2007-11-30

Clustering Methods for Delineating Regions of Spatial Stationarity

Jared M. Collings
Brigham Young University - Provo

CLUSTERING METHODS FOR DELINEATING REGIONS OF SPATIAL STATIONARITY

by

Jared M. Collings

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

December 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Jared M. Collings

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	William F. Christensen, Chair
Date	David G. Whiting
Date	Scott D. Grimshaw

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Jared M. Collings in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	William F. Christensen Chair, Graduate Committee

Accepted for the Department

Date	Scott D Grimshaw Graduate Coordinator

Accepted for the College

Date	Thomas W. Sederburg Associate Dean, College of Physical and Mathematical Science

ABSTRACT

CLUSTERING METHODS FOR DELINEATING REGIONS OF SPATIAL STATIONARITY

Jared M. Collings

Department of Statistics

Master of Science

The purpose of this paper is to examine and develop methods that can be used to delineate regions of stationarity. One of the major assumptions used in spatial estimation is that the data field is homogeneous with respect to the mean and the covariance function. As such, any spatial estimation presupposes that these criteria are met. With respect to analyses that may be considered new or experimental, however, there is no evidence that these assumptions will hold.

This paper seeks to further investigate data extracted by the use of Functional Magnetic Resonance Imaging (FMRI) as it is applied to brain tissue and how it measures blood flow to certain areas of the brain following the application of a stimulus. As a precursor to detailed spatial analysis of this kind of data, this paper develops methods of grouping data based on the necessary conditions for spatial statistical analysis.

ACKNOWLEDGEMENTS

The completion of this paper would not have been possible without the support and encouragement of many. It has been a long road to completion. I would like to thank my wife for pushing me as needed (and appropriate) and encouraging me that finishing was actually possible.

Additionally, I am very appreciative of the support from my parents who are able to remind me that I was not done, support me in finishing, and even provide technical assistance as necessary.

Lastly, I would like to extend appreciation to Dr. Christensen for introducing me to this area of study, for providing a great foundation of knowledge, and for allowing me to jump back into things after an extended break. Thanks to all for helping me finish.

CONTENTS

FIGURES

1. INTRODUCTION

In considering the human brain, one area of interest is the relationship between location and function. It is well known that the brain controls all functions of the body and that parts of the brain are specialized to certain tasks. One area of interest, however, is to determine whether or not parts of the brain associated with these certain tasks may be identified by analyzing blood flow to a given region after some sort of stimulus is applied.

In Silent Functional Magnetic Resonance Imaging (SFMRI), a series of audio impulses are given to a patient and blood flow in the brain is measured. The blood flow is measured in the "on" state, where the brain is subject to the stimulus, and in the "off" state, where the brain returns to a normal or resting state. Each area of measurement is a three-dimensional cube wherein the blood flow is measured, and these areas of measurement are referred to as voxels. A voxel is called "active" when there exists a significant difference between the "on" and "off" states.

Given the nature of blood flow, it is unreasonable to assume that the voxels are independent of each other. Thus, before estimating the level of activation for each voxel, the spatial dependence structure of the voxel measurements is estimated. Once such estimates of the spatial dependence are obtained, voxel activation maps exhibiting spatial smoothness can be created. However, in order to carry out spatial estimation of the voxel activation levels for the measured regions, it is necessary to subdivide the brain into regions that exhibit local stationarity. Stationarity is the condition where all the observations in the region are distributed with the same mean and variance covariance structures.

2. LITERATURE REVIEW

The first section of this chapter is devoted to explaining the nature of the FMRI data. In order to clarify the spatial nature of such data, a brief overview of spatial statistics will be given in Section 2.2. Section 2.3 outlines the process of hierarchical clustering, which will be employed to delineate regions of stationarity. A discussion of prior research and methodologies can be found in Section 2.4.

2.1 Brain Imaging Data (FMRI)

The idea that neuronal activity is connected to changes in brain metabolism and blood flow was suggested by Roy and Sherrington (1890) when they conducted highly invasive experiments with dogs, cats, and rabbits.

One method of measuring changes in brain blood flow is the high speed functional magnetic resonance imaging (FMRI), which makes use of the paramagnetic and diamagnetic properties of deoxyhemoglobin and oxyhemoglobin and the local, tissue-specific changes in blood volume, blood flow, and blood oxygenation in response to some controlled stimuli (Belliveau et al. 1991; Kwong et al. 1992; Ogawa et al., 1992). As applied to analyses of brain activity, FMRI serves to map neurological function to neuroanatomy, making it possible to connect function to location. One benefit of FMRI is that it becomes possible to obtain, in a few seconds, hundreds of noninvasive "moving pictures" of the brain as it processes stimuli.

Lange and Zeger (1997) gave a highly simplified explanation of the process of FMRI. In blood, there exists iron (hemoglobin) and iron is paramagnetic—the property of being weakly attracted to an external field. Deoxygenated blood, or "blue blood," is more paramagnetic than oxygenated blood, or "red blood." So, when brain activity increases, the flow and volume of red blood will also increase at the capillary level, causing an increase in the relative oxygen content. This increase in red blood volume results in a difference in the magnetic susceptibility of red blood with respect to blue blood. This difference can be used to produce a detectable local signal. Signals such as this can then be used to create contrast in a digital image.

Such digital images as produced by FMRI can be broken up into pixels (picture elements) in the two-dimensional case and into voxels (volume elements) in the three-dimensional case. Each two-dimensional image, or slice, consists of a 64 $\times$ 64 grid measuring the response to the stimulus. The physical size of the voxels are typically 3.125 mm $\times$ 3.125 mm $\times$ 5 mm. Lange and Zeger (1997) studied the visual cortex, where each voxel contained roughly 2 million neurons.

For this study, audio impulses are used as the stimuli. Instead of traditional FMRI measurements, silent functional magnetic resonance imaging (SFMRI) is used. SFMRI differs from FMRI in that only one measurement is taken during each recurrence of the stimulus state and the rest state. Measurements are taken alternately in an "on" state where an impulse is set upon the subject, denoted by (x), and an "off" state when the subject returns to a resting or non-stimulated state, denoted by (y). This process is repeated many times and the "activation" state of a voxel is assessed using some function of $\bar{x}$ and $\bar{y}$, like $\bar{x} - \bar{y}$.

The voxels of the brain that specialize in interpreting audio impulses are expected to have higher blood flow. These active voxels are expected to cluster together in contiguous groups with similar function, such as in the audio cortex. These hypothesized groups within the brain may possess mean, variance, or spatial structures that are vastly different from the structure of other sections. Traditionally, voxel activation is assessed via t-tests. For a discussion of traditional approaches, see Genovese (2000). One problem with this method lies in the fact that the data are noisy and, therefore, the activation maps created from individual t-tests are unrealistically noisy or rough. Additionally, there may be an insufficient number of observations to adequately test for significant activation.

In order to apply spatial prediction methods to create voxel activation maps, the usual approach is to assume stationarity for the region of interest to be mapped; however, in this case (brain activation), the stationarity assumption is violated. The voxels in different regions should not necessarily have the same means, nor should they necessarily have the same variances or spatial dependence structures. Moreover, the highly irregular manner in which brain tissue folds back on itself yields the potential for neighboring voxels to have dramatically different behavior. Hence, the voxel observations can be dramatically non-stationary.

2.2 Spatial Statistics

In this section, a short description of the general spatial model is given. In general, the variable of interest **Z** is given as a function of its location **s**, where $\mathbf{s} \in \Re^d$ is a point in d-dimensional Euclidean space. Because **s** is allowed to vary over the fixed index set $\mathbf{D} \subset \Re^d$ the following random field is generated:

$$\{\mathbf{Z}(\mathbf{s}) : \mathbf{s} \in \mathbf{D}\}.$$

In this paper, **Z**(**s**) is modeled in terms of mean structure, or large-scale variation, and in terms of the variogram, which models the dependent nature of the data. Let **Z**(**s**) = **μ**(**s**) + **δ**(**s**), $\mathbf{s} \in \mathbf{D}$, where **μ**(**s**) denotes the deterministic mean structure, or large-scale variation; that is, **μ**(**s**) is equal to E(**Z**(**s**)) and **δ**(**s**) denotes a zero-mean intrinsically stationary stochastic process with variogram $2\gamma(\mathbf{h})$.

The assumption of stationarity is upheld when the characteristics of the data elements do not vary across the region $\mathbf{D}$. In other words, the probability of a given fluctuation from the mean level (measured by **δ**(**s**)) is assumed to be the same for each point **Z**(**s**) within the region. The variogram, $2\gamma(\mathbf{h})$, from above is defined as $2\gamma(\mathbf{s}_1 - \mathbf{s}_2) \equiv \mathrm{var}(\mathbf{Z}(\mathbf{s}_1) - \mathbf{Z}(\mathbf{s}_2))$. (Note: $\gamma(\mathbf{s})$ has been referred to as the semivariogram. See Figure 2.1 for a graphical presentation of the semivariogram.)

The isotropic variogram and the semivariogram are solely functions of distance. That is, when $2\gamma(\mathbf{s}_1\text{-}\mathbf{s}_2)$ is a function solely of $\|\mathbf{s}_1\text{-}\mathbf{s}_2\|$, $2\gamma(\cdot)$ is said to be isotropic. In this case, $\gamma(\mathbf{h})$ can be redefined using $\gamma(\|\mathbf{h}\|)$ where $\|\cdot\|$ is the L_2 norm. Clearly $\gamma(\mathbf{h}) = \gamma(-\mathbf{h})$, where **h** is the distance between points and $\gamma(\mathbf{0}) = 0$. However, where $\gamma(\mathbf{h}) \rightarrow c_0 > 0$ as $\mathbf{h} \rightarrow 0$, c_0 is referred to as the nugget effect by Matheron (1962). This so-called nugget effect represents microscale variation that causes this discontinuity at the origin, and is comprised of two parts. The nugget effect, c_0, is due in part to measurement error, c_{ME}, and a white noise process that is assumed to occur on the very small scale, c_{MS}. Thus,

$$c_0 = c_{MS} + c_{ME.}$$

The sill is equal to twice the variance of **Z**(**s**). Notationally, $2\sigma^2 = \lim_{\|\mathbf{h}\|\to\infty} 2\gamma(\mathbf{h})$. Additionally, the sill, $2\sigma^2$, is equal to $2\gamma(\|\mathbf{s}_1\text{-}\mathbf{s}_2\|)$ when $\mathbf{s}_1$ and $\mathbf{s}_2$ are far enough apart to be considered independent.

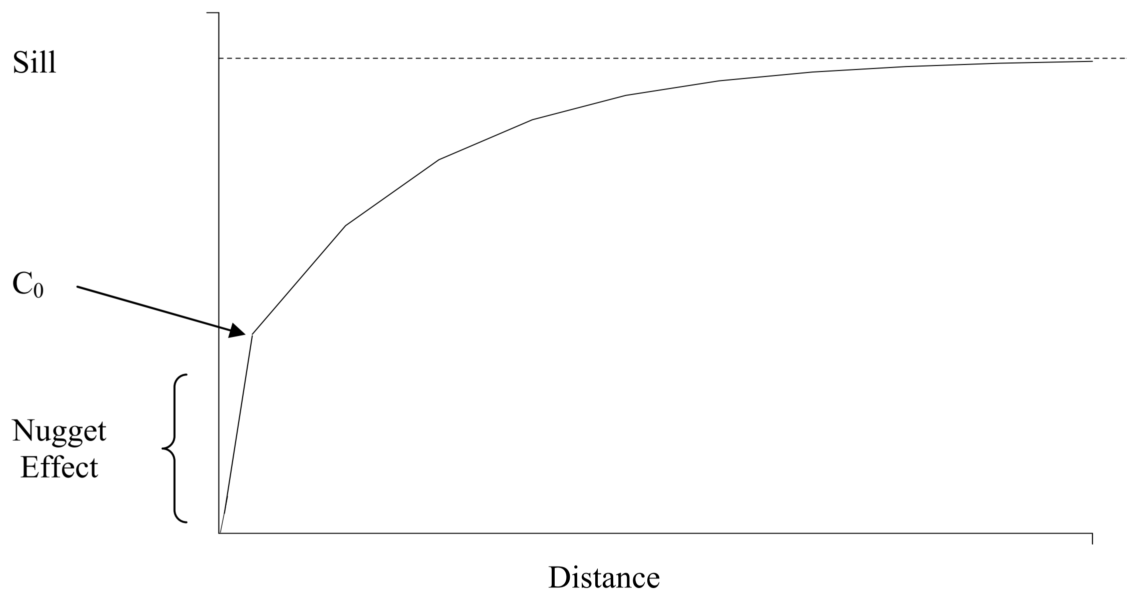


Figure 2.1 Semivariogram. The above diagram shows the behavior of the semivariogram. The value of the semivariogram approaches c_0 (the nugget effect) as the distance approaches zero. As the distance increases, the value of the semivariogram approaches its upper limit (the sill).

As is implied by the above definition for **δ**(**s**), the variogram, 2γ(**h**), can be estimated only when the process is stationary. The proposed approach for modeling non-stationary processes is to first discriminate, or delineate, areas of stationarity, after which it should be possible to proceed with spatial analysis within each neighborhood of stationarity.

2.2.1 Estimation of the variogram

Based on method-of-moments, Matheron (1962) suggested the following methodas a way of estimating the variogram. This method is often referred to as the classical estimator of the variogram. Let

$$2\hat{\gamma}(\mathbf{h}) \equiv \frac{1}{|N(\mathbf{h})|}\sum_{N(\mathbf{h})}\left(\mathbf{Z}(\mathbf{s}_i)-\mathbf{Z}(\mathbf{s}_j)\right)^2, \qquad \mathbf{h} \in \Re^d \tag{2.2.1}$$

where

$$N(\mathbf{h}) \equiv \{(\mathbf{s}_i, \mathbf{s}_j) : \mathbf{s}_i - \mathbf{s}_j = \mathbf{h}; i, j = 1, \ldots, n\} \tag{2.2.2}$$

and $|N(\mathbf{h})|$ is the number of distinct pairs in N(**h**).

Similarly, robust estimators of the variogram were introduced by Cressie and Hawkins (1980). They set

$$2\bar{\gamma}(\mathbf{h}) \equiv \frac{\left\{\frac{1}{|N(\mathbf{h})|}\sum_{N(\mathbf{h})}|\mathbf{Z}(\mathbf{s}_i)-\mathbf{Z}(\mathbf{s}_j)|^{1/2}\right\}^4}{\left(0.457+\frac{0.494}{|N(\mathbf{h})|}\right)} \tag{2.2.3}$$

and

$$2\tilde{\gamma}(\mathbf{h}) \equiv \frac{\left[med\left\{|\mathbf{Z}(\mathbf{s}_i)-\mathbf{Z}(\mathbf{s}_j)|^{1/2} : (\mathbf{s}_i, \mathbf{s}_j) \in N(\mathbf{h})\right\}\right]^4}{\left(0.457+\frac{0.494}{|N(\mathbf{h})|}\right)}. \tag{2.2.4}$$

Cressie and Hawkins also show through simulation that $2\bar{\gamma}(\cdot)$ is generally preferred over $2\tilde{\gamma}(\cdot)$.

2.3 Hierarchical Cluster Analysis

2.3.1 Clustering Methods

Cluster analysis is used to group observations, usually multivariate, into clusters—groups containing observations that are “close” in terms of Euclidean distance. The basic idea is to find an optimal grouping where the observations are similar within each cluster, and the clusters themselves are different from each other. One common approach to clustering is hierarchical clustering. This approach begins with *n* clusters, one for each observation, and ends with all *n* observations in the same cluster. In other words, the observations begin in separate clusters, then the two “closest” observations are combined into one new cluster. This process of combining the two closest clusters into a new single cluster continues until all the observations have been combined into a single cluster.

Euclidean distance is a common measure of similarity, or dissimilarity, among observations. For two vectors $\mathbf{x} = (x_1, x_2, \ldots, x_p)'$ and $\mathbf{y} = (y_1, y_2, \ldots, y_p)'$ the Euclidean distance between **x** and **y** is defined by

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})'(\mathbf{x}-\mathbf{y})} = \sqrt{\sum_{j=1}^{p}\left(x_j - y_j\right)^2}\,.$$

Another method of measuring distance is using the statistical difference defined by

$$D(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})'\,\Sigma^{-1}(\mathbf{x}-\mathbf{y})}\,.$$

This measure adjusts for differing variances and covariances among the differing groups. However, since there are assumed to be different groups within the data set, there is no obvious way to choose **Σ**.

Attention should also be given to the scale of measurements used in computing the Euclidean distance. Depending on the scale of measurements, the relative distances can change. To counter this, many authors recommend the usual standardizing of the variables—subtracting the mean and dividing by the standard deviation.

While the Euclidean distance between two points is clearly defined above, the distance between two clusters can be defined in several ways. The following are some of the most common methods for determining the distance between clusters (Fisher and Van Ness 1971; Van Ness 1973).

In the single linkage (or nearest neighbor) method, the distance between two clusters A and B is defined as the minimum distance from a point in A to a point in B. Thus,

$$\mathrm{D(A,B)} = \min\{d(\mathbf{y}_i,\mathbf{y}_j), \text{ for } \mathbf{y}_i \text{ in A and } \mathbf{y}_j \text{ in B}\}.$$

In the complete linkage (or farthest neighbor) method, the distance between two clusters A and B is defined as the maximum distance from a point in A to a point in B. Thus,

$$\mathrm{D(A,B)} = \max\{d(\mathbf{y}_i,\mathbf{y}_j), \text{ for } \mathbf{y}_i \text{ in A and } \mathbf{y}_j \text{ in B}\}.$$

In the average linkage method, the distance between two clusters A and B is defined as the average of the $n_A \times n_B$ distances from each of the n_A points in A to each of the n_B points in B. That is,

$$D(A,B) = \frac{1}{n_A n_B} \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} d(\mathbf{y}_i, \mathbf{y}_j).$$

2.3.2 Clustering Outcomes

The agglomerative hierarchical method of clustering begins with each point in the set as its own cluster. Then, the two closest clusters are grouped together, where closeness is determined by the linkage method is being used. This process is repeated until all the observations are grouped into a single cluster. The results are often displayed graphically in a tree diagram, which shows the development of the clusters. The method of determining the number of cluster ultimately used in analysis depends largely upon the judgment of the analyst and may vary according to the specific situation.

Generally, the use of single linkage differs from complete and average linkage in that it will create long string-like clusters. This phenomenon, known as chaining, generally yields undesirable clustering results. The results from complete and average linkage generally agree with each other. For additional information on displaying the results of hierarchical clustering, see Gordon (1987).

2.4 Methods of Clustering Spatial Data: History

2.4.1 Clustering

In the case of spatially dependent data, the results of normal clustering techniques do not always make sense given the nature of the data. With spatial data, one of the assumptions is that each observation is dependent on other nearby observations. In the case where similar observations exist but are separated by large distances it is often unreasonable to group them together.

This idea is supported by Legendre (1987), who found that imposing constraints on spatial data results in answers that are meaningful in context. He describes various forms of constrained clustering and then examines whether constrained clustering is needed to get meaningful results and how to determine if the groups found by constrained clustering are real. Legendre answers the first question through a simulated data set, where he found that constrained clustering recovered more information than unconstrained clustering.

Legendre's studies led to the conclusion that constrained clustering should always used when the data are assumed to be spatially (or temporally) correlated. Legendre also states, in answer to the second question, that it is possible to discern whether the groups found by constrained clustering are real. Legendre found that the spatially correlated data sets generally have larger differences between groups. Additionally, Legendre found that with constrained clustering techniques, the different linkage methods result in more similar groupings as compared to unconstrained clustering.

Wartenberg (in an unpublished manuscript cited by Legendre 1987) gives an example of what happens in situations of unconstrained clustering and constrained clustering through a study from the health sciences. Lung ailments may come from many different causes: occupational (such as coal mining), ambient (such as living near industrial areas), or personal habits (such as tobacco consumption), all of which can lead to differing severity of lung conditions. The unconstrained clustering tended to group the samples by severity of cases while the spatially constrained clustering was more likely to group observations with similar types of causes.

Another method of constrained clustering was also used by Huel, Petiot, and Lazar (1986) as they developed an algorithm for analyzing contiguous zones in a geographical epidemiology study. They developed an algorithm that groups the continuous geographical units that are similar with respect to the variables of interest. First, a distance measure between the zones is defined. This measure is defined in such a way as to prevent two or more of the distances measured from resulting in equal magnitudes; in other words, all distances calculated are unique and can be ranked from smallest to largest. Next, a distance threshold is chosen beyond which two zones cannot be grouped (i.e. if the distance between zones is less than the threshold, the zones are considered to be adjacent). Then, the two zones that have the smallest distance, subject to the constraint that the zones are adjacent, are grouped together. This process is continued until all the groups have been combined. Huel, Petiot, and Lazar then use this algorithm to group geographical zones in Normandy, France, where measurements of suicide rates were taken. In this study, the probability level from Cochran's test was defined as the distance for the algorithm. (Note that Cochran's test is useful for measuring samples on a categorical scale and measures whether or not different samples have the same number of successes and failures.)

One problem that arises in analysis of spatial data is that of non-stationarity. If, for example, the data are assumed to be spatial in nature and are assumed to have separate intrinsic groups, then estimating the variogram for the entire data set will violate the stationarity assumption. This problem is addressed by Haas (1990), in which the ecological effects of sulfuric and nitric acid (found in rainfall) on the environment in the United States were studied. This process of statistical estimation was complicated because of a strong spatial trend (mean non-stationarity) and an apparent spatial covariance structure dependent on location (covariance non-stationarity). Haas evaluated four statistical methods of estimating confidence intervals for the precipitation chemistry at the locations of measurement. Of interest is a method of estimation where the sample variogram is estimated using only data within a circular neighborhood immediately surrounding the estimate location—this method is called a window. This window moves along with the estimate location, where a new variogram estimate is calculated. This allows for the variance covariance structure to be a function of location and allows the model to account for more of the spatial trends.

However, the implementation of this moving window brings about the conflicting goals of having the window as small as possible while still allowing for a covariance model to be fitted using only the window's data. The smaller the window, the better the approximation of stationarity, while the larger the window, the more accurate the semi-variogram estimates. Haas (1990) suggests that the semi-variogram estimates at shorter lags have lower variance than estimates at longer lags given the same number of distance pairs; thus, fewer pairs are needed at shorter lag distances.

Another problem in the analysis of spatially correlated data is that of scale. Spatial resolution is associated with the smallest area for which data can be obtained. High resolution comes from small geographical units, which allow for an increase in the ability to differentiate nearby objects. However, small geographical areas, which usually have small populations, tend to have large variability.

Carvalho, Cruz and Nobre (1996) developed an algorithm to combine small, homogeneous areas in order to achieve minimum sample size requirements. As applied to census data, the algorithm begins with a specified census tract for which a search is performed among all adjacent tracts with the same risk profiles. In each census tract, a centroid is defined as an average of all the data collected in the tract. Using Euclidean distances between these centroids, a measurement is available to compare how close different tracts are to each other. The neighboring tract that is closest to the specified tract is subsequently combined with the specified tract to create a new tract. Then, if the desired sample size is achieved, the algorithm will move on to the next specified tract; otherwise, it will compare the new tract with its neighboring tracts, repeating the process until all census tracts have been specified by the algorithm. If, however, a census tract exists without neighboring tracts exhibiting similar risk profiles, it is defined as isolated.

The results of this agglomerative algorithm include (1) complete—the desired sample size is reached, (2) incomplete—the desired sample size is not reached, but there are no additional neighboring tracts of the same class, and (3) isolated—no neighboring tracts existed to be combined.

For additional studies in clustering see Anderson and Titterington (1997), Cuzick and Edwards (1990), Marshall (1991), and Raubertas (1988).

2.4.2 Methods of Evaluating Clustering Techniques

Two methods of assessing the efficiency of clustering models were recommended by Milligan (1983)—the Jaccard index and the Rand index. For a situation with n observations where the true groups are known, two half matrices (triangular n x n) are computed—one is computed according to the true groups and another is computed according to the classification method, or estimated groups. The *ij* position of these matrices contains a 1 when the *i* observation and the *j* observation are members of the same group, and a 0 otherwise. Then a summary two-way table is set up using the counts of 1s and 0s for each observation. The Jaccard index and the Rand index are calculated using such two-way tables (as in Figure 2.4.1) and are defined as

		Second Matrix	
		1	0
First matrix	1	a	b
	0	c	d

Figure 2.2 General Two Way Table. The two way tables summarizes the comparison counts, represented here by a, b, c, and d. The Rand and Jaccard indeces both, then, reference these counts.

$$Jaccard = \frac{a}{a+b+c},$$

$$Rand = \frac{a+d}{a+b+c+d} \ .$$

For example, in the simple situation where there are two groups, as given in Figure 2.4.2, and there are three estimated groups, as given in Figure 2.4.3, the indices would be computed by first constructing:

I	I
I	I
II	II

Figure 2.3 True Grouping

I	I
II	III
II	III

Figure 2.4 Estimated Grouping

To begin with, the two half matrices are computed for the true and estimated groups—given in Figure 2.4.4. Each cell in the data field is compared to each other cell. A value of 1 is assigned to the half matrix position if both cells are in the same group, otherwise the value assigned to the half matrix position is a 0. For the true groupings, the first cell (top left) is compared to the second cell (top right) and a value of 1 is assigned to the half matrix. The same is true for the third and fourth cells (middle left and right). A value of 0 is assigned for cells five and six (bottom left and right) because they are in different groups. The same process is continued for other columns of the half matrices.

$$\begin{array}{|c|c|c|c|c|c}
\hline
 & & & & & \\
\hline
1 & & & & & \\
\hline
1 & 1 & & & & \\
\hline
1 & 1 & 1 & & & \\
\hline
0 & 0 & 0 & 0 & & \\
\hline
0 & 0 & 0 & 0 & 1 & \\
\hline
\end{array}
\qquad
\begin{array}{|c|c|c|c|c|c}
\hline
 & & & & & \\
\hline
1 & & & & & \\
\hline
0 & 0 & & & & \\
\hline
0 & 0 & 0 & & & \\
\hline
0 & 0 & 1 & 0 & & \\
\hline
0 & 0 & 0 & 1 & 0 & \\
\hline
\end{array}$$

$$\begin{array}{cc|c|c|}
 & & \multicolumn{2}{c}{\text{Second Matrix}} \\
 & & 1 & 0 \\
\cline{3-4}
\text{First matrix} & 1 & 1 & 6 \\
\cline{3-4}
 & 0 & 2 & 6 \\
\cline{3-4}
\end{array}$$

$$\text{Jaccard Index} = \frac{1}{1+6+2} = \frac{1}{9}$$

$$\text{Rand Index} = \frac{1+6}{1+6+2+6} = \frac{7}{15}$$

Figure 2.5 Index Calculation Example. The half matrices are shown above. The matrix for the true groups is on the left and the matrix for the estimated groups is on the right. Below the matrices is the two-way table along with the calculations for both the Jaccard and Rand indices.

3. METHODS OF CLUSTERING OBSERVATIONS IN A NON-STATIONARY RANDOM FIELD

3.1 Data Structure

The previous approaches attempt to create contiguous clusters that have similar mean structures. In addition to having the same mean structure, however, the clusters should group observations that have similar variances and similar spatial structures. In other words, the non-stationary random field is to be broken down into contiguous regions (clusters) that exhibit spatial stationarity.

In order to attain localized estimates of the spatial variogram, the moving window approach introduced by Haas (1990) was used. In the moving window approach, a window, or neighborhood, around each point is used to estimate the variance structure at that point. In terms of this approach, the smaller the region, the better the approximation of stationarity. On the other hand, with larger windows, more data points are available for use in estimation.

In attempting to separate the observations into stationary regions, hierarchical clustering can be performed. If the data to be analyzed exhibits spatial characteristics, a location index should be included to encourage the creation of clusters that contain spatially contiguous observations.

For each data point, a data vector was obtained with the form

$$\mathbf{v}(\mathbf{s}) = (v_1(\mathbf{s}), v_2(\mathbf{s}), v_3(\mathbf{s}), v_4(\mathbf{s}), v_5(\mathbf{s}), v_6(\mathbf{s}), v_7(\mathbf{s})),$$

where $\mathbf{v}(\mathbf{s})$ represents the (possibly vector-valued) responses to be used in the clustering. In the case of SFMRI, $v_1(\mathbf{s})$ represents the observed level of activation for a specific location **s**, defined using the difference between stimulated measurements and the average blood flow for the non-stimulated measurements, $\overline{x}(\mathbf{s}) - \overline{y}(\mathbf{s})$; $v_2(\mathbf{s})$ represents the row (or x-coordinate) index; $v_3(\mathbf{s})$ represents the column (or y-coordinate) index; $v_4(\mathbf{s})$ is $2\overline{\gamma}_{\mathbf{s}}(1)$, referred to as the lag 1 estimate of the variogram; $v_5(\mathbf{s})$ is $2\overline{\gamma}_{\mathbf{s}}(2)$, the lag 2 estimate; $v_6(\mathbf{s})$ is $2\overline{\gamma}_{\mathbf{s}}(3)$, the lag 3 estimate; $v_7(\mathbf{s})$ is the estimated 90th percentile of the measurements $2\overline{\gamma}_{\mathbf{s}}(1)$, $2\overline{\gamma}_{\mathbf{s}}(2)$, $2\overline{\gamma}_{\mathbf{s}}(3)$, $2\overline{\gamma}_{\mathbf{s}}(4)$, $2\overline{\gamma}_{\mathbf{s}}(5)$, $2\overline{\gamma}_{\mathbf{s}}(6)$, and $2\overline{\gamma}_{\mathbf{s}}(7)$. The quantity $v_7(\mathbf{s})$ represents an approximation of the sill, which is equivalent to $2\sigma^2(\mathbf{s})$, the variance at location **s**.

To estimate the variogram, we recommend the use of the robust variogram estimator $2\overline{\gamma}(\cdot)$ as discussed by Cressie and Hawkins (see Section 2.2.1). For data points close to the natural boundary of a group, the estimates of the variogram are generally clustered into two different and distinct "clouds" (see Figure 3.1). The more robust (median-based) estimators generally provide variogram estimates that lie closer to the actual variogram cloud as compared to the non-robust (mean-based) estimators. This is particularly true for data points along a natural border between groups. The existence of multiple groups causes the distinct variogram clouds shown in Figure 3.1.

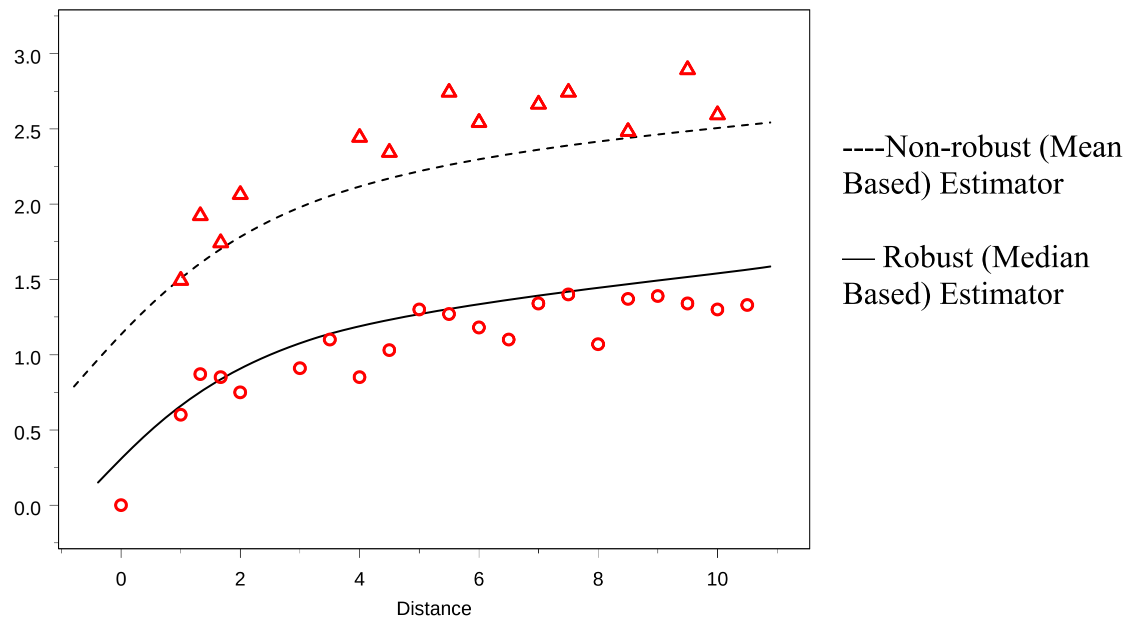


Figure 3.1 Robust vs. Non-Robust Estimators of the Variogram. The Δ's represent data points from one group and the ○'s represent data points from a second group.

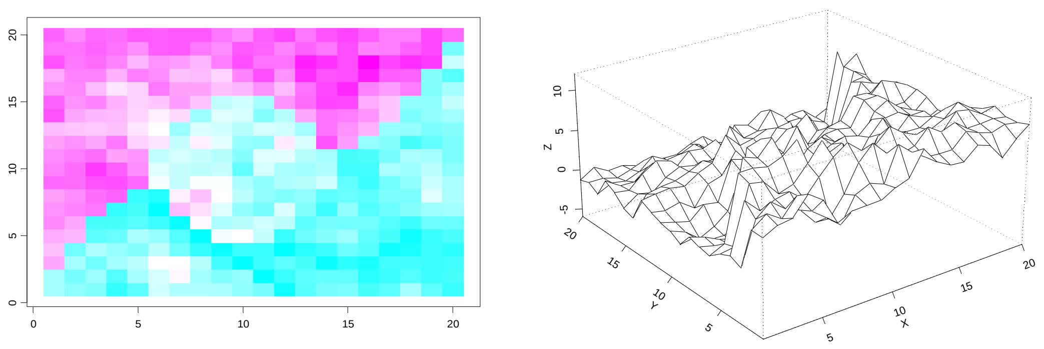
3.2 Proposed Methods

In this paper we will develop three methods of clustering data (as described later in this chapter). To compare and evaluate these methods, simulated data sets on a 20 x 20 grid are used. The grid is separated into three different regions—one in the upper left, one in the center and one in the lower right. Figure 3.2 provides two plots for each simulated data set. In the top plot of each pair, darker shades of purple represent low simulated values; the white squares represent intermediate simulated values; and darker shades of blue represent high simulated values. The bottom plot of each pair provides a three-dimensional view of the data set. In the first data set, the three different regions have different mean structures. This allows each region to be identified with relative ease. In the second data set, the three different regions have the same mean structure, but different variance structures; thus, the smoothness of the plotted points is very different across the regions and these regions are more difficult to identify.

The complete linkage clustering method, which clusters based solely on the distance between $\mathbf{Z}(\mathbf{s})$, is demonstrated on the two sample data sets and results in the clusters shown in Figures 3.3 and 3.4. As might be expected, this technique does a relatively good job at identifying the groups in Simulated Data Set 1 (Figure 3.3). However, as the number of groups increases, the clustered groupings become more and more fragmented. As might be expected, the complete linkage method is ineffective at identifying the groups in Simulated Data Set 2 (Figure 3.4). This technique is not able to identify any of the true groups. The same problems exist as with Data Set 1, in that the groups become increasingly fragmented as the number of groups increases.

In general, complete linkage is able to identify the three main regions for Simulated Data Set 1, but is not able to identify any of the regions in Simulated Data Set 2. In applying the complete linkage method to **z**(**s**) only the different mean structures are identified, whereas no information is identified with regard to the variance structures. Additionally, this method can produce non-contiguous groupings that cannot be used for spatial analysis; as the number of clusters increases so does the fragmented quality of the clusters. The three methods proposed in this chapter seek to address the limitations of complete linkage methods as applied to spatial analysis.

Simulated Data Set 1



Simulated Data Set 2

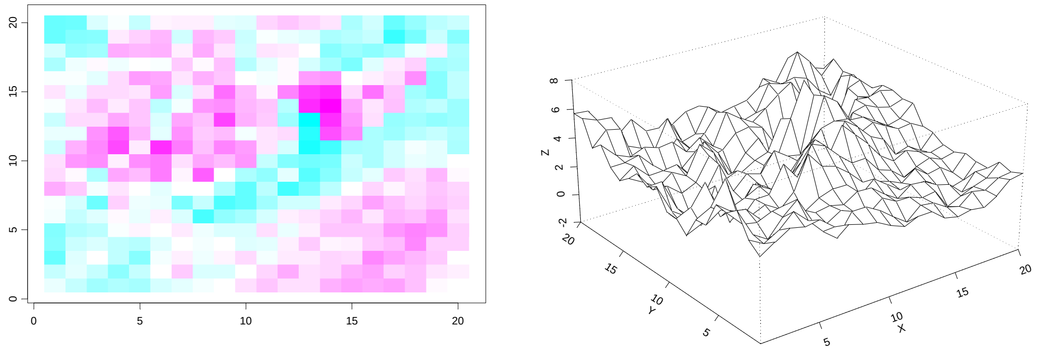


Figure 3.2 Simulated Data Sets. Each data set has three distinct regions: one in the top left, one in the center, and one in the bottom right. In the first data set the three regions have different mean structures. In the second data set the three regions have the same mean structure, but different variance structures. The shades of purple represent low simulated values; the white values represent intermediate simulated values; and the blue values represent high simulated values.

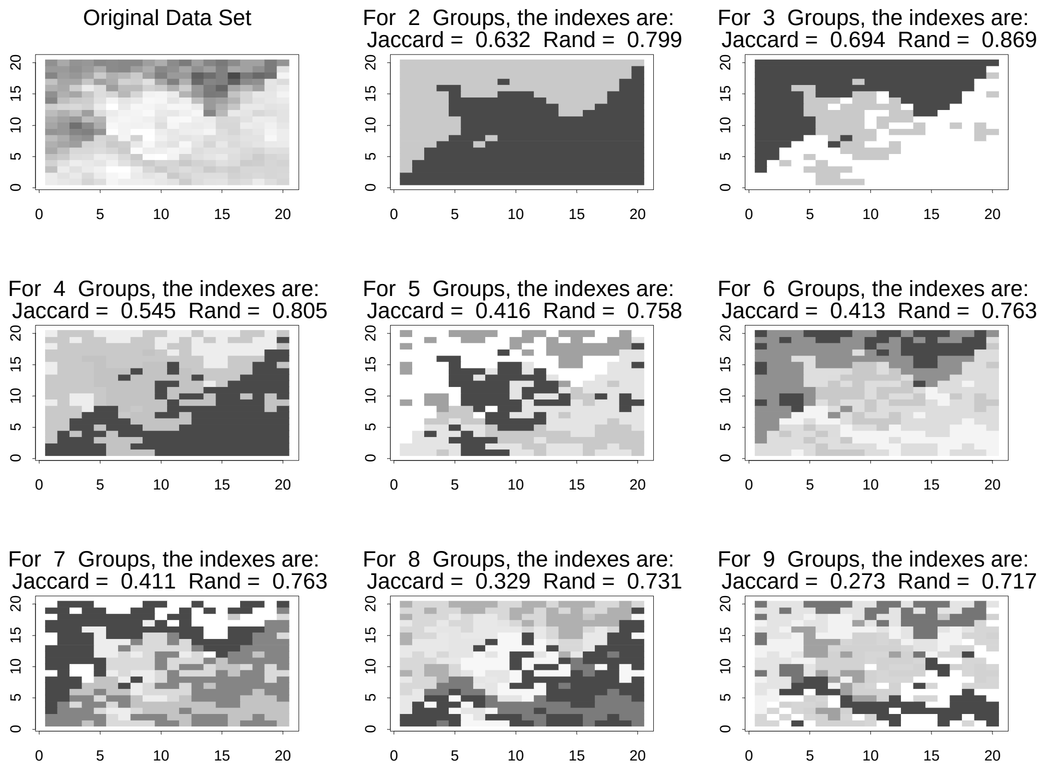


Figure 3.3 Complete Clustering Results for Simulated Data Set 1. The upper left plot shows the original three groups in the data set. The remaining plots represent the clustering produced by the Complete Linkage Method for 2 through 9 groups. The Jaccard and Rand indices are provided for each plot.

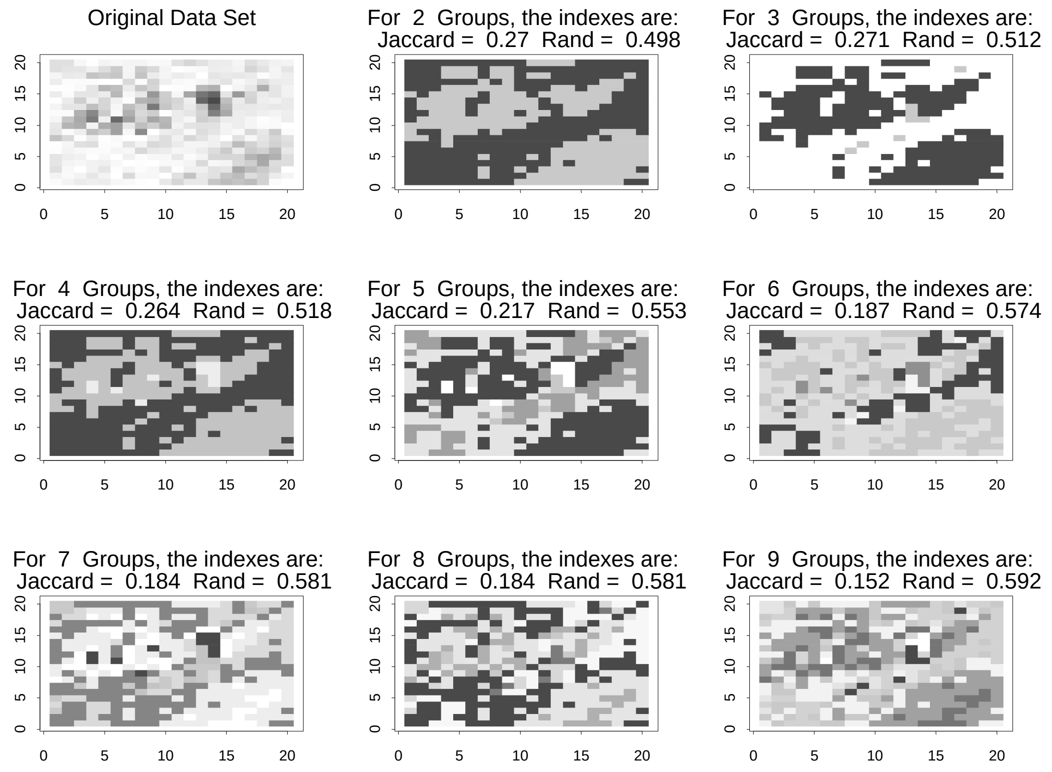


Figure 3.4 Complete clustering results for Simulated Data Set 2. The upper left plot shows the original three groups in the data set. The remaining plots represent the clustering produced by the Complete Linkage Method for 2 through 9 groups. The Jaccard and Rand indices are provided for each plot.

