Erasmus University Rotterdam

Bachelor Thesis

Evaluation of Different Clustering Algorithms: With Social Media Data in Tourism Domain

Author

Ji Heon Kim (445866)

Supervisor

U. Karaca

Second Reader

A. Castelein

July 7, 2019

Abstract

With an ever-growing amount of data, it has been more difficult to implement traditional data analytical methods as they are often computationally intensive. Traditional travel recommender systems (TSR) are no exception, and a possible solution to this problem is adopting clustering analysis. Through clustering, large datasets can be broken down into smaller datasets or clusters, and then methods like TSR can be independently applied to these individual clusters. However, clustering is often explorative and therefore, is often difficult to know which algorithm should be adopted for the datasets at hand. Therefore, both partitioning and hierarchical algorithms are explored to establish the best performing algorithm for the three tourism-related datasets. Algorithms were compared using performance measures like silhouette scores. Few of the algorithms including K-means, PAM, single-linkage and divisive were tested and according to the results, single-linkage and average methods of the agglomerative hierarchical algorithms produce the best performing clusters with the three datasets. This research provides insights into cluster analysis and more specifically, clustering tourism-related datasets.

Contents

1 Introduction

Many industries like E-commerce adopt recommender systems to process their consumer database and provide personalized recommendations to their customers (Schafer, Konstan, & Riedl, 2001). The tourism industry is no exception, as planning a vacation is often explorative. Customers rarely rely on prior experience about a destination as traveling to the same destination where they have personal knowledge about is uncommon (Renjith, Sreekumar, & Jathavedan, 2018). This forces travelers to not only explore the web for relevant information but also to process the gained data for their purpose. And this can cost consumers a significant amount of time. As a result, tourism-related sites often adopt travel recommender systems (TSR) to offer contextualized and personalized suggestions to facilitate the needs of their customers (Berka & Plößnig, 2004). With the ever-growing volume of data, the traditional TSR were proven to be inefficient. This is because TSR adopts highly computationally intensive methods (Jiang, Qian, Mei, & Fu, 2016). However, traditional TSR can still provide valuable insights by minimizing computational intensity and make TSR more applicable to larger datasets. A way to minimize computational time is to use clustering techniques. Clustering breaks down large datasets into smaller clusters by grouping similar elements into a cluster. Then, TSR or other algorithms are independently applied to these clusters, treating each cluster as a smaller dataset. As a result, this decreases the computational intensity as the size of the dataset that the algorithm is applied to is now manageable (Renjith et al., 2018). However, which algorithm one should use to create the "best" clusters for optimal results is still unknown. Researchers believe that there is no single "best" algorithm, but different methods are suitable for different datasets (Willett, 1988). Therefore, the goal of this research is to provide more insights into which clustering algorithm is suitable for the datasets in hand. And also for other similar datasets related to the tourism industry.

To achieve this goal, this paper is mainly divided into two sections, namely the replication and extension parts. First, the replication part is an attempt to reproduce the research by Renjith et al. in 2018. The same sets of clustering algorithms and performance measures used by the authors were explored to solidify their findings. However, I will further develop their achievements by exploring additional algorithms and comparing the performances of different algorithms. And with thorough comparisons, either the conclusions obtained by Renjith et al. (2018) will be re-established or different algorithm that outperforms the one set forth by the authors of the original paper could be suggested as a possible improvement to their results.

This study is relevant and important as U believe that companies and/or individuals with clustering problems can use the findings to decide which algorithm to use. This research also has the potential to help industries other than tourism-related industries. The datasets handled in different industries vary and that one cannot generalize the obtained results for other industries or even datasets. However, the methods used on this paper can be applied in many settings, and one can adopt the same techniques to establish different conclusions for many cases.

The rest of the paper is organized as follows. The next section is used to explore the findings of previous studies and to give readers more background information on clustering techniques and algorithms used for this research. Then, the methodology section will lay out the methods adopted for this research, followed by a short data section describing the datasets used for the experimental analysis. Section 5 contains the results and the interpretation of the performed analysis, and the conclusions made through the collected results will be presented in section 6. Finally, encountered limitations followed by suggestions for future research will be shared to close the paper.

2 Literature Review

2.1 Background Information

Before exploring the findings of previous studies, several questions must be answered first: what is cluster analysis and why and where is it used. Cluster analysis is, in simple terms, the breakdown of large datasets into smaller "clusters" by grouping similar elements together. And it is often performed for classification and this makes data analysis more efficient. There are many examples of data classification: geographical clustering of regions, types of cancer in the field of oncology and archaeological findings by date (Kaufman & Rousseeuw, 2009). As the examples show, clustering groups elements based on unique identifiable features or patterns. This can help in analyzing the data more efficiently as analysis algorithms perform well on smaller-sized datasets. When the number of data points increases, the computational power of these methods increases exponentially, often making them inapplicable to large datasets. Therefore, breaking down datasets into smaller clusters with similar elements can help reduce computation time and make analytical algorithms applicable. Data compression and vector quantization (Gersho & Gray, 1991), data mining and knowledge discovery (Fayyad, Piatetsky-Shapiro, & Smyth, 1996) and pattern recognition and classification (Duda & Hart, 1973) are a few examples of different fields that use clustering analysis.

Clustering uses many similarity measures to quantify the closeness between different objects. Although there is no measure that is universally best for all clustering problems (Huang, 2008), there are a few popular ones: Euclidean, Manhattan, Cosine, Jaccard and Minkowski (Renjith et al., 2018). However, it is important to remember that chosen measures should correspond to the characteristics of the dataset in question. Therefore, choosing a similarity measure depends on the data at hand. The measures and algorithms chosen for this experimental analysis will be further discussed in the later sections.

2.2 Relevant Studies

There are many established algorithms and many studies categorize these algorithms based on how each algorithm operates. For example, Jain and Dubes (1988) categorized them into two main categories: non-exclusive and exclusive. The difference between the two is that for exclusive clustering, element in the dataset can only belong to one subset and for non-exclusive, elements can belong to multiple subsets. For this research, exclusive algorithms will be mainly explored, but an algorithm from the class of non-exclusive is also considered. Within the class of exclusive clustering, it is further categorized into supervised and unsupervised algorithms. Supervised algorithms include linear and logistic regressions, and they need to know the algorithm's possible outcomes *a priori* to the implementation. Unsupervised algorithms, unlike their supervised counterparts, need not be trained beforehand with known outcomes. It will look for similarities by themselves to assign labels to each element for clustering. It is important to note that the methods adopted for this paper are all examples of unsupervised algorithms.

Jain and Dubes (1988) further categorized unsupervised algorithms into partitioning and hierarchical algorithms. Renjith et al. (2018) explored the performance of partitioning algorithms and I will further their research by looking at hierarchical algorithms. But first, what is the difference between the two classes of clustering algorithms? Partitioning algorithms divide the data into clusters in a single partition, whereas hierarchical algorithms partition the data in the nested sequence manner (Jain & Dubes, 1988). Partitioning algorithms form a pre-specified number of clusters, whereas hierarchical algorithms form all clusters by either merging or dividing different clusters.

From each category, there are a few well-established algorithms. First, from the partitioning class, K-means (MacQueen, 1967) and K-medoids (Kaufman & Rousseeuw, 1987) are the two most widely used algorithms. There are two types of hierarchical algorithms: agglomerative and divisive. The difference between the two algorithms will be discussed in the methodology section. Within the agglomerative approach, there are multiple methods: Single-Linkage, Complete-Linkage, and Group Average Method (Michener & Sokal, 1957). Different criterion functions used make these methods distinguishable. Researchers also found that agglomerative algorithms form higher quality clusters, but divisive algorithms are more efficient as they require low computational power (Zhao & Karypis, 2002).

For hierarchical algorithms, no number of clusters need to be set *a priori*, as it partitions the data in the nested sequence manner. However, for partitioning algorithms, which partition the data at once, a specific number of clusters need to be set beforehand. And researchers discovered that to achieve high quality clustering with partitioning technique, it is important to specify the optimal number of clusters for the dataset before the analysis (Milligan & Cooper, 1985). And there are several studies (Dunn, 1974; Kaufman & Rousseeuw, 2009; Milligan & Cooper, 1985) establishing different performance measures that are used establish the optimal number of clusters. In the experimental analysis section, these established results will be used to determine the correct number of clusters.

It is also important to note that there is no “ideal” clustering algorithm that fits all datasets (Willett, 1988). Therefore, researchers need to explore different algorithms and establish which method works best for the particular dataset at hand. For this reason, recent algorithms are more focused on efficiency and scalability to reduce computational costs and increase processing capability, rather than on universality of the algorithms (Jung, Park, Du, & Drake, 2003).

Although there are a copious number of studies on cluster analysis, there are only a few that explore tourism-related datasets. Furthermore, this research extends to mechanisms that were left unexplored in Renjith et al. (2018), the study will hopefully contribute additional insights to this field.

3 Methodology

This research attempts to find the best clustering algorithm for tourism-related datasets. The first part of the research will focus on replicating the results of Renjith et al. (2018). Therefore, algorithms and performance measures adopted in Renjith’s research will be considered first. In the second part of the research, additional algorithms that were not considered in the original paper will be explored. Before delving into the details of the algorithms used, the actual mechanical tools used for this study need to be discussed. **R** is a widely known statistical programming language used for data analysis (Ihaka & Gentleman, 1996). Therefore, the language has many valuable packages that can be used and applied for this study. Further explanations of specific and important functions used will be presented in future sections.

3.1 Partitioning Clustering

Renjith et al. (2018) only explored partitioning algorithms in their study; K-means (MacQueen, 1967), K-medoids (Kaufman & Rousseeuw, 1987), Clustering for Large Applications (CLARA) (Kaufman & Rousseeuw, 2009) and Fuzzy C-Means (Bezdek, Ehrlich, & Full, 1984) algorithms. And the same research was done again to solidify the results obtained by the authors.

3.1.1 Estimating the Optimal Number of Clusters

As mentioned before, it is well established fact that optimal number of clusters needs to be specified in order for partitioning algorithms to perform well and this can be done through the process of cluster validity (Charrad, Ghazzali, Boiteau, & Niknafs, 2014). There are many studies that test the performance of these validity measures. However, for this paper, the same method used by Renjith et al. (2018) will be adopted. Therefore, a package from **R** called **NbClust** (Charrad et al., 2014) was used to determine the optimal number of clusters for the algorithms. The package uses 30 indices including silhouette (Rousseeuw, 1987), Dunn (Dunn, 1974) and Trcovw (Milligan & Cooper, 1985). It uses either K-means or hierarchical agglomerative clustering (HAC) to cluster the data and determine the optimal number of clusters based on the 30 indices (Charrad et al., 2014).

