Team members:

Shishir Agrawal(2016B2A70674H)
Akash Goel(2015A3PS0321H)
Shubhanshu Sharma(2015A8PS0492H)
Anshuman Pati (2016B4A70470H)

Dataset Used:
Amino Acid dataset

Preprocessing Done
Levenshtein Distance is used to calculate distances between acids

1. K-Means/K-Mediods Constants:

K : (=5)

Algorithm: Partitioning Around Medoids (PAM) Initialize: select k of the n data points as the medoids
1.Associate each data point to the closest medoid.
2.While the cost of the configuration decreases:
3.For each medoid m, for each non-medoid data point o:
4.Swap mand o, associate each data point to the closest medoid, recompute the cost (sum of distances of points to their medoid)
5.If the total cost of the configuration increased in the previous step, undo the swap

2. Agglomerative Clustering Linkages used : max,min,average

Algorithm:

Let X = {x1 , x2 , x3 , ..., xn } be the set of data points.

1) Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.

2) Find the least distance pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

3) Increment the sequence number: m = m +1.Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to L(m) = d[(r),(s)].

4) Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new cluster, denoted (r,s) and old cluster(k) is defined in this way: d[(k), (r,s)] = min (d[(k),(r)], d[(k),(s)]).

5) If all points are in one cluster stop else move to step 2

3.Divisive Clustering Algorithm: DIANA

1. Initialize set of clusters S with one set containing all points

2. For each cluster in S find the point which is most dissimilar (point with maximum average distance). Add this point to the splinter set.

3. For each point in the original set compare the distance of the point with the set and its splinter set. If the splinter set is closer than add this point to the splinter set removing from the original set.

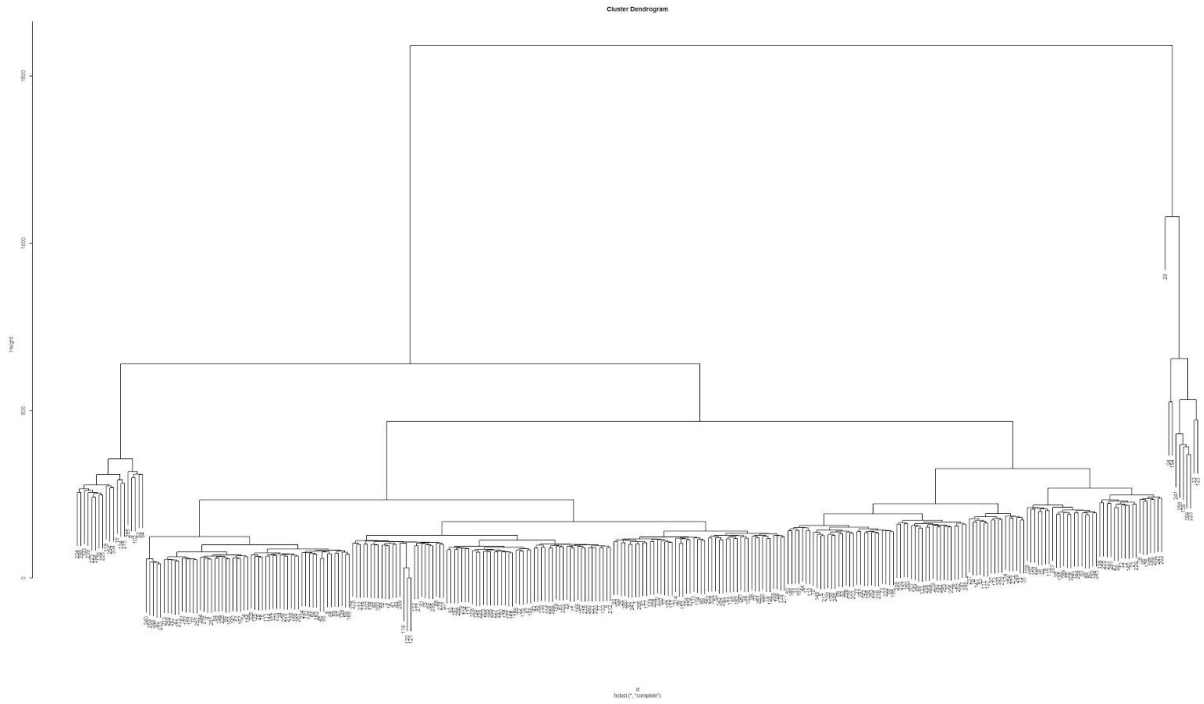
4. Continue until no such point exists. This splits the set into two sets.

