Advances in Dendrogram Seriation for Application to Visualization

Denise EARLE and Catherine B. HURLEY

Visualizations of statistical data benefit from systematic ordering of data objects to highlight features and structure. This article concerns ordering via dendrogram seriation based on hierarchical clustering of data objects. It describes DendSer, a general-purpose dendrogram seriation algorithm which when coupled with various seriation cost functions is easily adapted to different visualization settings. Comparisons are made with other dendrogram seriation algorithms and applications are presented. Supplementary materials for this article are available online.

Keywords: Anti-Robinson; Hierarchical clustering; Importance sorting; Reordering.

1. INTRODUCTION

Statistical data arrive for analysis in a data file, which is then read in to a data matrix or data frame. These structures necessarily impose a linear ordering on the cases and variables in the dataset. Unless the data have a time ordering or the objects are organized into groups, these linear orderings are not meaningful. However, in visualizations of multiple variables such as scatterplot matrices or parallel coordinate plots, or visualizations of multiple cases such as glyph arrays, these often meaningless linear orderings impact on the interpretability and effectiveness of the visualization. Many authors have demonstrated that data visualizations are greatly improved when the data components are systematically reordered. Bertin (1983) described the “reorderable matrix,” where he showed that reordering the rows and columns of a data matrix simplifies understanding. Friendly and Kwan (2003) demonstrated “effect ordering” of various data displays and for scatterplot matrices, Hurley (2004) ordered variables so that interesting panels were positioned close to the main diagonal. In each of these situations, reordering revealed patterns in data that were not obvious when the default order of variables or categories was used.

In the data analysis literature, the term seriation is used to describe a systematic reordering of data objects, with the goal of revealing structural information. Note that ecologists refer to this task as ordination. Kendall (1971) credited Petrie (1899) as being the first to use formal seriation methods. Robinson (1951) provided a mathematical framework for
the problem of seriation. He proposed a desired form for a similarity matrix, now known
as Robinson form, where entries are nonincreasing moving away from the main diagonal
along rows and columns. For historical background, see the Wilkinson and Friendly (2009)
article on the cluster heatmap and Liiv (2010) who provided a general overview.

Recent contributions to the combined areas of seriation and visualization include Eisen
et al. (1998), Chen (2002), Friendly and Kwan (2003), and Hurley (2004). Software ad-
vancements have also influenced the development of seriation and visualization tools, in
particular the R (R Core Team 2013) package seriation (Hahsler, Hornik, and Buchta
2008; Hahsler, Buchta, and Hornik 2013) which includes a range of algorithms.

There are two main challenges for seriation. The first is evaluating the “goodness” of
a permutation of a set of objects. In typical applications, good permutations place similar
objects nearby and dissimilar objects far apart. A path length cost function measuring the
sum of dissimilarities between adjacent objects captures this. Other cost functions measure
the closeness of a dissimilarity matrix to anti-Robinson form, see, for example, Wishart
(1999) and Hubert, Arabie, and Meulman (2001). For visualization applications, interobject
comparison measures other than similarities/dissimilarities are also useful (Hurley 2004).

The second challenge is finding a good permutation. Of course, for more than about
$n = 10$ objects, an exhaustive search of all possible $n!$ permutations is infeasible. There are
many heuristic techniques which produce good but not necessarily optimal permutations
for the seriation problem. These include traveling salesperson heuristics (see, e.g., Lawler
et al. 1985), simulated annealing algorithms (see, e.g., Brusco, Kohn, and Stahl 2008), and
dimension reduction techniques such as principal components analysis (as used by Friendly
and Kwan 2003, for example). This article focuses on one branch of seriation algorithms
called dendrogram seriation.

A dendrogram is a tree structure used to visualize the process of hierarchical clustering.
This article concerns dendrograms which are binary trees, as is the case with popular
hierarchical clustering algorithms, though the methodology also applies to the more general
setting. Figure 1(a) depicts a dendrogram showing the process of clustering five objects. The
horizontal axis shows the five objects (leaves). Each new cluster in the clustering process
is represented as a node in the dendrogram, labeled $N_1$, $N_2$, $N_3$, and $N_4$. The vertical axis
represents the dissimilarity level at which the cluster was formed, but is left unlabeled in this toy example.

The order of the leaves in a dendrogram provides a permutation of a set of objects. For a clustering of \( n \) objects, there are \( n - 1 \) dendrogram nodes, each of which may be rearranged resulting in a total of \( 2^{n-1} \) possible permutations of the objects. For example, the leaves of dendrogram of Figure 1(a) are in order b, d, a, e, c. Figure 1(b) visualizes the same hierarchical clustering as does Figure 1(a), but the leaf ordering is now a, e, d, b, c. On inspection, one sees that the positions of the clusters joined at nodes \( N_1 \) and \( N_2 \) have been translated, that is, interchanged.

Dendrogram seriation was introduced by Gruvaeus and Wainer (1972), whose motivation was to obtain a unique ordering of the objects from a hierarchical clustering. Since then, many authors (Degerman 1982; Gale, Halperin, and Costanzo 1984; Eisen et al. 1998; Alon et al. 1999; Wishart 1999; Bar-Joseph, Gifford, and Jaakkola 2001) have presented other dendrogram seriation algorithms, with the objective of placing similar objects nearby. In recent years, a popular application for these algorithms is heatmap visualization of dissimilarities based on gene expression data, see, for example, Eisen et al. (1998). Hurley (2004) used dendrogram seriation to improve other visualizations, namely scatterplot matrices and parallel coordinates, finding arrangements of variables that highlight features such as correlation, group separation, and nonnormality.

