Assignment 5: Clustering

Ricco Amezcua
CS422 Data Mining
Department of Computer Science
Illinois Institute of Technology

April 30, 2013

Abstract

This is a report for the fifth assignment of CS422 Data Mining. In this report, different clustering methods are looked at. The K Means, DB Scan, and Hierarchical are all tested and compared. Also, my own K Means algorithm implementation is compared against R's K Mean algorithm.

1 Problem Statement

This lab looks at various clustering algorithms. The clustering algorithms looked at are K Means (one being R's K Means and the other being my own implementation), Hierarchical clustering, and the DB Scan algorithm. The various algorithms are compared to each other using entropy to compare their results to a cluster's actual class. The sum of squared error is looked at when comparing the each K Means cluster to each other. Each cluster is also explored using both plots and similarity matrices.

My own K Means algorithm is explored further by graphing the sum of squared error for each iteration of the algorithm. The execution time is also compared to R's built in K Means algorithm.

Hierarchical clustering is tested by changing its clustering function. The functions tested are single, centroid, median, and average.

DB Scan is tested by changing the radius of each cluster. Its entropy is also looked at for classification.

The data set that is used is the iris data set. This data set has 4 continuos attributes and 3 classes. When there are 3 clusters, the cluster's classification will be compared to the iris' actual classes using entropy.

2 Proposed Solution

For R's K Means and my K Means, various values for k were used. Since the iris data set has 3 classes, the values for k were chosen to be 2 to 6. This was done to see how well both algorithms did when the value of k was far from the true value. It was also done to test empty clusters in my K Means algorithm.

For Hierarchical clustering, only the functions used for clustering were changed. Only a randomly taken sample of the data was used. This made it easier to analyze the clustering done with the algorithm.

For DB Scan, the minimum points was kept at 5 but the range of the clusters was changed. This was done to see how clusters would react when they had to compete for points. The values for the range were 0.4, 0.5, and 0.6 as this created a sets of clusters that were either close to the actual number of classes or slightly more.

My implementation of K Means followed the K Means algorithm. After the points were normalized, each point is randomly assigned to a cluster. Then the first centroids were found by taking the mean of each cluster. After this initialization, the actual algorithm began. First, k centroids were found. Then, each the centroid membership for each point was updated so that each point belong to the centroid it was closest to. This algorithm was repeated until all centroid stopped changing their membership. Their final membership would be their cluster classification.

If the k value is larger than the actual number of classes, it possible for empty clusters to appear. This handled by first checking if any empty clusters exist after giving each point its cluster membership. If there are any empty clusters, then those active clusters are first ranked by their sum of squared error. Then each empty cluster is given a one point from the highest ranking active clusters. This both solves the empty cluster problem and tries to reduce the overall sum of squared error.

3 Implementation Details

The source code can be found under the *code* folder. The file named irisCluster.R contains all of R's built in K Means, the Hierarchical clusters, and DB Scan. The file named iris_mykmeans.R contains my implementation of the K Means algorithm. The folder named *data* contains all of the data present in this document and the comparison matrices that were not included here.

4 Results and Discussion

Looking at the values for entropy, the R's K Means did the best in terms of correctly identify the clusters as classes. My K Means algorithm was a bit worse than the R's K Means. DB scan had the worst entropy. It is unknown why this is. It may be because the radiuses are too small and causing the clusters to cover points that it shouldn't. This is supported by figures 4.30, 4.31, and 4.32, where DB Scan has trouble defining clusters. Also, looking at DB Scans similarity matrices in figures 4.33, 4.34, and 4.35, it is clear that DB Scan has trouble correctly clustering the points.

Comparing the plots R's K Means and my K Means, my K Means was similar when k was 1, 2, 3, and 4. This is shown in their almost identical similarity matrices. However, both algorithms differ when k is 5 and 6. The clusters that are define are very different. This may be due to the centroids being initialized in different places. It may also be that R's K Means is better as clustering. Looking at the similarity matrices for k being 5 and k being 6 for R's K Means, the clusters are very close together, where as in my K Means, the clusters are either completely separated or completely mixed.

The figures 4.21 to 4.26 show the SSE values during my K Mean's runs. All of the plots show that no matter the k, the SSE for the first iteration of the algorithm gives the worst value for the SSE. This is to be expected as this is when the clusters are least defined. The plots also show that the SSE will improve greatly for a few iterations, then the SSE will change only by a slight amount. This is also to be expected as the later iterations will have most of the clusters well defined. There will only be a few points that will be switching clusters towards the end of the run.

Figures 4.26 to 4.29 show the results of the hierarchical clustering algorithm. All of the algorithms correctly clustered the setosa flowers. This is likely due to the fact that the setosa data is the most defined in the dataset. However, all of them had trouble separating the versicolor and virginica. The function that did the best was the single method, with all other methods doing about the same.

R's K Means was the best at clustering the iris data set. It had much lower entropy than DB Scan. It was also lower than my K Means. It also had a much faster run time than my algorithm. This may because it is taking advantage of some R functions that I am not using. The SSE for all runs is also much lower for R's K Means than my K Means.

Run Times for K Means		
Run	K Means	My Kmeans
2	0.01	1.739
3	0.01	2.067
4	0.01	9.020
5	0.01	8.577
6	0.01	8.364
Avg	0.01	5.953

SSE for K Means		
Run	K Means	My Kmeans
2	152.34	220.87
3	78.85	139.09
4	71.44	113.79
5	69.24	90.20
6	42.73	81.16

Entropy		
K Means	My K Means	DB Scan
0.249	0.467	0.706

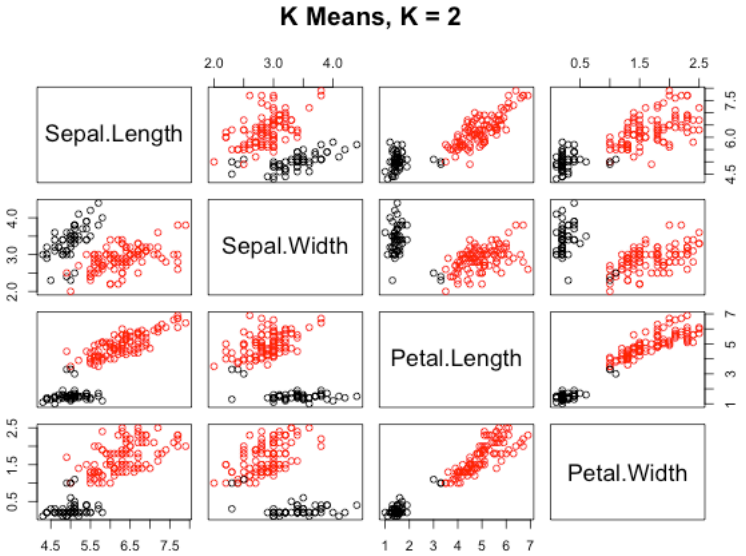

Figure 4.1:

