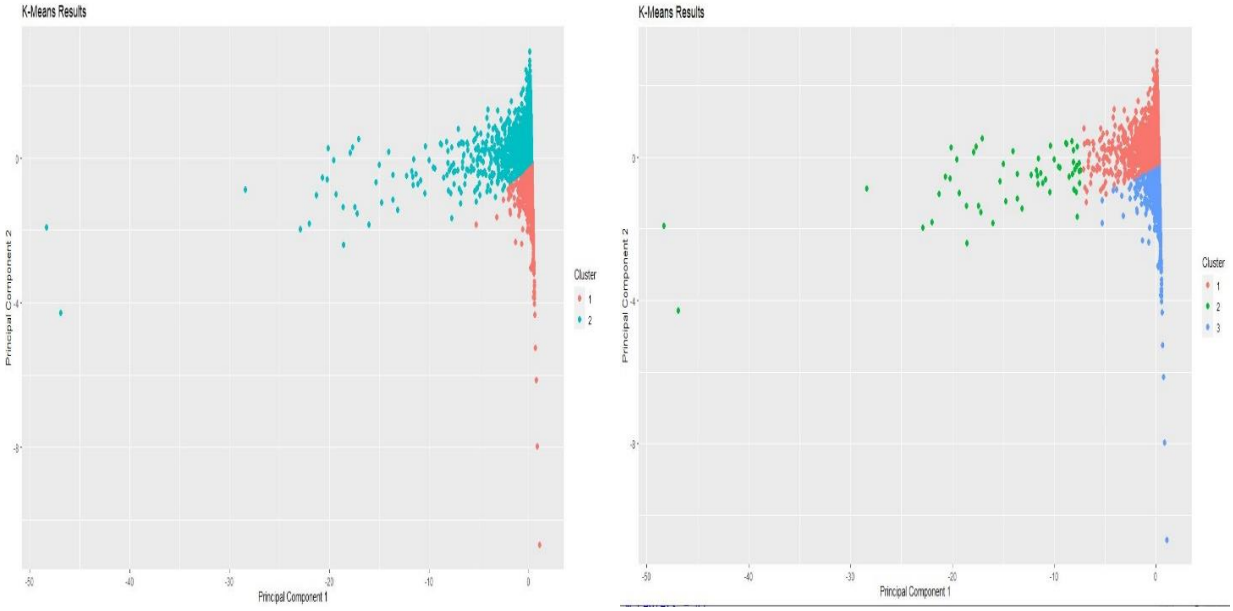
About Data

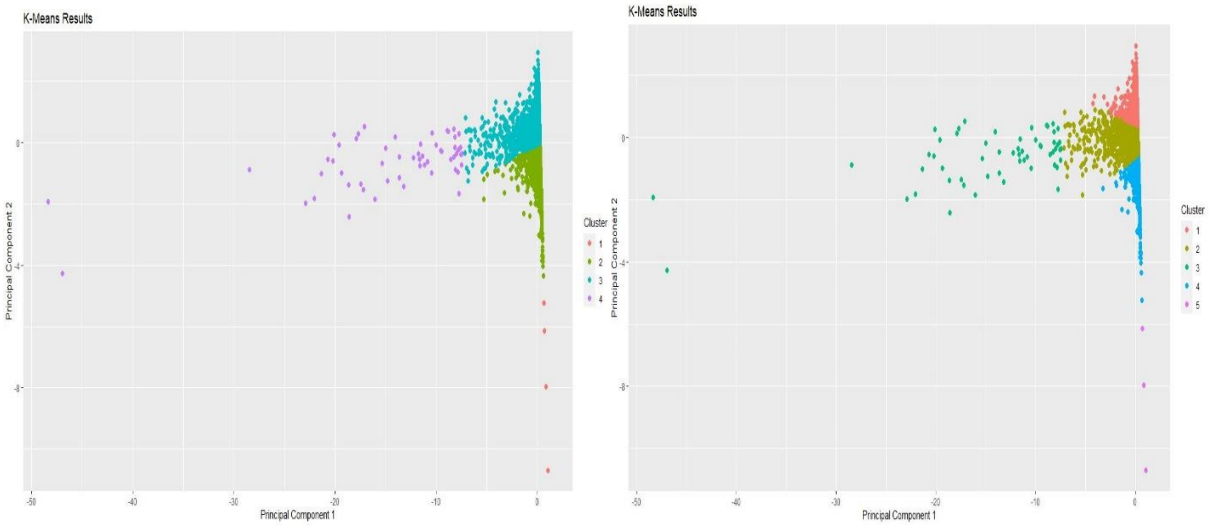
I chose “goodreads” dataset from Kaggle for this assignment which initially contained 10 features like Authors, Title, Average ratings and so on. After loading dataset into R programming language, I noticed that there are few features such as, isbn, isbn13 and Pageno which are not useful for our analysis. So, I started to do data preprocessing like the dimensions, dropping off few columns and knowing their class types which are essential before I start my analysis. I also noticed that dataset contains feature Language.code contains books of all languages but mostly this dataset includes books written in English, so I filtered data where Language.code = eng. After getting the required data for analysis, I started to do scaling on data features that are numeric.

Diagnostics for partitioning number of clusters

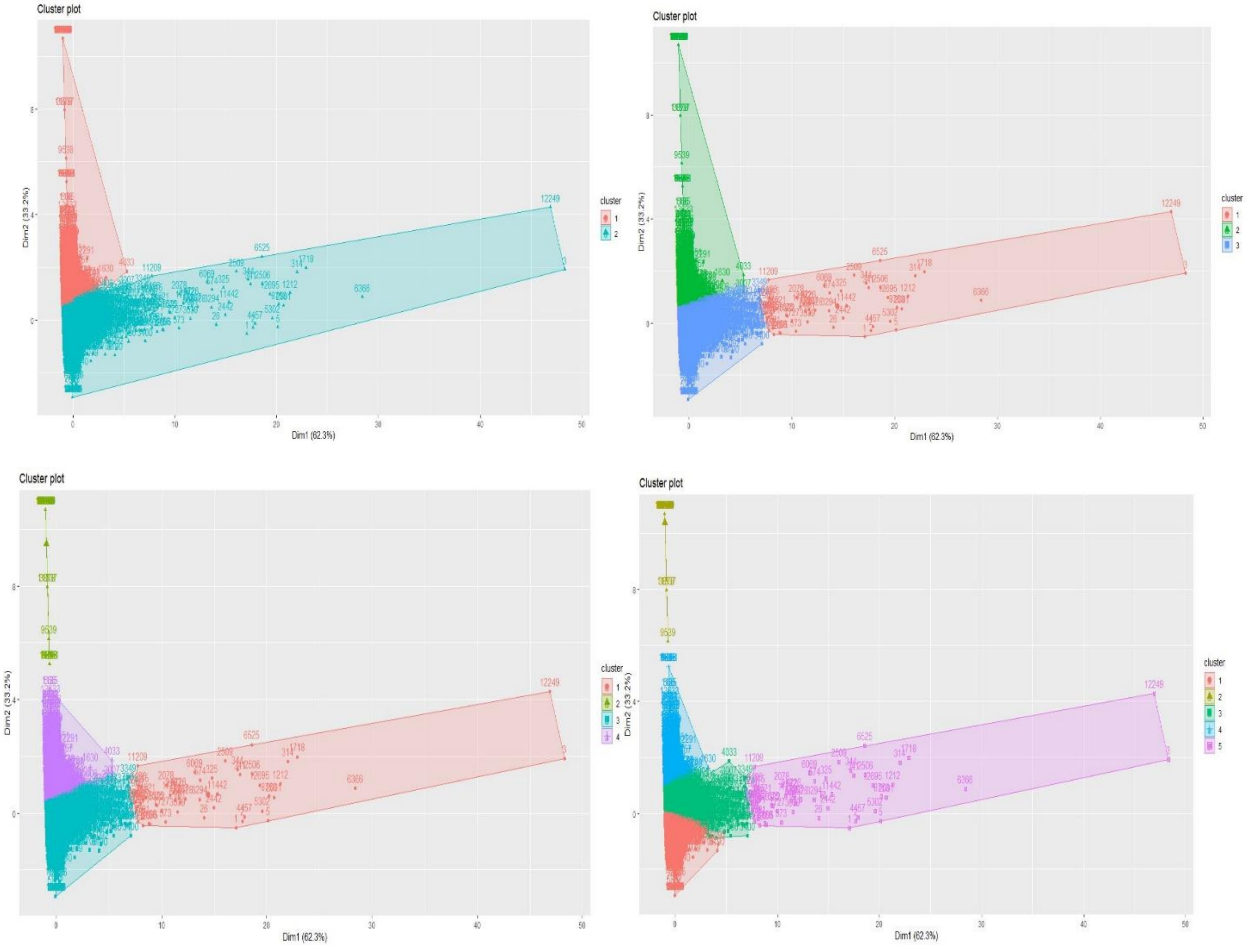
In order to decide how many clusters to be used, I tried three different clusters keeping k=2, k=3 and k=4. After giving the number of clusters to know whether they give me good clustering or no I plotted clusters one, two, three and four individually with PCA using plot.kmeans and fviz_cluster functions using useful and factoextra packages. I observed that different clusters do not have much of overlapping points which shows that they are good clustering.

