Evaluation of optimal clusterings found by cluster validation measures

by

Yu Shi

B.S., Arizona State University, 2017

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2020

Approved by:

Major Professor
Michael Higgins

Copyright

Abstract

There are many measures developed for assessing clustering algorithms. However, little work has been done to determine what type of clusterings these validation measures would consider "the best." In particular, if a clustering validation measure performs well, then it should be able to identify the "correct" clustering when when presented with all possible ways of clustering a dataset. We evaluate the performance of five clustering validation measures—Silhouette, Hubert-Gamma, R-squared, the Dunn family of indices, and the data Davies-Bouldin index—on five small clustered datasets. To obtain a large set of candidate clusterings, we view each dataset as a graph and form a connected bottleneck subgraph. On this subgraph, we identify all set-connected partitions—those whose blocks are connected—that satisfy a set of constraints on the number of blocks and the size of each block within the partition. We then apply the validation measure on each of the possible partitions to determine the clustering that each validation measure considers to be optimal. Based on test results, we find each measure has its own preferences. For example, the silhouette measure tends to be better at capturing connected regions, and many others measures prefer clusterings that contain many clusters. Finally, we compare the clusterings found by the validation measures to those obtained by other popular clustering methods including k-means, hierarchical agglomerative clustering (HAC), density-based spatial clustering of applications with noise (DBSCAN) and ordering points to identify the clustering structure (OPTICS).

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my major professor, Dr. Michael Higgins, for his help and guidance in my study and this report. Thank you for your patient to my questions. Every meeting can help me solve problems well and deeper understanding of statistics. I would like to thank my committee members, Dr. Haiyan Wang and Dr. Wu Cen for their suggestions and help of my report. I also want to thank my friends in Department of Statistics. We had a lot of fun and good times together. I would also like to thank my family for their support. Thank them for always supporting my decision and helping me.

Chapter 1

Introduction

Over the past few decades, there have been various methods proposed to perform the clustering of datasets (Hastie et al., 2001). As the number of clustering methods is on the rise, clustering is made applicable to more and more types of data. However, allowing for many competing clustering methods, it is potentially difficult to identify which method is most appropriate for a specific dataset. Consequently, clustering validation measures have been developed to compare these clustering methods for their effectiveness.

At present, there have been some work performed on assessing the efficacy of these validation measures. Some research focuses on the viability of validation measures (Brun et al., 2007). In this research, different types of models taken into consideration, clustering is performed using different methods, and validation measures are applied to compare them. Other research are aimed at validating the measures (Liu et al., 2010). Through the experiment on different types of datasets, clusterings are performed with different number of clusters for each set. Then, the measures are used to validate each clustering and determine whether the expected clustering can be identified. Some validation measures have been used to automate number of clusters in a clustering algorithm. For example, the `fviz_nbclust` function in the `factoextra` R package (Kassambara and Mundt, 2020) finds the optimal number of clusters for a given clustering method by applying the method many times to the dataset with varying number of clusters and finding which number of clusters leads to better

values of a selected validation measure.

In our study, we build on this literature by assessing the efficacy of five clustering validation measures: Silhouette, Hubert-Gamma, R-squared, the Dunn index, and the data Davies-Bouldin index. Our consideration is given to five small datasets for which some "reasonable" clusterings can be enumerated. Then, these clustering validation measures are applied across all potential clusterings to ascertain which one the measure judges as "optimal", thus allowing a comparison to be performed between the cluster validation measures.

1.1 Clustering

Clustering is a common statistical method that classifies and aggregates similar or close data points (based on their covariates) into different groups known as clusters. Clustering techniques are applied in such fields as machine learning (Taherkhani and Pierre, 2016), pattern recognition (Kalhori and Zarandi, 2015), and information retrieval (Jimenez and Vidal, 2004,2005).

There are a wide variety of clustering algorithms. In crisp clustering, each unit is assigned to a unique cluster. Fuzzy clustering requires that consideration is given to the correlation between the unit and the cluster. In fuzzy clustering, data points may fall into multiple clusters.

In our study, it is assumed that there are n units, as numbered 1 through n. Each unit i contains p covariates, denoted as $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$. In many clustering methods, the distance between units' covariates plays an essential role in the formation of clusters and the assessment of clustering. Among various methods applied for the calculation of distance are the Euclidean distance, the Manhattan distance and the Mahalanobis distance. The Euclidean distance is utilized in this study:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ip} - x_{jp})^2} = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}. \quad (1.1)$$

In the following part, various clustering methods will be discussed in detail.

K-means is known as one of the most commonly used clustering methods (Hartigan and Wong, 1979). It obtains a clustering that contains k clusters and all of the points in the same cluster are closest to their own cluster centroid. K-means aims at providing the clustering with minimal within-cluster variances. The objective function for k-means is expressed as follows

$$\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2. \tag{1.2}$$

Where the set of clusters $C = \{C_1, C_2, \ldots, C_k\}$, and c_i is the centroid of clusters. In k-means, the centroid is the mean of all vectors in the cluster. K-means clustering algorithms are subject to limitations and are frequent to identify locally optimal clusterings rather than globally optimal ones. In most cases, the number of clusters k should is supposed to defined in the first place.

For hierarchical agglomerative clustering (HAC) (Nielsen, 2016), distance is a critical factor. The purpose of HAC is to construct a hierarchy of clusters based on a single unit. The first step is to identify the units to be merged in the cluster, before selecting the two closest units based on the chosen linkage for merger. Then, HAC connects the two clusters based on the linkage, and finally aggregate all of the units. Depending on the type of dataset, HAC is required to select from different distance formulas. In HAC, Euclidean distance is used most, though the square Euclidean distance, Manhattan distance, and others are used as well. For HAC, there are different linkage criteria applied. Maximum (complete-linkage) clustering uses the following formula: $\max\{d(a,b) : a \in A, b \in B\}$, where d represents a distance function, while A and B refer to two clusters. Complete-linkage clustering would connect the two clusters via minimal value of the maximum distance between them. Minimum (single-linkage) clustering uses the following formula: $\min\{d(a,b) : a \in A, b \in B\}$. Single-linkage clustering would connect the two clusters via minimal value of the minimum distance between them. And unweighted average linkage clustering uses the following formula: $\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b)$. Unweighted average linkage clustering would connect the two clusters via minimal value of the average distance between them.

There are two clustering methods based on density models, including DBSCAN (Density-based spatial clustering of applications with noise) (Schubert et al., 2017) and OPTICS (Ordering points to identify the clustering structure) (Ankerst et al., 1999). Based on data density, DBSCAN aggregates the surrounding data points close to the center of the high density regions for the formation of clusters and marks outliers, which refer to the points fall under the low-density area. OPTICS is also based on density, similar to DBSCAN. Density-based clustering is mainly targeted at high-density areas, while those low-density areas are referred to as noise. DBSCAN involves two initial parameters, the size of the epsilon neighborhood and the number of minimum points in the epsilon region, respectively. The size of the epsilon is defined as the radius of the density cluster, and the epsilon region refers to the area of the core points. As the result of clustering is highly sensitive to these two parameters, different initial values will produce different clustering results. In OPTICS, it is sufficient to define the upper limit on the size of the epsilon neighborhood and the number of minimum points in the epsilon region, which allows OPTICS to perform clustering without a specific size limitation.

Although we have the above-mentioned solutions to obtain clusterings, it is often difficult to identify the structure of data during research applications. The efficacy of the clustering methods is determined by the structure of data. Since K-means method is required to define the number of clusters in advance, the datasets with more obvious distribution are more suitable for them. For dense, non-convex data, DBSCAN and OPTICS are potentially advantageous. As HAC is subject to no set constraints the whole graph can be sliced according to the number of clusters needed. Therefore, HAC can be applied to any datasets which can obtain the effective distance.

1.2 Measures for validating clustering

Following the clustering algorithm, the assessment of clustering is another significant part of the study. In the absence of a response, clustering validation measures can be taken to assess the efficacy of a clustering. Some popular cluster validation measures have been listed

by Halkidi et al. (2002). Herein, a few validation measures are detailed.

The silhouette value is taken as a measure to validate the consistency of the cluster (Rousseeuw, 1987). The silhouette value is calculated from the distance, which range from -1 to 1. Where high values indicate that the clustering has a good match, and vice versa. The Euclidean distance is often used in the silhouette measure. It is assumed that clustering algorithm is applied to assign data to k clusters $C = \{C_1, C_2, \ldots, C_k\}$. The silhouette value for a data point is

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad if \quad |C_i| > 1 \tag{1.3}$$

and $s(i) = 0$ if $|C_i| = 1$, where $a(i)$ represents the mean distance between the point i and other points in its cluster C_i and $b(i)$ refers to the minimum mean distance between the point i and all points other clusters $C_k \neq C_i$. That is to say, the $a(i)$ and $b(i)$ are defined as follows:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \quad and \quad b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j). \tag{1.4}$$

The smaller value of $a(i)$, the more similar the point i is to the rest of the cluster. A greater values of $b(i)$ implies that the point i is dissimilar from other clusters. As for the silhouette value, the mean value of all $s(i)$ is taken to assess the overall quality of clustering,

$$s = \frac{1}{k} \sum_{i \in C_i, C_i \in C} s(i). \tag{1.5}$$

The greater the value of s, the better the clustering is.

The Hubert Γ statistic is a measure used to test the similarity between two different clusters (Theodoridis and Koutroubas, 1999). It is obtained by the equation

$$\Gamma = \frac{1}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(i, j) Q(i, j). \tag{1.6}$$

Here, $P(i, j)$ indicates the distance between point i and j and $Q(i, j)$ is equal to the distance between centers of the clusters to which the points i and j belong. If the two points i and j fall into the same cluster, then $Q(i, j) = 0$, and there are no other clusters affected. If the

distance between two clusters c_i and c_j is close to the distance between two points i and j for which $i \in c_i$ and $j \in c_j$ for $i, j = 1, 2, \ldots, n$, then the Γ is assigned a large value. A larger value of Γ suggests a better clustering.

R-squared is a commonly used statistical measure (Sharma, 1996). In many cases, R-squared is applied to judge the goodness of fit for a regression model (Freels and Sinha, 2008). Moreover, it is suitable for the assessment of clustering. The form of the R-squared is expressed follows:

$$R^2 = \frac{SS_b}{SS_t} = \frac{SS_t - SS_w}{SS_t} = \frac{\left\{ \sum_{j=1,\ldots,v} \left[\sum_{k=1}^{n_j} (d_k - \bar{d}_k)^2 \right] \right\} - \left\{ \sum_{i=1,\ldots,c, j=1,\ldots,v} \left[\sum_{k=1}^{n_{ij}} (d_k - \bar{d}_k)^2 \right] \right\}}{\sum_{j=1,\ldots,v} \left[\sum_{k=1}^{n_j} (d_k - \bar{d}_k)^2 \right]}. \tag{1.7}$$

Where n_j represents the number of distances between each point and other points in the dataset, while n_{ij} refers to the number of distances between each point and other points in a cluster. SS_w indicates the sum of squares within group, SS_b denotes the sum of squares between groups, and SS_t stands for the total sum of squares. The d_k in the SS_t part means the distance between the point j and all other points, and $\bar{d}_k$ is defined as the mean of all d_k. For SS_w, d_k is the distance between the point j and all other points in its own cluster, with $\bar{d}_k$ as the mean value. R^2 ranges from 0 to 1. A larger value of R^2 means a smaller variance of clusters.

Dunn index is treated as a measure taken to validate that clusters are compact and well-separated (J.C.Dunn, 1974). They are two characteristics potentially indicative of a better clustering. It is defined as follows:

$$D_{n_c} = \min_{i=1,\ldots,n_c} \left\{ \min_{j=i+1,\ldots,n_c} \left(\frac{d(c_i, c_j)}{\max_{k=1,\ldots,n_c} diam(c_k)} \right) \right\}. \tag{1.8}$$

Where $d(c_i, c_j)$ is the dissimilarity function between two clusters c_i and c_j:

$$d(c_i, c_j) = \min_{x \in C_i, y \in C_j} d(x, y) \tag{1.9}$$

and $diam(c)$ indicates the diameter of cluster C

$$diam(C) = \max_{x,y \in C} d(x, y). \tag{1.10}$$

If $diam(c)$ is small, the clusters can be proven compact and well-separated. That is to say, a small $diam(c)$ indicates a large value of Dunn index.

Davies-Bouldin (DB) index provides a means to conduct internal assessment of clustering (Xiao et al., 2017). To obtain the DB value, a number of parameters are required to be determined. Firstly, we need to define $S_i = \left(\frac{1}{T_i} \sum_{j=1}^{T_i} |X_j - A_i|^2 \right)^{1/2}$ and $M_{i,j} = \|A_i - A_j\|_2 = \left(\sum_{k=1}^{n} |a_{i,k} - a_{j,k}|^2 \right)^{1/2}$, where A_i indicates the centroid of cluster C_i and T_i denotes the size of the cluster C_i. A_i is defined as the mean of all X_i in cluster C_i, and $a_{i,k}$ is denoted as a covariate in A_i. The next step is to obtain R, which is defined as:

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}}. \tag{1.11}$$

The last step is to obtain the value of DB, using the following formula:

$$DB = \frac{1}{N} \sum_{i=1}^{N} D_i. \tag{1.12}$$

Where D_i reoresents the maximum $R_{i,j}$ with $i \neq j$. S_i represents the scattering value within the cluster C_i, and $M_{i,j}$ refers to a measure of separation between cluster C_i and cluster C_j. The value of DB is indicative of the similarity between different clusters. A smaller DB values suggests a better clustering.

1.3 Enumerating All Set-connected Clusterings

Even for small datasets, enumerating all possible ways of clustering units requires prohibitive computation. Instead, we look at a small subset of these clusterings called *set-connected*

clusterings that seem likely to contain an "optimal" one for these validation measures.

These clusterings are generated as follows. We view a dataset a graph $G = (V, E)$. Each of the n units is treated as a vertex in V, and edges $ij \in E$ connect units i and j together. In this graph, a bottleneck subgraph $BG_\omega = (V, E_\omega)$ is constructed (Hochbaum and Shmoys, 1986; Higgins et al., 2016). The bottleneck subgraph with threshold ω is a subgraph of G where edges $ij \in E_\omega$ if and only if $\omega_{ij} \leq \omega$. The threshold ω is chosen to be as small as possible while ensuring that the graph G is *connected*—that is, a path of the edges in BG_ω connects every two vertices in BG_ω.

Upon forming this graph, we enumerate all ways of clustering the graph so that connected clusters are formed. These clusterings are called *set-connected* clusterings. To further restrict the number of clusterings, we also restrict the number of clusters and set minimum and maximum values on the number of units to be contained in each cluster. The algorithm for finding all set-connected clusterings can be found as part of the `redist` R package on CRAN (Fifield et al., 2016).

1.4 Test Clusterings

Our consideration is given to five datasets, each of which is comprised of a small number of units. Some datasets can distinguish the number of clusters accurately, and some datasets are symmetrical in the figure. These datasets would help us to validate the accuracy of obtaining the optimal clusterings and the preference of each measure.

For each dataset, the optimal clustering is identified for each validation measure by finding all set-connected clusterings as mentioned in Section 1.3 and by applying the validation measure on each of these clusterings. The basic process is shown in Figure 1.1.

Compared with the traditional clustering method, our method ensures the accuracy of optimality. Since there are thousands of clustering obtained altogether through set-connected partition algorithm, these clustering show reliability and are not randomly obtained. When thousands of reliable clusterings are assessed, the error will be reduced accordingly. It is a real possiblity that the traditional clustering method produces different results under the

same setting, i.e., K-means amd HAC, but the results of our method being fixed.

Additionally, the aforementioned popular clustering methods are adopted to process these datasets, including K-means, HAC, DBSCAN and OPTICS. Then, a comparison is performed between the clusterings obtained by these traditional methods and the "optimal" clustering obtained from measures. The clusterings obtained using these traditional methods are evaluated with the five validation measures we use, which is conducive to determining which type of clustering each measure prefers.

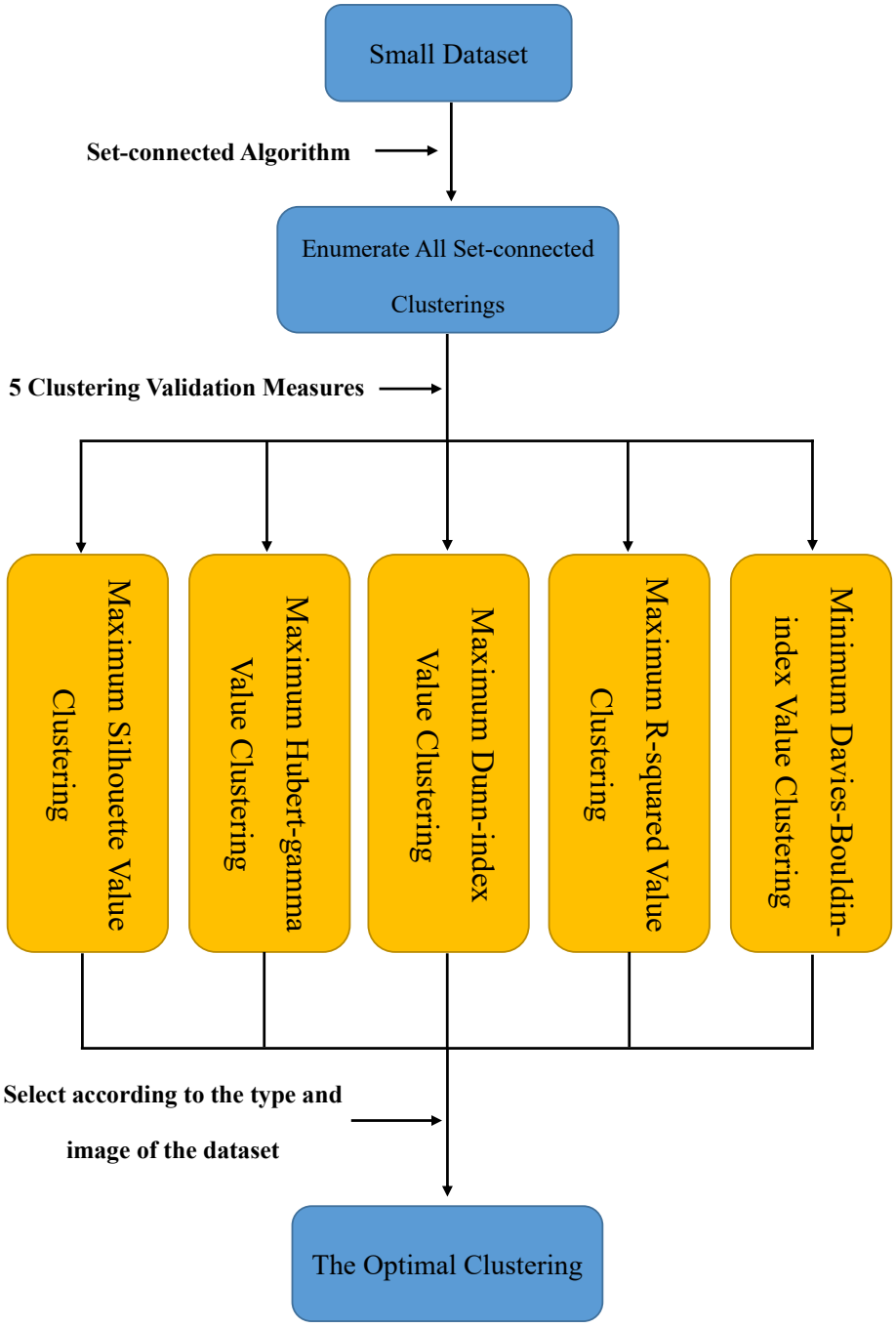

Figure 1.1: *Flow-chart for the implementation of finding the optimal clustering for each validation measure.*

Chapter 2

Results

Five different types of datasets are created. For each dataset, the pattern of clustering is different. After all the set-connected clusterings with the different number of clusters are lised for each dataset, the 5 measures as mentioned above are used to evaluate all clusterings. The optimal clustering of each measure is obtained through the comparison of scores. As for the 4 traditional clustering methods, the aforementioned steps are repeated. The clustering results as obtained by set-connected clustering method will be compared against those obtained using traditional clustering methods, which is helpful for verifying the viability and validity of these 5 clustering validation measures. In doing so, the correct clusterings and their preference for different types of clusterings can be identified.