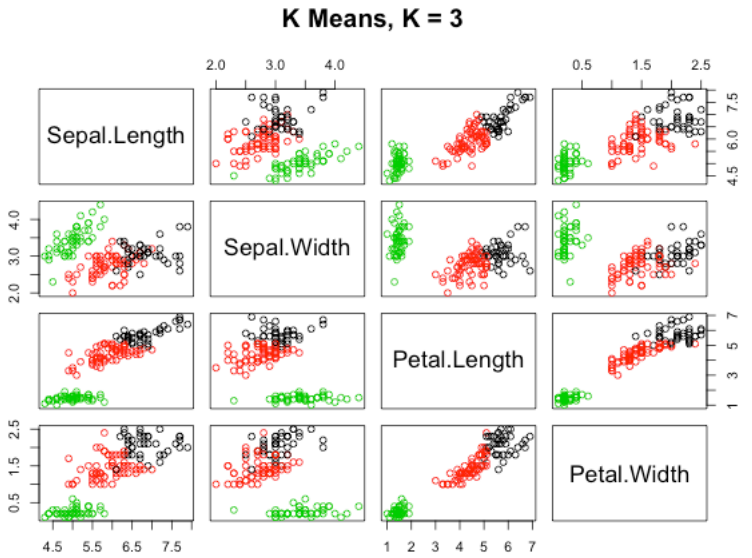

Figure 4.2:

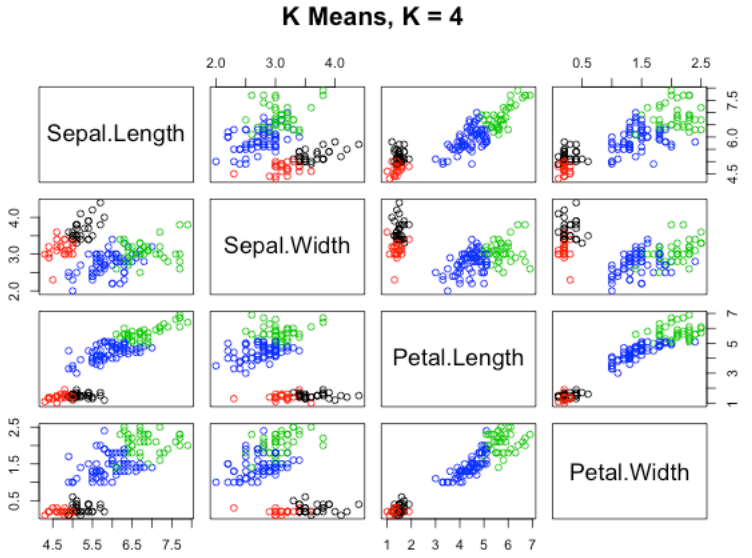

Figure 4.3:

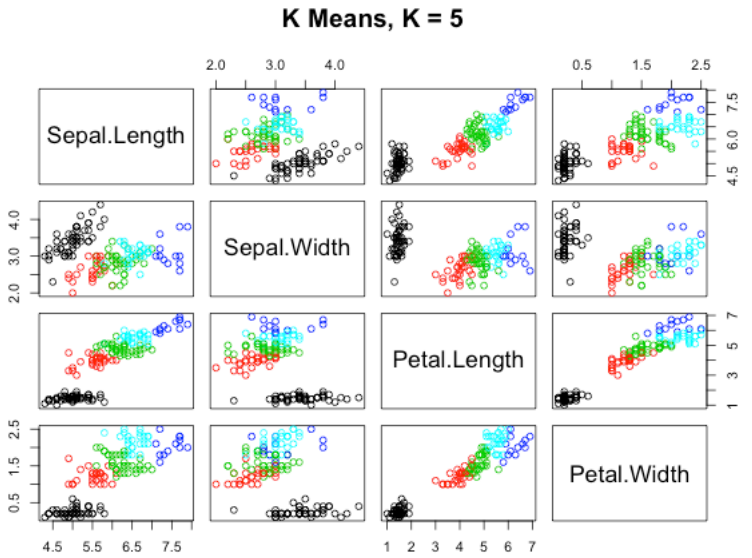

Figure 4.4:

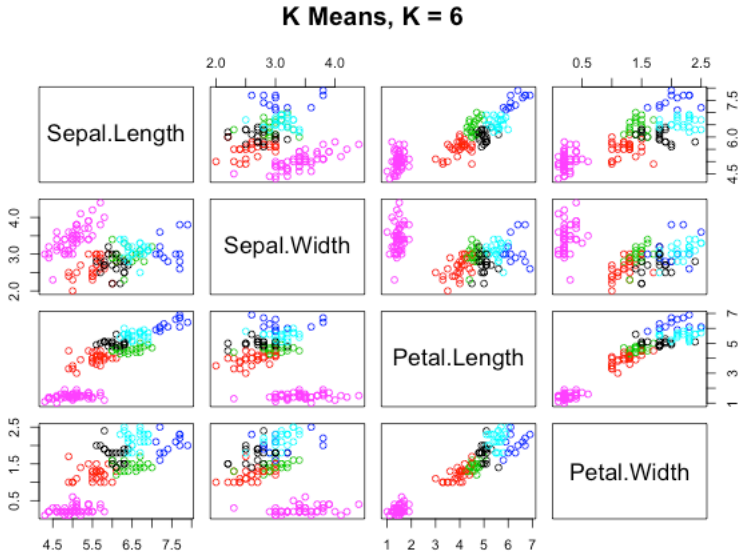

Figure 4.5:

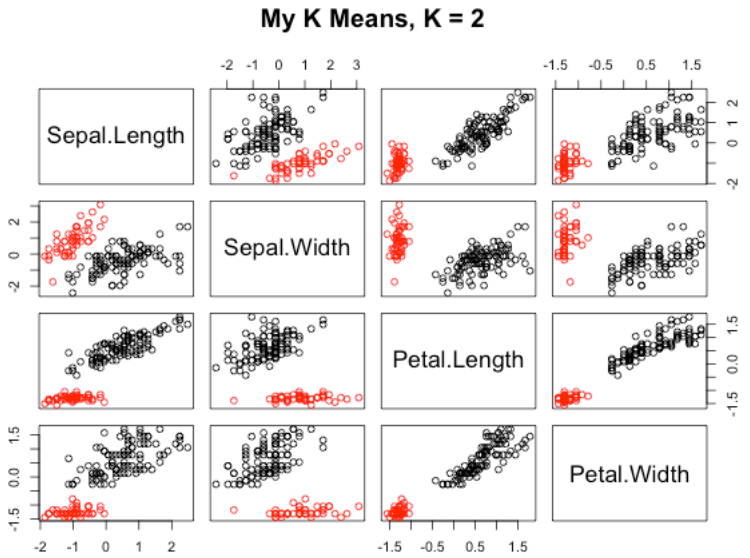

Figure 4.6:

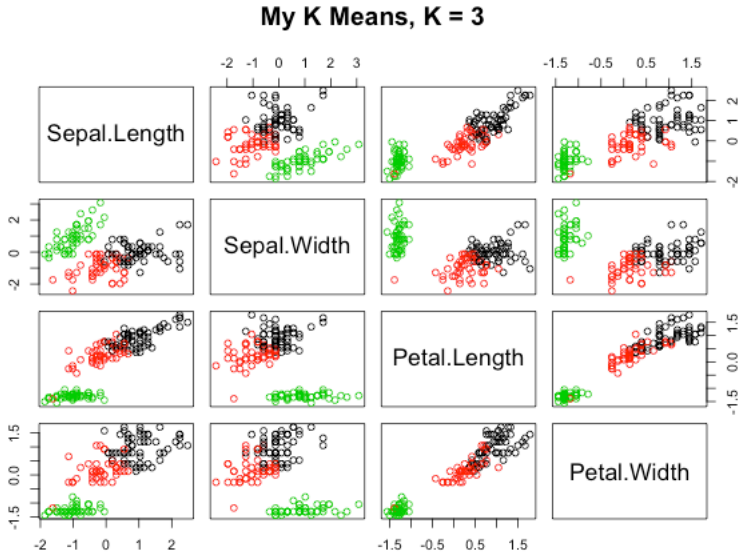

Figure 4.7:

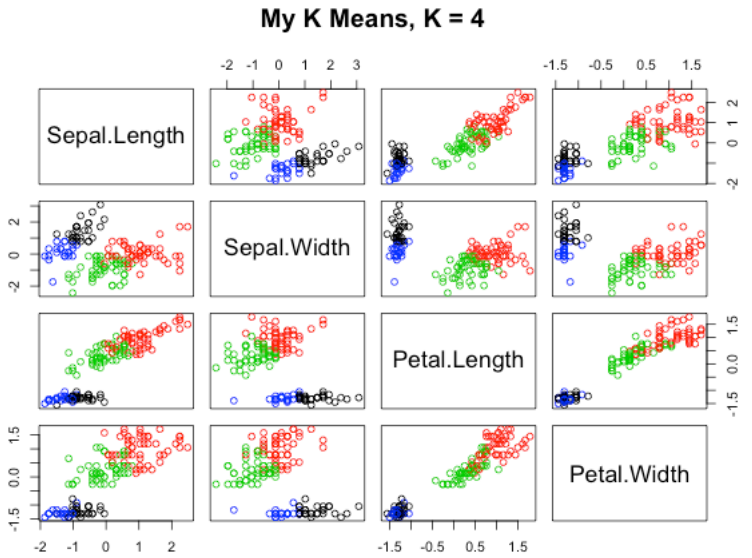

Figure 4.8:

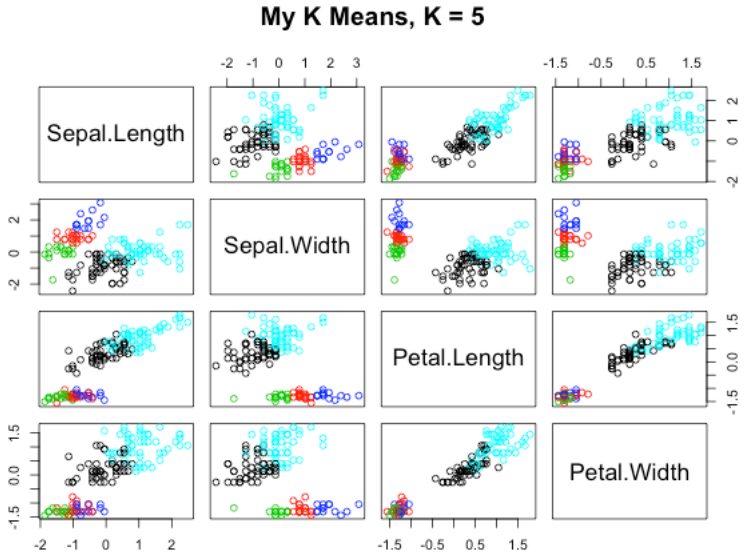

Figure 4.9:

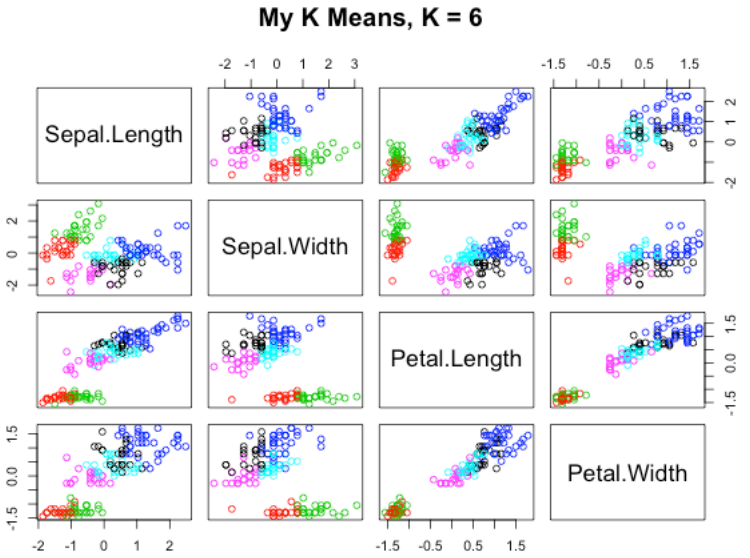
My K Means, K = 6
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width