In this article, we describe a general dendrogram seriation algorithm called DendSer. Our motivation is improvement of data visualizations, including but not limited to scatterplot matrices, parallel coordinate displays, glyph displays, and heatmaps. Hurley (2004) used the original Gruvaeus and Wainer (1972) (GW) algorithm for this purpose, however DendSer with various cost functions tackles this problem in a more systematic and general way. The work presented here is based on the unpublished PhD dissertation of Earle (2010) and a brief description of some topics appeared in Earle and Hurley (2011).

The article is organized as follows. The DendSer algorithm is described in Section 2.1. DendSer is a greedy algorithm in that it does not examine all \( 2^{n-1} \) permutations. However, the algorithm has settings which control the search space, and these are described in Section 2.2. We also describe a selection of cost functions in Section 2.3, each of which tailors dendrogram seriation to a specific visualization setting. In Section 3, we explore the DendSer algorithm, specifically, the search space parameters in Section 3.1, comparing with other dendrogram seriation algorithms in Section 3.2, and via toy examples in Section 3.3. Section 4 presents some visualization applications, which demonstrate our new cost functions and highlight the flexibility of DendSer. Finally, in Section 5 we make some concluding remarks.

## 2. GENERALIZED DENDROGRAM SERIATION

In our setup, we have \( n \) objects which we wish to layout for visualization purposes. We consider linear layouts only. Objects are typically statistical cases or variables. In our examples, we use visualizations which portray the objects directly (e.g., glyph displays of cases, dataset heatmaps), or show interobject comparisons or relationships (e.g., scatterplot matrices, parallel coordinate displays).
Each interobject comparison has an associated *seriation weight* $w_{ij}$ for objects $i$ and $j$, measuring the importance of the $(i, j)$ comparison in the visualization, with lower values indicating higher importance. A visualization that emphasizes important comparisons in a linear layout could have low weight pairs of objects in adjacent or perhaps nearby positions. Let $\pi$ denote a permutation of the $n$ objects and let $\mathcal{F}$ denote a cost function evaluating a permutation based on the seriation weights $w_{ij}$. The goal of seriation is then to find the permutation $\pi$ minimizing $\mathcal{F}$.

The particular choices for the cost function and seriation weights depend on the visualization. In many cases, $w_{ij}$ is a measure of object dissimilarity, and the cost function $\mathcal{F}$ measures path length or closeness to anti-Robinson form (e.g., Wishart 1999 and Hubert, Arabie, and Meulman 2001), both of which reward permutations which place low weight object pairs nearby. These along with some new suggestions for cost functions are presented in Section 2.3.

Generally, except for small $n$ it is not feasible to try and find the permutation $\pi$ minimizing $\mathcal{F}$. Instead, we adopt a two-stage minimization process:

1. Perform a hierarchical clustering using the seriation weights $w_{ij}$ in place of measures of dissimilarity. This yields a dendrogram $\Delta$.

2. Rearrange the nodes of $\Delta$ so that the leaf order minimizes $\mathcal{F}$.

Dendrogram seriation has a number of advantages over other techniques. It simplifies the problem of seriating $n$ objects by reducing the size of the search space from $n!$ permutations to $2^{n-1}$ permutations. It produces an ordering of objects consistent with the clustering structure, which is useful in many applications. Third, dendrogram seriation is applicable with a wide range of seriation cost functions. Finally, dendrogram seriation is efficient when compared with other techniques, and so is applicable to settings with large $n$.

The remainder of this section describes DendSer, our dendrogram seriation algorithm which implements stage two of the above minimization process.

### 2.1 DendSer: A Dendrogram Seriation Algorithm

The original (GW) dendrogram seriation algorithm of Gruvaeus and Wainer (1972) examines each node in turn and rearranges its left and right subnodes so that the two most low-weight objects at the edges of the subnodes are placed adjacently. Figure 2 illustrates

\[
\begin{align*}
\text{(a) Original} & & \text{(b) Reverse } N_r & & \text{(c) Reverse } N_l & & \text{(d) Reverse } N_l \text{ and } N_r \\
\end{align*}
\]

Figure 2. Four rearrangements of node $N$ examined by the Gruvaeus and Wainer (1972) algorithm. Here $a, \ldots, b$ denote leaves of $N_l$, $c, \ldots, d$ are leaves of $N_r$. $\bigcirc$ denotes a node whose leaf order is reversed.
this step for a single node $N$, with left and right subnodes $N_l$ and $N_r$ respectively. The leaves of $N_l$ and $N_r$ are $a$, $b$, $c$, and $d$ respectively. The leftmost dendrogram shows the starting arrangement of $N$. The following dendrograms reverse the order of the leaves of one or both of $N_l$ and $N_r$. The algorithm finds the smallest of $w_{bc}$, $w_{bd}$, $w_{ac}$, and $w_{ad}$ and selects the corresponding dendrogram rearrangement.