3.1.2 Crisp Algorithms

Crisp clustering algorithms are also known as hard or exclusive algorithms. These algorithms use the conventional set theory in which an element either belongs to a set or not. There is no partial membership. Below are a few famous crisp partitioning algorithms that were explored in this paper.

***K*-means** (MacQueen, 1967) algorithm is one of the most famous clustering algorithms. The algorithm partitions a dataset into k different clusters by maximizing the similarity of elements within a cluster and dissimilarity among k clusters. The standard algorithm described by Hartigan and Wong (1979) defines the total within cluster variation as the sum of squared distances between each member of a cluster and centroid of the cluster. Also, the reason Hartigan-Wong algorithm was used rather than MacQueen's (1967) is because the authors of the original paper also used Hartigan-Wong. Here, centroid is the mean value of all members of the cluster updated in each iteration. Within Cluster Variation is defined as

$$\mathrm{WCV}(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2 \tag{1}$$

where x_i represents an element in cluster C_k and μ_k, the centroid of cluster C_k. K-means algorithm minimizes WCV while assigning an element to a cluster if the sum of the squared distances between that element and to the centroid of the cluster is minimal. Readers can find the detailed algorithm under section 9.1 in the Appendix.

Next, Total WCV (TWCV) (2) also needs to be minimized for better clustering.

$$\text{TWCV} = \sum_{i=1}^{k} \text{WCV}(C_k) = \sum_{i=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2 \tag{2}$$

K-means algorithm has the advantages of easy implementation, flexibility and performance with large datasets. However, it is highly computationally intensive, sensitive to outliers and needs an accurate estimate of the actual number of clusters for better performance. For this research, the number of random sets of centers chosen was set to 50. This particular number was chosen because it produced similar clusters to that of (Renjith et al., 2018). Other inputs were set as default.

K-medoids (Kaufman & Rousseeuw, 1987) works similarly as K-means algorithm. It chooses an element that has the minimal average dissimilarity to all other elements in the dataset as the medoid. One of the most well-known K-medoids algorithms is the Partitioning Around Medoids (PAM) algorithm. PAM searches for k medoids in the dataset and form clusters around each of them. Furthermore, PAM minimizes the objective function, which is the sum of dissimilarities of all the elements in a cluster. The algorithm swaps the current medoid to a newer one after it checks whether the swap results in smaller objective value. Finally, when the function cannot be minimized any further, the algorithm has successfully identified k clusters. Again, readers can find a detailed algorithm for this study under Algorithm 2 in the Appendix section 9.1. It was discovered that default inputs for the function in **R** successfully replicated Renjith's results. Therefore, inputs were left untouched.

CLARA (Kaufman & Rousseeuw, 2009) extends the K-medoids algorithm and is used for larger datasets. Using this algorithm, the shortcomings of K-medoids algorithms like high computing time and large memory requirements can be avoided. This is achieved by the algorithm applying PAM on a sample of the dataset rather than the entire dataset. And if it draws the samples in a sufficiently random way, the identified K-medoids through applying PAM on different samples should correspond to the "true" medoids (medoids found on the whole dataset). Also, CLARA repeats the sampling process to avoid any sampling bias that can arise from applying the same algorithm multiple times on different samples. One can refer to Appendix section 9.1 for a detailed algorithm of CLARA. Also, the default values were used when adopting the CLARA function on **R**.

3.1.3 Fuzzy Algorithms

Zadeh (1965) defined fuzzy sets as sets containing objects with a continuum of grades of membership rather than either/or. There is a function that assigns each object a grade (between 0 and 1) of membership to a set. Since then, fuzzy set theory has been widely applied to clustering algorithms, and many well-known crisp algorithms have been fuzzifed (Chung & Lee, 1994). For this research, I will explore one fuzzy algorithm; Fuzzy C-Means algorithm.

Fuzzy C-Means (Bezdek et al., 1984) assigns an element a degree of membership to a cluster based on the distance of the element from the center of the cluster. If the element is closer to the center of the cluster, it will have higher membership to that cluster. Also, for each element, the sum of memberships across all cluster must be equal to 1.

$$\sum_{k=1}^{c} \mu_k(x_i) = 1 \tag{3}$$

c is the total number of clusters and $\mu_k(x_i)$ is the degree of membership of point x_i to cluster k. Next, the center of a cluster (centroid) is calculated the following way (4) where $\mu_{ik} = \mu_k(x_i)$ and m is the measure of fuzziness with $m \in [0, \infty]$.

$$C_k = \frac{\sum_{i=1}^{n} [\mu_k(x_i)]^m x_i}{\sum_{i=1}^{n} [\mu_k(x_i)]^m} = \frac{\sum_{i=1}^{n} [\mu_{ik}]^m x_i}{\sum_{i=1}^{n} [\mu_{ik}]^m} \tag{4}$$

The algorithm updates the degree of membership in each iteration (5) and minimizes the objective function (6) to get good clustering. d_{ik} is the Euclidean distance between x_i and centroid of cluster k, C_k.

$$\mu_{ik} = \frac{\left[\frac{1}{d_{ik}}\right]^{\frac{1}{(m-1)}}}{\sum_{k=1}^{c} \left[\frac{1}{d_{ik}}\right]^{\frac{1}{(m-1)}}} \tag{5}$$

$$J(U, c_1, c_2, ..., c_k) = \sum_{k=1}^{c} \sum_{j=1}^{n} (\mu_{ik})^m (d_{ik})^2 \tag{6}$$

The detailed algorithm can be found as Algorithm 4 in the Appendix section 9.1.

3.2 Hierarchical Clustering

Hierarchical algorithms are mainly grouped into two categories: agglomerative and divisive. Sneath and Sokal (1973) characterized agglomerative clustering as sequential, agglomerative, hierarchic, non-overlapping methods (SAHN). It start off by placing each element in their own cluster and every iteration the algorithm merges clusters close to each other and eventually form one large cluster with all entities. Divisive takes the opposite approach. It starts off with one large cluster and in each iteration, it splits the cluster to form clusters containing one entity. Unlike partitioning algorithms, hierarchical algorithms do not require a pre-determined number of clusters as they construct multiple partitions. Researchers need to decide which partition is optimal through graphical and other analysis. Both approaches will be explored for this study.

3.2.1 Agglomerative Clustering

For agglomerative algorithms, it is important to provide the criterion that the algorithm can use to determine the pairs of clusters to merge. There are many criteria functions available from past studies, but for this research, the four most commonly used criteria will be considered: Single-Linkage (SL), Complete-Linkage (CL), Group Average Method (UPGMA) and Ward's method. For simplicity, Euclidean distance will be used as the measure of similarities between elements. Also, it is important to note that for SL, CL and UPGMA one may specify other similarity measures such as the cosine similarity within vector-space model, but for Ward's method one must specify a distance measure.

Single-Linkage or also called the nearest neighbor method, measures the minimum distance (i.e. maximum similarity) between the two elements in the clusters that are being investigated. This is mathematically defined as

$$Sim(S_i, S_j) = \min_{d_i \in S_i, d_j \in S_j} \|d_i - d_j\| \tag{7}$$

Complete-Linkage or furthest neighbor method, it measures the maximum distance (i.e. minimum similarity) between the two elements in the clusters that are being investigated.

$$Sim(S_i, S_j) = \max_{d_i \in S_i, d_j \in S_j} \|d_i - d_j\| \tag{8}$$

As one can notice from (7) and (8), the two schemes base their merging decision solely on two elements in each cluster rather than considering all the data points inside the clusters in question. More precisely, CL uses the elements with the minimum similarity as the similarity measure between two clusters, whereas SL considers data points with maximum similarity. Therefore, they produce under-performing results (Zhao & Karypis, 2002). Therefore, one should also consider Group Average (UPGMA) and Ward's methods.

UPGMA (Michener & Sokal, 1957) determines the cluster similarity by comparing the average distance between all the elements in two clusters (9). This overcomes the limitations faced by single and complete linkage schemes, as all the data points in a cluster are considered in the merging decision-making process.

$$Sim(S_i, S_j) = \frac{1}{n_i n_j} \sum_{d_i \in S_i, d_j \in S_j} \|d_i - d_j\| \tag{9}$$

n_i denotes the size of cluster S_i.

Ward's (Ward, 1963) method is the only agglomerative approach that bases its clustering on the sum-of-square criterion function. As mentioned above, unlike the previous three approaches, a distance measure must be specified as the measure of similarity. For this paper, again, Euclidean distance was used for Ward's criterion.

$$Sim(S_i, S_j) = \sum_{d_i \in S_i, d_j \in S_j} \|d_i - d_j\|^2 \tag{10}$$

Ward's method minimizes the squared Euclidean distance as shown in 10. One can notice that the criterion above is like that of K-means partitioning clustering shown in 1. And for this reason, Ward's method is known to produce similar results as K-means clustering. However, Ward's method is performed with no structural constraint such as clustering embeddedness and thus can be used to create clustering which is used as the "starting point" for K-means (Murtagh & Legendre, 2014). Then the clustering created by Ward's method can be improved by applying K-means to it, and this is a much more direct and efficient approach of K-means. Although this approach will not be tested in this study, it could be something that can be further explored in the future.

There are two functions (**agnes()** and **hclust()**) in **R** that one could choose when implementing the mentioned clustering techniques. For this paper, function **agnes ()** from the **cluster** package in **R** was used. For SL, CL and UPGMA, it is a matter of preference to use **agnes()** or **hclust()**. However, for Ward's method, the two functions produce different results depending on the inputs specified by the users. Murtagh and Legendre (2014) state that **hclust()** function only produces the same result as **agnes()** when one gives ward.D2 as the input. This is because the specification ward.D for **hclust()** does not implement Ward's criterion as stated in 10. Thus, for this reason and consistency, **agnes()** function is used for all the methods.

3.2.2 Divisive Clustering

As mentioned above, divisive clustering adopts the top-down approach. It starts off by placing all the data points in the dataset in one cluster, then dividing the cluster based on the similarity measure (Euclidean distance in this case). The algorithm terminates when each data point in the dataset is placed in its own cluster. For this research, **diana()** function of **cluster** package was used.