Figure 4.10:

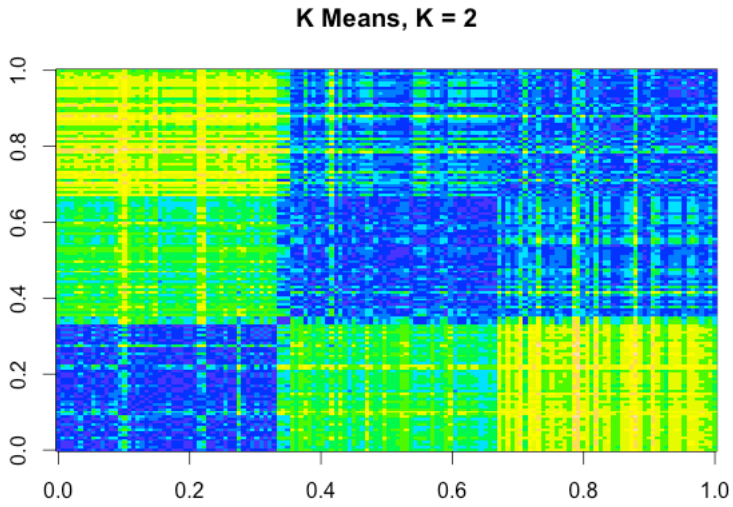

Figure 4.11:

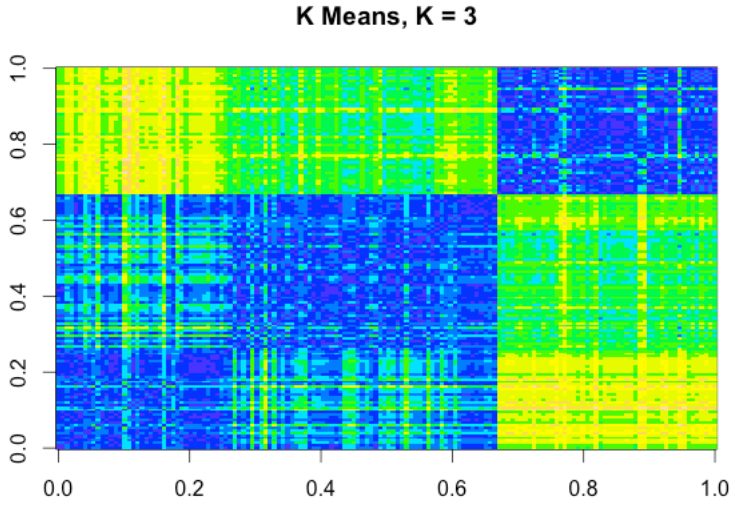

Figure 4.12:

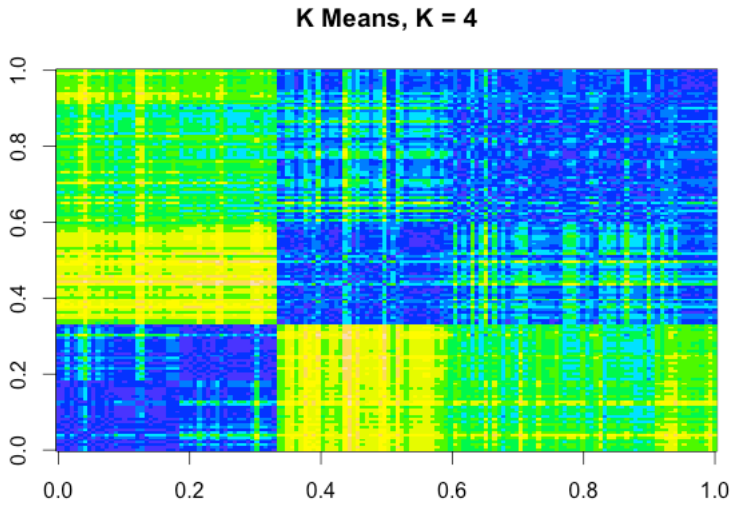

Figure 4.13:

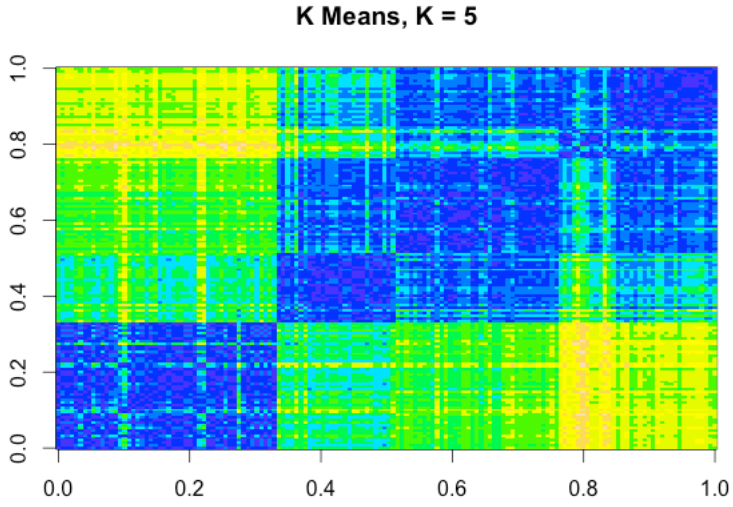

Figure 4.14:

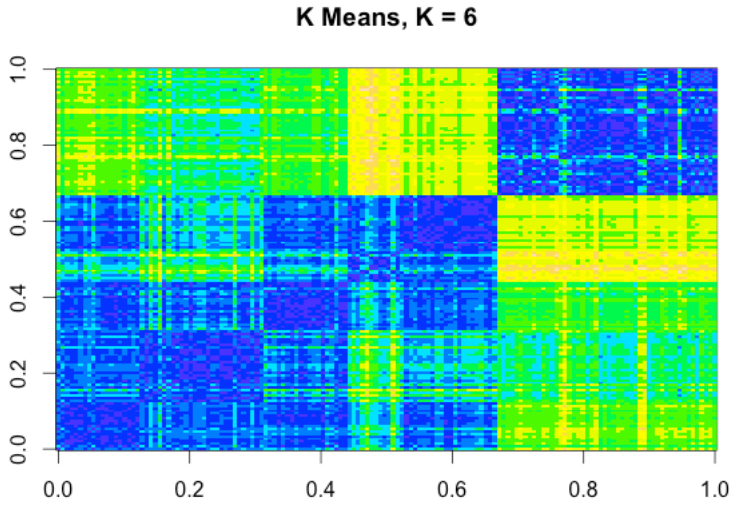

Figure 4.15:

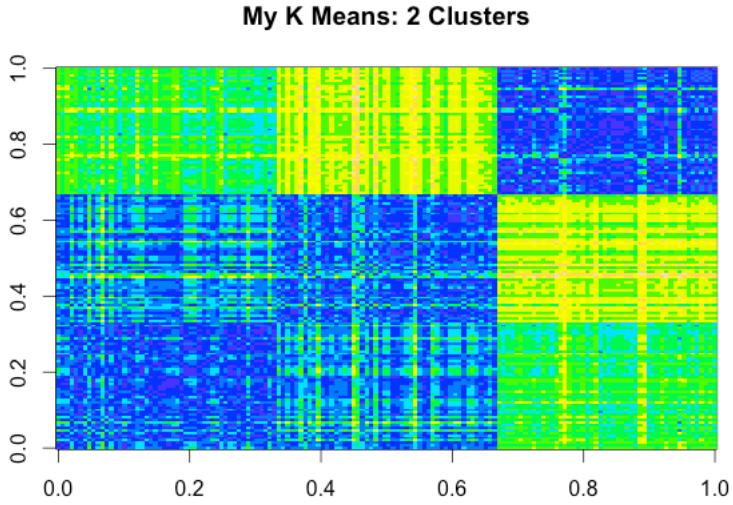

Figure 4.16:

Figure 4.17:

Figure 4.18:

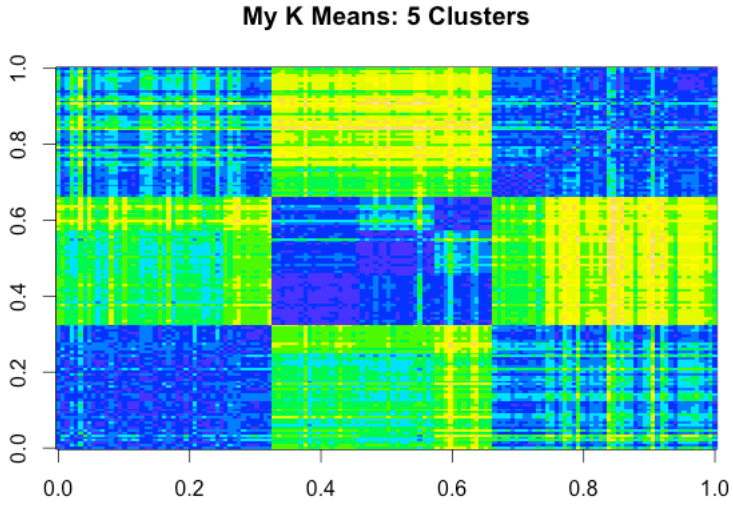

Figure 4.19:

Figure 4.20:

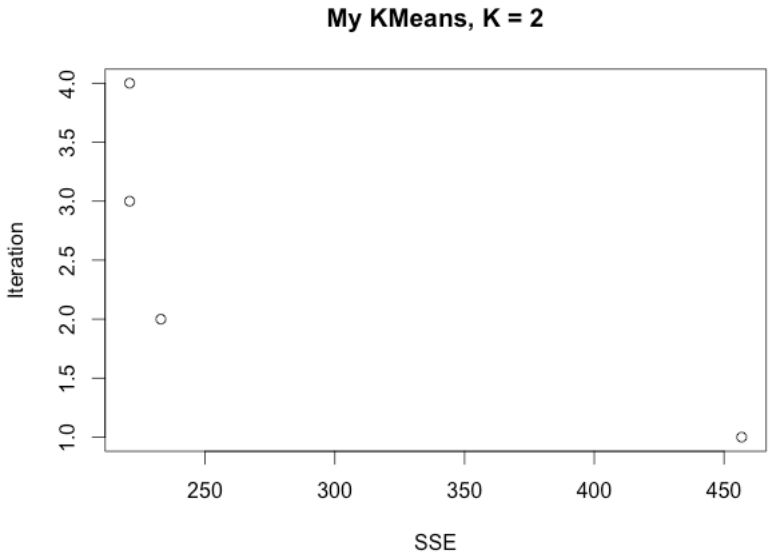

Figure 4.21:

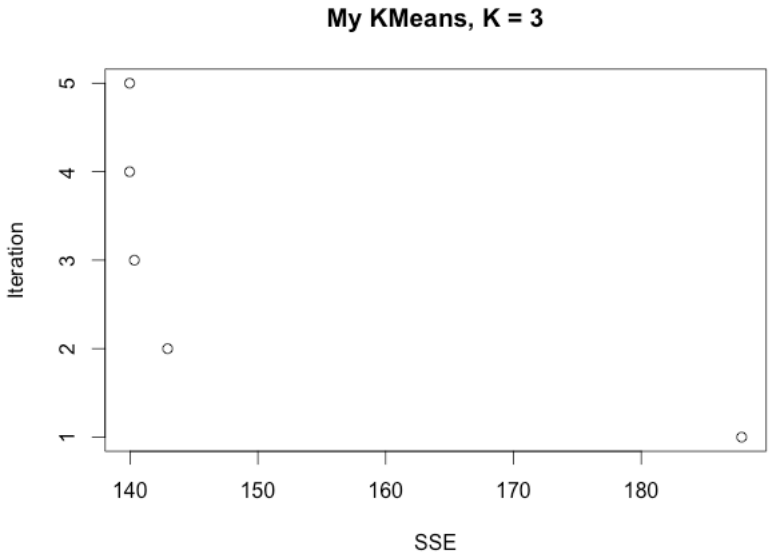

Figure 4.22:

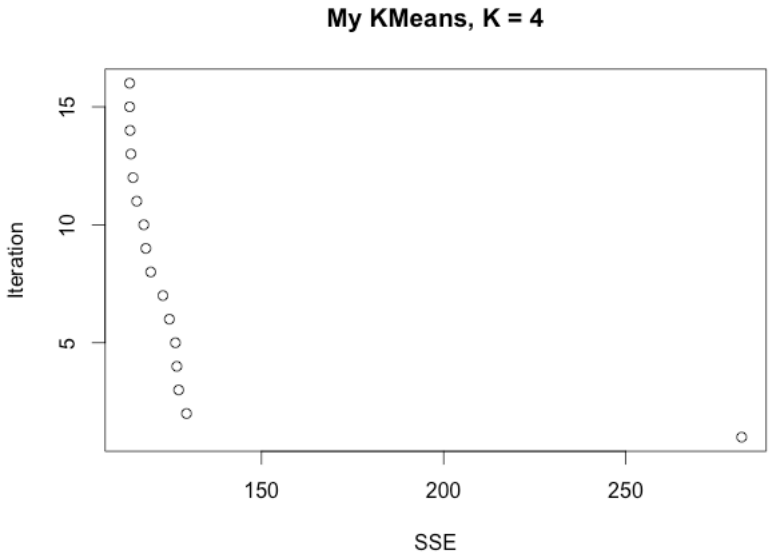

Figure 4.23:

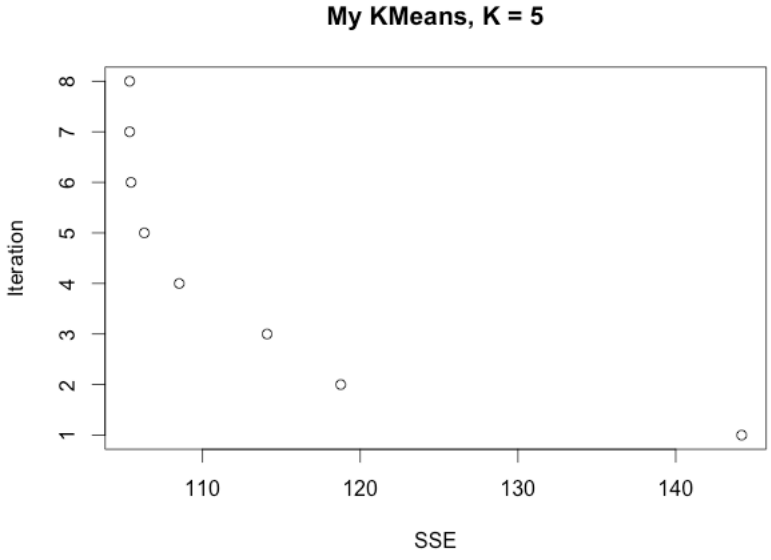

Figure 4.24:

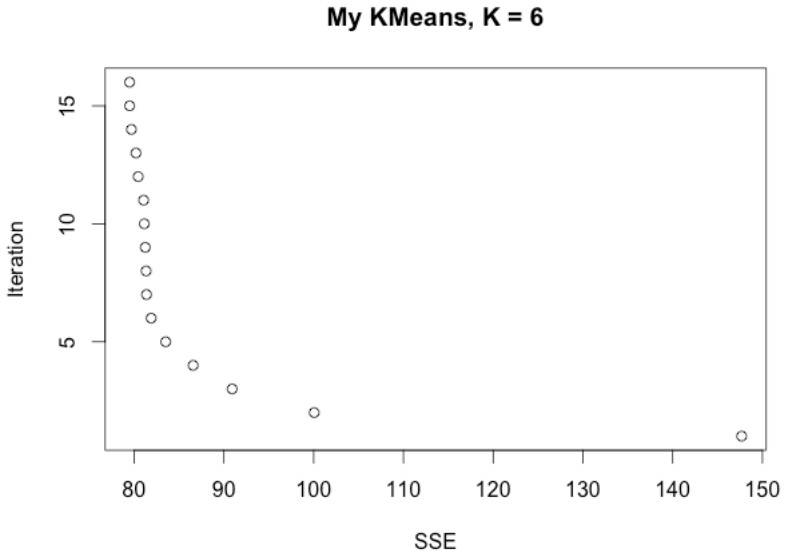

Figure 4.25:

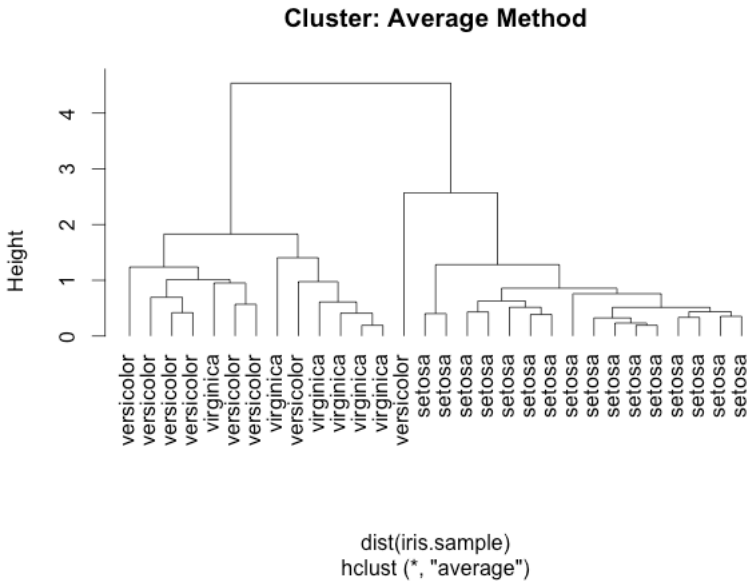

Figure 4.26:

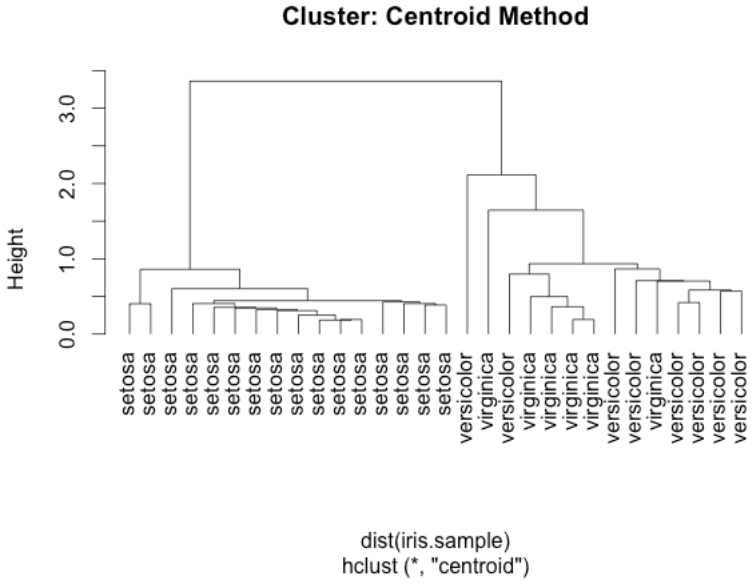

Figure 4.27:

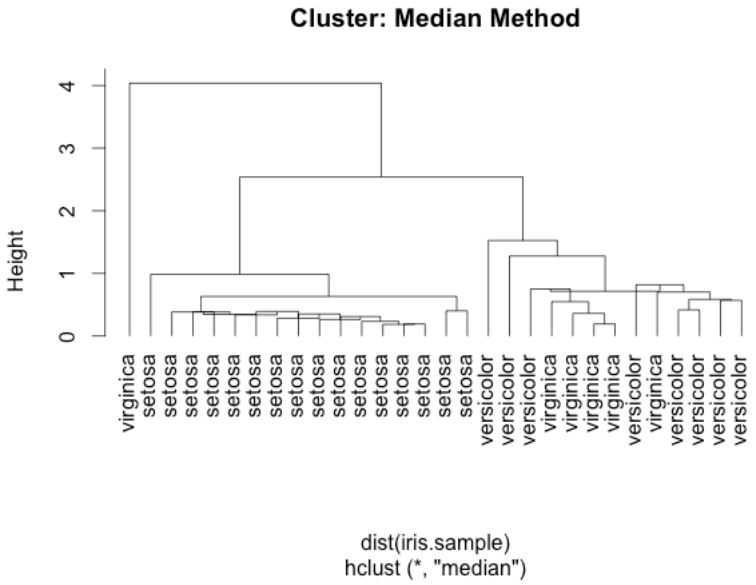

Figure 4.28:

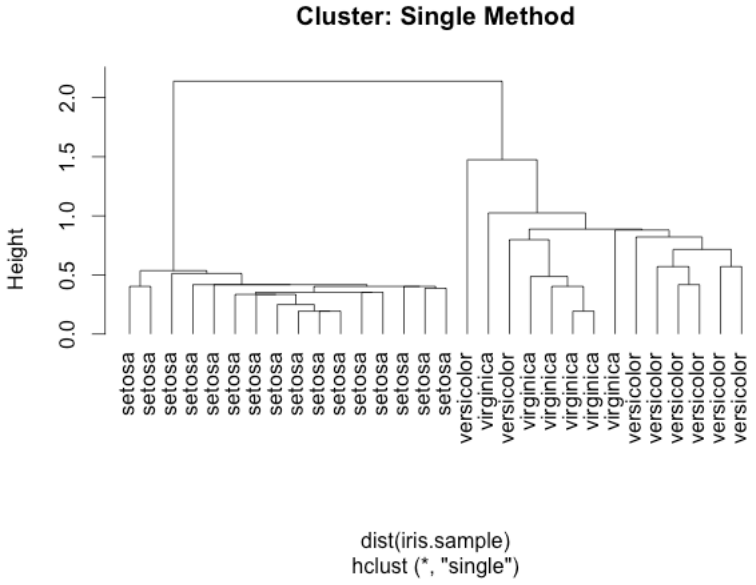

Figure 4.29:

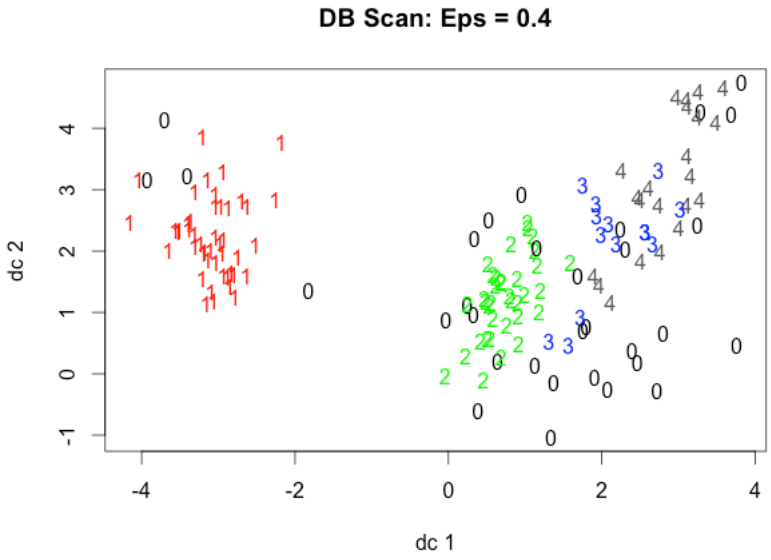

Figure 4.30:

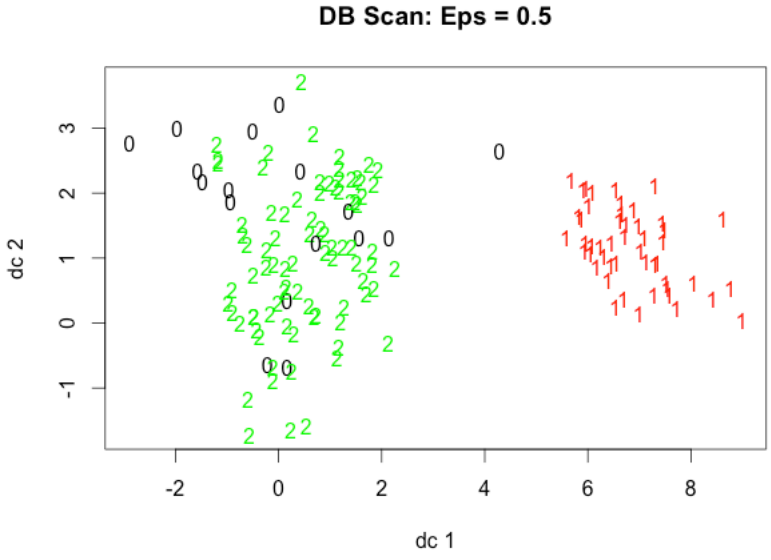

Figure 4.31:

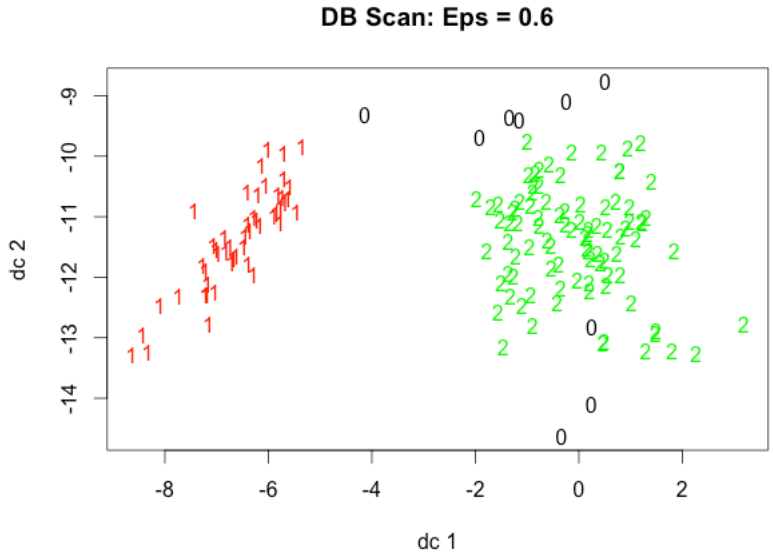

Figure 4.32:

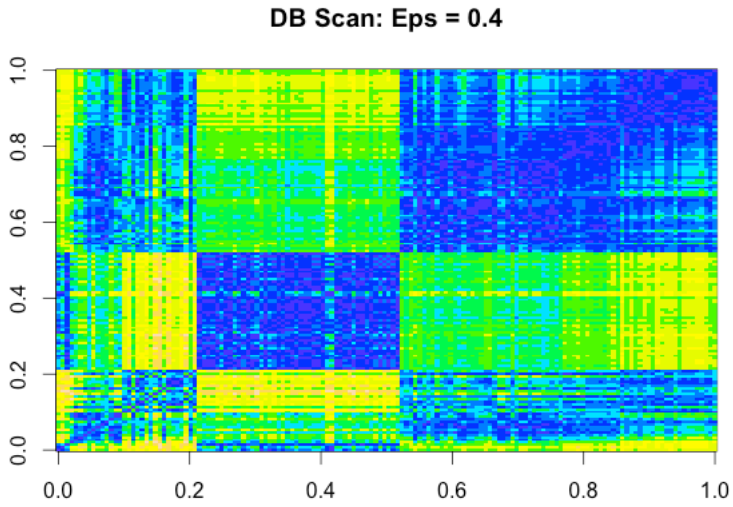

Figure 4.33:

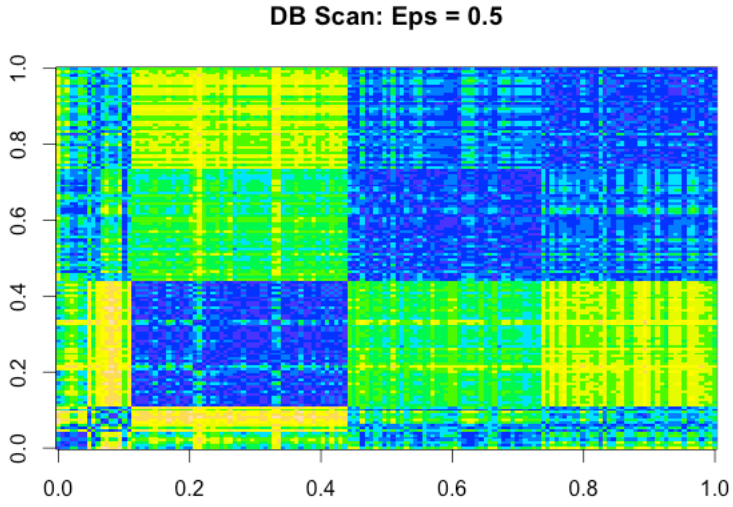

Figure 4.34:

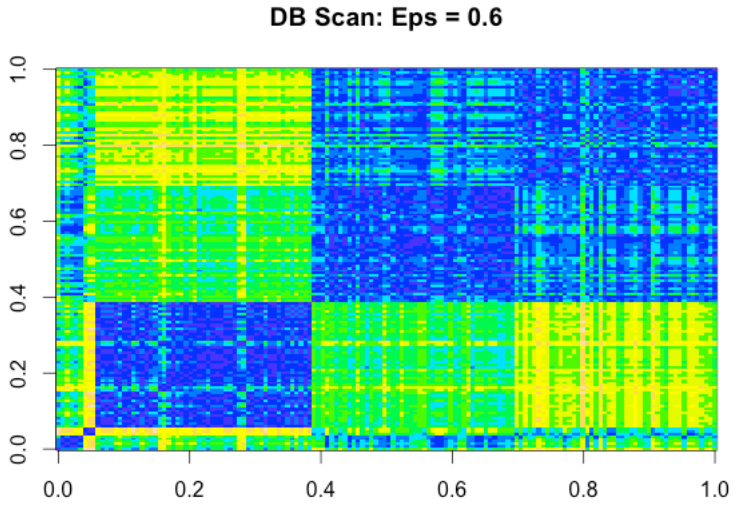

Figure 4.35: