Stat 432 HW 07

Name: Ahmadreza Eslamminia, netID: ae15

Summer 2024

Include the R code for this HW.

```
knitr::opts_chunk$set(echo = TRUE)
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.3
```

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.3.3
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(tibble)
```

```
## Warning: package 'tibble' was built under R version 4.3.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.3.3
```

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.3.3
```

```
library(jpeg)
library(ggplot2)
library(scatterplot3d)

#add more libraries as needed.
```

Question 1 (K-means clustering)

In this question, we will replicate the image pixels K-means clustering example in our lectures. Perform the following:

(a) Choose a colored `.jpg` image by your own. The resolution should be at least 200×200 but no larger than 800×800. You may reduce the resolution of your image before loading it into R. Plot this image in R.

```
img <- readJPEG("image.jpg")

img_dim <- dim(img)
par(mar=rep(0.2,4))
plot(c(0, img_dim[2]), c(0, img_dim[1]), xaxt='n', yaxt='n', bty='n', pch='', xlab='', ylab='')
rasterImage(img, 0, 0, img_dim[2], img_dim[1])
```

The image is token by myself :))

(b) "vectorize'' this image dataset by converting it to a matrix with dimension: Number of Pixels $\times$ 3, where 3 represents the dimension of (RGB) colors.

```
img_vectorized <- apply(img, 3, c)
dim(img_vectorized)
```

```
## [1] 307200      3
```

```r
head(img_vectorized)
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.3921569 0.7764706 0.5372549
## [2,] 0.3490196 0.7333333 0.4941176
## [3,] 0.3294118 0.7176471 0.4666667
## [4,] 0.3411765 0.7294118 0.4784314
## [5,] 0.3450980 0.7333333 0.4823529
## [6,] 0.3647059 0.7529412 0.5019608
```

(c) Perform k-means with $k = 2, 5, 10, 30, 50, 100$ by treating the pixels as observations. For each k setting, after obtaining the clusters, reconstruct a new image that replaces the pixel colors in each cluster by its cluster mean color. Plot these six new images. Use `par(mfrow = c(2, 3)` to organize your 6 new figures with two rows of three images per row. Label each figure with their corresponding k.

```r
perform_kmeans_and_plot <- function(img_vectorized, k, img_dim) {
  kmeans_fit <- kmeans(img_vectorized, centers=k)
  new_img_vectorized <- kmeans_fit$centers[kmeans_fit$cluster,]

  # Convert  back
  new_img <- array(0, dim=img_dim)
  new_img[,,1] <- matrix(new_img_vectorized[,1], img_dim[1], img_dim[2])
  new_img[,,2] <- matrix(new_img_vectorized[,2], img_dim[1], img_dim[2])
  new_img[,,3] <- matrix(new_img_vectorized[,3], img_dim[1], img_dim[2])

  return(new_img)
}

k_values <- c(2, 5, 10, 30, 50, 100)


par(mfrow=c(2, 3), mar=rep(0.2, 4))
for (k in k_values) {
  new_img <- perform_kmeans_and_plot(img_vectorized, k, img_dim)
  plot(c(0, img_dim[2]), c(0, img_dim[1]), xaxt='n', yaxt='n', bty='n', pch='', xlab='', ylab='')
  rasterImage(new_img, 0, 0, img_dim[2], img_dim[1])
  title(main=paste("k =", k))
}
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 15360000)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 15360000)
```