2.1 Datasets

With regard to the dataset 1 shown in Figure 2.1, there are 20 points evenly distributed in the four corners of the plot. As the set-connected algorithm of dataset 1, we try, 3 to 6, different numbers of clusters, and with the minimum number of a cluster set to 3 and the maximum number to 7. When the number of clusters is set to 3, 84 potential set-connected clusterings will be enumerated. When we set the number of clusters to 4, there will be 6897 clusterings. And if the number of clusters is reset to 5 and 6, the number of potential

set-connected clusterings will be 12012 and 2562, respectively.

As for the dataset 2 plot shown in Figure 2.1, it can be seen clearly that there are 23 units contained in five clusters of a varying size. A trial is also conducted on 3 to 6 number of clusters using the set-connected algorithm for dataset 2, with the minimum number of a cluster set to 3. In order to complete the test, the maximum number of a cluster is set to 10 for 3 and 4 clusters and the number is set to 7 for 5 and 6 clusters. There are 481 potential set-connected clusterings with the cluster number of 3 and 12503 potential set-connected clusterings with cluster number of 4. And for the number of clusters set to 5 and 6, there will be 22346 and 39585 set-connected clusterings, respectively.

As for dataset 3, 4, and 5, they are all symmetrical. It is difficult for us to distinguish between several combinations intuitively. There are 20 points contained in the dataset 3 shown in Figure 2.1. Since dataset 3 is centrosymmetric, it is expected that all points may be divided into two clusters. Therefore, trial is conducted on 2 to 4 clusters using the set-connected algorithm, with the minimum number of a cluster set to 4 and the maximum number set to 12. There are 319 potential set-connected clusterings when the clusters are set to 2. If the clusters are set to 3 and 4, the potential results will be 4708 and 4324, respectively.

As for dataset 4, there are 18 points, and the plot is symmetrical on the horizontal axis shown in Figure 2.1. It is expected to obtain the clustering with two clusters symmetrically up and down, or divide the clustering with 3 clusters into three parts. Thus, trial is conducted on 2 to 4 number of clusters on set-connected clustering. The minimum number for each cluster is set to 4, and the maximum number is set to 10. When the cluster number is set to 2, there are 437 potential set-connected clusterings. If the cluster numbers is set to 3 and 4, then the potential number of set-connected clusterings will be 4938 and 2763, respectively.

There are 21 points contained in dataset 5. From Figure 2.1, it can be seen clearly that dataset 5 is vertically symmetrical and distributed across three areas for which it is expected that clustering will involve 2 or 3 clusters. A trial is conducted on 2 to 4 number of clusters with set-connected clusterings in the first place. Through previous experiments, however, it can be found out that there are many measures showing preference for more clusters.

Then, 5 clusters setting is added for dataset 5. Moreover, the same is done for k-means and HAC. The minimum number for each cluster is set to 4, and the maximum number to 14 for number of cluster as 2 and number to 10 for number of cluster as 3 to 5. There are as few as 8 potential set-connected clusterings when the number of cluster is set as 2, and 45 potential set-connected clusterings with the cluster number set as 3. If the cluster number is set to 4 and 5, the potential set-connected clusterings will be 104 and 16, respectively.

2.2 Implementation and Results

With regard to dataset 1, the cluster validation measures are applied to test all of these clusterings, and the clustering is selected by choosing the maximum silhouette value, the maximum Hubert-gamma value, the maximum R^2, the maximum Dunn index and the minimum DB index of all set-connected clusterings. We list all results on Table 2.1, Figure 2.2 and Figure 2.3. Since k-means and HAC can define the number of clusters, a trial is conducted on 3 to 5 clusters for each of them and the measures intended for these two method are taken as comparison. After the result for all set-connected clusterings is compared, the clustering with 4 clusters is chosen for silhouette, Hubert gamma, Dunn index and Davies-Bouldin index, while the clustering with 6 clusters is chosen for R^2 measure. For the k-means method, silhouette and Dunn index are applied to obtain the clustering with 4 clusters. Hubert gamma and Davies-Bouldin index are applied to obtain the clustering with 6 clusters, and R^2 shows the clustering involves 5 clusters. For HAC method, silhouette, Dunn index and Davies-Bouldin index indicate that the clustering involves 4 clusters, and the clustering with 6 clusters is obtained by Hubert gamma and R^2. The clustering obtained by DBSCAN involves 4 clusters, while the clustering obtained by OPTICS involves 5 clusters. All of the clusterings with 4 clusters involved produce a reasonable result as expected.

The testing results for dataset 2 are listed in Table 2.2, Figure 2.4 and Figure 2.5. For the set-connected method, the optimal clustering with 5 clusters is obtained using silhouette and Dunn index. Hubert gamma, R^2 and Davies-Bouldin index are applied to obtain the clustering with 6 clusters. As for k-means, only silhouette is applied to obtain the clustering

Table 2.1: *Cluster validation measures results for dataset 1*

Clustering method	Measures for assessing clustering	Values of optimal clusterings (number of clusters)
Set-connected Clusterings	Silhouette	0.620715(4)
	Hubert Gamma	4.289470(4)
	R-squared	0.993003(6)
	Dunn index	1.414214(4)
	Davies-Bouldin index	0.800000(4)
K-means	Silhouette	0.620715(4)
	Hubert Gamma	4.366474(6)
	R-squared	0.995777(6)
	Dunn index	1.414214(4)
	Davies-Bouldin index	0.609096(6)
Hierarchical	Silhouette	0.620715(4)
	Hubert Gamma	4.366486(6)
	R-squared	0.995777(6)
	Dunn index	1.414214(4)
	Davies-Bouldin index	0.800000(4)
DBSCAN	Silhouette	0.620715(4)
	Hubert Gamma	4.289470(4)
	R-squared	0.992213(4)
	Dunn index	1.414214(4)
	Davies-Bouldin index	0.800000(4)
OPTICS	Silhouette	0.374396(5)
	Hubert Gamma	3.852513(5)
	R-squared	0.985447(5)
	Dunn index	0.395285(5)
	Davies-Bouldin index	1.477132(5)

has 5 clusters. Hubert gamma, R^2, Dunn index and Davies-Bouldin index indicate that the clustering involves 6 clusters. For HAC, the clustering with 5 clusters are chosen by silhouette and Dunn index. Hubert gamma, R^2 and Davies-Bouldin index are applied to obtain the clustering involving 6 clusters. The clustering obtained by DBSCAN contains 5 clusters, while the clustering obtained by OPTICS involves 5 clusters. In addition to the result obtained by OPTICS, the clustering with 5 clusters produces reasonable result.

As for dataset 3, the final results are shown in Table 2.3, Figure 2.6 and Figure 2.7. For set-connected clusterings, only silhouette is applied to obtain the result for clustering that involves 2 clusters. Hubert gamma, R^2, Dunn index and Davies-Bouldin index are

Table 2.2: *Cluster validation measures results for dataset 2*

Clustering method	Measures for assessing clustering	Values of optimal clusterings (number of clusters)
Set-connected Clusterings	Silhouette	0.624640(5)
	Hubert Gamma	3.701114(6)
	R-squared	0.997546(6)
	Dunn index	1.382932(5)
	Davies-Bouldin index	0.657519(6)
K-means	Silhouette	0.624640(5)
	Hubert Gamma	3.669669(6)
	R-squared	0.995138(6)
	Dunn index	1.044031(6)
	Davies-Bouldin index	0.655834(6)
Hierarchical	Silhouette	0.624640(5)
	Hubert Gamma	3.682135(6)
	R-squared	0.996336(6)
	Dunn index	1.382932(5)
	Davies-Bouldin index	0.656256(6)
DBSCAN	Silhouette	0.624640(5)
	Hubert Gamma	3.659150(5)
	R-squared	0.994573(5)
	Dunn index	1.382932(5)
	Davies-Bouldin index	0.702293(5)
OPTICS	Silhouette	0.547919(5)
	Hubert Gamma	3.534014(5)
	R-squared	0.983944(5)
	Dunn index	0.685763(5)
	Davies-Bouldin index	0.852880(5)

applied to obtain the results for clustering that contains 4 clusters. K-means and HAC has the same situation as set-connected clusterings. Although the clustering have the same clusters, the clustering for each of them are different. The clustering obtained by DBSCAN contains 3 clusters, while the clustering obtained by OPTICS involves 3 clusters. All of the clusterings with 2 clusters produce the results as expected. In addition to the clustering obtained by DB index using set-connected clusterings method, although the other clustering with 4 clusters are different, they are all centrosymmetric structures. From this perspective, these clusterings with 4 clusters are deemed reasonable as well. And the results of DBSCAN and OPTICS can be ignored basically which is due to plenty of noise shown in the figure.

Table 2.3: *Cluster validation measures results for dataset 3*

Clustering method	Measures for assessing clustering	Values of optimal clusterings (number of clusters)
Set-connected Clusterings	Silhouette	0.528297(2)
	Hubert Gamma	3.298002(4)
	R-squared	0.983940(4)
	Dunn index	0.755511(4)
	Davies-Bouldin index	0.963887(4)
K-means	Silhouette	0.528297(2)
	Hubert Gamma	3.396084(4)
	R-squared	0.985566(4)
	Dunn index	1.035744(4)
	Davies-Bouldin index	0.998422(4)
Hierarchical	Silhouette	0.528297(2)
	Hubert Gamma	3.330704(4)
	R-squared	0.975390(4)
	Dunn index	0.751617(4)
	Davies-Bouldin index	1.053518(4)
DBSCAN	Silhouette	-0.037873(3)
	Hubert Gamma	1.026219(3)
	R-squared	0.558376(3)
	Dunn index	0.202949(3)
	Davies-Bouldin index	3.263959(3)
OPTICS	Silhouette	-0.330371(3)
	Hubert Gamma	0.452710(3)
	R-squared	0.296555(3)
	Dunn index	0.202949(3)
	Davies-Bouldin index	3.579808(3)

The final results for dataset 4 are listed in Table 2.4, Figure 2.8 and Figure 2.9. As for set-connected algorithm, the clustering with 2 clusters by silhouette and the clustering with 3 clusters by Dunn index are chosen. Hubert gamma, R^2 and Davies-Bouldin index are used to screen out the clustering with 4 clusters involved. Through K-means and HAC has the same situation as set-connected clusterings, the clustering for each of them is different. The clustering obtained by DBSCAN contains 3 clusters, while the clustering obtained by OPTICS involves 4 clusters. For the clustering with 2 clusters, set-connected with silhouette and k-means compound the expected result. And set-connected algorithm by Dunn index, k-means, HAC and DBSCAN get the expected result for clustering with 3 clusters also. For

other clusterings obtained, they are shown not to be as reasonable as required.

Table 2.4: *Cluster validation measures results for dataset 4*

Clustering method	Measures for assessing clustering	Values of optimal clusterings (number of clusters)
Set-connected Clusterings	Silhouette	0.508489(2)
	Hubert Gamma	9.244584(4)
	R-squared	0.980601(4)
	Dunn index	0.707955(3)
	Davies-Bouldin index	1.260339(4)
K-means	Silhouette	0.508489(2)
	Hubert Gamma	9.466156(4)
	R-squared	0.973296(4)
	Dunn index	0.707955(3)
	Davies-Bouldin index	1.329216(4)
Hierarchical	Silhouette	0.510266(2)
	Hubert Gamma	9.466156(4)
	R-squared	0.973296(4)
	Dunn index	0.707955(3)
	Davies-Bouldin index	1.329216(4)
DBSCAN	Silhouette	0.436226(3)
	Hubert Gamma	9.068886(3)
	R-squared	0.955676(3)
	Dunn index	0.707955(3)
	Davies-Bouldin index	1.375237(3)
OPTICS	Silhouette	0.278079(4)
	Hubert Gamma	7.308639(4)
	R-squared	0.930078(4)
	Dunn index	0.359410(4)
	Davies-Bouldin index	3.958782(4)

The final results for dataset 5 are listed in Table 2.5, Figure 2.10 and Figure 2.11. As for set-connected clusterings, the clustering with 3 clusters is chosen by silhouette, and the clustering with 4 clusters is chosen by Hubert gamma. Besides, the clustering with 5 clusters is chosen by R^2, Dunn index and Davies-Bouldin index. For k-means, silhouette demonstrates that the optimal clustering contains 3 clusters, and other measures show that the optimal clustering involves 5 clusters. For HAC, the clustering with 3 clusters also shows the highest silhouette values, which is however different from the clustering obtained by set-connected and k-means. The clustering with 4 clusters is chosen for Dunn index, with other

measures showing that the optimal clustering contains 5 clusters. The clustering obtained by DBSCAN involves 3 clusters, while the clustering obtained by OPTICS contains 3 clusters. For the clustering with 3 clusters, set-connected with silhouette and k-means compound the expected result. The clustering obtained by DBSCAN is also reasonable, despite some difference.

Table 2.5: *Cluster validation measures results for dataset 5*

Clustering method	Measures for assessing clustering	Values of optimal clusterings (number of clusters)
Set-connected Clusterings	Silhouette	0.506199(3)
	Hubert Gamma	9.696242(4)
	R-squared	0.983315(5)
	Dunn index	0.614636(5)
	Davies-Bouldin index	1.215449(5)
K-means	Silhouette	0.506199(3)
	Hubert Gamma	9.902522(5)
	R-squared	0.974304(5)
	Dunn index	0.620484(5)
	Davies-Bouldin index	1.118323(5)
Hierarchical	Silhouette	0.500292(3)
	Hubert Gamma	9.884463(5)
	R-squared	0.972620(5)
	Dunn index	0.583095(4)
	Davies-Bouldin index	1.174966(5)
DBSCAN	Silhouette	0.383481(3)
	Hubert Gamma	8.468460(3)
	R-squared	0.922672(3)
	Dunn index	0.406885(3)
	Davies-Bouldin index	2.024551(3)
OPTICS	Silhouette	0.365489(3)
	Hubert Gamma	8.483592(3)
	R-squared	0.920787(3)
	Dunn index	0.339935(3)
	Davies-Bouldin index	2.043995(3)

2.3 Discussion

After the comparison between our set-connected methods and those traditional methods, the viability of set-connected method was confirmed in first place. Then, some reasonable results were obtained as expected, and some of the results looking odd. Nevertheless, some new discoveries were made.

The first one is that silhouette is the most effective measures identified in our study. The silhouette is most likely to provide the most reasonable clustering as we expected for all datasets for most of the clustering methods, especially the set-connected method. Besides, it is effective in capturing the clustering with "non-compact" clusters. Secondly, other validity measures show preference for the clustering with more clusters in most part, and R^2 even showed this feature in dataset 1. It is also the case for datasets 2, 3 and 4, for which one more clusters was added for data 5. These results also confirmed this argument, as R^2, Dunn index and Davies-Bouldin index pick the clustering which has the highest cluster number. Among the tradition clustering methods, the DBSCAN method is also excellent in picking up "non-compact" clusters. The DBSCAN method identifies the reasonable clustering well except dataset 3. Due to the inability to set the number of clusters for DBSCAN, some set of exact clustering can be obtained directly on a frequent basis. After the clustering with noise is excluded, the cluster number of clustering picked by DBSCAN can be taken as a reference for our method. For k-means and HAC, the expected clustering can be obtained when the correct cluster numbers are set in most of the situations. It is possible for them to generate some clusterings with only 1 or 2 units involved in a cluster when too many clusters are chosen. This will affect the evaluation of the measures for their validity to some extent. But for validity measures for k-means in dataset 3, they show more reasonable values on clustering with 4 clusters, which is because the size of each cluster is different from that in our method. Therefore, the measures for k-means can provide some guidance on the size of cluster for our set-connected algorithm. This is because OPTICS failed to produce a satisfactory result, a result of which the cluster number of the clustering can be taken as reference after the removal of noises from OPTICS results.

