HW 1 - 521

Emma Schmidt

9/7/2021

Question 1: Basics of SVD

Generate a random matrix M of size n×n for n in (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048). And then plot the time taken to (i) to generate this matrix, and (ii) to compute the SVD of the matrix as n increases. Note that you need to plot two figures. You may want to use Sys.time for the same. DO NOT report the SVD values, just plot the time as a function of n. Do you see some scaling with n? Justify your observations. It may be useful to plot t.

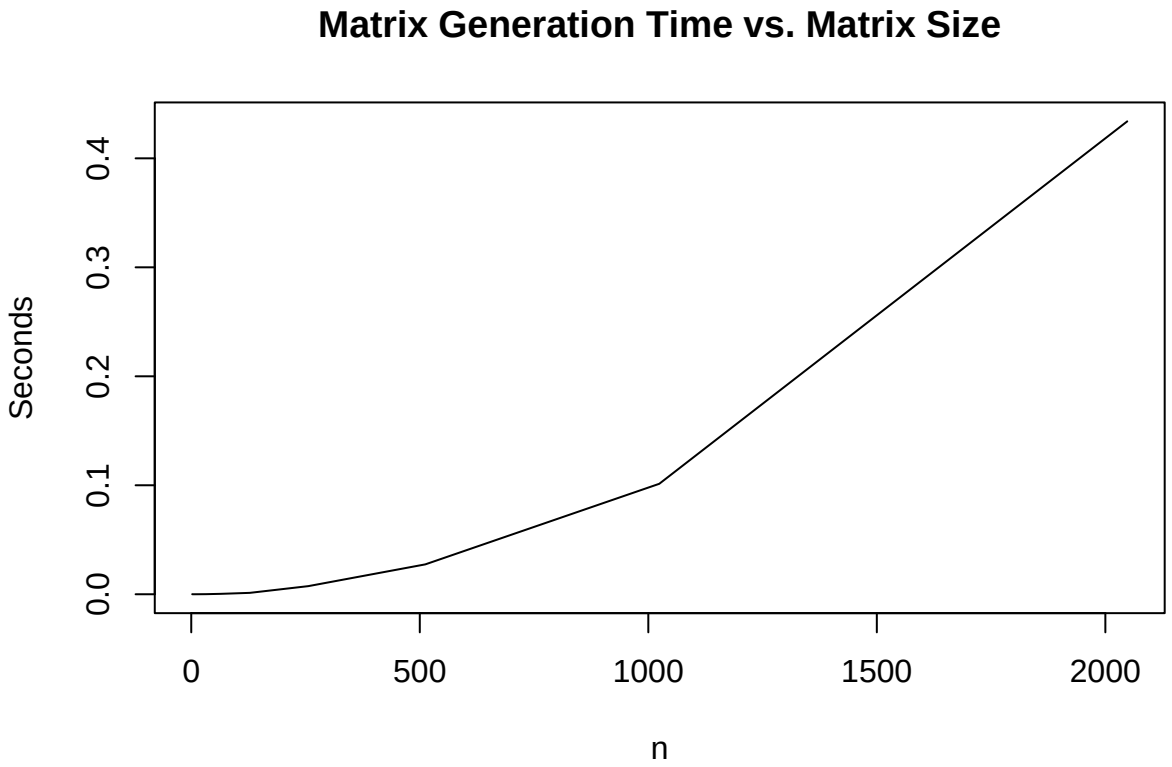
i) Time to generate matrix

```
set.seed(45)

n <- c(2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048)
v <- c()

for(i in seq_along(n)){
  start <- Sys.time()
  M <- matrix(rnorm((n[i])^2), nrow = n[i], ncol = n[i])
  end <- Sys.time()
  v[i] <- end - start
}

plot(n, v, type = "l", xlab = "n", ylab = "Seconds",
     main = "Matrix Generation Time vs. Matrix Size")
```



As n increases the time to generate the matrix roughly increases exponentially.

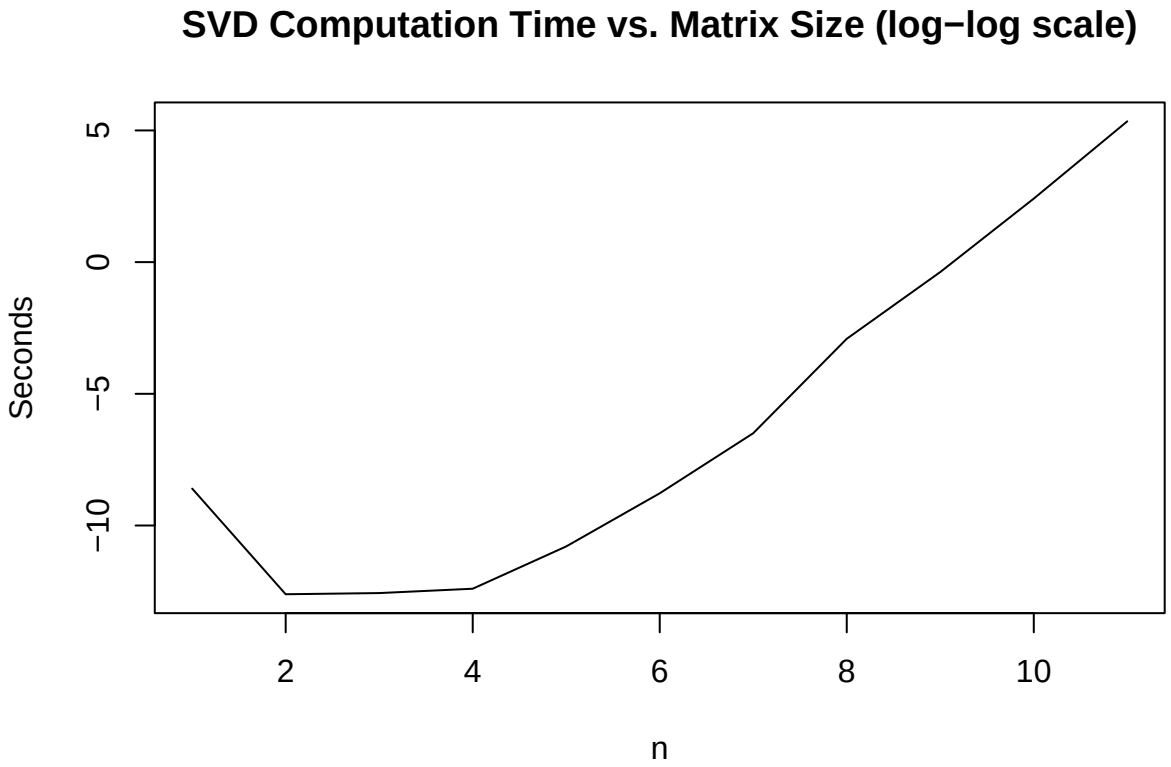
ii) Time to compute SVD

```
set.seed(45)

n <- c(2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048)
v <- c()

for(i in seq_along(n)){
  start <- Sys.time()
  M <- matrix(rnorm((n[i])^2), nrow = n[i], ncol = n[i])
  svd(M)
  end <- Sys.time()
  v[i] <- end - start
}

plot(log2(n), log2(v), type = "l", xlab = "n", ylab = "Seconds",
     main = "SVD Computation Time vs. Matrix Size (log-log scale)")
```



Consistent with our findings above, as n increases, the time to execute SVD exponentially increases.