3.3 Cluster Evaluation

After forming the clusters with the mentioned algorithms, it is important to compare their performances with different criteria. The main criteria used for this paper are Silhouette Width (Rousseeuw, 1987) and Dunn index (Dunn, 1974).

Silhouette calculates the average distance between the clusters formed and reports a value between -1 and 1. Silhouette 1 shows that the element is well allocated in the appropriate cluster with intra-cluster dissimilarity being smaller than the lowest inter-cluster dissimilarity. And naturally, silhouette value -1 shows the opposite, the worst possible clustering.

The Dunn index is another popular measure to determine the quality of clusters formed. Dunn index is defined as the ratio between minimum inter-cluster distance and maximum intra-cluster distance. Intra-cluster distance could also be considered as the diameter of the cluster and to achieve high quality clustering, the diameter of each cluster should be as small as possible. Therefore, with the largest possible inter-cluster distance and the smallest possible intra-cluster distance, Dunn index should be maximized.

Dunn2 is another index from the Dunn Index family, Entropy measures the distribution of cluster memberships and Wb.Ratio is defined as the average within similarity of the cluster by average between similarity of clusters. These measures were included as they were also included in the paper by Renjith et al. (2018) but they will not be heavily discussed when comparing performances of different algorithms.

Furthermore, it was stated that hierarchical algorithms do not require an optimal number of clusters to be specified *a priori*. However, to compare hierarchical with partitioning algorithms, the same number of clusters for each dataset will be used. Then, using the same performance measures, the clusters will be compared.

4 Data

Three real-time social media datasets were used for the experimental analysis. Data points were collected from online reviews and ratings of different tourist attractions on three different geographical locations representing customers' interests. Table 1 below gives more detail on each dataset used. All the datasets used were retrieved from UCI Machine Learning Repository.[1]

Table 1: Summary of the three datasets

	Dataset 1	Dataset 2	Dataset 3
Description	The dataset contains destination reviews across South India published by users on holidayiq.com until October 2014.	The dataset contains rating information on destinations across East Asia collected by TripAdvisor.com. Each rating is mapped from Excellent (4) to Terrible (0) and average rating is used against each category per user.	The dataset contains average Google ratings for different destination categories across Europe.
Brief Overview	Dataset contains more than 1500 reviews by 249 reviewers and each of them falls into one of 6 destination related categories.	Dataset contains 980 ratings on 10 different destination related categories.	Dataset contains 5456 records with 24 destination related categories.

In the original paper, it was stated that they normalized the data with **R** functions such as **scale()** to ensure that no one large data point influences the clustering process. However, it was found that not scaling the data gives similar results obtained by the original authors. This may be due to several reasons. One clear reason could be that, as most of the data points are ratings of a tourist sight, they will all have a similar range. For example, dataset 2 contains data points that range from 0 to 4. And as most data points have similar values, they need not be scaled.

5 Experimental Analysis

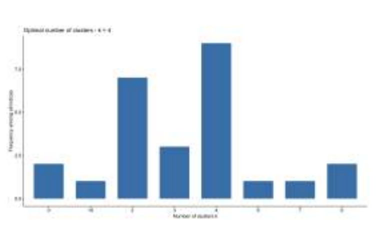
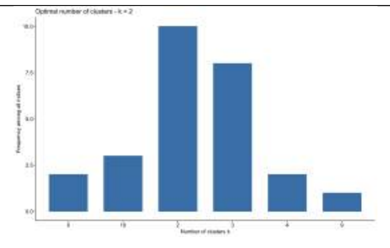
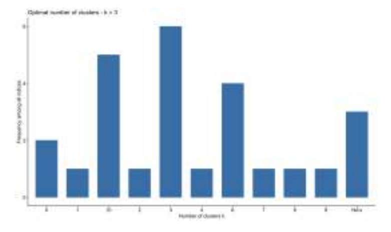
In this section, the results of different algorithms for each dataset will be presented along with a detailed interpretation. First, the results of the partitioning algorithms will be presented followed by the results for hierarchical algorithms.

5.1 Partitioning Algorithms

As discussed in the methodology section, **NbClust()** function was used to determine the optimal number of clusters for the datasets. One needs to give several inputs to initialize the function, such as the method of clustering and similarity measures. To get the same results as Renjith et al. (2018), the "complete" approach was specified for the clustering method and Euclidean distance for the similarity measure and 10 as the maximum number of clusters. Table 2 shows that 4, 2 and 3 are the recommended number of clusters for datasets 1, 2, and 3, respectively.

[1]https://archive.ics.uci.edu/ml/index.php

Table 2: Display of NbClust Results

Dataset 1	Dataset 2	Dataset 3
		

For datasets 1 and 3, the same number of recommended clusters as Renjith et al. (2018) was obtained. However, for dataset 2, the result was different. Their results suggested that the optimal number of clusters for dataset 2 was 8, whereas the function suggested 2 on my attempt. For further analysis, this difference is taken into consideration, and I will discuss their respective performances to establish the optimal number of clusters for this dataset.

Next, using *K*-means, PAM, CLARA and fuzzy C-means (FCM) algorithms, clustering was performed. Then, as displayed in Table 3, **fviz_cluster()** function was used to graph the clusters. Again, results for datasets 1 and 2 are like what Renjith et al. (2018) presented in their paper.

Table 4 summarizes the results obtained for silhouette analysis. As discussed earlier, silhouette width indicates the average distance between the clusters formed and takes the value 1 when the best clusters are produced.

Table 3: Clusters generated by Partitioning Algorithms

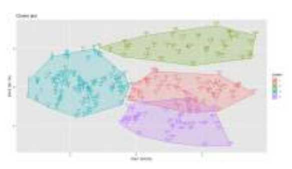
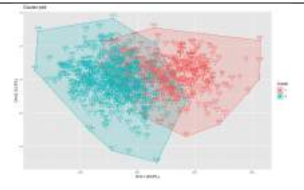
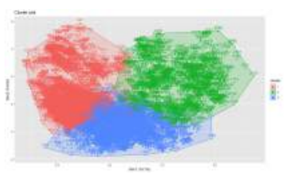
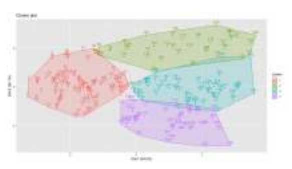
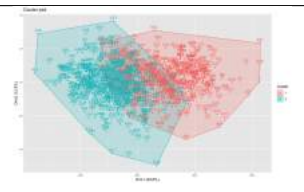
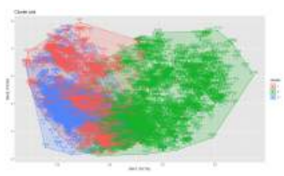
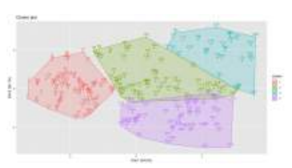
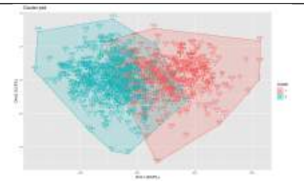
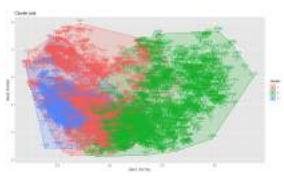
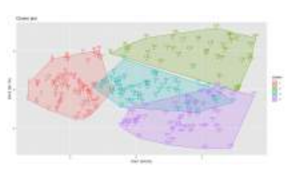
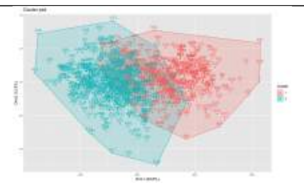
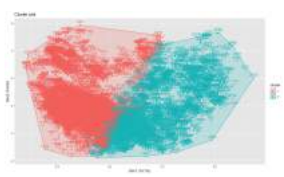
	Dataset 1	Dataset 2	Dataset 3
K-means			
K-medoids			
CLARA			
Fuzzy CM			

Table 4: Results of Silhouette Analysis for Partitioning Algorithms

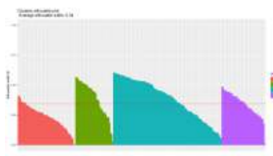
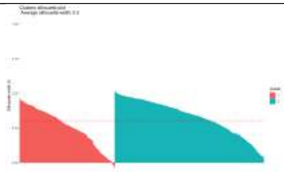
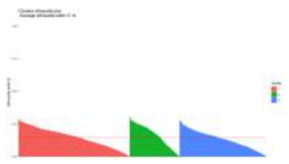
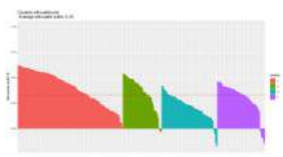
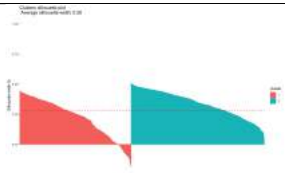
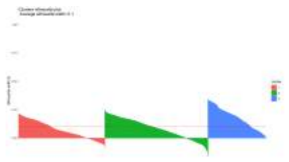
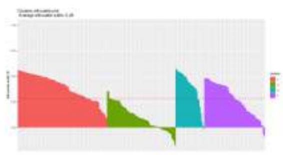
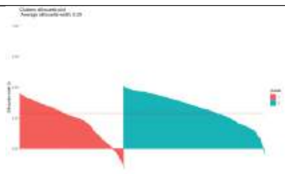
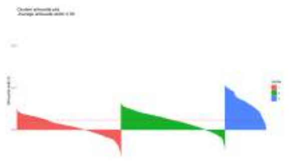
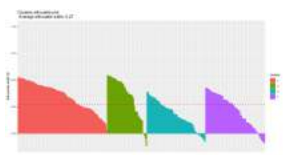
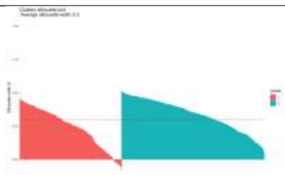
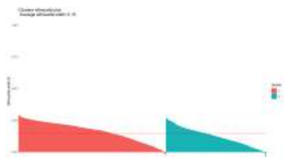
	Dataset 1	Dataset 2	Dataset 3
K-means			
K-medoids			
CLARA			
Fuzzy CM			