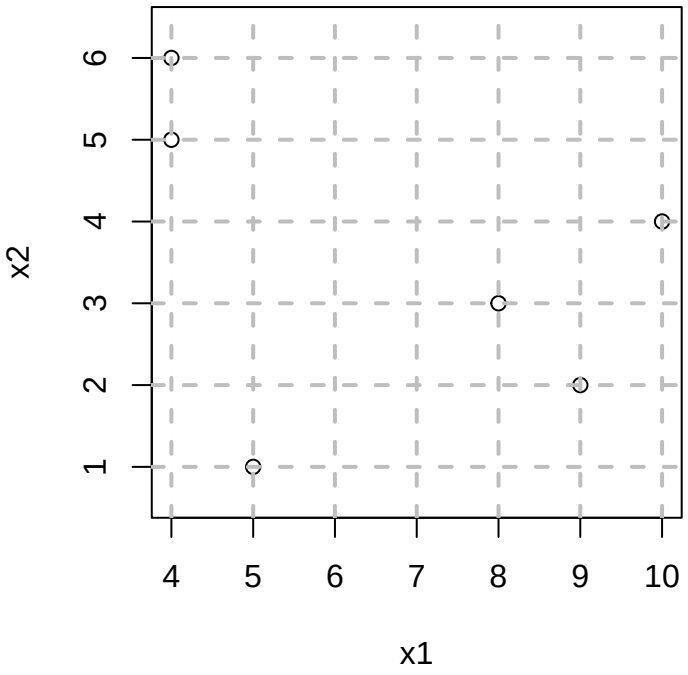
```
## Warning: did not converge in 10 iterations
```


K = 2
K = 5
K = 10
K = 30
K = 50
K = 100

Question 2 (K-means clustering)

In this problem, you will perform K-means clustering *manually*, with K=2, on a small example. The observations are as follows:

Obs	x1	x2	initial.cluster
1	5	1	1
2	9	2	1
3	8	3	2
4	4	6	2
5	4	5	2
6	10	4	2



(a) Compute the centroid for the initial cluster.

```
data <- data.frame(
  Obs = 1:6,
  x1 = c(5, 9, 8, 4, 4, 10),
  x2 = c(1, 2, 3, 6, 5, 4),
  initial.cluster = c(1, 1, 2, 2, 2, 2)
)

#  centroides
centroid1 <- colMeans(data[data$initial.cluster == 1, 2:3])
centroid2 <- colMeans(data[data$initial.cluster == 2, 2:3])

centroid1
```

```
##  x1  x2
## 7.0 1.5
```

```r
centroid2
```

```
##  x1  x2
## 6.5 4.5
```

(b) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```r
euclidean_distance <- function(a, b) {
  sqrt(sum((a - b)^2))
}

# Assign
assign_clusters <- function(data, centroids) {
  distances <- sapply(1:nrow(data), function(i) {
    c(euclidean_distance(data[i, 2:3], centroids[[1]]),
      euclidean_distance(data[i, 2:3], centroids[[2]]))
  })
  new_clusters <- apply(distances, 2, which.min)
  return(new_clusters)
}