Question 2: Power Method

2.1 First Eigenvector: Write code that implements the full procedure of the power method to find the largest eigenvalue and its corresponding eigenvector. Use your code to find the largest eigenvalue of the following matrix.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & -1 & 4 \\ 3 & 4 & -5 \end{bmatrix}$$

Compare results with those provided by eigen()

```
set.seed(45)

power_method <- function(matrix) {
  data <- rnorm(nrow(matrix))
  w <- data
  sk_new <- 1
  sk_old <- 0
  vector <- matrix %*% w
  while(abs(sk_new - sk_old) > .00001) {
    sk_new <- vector[which.max(abs(vector))]
    sk_old <- sk_new
    w_new <- (matrix %*% w)/sk_new
    w <- w_new
    vector <- matrix %*% w
```

```
    sk_new <- vector[which.max(abs(vector))]
  }
  return(c(w_new, sk_new))
}

A <- matrix(c(1, 2, 3, 2, -1, 4, 3, 4, -5), ncol = 3, nrow = 3)
power_method(A)
```

```
## [1] -0.2223262 -0.5253032  1.0000000 -7.7681913
```

```
eigen(A)
```

```
## eigen() decomposition
## $values
## [1]  4.610843 -1.842654 -7.768189
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] -0.6890036  0.6985555 -0.1931172
## [2,] -0.5672220 -0.6856083 -0.4562899
## [3,] -0.4511466 -0.2048451  0.8686226
```

2.2 Deflation and Power Method:

$$B = \begin{bmatrix} 5 & 1 & 0 \\ 1 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

a) Apply your power method on matrix B to get an approximation of the first eigenvector and eigenvalue

```
set.seed(45)

B <- matrix(c(5, 1, 0, 1, 4, 0, 0, 0, 1), 3, 3)
power_method(B)
```

```
## [1] 1.000000e+00 6.180490e-01 3.223777e-20 5.618049e+00
```

```
eigen(B)
```

```
## eigen() decomposition
## $values
## [1] 5.618034 3.381966 1.000000
##
## $vectors
##           [,1]       [,2] [,3]
## [1,] 0.8506508  0.5257311    0
## [2,] 0.5257311 -0.8506508    0
## [3,] 0.0000000  0.0000000    1
```

b) Deflate matrix B and apply the power method on the residual matrix B1 to obtain the second eigenvalue and eigenvector

```r
set.seed(45)

deflation <- function(matrix) {
  w_new <- power_method(matrix)[1:nrow(matrix)]
  sk_new <- power_method(matrix)[nrow(matrix)+1]
  deflation_matrix <- matrix - (sk_new * w_new %*% t(w_new))
  deflation_matrix
}

power_method(deflation(B))
```

```
## [1] -6.180563e-01  1.000000e+00  1.882614e-17  3.381969e+00
```

```r
eigen(deflation(B))
```

```
## eigen() decomposition
## $values
## [1]  3.381966  1.000000 -2.146002
##
## $vectors
##               [,1]         [,2]          [,3]
## [1,]  5.257419e-01 0.000000e+00  8.506441e-01
## [2,] -8.506441e-01 4.198509e-18  5.257419e-01
## [3,]  3.571437e-18 1.000000e+00 -2.207332e-18
```

c) Deflate the matrix B1 again and apply the power method to obtain the third eigenvalue and eigenvector

```r
set.seed(45)

power_method(deflation(deflation(B)))
```

```
## [1]  1.000000e+00  6.180827e-01 -1.245997e-07 -2.146031e+00
```

```r
eigen(deflation(deflation(B)))
```

```
## eigen() decomposition
## $values
## [1]  1.000000 -1.291721 -2.145783
##
## $vectors
##               [,1]          [,2]         [,3]
## [1,]  2.224974e-15  5.257137e-01 8.506616e-01
## [2,] -4.109844e-15 -8.506616e-01 5.257137e-01
## [3,]  1.000000e+00 -4.665786e-15 2.679010e-16
```

Question 3: Principle Component Analysis

For this question use the USArrests dataset

a) Use apply() function to compute the mean and variance of all four columns

```
apply(USArrests, 2, mean)
```

```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
apply(USArrests, 2, var)
```

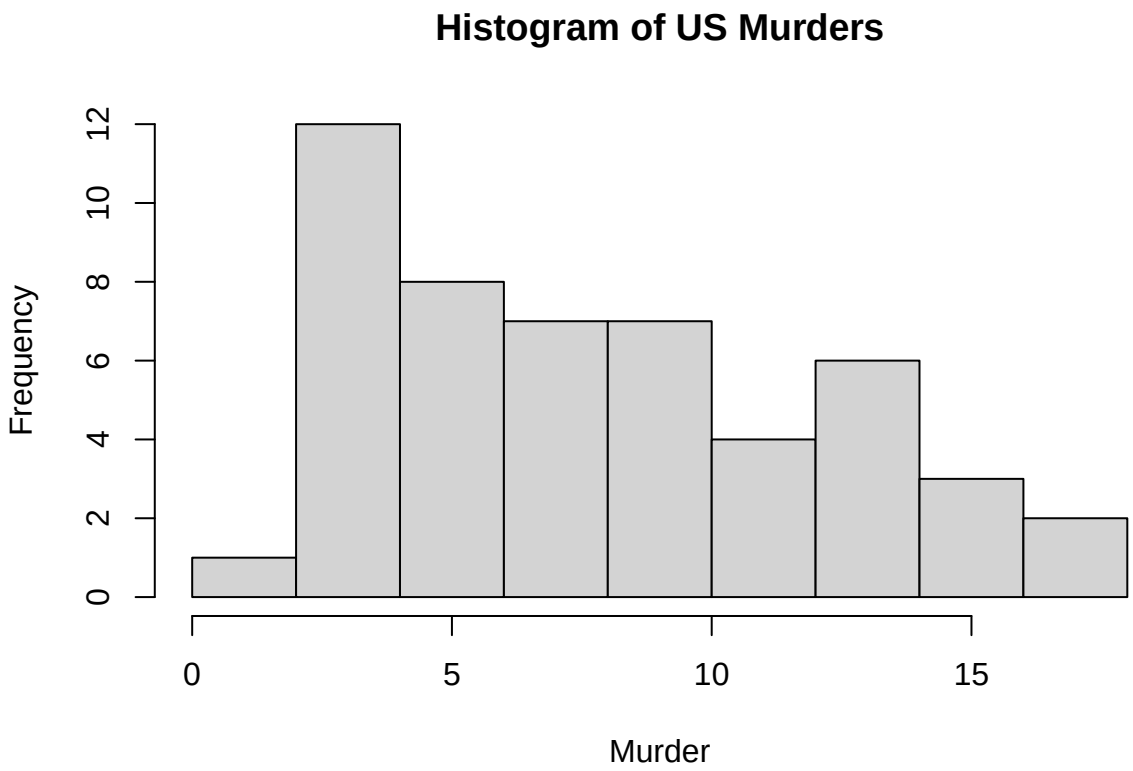
```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