Plots with PCA using plot.kmeans()

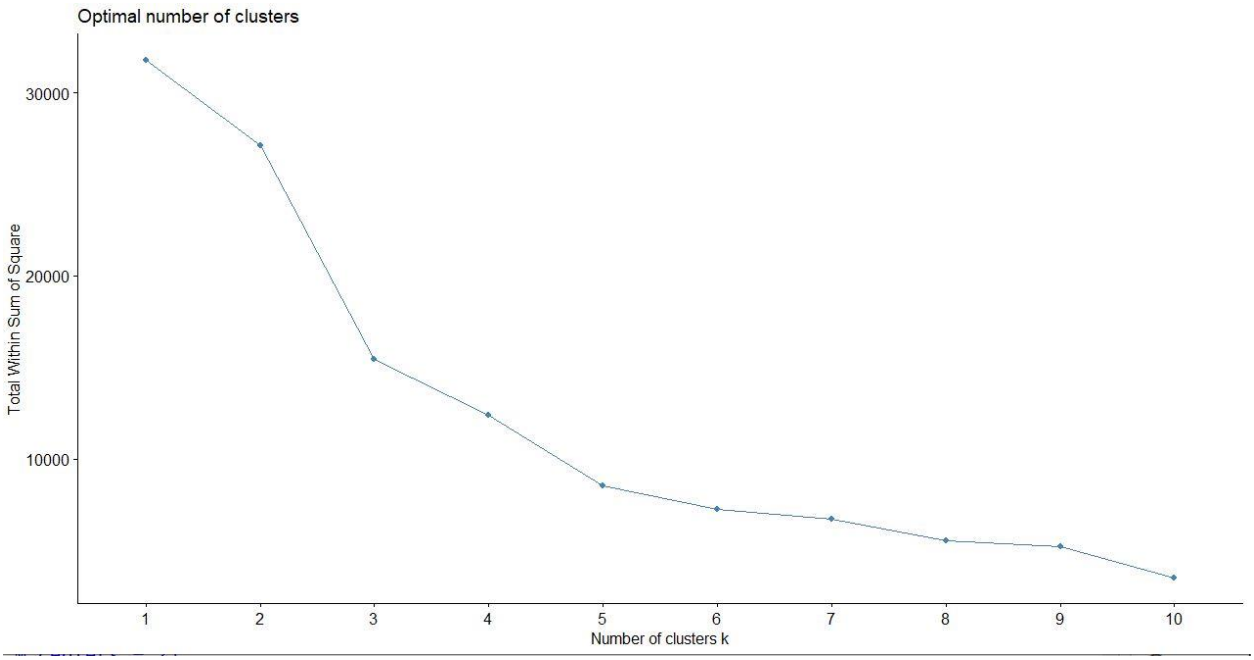




Plots with PCA using fviz_cluster()



Now, to decide on which cluster to pick I did Elbow method using wss, the only condition I used is to consider the cluster which gives me the least within squares using a smaller number of clusters.



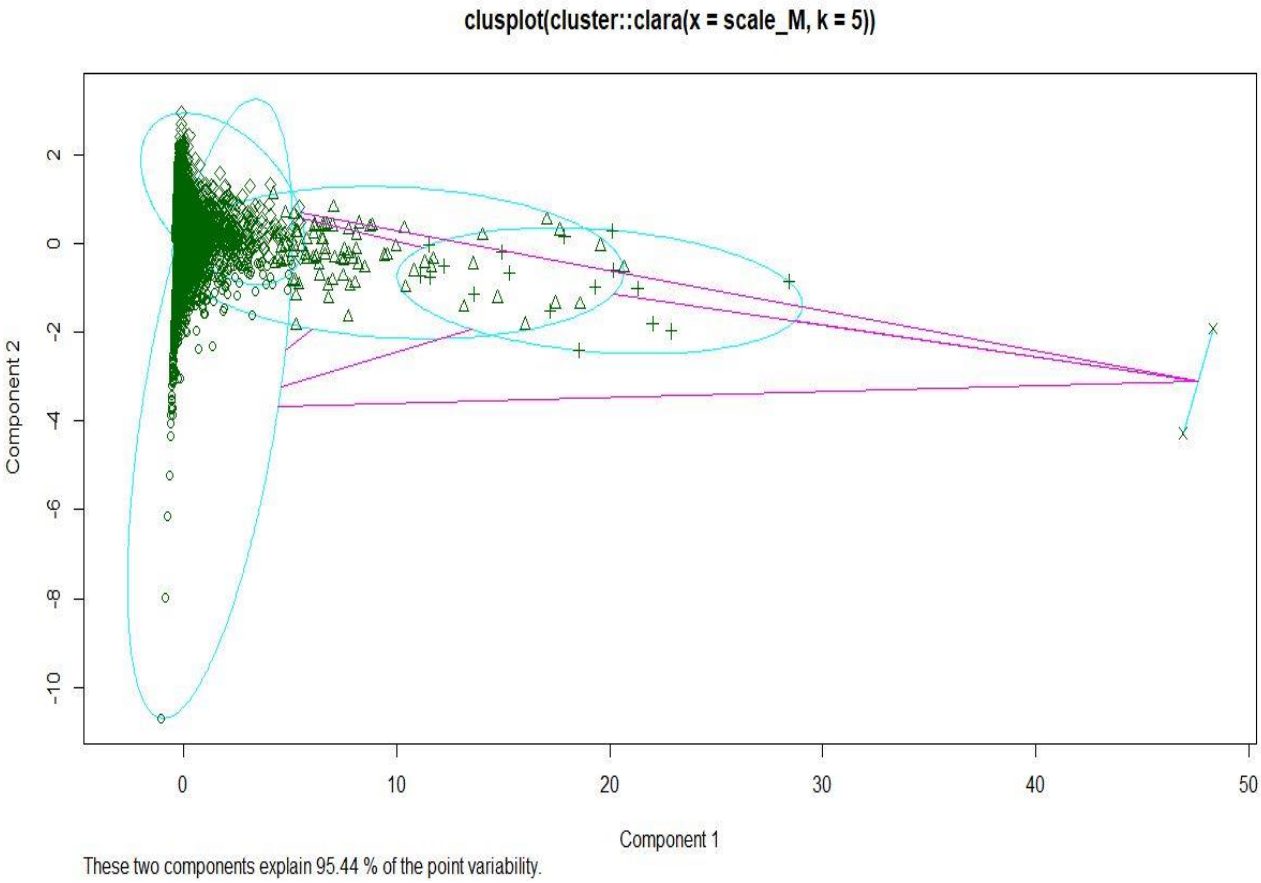
As one can observe from the above graph, the trend keeps going on with increase in number of clusters and decrease in Within Sum of Squares. There is comparatively less drop of WSS value from one-two cluster than two-three and though there is a little drop of WSS value from three-four, the four-five value drop is little big comparatively. So, I decided to use k=5.

