

Dissimilarity Plots: A Visual Exploration Tool for Partitional Clustering



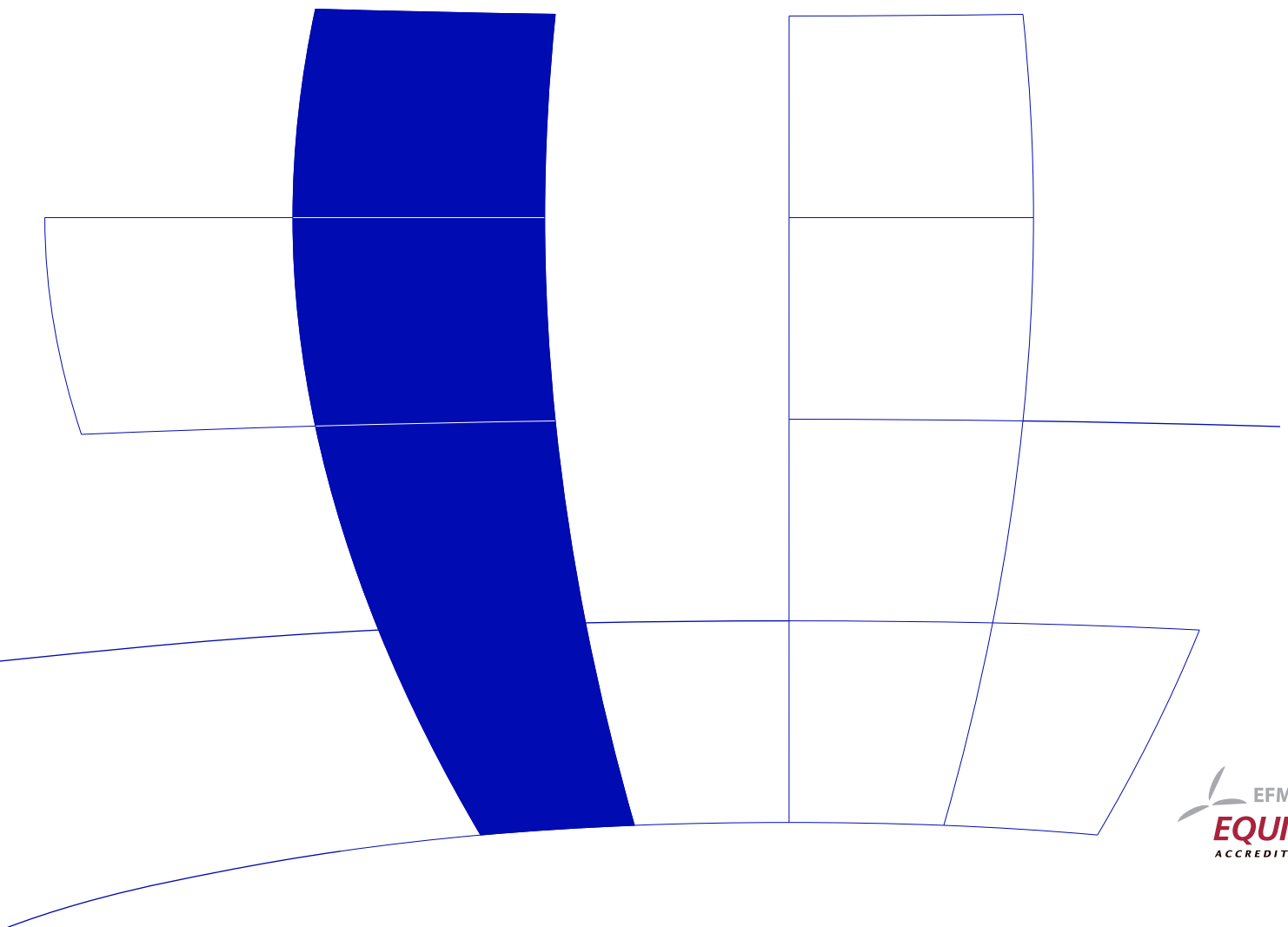
Michael Hahsler, Kurt Hornik

Department of Statistics and Mathematics
WU Wirtschaftsuniversität Wien

Research Report Series

Report 89
September 2009

<http://statmath.wu.ac.at/>



Dissimilarity Plots: A Visual Exploration Tool for Partitional Clustering

Michael Hahsler

Department of Computer Science and Engineering,
Lyle School of Engineering,
Southern Methodist University

Kurt Hornik

Department of Statistics and Mathematics,
Wirtschaftsuniversität Wien

Abstract

For hierarchical clustering, dendrograms provide convenient and powerful visualization. Although many visualization methods have been suggested for partitional clustering, their usefulness deteriorates quickly with increasing dimensionality of the data and/or they fail to represent structure between and within clusters simultaneously. In this paper we extend (dissimilarity) matrix shading with several reordering steps based on seriation. Both methods, matrix shading and seriation, have been well-known for a long time. However, only recent algorithmic improvements allow to use seriation for larger problems. Furthermore, seriation is used in a novel stepwise process (within each cluster and between clusters) which leads to a visualization technique that is independent of the dimensionality of the data. A big advantage is that it presents the structure between clusters and the micro-structure within clusters in one concise plot. This not only allows for judging cluster quality but also makes mis-specification of the number of clusters apparent. We give a detailed discussion of the construction of dissimilarity plots and demonstrate their usefulness with several examples.

1 Introduction

Assessing the quality of an obtained cluster solution has been a research topic since the invention of cluster analysis. This is especially important since all popular clustering algorithms produce a clustering even for data without a “cluster” structure. The quality of clustering or individual clusters is typically judged by intra and inter-cluster similarities. High intra-cluster similarity (between objects in the same cluster) and at the same time low inter-cluster similarity (between different clusters) indicate a good clustering. To present these similarities visualizations are helpful for judging the quality of a clustering and to explore the cluster structure.

For hierarchical clustering dendrograms (Hartigan, 1967) are available which show the hierarchical structure of the clustering as a binary tree. Similarities between clusters and between objects are represented in the plot by the height of internal nodes of the tree. Cluster quality can be judged by looking at the distance between the internal nodes that separate clusters and the nodes that separate the objects in each cluster. Unfortunately such a convenient visualization is only possible for hierarchical/nested clusterings.

For an arbitrary partition of data into k clusters, the original objects can be projected into 2-dimensional space using dimensionality reduction methods (e.g., principal component analysis or multi-dimensional scaling). Objects belonging to the same cluster can be marked and separation between clusters can be judged visually (Pison, Struyf, and Rousseeuw, 1999). This type of visualization works very well if the dimensionality reduction is able to preserve a large portion of the variability in the original data which is typically difficult for higher dimensional data.

Another approach is to visualize metrics calculated from inter and intra-cluster similarities to judge cluster quality. For example, silhouette width (Rousseeuw, 1987; Kaufman and Rousseeuw, 1990) is a measure for how much an object belongs to its cluster (intra cluster similarity) compared to how close it is to objects in its nearest neighboring clusters. Silhouette widths are typically visualized using a barplot where the objects are ordered by cluster and decreasing silhouette width. However, this type of visualization only provides a diagnostic tool for cluster quality and does not allow to analyze the structure of the data. These and several other visualization methods (e.g., based on self-organizing maps and neighborhood graphs) are reviewed in Leisch (2008).