3.2.1 Basic Window Method

In order to obtain estimates of $v_4(\mathbf{s})$, $v_5(\mathbf{s})$, $v_6(\mathbf{s})$, and $v_7(\mathbf{s})$ that are valid at the specific location **s**, the moving window described by Haas (1987) is used. For each position in the data field (usually a grid), the variogram is estimated at distances of 1 through 7. For $2\bar{\gamma}_{\mathbf{s}}(1)$, or lag 1, the window consists of all the data points within a window of ± 1 row or column of the specified position. For $2\bar{\gamma}(2)$, or lag 2, all points within a window of ± 2 are used. The window size continues to grow in this manner until $2\bar{\gamma}(7)$, or lag 7, is estimated. For positions along the edges or in the corners, portions of the window will be outside of the data grid. In this case, only the positions that are aligned within the window and the grid are used for estimating the appropriate lags. Finally, the 90 percentile among $2\bar{\gamma}_{\mathbf{s}}(1)$, $2\bar{\gamma}_{\mathbf{s}}(2)$, $2\bar{\gamma}_{\mathbf{s}}(3)$, $2\bar{\gamma}_{\mathbf{s}}(4)$, $2\bar{\gamma}_{\mathbf{s}}(5)$, $2\bar{\gamma}_{\mathbf{s}}(6)$, and $2\bar{\gamma}_{\mathbf{s}}(7)$ is found at each specified location. After the data vector $\mathbf{v}(\mathbf{s})$ is constructed, the measurements in $\mathbf{v}(\mathbf{s})$ are centered and scaled.

Weights are applied to the data vector depending upon the composition of the data field. A weighting of (3, 1, 1) is applied to the mean structure and a weighting of (2, 1, 1, 0, 1) is applied to the spatial structure. The weights are chosen to be similar in their total effect on the data vector $\mathbf{v}(\mathbf{s})$. The weights are then adjusted by multiplying the mean structure weighting by a and the spatial structure weighting by $1-a$, for increasing values of a. These weighted data vectors are then clustered using the hierarchical clustering method described above. Because both the Jaccard and the Rand indices evaluate clustering processes in basically the same way, only the Jaccard index is shown

Using the Jaccard index as the criterion for analysis of the Basic Window Method, Figures 3.5 and 3.6 provide a summary of how well this method identifies the groups within Data Examples 1 and 2, respectively. (Additional images of the hierarchical clusters produced by application of the Basic Window Method to Simulated Data Sets 1 and 2 can be found in Appendix A.)

Generally, as the number of groups increases, the value of the Jaccard index decreases. Recall that Simulated Data Set 1 contains differing mean structures and similar spatial structures (see Figure 3.2). The Basic Window Method performs best with equal weighting between mean structure and spatial structure (that is, when $a = 0.50$) for Simulated Data Set 1. Recall that Simulated Data Set 2 has a similar mean structure throughout the region (see Figure 3.3). The Basic Window Method using only spatial information ($a = 0.0$) is at least as good as the subsequent iterations that include the mean structure data (i.e. $a > 0$).

In summary, the Basic Window Method includes additional information in the clustering process and thus provides an improvement over traditional clustering techniques. However, in order to perform spatial analysis on a data structure, the data within each cluster must be contiguous. The Basic Window Method fails to adequately address this issue.

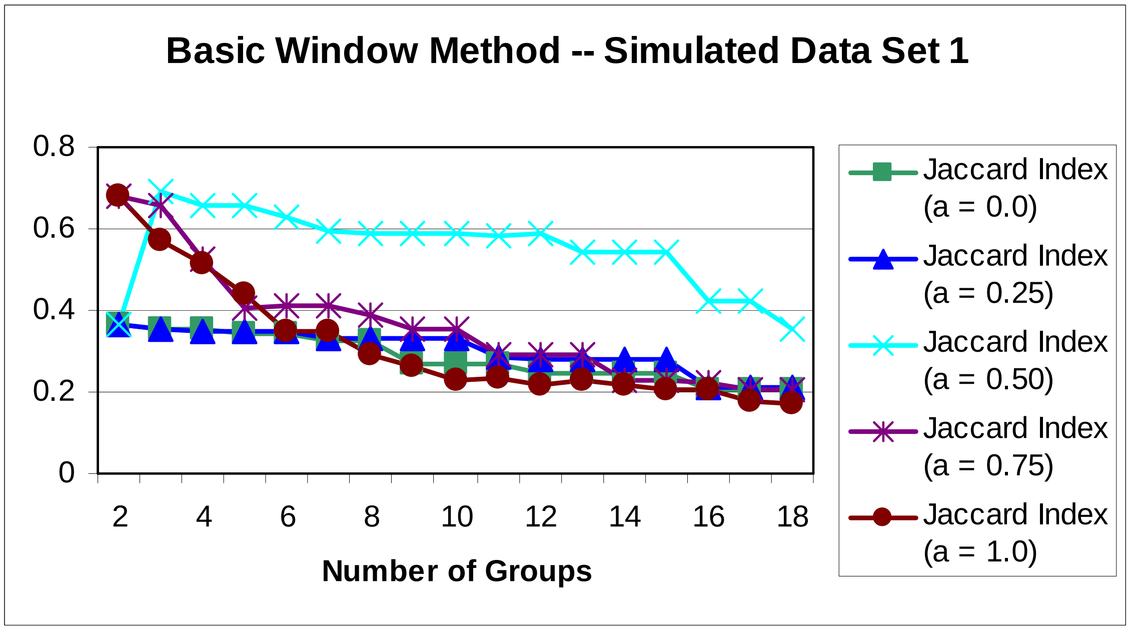


Figure 3.5 Basic Window Clustering for Simulated Data Set 1. The vertical axis shows the Jaccard Index as a function of the number of groups with various weights being given to the mean spatial structure and the spatial structure. The relative weight given to the mean structure is represented by *a*, with relative weight *1-a* given to the spatial structure.

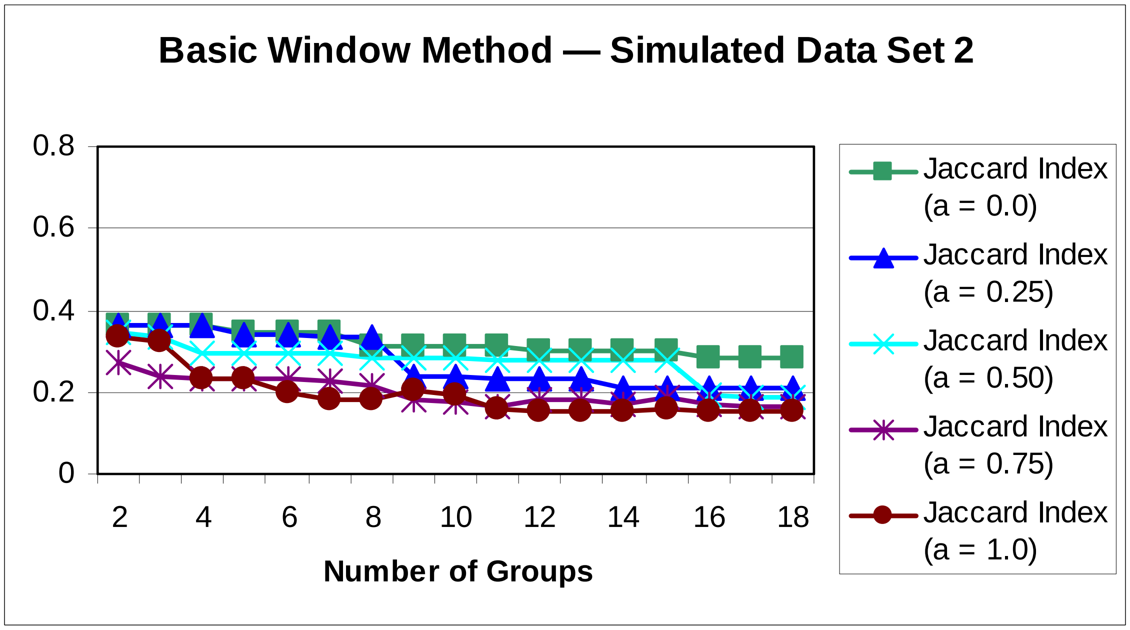


Figure 3.6 Basic Window Clustering for Simulated Data Set 2. The vertical axis shows the Jaccard Index as a function of the number of groups with various weights being given to the mean spatial structure and the spatial structure. The relative weight given to the mean structure is represented by *a*, with relative weight *1-a* given to the spatial structure.

3.2.2 Adjacent Constraint Method

As mentioned in the previous section, one of the difficulties with the Basic Window Method lies in the fact that it will often produce groups that fail to be spatially contiguous. The solution to this seems to be easy; one could simply increase the weights applied to the location index of $\mathbf{v}(\mathbf{s})$—$v_2(\mathbf{s})$ and $v_3(\mathbf{s})$. However, in order to achieve spatially contiguous groups, the location indices begin to have a greater effect than the other elements of the data vector. This results in groups that tend to reflect the four quadrants of the grid rather than producing groups of similar data points.

Instead of including the location indices of the data vector, the next proposed method—the Adjacent Constraint Method—utilizes the addition of an adjacency constraint. In order for two observations or groups to be clustered together, they must meet the requirement of being adjacent to each other. The data vectors for each position are constructed according to the method described in section 3.1.

The imposition of this constraint on the clustering method allow for the results to make sense in terms of the problem. Because the desired result of these clustering methods is to separate regions for the application of spatial analysis, the delineated regions must meet the required assumptions implicit in such analysis—one such assumption is that of contiguous regions. Legendre gives examples of constraints in ecological theory where the clusters are constrained to be contiguous; he then shows that constrained clustering methods always recover a larger fraction of the information when compared with the unconstrained methods (Legendre 1987).

After constructing $\mathbf{v}(\mathbf{s})$ for all $\mathbf{s}$ in the random field, the Adjacent Constraint Method compares the data vectors for all adjacent observations and then groups the two observations that are closest together. In this case, adjacency is determined by how far apart the two observations are located within the grid. Locations are considered to be adjacent when the Euclidean distance between the two observations is less than 1.5. For data sets that are set up in two-dimensional grids, this is equivalent to the eight nearest neighbors. The process continues combining data in the same manner until all the observations are combined into groups. Note that for an observation to be adjacent to a group of observations, the observation only needs to be adjacent to one of the observations in the group. Similarly, for two groups to be adjacent, at least one observation from the first group must be adjacent to at least one observation from the second group. This algorithm is similar to the one used by Huel, Petiot, and Lazar (1986).

In addition to the imposition of this adjacency constraint, the weights applied to the data vector, $\mathbf{v}(\mathbf{s})$, are also adjusted. Weights are applied to the data vector in two ways. Recall that the data vector may be split into two different sections—a section containing mean structure data points and a section containing spatial structure data points. Weights are first applied to these structures—a weight on the mean structure, represented by a, and a weight on the spatial structure, represented by $1-a$. Where a is constrained to be between 0 and 1.

The second way that weights are applied to the data vector is in determining the relative importance of data elements within the mean structure and spatial structure. These weights are not limited, but should be equal in total magnitude. Some care should be taken to determine these weights, as they have a large impact on the results. Recall from Chapter 2 that the data vector is of the form

$$\mathbf{v}(\mathbf{s}) = (v_1(\mathbf{s}), v_2(\mathbf{s}), v_3(\mathbf{s}), v_4(\mathbf{s}), v_5(\mathbf{s}), v_6(\mathbf{s}), v_7(\mathbf{s})).$$

The mean structure is reflected in the values ($v_1(\mathbf{s})$, $v_2(\mathbf{s})$, $v_3(\mathbf{s})$) and the spatial structure is reflected in the remaining elements, ($v_4(\mathbf{s})$, $v_5(\mathbf{s})$, $v_6(\mathbf{s})$, $v_7(\mathbf{s})$). The weights applied to the mean structure are 5, 0, and 0, respectively. Note that weights of zero are applied to the *x* and *y* coordinates, which effectively removes their effect on clustering. These effects are removed because the Adjacent Constraint Method of clustering forces groups to be contiguous, thereby removing the need to include location elements in the data vector. The weights applied to the variance structure are 2, 1, 1, and 1. This weight structure seeks to reflect the relative importance of each lag estimate on clustering. Essentially, the weights indicate that the most important data element for clustering based on a spatial perspective is the estimate of lag 1, followed by the estimate of the sill (using the 90th percentile of the lags as the estimate), then the estimates for lags 2 and 3.

As in the previous section, the effectiveness of the Adjacent Constraint Method may be evaluated by examining the Jaccard indexes at differing values of *a* and differing numbers of separated groups. The Jaccard index will generally decrease as the number of groups increase. With the Adjacent Constraint Method, however, the summary of Jaccard indices show a general increase up to a certain number of groups, and then the customary decrease (see Figures 3.7 and 3.8). This is especially evident for Simulated Data Set 1 in the case where $a = 0.25$, and can be seen for Simulated Data Set 2 with $a = 0.0$. As each Simulated Data Set contains only three "true" groups, this seems a little counterintuitive. The Jaccard indices reach a maximum at around 15 to 17 groups. Upon further examination, there appear to be many small groups of elements that are clustered along the borders of the natural groups. This phenomenon is examined in more detail in the following section. The hierarchical clustering produced by application of the Adjacent Constraint Method to Simulated Data Sets 1 and 2 can be found in Appendix A.

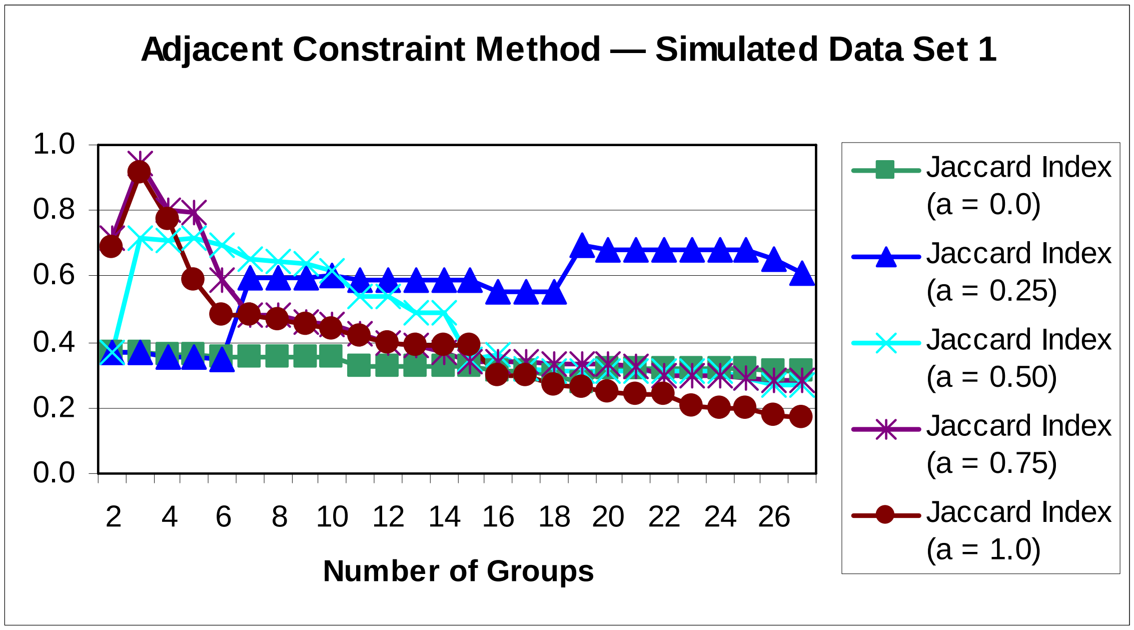


Figure 3.7 Adjacent Constraint Clustering Results for Simulated Data Set 1. The vertical axis shows the value of the Jaccard Index as a function of groups using the Adjacent Constraint Method. The relative weight given to the mean structure is *a* and the relative weight given to the spatial structure is *1-a*.

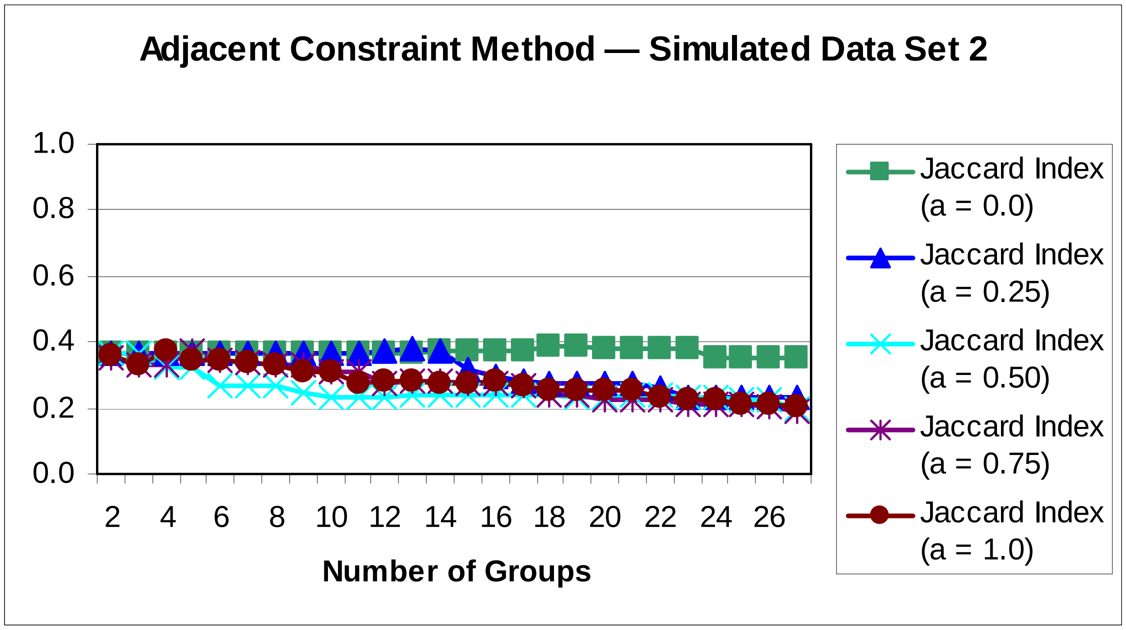


Figure 3.8 Adjacent Constraint Clustering Results for Simulated Data Set 2. The vertical axis shows the value of the Jaccard Index as a function of groups using the Adjacent Constraint Method. The relative weight given to the mean structure is *a* and the relative weight given to the spatial structure is *1-a*.

3.2.3 Two-Part Method

While using the two methods described above on the simulated data, the non-stationarity is clearly evident at the border of two different regions. This leads to estimated clusters that will be referred to as border clusters. When points reside along the natural borders between regions, the estimation of the variance and spatial dependence structure is highly affected by non-stationarity, or the vast difference between the natural regions. When near such regional borders, the window used in the previous methods (described in sections 3.2.1 and 3.2.2) spans the two regions and points from both regions are included in the estimation process. This leads to the artificial creation of small, often chain-like groups along the diagonal division between the three true groups. See Figure 3.9 for an illustration of such border blusters. The creation of border clusters is particularly evident when bordering regions have dramatically different mean levels for $\mathbf{Z}(\mathbf{s})$, yielding exaggerated values of $v_7(\mathbf{s})$ or $2\sigma^2(\mathbf{s})$. See, for example, the clusters formed where the known regional boundaries are located in Figure 3.7.

In order to account for the potential non-stationarity of the analyzed regions and in an attempt to prevent the creation of border clusters, the clustering process was split into two steps. The first step is to perform a cluster analysis only on $v_1(\mathbf{s})$—that is $\mathbf{Z}(\mathbf{s})$. As before, $v_2(\mathbf{s})$ and $v_3(\mathbf{s})$ (the row and column indices) are not included as the clustering is already subject to the data elements being adjacent. Since this initial clustering is an attempt to get better estimates of spatial structure, a stricter criterion of adjacent is used. To prevent the chaining of data elements (where a sequence of data elements only border each other diagonally), the physical distance must equal 1.0—only elements directly above, below, or to either side will be considered for clustering.

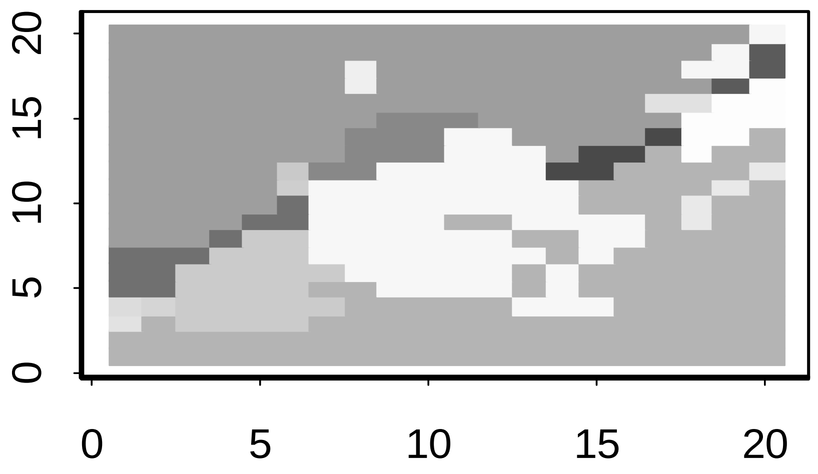


Figure 3.9 Border Clusters. This plot illustrates the artificial creation of groups on or near the borders of the natural group divisions.

This initial clustering is considered to be finished when each estimate of $v_4(\mathbf{s})$—lag 1—is close enough in relation to all other estimates of $v_4(\mathbf{s})$ within a specific region. The estimates are considered close enough if they fall within the central 99.9% of the range of a chi-squared distribution with a mean equal to the average estimate of $v_4(\mathbf{s})$ within the region. In essence, this method prevents the estimation of extreme values for $v_4(\mathbf{s})$ that result from differing mean structures between groups (as exhibited by border clusters), not from natural variation within a group. For this initial clustering, the less robust classical estimator of the variogram is used. Choosing a less robust estimator will allow for more extreme values of $v_4(\mathbf{s})$ where different underlying groups fall within a specific region, which this method seeks to identify.

Within these previously determined regions of significantly different mean structures, $\mathbf{Z}(\mathbf{s})$, the data elements are clustered by variance and spatial dependence structure using the Adjacent Constraint Method as described in Section 3.2.2. This method will have the greatest impact on data regions where there is significant fluctuation in the mean structure (as with Simulated Data Set 1). For regions where the mean structure is fairly constant, this method is expected to perform approximately the same as the Adjacent Constraint Method—in other words, all estimates of $v_4(\mathbf{s})$ may be assumed to fall within the chi-squared distribution with the required cutoff. These conclusions are supported by Figures 3.10 and 3.11. Figure 3.11 shows the Jaccard Index values for varying numbers of groups using the Two-Part Method on Simulated Data Set 1. The problem created by the border clusters is mitigated somewhat, but there is still evidence of such regions (for example, when $a = 0.25$). Figure 3.12 shows the Jaccard Index values for varying numbers of groups using the Two-Part Method on Simulated

Data Set 2. As expected, the results for Simulated Data Set 2 appear to be generally the same as those obtained with the Adjacent Constraint Method.

The hierarchical cluster results produced by application of the Two-Part Method to Simulated Data Set s 1 and 2 may be found in Aappendix A.

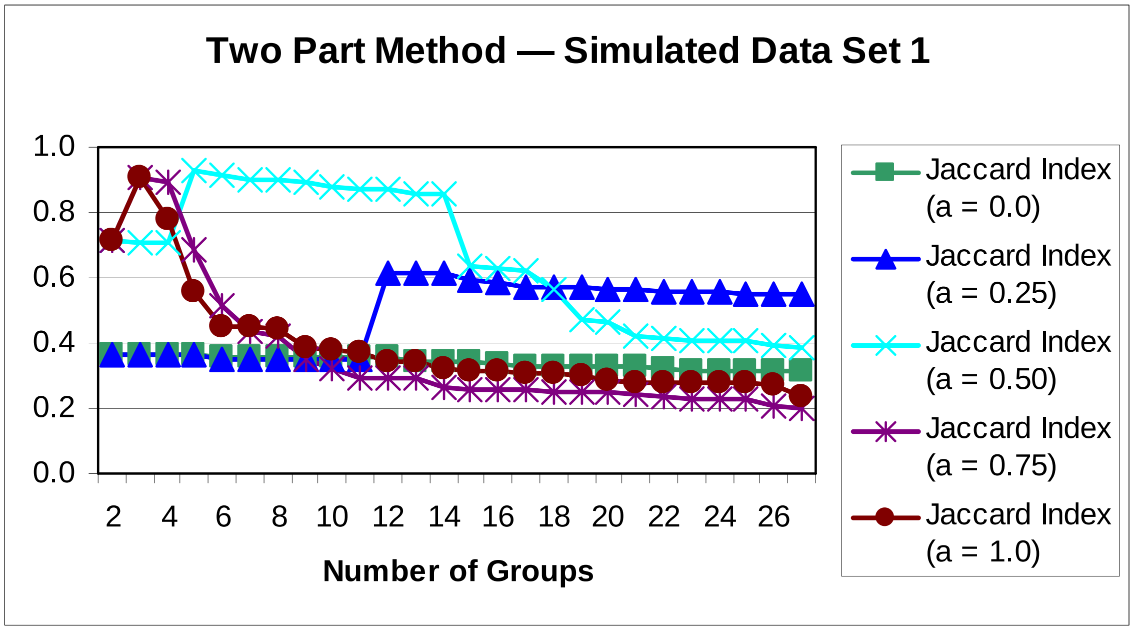


Figure 3.10 Two-Part Clustering Results for Simulated Data Set 1. The vertical axis shows the value of the Jaccard Index as a function of groups using the Two-Part Method. The relative weight given to the mean structure is *a* and the relative weight given to the spatial structure is *1-a*.

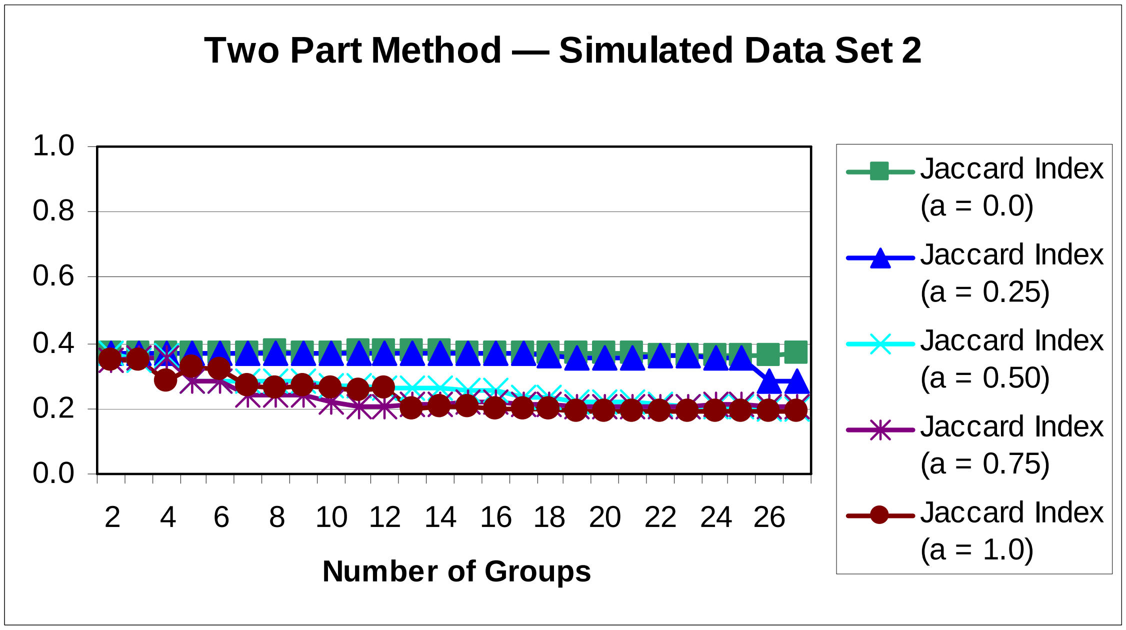


Figure 3.11 Two-Part Clustering Results for Simulated Data Set 2. The vertical axis shows the value of the Jaccard Index as a function of groups using the Two-Part Method. The relative weight given to the mean structure is *a* and the relative weight given to the spatial structure is *1-a*.

4. RESULTS

4.1 Simulated Examples

Each of the three previously described methods is used to cluster a series of simulated data sets. Four separate data sets were simulated with differing mean and covariance structures (as shown in Figures 4.1 and 4.2). Each data set has three different regions. The left side of the grid is one data region, the center is the second data region, and the right side is the third data region. The regions vary by mean and variance structure. Lower variance and higher range regions appear to be more smooth. The following provides a short description of each simulated data set's characteristics.

<u>Data Set 1:</u>
Region 1 exhibits high mean, moderate variance, and moderate range
Region 2 exhibits low mean, moderate variance, and moderate range
Region 3 exhibits moderate mean, moderate variance, and moderate range

<u>Data Set 2:</u>
Region 1 exhibits moderate mean, moderate variance, and moderate range
Region 2 exhibits moderate mean, small variance, and moderate range
Region 3 exhibits moderate mean, large variance, and moderate range

<u>Data Set 3:</u>
Region 1 exhibits moderate mean, moderate variance, and small range
Region 2 exhibits moderate mean, moderate variance, and large range
Region 3 exhibits moderate mean, moderate variance, and moderate range

<u>Data Set 4:</u>
Region 1 exhibits high mean, moderate variance, and small range
Region 2 exhibits low mean, small variance, and large range
Region 3 exhibits moderate mean, large variance, and moderate range

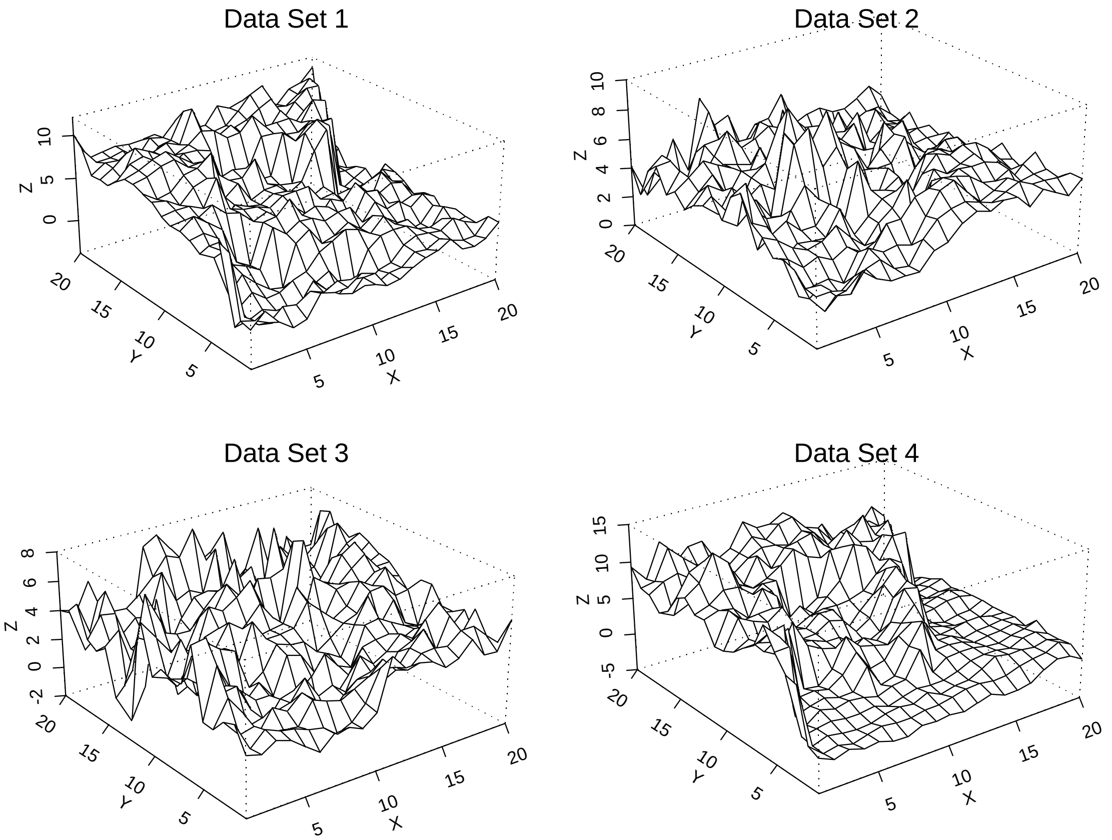


Figure 4.1 Sample Data Sets (Perspective Plot). Each data set has three different regions. The left side of the grid is one data region, the center is the second data region, and the right side is the third data region. The regions vary by mean and variance structure. Lower variance and higher range regions appear to be more smooth.

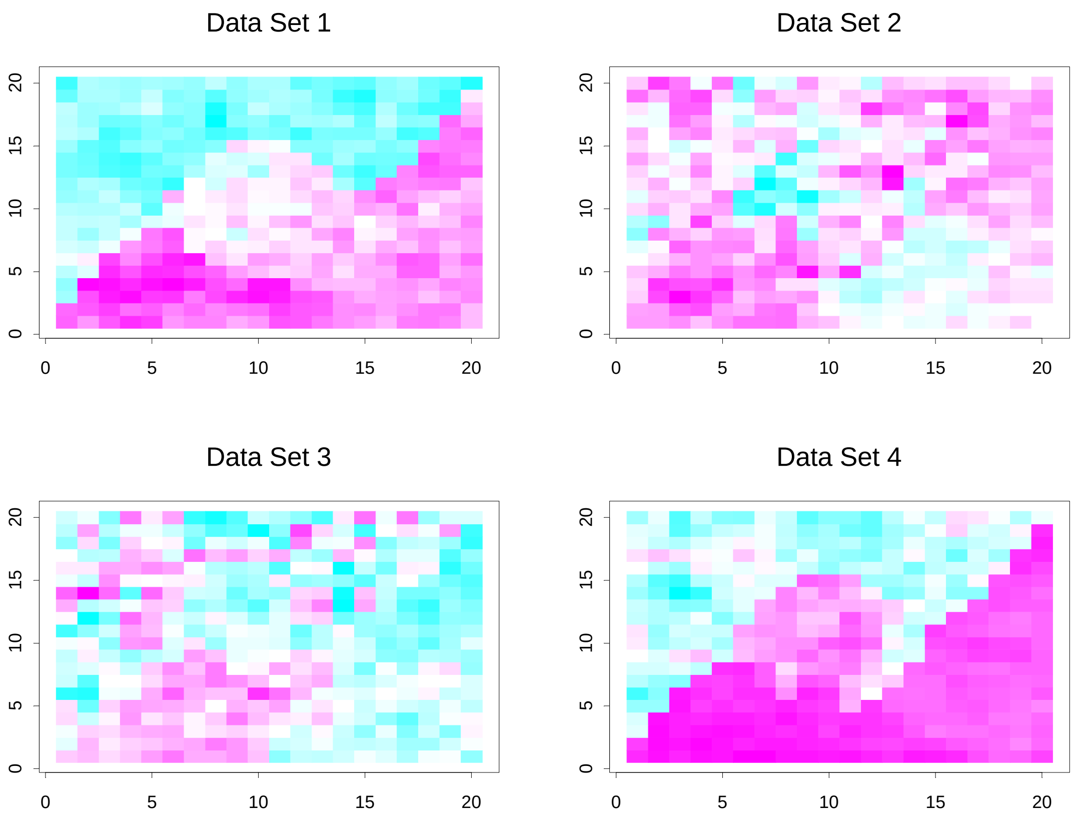


Figure 4.2 Sample Data Sets (Image View). This figure provides an alternate view of each of the data sets. Each data set has three different regions. The upper left portion of the grid is one data region, the center is the second, and the lower right portion is the third. The regions vary by mean and variance structure. Darker shades of purple represent low simulated values, the white values represent intermediate simulated values, and the darker shades of blue represent high simulated values.

The reasons for comparing and contrasting these three methods are as follows: 1) to determine how well the methods correctly identify the true groupings within each data sets—measured according to the Jaccard index value, and 2) to determine how useful the results will be in the view of performing spatial analysis on the resulting clusters.

4.2 Evaluation of Performance

The Jaccard index values resulting from the clustering methods proposed in this paper are shown in tables and charts in Appendix A (chapter 4 tables). Figure 4.3 presents a portion of these results for the case where $a = 0.50$. Different values of a were used for clustering the sample data sets. In most instances, however, a produced the highest values of the Jaccard index when set to 0.50.

An analysis of these results indicates that the Adjacent Constraint Method and the Two-Part Method perform at least as well as the Basic Window Method and the value of the Jaccard index is generally higher for these two methods. This result is consistent with the findings of Legendre (see Section 2.4) in that through imposing constraints on the clustering criteria more of the information is retained. It is evident that when delineating regions of spatial stationarity it is important to impose constraints upon the clustering criteria. These constraints will not only allow the clustering methods to retain more information, and thus perform better, but they provide the added benefit of resulting in clusters where the data points are spatially contiguous, as required for subsequent spatial analysis.

In general, the first groups that are identified through the proposed clustering techniques do not reflect the true groups. Initially, the regions that stand out represent small artificial groups found along the natural borders of the true groups. The Two-Part Method

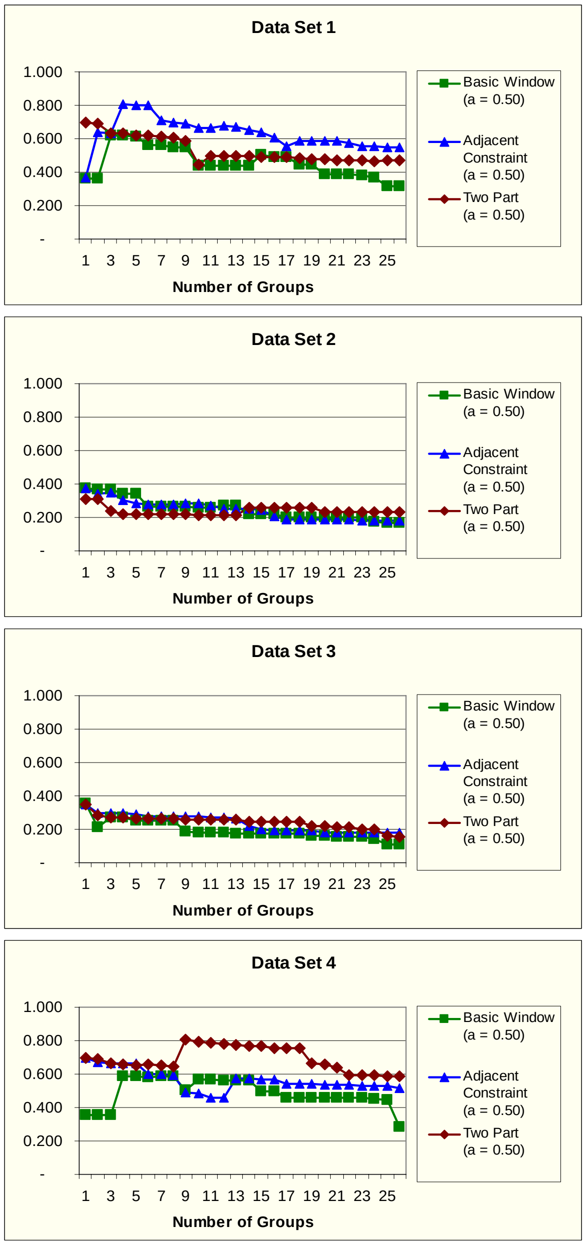


Figure 4.3 Sample Data Sets (Graphical Clustering Results *a* = 0.50). Graphical summary of the clustering results (based on the Jaccard Index) for the four simulated data sets using the Basic Window method, the Adjacent Constraint Method and the Two-Part Method. The vertical axis gives the Jaccard Index value as a function of the number of groups.

Data Set 2 Summary Statistics

	Group 1		Group 2		Group 3	
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Activation Level — $v_1(s)$	4.3	1.1	4.2	1.2	4.9	1.9
Lag 1 Estimate — $v_4(s)$	1.3	1.3	0.4	0.4	2.4	2.3
Lag 2 Estimate — $v_5(s)$	1.6	1.5	0.7	0.7	3.1	2.9
Lag 3 Estimate — $v_6(s)$	1.7	1.9	0.8	0.8	3.7	4.2
Estimate of 90th Percentile — $v_7(s)$	2.6	2.1	1.9	1.6	5.4	5.8

Figure 4.4 Summary statistics for Data Set 2. The relatively high standard deviations for the estimates of the spatial structure (with comparison to the average estimate values) make it difficult to accurately delineate the simulated regions within this data set.

mitigates this problem somewhat, but does not completely remove it. Figure 4.4 illustrates this for Data Set 4. Notice that the Two-Part Method does not completely solve the problem of the creation of small groups generally found along the natural borders. In fact, it is not until there are 10 groups that we see the division of all three true groups.

Because of this initial artificial border grouping issue, the Jaccard index starts off relatively low and then may exhibit large increases. Similarly, the final results generally indicate the presence of more groups than there actually are. In the simulated case, there are three actual groups, but Figures 4.3 and 4.4 indicate the presence of anywhere from five to ten groups.

The proposed clustering techniques perform better with regard to Data Sets 1 and 4. By construction, Data Sets 1 and 4 have differing mean structures across groups, whereas Data Sets 2 and 3 have the same mean structure with differing spatial structures. This suggests that the estimates of the variogram are inadequately reflecting differences in the spatial structure. (See Figure 4.4 for the summary statistics of Data Set 2). Recall that these methods employ a moving window approach. As a result, there are relatively few data points available for the estimates for each lag. For future studies, it would be worthwhile to investigate methods to improve these estimates, or to increase the number of data points used in estimation.

One possible way to improve the variogram estimation would be to calculate the variogram for the specified location using the methods proposed in this paper. Then, as a second step, calculate the variogram for each data point adjacent to the specified location (again using the methods proposed in this paper). Lastly, take the average of all the estimated variograms as the estimate of the variogram at the specified location. This method of estimation might also mitigate the impact of the border clusters by smoothing some of the extreme estimates of the variogram that are found along natural borders.