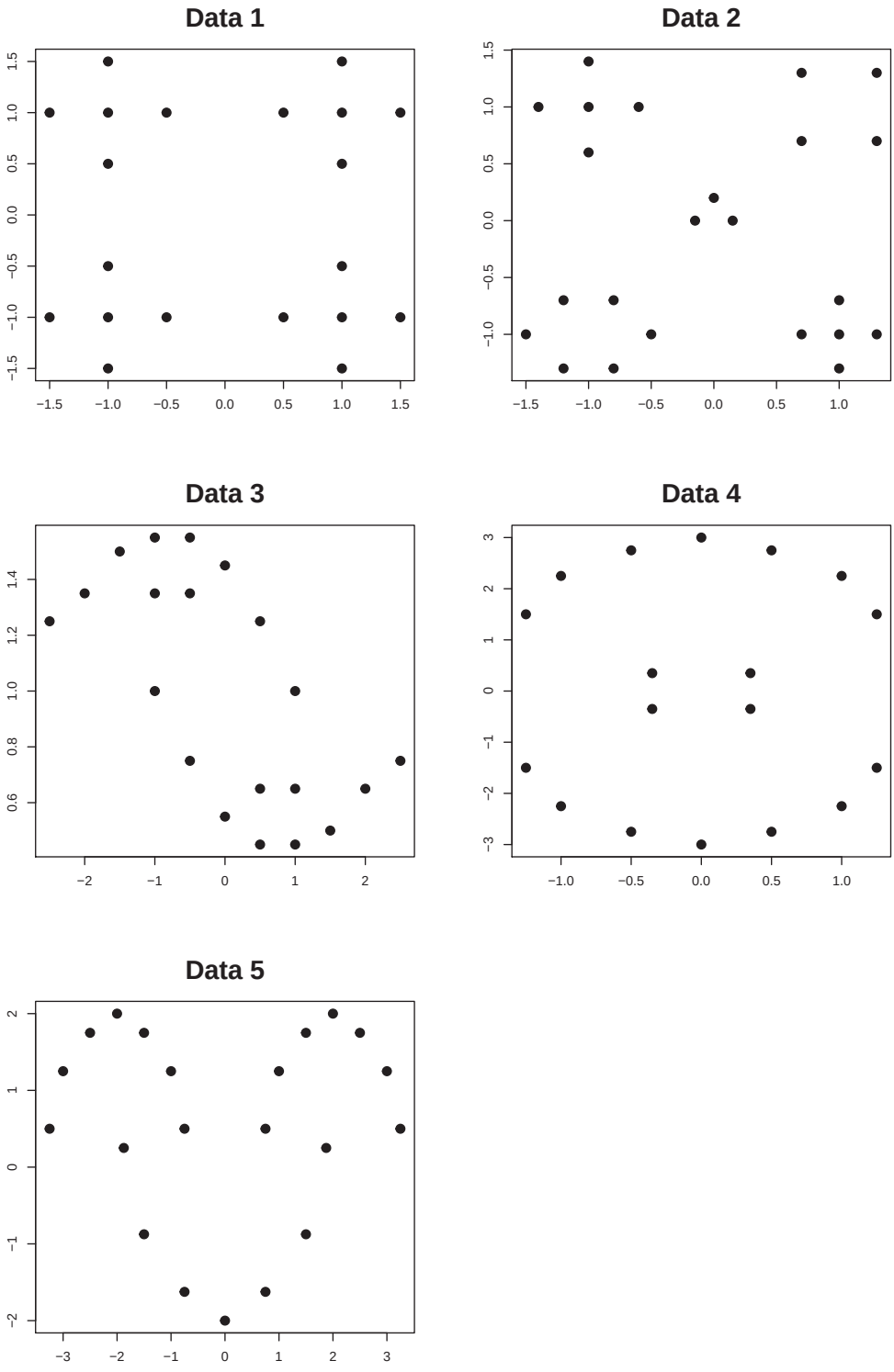

Figure 2.1: *Plots of all datasets.*

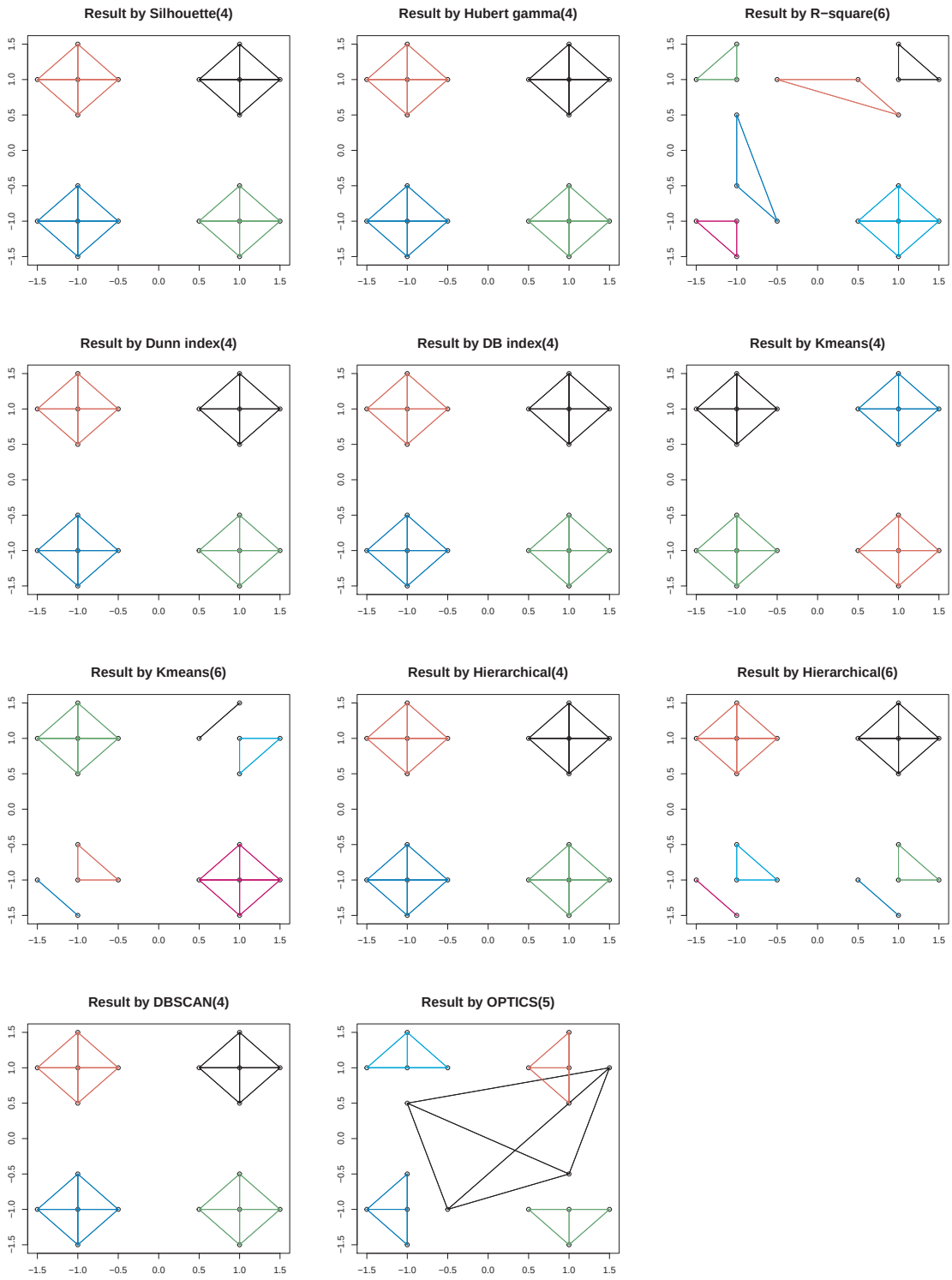

Figure 2.2: *Results of clustering methods for dataset 1. We connect all units in the same clusters by using the same color. The content in parentheses behind the number in parenthesis indicates the number of clusters.*

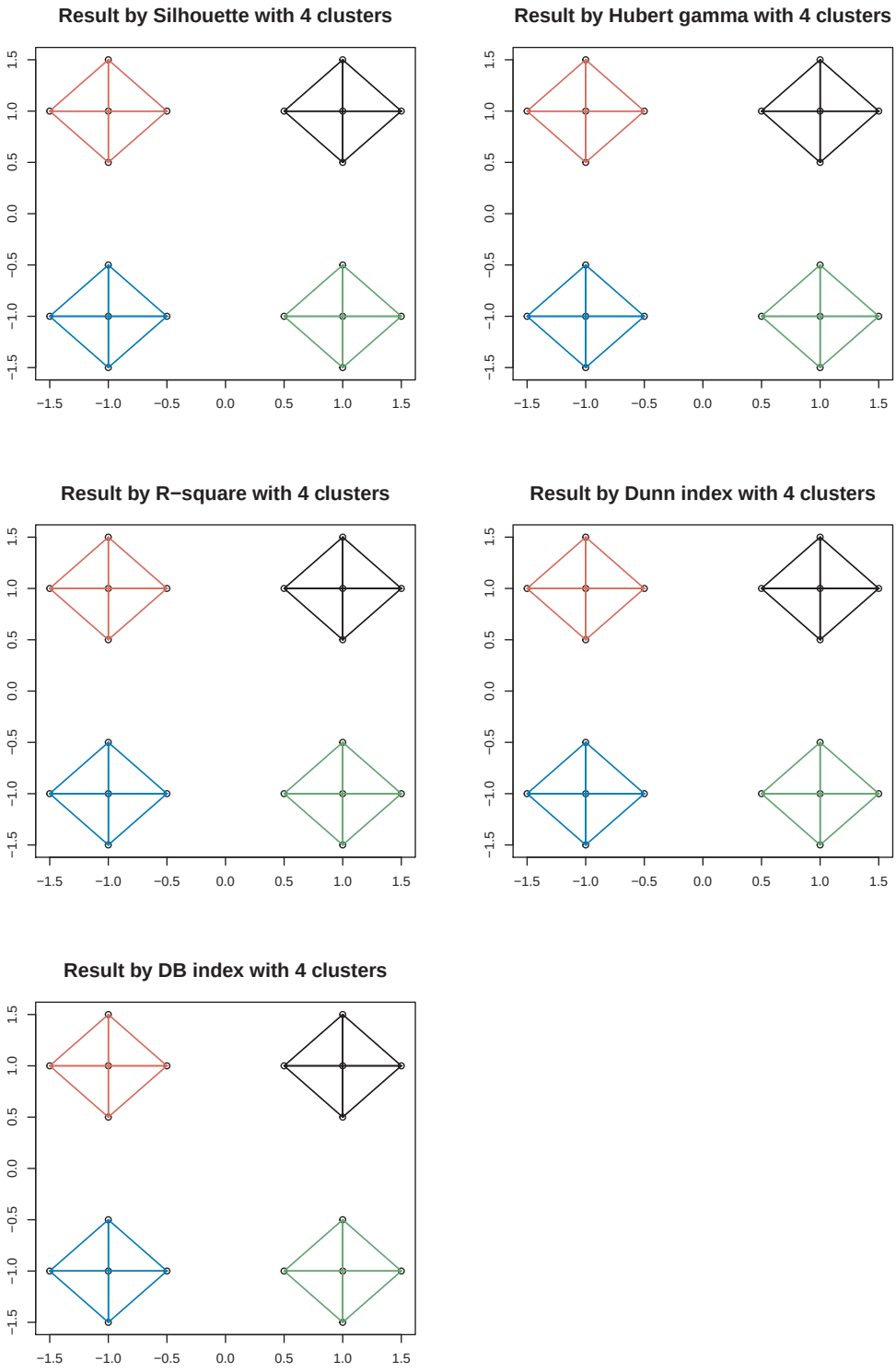

Figure 2.3: *The result of five validation measures given 4 clusters on dataset 1.*

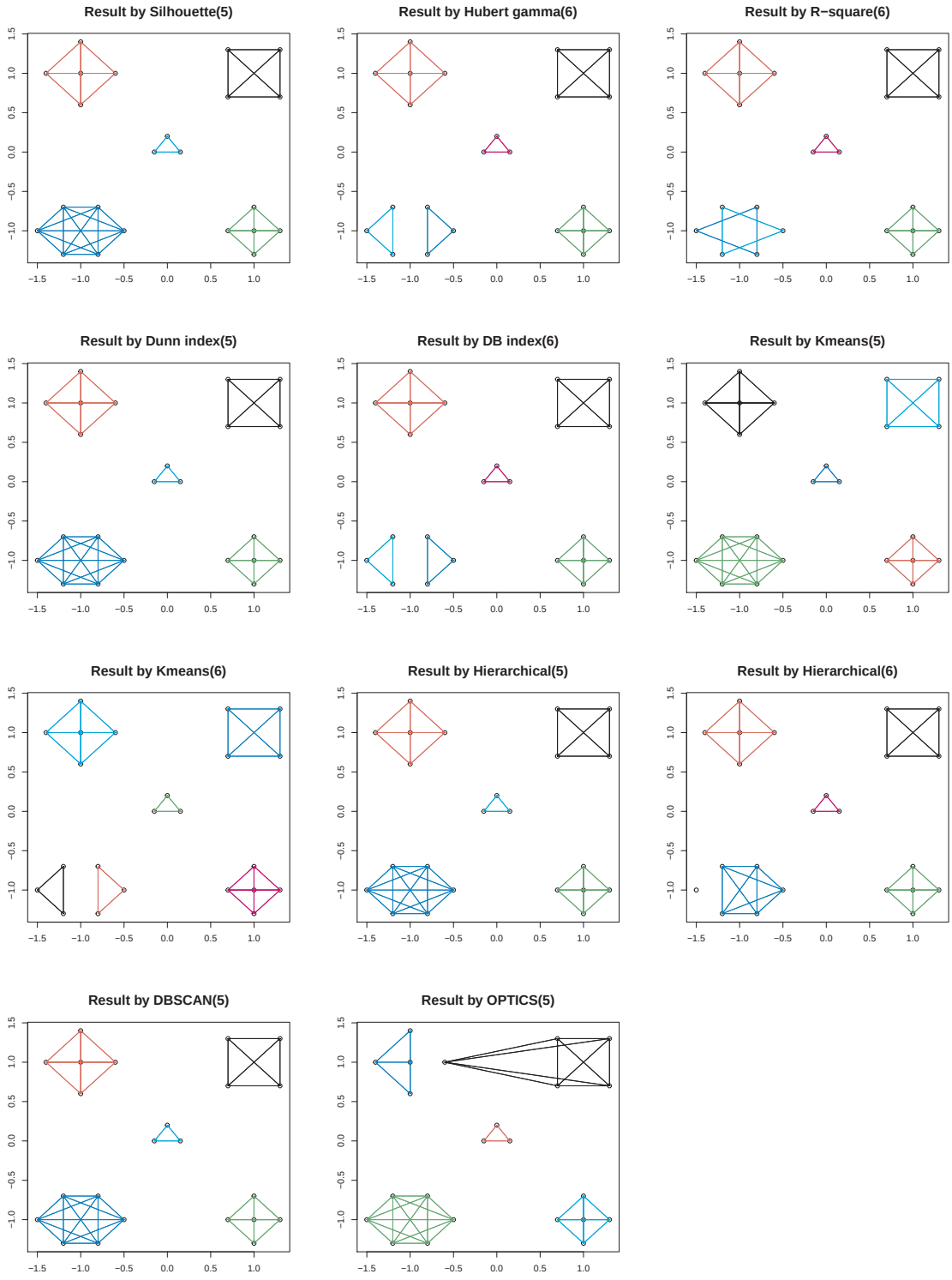

Figure 2.4: *Results for dataset 2. We connect all units in the same clusters by using the same color. The number in parenthesis indicates the number of clusters.*

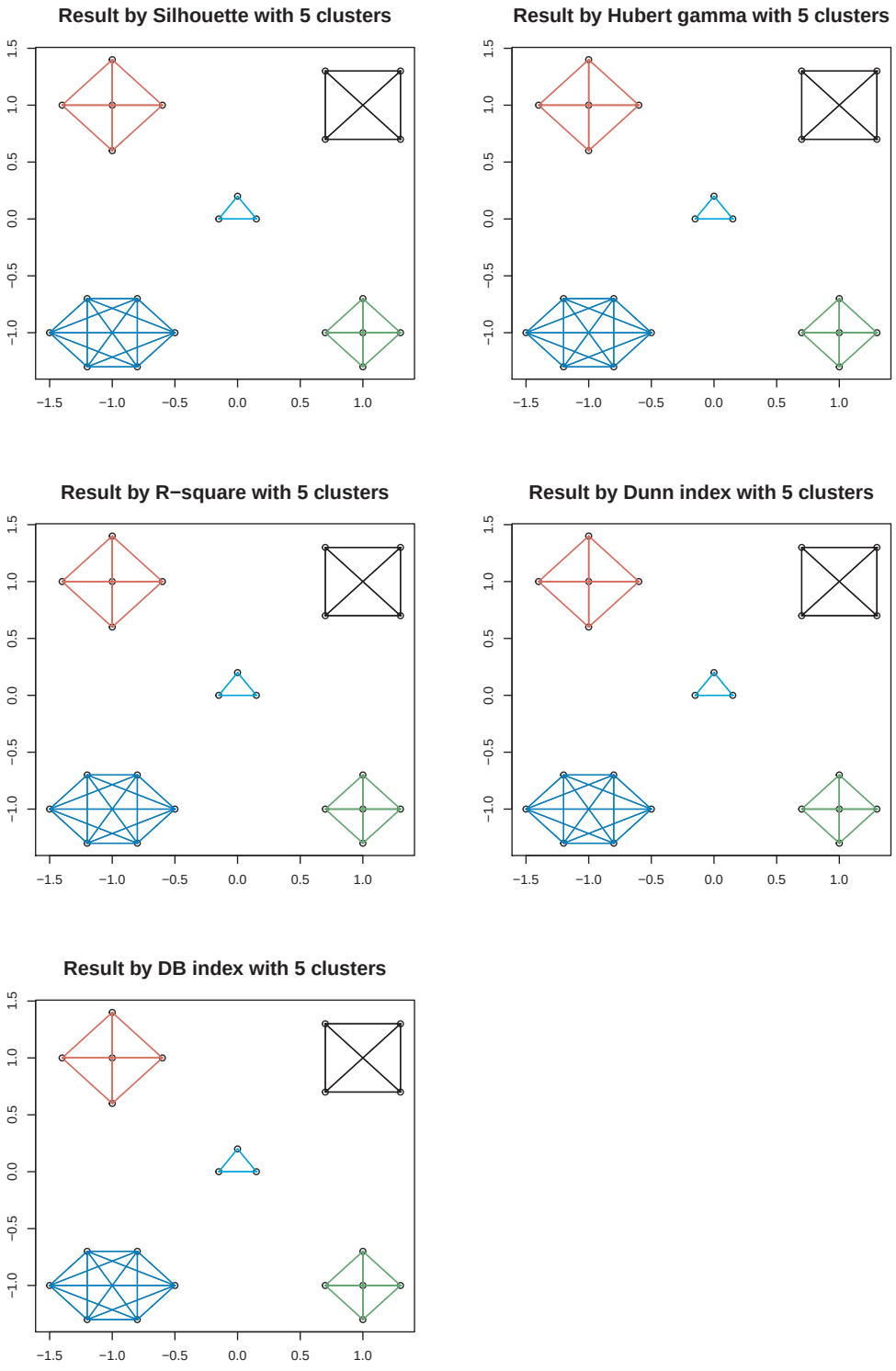

Figure 2.5: *The result of five validation measures given 5 clusters on dataset 2.*

