1. Introduction/Objective

This project aims at finding and analysing clustering solutions (optimal) obtained by k-means/k-medoids on the bases of different measure such as distance function, number of clusters, cluster size. We will also focus on cluster stability as the above-mentioned algorithms are not completely deterministic and can depend on the initial (typically randomly determined) starting points. The dataset used here is HouseVotes84 which includes votes for each of the U.S house of representative's congressmen on 16 key votes given as V1 to V16 with 435 observation.

2. Background and Research

Clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. It is an unsupervised learning technique which is mostly used in machine learning, segment analysis etc. There are many clustering algorithms which uses different parameters but ultimately boils down to having *minimum intra-cluster distance* and *maximum inter-cluster distance*.

There are various types of clustering namely, **Exclusive Clustering, Overlapping Clustering** and **Hierarchical Clustering** however in this project we will focus on Explicit Clustering and Overlapping Clustering (to some extent).

Explicit Clustering is to cluster the data into partition such that every data point belongs to one and only one cluster, in other words, the cluster membership is either 1 or 0 where as in overlapping clustering the cluster membership is a probabilistic statement or a probability of point belonging to different clusters.

K-Means is the most classic explicit clustering algorithm which uses an iterative refinement technique. It aims at minimizing the objective function

$\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left||x_n - \mu_k|\right|^2$ where z_{nk} is the binary vector with value 1 at position n for nth cluster with minimum $\left||x_n - \mu_k|\right|^2$ which is the distance function K-Means uses. Since the distance is squared it is more influenced by the outliers. **Since it uses squared euclidian distance it does not make much sense to use it over a binary data** thus we need some algorithm that allows us to use select out choice of distance function and gives a generalised cost/objective function.

K-Medoids is a more general clustering algorithm that works in the similar way of K-Means with subtle difference which will be explained further. It uses a cost function $\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} d(x_i, \mu_k)$ with $d(x_i, \mu_k)$ as the distance function which allows us to pass a custom dissimilarity matrix to the algorithm.

2.1 What is K-Medoids?

Partitioning Around Medoids or the K-medoids algorithm is a partitional clustering algorithm which is slightly modified from the K-Means algorithm. They both attempt to minimize the squared-error (as mentioned above) but the K-medoids algorithm is more robust to noise than K-means algorithm. In K-means algorithm, they choose means as the centroids but, in the K-Medoids, data points are chosen to be the medoids.

A medoid can be defined as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal. The difference between k-means and k-medoids is analogous to the difference between mean and median: where mean indicates the average value of all data items collected, while median indicates the value around that which all data items are evenly distributed around it. The basic idea of this algorithm is to first compute the K representative objects which are called as medoids. After finding the set of medoids, each object of the data set is assigned to the nearest medoid. That is, observation x is put into cluster k, when medoid mk_x is nearer than any other medoid m_w.

We will use PAM for implementing K-Medoids. Although PAM does not scale well for large data set it is very much efficient for small dataset with time complexity of $O(k(n-k)^2)$

2.2 How K-Medoids Works?

The algorithm involves 2 steps **Build-Step** and **Swap-Step.** The build step selects k "centrally located" observations, to be as initial medoids unlike the random k points selected by K-Means. Next the Swap-Step checks if the objective/cost function can be reduced by interchanging a selected observation with another observation, if yes, the swap is performed. This continues until the objective/cost function can no longer be decreased.

- Select k random points as the medoids from the given n observations/datapoint of the dataset. (Note that chooses from within dataset)
- Associate each data point to the closest medoid by using the distance measure passed by the user $d(x_i, \mu_k)$
- For each pair of selected object/observations S and non-selected observation N calculate the swap cost as described above
- If the cost i.e. $\mathrm{TC_{SN}} < 0$, S is replaced by N
- Repeat above steps until. $TC_{SN} > 0$ and no replacement can further be done

2.3 Distance Measures (Binary Distance Measure)

The first step of most multivariate analysis is to calculate a matrix of distances or similarities among a set of items in a multidimensional space. A few of the things to be taken care while choosing the distance measures is Noise and Outliers. These can be exaggerated by the distance measure when thy have undue influence on the results perhaps obscuring meaningful patterns. There are different types of distance measure

suited for different type of data such as Sorenson, Relative Sorenson, Euclidean, Relative Euclidean etc but since we have binary data in place, we will not go for the conventional distance measure rather we would go for Binary Similarity/Dissimilarity.

I, J	I (Presence)	0 (Absence)	Sum
I (Presence)	$a = i.j$	b = $\bar{i}.j$	$a+b$
0 (Absence)	c = $i.\bar{j}$	d = $\bar{i}.\bar{j}$	$c+d$
Sum	$a+c$	$b+d$	N = $a+b+c+d$

Suppose that two objects or patterns, I and J are represented by the binary feature vector form. Let N be the number of features (attributes) or dimension of the feature vector. Definitions of binary similarity and distance measures are expressed by Operational Taxonomic Units in a 2 x 2 contingency table where a is the number of features where the values of I and J are both 1 (or presence), meaning 'positive matches', b is the number of attributes where the value of I and J is (0,1), meaning 'I absence mismatches', c is the number of attributes where the value of I and J is (1,0), meaning 'J absence mismatches', and d is the number of attributes where both I and J have 0 (or absence), meaning 'negative matches'. The diagonal sums $a+d$ represents the total number of matches between I and J, the other diagonal sum $b+c$ represents the total number of mismatches between I and J. The total sum of the 2x2 table, $a+b+c+d$ is always equal to N.

3. Implementation

The first part of the implementation is to choose a distance function which is the most important part of PAM (K-Medoids). Although we must use different distance function and compare the result, I would like to explain my choice of distance function and why it would fit best for the binary data we have. Referring to the research paper "A Survey of Binary Similarity and Distance Measure" by Pace University, NY. The inclusion of negative matches d (referring to the 2x2 contingency table) in the similarity measure do not mean necessarily any similarity between 2 observations because an infinite number of attributes is possibly lacking in in two observations. Also looking at our dataset we are more interested in positive matches rather than negative matches. Example of such Similarity Measure includes Jaccard, Sokal & Sneath-I, Faith and dissimilarity measure include Pattern Difference. (All of the mentioned similarity and dissimilarity measures can be referred in the link mentioned above)

Before diving into the implementation of k-Medoids we also need to find the optimal number of clusters. For this I will use *NbClust* method from the package NbClust. NbClust package provides 30 indices for determining the number of clusters and proposes to user the best clustering scheme from the different results obtained by varying all combinations of number of clusters, distance measures, and clustering methods.

As we know that neither the K-Means or K-Medoids is a stable algorithm i.e. we cannot rely on a result returned by a single execution because of the randomness of the starting point and the problem that both may converge to local minima thus we need to find the optimal number of clusters in the data. There could be several methods as named below

- Plotting total within sum of square vs number of cluster (Knee Plot)
- Using NbClust library to determine the optimal number of clusters, it uses a huge number of cluster suitability measuring criteria
- Using vegan package which uses Calinski criterion which is like finding ratio of between-cluster-variance/within-cluster-variance

For this project we will be using the NbClust package and method.

3.1 How does NbClust determines optimal number of clusters

NbClust gives us 30 indices like KL (krzanowski & Lai), Silhouette, C-Index (Hubert's method) etc. which determines optimal number of clusters at their own level and the maximum indices determining x as the optimal number, that is chosen as the optimal number of clusters. For this project we will focus on Silhouette Index

3.2 Silhouette Index (Finding Optimal Clusters)

The silhouette index is defined as:

Silhouette = $\frac{\sum_{i=1}^{n} S(i)}{n}$, $Silhouette \in [-1,1]$

Where $S(i) = \frac{b(i)-a(i)}{max a(i),b(i)}$, $a(i)$ is the average dissimilarity of the i[th] object to all other objects of cluster $C_r$and $b(i)$ is the minimum of average dissimilarity of the i[th] object to all other object ins of cluster C_s.

The maximum value f the index is used to determine the optimal number of clusters, $S(i)$ is not defined for k = 1, i.e. only one cluster hence the min cluster for NbClust is 2.

3.3 Using NbClust

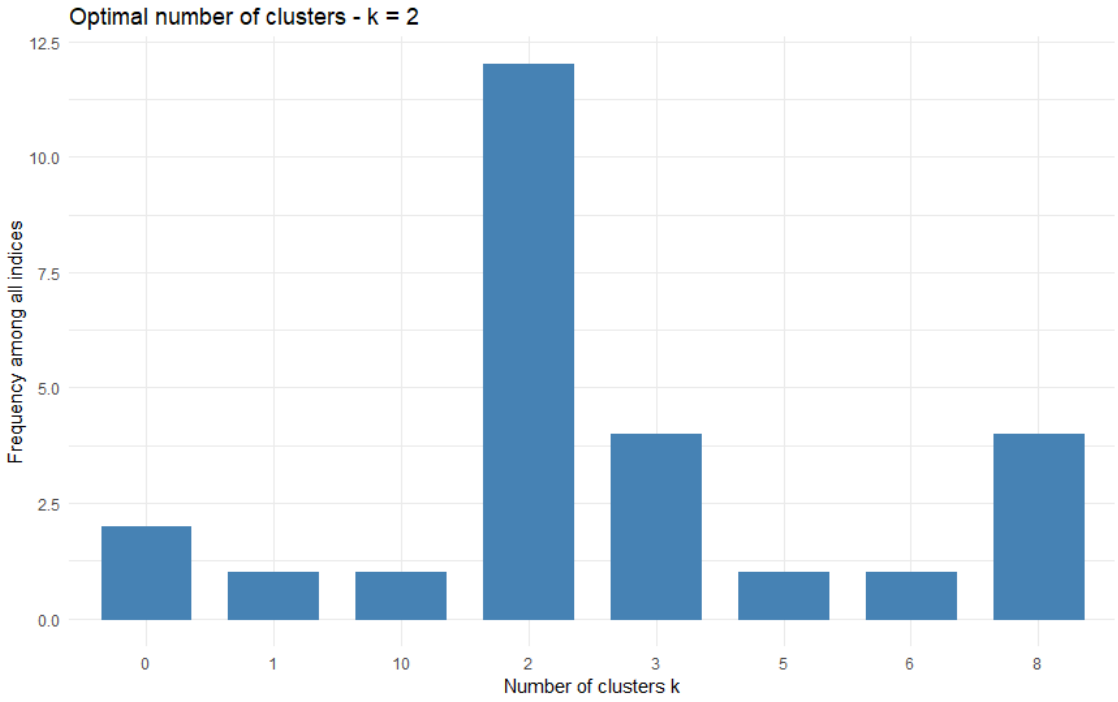
Keeping in mind the above points about the NbClust and the distance measure and our algorithm we will now execute NbClust and find the optimal number of clusters

Note: I have passed the distance measure of my choice via *diss*. I am not interested in negative matches and hence I have chosen Jaccard, Sokal and Dice distance measure as they ignore the negative matches (As of now)

NB <- NbClust (USCD, diss = Sim, min.nc = 2, distance = NULL, max.nc = 10, method = "centroid", index ="all")

We can now print the NB variable to get the details of what it has done, also it gives the values of the optimal clusters thrown out by different measure and se can see under "Silhouette" that the optimal cluster chosen is two. We will also look at the summary using *fviz_nbclust* which will show us better summary with a histogram of what happened. It gives us summary of what ever indices chose as the optimal cluster and chooses the maximum out of it.

Note: this result is when the "Jaccard" distance measure is used. From previous studies of binary data, we wound that Jaccard works best and gives a good rand index (this will be discussed later)



3.3 Running K-Medoids Algorithm

Now that we know that the optimal number of clusters are two, we can go ahead and experiment with K-Medoids and throw in different distance measure and see what the results are and how they vary. (Although we said earlier to not take in consideration negative match, we will verify it here if our assumption is right about the involvement of negative matches in clustering)

- Implemented a function which executes pam with number of clusters 2 (Because we found optimal cluster above)
- This function takes in different methods to execute rand pam with starting from 1 which is Jaccard to 10 which is S2 coefficient of Gower & Legendre
- On each iteration I take out the classifications and create a table with the original data to see how well the algorithm performed
- Once the table is created, I pass it to *classAgreement ()*, and it returns a rand index which is stored in a vector
- Once all iterations are completed, I find the index of max element in the vector that has max rand index (we can sort this in place to find the top performing distance functions on our data) and return it

Note: While running the algorithm I found that 7, 8, 9 methods of *dist.binary()* were not working, also the simple matching gave the max rand index and thus chosen as the best fit for this data.

3.4 Using Silhouette to find the goodness of fit

Now that we have which distance function work, in this case it is "**Simple Matching Coefficient**" $S = (a + d)/(a + b + c + d)$ we will plot a silhouette and clusters to visualize and analyse the clustering based on the same.

4. Interpretations and Findings (Simple Matching)

Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighbouring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighbouring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

- After thoroughly analysing the silhouette plot, we find that we have 10 data points that are wrongly assigned i.e. they have a negative silhouette score.
- The **medoids** at which K-Medoids converged were **83 and 23**
- Size of the 2 clusters are 210 and 225 with max dissimilarity of 0.75 each and average dissimilarity of 0.43557 and 0.44956 respectively
- The **average silhouette width is greater than 0.5** which is good

Considering the data points with negative silhouette vales we need to pay special attention to them and hence we will find out which all are the points having this issue. I used below code snippet to find out the points of interest and observed them.

```
sum(Model_2$silinfo$widths[,3] <= 0)
```

4.1 What if the number of clusters were not optimal

In this case I choose the number of clusters as 3 keeping the best performing distance measure i.e. Simple Matching. The average silhouette width reduces drastically with increasing number of wrongly assigned data points with negative silhouette values.

For K = 3 the average silhouette width is reduced to 0.21 while the silhouette width for 2 clusters were pulled down to 0.12 and 0.08 which are not good values to consider. We see a similar pattern as we keep on increasing the number of clusters, the average silhouette width decreases along with the individual cluster's silhouette width.

4. Conclusion / Extended Work

After the execution of the algorithm and the analysis of the data it was possible to tell that the clusters were well grouped and correlated with the V1:V16 of each data point. In the data there was a total of 435 elements, 267 democrats and 168 from *republican* and the algorithm clustered the elements from *Republican* as cluster 1, the ones from *Democrats* as cluster 2. After verifying the results, we find that out of 435 elements, 379 were correctly clustered, giving an error margin of 12%—which is a very good result.

This project can be extended to analyse why some of the points gave negative silhouette values, although we do have the points of interests, but I could not find relevant information about them.

5. References

Figure A: Silhouette and Clusters Identified by K-Medoids (Best fit with Optimal Clusters)