More recent dendrogram seriation algorithms (Degerman 1982; Gale, Halperin, and Costanzo 1984; Eisen et al. 1998; Alon et al. 1999; Wishart 1999; Bar-Joseph, Gifford, and Jaakkola 2001; Morris, Asnake, and Yen 2003; Forina et al. 2007; Tien et al. 2008; Wu, Tien, and Chen 2010; Chae and Chen 2011) differ from the above method in (1) the rearrangements evaluated at each node $N$, (2) the criterion used to compare the rearrangements, and (3) the order and choice of nodes to be visited. By comparison, our proposed DendSer algorithm offers a choice of options for (1), (2), and (3), enabling dendrogram seriation to be tailored to a range of visualization settings.

First, we introduce some notation. Let $N_1, N_2, \ldots, N_{n-1}$ denote the $n-1$ dendrogram nodes of $\Delta$ listed in order of increasing join height. Let $\Pi(N; \Delta)$ denote the set of all permutations of the leaves of dendrogram $\Delta$ obtained by rearrangements of the subtree at node $N$. At the root node $R \equiv N_{n-1}$, the set $\Pi(R; \Delta)$ consists of all $2^{n-1}$ permutations of $n$ objects permitted by the structure imposed by $\Delta$. As before, let $F$ be a function measuring the cost of a permutation. Then, the goal of dendrogram seriation is to find the permutation $\pi^*$ of dendrogram leaves minimizing $F$, that is,

$$\pi^* = \arg\min_{\pi \in \Pi(R; \Delta)} F(\pi).$$

However, DendSer is a greedy algorithm in that it does not examine all $2^{n-1}$ permutations. Rather it evaluates $F$ on a much smaller number of permutations of dendrogram leaves obtained by rearrangements at each node. Let $\Pi(N; \Delta)$ denote some small set of permutations of the leaves of dendrogram $\Delta$ obtained by rearrangements of the subtree at node $N$. Of course $\Pi(N; \Delta) \subset \Pi(N; \Delta)$. This set of permutations $\Pi(N; \Delta)$ is provided by a function $\Theta$ which we call a node operator. Choices for $\Theta$ are described in Section 2.2.

The DendSer algorithm examines the nodes $N_j$ of $\Delta$ in turn, at each stage updating $\Delta$ with the permutation

$$\pi_{\text{new}} = \arg\min_{\pi \in \Pi(N_j; \Delta)} F(\pi).$$

One iteration is completed when all $n-1$ nodes are visited. Further iterations are performed until one complete iteration leaves $\pi$ unchanged. Pseudo-code for DendSer is given as Algorithm 1.

The DendSer seriation algorithm has a number of options: (i) the choice of cost function $F$, (ii) the choice of node operator $\Theta$, and (iii) the node visit order. Choices for $\Theta$ and $F$ are presented in following subsections.

The efficiency of DendSer is essentially determined by the choice of $F$. At each node, optionally updating the dendrogram is $O(n)$, so the overall efficiency is $O(n) \times (O(n) + O(F))$, with the proportionality constant determined by the number of iterations and choice of the node operator $\Theta$. 

2.2 NODE OPERATORS

A node operator is a function \( \Theta \) that provides a set of permutations of dendrogram leaves by rearrangements of the subtree at a particular node. The choice of the node operator affects the search space of permutations explored.

There are two fundamental node operators, namely node translation and node reflection. Following Forina et al. (2007), we use the term translation to describe interchanging the left and right subnodes of a node. The reflection of a node reverses the order of its leaves (referred to as rotation in Forina et al. 2007). For nodes with two leaves, node reflection is the same as node translation. Any permutation \( \pi \) in \( \tilde{\Pi}(R; \Delta) \) may be obtained by a series of node translations or a series of node reflections.

\begin{algorithm}
\begin{algorithmic}
\Require Dendrogram: \( \Delta \), cost function: \( F \), node operator: \( \Theta \), node visit order: dir, maxloops: maximum \# iterations
\State \( n \leftarrow \# \text{leaves in } \Delta \)
\If{\( \text{dir} = \text{“up”} \)}
\State \( \Delta \)-nodes \( \leftarrow \{N_1, N_2, \ldots, N_{n-1}\} \)
\Else
\State \( \Delta \)-nodes \( \leftarrow \{N_{n-1}, N_{n-2}, \ldots, N_1\} \)
\EndIf
\State nloops \( \leftarrow 0 \)
\State \( \pi_{\text{cur}} \leftarrow \) initial permutation of leaves of \( \Delta \)
\State stoploop \( \leftarrow \) FALSE
\While{\text{not} stoploop}
\State \( \pi_0 \leftarrow \pi_{\text{cur}} \)
\State nloops \( \leftarrow \) nloops + 1
\For{\( N_i \) in \( \Delta \)-nodes}
\State \( \Pi(N_i; \Delta) \leftarrow \Theta(N_i; \Delta) \)
\State \( \pi_{\text{new}} \leftarrow \arg \min_{\pi \in \Pi(N_i; \Delta)} F(\approx) \)
\If{\( \pi_{\text{new}} \neq \pi_{\text{cur}} \)}
\State update \( \Delta \) according to \( \pi_{\text{new}} \)
\State \( \pi_{\text{cur}} \leftarrow \pi_{\text{new}} \)
\EndIf
\EndFor
\State stoploop \( \leftarrow \pi_{\text{cur}} = \pi_0 \) \text{ or } nloops = \text{maxloops}
\EndWhile
\State \Return \( \pi_{\text{cur}} \)
\end{algorithmic}
\end{algorithm}

Recall Figure 1 which illustrated the effect of translating two nodes of a four-node dendrogram. Figure 3 is a revised version of Figure 1 showing in addition the effect of reflecting the same two nodes. In Figure 3, translation of the nodes \( N_1 \) and \( N_3 \) in (a) gives the permutation a, e, b, d, c as shown in (b). Reflection of the nodes \( N_1 \) and \( N_3 \) in (a) gives the permutation e, a, b, d, c as shown in (c). Alternatively, starting from (a), the permutation...
Figure 2 illustrated a node operator which offers a choice of up to four permutations at each node. For a node $N$ whose left and right subnodes are $N_l$ and $N_r$, these four choices consist of reflecting none, one, or both of $N_l$ and $N_r$. Since this is reflection at a depth of one, we define the result of applying node operator $R_1$ as $R_1(N; \Delta) = R_0(N_l; \Delta) \cup R_0(N_r; \Delta) \cup R_0(N_l, N_r; \Delta)$. Similarly, we define a node operator $T_1$ which translates none, one, or both of $N_l$ and $N_r$.

We also consider node operators combining depth levels. Let $R_{01}$ denote an operator which at a node $N$ offers a choice of up to eight permutations, each reflecting some subset of $N_l$, $N_r$, and $N$ itself. Similarly, denote by $T_{01}$ an operator which translates some subset of $N_l$, $N_r$, and $N$ itself. Clearly, one could continue to define more operators by extending depth levels and optionally combining reflection and translation. In Section 3.1, we investigate which of the operators $T_0, R_0, C_0, R_1, T_1, R_{01}$, and $T_{01}$ coupled with an up or down node visit order gives the best results.

2.3 **Seriation Cost Functions for Visualization**

We consider cost functions $\mathcal{F}$ which evaluate a permutation $\pi$ of a set of objects. We focus on a few cost functions used in the applications of Section 4. Alternatively, the user may choose from the criteria provided by the R package *seriation* (Hahsler, Hornik, and Buchta 2008; Hahsler, Buchta, and Hornik 2013) or they may input their own user-defined measure. Generally, cost functions are based on interobject seriation weights $w_{\pi(i), \pi(j)}$, where $w_{\pi(i), \pi(j)}$ is the seriation weight for the objects at positions $i$ and $j$ in the permutation $\pi$.

2.3.1 **Path Length.** The path length cost function for a permutation $\pi$ is

$$PL(\pi) = \sum_{i=1}^{n-1} w_{\pi(i), \pi(i+1)}.$$ 

(a) Given ordering  
(b) Translate two nodes  
(c) Rotate two nodes

Figure 3. A dendrogram visualizing a hierarchical clustering of five objects. $\leftrightarrow$ denotes a node whose branches are translated. $\bigcirc$ denotes a node whose leaf order is reversed.

a, e, b, d, c could have been obtained by rotations at $N_2$ and $N_3$, while the permutation e, a, b, d, c could have been obtained by translations at $N_2$ and $N_3$.

We denote the node translation and reflection operators by $T_0$ and $R_0$, respectively. Each of these operators offers a choice of two permutations at each node. We also consider an operator $C_0$ offering the choice of translation or reflection, that is, $C_0(N, \Delta) = T_0(N, \Delta) \cup R_0(N, \Delta)$. In a later section, we show that DendSer with node operator $C_0$ converges to permutations with lower cost function values than either of $T_0$ or $R_0$.
A permutation minimizing PL aims to place low weight pairs of objects adjacently. The path length criterion has been successfully applied to a variety of statistical visualizations. Hurley (2004) described why this criterion is helpful for seriating variables in a parallel coordinates plot. Bar-Joseph, Gifford, and Jaakkola (2001) showed that minimizing the path length of a permutation helps reveal biological structure in heatmaps of gene expression data.

Finding the permutation minimizing PL is a variant of the well-known traveling salesperson problem (TSP), where there is no return to the starting city. As the TSP is NP-hard, heuristic algorithms are employed which do not necessarily find the global optimal path. In the case where paths are restricted to sequences of leaves from a dendrogram, Bar-Joseph, Gifford, and Jaakkola (2001) described an algorithm which finds the optimal path. Their optimal leaf ordering (OLO) algorithm is $O(n^3)$, while sequential algorithms such as DendSer and the DSA algorithm of Forina et al. (2007) are $O(n^2)$ but yield suboptimal solutions.

2.3.2 Lazy Path Length. Here, we propose a new criterion called lazy path length which is a variation of the path length criterion. The lazy path length (LPL) cost function for a permutation $\pi$ is

$$LPL(\pi) = \sum_{i=1}^{n-1} (n - i)w_{\pi(i),\pi(i+1)}.$$ (2)

LPL is a weighted measure of path length, where weights are $n - 1, n - 2, \ldots, 1$. It rewards permutations where the path is short and where the seriation weights are generally increasing.

The LPL problem may be regarded as a variant of the TSP, where the salesperson wants to travel a short distance in total, but he/she is also rather lazy and wants to postpone the larger distances as much as possible. (Gutin and Punnen (2002) discussed other TSP variants which also aim to satisfy two goals.)

In visualization applications, LPL-based seriation offers a form of importance sorting. Sections 4.1 and 4.3 present examples where seriating variables using the LPL criterion is effective in making interesting features more prominent in parallel coordinate plots and scatterplot matrices.

2.3.3 Anti-Robinson Form. A symmetric matrix where entries $w_{ij}$ are nondecreasing moving away from the main diagonal along rows and columns is said to have anti-Robinson form (Robinson 1951), that is, $w_{ij} \leq w_{ik}$ and $w_{jk} \leq w_{ik}$, for $1 \leq i < j < k \leq n$. Anti-Robinson form is a natural concept for seriation. If a matrix $W$ has anti-Robinson form, then object pairs with low seriation weights are close together in the corresponding permutation and object pairs with high weight are far apart.

Anti-Robinson form has several applications in seriating statistical graphics. Gale, Halperin, and Costanzo (1984) and Wishart (1999) aimed to seriate dissimilarity matrices so that they were as close as possible to anti-Robinson form, to produce more visually appealing heatmaps. Hurley (2004) described how anti-Robinson form is a desirable pattern for a scatterplot matrix. Morris, Asnake, and Yen (2003) used anti-Robinson form to improve network visualization.

Many algorithms exist that uncover anti-Robinson form if it is present (see, e.g., Hubert 1974). However, in most cases it is only possible to reorder a matrix so that it follows
approximate anti-Robinson form. There are many ways of measuring closeness of a matrix
to anti-Robinson form (see, e.g., Hubert, Arabie, and Meulman 2001, p. 55). In the present
article, we use the following measure:

\[ \text{ARc}(\pi) = \sum_{|i - j| \leq n-1} (n - |i - j|) w_{\pi(i), \pi(j)}. \]  

(3)

ARc recasts the matrix product proposal of Hubert, Arabie, and Meulman (2001) as a cost
function. ARc is calculated in \( O(n^2) \) steps. Other anti-Robinson measures check whether
\( w_{ij} \leq w_{ik} \) and \( w_{jk} \leq w_{ik} \), for \( 1 \leq i < j < k \leq n \) and take \( O(n^3) \) steps.

2.3.4 Banded Anti-Robinson Form. Banded anti-Robinson form is a relaxed version of
anti-Robinson form. It requires anti-Robinson pattern for entries in the first \( b \) off-diagonals.
Also, entries within band \( b \) of the main diagonal should be smaller than those outside the
band. We define a symmetric matrix \( W = [w_{ij}] \) to have banded anti-Robinson form of size
\( b \) with \( 1 < b < n \) if, for \( 1 \leq i < j < k \leq n \)

1. \( w_{ij} \leq w_{ik} \) and \( w_{jk} \leq w_{ik} \), for \( |i - k| \leq b \).
2. \( w_{ik} \leq w_{i'k'} \), for \( |i - k| \leq b \) and \( |i' - k'| > b \).

We use the following cost function to measure closeness of a matrix to banded anti-Robinson
form

\[ \text{BAR}(\pi) = \sum_{|i - j| \leq b} (b + 1 - |i - j|) w_{\pi(i), \pi(j)}. \]  

(4)

When \( b = 1 \), the BAR cost function is identical to PL of Equation (1) and when \( b = n - 1 \), BAR is identical to ARc as given in Equation (3). Both ARc and BAR are useful
for smoothing out heatmaps and revealing structure between objects and clusters. BAR
is a compromise between the extremes of path length, which measures local structure in
the sense of using the first diagonal in the seriated weight matrix, and ARc, which uses
all \( n - 1 \) diagonals. In the examples of Sections 3 and 4, we use our default choice of
\( b = \lfloor n/5 \rfloor \), which we have found works well.

2.3.5 Leaf Sort. A simple and efficient way to rearrange a dendrogram is to optionally
translate each node on comparing its left and right branch. If the comparator depends on
the membership of objects within each branch and not their positions, evaluation of the
\( n - 1 \) permutations obtained via node translation suffices to find the optimal arrangement.

Suppose that each leaf object has an associated weight. Then, the left and right branches
could be compared via their average weight placing the lower weight branch on the left at
each stage. The end result is a dendrogram where leaf weights generally increase, combining
the benefits of clustering with those of sorting. This procedure was used by Eisen et al.
(1998), with weights given by an external variable, while Gale, Halperin, and Costanzo
(1984), Tien et al. (2008), and Chae and Chen (2011) used positions in an externally
produced ordering as weights. Scores from the first principal component are in many cases
a reasonable choice of leaf weight. If the object weight is some measure of importance
or interestingness, then this method of seriation produces dendrogram-based importance
sorted arrangements.
For a set of \( n \) objects with weights \( v_i \), for \( i = 1, 2, \ldots, n \), a leaf sort cost function is defined as

\[
\text{LS}(\pi) = - \sum_{i=1}^{n} i v_{\pi(i)}. \tag{5}
\]

It is straightforward to show that translating nodes to reduce LS is the same as placing the branch with the lower average weight on the left. This corresponds to one iteration of DendSer with node operator \( T_0 \). Furthermore, the permutation \( \pi \in \Pi(R; \Delta) \) minimizing LS also maximizes the Pearson correlation between the leaf weights and their positions in the permutation. Other correlation measures would similarly produce leaf sorted dendrograms; indeed Degerman (1982) and later Gale, Halperin, and Costanzo (1984) used Kendall’s \( \tau \).

Note that the \texttt{reorder.dendrogram} function in the R package \texttt{stats} (R Core Team 2013) also offers a version of dendrogram leaf sorting. One might expect that \texttt{reorder.dendrogram} with \texttt{agglo.fun=mean} would give the same result as the leaf sorting procedure described above. However, it calculates a node weight by averaging the weight of its left and right branches, rather than weighting the average based on the number of leaves per branch.

### 2.4 Other Dendrogram Seriation Algorithms

Many previously proposed dendrogram seriation algorithms (Gruvaeus and Wainer 1972; Degerman 1982; Gale, Halperin, and Costanzo 1984; Eisen et al. 1998; Alon et al. 1999; Wishart 1999; Tien et al. 2008; Wu, Tien, and Chen 2010; Chae and Chen 2011) fit in to the DendSer framework, with particular fixed choices for the options listed above. For example, the GW algorithm Gruvaeus and Wainer (1972) uses node operator \( R_1 \) as illustrated in Figure 2 and a cost function \( \mathcal{F} \) that compares permutations local to the current node only. Only one iteration is required as a second iteration will leave \( \pi \) unchanged.

The algorithms of Morris, Asnake, and Yen (2003) and Forina et al. (2007) vary slightly from the DendSer framework. Morris, Asnake, and Yen (2003) sought anti-Robinson structure using a simulated annealing-based algorithm that visits nodes in random order. The DSA algorithm of Forina et al. (2007) looks for permutations with short path length and visits groups of (nonnested) nodes simultaneously. The \texttt{symOrder} and \texttt{HC-SYM} algorithms of Chae and Chen (2011) are recent additions to the catalog of dendrogram seriation algorithms. The first step of \texttt{symOrder} produces an ordering of the objects in the left and right subtrees of a node \( N_i \) using a measure called the bilateral symmetric distance. The order \( \pi_1 \) produced is not necessarily consistent with the dendrogram, that is, \( \pi_1 \notin \Pi(N_i; \Delta) \). The second step of \texttt{symOrder} produces \( \pi_2 \in \Pi(N_i; \Delta) \) using dendrogram leaf sorting where weights are leaf positions in \( \pi_1 \). The \texttt{HC-SYM} algorithm applies \texttt{symOrder} at a selection of nodes, in decreasing order of height.

### 3. Exploring Dendrogram Seriation

#### 3.1 DendSer Choices

Here, we investigate which of the DendSer settings, that is, node operators \( \Theta \), coupled with an up or down node visit order produces the best result for a given dendrogram \( \Delta \) and
cost function $F$. We also investigate whether an initial GW step helps to reduce the cost function.

To investigate performance on a range of datasets, we decided to use datasets from the HSAUR2 package (Everitt and Hothorn 2013), selecting all datasets with two or more numeric variables and 30 or more cases. This left us with 18 datasets, ranging in size from about 30 to 2000 cases; for the two datasets with more than 600 cases, we used a random selection of 600 cases. For each dataset, seriation weights are Euclidean distances between cases based on standardized variables, and dendrograms are calculated based on average, single, complete and Ward’s linkage. Using the PL cost function, each dendrogram was seriated using each of seven node operators and each of two visit orders. The process was repeated starting with an initial GW step in each case. For each DendSer run, we recorded the ratio of PL costs for the seriated relative to the initial dendrogram, and then averaged these ratios across the datasets and linkages. These steps were repeated for the cost functions LPL, ARc, and BAR.

Figure 4 shows the results for cost functions PL and ARc. The results for cost functions LPL and BAR (see Figure A of the supplementary materials) are similar to those for PL. For each cost function, node operators are ordered from best to worst and $R_{01}$ is the best performer in each case. $C_0$ also performs strongly, giving cost scores slightly below those of $R_{01}$ but with much less effort, checking at most two new permutations per node visit instead of up to seven for $R_{01}$. The conventional choice of node operator, $T_0$ performs well with ARc only, where it gives cost scores slightly below those of $R_{01}$ but more efficiently, checking only one new permutation per node visit. Generally, an initial GW step and visiting notes from the root downward is beneficial for cost reduction and also reduces the number of iterations required (see Figure B of the supplementary materials). Based on these findings, $R_{01}$, an initial GW step and visiting nodes in the “down” direction are the preferred choices for all four cost functions. We also verified that this setting gives the best, or close to the best result across most datasets and linkages (Figures C–F of the supplementary materials).

Convergence of DendSer is typically two or three iterations for the recommended settings. However, raw times are more relevant as visualization applications require a seriation

![Figure 4](https://via.placeholder.com/150)

**Figure 4.** Evaluating DendSer settings. Shown are cost reductions averaged over 72 dendrograms (of HSAUR2 datasets) seriated with cost functions PL and ARc in panels (a) and (b) and 7 node operators (ordinate). Symbols △ and ▽ denote up and down node visit orders, respectively. Filled symbols result from seriations with an initial GW step.
Table 1. Average DendSer timings in seconds for 600 objects

| Settings       | PL  | LPL | ARc | BAR |
|----------------|-----|-----|-----|-----|
| Optimal: $R_{01}$-GW-down | 0.42 | 0.48 | 7.8  | 3.6  |
| Faster: $C_0$-GW-down | 0.38 | 0.38 | –    | 2.2  |
| Faster: $T_0$-down | –   | –   | 1.9  | –    |

algorithm that gives practically instantaneous results. Table 1 gives average times for seriating dendrograms of 600 objects on a 2.66 GHz MacBook.

The first row gives times for the optimal $R_{01}$-GW-down setting, the next two rows gives times from more efficient settings, selected from the results of Figure 4. The substantially faster times for DendSer with costs PL and LPL is expected, as for a given permutation, these costs as calculated in $O(n)$ operations as compared to $O(n^2)$ for ARc and BAR. In a compromise between optimality and speed, our current version of DendSer defaults to faster settings for ARc/BAR when $n$ is 300 or above, and for PL/LPL when $n$ is 1000 or above.

### 3.2 COMPARISON WITH OTHER DENDROGRAM SERIATION ALGORITHMS

In this section, we compare cost scores from dendrograms seriated with DendSer with those from other dendrogram seriation algorithms. As in Section 3.1, comparisons are based on 72 dendrograms of 18 datasets from the HSAUR2 package (Everitt and Hothorn 2013) and for DendSer we use the optimal settings discussed in Section 3.1. DendSer in conjunction with our new LPL cost function was omitted from this comparison, as it has no obvious competitor algorithm.

#### 3.2.1 Minimizing Path Length

For minimizing path length we compare DendSer with cost function PL with the result of a GW-step (available in package gclus, Hurley 2012) and with the optimal leaf ordering algorithm (OLO) of Bar-Joseph, Gifford, and Jaakkola (2001) (available in package seriation Hahsler, Hornik, and Buchta 2008; Hahsler, Buchta, and Hornik 2013). We also include as “LS-MDS” the results from leaf sorted dendrograms, where leaf weights are coordinates on the first axis of classical multidimensional scaling carried out on the seriations weight matrix $W$. (As the seriation weights here are Euclidean distances, the MDS coordinates coincide with scores from the first principal component.) For comparison purposes, we calculate path lengths of orderings obtained from the traveling salesperson farthest insertion heuristic, as provided in package TSP (Hahsler and Hornik 2007). (The conventional TSP problem seeks optimal closed paths. Algorithms are easily adjusted to find open paths by adding a dummy node equidistant to all other nodes, finding the optimal closed path, and then cutting the path by removing the dummy node.)

In Figure 5(a), plotted points represent the ratio of PL scores from the associated algorithm to the score from the initial dendrogram ordering, averaged across the datasets. OLO on average reduces the PL cost score by about 73% of that of the original dendrogram order. The reduction obtained by DendSer-PL is slightly higher. The (nondendrogram) farthest insertion TSP method reduces PL cost by 72%, though other insertion methods do not beat OLO.
In Figure 5(b), we compare the PL scores of OLO seriated dendrograms by linkage relative to PL scores of paths minimizing the TSP heuristic. Ward’s linkage is the clear winner of the four linkages and even produces lower PL scores than TSP. The poor result for single linkage is surprising as it merges clusters based on nearest neighbors. These findings are consistent across most of the 18 datasets.

TSP with farthest insertion is also comparatively slow, taking four seconds to seriate 600 objects on a 2.66 GHz MacBook, as compared to fractions of seconds for OLO and DendSer-PL. DendSer-PL produces suboptimal solutions but runs in $O(n^2)$ steps as compared to $O(n^3)$ for OLO. In our experiments, the speed advantage of DendSer-PL does not become apparent until $n$ is 1000 or more (due to the efficient C implementation of OLO). For example, it seriates $n = 2000$ objects in 2.5 sec as compared to about 6 sec for OLO.

From the experiments in this subsection, if the main goal is seriation for low path length, then, of the methods considered, OLO (or DendSer-PL for larger $n$) on the Ward linkage dendrogram is the method of choice.

3.2.2 Finding Anti-Robinson Form. With the goal of finding anti-Robinson form, we compare DendSer with cost ARc to OLO, LS-MDS, and GW. Non-dendrogram based competitors include the ARSA algorithm (Brusco, Köhn, and Stahl 2008) which uses simulated annealing to find a seriation minimizing ARc, and the R2E elliptical seriation algorithm devised by Chen (2002) for the purpose of finding AR structure. The R2E algorithm iteratively calculates correlations of the seriation weight matrix $W$ until the correlation matrix has rank 2.

In Figure 6(a), plotted points represent the ratio of ARc from the associated algorithm to the score from the initial dendrogram ordering, averaged across the datasets. ARSA gives the lowest ARc scores and DendSer-ARc is best of the dendrogram based algorithms. R2E and LS-MDS also produce good reductions in ARc. Figure 6(b) shows the results for BAR cost where DendSer-BAR gives the best result followed by R2E.

Figure 6(c) compares ARc scores by linkage with ARSA. Of the linkage seriations, average linkage has the lowest average score, though there is a lot of variability across datasets. In any case, ARSA beats each of the linkage seriations. The comparison by
linkage is not shown for BAR but again, average linkage dendrograms have the lowest average score.

Comparing timings, both ARSA and R2E are slower than DendSer, substantially so in the case of ARSA. Seriating \( n = 600 \) objects takes ARSA about 3 min compared with seconds for DendSer (see Table 1). We conclude that ARSA is not a practical method for visualization applications unless \( n \) is small. Neither is DendSer with either of ARc or BAR suitable for large \( n \), as in our experiments they take about 3 min to seriate \( n = 2000 \) objects. LS-MDS would seem to provide a much faster alternative to DendSer-ARc in this case.