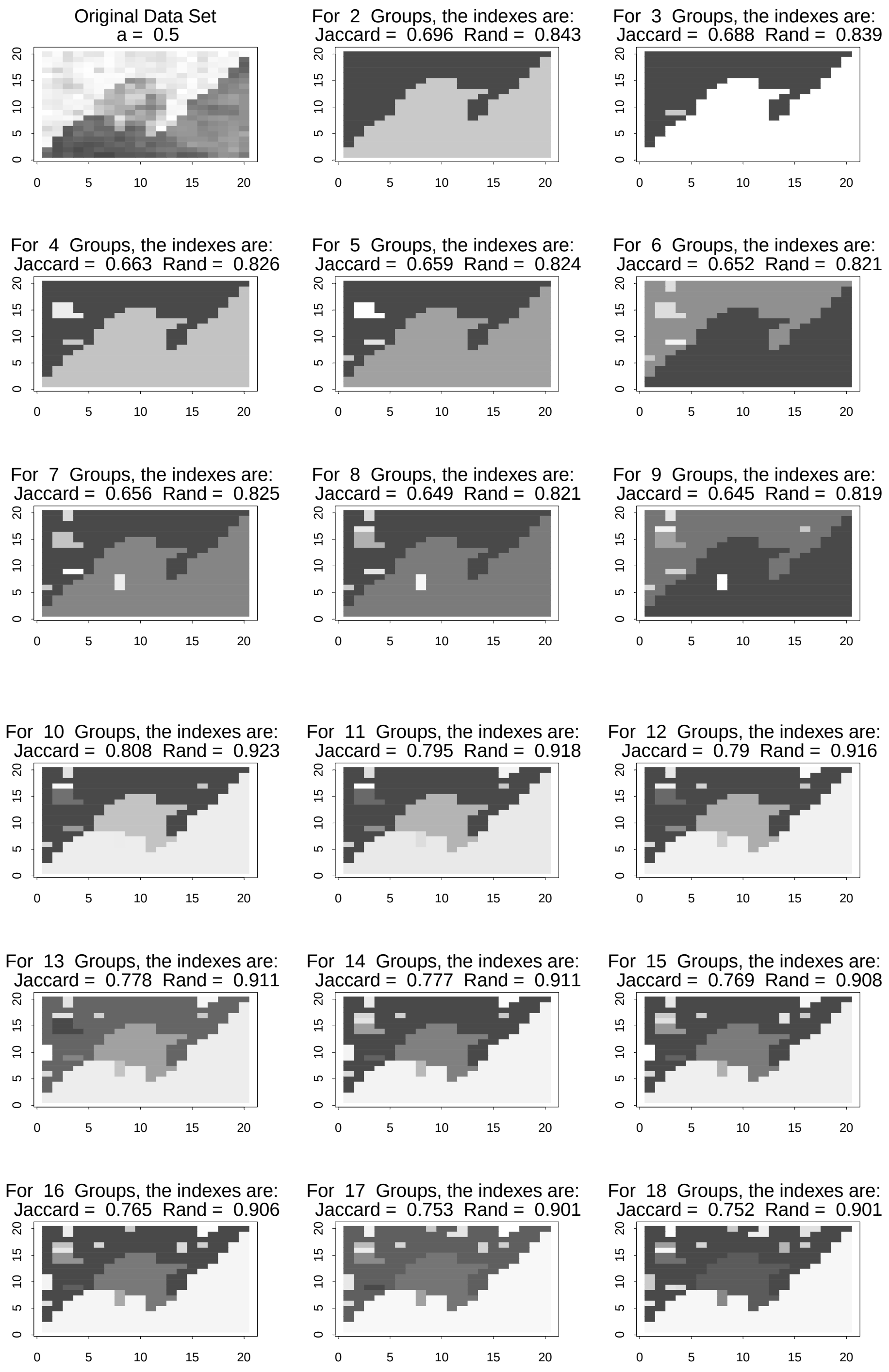


Figure 4.5 Graphical Clustering Results (Data Set 4). Graphical clustering results for Data Set 4 (with *a* = 0.50) for *n* groups where *n* = 2, 3, …, 18. The upper left plot shows the true groups.

From a practical standpoint and as described in previous sections, the Two-Part and the Adjacent Constraint Methods are superior to the Basic Window Method when the aim is to perform spatial analyses. To perform spatial statistical procedures on a data region, all of the data points within the region must be contiguous, as will be the case with the Two-Part Method and the Adjacent Constraint Method. This is not to say that the clustered groups produced using the Two-Part Method and the Adjacent Constraint Methods are necessarily ideal for spatial analysis. These two methods may produce groupings that include many groups with a small number of data points (border clusters). This necessitates careful judgment before performing spatial analysis; groups with small numbers of data points may need to be manually combined with neighboring groups.

5. SFMRI DATA ANALYSIS

5.1 Clustering of SFMRI Data

This section details the results of applying the proposed clustering techniques to SFMRI data. The data represent measurements of blood flow to areas of the brain (as described in chapter 2). Recall that measurements within the voxels are taken in the "on" (stimulated) and the "off" (non-stimulated) positions. Figure 5.1 shows the average blood flow for these two conditions within the defined voxels.

5.2 Evaluation of Clustering

Upon inspection of the SFMRI data set (see Figure 5.1), there is evidence of differences in blood flow across the various regions of the brain. These changes in mean structure indicate that increasing the weights applied to the mean structure might be appropriate. Because of this, clustering results for $a = 0.75$ are chosen as representative of the underlying groups (even though the clustering was done for all five values of a).

When evaluating the clusters produced as a result of applying the clustering techniques, it becomes apparent that each additional group does not necessarily add information to the clustering (See Figure 5.2). Note the groups added that appear to be mere border clusters. For example, groups 4 through 8 do not add any real value to the identification of the underlying regions. As more groups are split out, some contain very few data points.

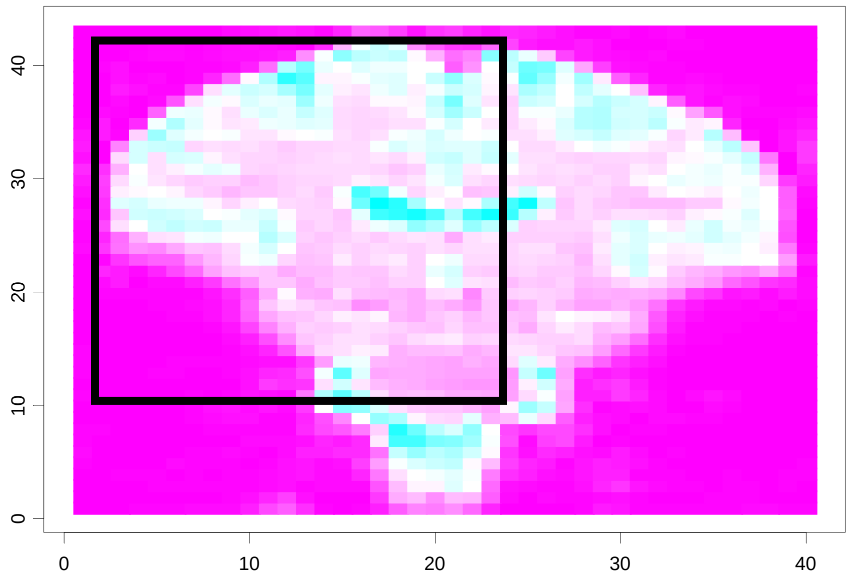


Figure 5.1 SFMRI Brain Data (Image Plot). This figure represents the average blood flow between the "on" and "off" conditions to the defined voxels of the brain. The highlighted area is the area to which the clustering techniques are applied. Darker shades of purple represent lower measurements of blood flow, the white values represent intermediate measurements of blood flow, and darker blue shades represent higher measurements of blood flow.

Also, as the larger groups emerge within the target region, these smaller groups are seen to be along the edges of, or between the larger groups—evidence that they are likely not true groups, but merely manifestations of the border cluster issue. Moreover, these groups also increase the number of groups that must be separated before all the underlying true groups show up in the cluster analysis.

It is also useful to compare results from both the Adjacent Constraint Method and Two-Part Methods. A representative sample of this information is presented in Figure 5.3, giving the image plots for all three clustering methods with $a = 0.75$ and 12 groups. Overall, there tends to be agreement in the location of the larger groups. In identifying the general underlying differences across regions of the brain, the small groups that are inconsistent across methods may be attributed to the border cluster issue. Also, the voxels outside the outline of the brain clearly do not reflect blood flow measures, and any clusters that show up there may be ignored, as is the case with the Basic Window Method where some groups are identified outside the measurement area. As was shown in Chapter 4, both of the proposed methods perform similarly in terms of accuracy. Thus, they may be used as a cross-check to determine which groups reflect underlying structure and which groups are likely to be border clusters.

These techniques as applied to the SFMRI data set enable the delineation of the different regions within the brain. However, it is insufficient to take the resulting groups straight into an analytic application. Care and judgment must be applied to ensure that the resulting groups are appropriate and reflective of underlying information. Any groups that are artificially split out should be rejoined to an adjacent group. As indicated, these groups may be identified through analysis of the cluster images as well as cross validation between clustering methods. The subsequent application of spatial statistical methods on the delineated regions should provide increased understanding of the structure within these distinct regions of the brain.

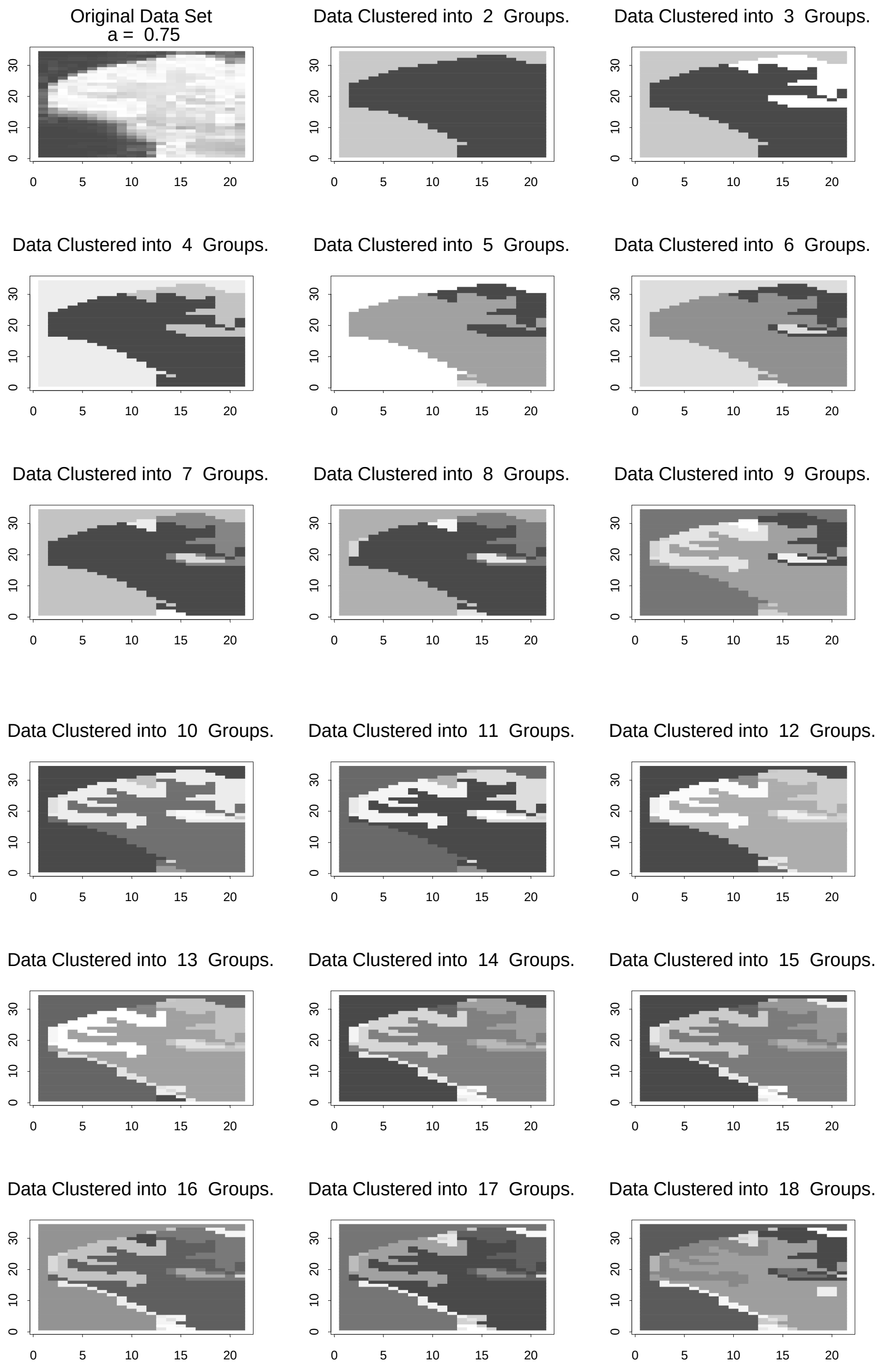


Figure 5.2 Graphical Clustering Results (Adjacent Constraint Method *a* = 0.75). Clustering results for the Adjacent Constraint Method with *a* = 0.75, for *n* groups with *n* = 2, 3, …, 18. The upper left plot shows the original data.

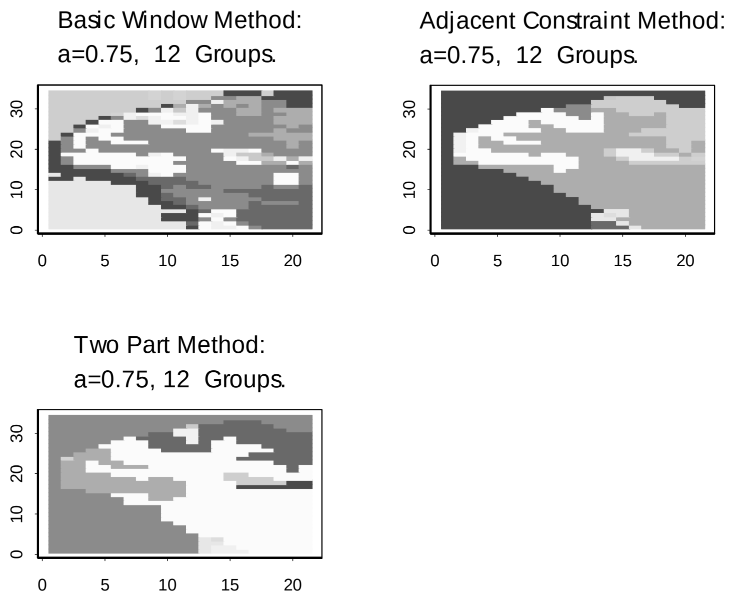


Figure 5.3 Clustering Results for the SFMRI Brain Data. Comparison of the three clustering techniques as applied to the SFMRI data with $a = 0.75$ and 12 groups.

6. CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

6.1 Conclusions

In general, the application of constraints greatly improved the usefulness of the clustering methods. The clustering of contiguous regions allows for further spatial analysis to take place. However, these methods still have difficulty in grouping all of the data elements into regions of sufficient size to perform this type of analysis. As with all clustering methods, the results of these methods are unable to produce definite results as to the correct number and placement of the true groups. Some judgment is required for the application of the clustering results.

Of the three methods presented, the Two-Part Method appears to be best equipped for delineating regions of spatial stationarity. This method is able to anticipate and account for potential problems due to non-stationarity of the mean structure, while still preserving the benefits gained from forcing grouped data elements to be adjacent to each other.

The last item of note is in regard to the weights applied to the data structure. Since the results must be viewed as indications of grouping rather than concrete evidence, it is useful to use various values of a. The weights must also be reasonable with regard to the data to be analyzed. If it can be assumed that the data field has similar mean structure, selection of a value of a that up-weights the variance components would be appropriate, whereas up-weighting the mean structure would not make as much sense.

6.2 Directions for Further Research

Further research options in this field exist primarily in two distinct areas: 1) how to improve the clustering of groups and analysis of how data regions are delineated, and 2) what to do with the data after the groups have been delineated.

With regards to clustering of the groups and the analysis of how the data regions are delineated, it would be useful to examine the extreme estimates of the variance structure more closely. In this paper, it was noted that one of the problems was the clustering and grouping of the border regions. If a method could be developed that could more accurately identify where border regions exist, one could divide a region based on where the most likely place for a border exists. Instead of trying to group data elements within a region, one could attempt to divide a region into homogenous groups.

One difficulty that exists with these methods is that they tend to produce various small clusters within the entire data region. A method to prevent or limit these small clusters would be very useful. Alternatively, one could examine how to recombine these small clusters with others in order to produce regions of sufficient size to perform additional spatial analysis.

Improvements to the clustering methods may also be realized through improved estimates of the variogram. As discussed in Chapter 4, there are various ways to increase the amount of data used to estimate the variogram for each location.

Another question of interest would be how to better identify regions with differing spatial structures. The approach of combining data points with the closest covariance structures seems to favor clustering results that produce many groups with very similar spatial structures rather than producing fewer groups with more variability in spatial structure between groups.

After data regions have been delineated, spatial statistical methods may be used for estimation within the region or space. With regards to the SFMRI data, these methods may be used to identify specifically which areas of the brain are associated with specific types of impulses and ultimately allow researchers to predict the physiological effects of such specified impulses or stimuli

BIBLIOGRAPHY

Anderson, N. H., and Titterington, D. M. (1997), "Some Methods for Investigating Spatial Clustering, with Epidemiological Applications," *Journal of the Royal Statistical SocietY, Series A (Statistics in Society)*, 160, 87-105.

Belliveau, J. W., Kennedy, D. N., McKinstry, R. C., Buchbinder, B. R., Weisskoff, R. M., Cohen, M. S., Vevea, J. M., Brady, T. J., and Rosen, B. R. (1991), "Functional Mapping of the Human Visual Cortex by Magnetic Resonance Imaging," *Science*, 254, 716-719.

Carvalho, M. S., Cruz, O. G., and Nobre, F. F. (1996), "Spatial Partitioning Using Multivariate Cluster Analysis and a Contiguity Algorithm," *Statistics in Medicine*, 15, 1885-1894.

Cressie, N., and Hawkins, D. M. (1980), "Robust Estimation of the Variogram, I." *Journal of the International Association for Mathematical Geology*, 12, 115-125.

Cuzick, J., and Edwards, R. (1990), "Spatial Clustering for Inhomogeneous Populations," *Journal of the Royal Statistical Society, Series B (Methodological)*, 52, 73-104.

Fisher, L., and Van Ness, J. W. (1971), "Admissible Clustering Procedures," *Biometrika*, 58, 91-104.

Genovese, Christopher R. (2000) "A Bayesian Time-Course Model for Functional Magnetic Resonance Imaging Data," *Journal of the American Statistical Association*, 95, 691-703.

Gordon, A. D. (1987), “A Review of Hierarchical Classification,” *Journal of the Royal Statistical Society. Series A (General)*, 150, 119-137.

Haas, T. C. (1990), “Lognormal and Moving Window Methods of Estimating Acid Deposition,” *Journal of the American Statistical Association*, 85, 950-963.

Huel, G., Petiot, J. G., and Lazar, P. (1986), “Algorithm for the Grouping of Contiguous Geographical Zones,” *Statistics in Medicine*, 5, 171-181.

Kwong, K. K., Belliveau, J. W., Chesler, D. A., Goldberg, I. E., Weisskoff, R. M., Poncelet, B. P., Kennedy, D. N., Hoppel, B. E., Cohen, M. S., Turner, R., Cheng, H., Brady, T. J., and Rosen, B. R. (1992), “Dynamic Magnetic Resonance Imaging of Human Brain Activity During Primary Sensory Stimulation,” *Proc. Natn. Acad. Sci.*, 89, 5675-5679.

Lange, N., and Zeger, S. L. (1997), “Non-Linear Fourier Time Series Analysis for Human Brain Mapping by Functional Magnetic Resonance Imaging,” *Applied Statistics*, 46, 1-29.

Legendre, P. (1987), “Constrained Clustering,” In *Developments in Numerical Ecology,* P. Legendre and L. Legendre, eds. Springer, Berlin, 289-307.

Marshall, R. J. (1991), “A Review of Methods for the Statistical Analysis of Spatial Patterns of Disease,” *Journal of the Royal Statistical Society, Series A (Statistics in Society)*, 154, 421-441.

Matheron, G. (1962), "*Traite de Geostatistique Appliquee, Tome* ***I****. Memoires du Bureau de Recherches Geologiques et Minieres*," **No. 14**, Editions Technip, Paris.

Milligan, G. W. (1983), “Characteristics of four external criterion measures,” *In* J. Felsenstein [ed.] Numerical Taxonomy. NATO Advanced Study Institute Series G (Ecological Sciences), No. 1. Springer-Verlag, Berlin, 167-173.

Owaga, S., Tank, D. W., Menon, R., Ellermann, J. M., Kim, S. G., Merkle, H., and Ugurbil, K. (1992), “Intrinsic Signal Changes Accompanying Sensory Stimulation: Functional Brain Mapping with Magnetic Resonance Imaging,” *Proc. Natl. Acad. Sci.*, 89, 5951-5955.

Raubertas, R. F. (1988), “Spatial and Temporal Analysis of Disease Occurrence for Detection of Clustering,” *Biometrics*, 44, 1121-1129.

Roy, C. S. and Sherrington, C. S. (1890), “The Regulation of the Blood-Supply of the Brain,” *J. Physiol. Lond.*, 11, 85-108.

Van Ness, J. W. (1973), “Admissible Clustering Procedures,” *Biometrika*, 60, 422-424.

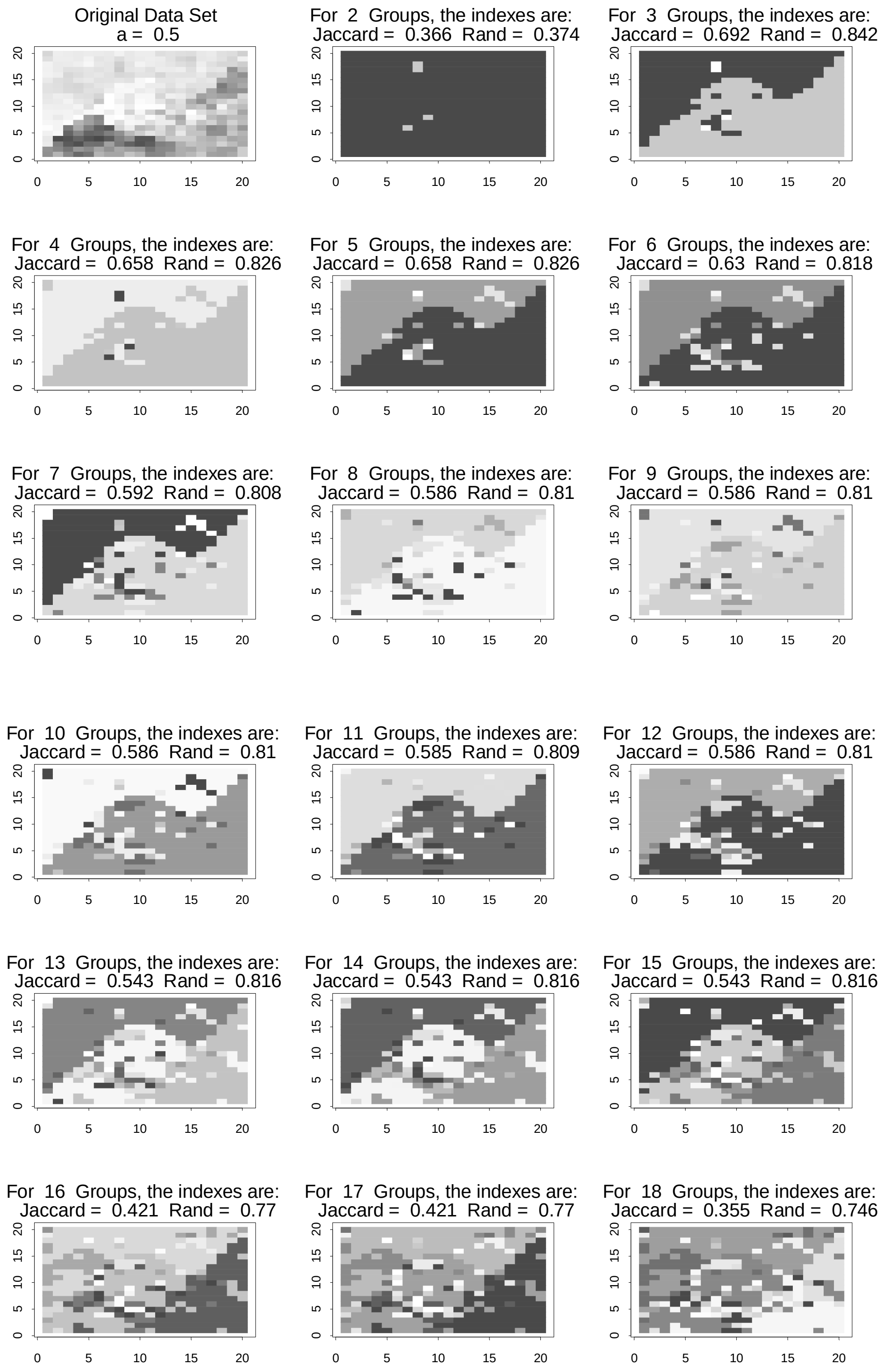
A. HIERARCHICAL CLUSTER IMAGES

A.1 Graphical Examples

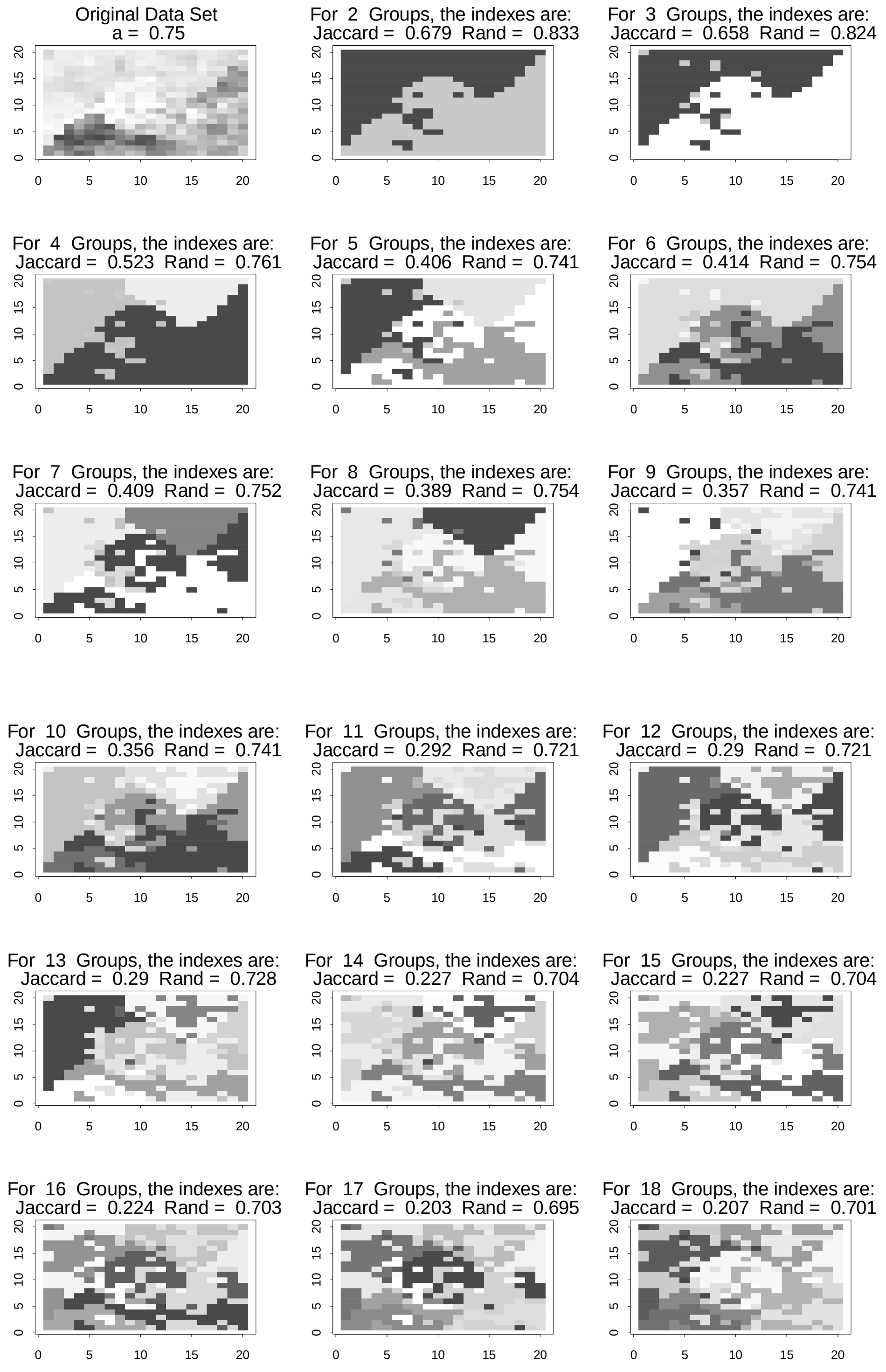
The following images provide a selection of clustering resulting from applying the three proposed methods to the two data examples from Chapter 3.Each chart indicates the method used as well as the weight used for *a*.

Basic Window Method—Simulated Data Set 1

Simulated Data Set 1 Basic Window Method (*a* = 0.50)

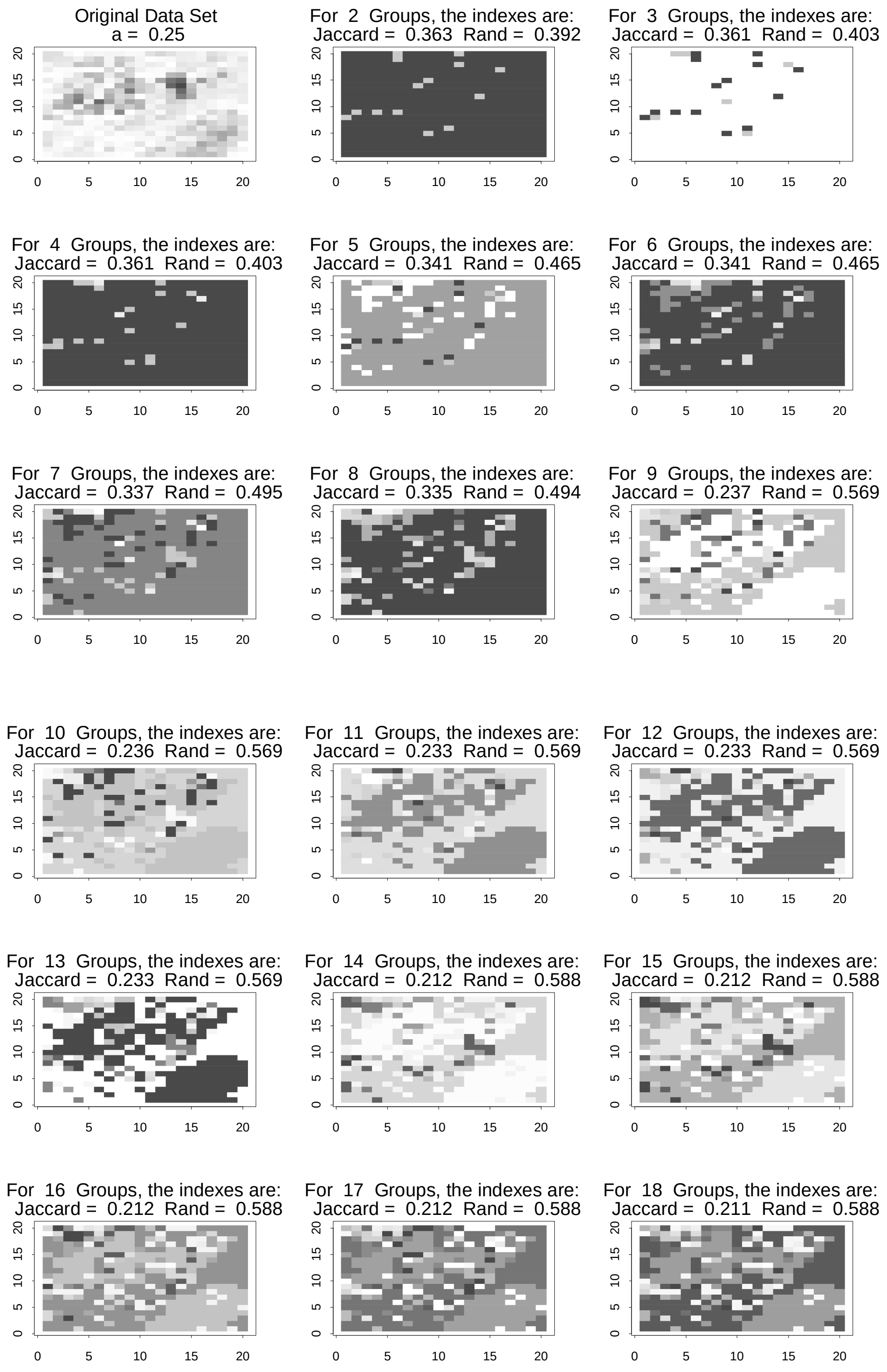


Simulated Data Set 1 Basic Window Method ($a = 0.75$)

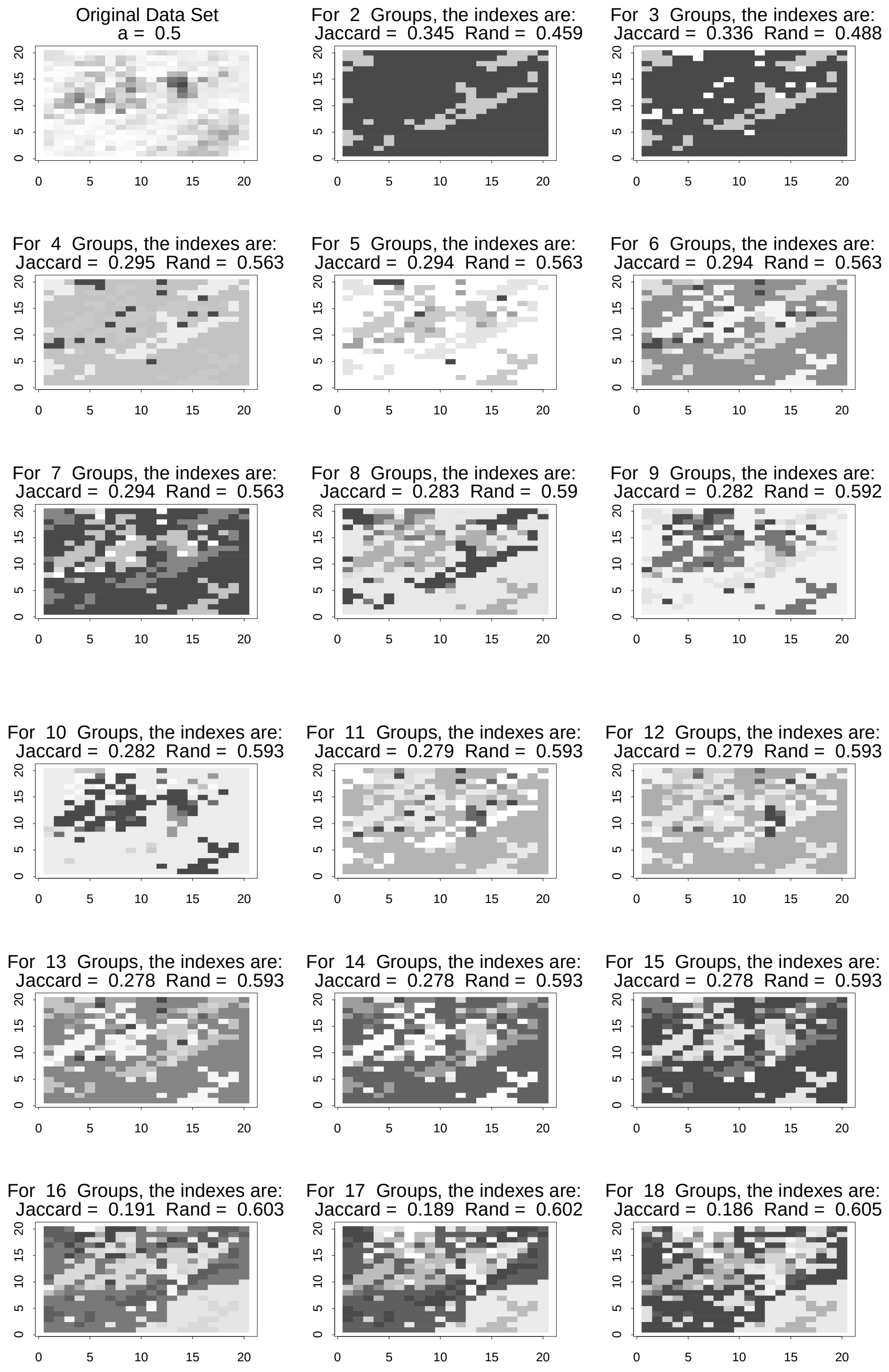


Basic Window Method— Simulated Data Set 2

Simulated Data Set 2 Basic Window Method (a = 0.25)



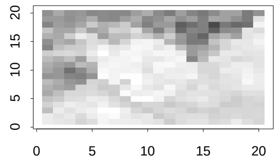
Simulated Data Set 2 Basic Window Method (a = 0.50)



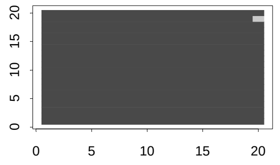
Adjacent Constraint Method— Simulated Data Set 1

Simulated Data Set 1 Adjacent Constraint Method (*a* = 0.50)

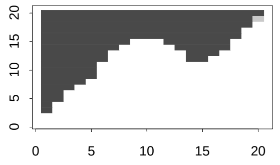
Original Data Set
a = 0.5



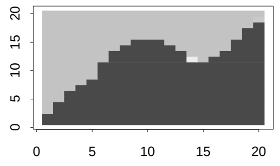
For 2 Groups, the indexes are:
Jaccard = 0.367 Rand = 0.369



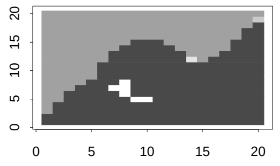
For 3 Groups, the indexes are:
Jaccard = 0.713 Rand = 0.853



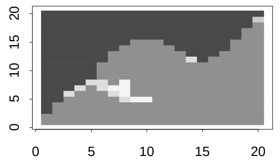
For 4 Groups, the indexes are:
Jaccard = 0.709 Rand = 0.851



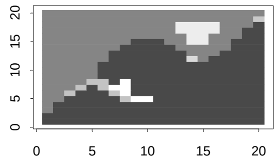
For 5 Groups, the indexes are:
Jaccard = 0.717 Rand = 0.858



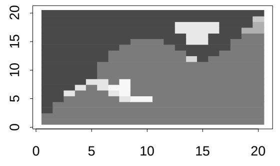
For 6 Groups, the indexes are:
Jaccard = 0.698 Rand = 0.85



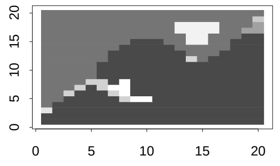
For 7 Groups, the indexes are:
Jaccard = 0.652 Rand = 0.827



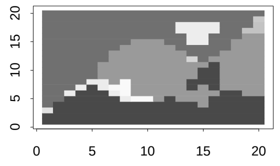
For 8 Groups, the indexes are:
Jaccard = 0.643 Rand = 0.824



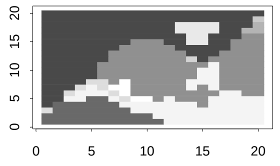
For 9 Groups, the indexes are:
Jaccard = 0.639 Rand = 0.822



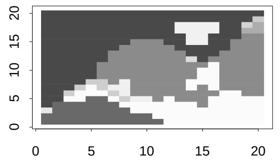
For 10 Groups, the indexes are:
Jaccard = 0.615 Rand = 0.843



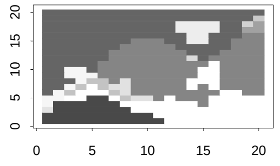
For 11 Groups, the indexes are:
Jaccard = 0.54 Rand = 0.812



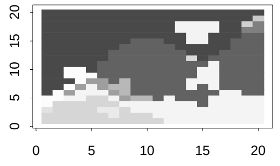
For 12 Groups, the indexes are:
Jaccard = 0.536 Rand = 0.811



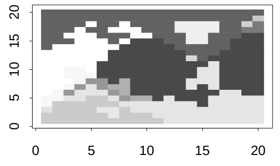
For 13 Groups, the indexes are:
Jaccard = 0.489 Rand = 0.791



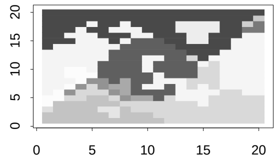
For 14 Groups, the indexes are:
Jaccard = 0.486 Rand = 0.79



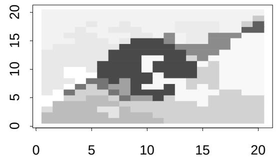
For 15 Groups, the indexes are:
Jaccard = 0.35 Rand = 0.735



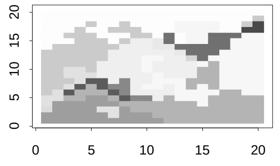
For 16 Groups, the indexes are:
Jaccard = 0.361 Rand = 0.761



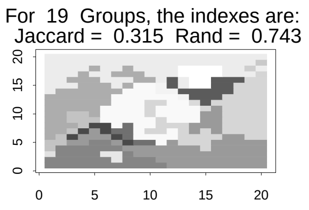
For 17 Groups, the indexes are:
Jaccard = 0.322 Rand = 0.746

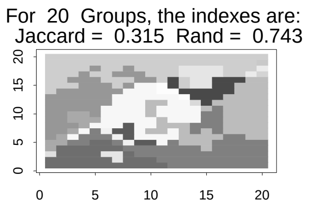


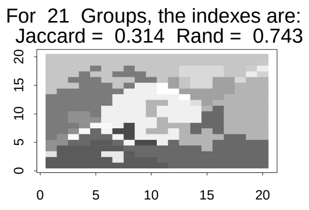
For 18 Groups, the indexes are:
Jaccard = 0.315 Rand = 0.743

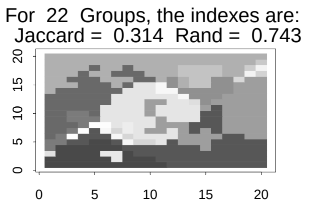


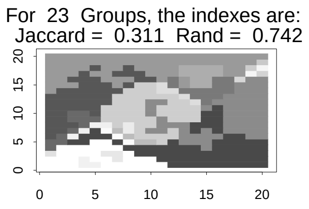
Simulated Data Set 1 Adjacent Constraint Method (*a* = 0.50) (continued)

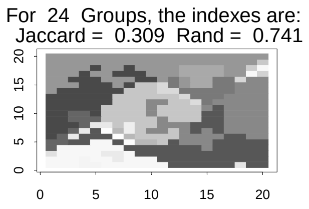


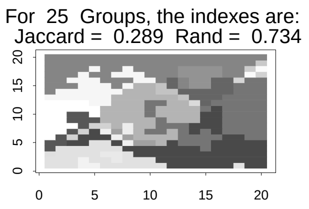


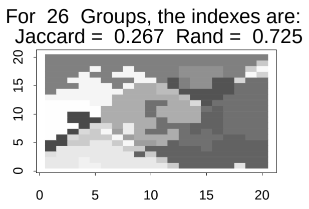


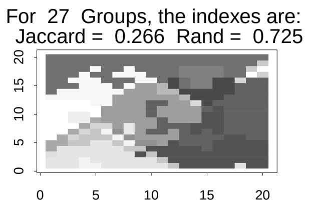




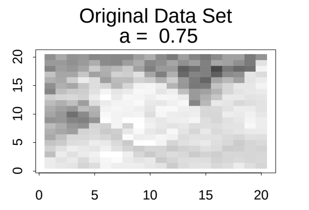


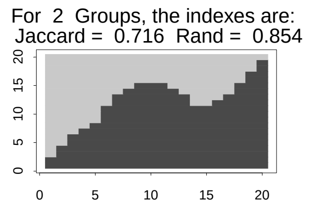


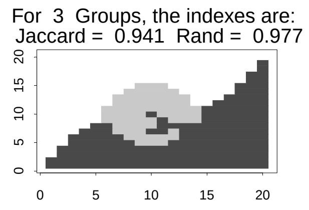


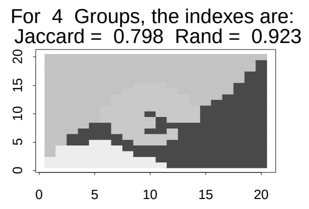


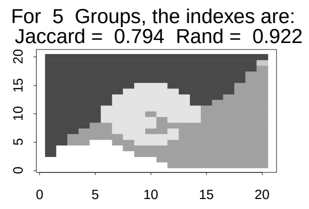
Simulated Data Set 1 Adjacent Constraint Method (*a* = 0.75)