# Initial centroides
centroids <- list(centroid1, centroid2)

# Assign clusters
data$cluster <- assign_clusters(data, centroids)

data
```

```
##   Obs x1 x2 initial.cluster cluster
## 1   1  5  1               1       1
## 2   2  9  2               1       1
## 3   3  8  3               2       1
## 4   4  4  6               2       2
## 5   5  4  5               2       2
## 6   6 10  4               2       2
```

(c) Repeat (a) and (b) until the answers obtained stop changing.

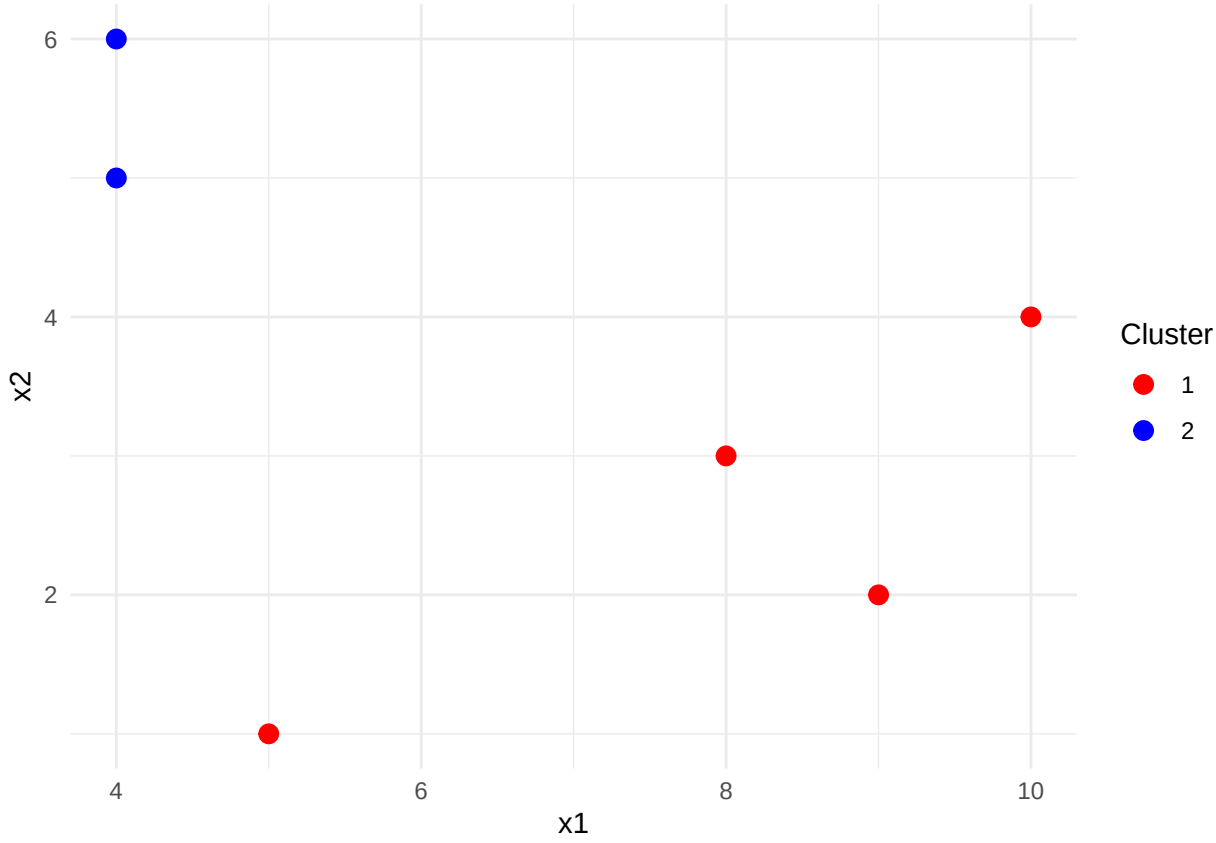
```r
converged <- FALSE

while (!converged) {

  new_centroid1 <- colMeans(data[data$cluster == 1, 2:3])
  new_centroid2 <- colMeans(data[data$cluster == 2, 2:3])

  # Assign
  new_clusters <- assign_clusters(data, list(new_centroid1, new_centroid2))

  if (all(new_clusters == data$cluster)) {
    converged <- TRUE
```

```
  } else {
    data$cluster <- new_clusters
  }

  # Update
  centroids <- list(new_centroid1, new_centroid2)
}

data
```

```
##   Obs x1 x2 initial.cluster cluster
## 1   1  5  1               1       1
## 2   2  9  2               1       1
## 3   3  8  3               2       1
## 4   4  4  6               2       2
## 5   5  4  5               2       2
## 6   6 10  4               2       1
```

(d) Plot the observations according to the cluster labels obtained.

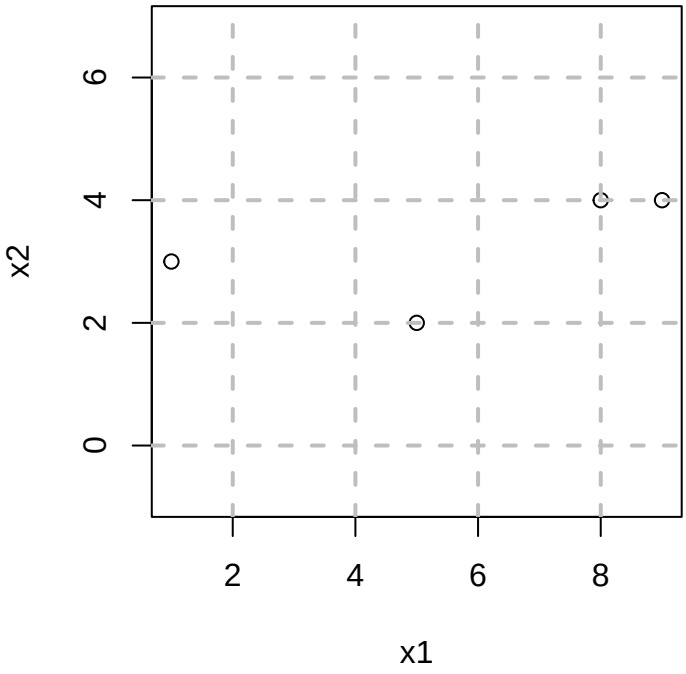
```
library(ggplot2)

