Data set analysis for Berlin Crime 2012 – 2019

Mert Efe

Middle East Technical University, Turkey, e259030@metu.edu.tr

Abdullah Burkan Bereketoglu

Middle East Technical University, Turkey, burkan.bereketoglu@metu.edu.tr

In the lecture *CEng 574 Statistical Data Analysis* by Volkan Atalay, a professor at the Middle East Technical University we learned how to make an analysis on a data set regarding several projection and clustering methods. In the assignments we had to apply the theoretical knowledge from the lecture to practice, using R as the programming language and a self-selected data set. In the following paper we are doing a data set analysis for the *Crimes in Berlin, Germany, 2012-2019* data set from Kaggle and discuss our results.

Additional Keywords and Phrases: projection methods, clustering methods, data set analysis, Berlin, crimes

ACM Reference Format:
Mert Efe, Abdullah Burkan Bereketoglu, 2022. Data set analysis for Berlin Crime 2012 – 2019.

1 INTRODUCTION

Berlin is the capital city of Germany. It has about 3.6 million inhabitants and consists of twelve different districts. It is a very special city with its own special history. For a period in history, Berlin was divided in two, and a wall was built. Nowadays, Berlin is a multicultural city with many interesting places. Many people came to Berlin to see historical artifacts such as The Berlin Wall, The Kaiser Wilhelm Memorial Church or visit cultural places such as one of Berlin's many museums. Additions to that, Berlin has lively nightlife.

However, how many beautiful places Berlin has, there also exists crime. As in many other cities there are also *Theft*, *Injury*, *Robbery* and so on.

As mentioned earlier, Berlin consists of twelve different districts. These twelves districts are *Mitte*, *Reinickendorf*, *Pankow*, *Lichtenberg*, *Marzahn-Hellersdorf*, *Treptow-Köpenick*, *Neukölln*, *Friedrichshein-Kreuzberg*, *Tempelhof-Schöneberg*, *Steglitz-Zehlendorf*, *Charlottenburg-Wilmersdorf* and *Spandau*.

In this paper, we analyze the *"Crimes in Berlin, Germany, 2012-2019"* data set we found at Kaggle. For our analysis, we want to use projection methods such as Principal Component Analysis, Multidimensional Scaling algorithms, and t-distributed stochastic neighbor embedding, called t-SNE. Moreover, we want to use three different clustering approaches. We are applying Hierarchical Clustering, K-Means, and K-Medoids to cluster the data and evaluate them with clustering evaluation criterions called Rousseeuw's Silhouette, Sum-Of-Squares, and Davies-Bouldin index.

The paper is structured as follows: In Section 2, we are describing which projection and clustering methods we are apply to the data set and how to evaluate them. In Section 3 the results of these methods are shown. Last, we are discussing the results and give an outlook on what other analysis approach could be done in Section 4.

1.1 Data set

We will discuss here how our data set is built and where we found it. Also, we look to some statistics and get knowledge about the correlation between the crimes. Afterwards, we will talk about our further analysis and how we compressed the data set for our analysis.

1.1.1 Overview

We found our data set from Kaggle. It is called *"Crimes in Berlin, Germany, 2012-2019"* and describes the occurrences of 16 different kind of crimes for each district region in a district from 2012 to 2019. The data set consists of 1200 samples and 20 features, where four features are categorical, and the remaining 16 features are numerical. A section of the data set is given in Table 1.

Looking to Figure 1 where the pie chart for the total number of crimes for 2012 and 2019 is visualized, we can observe 14.389 fewer crimes in 2019 than in 2012. Also, we can observe that the order of crimes changed from 2012 to 2019. For example, *Damage* was the third often crime in 2012 and *Injury* the fourth. In 2019 they swapped the places.

Table 1: Section of the data set for first sample with first 8 features

Year	District	Code	Location	Robbery	Street_robbery	Injury	Agg_assault
2012	Mitte	10111	Tiergarten Süd	70	46	586	194

1.1.2 Correlation Matrix

To know how the crimes are correlated to each other, we can use the correlation matrix. It is a symmetric matrix, where the crimes are listed on the x-axis and on the y-axis [1]. The correlation coefficient can become a value between -1 and 1 and describes how the crimes are correlated [1]. A value close to 1 indicates that these features are correlating to each other [1]. If one feature increases or decreases, the other follows the same pattern [1]. A value close to -1 implies the opposite [1]. It is a negative correlation [1]. Here would feature y decrease when feature x increases or vice versa [1]. A correlation coefficient close to 0 means no correlation [1]. So, the features are not influencing each other [1].

We can look the corresponding correlation matrix for our data set in Figure 1. As we can observe, nearly no correlation for *Drugs* with *Car* and a high correlation for *Robbery* and *Street_robbery*. Our data set has nowhere a negative correlation. Most crimes have a correlation coefficient higher than 0.5.

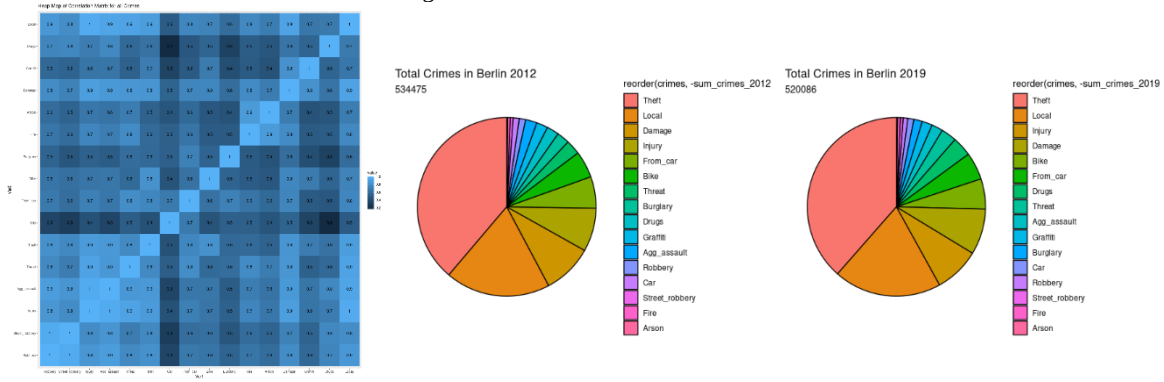

Figure 1: On the left side the correlation matrix for crimes is visualized. Darker patterns indicate not strong correlation, while bright tiles are indicating strong correlation. On the middle the pie charts demonstrate the total amount of crimes for 2012 ordered to the occurrence for each crime. On the right the same pie chart is visualized for 2019.

1.1.3 Further Analysis

We decided to drop the column *Code* and *Location* and summarize all crimes for each district for each year for our further analysis to keep track of districts instead of their corresponding regions. So, our new data set consists of 96 samples

with 18 features, where two features are categorical, and the remaining 16 features are numerical. A section of the new data set is given in Table 2.

Table 2: Section of the summarized data set for first sample with first 6 features

Year	District	Robbery	Street_robbery	Injury	Agg_assault
2012	Mitte	1253	610	7921	2362

2 DATA ANALYSIS

This Section is about the projection and clustering method we are going to use for our data set analysis. Also, we are talking about how to validate and evaluate the results.

2.1 Projection

In lectures we learned that there are different projection and mapping methods to reduce dimensionality. References [2], [3], [4] are ordered for following subjects excerpted from the lectures accordingly. In the data set three different types of projection and mapping methods are applied. Principal Component Analysis [2], Multidimensional scaling (metric & non-metric) [3], and t-distributed stochastic neighbor embedding [4], our goal was to map or make projection of the high dimensional data into a visualizable format in different senses.

2.1.1 Principal Component Analysis

In our analysis after looking at the correlations between the features and analyzing the preliminary data table we can now cover more grounds with techniques that will help us to plot our full data in 2-D, 3-D plots. Here to achieve the dimensionality reduction that is desired, we can apply a technique called PCA (Principal Component Analysis).

Mainly what PCA does is to pick the most variability in the coordinate system of n-D, then take the orthogonal (perpendicular) line as PC2 or the second principal component which is in theory the second most variance, proportional to the whole data set occurs [5]. With the PCA applied, we now have the data set have switched its coordinate frame to the one that has PC1 with most variance, then PC2 with second-most, if exist PC3 third most, PC4, PC5 and it goes on in that pattern, so it can be said that the frame is now selected such that it will cover most variance by its arms [5]. On the other hand, PCA has some limitations, so is not the best to use in each case, these limitations are the assumption of linearity, distribution being Gaussian, hence mean and variance is only necessary statistics [6].

- The first part is to the computation of the covariance matrix for the PCA which has several steps that will be discussed after the intensive steps and other time complexity related parts. $O(N * D * \min(N, D))$ [6]
- Second step is to compute the eigenvalue decomposition (EVD) of the covariance matrix which is computationally intensive and has a worst-case cost as $O(D^3)$ [6].
- Hence, resulting overall complexity for PCA is $O(N * D * \min(N, D) + D^3)$ [6]

Let's also discuss what algorithm do we use to make the computation of the EVD of the covariance matrix and how to summarize it in a few steps in detail. Given a matrix that NxD since we used N and D at every place, we should continue in that manner to preserve the similarity that is needed in the complexity function., in this matrix we have a target dimension that is k, then the algorithm of PCA makes Nxk matrix but not NxD matrix such that the columns of this matrix are the principal components of the matrix of NxD.

To calculate the covariance matrix from this point is first:

- Compute the mean-centered matrix of input matrix [5]
- Then compute the covariance matrix of the input matrix [5]
- As the third step we will have the calculation of Eigenvalue decomposition of covariance matrix.[5].
- Last step is delivery of principal components of main matrix, which are the associated eigenvectors of the matrix. [5]

For R, PCA can be calculated with the prcomp() method, which is provided form the stats package [7].

2.1.2 Multidimensional Scaling

Similar to PCA in its results, but different on the application part, Multidimensional scaling (MDS) is a technique that makes a map that shows the relative positions (distance metric) of a number of objects, given only a table of distances between them. The map can consist of one, two, or even more dimensions. MDS calculates either non-metric or metric solutions, also there are non-linear and linear approaches, but these are not calculation metrics rather techniques to be used. The distance table in MDS is known as the proximity matrix [8]. The proximity matrix arises either directly from the experiment conducted or may arise indirectly as a correlation matrix [8]. Proximity measures that are used in the MDS mapping quantify how "close" every two objects are. In MDS there are mostly three forms, dissimilarities, similarities, and correlations [9]. Definition of the similarities and dissimilarities can be given as, Similarities indicating a symmetrical matrix that is going to be converted to dissimilarities and represents the closeness of points in the data set [9]. Dissimilarities, on the other hand, are distances between two objects inside the data [9]. It can be directly measured, approximated, or calculated from the differences in a scale such as 1 is highest pain 10 is least., also symmetrical [9].

The two general methods that are described above as the calculation solutions are called CMS or Classical (Metric) Multidimensional Scaling, and Non-Metric Multidimensional Scaling (Sammon's, etc.). CMS reproduces the original metric or distances, such as Euclidean distances, Manhattan distances, Minkowski distances, or other related distance metrics. Non-Metric Multidimensional Scaling (Sammon's), assumes that only the ranks of the distances that wanted to be reproduced are known. Therefore, these types of non-metric MDS produce a map that tries to reproduce these ranks that are known. Furthermore, distances themselves for non-metric types are not reproduced. After the MDS is applied from the function *sammon()* of package *MASS* or *cmdscale()* of *stats* package, there is a value part of the MDS which is named stress, and that indicates goodness-of-fit, if it is really close to 0 for many data sets means the fit is perfect [10, 11]. If not close to 0 that means fit of the mapping is not good. Also, it is good to add that in current years, some scientists say that stress dependency is mostly related to distance matrix quality and object counts in the matrix to be acceptable rather than a score [9].

To apply MDS, one at first needs to decide whether they want their data mapped to two, three dimensions, if not MDS is mostly useless [9]. To determine the optimal number of dimensions, factor analysis can be done, and number of factors can be used [9].

2.1.2.1 Principial Coordinate Analysis

Algorithm that Torgerson (1952), which is mentioned in the course, is used for Metric MDS, also called Principal Coordinate Analysis. The algorithm is given as: There exists a distance matrix that will map the inter-point configurations of points X in low dimensional space. Most of the time, low dimension refers to one, two and three dimensions. Feasible to use with true distances [8]. Moreover, complexity of the metric MDS algorithm is $O(n^3)$ [9].

1. First, we need to calculate the distance matrix for our data set.

2. Then we are applying eigenvalue decomposition to it. The eigenvectors are stored in a matrix, descending ordered to its eigenvalues.
3. Now we take the first k eigenvectors we want to keep for our analysis.

2.1.3 Non-Metric Multidimensional Scaling

For the Non-Metric MDS we rather use a simplistic approach of an algorithm to make the mapping to lower dimensions. Here are the algorithm and the complexities. Feasible to use with no true distance data [8]. Computational cost higher than Metric MDS [8].

- Dissimilarity matrix is used rather than true distance matrix due to only information is on the rank order of dissimilarities.
- By that and usage of steepest descent with pre-applied metric Multidimensional technique to configure, minimization of stress is achieved in this algorithm. Iteration due to the steepest descent is really important, hence changes the time complexity to $O(n^2 * p)$, in which p is iteration total count, and space complexity is $O(n^2)$, these values are mostly for the *sammon()* of *MASS* package that is used in our research [10].

2.1.4 t-distributed stochastic neighbor embedding

t-Distributed Stochastic Neighbor Embedding is as the PCA and MDS is an unsupervised technique, but in addition to that it is also a non-linear technique that is primarily used for data exploration and as PCA and MDS to visualize high-dimensional data, and t-SNE does it by reducing the dimensionality [12]. t-SNE preserves only small pairwise distances or local similarities which is tuned by t-SNE hyperparameters, to give more emphasis on the t-SNE, many data scientists use the Swiss Roll data set as an example to make one understand why linear dimensionality reduction is not enough for such data sets [12]. In the core, t-SNE wants to accomplish to give the user to have a result that will give opinions about how is data arranged in a high dimensional space [13]. Also, it is relatively newer technique compared to PCA [13]. On the Swiss Roll Data set t-SNE correctly moves in the data set to find the accurate distance, but linear models directly match two points [14]. So basically if there is linearly nonseparable data that needs dimensionality reduction, we will use t-SNE. t-SNE is also really useful in CNN networks [15]. The package we used is *Rtsne* and the method is also called *Rtsne()*, with three important hyperparameters; *perplexity*, *max_iter*, *initial_dim* [16]. Lastly, it should be added that t-SNE has time and space complexity of $O(n^2)$, where n is the number of data points that constrains the application of the technique [13]. t-SNE's iterative nature makes direct implementation rather slow, and due to its complexity, t-SNE with big data will require too much time and cost will be too high [13]. So, we also need to tune, not only perplexity, but this iterative nature of the t-SNE [13]. It can be said that one needs to look at several iterations and decide on the first one which shows significant clusters after the application, with that in hand computation cost can be reduced to some extent [13]. One also needs to by an Autoencoder or PCA, reduce the initial dimensions, due to the probabilistic nature and the complexity of the t-SNE [13].

The main deal for t-SNE is to make mapping near on a manifold that is close in low dimensional space such as in two dimensions or three dimensions. To be able to achieve this as t-SNE algorithm:

1. We start with measuring the Euclidean distances in the high dimensions & convert these distances into probabilities of picking each point as a neighbor, in this case our similarities are proportional to probability results, at this point we will use Gaussian distribution probability and density for each point.

2. Continue with again Euclidean distances measurement and conversion of probability, hence same as the first step for low dimensionality, however we used t-distribution with heavier tails.
3. Then continue with the minimization of the difference of the conditional probabilities by such as KL-divergence.

2.2 Clustering

As we learned in the lecture, there are different clustering approaches to cluster the data. All following subsections are according to references [17], [18], [19].

We applied three different types of clustering for our data set. First, we will talk about Hierarchical Clustering [17], then we will go forward with K-Means [18], and last, we will discuss K-Medoids [19]. The goal of clustering is to group data points with similar patterns.

2.2.1 Hierarchical Clustering

First, there exist two kinds of Hierarchical Clustering: Agglomerative and Divisive. They are different in that Agglomerative Hierarchical Clustering starts with each point as an individual cluster and merges the two nearest points depending on the linkage condition. Divisive Hierarchical Clustering starts with all points as one cluster and splits each, depending on a linkage condition. For our analysis, we decided to use Agglomerative Hierarchical Clustering. Therefore, we used the *agnes()*-function in R from the *cluster* package [20]. The algorithm is relatively easy.

1. We need to calculate the similarity or dissimilarity matrix for our given data set.
2. Then we need to calculate the distances to the other points for each point with a metric like Euclidean distance.
3. We are merging the two closest points depending on a linkage condition and updating the similarity, or the dissimilarity matrix, depending on which one we used.
4. We then repeated Steps 2 and 3, until we formed a single cluster containing all points, which would be the root.

For Hierarchical Clustering, there exists not only different methods, but also different types of linkages on how the clusters are merged. We consider the following linkage types: complete, single, and average for our analysis. Complete-Linkage calculates the maximum distances of two points for two different clusters. Single-Linkage is the opposite and calculates the minimum distance. In average linkage, the distances for all points for a cluster are calculated to another cluster and divided by the number of points for both clusters.

The space complexity for Hierarchical Clustering is $O(n^2)$, because the similarity/dissimilarity matrix for n points is stored. Since we need to calculate the distance for each point in the similarity/dissimilarity matrix and after merging, we need to update the matrix, time complexity is $O(n^3)$.

2.2.2 K-Means

K-Means is an iterative partitioning clustering algorithm. The algorithm consists of one hyperparameter called k, which defines how many clusters are formed. If k is chosen, it is not possible that the algorithm founds more or less clusters then k. The only thing that can happen is that an empty cluster can be formed if all data points are closer to another cluster. Also, K-Means tries to minimize an error criterion such as the sum-of-squares. So, similar samples are assigned to the same cluster while different samples are assigned to another one. However, it must be noted that the sum-of-squares is negatively correlated to the number of clusters.

In R, the function to perform K-Means is called *kmeans()* and is provided in the *stats* package [21]. The algorithm is easy to understand and works as follows:

1. k centroids are initialized randomly in the data set. The centroids can be a random sample or consist of random values. The centroid defines a cluster.
2. Calculate for each data point the distance to each centroid. Therefore, a distance metric such as the Euclidean distance can be used.
3. Assign each data point to their closest centroid.
4. Update the centroid by calculating the mean over all assigned data points.
5. Repeat steps 2 to 5 until no data point is changing the cluster, so until it converged or until a previously defined maximum number of iterations is reached.

As we can see in step 1, k centroids are initialized randomly. That is the reason why K-Means can produce different results for different runs. One result could be better than another one. Because of that, the R function *kmeans()* has a configuration where we can decide how many times the algorithm should be performed [22]. Over all runs, it takes the best one. We decided to have 25 runs for our data set because we read in [22] that this is often recommended.

In step 3, we learned that a data point would be assigned to its closest cluster. That is the reason why K-Means is sensitive to outliers. In step 4, the centroids are updated, and outlier forces to move a centroid to a worse place. Therefore, it is better to perform K-Means after the outliers are eliminated from the data set.

The space complexity of K-Means is $O((n + k) * d)$ [23]. We need to store n – data points with d – dimensions and k – clusters, which are defined by centroids. Again, the centroids consist of d – dimensions. The time complexity for K-Means is $O(n * k * i * d)$, because we calculate for i – Iterations the distance for a d – dimensional vector for n – data points and reassign them to k – clusters.

2.2.3 K-Medoids

K-Medoids is an alternative to K-Means. It works similarly. However, instead of using centroids we use representative objects, called medoids. The medoids are the medians of a cluster. That is the reason why K-Medoids is more robust against outliers. The algorithm consists of one hyperparameter called k, which defines the number of clusters. The same applies here, if k is chosen, there is no opportunity to get more or less clusters then k. K-Medoids tries to minimize the sum of dissimilarities for each data point for each cluster by regarding their nearest medoid.

In R the K-Medoids method, which we used, is provided in the *cluster* package [24]. The method is called *pam()*, abbreviated for *Partitioning Around Medoids* [24]. The advantage of *Partitioning Around Medoids* is that it works well for small data sets. Since we shrunk our 1200 sample huge data set to 96 samples, it is very suitable for us.

K-Medoids works as follows:

1. k medoids are selected randomly [25].
2. Calculate for each data point the distance to each medoid. Therefore, a distance metric such as the Euclidean distance can be used [25].
3. Assign each data point to their closest medoid [25].
4. Compute the swap cost [25].
5. For each cluster, swap the medoid with one non-medoid and repeat steps 2 to 4. If the swap decreases the sum of dissimilarities, keep this set of medoids and repeat step 5, otherwise stop [25].

The time complexity for K-Medoids is $O(i * k * (n - k)^2)$ [26] because we need to calculate for each non-medoid data point the distance to each medoid. I did not find something for the space complexity for K-Medoids, however, I think it is also $O((n + k) * d)$ such as for K-Means, because we need to store n – data points in total and need to remember which

one was the medoid, which would describe the k in the inner term. Since the data points are d – dimensional we need to store d – dimensions for each data point.

2.3 Cluster Evaluation and Validation

After clustering the data, we need to find out, which results provides the best results. To find this out, there are different types of evaluation criteria. All following subsections are according to references [27].

2.3.1 External Criteria

External criteria would be the evaluation comparing the results with our labels. Since our data set does not contain any labels, we cannot perform external criteria.

2.3.2 Internal Criteria

Internal criterion is the evaluation approach by disregarding labels and evaluating how well defined the clusters are. In the lecture, we learned different types of internal criteria. Silhouette, sum-of-squares error and cluster validity via correlation are just a few examples.

In Section K-Means we said that K-Means tries to minimize sum-of-squares. To use the sum-of-squares validation approach we can visualize total within sum-of-squares for n clusters. Afterwards, it can be evaluated with the Elbow-Method. The visualization of the total sum-of-squares looks such as an arm. The Elbow-Method determines the optimal k by evaluating the total sum-of-squares for n clusters, and choosing the cluster, where the bend lies.

Silhouette, also called Rousseeuw's Silhouette describes, how well data points are clustered regarding the intra- and inter-cluster distances. If the Silhouette coefficient is close to 1 it implies that the data points are well clustered. If it would be close to 0, it indicated that the data point should be better assigned to another cluster. In the last case, for Silhouette coefficient below 0 it indicates that the data point is misclassified. For using the silhouette approach to compare different numbers of clusters, we need to calculate the average Silhouette coefficient.

In R the method we used to perform Rousseeuw's Silhouette is called *index.S()* and is provided in the *clusterSim* package [28].

2.3.3 Relative Criteria

Relative criteria compare one clustering result with another, which is created with the same algorithm but with different parameters. Davies-Bouldin index is one representative for relative criteria, which we used for our data set.

The index describes how compact the clusters are and how far away each center is from another center. A Davies-Bouldin index close to 0 implies a good clustering.

To perform relative criteria in R we used the *index.DB()* method, which is a method from the *clusterSim* package [29], such as Rousseeuw's Silhouette.

3 RESULTS

In this Section we are going to discuss our results for the different kind of projection and clustering methods.

3.1.1 Principal Component Analysis

To be able to visualize the new data set with 96 observations and 18 features also to make analysis on the data set in 2-D or 3-D to observe patterns, similarities, dissimilarities, the first thing that should be done is to reduce the dimensionality

without losing too much of the variance itself. This can be done with various techniques which will be covered in this paper. Here with PCA, it can be said that by linearly reducing the dimensionality sometimes be faulty as described in the introduction of PCA.

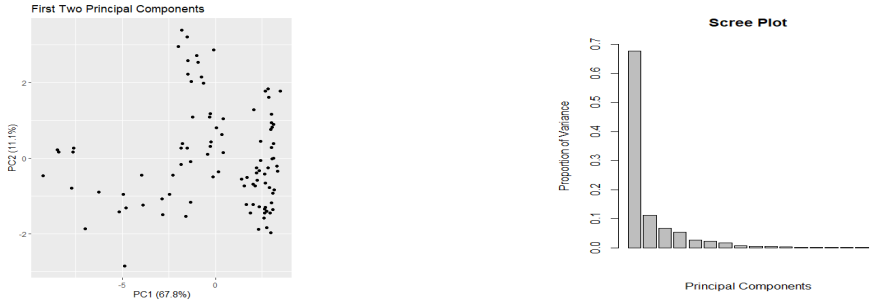

Figure 2: In the left we can observe how the data is spread corresponding to PC1 and PC2. The right Scree Plot shows the proportion of variance for each PC.

When looking at the results of the PCA, it is seen that most of the variance is on the PC1 which is the first eigenvalue of the eigenvectors of the input matrix as described in the algorithm part. PC1 with 67.8% variance, which can be seen from Figure 2 is spread widely, which is good for to reduce the dimensionality to lower dimensions, if less number of principal components at total pass the eighty to eighty-five percent total proportion of variance that means reduction to that specific minimum will be satisfied without losing much of the meaningful data and reducing noise at a small cost, also gaining the ability to visualize the data in a visualizable manner.

Since the total proportion of variance is ~ 80% for the PC1 and PC2, it can be deduced that the data set can be successful, without losing much information that can be reduced to two dimensions. One can also discuss that the data should sharply pass this threshold, hence it is possible to also think the data with three significant dimensions, therefore with PC1, PC2, and PC3 to make a more precise dimensionality reduction.

3.1.2 Multidimensional Scaling

Results of the MDS for both non-metric and metric at the end show a similar pattern, and it is also showing similar results to the PCA applied. In the analysis, two distance methods, and two general methods are used. As two-dimensional data had enough variability, that is looked for as ~80%, directly applied dimension as two. These results give information about the pairwise distances, since distances are preserved compared to PCA, and the distance matrix is also still available. In the end, dissimilarities, as expected, show a similar pattern as PCA.

If asked why, it is due to dissimilarities increase in a path with spread, and spread creates variance, variance is the core of the approach of PCA to reduce the dimensionality, hence if the high variance in a path, then there can be said that the dissimilarities are also varying widely, compared to the other dimensions.

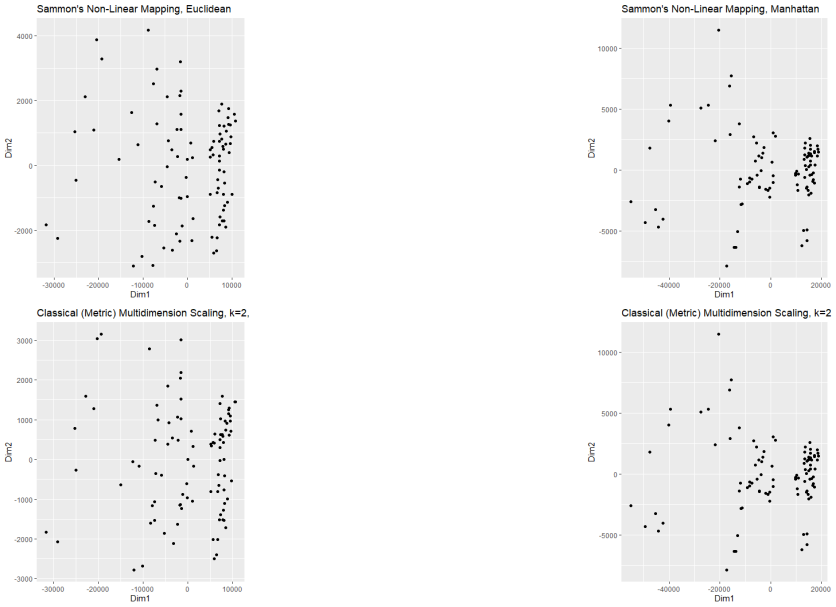

Figure 3: The scatter plots show different results for each MDS method for different distance metrics.

3.1.3 t-distributed stochastic neighbor embedding

Results as seen, indicates that we have three clusters which are similar if intra-cluster points at each cluster compared with the one that is also in the same cluster. Furthermore, results also show perplexity that needed to be tuned is 4 as optimal, initial PCs are 10, which also shows that linear dimensionality reduction can be applied to the data set, also by visually observing, it is deducted that optimal maximum iteration count is 500.

Moreover, the Shepard diagram shows us that the rank correlation of the t-SNE is not high since the plot is not linear, hence the similarities inside the clusters may be misleading.

By the results, one can conclude that if the results were more significant, at the goodness-of-fit test, total crimes or crime types might be similar inside the clusters, and also when looked from the data set visually one can actually see that these dates, some of them are indeed similar at total crimes and similar crime counts at various features. The issue is not all of them are similar in intra-cluster, and for one of the clusters none of them are relevant to each other, that one is the bottom left cluster, maybe the algorithm put it there to resemble its relevancy, but there cannot be any proof on that, since there is no such mathematical background for t-SNE.

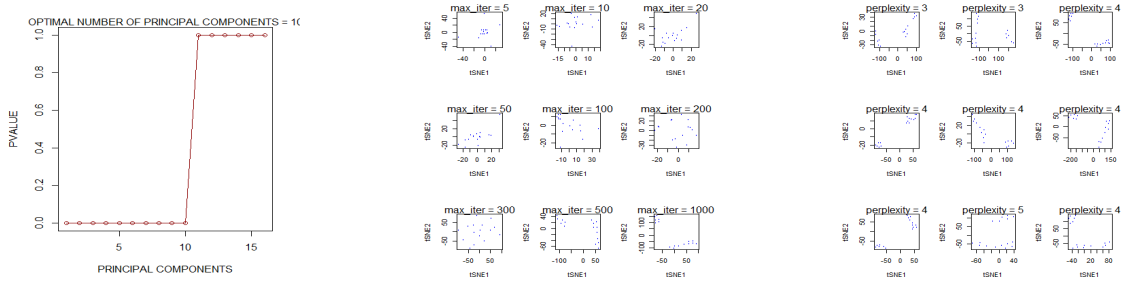

Figure 4: Hyperparameter tuning for t-SNE

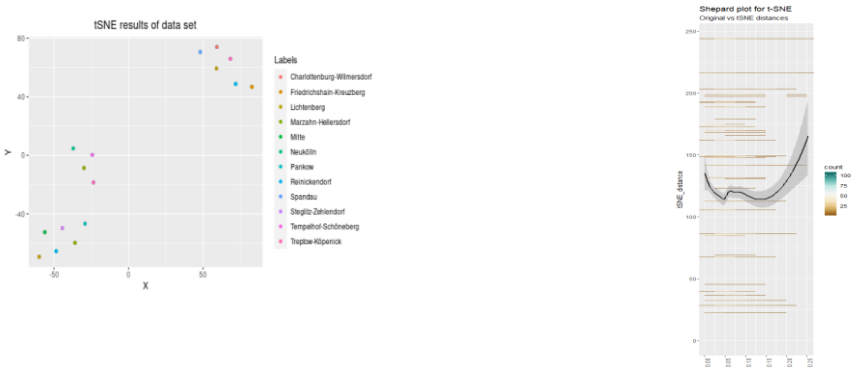

Figure 5: Left we have the t-SNE cluster plot and right we have the Shepard Diagram

3.1.4 Hierarchical Clustering

To decide, which linkage type is the best, we need to find out which linkage-type offers us the best clustering results for over data set. The *agnes()* method also calculates an agglomerative coefficient, which value indicates, how well the cluster structures are [30]. The agglomerative coefficient needs to be close to 1 [30]. The best agglomerative coefficient was given for the complete linkage for our data set. So, we continue our further analysis for Hierarchical Clustering with the complete linkage as the condition in which clusters are merged depending on the calculated distances.

To visualize our results, we can create a Dendrogram, a tree-like diagram that illustrates when data points are formed to a cluster and merged with other ones. We can also show our clustering results in a two-dimensional projection of our data. Therefore, we need to perform Principal Component Analysis first because our data set consists of nine features, and we cannot visualize all of them. In Figure 6 are our result performing Hierarchical Clustering with complete linkage.

To find the optimal k we applied Rousseeuw's Silhouette and Davies-Bouldin index. Also, in Figure 6 we can observe that for Hierarchical Clustering with complete linkage, the optimal k for Rousseeuw's Silhouette is given for $k = 2$ and for the Davies-Bouldin index it is also $k = 2$.

Our results are visualized in Figure 6. Since we chose k as k=2 we only have two clusters. Regarding Figure 8 we can see, after adapting the Hierarchical Clustering results to it that *Mitte* and *Friedrichshain-Kreuzberg* are in the same cluster. The cluster contains each district for each year. The other cluster is formed by *Neukölln*, Pankow, *Tempelhof-Schöneberg*, *Charlottenburg-Wilmersdorf*, *Lichtenberg*, *Treptow-Köpenick*, *Steglitz-Zehlendorf*, *Marzahn-Hellersdorf*, *Spandau*, and *Reinickendorf*. Also, here, for each district each year lies in the cluster.

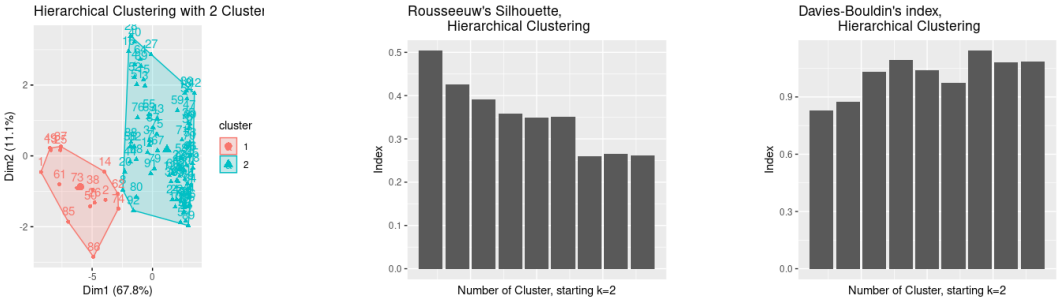

Figure 6: Left we have the visualization for Hierarchical Clustering with two clusters. On the middle, Rousseeuw's Silhouette for ten clusters are shown for the internal criteria evaluation and on the right side the Davies-Bouldin's index for ten clusters is given for the relative criteria evaluation.

Our results for K-Means are shown in Figure 7. To find the optimal k and to be consist to the evaluation criteria, we also used for K-Means Rousseeuw's Silhouette and Davies-Bouldin index to determine the optimal k. However, we also used the Elbow-Method to determine the optimal k regarding the total within sum-of-squares.

Rousseeuw's Silhouette demonstrates that the optimal k is $k = 2$. Same applies for Davies-Bouldin index. But, looking the evaluation with the Elbow-Method with respect to the total within sum-of-squares the optimal k would be $k = 4$.

The clustering result in Figure 7 is visualized as $k = 4$ as the optimal k. Using Principal Component Analysis, we can project our data in two dimensions and visualize them, colored by their assigned clusters. We can observe that *Mitte* and *Friedrichshain-Kreuzberg* are the smallest clusters containing themselves for each year. *Neukölln*, *Pankow*, *Tempelhof-Schöneberg*, and *Charlottenburg-Wilmersdorf* forms the middle cluster. Again, the cluster contains for each district each year. The biggest cluster is formed by the districts: *Lichtenberg*, *Treptow-Köpenick*, *Steglitz-Zehlendorf*, *Marzahn-Hellersdorf*, *Spandau*, and *Reinickendorf*.

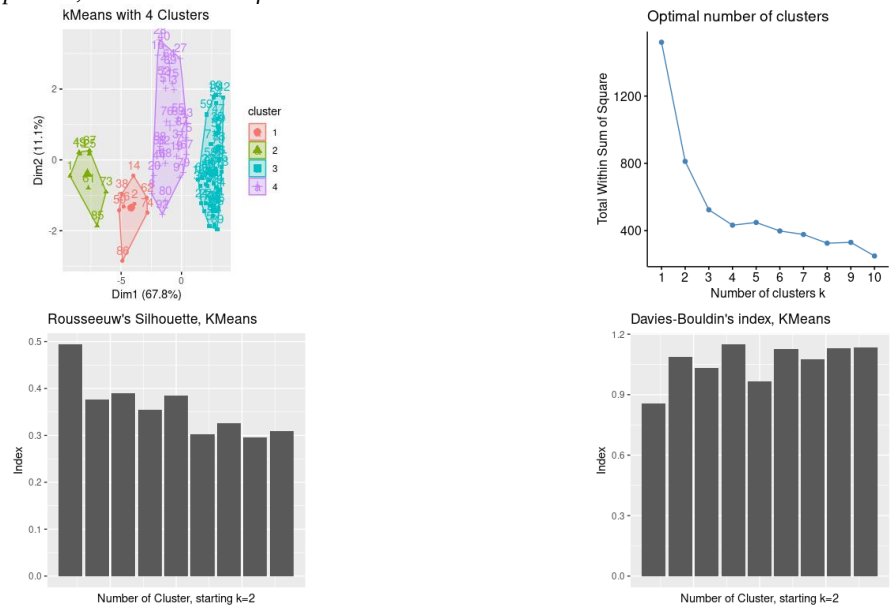

Figure 7: On the left top corner the clustering result for $k = 4$ is given. On the right top corner, the total within sum-of-squares is shown to apply the Elbow-Method. On the bottom left corner, we have the Rousseeuw's Silhouette given for ten clusters and on the bottom right corner the Davies-Bouldin's index.

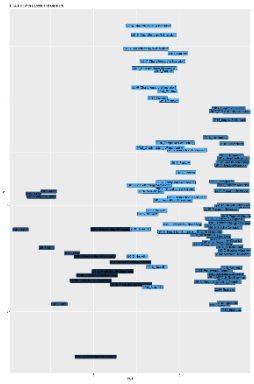

Figure 8: Clustering results for K-Means with $k = 4$ with labeled districts.

3.1.6 K-Medoids

To obtain the optimal value for k we can visualize the total sum-of-squares for n clusters and use the Elbow-Method to identify, where the bend is. In Figure 9 the total within sum-of-squares for 10 clusters are visualized. We can observe that the optimal k is k=4. Also, we can used Rousseeuw's Silhouette and Davies-Bouldin index to determine it, which are also shown in Figure 9. Rousseeuw's Silhouette indicates the same results as the Elbow-Method for the total within sum-of-squares. However, Davies-Bouldin index indicates $k = 2$ as the optimal one too.

If we visualize the data now in the 2D-projection and use $k = 4$ as the optimal k, we can observe that we got the exact same result as for K-Means, which is quite interesting.

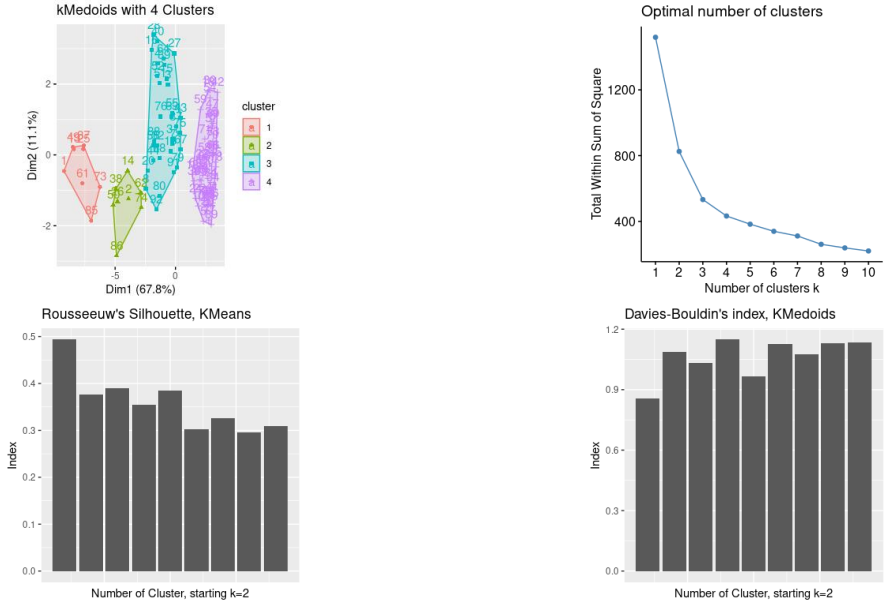

Figure 9: On the left top corner the clustering result for $k = 4$ is given. On the right top corner, the total within sum-of-squares is shown to apply the Elbow-Method. On the bottom left corner, we have the Rousseeuw's Silhouette given for ten clusters and on the bottom right corner the Davies-Bouldin's index.

3.1.7 Finding best cluster algorithm

R provides in the *clValid* package a method called *clValid()*, which evaluates the clustering results for the given clustering algorithms with three validation types [31]. For our data set, the best validation score is reached for Hierarchical Clustering with complete linkage for $k = 2$ for connectivity and silhouette and $k = 3$ for the Dunn index, another relative criterion.

4 CONCLUSION

As to conclude, we will discuss about our results compared to real life expectations and give a perspective for future work.

4.1.1 Real Life Expectations

Intuitively, we would cluster the districts differently. We were surprised how the districts are clustered. Of course, we are talking on a subjective perception. It is interesting to see the real results with some data. Our analysis shows that *Mitte* builds a cluster with only themselves. Regarding the data and calculating the total number of crimes, it came out that *Mitte* happens the most crimes, followed by *Friedrichshain-Kreuzberg*. That could be the reason why there are grouped together as the optimal k for Hierarchical Clustering and K-Means. Looking at the not-summarized data, we can obtain that the reason might be the district regions in *Mitte*. In *Mitte* is a district region called *Alexanderplatz*. It is a crowded place where many trains and bus stations exist. Also, there are some sights directly there or in the near, like the *Fernsehturm*. We think *Alexanderplatz* is almost comparable with *Kızılay*. While the other district regions in *Mitte* have the total amount of crimes round about 6.000-8.000, *Alexanderplatz* forms a huge outlier with roundabout 20.000.

4.1.2 Future work

Further analysis of the data can start with looking at the locations which we combined with their district relation to zoom-in the picture to understand more local behaviors. Later, if people flow rate can be delivered from the municipalities and other related local authorities there can be done a data stream analysis, from which we can analyze the real time locations crime rates. Now, we only know the crime amount rather than the rate, since we do not have the information about the population, and how many people pass the place each day. So, touristic places or transport hubs could be determined.

REFERENCES

[1] Zach, "How to read a correlation matrix," *Statology*, 15-Sep-2020. [Online]. Available: https://www.statology.org/how-to-read-a-correlation-matrix/. [Accessed: 22-Jan-2022].

[2] Volkan Atalay, Fall 2020, *"*DATA PROJECTION*", CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[3] Volkan Atalay, Fall 2020, *"DATA PROJECTION-Part 2 Multi Dimensional Scaling (MDS)", CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[4] Volkan Atalay, Fall 2020, *"Non-Linear Projection and Embedding Methods", CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[5] A. Banerjee, "Computational complexity of PCA," Medium, 25-Oct-2020. [Online]. Available: https://alekhyo.medium.com/computational-complexity-of-pca-4cb61143b7e5. [Accessed: 19-Jan-2022].

[6] H. Cardot and D. Degras, "Online principal component analysis in high dimension: Which algorithm to choose?" arXiv.org, 11–Nov-2015. [Online]. Available: https://arxiv.org/abs/1511.03688. [Accessed: 19-Jan-2022]

[7] "PRCOMP: Principal Components Analysis," RDocumentation, 11-Feb-2021. [Online]. Available: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp. [Accessed: 22-Jan-2022].

[8] "Chapter 435 Multidimensional Scaling," Multidimensional Scaling. [Online]. Available: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Multidimensional_Scaling.pdf. [Accessed: 20-Jan-2022].

[9] S. Jung, “Lecture 8: Multidimensional Scaling - University of Pittsburgh,” Advanced Applied Multivariate Analysis, 2013. [Online]. Available: https://www.stat.pitt.edu/sungkyu/course/2221Fall13/lec8_mds_combined.pdf. [Accessed: 20-Jan-2022].

[10] “Sammon: Sammon's non-linear mapping,” RDocumentation, 05-Apr-2021. [Online]. Available: https://www.rdocumentation.org/packages/MASS/versions/7.3-54/topics/sammon. [Accessed: 22-Jan-2022].

[11] “Cmdscale: Classical (metric) multidimensional scaling,” RDocumentation, 05-Apr-2021. [Online]. Available: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/cmdscale. [Accessed: 22-Jan-2022].

[12] D. Surendran, “Swiss Roll Dataset,” a Swiss Roll Dataset, 16-May-2004. [Online]. Available: http://people.cs.uchicago.edu/~dinoj/manifold/swissroll.html. [Accessed: 20-Jan-2022].

[13] “How T-SNE works¶,” How t-SNE works - openTSNE 0.3.13 documentation. [Online]. Available: https://opentsne.readthedocs.io/en/latest/tsne_algorithm.html. [Accessed: 20-Jan-2022].

[14] K. E. (burnpiro), “T-SNE clearly explained,” Medium, 22-Apr-2020. [Online]. Available: https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a. [Accessed: 20-Jan-2022].

[15] N. Pezzotti, B. P. F. Lelieveldt, L. van der Maaten, T. Höllẗö E. Eisemannö and A. Vilanova, “Approximated and user steerable tsne for progressive visual analytics,” arXiv.org, 16-Jun-2016. [Online]. Available: https://arxiv.org/abs/1512.01655. [Accessed: 20-Jan-2022].

[16] “Rtsne: Barnes-hut implementation of T-distributed stochastic neighbor embedding,” RDocumentation, 05-Apr-2021. [Online]. Available: https://www.rdocumentation.org/packages/Rtsne/versions/0.15/topics/Rtsne. [Accessed: 22-Jan-2022].

[17] Volkan Atalay, Fall 2020, *“HIERARCHICAL CLUSTERING”, CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[18] Volkan Atalay, Fall 2020, *“(Some) PARTITIONING ALGORITHMS”, CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[19] Volkan Atalay, Fall 2020, *“Clustering beyond k-means”, CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[20] “Agnes: Agglomerative nesting (hierarchical clustering),” *RDocumentation*. [Online]. Available: https://www.rdocumentation.org/packages/cluster/versions/2.1.2/topics/agnes. [Accessed: 22-Jan-2022].

[21] “Kmeans: K-means clustering,” *RDocumentation*. [Online]. Available: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans. [Accessed: 22-Jan-2022].

[22] “K-means cluster analysis,” *K-means Cluster Analysis · UC Business Analytics R Programming Guide*. [Online]. Available: https://uc-r.github.io/kmeans_clustering. [Accessed: 20-Jan-2022].

[23] Abdul Manan, “Computing complexity of kmeans algorithm,” Stack Overflow, 01-Aug-1962. [Online Forum]. Available: https://stackoverflow.com/questions/25362496/computing-complexity-of-kmeans-algorithm. [Accessed: 18-Jan-2022].

[24] “Pam: Partitioning around Medoids,” RDocumentation. [Online]. Available: https://www.rdocumentation.org/packages/cluster/versions/2.1.2/topics/pam. [Accessed: 22-Jan-2022].

[25] Kashi Sai Prasad, “K medoids - pam clustering,” YouTube, 27-Jun-2021. [Online]. Available: https://www.youtube.com/watch?v=NyueAYkWqwk. [Accessed: 20-Jan-2022].

[26] M. Tiwari and R. Singh, “[PDF] comparative investigation of k-means and K-medoid algorithm on Iris Data: Semantic scholar,” [PDF] Comparative Investigation of K-Means and K-Medoid Algorithm on Iris Data | Semantic Scholar, 01-Nov-2012. [Online]. Available: https://www.semanticscholar.org/paper/Comparative-Investigation-of-K-Means-and-K-Medoid-Tiwari-Singh/57713d3bbeba598ef6a452bec0a55045e1e82f27. [Accessed: 22-Jan-2022].

[27] Volkan Atalay, Fall 2020, *“Evaluation and Validation of Clusters”, CEng 574 Statistical Data Analysis.* [Lecture Slides]. [Accessed: 20-Jan-2022]

[28] “Index.S: Calculates Rousseeuw's silhouette internal cluster quality index,” *RDocumentation*. [Online]. Available: https://www.rdocumentation.org/packages/clusterSim/versions/0.49-2/topics/index.S. [Accessed: 22-Jan-2022].

[29] “Index.DB: Calculates Davies-Bouldin's index,” *RDocumentation*. [Online]. Available: https://www.rdocumentation.org/packages/clusterSim/versions/0.49-2/topics/index.DB. [Accessed: 22-Jan-2022].

[30] “Hierarchical Cluster Analysis,” Hierarchical Cluster Analysis · UC Business Analytics R Programming Guide. [Online]. Available: https://uc-r.github.io/hc_clustering. [Accessed: 20-Jan-2022].

[31] “CLVALID: Validate cluster results,” *RDocumentation*. [Online]. Available: https://www.rdocumentation.org/packages/clValid/versions/0.7/topics/clValid. [Accessed: 22-Jan-2022].