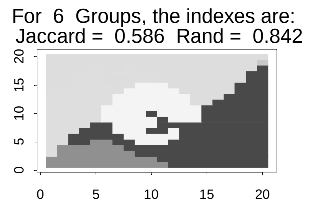


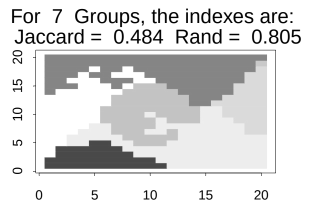


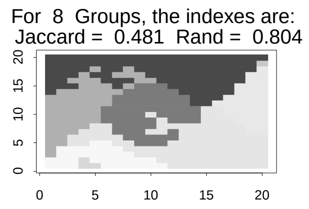


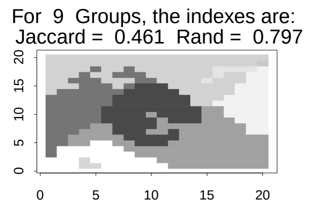






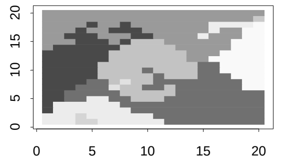




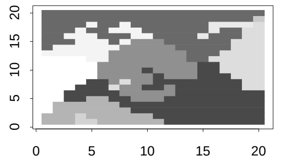


Simulated Data Set 1 Adjacent Constraint Method (a = 0.75) (continued)

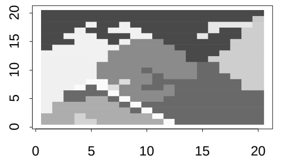
For 10 Groups, the indexes are: Jaccard = 0.457 Rand = 0.795



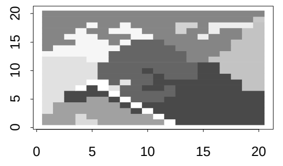
For 11 Groups, the indexes are: Jaccard = 0.426 Rand = 0.784



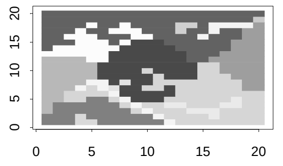
For 12 Groups, the indexes are: Jaccard = 0.4 Rand = 0.774



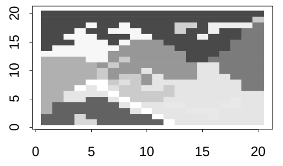
For 13 Groups, the indexes are: Jaccard = 0.391 Rand = 0.771



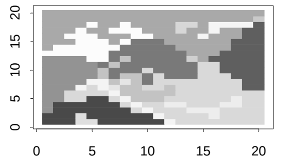
For 14 Groups, the indexes are: Jaccard = 0.37 Rand = 0.763



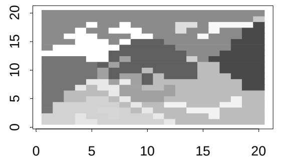
For 15 Groups, the indexes are: Jaccard = 0.343 Rand = 0.753



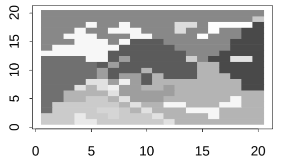
For 16 Groups, the indexes are: Jaccard = 0.34 Rand = 0.752



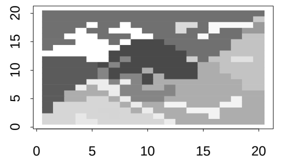
For 17 Groups, the indexes are: Jaccard = 0.337 Rand = 0.751



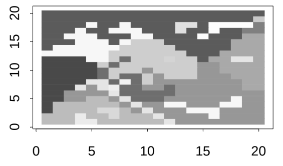
For 18 Groups, the indexes are: Jaccard = 0.335 Rand = 0.75



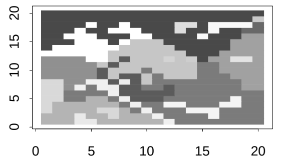
For 19 Groups, the indexes are: Jaccard = 0.332 Rand = 0.749



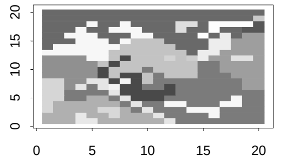
For 20 Groups, the indexes are: Jaccard = 0.33 Rand = 0.749



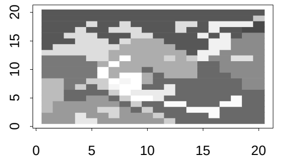
For 21 Groups, the indexes are: Jaccard = 0.323 Rand = 0.746



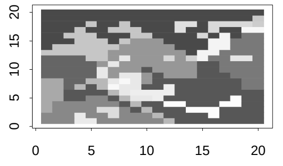
For 22 Groups, the indexes are: Jaccard = 0.3 Rand = 0.737



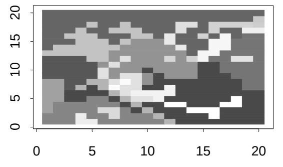
For 23 Groups, the indexes are: Jaccard = 0.297 Rand = 0.736



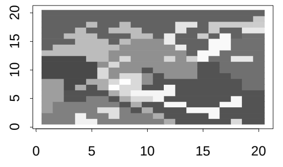
For 24 Groups, the indexes are: Jaccard = 0.297 Rand = 0.736



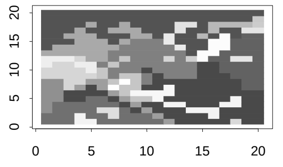
For 25 Groups, the indexes are: Jaccard = 0.289 Rand = 0.733



For 26 Groups, the indexes are: Jaccard = 0.287 Rand = 0.732

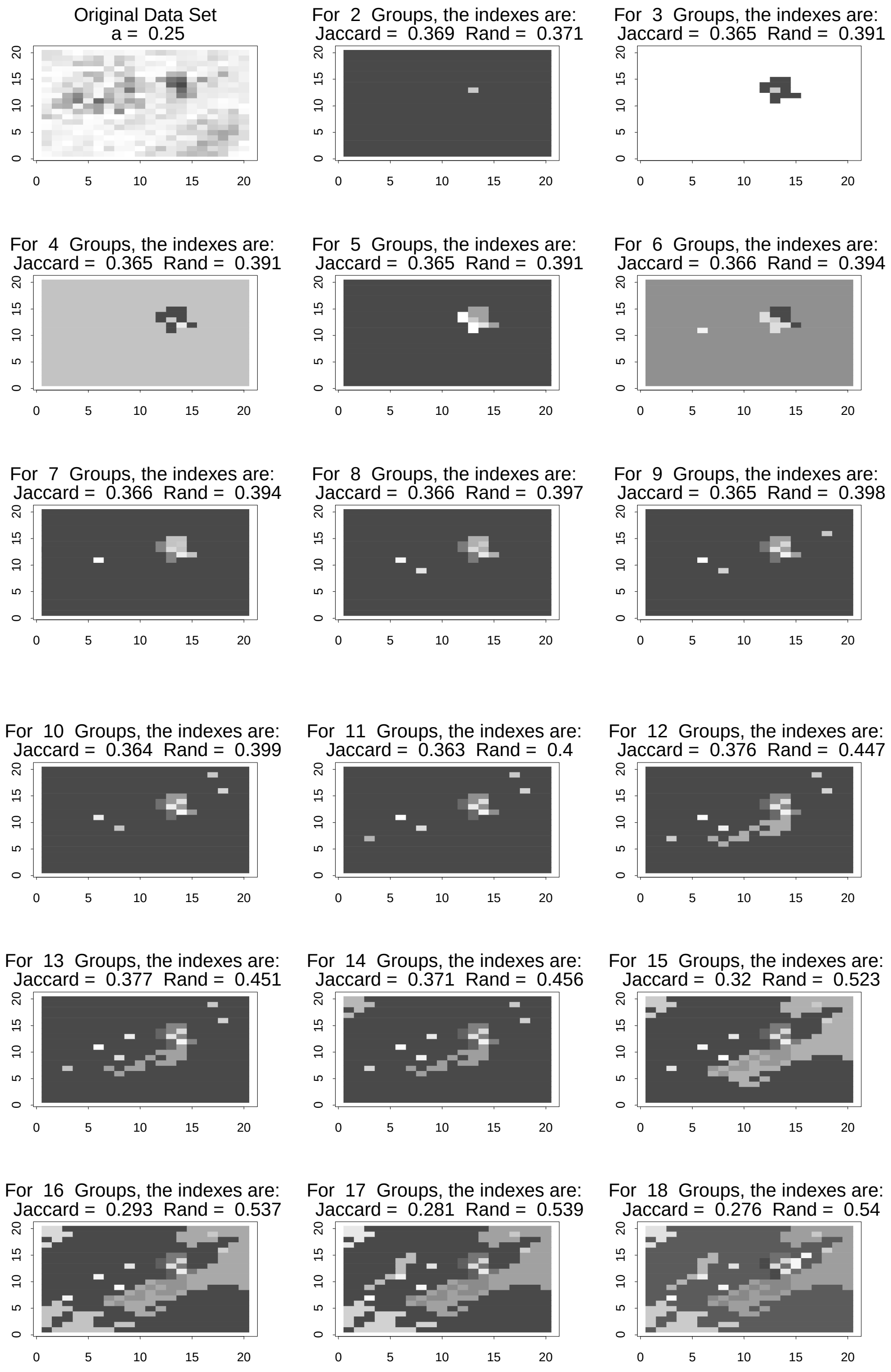


For 27 Groups, the indexes are: Jaccard = 0.283 Rand = 0.731



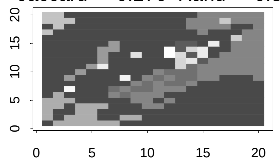
Adjacent Constraint Method— Simulated Data Set 2

Simulated Data Set 2 Adjacent Constraint Method (a = 0.25)

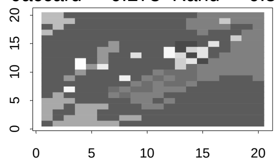


Simulated Data Set 2 Adjacent Constraint Method (*a* = 0.25) (continued)

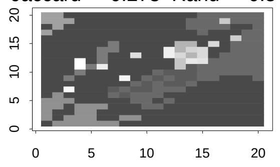
For 19 Groups, the indexes are: Jaccard = 0.276 Rand = 0.54



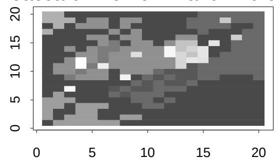
For 20 Groups, the indexes are: Jaccard = 0.273 Rand = 0.541



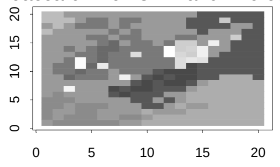
For 21 Groups, the indexes are: Jaccard = 0.273 Rand = 0.541



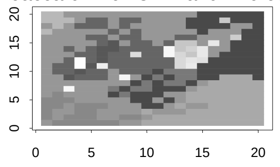
For 22 Groups, the indexes are: Jaccard = 0.262 Rand = 0.612



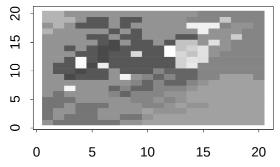
For 23 Groups, the indexes are: Jaccard = 0.232 Rand = 0.652



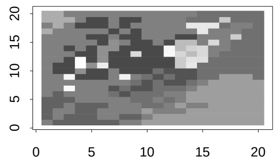
For 24 Groups, the indexes are: Jaccard = 0.231 Rand = 0.652



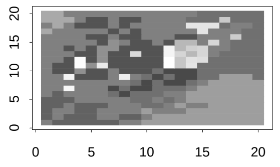
For 25 Groups, the indexes are: Jaccard = 0.232 Rand = 0.654



For 26 Groups, the indexes are: Jaccard = 0.231 Rand = 0.654

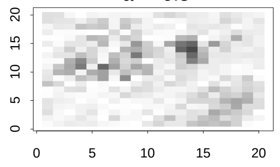


For 27 Groups, the indexes are: Jaccard = 0.23 Rand = 0.654

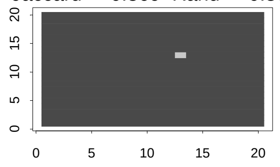


Simulated Data Set 2 Adjacent Constraint Method (*a* = 0.50)

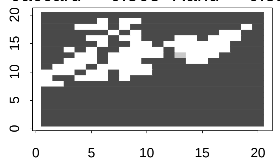
Original Data Set a = 0.5



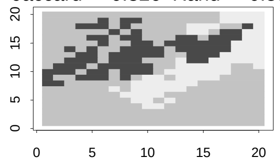
For 2 Groups, the indexes are: Jaccard = 0.369 Rand = 0.371



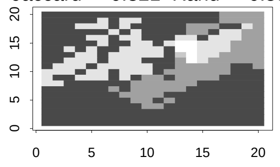
For 3 Groups, the indexes are: Jaccard = 0.363 Rand = 0.515



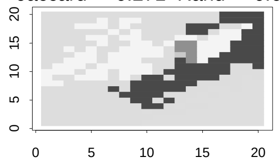
For 4 Groups, the indexes are: Jaccard = 0.326 Rand = 0.586



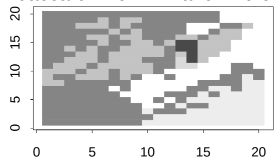
For 5 Groups, the indexes are: Jaccard = 0.321 Rand = 0.584



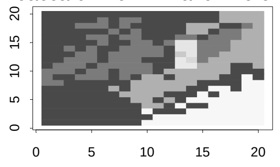
For 6 Groups, the indexes are: Jaccard = 0.271 Rand = 0.61



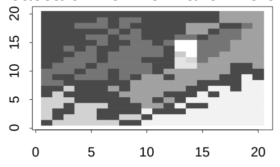
For 7 Groups, the indexes are: Jaccard = 0.27 Rand = 0.611



For 8 Groups, the indexes are: Jaccard = 0.27 Rand = 0.611

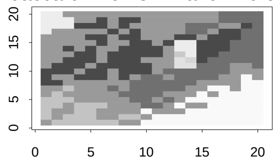


For 9 Groups, the indexes are: Jaccard = 0.246 Rand = 0.622

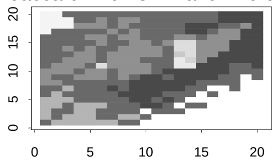


Simulated Data Set 2 Adjacent Constraint Method ($a = 0.50$) (continued)

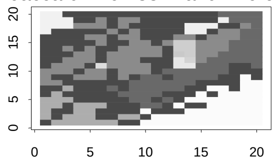
For 10 Groups, the indexes are:
Jaccard = 0.232 Rand = 0.621



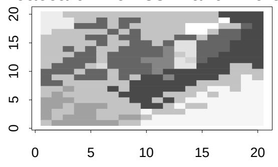
For 11 Groups, the indexes are:
Jaccard = 0.232 Rand = 0.621



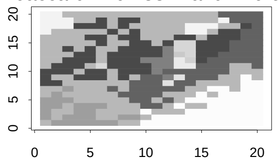
For 12 Groups, the indexes are:
Jaccard = 0.233 Rand = 0.625



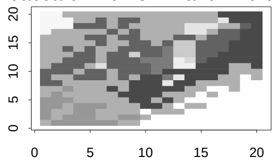
For 13 Groups, the indexes are:
Jaccard = 0.238 Rand = 0.64



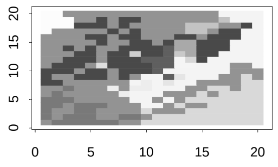
For 14 Groups, the indexes are:
Jaccard = 0.238 Rand = 0.64



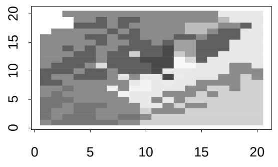
For 15 Groups, the indexes are:
Jaccard = 0.237 Rand = 0.641



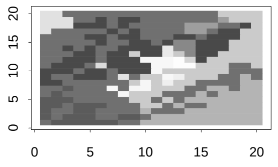
For 16 Groups, the indexes are:
Jaccard = 0.236 Rand = 0.642



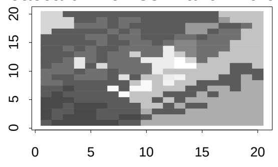
For 17 Groups, the indexes are:
Jaccard = 0.236 Rand = 0.642



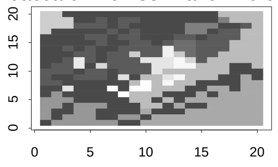
For 18 Groups, the indexes are:
Jaccard = 0.236 Rand = 0.642



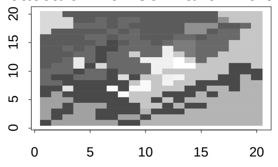
For 19 Groups, the indexes are:
Jaccard = 0.233 Rand = 0.641



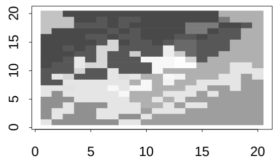
For 20 Groups, the indexes are:
Jaccard = 0.233 Rand = 0.642



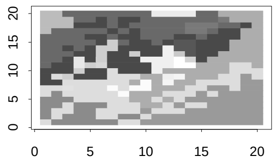
For 21 Groups, the indexes are:
Jaccard = 0.236 Rand = 0.685



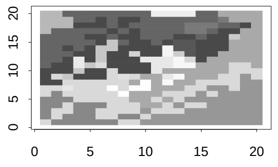
For 22 Groups, the indexes are:
Jaccard = 0.238 Rand = 0.691



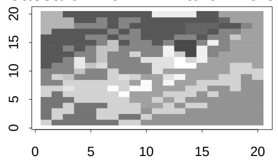
For 23 Groups, the indexes are:
Jaccard = 0.23 Rand = 0.688



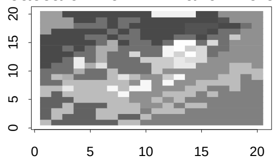
For 24 Groups, the indexes are:
Jaccard = 0.229 Rand = 0.688



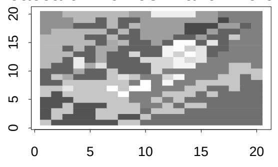
For 25 Groups, the indexes are:
Jaccard = 0.224 Rand = 0.687



For 26 Groups, the indexes are:
Jaccard = 0.224 Rand = 0.687

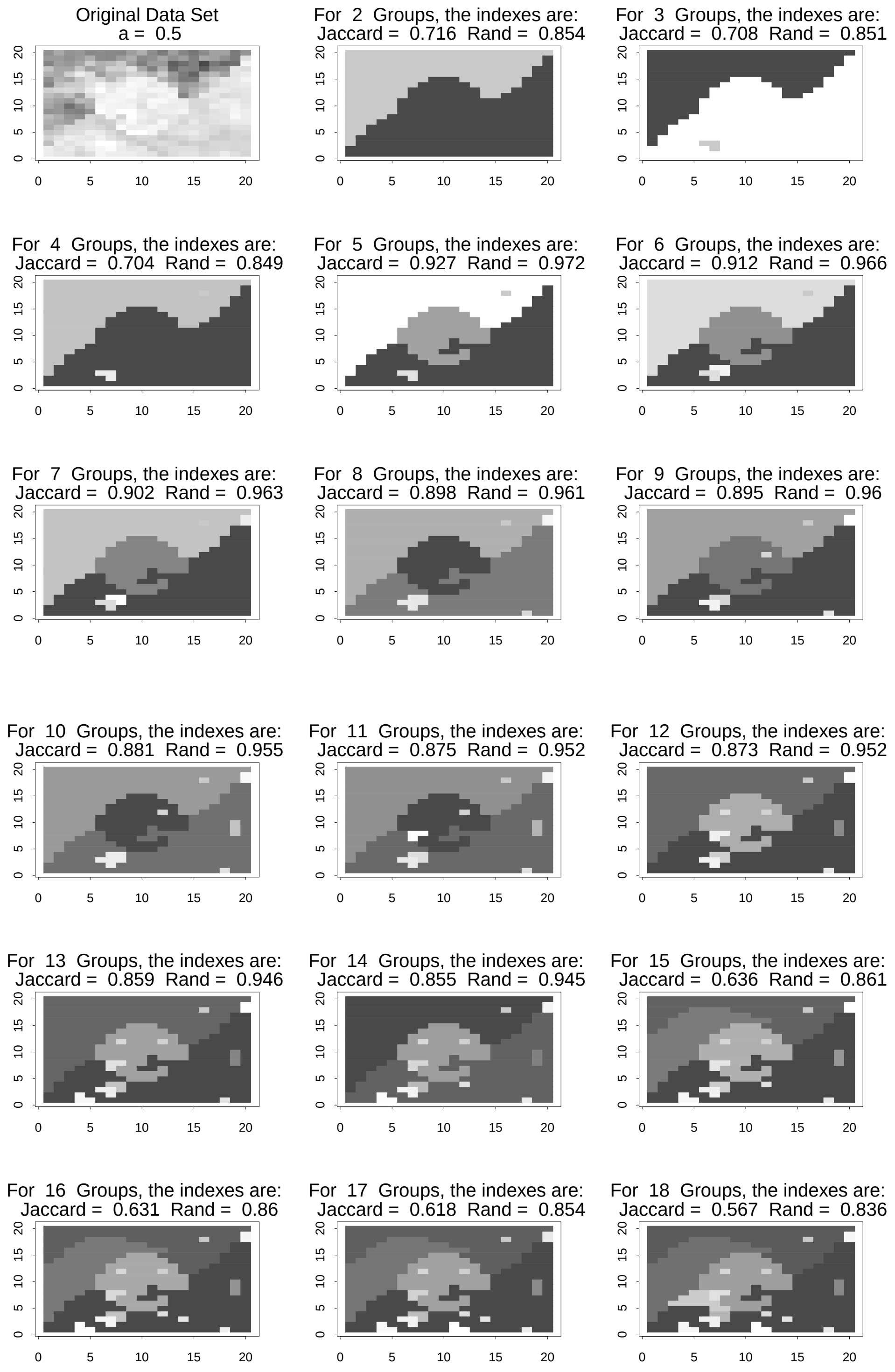


For 27 Groups, the indexes are:
Jaccard = 0.195 Rand = 0.675



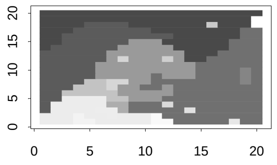
Two-Part Method— Simulated Data Set 1

Simulated Data Set 1 Two-Part Method (*a* = 0.50)

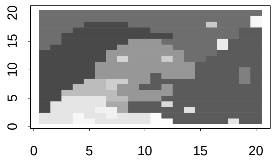


Simulated Data Set 1 Two-Part Method (*a* = 0.50) (continued)

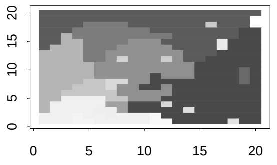
For 19 Groups, the indexes are:
Jaccard = 0.472 Rand = 0.801



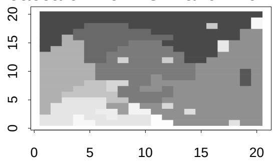
For 20 Groups, the indexes are:
Jaccard = 0.466 Rand = 0.798



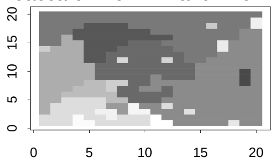
For 21 Groups, the indexes are:
Jaccard = 0.423 Rand = 0.782



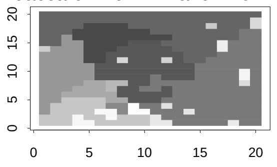
For 22 Groups, the indexes are:
Jaccard = 0.413 Rand = 0.779



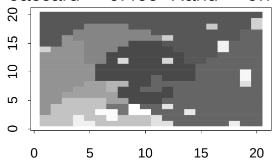
For 23 Groups, the indexes are:
Jaccard = 0.41 Rand = 0.778



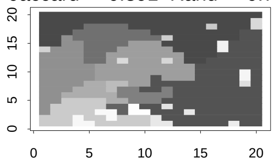
For 24 Groups, the indexes are:
Jaccard = 0.41 Rand = 0.778



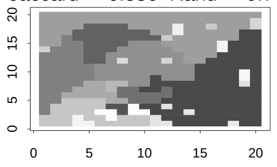
For 25 Groups, the indexes are:
Jaccard = 0.409 Rand = 0.777



For 26 Groups, the indexes are:
Jaccard = 0.391 Rand = 0.771

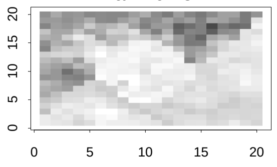


For 27 Groups, the indexes are:
Jaccard = 0.386 Rand = 0.768

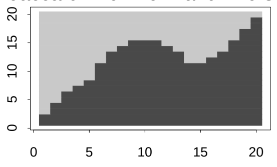


Simulated Data Set 1 Two-Part Method (*a* = 0.75)

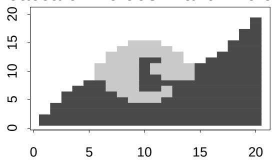
Original Data Set
a = 0.75



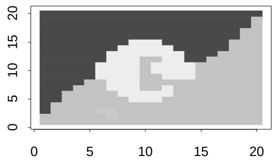
For 2 Groups, the indexes are:
Jaccard = 0.716 Rand = 0.854



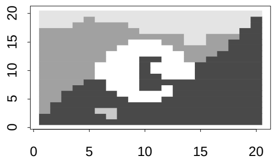
For 3 Groups, the indexes are:
Jaccard = 0.908 Rand = 0.964



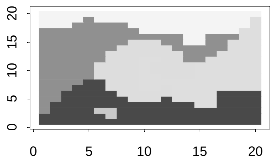
For 4 Groups, the indexes are:
Jaccard = 0.894 Rand = 0.958



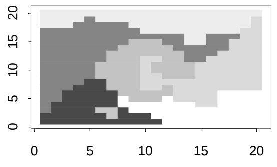
For 5 Groups, the indexes are:
Jaccard = 0.688 Rand = 0.877



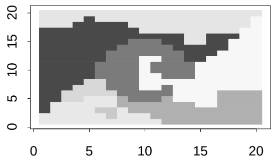
For 6 Groups, the indexes are:
Jaccard = 0.512 Rand = 0.815



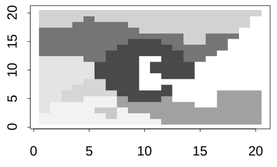
For 7 Groups, the indexes are:
Jaccard = 0.433 Rand = 0.785



For 8 Groups, the indexes are:
Jaccard = 0.42 Rand = 0.78

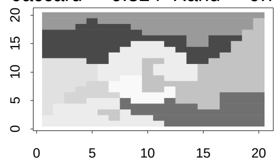


For 9 Groups, the indexes are:
Jaccard = 0.347 Rand = 0.753

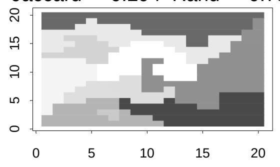


Simulated Data Set 1 Two-Part Method (*a* = 0.75) (continued)

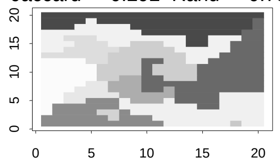
For 10 Groups, the indexes are:
Jaccard = 0.324 Rand = 0.744



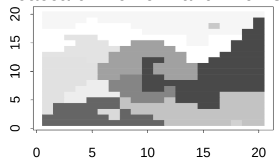
For 11 Groups, the indexes are:
Jaccard = 0.294 Rand = 0.733



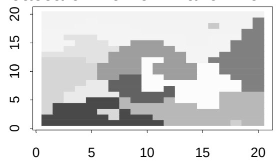
For 12 Groups, the indexes are:
Jaccard = 0.292 Rand = 0.732



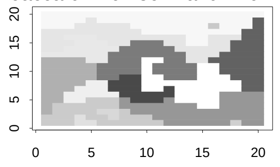
For 13 Groups, the indexes are:
Jaccard = 0.29 Rand = 0.731



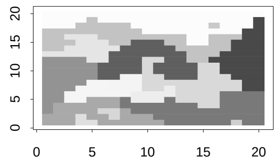
For 14 Groups, the indexes are:
Jaccard = 0.262 Rand = 0.725



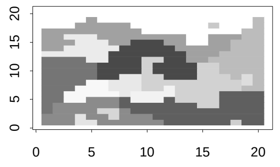
For 15 Groups, the indexes are:
Jaccard = 0.259 Rand = 0.724



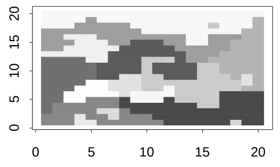
For 16 Groups, the indexes are:
Jaccard = 0.256 Rand = 0.723



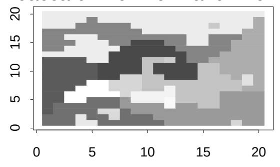
For 17 Groups, the indexes are:
Jaccard = 0.254 Rand = 0.722



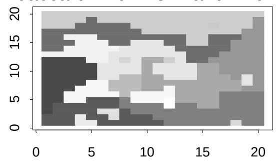
For 18 Groups, the indexes are:
Jaccard = 0.252 Rand = 0.721



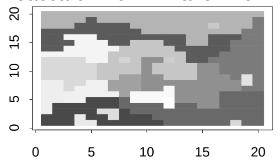
For 19 Groups, the indexes are:
Jaccard = 0.249 Rand = 0.72



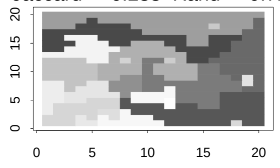
For 20 Groups, the indexes are:
Jaccard = 0.248 Rand = 0.72



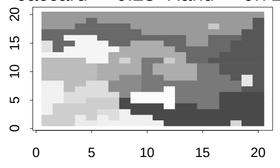
For 21 Groups, the indexes are:
Jaccard = 0.24 Rand = 0.717



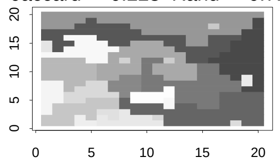
For 22 Groups, the indexes are:
Jaccard = 0.233 Rand = 0.714



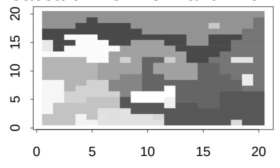
For 23 Groups, the indexes are:
Jaccard = 0.23 Rand = 0.713



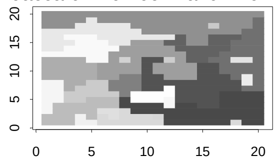
For 24 Groups, the indexes are:
Jaccard = 0.228 Rand = 0.712



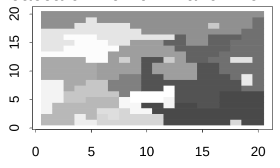
For 25 Groups, the indexes are:
Jaccard = 0.226 Rand = 0.712



For 26 Groups, the indexes are:
Jaccard = 0.209 Rand = 0.705



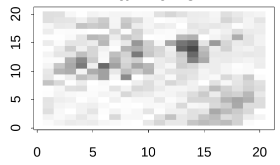
For 27 Groups, the indexes are:
Jaccard = 0.201 Rand = 0.702



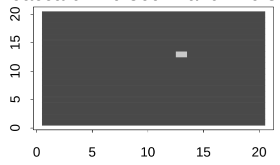
Two-Part Method— Simulated Data Set 2

Simulated Data Set 2 Two-Part Method (*a* = 0.25)

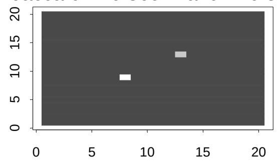
Original Data Set
a = 0.25



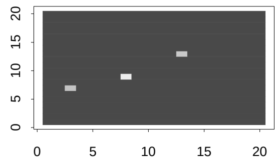
For 2 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.371



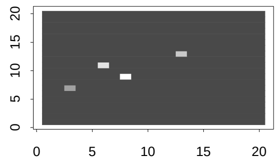
For 3 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.375



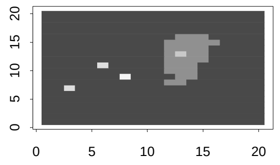
For 4 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.376



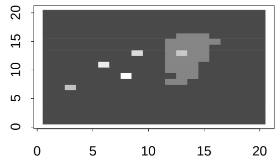
For 5 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.379



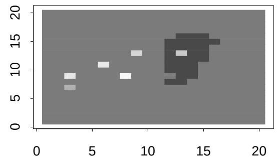
For 6 Groups, the indexes are:
Jaccard = 0.367 Rand = 0.437



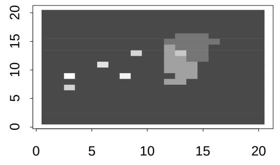
For 7 Groups, the indexes are:
Jaccard = 0.368 Rand = 0.44



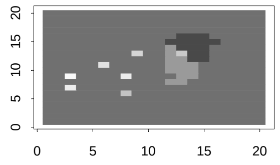
For 8 Groups, the indexes are:
Jaccard = 0.367 Rand = 0.441



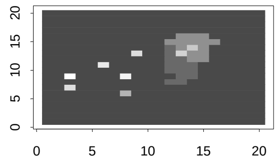
For 9 Groups, the indexes are:
Jaccard = 0.368 Rand = 0.444



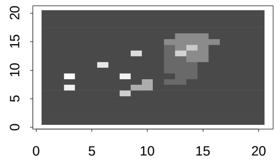
For 10 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.447



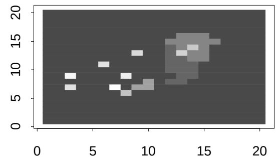
For 11 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.447



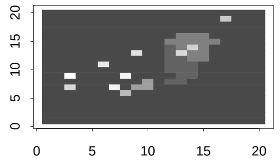
For 12 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.457



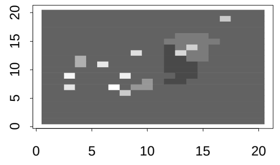
For 13 Groups, the indexes are:
Jaccard = 0.372 Rand = 0.46



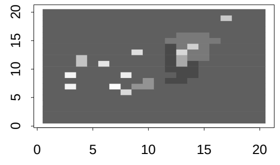
For 14 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.461



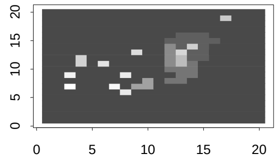
For 15 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.463



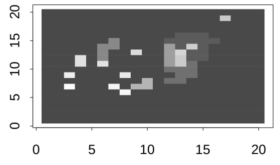
For 16 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.462



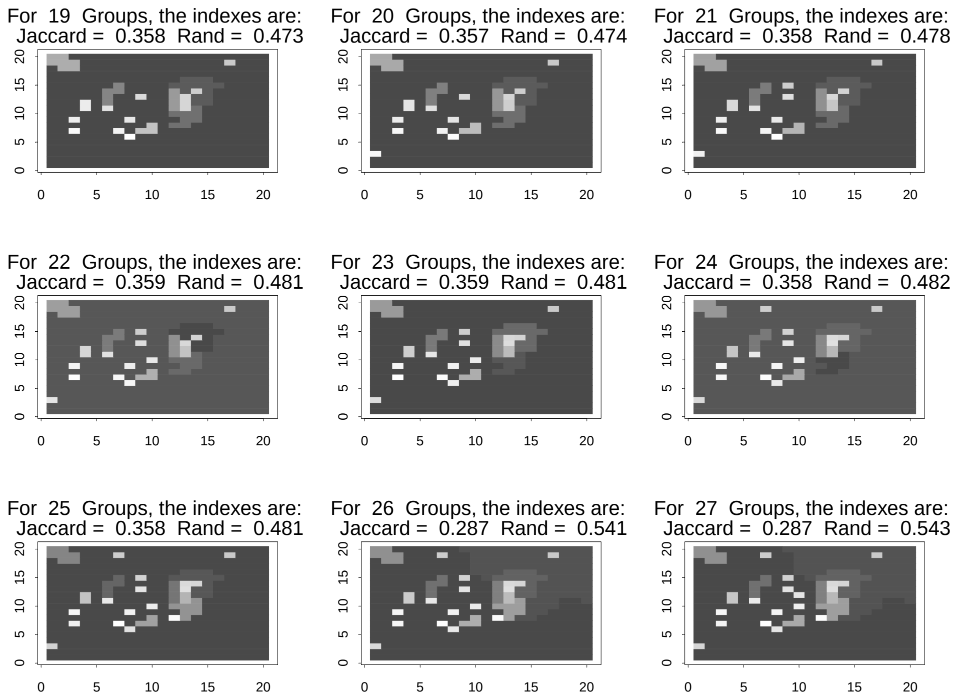
For 17 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.462



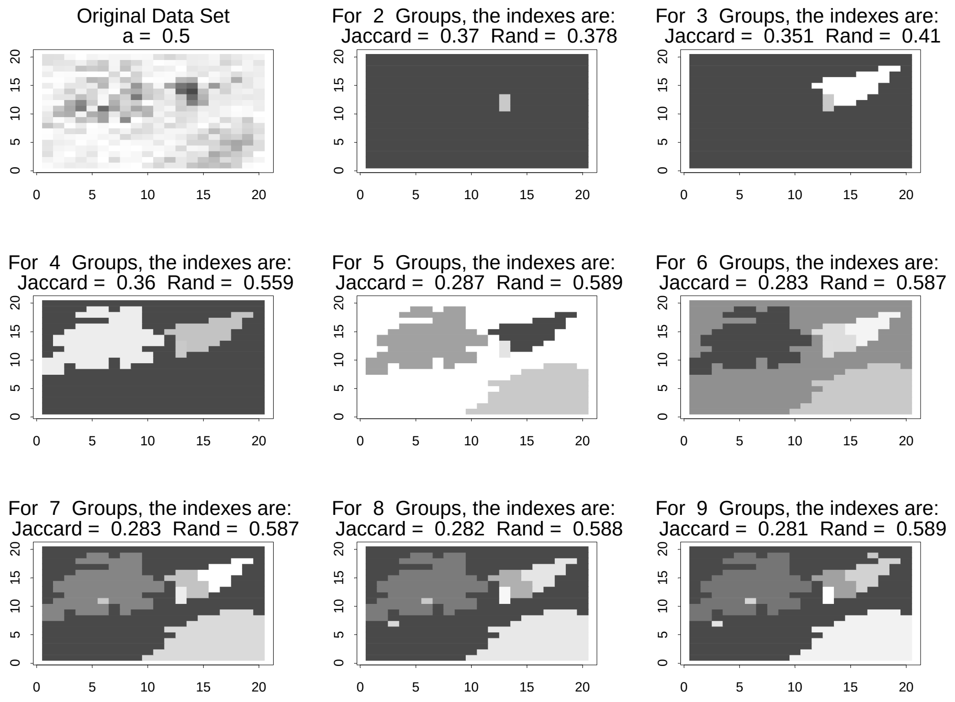
For 18 Groups, the indexes are:
Jaccard = 0.364 Rand = 0.467



Simulated Data Set 2 Two-Part Method (*a* = 0.25) (continued)

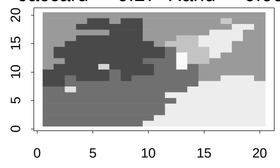


Simulated Data Set 2 Two-Part Method (*a* = 0.50)

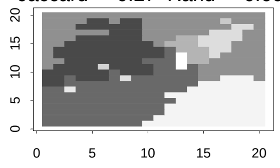


Simulated Data Set 2 Two-Part Method (a = 0.50) (continued)

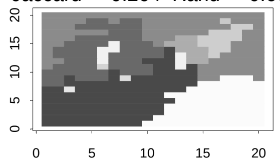
For 10 Groups, the indexes are: Jaccard = 0.27 Rand = 0.663



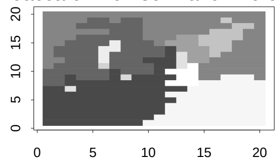
For 11 Groups, the indexes are: Jaccard = 0.27 Rand = 0.664



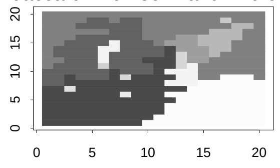
For 12 Groups, the indexes are: Jaccard = 0.264 Rand = 0.662



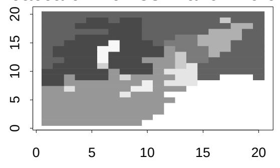
For 13 Groups, the indexes are: Jaccard = 0.259 Rand = 0.665



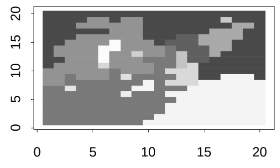
For 14 Groups, the indexes are: Jaccard = 0.259 Rand = 0.666



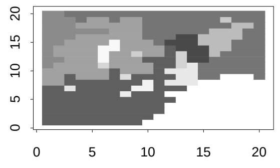
For 15 Groups, the indexes are: Jaccard = 0.258 Rand = 0.667



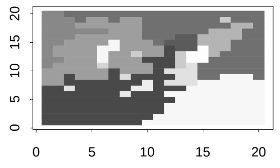
For 16 Groups, the indexes are: Jaccard = 0.257 Rand = 0.667



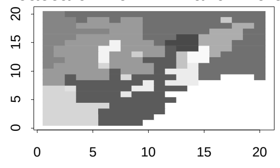
For 17 Groups, the indexes are: Jaccard = 0.232 Rand = 0.665



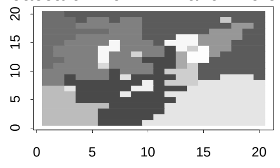
For 18 Groups, the indexes are: Jaccard = 0.231 Rand = 0.664



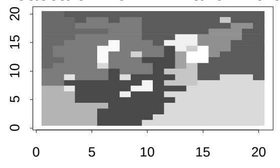
For 19 Groups, the indexes are: Jaccard = 0.217 Rand = 0.67



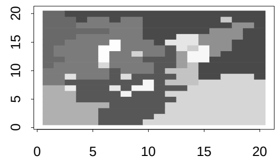
For 20 Groups, the indexes are: Jaccard = 0.217 Rand = 0.669



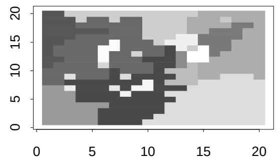
For 21 Groups, the indexes are: Jaccard = 0.217 Rand = 0.67



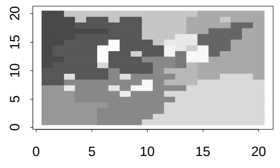
For 22 Groups, the indexes are: Jaccard = 0.216 Rand = 0.67



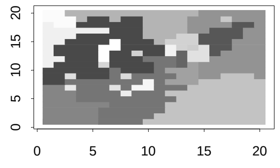
For 23 Groups, the indexes are: Jaccard = 0.209 Rand = 0.678



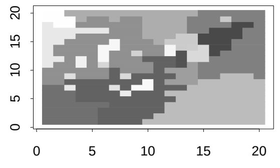
For 24 Groups, the indexes are: Jaccard = 0.207 Rand = 0.678



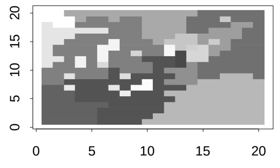
For 25 Groups, the indexes are: Jaccard = 0.204 Rand = 0.677



For 26 Groups, the indexes are: Jaccard = 0.201 Rand = 0.676



For 27 Groups, the indexes are: Jaccard = 0.201 Rand = 0.676

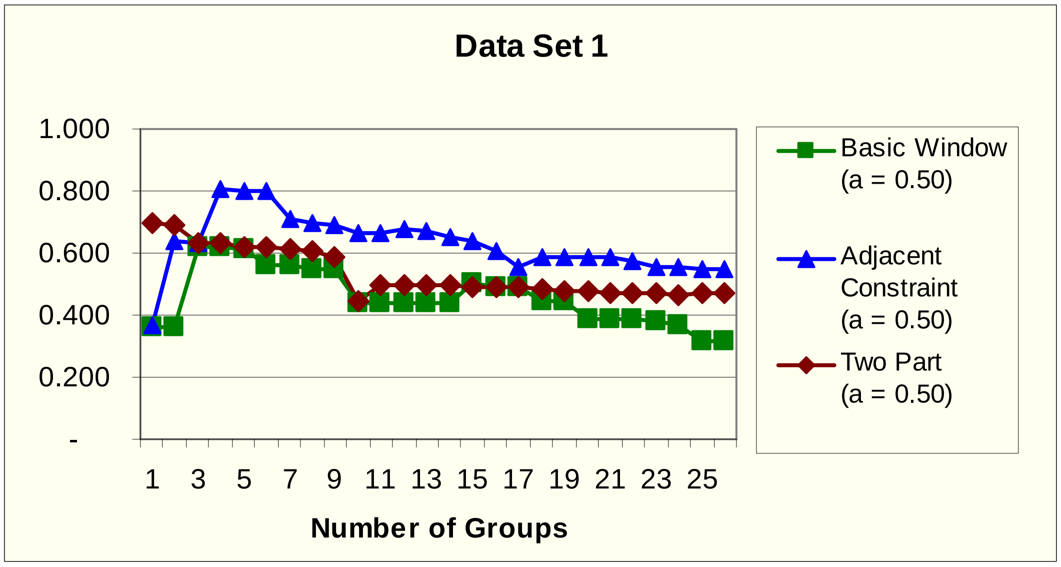


A.2 Numerical and Graphical Results

The following Tables and associated charts show the Jaccard indexes resulting from the application of each of the three proposed clustering methods as applied to the four simulated data sets. Select weights of *a* are used for each of of the clustering methods (values of *a* vary between 0.00, 0.25, 0.50, 0.75 and 1.00).