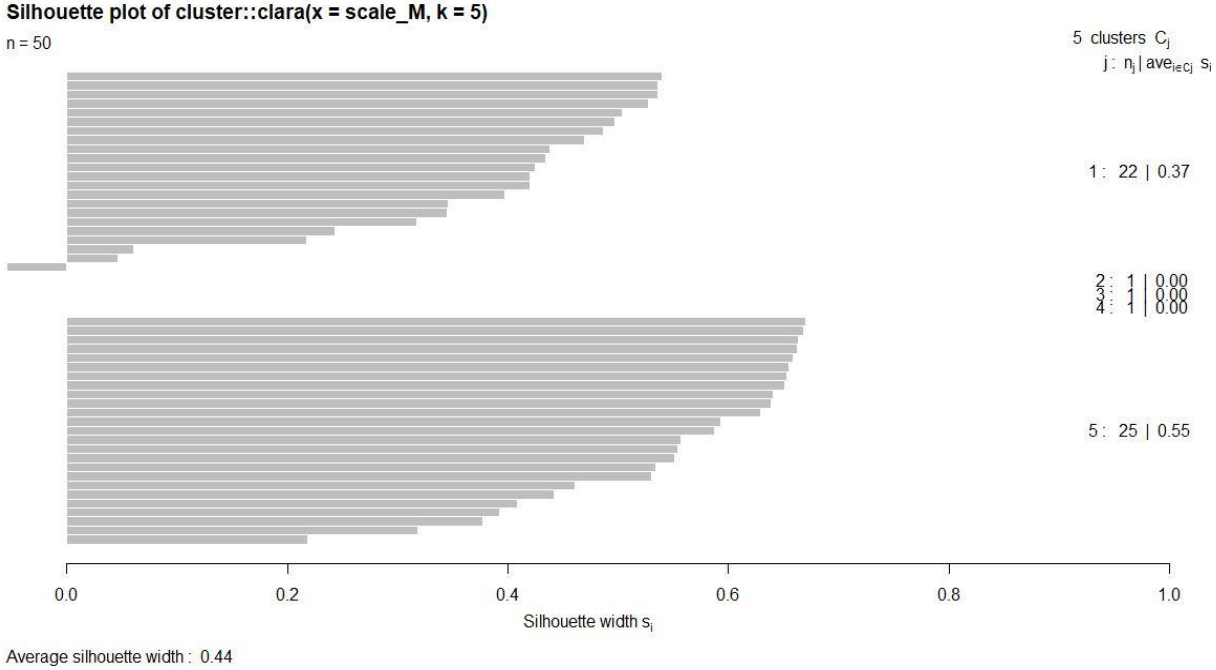
Diagnostics for partitioning algorithm

To decide which partition algorithm to use I ran algorithms like CLARA (Clustering Large Applications), Fuzzy and PAM (Partition Around Medoids)

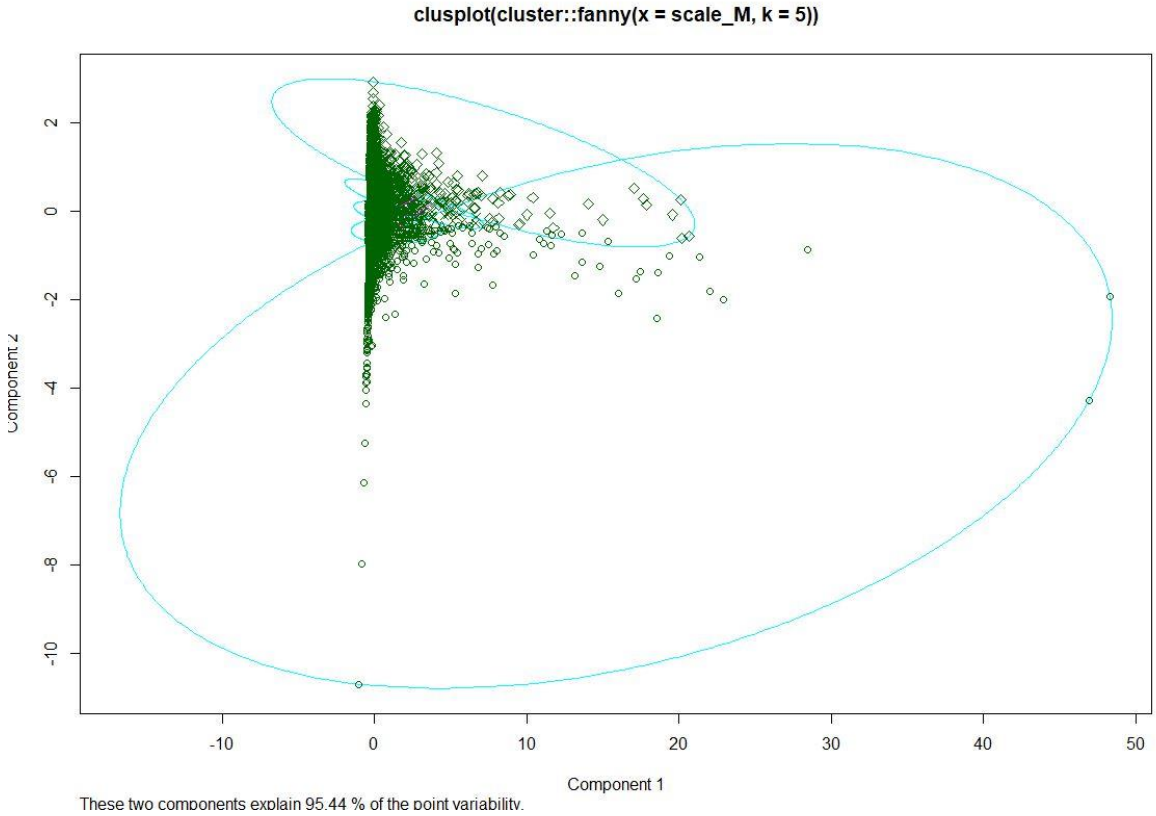
I chose CLARA partitioning algorithm as it deals with data containing a large number of objects and also reduces computing time. It plots given k value 5 along first principal component 1 and second principal component.

The below Silhouette plot also shows the bars are towards right except one which indicates all points belong to its cluster.

