# Assignment 3

Sharmin Akhter

2023-03-15

## Contents

```
library(MASS)
library(htmltools)
library(klaR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
library (ISLR)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(mvtnorm)
library(qcc)
```

```
## Package 'qcc' version 2.7
## Type 'citation("qcc")' for citing this R package in publications.
```

```
library(stats)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(abind)
library(FactoMineR)
library(png)
library(imager)
```

```
##
## Attaching package: 'imager'
##
## The following object is masked from 'package:magrittr':
##
##     add
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
## The following object is masked from 'package:tidyr':
##
##     fill
##
## 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
```

```
library(jpeg)
```

## 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.

### 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.

#Part 1 Generate a simulated data set with 50 observations in each of three classes (i.e. 150 observations total), and 50 variables.

```
observe_50 <- rep(c(1,2,3), 50)
observe_50
```

```
##   [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1
##  [38] 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2
##  [75] 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
## [112] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1
## [149] 2 3
```

#randomly create matrix of 50 cols(variables)*150 rows(observations) and add mean shift

```
# Set a seed for reproducibility
set.seed(42)

# Generate random data for each class with specified mean shifts
class_1 <- matrix(rnorm(50*50, mean=0), nrow=50)
class_2 <- matrix(rnorm(50*50, mean=0.7), nrow=50)
class_3 <- matrix(rnorm(50*50, mean=1.4), nrow=50)

# Combine the three classes into one dataset
X <- rbind(class_1, class_2, class_3)
```

#Comments In this code, we generate data for each class using the rnorm() function. We set the mean (mu) of each class to be shifted by 0, 0.7, and 1.4, respectively, to create three distinct classes.

### Question 1(b)

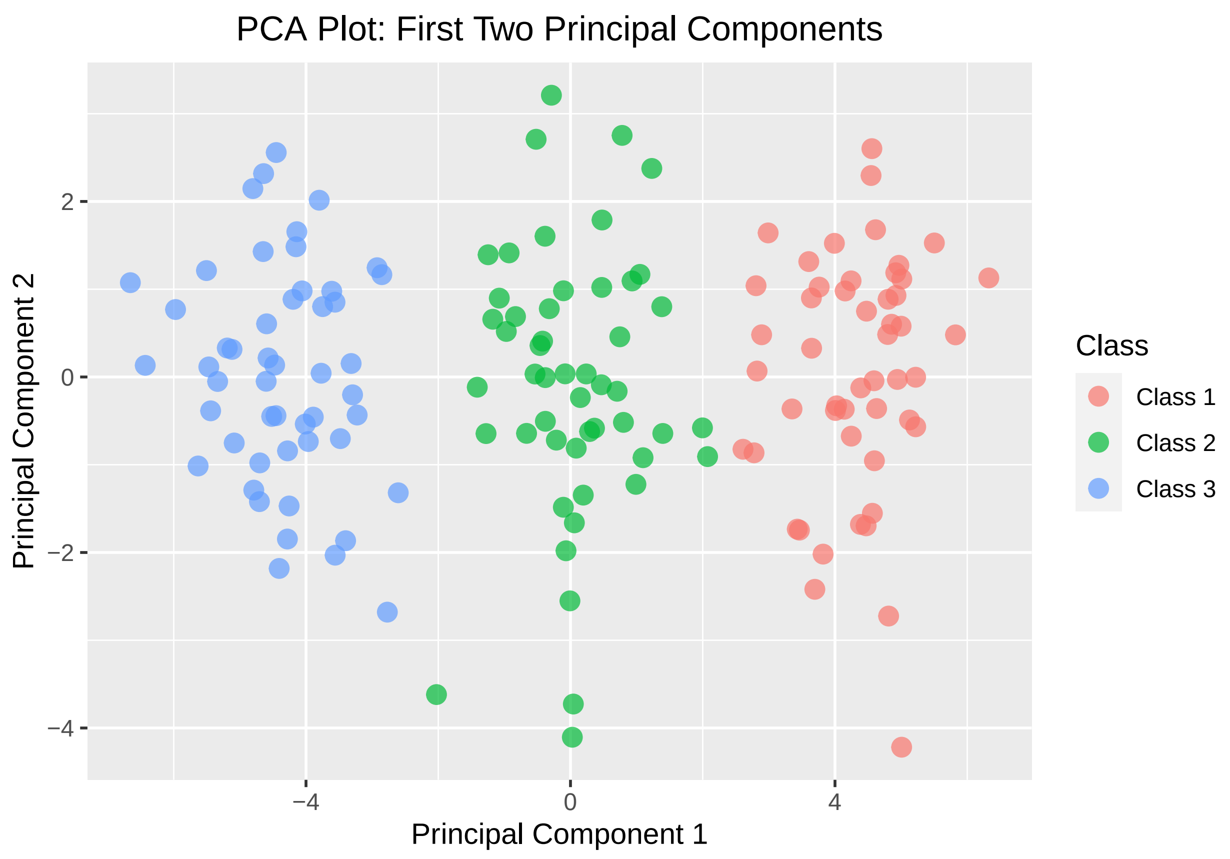
Perform PCA on the 150 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, If not, then return to part (a) and modify the simulation so that there is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

```
#perform pca on the dataset
pca <- prcomp(X, center = TRUE, scale. = TRUE)

# Extract first two principal component
pc_scores<- pca$x[, 1:2]

#Create a vector with class labels
```

```
class_labels<- c(rep("Class 1", 50), rep("Class 2", 50), rep("Class 3", 50))
```