```
apply(USArrests, 2, sd)
```

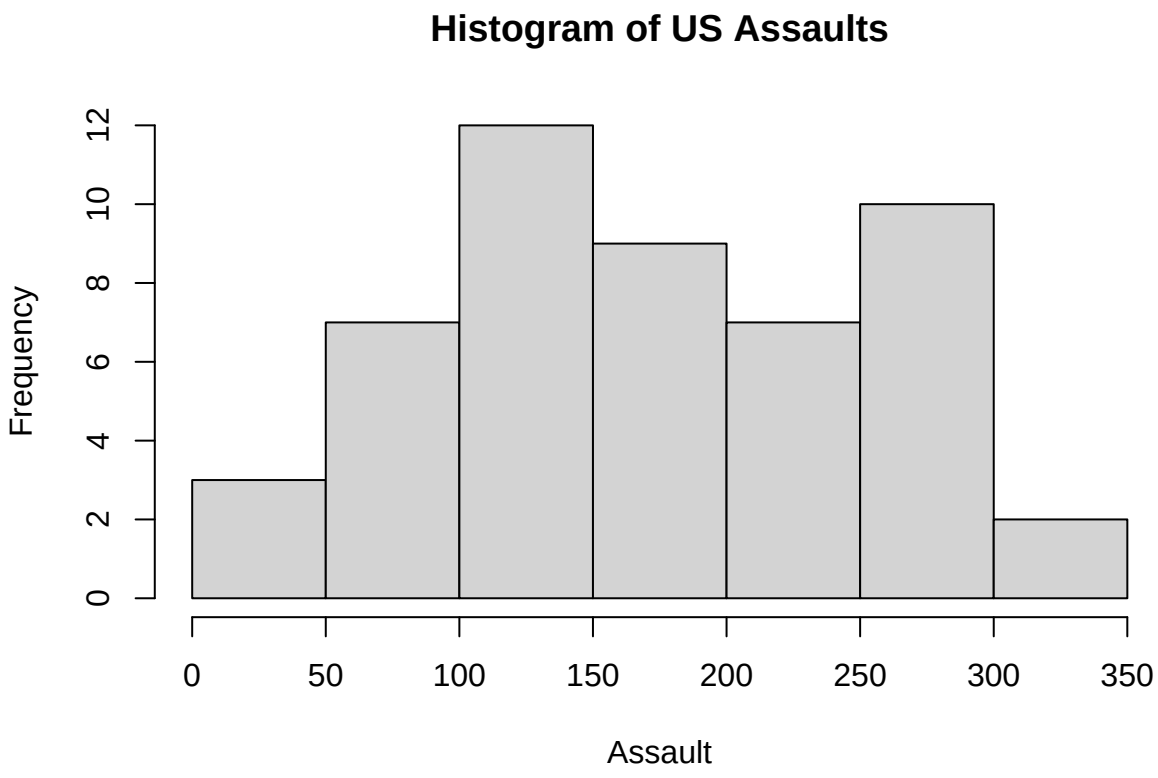
```
##    Murder   Assault  UrbanPop      Rape
##  4.355510 83.337661 14.474763  9.366385
```

b) Plot a histogram for each of the four columns

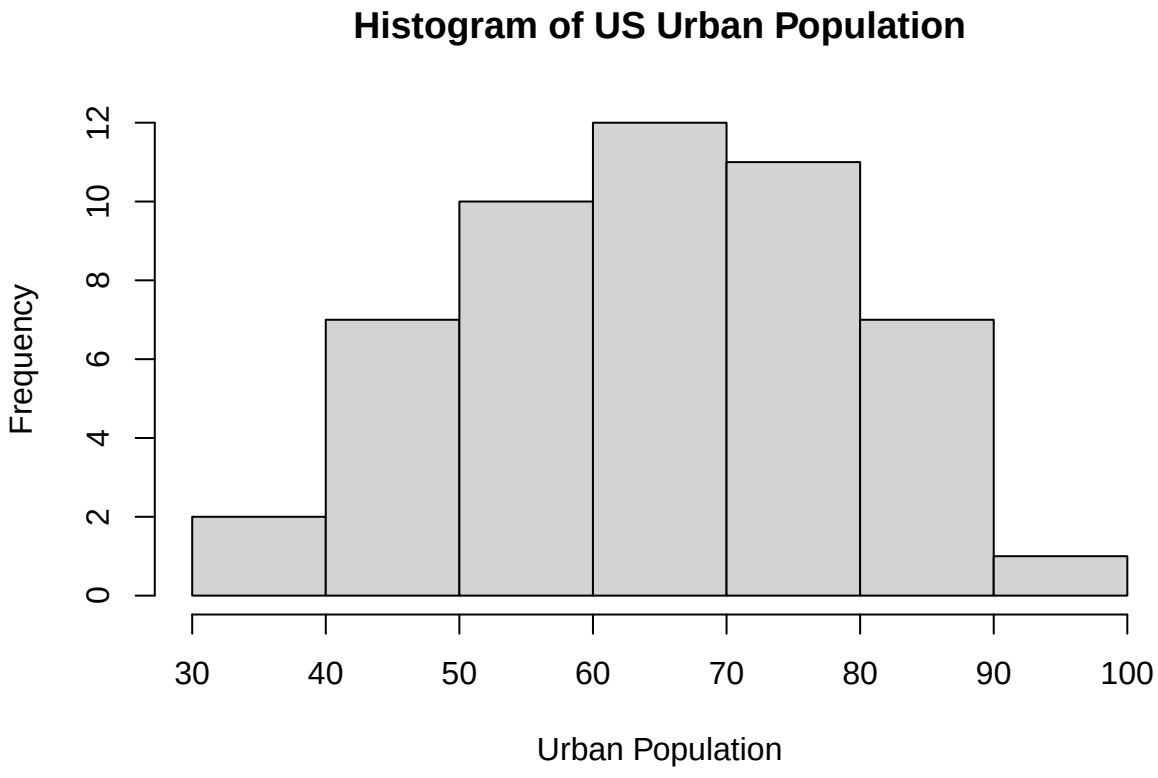
```
hist(USArrests$Murder, xlab = "Murder", main = "Histogram of US Murders")
```



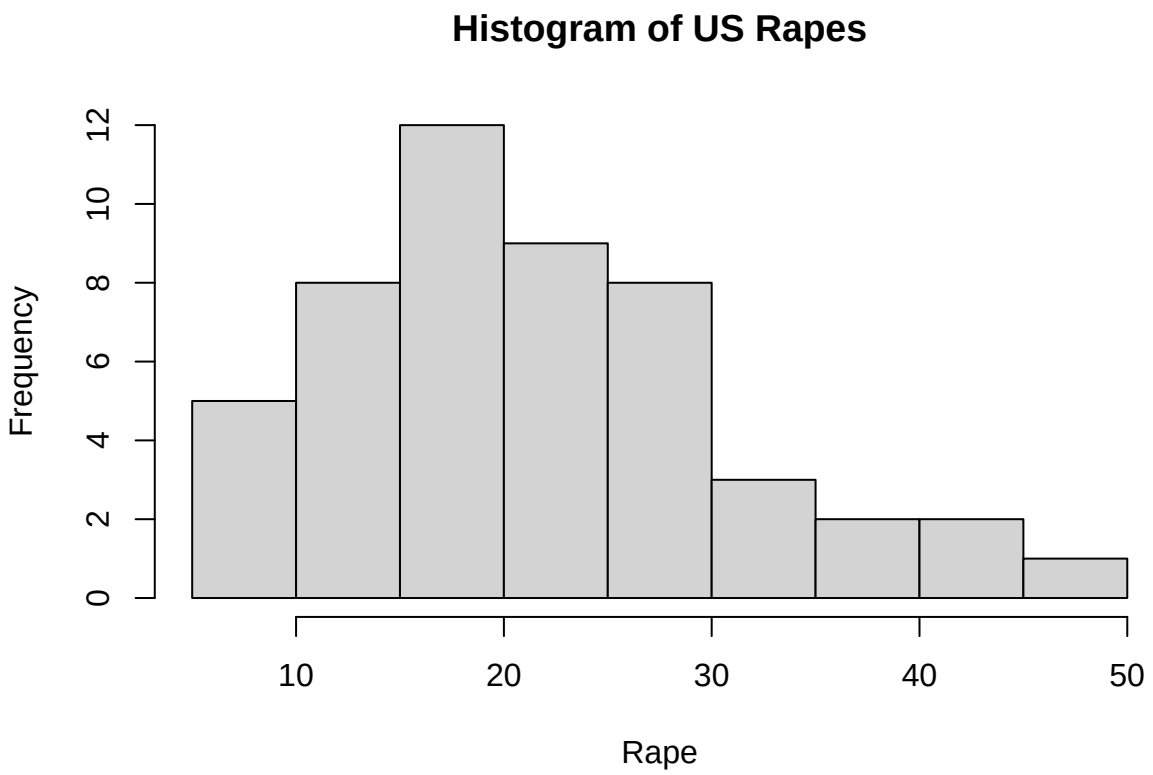