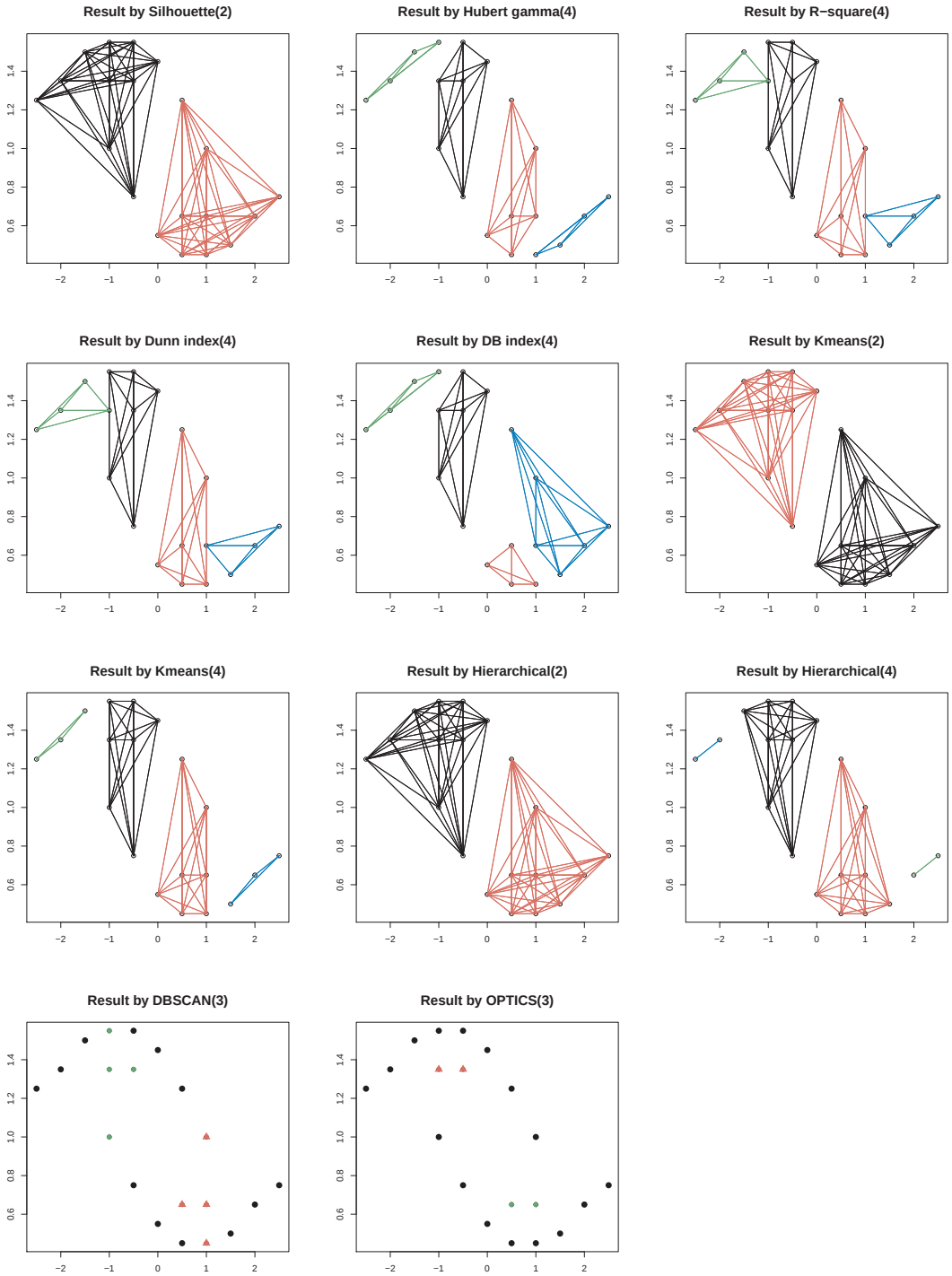

Figure 2.6: *Results for dataset 3. We connect all units in the same clusters by using the same color. Since the connections of DBSCAN and OPTICS are very chaotic, then we use different colors and different shapes to represent clusters. The number in parenthesis indicates the number of clusters.*

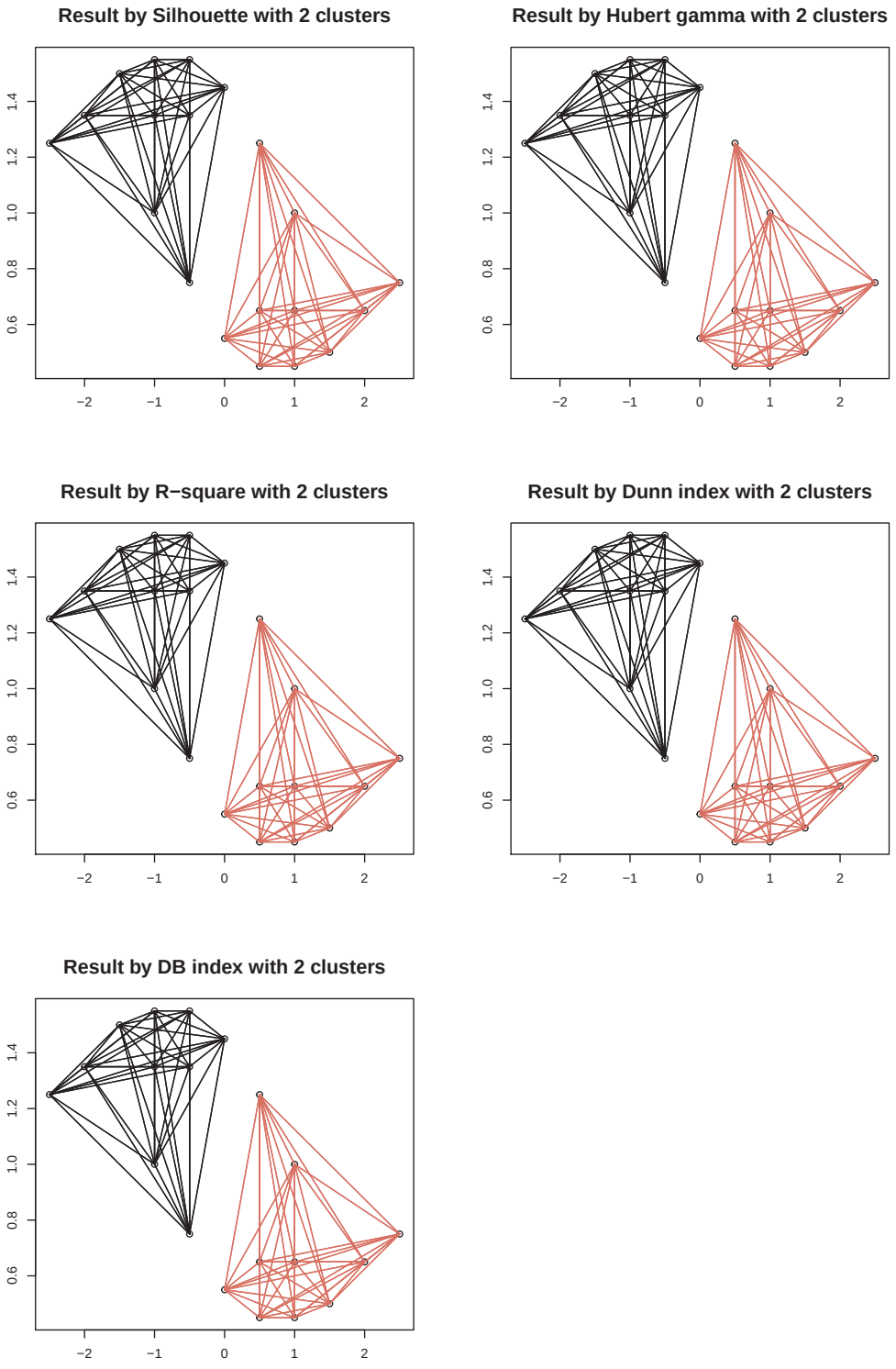

Figure 2.7: *The result of five validation measures given 2 clusters on dataset 3.*

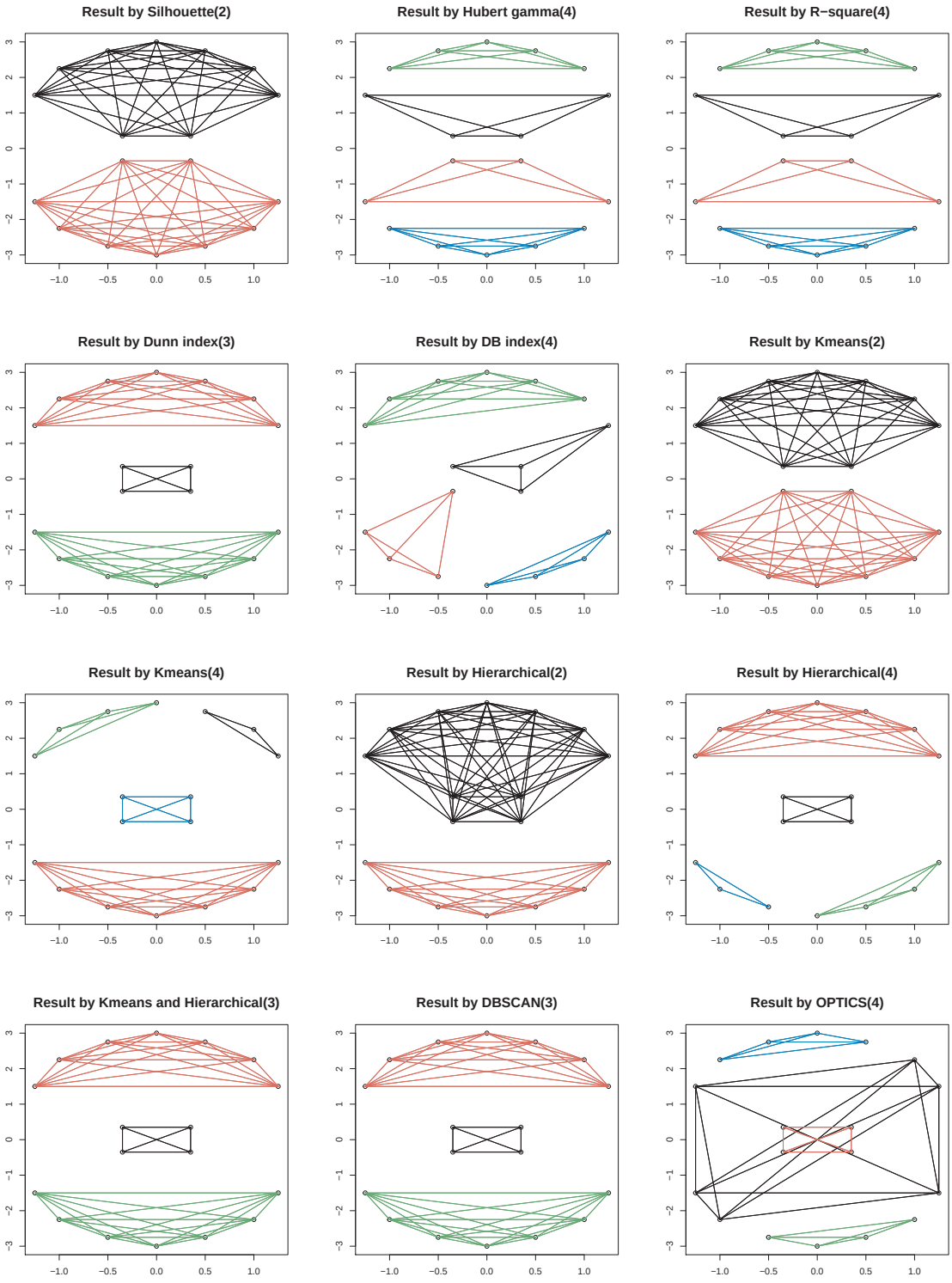

Figure 2.8: *These are results for dataset 4. We connect all units in the same clusters by using the same color. The number in parenthesis indicates the number of clusters.*

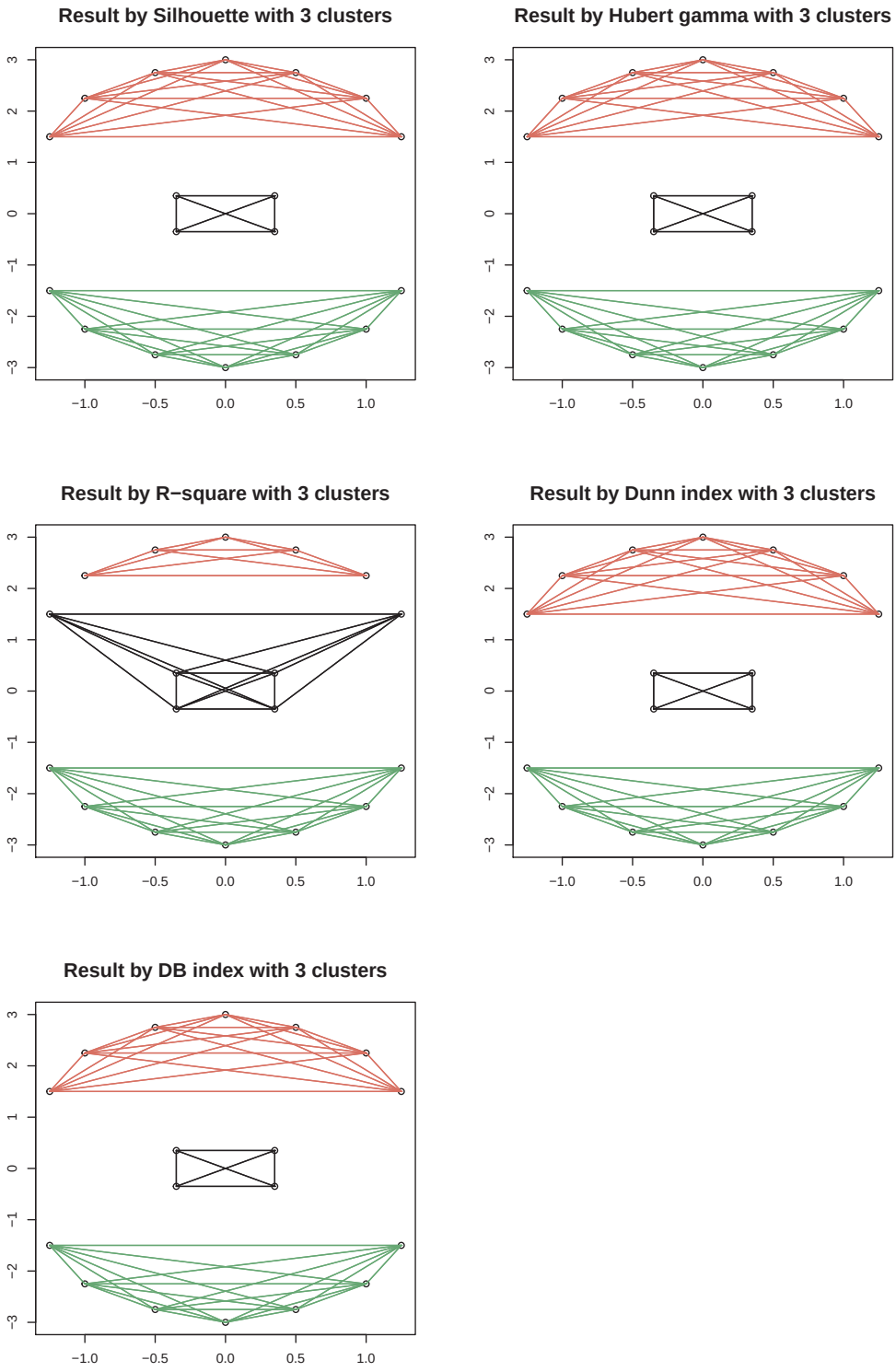

Figure 2.9: *The result of five validation measures given 3 clusters on dataset 4.*

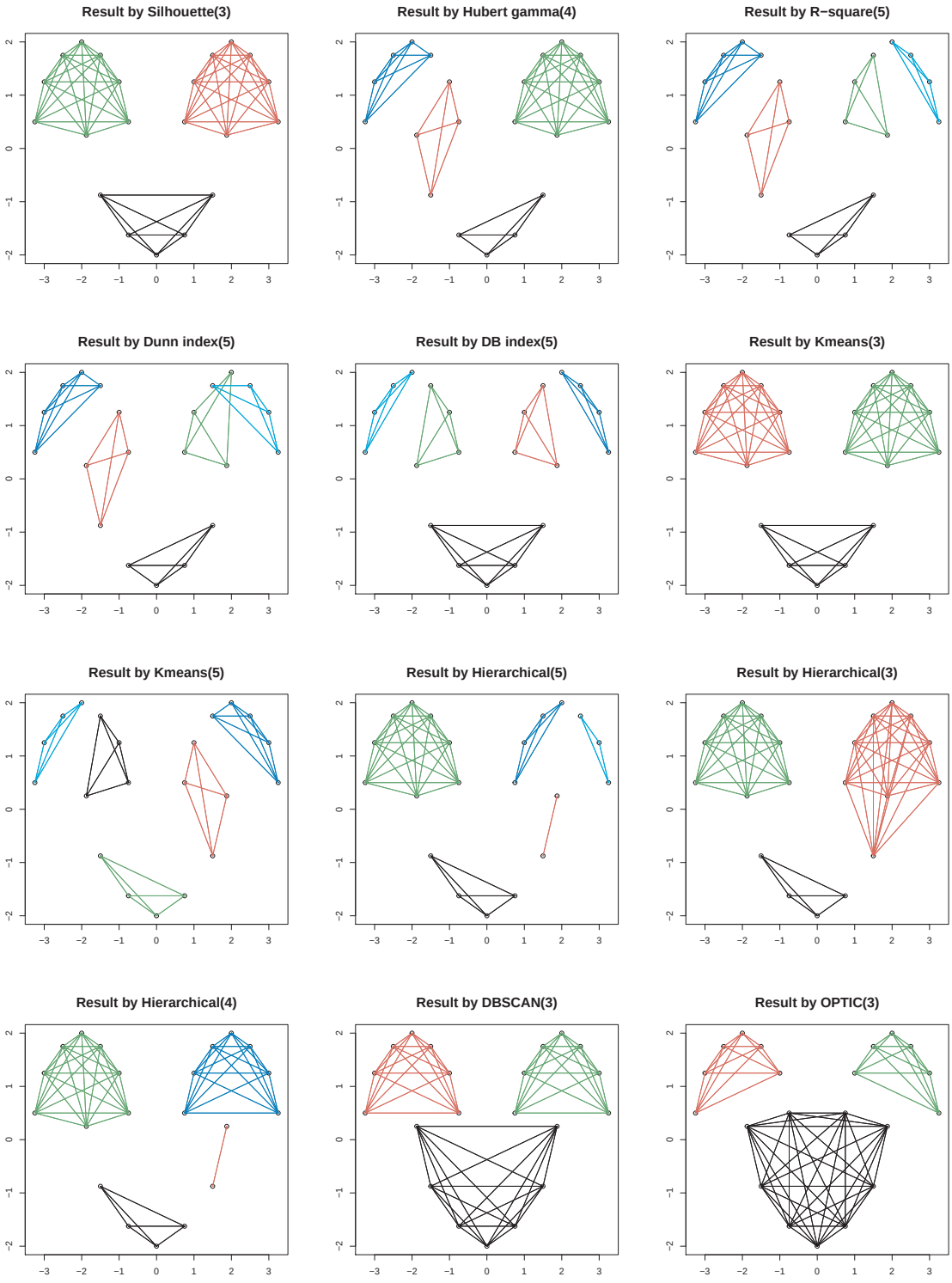

Figure 2.10: *These are results for dataset 5. We connect all units in the same clusters by using the same color. The number in parenthesis indicates the number of clusters.*