The visualization technique presented in this paper is based on a different technique called matrix shading (see, e.g., Sneath and Sokal, 1973; Ling, 1973; Gale, Halperin, and Costanzo, 1984). For matrix shading, each value in the matrix is represented by a square with the intensity of the color depending on the value. The presentation is improved by reordering the rows and columns. Reordering matrices is a long known technique. For example Jacques Bertin devotes a whole chapter of his book “Graphics and Graphic Information Processing” (Bertin, 1981, which was first published in French in 1967) to this topic. More recently matrix reordering was applied to mosaic displays for multi-way contingency tables (Friendly, 1994), distance matrices (Wishart, 1999), correlation matrices (Friendly, 2002), and scatter plot matrices (Hurley, 2004). For these applications reordering is typically done using heuristics. For example matrix shading is often used in connection with hierarchical clustering, where the order of the dendrogram leaf nodes is used to arrange the matrix yielding a cluster heat map (Eisen, Spellman, Brownadagger, and Botstein, 1998; Wilkinson and Friendly, 2009). Since the order of leaf nodes in a dendrogram is not unique (each subtree can be rotated) and to further improve the presentation, the leaf nodes can be reordered using heuristics (e.g., Gruvaeus and Wainer, 1972). Only more recently Bar-Joseph, Demaine, Gifford, and Jaakkola (2001) developed an $O(n^4)$ algorithm that finds the optimal order of leaf nodes which minimizes the sum of distances between the nodes in the order.

The idea to apply matrix shading not only with hierarchical clustering but also in the context of partitional clustering is obvious and is used in a method called *CLUSION* suggested by Strehl and Ghosh (2003) for a graph-based partitional clustering. In this method the dissimilarity matrix is arranged such that all objects pertaining to a single cluster appear in consecutive order in the matrix. The authors call this *coarse seriation*. The result of a “good” clustering should be a matrix with low dissimilarity values forming blocks around the main diagonal corresponding to the clusters. However, using coarse seriation, the order of the clusters is only an artifact of the cluster algorithm and the objects within each cluster are unordered potentially obscuring structure in the data.

The dissimilarity plot method presented in this paper applies state-of-the-art seriation methods to find, given a partitional clustering, the optimal or near optimal positions of clusters and objects within clusters in the shaded matrix. It aims at visualizing global structure (similarity between different clusters is reflected by their position relative to each other) as well as the micro structure within each cluster (position of objects) to be able to judge cluster quality and give an indication if the number of clusters was mis-specified. Seriation is a combinatorial problem and thus in general very difficult to solve for all but extremely small problems. Recently a very efficient algorithm for the seriation problem based on branch-and-bound (Brusco and Stahl, 2005) and a heuristic that combines dynamic programming combined with simulated annealing (Brusco, Köhn, and Stahl, 2008) have been developed. This algorithmic progress allows us to use seriation for visualization of clusterings of larger data sets.

The rest of the paper is organized as follows. In Section 2 we introduce seriation as the basic method to optimally position clusters and objects in the plot. The method used to produce dissimilarity plots is described in Section 3. Section 4 presents several examples to show how dissimilarity plots compare to other methods. We conclude the paper with Section 5.

2 Seriation

Seriation is a basic problem in combinatorial data analysis (Arabie and Hubert, 1996) with the aim to arrange all objects in a set in a linear order given available data and some loss function, in order to reveal structural information. Solving problems in combinatorial data

analysis requires the solution of discrete optimization problems which, in the most general case, involves evaluating all feasible solutions. Due to the combinatorial nature, the number of possible solutions grows with problem size (number of objects, n) by the order $O(n!)$. This makes a brute-force enumerative approach infeasible for all but very small problems. To solve larger problems (currently with up to 40 objects), partial enumeration methods can be used. For example, Hubert, Arabie, and Meulman (1987) propose dynamic programming and Brusco and Stahl (2005) use a branch-and-bound strategy. For even larger problems heuristics can be employed. Recently a heuristic which combines dynamic programming with simulated annealing was developed by Brusco et al. (2008). This heuristic produces very good average results and allows to find close to optimal solutions for much larger problems.

To seriate a set of n objects $\mathcal{O} = \{O_1, \dots, O_n\}$ one typically starts with an $n \times n$ symmetric dissimilarity matrix $\mathbf{D} = (d_{ij})$ where d_{ij} for $1 \leq i, j \leq n$ represents the dissimilarity between objects O_i and O_j , and $d_{ii} = 0$ for all i . We define a permutation function Ψ as a function which reorders the objects in \mathbf{D} by simultaneously permuting rows and columns, i.e., $\Psi(\mathbf{D}) = \mathbf{A}\mathbf{D}\mathbf{A}$, where \mathbf{A} is a permutation matrix with exactly one 1 per row and column and otherwise only 0s. The permutation matrix \mathbf{A} can be obtained by permuting the rows of an $n \times n$ identity matrix according to the order the objects in \mathcal{O} should have in the permutation.

The seriation problem is to find a permutation function Ψ^* which optimizes the value of a given loss function L . This results in the optimization problem

$$\Psi^* = \underset{\Psi}{\operatorname{argmin}} L(\Psi(\mathbf{D})). \quad (1)$$

Next, we introduce a small selection of loss functions which can be used for seriation of objects and clusters in dissimilarity plot. A more comprehensive list of loss function for seriation can be found in Hahsler, Hornik, and Buchta (2008).

2.1 Column/row gradient measures

A symmetric dissimilarity matrix where the values in all rows and columns only increase when moving away from the main diagonal is called a perfect *anti-Robinson matrix* after the statistician Robinson (1951). Formally, an $n \times n$ dissimilarity matrix \mathbf{D} is in anti-Robinson form if and only if the following two gradient conditions hold (Hubert et al., 1987):

$$\text{within rows: } d_{ik} \leq d_{ij} \quad \text{for } 1 \leq i < k < j \leq n; \quad (2)$$

$$\text{within columns: } d_{kj} \leq d_{ij} \quad \text{for } 1 \leq i < k < j \leq n. \quad (3)$$

In an anti-Robinson matrix the smallest dissimilarity values appear close to the main diagonal, therefore, the closer objects are together in the order of the matrix, the higher their similarity. This provides a natural objective for seriation.

It has to be noted that \mathbf{D} can be brought into a perfect anti-Robinson form by row and column permutation whenever \mathbf{D} is an ultrametric or \mathbf{D} has an exact Euclidean representation in a single dimension (Hubert et al., 1987). However, for most data only an approximation to the anti-Robinson form is possible.