Table 5: Performance Measures for Partitioning Algorithms

	Algorithms	Avg. SilWidth	Dunn	Dunn2	Entropy	Wb.Ratio
Dataset 1	*K*-means	0.342257	0.068746	1.430012	1.294179	0.525884
	K-medoids	0.330981	0.020773	1.384847	1.306847	0.531002
	CLARA	0.281170	0.047632	1.316865	1.318452	0.557269
	Fuzzy CM	0.274475	0.056989	0.557269	1.345672	0.552480
Dataset 2	*K*-means	0.300853	0.040848	1.368354	0.668657	0.682086
	K-medoids	0.283778	0.068858	1.318059	0.689110	0.700440
	CLARA	0.288060	0.065830	1.333161	0.68170	0.694519
	Fuzzy CM	0.296033	0.064801	1.347550	0.679078	0.687029
Dataset 3	*K*-means	0.144376	0.012788	1.168382	1.049308	0.812093
	K-medoids	0.10160	0.001119	0.990476	1.072336	0.863801
	CLARA	0.089959	0.012444	0.970902	1.026605	0.872036
	Fuzzy CM	0.146124	0.005367	1.143908	0.674584	0.846393

Table 5 reports the values of different performance measures used: silhouette width, Dunn, Dunn2, Entropy and Wb.Ratio. These measures were calculated using **cluster.stats()** function. From Table 4 and Table 5 one can notice that for datasets 1 and 2, *K*-means approach gave the highest silhouette score. For dataset 3, the FCM method gave the highest score, but *K*-means was close with the difference between the two being smaller than 0.02. Therefore, solely basing on silhouette analysis, *K*-means is the best performing algorithm for all three datasets. Next, one can derive a similar conclusion looking at the Dunn index. For datasets 1 and 3, K-means has the highest Dunn index. This was not the case for dataset 2, where K-means had the lowest Dunn index. However, for two out of three datasets, K-means gave the highest Dunn index score, proving that K-means performs well.

As mentioned before, for dataset 2, **NbClust()** suggested different number as the optimal number of clusters as Renjith et al. (2018). Table 6 shows the results obtained by the authors for dataset 2 with 8 as the optimal number of clusters. Now, comparing their values, one can establish the "true" optimal number of clusters for dataset 2.

Table 6: Performance Measures obtained by Renjith et al. for Dataset 2

	Algorithms	Avg. SilWidth	Dunn	Dunn2	Entropy	Wb.Ratio
Dataset 2	K-Means	0.305087	0.009681	0.902415	2.033419	0.307399
	K-Medoids	0.282551	0.010317	0.814189	2.026279	0.307633
	CLARA	0.284243	0.017442	0.857498	2.050770	0.316814
	Fuzzy CM	0.280955	0.012449	0.808759	2.037319	0.308307

First, the same conclusion was made regarding K-means algorithm outperforming other algorithms for dataset 2. Comparing table 5 and 6, it can be concluded that the values for average silhouette width obtained by Renjith et al. (2018) and for this research were very similar; on average, the difference between the values was less than 0.005. This shows that cluster similarity with 2 as the optimal number of clusters as opposed to 8 is indistinguishable. This result is surprising, as it was expected that the silhouette score to be higher with an increase in the number of clusters. The conjecture is plausible as clusters have silhouette width of 1 ("perfect" clustering) when intra-cluster dissimilarity (compact clustering) is smaller than the smallest inter-cluster dissimilarity (clusters as far away from each other as possible). And if the number of clusters increases, the intra-cluster dissimilarity was expected to decrease as the dataset gets divided into more but smaller clusters. Therefore, it is natural to expect low intra-cluster dissimilarity for 8 clusters as opposed to 2 and thus resulting in higher silhouette width for Renjith et al. (2018). However, the shown results of similar average silhouette width could be explained by the low inter-cluster dissimilarity of their clusters. In short, clusters in the original paper were located relatively close to each other compared to clusters obtained on this research.

The above explanation is further supported by the Dunn index values. My Dunn index values were significantly higher than those of Renjith et al. (2018). As mentioned above, Dunn index is the ratio between inter and intra-cluster dissimilarity and should be maximized by having largest possible inter-cluster dissimilarity and smallest possible intra-cluster dissimilarity. It was also established that Renjith's clusters should have lower intra-cluster dissimilarity because of smaller-sized clusters. Therefore, lower Dunn indices even with smaller intra-cluster dissimilarities, show that clusters of Renjith et al. (2018) have relatively low inter-cluster dissimilarity.

Table 7: Performance Measures obtained with 8 as the number of clusters

	Algorithms	Avg. SilWidth	Dunn	Dunn2	Entropy	Wb.Ratio
Dataset 2	K-Means	0.168207	0.044551	0.757441	1.965414	0.593278
	K-Medoids	0.117727	0.020978	0.930114	2.040207	0.645674
	CLARA	0.117727	0.020978	0.930114	2.040207	0.645674
	Fuzzy CM	0.033346	0.022517	0.842731	1.836772	0.677664

Now, "hypothetical" performance measures will be discussed. Table 7 reports the values I would have obtained if 8 were given from **NbClust()** function. Also, although the difference between the clusters will be mainly discussed through the performance measures, one can find the cluster plots and results of silhouette analysis of dataset 2 with 8 clusters in Appendix 9.2 under table 12. Now, comparing tables 6 and 7, one can notice, even after creating 8 clusters, the values obtained during this research are very different to that of Renjith et al. (2018). For K-means, K-medoids and CLARA, the average silhouette width obtained for this research is almost 50% lower than that of Renjith's. For FCM, it was about 88% lower than Renjith's. However, Dunn values reported in Table 7 are higher than the values in Table 6. Both results are unexpected, as the entire analysis was performed with the same dataset as Renjith et al. (2018) using the methods. This issue should be further investigated for clarification. However, this is beyond this research and will be more discussed in the Limitations section.

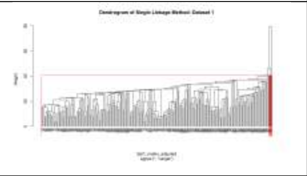
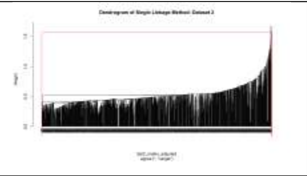
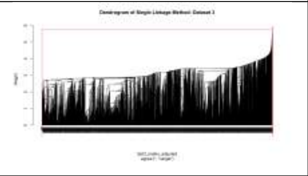
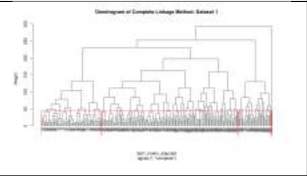
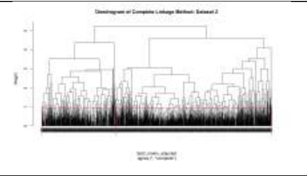
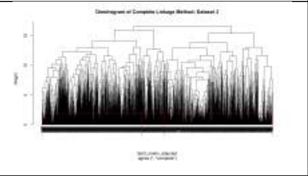
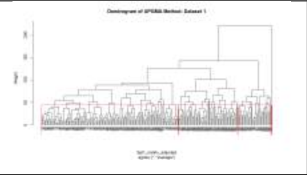
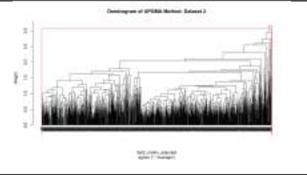
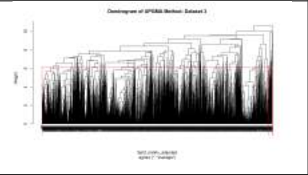
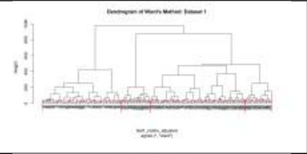
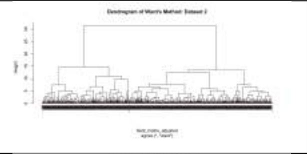
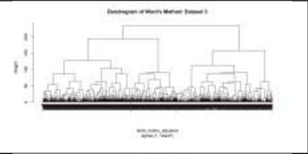
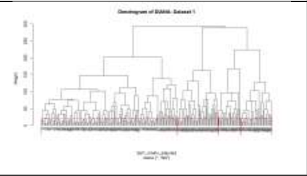
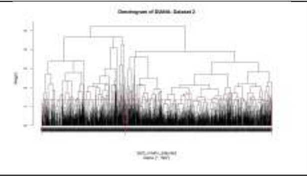
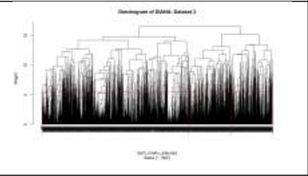
In conclusion, 2 as the optimal number of clusters seems to be better for this dataset. This is because the 2 clusters obtained initially (Table 5) not only outperform Renjith's clusters (Table 6) but also the "hypothetical" clusters (Table 7).

5.2 Hierarchical Algorithms

For hierarchical algorithms, "dendrograms" were used to examine the clusters. Dendrograms are tree diagrams that show which clusters are merged (the agglomerative approach) or separated (the divisive approach) in each iteration. The vertical axis represents the relative closeness (similarity) between the clusters and by looking at when the two lines from each cluster join, one deduce at which point the two clusters were merged or separated. And that point shows how close those two clusters were. The table below shows the dendrograms for different hierarchical clustering techniques for three datasets.

The red boxes represent how each data point was placed into a pre-specified number of sub-clusters through these algorithms. One can deduce that the size of a box represents the size of the respective cluster. This finding is solidified by comparing the dendrograms with the cluster plots in Table 9. Taking dataset 2 as an example, it can be seen that for SL and UPGMA methods, there is one large cluster and one cluster with just a single element. The respective dendrogram also conveys the same information with one large rectangle and the last element in its own box. Next, a few trends that can be noticed from the dendrograms presented in Table 8 will be discussed.