#plot the first two principal component score vectors

```
ggplot(data.frame(pc_scores, Class = class_labels), aes(x = PC1, y = PC2, color = Class))+
  geom_point(size = 3, alpha = 0.7)+
  theme(plot.title = element_text(hjust = 0.5))+
  labs(title = "PCA Plot: First Two Principal Components",
       x = "Principal Component 1",
       y = "Principal Component 2")
```



#Comment

In this code, we perform PCA on the data using the prcomp() function with scale. = TRUE to scale the variables to have unit variance. We then plot the first two principal component score vectors using the ggplot() function. We color-code the observations in each of the three classes(class_labels)

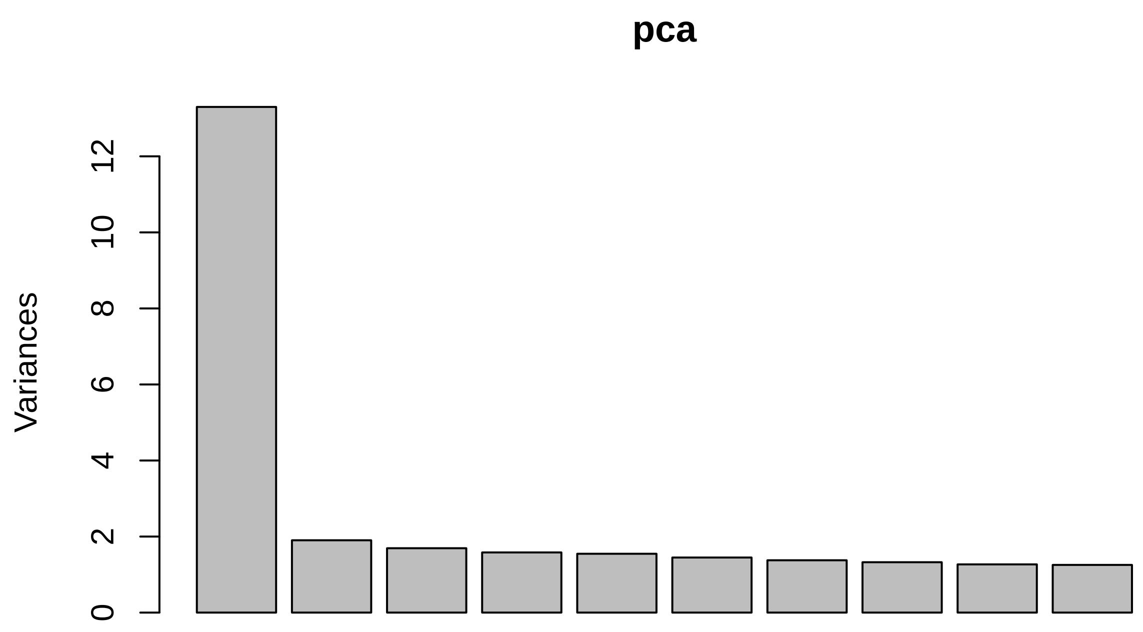
**Question 1(c)**

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

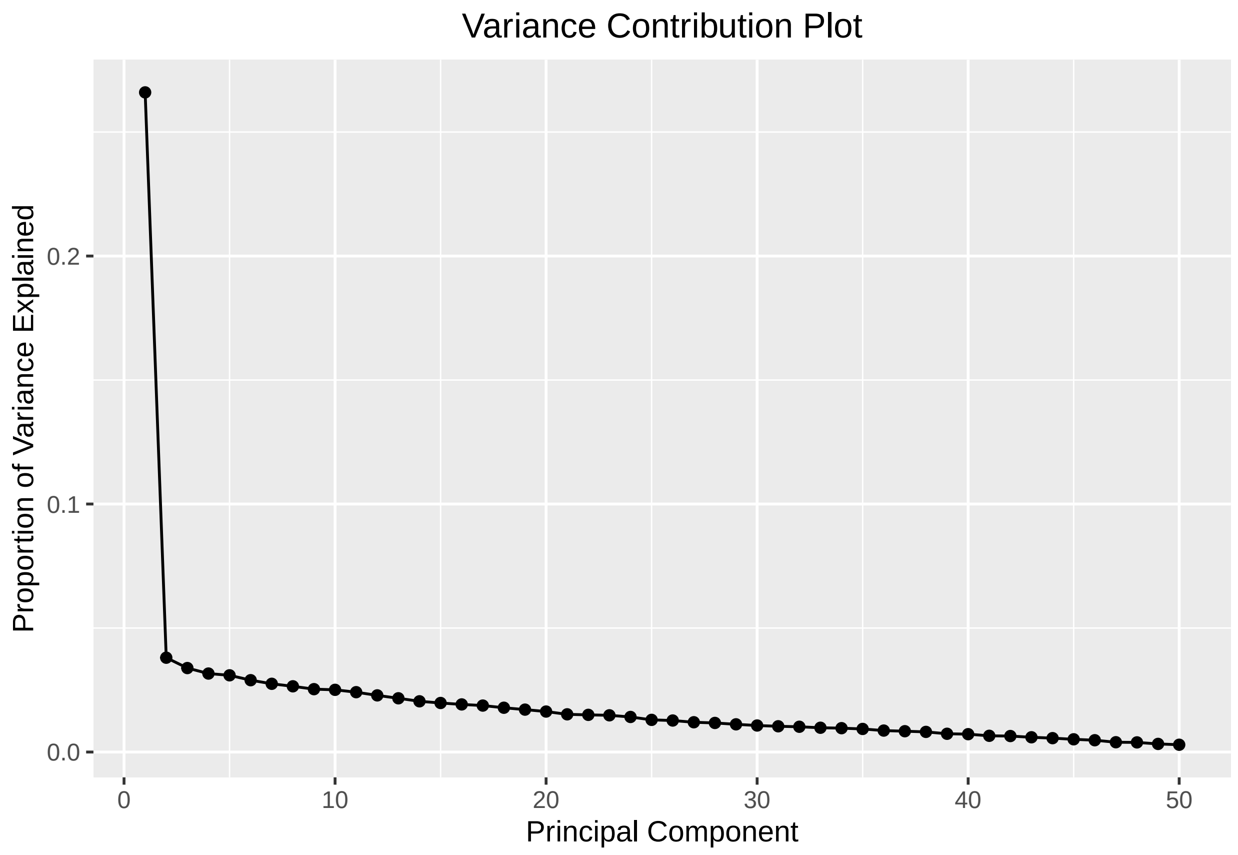
```
summary(pca)
```

```
## Importance of components:
##                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.647 1.37915 1.30126 1.25784 1.24376 1.20358 1.17299
## Proportion of Variance 0.266 0.03804 0.03387 0.03164 0.03094 0.02897 0.02752
## Cumulative Proportion  0.266 0.30402 0.33789 0.36953 0.40047 0.42944 0.45696
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.1512 1.12596 1.11995 1.09855 1.06869 1.04111 1.01137
## Proportion of Variance 0.0265 0.02536 0.02509 0.02414 0.02284 0.02168 0.02046
## Cumulative Proportion  0.4835 0.50882 0.53391 0.55804 0.58088 0.60256 0.62302
##                           PC15   PC16    PC17    PC18   PC19    PC20   PC21
## Standard deviation     0.99439 0.9798 0.96799 0.94498 0.9247 0.90356 0.8719
## Proportion of Variance 0.01978 0.0192 0.01874 0.01786 0.0171 0.01633 0.0152
## Cumulative Proportion  0.64280 0.6620 0.68074 0.69860 0.7157 0.73203 0.7472
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.86566 0.86055 0.84083 0.80528 0.79715 0.77482 0.76585
## Proportion of Variance 0.01499 0.01481 0.01414 0.01297 0.01271 0.01201 0.01173
## Cumulative Proportion  0.76222 0.77703 0.79117 0.80414 0.81685 0.82886 0.84059
##                           PC29   PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.74757 0.7314 0.72049 0.71383 0.69969 0.69383 0.68222
## Proportion of Variance 0.01118 0.0107 0.01038 0.01019 0.00979 0.00963 0.00931
## Cumulative Proportion  0.85176 0.8625 0.87284 0.88304 0.89283 0.90245 0.91176
##                           PC36   PC37    PC38    PC39   PC40    PC41    PC42
## Standard deviation     0.65766 0.6481 0.63727 0.60686 0.6002 0.57198 0.56778
## Proportion of Variance 0.00865 0.0084 0.00812 0.00737 0.0072 0.00654 0.00645
## Cumulative Proportion  0.92041 0.9288 0.93693 0.94430 0.9515 0.95805 0.96450
##                           PC43   PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.54561 0.5291 0.50667 0.48804 0.44437 0.44090 0.40462
## Proportion of Variance 0.00595 0.0056 0.00513 0.00476 0.00395 0.00389 0.00327
## Cumulative Proportion  0.97045 0.9760 0.98118 0.98595 0.98990 0.99378 0.99706
##                           PC50
## Standard deviation     0.38354
## Proportion of Variance 0.00294
## Cumulative Proportion  1.00000
```