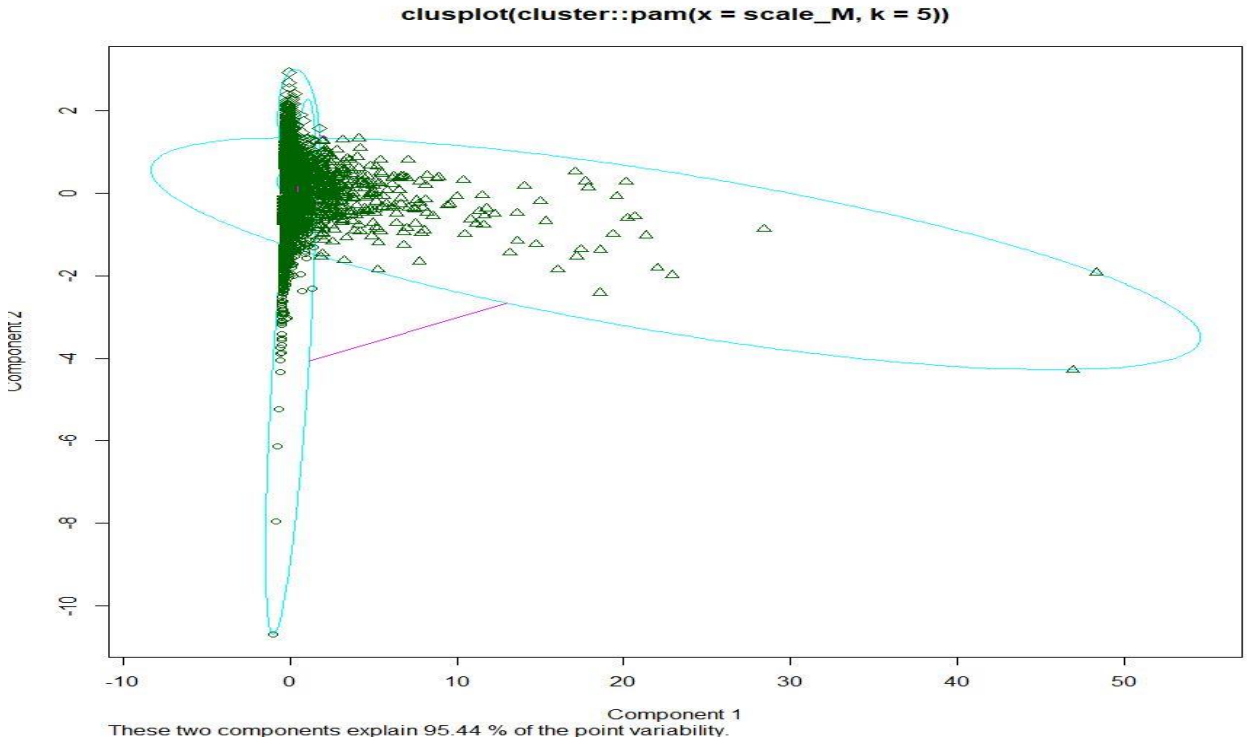

clusplot(cluster::clara(x = scale_M, k = 5))
Component 2
Component 1
These two components explain 95.44 % of the point variability.


Silhouette plot of cluster::clara(x = scale_M, k = 5)
n = 50
5 clusters Cj
j : nj | avei∈Cj si
1 : 22 | 0.37
2 : 1 | 0.00
3 : 1 | 0.00
4 : 1 | 0.00
5 : 25 | 0.55
Silhouette width si
Average silhouette width : 0.44