A suitable loss measure which quantifies the divergence of a matrix from the anti-Robinson form was given by Hubert et al. (1987) as

$$L(\mathbf{D}) = \sum_{i < k < j} f(d_{ik}, d_{ij}) + \sum_{i < k < j} f(d_{kj}, d_{ij}) \quad (4)$$

where $f(\cdot, \cdot)$ is a function which defines how a violation or satisfaction of a gradient condition for an object triple $(O_i, O_k$ and $O_j)$ is counted. Hubert et al. (1987) suggest two functions. The first function is given by:

$$f(z, y) = \operatorname{sign}(y - z) = \begin{cases} -1 & \text{if } z > y; \\ 0 & \text{if } z = y; \\ +1 & \text{if } z < y. \end{cases} \quad (5)$$

It results in the raw number of triples violating the gradient constraints minus triples which satisfy the constraints.

The second function is defined as:

$$f(z, y) = |y - z| \text{sign}(y - z) = y - z \quad (6)$$

It weighs each satisfaction or violation by its magnitude given by the absolute difference between the values.

2.2 Anti-Robinson events

An even simpler loss function can be created in the same way as the gradient measures above by concentrating on violations only.

$$L(\mathbf{D}) = \sum_{i < k < j} f(d_{ik}, d_{ij}) + \sum_{i < k < j} f(d_{kj}, d_{ij}) \quad (7)$$

To only count the violations we use

$$f(z, y) = I(z, y) = \begin{cases} 1 & \text{if } z < y \text{ and} \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

$I(\cdot)$ is an indicator function returning 1 only for violations. Chen (2002) presented a formulation for an equivalent loss function and called the violations *anti-Robinson events*. Chen (2002) also introduced a weighted versions of the loss function resulting in

$$f(z, y) = |y - z| I(z, y) \quad (9)$$

using the absolute deviations as weights.

2.3 Hamiltonian path length

The dissimilarity matrix \mathbf{D} can be represented as a finite weighted graph $G = (\Omega, E)$ where the set of objects Ω constitute the vertices and each edge $e_{ij} \in E$ between the objects $O_i, O_j \in \Omega$ has a weight w_{ij} associated which represents the dissimilarity d_{ij} .

Such a graph can be used for seriation (see, e.g., Hubert, 1974; Caraux and Pinloche, 2005). An order Ψ of the objects can be seen as a path through the graph where each node is visited exactly once, i.e., a Hamiltonian path. Minimizing the Hamiltonian path length results in a seriation optimal with respect to dissimilarities between neighboring objects. The loss function based on the Hamiltonian path length is:

$$L(\mathbf{D}) = \sum_{i=1}^{n-1} d_{i, i+1}. \quad (10)$$

The length of the Hamiltonian path is equal to the value of the *minimal span loss function* (as used by Chen, 2002), and both notions are closely related to the *traveling salesperson problem (TSP)* (Gutin and Punnen, 2002). For the TSP exist specialized solvers (e.g., Concorde by Applegate, Bixby, Chvátal, and Cook (2006)) and good heuristics (e.g., Lin and Kernighan, 1973) which are more efficient than general seriation algorithms.

3 Dissimilarity plots

The aim of visualizing a clustering solution is to help to make the structure or lack thereof implied by the cluster solution apparent. To achieve this goal we use matrix shading to display the dissimilarity matrix with the following improvements:

1. We arrange the clusters in such a way that clusters which contain objects that are more similar are displayed closer together. This helps to understand the relationship between clusters. For example, if the clustering algorithm breaks apart a natural group in the data, the two formed clusters should be displayed next to each other.

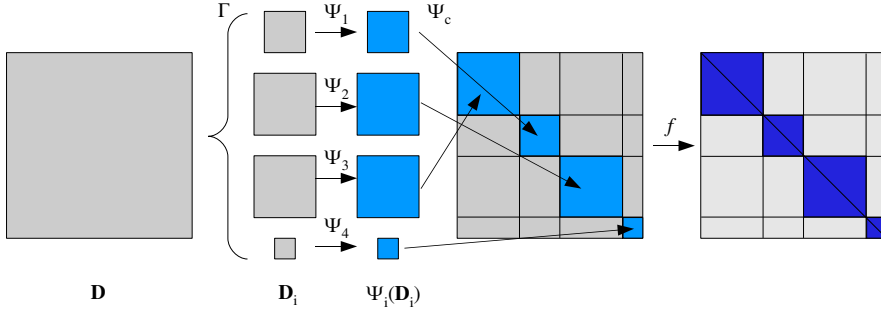


Figure 1: Example of the application of seriation for dissimilarity plot with four clusters.

2. We arrange the objects in each cluster such that more similar objects are displayed closer together. This way the micro-structure inside each cluster can be analyzed. For example a cluster can contain two or more quite distinct groups which might indicate that the number of clusters used for clustering was too small.
3. We use a color scheme that is equivalent to a monotone, possibly non-linear transformation of the (dis)similarity data to highlight different aspects of the clustering (e.g., cluster compactness or inter cluster similarities).

Figure 1 shows the steps needed for visualizing data using dissimilarity plot. As input we have a dissimilarity matrix \mathbf{D} and the assignment function $\Gamma : \mathcal{O} \rightarrow \{1, \dots, k\}$ provided by a partitioning clustering algorithm, which assigns a cluster number to each object. This function is used to split the set of objects \mathcal{O} into k (number of clusters) partitions:

$$\mathcal{O}_i = \{o \in \mathcal{O} \mid \Gamma(o) = i\} \quad \text{for } i = 1 \dots k. \quad (11)$$

We create for each set \mathcal{O}_i representing cluster i a sub-matrix \mathbf{D}_i containing only the dissimilarities between the objects in \mathcal{O}_i .

To reveal structural information within each cluster, we use seriation on the set of objects for each cluster using the corresponding dissimilarity sub-matrix \mathbf{D}_i resulting in k permutation functions Ψ_i which are used to permute the column and rows of \mathbf{D}_i representing objects in each cluster. Note that we never have to apply the expensive seriation operation to the whole dissimilarity matrix. We always only seriate sub-matrices which speeds up computation significantly.

Next, we determine the order of clusters for display. To position the clusters in the dissimilarity plot seriation is applied to an inter-cluster dissimilarity matrix \mathbf{D}_c to find a cluster permutation function Ψ_c which determines the order of clusters in the plot. To construct \mathbf{D}_c we have to compute aggregate dissimilarities between all pairs of clusters given dissimilarities between all elements of the clusters in \mathbf{D} . For hierarchical clustering dissimilarities between two clusters represented by two sets of objects \mathcal{X} and \mathcal{Y} are typically calculated by one of the following methods.

$$\text{complete-link: } d_c(\mathcal{X}, \mathcal{Y}) = \max\{d(x, a) : x \in \mathcal{X}, y \in \mathcal{Y}\} \quad (12)$$

$$\text{single-link: } d_s(\mathcal{X}, \mathcal{Y}) = \min\{d(x, a) : x \in \mathcal{X}, y \in \mathcal{Y}\} \quad (13)$$

$$\text{average-link: } d_a(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}| \cdot |\mathcal{Y}|} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} d(x, y) \quad (14)$$

Complete-link (single-link) respectively use the largest (smallest) possible dissimilarity between any two objects, one of each set. Average-link computes the average of all pairwise dissimilarities between objects of the two sets. In set theory the *Hausdorff metric* (Hausdorff, 2001) is used to calculate the dissimilarity between two sets defined from pairwise dissimilarities between the elements in the two set. The metric is defined as:

$$d_H(\mathcal{X}, \mathcal{Y}) = \max\{\sup_{x \in \mathcal{X}} \inf_{y \in \mathcal{Y}} d(x, y), \sup_{y \in \mathcal{Y}} \inf_{x \in \mathcal{X}} d(x, y)\} \quad (15)$$

The Hausdorff metric pairs up each element from one set with the most similar element from the other set and then finds the largest dissimilarity of paired up points.