ggplot(data, aes(x = x1, y = x2, color = as.factor(cluster))) +
  geom_point(size = 3) +
  scale_color_manual(values = c('red', 'blue')) +
  labs(color = "Cluster") +
  theme_minimal()
```



Question 3 (Hierarchical clustering)

In this problem, you will perform hierarchical clustering *manually* on a small example. The observations are as follows:

Obs	x1	x2
1	1	3
2	5	2
3	9	4
4	8	4



(a) Conduct the hierarchical clustering based on complete linkage. For each step of the hierarchical clustering, report the new fusion and the corresponding dissimilarity (distance).

```
data <- data.frame(
  Obs = 1:4,
  x1 = c(1, 5, 9, 8),
  x2 = c(3, 2, 4, 4)
)

# distance matrix
dist_matrix <- dist(data[,2:3])


hc_complete <- hclust(dist_matrix, method = "complete")

cat("Complete linkages Fusions:\n")
```

```
## Complete linkages Fusions:
```

```r
print(hc_complete$merge)
```

```
##      [,1] [,2]
## [1,]   -3   -4
## [2,]   -1   -2
## [3,]    1    2
```

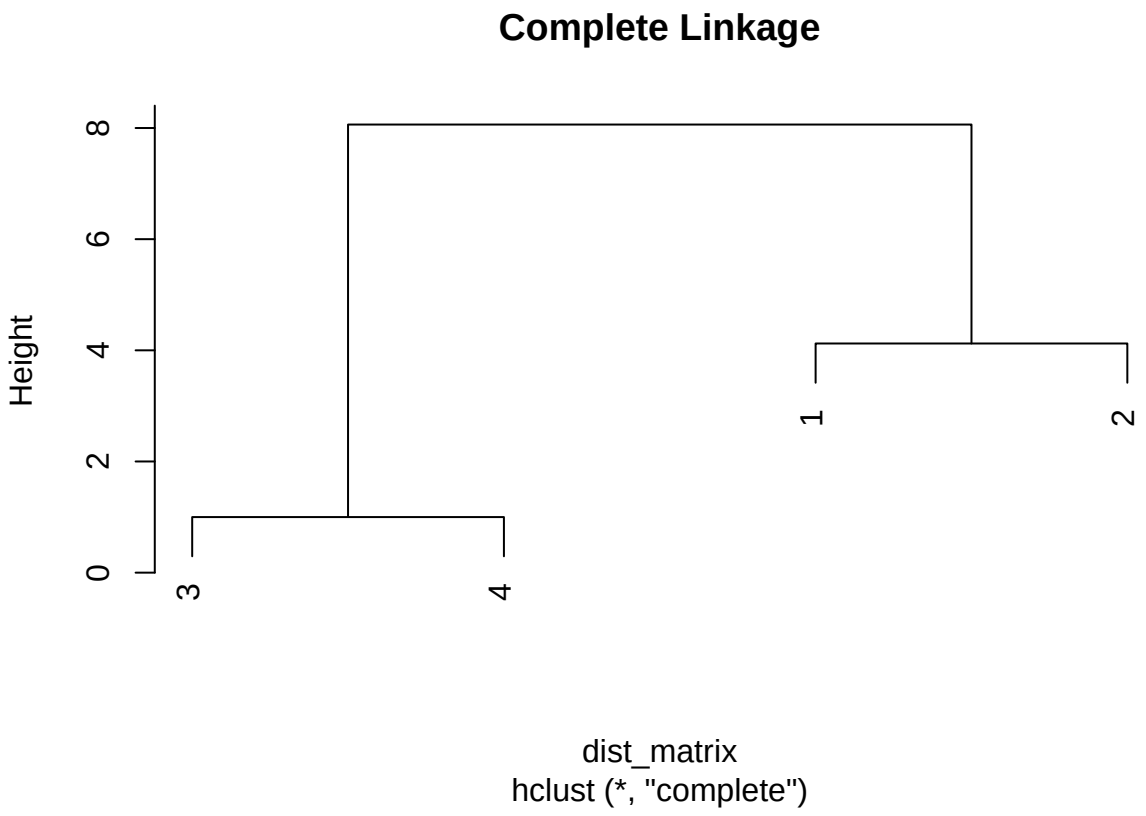
```r
cat("\nComplete Linkage Disimilarites:\n")
```