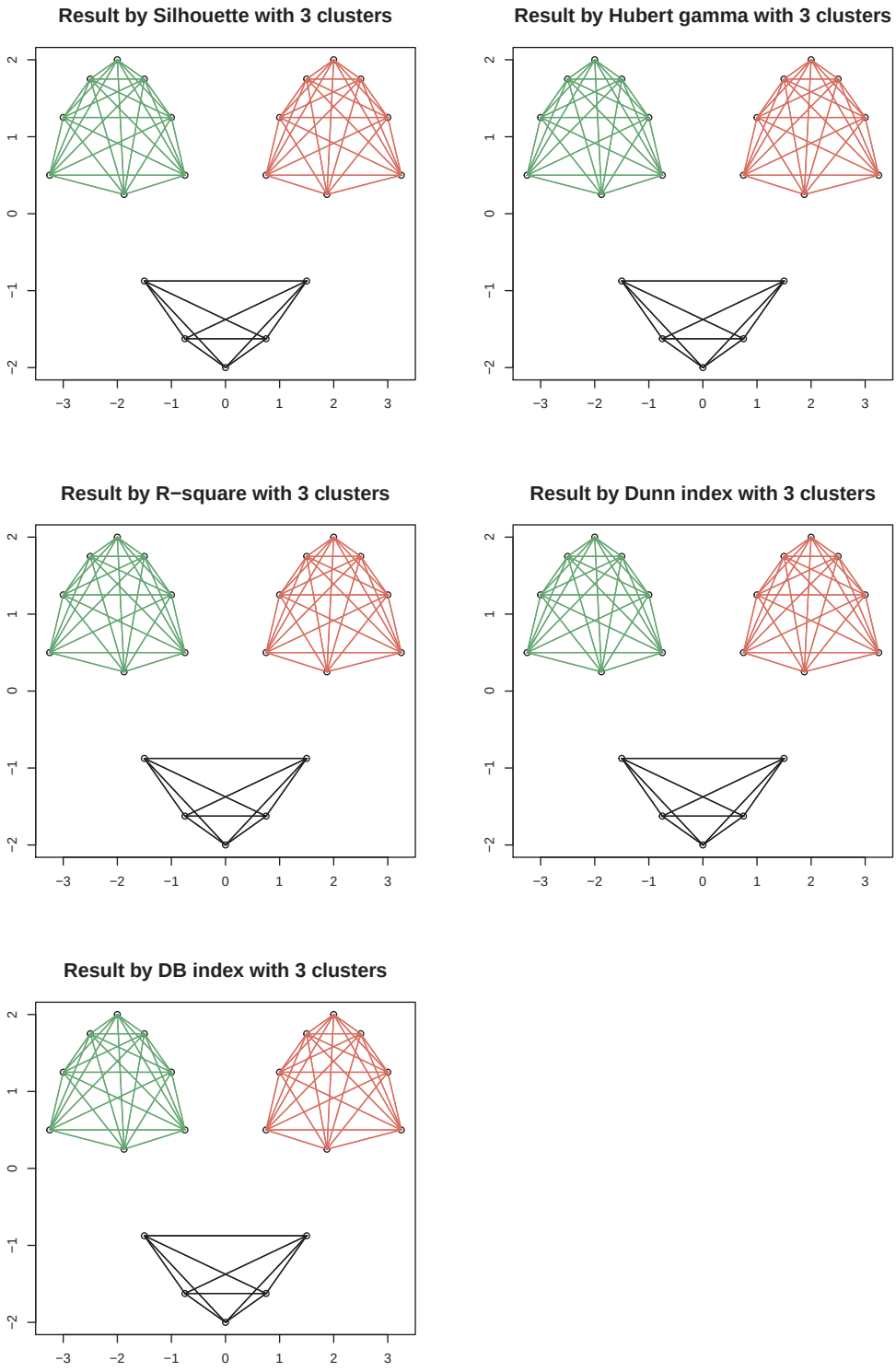

Figure 2.11: *The result of five validation measures given 3 clusters on dataset 5.*

Chapter 3

Conclusion

3.1 Conclusion

In this study, we discussed the preferences of 5 popular cluster validation measures and the viability of applying them to identify the "correct" clustering among all potential clusterings.

First of all, clustering verification measures were taken to test all potential clustering that can identify the "correct" clustering. We find each measure has its own preferences. The silhouette measure seems to be most effective in capturing connected regions. The Dunn index performs well for the data with relatively obvious known cluster numbers. The other three measures, including-Hubert-Gamma, R-squared, and Davies-Bouldin index, prefer clusterings that contain a large number of clusters.

3.2 Future Study

In small dataset experiments, our method can be used to evaluate the optimal clustering. Faced with relatively complex data, our method can also give a variety of possible clusterings which selected by the cluster validation measures. According to different data types and requirements of experimental, measures can be added or changed to improve the method.

In the process of obtaining the set-connected clusterings, some datasets may be unlikely

to connect units by connecting the maximum distance of the minimum distance between all units, which will make us unable to enumerate set-connected clustering. We may aim to automate the procedure to obtain all set-connected clusterings.

Finally, this type of method may be useful in the design of experiments and observational studies. For example, given a measure that assesses the projected power of an experiment, the procedure uncovering optimal configuration of experimental blocks.

Bibliography

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning data mining, inference, and prediction.* New York : Springer, 2001.

Marcel Brun, Chao Sima, Jianping Hua, James Lowey, Brent Carroll, Edward Suh, and Edward R. Dougherty. Model-based evaluation of clustering validation measures. *Pattern Recognition.*, 40(3):807–824, 2007.

Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of internal clustering validation measures. In *2010 IEEE International Conference on Data Mining*, pages 911–916, 2010.

Alboukadel Kassambara and Fabian Mundt. *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*, 2020. URL https://CRAN.R-project.org/package=factoextra. R package version 1.0.7.

Nasrin Taherkhani and Samuel Pierre. Centralized and localized data congestion control strategy for vehicular ad hoc networks using a machine learning clustering algorithm. *IEEE/IET Electronic Library.*, 17:3275–3285, 2016.

M.Rostam Niakan Kalhori and M.H. Fazel Zarandi. Interval type-2 credibilistic clustering for pattern recognition. *Pattern Recognition.*, 48:3652, 2015.

D Jimenez and V Vidal. Parallel implementation of information retrieval clustering models. *High Performance Computing For Computational Science.*, 3402:129–141, 2004,2005.

J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics).*, 28:100–108, 1979.

Frank Nielsen. *Introduction to HPC with MPI for data science.* Cham : Springer, 2016.

Erich Schubert, Jörg Sander, Martin Ester, Hans Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Transactions on Database Systems.*, 42:1–21, 2017.

M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J.Sander. Optics: Ordering points to identify the clustering structure. *SIGMOD Record (ACM Special Interest Group on Management of Data).*, 28:49–60, 1999.

Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Clustering validity checking methods: part ii. *ACM SIGMOD Record.*, 31:19–27, 2002.

Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics.*, 20:53–65, 1987.

S. Theodoridis and K. Koutroubas. *Pattern recognition.* Academic Press, 1999.

S.C. Sharma. *Applied Multivariate Techniques.* John Wiley & Sons, 1996.

S. Freels and K. Sinha. R -squared for general regression models in the presence of sampling weights. *Statistics and Probability Letters.*, 78(12):1671–1672, 2008.

J.C.Dunn. Well-separated clusters and the optimal fuzzy partitions. *Journal of Cybernet.*, 4:95–104, 1974.

Junwei Xiao, Jianfeng Lu, and Xiangyu Li. Davies bouldin index based hierarchical initialization k-means. *Intelligent Data Analysis.*, 21(6):1327–1338, 2017.

Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM.*, 33:533–550, 1986.

Michael J. Higgins, Fredrik Sävje, and Jasjeet S. Sekhon. Improving massive experiments with threshold blocking.(colloquium paper: Statistics). *Proceedings of the National Academy of Sciences of the United States.*, 113(27):7369, 2016.

Ben Fifield, Alexander Tarr, Michael Higgins, and Kosuke Imai. redist: Markov chain monte carlo methods for redistricting simulation. Available at The Comprehensive R Archive Network (CRAN), 2016. URL https://CRAN.R-project.org/package=redist.