The last step in Figure 1 deals with the used color palette. Although this is an important step, color palettes are rarely discussed for matrix shading. Most applications only use gray scale and implicitly assume that the values in the matrix should be perceived to go linearly from a very dark gray (or black) for very low dissimilarity to a very light gray (or white) for high dissimilarity. This somewhat non-intuitive convention to use black for low values and white for high values follows the fact that what is visualized and thus rendered dark are the clusters represented by areas of low dissimilarity. Another way of looking at this is that actually similarities are visualized with areas of high similarity rendered dark. Dissimilarities can always be transformed to similarities. This transformation is typically non-linear since it maps dissimilarity values with possibly unbounded domain $[0, \infty)$ to similarity values in $[0, 1]$. This creates the problem that if dissimilarities are shown in a linear gray scale then similarities are not displayed linearly and vice versa.

A solution is to abandon the idea of a linear scale and focus on the intended purpose of the visualization, e.g., on the compactness of clusters, on the micro structure within clusters, or on the interaction between clusters. This can be done by using monotone and possibly non-linear transformations of the dissimilarity values to the gray values used for visualization. Here we assume that we have an output device which can display points in a gray scale going from 0 (white or a light gray) to 1 (black or a dark gray) which is perceived to be linear by the user. We define the transformation function as

$$f : [0, d_{max}] \rightarrow [0, 1], \quad (16)$$

where d_{max} is some maximal dissimilarity value, such that for all $d \in \mathbf{D}$ we have $d \leq d_{max}$.

Some examples for useful transformations are presented in Table 1 and depicted in Figure 2. Next to the linear transformation (a) simple non-linear transformations ((b) and (c)) can be used. (b) focuses the attention on compact clusters and reduces high dissimilarity “noise” while (c) makes even areas of higher dissimilarity visible. Note that the linear transformation (a) is just a special case of the non-linear transformation with $p = 1$. (d) and (e) show two possible transformations that highlight areas of lower dissimilarity. (d) simply cuts the palette off after the threshold while (e) uses a shifted Logistic distribution function for a gradual change. Note that the last two transformations should be applied with care since they can create the illusion that the found clusters are stronger than they are in reality.

With these transformations high dissimilarity “noise” can be reduced (see Table 1 (b) to (e)) which lets the user concentrate on the clusters. Or higher dissimilarities can be pronounced (see Table 1 (f)) to analyze inter cluster relationships which are typically found at slightly higher dissimilarity levels.

To use color instead of gray values, a palette that is perceived to go linearly from very low intensity to high intensity of a certain color is required. The usually employed RGB color space with colors defined by red-green-blue triplets is not perception based. Changing the values of the triplet linearly results in changes that are perceived strongly non-linear and thus will influence and distort the impression of the matrix shading. An example of a perception based colorspace is HCL where colors are defined by triplets of hue (color on a 360 degree wheel), chroma (intensity) and luminance (brightness). In this colorspace colors are balanced and choosing a linear palette is reduced to constructing a linear path through the luminance-chroma space at a given hue (color) resulting in a so-called sequential palette (Zeileis, Hornik, and Murrell, 2009). Once we have a linear palette the transformations in Table 1 can be applied in the same way as for gray scale.

Since we deal with symmetric dissimilarity matrices, the display is mirrored around the diagonal which is redundant. We can use the lower triangle of the plot to display the aggregated dissimilarities within and between clusters already calculated for the inter-cluster dissimilarity matrix \mathbf{D}_c .

In the following section we will present several examples which show the usefulness of dissimilarity plots.

Figure 2	Description	Transformation functions
(a)	Linear palette	$f(d) = 1 - d/d_{max}$
(b), (c)	Non-linear palette using the power p	$f(d) = (1 - d/d_{max})^p$
(d)	Linear palette with threshold t	$f(d) = \begin{cases} 1 - d/d_{max} & \text{if } d \leq t, \\ 0 & \text{otherwise.} \end{cases}$
(e)	Non-linear palette (Logistic distribution function) with threshold t and scale parameter s	$f(d) = \frac{1}{(1 + \exp((d-t)/s))}$

Table 1: Examples of transformation functions.

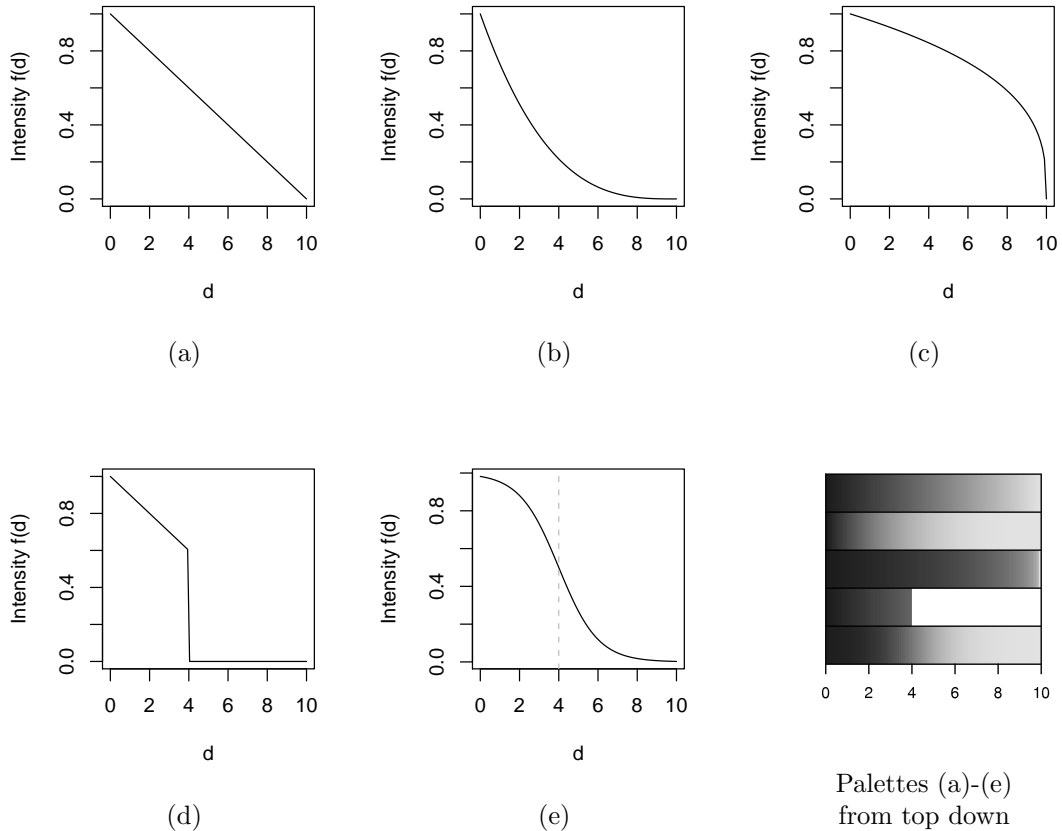


Figure 2: Transformation functions from Table 1 with $d_{max} = 10$: (a) linear palette, (b) sub-linear with $p = 3$, (c) super-linear with $p = 1/3$, (d) linear with threshold at $t = 4$, and (e) non-linear (logistics distribution function) with $t = 4$ and $s = 1$.