5. For complete splitting split all sets until all sets have only a single element.For partial splitting till K split the set in S which is most heterogeneous (most diameter measure)

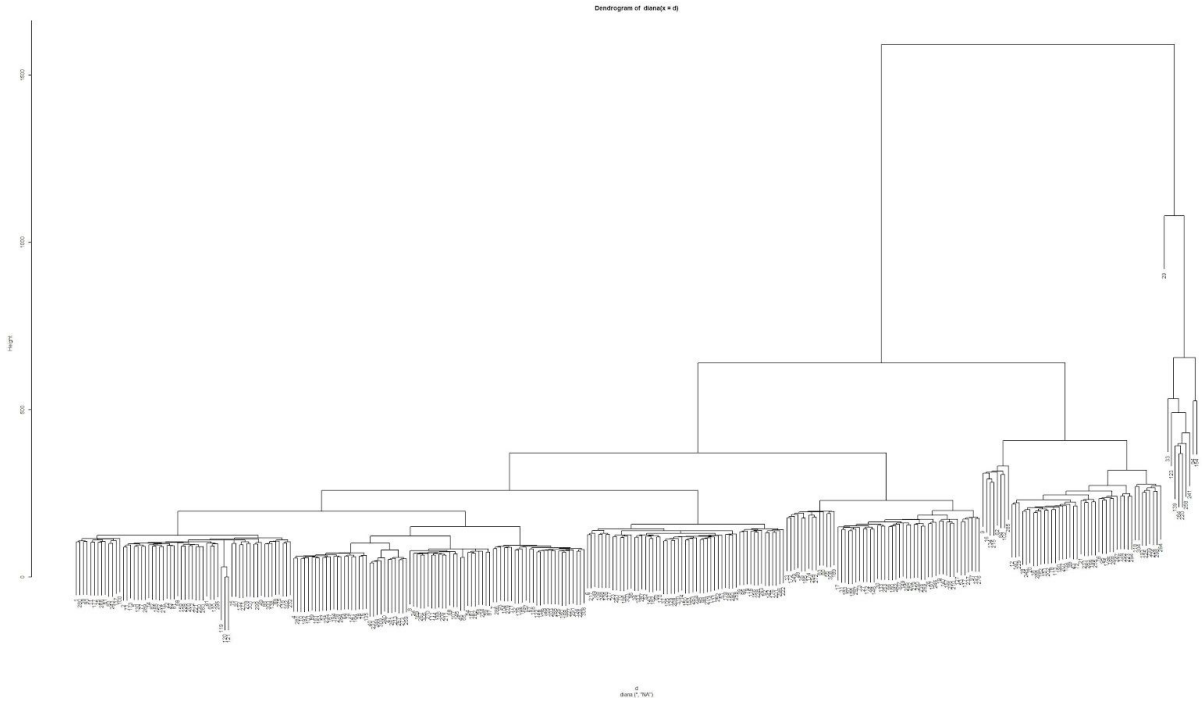
Time: 1.45s (K = 3)

VISUALIZATIONS:

Agglomerative Clustering (Min)



Divisive Analysis Clustering (DIANA)



Conclusion:

The procedure comparisons between K-Means, Agglomerative and Divisive algorithms are done above and in the python programs itself. Here, we compare the results produced by the above three algorithms.

Comparison between K-Means and Hierarchical: For this dataset Agglomerative hierarchical clustering with complete linkage produces the results close to those of K-Means. However, we cannot say one method is superior to the other unless we know the context in which the knowledge of clusters is used.

Comparison between Agglomerative and Divisive clustering (Hierarchical): Agglomerative is simpler to implement. Divisive algorithm results are not close to any of those Agglomerative algorithms' results for this dataset.