Appendix A

R-Code

```
#rm(list = ls())

source("countPartitionsRWrapper.R")
#library("fpc")
library(dbscan)

########## data set ########################

smallstar = function(center, radius){
  x1 = center[1]
  x2 = center[2]
  r = radius

  matrix(c(x1 + r, x2,
           x1 - r, x2,
           x1, x2 + r,
           x1, x2 - r,
```

```
          x1, x2), ncol = 2, byrow = TRUE)
}

smallsquare = function(center, radius){
  x1 = center[1]
  x2 = center[2]
  r = radius

  matrix(c(x1 + r, x2 + r,
           x1 - r, x2 + r,
           x1 + r, x2 - r,
           x1 - r, x2 - r),
         ncol = 2, byrow = TRUE)
}

smalltriangle = function(center, radius){
  x1 = center[1]
  x2 = center[2]
  r = radius

  matrix(c(x1 + .75*r, x2,
           x1 - .75*r, x2,
           x1, x2 + r),
         ncol = 2, byrow = TRUE)
}

smallhex = function(center, radius){
  x1 = center[1]
  x2 = center[2]
```

```
  r = radius

  matrix(c(x1 + r, x2 + 1.5*r,
           x1 + r, x2 - 1.5*r,
           x1 - r, x2 + 1.5*r,
           x1 - r, x2 - 1.5*r,
           x1 + 2.5*r, x2,
           x1 - 2.5*r, x2),
         ncol = 2, byrow = TRUE)
}

smallarc = function(center, radius, parity = 1){
  x1 = center[1]
  x2 = center[2]
  r = radius

  ret = matrix(c(x1, x2,
                 x1 - .5*r, x2 - .25*r,
                 x1 + .5*r, x2 - .25*r,
                 x1 - r, x2 - .75*r,
                 x1 + 1*r, x2 - .75*r,
                 x1 - 1.25*r, x2 - 1.5*r,
                 x1 + 1.25*r, x2 - 1.5*r),
               ncol = 2, byrow = TRUE)

  ret = ret*parity
}
```

```
smalltail = function(start,radius,parity = 1, flip = 1){
  x1 = start[1]
  x2 = start[2]
  r = radius

  ret = matrix(c(x1, x2,
                 x1 +.5*r, x2 - .25*r,
                 x1 + r, x2 - .45*r,
                 x1 + 1.5*r, x2 - .35*r,
                 x1 + 1.5*r, x2 - .55*r,
                 x1 + 2*r, x2 - .35*r,
                 x1 + 2*r, x2 - .55*r,
                 x1 + 2.5*r, x2 - .5*r,
                 x1 + 3*r, x2 - .35*r,
                 x1 + 3.5*r, x2 - .25*r),
               ncol = 2, byrow = TRUE)

  ret[,1] = ret[,1]*flip
  ret*parity

}

g1 = rbind(
  smallstar(c(1,1),.5),
  smallstar(c(-1,1),.5),
  smallstar(c(1,-1),.5),
  smallstar(c(-1,-1),.5)
)
```

```
g2 = rbind(
  smallsquare(c(1,1),.3),
  smallstar(c(-1,1),.4),
  smallstar(c(1,-1),.3),
  smallhex(c(-1,-1),.2),
  smalltriangle(c(0,0),.2)
)

g3 = rbind(smalltail(c(-1,1), 1),
           smalltail(c(-1,-1), 1, parity = -1))

g4 = rbind(smallsquare(c(0,0), radius = .35),
           smallarc(c(0,3),1),
           smallarc(c(0,3),1,parity = -1))

g5 = rbind(smallarc(c(0,2),1.5,parity = -1),
           smallarc(c(-2,2),1),
           smallarc(c(2,2),1))



########## functions  ############################

distance <- function(x,y){
  a = sqrt((x[1]-y[1])^2+(x[2]-y[2])^2)
  as.numeric(a)
}

#sihouette function
```

```
silh <- function(st,np){
  npl = nrow(np)
  xx = rep(0,npl)
  np = cbind(np,xx)
  nst = length(unique(st))
  np[,3] = st
  np = np[order(np[,3]),]
  newnp = NULL
  renp = np
  cnum = rep(0,nst)

  for (i in 1:nst) {
    cnum[i] = sum(st==i)
    pp = 1:cnum[i]
    newnp[[i]] = renp[pp,]
    renp = renp[-pp,]
  }

  aa = np[,1:2]
  dm <- as.matrix(dist(aa))
  nr <- nrow(dm)
  diag(dm) = NA
  dm <- t(matrix(t(dm)[which(!is.na(dm))],nrow = (nr-1),ncol = nr))

  s =NULL
  for (j in 1:npl) {
    nc <- np[j,3]
    newcnum = cnum
```

```
    newcnum[nc] = newcnum[nc] - 1
    ds = dm[j,]
    zz = NULL
    for (k in 1:nst) {
      nn = newcnum[k]
      nnn = 1:nn
      zz[[k]] = ds[nnn]
      ds = ds[-nnn]
    }
    zz <- lapply(zz, mean)
    a = zz[[nc]]
    zz[[nc]] = NULL
    b = min(unlist(zz))
    s[[j]] = (b-a)/max(a,b)
  }
  sa = mean(unlist(s))
}

#R-square function

rsf <- function(st,np){

  npl = nrow(np)
  xx = rep(0,npl)
  np = cbind(np,xx)
  nst = length(unique(st))
  np[,3] = st
  np = np[order(np[,3]),]
  newnp = NULL
```

```
renp = np
cnum = rep(0,nst)

for (i in 1:nst) {
  cnum[i] = sum(st==i)
  pp = 1:cnum[i]
  newnp[[i]] = renp[pp,]
  renp = renp[-pp,]

}

aa = np[,1:2]
dm <- as.matrix(dist(aa))
nr <- nrow(dm)
nr <- nrow(dm)
diag(dm) = NA
dm <- t(matrix(t(dm)[which(!is.na(dm))],nrow = (nr-1),ncol = nr))
meanxk <- apply(dm,1,mean)
mxkm <- matrix(meanxk,nrow = length(meanxk),
               ncol = length(meanxk)-1,byrow = FALSE)
tt <- (dm-mxkm)^2
sst <- sum(tt)
wdm = dm
w = rep(0,nst)
for (k in 1:nst) {
  nc = cnum[k]
  if(nc > 2){
    nc1 = nc - 1
    calm <- wdm[1:nc,1:nc1]
```

```
      meanxk2 <- apply(calm, 1, mean)
      mxkm2 <- matrix(meanxk2,nrow = length(meanxk2),
                      ncol = length(meanxk2)-1,byrow = FALSE)
      ww <- (calm - mxkm2)^2
      w[k] = sum(ww)
    }else{
      w[k] = 0
    }
    if(k==nst){

    }else{
      wdm = wdm[-(1:nc),-(1:nc)]
    }
  }

  ssw <- sum(w)
  rs = (sst-ssw)/sst
  rs

}


#Hubert Gamma statistic

hubertgamma <- function(st,np){

  p <- cp(np)
  q <- cq(st,np)
  n = nrow(np)
```

```
  m = n*(n-1)/2
  hgamma = (1/(2*m))*sum(p*t(q))
  hgamma
}



#Create P

cp <- function(np){
  point <- np[,1:2]
  p <- as.matrix(dist(point))
  p
}

#Create Q

cq <- function(st,np){
  npl = nrow(np)
  xx = rep(0,npl)
  np = cbind(np,xx)
  nst = length(unique(st))
  np[,3] = st
  np = np[order(np[,3]),]


  newnp = NULL
  renp = np


  cnum = rep(0,nst)
  cpoint = NULL
```

```
  for (i in 1:nst) {
    cnum[i] = sum(st==i)
    pp = 1:cnum[i]
    newnp[[i]] = renp[pp,]
    renp = renp[-pp,]
    if(cnum[i] > 1){
      cpoint[[i]] = apply(newnp[[i]], 2, mean)
    }else{
      cpoint[[i]] = newnp[[i]]
    }
  }

  q <- matrix(rep(0,npl),npl,3)

  for (x in 1:npl) {
    q[x,] = cpoint[[st[x]]]
  }
  q <- q[,1:2]
  q <- as.matrix(dist(q))
  q
}

#Dunn index

dn <- function(st,np){
  npl = nrow(np)
  xx = rep(0,npl)
  np = cbind(np,xx)
```

```
nst = length(unique(st))
np[,3] = st
np = np[order(np[,3]),]
newnp = NULL
renp = np
cnum = rep(0,nst)

for (i in 1:nst) {
  cnum[i] = sum(st==i)
  pp = 1:cnum[i]
  newnp[[i]] = renp[pp,]
  renp = renp[-pp,]
}
diamall <- rep(0,nst)

for (j in 1:nst) {
  diamall[j] = max(dist(newnp[[j]]))
}

diam = max(diamall)
dall <- rep(0,nst-1)
aaa <- as.matrix(dist(np))
nst1 = nst - 1

for (k in 1:nst1) {
  c = sum(st==k)
  cc = 1:c
  dall[k] = min(aaa[cc,-cc])
  aaa <- aaa[-cc,-cc]
```

```
  }

  d = min(dall)
  dnc = d/diam
  dnc
}


#Davies-Bouldin index

DBindex <- function(st,np){
  npl = nrow(np)
  xx = rep(0,npl)
  np = cbind(np,xx)
  nst = length(unique(st))
  np[,3] = st
  np = np[order(np[,3]),]

  newnp = NULL
  renp = np

  cnum = rep(0,nst)
  A <- NULL
  for (i in 1:nst) {
    cnum[i] = sum(st==i)
    pp = 1:cnum[i]
    newnp[[i]] = renp[pp,]
    renp = renp[-pp,]
    if(cnum[i]>1){
```

```
      A[[i]] = apply(newnp[[i]], 2, mean)
    }else{
      A[[i]] = newnp[[i]]
    }
  }
  A = matrix(unlist(A),nrow = nst, byrow = TRUE)
  M = as.matrix(dist(A))

  s <- rep(0,nst)

  for (j in 1:nst) {
    newa <- rbind(A[j,],newnp[[j]])
    newa = newa[,1:2]
    t = cnum[j]
    s[j] = sum(dist(newa)[1:t])*sqrt(1/t)
  }
  d = rep(0,nst)

  for (k in 1:nst) {
    r = (s[k]+s)/M[k,]
    r = r[-k]
    d[k] = max(r)
  }

  DB = 1/nst*sum(d)
  DB
}
```

```
# Print function for Clustering
# print out different clusters separately and connect the dots
# in the clusters.

connectplot <- function(data,cl,name){
  plot(data[,1],data[,2], main = name, xlab = "",ylab = "",cex.main=1.5)
  cl = unlist(cl)
  ncl = length(unique(cl))
  npl = nrow(data)
  xx = rep(0,npl)
  data = cbind(data,xx)
  data[,3] = cl
  data = data[order(data[,3]),]

  if(data[1,3]==0){
    data[,3] = data[,3] + 1
    cl = cl + 1
  }

  newdata = NULL
  redata = data
  cnum = rep(0,ncl)

  for (i in 1:ncl) {
    cnum[i] = sum(cl==i)
    pp = 1:cnum[i]
    newdata[[i]] = redata[pp,]
    redata = redata[-pp,]
  }
```

```
  for (a in 1:ncl) {
    if(cnum[a]==1){
      mp = newdata[[a]]

    }else{
      mp = newdata[[a]]
      nm = nrow(mp)

      for (b in 1:nm) {
        for (c in 1:nm) {

          segments(mp[b,1],mp[b,2],mp[c,1],mp[c,2],col = a)

        }
      }
    }
  }
}



# Plot function(2), use only when the image is
# too cluttered for DBSCAN and OPTICS

doplot <- function(data,cl,name){
  plot(data[,1],data[,2], main = name, xlab = "",ylab = "",cex.main=1.5)
  cl = unlist(cl)
  ncl = length(unique(cl))
  npl = nrow(data)
```

```
  xx = rep(0,npl)
  data = cbind(data,xx)
  data[,3] = cl
  data = data[order(data[,3]),]

  if(data[1,3]==0){
    data[,3] = data[,3] + 1
    cl = cl + 1
  }

  newdata = NULL
  redata = data
  cnum = rep(0,ncl)

  for (i in 1:ncl) {
    cnum[i] = sum(cl==i)
    pp = 1:cnum[i]
    newdata[[i]] = redata[pp,]
    redata = redata[-pp,]
  }

  for (a in 1:ncl) {
    mp = newdata[[a]]
    points(mp[,1],mp[,2],pch = (15+a),col=a,cex=1.5)
  }

}
```

```
######################### Main part ############################



#########################################################
#
# Step 1 : Find the threshold for bottleneck subgraph a dataset.
# Step 2 : Using the threshold to create the bottleneck subgraph.
# Step 3 : Enumerating All Set-connected Clusterings by using set-connected
#          algorithm(generatePartitions) with different numbers of clusters.
# Step 4 : Using 5 clustering validation measures|Silhouette, Hubert-Gamma,
#          R-squared, the Dunn index, and the data Davies-Bouldin index|on
#          all Set-connected Clusterings to evaluate the score.
# Step 5 : Find the "Optimal" clustering for each clustering
#          validation measures.
# Step 6 : Using 4 traditional clustering method-K-means, HAC, DBSCAN,
#          OPTICS-to create clustering and use 5 clustering validation
#          measures evaluate the score.
# Step 7 : Find the "Optimal" clustering for each clustering validation
#          measures from traditional clusterings.
# Step 8 : Plot the "Optimal" clustering by Set-connected algorithm
#          and all best clusterings selected by 5 clustering validation
#          measures for Set-connected Clusterings.
# Step 9 : Plot the "Optimal" number of clusterings for which clusterings
#          selected by clustering validation measures are different
#          from the "Optimal" one.
# Step 10: Plot all clusterings selected by 5 clustering validation
#          measures of traditional clustering method.
```

```
#
#########################################################



# For data set 1


plot(g1[,1],g1[,2])


# Find the threshold for bottleneck subgraph for data set 1,
# to make sure all point can connect within the same group.
dis1 <- as.matrix(dist(g1))
diag(dis1) <- Inf
dis11 <- apply(dis1, 1, sort, decreasing=F)
d14 <- min(dis11[5,])

adjlist = NULL
num = nrow(g1)

for (i in 1:num) {
  ntivec = NULL
  for (j in 1:num) {
    dd = distance(g1[i,],g1[j,])
    dd
    if(dd!=0){
      if(dd<=d14){
        segments(g1[i,1],g1[i,2],g1[j,1],g1[j,2])
```

```
        ntivec = c(ntivec , j)
      }
    }else{}
  }
  adjlist[[i]]= ntivec
}

# set-connected algorithm to enumerating All Set-connected Clusterings,
# define the size from 3-7 for each cluster,
# and set the number from 3-4.
st13 = generatePartitions(adjlist,numBlocks = 3,
                          numConstraintLow = 3, numConstraintHigh = 7)
st14 = generatePartitions(adjlist,numBlocks = 4,
                          numConstraintLow = 3, numConstraintHigh = 7)
st15 = generatePartitions(adjlist,numBlocks = 5,
                          numConstraintLow = 3, numConstraintHigh = 7)
st16 = generatePartitions(adjlist,numBlocks = 6,
                          numConstraintLow = 3, numConstraintHigh = 7)


# data 1 cluster 3
# evaluate all set-connected clustering by using
# 5 clustering validation measures
n13 = length(st13)
ss13 = rep(0,n13)
hb13 = rep(0,n13)
rq13 = rep(0,n13)
df13 = rep(0,n13)
db13 = rep(0,n13)
```

```
for (i in 1:n13) {
  stt = st13[[i]]
  ss13[i] = silh(stt,g1)
  hb13[i] = hubertgamma(stt,g1)
  rq13[i] = rsf(stt,g1)
  df13[i] = dn(stt,g1)
  db13[i] = DBindex(stt,g1)
}

mssc13 = which.max(ss13)#2
mhbc13 = which.max(hb13)#2
mrqc13 = which.max(rq13)#2
mdfc13 = which.max(df13)#2
mdbc13 = which.min(db13)#45

# data 1 cluster 4

n14 = length(st14)
ss14 = rep(0,n14)
hb14 = rep(0,n14)
rq14 = rep(0,n14)
df14 = rep(0,n14)
db14 = rep(0,n14)

for (i in 1:n14) {
  stt = st14[[i]]
  ss14[i] = silh(stt,g1)
  hb14[i] = hubertgamma(stt,g1)
```

```
  rq14[i] = rsf(stt,g1)
  df14[i] = dn(stt,g1)
  db14[i] = DBindex(stt,g1)
}

mssc14 = which.max(ss14)#1757
mhbc14 = which.max(hb14)#1757
mrqc14 = which.max(rq14)#1757
mdfc14 = which.max(df14)#1757
mdbc14 = which.min(db14)#1757



# data 1 cluster 5

n15 = length(st15)
ss15 = rep(0,n15)
hb15 = rep(0,n15)
rq15 = rep(0,n15)
df15 = rep(0,n15)
db15 = rep(0,n15)

for (i in 1:n15) {
  stt = st15[[i]]
  ss15[i] = silh(stt,g1)
  hb15[i] = hubertgamma(stt,g1)
  rq15[i] = rsf(stt,g1)
  df15[i] = dn(stt,g1)
  db15[i] = DBindex(stt,g1)
}
```

```
mssc15 = which.max(ss15)#1170
mhbc15 = which.max(hb15)#1170
mrqc15 = which.max(rq15)#1170
mdfc15 = which.max(df15)#39
mdbc15 = which.min(db15)#4840


# data 1 cluster 6

n16 = length(st16)
ss16 = rep(0,n16)
hb16 = rep(0,n16)
rq16 = rep(0,n16)
df16 = rep(0,n16)
db16 = rep(0,n16)

for (i in 1:n16) {
  stt = st16[[i]]
  ss16[i] = silh(stt,g1)
  hb16[i] = hubertgamma(stt,g1)
  rq16[i] = rsf(stt,g1)
  df16[i] = dn(stt,g1)
  db16[i] = DBindex(stt,g1)
}

mssc16 = which.max(ss16)#688
mhbc16 = which.max(hb16)#688
mrqc16 = which.max(rq16)#688
```

```
mdfc16 = which.max(df16)#2
mdbc16 = which.min(db16)#900


# List all "optimal" scores for each number of clusters,
# and then compare all these scores to find which clustering
# has the optimal score in its own measure.

vs13 = max(ss13)
vs14 = max(ss14)
vs15 = max(ss15)
vs16 = max(ss16)

vs1 = c(vs13,vs14,vs15,vs16)
max(vs1)
which.max(vs1)

vh13 = max(hb13)
vh14 = max(hb14)
vh15 = max(hb15)
vh16 = max(hb16)

vh1 = c(vh13,vh14,vh15,vh16)
max(vh1)
which.max(vh1)

vr13 = max(rq13)
vr14 = max(rq14)
vr15 = max(rq15)
```

```
vr16 = max(rq16)


vr1 = c(vr13,vr14,vr15,vr16)
max(vr1)
which.max(vr1)


vf13 = max(df13)
vf14 = max(df14)
vf15 = max(df15)
vf16 = max(df16)


vf1 = c(vf13,vf14,vf15,vf16)
max(vf1)
which.max(vf1)


vb13 = min(db13)
vb14 = min(db14)
vb15 = min(db15)
vb16 = min(db16)


vb1 = c(vb13,vb14,vb15,vb16)
min(vb1)
which.min(vb1)



# K-means also try 3-4 clusters and evaluate all clusterings
# by 5 clustering validation measures.
# And compare the scores with optimal set-connected clusterings.
km13 <- kmeans(g1,3)$cluster
```

```
km14 <- kmeans(g1,4)$cluster