```
hist(USArrests$Assault, xlab = "Assault", main = "Histogram of US Assaults")
```




```
hist(USArrests$UrbanPop, xlab = "Urban Population", main = "Histogram of US Urban Population")
```

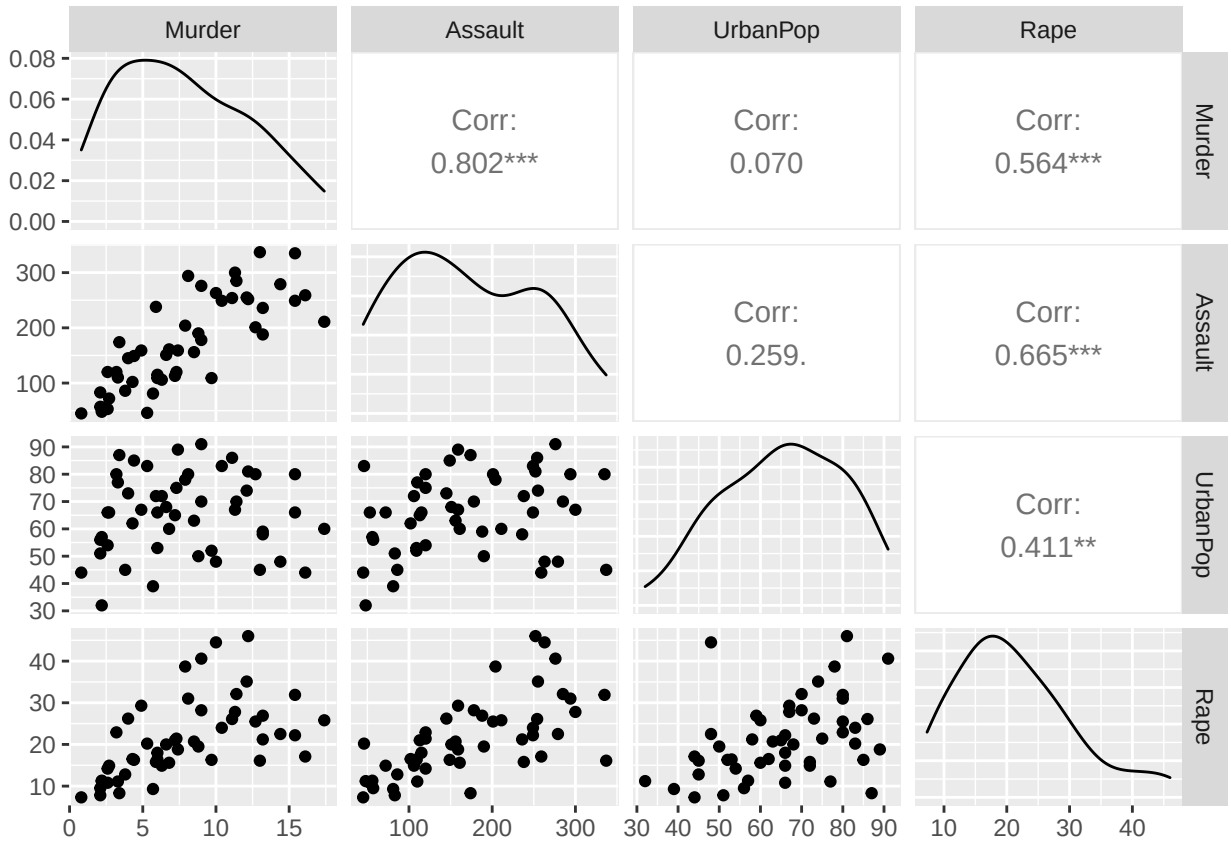