Fuzzy Algorithm

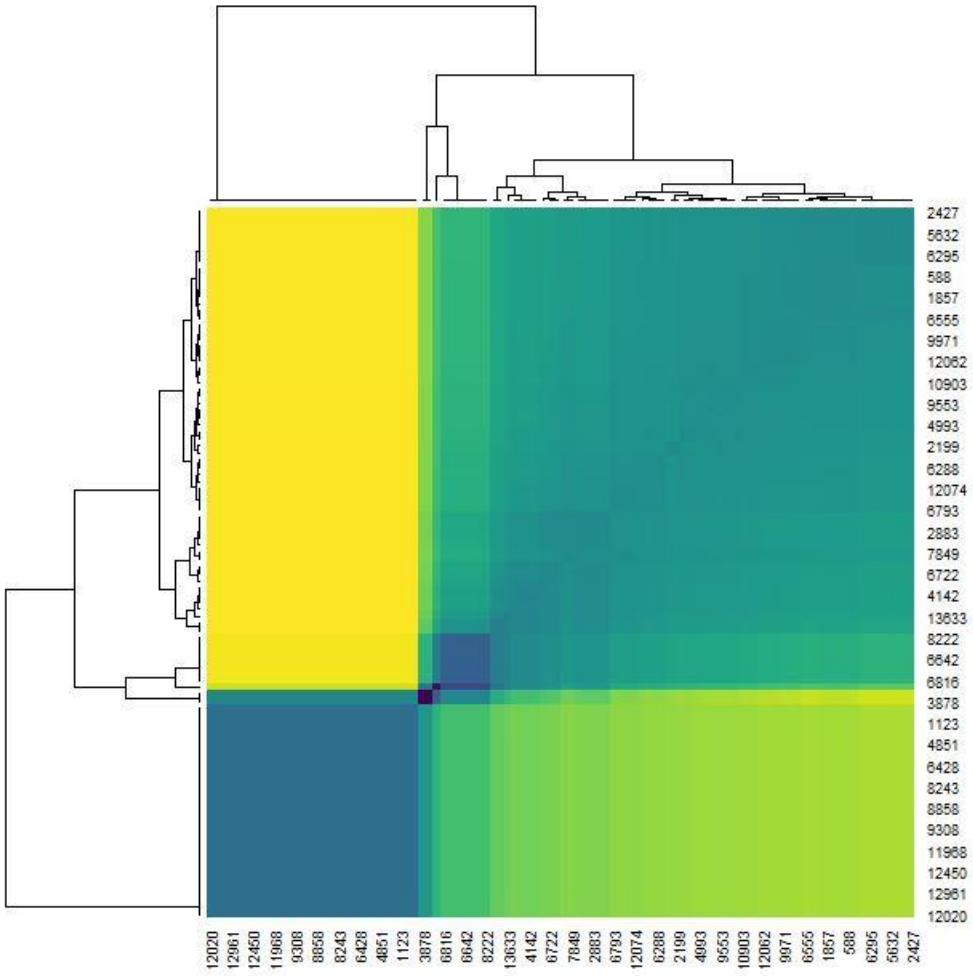


PAM algorithm

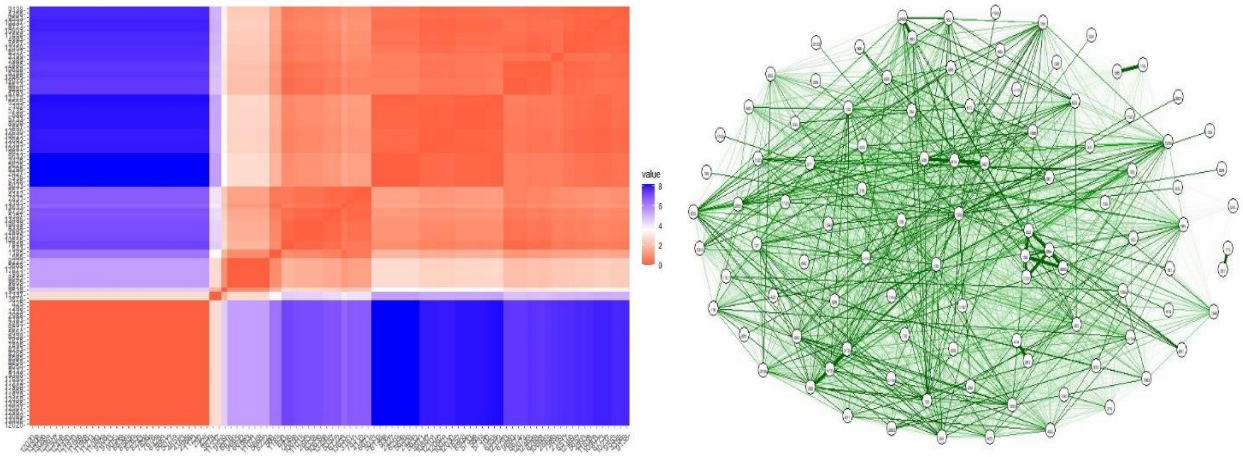


Diagnostics for Hierarchical parameters and algorithm

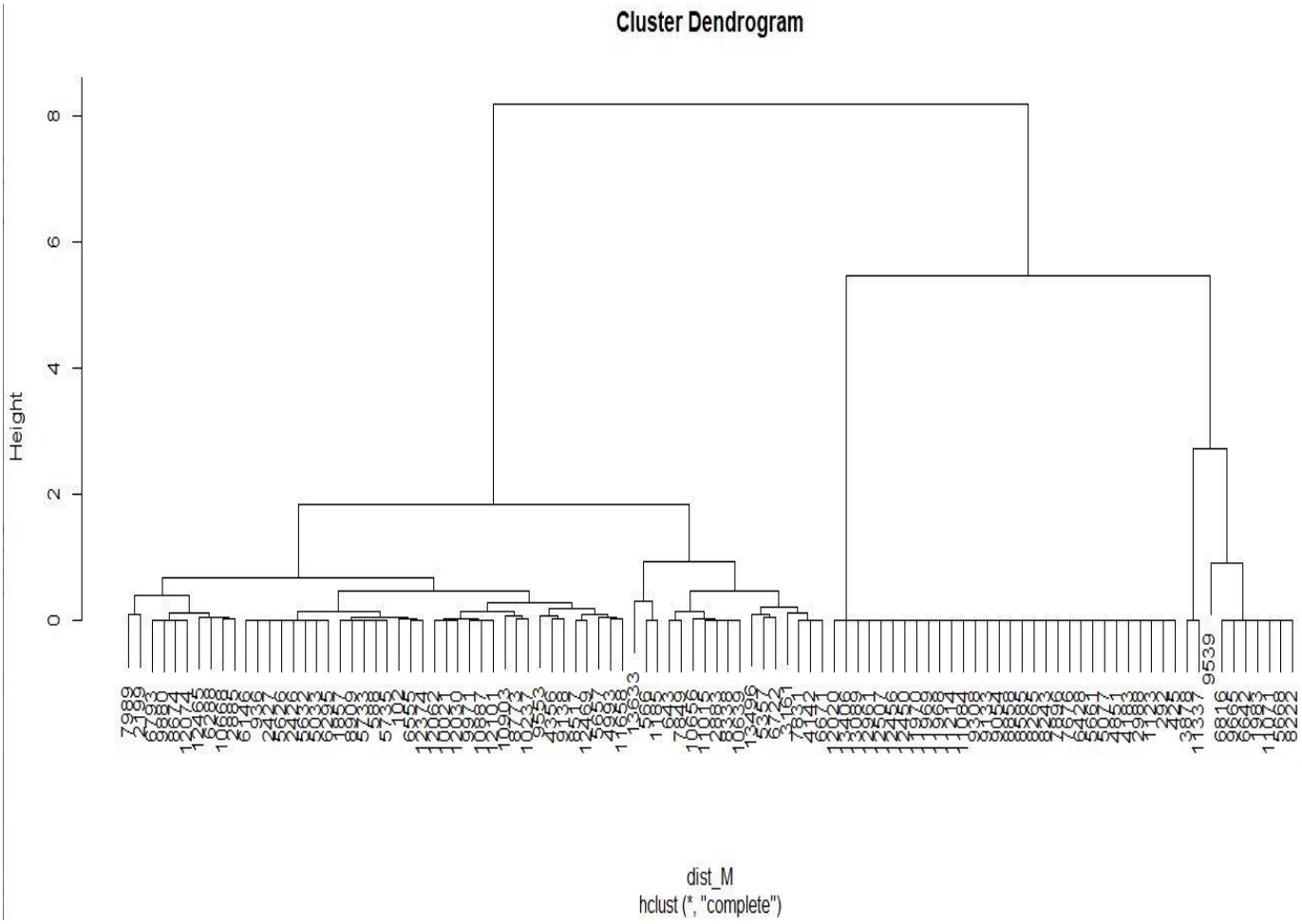
To compute distance matrix, I choose first 100 rows to get 100 by 100 matrix. To visualize the distance matrix, I used heatmap with virisdis color palate where darker values indicate that two points belong to same cluster and the darkest values is the distance from an observation to itself which will always be zero and it appeared like below:



There are other ways to visualize the distance matrix as well, like using fviz_dist() in factoextra package and using qgraph.



Using distance matrix as input I plotted hclust() algorithm and the graph looked like below



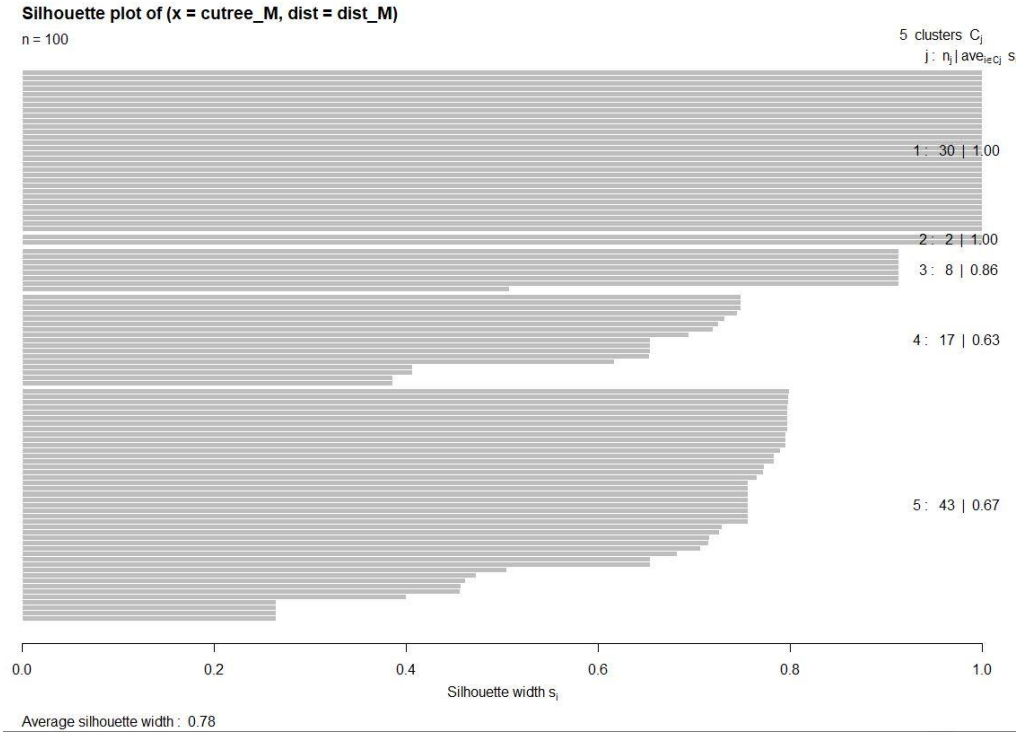
Cutree for hclust()

Using Cutree we can determine how the data points go into each cluster. The below is for first 100 sampled data points.

```
> cutree_M
  292   425  1123  2988  4183  4851  5077  5661  6428  7676  7896  8243  8265  8585  8858  9054  9133  9308 11084 11214 11968 11969 11970 12020 12450 12456 12507 12961 13089
    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
13406  3878 11337  9539  1983  5668  6642  6816  8222  9665 11071   566  1185 13633  4142  6671  7831  3161  5357  6722 13496  2883  8338 10639 11015 10656   643  7849  6793
    1     2     2     3     3     3     3     3     3     3     3     4     4     4     4     4     4     4     4     4     4     4     4     4     4     4     4     4     5
 8674  9880 12074 10668  2885  6288 12465  7989  2199  8517 12469  4993  4356  5657  9138  9553 11658  8773 10237 10903 10021 12030 12062  9971 10887 12101   588  1857  5733
    5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5     5
 5735  8909   102  6555 12374   936  2426  2427  5033  5626  5632  6146  6295
    5     5     5     5     5     5     5     5     5     5     5     5     5
```