Table 8: Dendrograms obtained by different Hierarchical Algorithms

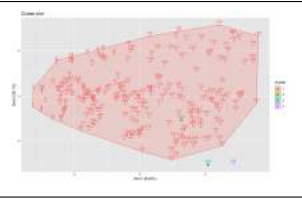
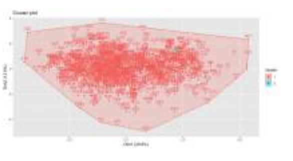
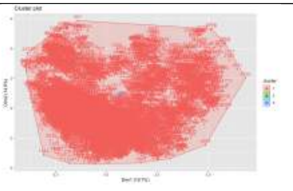
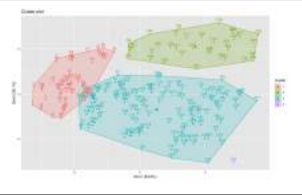
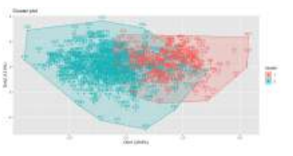
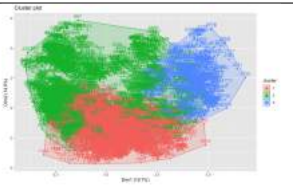
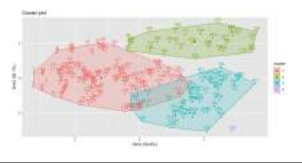
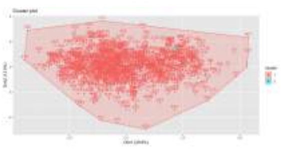
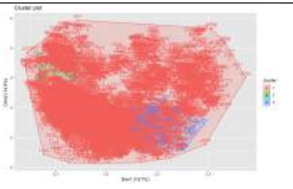
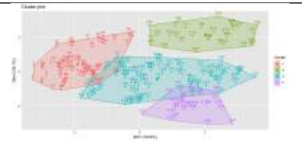
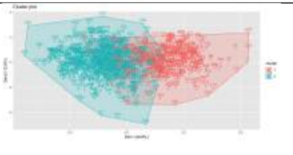
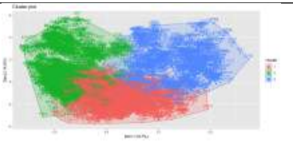
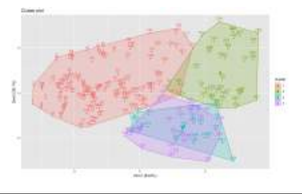
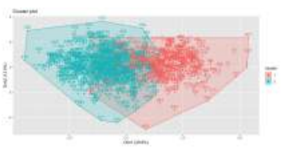
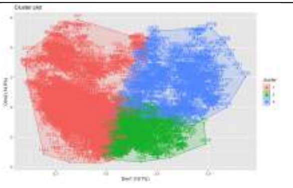
	Dataset 1	Dataset 2	Dataset 3
Single			
Complete			
Average			
Ward			
Divisive			

First, dendrograms formed by the SL method has the shape of an exponential graph. The similarity maximization mechanism of the single-linkage method may explain this shape. The method merges two closest clusters, and because of this, it often merges clusters nearer the boundaries. This results in long, thin shaped clusters. Furthermore, this "domino-like" merging pattern could be causing the exponential shapes on dendrograms. This could also be the reason the SL method places a majority of the elements in one cluster and fewer data points in other clusters. However, it is important to note that this observed pattern is unique to the datasets at hand and is not a general rule. To give more conclusive remarks on this, further research on the SL algorithm should be performed, but this is beyond this study.

Second, from Table 8 and 9, it can be seen that data point 244 in the dataset forms its own cluster with agglomerative methods. This shows that this element is placed further away from the rest of the data points, signalling that it differs from the rest of the data points.

Also from comparing the dendrograms, it can be noticed that Ward method is the only algorithm that distributed the data points relatively fairly compared to other hierarchical algorithms. Ward method did not form clusters with just one element or placed the majority of the data points in one cluster. The same criterion function as K-means could be causing this observed pattern, creating similar clusters as K-means. This will be further explored later with performance measures.

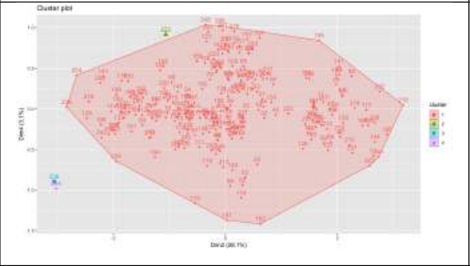
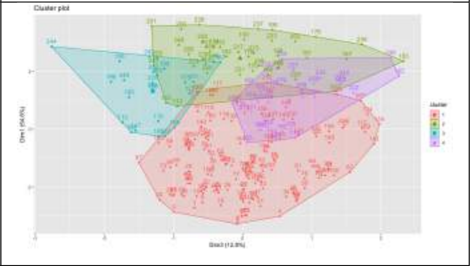
Table 9: Cluster Plots for Hierachical Algorithms

	Dataset 1	Dataset 2	Dataset 3
Single			
Complete			
Average			
Ward			
Divisive			

Now, the shapes of the clusters formed will be considered. Although the number of clusters need not be pre-specified for hierarchical algorithms, to "cut" the tree and form a specific number of sub-clusters for analysis, a number needs to given. Furthermore, to compare hierarchical methods with partitioning ones, the same number of clusters needs to be used. With this given number, the **cutree()** function was used to form sub-clusters. Table 9 shows the cluster plots produced by hierarchical algorithms. It can be seen that the clusters formed with hierarchical algorithms do not have similar shapes to those formed with partitioning algorithms, with Ward's method being an exception. This is probably because partitioning and hierarchical algorithms adopt very different approaches in grouping the data producing differently shaped clusters. However, again, as Ward's method uses the same criterion function as K-means, clusters produced by Ward's method strongly resemble those formed by K-means.

It is also interesting to note that some clusters are entirely overlapped by another cluster. For example, in Table 9, the purple and blue clusters are located in the same position, signalling that they are similar. This made me question why the algorithm did not merge the two clusters. And this is because of the dimensionality of the plots. The plots presented above are plotted with dimensions 1 and 2. However, dataset 1 has six attributes, meaning that the dataset forms a six-dimensional space. Therefore, different dimensions need to be examined to get a clearer understanding of where exactly the clusters are located. As a result, the clusters that showed overlapping traits for dataset 1 (Single and Divisive) with different dimensions were plotted again, as shown in Table 10.

Table 10: Cluster Plot on different dimensions for Dataset 1

	Single	Divisive
Dataset 1		

For the single-linkage method, overlapping clusters separate into different locations when dimensions 2 and 4 were chosen. For the divisive method, dimensions 1 and 3 were chosen to separate the overlapping clusters. This analysis could be further explained by linking it to what the actual attributes represent in the dataset. For single-linkage method, the overlapping clusters could contain elements with an equivalent number of reviews on sports complex and religious institutes (attributes 1 and 2) and as a result, placing them in the similar region when plotted with respect to those attributes. However, when plotted with respect to reviews on religious institutes and theaters (attributes 2 and 4) the overlapping clusters are not overlapping anymore showing that the clusters are dissimilar enough to be two “separate” clusters but just have certain similar attributes. Although, due to time constraint, it was not possible to perform the same investigation on datasets 2 and 3 but it is something that could be further researched to provide more perspective on the clusters formed.

Next, table 11 quantifies the performances of different hierarchical algorithms through the same measures used for partitioning algorithms.

Table 11: Performance Measures for Hierarchical Algorithms

	Algorithms	Avg. SilWidth	Dunn	Dunn2	Entropy	Wb.Ratio
Dataset 1	Single	-0.028252	0.160830	1.018122	0.078451	0.668097
	Complete	0.249552	0.107526	1.316133	0.963495	0.643577
	UPGMA	0.324723	0.143225	1.531099	0.960173	0.577432
	Ward	0.249880	0.103181	1.227228	1.242127	0.598573
	Divisive	0.256108	0.082046	1.090172	1.114772	0.614906
Dataset 2	Single	0.410669	0.317504	1.791281	0.008048	0.558260
	Complete	0.274864	0.071774	1.401415	0.627177	0.712848
	UPGMA	0.410669	0.317504	1.791281	0.008048	0.558260
	Ward	0.279313	0.078543	1.287716	0.662340	0.696410
	Divisive	0.295324	0.069094	1.288207	0.654120	0.679564
Dataset 3	Single	0.122825	0.339032	1.211943	0.005027	0.806841
	Complete	0.097531	0.063444	1.095375	0.945593	0.864492
	UPGMA	0.087262	0.204934	1.160448	0.096570	0.857410
	Ward	0.12870	0.133506	1.152602	1.054129	0.827442
	Divisive	0.132637	0.039734	1.136231	0.907941	0.833059

As it can be seen from the table above, for dataset 1, UPGMA is performing well in terms of average silhouette width followed closely by CL and the divisive algorithm. UPGMA also performs well in terms of the Dunn index, whereas the divisive algorithm performs poorly according to the Dunn index. For dataset 2, it is interesting to see that the SL and UPGMA produced the best performing clusters with identical scores. This is unusual but not surprising, as the cluster shapes presented in Table 9 show that the SL and UPGMA methods produced identically shaped clusters. This is something that is quite unique to this dataset. For dataset 3, the divisive algorithm is the best performing algorithm in terms of silhouette scores, followed by the SL method with the second highest silhouette score. However, SL has the highest Dunn index, whereas divisive does not have a very high Dunn index value. Therefore, it can be concluded that, for dataset 3, SL method is the best suited method.

Furthermore, it is interesting to note that UPGMA does not always outperform SL and CL. For dataset 1, UPGMA outperforms the other two but for datasets 2 and 3, this is not true. It was mentioned in the methodology section that UPGMA often outperforms SL and CL as it bases its merging decision on all the elements in the clusters. However, again, this is true on average and does not apply to every dataset, as evidenced by Table 11.

Now, the obtained results with those of the partitioning algorithms will be compared. For datasets 1 and 3, partitioning algorithms on average have a higher silhouette score, whereas for dataset 2, hierarchical algorithms produced clusters with higher silhouette scores. However, for all three datasets, clusters through hierarchical algorithms have higher Dunn index. Also, it can be easily seen from Tables 5 and 11 that the difference in Dunn indices between the two classes of algorithms is on average larger than the difference in silhouette scores. This suggests that although hierarchical algorithms produce slightly lower silhouette scores than partitioning algorithms, they produce much higher Dunn indices to offset the difference in silhouette scores.

The observation could be supported by considering the definitions of the two performance measures. They are both based on intra-and inter-cluster dissimilarities, with Dunn being the ratio of the two. Higher silhouette scores could mean that the partitioning algorithms may have produced more compact clusters (smaller intra-cluster dissimilarity) than their hierarchical counterparts. However, higher Dunn indices for hierarchical algorithms show that although the clusters may not be as compact as the ones produced by partitioning algorithms, but the inter-cluster dissimilarities must be higher than the ones produced with partitioning algorithms. Furthermore, bigger difference between Dunn indices compare to that of silhouette scores shows that the lower inter-cluster dissimilarities for partitioning algorithms is much more significant than the higher intra-cluster dissimilarities of hierarchical algorithms. So intuitively, this could be interpreted as that hierarchical algorithms are better suited for the datasets at hand. However, again, it is important to note that this finding is only valid for the given datasets and no conclusion such as "hierarchical algorithms are better than partitioning algorithms" can be derived.