```
hist(USArrests$Rape, xlab = "Rape", main = "Histogram of US Rapes")
```



c) Explore and plot any correlations between the four columns

```
ggpairs(USArrests, progress = F)
```



Assault and murder show a strong, positive correlation. Rape vs murder and rape vs assault both show moderate positive correlation. Urban population looks to have some positive correlation with rape, but less correlation with both murder and assault.

d) Use prcomp() function to perform principle component analysis. Make sure you standardize the data matrix. Print a summary at the end

```
arrests_pca <- prcomp(USArrests, scale = TRUE)
summary(arrests_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

e) Obtain the principal vectors and store them in a matrix, include row and column names. Display first three loadings

```
loadings <- arrests_pca$rotation
loadings[,1:3]
```

```
##                 PC1        PC2        PC3
## Murder   -0.5358995  0.4181809 -0.3412327
## Assault  -0.5831836  0.1879856 -0.2681484
## UrbanPop -0.2781909 -0.8728062 -0.3780158
## Rape     -0.5434321 -0.1673186  0.8177779
```

f) Obtain principle components and store them in a matrix, include row and column names. Display first three PCs

```
scores <- arrests_pca$x
scores[,1:3]
```

```
##                       PC1         PC2         PC3
## Alabama        -0.97566045  1.12200121 -0.43980366
## Alaska         -1.93053788  1.06242692  2.01950027
## Arizona        -1.74544285 -0.73845954  0.05423025
## Arkansas        0.13999894  1.10854226  0.11342217
## California     -2.49861285 -1.52742672  0.59254100
## Colorado       -1.49934074 -0.97762966  1.08400162
## Connecticut     1.34499236 -1.07798362 -0.63679250
## Delaware       -0.04722981 -0.32208890 -0.71141032
## Florida        -2.98275967  0.03883425 -0.57103206
## Georgia        -1.62280742  1.26608838 -0.33901818
## Hawaii          0.90348448 -1.55467609  0.05027151
## Idaho           1.62331903  0.20885253  0.25719021
## Illinois       -1.36505197 -0.67498834 -0.67068647
## Indiana         0.50038122 -0.15003926  0.22576277
## Iowa            2.23099579 -0.10300828  0.16291036
## Kansas          0.78887206 -0.26744941  0.02529648
## Kentucky        0.74331256  0.94880748 -0.02808429
## Louisiana      -1.54909076  0.86230011 -0.77560598
## Maine           2.37274014  0.37260865 -0.06502225
## Maryland       -1.74564663  0.42335704 -0.15566968
## Massachusetts   0.48128007 -1.45967706 -0.60337172
## Michigan       -2.08725025 -0.15383500  0.38100046
## Minnesota       1.67566951 -0.62590670  0.15153200
## Mississippi    -0.98647919  2.36973712 -0.73336290
## Missouri       -0.68978426 -0.26070794  0.37365033
## Montana         1.17353751  0.53147851  0.24440796
## Nebraska        1.25291625 -0.19200440  0.17380930
## Nevada         -2.84550542 -0.76780502  1.15168793
## New Hampshire   2.35995585 -0.01790055  0.03648498
## New Jersey     -0.17974128 -1.43493745 -0.75677041
## New Mexico     -1.96012351  0.14141308  0.18184598
## New York       -1.66566662 -0.81491072 -0.63661186
## North Carolina -1.11208808  2.20561081 -0.85489245
## North Dakota    2.96215223  0.59309738  0.29824930
## Ohio            0.22369436 -0.73477837 -0.03082616
## Oklahoma        0.30864928 -0.28496113 -0.01515592
## Oregon         -0.05852787 -0.53596999  0.93038718
## Pennsylvania    0.87948680 -0.56536050 -0.39660218
## Rhode Island    0.85509072 -1.47698328 -1.35617705
## South Carolina -1.30744986  1.91397297 -0.29751723
## South Dakota    1.96779669  0.81506822  0.38538073
## Tennessee      -0.98969377  0.85160534  0.18619262
## Texas          -1.34151838 -0.40833518 -0.48712332
## Utah            0.54503180 -1.45671524  0.29077592
## Vermont         2.77325613  1.38819435  0.83280797
## Virginia        0.09536670  0.19772785  0.01159482
## Washington      0.21472339 -0.96037394  0.61859067
```

```
## West Virginia   2.08739306  1.41052627  0.10372163
## Wisconsin       2.05881199 -0.60512507 -0.13746933
## Wyoming         0.62310061  0.31778662 -0.23824049
```

g) Obtain the eigenvalues and store them as a vector. Display the entire vector, and compute their sum

```
eigenvalues <- arrests_pca$sd^2
e_vector <- c(eigenvalues)
e_vector
```

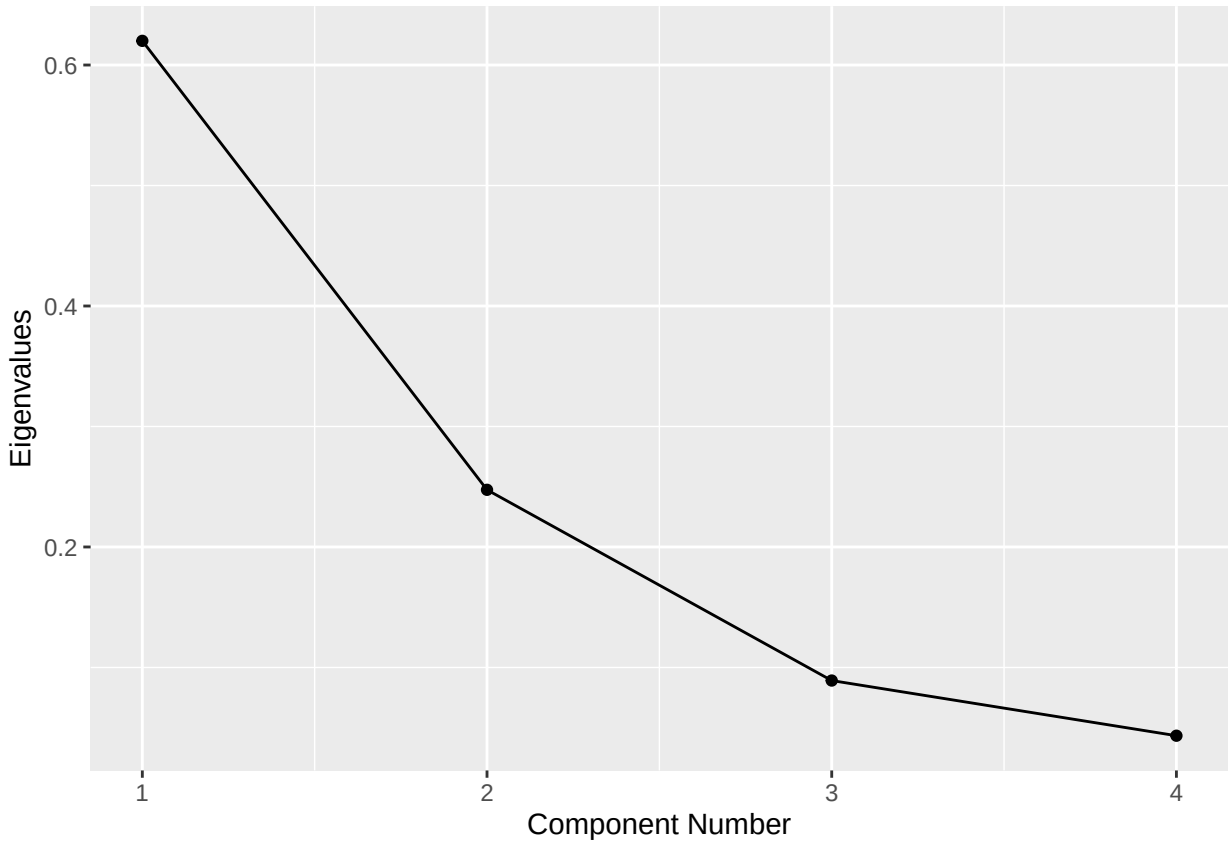
```
## [1] 2.4802416 0.9897652 0.3565632 0.1734301
```

```
sum(e_vector)
```

```
## [1] 4
```

h) Creat a scree-plot of the eigenvalues

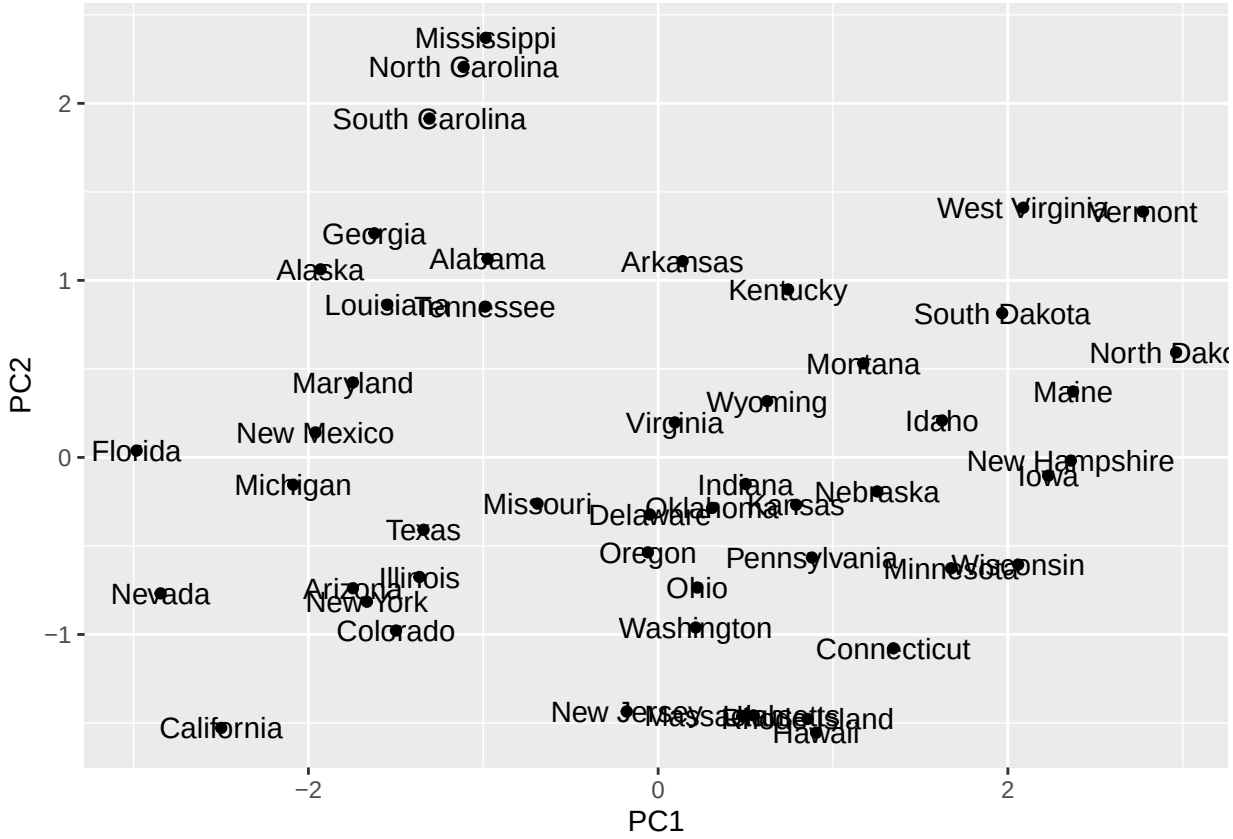
```
ggplot() + geom_point(aes(x = 1:length(eigenvalues), y=eigenvalues/sum(eigenvalues))) +
labs(x = "Component Number", y = "Eigenvalues") + geom_line(aes(x = 1:length(eigenvalues), y=eigenvalues
```



In scree plots the elbow is of interest. The elbow appears to occur roughly at component 3. We can interpret this as the place where most of the variance has been explained, and further components don't offer a ton of significance.

i) Create a scatterplot based on the 1st and 2nd PCs. Which state stands out? How do you read/interpret this chart?

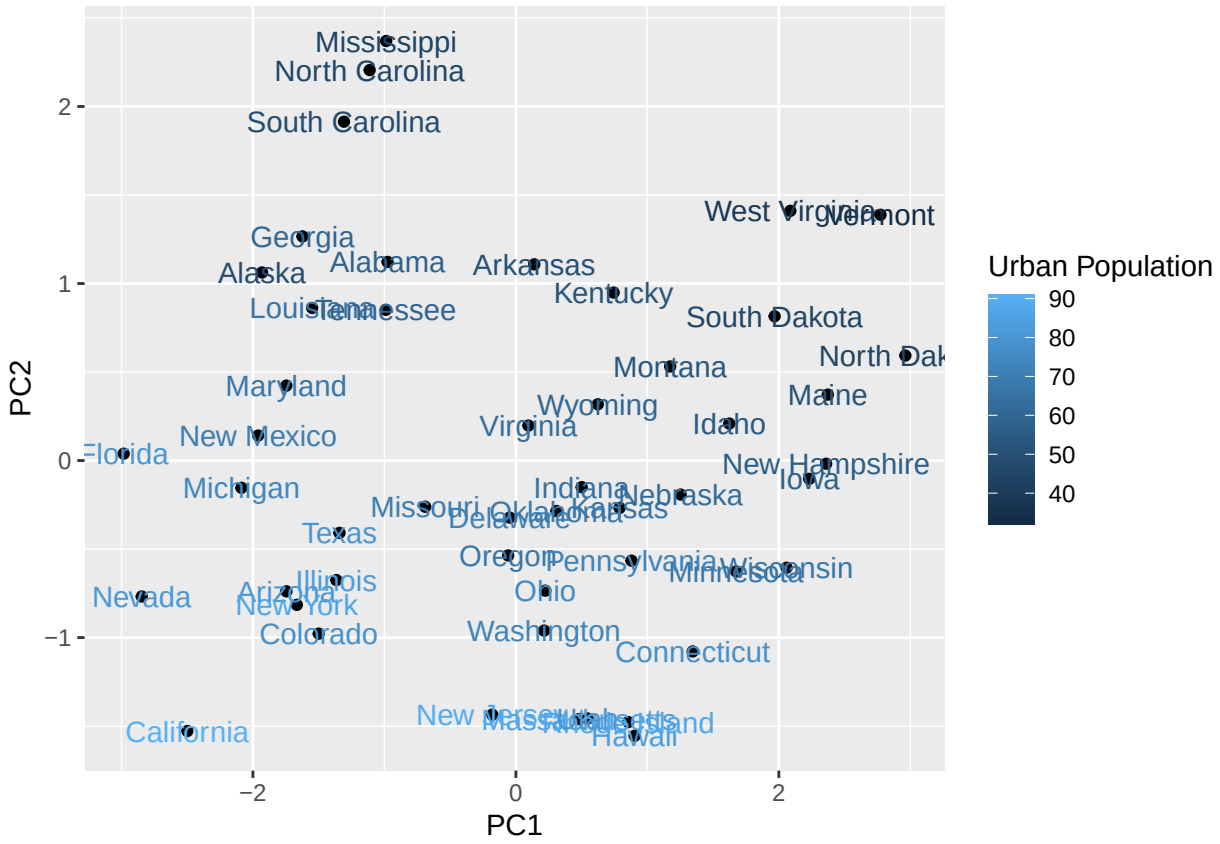
```
ggplot() + geom_point(aes(x = scores[, 1], y=scores[, 2])) +
geom_text(aes(x = scores[, 1], y=scores[, 2], label = rownames(USArrests))) +
labs(x = "PC1", y = "PC2")
```



Florida and Nevada and California are all states that stand out to me. Florida and Nevada are on the PC1 extreme, and California appears to be an extreme for both PC1 and PC2. Based on the loadings, it looks like murder, rape, assault, and urban population all contribute to a negative PC1. It makes sense to me that Florida, Nevada, and California are heavily negative because they all have popular urban vacation destinations, and people tend to drink alcohol at these locations, which I believe may be a contributing factor to poor decision making and thus the negative PC1 reading. The PC2 loading shows that urban population would push a state in the negative direction, so it makes sense that California is very negative, as LA is one of the biggest urban areas in the United States.

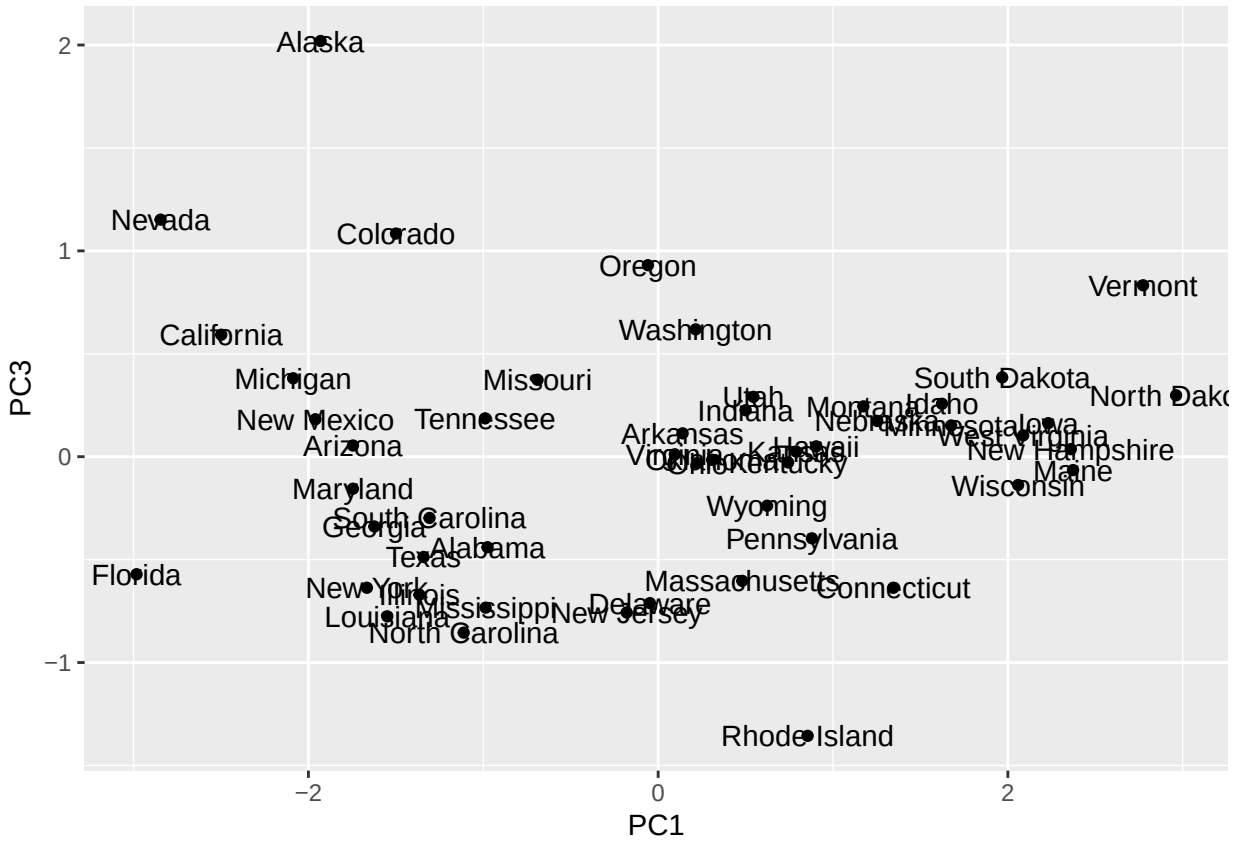
j) Create the same scaterplot, but color states according to the variable UrbanPop

```
ggplot() + geom_point(aes(x = scores[, 1], y=scores[, 2])) +
geom_text(aes(x = scores[, 1], y=scores[, 2], label = rownames(USArrests), color = USArrests$UrbanPop))
```