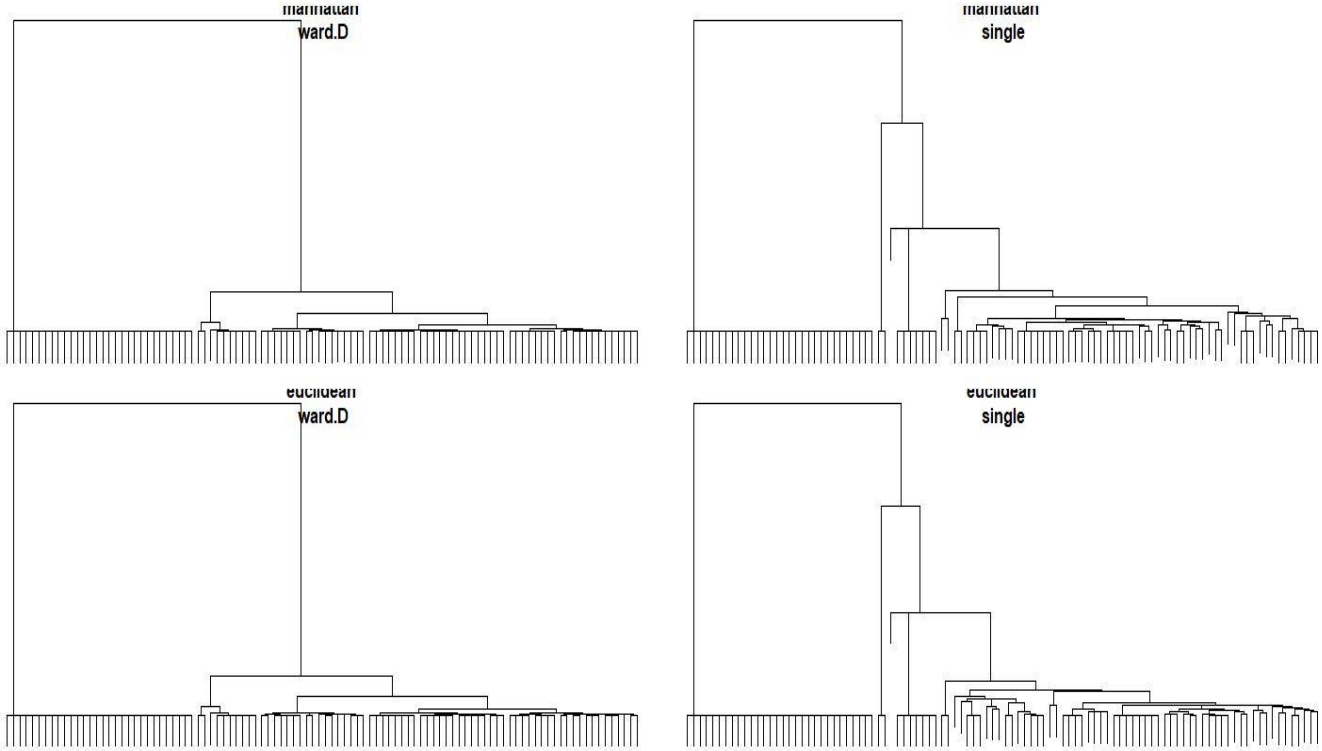
Silhouette for hclust()

Silhouette plots help us evaluate how well each point fits with the rest of its cluster. In my silhouette plots all bars point to the right which indicates good clustering.

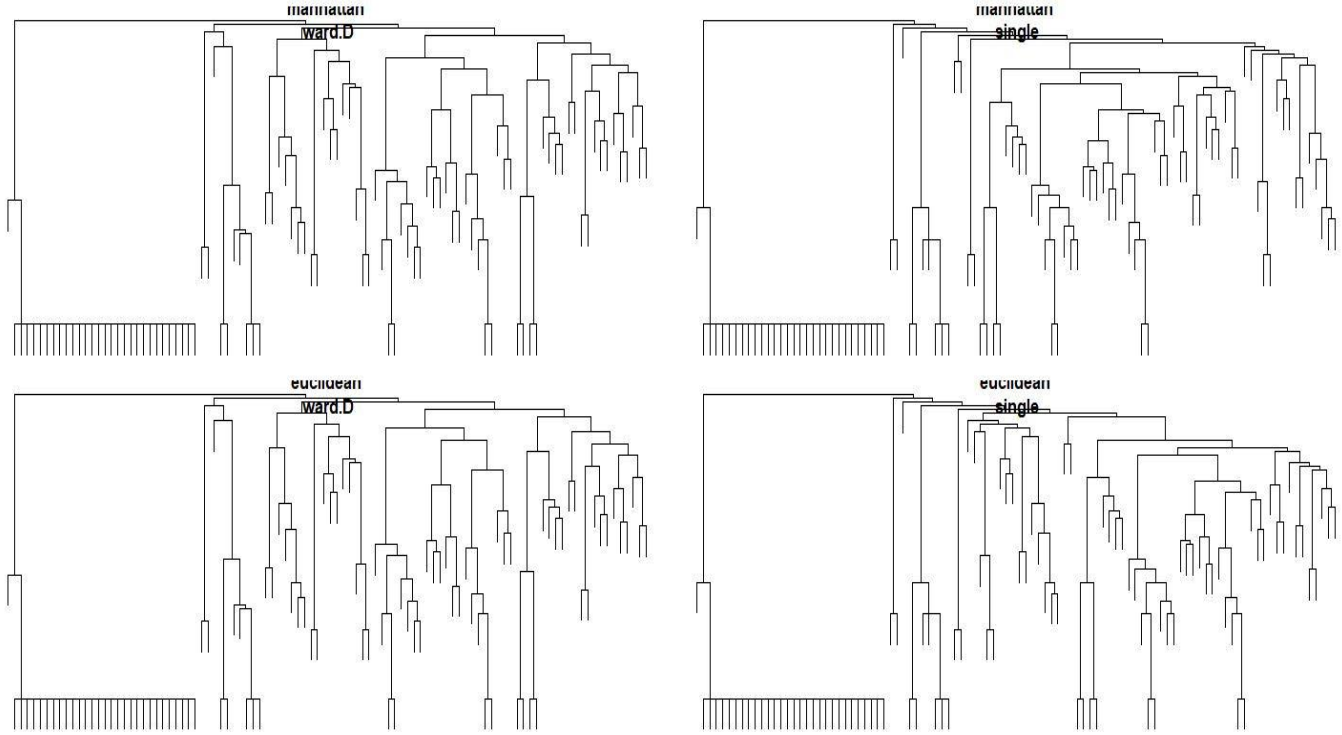


To determine outliers and partition data into equally sized groups I plotted hierarchical clustering with distance methods Manhattan which uses mean absolute error to evaluate the model and Euclidean which uses sum of squares error to evaluate the model. And when it comes to clustering methods, I chose Ward.D and Single. Reason to use ward.D and single methods is that, Ward.D works to minimize standard deviation and does a good job of creating clusters that have approximately equal size groups and Single linkage does a good job of isolating observations that are dissimilar from the rest of the data set, and does a good job in outlier detection.

Since data is huge the dendrogram appeared clumsy initially so I sampled the data with first 100 observations.

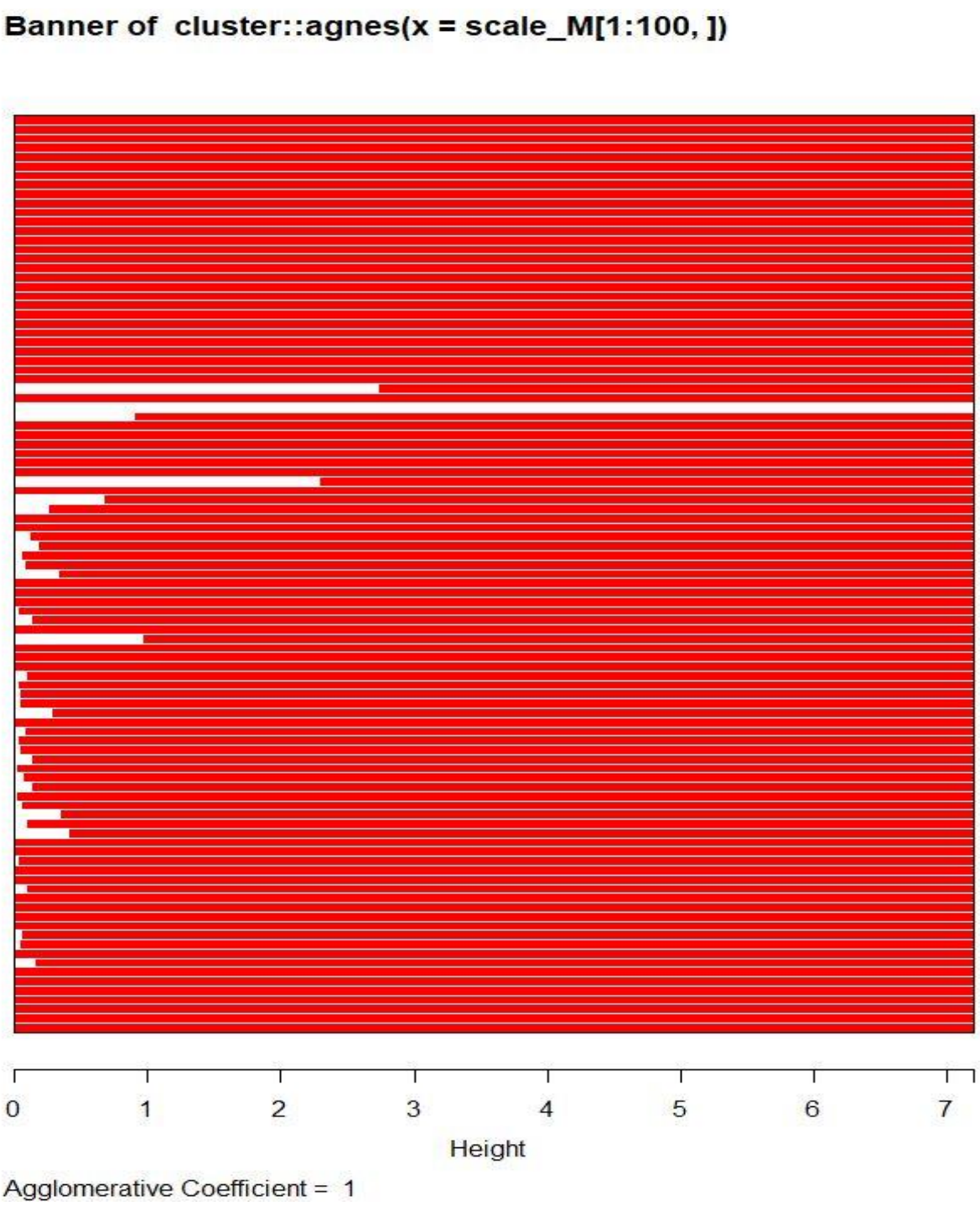


One problem with comparing dendrograms is that the different metrics and different algorithms measure dissimilarity differently. Therefore, I converted the heights to a common scale using rank() function which sets all heights to integer values, makes it easier to compare dendrograms, and spreads out the dendrogram splits.

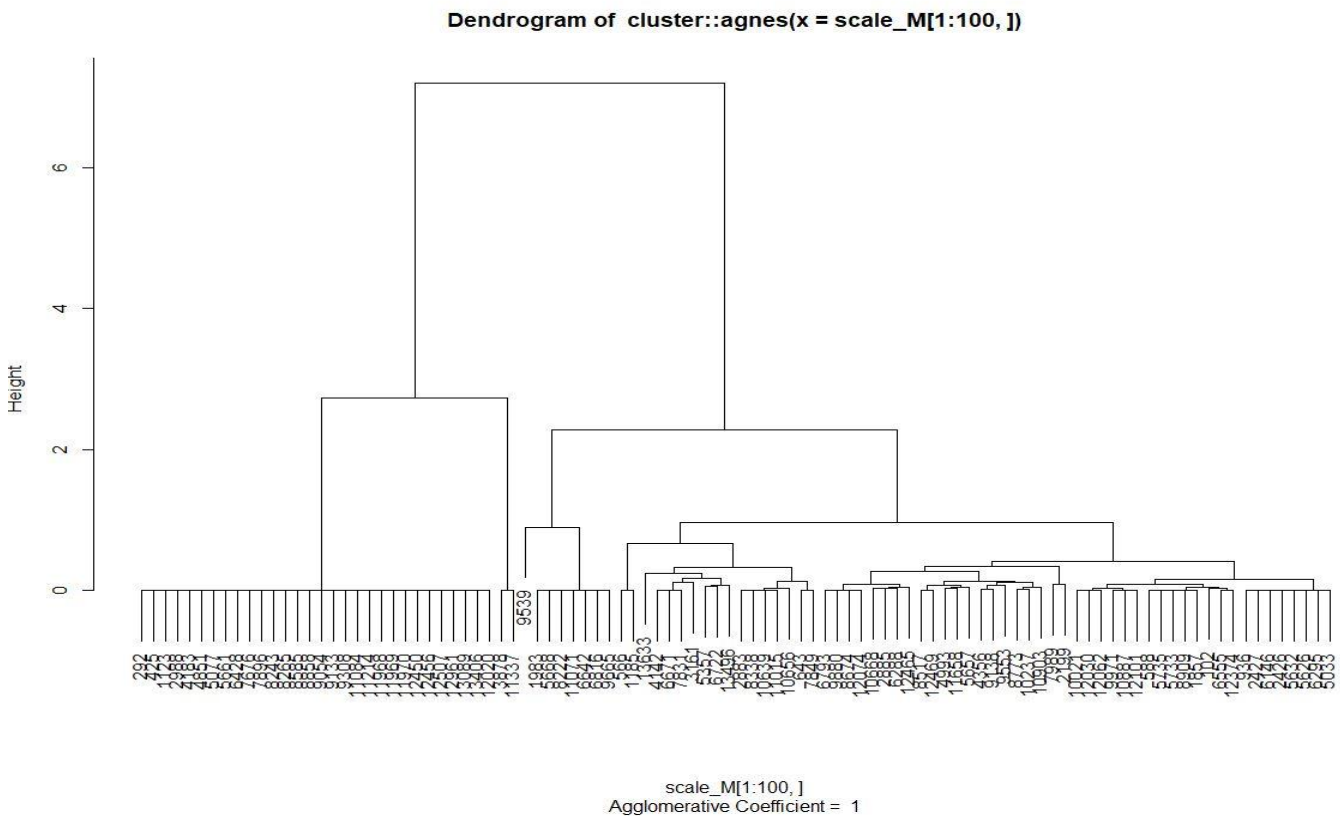


Now the above graph looks clear and if one observes keenly one can notice there are few outliers in Manhattan Single linkage and in Euclidean Single Linkage. When it comes to Manhattan ward.D and Euclidean ward.D one can observe all the clusters are of equal size which contains equal data points.