4 Examples

In this section we present several examples to show the potential of dissimilarity plots. For the examples we use the column/row gradient measure as the loss function for seriation and aggregate dissimilarities between clusters using average pairwise dissimilarity. The seriation of clusters is done using branch-and-bound to find the optimal solution (Brusco and Stahl, 2005). For the seriation for all objects in each cluster we use a simulated annealing heuristic (Brusco et al., 2008). The implementation of both algorithms is provided by Michael Brusco and is available in the R extension package `seriation` (Hahsler, Buchta, and Hornik, 2009).

4.1 Easily distinguishable groups

First we look at the *Ruspini* data set (Ruspini, 1970) which is popular for illustrating clustering techniques. It consists of 75 points in two-dimensional space with four clearly distinguishable groups.

We calculated a dissimilarity matrix using the euclidean distance and we used a k -medoids clustering algorithm (partitioning around medoids (PAM); Kaufman and Rousseeuw, 1990) to produce a partition with $k = 4$ clusters. We present several visualizations of the clustering in Figure 3. Figure 3(a) plots the data in two-dimensional space using principal components analysis (PCA). The four clusters are clearly separated and it is visible that clusters 2 and 4 are closer together. Figure 3(b) shows the silhouette widths for all objects. The clean clustering is represented here by the fact that all objects have large silhouette widths indicating that they fit well in the cluster they are assigned to and that they are far from the objects in the next closest cluster. Figure 3(c) shows the unordered dissimilarity matrix. Dissimilarity values are represented by gray values given in the color key below the plot. Figure 3(d) is the result of the dissimilarity plot. The clearly visible dark squares on the diagonal and the lighter off-diagonal blocks indicate that the clusters are well defined. The position of the clusters and the average cluster dissimilarity plotted in the lower left triangle indicate that clusters 3 and 1 and clusters 2 and 4 are closer together while 3 and 4 (the endpoints of the plot) are the most dissimilar.

For this example all techniques provided good results which was to be expected given the “easy” to cluster data set. Next we will see how dissimilarity plots help to identify a mis-specification of the number of clusters.

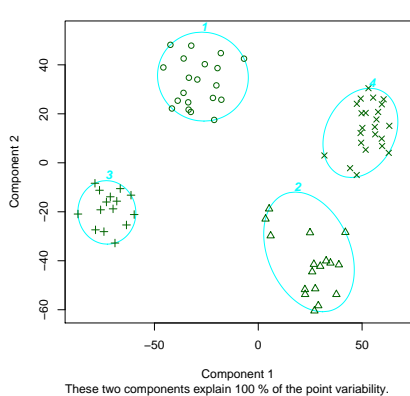
```
R> clusplot(cl_ruspini, labels = 4, lines = 0, main = "")
R> plot(silhouette(cl_ruspini), main = "")
R> dissplot(d_ruspini, method = NA, options = list(main = ""))
R> dissplot(d_ruspini, cl_ruspini$cluster, method = "BBURCG",
+ options = list(main = ""))
```

4.2 Mis-specification of the number of clusters

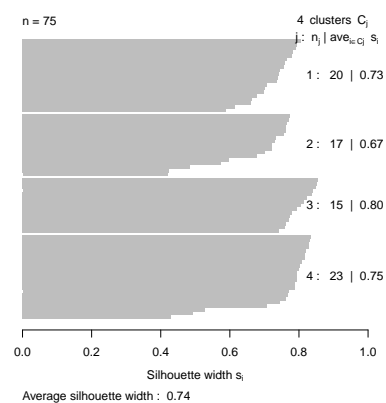
We use again the *Ruspini* data set with four groups but this time we mis-specify the number of clusters and use first $k = 3$ and then $k = 7$. We again use PAM to create the clusterings.

Figure 4 shows the two dissimilarity plots. In Figure 4(a) the number of clusters is $k = 3$. Clusters 2 and 3 are clearly visible as two dark squares. However, cluster 1 is on average less compact (triangle below the diagonal) and looking at the structure of the objects within the cluster reveals two clearly distinct groups (the two dark triangles above the diagonal). This indicates that the true number of groups must be larger than the number of clusters used for clustering.

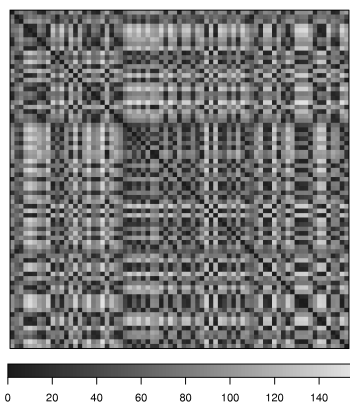
The dissimilarity plot in Figure 4(b) shows the result for a too large number of clusters. Here the clustering algorithm breaks some natural groups into several clusters to meet the requirement of seven clusters. The dissimilarity plot rearranges the clusters to show that actually some clusters belong together to form a natural group making the mis-specification of the numbers of clusters apparent.



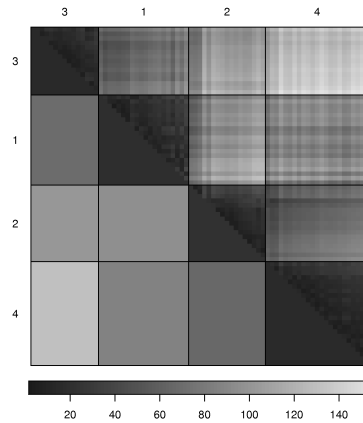
(a)



(b)



(c)



(d)

Figure 3: Plots for the Ruspini data set with 4 clearly separated clusters. (a) Projection of the data on its first two principal components, (b) silhouette plot, (c) matrix shading of the unsorted dissimilarity matrix, and (d) seriated dissimilarity plot.

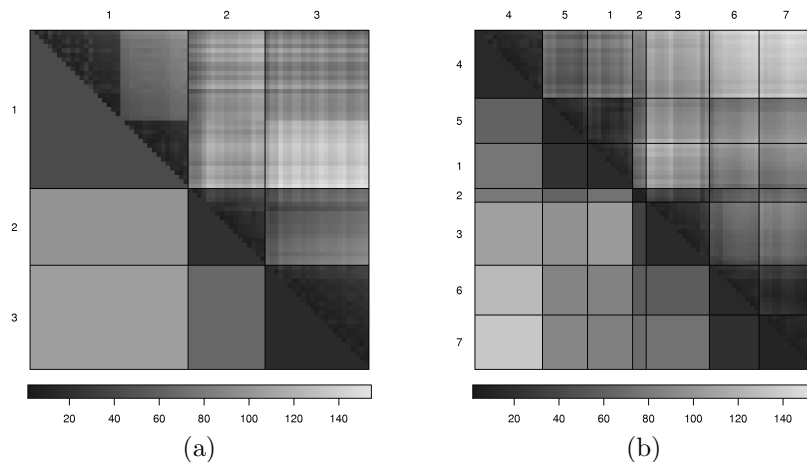


Figure 4: Dissimilarity plots for the Ruspini data set with mis-specified number of clusters. (a) Three instead of four clusters, and (b) Seven instead of four clusters

```
R> cl_ruspini_3 <- pam(d_ruspini, k = 3)
R> dissplot(d_ruspini, cl_ruspini_3$cluster, method = list("BBURCG",
+ "ARSA"), options = list(main = ""))

R> cl_ruspini_7 <- pam(d_ruspini, k = 7)
R> res <- dissplot(d_ruspini, cl_ruspini_7$cluster, method = list("BBURCG",
+ "ARSA"), options = list(main = ""))
```

4.3 Data without structure

Next we look at data which do not contain any structure. Clustering algorithms will still find the specified number of clusters and it is important to identify the clustering result as an artifact rather than a found grouping in the data.

We generate random data for 250 objects in five-dimensional space. The data point of each object is chosen randomly from a standard normal distribution. Distances between objects are calculated using euclidean distance and clustering is performed for $k = 10$ with PAM.

The results are presented in Figure 5. The projection of the data onto its first two principal components in Figure 5(a) shows that all clusters overlap and no structure is visible. This is a sign of a “bad” clustering, however, the first two principal components only account for 41.92% of the variability of the data indicating that separation between the clusters not visible in the plot might exist. Figure 5(b) shows rather narrow silhouettes with even several negative values which indicates that several objects are closer to objects in other clusters than to its own medoid. Narrow silhouettes suggests a “weak” clustering, but they can also be the result of the dimensionality of the data set. The dissimilarity plot in Figure 5(c) shows almost no variation in gray value over the whole matrix. To focus on areas of low dissimilarity which are potentially useful clusters, we use in Figure 5(d) a non-linear palette (with $p = 3$). However, the squares on the diagonal fade together with the rest of the plot towards a lighter gray, a clear sign that the clustering produced no useful results.

```
R> n <- 250
R> m <- 5
R> random <- matrix(rnorm(n * m), ncol = m)
R> d_random <- dist(random)
R> cl_random <- pam(d_random, k = 10)

R> clusplot(cl_random, labels = 4, lines = 0, main = "")
R> plot(silhouette(cl_random), main = "")
```

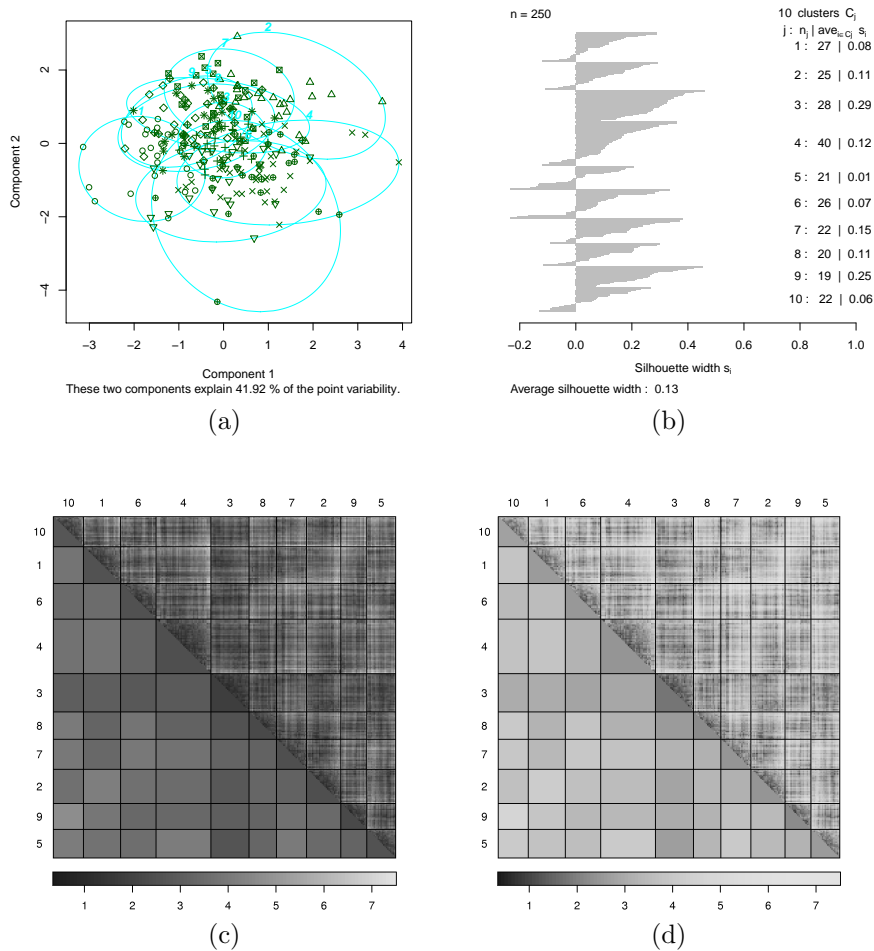


Figure 5: Plots for random data. (a) Projection of the data onto its first two principal components, (b) silhouette plot, (c) dissimilarity plot with linear palette, and (d) dissimilarity plot with a non-linear palette (cubic; $p = 3$) to highlight strong clusters.

```
R> res <- dissplot(d_random, cl_random$clustering, method = list("BBURCG",
+ "ARSA"), options = list(main = ""))
```

4.4 High-dimensional data

To show how dissimilarity plots work with higher-dimensional data, we use the *Votes* data set available via the UCI Repository of Machine Learning Databases (Asuncion and Newman, 2007). This data set includes votes (voted for, voted against and unknown) for each of the U.S. House of the 435 Representatives congressmen on the 16 key votes during the second session of 1984. Hence the dimensionality of the data set is 16.

We preprocessed the data such that votes in favor are coded as 1 and everything else (vote against and unknown) is coded as 0. Then we used the Jaccard index (Sneath and Sokal, 1973) to calculate a dissimilarity matrix between congressmen. This is the number of votes two congressmen both voted in favor divided by the total number of votes any of the two voted in favor.

For clustering we used again PAM. To decide on the number of clusters we plotted the average silhouette values for $k = 2, 3, \dots, 30$. The first bump in the plot occurred at $k = 12$ which we selected as the number of clusters.

The results of the clustering are shown in Figure 6. Due to the high dimensionality, the projection of the objects onto its first two principal components in Figure 6(a) only explains 40.45% of the variability and does not reveal too much other than that all the clusters seem to be arranged in a circle. The silhouette plot in Figure 6(b) shows a rather small average silhouette width of 0.14 which can be the result of the higher dimensionality of the data. From the individual silhouettes, it seems that cluster 3 is especially bad since most of its points are closer to points in other clusters than to its own medoid resulting in an on average negative silhouette width for the cluster. Figure 6(c) shows the dissimilarity plot. A light block separates two clearly defined groups, a larger group including clusters 7, 5, 8, 4, 6 and 12 and the smaller group including clusters 10, 11, 2, 1. The separation becomes even clearer in Figure 6(d) where we removed all dissimilarity values greater than 0.7. Two clusters (3 and 9) are related to both groups but have more similarity to the smaller group.

Since the Votes data set contains also class labels (party affiliation of congressmen), we can create a cross-table for clusters and classes. The result is presented in Table 2. To make it easier to compare it with the dissimilarity plot, we arranged the clusters in the table in the same order as in the plot. The larger block in the dissimilarity plot contains almost exclusively Democrats while the smaller cluster is dominated by Republicans. Clusters 3 and 9 consists of mostly Democrats and Republicans, respectively, who seem to share many views with Republicans but also vote frequently with Democrats.

After analyzing the dissimilarity plot, the projection of the clusters in Figure 6(a) makes more sense. The clusters run in a half-circle from cluster 2 to cluster 5 (using Table 2 reveals that is from Republicans to Democrats) and cluster 3 lays across all other clusters. However, this interpretation can only be reached after studying the dissimilarity plot.

```
R> set.seed(1234)
R> library(cba)
R> data(Votes)
R> x <- as.dummy(Votes[-17])
R> d_votes <- dist(x, method = "binary")
R> labels_votes <- pam(d_votes, k = 12, cluster.only = TRUE)

R> clusplot(d_votes, diss = TRUE, labels_votes, labels = 4,
+         lines = 0, main = "")

R> plot(silhouette(labels_votes, d_votes), main = "")

R> dp_votes <- dissplot(d_votes, labels_votes, method = list("BBURCG",
+ "ARSA"), options = list(main = ""))

R> plot(dp_votes, options = list(main = "", threshold = 0.7))

R> library(xtable)
R> tab <- table(labels_votes, Votes$Class)
R> n <- 1:nrow(tab)
R> tab <- cbind(n, tab)[dp_votes$cluster_order, ]
R> colnames(tab) <- c("Cluster", "Democrats", "Republicans")
R> rownames(tab) <- NULL
R> tabx <- xtable(tab, caption = "Cluster composition", label = "tab:votes")
R> print(tabx, type = "latex", floating = TRUE, table.placement = "tbp")
```

5 Conclusion

This paper demonstrates that the two well-known methods of matrix shading and seriation can be combined into dissimilarity plots, a new and powerful visualization technique that provides many advantages over existing techniques for visualizing partitionial clustering. Most notably the new method is independent on the dimensionality of the data set and by reordering clusters and objects within clusters provides a very concise structural representation of the clustering. We have shown that dissimilarity plots are also helpful in spotting the mis-specification of the number of clusters used for clustering.

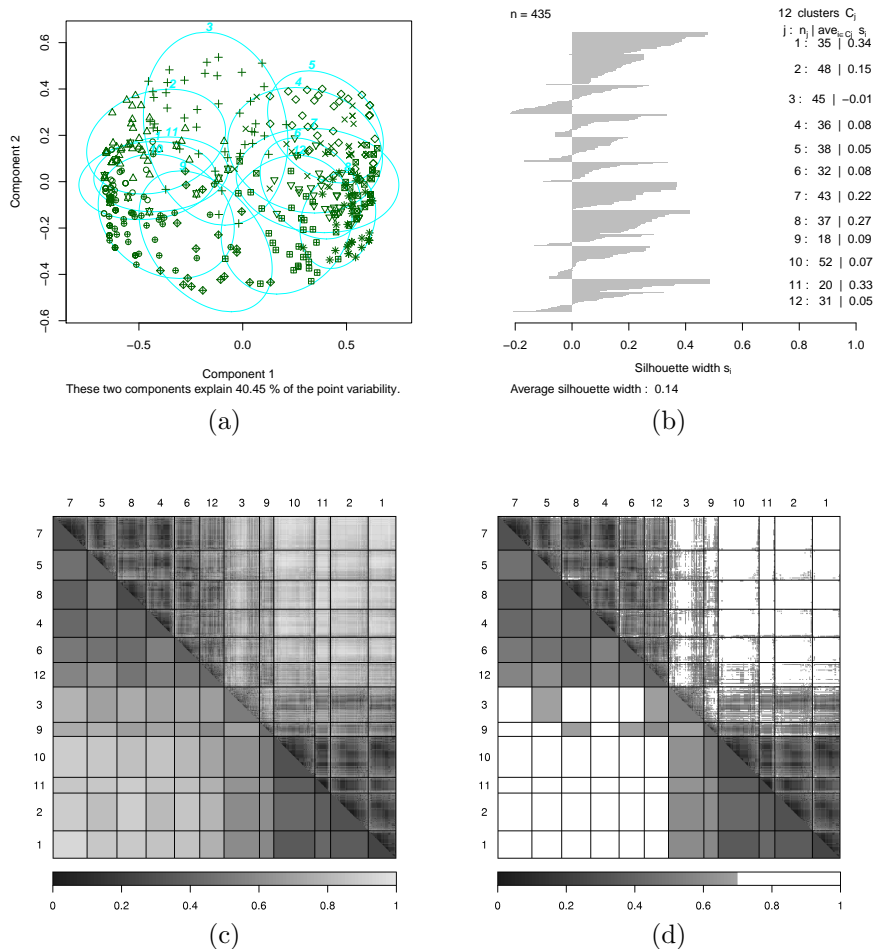


Figure 6: Plots for the Votes data set. (a) Projection of the data on its first two principal components, (b) silhouette plot, (c) dissimilarity plot, and (d) dissimilarity plot with shading threshold.

Appendix: Software

A R extension package called **seriation** is available from the Comprehensive R Archive Network web site (CRAN.R-project.org) or can be directly installed from within R. The package contains the seriation methods and the visualization technique discussed in this paper.

The dissimilarity plots function is called by

```
dissplot(x, labels = NULL, method = NULL, control = NULL, options = NULL)
```

where x is the dissimilarity matrix, **labels** contains a vector with integers representing the assignment function Γ , **method** selects the used seriation method, **control** can contain additional options for the chosen seriation method, and **options** can be used to modify the display (color palette, title, where to display between cluster dissimilarities, etc.).

In the current version the following seriation methods are available: branch-and-bound solver for the gradient measure, simulated annealing and dynamic programming for the gradient measure, exact solver and various heuristics for the Hamiltonian path problem, hierarchical clustering (used between clusters or within clusters) plus optimal leaf ordering or with the heuristic by Gruvaeus and Wainer. A detailed description these seriation methods can be found in the package description (Hahsler et al., 2009).

For HCL color support the R extension package **colorspace** (Ihaka, Murrell, Hornik, and Zeileis, 2008) is used. Different options for the transformation function f are implemented

	Cluster	Democrats	Republicans
1	7	42	1
2	5	38	0
3	8	36	1
4	4	34	2
5	6	32	0
6	12	26	5
7	3	41	4
8	9	2	16
9	10	3	49
10	11	5	15
11	2	7	41
12	1	1	34

Table 2: Cluster composition

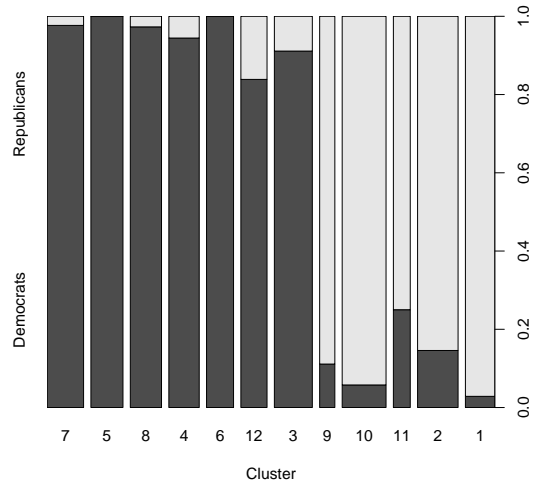


Figure 7: Visualization of cluster composition as a spine plot with reordered clusters.

directly into `disssplot()` via the `options` parameter. Linear and non-linear (using the power transformation) gray scales and color palettes can be directly selected and also a transformation function with threshold is available. Arbitrary transformations can be applied by creating a custom color palette using `hc1()` defined in the R core package `grDevices`.

The resolution of the used display (current mass market displays offer a resolution of up to 1920×1200 pixels) restricts the size of the dissimilarity matrix that can be displayed. For larger data sets several methods are possible:

1. Sampling of objects. This reduces the size of the dissimilarity matrix and therefore also speeds up the construction of the dissimilarity plot. However, details are sacrificed.
2. Image downsampling. After the full size reordered dissimilarity matrix is created the size of the image is reduced to fit the display (e.g., by using pixel skipping, pixel averaging or 2D discrete wavelet transformation).
3. Splitting the plot. To retain all information in the plot, in a first stage only a plot with dissimilarities between clusters can be displayed. Then the user can zoom into individual clusters using the whole available display for only a single cluster.

The first two options are already supported by R. Sampling is a built-in function and image downsampling is done by the graphics device. The third option currently needs to be done manually and a more convenient solution is part of future development.

References

- D. Applegate, R. Bixby, V. Chvátal, and W. Cook. *Concorde TSP Solver*, 2006. URL <http://www.tsp.gatech.edu/concorde/>.
- P. Arabie and L. J. Hubert. An overview of combinatorial data analysis. In P. Arabie, L. J. Hubert, and G. D. Soete, editors, *Clustering and Classification*, pages 5–63. World Scientific, River Edge, NJ, 1996.
- A. Asuncion and D. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. URL <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- Z. Bar-Joseph, E. D. Demaine, D. K. Gifford, and T. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(1):22–29, 2001.
- J. Bertin. *Graphics and Graphic Information Processing*. Walter de Gruyter, Berlin, 1981. Translated by William J. Berg and Paul Scott.
- M. Brusco and S. Stahl. *Branch-and-Bound Applications in Combinatorial Data Analysis*. Springer, 2005.
- M. Brusco, H. F. Köhn, and S. Stahl. Heuristic implementation of dynamic programming for matrix permutation problems in combinatorial data analysis. *Psychometrika*, 73:503–522, Sep 2008.
- G. Caraux and S. Pinloche. Permutmatrix: A graphical environment to arrange gene expression profiles in optimal linear order. *Bioinformatics*, 21(7):1280–1281, 2005.
- C.-H. Chen. Generalized association plots: Information visualization via iteratively generated correlation matrices. *Statistica Sinica*, 12(1):7–29, 2002.
- M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science of the United States*, 95(25):14863–14868, December 1998.
- M. Friendly. Mosaic displays for multi-way contingency tables. *Journal of the American Statistical Association*, 89(425):190–200, 1994.
- M. Friendly. Corrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 56(4):316–324, Nov 2002.

- N. Gale, W. C. Halperin, and C. M. Costanzo. Unclassed matrix shading and optimal ordering in hierarchical cluster analysis. *Journal of Classification*, 1:75–92, 1984.
- G. Gruvaeus and H. Wainer. Two additions to hierarchical cluster analysis. *British Journal of Mathematical and Statistical Psychology*, 25:200–206, 1972.
- G. Gutin and A. P. Punnen, editors. *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Kluwer, Dordrecht, 2002.
- M. Hahsler, K. Hornik, and C. Buchta. Getting things in order: An introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, March 2008.
- M. Hahsler, C. Buchta, and K. Hornik. *seriation: Infrastructure for seriation*, 2009. URL <http://CRAN.R-project.org/package=seriation>. R package version 1.0-1.
- J. A. Hartigan. Representation of similarity matrices by trees. *Journal of the American Statistical Association*, 62(320):1140–1158, 1967.
- F. Hausdorff. *Set Theory*. American Mathematical Society, 5th edition, 2001.
- L. Hubert, P. Arabie, and J. Meulman. *Combinatorial Data Analysis: Optimization by Dynamic Programming*. Society for Industrial Mathematics, 1987.
- L. J. Hubert. Some applications of graph theory and related nonmetric techniques to problems of approximate seriation: The case of symmetric proximity measures. *British Journal of Mathematical Statistics and Psychology*, 27:133–153, 1974.
- C. B. Hurley. Clustering visualizations of multidimensional data. *Journal of Computational and Graphical Statistics*, 13(4):788–806, Dec 2004.
- R. Ihaka, P. Murrell, K. Hornik, and A. Zeileis. *colorspace: Color Space Manipulation*, 2008. URL <http://CRAN.R-project.org/package=colorspace>. R package version 1.0-0.
- L. Kaufman and P. J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*. John Wiley and Sons, New York, 1990.
- F. Leisch. Visualizing cluster analysis and finite mixture models. In C. Chen, W. Härdle, and A. Unwin, editors, *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics. Springer Verlag, 2008.
- S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.
- R. L. Ling. A computer generated aid for cluster analysis. *Communications of the ACM*, 16(6):355–361, 1973.
- G. Pison, A. Struyf, and P. J. Rousseeuw. Displaying a clustering with clusplot. *Computational Statistics & Data Analysis*, 30(4):381–392, June 1999.
- W. S. Robinson. A method for chronologically ordering archaeological deposits. *American Antiquity*, 16:293–301, 1951.
- P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.
- E. H. Ruspini. Numerical methods for fuzzy clustering. *Information Science*, 2:319–350, 1970.
- P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman and Company, San Francisco, 1973.
- A. Strehl and J. Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.
- L. Wilkinson and M. Friendly. The history of the cluster heat map. *The American Statistician*, 63(2):179–184, May 2009.

- D. Wishart. Clustangraphics3: Interactive graphics for cluster analysis. In W. Gaul and H. Locarek-Junge, editors, *Data Analysis and Knowledge Organization: Classification in the Information Age*, Studies in Classification, pages 268–275. Springer, 1999.
- A. Zeileis, K. Hornik, and P. Murrell. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9):3259–3270, 2009. ISSN 0167-9473.