CSC591: Foundations of Data Science HW5: Bayesian Inference, Missing Data Analysis
Released: 11/25/15 Due: **12/04/15 (23:55pm);** (One day late: -25%; -100% after that).

Student Name: Parth Satra
Student ID: 200062999

R. Bonus Question (R implementation) (**4% of grade**) (Please note that this is completely optional; use your time wisely as the implementation may take time).
(You can use any 2-d data, real or simulated for implementation; test data will be provided later to answer part b of this question)
(**a**) Implement G-Means (paper is provided under additional resources) (Algorithm 1, listed on page 3). (submit code as separate file; make single zip file)
(**b**) Generate 2-d plots (scatter plots and draw ellipsoids) (data will be provided later), include these plots as part of h/w solution)

Answer

The code given below generates the G-Means. The reference for the algorithm is taken from the paper "**Learning the k in k-means by Greg Hamerly, Charles Elkan**" shared on course moodle page.

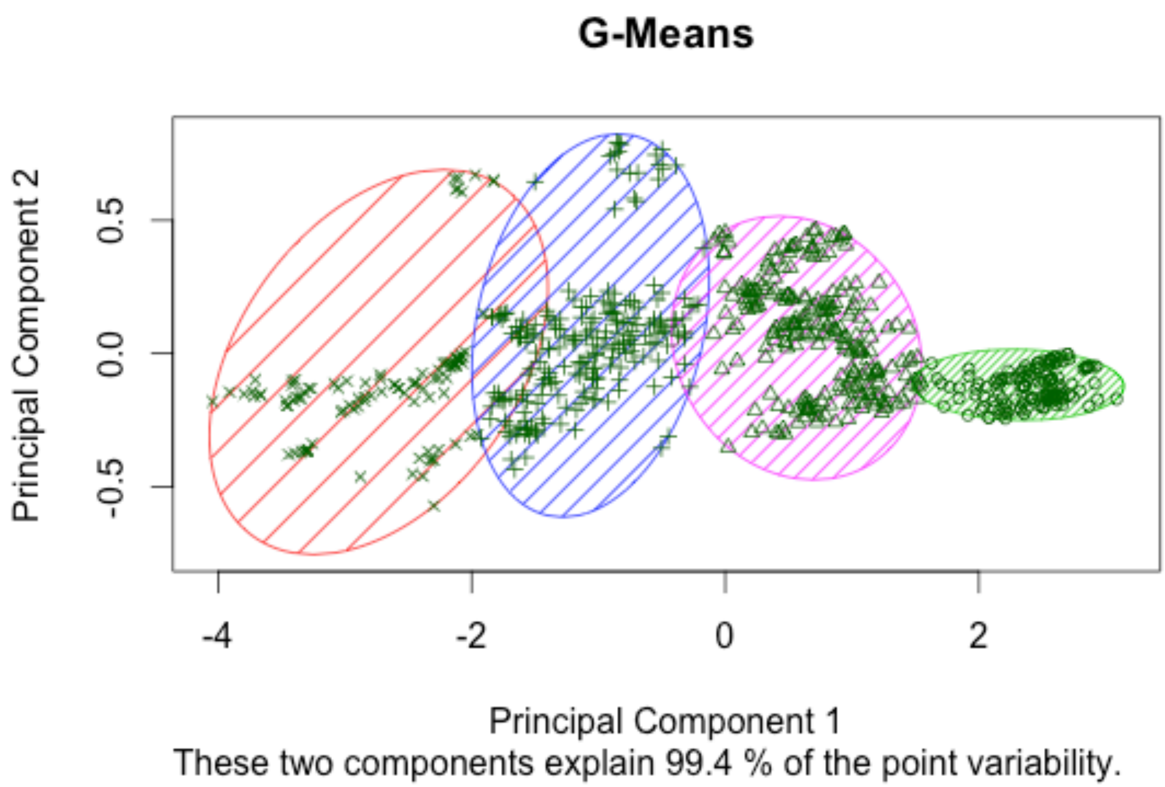
The accompanied README.txt file contains the required steps to run the code.

Code:

```
rm(list = ls())
library(ADGofTest)
library(cluster)

# Read data
data <- read.csv("hw5-3d-data.csv", header = TRUE)
alpha = 0.005
num_centers = 1;
centers = data[0, ]
clusters <- kmeans(data, 1)

# Run kmeans for the desired number of clusters
while(TRUE) {
 if(num_centers != 1) {
  clusters <- kmeans(data, centers = centers)
 }
 next_centers = data[0, ]
 # Set of datapoints assigned to center cj
 for(i in 1:nrow(clusters$centers)) {
  data_set <- data[clusters$cluster == i,]
```

```
    # Use a statistical test to detect if each data set follows a Gaussian distribution
    # Performing PCA to get new(better) centers
    p_comp <- prcomp(data_set)
    lambda <- p_comp$sdev[1]
    p_vector <- p_comp$rotation[,1]
    p_vector <- p_vector * sqrt(2 * lambda / pi)
    new_centers = rbind(clusters$centers[i,] - p_vector, clusters$centers[i,] + p_vector)

    # Run kmeans to get the new centers for the dataset
    new_clusters <- kmeans(data_set, new_centers)

    # Calculate direction between the two centers.
    direction <- new_clusters$centers[1, ] - new_clusters$centers[2, ]
    distance <- norm(as.matrix(t(direction)), "f")

    # Project the data onto the new centers
    projection <- (as.matrix(data_set) %*% direction) / (distance ^ 2)
    projection <- scale(projection)

    # Perform AD-Test
    ad <- ad.test(projection, pnorm)
    if(ad$p.value <= alpha) {
      next_centers <- rbind(next_centers, new_clusters$centers)
    } else {
      next_centers <- rbind(next_centers, clusters$centers[i,])
    }
  }
  centers <- next_centers
  if(num_centers == nrow(centers)) {
    break
  } else {
    num_centers = nrow(centers)
  }
}

final_cluster <- kmeans(data, centers)
clusplot(data, final_cluster$cluster, lines = 3, cex = 0.7, color = TRUE,
        main = "G-Means", shade = TRUE, xlab = "Principal Component 1",
        ylab = "Principal Component 2")
```

Output:


G-Means
Principal Component 2
0.5
0.0
-0.5
-4
-2
0
2
Principal Component 1
These two components explain 99.4 % of the point variability.