Another algorithm I chose after hclust() is Agnes algorithm which is similar to hclust(). Since data is huge I sampled it with first 100 observations. The first plot I got after running this is banner plot, which tells when does an observation merges with other observations. The height of the plot is seven and least value is zero



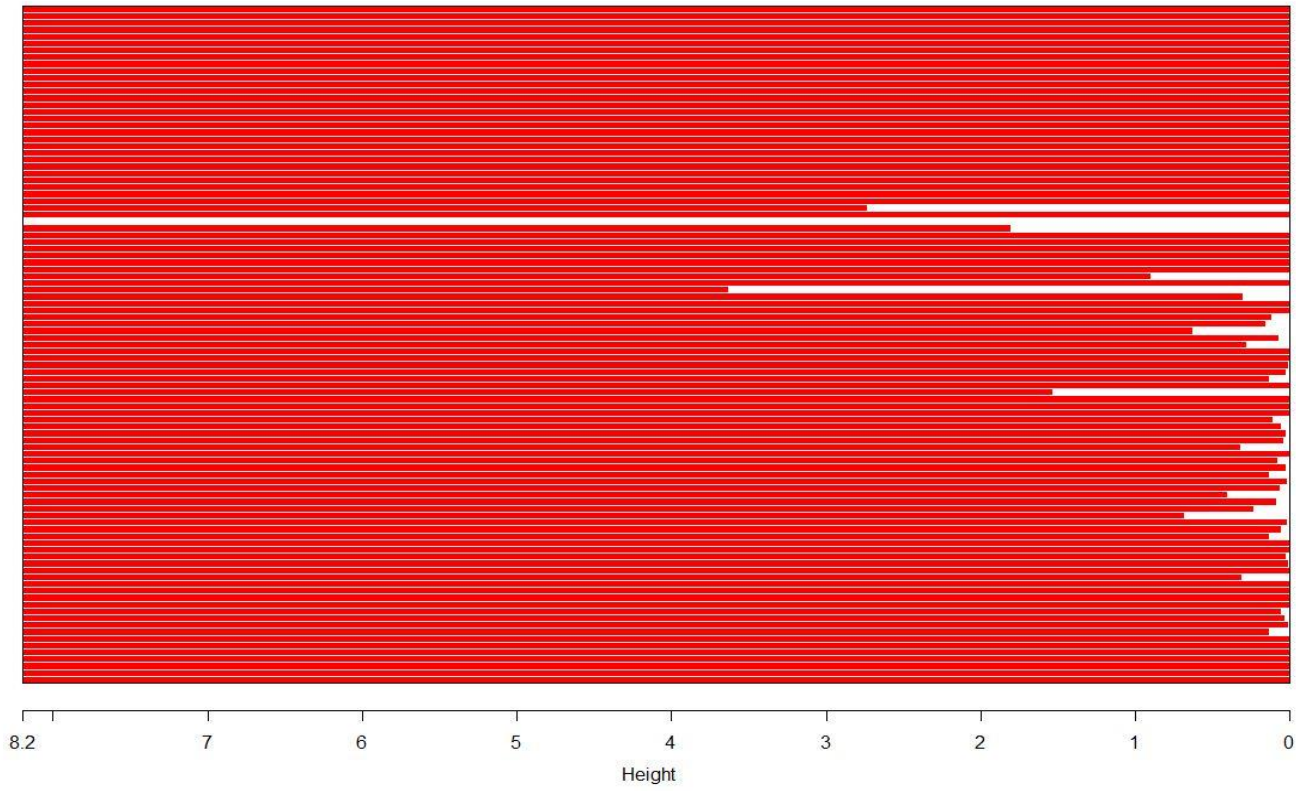
This banner plot represents how much of a difference in the height of a different merging on the dendrogram.



Diana Algorithm

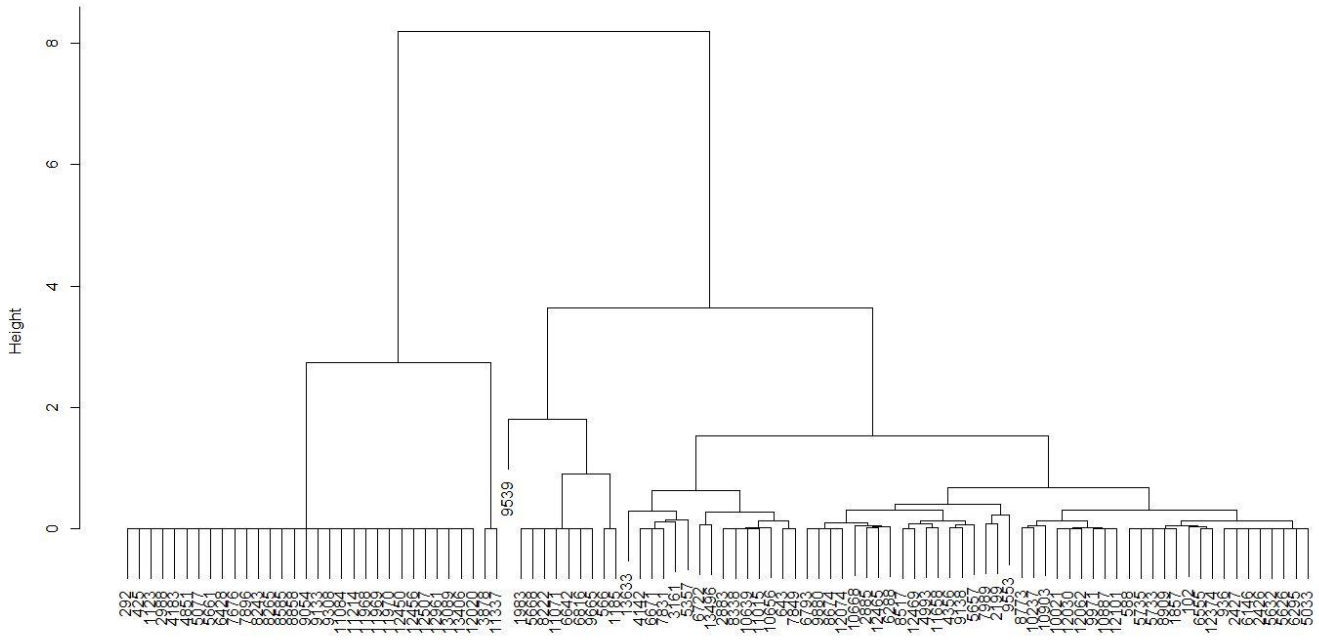
This algorithm is compiled using diana() function in cluster package. Diana is a type of divisive algorithm which divides the data into smaller and smaller subsets.

Banner of cluster::diana(x = scale_M[1:100,])



Divisive Coefficient = 1

Dendrogram of cluster::diana(x = scale_M[1:100,])



scale_M[1:100,]
Divisive Coefficient = 1

I did not do mona() algorithm as it is not suitable for my dataset. So, the final methods I chose is hclust() and agnes() algorithms.

I chose Agglomerative clustering algorithm cause its mechanism suits my dataset, having different features like Average rating, text rating and rating count which determines the best rating a book can be read by more people and becomes popular. **Also, AGNES is sensitive to outliers as well. Other algorithm, hclust() is suitable cause it has suitable metrics to determine the outliers and to partitioning the data into equal sizes such as Single linkage algorithm and ward.D algorithm respectively. I don't think diana() and mona() algorithms would suit my dataset properly as final aim is to determine best book through its ratings and not to determine best ratings through books.**