20)
km.out$tot.withinss

## [1] 363.0774

plot(m.comp, col=(km.out$cluster+1), main="K-Means Clustering Results with
K=3 on the first two principal components", xlab = "", ylab = "", pch=20,
cex=2)
```



```
table(true_class, km.out$cluster)

##
## true_class  1  2  3
##          1 49  1  0
##          2  0 50  0
##          3  0  0 50
```

- The clustering result are almost perfectly clustered except for one observation in class 1 that is classified to to a different cluster. The PCA carries enough information.

```
res <- kmeans(scale(sim.data), 3, nstart = 20)
res$tot.withinss

## [1] 5521.778

plot(scale(sim.data), col=(res$cluster+1), main="K-Means Clustering Results
with K=3", xlab = "", ylab = "", pch=20, cex=2)
```



```
table(true_class, res$cluster)

##
## true_class  1  2  3
##          1  0  0 50
##          2  0 49  1
##          3 50  0  0
```

- The scaling does not change the result. The clustering result are almost perfectly clustered except for one observation in class 2 that is classified to to a different cluster. The PCA carries enough information.