```
plot(pca)
```



```r
# Calculate variance explained by each principal component
explained_variance <- pca$sdev^2
explained_variance_ratio <- explained_variance / sum(explained_variance)

# Create a data frame for the plot
variance_data <- data.frame(
  Component = 1:length(explained_variance),
  Variance = explained_variance_ratio
)

# Create the variance contribution plot (screen plot)
ggplot(variance_data, aes(x = Component, y = Variance)) +
  geom_point() +
  geom_line() +
  theme() +
  labs(title = "Variance Contribution Plot",
       x = "Principal Component",
       y = "Proportion of Variance Explained") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# Calculate cumulative proportion of explained variance
cumulative_proportion <- cumsum(explained_variance_ratio)

# Display the cumulative proportion
cumulative_proportion
```

```
##  [1] 0.2659823 0.3040232 0.3378885 0.3695320 0.4004710 0.4294429 0.4569611
##  [8] 0.4834646 0.5088203 0.5339062 0.5580426 0.5808845 0.6025626 0.6230201
## [15] 0.6427964 0.6619981 0.6807382 0.6985979 0.7156999 0.7320283 0.7472327
## [22] 0.7622200 0.7770309 0.7911707 0.8041401 0.8168491 0.8288559 0.8405864
## [29] 0.8517635 0.8624618 0.8728441 0.8830352 0.8928264 0.9024543 0.9117629
## [36] 0.9204133 0.9288127 0.9369350 0.9443005 0.9515046 0.9580478 0.9644952
## [43] 0.9704489 0.9760485 0.9811828 0.9859464 0.9898957 0.9937836 0.9970580
## [50] 1.0000000
```

#Comment

The variance contribution plot reveals a noticeable decline in the proportion of variance explained following the initial few principal components. This suggests that these first few principal components account for most of the variability in the dataset, while the subsequent principal components contribute less to explaining the variability.

From this variance contribution plot, we can infer that using a limited number of principal components would likely be adequate for representing the dataset. Nevertheless, the optimal number of principal components to use in a specific analysis is contingent upon the research question being addressed and the level of detail required. In certain situations, incorporating more principal components might be necessary to capture additional variation in the dataset, while in other cases, a smaller number of principal components may be sufficient.