```
##
## Complete Linkage Disimilarites:
```

```r
print(hc_complete$height)
```

```
## [1] 1.000000 4.123106 8.062258
```

```r
# dendrograms
plot(hc_complete, main = "Complete Linkage")
```



(b) Conduct the hierarchical clustering based on single linkage. For each step of the hierarchical clustering, report the new fusion and the corresponding dissimilarity (distance).

```r
hc_single <- hclust(dist_matrix, method = "single")

cat("Single Linkage Fusions:\n")
```

```
## Single Linkage Fusions:
```

```r
print(hc_single$merge)
```

```
##      [,1] [,2]
## [1,]   -3   -4
## [2,]   -2    1
## [3,]   -1    2
```

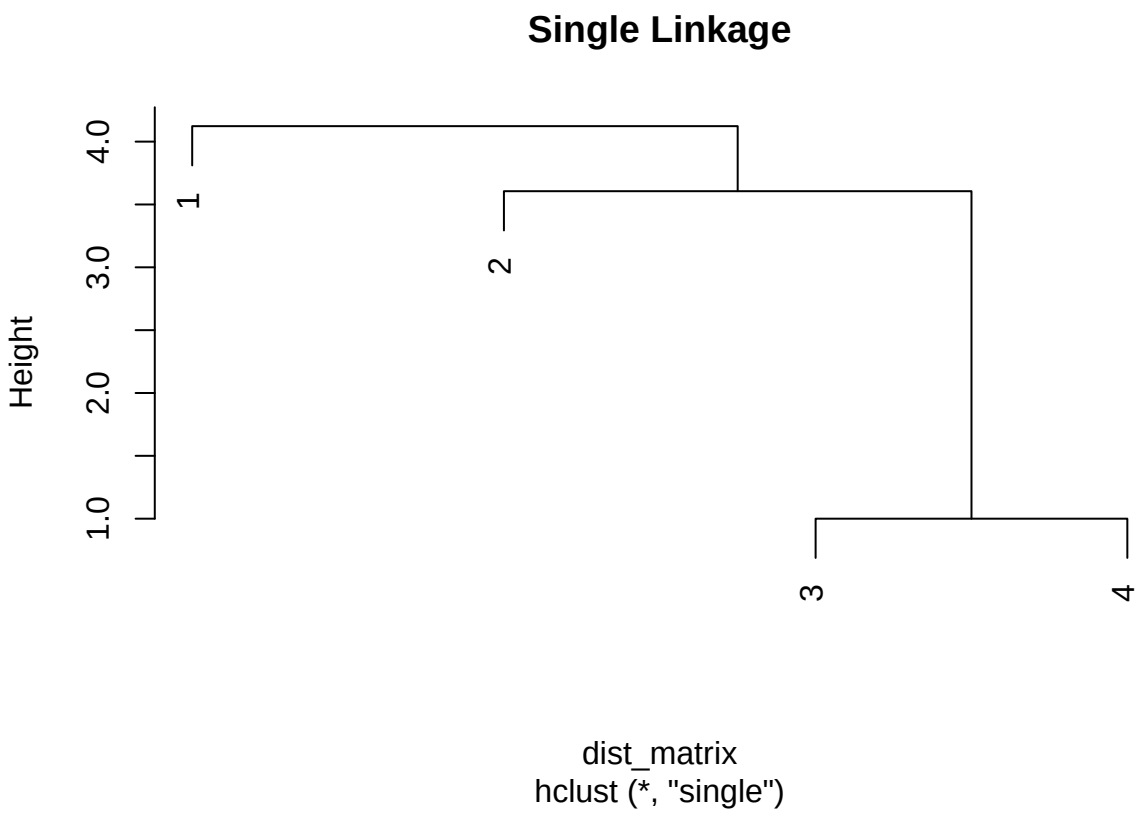
```r
cat("\nSingle Linkage Dissimilarities:\n")
```