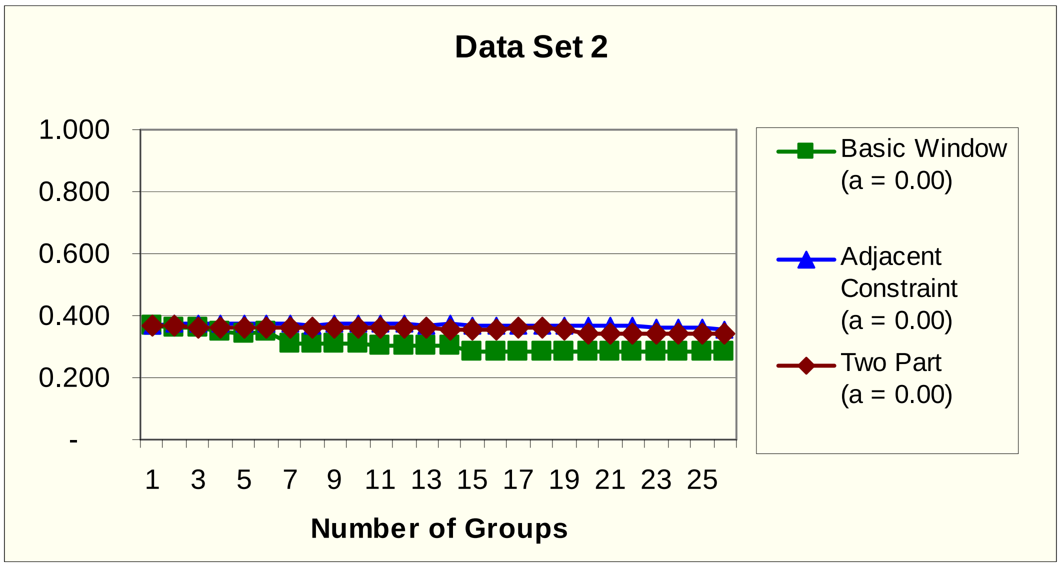
Data Set 1 (*a* = 0.50)

Number of Groups	Basic Window (*a* = 0.50)	Adjacent Constraint (*a* = 0.50)	Two Part (*a* = 0.50)
2	0.360	0.366	0.696
3	0.360	0.640	0.688
4	0.617	0.632	0.631
5	0.617	0.804	0.631
6	0.611	0.802	0.619
7	0.564	0.802	0.619
8	0.564	0.708	0.615
9	0.546	0.698	0.607
10	0.546	0.689	0.588
11	0.440	0.665	0.444
12	0.440	0.665	0.497
13	0.440	0.676	0.497
14	0.440	0.671	0.495
15	0.440	0.653	0.495
16	0.505	0.637	0.493
17	0.493	0.604	0.492
18	0.493	0.558	0.490
19	0.445	0.588	0.486
20	0.443	0.588	0.478
21	0.384	0.587	0.475
22	0.384	0.587	0.472
23	0.384	0.574	0.469
24	0.381	0.556	0.468
25	0.368	0.552	0.466
26	0.315	0.547	0.474
27	0.315	0.547	0.472



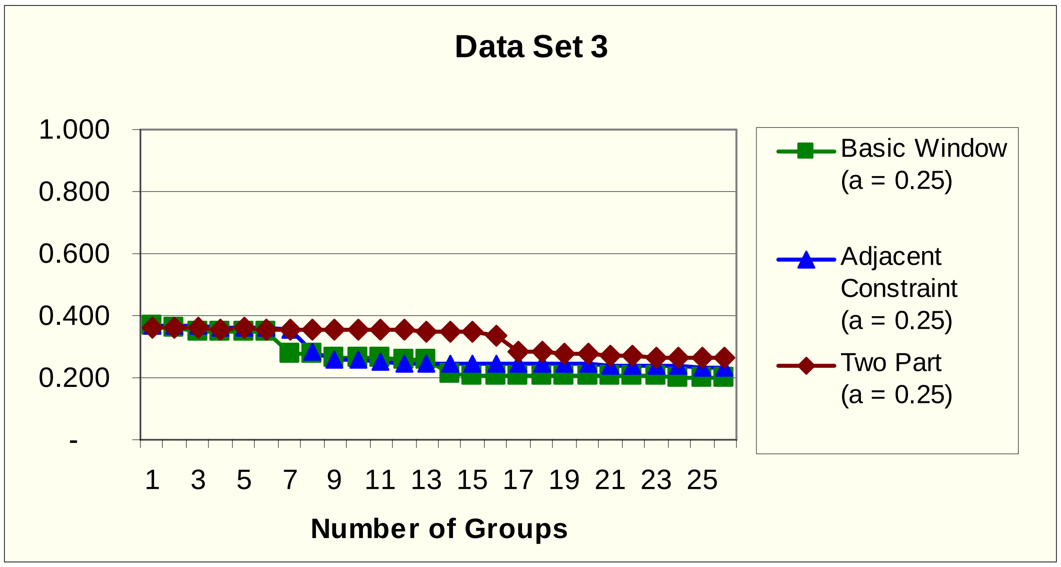
Data Set 2 (*a* = 0.00)

Number of Groups	Basic Window (*a* = 0.00)	Adjacent Constraint (*a* = 0.00)	Two Part (*a* = 0.00)
2	0.365	0.369	0.369
3	0.361	0.373	0.365
4	0.361	0.374	0.364
5	0.347	0.375	0.361
6	0.340	0.371	0.363
7	0.347	0.371	0.362
8	0.311	0.371	0.362
9	0.311	0.370	0.361
10	0.311	0.372	0.361
11	0.310	0.371	0.362
12	0.302	0.372	0.361
13	0.302	0.372	0.360
14	0.302	0.370	0.359
15	0.302	0.371	0.358
16	0.286	0.370	0.357
17	0.285	0.370	0.358
18	0.284	0.370	0.359
19	0.284	0.370	0.360
20	0.284	0.368	0.356
21	0.284	0.367	0.343
22	0.284	0.366	0.344
23	0.283	0.365	0.343
24	0.283	0.364	0.342
25	0.283	0.362	0.342
26	0.283	0.362	0.342
27	0.283	0.358	0.341



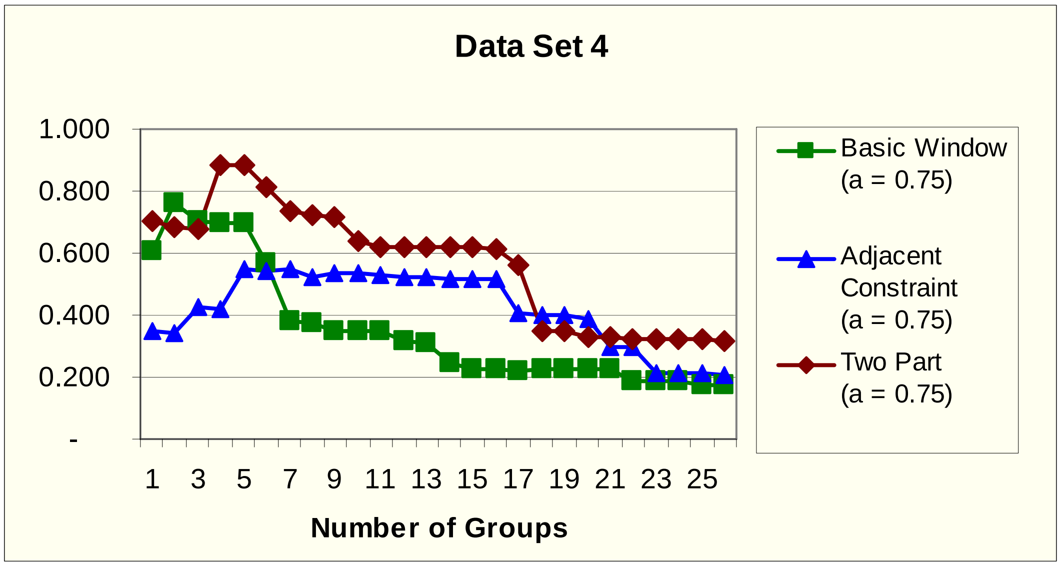
Data Set 3 (*a* = 0.25)

Number of Groups	Basic Window (*a* = 0.25)	Adjacent Constraint (*a* = 0.25)	Two Part (*a* = 0.25)
2	0.365	0.367	0.364
3	0.362	0.366	0.360
4	0.351	0.365	0.359
5	0.350	0.362	0.358
6	0.350	0.361	0.359
7	0.350	0.360	0.358
8	0.278	0.356	0.357
9	0.277	0.282	0.356
10	0.262	0.261	0.355
11	0.262	0.260	0.354
12	0.262	0.252	0.354
13	0.258	0.248	0.354
14	0.258	0.247	0.351
15	0.210	0.247	0.350
16	0.209	0.247	0.348
17	0.209	0.246	0.338
18	0.208	0.246	0.282
19	0.208	0.243	0.282
20	0.206	0.243	0.276
21	0.205	0.242	0.277
22	0.205	0.238	0.271
23	0.205	0.237	0.271
24	0.205	0.237	0.267
25	0.202	0.236	0.267
26	0.202	0.231	0.266
27	0.202	0.231	0.266

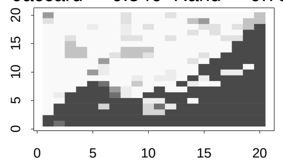


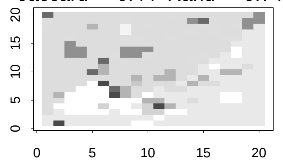
Data Set 4 (*a* = 0.75)

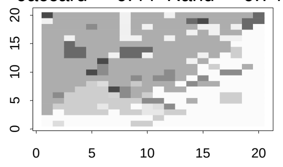
Number of Groups	Basic Window (*a* = 0.75)	Adjacent Constraint (*a* = 0.75)	Two Part (*a* = 0.75)
2	0.609	0.347	0.705
3	0.760	0.342	0.686
4	0.702	0.423	0.679
5	0.696	0.420	0.885
6	0.695	0.546	0.881
7	0.565	0.544	0.811
8	0.383	0.550	0.736
9	0.377	0.520	0.725
10	0.350	0.538	0.715
11	0.350	0.534	0.636
12	0.349	0.531	0.622
13	0.316	0.525	0.621
14	0.308	0.524	0.620
15	0.245	0.519	0.619
16	0.227	0.516	0.617
17	0.225	0.516	0.615
18	0.221	0.408	0.564
19	0.226	0.403	0.349
20	0.225	0.403	0.346
21	0.226	0.389	0.328
22	0.224	0.298	0.326
23	0.190	0.297	0.325
24	0.190	0.213	0.322
25	0.190	0.212	0.320
26	0.174	0.211	0.320
27	0.174	0.207	0.319

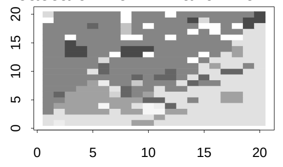


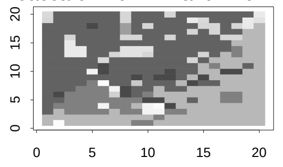
Data Set 1 Basic Window Method (*a* = 0.50)

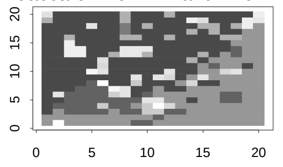

For 10 Groups, the indexes are:
Jaccard = 0.546 Rand = 0.789

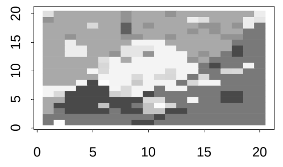

For 11 Groups, the indexes are:
Jaccard = 0.44 Rand = 0.742

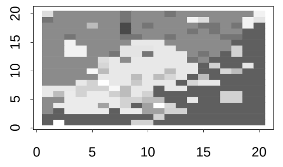

For 12 Groups, the indexes are:
Jaccard = 0.44 Rand = 0.742

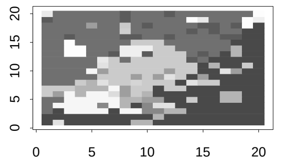

For 13 Groups, the indexes are:
Jaccard = 0.44 Rand = 0.742

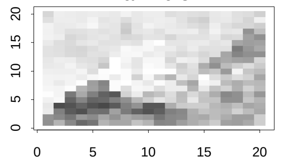

For 14 Groups, the indexes are:
Jaccard = 0.44 Rand = 0.743

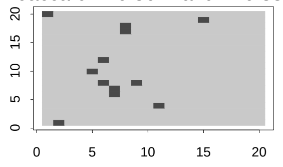

For 15 Groups, the indexes are:
Jaccard = 0.44 Rand = 0.743

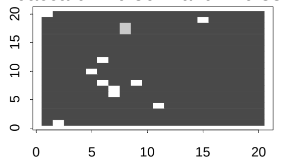

For 16 Groups, the indexes are:
Jaccard = 0.505 Rand = 0.81

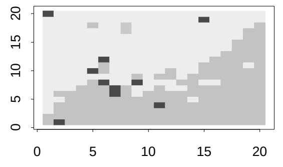

For 17 Groups, the indexes are:
Jaccard = 0.493 Rand = 0.806

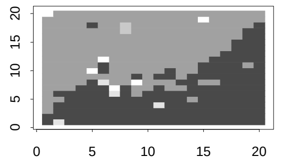

For 18 Groups, the indexes are:
Jaccard = 0.493 Rand = 0.806

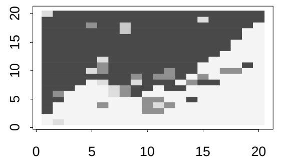

Original Data Set
a = 0.5

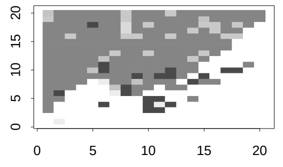

For 2 Groups, the indexes are:
Jaccard = 0.36 Rand = 0.383

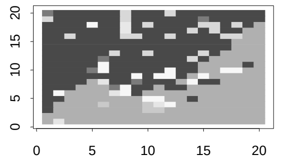

For 3 Groups, the indexes are:
Jaccard = 0.36 Rand = 0.383

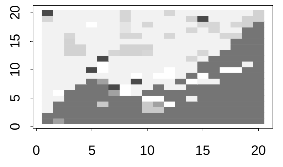

For 4 Groups, the indexes are:
Jaccard = 0.617 Rand = 0.8


For 5 Groups, the indexes are:
Jaccard = 0.617 Rand = 0.801


For 6 Groups, the indexes are:
Jaccard = 0.611 Rand = 0.806

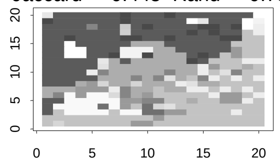

For 7 Groups, the indexes are:
Jaccard = 0.564 Rand = 0.79


For 8 Groups, the indexes are:
Jaccard = 0.564 Rand = 0.79

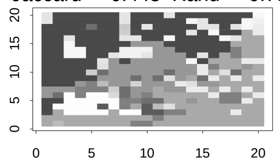

For 9 Groups, the indexes are:
Jaccard = 0.546 Rand = 0.789

Data Set 1 Basic Window Method (a = 0.50) (continued)

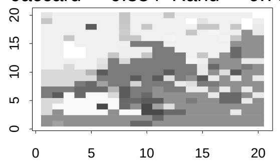
For 19 Groups, the indexes are:
Jaccard = 0.445 Rand = 0.788



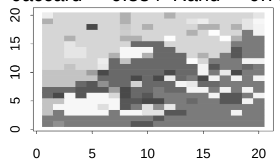
For 20 Groups, the indexes are:
Jaccard = 0.443 Rand = 0.788



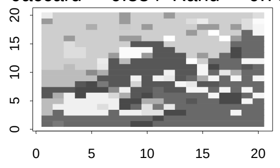
For 21 Groups, the indexes are:
Jaccard = 0.384 Rand = 0.766



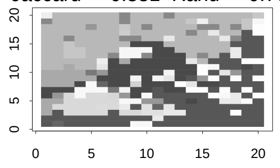
For 22 Groups, the indexes are:
Jaccard = 0.384 Rand = 0.766



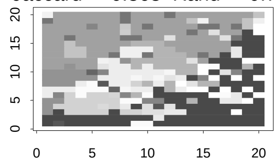
For 23 Groups, the indexes are:
Jaccard = 0.384 Rand = 0.766



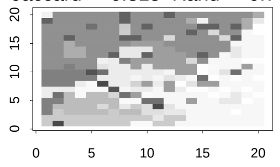
For 24 Groups, the indexes are:
Jaccard = 0.381 Rand = 0.765



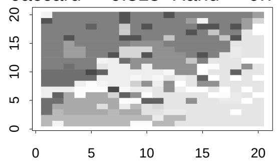
For 25 Groups, the indexes are:
Jaccard = 0.368 Rand = 0.762



For 26 Groups, the indexes are:
Jaccard = 0.315 Rand = 0.743

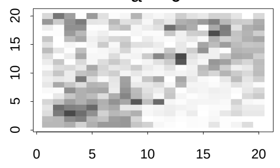


For 27 Groups, the indexes are:
Jaccard = 0.315 Rand = 0.743

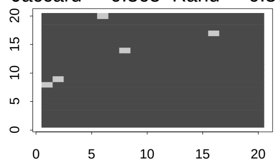


Data Set 2 Basic Window Method (a = 0.00)

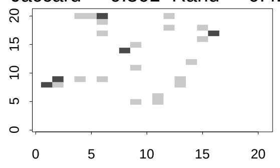
Original Data Set
a = 0



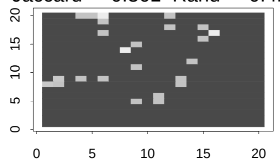
For 2 Groups, the indexes are:
Jaccard = 0.365 Rand = 0.375



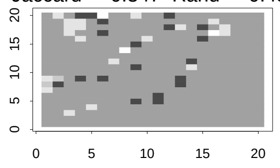
For 3 Groups, the indexes are:
Jaccard = 0.361 Rand = 0.411



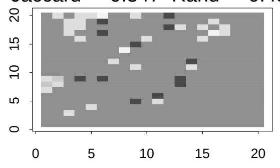
For 4 Groups, the indexes are:
Jaccard = 0.361 Rand = 0.411



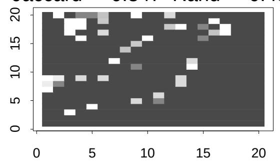
For 5 Groups, the indexes are:
Jaccard = 0.347 Rand = 0.434



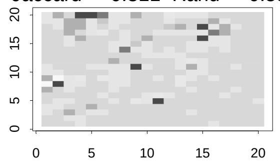
For 6 Groups, the indexes are:
Jaccard = 0.347 Rand = 0.434



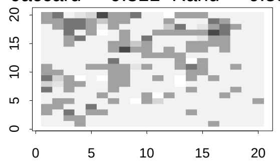
For 7 Groups, the indexes are:
Jaccard = 0.347 Rand = 0.434



For 8 Groups, the indexes are:
Jaccard = 0.311 Rand = 0.553

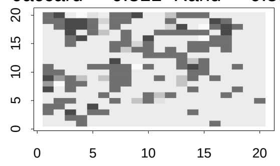


For 9 Groups, the indexes are:
Jaccard = 0.311 Rand = 0.553

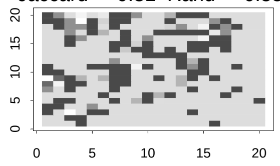


Data Set 2 Basic Window Method (*a* = 0.00) (continued)

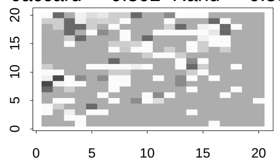
For 10 Groups, the indexes are:
Jaccard = 0.311 Rand = 0.553



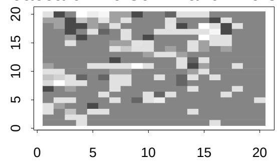
For 11 Groups, the indexes are:
Jaccard = 0.31 Rand = 0.553



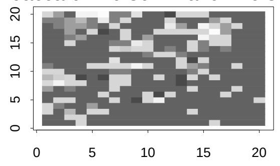
For 12 Groups, the indexes are:
Jaccard = 0.302 Rand = 0.557



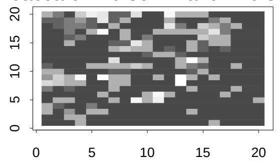
For 13 Groups, the indexes are:
Jaccard = 0.302 Rand = 0.557



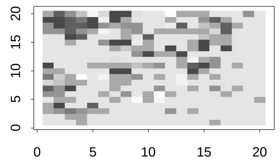
For 14 Groups, the indexes are:
Jaccard = 0.302 Rand = 0.557



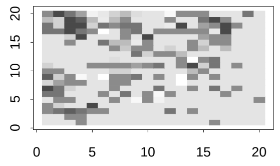
For 15 Groups, the indexes are:
Jaccard = 0.302 Rand = 0.557



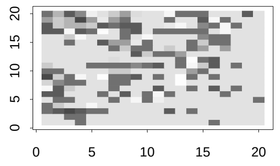
For 16 Groups, the indexes are:
Jaccard = 0.286 Rand = 0.583



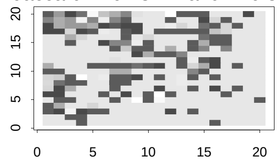
For 17 Groups, the indexes are:
Jaccard = 0.285 Rand = 0.583



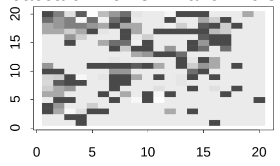
For 18 Groups, the indexes are:
Jaccard = 0.284 Rand = 0.583



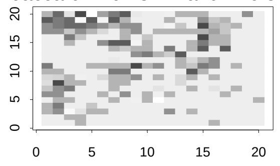
For 19 Groups, the indexes are:
Jaccard = 0.284 Rand = 0.583



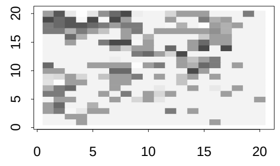
For 20 Groups, the indexes are:
Jaccard = 0.284 Rand = 0.583



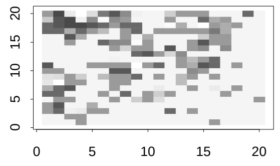
For 21 Groups, the indexes are:
Jaccard = 0.284 Rand = 0.584



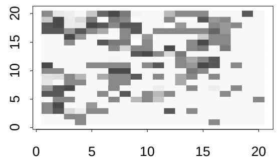
For 22 Groups, the indexes are:
Jaccard = 0.284 Rand = 0.584



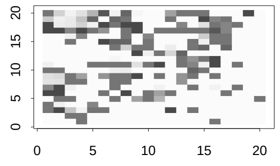
For 23 Groups, the indexes are:
Jaccard = 0.283 Rand = 0.584



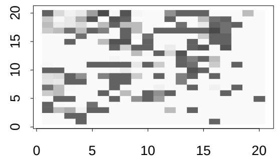
For 24 Groups, the indexes are:
Jaccard = 0.283 Rand = 0.584



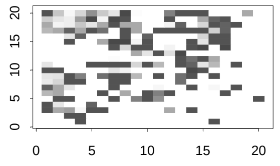
For 25 Groups, the indexes are:
Jaccard = 0.283 Rand = 0.584



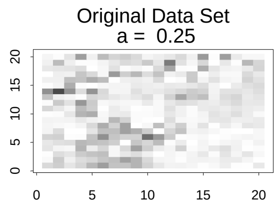
For 26 Groups, the indexes are:
Jaccard = 0.283 Rand = 0.584

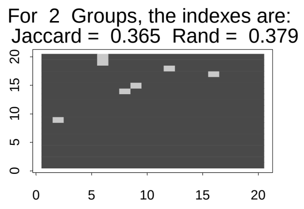


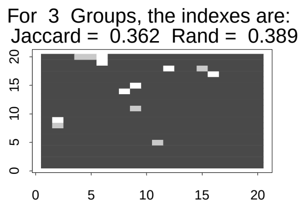
For 27 Groups, the indexes are:
Jaccard = 0.283 Rand = 0.584

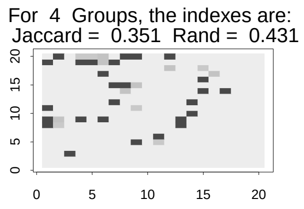


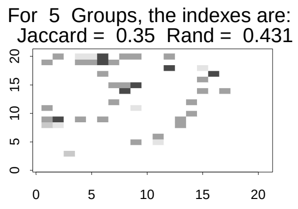
Data Set 3 Basic Window Method (*a* = 0.25)

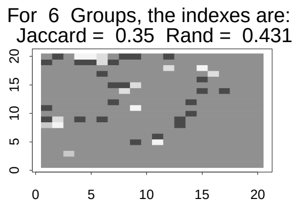


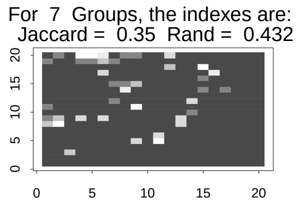


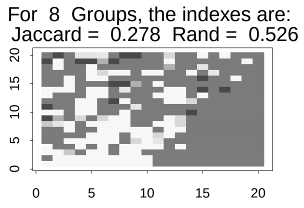


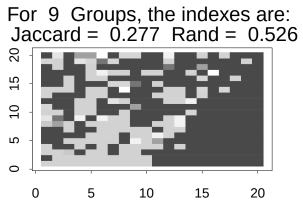


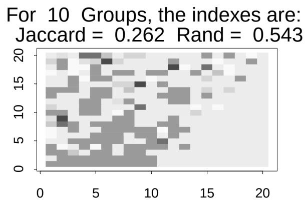


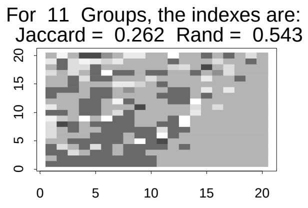


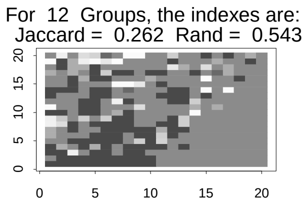


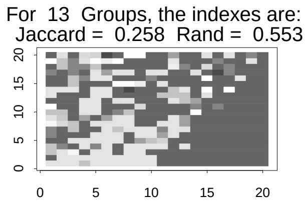


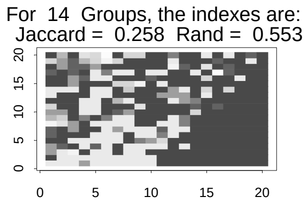


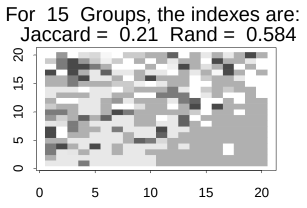


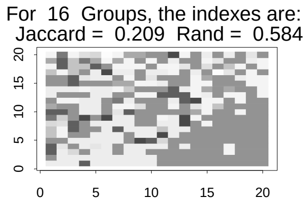


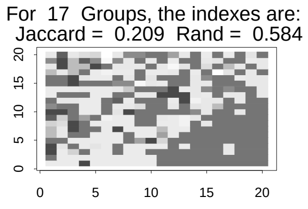


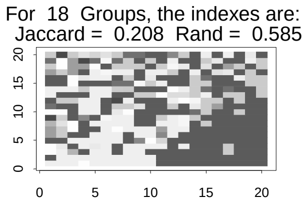






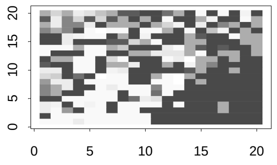




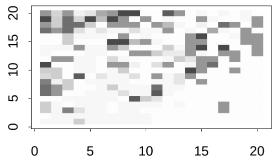


Data Set 3 Basic Window Method (*a* = 0.25) (continued)

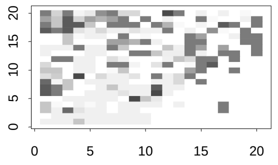
For 19 Groups, the indexes are: Jaccard = 0.208 Rand = 0.585



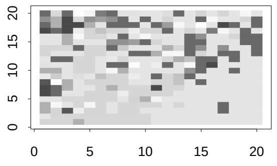
For 20 Groups, the indexes are: Jaccard = 0.206 Rand = 0.591



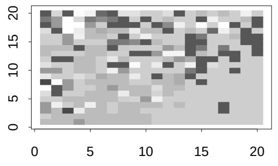
For 21 Groups, the indexes are: Jaccard = 0.205 Rand = 0.591



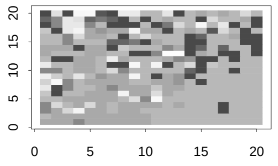
For 22 Groups, the indexes are: Jaccard = 0.205 Rand = 0.591



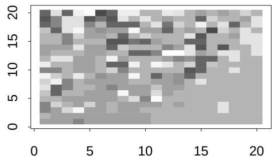
For 23 Groups, the indexes are: Jaccard = 0.205 Rand = 0.591



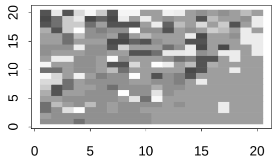
For 24 Groups, the indexes are: Jaccard = 0.205 Rand = 0.591



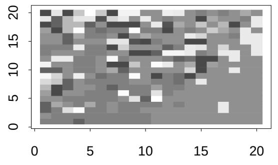
For 25 Groups, the indexes are: Jaccard = 0.202 Rand = 0.594



For 26 Groups, the indexes are: Jaccard = 0.202 Rand = 0.594

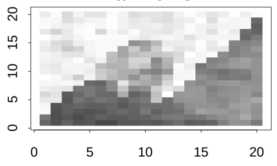


For 27 Groups, the indexes are: Jaccard = 0.202 Rand = 0.594

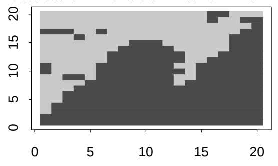


Data Set 4 Basic Window Method (*a* = 0.75)

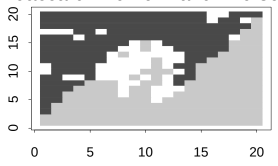
Original Data Set a = 0.75



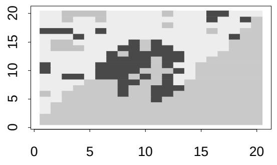
For 2 Groups, the indexes are: Jaccard = 0.609 Rand = 0.784



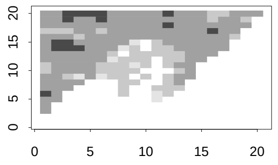
For 3 Groups, the indexes are: Jaccard = 0.76 Rand = 0.897



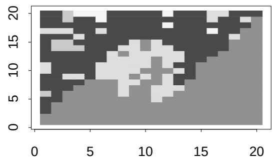
For 4 Groups, the indexes are: Jaccard = 0.702 Rand = 0.873



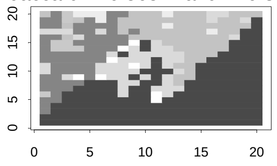
For 5 Groups, the indexes are: Jaccard = 0.696 Rand = 0.871



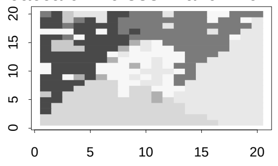
For 6 Groups, the indexes are: Jaccard = 0.695 Rand = 0.87



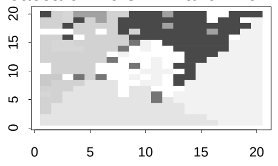
For 7 Groups, the indexes are: Jaccard = 0.565 Rand = 0.817



For 8 Groups, the indexes are: Jaccard = 0.383 Rand = 0.753

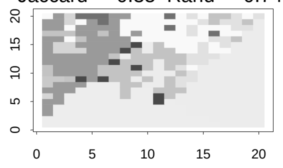


For 9 Groups, the indexes are: Jaccard = 0.377 Rand = 0.751

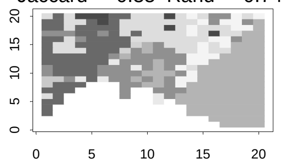


Data Set 4 Basic Window Method (*a* = 0.75) (continued)

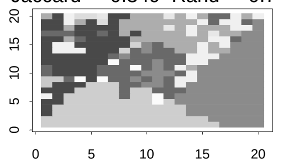
For 10 Groups, the indexes are:
Jaccard = 0.35 Rand = 0.742



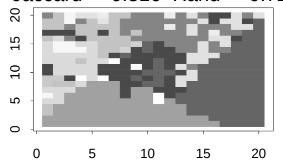
For 11 Groups, the indexes are:
Jaccard = 0.35 Rand = 0.742



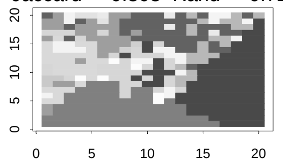
For 12 Groups, the indexes are:
Jaccard = 0.349 Rand = 0.741



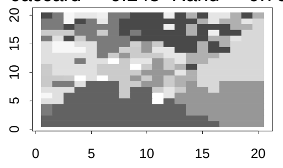
For 13 Groups, the indexes are:
Jaccard = 0.316 Rand = 0.728



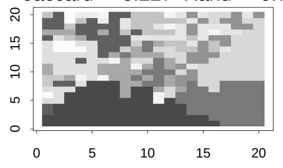
For 14 Groups, the indexes are:
Jaccard = 0.308 Rand = 0.727



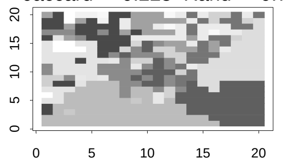
For 15 Groups, the indexes are:
Jaccard = 0.245 Rand = 0.707



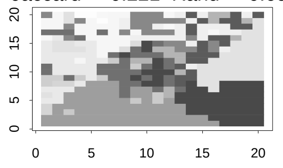
For 16 Groups, the indexes are:
Jaccard = 0.227 Rand = 0.7



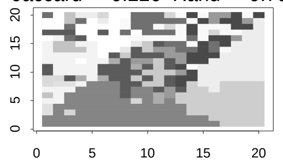
For 17 Groups, the indexes are:
Jaccard = 0.225 Rand = 0.7



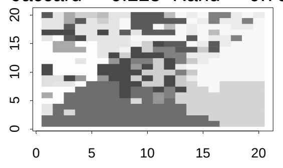
For 18 Groups, the indexes are:
Jaccard = 0.221 Rand = 0.699



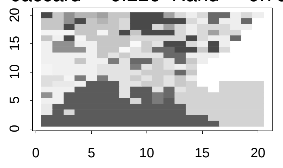
For 19 Groups, the indexes are:
Jaccard = 0.226 Rand = 0.706



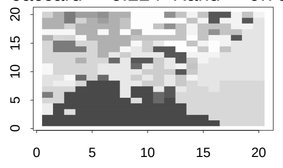
For 20 Groups, the indexes are:
Jaccard = 0.225 Rand = 0.706



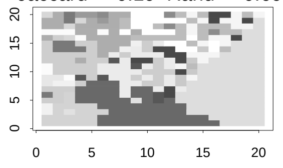
For 21 Groups, the indexes are:
Jaccard = 0.226 Rand = 0.709



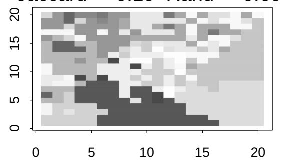
For 22 Groups, the indexes are:
Jaccard = 0.224 Rand = 0.708



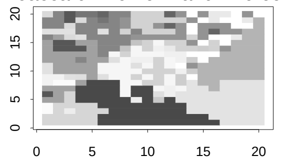
For 23 Groups, the indexes are:
Jaccard = 0.19 Rand = 0.697



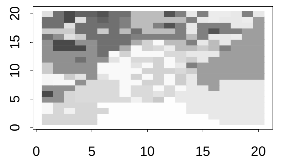
For 24 Groups, the indexes are:
Jaccard = 0.19 Rand = 0.698



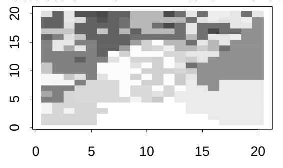
For 25 Groups, the indexes are:
Jaccard = 0.19 Rand = 0.697



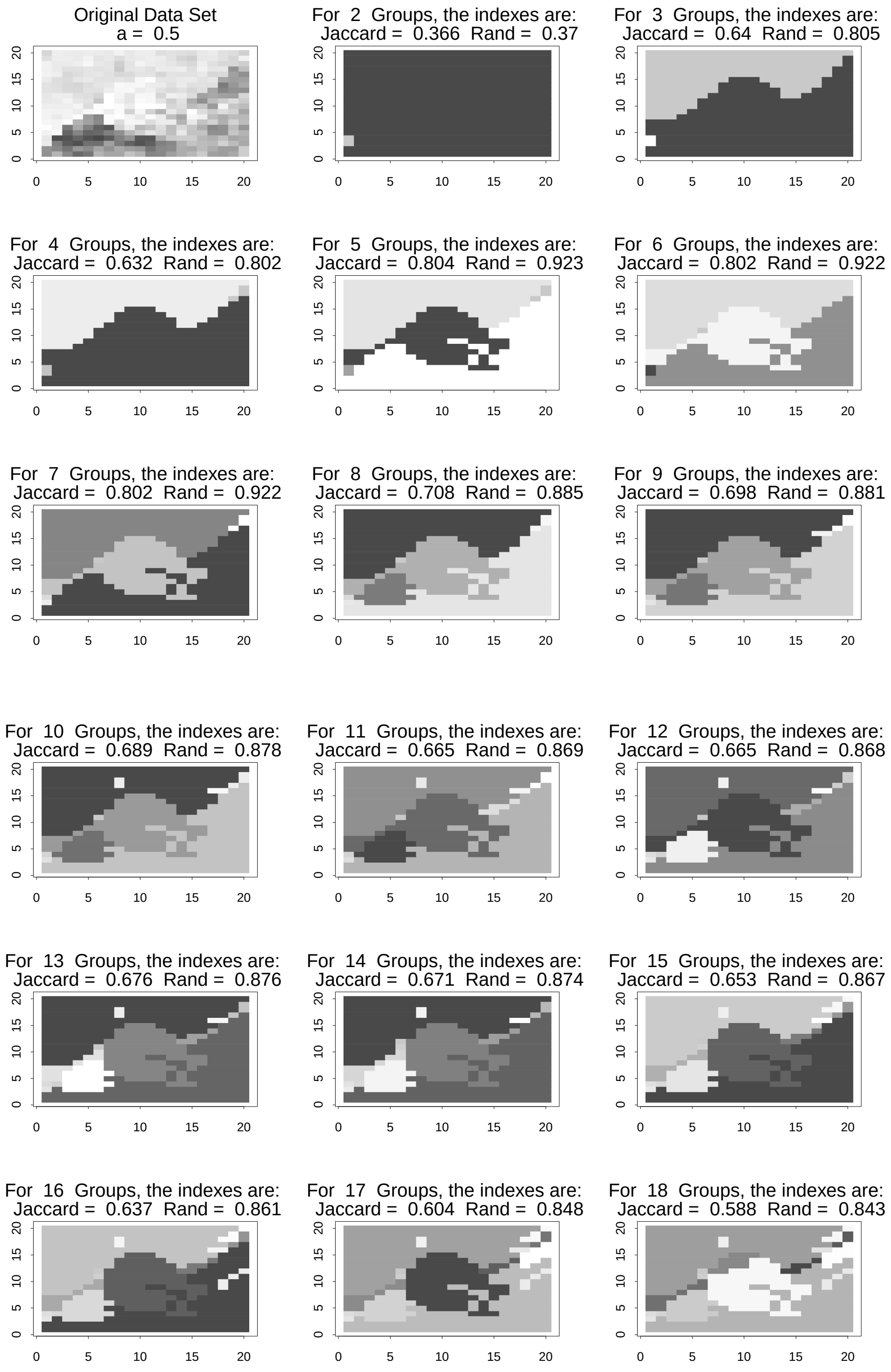
For 26 Groups, the indexes are:
Jaccard = 0.174 Rand = 0.694



For 27 Groups, the indexes are:
Jaccard = 0.174 Rand = 0.694

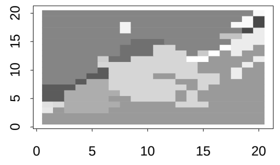


Data Set 1 Adjacent Constraint Method (*a* = 0.50)

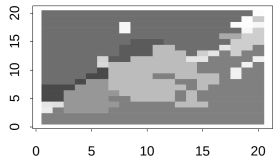


Data Set 1 Adjacent Constraint Method (*a* = 0.50) (continued)

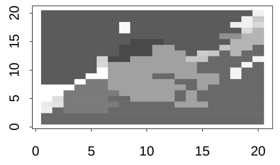
For 19 Groups, the indexes are:
Jaccard = 0.588 Rand = 0.843



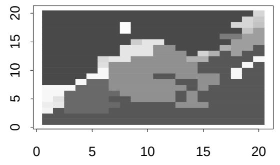
For 20 Groups, the indexes are:
Jaccard = 0.588 Rand = 0.843



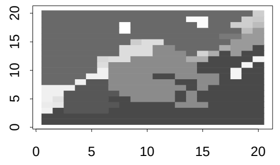
For 21 Groups, the indexes are:
Jaccard = 0.587 Rand = 0.842



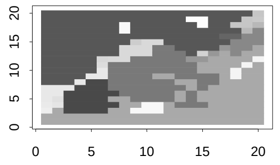
For 22 Groups, the indexes are:
Jaccard = 0.587 Rand = 0.842



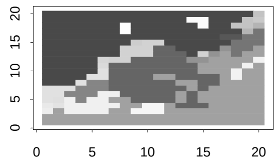
For 23 Groups, the indexes are:
Jaccard = 0.574 Rand = 0.837



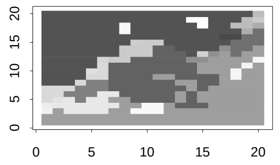
For 24 Groups, the indexes are:
Jaccard = 0.556 Rand = 0.831



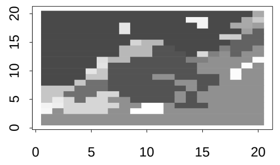
For 25 Groups, the indexes are:
Jaccard = 0.552 Rand = 0.829



For 26 Groups, the indexes are:
Jaccard = 0.547 Rand = 0.827

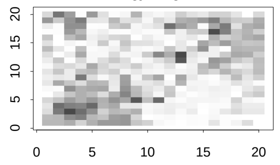


For 27 Groups, the indexes are:
Jaccard = 0.547 Rand = 0.827

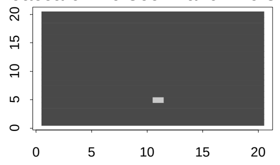


Data Set 2 Adjacent Constraint Method (*a* = 0.00)

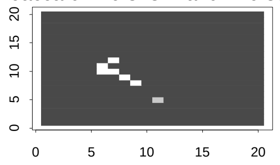
Original Data Set
a = 0



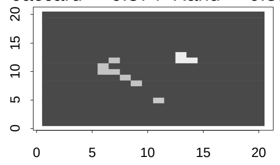
For 2 Groups, the indexes are:
Jaccard = 0.369 Rand = 0.371



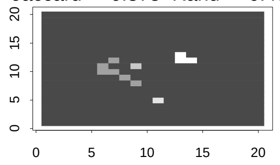
For 3 Groups, the indexes are:
Jaccard = 0.373 Rand = 0.391



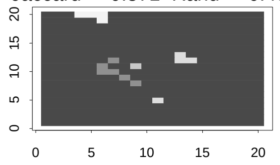
For 4 Groups, the indexes are:
Jaccard = 0.374 Rand = 0.399



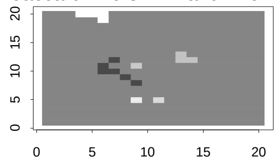
For 5 Groups, the indexes are:
Jaccard = 0.375 Rand = 0.402



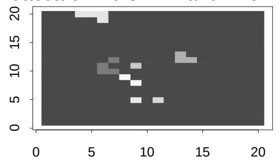
For 6 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.405



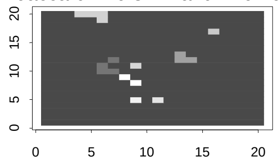
For 7 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.409



For 8 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.409

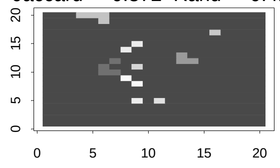


For 9 Groups, the indexes are:
Jaccard = 0.37 Rand = 0.409

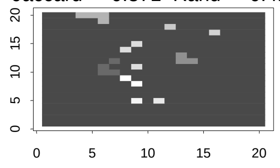


Data Set 2 Adjacent Constraint Method (*a* = 0.00) (continued)

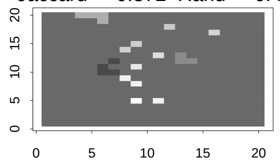
For 10 Groups, the indexes are:
Jaccard = 0.372 Rand = 0.416



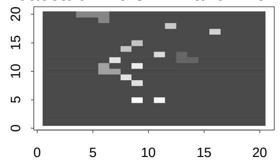
For 11 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.417



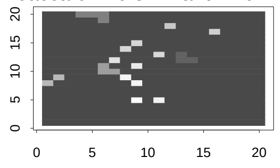
For 12 Groups, the indexes are:
Jaccard = 0.372 Rand = 0.42



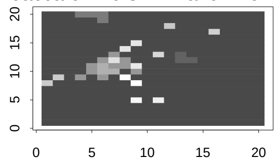
For 13 Groups, the indexes are:
Jaccard = 0.372 Rand = 0.42



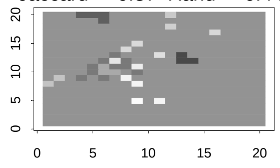
For 14 Groups, the indexes are:
Jaccard = 0.37 Rand = 0.422



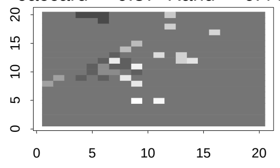
For 15 Groups, the indexes are:
Jaccard = 0.371 Rand = 0.445



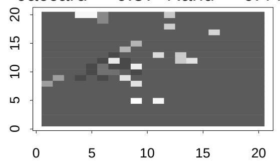
For 16 Groups, the indexes are:
Jaccard = 0.37 Rand = 0.446



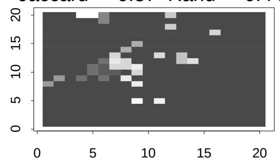
For 17 Groups, the indexes are:
Jaccard = 0.37 Rand = 0.446



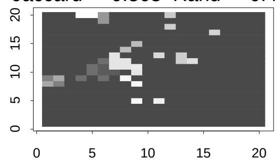
For 18 Groups, the indexes are:
Jaccard = 0.37 Rand = 0.446



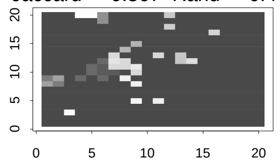
For 19 Groups, the indexes are:
Jaccard = 0.37 Rand = 0.446



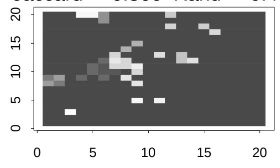
For 20 Groups, the indexes are:
Jaccard = 0.368 Rand = 0.447



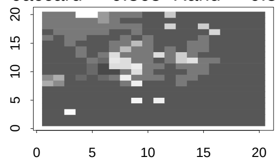
For 21 Groups, the indexes are:
Jaccard = 0.367 Rand = 0.448



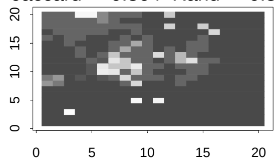
For 22 Groups, the indexes are:
Jaccard = 0.366 Rand = 0.449



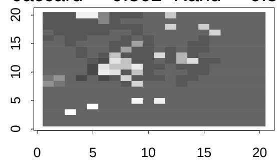
For 23 Groups, the indexes are:
Jaccard = 0.365 Rand = 0.574



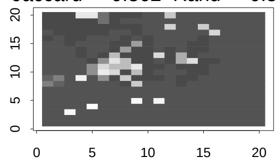
For 24 Groups, the indexes are:
Jaccard = 0.364 Rand = 0.574



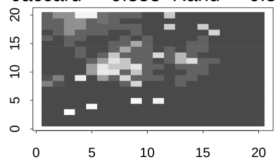
For 25 Groups, the indexes are:
Jaccard = 0.362 Rand = 0.573



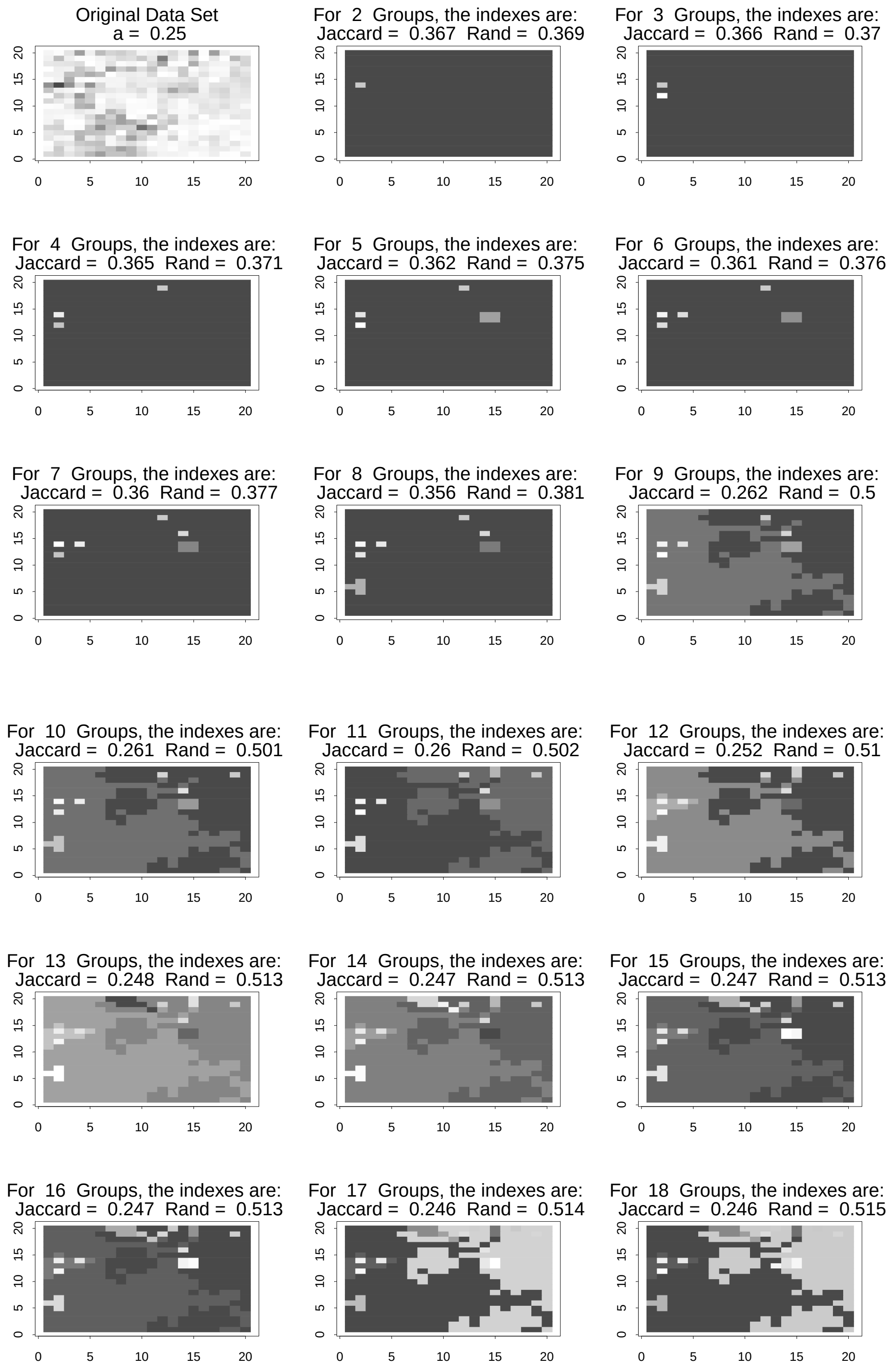
For 26 Groups, the indexes are:
Jaccard = 0.362 Rand = 0.573



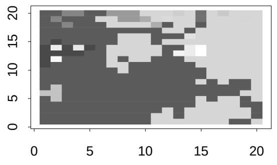
For 27 Groups, the indexes are:
Jaccard = 0.358 Rand = 0.572

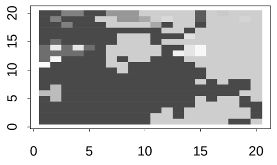


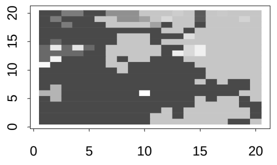
Data Set 3 Adjacent Constraint Method (a = 0.25)

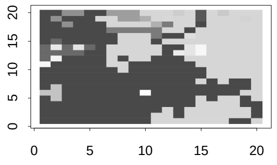


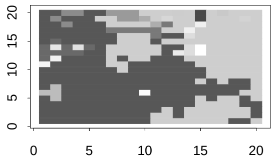
Data Set 3 Adjacent Constraint Method ($a = 0.25$) (continued)

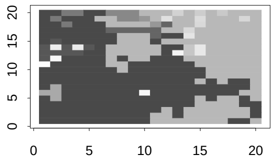

For 19 Groups, the indexes are: Jaccard = 0.243 Rand = 0.519

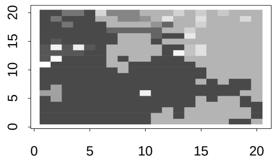

For 20 Groups, the indexes are: Jaccard = 0.243 Rand = 0.519

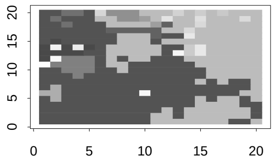

For 21 Groups, the indexes are: Jaccard = 0.242 Rand = 0.52

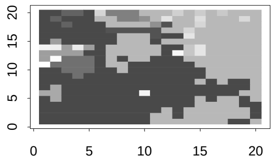

For 22 Groups, the indexes are: Jaccard = 0.238 Rand = 0.527


For 23 Groups, the indexes are: Jaccard = 0.237 Rand = 0.529

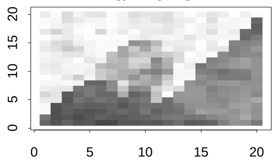

For 24 Groups, the indexes are: Jaccard = 0.237 Rand = 0.529

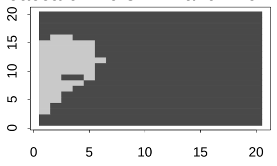

For 25 Groups, the indexes are: Jaccard = 0.236 Rand = 0.529

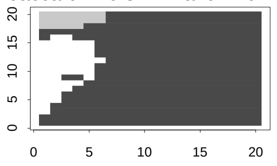

For 26 Groups, the indexes are: Jaccard = 0.231 Rand = 0.542

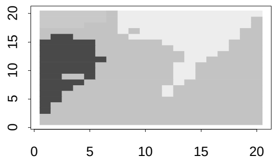

For 27 Groups, the indexes are: Jaccard = 0.231 Rand = 0.542

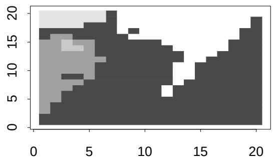
Data Set 4 Adjacent Constraint Method ($a = 0.75$)

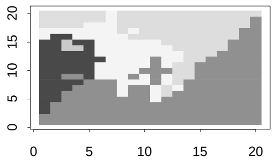

Original Data Set a = 0.75

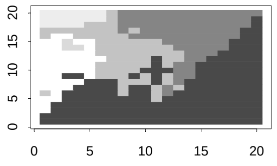

For 2 Groups, the indexes are: Jaccard = 0.347 Rand = 0.441

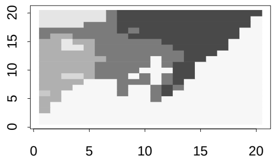

For 3 Groups, the indexes are: Jaccard = 0.342 Rand = 0.468

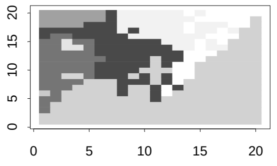

For 4 Groups, the indexes are: Jaccard = 0.423 Rand = 0.665


For 5 Groups, the indexes are: Jaccard = 0.42 Rand = 0.663

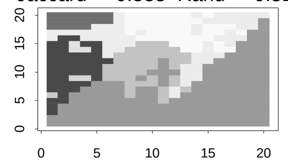

For 6 Groups, the indexes are: Jaccard = 0.546 Rand = 0.805

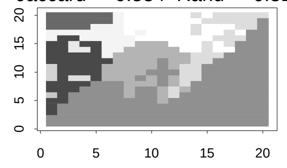

For 7 Groups, the indexes are: Jaccard = 0.544 Rand = 0.805

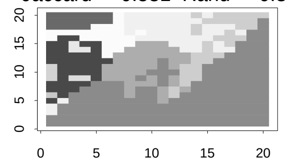

For 8 Groups, the indexes are: Jaccard = 0.55 Rand = 0.809

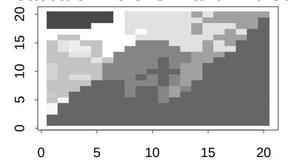

For 9 Groups, the indexes are: Jaccard = 0.52 Rand = 0.799

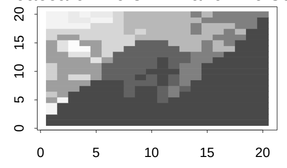
Data Set 4 Adjacent Constraint Method (*a* = 0.75) (continued)

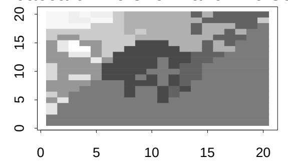

For 10 Groups, the indexes are: Jaccard = 0.538 Rand = 0.813

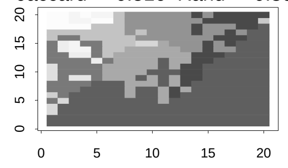

For 11 Groups, the indexes are: Jaccard = 0.534 Rand = 0.812

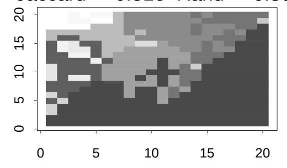

For 12 Groups, the indexes are: Jaccard = 0.531 Rand = 0.81

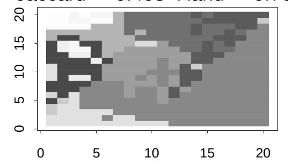

For 13 Groups, the indexes are: Jaccard = 0.525 Rand = 0.808

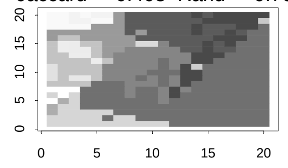

For 14 Groups, the indexes are: Jaccard = 0.524 Rand = 0.808

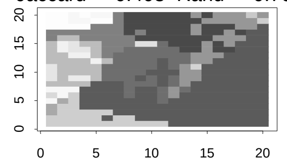

For 15 Groups, the indexes are: Jaccard = 0.519 Rand = 0.806

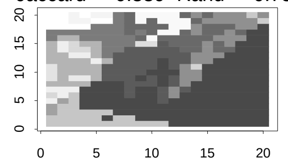

For 16 Groups, the indexes are: Jaccard = 0.516 Rand = 0.805

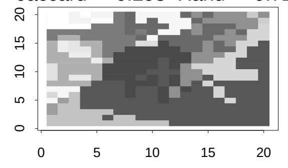

For 17 Groups, the indexes are: Jaccard = 0.516 Rand = 0.805

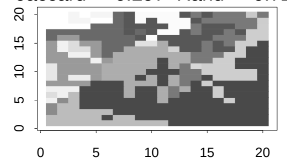

For 18 Groups, the indexes are: Jaccard = 0.408 Rand = 0.764

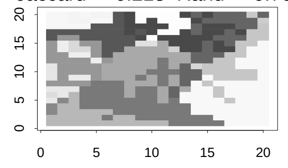

For 19 Groups, the indexes are: Jaccard = 0.403 Rand = 0.762

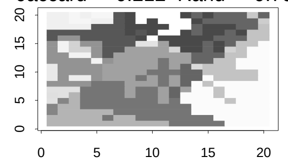

For 20 Groups, the indexes are: Jaccard = 0.403 Rand = 0.762

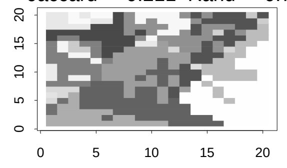

For 21 Groups, the indexes are: Jaccard = 0.389 Rand = 0.757

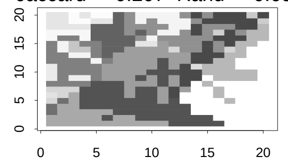

For 22 Groups, the indexes are: Jaccard = 0.298 Rand = 0.725


For 23 Groups, the indexes are: Jaccard = 0.297 Rand = 0.724

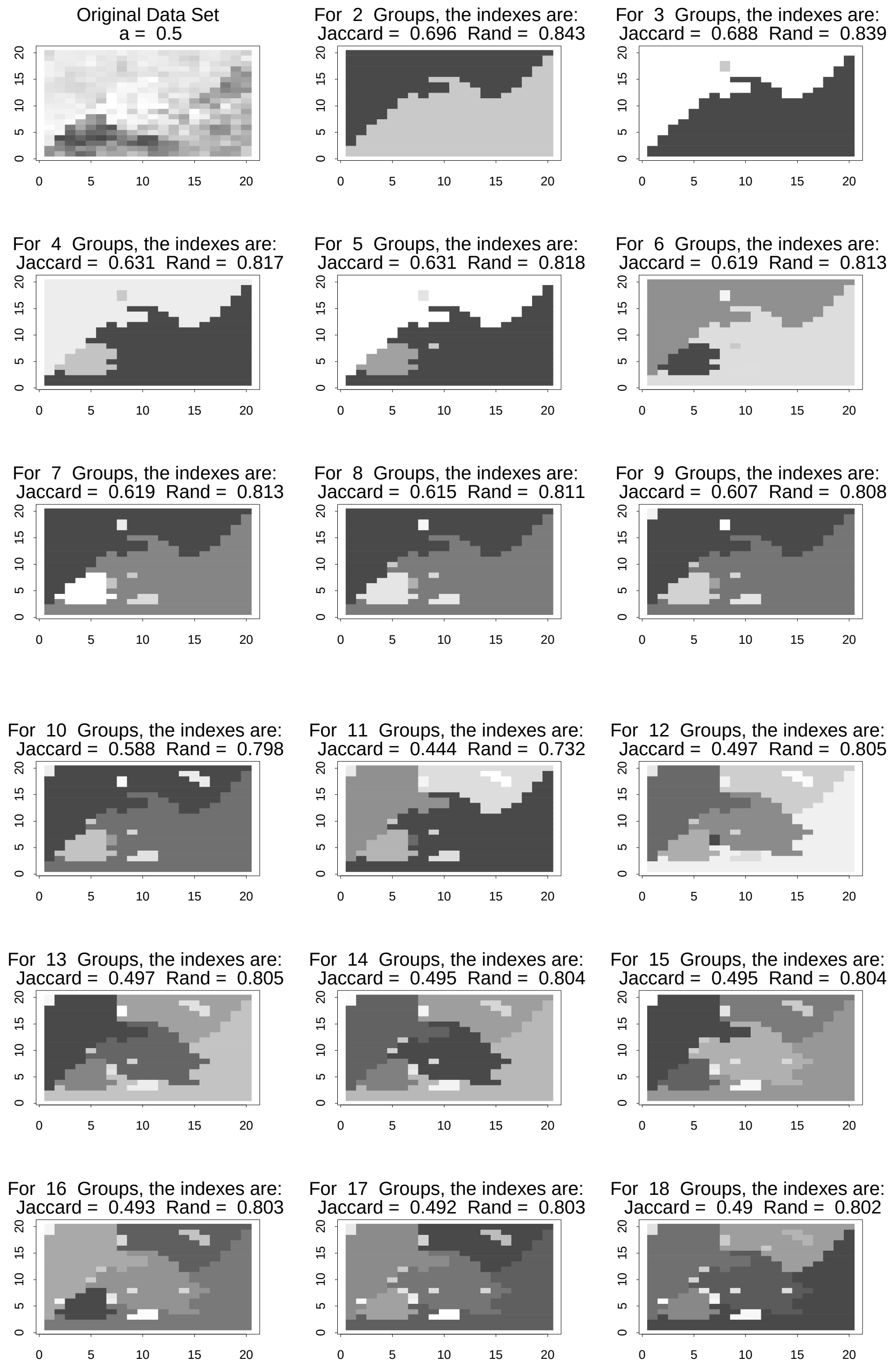

For 24 Groups, the indexes are: Jaccard = 0.213 Rand = 0.701


For 25 Groups, the indexes are: Jaccard = 0.212 Rand = 0.701


For 26 Groups, the indexes are: Jaccard = 0.211 Rand = 0.7

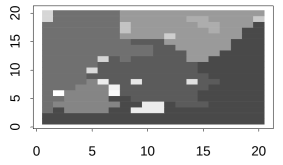

For 27 Groups, the indexes are: Jaccard = 0.207 Rand = 0.699

Data Set 1 Two-Part Method (*a* = 0.50)

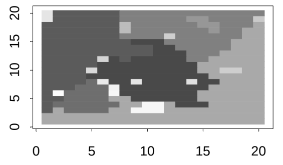


Data Set 1 Two-Part Method (*a* = 0.50) (continued)

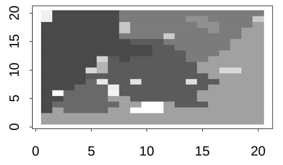
For 19 Groups, the indexes are: Jaccard = 0.486 Rand = 0.801



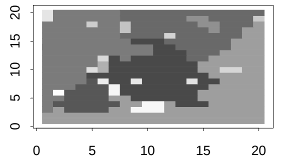
For 20 Groups, the indexes are: Jaccard = 0.478 Rand = 0.798



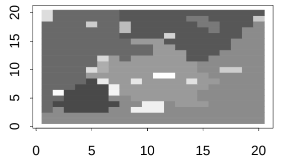
For 21 Groups, the indexes are: Jaccard = 0.475 Rand = 0.797



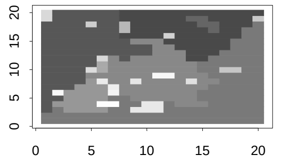
For 22 Groups, the indexes are: Jaccard = 0.472 Rand = 0.796



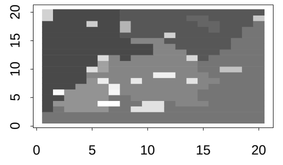
For 23 Groups, the indexes are: Jaccard = 0.469 Rand = 0.795



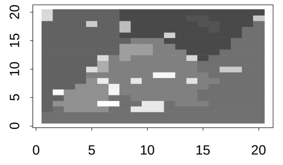
For 24 Groups, the indexes are: Jaccard = 0.468 Rand = 0.794



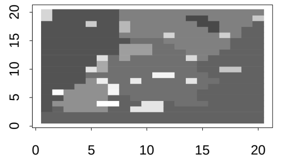
For 25 Groups, the indexes are: Jaccard = 0.466 Rand = 0.793



For 26 Groups, the indexes are: Jaccard = 0.474 Rand = 0.801

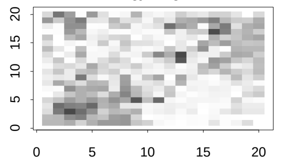


For 27 Groups, the indexes are: Jaccard = 0.472 Rand = 0.8

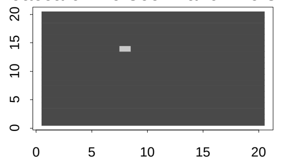


Data Set 2 Two-Part Method (*a* = 0.00)

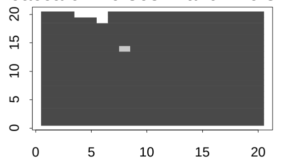
Original Data Set a = 0



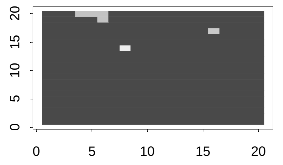
For 2 Groups, the indexes are: Jaccard = 0.369 Rand = 0.371



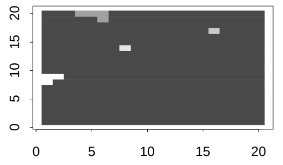
For 3 Groups, the indexes are: Jaccard = 0.365 Rand = 0.375



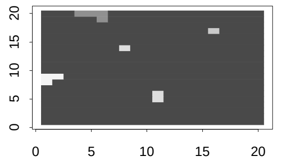
For 4 Groups, the indexes are: Jaccard = 0.364 Rand = 0.376



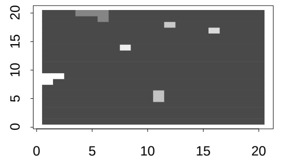
For 5 Groups, the indexes are: Jaccard = 0.361 Rand = 0.379



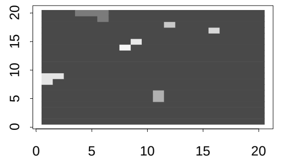
For 6 Groups, the indexes are: Jaccard = 0.363 Rand = 0.385



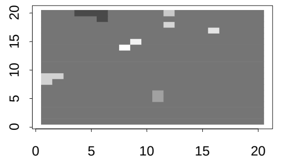
For 7 Groups, the indexes are: Jaccard = 0.362 Rand = 0.386



For 8 Groups, the indexes are: Jaccard = 0.362 Rand = 0.389

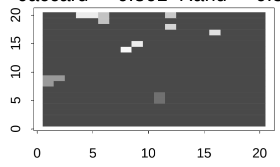


For 9 Groups, the indexes are: Jaccard = 0.361 Rand = 0.39

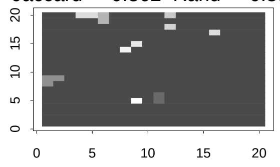


Data Set 2 Two-Part Method (*a* = 0.00) (continued)

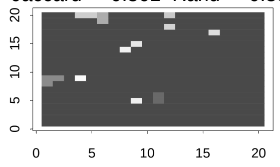
For 10 Groups, the indexes are: Jaccard = 0.361 Rand = 0.39



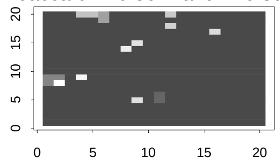
For 11 Groups, the indexes are: Jaccard = 0.362 Rand = 0.393



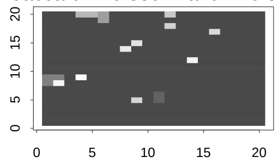
For 12 Groups, the indexes are: Jaccard = 0.361 Rand = 0.394



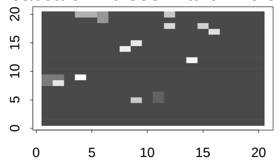
For 13 Groups, the indexes are: Jaccard = 0.36 Rand = 0.395



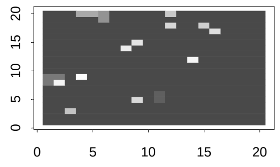
For 14 Groups, the indexes are: Jaccard = 0.359 Rand = 0.396



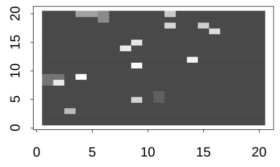
For 15 Groups, the indexes are: Jaccard = 0.358 Rand = 0.397



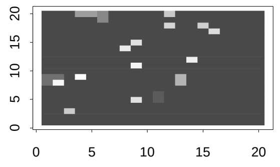
For 16 Groups, the indexes are: Jaccard = 0.357 Rand = 0.398



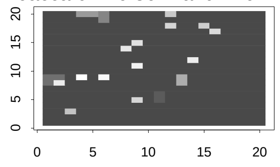
For 17 Groups, the indexes are: Jaccard = 0.358 Rand = 0.401



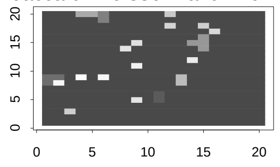
For 18 Groups, the indexes are: Jaccard = 0.359 Rand = 0.407



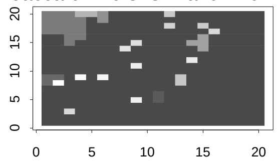
For 19 Groups, the indexes are: Jaccard = 0.36 Rand = 0.411



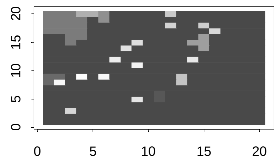
For 20 Groups, the indexes are: Jaccard = 0.356 Rand = 0.414



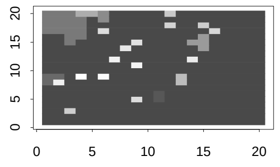
For 21 Groups, the indexes are: Jaccard = 0.343 Rand = 0.436



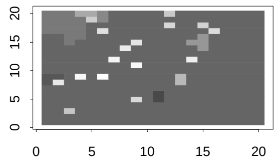
For 22 Groups, the indexes are: Jaccard = 0.344 Rand = 0.439



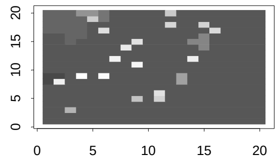
For 23 Groups, the indexes are: Jaccard = 0.343 Rand = 0.44



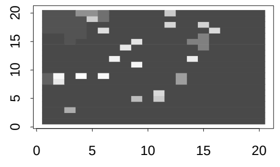
For 24 Groups, the indexes are: Jaccard = 0.342 Rand = 0.441



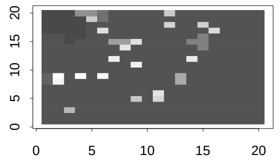
For 25 Groups, the indexes are: Jaccard = 0.342 Rand = 0.441



For 26 Groups, the indexes are: Jaccard = 0.342 Rand = 0.441

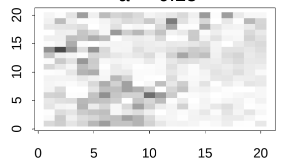


For 27 Groups, the indexes are: Jaccard = 0.341 Rand = 0.444

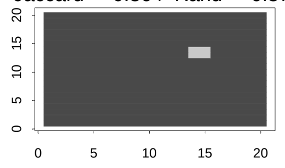


Data Set 3 Two-Part Method (a = 0.25)

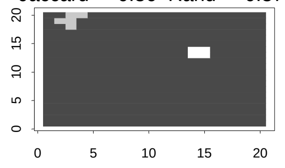
Original Data Set
a = 0.25



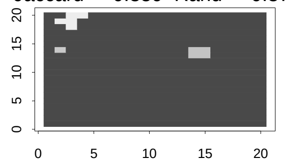
For 2 Groups, the indexes are:
Jaccard = 0.364 Rand = 0.372



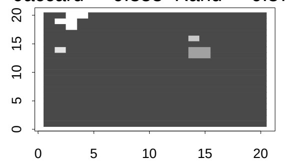
For 3 Groups, the indexes are:
Jaccard = 0.36 Rand = 0.377



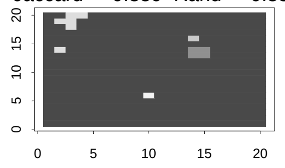
For 4 Groups, the indexes are:
Jaccard = 0.359 Rand = 0.378



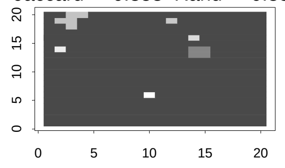
For 5 Groups, the indexes are:
Jaccard = 0.358 Rand = 0.379



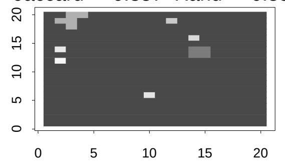
For 6 Groups, the indexes are:
Jaccard = 0.359 Rand = 0.382



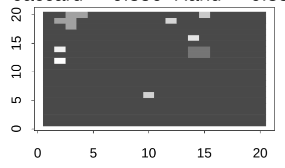
For 7 Groups, the indexes are:
Jaccard = 0.358 Rand = 0.383



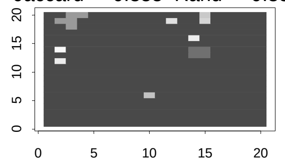
For 8 Groups, the indexes are:
Jaccard = 0.357 Rand = 0.384



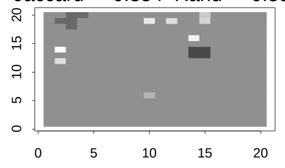
For 9 Groups, the indexes are:
Jaccard = 0.356 Rand = 0.385



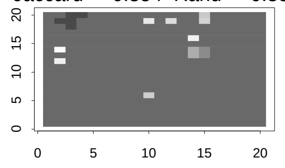
For 10 Groups, the indexes are:
Jaccard = 0.355 Rand = 0.386



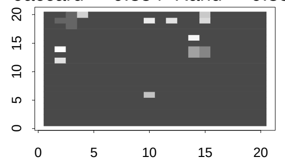
For 11 Groups, the indexes are:
Jaccard = 0.354 Rand = 0.387



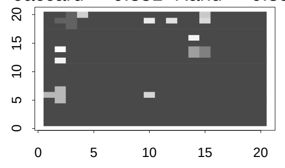
For 12 Groups, the indexes are:
Jaccard = 0.354 Rand = 0.387



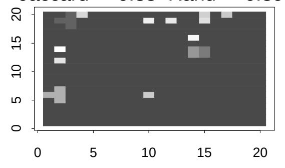
For 13 Groups, the indexes are:
Jaccard = 0.354 Rand = 0.387



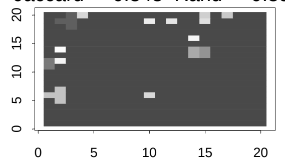
For 14 Groups, the indexes are:
Jaccard = 0.351 Rand = 0.392



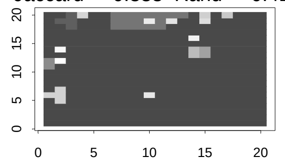
For 15 Groups, the indexes are:
Jaccard = 0.35 Rand = 0.393



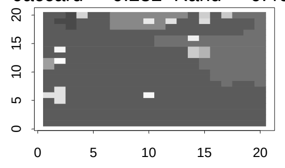
For 16 Groups, the indexes are:
Jaccard = 0.348 Rand = 0.395



For 17 Groups, the indexes are:
Jaccard = 0.338 Rand = 0.417

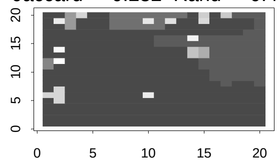


For 18 Groups, the indexes are:
Jaccard = 0.282 Rand = 0.465

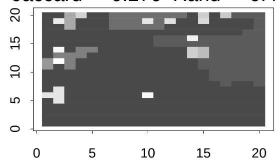


Data Set 3 Two-Part Method (*a* = 0.25) (continued)

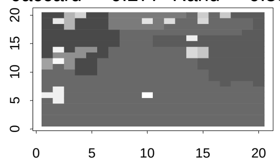
For 19 Groups, the indexes are:
Jaccard = 0.282 Rand = 0.465



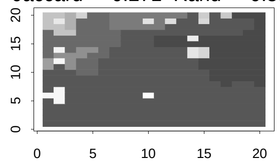
For 20 Groups, the indexes are:
Jaccard = 0.276 Rand = 0.473



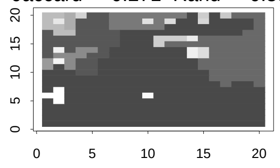
For 21 Groups, the indexes are:
Jaccard = 0.277 Rand = 0.555



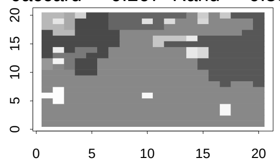
For 22 Groups, the indexes are:
Jaccard = 0.271 Rand = 0.551



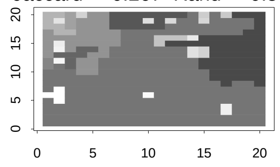
For 23 Groups, the indexes are:
Jaccard = 0.271 Rand = 0.553



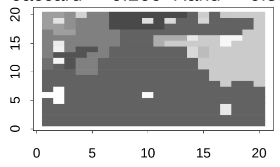
For 24 Groups, the indexes are:
Jaccard = 0.267 Rand = 0.553



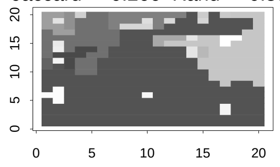
For 25 Groups, the indexes are:
Jaccard = 0.267 Rand = 0.552



For 26 Groups, the indexes are:
Jaccard = 0.266 Rand = 0.553

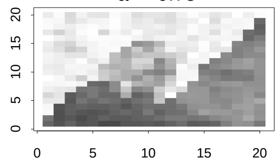


For 27 Groups, the indexes are:
Jaccard = 0.266 Rand = 0.553

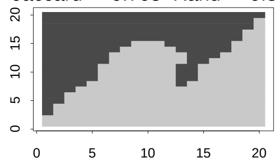


Data Set 4 Two-Part Method (*a* = 0.75)

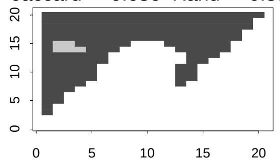
Original Data Set
a = 0.75



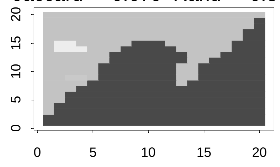
For 2 Groups, the indexes are:
Jaccard = 0.705 Rand = 0.848



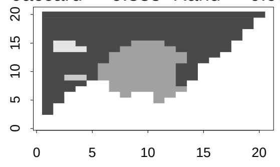
For 3 Groups, the indexes are:
Jaccard = 0.686 Rand = 0.839



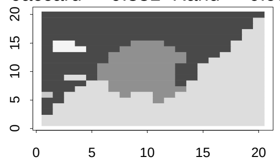
For 4 Groups, the indexes are:
Jaccard = 0.679 Rand = 0.835



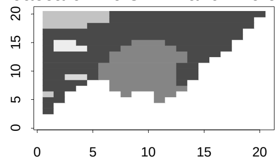
For 5 Groups, the indexes are:
Jaccard = 0.885 Rand = 0.955



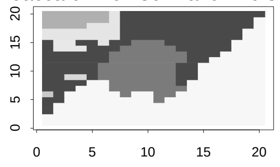
For 6 Groups, the indexes are:
Jaccard = 0.881 Rand = 0.953



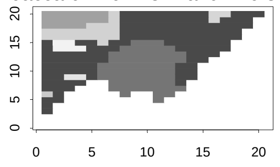
For 7 Groups, the indexes are:
Jaccard = 0.811 Rand = 0.927



For 8 Groups, the indexes are:
Jaccard = 0.736 Rand = 0.898

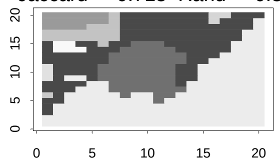


For 9 Groups, the indexes are:
Jaccard = 0.725 Rand = 0.894

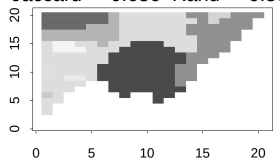


Data Set 4 Two-Part Method (*a* = 0.75) (continued)

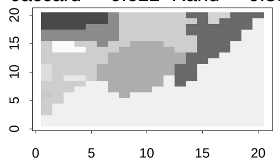
For 10 Groups, the indexes are:
Jaccard = 0.715 Rand = 0.89



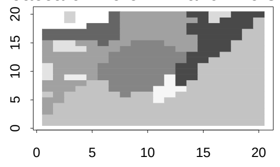
For 11 Groups, the indexes are:
Jaccard = 0.636 Rand = 0.862



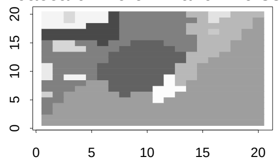
For 12 Groups, the indexes are:
Jaccard = 0.622 Rand = 0.856



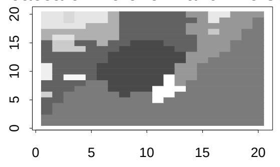
For 13 Groups, the indexes are:
Jaccard = 0.621 Rand = 0.856



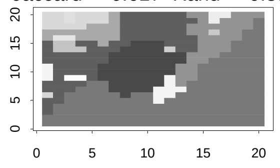
For 14 Groups, the indexes are:
Jaccard = 0.62 Rand = 0.856



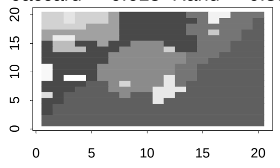
For 15 Groups, the indexes are:
Jaccard = 0.619 Rand = 0.855



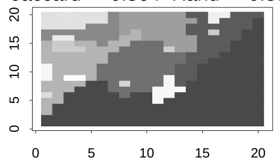
For 16 Groups, the indexes are:
Jaccard = 0.617 Rand = 0.855



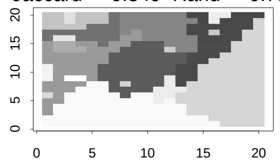
For 17 Groups, the indexes are:
Jaccard = 0.615 Rand = 0.854



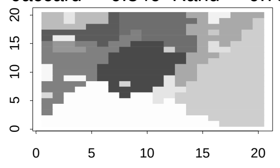
For 18 Groups, the indexes are:
Jaccard = 0.564 Rand = 0.834



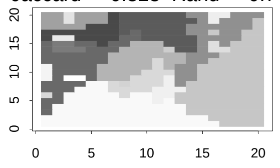
For 19 Groups, the indexes are:
Jaccard = 0.349 Rand = 0.756



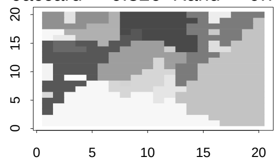
For 20 Groups, the indexes are:
Jaccard = 0.346 Rand = 0.755



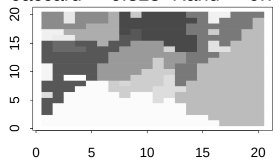
For 21 Groups, the indexes are:
Jaccard = 0.328 Rand = 0.748



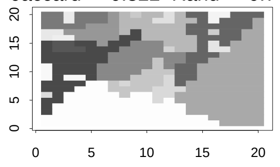
For 22 Groups, the indexes are:
Jaccard = 0.326 Rand = 0.748



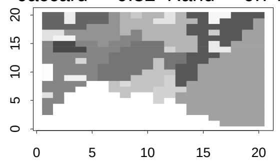
For 23 Groups, the indexes are:
Jaccard = 0.325 Rand = 0.747



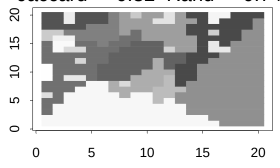
For 24 Groups, the indexes are:
Jaccard = 0.322 Rand = 0.746



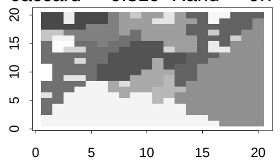
For 25 Groups, the indexes are:
Jaccard = 0.32 Rand = 0.745



For 26 Groups, the indexes are:
Jaccard = 0.32 Rand = 0.745



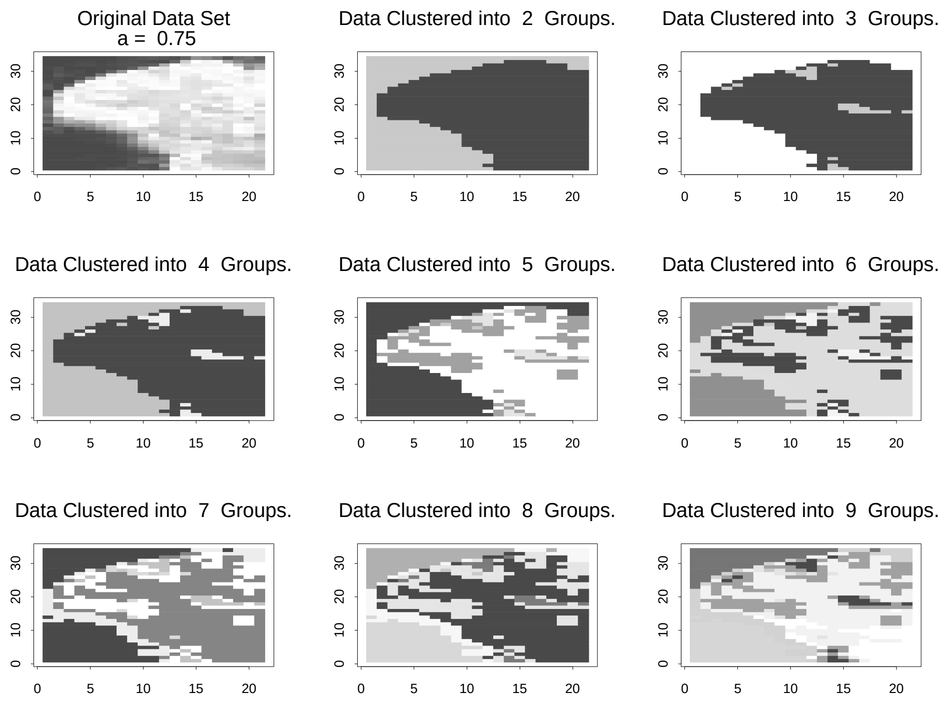
For 27 Groups, the indexes are:
Jaccard = 0.319 Rand = 0.745



A.3 SFMRI Brain Data Graphical Analysis

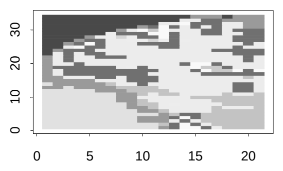
The following images show the clustering resulting from applying the three proposed methods to the SFMRI data from Chapter 5. A weight of 0.75 was selected for *a*.

SFMRI Brain Data Set Basic Window Method (*a* = 0.75)

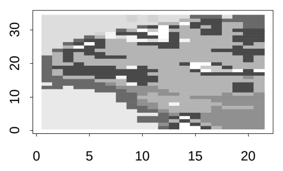


SFMRI Brain Data Set Basic Window Method (*a* = 0.75) (continued)

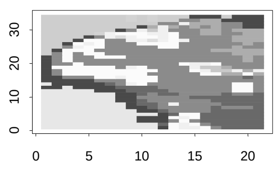
Data Clustered into 10 Groups.



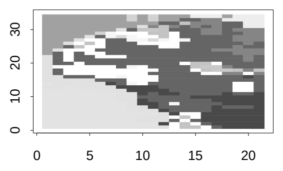
Data Clustered into 11 Groups.



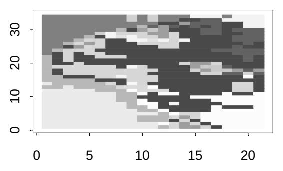
Data Clustered into 12 Groups.



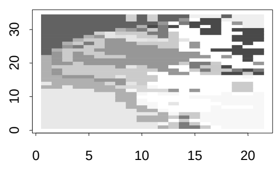
Data Clustered into 13 Groups.



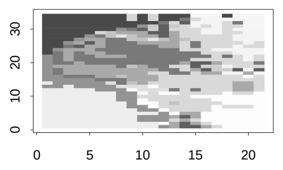
Data Clustered into 14 Groups.



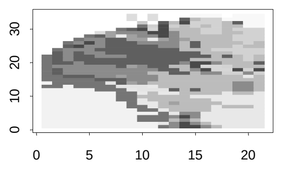
Data Clustered into 15 Groups.



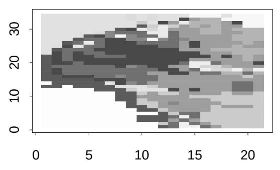
Data Clustered into 16 Groups.



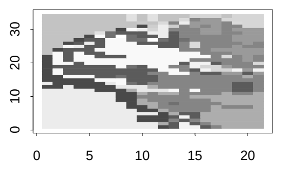
Data Clustered into 17 Groups.



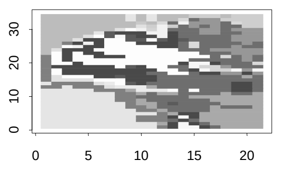
Data Clustered into 18 Groups.



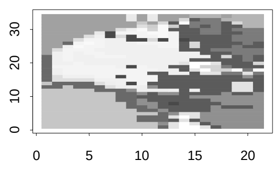
Data Clustered into 19 Groups.



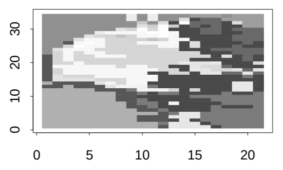
Data Clustered into 20 Groups.



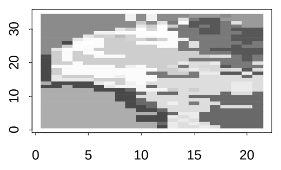
Data Clustered into 21 Groups.



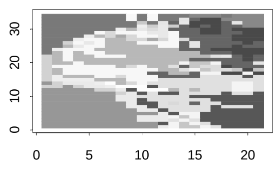
Data Clustered into 22 Groups.



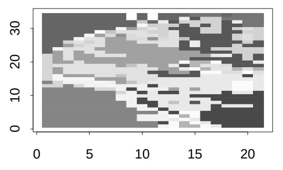
Data Clustered into 23 Groups.



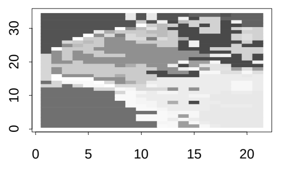
Data Clustered into 24 Groups.



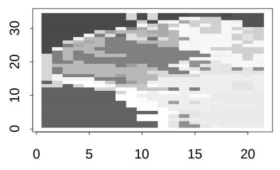
Data Clustered into 25 Groups.



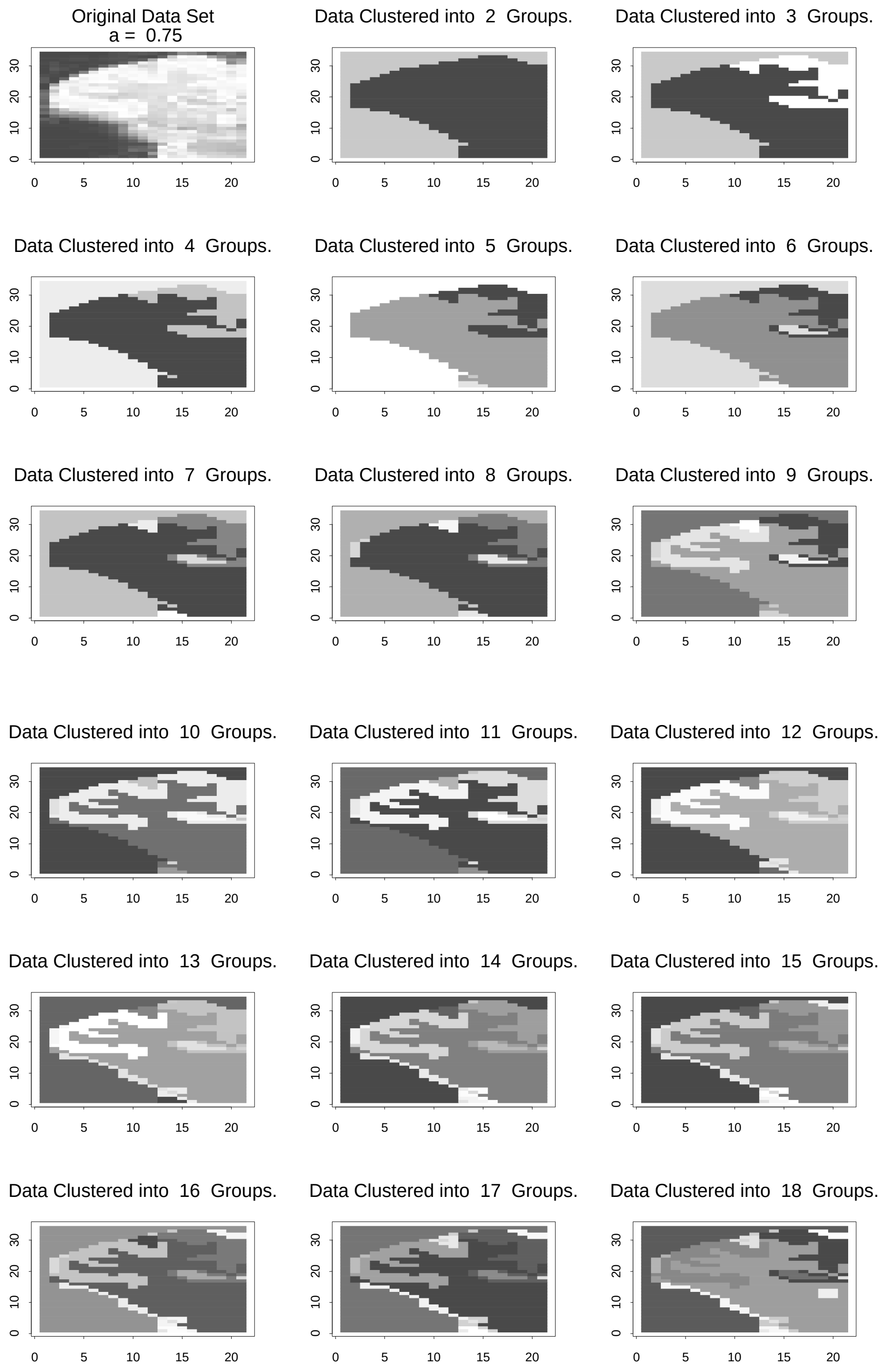
Data Clustered into 26 Groups.



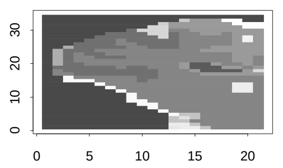
Data Clustered into 27 Groups.

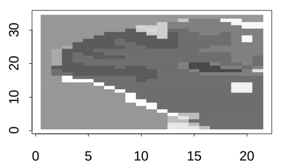


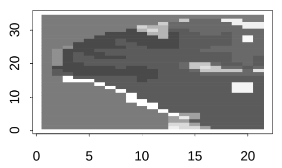
SFMRI Brain Data Set Adjacent Constraint Method (a = 0.75)

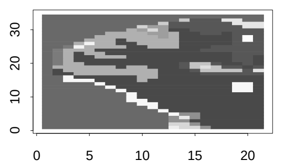


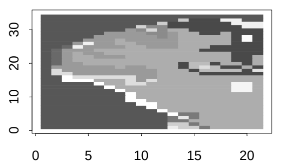
SFMRI Brain Data Set Adjacent Constraint Method (*a* = 0.75) (continued)

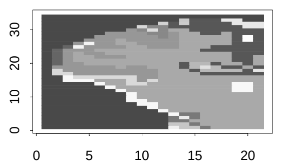

Data Clustered into 19 Groups.

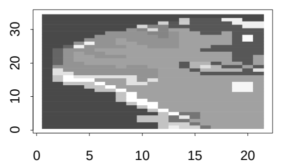

Data Clustered into 20 Groups.

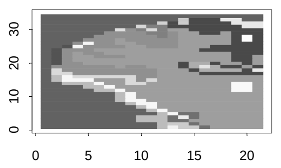

Data Clustered into 21 Groups.

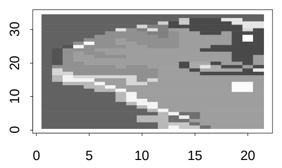

Data Clustered into 22 Groups.


Data Clustered into 23 Groups.

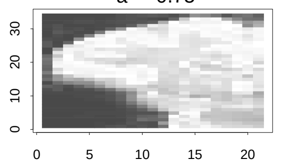

Data Clustered into 24 Groups.

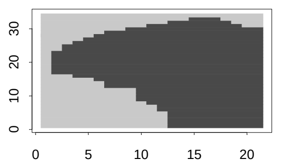

Data Clustered into 25 Groups.

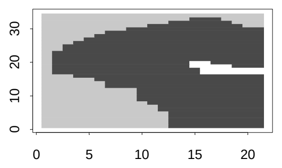

Data Clustered into 26 Groups.

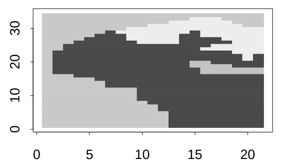

Data Clustered into 27 Groups.

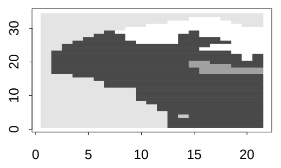
SFMRI Brain Data Set Two-Part Method (*a* = 0.75)

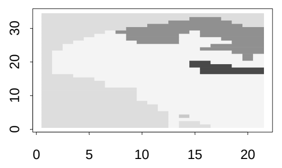

Original Data Set
a = 0.75

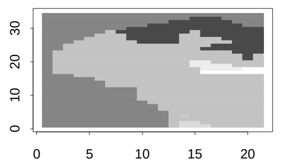

Data Clustered into 2 Groups.

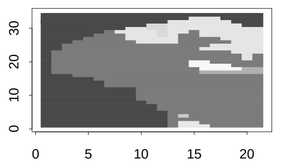

Data Clustered into 3 Groups.

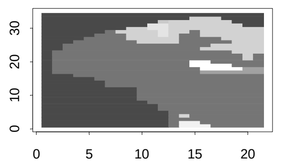

Data Clustered into 4 Groups.


Data Clustered into 5 Groups.


Data Clustered into 6 Groups.

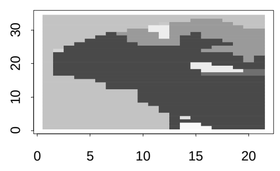

Data Clustered into 7 Groups.


Data Clustered into 8 Groups.

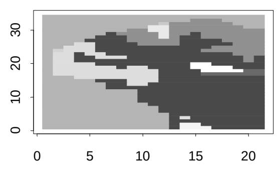

Data Clustered into 9 Groups.

SFMRI Brain Data Set Two-Part Method (a = 0.75) (continued)

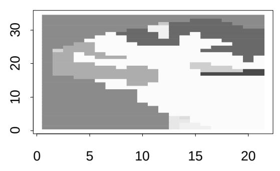
Data Clustered into 10 Groups.



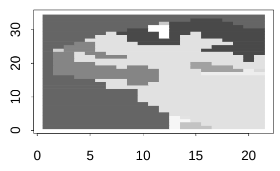
Data Clustered into 11 Groups.



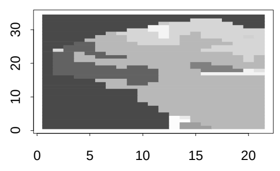
Data Clustered into 12 Groups.



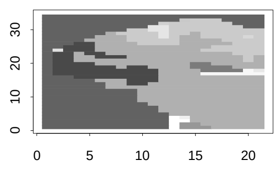
Data Clustered into 13 Groups.



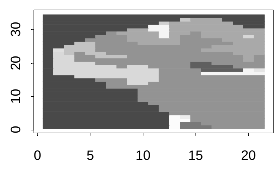
Data Clustered into 14 Groups.



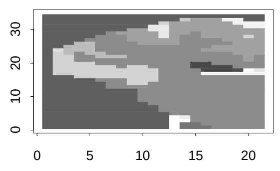
Data Clustered into 15 Groups.



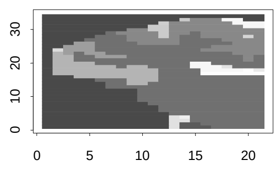
Data Clustered into 16 Groups.



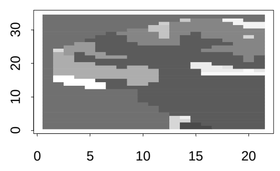
Data Clustered into 17 Groups.



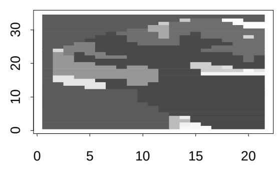
Data Clustered into 18 Groups.



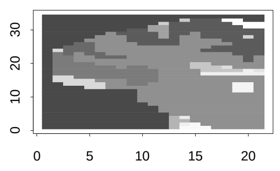
Data Clustered into 19 Groups.



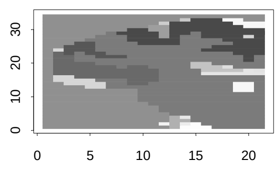
Data Clustered into 20 Groups.



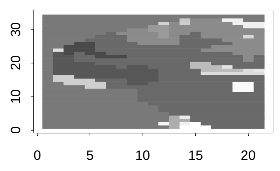
Data Clustered into 21 Groups.



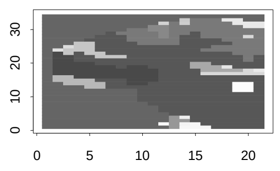
Data Clustered into 22 Groups.



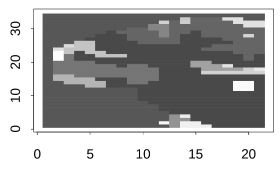
Data Clustered into 23 Groups.



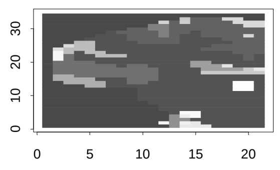
Data Clustered into 24 Groups.



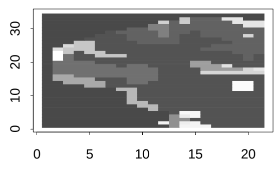
Data Clustered into 25 Groups.



Data Clustered into 26 Groups.



Data Clustered into 27 Groups.



B. SPLUS CODE

B.1 Data Generation

```
##################################################################################
##################################################################################
#for (s in 1:wd)
#{
#  w[s,j[j<(s-10)^3/100+10]] <- g2[s,j[j<(s-10)^3/100+10]]
#  grp[s,j[j<(s-10)^3/100+10]] <- "Grp 2"
#  w[s,j[(s-10)^2/2+(j-10)^2/3<9]] <- g3[s,j[(s-10)^2/2+(j-10)^2/3<9]]
#  grp[s,j[(s-10)^2/2+(j-10)^2/3<9]] <- "Grp 3"
#}

#----------------------------------------------------------------------------#
###The following sequence of code simulates the following regions         ###
#----------------------------------------------------------------------------#

grid <- expand.grid(x=1:wd,y=1:wd)

#----------------------------------------------------------------------------#
###Test Region 1                                                           ###
###1. Region 1 => high mean, moderate variance, moderate range             ###
###1. Region 2 => low mean, moderate variance, moderate range              ###
###1. Region 3 => moderate mean, moderate variance, moderate range         ###
#----------------------------------------------------------------------------#

testdata.1.1 <- matrix(8+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
testdata.1.2 <- matrix(0+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
testdata.1.3 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
testdata.1.grid <- testdata.1.1
testdata.1.grid[grp=="Grp 2"] <- testdata.1.2[grp=="Grp 2"]
testdata.1.grid[grp=="Grp 3"] <- testdata.1.3[grp=="Grp 3"]


#----------------------------------------------------------------------------#
###Test Region 2                                                           ###
###2. Region 1 => moderate mean, moderate variance, moderate range         ###
###2. Region 2 => moderate mean, small variance, moderate range            ###
###2. Region 3 => moderate mean, large variance, moderate range            ###
#----------------------------------------------------------------------------#

testdata.2.1 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
testdata.2.2 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=1,nc=10000),wd,wd)
testdata.2.3 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=5,nc=10000),wd,wd)
testdata.2.grid <- testdata.2.1
testdata.2.grid[grp=="Grp 2"] <- testdata.2.2[grp=="Grp 2"]
testdata.2.grid[grp=="Grp 3"] <- testdata.2.3[grp=="Grp 3"]

#----------------------------------------------------------------------------#
###Test Region 3                                                           ###
###3. Region 1 => moderate mean, moderate variance, small range            ###
###3. Region 2 => moderate mean, moderate variance, large range            ###
###3. Region 3 => moderate mean, moderate variance, moderate range         ###
#----------------------------------------------------------------------------#

testdata.3.1 <- matrix(4+rfsim(grid,covfun=exp.cov,range=1,sill=3,nc=10000),wd,wd)
testdata.3.2 <- matrix(4+rfsim(grid,covfun=exp.cov,range=5,sill=3,nc=10000),wd,wd)
testdata.3.3 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
testdata.3.grid <- testdata.3.1
testdata.3.grid[grp=="Grp 2"] <- testdata.3.2[grp=="Grp 2"]
testdata.3.grid[grp=="Grp 3"] <- testdata.3.3[grp=="Grp 3"]


#--------------------------------------------------------------------------#
###Test Region 4                                                         ###
###4. Region 1 => high mean, moderate variance, small range              ###
###4. Region 2 => low mean, small variance, large range                  ###
###4. Region 3 => moderate mean, large variance, moderate range          ###
#--------------------------------------------------------------------------#

testdata.4.1 <- matrix(8+rfsim(grid,covfun=exp.cov,range=1,sill=3,nc=10000),wd,wd)
testdata.4.2 <- matrix(0+rfsim(grid,covfun=exp.cov,range=5,sill=1,nc=10000),wd,wd)
testdata.4.3 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=5,nc=10000),wd,wd)
testdata.4.grid <- testdata.4.1
testdata.4.grid[grp=="Grp 2"] <- testdata.4.2[grp=="Grp 2"]
testdata.4.grid[grp=="Grp 3"] <- testdata.4.3[grp=="Grp 3"]


#--------------------------------------------------------------------------#
###Chart all regions                                                     ###
#--------------------------------------------------------------------------#

par(mfrow=c(2,2))
image(testdata.1.grid)
title("Image of Test Data 1")
image(testdata.2.grid)
title("Image of Test Data 2")
image(testdata.3.grid)
title("Image of Test Data 3")
image(testdata.4.grid)
title("Image of Test Data 4")
persp(testdata.1.grid)
title("Image of Test Data 1")
persp(testdata.2.grid)
title("Image of Test Data 2")
persp(testdata.3.grid)
title("Image of Test Data 3")
persp(testdata.4.grid)
title("Image of Test Data 4")
```

B.2 General Functions

```
#-------------------------------------------------------------------------#
###Defined Functions                                                   ###
#-------------------------------------------------------------------------#

#-------------------------------------------------------------------------#
###The following three functions are estimators of the variogram         ###
#-------------------------------------------------------------------------#

gamma.classic <- function(zdat) {
   nh <- length(zdat)
   gval <- .5*(sum(zdat*zdat))/nh
   return (gval)
}

gamma.bar <- function(zdat) {
   nh <- length(zdat)
   bh <- 0.457
   gval <- .5*((sum(sqrt(abs(zdat)))/nh)^4/(bh+.494/nh))
   return (gval)
}

gamma.tilda <- function(zdat) {
   nh <- length(zdat)
   bh <- 0.457
   gval <- .5*((median(sqrt(abs(zdat)))^4)/(bh+.494/nh))
   return (gval)
}

#-------------------------------------------------------------------------#
###Function to test if ind < 0                                         ###
#-------------------------------------------------------------------------#

test.fun <- function (ind) {
   if (ind < 0) return (T)
   return (F)
}

#-------------------------------------------------------------------------#
###Calculates groups to be similar to S-Plus output                    ###
### (for use in hclust function)                                       ###
#-------------------------------------------------------------------------#

order.fun <- function (t,m,i){
   index.history <- NULL
   index.history <- c(i,index.history)
   column.history <- NULL
   column.history <- c(1,column.history)


   while (length(t) < i+1){
      while (m[index.history[1],column.history[1]] > 0) {
         index.history <- c(m[index.history[1],column.history[1]],index.history)
         column.history <- c(1,column.history)
      }

      t <- c(t,-m[index.history[1],column.history[1]])

      if (column.history[1] == 1) {
         column.history[1] <- 2
      }
      else {
          while ((column.history[1] == 2) && (length(column.history >0))) {
              index.history <- index.history[-1]
              column.history <- column.history[-1]
          }
          column.history[1] <- 2
      }

    }

    return(t)


  }


  ord.fun <- function (t,m,i) {
    ind1 <- m[i,1]
    if (test.fun(ind1)) t <- c(t,-ind1) else t <- ord.fun(t,m,ind1)

    ind2 <- m[i,2]
    if (test.fun(ind2)) t <- c(t,-ind2) else t <- ord.fun(t,m,ind2)

    return(t)
  }

  #---------------------------------------------------------------------------#
  ###Calculated the Jaccard and Rand Indexes                                  ###
  #---------------------------------------------------------------------------#

  get.index <- function(group.mat, cluster.mat) {
    v.known <- c(group.mat)
    v.clust <- c(cluster.mat)
    nn <- length(v.known)
    mat.known <- matrix(-1,nn,nn)
    mat.clust <- matrix(-1,nn,nn)

    for (i in 1:(nn-1)) {
       mat.known[i,(i+1):nn] <- (v.known[(i+1):nn] == v.known[i])
       mat.clust[i,(i+1):nn] <- (v.clust[(i+1):nn] == v.clust[i])
    }

    count.mat <- mat.known + mat.clust
    a <- length(count.mat[count.mat==2])
    bc.sum <- length(count.mat[count.mat==1])
    d <- length(count.mat[count.mat==0])    index <-
c(Jaccard=a/(a+bc.sum),Rand=(a+d)/(a+bc.sum+d))
    return(index)
  }
```

```
#----------------------------------------------------------------------------#
###Constrained Clustering (using complete linkage, or maximum, as distance ###
###criteria)                                                               ###
#----------------------------------------------------------------------------#
hclust.adj.com <- function(data.vec,nr,nc,adjacent.distance) {

   ntot <- nr*nc

#----------------------------------------------------------------------------#
###general positions for any data matrix                                   ###
###data.pos represents a matrix with the pairings of each point of the     ###
###data matrix (i.e. column 1 provides the row number, column 2 the column###
###number)                                                                 ###
#----------------------------------------------------------------------------#

   data.pos <- matrix(c(rep(1:nr, times=nc),rep(1:nc,
each=nr)),ncol=2,byrow=F,dimnames=list(NULL,c("row","col")))

#----------------------------------------------------------------------------#
###temp.dist contains the distances between each data point of the data   ###
###matrix                                                                  ###
#----------------------------------------------------------------------------#

   temp.dist <- dist(data.vec)

#----------------------------------------------------------------------------#
###temp.dist.pos contains the distances between each physical location in ###
###the data matrix                                                         ###
#----------------------------------------------------------------------------#

   temp.dist.pos <- dist(data.pos)

#----------------------------------------------------------------------------#
###lists the observation pairs that correspond to the distances from the  ###
###S-PLUS dist function                                                    ###
#----------------------------------------------------------------------------#

   tt.c1 <- tt.c2 <- NULL
   for (i in 1:(ntot-1)) {
      temp.seq <- seq(1:(ntot-i))+i
      tt.c2 <- c(tt.c2,temp.seq)
      tt.c1 <- c(tt.c1,numeric(length(temp.seq))+i)
   }
   tt <- matrix(-c(tt.c1,tt.c2),ncol=2,byrow=F)

#----------------------------------------------------------------------------#
###temp.merge, temp.height, and temp.order match the output from the      ###
###S-PLUS hclust function dist.mat contains the distance between data     ###
###points, physical locations and the data index for which the distances  ###
###correspond                                                              ###
#----------------------------------------------------------------------------#

   temp.merge <- matrix(0,nrow=ntot-1,ncol=2)
   temp.height <- temp.order <- numeric(length = ntot-1)
   dist.mat <-
matrix(c(temp.dist,temp.dist.pos,tt),ncol=4,byrow=F,dimnames=list(NULL,c("dist","pos
dist","","")))

#----------------------------------------------------------------------------#
###This algorithm clusters based on maximum distance                       ###
#----------------------------------------------------------------------------#
```

```
  for (i in 1:(ntot-1)) {
    if (length(dist.mat) > 4) {

#-------------------------------------------------------------------------#
###records the point number and row and column coordinates of the         ###
###observations with the smallest distance.                                ###
###Note that the physical locations must be adjacent                       ###
### (i.e. < adjacent.distance)                                             ###
#-------------------------------------------------------------------------#

        min.dist <- min(dist.mat[(dist.mat[,2] <
    adjacent.distance)&(dist.mat[,1]>0),1])
        temp.pos <- which(dist.mat[,1]==min.dist)[1]
        temp.merge[i,] <- dist.mat[temp.pos,3:4][order(dist.mat[temp.pos,3:4])]

#-------------------------------------------------------------------------#
###records the min distance between points                                 ###
#-------------------------------------------------------------------------#

        temp.height[i] <- min.dist

#-------------------------------------------------------------------------#
###adjust the distance matrix dist.mat by combining the joined groups     ###
#-------------------------------------------------------------------------#

        dist.mat <-
    dist.mat[!(((dist.mat[,3]==temp.merge[i,][1])&(dist.mat[,4]==temp.merge[i,][2]))|
    ((dist.mat[,3]==temp.merge[i,][2])&(dist.mat[,4]==temp.merge[i,][1]))),]

#-------------------------------------------------------------------------#
###change the negative point number to a positive group number            ###
### (to match S-Plus requirements)                                         ###
#-------------------------------------------------------------------------#

        temp.pairs <- temp.merge[i,]
        dist.mat[(dist.mat[,3]==temp.pairs[1])|(dist.mat[,3]==temp.pairs[2]),3] <- i
        dist.mat[(dist.mat[,4]==temp.pairs[1])|(dist.mat[,4]==temp.pairs[2]),4] <- i

#-------------------------------------------------------------------------#
###recalculate the distances between groups in dist.mat                   ###
###the measured distance between groups is the max distance from the point###
###in question to any point within the group.                              ###
###the physical distance is the minimum distance between groups           ###
#-------------------------------------------------------------------------#

        pos.lst <- which((dist.mat[,3]==i)|(dist.mat[,4]==i))
        delete.row <- NULL
        for (k in 1:(length(pos.lst))) {
           temp.pairs <- dist.mat[pos.lst[k],3:4]
           ind.pos <-
    which(((dist.mat[pos.lst,3]==temp.pairs[1])&(dist.mat[pos.lst,4]==temp.pairs[2]))
    |((dist.mat[pos.lst,3]==temp.pairs[2])&(dist.mat[pos.lst,4]==temp.pairs[1])))
           dist.mat[pos.lst[ind.pos],1] <- max(dist.mat[pos.lst[ind.pos],1])
           dist.mat[pos.lst[ind.pos],2] <- min(dist.mat[pos.lst[ind.pos],2])
           delete.row <- c(delete.row,pos.lst[ind.pos[2:length(ind.pos)]])
        }
        dist.mat <- dist.mat[-unique(delete.row),]
    }

    else {
        temp.merge[i,] <- dist.mat[3:4]
        temp.height[i] <- dist.mat[1]
    }
```

```
   }

#-----------------------------------------------------------------------------#
###combine data into a list---similar to output of hclust function in S-Plus###
#-----------------------------------------------------------------------------#

   mlen <- length(temp.merge[,1])
   temp.order <- NULL
   temp.order <- order.fun(temp.order,temp.merge,mlen)
   link.adj <- list(temp.merge,temp.height,temp.order)
   names(link.adj) <- c("merge","height","order")

   return (link.adj)
}


#-----------------------------------------------------------------------------#
###Constrained Clustering (using average linkage as distance criteria)     ###
#-----------------------------------------------------------------------------#

hclust.adj.ave <- function(data.vec,nr,nc,adjacent.distance) {

   ntot <- nr*nc

#-----------------------------------------------------------------------------#
###general positions for any data matrix                                   ###
###data.pos represents a matrix with the pairings of each point of the     ###
###data matrix (i.e. column 1 provides the row number, column 2 the        ###
###column number)                                                          ###
#-----------------------------------------------------------------------------#

   data.pos <- matrix(c(rep(1:nr, times=nc),rep(1:nc,
each=nr)),ncol=2,byrow=F,dimnames=list(NULL,c("row","col")))

#-----------------------------------------------------------------------------#
###temp.dist contains the distances between each data point of the data    ###
###matrix                                                                  ###
#-----------------------------------------------------------------------------#

   temp.dist <- dist(data.vec)

#-----------------------------------------------------------------------------#
###temp.dist.pos contains the distances between each physical location in ###
###the data matrix                                                         ###
#-----------------------------------------------------------------------------#

   temp.dist.pos <- dist(data.pos)

#-----------------------------------------------------------------------------#
###lists the observation pairs that correspond to the distances from the  ###
###S-PLUS dist function                                                    ###
#-----------------------------------------------------------------------------#

   tt.c1 <- tt.c2 <- NULL
   for (i in 1:(ntot-1)) {
      temp.seq <- seq(1:(ntot-i))+i
      tt.c2 <- c(tt.c2,temp.seq)
      tt.c1 <- c(tt.c1,numeric(length(temp.seq))+i)
   }
   tt <- matrix(-c(tt.c1,tt.c2),ncol=2,byrow=F)

#-----------------------------------------------------------------------------#
###temp.merge, temp.height, and temp.order match the output from the       ###
```

```
###S-PLUS hclust function dist.mat contains the distance between data      ###
###points, physical locations and the data index for which the distances  ###
###correspond                                                              ###
#--------------------------------------------------------------------------#

  temp.merge <- matrix(0,nrow=ntot-1,ncol=2)
  temp.height <- temp.order <- numeric(length = ntot-1)
  dist.mat <-
  matrix(c(temp.dist,temp.dist.pos,tt),ncol=4,byrow=F,dimnames=list(NULL,c("dist","po
  s dist","","")))

#--------------------------------------------------------------------------#
###This algorithm clusters based on average distance                       ###
#--------------------------------------------------------------------------#

  for (i in 1:(ntot-1)) {
    if (length(dist.mat) > 4) {

#--------------------------------------------------------------------------#
###records the point number and row and column coordinates of the          ###
###observations with the smallest distance.                                ###
###Note that the physical locations must be adjacent                       ###
### (i.e. < adjacent.distance)                                             ###
#--------------------------------------------------------------------------#

       min.dist <- min(dist.mat[(dist.mat[,2] <
  adjacent.distance)&(dist.mat[,1]>0),1])
       temp.pos <- which(dist.mat[,1]==min.dist)[1]
       temp.merge[i,] <- dist.mat[temp.pos,3:4][order(dist.mat[temp.pos,3:4])]

#--------------------------------------------------------------------------#
###records the min distance between points                                 ###
#--------------------------------------------------------------------------#

       temp.height[i] <- min.dist

#--------------------------------------------------------------------------#
###adjust the distance matrix dist.mat by combining the joined groups      ###
#--------------------------------------------------------------------------#

       dist.mat <-
  dist.mat[!(((dist.mat[,3]==temp.merge[i,][1])&(dist.mat[,4]==temp.merge[i,][2]))|((
  dist.mat[,3]==temp.merge[i,][2])&(dist.mat[,4]==temp.merge[i,][1]))),]

#--------------------------------------------------------------------------#
###change the negative point number to a positive group number             ###
### (to match S-Plus requirements)                                         ###
#--------------------------------------------------------------------------#

       temp.pairs <- temp.merge[i,]
       dist.mat[(dist.mat[,3]==temp.pairs[1])|(dist.mat[,3]==temp.pairs[2]),3] <- i
       dist.mat[(dist.mat[,4]==temp.pairs[1])|(dist.mat[,4]==temp.pairs[2]),4] <- i

#--------------------------------------------------------------------------#
###recalculate the distances between groups in dist.mat                    ###
###the measured distance between groups is the average distance from the  ###
###point in question to any point within the group.                        ###
###the physical distance is the minimum distance between groups            ###
#--------------------------------------------------------------------------#

       pos.lst <- which((dist.mat[,3]==i)|(dist.mat[,4]==i))
       delete.row <- NULL
       for (k in 1:(length(pos.lst))) {
            temp.pairs <- dist.mat[pos.lst[k],3:4]
            ind.pos <-
which(((dist.mat[pos.lst,3]==temp.pairs[1])&(dist.mat[pos.lst,4]==temp.pairs[2]))|((
dist.mat[pos.lst,3]==temp.pairs[2])&(dist.mat[pos.lst,4]==temp.pairs[1])))
            dist.mat[pos.lst[ind.pos],1] <- mean(dist.mat[pos.lst[ind.pos],1])
            dist.mat[pos.lst[ind.pos],2] <- min(dist.mat[pos.lst[ind.pos],2])
            delete.row <- c(delete.row,pos.lst[ind.pos[2:length(ind.pos)]])
         }
         dist.mat <- dist.mat[-unique(delete.row),]
      }
      else {
         temp.merge[i,] <- dist.mat[3:4]
         temp.height[i] <- dist.mat[1]
      }
   }

#----------------------------------------------------------------------------#
###combine data into a list---similar to output of hclust function in S-Plus###
#----------------------------------------------------------------------------#

   mlen <- length(temp.merge[,1])
   temp.order <- NULL
   temp.order <- order.fun(temp.order,temp.merge,mlen)
   link.adj <- list(temp.merge,temp.height,temp.order)
   names(link.adj) <- c("merge","height","order")

   return (link.adj)
}
```