km15 <- kmeans(g1,5)$cluster
km16 <- kmeans(g1,6)$cluster

ss13k = silh(km13,g1)
hb13k = hubertgamma(km13,g1)
rq13k = rsf(km13,g1)
df13k = dn(km13,g1)
db13k = DBindex(km13,g1)

ss14k = silh(km14,g1)
hb14k = hubertgamma(km14,g1)
rq14k = rsf(km14,g1)
df14k = dn(km14,g1)
db14k = DBindex(km14,g1)

ss15k = silh(km15,g1)
hb15k = hubertgamma(km15,g1)
rq15k = rsf(km15,g1)
df15k = dn(km15,g1)
db15k = DBindex(km15,g1)

ss16k = silh(km16,g1)
hb16k = hubertgamma(km16,g1)
rq16k = rsf(km16,g1)
df16k = dn(km16,g1)
db16k = DBindex(km16,g1)

vs1k = c(ss13k,ss14k,ss15k,ss16k)
```

```
max(vs1k)
which.max(vs1k)

vh1k = c(hb13k,hb14k,hb15k,hb16k)
max(vh1k)
which.max(vh1k)

vr1k = c(rq13k,rq14k,rq15k,rq16k)
max(vr1k)
which.max(vr1k)

vf1k = c(df13k,df14k,df15k,df16k)
max(vf1k)
which.max(vf1k)

vd1k = c(db13k,db14k,db15k,db16k)
min(vd1k)
which.min(vd1k)

# HAC also try 3-4 clusters and evaluate all clusterings
# by 5 clustering validation measures.
# And compare the scores with optimal set-connected clusterings.
hc1 <- hclust(d=dist(g1))
hc13 <- cutree(hc1, 3)
hc14 <- cutree(hc1, 4)
hc15 <- cutree(hc1, 5)
hc16 <- cutree(hc1, 6)

ss13h = silh(hc13,g1)
```

```
hb13h = hubertgamma(hc13,g1)
rq13h = rsf(hc13,g1)
df13h = dn(hc13,g1)
db13h = DBindex(hc13,g1)

ss14h = silh(hc14,g1)
hb14h = hubertgamma(hc14,g1)
rq14h = rsf(hc14,g1)
df14h = dn(hc14,g1)
db14h = DBindex(hc14,g1)

ss15h = silh(hc15,g1)
hb15h = hubertgamma(hc15,g1)
rq15h = rsf(hc15,g1)
df15h = dn(hc15,g1)
db15h = DBindex(hc15,g1)

ss16h = silh(hc16,g1)
hb16h = hubertgamma(hc16,g1)
rq16h = rsf(hc16,g1)
df16h = dn(hc16,g1)
db16h = DBindex(hc16,g1)

vs1h = c(ss13h,ss14h,ss15h,ss16h)
max(vs1h)
which.max(vs1h)

vh1h = c(hb13h,hb14h,hb15h,hb16h)
max(vh1h)
```

```
which.max(vh1h)

vr1h = c(rq13h,rq14h,rq15h,rq16h)
max(vr1h)
which.max(vr1h)

vf1h = c(df13h,df14h,df15h,df16h)
max(vf1h)
which.max(vf1h)

vd1h = c(db13h,db14h,db15h,db16h)
min(vd1h)
which.min(vd1h)

# DBSCAN
db1 <- dbscan(g1,eps=0.5,MinPts = 3)$cluster
ss1d = silh(db1,g1)#0.6207152
hb1d = hubertgamma(db1,g1)#4.28947
rq1d = rsf(db1,g1)#0.9922129
df1d = dn(db1,g1)#1.414214
db1d = DBindex(db1,g1)#0.8

# OPTICS
op10 <- optics(g1,eps = 3,minPts = 3)
op1 <- extractDBSCAN(op10,eps_cl = 0.5)$cluster
op1 = op1 + 1
ss1o = silh(op1,g1)#0.3743962
hb1o = hubertgamma(op1,g1)#3.852513
rq1o = rsf(op1,g1)#0.9854472
```

```
df1o = dn(op1,g1)#0.3952847
db1o = DBindex(op1,g1)#1.477132




# Plot all datasets
pdf("data.pdf",width = 8, height = 12,colormodel='cmyk')
par(mfrow=c(3,2))
plot(g1[,1],g1[,2],main = "Data 1",xlab = "",ylab = "",
     pch = 19,cex = 1.5,cex.main=2)
plot(g2[,1],g2[,2],main = "Data 2",xlab = "",ylab = "",
     pch = 19,cex = 1.5,cex.main=2)
plot(g3[,1],g3[,2],main = "Data 3",xlab = "",ylab = "",
     pch = 19,cex = 1.5,cex.main=2)
plot(g4[,1],g4[,2],main = "Data 4",xlab = "",ylab = "",
     pch = 19,cex = 1.5,cex.main=2)
plot(g5[,1],g5[,2],main = "Data 5",xlab = "",ylab = "",
     pch = 19,cex = 1.5,cex.main=2)
dev.off()




# Plot dataset1
pdf("data1result.pdf",width = 12, height = 16,colormodel='cmyk')
par(mfrow=c(4,3))
connectplot(g1,st14[mssc14],"Result by Silhouette(4)")
connectplot(g1,st14[mhbc14],"Result by Hubert gamma(4)")
```

```
connectplot(g1,st16[mrqc16],"Result by R-square(6)")
connectplot(g1,st14[mdfc14],"Result by Dunn index(4)")
connectplot(g1,st14[mdbc14],"Result by DB index(4)")
connectplot(g1,km14,"Result by Kmeans(4)")
connectplot(g1,km16,"Result by Kmeans(6)")
connectplot(g1,hc14,"Result by Hierarchical(4)")
connectplot(g1,hc16,"Result by Hierarchical(6)")
connectplot(g1,db1,"Result by DBSCAN(4)")
connectplot(g1,op1,"Result by OPTICS(5)")
dev.off()
```

```
pdf("data1right.pdf",width = 8, height = 12,colormodel='cmyk')
par(mfrow=c(3,2))
connectplot(g1,st14[mssc14],"Result by Silhouette with 4 clusters")
connectplot(g1,st14[mhbc14],"Result by Hubert gamma with 4 clusters")
connectplot(g1,st14[mrqc14],"Result by R-square with 4 clusters")
connectplot(g1,st14[mdfc14],"Result by Dunn index with 4 clusters")
connectplot(g1,st14[mdbc14],"Result by DB index with 4 clusters")
dev.off()
```

```
# For data set 2


plot(g2[,1],g2[,2])


dis2 <- as.matrix(dist(g2))
diag(dis2) <- Inf
dis21 <- apply(dis2, 1, sort, decreasing=F)
d2 <- min(dis21[7,])



adjlist2 = NULL
num2 = nrow(g2)

for (i in 1:num2) {
  ntivec = NULL
  for (j in 1:num2) {
    dd = distance(g2[i,],g2[j,])
    dd
    if(dd!=0){
      if(dd<=d2){
        segments(g2[i,1],g2[i,2],g2[j,1],g2[j,2])
        ntivec = c(ntivec , j)
      }
    }else{}
  }
  adjlist2[[i]]= ntivec
}
```

```
st23 = generatePartitions(adjlist2,numBlocks = 3,
                          numConstraintLow = 3, numConstraintHigh = 10)
st24 = generatePartitions(adjlist2,numBlocks = 4,
                          numConstraintLow = 3, numConstraintHigh = 10)
st25 = generatePartitions(adjlist2,numBlocks = 5,
                          numConstraintLow = 3, numConstraintHigh = 7)
st26 = generatePartitions(adjlist2,numBlocks = 6,
                          numConstraintLow = 3, numConstraintHigh = 7)



# data 2 cluster 3

n23 = length(st23)
ss23 = rep(0,n23)
hb23 = rep(0,n23)
rq23 = rep(0,n23)
df23 = rep(0,n23)
db23 = rep(0,n23)

for (i in 1:n23) {
  stt = st23[[i]]
  ss23[i] = silh(stt,g2)
  hb23[i] = hubertgamma(stt,g2)
  rq23[i] = rsf(stt,g2)
  df23[i] = dn(stt,g2)
  db23[i] = DBindex(stt,g2)
}

mssc23 = which.max(ss23)#252
```

```
mhbc23 = which.max(hb23)#252
mrqc23 = which.max(rq23)#105
mdfc23 = which.max(df23)#451
mdbc23 = which.min(db23)#451


# data 2 cluster 4

n24 = length(st24)
ss24 = rep(0,n24)
hb24 = rep(0,n24)
rq24 = rep(0,n24)
df24 = rep(0,n24)
db24 = rep(0,n24)

for (i in 1:n24) {
  stt = st24[[i]]
  ss24[i] = silh(stt,g2)
  hb24[i] = hubertgamma(stt,g2)
  rq24[i] = rsf(stt,g2)
  df24[i] = dn(stt,g2)
  db24[i] = DBindex(stt,g2)
}

mssc24 = which.max(ss24)#2594
mhbc24 = which.max(hb24)#2594
mrqc24 = which.max(rq24)#2071
mdfc24 = which.max(df24)#2070
mdbc24 = which.min(db24)#2070
```

```
# data 2 cluster 5

n25 = length(st25)
ss25 = rep(0,n25)
hb25 = rep(0,n25)
rq25 = rep(0,n25)
df25 = rep(0,n25)
db25 = rep(0,n25)

for (i in 1:n25) {
  stt = st25[[i]]
  ss25[i] = silh(stt,g2)
  hb25[i] = hubertgamma(stt,g2)
  rq25[i] = rsf(stt,g2)
  df25[i] = dn(stt,g2)
  db25[i] = DBindex(stt,g2)
}

mssc25 = which.max(ss25)#7812
mhbc25 = which.max(hb25)#7812
mrqc25 = which.max(rq25)#7812
mdfc25 = which.max(df25)#7812
mdbc25 = which.min(db25)#7812

# data 2 cluster 6

n26 = length(st26)
ss26 = rep(0,n26)
```

```
hb26 = rep(0,n26)
rq26 = rep(0,n26)
df26 = rep(0,n26)
db26 = rep(0,n26)

for (i in 1:n26) {
  stt = st26[[i]]
  ss26[i] = silh(stt,g2)
  hb26[i] = hubertgamma(stt,g2)
  rq26[i] = rsf(stt,g2)
  df26[i] = dn(stt,g2)
  db26[i] = DBindex(stt,g2)
}

mssc26 = which.max(ss26)#27024
mhbc26 = which.max(hb26)#27024
mrqc26 = which.max(rq26)#27025
mdfc26 = which.max(df26)#27027
mdbc26 = which.min(db26)#27024

# Find the best score for each measure.

vs23 = max(ss23)
vs24 = max(ss24)
vs25 = max(ss25)
vs26 = max(ss26)

vs2 = c(vs23,vs24,vs25,vs26)
max(vs2)
```

```
which.max(vs2)


vh23 = max(hb23)
vh24 = max(hb24)
vh25 = max(hb25)
vh26 = max(hb26)

vh2 = c(vh23,vh24,vh25,vh26)
max(vh2)
which.max(vh2)

vr23 = max(rq23)
vr24 = max(rq24)
vr25 = max(rq25)
vr26 = max(rq26)

vr2 = c(vr23,vr24,vr25,vr26)
max(vr2)
which.max(vr2)

vf23 = max(df23)
vf24 = max(df24)
vf25 = max(df25)
vf26 = max(df26)

vf2 = c(vf23,vf24,vf25,vf26)
max(vf2)
which.max(vf2)
```

```
vb23 = min(db23)
vb24 = min(db24)
vb25 = min(db25)
vb26 = min(db26)

vb2 = c(vb23,vb24,vb25,vb26)
min(vb2)
which.min(vb2)

# K-means
km23 <- kmeans(g2,3)$cluster
km24 <- kmeans(g2,4)$cluster
km25 <- kmeans(g2,5)$cluster
km26 <- kmeans(g2,6)$cluster

ss23k = silh(km23,g2)
hb23k = hubertgamma(km23,g2)
rq23k = rsf(km23,g2)
df23k = dn(km23,g2)
db23k = DBindex(km23,g2)

ss24k = silh(km24,g2)
hb24k = hubertgamma(km24,g2)
rq24k = rsf(km24,g2)
df24k = dn(km24,g2)
db24k = DBindex(km24,g2)

ss25k = silh(km25,g2)
hb25k = hubertgamma(km25,g2)
```

```
rq25k = rsf(km25,g2)
df25k = dn(km25,g2)
db25k = DBindex(km25,g2)

ss26k = silh(km26,g2)
hb26k = hubertgamma(km26,g2)
rq26k = rsf(km26,g2)
df26k = dn(km26,g2)
db26k = DBindex(km26,g2)

vs2k = c(ss23k,ss24k,ss25k,ss26k)
max(vs2k)
which.max(vs2k)

vh2k = c(hb23k,hb24k,hb25k,hb26k)
max(vh2k)
which.max(vh2k)

vr2k = c(rq23k,rq24k,rq25k,rq26k)
max(vr2k)
which.max(vr2k)

vf2k = c(df23k,df24k,df25k,df26k)
max(vf2k)
which.max(vf2k)

vd2k = c(db23k,db24k,db25k,db26k)
min(vd2k)
which.min(vd2k)
```

```
# HAC
hc2 <- hclust(d=dist(g2))
hc23 <- cutree(hc2, 3)
hc24 <- cutree(hc2, 4)
hc25 <- cutree(hc2, 5)
hc26 <- cutree(hc2, 6)

ss23h = silh(hc23,g2)
hb23h = hubertgamma(hc23,g2)
rq23h = rsf(hc23,g2)
df23h = dn(hc23,g2)
db23h = DBindex(hc23,g2)

ss24h = silh(hc24,g2)
hb24h = hubertgamma(hc24,g2)
rq24h = rsf(hc24,g2)
df24h = dn(hc24,g2)
db24h = DBindex(hc24,g2)

ss25h = silh(hc25,g2)
hb25h = hubertgamma(hc25,g2)
rq25h = rsf(hc25,g2)
df25h = dn(hc25,g2)
db25h = DBindex(hc25,g2)

ss26h = silh(hc26,g2)
hb26h = hubertgamma(hc26,g2)
rq26h = rsf(hc26,g2)
```

```
df26h = dn(hc26,g2)
db26h = DBindex(hc26,g2)

vs2h = c(ss23h,ss24h,ss25h,ss26h)
max(vs2h)
which.max(vs2h)

vh2h = c(hb23h,hb24h,hb25h,hb26h)
max(vh2h)
which.max(vh2h)

vr2h = c(rq23h,rq24h,rq25h,rq26h)
max(vr2h)
which.max(vr2h)

vf2h = c(df23h,df24h,df25h,df26h)
max(vf2h)
which.max(vf2h)

vd2h = c(db23h,db24h,db25h,db26h)
min(vd2h)
which.min(vd2h)

# DBSCAN
db2 <- dbscan(g2,eps=0.5,MinPts = 3)$cluster
db2 = db2 + 1
ss2d = silh(db2,g2)
hb2d = hubertgamma(db2,g2)
rq2d = rsf(db2,g2)
```

```
df2d = dn(db2,g2)
db2d = DBindex(db2,g2)

# OPTICS
op20 <- optics(g2,eps = 3,minPts = 3)
op2 <- extractDBSCAN(op20,eps_cl = 0.5)$cluster
op2 = op2 + 1
ss2o = silh(op2,g2)
hb2o = hubertgamma(op2,g2)
rq2o = rsf(op2,g2)
df2o = dn(op2,g2)
db2o = DBindex(op2,g2)




pdf("data2result.pdf",width = 12, height = 16,colormodel='cmyk')
par(mfrow=c(4,3))
connectplot(g2,st25[mssc25],"Result by Silhouette(5)")
connectplot(g2,st26[mhbc26],"Result by Hubert gamma(6)")
connectplot(g2,st26[mrqc26],"Result by R-square(6)")
connectplot(g2,st25[mdfc25],"Result by Dunn index(5)")
connectplot(g2,st26[mdbc26],"Result by DB index(6)")
connectplot(g2,km25,"Result by Kmeans(5)")
connectplot(g2,km26,"Result by Kmeans(6)")
connectplot(g2,hc25,"Result by Hierarchical(5)")
connectplot(g2,hc26,"Result by Hierarchical(6)")
connectplot(g2,db2,"Result by DBSCAN(5)")
```

```
connectplot(g2,op2,"Result by OPTICS(5)")
dev.off()


pdf("data2right.pdf",width = 8, height = 12,colormodel='cmyk')
par(mfrow=c(3,2))
connectplot(g2,st25[mssc25],"Result by Silhouette with 5 clusters")
connectplot(g2,st25[mhbc25],"Result by Hubert gamma with 5 clusters")
connectplot(g2,st25[mrqc25],"Result by R-square with 5 clusters")
connectplot(g2,st25[mdfc25],"Result by Dunn index with 5 clusters")
connectplot(g2,st25[mdbc25],"Result by DB index with 5 clusters")
dev.off()




# For data set 3
plot(g3[,1],g3[,2])

dis3 <- as.matrix(dist(g3))
diag(dis3) <- Inf
dis31 <- apply(dis3, 1, sort, decreasing=F)
d3 <- min(dis31[6,])


adjlist3 = NULL
num3 = nrow(g3)

for (i in 1:num3) {
```

```
  ntivec = NULL
  for (j in 1:num3) {
    dd = distance(g3[i,],g3[j,])
    dd
    if(dd!=0){
      if(dd<=d3){
        segments(g3[i,1],g3[i,2],g3[j,1],g3[j,2])
        ntivec = c(ntivec , j)
      }
    }else{}
  }
  adjlist3[[i]]= ntivec
}
st32 = generatePartitions(adjlist3,numBlocks = 2,
                          numConstraintLow = 4, numConstraintHigh = 12)
st33 = generatePartitions(adjlist3,numBlocks = 3,
                          numConstraintLow = 4, numConstraintHigh = 12)
st34 = generatePartitions(adjlist3,numBlocks = 4,
                          numConstraintLow = 4, numConstraintHigh = 12)



# data 3 cluster 2


n32 = length(st32)
ss32 = rep(0,n32)
hb32 = rep(0,n32)
rq32 = rep(0,n32)
df32 = rep(0,n32)
db32 = rep(0,n32)
```

```
for (i in 1:n32) {
  stt = st32[[i]]
  ss32[i] = silh(stt,g3)
  hb32[i] = hubertgamma(stt,g3)
  rq32[i] = rsf(stt,g3)
  df32[i] = dn(stt,g3)
  db32[i] = DBindex(stt,g3)
}

mssc32 = which.max(ss32)#93
mhbc32 = which.max(hb32)#93
mrqc32 = which.max(rq32)#93
mdfc32 = which.max(df32)#93
mdbc32 = which.min(db32)#93

# data 3 cluster 3

n33 = length(st33)
ss33 = rep(0,n33)
hb33 = rep(0,n33)
rq33 = rep(0,n33)
df33 = rep(0,n33)
db33 = rep(0,n33)

for (i in 1:n33) {
  stt = st33[[i]]
  ss33[i] = silh(stt,g3)
  hb33[i] = hubertgamma(stt,g3)
```

```
  rq33[i] = rsf(stt,g3)
  df33[i] = dn(stt,g3)
  db33[i] = DBindex(stt,g3)
}

mssc33 = which.max(ss33)#1736
mhbc33 = which.max(hb33)#1400
mrqc33 = which.max(rq33)#1400
mdfc33 = which.max(df33)#1400
mdbc33 = which.min(db33)#1097

# data 3 cluster 4


n34 = length(st34)
ss34 = rep(0,n34)
hb34 = rep(0,n34)
rq34 = rep(0,n34)
df34 = rep(0,n34)
db34 = rep(0,n34)

for (i in 1:n34) {
  stt = st34[[i]]
  ss34[i] = silh(stt,g3)
  hb34[i] = hubertgamma(stt,g3)
  rq34[i] = rsf(stt,g3)
  df34[i] = dn(stt,g3)
  db34[i] = DBindex(stt,g3)
}
```

```
mssc34 = which.max(ss34)#3310
mhbc34 = which.max(hb34)#2626
mrqc34 = which.max(rq34)#2710
mdfc34 = which.max(df34)#2710
mdbc34 = which.min(db34)#2612

# Find the best score for each measure.
vs32 = max(ss32)
vs33 = max(ss33)
vs34 = max(ss34)

vs3 = c(vs32,vs33,vs34)
max(vs3)
which.max(vs3)

vh32 = max(hb32)
vh33 = max(hb33)
vh34 = max(hb34)

vh3 = c(vh32,vh33,vh34)
max(vh3)
which.max(vh3)

vr32 = max(rq32)
vr33 = max(rq33)
vr34 = max(rq34)

vr3 = c(vr32,vr33,vr34)
```

```
max(vr3)
which.max(vr3)

vf32 = max(df32)
vf33 = max(df33)
vf34 = max(df34)

vf3 = c(vf32,vf33,vf34)
max(vf3)
which.max(vf3)

vd32 = min(db32)
vd33 = min(db33)
vd34 = min(db34)

vd3 = c(vd32,vd33,vd34)
min(vd3)
which.min(vd3)



# K-means

km32 <- kmeans(g3,2)$cluster
km33 <- kmeans(g3,3)$cluster
km34 <- kmeans(g3,4)$cluster

ss32k = silh(km32,g3)
hb32k = hubertgamma(km32,g3)
rq32k = rsf(km32,g3)
```

```
df32k = dn(km32,g3)
db32k = DBindex(km32,g3)

ss33k = silh(km33,g3)
hb33k = hubertgamma(km33,g3)
rq33k = rsf(km33,g3)
df33k = dn(km33,g3)
db33k = DBindex(km33,g3)

ss34k = silh(km34,g3)
hb34k = hubertgamma(km34,g3)
rq34k = rsf(km34,g3)
df34k = dn(km34,g3)
db34k = DBindex(km34,g3)

vs3k = c(ss32k,ss33k,ss34k)
max(vs3k)
which.max(vs3k)

vh3k = c(hb32k,hb33k,hb34k)
max(vh3k)
which.max(vh3k)

vr3k = c(rq32k,rq33k,rq34k)
max(vr3k)
which.max(vr3k)

vf3k = c(df32k,df33k,df34k)
max(vf3k)
```

```
which.max(vf3k)

vd3k = c(db32k,db33k,db34k)
min(vd3k)
which.min(vd3k)

# HAC
hc3 <- hclust(d=dist(g3))
hc32 <- cutree(hc3, 2)
hc33 <- cutree(hc3, 3)
hc34 <- cutree(hc3, 4)

ss32h = silh(hc32,g3)
hb32h = hubertgamma(hc32,g3)
rq32h = rsf(hc32,g3)
df32h = dn(hc32,g3)
db32h = DBindex(hc32,g3)

ss33h = silh(hc33,g3)
hb33h = hubertgamma(hc33,g3)
rq33h = rsf(hc33,g3)
df33h = dn(hc33,g3)
db33h = DBindex(hc33,g3)

ss34h = silh(hc34,g3)
hb34h = hubertgamma(hc34,g3)
rq34h = rsf(hc34,g3)
df34h = dn(hc34,g3)
db34h = DBindex(hc34,g3)
```

```
vs3h = c(ss32h,ss33h,ss34h)
max(vs3h)
which.max(vs3h)

vh3h = c(hb32h,hb33h,hb34h)
max(vh3h)
which.max(vh3h)

vr3h = c(rq32h,rq33h,rq34h)
max(vr3h)
which.max(vr3h)

vf3h = c(df32h,df33h,df34h)
max(vf3h)
which.max(vf3h)

vd3h = c(db32h,db33h,db34h)
min(vd3h)
which.min(vd3h)

# DBSCAN
db3o <- fpc::dbscan(g3,eps=0.5,MinPts = 4)
db3 <- db3o$cluster
db3 = db3 + 1
ss3d = silh(db3,g3)
hb3d = hubertgamma(db3,g3)
rq3d = rsf(db3,g3)
df3d = dn(db3,g3)
```

```
db3d = DBindex(db3,g3)

# OPTICS
op30 <- optics(g3,eps = 4,minPts = 4)
op3o <- extractDBSCAN(op30,eps_cl = 0.5)
op3 <- op3o$cluster
op3 = op3 + 1
ss3o = silh(op3,g3)
hb3o = hubertgamma(op3,g3)
rq3o = rsf(op3,g3)
df3o = dn(op3,g3)
db3o = DBindex(op3,g3)



pdf("data3result.pdf",width = 12, height = 16,colormodel='cmyk')
par(mfrow=c(4,3))
connectplot(g3,st32[mssc32],"Result by Silhouette(2)")
connectplot(g3,st34[mhbc34],"Result by Hubert gamma(4)")
connectplot(g3,st34[mrqc34],"Result by R-square(4)")
connectplot(g3,st34[mdfc34],"Result by Dunn index(4)")
connectplot(g3,st34[mdbc34],"Result by DB index(4)")
connectplot(g3,km32,"Result by Kmeans(2)")
connectplot(g3,km34,"Result by Kmeans(4)")
connectplot(g3,hc32,"Result by Hierarchical(2)")
connectplot(g3,hc34,"Result by Hierarchical(4)")
doplot(g3,db3,"Result by DBSCAN(3)")
doplot(g3,op3,"Result by OPTICS(3)")
dev.off()
```

```
pdf("data3right.pdf",width = 8, height = 12,colormodel='cmyk')
par(mfrow=c(3,2))
connectplot(g3,st32[mssc32],"Result by Silhouette with 2 clusters")
connectplot(g3,st32[mhbc32],"Result by Hubert gamma with 2 clusters")
connectplot(g3,st32[mrqc32],"Result by R-square with 2 clusters")
connectplot(g3,st32[mdfc32],"Result by Dunn index with 2 clusters")
connectplot(g3,st32[mdbc32],"Result by DB index with 2 clusters")
dev.off()




# For data set 4

plot(g4[,1],g4[,2])

dis4 <- as.matrix(dist(g4))
diag(dis4) <- Inf
dis41 <- apply(dis4, 1, sort, decreasing=F)
d4 <- min(dis41[5,])

adjlist4 = NULL
num4 = nrow(g4)

for (i in 1:num4) {
  ntivec = NULL
  for (j in 1:num4) {
    dd = distance(g4[i,],g4[j,])
```

```
    dd
    if(dd!=0){
      if(dd<=d4){
        segments(g4[i,1],g4[i,2],g4[j,1],g4[j,2])
        ntivec = c(ntivec , j)
      }
    }else{}
  }
  adjlist4[[i]]= ntivec
}

st42 = generatePartitions(adjlist4,numBlocks = 2,
                          numConstraintLow = 4, numConstraintHigh = 10)
st43 = generatePartitions(adjlist4,numBlocks = 3,
                          numConstraintLow = 4, numConstraintHigh = 10)
st44 = generatePartitions(adjlist4,numBlocks = 4,
                          numConstraintLow = 4, numConstraintHigh = 10)



# data 4 cluster 2

n42 = length(st42)
ss42 = rep(0,n42)
hb42 = rep(0,n42)
rq42 = rep(0,n42)
df42 = rep(0,n42)
db42 = rep(0,n42)

for (i in 1:n42) {
```

```
  stt = st42[[i]]
  ss42[i] = silh(stt,g4)
  hb42[i] = hubertgamma(stt,g4)
  rq42[i] = rsf(stt,g4)
  df42[i] = dn(stt,g4)
  db42[i] = DBindex(stt,g4)
}

mssc42 = which.max(ss42)#302
mhbc42 = which.max(hb42)#302
mrqc42 = which.max(rq42)#302
mdfc42 = which.max(df42)#302
mdbc42 = which.min(db42)#302

# data 4 cluster 3

n43 = length(st43)
ss43 = rep(0,n43)
hb43 = rep(0,n43)
rq43 = rep(0,n43)
df43 = rep(0,n43)
db43 = rep(0,n43)

for (i in 1:n43) {
  stt = st43[[i]]
  ss43[i] = silh(stt,g4)
  hb43[i] = hubertgamma(stt,g4)
  rq43[i] = rsf(stt,g4)
  df43[i] = dn(stt,g4)
```

```
  db43[i] = DBindex(stt,g4)
}

mssc43 = which.max(ss43)#1
mhbc43 = which.max(hb43)#1
mrqc43 = which.max(rq43)#1219
mdfc43 = which.max(df43)#1
mdbc43 = which.min(db43)#1



# data 4 cluster 4

n44 = length(st44)
ss44 = rep(0,n44)
hb44 = rep(0,n44)
rq44 = rep(0,n44)
df44 = rep(0,n44)
db44 = rep(0,n44)

for (i in 1:n44) {
  stt = st44[[i]]
  ss44[i] = silh(stt,g4)
  hb44[i] = hubertgamma(stt,g4)
  rq44[i] = rsf(stt,g4)
  df44[i] = dn(stt,g4)
  db44[i] = DBindex(stt,g4)
}

mssc44 = which.max(ss44)#4
```

```
mhbc44 = which.max(hb44)#25
mrqc44 = which.max(rq44)#25
mdfc44 = which.max(df44)#93
mdbc44 = which.min(db44)#4

# Find the best score for each measure.

vs42 = max(ss42)
vs43 = max(ss43)
vs44 = max(ss44)

vs4 = c(vs42,vs43,vs44)
max(vs4)
which.max(vs4)

vh42 = max(hb42)
vh43 = max(hb43)
vh44 = max(hb44)

vh4 = c(vh42,vh43,vh44)
max(vh4)
which.max(vh4)

vr42 = max(rq42)
vr43 = max(rq43)
vr44 = max(rq44)

vr4 = c(vr42,vr43,vr44)
max(vr4)
```

```
which.max(vr4)

vf42 = max(df42)
vf43 = max(df43)
vf44 = max(df44)

vf4 = c(vf42,vf43,vf44)
max(vf4)
which.max(vf4)

vd42 = min(db42)
vd43 = min(db43)
vd44 = min(db44)

vd4 = c(vd42,vd43,vd44)
min(vd4)
which.min(vd4)

# K-means
km42 <- kmeans(g4,2)$cluster
km43 <- kmeans(g4,3)$cluster
km44 <- kmeans(g4,4)$cluster

ss42k = silh(km42,g4)
hb42k = hubertgamma(km42,g4)
rq42k = rsf(km42,g4)
df42k = dn(km42,g4)
db42k = DBindex(km42,g4)
```

```
ss43k = silh(km43,g4)
hb43k = hubertgamma(km43,g4)
rq43k = rsf(km43,g4)
df43k = dn(km43,g4)
db43k = DBindex(km43,g4)

ss44k = silh(km44,g4)
hb44k = hubertgamma(km44,g4)
rq44k = rsf(km44,g4)
df44k = dn(km44,g4)
db44k = DBindex(km44,g4)

vs4k = c(ss42k,ss43k,ss44k)
max(vs4k)
which.max(vs4k)

vh4k = c(hb42k,hb43k,hb44k)
max(vh4k)
which.max(vh4k)

vr4k = c(rq42k,rq43k,rq44k)
max(vr4k)
which.max(vr4k)

vf4k = c(df42k,df43k,df44k)
max(vf4k)
which.max(vf4k)

vd4k = c(db42k,db43k,db44k)
```

```
min(vd4k)
which.min(vd4k)

# HAC
hc4 <- hclust(d=dist(g4))
hc42 <- cutree(hc4, 2)
hc43 <- cutree(hc4, 3)
hc44 <- cutree(hc4, 4)

ss42h = silh(hc42,g4)
hb42h = hubertgamma(hc42,g4)
rq42h = rsf(hc42,g4)
df42h = dn(hc42,g4)
db42h = DBindex(hc42,g4)

ss43h = silh(hc43,g4)
hb43h = hubertgamma(hc43,g4)
rq43h = rsf(hc43,g4)
df43h = dn(hc43,g4)
db43h = DBindex(hc43,g4)

ss44h = silh(hc44,g4)
hb44h = hubertgamma(hc44,g4)
rq44h = rsf(hc44,g4)
df44h = dn(hc44,g4)
db44h = DBindex(hc44,g4)

vs4h = c(ss42h,ss43h,ss44h)
max(vs4h)
```

```
which.max(vs4h)

vh4h = c(hb42h,hb43h,hb44h)
max(vh4h)
which.max(vh4h)

vr4h = c(rq42h,rq43h,rq44h)
max(vr4h)
which.max(vr4h)

vf4h = c(df42h,df43h,df44h)
max(vf4h)
which.max(vf4h)

vd4h = c(db42h,db43h,db44h)
min(vd4h)
which.min(vd4h)



# DBSCAN
db4 <- dbscan(g4,eps=1.4,MinPts = 4)$cluster
ss4d = silh(db4,g4)
hb4d = hubertgamma(db4,g4)
rq4d = rsf(db4,g4)
df4d = dn(db4,g4)
db4d = DBindex(db4,g4)

# OPTICS
op40 <- optics(g4,eps = 4,minPts = 4)
```

```
op4 <- extractDBSCAN(op40,eps_cl = 1)$cluster
op4 = op4 + 1
ss4o = silh(op4,g4)
hb4o = hubertgamma(op4,g4)
rq4o = rsf(op4,g4)
df4o = dn(op4,g4)
db4o = DBindex(op4,g4)



pdf("data4result.pdf",width = 12, height = 16,colormodel='cmyk')
par(mfrow=c(4,3))
connectplot(g4,st42[mssc42],"Result by Silhouette(2)")
connectplot(g4,st44[mhbc44],"Result by Hubert gamma(4)")
connectplot(g4,st44[mrqc44],"Result by R-square(4)")
connectplot(g4,st43[mdfc43],"Result by Dunn index(3)")
connectplot(g4,st44[mdbc44],"Result by DB index(4)")
connectplot(g4,km42,"Result by Kmeans(2)")
connectplot(g4,km44,"Result by Kmeans(4)")
connectplot(g4,hc42,"Result by Hierarchical(2)")
connectplot(g4,hc44,"Result by Hierarchical(4)")
connectplot(g4,hc43,"Result by Kmeans and Hierarchical(3)")
connectplot(g4,db4,"Result by DBSCAN(3)")
connectplot(g4,op4,"Result by OPTICS(4)")
dev.off()

pdf("data4right.pdf",width = 8, height = 12,colormodel='cmyk')
par(mfrow=c(3,2))
connectplot(g4,st43[mssc43],"Result by Silhouette with 3 clusters")
```

```
connectplot(g4,st43[mhbc43],"Result by Hubert gamma with 3 clusters")
connectplot(g4,st43[mrqc43],"Result by R-square with 3 clusters")
connectplot(g4,st43[mdfc43],"Result by Dunn index with 3 clusters")
connectplot(g4,st43[mdbc43],"Result by DB index with 3 clusters")
dev.off()
```

```
# For data set 5
plot(g5[,1],g5[,2])

dis5 <- as.matrix(dist(g5))
diag(dis5) <- Inf
dis51 <- apply(dis5, 1, sort, decreasing=F)
d5 <- min(dis51[4,])


adjlist5 = NULL
num5 = nrow(g5)

for (i in 1:num5) {
  ntivec = NULL
  for (j in 1:num5) {
    dd = distance(g5[i,],g5[j,])
    dd
    if(dd!=0){
      if(dd<=d5){
```

```
        segments(g5[i,1],g5[i,2],g5[j,1],g5[j,2])
        ntivec = c(ntivec , j)
      }
    }else{}
  }
  adjlist5[[i]]= ntivec
}

st52 = generatePartitions(adjlist5,numBlocks = 2,
                          numConstraintLow = 4, numConstraintHigh = 14)
st53 = generatePartitions(adjlist5,numBlocks = 3,
                          numConstraintLow = 4, numConstraintHigh = 10)
st54 = generatePartitions(adjlist5,numBlocks = 4,
                          numConstraintLow = 4, numConstraintHigh = 10)
st55 = generatePartitions(adjlist5,numBlocks = 5,
                          numConstraintLow = 4, numConstraintHigh = 10)


# data 5 cluster 2
n52 = length(st52)
ss52 = rep(0,n52)
hb52 = rep(0,n52)
rq52 = rep(0,n52)
df52 = rep(0,n52)
db52 = rep(0,n52)

for (i in 1:n52) {
  stt = st52[[i]]
  ss52[i] = silh(stt,g5)
```

```
  hb52[i] = hubertgamma(stt,g5)
  rq52[i] = rsf(stt,g5)
  df52[i] = dn(stt,g5)
  db52[i] = DBindex(stt,g5)
}

mssc52 = which.max(ss52)#1
mhbc52 = which.max(hb52)#1
mrqc52 = which.max(rq52)#1
mdfc52 = which.max(df52)#5
mdbc52 = which.min(db52)#1



# data 5 cluster 3

n53 = length(st53)
ss53 = rep(0,n53)
hb53 = rep(0,n53)
rq53 = rep(0,n53)
df53 = rep(0,n53)
db53 = rep(0,n53)

for (i in 1:n53) {
  stt = st53[[i]]
  ss53[i] = silh(stt,g5)
  hb53[i] = hubertgamma(stt,g5)
  rq53[i] = rsf(stt,g5)
  df53[i] = dn(stt,g5)
  db53[i] = DBindex(stt,g5)
```

```
}

mssc53 = which.max(ss53)#5
mhbc53 = which.max(hb53)#5
mrqc53 = which.max(rq53)#5
mdfc53 = which.max(df53)#5
mdbc53 = which.min(db53)#5



# data 5 cluster 4

n54 = length(st54)
ss54 = rep(0,n54)
hb54 = rep(0,n54)
rq54 = rep(0,n54)
df54 = rep(0,n54)
db54 = rep(0,n54)

for (i in 1:n54) {
  stt = st54[[i]]
  ss54[i] = silh(stt,g5)
  hb54[i] = hubertgamma(stt,g5)
  rq54[i] = rsf(stt,g5)
  df54[i] = dn(stt,g5)
  db54[i] = DBindex(stt,g5)
}

mssc54 = which.max(ss54)#3
```

```
mhbc54 = which.max(hb54)#3
mrqc54 = which.max(rq54)#3
mdfc54 = which.max(df54)#3
mdbc54 = which.min(db54)#3

# data 5 cluster 5

n55 = length(st55)
ss55 = rep(0,n55)
hb55 = rep(0,n55)
rq55 = rep(0,n55)
df55 = rep(0,n55)
db55 = rep(0,n55)

for (i in 1:n55) {
  stt = st55[[i]]
  ss55[i] = silh(stt,g5)
  hb55[i] = hubertgamma(stt,g5)
  rq55[i] = rsf(stt,g5)
  df55[i] = dn(stt,g5)
  db55[i] = DBindex(stt,g5)
}

mssc55 = which.max(ss55)#4
mhbc55 = which.max(hb55)#4
mrqc55 = which.max(rq55)#4
mdfc55 = which.max(df55)#1
mdbc55 = which.min(db55)#16
```

```
# Find the best score for each measure.

vs52 = max(ss52)
vs53 = max(ss53)
vs54 = max(ss54)
vs55 = max(ss55)
vs5 = c(vs52,vs53,vs54,vs55)
max(vs5)
which.max(vs5)

vh52 = max(hb52)
vh53 = max(hb53)
vh54 = max(hb54)
vh55 = max(hb54)
vh5 = c(vh52,vh53,vh54,vh55)
max(vh5)
which.max(vh5)

vr52 = max(rq52)
vr53 = max(rq53)
vr54 = max(rq54)
vr55 = max(rq55)
vr5 = c(vr52,vr53,vr54,vr55)
max(vr5)
which.max(vr5)

vf52 = max(df52)
vf53 = max(df53)
```

```
vf54 = max(df54)
vf55 = max(df55)
vf5 = c(vf52,vf53,vf54,vf55)
max(vf5)
which.max(vf5)

vd52 = min(db52)
vd53 = min(db53)
vd54 = min(db54)
vd55 = min(db55)
vd5 = c(vd52,vd53,vd54,vd55)
min(vd5)
which.min(vd5)

# K-means

km52 <- kmeans(g5,2)$cluster
km53 <- kmeans(g5,3)$cluster
km54 <- kmeans(g5,4)$cluster
km55 <- kmeans(g5,5)$cluster

ss52k = silh(km52,g5)
hb52k = hubertgamma(km52,g5)
rq52k = rsf(km52,g5)
df52k = dn(km52,g5)
db52k = DBindex(km52,g5)

ss53k = silh(km53,g5)
hb53k = hubertgamma(km53,g5)
```

```
rq53k = rsf(km53,g5)
df53k = dn(km53,g5)
db53k = DBindex(km53,g5)

ss54k = silh(km54,g5)
hb54k = hubertgamma(km54,g5)
rq54k = rsf(km54,g5)
df54k = dn(km54,g5)
db54k = DBindex(km54,g5)

ss55k = silh(km55,g5)
hb55k = hubertgamma(km55,g5)
rq55k = rsf(km55,g5)
df55k = dn(km55,g5)
db55k = DBindex(km55,g5)



vs5k = c(ss52k,ss53k,ss54k,ss55k)
max(vs5k)
which.max(vs5k)

vh5k = c(hb52k,hb53k,hb54k,hb55k)
max(vh5k)
which.max(vh5k)

vr5k = c(rq52k,rq53k,rq54k,rq55k)
max(vr5k)
which.max(vr5k)
```

```
vf5k = c(df52k,df53k,df54k,df55k)
max(vf5k)
which.max(vf5k)

vd5k = c(db52k,db53k,db54k,db55k)
min(vd5k)
which.min(vd5k)



# HAC
hc5 <- hclust(d=dist(g5))
hc52 <- cutree(hc5, 2)
hc53 <- cutree(hc5, 3)
hc54 <- cutree(hc5, 4)
hc55 <- cutree(hc5, 5)

ss52h = silh(hc52,g5)
hb52h = hubertgamma(hc52,g5)
rq52h = rsf(hc52,g5)
df52h = dn(hc52,g5)
db52h = DBindex(hc52,g5)

ss53h = silh(hc53,g5)
hb53h = hubertgamma(hc53,g5)
rq53h = rsf(hc53,g5)
df53h = dn(hc53,g5)
db53h = DBindex(hc53,g5)

ss54h = silh(hc54,g5)
```

```
hb54h = hubertgamma(hc54,g5)
rq54h = rsf(hc54,g5)
df54h = dn(hc54,g5)
db54h = DBindex(hc54,g5)

ss55h = silh(hc55,g5)
hb55h = hubertgamma(hc55,g5)
rq55h = rsf(hc55,g5)
df55h = dn(hc55,g5)
db55h = DBindex(hc55,g5)

vs5h = c(ss52h,ss53h,ss54h,ss55h)
max(vs5h)
which.max(vs5h)

vh5h = c(hb52h,hb53h,hb54h,hb55h)
max(vh5h)
which.max(vh5h)

vr5h = c(rq52h,rq53h,rq54h,rq55h)
max(vr5h)
which.max(vr5h)

vf5h = c(df52h,df53h,df54h,df55h)
max(vf5h)
which.max(vf5h)

vd5h = c(db52h,db53h,db54h,db55h)
min(vd5h)
```

```
which.min(vd5h)

# DBSCAN
db5 <- dbscan(g5,eps=1.3,MinPts = 4)$cluster
db5 = db5 + 1
ss5d = silh(db5,g5)
hb5d = hubertgamma(db5,g5)
rq5d = rsf(db5,g5)
df5d = dn(db5,g5)
db5d = DBindex(db5,g5)

# OPTICS
op50 <- optics(g5,eps = 4,minPts = 4)
op5 <- extractDBSCAN(op50,eps_cl = 1.3)$cluster
op5 = op5 + 1
ss5o = silh(op5,g5)
hb5o = hubertgamma(op5,g5)
rq5o = rsf(op5,g5)
df5o = dn(op5,g5)
db5o = DBindex(op5,g5)

pdf("data5result.pdf",width = 12, height = 16,colormodel='cmyk')
par(mfrow=c(4,3))
connectplot(g5,st53[mssc53],"Result by Silhouette(3)")
connectplot(g5,st54[mhbc54],"Result by Hubert gamma(4)")
connectplot(g5,st55[mrqc55],"Result by R-square(5)")
connectplot(g5,st55[mdfc55],"Result by Dunn index(5)")
connectplot(g5,st55[mdbc55],"Result by DB index(5)")
connectplot(g5,km53,"Result by Kmeans(3)")
```

```
connectplot(g5,km55,"Result by Kmeans(5)")
connectplot(g5,hc55,"Result by Hierarchical(5)")
connectplot(g5,hc53,"Result by Hierarchical(3)")
connectplot(g5,hc54,"Result by Hierarchical(4)")
connectplot(g5,db5,"Result by DBSCAN(3)")
connectplot(g5,op5,"Result by OPTIC(3)")
dev.off()


pdf("data5right.pdf",width = 8, height = 12,colormodel='cmyk')
par(mfrow=c(3,2))
connectplot(g5,st53[mssc53],"Result by Silhouette with 3 clusters")
connectplot(g5,st53[mhbc53],"Result by Hubert gamma with 3 clusters")
connectplot(g5,st53[mrqc53],"Result by R-square with 3 clusters")
connectplot(g5,st53[mdfc53],"Result by Dunn index with 3 clusters")
connectplot(g5,st53[mdbc53],"Result by DB index with 3 clusters")
dev.off()
```