```
##
## Single Linkage Dissimilarities:
```

```r
print(hc_single$height)
```

```
## [1] 1.000000 3.605551 4.123106
```

```r
plot(hc_single, main = "Single Linkage")
```



Question 4 (PCA)

(a) Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(123)


n <- 20
p <- 50

# Generate
class1 <- matrix(rnorm(n * p, mean = 0), nrow = n, ncol = p)
class2 <- matrix(rnorm(n * p, mean = 3), nrow = n, ncol = p)
class3 <- matrix(rnorm(n * p, mean = 6), nrow = n, ncol = p)

# Combine
data <- rbind(class1, class2, class3)

# class labeling
labels <- factor(rep(1:3, each = n))


head(data)
```

```
##             [,1]       [,2]       [,3]       [,4]         [,5]        [,6]
## [1,] -0.56047565 -1.0678237 -0.6947070  0.3796395  0.005764186 -0.71040656
## [2,] -0.23017749 -0.2179749 -0.2079173 -0.5023235  0.385280401  0.25688371
## [3,]  1.55870831 -1.0260044 -1.2653964 -0.3332074 -0.370660032 -0.24669188
## [4,]  0.07050839 -0.7288912  2.1689560 -1.0185754  0.644376549 -0.34754260
## [5,]  0.12928774 -0.6250393  1.2079620 -1.0717912 -0.220486562 -0.95161857
## [6,]  1.71506499 -1.6866933 -1.1231086  0.3035286  0.331781964 -0.04502772
##            [,7]       [,8]       [,9]      [,10]      [,11]       [,12]
## [1,]  0.1176466  0.7017843  1.0527115 -1.0633261  2.1988103 -0.57397348
## [2,] -0.9474746 -0.2621975 -1.0491770  1.2631852  1.3124130  0.61798582
## [3,] -0.4905574 -1.5721442 -1.2601552 -0.3496504 -0.2651451  1.10984814
## [4,] -0.2560922 -1.5146677  3.2410399 -0.8655129  0.5431941  0.70758835
## [5,]  1.8438620 -1.6015362 -0.4168576 -0.2362796 -0.4143399 -0.36365730
## [6,] -0.6519499 -0.5309065  0.2982276 -0.1971759 -0.4762469  0.05974994
##           [,13]       [,14]      [,15]      [,16]      [,17]       [,18]
## [1,] -0.7886220 -0.52111732 -1.6674751 -0.7152422  0.2374303  0.62418747
## [2,] -0.5021987 -0.48987045  0.7364960 -0.7526890  1.2181086  0.95900538
## [3,]  1.4960607  0.04715443  0.3860266 -0.9385387 -1.3387743  1.67105483
## [4,] -1.1373036  1.30019868 -0.2656516 -1.0525133  0.6608203  0.05601673
## [5,] -0.1790516  2.29307897  0.1181445 -0.4371595 -0.5229124 -0.05198191
## [6,]  1.9023618  1.54758106  0.1340386  0.3311792  0.6837455 -1.75323736
##           [,19]       [,20]       [,21]      [,22]      [,23]      [,24]
## [1,] -0.2052993  0.03455107 -0.07355602  1.7795029 -1.2410445  0.8719650
## [2,]  0.6511933  0.19023032 -1.16865142  0.2864244  0.4547693 -0.3484724
## [3,]  0.2737665  0.17472640 -0.63474826  0.1263159  0.6599026  0.5185038
## [4,]  1.0246732 -1.05501704 -0.02884155  1.2722668 -0.1998898 -0.3906850
## [5,]  0.8176594  0.47613328  0.67069597 -0.7184662 -0.6451140 -1.0927872
## [6,] -0.2097932  1.37857014 -1.65054654 -0.4503386  0.1653210  1.2100105
##           [,25]       [,26]      [,27]      [,28]      [,29]      [,30]
```