k) Create a scatterplot based on the 1st and 3rd PC

```
ggplot() + geom_point(aes(x = scores[, 1], y=scores[, 3])) +
geom_text(aes(x = scores[, 1], y=scores[, 3], label = rownames(USArrests))) +
labs(x = "PC1", y = "PC3")
```



The PC1 vs PC2 plot had a lot of variation across both PC's. In the PC1 vs PC3 plot, there is a lot less variation in the PC3 component. All but four of the state have a PC3 value that lies between -1, and 1. That was my biggest takeaway in terms of visual difference between the two plots.

Question 4: K-Means and PCA

In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data

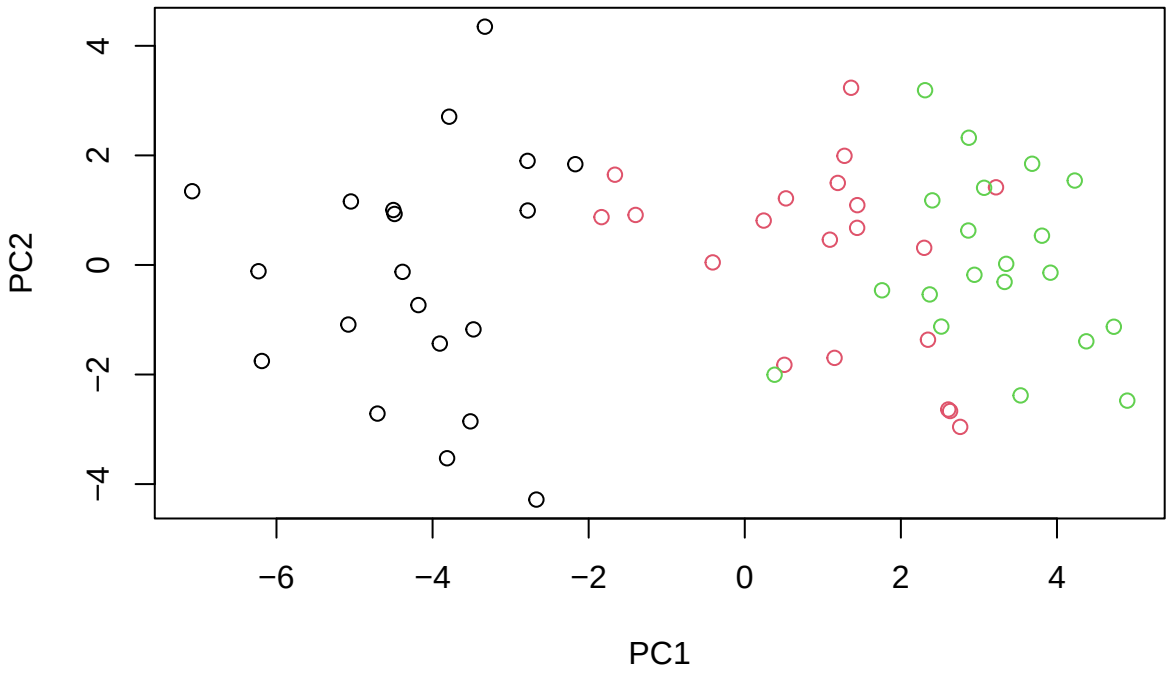
a) Generate a simulated dataset with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables

```
set.seed(45)

data <- rbind(matrix(rnorm(20*50, mean = .42), nrow = 20),
matrix(rnorm(20*50, mean=1.10), nrow = 20),
matrix(rnorm(20*50, mean=1.45), nrow = 20))
```

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

```
pca_data <- prcomp(data)
data_scores <- pca_data$x
plot(data_scores[,1:2], col=c(rep(1,20), rep(2,20), rep(3,20)))
```



c) Perform K-means clustering of the observations with K = 3. How weel do the clusters that you obtained in K-means clustering compare to the true labels?

```
set.seed(45)

kmean_3 <- kmeans(data, 3)
true <- c(rep(1,20), rep(2,20), rep(3,20))
table(kmean_3$cluster, true)
```

```
##    true
##      1  2  3
##   1  0 11 12
##   2 20  3  0
##   3  0  6  8
```

The first true class is executed well, but the second one is jumbled into all three clusters. The final true class is is split between two of the clusters

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

```
set.seed(45)

kmean_2 <- kmeans(data, 2)
true <- c(rep(1,20), rep(2,20), rep(3,20))
table(kmean_2$cluster, true)
```

```
##    true
##      1  2  3
##   1  0 17 20
##   2 20  3  0
```

It appears that the observations were fairly well classified

e) Perform K-means clustering with K = 4. Describe your results

```
set.seed(45)

kmean_4 <- kmeans(data, 4)
true <- c(rep(1,20), rep(2,20), rep(3,20))
table(kmean_4$cluster, true)
```

```
##    true
##      1  2  3
##   1  0  8 14
##   2 18  0  0
##   3  0  4  6
##   4  2  8  0
```

With four clusters we do see some misclassification. Three of our true groups are split between two clusters, but one of our true groups gets split into three separate clusters.

f) Perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data

```
set.seed(45)

kmean_3sc <- kmeans(data_scores, 3)
true <- c(rep(1,20), rep(2,20), rep(3,20))
table(kmean_3sc$cluster, true)
```

```
##    true
##      1  2  3
##   1  0 11 12
##   2 20  3  0
##   3  0  6  8
```

As we saw in a previous case one of the true groups is successfully clustered, while the others have some mix ups

g) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have a standard deviation of one

```
set.seed(45)

km3_scale <- kmeans(scale(data), 3)
true <- c(rep(1,20), rep(2,20), rep(3,20))
table(km3_scale$cluster, true)
```

```
##    true
##      1  2  3
##   1  0 12 12
##   2 20  3  0
##   3  0  5  8
```

Again with the scaling we see only one of the true groups get correctly clustered. The results are very similar to that of b)