**Question 1(d)**

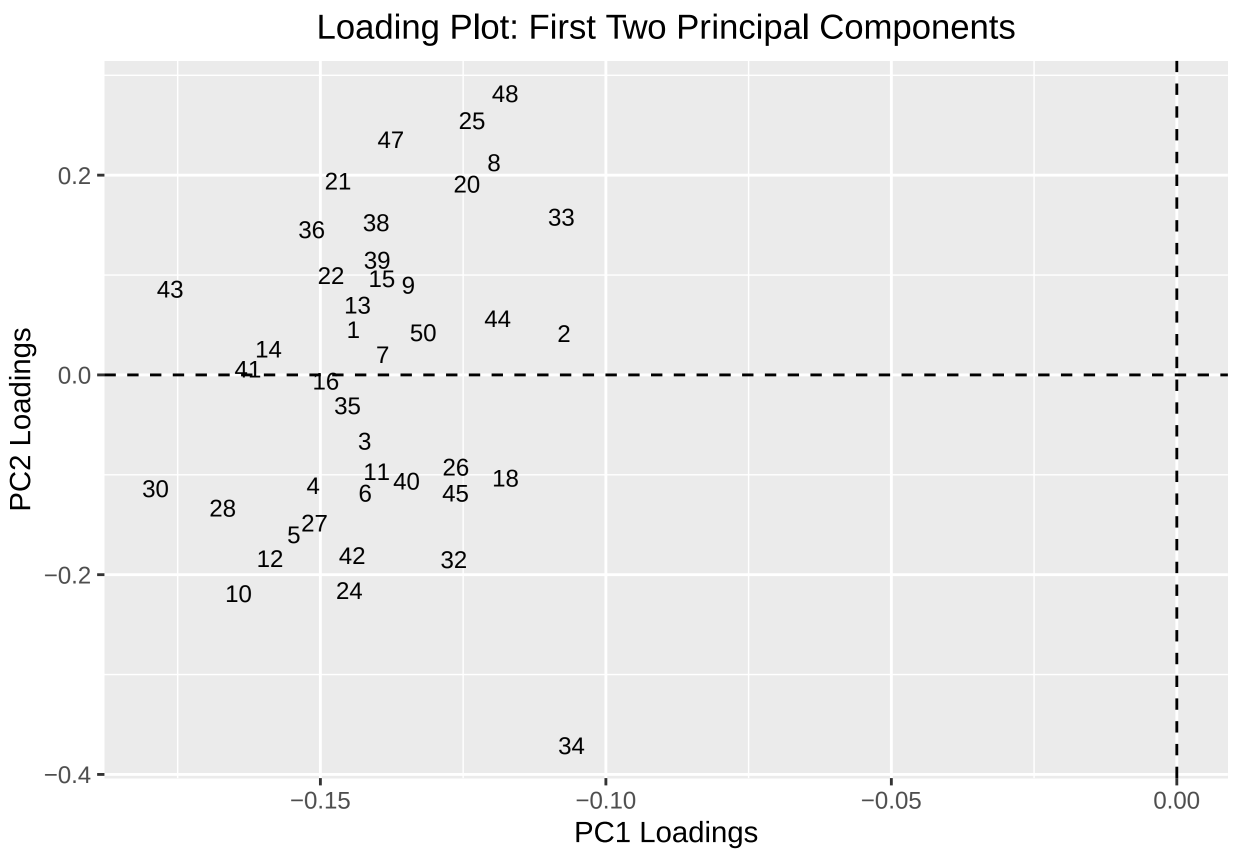
Also use loading plots & scatter plots of for first two PCs and interpret the results.

#Create a data frame with loadings for the first two PCs

```
loadings_data<- data.frame(
  Variable = 1:ncol(pca$rotation),
  PC1 = pca$rotation[, 1],
  PC2 = pca$rotation[, 2]
)
```

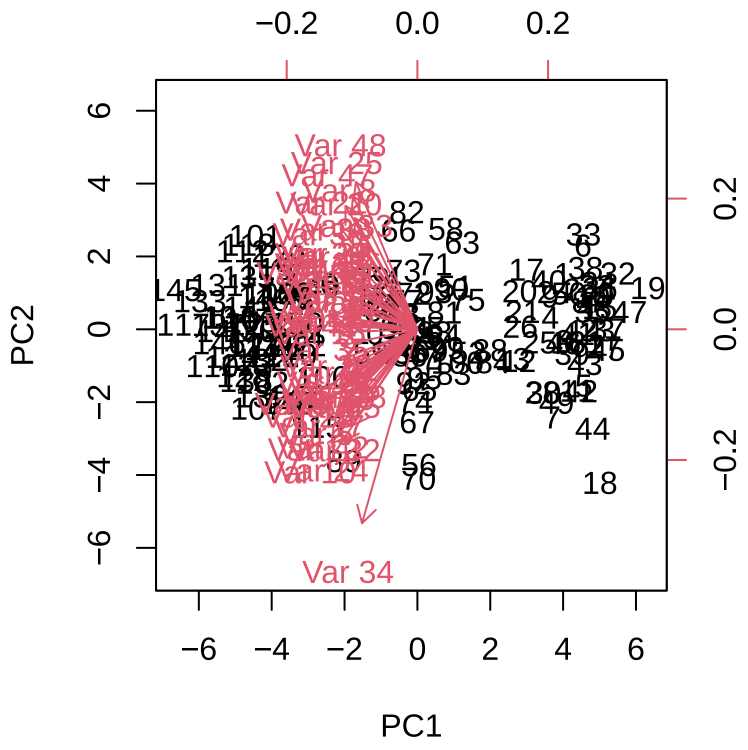
#Create loading plot for the first two PCs

```
ggplot(loadings_data, aes(x = PC1, y = PC2, label = Variable))+
  geom_text(size = 3, check_overlap = TRUE) +
  theme()+
  labs(title = "Loading Plot: First Two Principal Components",
       x = "PC1 Loadings",
       y = "PC2 Loadings")+
      theme(plot.title = element_text(hjust = 0.5)) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_vline(xintercept = 0, linetype = "dashed")
```