Finally, the results obtained by Ward's method to those of K-means will be compared. It was expected the measures obtained through these two methods to be similar, as they share the criterion function. However, it can be deduced from the tables that they are not very close to each other. In fact, the measures for the divisive method seem to be closer to measures of K-means than Ward's method. There may be supporting explanations for this observation, or it could be that this finding is unique to the datasets we have. And this is something that could be researched in the future.

6 Conclusion

This paper was an attempt to replicate the results of Renjith et al. (2018) and to further develop their findings. And as a result, the algorithms described in the original paper and additional methods were successfully explored on three tourism-related datasets from the UCI Machine Learning Repository. First, it was established that for dataset 2, forming two clusters seem perform better than producing eight clusters as presented by Renjith et al. (2018). Aside from that, I managed to re-establish all the results of the author and concluded that K-means produced best clusters among the explored partitioning algorithms.

Next, as an extension, many hierarchical algorithms were explored. Both agglomerative and divisive algorithms were tested for this research. It was concluded that UPGMA method performed the best for dataset 1, SL and UPGMA for dataset 2 and SL for dataset 3. So, it can be said that SL and UPGMA are best performing algorithms for the datasets at hand. Furthermore, it was successfully established that, on average, hierarchical algorithms produced better performing clusters than partitioning algorithms. Therefore, I concluded that hierarchical algorithms are better suited for the datasets used for the experimental analysis. However, this research was far from perfect. One should remember that the results found can only provide insights into the explored algorithms and no general rule can be derived for all existing datasets, partitioning and hierarchical algorithms. In the next section, more specific limitations faced will be discussed.

7 Limitations and Suggestions for Future Research

There are multiple limitations faced at different stages of the study. First, even when the same function on the same dataset was used, I failed to obtain the same results as Renjith et al. (2018). Although the difference in the results was successfully justified, but the fact that different results were obtained with the same datasets is something that needs to be further looked into. The dataset in question (dataset 2) could have been contaminated or simply different from the dataset that Renjith and the team used. Both options are possible, as the dataset was downloaded from an open-source library.

Second, only silhouette scores and the Dunn index were used to make conclusions about the algorithms. However, there are many measures available from different studies. Different conclusions could have been made if other measures were used. There are also studies showing that certain measures outperform the Dunn index, proving that more measures need to be considered if more concrete conclusions are to be made.

Third, I only managed to look at overlapping clusters for dataset 1 due to time constraints. However, it would provide so much more valuable insights into the clusters formed if similar dimensional analysis could be performed on the other two datasets. Through this, one could see which attributes are significant in forming clusters.

Last, if there was sufficient time, it would have been better to implement other algorithms myself. For example, there are many variations of K-means, such as K-means based on simulated annealing or other hierarchical algorithms that do not have a pre-implemented function for. By implementing algorithms, I would have had more algorithms to compare producing more sound results.

Therefore, for future research, more algorithms and performance measures should be implemented. And if possible, it would be helpful to implement the algorithms on more datasets. I believe that this will make the research more general and applicable to different situations.

8 Acknowledgement

I would like to thank Mr Utku Karaca for his aid and counsel during the writing process of this paper. The discussions with him helped me get a better grasp of the methodology and guided me to a feasible direction for the research.

References

Berka, T., & Plößnig, M. (2004). Designing recommender systems for tourism. In Proceedings of enter (p. 26-28).

Bezdek, J. C., Ehrlich, R., & Full, W. (1984). Fcm: The fuzzy c-means clustering algorithm. Computers Geosciences, 10(2), 191 - 203.

Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). Nbclust: An r package for determining the relevant number of clusters in a data set. Journal of Statistical Software, Articles, 61(6).

Chung, F. L., & Lee, T. (1994). Fuzzy competitive learning. Neural Networks, 7(3), 539 - 551.

Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis. New York: John Willey & Sons.

Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. Journal of Cybernetics, 4(1), 95-104. doi: 10.1080/01969727408546059

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). Knowledge discovery and data mining: Towards a unifying framework. In Proceedings of the second international conference on knowledge discovery and data mining (pp. 82–88). AAAI Press.

Gersho, A., & Gray, R. M. (1991). Vector quantization and signal compression. Norwell, MA, USA: Kluwer Academic Publishers.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-Means clustering algorithm. Applied Statistics, 28(1), 100–108.

Huang, A. (2008). Similarity measures for text document clustering. Proceedings of the 6th New Zealand Computer Science Research Student Conference.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5(3), 299-314. doi: 10.1080/10618600.1996.10474713

Jain, A. K., & Dubes, R. C. (1988). Algorithms for clustering data. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Jiang, S., Qian, X., Mei, T., & Fu, Y. (2016, March). Personalized travel sequence recommendation on multi-source big social media. IEEE Transactions on Big Data, 2(1), 43-56.

Jung, Y., Park, H., Du, D., & Drake, B. L. (2003, Jan 01). A decision criterion for the optimal number of clusters in hierarchical clustering. Journal of Global Optimization, 25(1), 91–111.

Kaufman, L., & Rousseeuw, P. (2009). Finding groups in data: an introduction to cluster analysis. Wiley.

Kaufman, L., & Rousseeuw, P. J. (1987). Clustering by means of medoids. North Holland.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth berkeley symposium on mathematical statistics and probability, volume 1: Statistics (pp. 281–297). University of California Press.

Michener, C. D., & Sokal, R. R. (1957). A quantitative approach to a problem in classification. Evolution, 11(2), 130-162. doi: 10.1111/j.1558-5646.1957.tb02884.x

Milligan, G. W., & Cooper, M. C. (1985, Jun 01). An examination of procedures for determining the number of clusters in a data set. Psychometrika, 50(2), 159–179. doi: 10.1007/BF02294245

Murtagh, F., & Legendre, P. (2014, Oct 01). Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? Journal of Classification, 31(3), 274–295. doi: 10.1007/s00357-014-9161-z

Renjith, S., Sreekumar, A., & Jathavedan, M. (2018, Dec). Evaluation of partitioning clustering algorithms for processing social media data in tourism domain. In 2018 ieee recent advances in intelligent computational systems (raics) (p. 127-131). doi: 10.1109/RAICS.2018.8635080

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53 - 65.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. Data Mining and Knowledge Discovery, 5(1), 115–153.

Sneath, P. H. A., & Sokal, R. R. (1973). Numerical taxonomy: The principles and practice of numerical classification. San Francisco, W.H. Freeman and Company., USA: W. H. Freeman.

Ward, J. H. J. (1963). Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, 58(301), 236-244. doi: 10.1080/01621459.1963.10500845

Willett, P. (1988). Recent trends in hierarchic document clustering: A critical review. Information Processing Management, 24(5), 577 - 597.

Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338 - 353.

Zhao, Y., & Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. In Proceedings of the eleventh international conference on information and knowledge management (pp. 515–524). New York, NY, USA: ACM. doi: 10.1145/584792.584877

9 Appendix

9.1 Algorithms

Algorithm 1 *K*-means Aglorithm

```
1. Set the total number of clusters, k
2. Determine initial k centroids to start with. Random sampling is a conventional choice
3. For each element in the dataset:
    a. Calculate the distance between each of k centroids and the element
    b. Put the element in the cluster with the closest centroid
4. For each k clusters:
    a. Update the centorids by calculating the new mean.
5. Minimize WCV. Repeat steps 3,4 until clusters remain unchanged or maximum number of iterations
is reached =0
```

Algorithm 2 *K*-medoids Algorithm

```
1. Set the total number of clusters, k
2. Determine initial k medoids to start with. Unlike K-means, medoids are always chosen from the
members of the dataset
3. For each element in the dataset:
    a. Calculate the distance between each of k medoids and the element
    b. Put the element in the cluster with the closest medoid
4. For each k clusters:
    a. Look for an element that will increase the similarity within the cluster
    b. If found, set the newly found element as the new medoid for the cluster
5. If a new medoid was found in step 4, go to step 3 and if not, end the algorithm =0
```

Algorithm 3 CLARA Algorithm

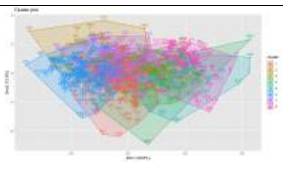
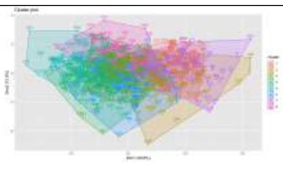
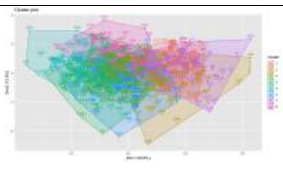
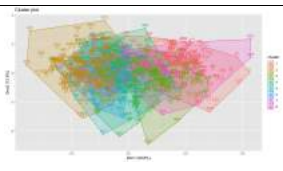
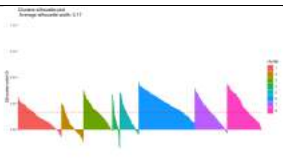
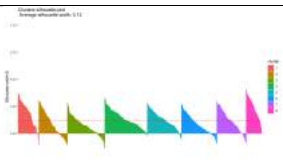
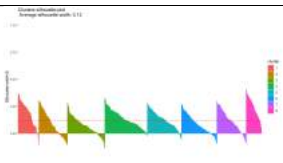
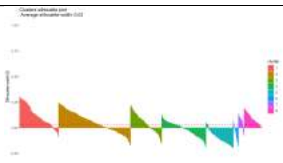
```
1. Form number of subsets of the dataset
2. For each subset, apply PAM algorithm then choose corresponding K-medoids for the subset.
3. For each element in the dataset:
    a. Calculate the distance between each of k medoids and the element
    b. Put the element in the cluster with the closest medoid
4. For each subset:
    a. Evaluate the goodness of the clustering by calculating the average dissimilarities of the elements
to their closest medoid
5. Select the subset with the minimum dissimilarity =0
```

Algorithm 4 Fuzzy C-Means Algorithm

```
1. Set the total number of clusters, k
2. Determine initial k centroids to start with.
3. Calculate the membership matrix for each element in the dataset for k initial clusters. Note that
sum of membership of an element to k clusters must equal to 1
4. For each k clusters:
    a. Calculate new centroids with new membership status of the elements
5. For each element in the dataset:
    a. Calculate the new dissimilarity between the element and the the new centroid
    b. Update the membership matrix
6. Minimize the objective function. Repeat steps 4 and 5 until there is no change in formed centroids
or the maximum number of iterations has been reached =0
```

9.2 Cluster Plots and Silhouette Plots for Dataset 2

Table 12: Clusters And Silhouette Plots for optimal number of clusters as 8

	K-means	K-medoids	CLARA	Fuzzy C-means
ClustPlot				
SilPlot				

9.3 Codes used

Below are the codes written on **R** for this research. It is categorized for each dataset.