```
## [1,] 0.02045071 -0.60189285  0.9672673 -0.2732481 -0.8338436 -0.3164159
## [2,] 0.31405766 -0.99369859 -0.1082801 -0.4686998  0.5787224 -0.1023465
## [3,] 1.32821470  1.02678506 -0.6984207  0.7041673 -1.0875807 -1.1815592
## [4,] 0.12131838  0.75106130 -0.2759452 -1.1973635  1.4840309  0.4986580
## [5,] 0.71284232 -1.50916654  1.1146485  0.8663661 -1.1862066 -1.0389564
## [6,] 0.77886003 -0.09514745  0.5500440  0.8641525  0.1010792 -0.2262220
##            [,31]      [,32]      [,33]       [,34]      [,35]      [,36]
## [1,]  1.07401226 -1.4152819 -0.8341882  0.86364843  0.3823051 -0.7282191
## [2,] -0.02734697  0.3183903  0.2710668  1.38051453  0.9821130 -1.5404424
## [3,] -0.03333034  0.8464363  0.1573533  1.96624802 -0.7273835 -0.6930946
## [4,] -1.51606762  0.1781902  0.6297117 -0.02839505 -0.9968390  0.1188494
## [5,]  0.79038534 -0.8752555 -0.3957980 -2.24905109 -1.0416889 -1.3647095
## [6,] -0.21073418  0.9411658  0.8993541  0.03152600 -0.4145887  0.5899827
##             [,37]      [,38]      [,39]        [,40]      [,41]       [,42]
## [1,] -1.041673294  0.7116019  0.5397906  0.813400374  0.3562833 -1.22898618
## [2,] -1.728304515  0.9902622  0.6164557  0.904435464 -0.6580102  0.74229702
## [3,]  0.641830028  2.3829267  0.6165678  0.002691661  0.8552022 -0.08291994
## [4,] -1.529310531  0.6644159 -1.6921015 -1.176692158  1.1529362  0.78981792
## [5,]  0.001683688  0.2073812  0.3687421 -1.318220727  0.2762746 -0.26770642
## [6,]  0.250247821 -2.2106331  0.9678592 -0.592997366  0.1441047 -0.59189210
##            [,43]      [,44]      [,45]      [,46]       [,47]      [,48]
## [1,] -0.75939812  0.9112097  0.4770372 -1.0141142 -0.41401586  1.0490866
## [2,]  2.68485900  0.1424584 -0.9719942 -0.7913139 -0.21215053 -0.3599751
## [3,] -0.45839014 -1.3894835 -0.1852022  0.2995937 -0.03653722 -1.6859165
## [4,]  0.06424356 -0.8660377  1.2209637  1.6390519  0.36501875 -0.8445834
## [5,]  0.64979187 -0.1632849  0.5412841  1.0846170  0.66515988 -0.4577605
## [6,] -0.02601863  2.5530261  0.4573573 -0.6245675  1.31782088  0.1036380
##           [,49]      [,50]
## [1,] 0.78417088  2.3050620
## [2,] 2.29961936 -1.1246037
## [3,] 0.15670299 -0.3054696
## [4,] 0.04673353 -0.5167594
## [5,] 0.09658583  1.5123954
## [6,] 0.06976623 -0.7694849
```

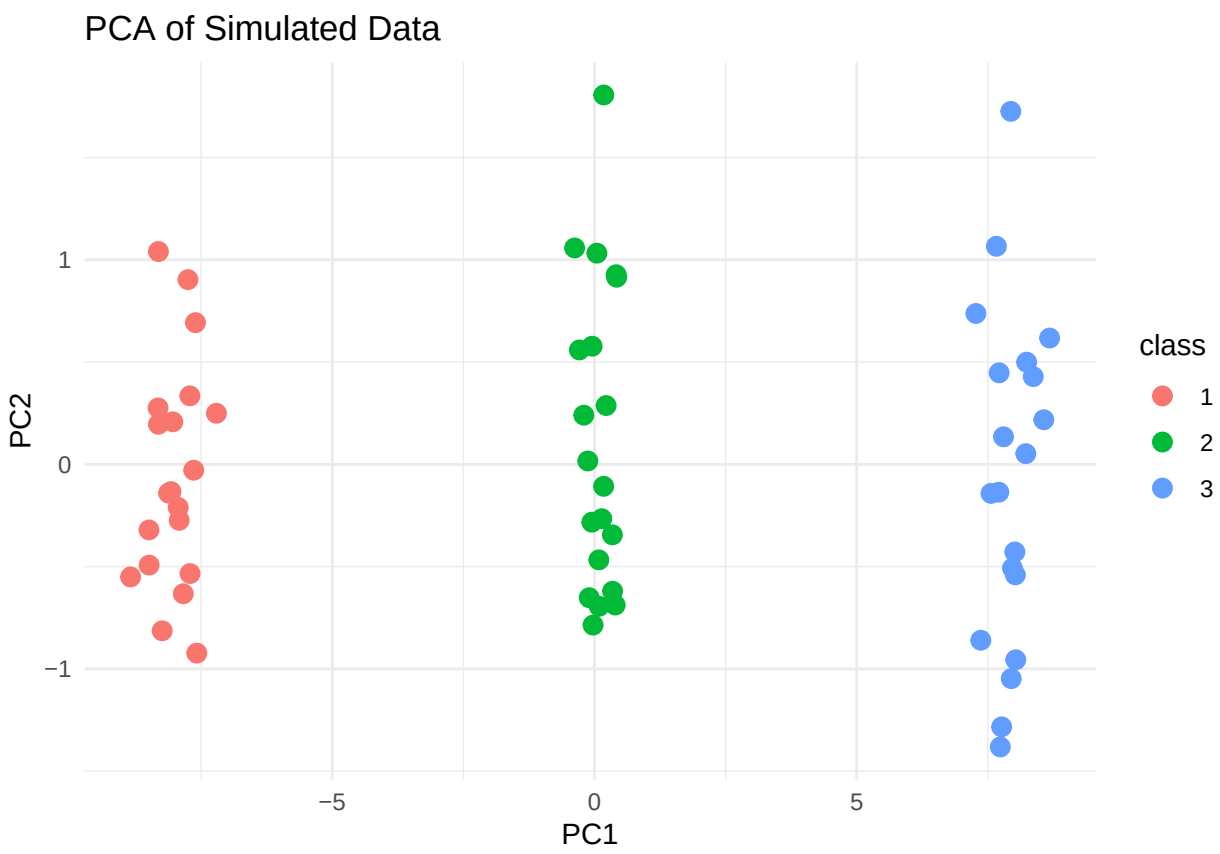
Hint: There are a number of functions in R that you can use to generate data. One example is the `rnorm()` function; `runif()` is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

(b) Perform PCA on the 60 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 do not appear separated in this plot, then return to part (a) and modify the simulation so that there is greater separation between the three classes.

```r
pca_result <- prcomp(data, scale. = TRUE)


pca_data <- data.frame(pca_result$x, class = labels)


ggplot(pca_data, aes(x = PC1, y = PC2, color = class)) +
  geom_point(size = 3) +
```

```
  labs(title = "PCA of Simulated Data", x = "PC1", y = "PC2") +
  theme_minimal()
```



‘‘‘