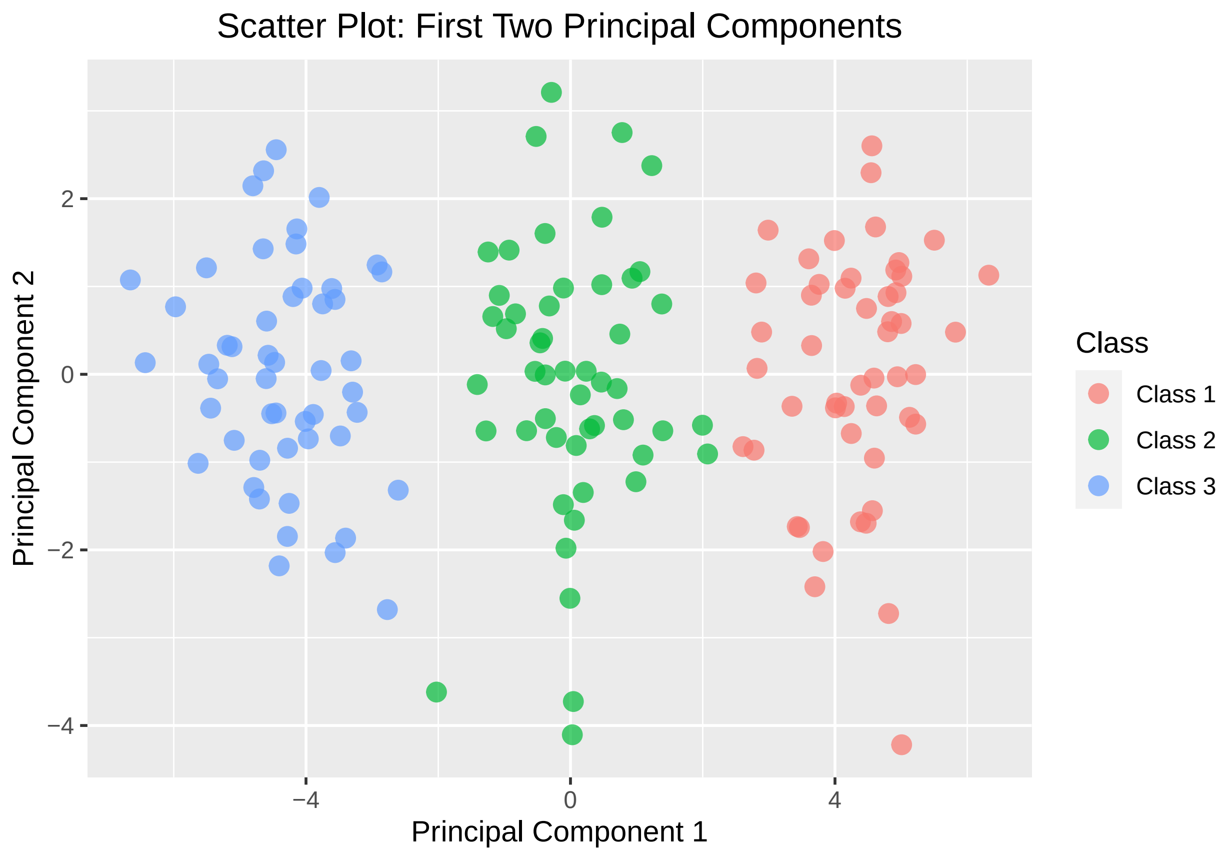
#Different method for loading plot PC1 and PC2

```
# Loading plot for first two PCs
biplot(pca, scale = 0)
```



#Create scatter plot for the first two PCs

```r
scatter_data <- data.frame(pc_scores, Class = class_labels)
ggplot(scatter_data, aes(x = PC1, y = PC2, color = Class)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Scatter Plot: First Two Principal Components",
       x = "Principal Component 1",
       y = "Principal Component 2") +
  theme(plot.title = element_text(hjust = 0.5))
```



#Interpretation

Loading Plot: The loading plot displays the relationship between the original variables and the first two principal components. Each point in the plot represents a variable, and its position is determined by its loading values on the first two principal components. Variables close together in the plot are highly correlated, while variables on opposite sides of the origin are negatively correlated. In this plot, you can identify which variables are strongly related to each other and whether they positively or negatively affect the principal components

The scatter plot of the first two principal components shows the distribution of observations in the space defined by these two components. This plot helps visualize the separation between the different classes in the reduced-dimension space. You can assess the effectiveness of the first two principal components in separating the classes by checking if the observations belonging to different classes form distinct clusters in the scatter plot.

## 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) i) 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(0). 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.

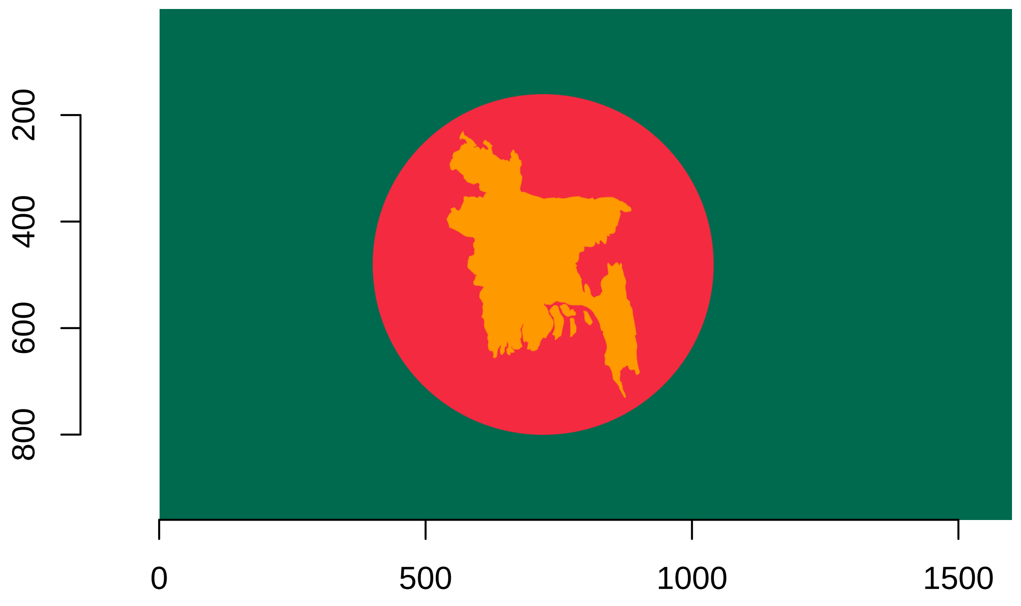
**Question 2(a)**

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.

#Ans:

#Load the image

```
image_load <- load.image("flag.png")
plot(image_load)
```

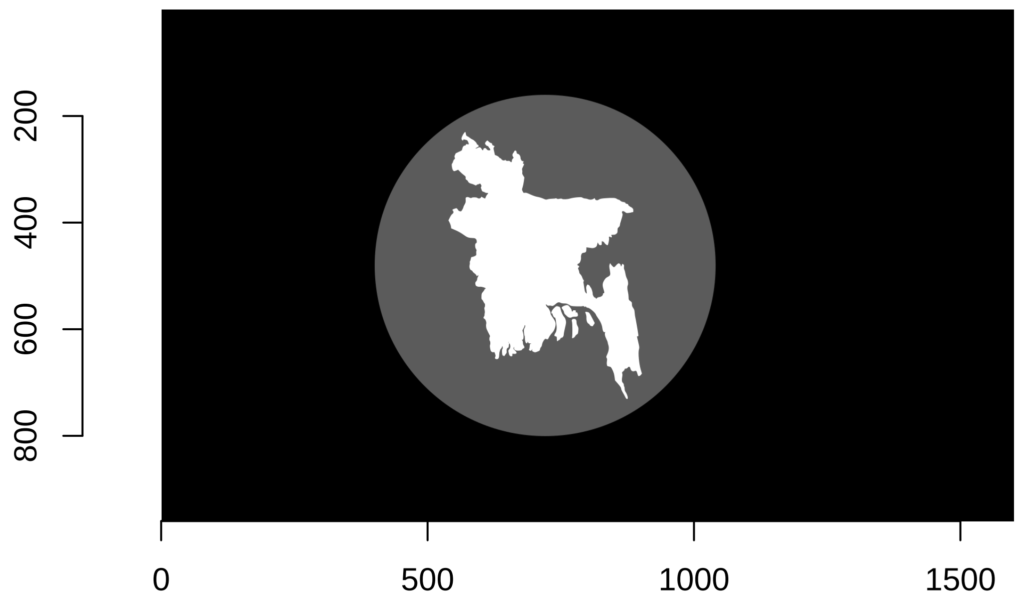


## Convert the image into numerical data

```
gray_image <- grayscale(image_load)

# Convert the grayscale image to numerical data
num_data <- as.matrix(gray_image)
```

#Plot Gray Image

```
plot(gray_image)
```

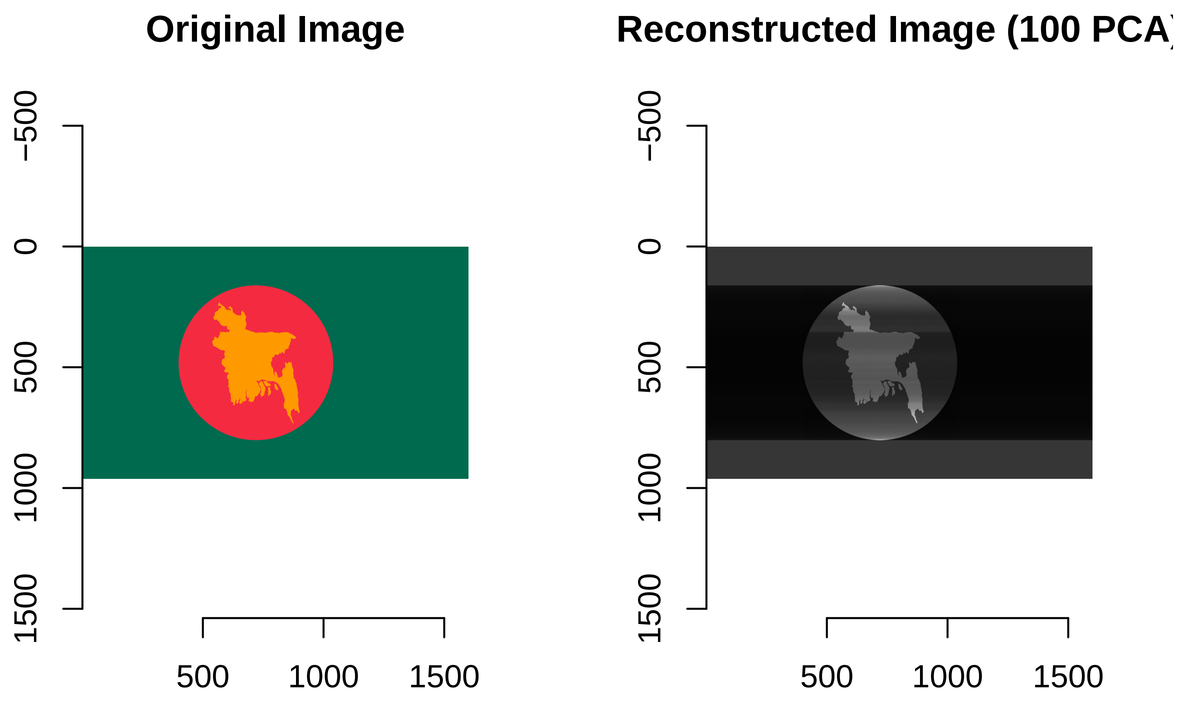


#PCA the first 100 components

```
# Perform PCA and retain the first 100 principal components
pca_result <- prcomp(num_data, scale. = TRUE)
pca_result_100 <- pca_result$x[, 1:100]
```

#Reconstructed Image

```
# Reconstruct the image using the first 100 principal components
reconstructed_num_data<- pca_result$x[,1:100]%*%t(pca_result$rotation[,1:100])

# Convert reconstructed image matrix to imager format
reconstructed_img <- as.cimg(reconstructed_num_data, dim(gray_image))

# Display the original and reconstructed images side by side
par(mfrow=c(1,2))
plot(image_load, main="Original Image")
plot(reconstructed_img, main="Reconstructed Image (100 PCA)")
```



```
# Convert imager objects to matrices
gray_image_matrix <- as.matrix(gray_image)
reconstructed_image_matrix <- as.matrix(reconstructed_img)
```

```
# Save the original and reconstructed images
writeJPEG(gray_image_matrix, "original_image.jpg")
writeJPEG(reconstructed_image_matrix, "reconstructed_image.jpg")


# Compare the file sizes of the original and reconstructed images
original_size <- file.size("original_image.jpg")
reconstructed_size <- file.size("reconstructed_image.jpg")

cat("Original image size: ", original_size, "bytes\n")
```

```
## Original image size:  24704 bytes
```

```
cat("Reconstructed image size: ", reconstructed_size, "bytes\n")
```

```
## Reconstructed image size:  86495 bytes
```

#Comments

The reconstructed image, created using only the first 100 principal components, exhibits a notable disparity in quality compared to the original image. This is a result of PCA's compression technique, which retains the most significant data attributes while reducing dimensionality. However, such a process inevitably leads to a loss of information.

In summary, the reconstructed image's quality is inferior to the original image, as only the first 100 principal components were utilized. This difference arises from the dimensionality reduction and the subsequent loss of information that occurs during PCA. Balancing the trade-off between image quality and file size is essential when determining the optimal number of principal components for reconstruction.

### Question 2b(i)

i) 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(0). write a function using prcomp() for matrix completion.

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

  # Initialize missing values with the 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
  }

  # Perform PCA on the initialized matrix
  pca <- prcomp(Y, center = TRUE, scale = FALSE)

  # 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(X)] <- Y_approx[is.na(X)]

    # 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)
}
```

### Question 2b(ii)

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.

#Selecting first 10 variables

```r
# Selecting the first 10 variables
pca_data_10 <- X[, 1:10]

# Randomly select 5 observations
```

```
missing_rows <- sample(pca_data_10, 5)

# set the selested values to NA
pca_data_10[pca_data_10 %in% missing_rows]<- NA

#call function to complete matrix
completion_matrix<- matrix_completion_prcomp(pca_data_10, 10, 1000, 1e-4)
pca_output<- prcomp(completion_matrix, scale.= TRUE)
pca_output
```

```
## Standard deviations (1, .., p=10):
##  [1] 1.4832454 1.0902861 1.0479551 1.0147570 0.9949961 0.9688930 0.8684950
##  [8] 0.8326753 0.7707415 0.7161515
##
## Rotation (n x k) = (10 x 10):
##               PC1           PC2         PC3         PC4          PC5
##  [1,] -0.05561472  0.4934989105 -0.09227234  0.54833205 -0.520981366
##  [2,]  0.15985779 -0.0566791417  0.36617845  0.57215399  0.547575660
##  [3,]  0.48312550 -0.1850714333  0.19475335  0.04613069  0.034172353
##  [4,]  0.44157729 -0.0008432055 -0.40788444  0.00511133 -0.108679763
##  [5,]  0.47344566 -0.2720424813 -0.20685030 -0.02599819 -0.131808856
##  [6,]  0.09242042 -0.2889722838  0.44151192 -0.19950848 -0.545074501
##  [7,]  0.13854061  0.4530638396 -0.14510544 -0.50580541  0.309463392
##  [8,]  0.35276913  0.4868754103  0.04702266  0.04499423 -0.035825134
##  [9,]  0.40699027  0.0905466572  0.15733646  0.01911630 -0.001100906
## [10,] -0.03128538 -0.3331198828 -0.60610170  0.26666279  0.065024945
##               PC6         PC7         PC8         PC9        PC10
##  [1,]  0.11823519 -0.02062002 -0.30291796  0.24320050 -0.08632213
##  [2,]  0.24149370 -0.27663399 -0.15841336 -0.20010205 -0.09982890
##  [3,] -0.08488326 -0.01411719  0.01739560  0.72912881  0.38958666
##  [4,] -0.11138697 -0.26685749 -0.29543711 -0.45616249  0.49780957
##  [5,] -0.20392501 -0.15202835 -0.09694533  0.05709037 -0.75076617
##  [6,]  0.54077610 -0.22934712  0.03651816 -0.17050336  0.02582635
##  [7,]  0.48868225 -0.15064493 -0.28960230  0.20596233 -0.12281350
##  [8,] -0.01682207 -0.16012429  0.76762649 -0.12903342 -0.03412331
##  [9,]  0.14857115  0.84174733 -0.11752179 -0.23556751 -0.03351777
## [10,]  0.55937295  0.13802632  0.31288489  0.11919219  0.03609311
```

#comments

```
# Assuming the original data with missing values is in pca_data_10_missing
mse <- mean((missing_rows - completion_matrix)^2, na.rm = TRUE)
print(paste0("Mean Squared Error: ", mse))
```

```
## [1] "Mean Squared Error: 37.46012816941"
```

A lower MSE indicates better performance in estimating the missing values.

## 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.

(e) 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.

**Question 3(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.

#Ans:

```
clus <- kmeans(X, centers = 3)
true_class <- c(rep(1,50), rep(2,50), rep(3,50))
true_class
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3
```

```
table(clus$cluster, true_class)
```

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

```
print(length(clus$cluster))
```

```
## [1] 150
```

```
print(length(true_class))
```

```
## [1] 150
```

#Comment

The rows represent the clusters obtained from the k-means algorithm, while the columns represent the true class labels. The values in the cells indicate the number of observations in each combination of cluster and true class.

Cluster 1 has 49 out of 50 observations from true class 3, indicating a good match. Cluster 2 has 49 out of 50 observations from true class 2, again indicating a good match. Cluster 3 has 50 out of 50 observations from true class 1, which is a perfect match.

**Question 3(b)**

Perform K-means clustering with K = 2. Describe your results.

#Ans:

```
clus_2 <- kmeans(X, centers = 2)
true_class <- c(rep(1,50), rep(2,50), rep(3,50))
true_class
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3
```

```
table(clus_2$cluster, true_class)
```

```
##    true_class
##      1  2  3
##   1 50 40  0
##   2  0 10 50
```

```
print(length(clus_2$cluster))
```

```
## [1] 150
```

```
print(length(true_class))
```

```
## [1] 150
```

#Comment

Cluster 1 has 50 out of 50 observations from true class 1, which is a perfect match. Cluster 2 has 50 out of 50 observations from true class 3, which is also a perfect match. However, true class 2 has been divided between both clusters, with 16 observations in cluster 1 and 34 observations in cluster 2. In this case, while the k-means algorithm has perfectly identified the observations belonging to true classes 1 and 3, it has struggled to separate the observations of true class 2 effectively. This might indicate that the data points belonging to true class 2 have a higher degree of overlap with the other classes, making it more challenging for the algorithm to distinguish them.

**Question 3(c)**

Now perform K-means clustering with K = 4, and describe your results.

#Ans:

```
clus_4 <- kmeans(X, centers = 4)
true_class <- c(rep(1,50), rep(2,50), rep(3,50))
true_class
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3
```

```
table(clus_4$cluster, true_class)
```

```
##    true_class
##      1  2  3
##    1  1 25  0
##    2  0 25  0
##    3 49  0  0
##    4  0  0 50
```

#Comment

Cluster 4 has 50 out of 50 observations from true class 3, which is a perfect match. Cluster 2 has 25 out of 25 observations from true class 2, which is also a perfect match.

### Question 3(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.

#Ans:

```r
# Perform PCA
pca_result <- prcomp(X, center = TRUE, scale. = TRUE)

# Extract the first two principal component score vectors
pc1 <- pca_result$x[,1]
pc2 <- pca_result$x[,2]

# Combine the first two PCs into a new matrix
pc_scores <- cbind(pc1, pc2)

# Perform k-means clustering on the new matrix with K = 3
clus_pca <- kmeans(pc_scores, centers = 3)

# Compare the cluster assignments with the true class labels
table(clus_pca$cluster, true_class)
```

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

#Comment

Cluster 1 has 50 out of 50 observations from true class 3, which is a perfect match. Cluster 2 has 50 out of 50 observations from true class 2, which is also a perfect match. Cluster 3 has 50 out of 50 observations from true class 1, which is another perfect match. In this case, the k-means algorithm has perfectly identified the observations belonging to all three true classes when applied to the first two principal component score vectors. This indicates that the first two principal components have captured the majority of the variance in the dataset, leading to better separation of the classes.

### Question 3(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.

#Ans:

```r
# Standardize the data (mean = 0, sd = 1)
scaled_data <- scale(X)
```

```r
# Perform k-means clustering on the scaled data with K = 3
clus_scaled <- kmeans(scaled_data, centers = 3)

# Compare the cluster assignments with the true class labels
table(clus_scaled$cluster, true_class)
```

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

#Comment

Cluster 2 has 50 out of 50 observations from true class 1, which is a perfect match. Cluster 3 has 49 out of 50 observations from true class 2, which is a good match as well. The k-means algorithm has performed well in identifying the observations belonging to true classes 1, 2, and 3 when applied to the scaled data. Although there are a few misclassified observations, the overall clustering results are quite accurate.

#How do these results compare to those obtained in Q1? Explain. The scaling doesn't change the result in this part. The clustering result perfectly clustered for one observation in class 3.