9.3.1 Dataset 1

```
Set1_matrix <- data.matrix(Set1)
Set1_matrix_adjusted <- Set1_matrix[,-1]


res1<-NbClust(Set1_matrix_adjusted, diss=NULL, distance = "euclidean", min.nc=2, max.
    ↪ nc=10,
              method = "complete", index = "all")
fviz_nbclust(res1, method = "silhouette", diss = NULL, k.max = 10, nboot = 100,
            verbose = interactive(), barfill = "steelblue", barcolor = "steelblue",
            linecolor = "steelblue", print.summary = TRUE)
```

```
set.seed(76964057)

#Kmeans
res1_kmeans <- kmeans(Set1_matrix_adjusted, centers = 4, nstart = 50, algorithm = "
    ↪ Hartigan-Wong")
fviz_cluster(res1_kmeans, data = Set1_matrix_adjusted)

#PAM
res1_pam <- pam(Set1_matrix_adjusted, k = 4, diss = FALSE, metric = "euclidean", stand
    ↪ = FALSE, do.swap = TRUE)
fviz_cluster(res1_pam, data = Set1_matrix_adjusted)

#CLARA
res1_clara <- clara(Set1_matrix_adjusted, k = 4, metric = "euclidean", stand = FALSE,
    ↪ pamLike = FALSE)
fviz_cluster(res1_clara, data = Set1_matrix_adjusted)

#FCM
res1_fcm <- fanny(Set1_matrix_adjusted, 4, metric = "SqEuclidean")
fviz_cluster(res1_fcm, data = Set1_matrix_adjusted)

#Silhouette Analysis Kmeans
sil1_kmeans <- silhouette(res1_kmeans$cluster, dist(Set1_matrix_adjusted))
fviz_silhouette(sil1_kmeans, label = FALSE, print.summary = FALSE)

#Silhouette Analysis PAM
sil1_pam <- silhouette(res1_pam$cluster, dist(Set1_matrix_adjusted))
fviz_silhouette(sil1_pam, label = FALSE, print.summary = FALSE)

#Silhouette Analysis CLARA
sil1_clara <- silhouette(res1_clara$cluster, dist(Set1_matrix_adjusted))
fviz_silhouette(sil1_clara, label = FALSE, print.summary = FALSE)

#Silhouette Analysis FCM
sil1_fcm <- silhouette(res1_fcm$cluster, dist(Set1_matrix_adjusted))
fviz_silhouette(sil1_fcm, label = FALSE, print.summary = FALSE)

#Cluster Analysis Kmeans
dd1 <- dist (Set1_matrix_adjusted, method = "euclidean")
cluster.stats(dd1, res1_kmeans$cluster, alt.clustering = NULL, silhouette = TRUE)
```

```
#Cluster Analysis PAM
cluster.stats(dd1, res1_pam$cluster, alt.clustering = NULL, silhouette = TRUE)

#Cluster Analysis CLARA
cluster.stats(dd1, res1_clara$cluster, alt.clustering = NULL, silhouette = TRUE)

#Cluster Analysis FCM
cluster.stats(dd1, res1_fcm$cluster, alt.clustering = NULL, silhouette = TRUE)

#Agglomerative Hierarchical Algorithms

#Singly-Linked
ah_single_1 <- agnes(Set1_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, method = "single")
pltree(ah_single_1, cex = 0.6, hang = -1, main = "Dendrogram␣of␣Single␣Linkage␣Method:
    ↪ ␣Dataset␣1")
rect.hclust(ah_single_1, k =4, border = "red")
dend_sl_1 <- as.dendrogram(ah_single_1)
sub_grp_sl_1 <- cutree(ah_single_1, k=4)
fviz_cluster(list(data = Set1_matrix_adjusted, cluster = sub_grp_sl_1), axes = c(2,4))
cluster.stats(dd1, sub_grp_sl_1, alt.clustering = NULL, silhouette = TRUE)

#Complete-Link
ah_complete_1 <- agnes(Set1_matrix_adjusted, diss = FALSE, metric ="euclidean", stand
    ↪ = FALSE, method = "complete")
pltree(ah_complete_1, cex = 0.6, hang=-1, main = "Dendrogram␣of␣Complete␣Linkage␣
    ↪ Method:␣Dataset␣1")
rect.hclust(ah_complete_1, k =4, border = "red")
dend_cl_1 <- as.dendrogram(ah_complete_1)
sub_grp_cl_1 <- cutree(ah_complete_1, k=4)
fviz_cluster(list(data = Set1_matrix_adjusted, cluster = sub_grp_cl_1), axes = c(1,3))
cluster.stats(dd1, sub_grp_cl_1, alt.clustering = NULL, silhouette = TRUE)

#UPGMA
ah_upgma_1 <- agnes(Set1_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, method = "average")
pltree(ah_upgma_1, cex = 0.6, hang = -1, main = "Dendrogram␣of␣UPGMA␣Method:␣Dataset␣1
    ↪ ")
rect.hclust(ah_upgma_1, k =4, border = "red")
```

```
dend_upgma_1 <- as.dendrogram(ah_upgma_1)
sub_grp_upgma_1 <- cutree(ah_upgma_1, k=4)
fviz_cluster(list(data = Set1_matrix_adjusted, cluster = sub_grp_upgma_1))
cluster.stats(dd1, sub_grp_upgma_1, alt.clustering = NULL, silhouette = TRUE)

#Ward's
ah_wards_1 <- agnes(Set1_matrix_adjusted, diss = FALSE, metric = "euclidean", stand =
    ↪ FALSE, method = "ward")
pltree(ah_wards_1, cex = 0.6, hang = -1, main = "Dendrogram␣of␣Ward's␣Method:␣Dataset␣
    ↪ 1")
rect.hclust(ah_wards_1, k =4, border = "red")
dend_wards_1 <- as.dendrogram(ah_wards_1)
sub_grp_wards_1 <- cutree(ah_wards_1, k =4)
fviz_cluster(list(data = Set1_matrix_adjusted, cluster = sub_grp_wards_1))
cluster.stats(dd1, sub_grp_wards_1, alt.clustering = NULL, silhouette = TRUE)

#Divisive Hierarchical Algorithm

#DIANA
dh_diana_1 <- diana(Set1_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, stop.at.k = FALSE)
pltree(dh_diana_1, cex = 0.6, hang = -1, main = "Dendrogram␣of␣DIANA:␣Dataset␣1")
rect.hclust(dh_diana_1, k =4, border = "red")
dend_diana_1 <- as.dendrogram(dh_diana_1)
sub_grp_diana_1 <- cutree(dh_diana_1, k=4)
fviz_cluster(list(data = Set1_matrix_adjusted, cluster = sub_grp_diana_1), axes = c
    ↪ (3,1))
cluster.stats(dd1, sub_grp_diana_1, alt.clustering = NULL, silhouette = TRUE)
```

9.3.2 Dataset 2

```
Set2_matrix <- data.matrix(Set2)
Set2_matrix_adjusted <- Set2_matrix[,-1]

res2<-NbClust(Set2_matrix_adjusted, diss=NULL, distance = "euclidean", min.nc=2, max.
    ↪ nc=15,
            method = "centroid", index = "all")
fviz_nbclust(res2, method = "silhouette", diss = NULL, k.max = 10, nboot = 100,
           verbose = interactive(), barfill = "steelblue", barcolor = "steelblue",
           linecolor = "steelblue", print.summary = TRUE)
```

```
set.seed(76964057)

#Kmeans
res2_kmeans <- kmeans(Set2_matrix_adjusted, centers = 2, nstart = 50, algorithm = "
    ↪ Hartigan-Wong")
fviz_cluster(res2_kmeans, data = Set2_matrix_adjusted)

#PAM
res2_pam <- pam(Set2_matrix_adjusted, k = 2, diss = FALSE, metric = "euclidean", stand
    ↪ = FALSE, do.swap = TRUE)
fviz_cluster(res2_pam, data = Set2_matrix_adjusted)

#CLARA
res2_clara <- clara(Set2_matrix_adjusted, k = 2, metric = "euclidean", stand = FALSE,
    ↪ sample = 500, sampsize= nrow(Set2_matrix_adjusted), pamLike = FALSE)
fviz_cluster(res2_clara, data = Set2_matrix_adjusted)

#FCM
res2_fcm <- fanny(Set2_matrix_adjusted, 2, metric = "SqEuclidean")
fviz_cluster(res2_fcm, data = Set2_matrix_adjusted)

#Silhouette Analysis Kmeans
sil2_kmeans <- silhouette(res2_kmeans$cluster, dist(Set2_matrix_adjusted))
fviz_silhouette(sil2_kmeans, label = FALSE, print.summary = FALSE)

#Silhouette Analysis PAM
sil2_pam <- silhouette(res2_pam$cluster, dist(Set2_matrix_adjusted))
fviz_silhouette(sil2_pam, label = FALSE, print.summary = FALSE)

#Silhouette Analysis CLARA
sil2_clara <- silhouette(res2_clara$cluster, dist(Set2_matrix_adjusted))
fviz_silhouette(sil2_clara, label = FALSE, print.summary = FALSE)

#Silhouette Analysis FCM
sil2_fcm <- silhouette(res2_fcm$cluster, dist(Set2_matrix_adjusted))
fviz_silhouette(sil2_fcm, label = FALSE, print.summary = FALSE)

#Cluster Analysis Kmeans
dd2 <- dist (Set2_matrix_adjusted, method = "euclidean")
cluster.stats(dd2, res2_kmeans$cluster, alt.clustering = NULL, silhouette = TRUE)
```

```
#Cluster Analysis PAM
cluster.stats(dd2, res2_pam$cluster, alt.clustering = NULL, silhouette = TRUE)

#Cluster Analysis CLARA
cluster.stats(dd2, res2_clara$cluster, alt.clustering = NULL, silhouette = TRUE)

#Cluster Analysis FCM
cluster.stats(dd2, res2_fcm$cluster, alt.clustering = NULL, silhouette = TRUE)

#Agglomerative Hierarchical Algorithms

#Singly-Linked
ah_single_2 <- agnes(Set2_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, method = "single")
pltree(ah_single_2, cex = 0.6, hang = -1, main = "Dendrogram of Single Linkage Method:
    ↪  Dataset 2")
rect.hclust(ah_single_2, k =2, border = "red")
dend_sl_2 <- as.dendrogram(ah_single_2)
sub_grp_sl_2 <- cutree(ah_single_2, k=2)
fviz_cluster(list(data = Set2_matrix_adjusted, cluster = sub_grp_sl_2),axes = c(6,6))
cluster.stats(dd2, sub_grp_sl_2, alt.clustering = NULL, silhouette = TRUE)

#Complete-Link
ah_complete_2 <- agnes(Set2_matrix_adjusted, diss = FALSE, metric ="euclidean", stand
    ↪ = FALSE, method = "complete")
pltree(ah_complete_2, cex = 0.6, hang=-1, main = "Dendrogram of Complete Linkage
    ↪ Method: Dataset 2")
rect.hclust(ah_complete_2, k =2, border = "red")
dend_cl_2 <- as.dendrogram(ah_complete_2)
sub_grp_cl_2 <- cutree(ah_complete_2, k=2)
fviz_cluster(list(data = Set2_matrix_adjusted, cluster = sub_grp_cl_2))
cluster.stats(dd2, sub_grp_cl_2, alt.clustering = NULL, silhouette = TRUE)

#UPGMA
ah_upgma_2 <- agnes(Set2_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, method = "average")
pltree(ah_upgma_2, cex = 0.6, hang = -1, main = "Dendrogram of UPGMA Method: Dataset
    ↪ 2")
rect.hclust(ah_upgma_2, k =2, border = "red")
```

```
dend_upgma_2 <- as.dendrogram(ah_upgma_2)
sub_grp_upgma_2 <- cutree(ah_upgma_2, k=2)
fviz_cluster(list(data = Set2_matrix_adjusted, cluster = sub_grp_upgma_2))
cluster.stats(dd2, sub_grp_upgma_2, alt.clustering = NULL, silhouette = TRUE)

#Ward's
ah_wards_2 <- agnes(Set2_matrix_adjusted, diss = FALSE, metric = "euclidean", stand =
    ↪ FALSE, method = "ward")
pltree(ah_wards_2, cex = 0.6, hang = -1, main = "Dendrogram of Ward's Method: Dataset
    ↪ 2")
rect.hclust(ah_wards_2, k =2, border = "red")
dend_wards_2 <- as.dendrogram(ah_wards_2)
sub_grp_wards_2 <- cutree(ah_wards_2, k =2)
fviz_cluster(list(data = Set2_matrix_adjusted, cluster = sub_grp_wards_2))
cluster.stats(dd2, sub_grp_wards_2, alt.clustering = NULL, silhouette = TRUE)

#Divisive Hierarchical Algorithm

#DIANA
dh_diana_2 <- diana(Set2_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, stop.at.k = FALSE)
pltree(dh_diana_2, cex = 0.6, hang = -1, main = "Dendrogram of DIANA: Dataset 2")
rect.hclust(dh_diana_2, k =2, border = "red")
dend_diana_2 <- as.dendrogram(dh_diana_2)
sub_grp_diana_2 <- cutree(dh_diana_2, k=2)
fviz_cluster(list(data = Set2_matrix_adjusted, cluster = sub_grp_diana_2))
cluster.stats(dd2, sub_grp_diana_2, alt.clustering = NULL, silhouette = TRUE)
```

9.4 Dataset 3

```
Set3_matrix <- data.matrix(Set3)
Set3_matrix_adjusted <- Set3_matrix[,-1]

res3<-NbClust(Set3_matrix_adjusted, diss= NULL, distance = "euclidean", min.nc=2, max.
    ↪ nc=10,
            method = "complete", index = "all")
fviz_nbclust(res3, method = "silhouette", diss = NULL, k.max = 10, nboot = 100,
            verbose = interactive(), barfill = "steelblue", barcolor = "steelblue",
            linecolor = "steelblue", print.summary = TRUE)
```

```
set.seed(76964057)

#Kmeans
res3_kmeans <- kmeans(Set3_matrix_adjusted, centers = 3, nstart = 7, algorithm = "
    ↪ Hartigan-Wong")
fviz_cluster(res3_kmeans, data = Set3_matrix_adjusted)

#PAM
res3_pam <- pam(Set3_matrix_adjusted, k = 3, diss = FALSE, metric = "euclidean", stand
    ↪ = FALSE, do.swap = TRUE,pamonce =0)
fviz_cluster(res3_pam, data = Set3_matrix_adjusted)

#CLARA
res3_clara <- clara(Set3_matrix_adjusted, k=3, metric ="euclidean", stand = FALSE,
    ↪ pamLike = FALSE)
fviz_cluster(res3_clara, data = Set3_matrix_adjusted)

#FCM
res3_fcm <- fanny(Set3_matrix_adjusted, 3, metric = "SqEuclidean")
fviz_cluster(res3_fcm, data = Set3_matrix_adjusted)

#Silhouette Analysis Kmeans
sil3_kmeans <- silhouette(res3_kmeans$cluster, dist(Set3_matrix_adjusted))
fviz_silhouette(sil3_kmeans, label = FALSE, print.summary = FALSE)

#Silhouette Analysis PAM
sil3_pam <- silhouette(res3_pam$cluster, dist(Set3_matrix_adjusted))
fviz_silhouette(sil3_pam, label = FALSE, print.summary = FALSE)

#Silhouette Analysis CLARA
sil3_clara <- silhouette(res3_clara$cluster, dist(Set3_matrix_adjusted))
fviz_silhouette(sil3_clara, label = FALSE, print.summary = FALSE)

#Silhouette Analysis FCM
sil3_fcm <- silhouette(res3_fcm$cluster, dist(Set3_matrix_adjusted))
fviz_silhouette(sil3_fcm, label = FALSE, print.summary = FALSE)

#Cluster Analysis Kmeans
dd3 <- dist (Set3_matrix_adjusted, method = "euclidean")
cluster.stats(dd3, res3_kmeans$cluster, alt.clustering = NULL, silhouette = TRUE)
```

```
#Cluster Analysis PAM
cluster.stats(dd3, res3_pam$cluster, alt.clustering = NULL, silhouette = TRUE)

#Cluster Analysis CLARA
cluster.stats(dd3, res3_clara$cluster, alt.clustering = NULL, silhouette = TRUE)

#Cluster Analysis FCM
cluster.stats(dd3, res3_fcm$cluster, alt.clustering = NULL, silhouette = TRUE)

#Agglomerative Hierarchical Algorithms

#Singly-Linked
ah_single_3 <- agnes(Set3_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, method = "single")
pltree(ah_single_3, cex = 0.6, hang = -1, main = "Dendrogram of Single Linkage Method:
    ↪  Dataset 3")
rect.hclust(ah_single_3, k =3, border = "red")
dend_sl_3 <- as.dendrogram(ah_single_3)
sub_grp_sl_3 <- cutree(ah_single_3, k=3)
fviz_cluster(list(data = Set3_matrix_adjusted, cluster = sub_grp_sl_3))
cluster.stats(dd3, sub_grp_sl_3, alt.clustering = NULL, silhouette = TRUE)

#Complete-Link
ah_complete_3 <- agnes(Set3_matrix_adjusted, diss = FALSE, metric ="euclidean", stand
    ↪ = FALSE, method = "complete")
pltree(ah_complete_3, cex = 0.6, hang=-1, main = "Dendrogram of Complete Linkage
    ↪ Method: Dataset 3")
rect.hclust(ah_complete_3, k =3, border = "red")
dend_cl_3 <- as.dendrogram(ah_complete_3)
sub_grp_cl_3 <- cutree(ah_complete_3, k=3)
fviz_cluster(list(data = Set3_matrix_adjusted, cluster = sub_grp_cl_3))
cluster.stats(dd3, sub_grp_cl_3, alt.clustering = NULL, silhouette = TRUE)

#UPGMA
ah_upgma_3 <- agnes(Set3_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, method = "average")
pltree(ah_upgma_3, cex = 0.6, hang = -1, main = "Dendrogram of UPGMA Method: Dataset
    ↪ 3")
rect.hclust(ah_upgma_3, k =3, border = "red")
```

```
dend_upgma_3 <- as.dendrogram(ah_upgma_3)
sub_grp_upgma_3 <- cutree(ah_upgma_3, k=3)
fviz_cluster(list(data = Set3_matrix_adjusted, cluster = sub_grp_upgma_3))
cluster.stats(dd3, sub_grp_upgma_3, alt.clustering = NULL, silhouette = TRUE)

#Ward's
ah_wards_3 <- agnes(Set3_matrix_adjusted, diss = FALSE, metric = "euclidean", stand =
    ↪ FALSE, method = "ward")
pltree(ah_wards_3, cex = 0.6, hang = -1, main = "Dendrogram of Ward's Method: Dataset
    ↪ 3")
rect.hclust(ah_wards_3, k =3, border = "red")
dend_wards_3 <- as.dendrogram(ah_wards_3)
sub_grp_wards_3 <- cutree(ah_wards_3, k =3)
fviz_cluster(list(data = Set3_matrix_adjusted, cluster = sub_grp_wards_3))
cluster.stats(dd3, sub_grp_wards_3, alt.clustering = NULL, silhouette = TRUE)

#Divisive Hierarchical Algorithm

#DIANA
dh_diana_3 <- diana(Set3_matrix_adjusted, diss = FALSE, metric ="euclidean", stand =
    ↪ FALSE, stop.at.k = FALSE)
pltree(dh_diana_3, cex = 0.6, hang = -1, main = "Dendrogram of DIANA: Dataset 3")
rect.hclust(dh_diana_3, k =3, border = "red")
dend_diana_3 <- as.dendrogram(dh_diana_3)
sub_grp_diana_3 <- cutree(dh_diana_3, k=3)
fviz_cluster(list(data = Set3_matrix_adjusted, cluster = sub_grp_diana_3))
cluster.stats(dd3, sub_grp_diana_3, alt.clustering = NULL, silhouette = TRUE)
```