B.3 Basic Window (Central) Method Function

```
#-----------------------------------------------------------------------------#
###Beginning of Code for the Central Method Function                       ###
#-----------------------------------------------------------------------------#

Central.Method.Cluster <- function(data.grid, distance.type, estimator.type){
#-----------------------------------------------------------------------------#
###"distance.type" may be equal to "average" or "compact"                    ###
###"estimator.type" may be equal to "gamma bar", "gamma tilda" or "classical"###
#-----------------------------------------------------------------------------#

data.rows <- nrow(data.grid)
data.columns <- ncol(data.grid)
sz <- data.rows*data.columns

#-----------------------------------------------------------------------------#
###defining the measurement value, and the location (row/column number)    ###
###denoted by x and y                                                       ###
#-----------------------------------------------------------------------------#

zval <- c(data.grid)
c.row <- diag(c(1:data.rows))
c.col <- diag(c(1:data.columns))
iwd.row <- diag(c(1),data.columns)
iwd.col <- diag(c(1),data.rows)
yval <- diag(kronecker(c.col,iwd.col))
xval <- diag(kronecker(iwd.row,c.row))

#-----------------------------------------------------------------------------#
###Step 1 -- Get estimates of the variogram                                 ###
#-----------------------------------------------------------------------------#

#-----------------------------------------------------------------------------#
###Estimates of the variogram are calculated at lags 1 through 7           ###
###For each cell in the data grid, this function determines the other data###
###points to be incluced in the calculations of the variogram at each      ###
###lag (1 through 7)                                                        ###
#-----------------------------------------------------------------------------#
for (i in 1:7) {

   for (cc in 1:data.columns) {
      h.max <- min(i,data.columns-cc)
      h.min <- max(1-cc,-i)

      for (rr in 1:data.rows) {
         v.max <- min(i,data.rows-rr)
         v.min <- max(1-rr,-i)

         temp.pos.c <-
   diag(kronecker(diag(c(h.min:h.max)),diag(length(v.min:v.max))))
         temp.pos.r <-
   diag(kronecker(diag(length(h.min:h.max)),diag(c(v.min:v.max))))
         temp.dist <-
   matrix(c(temp.pos.r,temp.pos.c),ncol=2,byrow=F,dimnames=list(NULL,c("row","col"))
   )
         dist.pos <- sqrt(temp.dist[,1]^2+temp.dist[,2]^2)
         temp.window <- temp.dist[(dist.pos > i-.5)&(dist.pos < i+.5),]
         temp.window[,1] <- temp.window[,1]+rr
         temp.window[,2] <- temp.window[,2]+cc

         diff.use <- data.grid[temp.window]-data.grid[rr,cc]
```

```
          temp.b <- c(temp.b, gamma.bar(diff.use))
          temp.t <- c(temp.t, gamma.tilda(diff.use))
          temp.cl <- c(temp.cl, gamma.classic(diff.use))
       }
    }
}

#-----------------------------------------------------------------------------#
###estimates of the variogram at each lag are split into separate arrays   ###
### (1 through 7)                                                          ###
#-----------------------------------------------------------------------------#

if (estimator.type == "gamma bar") {
#Gamma Bar Estmator
    lag.b1 <- temp.b[1:sz]
    lag.b2 <- temp.b[(sz+1):(2*sz)]
    lag.b3 <- temp.b[(2*sz+1):(3*sz)]
    lag.b4 <- temp.b[(3*sz+1):(4*sz)]
    lag.b5 <- temp.b[(4*sz+1):(5*sz)]
    lag.b6 <- temp.b[(5*sz+1):(6*sz)]
    lag.b7 <- temp.b[(6*sz+1):(7*sz)]

##90th Lag Percentile
    q90.b <- numeric(sz)
    for (i in 1:sz) {
       q90.b[i] <-
quantile(c(lag.b1[i],lag.b2[i],lag.b3[i],lag.b4[i],lag.b5[i],lag.b6[i],lag.b7[i]),.9
,na.rm=T)
    }

    measure.b <-
matrix(c(xval,yval,zval,lag.b1,lag.b2,lag.b3,lag.b4,q90.b),nrow=sz,ncol=8,byrow=F,di
mnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90")))
    mstand.b <- scale(measure.b,center=T,scale=T)

    return (mstand.b)

}

if (estimator.type == "gamma tilda") {
#Gamma Tilda Estmator
    lag.t1 <- temp.t[1:sz]
    lag.t2 <- temp.t[(sz+1):(2*sz)]
    lag.t3 <- temp.t[(2*sz+1):(3*sz)]
    lag.t4 <- temp.t[(3*sz+1):(4*sz)]
    lag.t5 <- temp.t[(4*sz+1):(5*sz)]
    lag.t6 <- temp.t[(5*sz+1):(6*sz)]
    lag.t7 <- temp.t[(6*sz+1):(7*sz)]

##90th Lag Percentile
    q90.t <- numeric(sz)
    for (i in 1:sz) {
       q90.t[i] <-
quantile(c(lag.t1[i],lag.t2[i],lag.t3[i],lag.t4[i],lag.t5[i],lag.t6[i],lag.t7[i]),.9
,na.rm=T)
    }

    measure.t <-
matrix(c(xval,yval,zval,lag.t1,lag.t2,lag.t3,lag.t4,q90.t),nrow=sz,ncol=8,byrow=F,di
mnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90")))
    mstand.t <- scale(measure.t,center=T,scale=T)

    return (mstand.t)
```

```
   }

if (estimator.type == "classical") {
#Classical Estmator
   lag.cl1 <- temp.cl[1:sz]
   lag.cl2 <- temp.cl[(sz+1):(2*sz)]
   lag.cl3 <- temp.cl[(2*sz+1):(3*sz)]
   lag.cl4 <- temp.cl[(3*sz+1):(4*sz)]
   lag.cl5 <- temp.cl[(4*sz+1):(5*sz)]
   lag.cl6 <- temp.cl[(5*sz+1):(6*sz)]
   lag.cl7 <- temp.cl[(6*sz+1):(7*sz)]

##90th Lag Percentile
   q90.cl <- numeric(sz)
   for (i in 1:sz) {
      q90.cl[i] <-
   quantile(c(lag.cl1[i],lag.cl2[i],lag.cl3[i],lag.cl4[i],lag.cl5[i],lag.cl6[i],lag.
   cl7[i]),.9,na.rm=T)
   }

   measure.cl <-
   matrix(c(xval,yval,zval,lag.cl1,lag.cl2,lag.cl3,lag.cl4,q90.cl),nrow=sz,ncol=8,by
   row=F,dimnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90"
   )))
   mstand.cl <- scale(measure.cl,center=T,scale=T)

   return (mstand.cl)

}

}
```

B.4 Adjacent Constraint Method Function

```
#---------------------------------------------------------------------------#
###Beginning of Code for Adjacent Constraint Method Function              ###
#---------------------------------------------------------------------------#

Adjacent.Constraint.Cluster <- function(data.grid, distance.type, adjacent.distance,
maximum.groups, estimator.type){

data.rows <- nrow(data.grid)
data.columns <- ncol(data.grid)
sz <- data.rows*data.columns

#---------------------------------------------------------------------------#
###defining the measurement value, and the location (row/column number)   ###
###denoted by x and y                                                     ###
#---------------------------------------------------------------------------#
zval <- c(data.grid)
c.row <- diag(c(1:data.rows))
c.col <- diag(c(1:data.columns))
iwd.row <- diag(c(1),data.columns)
iwd.col <- diag(c(1),data.rows)
yval <- diag(kronecker(c.col,iwd.col))
xval <- diag(kronecker(iwd.row,c.row))

#---------------------------------------------------------------------------#
###Step 1 -- Get estimates of the variogram                               ###
#---------------------------------------------------------------------------#

temp.b <- numeric(0)
temp.t <- numeric(0)
temp.cl <- numeric(0)

#---------------------------------------------------------------------------#
###Estimates of the variogram are calculated at lags 1 through 7          ###
###For each cell in the data grid, this function determines the other data###
###points to be incluced in the calculations of the variogram at each lag ###
### (1 through 7)                                                         ###
#---------------------------------------------------------------------------#
for (i in 1:7) {

   for (cc in 1:data.columns) {
      h.max <- min(i,data.columns-cc)
      h.min <- max(1-cc,-i)

      for (rr in 1:data.rows) {
            v.max <- min(i,data.rows-rr)
            v.min <- max(1-rr,-i)

            temp.pos.c <-
diag(kronecker(diag(c(h.min:h.max)),diag(length(v.min:v.max))))
            temp.pos.r <-
diag(kronecker(diag(length(h.min:h.max)),diag(c(v.min:v.max))))
            temp.dist <-
matrix(c(temp.pos.r,temp.pos.c),ncol=2,byrow=F,dimnames=list(NULL,c("row","col")))
            dist.pos <- sqrt(temp.dist[,1]^2+temp.dist[,2]^2)

            temp.window <- temp.dist[(dist.pos > i-.5)&(dist.pos < i+.5),]
            temp.window[,1] <- temp.window[,1]+rr
            temp.window[,2] <- temp.window[,2]+cc

            diff.use <- data.grid[temp.window]-data.grid[rr,cc]
```

```
          temp.b <- c(temp.b, gamma.bar(diff.use))
          temp.t <- c(temp.t, gamma.tilda(diff.use))
          temp.cl <- c(temp.cl, gamma.classic(diff.use))

    }
  }
}

#----------------------------------------------------------------------------#
###estimates of the variogram at each lag are split into separate arrays   ###
### (1 through 7)                                                           ###
#----------------------------------------------------------------------------#

if (estimator.type == "gamma bar") {
#Gamma Bar Estmator
  lag.b1 <- temp.b[1:sz]
  lag.b2 <- temp.b[(sz+1):(2*sz)]
  lag.b3 <- temp.b[(2*sz+1):(3*sz)]
  lag.b4 <- temp.b[(3*sz+1):(4*sz)]
  lag.b5 <- temp.b[(4*sz+1):(5*sz)]
  lag.b6 <- temp.b[(5*sz+1):(6*sz)]
  lag.b7 <- temp.b[(6*sz+1):(7*sz)]

##90th Lag Percentile
  q90.b <- numeric(sz)
  for (i in 1:sz) {
    q90.b[i] <-
  quantile(c(lag.b1[i],lag.b2[i],lag.b3[i],lag.b4[i],lag.b5[i],lag.b6[i],lag.b7[i]),.
  9,na.rm=T)
  }

  measure.b <-
  matrix(c(xval,yval,zval,lag.b1,lag.b2,lag.b3,lag.b4,q90.b),nrow=sz,ncol=8,byrow=F,d
  imnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90")))
  mstand.b <- scale(measure.b,center=T,scale=T)

  return (mstand.b)

}

if (estimator.type == "gamma tilda") {
#Gamma Tilda Estmator
  lag.t1 <- temp.t[1:sz]
  lag.t2 <- temp.t[(sz+1):(2*sz)]
  lag.t3 <- temp.t[(2*sz+1):(3*sz)]
  lag.t4 <- temp.t[(3*sz+1):(4*sz)]
  lag.t5 <- temp.t[(4*sz+1):(5*sz)]
  lag.t6 <- temp.t[(5*sz+1):(6*sz)]
  lag.t7 <- temp.t[(6*sz+1):(7*sz)]

##90th Lag Percentile
  q90.t <- numeric(sz)
  for (i in 1:sz) {
    q90.t[i] <-
  quantile(c(lag.t1[i],lag.t2[i],lag.t3[i],lag.t4[i],lag.t5[i],lag.t6[i],lag.t7[i]),.
  9,na.rm=T)
  }

  measure.t <-
  matrix(c(xval,yval,zval,lag.t1,lag.t2,lag.t3,lag.t4,q90.t),nrow=sz,ncol=8,byrow=F,d
  imnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90")))
  mstand.t <- scale(measure.t,center=T,scale=T)
```

```
    return (mstand.t)

}

if (estimator.type == "classical") {
#Classical Estmator
    lag.cl1 <- temp.cl[1:sz]
    lag.cl2 <- temp.cl[(sz+1):(2*sz)]
    lag.cl3 <- temp.cl[(2*sz+1):(3*sz)]
    lag.cl4 <- temp.cl[(3*sz+1):(4*sz)]
    lag.cl5 <- temp.cl[(4*sz+1):(5*sz)]
    lag.cl6 <- temp.cl[(5*sz+1):(6*sz)]
    lag.cl7 <- temp.cl[(6*sz+1):(7*sz)]

##90th Lag Percentile
    q90.cl <- numeric(sz)
    for (i in 1:sz) {
        q90.cl[i] <-
quantile(c(lag.cl1[i],lag.cl2[i],lag.cl3[i],lag.cl4[i],lag.cl5[i],lag.cl6[i],
lag.cl7[i]),.9,na.rm=T)
    }

    measure.cl <-
matrix(c(xval,yval,zval,lag.cl1,lag.cl2,lag.cl3,lag.cl4,q90.cl),nrow=sz,ncol
=8,byrow=F,dimnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3",
"lag4","q90")))
    mstand.cl <- scale(measure.cl,center=T,scale=T)

    return (mstand.cl)

}
}
```

B.5 Two Part Method Function

```
#------------------------------------------------------------------------------#
###Beginning of Code for Two Part Method Function                           ###
#------------------------------------------------------------------------------#

Two.Part.Method.Cluster <- function(data.grid, distance.type, adjacent.distance,
   maximum.groups, estimator.type){

data.rows <- nrow(data.grid)
data.columns <- ncol(data.grid)
sz <- data.rows*data.columns

#------------------------------------------------------------------------------#
###defining the measurement value, and the location (row/column number)     ###
##denoted by x and y                                                        ###
#------------------------------------------------------------------------------#
zval <- c(data.grid)
c.row <- diag(c(1:data.rows))
c.col <- diag(c(1:data.columns))
iwd.row <- diag(c(1),data.columns)
iwd.col <- diag(c(1),data.rows)
yval <- diag(kronecker(c.col,iwd.col))
xval <- diag(kronecker(iwd.row,c.row))

#------------------------------------------------------------------------------#
###Step 1 -- Cluster Based upon Mean Structure (ignore location, limiting ###
###distance to be "1.1")                                                    ###
#------------------------------------------------------------------------------#

measure1 <- matrix(c(xval,yval,zval),ncol=3,byrow=F)
mstd1 <- scale(measure1, center=T ,scale=T)
ms.w <- mstd1%*%diag(c(0,0,1))

if (distance.type == "complete")
   mean.adj.comp <- hclust.adj.com(ms.w,data.rows,data.columns, adjacent.distance)
if (distance.type == "average")
   mean.adj.comp <- hclust.adj.ave(ms.w,data.rows,data.columns, adjacent.distance)

#------------------------------------------------------------------------------#
###Step 2 -- Get "good" estimates of the variogram                          ###
#------------------------------------------------------------------------------#

#------------------------------------------------------------------------------#
###Initialize a loop to combine the separated groups until variogram        ###
###estimates are "close"                                                    ###
###"close" is determined by whether or not the variograms all fall within ###
###the chi-square distribution with 99.9% probability                       ###
###the variogram estimator is gamma bar                                     ###
#------------------------------------------------------------------------------#

done.test <- 0
loop.number <- 1

while (done.test == 0) {
   tree.mean.temp <-
   matrix(cutree(mean.adj.comp,k=loop.number),nrow=data.rows,ncol=data.columns,
   byrow=F)

   temp.b <- numeric(0)
   temp.t <- numeric(0)
   temp.cl <- numeric(0)
```

```
#--------------------------------------------------------------------------#
###Estimates of the variogram are calculated at lags 1 through 7          ###
###For each cell in the data grid, this function determines the other data###
###points to be incluced in the calculations of the variogram at each lag ###
### (1 through 7)                                                         ###
#--------------------------------------------------------------------------#
   for (i in 1:7) {

      for (cc in 1:data.columns) {
         h.max <- min(i,data.columns-cc)
         h.min <- max(1-cc,-i)

         for (rr in 1:data.rows) {
            v.max <- min(i,data.rows-rr)
            v.min <- max(1-rr,-i)

            temp.group <- tree.mean.temp[rr,cc]
            temp.pos.c <-
diag(kronecker(diag(c(h.min:h.max)),diag(length(v.min:v.max))))
            temp.pos.r <-
diag(kronecker(diag(length(h.min:h.max)),diag(c(v.min:v.max))))
            temp.dist <-
matrix(c(temp.pos.r,temp.pos.c),ncol=2,byrow=F,dimnames=list(NULL,c("row","col")))
            dist.pos <- sqrt(temp.dist[,1]^2+temp.dist[,2]^2)
            temp.window <- temp.dist[(dist.pos > i-.5)&(dist.pos < i+.5),]
            temp.window[,1] <- temp.window[,1]+rr
            temp.window[,2] <- temp.window[,2]+cc

            diff <- data.grid[temp.window]-data.grid[rr,cc]
            diff.use <- diff[which(tree.mean.temp[temp.window]==temp.group)]

            if (length(diff.use) > 0) {
               temp.b <- c(temp.b, gamma.bar(diff.use))
               temp.t <- c(temp.t, gamma.tilda(diff.use))
               temp.cl <- c(temp.cl, gamma.classic(diff.use))
            }
            else {
               temp.b <- c(temp.b, -1)
               temp.t <- c(temp.t, -1)
               temp.cl <- c(temp.cl, -1)
            }
         }
      }
   }

   lag.1.b <- temp.b[1:400]
   lag.1.t <- temp.t[1:400]
   lag.1.cl <- temp.cl[1:400]

   if (max(lag.1.b) < qchisq(.999,mean(lag.1.b)))
      done.test <- 1

   loop.number = loop.number + 1

}

#--------------------------------------------------------------------------#
###estimates of the variogram at each lag are split into separate arrays  ###
### (1 through 7)                                                         ###
#--------------------------------------------------------------------------#

if (estimator.type == "gamma bar") {
   ##Gamma Bar Estmator
   lag.b1 <- temp.b[1:sz]
   lag.b2 <- temp.b[(sz+1):(2*sz)]
   lag.b3 <- temp.b[(2*sz+1):(3*sz)]
   lag.b4 <- temp.b[(3*sz+1):(4*sz)]
   lag.b5 <- temp.b[(4*sz+1):(5*sz)]
   lag.b6 <- temp.b[(5*sz+1):(6*sz)]
   lag.b7 <- temp.b[(6*sz+1):(7*sz)]

##lag set to -1 if region is not large enough to calculate.
##in this case, set estimate to value of next lowest lag
   lag.b2[lag.b2==-1] <- lag.b1[lag.b2==-1]
   lag.b3[lag.b3==-1] <- lag.b2[lag.b3==-1]
   lag.b4[lag.b4==-1] <- lag.b3[lag.b4==-1]
   lag.b5[lag.b5==-1] <- lag.b4[lag.b5==-1]
   lag.b6[lag.b6==-1] <- lag.b5[lag.b6==-1]
   lag.b7[lag.b7==-1] <- lag.b6[lag.b7==-1]

##90th Lag Percentile
   q90.b <- numeric(sz)
   for (i in 1:sz) {
      q90.b[i] <-
   quantile(c(lag.b1[i],lag.b2[i],lag.b3[i],lag.b4[i],lag.b5[i],lag.b6[i],lag.b7[i])
   ,.9,na.rm=T)
   }

   measure.b <-
   matrix(c(xval,yval,zval,lag.b1,lag.b2,lag.b3,lag.b4,q90.b),nrow=sz,ncol=8,byrow=F
   ,dimnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90")))
   mstand.b <- scale(measure.b,center=T,scale=T)

   return (mstand.b)

}

if (estimator.type == "gamma tilda") {
#Gamma Tilda Estmator
   lag.t1 <- temp.t[1:sz]
   lag.t2 <- temp.t[(sz+1):(2*sz)]
   lag.t3 <- temp.t[(2*sz+1):(3*sz)]
   lag.t4 <- temp.t[(3*sz+1):(4*sz)]
   lag.t5 <- temp.t[(4*sz+1):(5*sz)]
   lag.t6 <- temp.t[(5*sz+1):(6*sz)]
   lag.t7 <- temp.t[(6*sz+1):(7*sz)]

##lag set to -1 if region is not large enough to calculate.
##in this case, set estimate to value of next lowest lag
   lag.t2[lag.t2==-1] <- lag.t1[lag.t2==-1]
   lag.t3[lag.t3==-1] <- lag.t2[lag.t3==-1]
   lag.t4[lag.t4==-1] <- lag.t3[lag.t4==-1]
   lag.t5[lag.t5==-1] <- lag.t4[lag.t5==-1]
   lag.t6[lag.t6==-1] <- lag.t5[lag.t6==-1]
   lag.t7[lag.t7==-1] <- lag.t6[lag.t7==-1]

##90th Lag Percentile
   q90.t <- numeric(sz)
   for (i in 1:sz) {
      q90.t[i] <-
   quantile(c(lag.t1[i],lag.t2[i],lag.t3[i],lag.t4[i],lag.t5[i],lag.t6[i],lag.t7[i])
   ,.9,na.rm=T)
   }
   measure.t <-
matrix(c(xval,yval,zval,lag.t1,lag.t2,lag.t3,lag.t4,q90.t),nrow=sz,ncol=8,byrow
   =F,dimnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3","lag4","q90")))
   mstand.t <- scale(measure.t,center=T,scale=T)

   return (mstand.t)

}

if (estimator.type == "classical") {
#Classical Estmator
   lag.cl1 <- temp.cl[1:sz]
   lag.cl2 <- temp.cl[(sz+1):(2*sz)]
   lag.cl3 <- temp.cl[(2*sz+1):(3*sz)]
   lag.cl4 <- temp.cl[(3*sz+1):(4*sz)]
   lag.cl5 <- temp.cl[(4*sz+1):(5*sz)]
   lag.cl6 <- temp.cl[(5*sz+1):(6*sz)]
   lag.cl7 <- temp.cl[(6*sz+1):(7*sz)]

##lag set to -1 if region is not large enough to calculate.
##in this case, set estimate to value of next lowest lag
   lag.cl2[lag.cl2==-1] <- lag.cl1[lag.cl2==-1]
   lag.cl3[lag.cl3==-1] <- lag.cl2[lag.cl3==-1]
   lag.cl4[lag.cl4==-1] <- lag.cl3[lag.cl4==-1]
   lag.cl5[lag.cl5==-1] <- lag.cl4[lag.cl5==-1]
   lag.cl6[lag.cl6==-1] <- lag.cl5[lag.cl6==-1]
   lag.cl7[lag.cl7==-1] <- lag.cl6[lag.cl7==-1]

##90th Lag Percentile
   q90.cl <- numeric(sz)
   for (i in 1:sz) {
      q90.cl[i] <-
quantile(c(lag.cl1[i],lag.cl2[i],lag.cl3[i],lag.cl4[i],lag.cl5[i],lag.cl6[i],
lag.cl7[i]),.9,na.rm=T)
   }

   measure.cl <-
matrix(c(xval,yval,zval,lag.cl1,lag.cl2,lag.cl3,lag.cl4,q90.cl),nrow=sz,ncol=8,
   byrow=F,dimnames=list(NULL,c("xval","yval","zval","lag1","lag2","lag3",
   "lag4","q90")))
   mstand.cl <- scale(measure.cl,center=T,scale=T)

   return (mstand.cl)

}
}
```

B.6 Simulated Data Examples

```
#-----------------------------------------------------------------------------#
###Chapter 3 Examples                                                       ###
#-----------------------------------------------------------------------------#
chapter.3.1.1 <- matrix(0+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
chapter.3.1.2 <- matrix(8+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
chapter.3.1.3 <- matrix(4+rfsim(grid,covfun=exp.cov,range=3,sill=3,nc=10000),wd,wd)
Chapter.3.Data.Set.1 <- chapter.3.1.1
Chapter.3.Data.Set.1[grp=="Grp 2"] <- chapter.3.1.2[grp=="Grp 2"]
Chapter.3.Data.Set.1[grp=="Grp 3"] <- chapter.3.1.3[grp=="Grp 3"]

chapter.3.2.1 <- matrix(4+rfsim(grid,covfun=exp.cov,range=5,sill=3,nc=10000),wd,wd)
chapter.3.2.2 <- matrix(4+rfsim(grid,covfun=exp.cov,range=5,sill=1,nc=10000),wd,wd)
chapter.3.2.3 <- matrix(4+rfsim(grid,covfun=exp.cov,range=5,sill=5,nc=10000),wd,wd)
Chapter.3.Data.Set.2 <- chapter.3.2.1
Chapter.3.Data.Set.2[grp=="Grp 2"] <- chapter.3.2.2[grp=="Grp 2"]
Chapter.3.Data.Set.2[grp=="Grp 3"] <- chapter.3.2.3[grp=="Grp 3"]

par(mfrow=c(1,1))
image(Chapter.3.Data.Set.1)
persp(Chapter.3.Data.Set.1)
image(Chapter.3.Data.Set.2)
persp(Chapter.3.Data.Set.2)

example.3.1.cluster <- hclust(dist(c(Chapter.3.Data.Set.1)), method = 'compact')

par(mfrow=c(3,3))
image(Chapter.3.Data.Set.1)
title(main=paste("Original Data Set"))
for (i in 2:9) {
   tree.adj.constraint <-
   matrix(cutree(example.3.1.cluster,k=i),nrow=data.rows,ncol=data.columns,byrow=F)
   index <- get.index(grp.mat,tree.adj.constraint)
   image(tree.adj.constraint)
   title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
   round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
}

example.3.2.cluster <- hclust(dist(c(Chapter.3.Data.Set.2)), method = 'compact')

par(mfrow=c(3,3))
image(Chapter.3.Data.Set.2)
title(main=paste("Original Data Set"))
for (i in 2:9) {
   tree.adj.constraint <-
   matrix(cutree(example.3.2.cluster,k=i),nrow=data.rows,ncol=data.columns,byrow=F)
   index <- get.index(grp.mat,tree.adj.constraint)
   image(tree.adj.constraint)
   title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
   round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
}


###Chapter 3 Basic Window Method Example
Chapter.3.Data.Set.1.Central <- Central.Method.Cluster(Chapter.3.Data.Set.1,
   "complete", "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- Chapter.3.Data.Set.1.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.3.1 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(testdata.1.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:18) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.3.1,k=i),nrow=data.rows,ncol=data.columns,
   byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

Chapter.3.Data.Set.2.Central <- Central.Method.Cluster(Chapter.3.Data.Set.2,
"complete", "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- Chapter.3.Data.Set.2.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.3.2 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(Chapter.3.Data.Set.2)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:18) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.3.2,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

###Chapter 3 Adjacent Contstraint Method Example

Chapter.3.Data.Set.1.Adjacent <- Adjacent.Constraint.Cluster (Chapter.3.Data.
Set.1, "complete", 1.5, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- Chapter.3.Data.Set.1.Adjacent%*%weight.adjcon

   constraint.cluster.3.1 <- hclust.adj.com(msw,data.rows,data.columns,1.5)

   par(mfrow=c(3,3))
   image(Chapter.3.Data.Set.1)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.3.1,k=i),nrow=data.rows,ncol=data.columns,byrow=
   F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
   round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

Chapter.3.Data.Set.2.Adjacent <- Adjacent.Constraint.Cluster (Chapter.3.Data.Set.2,
   "complete", 1.5, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- Chapter.3.Data.Set.2.Adjacent%*%weight.adjcon

   constraint.cluster.3.2 <- hclust.adj.com(msw, data.rows, data.columns, 1.5)

   par(mfrow=c(3,3))
   image(Chapter.3.Data.Set.2)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.3.2,k=i),nrow=data.rows,ncol=data.columns,byrow=
   F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
   round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

###Chapter 3 Two Part Method Example

Chapter.3.Data.Set.1.TwoPart <- Two.Part.Method.Cluster (Chapter.3.Data.Set.1,
   "complete", 1.1, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- Chapter.3.Data.Set.1.TwoPart%*%weight.adjcon

   constraint.cluster.3.1 <- hclust.adj.com(msw, data.rows, data.columns, 1.1)

   par(mfrow=c(3,3))
   image(Chapter.3.Data.Set.1)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.3.1,k=i),nrow=data.rows,ncol=data.columns,byrow=
   F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
   round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

Chapter.3.Data.Set.2.TwoPart <- Two.Part.Method.Cluster (Chapter.3.Data.Set.2,
"complete", 1.1, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- Chapter.3.Data.Set.2.TwoPart%*%weight.adjcon

   constraint.cluster.3.2 <- hclust.adj.com(msw, data.rows, data.columns, 1.1)

   par(mfrow=c(3,3))
   image(Chapter.3.Data.Set.2)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.3.2,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

#--------------------------------------------------------------------------#
###Analysis of Simulated Data Sets with above functions                  ###
#--------------------------------------------------------------------------#
grp.mat <- grp
grp.mat[grp.mat == "Grp 1"] <- 1
grp.mat[grp.mat == "Grp 2"] <- 2
grp.mat[grp.mat == "Grp 3"] <- 3
grp.mat <- matrix(as.numeric(grp.mat),20,20)

#--------------------------------------------------------------------------#
###Central Method - Data Set 1                                            ###
#--------------------------------------------------------------------------#

datacluster.1.Central <- Central.Method.Cluster(testdata.1.grid, "complete",
 "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.1.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.1 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(testdata.1.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.1,k=i),nrow=data.rows,ncol=data.columns,byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
      round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}
#---------------------------------------------------------------------------#
###Central Method - Data Set 2                                            ###
#---------------------------------------------------------------------------#

datacluster.2.Central <- Central.Method.Cluster(testdata.2.grid, "complete", "gamma
      bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.2.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.2 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(testdata.2.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
      matrix(cutree(constraint.cluster.2,k=i),nrow=data.rows,ncol=data.columns,byrow
      =F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
      round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

#---------------------------------------------------------------------------#
###Central Method - Data Set 3                                            ###
#---------------------------------------------------------------------------#

datacluster.3.Central <- Central.Method.Cluster(testdata.3.grid, "complete", "gamma
      bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.3.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.3 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(testdata.3.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
```

```
            tree.adj.constraint <-
            matrix(cutree(constraint.cluster.3,k=i),nrow=data.rows,ncol=data.columns
            ,byrow=F)
            index <- get.index(grp.mat,tree.adj.constraint)
            image(tree.adj.constraint)
               title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
            round(index[1],digits=3), " Rand = ",round(index[2],digits=3),
            collapse = ""))
            }

}
```

```
#----------------------------------------------------------------------------#
###Central Method - Data Set 4                                            ###
#----------------------------------------------------------------------------#

datacluster.4.Central <- Central.Method.Cluster(testdata.4.grid, "complete", "gamma
bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.4.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.4 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(testdata.4.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.4,k=i),nrow=data.rows,ncol=data.columns,byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

#----------------------------------------------------------------------------#
###Adjacent Contstraint Method -- Data Set 1                              ###
#----------------------------------------------------------------------------#

datacluster.1.Adjacent <- Adjacent.Constraint.Cluster (testdata.1.grid, "complete",
1.5, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.1.Adjacent%*%weight.adjcon

   constraint.cluster.1 <- hclust.adj.com(msw, data.rows, data.columns, 1.5)

   par(mfrow=c(3,3))
   image(testdata.1.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.1,k=i),nrow=data.rows,ncol=data.columns,byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}
```

```
#--------------------------------------------------------------------------#
###Adjacent Contstraint Method -- Data Set 2                            ###
#--------------------------------------------------------------------------#

datacluster.2.Adjacent <- Adjacent.Constraint.Cluster (testdata.2.grid,
 "complete", 1.5, 20, "gamma bar")

for (inc in 0:4) {
  w.mean <- .25*inc
  w.spatial <- 1-w.mean
  weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
  msw <- datacluster.2.Adjacent%*%weight.adjcon

  constraint.cluster.2 <- hclust.adj.com(msw, data.rows, data.columns, 1.5)

  par(mfrow=c(3,3))
  image(testdata.2.grid)
  title(main=paste("Original Data Set\n a = ", inc*.25))
  for (i in 2:27) {
      tree.adj.constraint <-
  matrix(cutree(constraint.cluster.2,k=i),nrow=data.rows,ncol=data.columns
,byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
  round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
  }

}

#--------------------------------------------------------------------------#
###Adjacent Contstraint Method -- Data Set 3                            ###
#--------------------------------------------------------------------------#

datacluster.3.Adjacent <- Adjacent.Constraint.Cluster (testdata.3.grid,
 "complete", 1.5, 20, "gamma bar")

for (inc in 0:4) {
  w.mean <- .25*inc
  w.spatial <- 1-w.mean
  weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
  msw <- datacluster.3.Adjacent%*%weight.adjcon

  constraint.cluster.3 <- hclust.adj.com(msw, data.rows, data.columns, 1.5)

  par(mfrow=c(3,3))
  image(testdata.3.grid)
  title(main=paste("Original Data Set\n a = ", inc*.25))
  for (i in 2:27) {
      tree.adj.constraint <-
  matrix(cutree(constraint.cluster.3,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
  round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
  }

 }
```

```
#-----------------------------------------------------------------------#
###Adjacent Contstraint Method -- Data Set 4                          ###
#-----------------------------------------------------------------------#

datacluster.4.Adjacent <- Adjacent.Constraint.Cluster (testdata.4.grid,
"complete", 1.5, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.4.Adjacent%*%weight.adjcon

   constraint.cluster.4 <- hclust.adj.com(msw, data.rows, data.columns, 1.5)

   par(mfrow=c(3,3))
   image(testdata.4.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.4,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}

#-----------------------------------------------------------------------#
###Two Part Method                                                     ###
#-----------------------------------------------------------------------#
###The data points are run through the clustering fuction subject to the  ###
###weights. Weights are assigned to give initial equal weighting to      ###
###non-spatial data points (i.e. the measured data) and the spatial data  ###
###points (i.e. the variogram estimators) The clustering is done 5 times  ###
###with added loads to non-spatial and spatial data points             ###
###The clustering is then graphically displayed                        ###
#-----------------------------------------------------------------------#


#-----------------------------------------------------------------------#
###Two Part Method -- Data Set 1                                       ###
#-----------------------------------------------------------------------#
datacluster.1.TwoPart <- Two.Part.Method.Cluster (testdata.1.grid, "complete"
, 1.1, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.1.TwoPart%*%weight.adjcon

   constraint.cluster.1 <- hclust.adj.com(msw, data.rows, data.columns, 1.1)

   par(mfrow=c(3,3))
   image(testdata.1.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.1,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
```

```
       index <- get.index(grp.mat,tree.adj.constraint)
       image(tree.adj.constraint)
    title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
  round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
  }

}

#-----------------------------------------------------------------------------#
###Two Part Method -- Data Set 2                                            ###
#-----------------------------------------------------------------------------#
datacluster.2.TwoPart <- Two.Part.Method.Cluster (testdata.2.grid, "complete",
 1.1, 20, "gamma bar")

for (inc in 0:4) {
  w.mean <- .25*inc
  w.spatial <- 1-w.mean
  weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
  msw <- datacluster.2.TwoPart%*%weight.adjcon

  constraint.cluster.2 <- hclust.adj.com(msw, data.rows, data.columns, 1.1)

  par(mfrow=c(3,3))
  image(testdata.2.grid)
  title(main=paste("Original Data Set\n a = ", inc*.25))
  for (i in 2:27) {
    tree.adj.constraint <-
  matrix(cutree(constraint.cluster.2,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
    index <- get.index(grp.mat,tree.adj.constraint)
    image(tree.adj.constraint)
    title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
  round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
  }

}
#-----------------------------------------------------------------------------#
###Two Part Method -- Data Set 3                                            ###
#-----------------------------------------------------------------------------#
datacluster.3.TwoPart <- Two.Part.Method.Cluster (testdata.3.grid, "complete",
 1.1, 20, "gamma bar")

for (inc in 0:4) {
  w.mean <- .25*inc
  w.spatial <- 1-w.mean
  weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
  msw <- datacluster.3.TwoPart%*%weight.adjcon

  constraint.cluster.3 <- hclust.adj.com(msw, data.rows, data.columns, 1.1)

  par(mfrow=c(3,3))
  image(testdata.3.grid)
  title(main=paste("Original Data Set\n a = ", inc*.25))
  for (i in 2:27) {
    tree.adj.constraint <-
  matrix(cutree(constraint.cluster.3,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
    index <- get.index(grp.mat,tree.adj.constraint)
    image(tree.adj.constraint)
    title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
  round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
  }
}
```

```
#----------------------------------------------------------------------------#
###Two Part Method -- Data Set 4                                           ###
#----------------------------------------------------------------------------#
datacluster.4.TwoPart <- Two.Part.Method.Cluster (testdata.4.grid, "complete",
 1.1, 20, "gamma bar")

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- datacluster.4.TwoPart%*%weight.adjcon

   constraint.cluster.4 <- hclust.adj.com(msw, data.rows, data.columns, 1.1)

   par(mfrow=c(3,3))
   image(testdata.4.grid)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
matrix(cutree(constraint.cluster.4,k=i),nrow=data.rows,ncol=data.columns,
byrow=F)
      index <- get.index(grp.mat,tree.adj.constraint)
      image(tree.adj.constraint)
      title(main=paste("For ",i," Groups, the indexes are: \n Jaccard = ",
round(index[1],digits=3), " Rand = ",round(index[2],digits=3),collapse = ""))
   }

}
```

B.7 SFMRI Brain Data Analysis

```
#-----------------------------------------------------------------------------#
###Brain Data                                                              ###
#-----------------------------------------------------------------------------#

###Brain Data Matrix is 64 x 64

a6s10.sort <- a6s10
a6s10.sort[,2] <- 65 - a6s10.sort[,2]
a6s10.sort <- a6s10.sort[order(a6s10.sort[,2],a6s10.sort[,1]), ]

a6s10.mean.0 <- matrix(apply(a6s10.sort[,4:39], 1, mean), nrow = 64, ncol = 64,
   byrow = F)

###equivalent method to plot the data
a6s10.mean.1 <- interp(a6s10[, 1], 65-a6s10[, 2], apply(a6s10[,4:39], 1, mean),
 xo = c(1:64), yo = c(1:64))

on.array.sort  <- c(a6s10.sort[, 4], a6s10.sort[, 6], a6s10.sort[, 8],
   a6s10.sort[,10],
   a6s10.sort[,12], a6s10.sort[,14], a6s10.sort[,16], a6s10.sort[,18],
   a6s10.sort[,20],
   a6s10.sort[,22], a6s10.sort[,24], a6s10.sort[,26], a6s10.sort[,28],
   a6s10.sort[,30],
   a6s10.sort[,32], a6s10.sort[,34], a6s10.sort[,36], a6s10.sort[,38])

a6s10.on <- matrix(on.array.sort, nrow = 4096, ncol = 18, byrow = F)
a6s10.on.mean <- apply(a6s10.on, 1, mean)

off.array.sort <- c(a6s10.sort[, 5], a6s10.sort[, 7], a6s10.sort[, 9],
   a6s10.sort[,11],
   a6s10.sort[,13], a6s10.sort[,15], a6s10.sort[,17], a6s10.sort[,19],
   a6s10.sort[,21],
   a6s10.sort[,23], a6s10.sort[,25], a6s10.sort[,27], a6s10.sort[,29],
   a6s10.sort[,31],
   a6s10.sort[,33], a6s10.sort[,35], a6s10.sort[,37], a6s10.sort[,39])

a6s10.off <- matrix(off.array.sort, nrow = 4096, ncol = 18, byrow = F)
a6s10.off.mean <- apply(a6s10.off, 1, mean)

a6s10.mean.diff.0 <- matrix(a6s10.on.mean - a6s10.off.mean, nrow = 64, ncol
 = 64, byrow = F)

###Captures all meaningful brain data
a6s10.mean <- a6s10.mean.0[13:52,6:48]
a6s10.mean.diff <- a6s10.mean.diff.0[13:52,6:48]

###Captures portion for computing (physical computing restraints)
a6s10.mean <- a6s10.mean.0[14:34,15:48]
a6s10.mean.diff <- a6s10.mean.diff.0[14:34,15:48]

par(mfrow=c(2,2))

image(a6s10.mean.0)
persp(a6s10.mean.0)

image(a6s10.mean)
persp(a6s10.mean)

image(a6s10.mean.diff)
persp(a6s10.mean.diff)
```

```
###increase the following to help with the dynamic memory problems
options(memory = 14336*1024^2)

#-----------------------------------------------------------------------------#
###Central Method - Brain Data                                              ###
#-----------------------------------------------------------------------------#

braincluster.1.Central <- Central.Method.Cluster(a6s10.mean, "complete", "gamma
   bar")

braindata.rows <- nrow(a6s10.mean)
braindata.columns <- ncol(a6s10.mean)

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- braincluster.1.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.1 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(a6s10.mean)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.1,k=i),nrow=braindata.rows,ncol=
   braindata.columns,byrow=F)
      image(tree.adj.constraint)
      title(main=paste("Data Clustered into ",i," Groups."))
   }
}

braincluster.2.Central <- Central.Method.Cluster(a6s10.mean.diff, "complete",
 "gamma bar")

braindata.rows <- nrow(a6s10.mean.diff)
braindata.columns <- ncol(a6s10.mean.diff)

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(1,1,3),w.spatial*c(2,1,1,0,1)))
   msw <- braincluster.2.Central%*%weight.adjcon

   msw <- dist(data.matrix(msw))

   constraint.cluster.2 <- hclust(msw, method = 'compact')

   par(mfrow=c(3,3))
   image(a6s10.mean.diff)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.2,k=i),nrow=braindata.rows,ncol=
braindata.columns,byrow=F)
      image(tree.adj.constraint)
      title(main=paste("Data Clustered into ",i," Groups."))
   }
}

#--------------------------------------------------------------------------#
###Adjacent Contstraint Method   -- Brain Data                           ###
#--------------------------------------------------------------------------#
braincluster.1.Adjacent <- Adjacent.Constraint.Cluster (a6s10.mean, "complete",
 1.5, "gamma bar")

braindata.rows <- nrow(a6s10.mean)
braindata.columns <- ncol(a6s10.mean)

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- braincluster.1.Adjacent%*%weight.adjcon

   constraint.cluster.1 <- hclust.adj.com(msw, braindata.rows, braindata.columns,
   1.5)

   par(mfrow=c(3,3))
   image(a6s10.mean)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.1,k=i),nrow=braindata.rows,ncol=
   braindata.columns,byrow=F)
      image(tree.adj.constraint)
      title(main=paste("Data Clustered into ",i," Groups."))
   }
}


braincluster.2.Adjacent <- Adjacent.Constraint.Cluster (a6s10.mean.diff,
 "complete", 1.5, "gamma bar")

braindata.rows <- nrow(a6s10.mean.diff)
braindata.columns <- ncol(a6s10.mean.diff)

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- braincluster.2.Adjacent%*%weight.adjcon

   constraint.cluster.2 <- hclust.adj.com(msw, braindata.rows,
    braindata.columns, 1.5)

   par(mfrow=c(3,3))
   image(a6s10.mean.diff)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.2,k=i),nrow=braindata.rows,ncol=
   braindata.columns,byrow=F)
      image(tree.adj.constraint)
      title(main=paste("Data Clustered into ",i," Groups."))
   }
}

#--------------------------------------------------------------------------#
###Two Part Method -- Brain Data                                         ###
#--------------------------------------------------------------------------#
```

```
braincluster.1.TwoPart <- Two.Part.Method.Cluster (a6s10.mean, "complete", 1.1,
   "gamma bar")

braindata.rows <- nrow(a6s10.mean)
braindata.columns <- ncol(a6s10.mean)

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- braincluster.1.TwoPart%*%weight.adjcon

   constraint.cluster.1 <- hclust.adj.com(msw, braindata.rows,
braindata.columns, 1.1)

   par(mfrow=c(3,3))
   image(a6s10.mean)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.1,k=i),nrow=braindata.rows,ncol=
   braindata.columns,byrow=F)
      image(tree.adj.constraint)
      title(main=paste("Data Clustered into ",i," Groups."))
   }
}

braincluster.2.TwoPart <- Two.Part.Method.Cluster (a6s10.mean.diff,
"complete", 1.1, "gamma bar")

braindata.rows <- nrow(a6s10.mean.diff)
braindata.columns <- ncol(a6s10.mean.diff)

for (inc in 0:4) {
   w.mean <- .25*inc
   w.spatial <- 1-w.mean
   weight.adjcon <- diag(c(w.mean*c(0,0,5),w.spatial*c(2,1,1,0,1)))
   msw <- braincluster.2.TwoPart%*%weight.adjcon

   constraint.cluster.2 <- hclust.adj.com(msw, braindata.rows,
   braindata.columns, 1.1)

   par(mfrow=c(3,3))
   image(a6s10.mean.diff)
   title(main=paste("Original Data Set\n a = ", inc*.25))
   for (i in 2:27) {
      tree.adj.constraint <-
   matrix(cutree(constraint.cluster.2,k=i),nrow=braindata.rows,ncol=
   braindata.columns,byrow=F)
      image(tree.adj.constraint)
      title(main=paste("Data Clustered into ",i," Groups."))
   }
}
```