### 3.3 Comparison via Toy Examples

Using some simple artificial datasets, we demonstrate how the choice of cost function affects the dendrogram seriation results. We also investigate the performance of some nondendrogram seriation algorithms on these datasets.

We randomly generated four artificial datasets each with \( n = 100 \) observations on two variables, to have linear, curved, circular, and clustered patterns. The datasets along with seriations from OLO (minimizing cost function PL), and those from DendSer using BAR, ARc and LS-MDS are shown in Figure 7. In each case, we started with the Euclidean distance average linkage dendrogram and used DendSer settings recommended in Section 3.1.

A good seriation result would place similar objects nearby in the sequence and dissimilar objects far apart. In the scatterplots of Figure 7, good seriations appear as paths that do not take unnecessary detours. Also, the heatmaps should have a smooth appearance ideally with colors moving through a dark to light color gradient as distance from the diagonal increases, that is, the typical anti-Robinson pattern. For the linear data, all four seriation algorithms produce a path that goes from one end of the data to the other, but the OLO path has many small wiggles and results in a less smooth heatmap than the other three. For the curved data pattern, the path produced by LS-MDS takes a few obvious detours which appears as a checkerboard pattern in the heatmap. Here, the efforts of LS-MDS to produce a linear path (ordering the points along the first principal component subject to the restrictions of the dendrogram) are clearly apparent. Neither of the cost functions LS-MDS or ARc find the circular path. Seriation with ARc looks for an anti-Robinson pattern in the
Figure 7. Dendrogram seriation. Results from four methods (in columns) for four synthetic datasets, shown both as a scatterplot and heatmap. For heatmaps, dark to light colors represent low to high Euclidean distances.
distance matrix, but data on a circle do not have this pattern. Seriation with OLO and BAR both find the circular path, and the associated heatmaps are quite smooth in appearance. The dark color regions in the upper right and lower left heatmap corners are indicative of circular structure, with the remainder of the heatmap having the ideal anti-Robinson pattern. In our last example, the data consist of four clusters. OLO and BAR produce paths that move through the clusters in a circular pattern, while ARc and LS-MDS visit the clusters in a zig-zag pattern. The circular paths avoid the large diagonal jump between two of the clusters and produce visibly smooth heatmaps with an anti-Robinson pattern except in the lower left and top right heatmap corners.

For these examples, GW paths (not shown) are broadly similar to those of OLO, though less smooth. The cost functions PL, ARc, and BAR differ in the amount of smoothness they try to impose on the heatmap. PL evaluates a seriation by summing the entries along the main diagonal, BAR uses a few diagonals, while ARc uses all entries. LS-MDS also uses all entries along with scores from the first principal component. Thus, the choice between cost functions is a choice between a “local” cost function (PL), an intermediate cost function (BAR), and global cost functions (ARc and LS-MDS). As can be seen from Example 7, LS-MDS does not give good results when data have a curved pattern. (Of course, seriation in order of scores from the first principal component without the dendrogram restriction would give far worse results.) The global cost functions do not work well for data with a circular pattern. Overall, the BAR cost function offers a good compromise between local and global smoothness. For this reason, it is our preferred cost function for improving heatmap visualizations.

It is interesting to compare the performance of some nondendrogram seriation algorithms on these datasets. Figure 8 shows the paths obtained using the TSP farthest insertion heuristic, ARSA and R2E. (Heatmaps are shown in Figure G of supplementary materials). Of the three methods, R2E produces satisfactory results for all four datasets. TSP “fails” on the linear data, producing a path going from southwest to northeast and back again. ARSA does poorly on the circular data. Though it and DendSer-ARc seek to minimize the same cost function, the fact that DendSer respects the hierarchical clustering structure prevents it from producing badly scrambled paths.

Wilkinson (2005) also conducted a comparison of seriation algorithms. They subjected various artificial datasets to random permutations and evaluated each algorithm on how well it recovered the original order. In their study, seriation via hierarchical clustering using the simple Gruvaeus and Wainer (1972) method gave good results. Overall, they recommended seriation using coordinates from the first principal components dimension. However, in the curve, circle, and cluster patterns of Figure 7, sorting along the first principal components dimension would not give good results.

4. APPLICATIONS

In this section, we explore three datasets using heatmaps, glyph arrays, parallel coordinate displays, and scatterplot matrices, where visualizations benefit from seriation with DendSer.
4.1 POTTERY DATA

In our first example, we use seriation methods to explore the pottery data from Tubb, Parker, and Nickless (1980). The data consist of nine chemical measurements of 45 pieces of pottery from five kiln sites in Britain. The first step in analyzing these data is to see if the pottery pieces cluster into distinct groups based on their chemical composition and so we performed a hierarchical clustering using scaled Euclidean distance and average linkage. Figure 9(a) shows a heatmap of the distances between the pottery pieces, ordered using the BAR-serrated dendrogram. (We use cost function BAR here because of our findings of Section 3.3.)

From a visual assessment, the pottery samples appear to fall into three clusters. A heatmap using the order from the initial, unseriated dendrogram (not shown) has a more blotchy appearance which complicates the assessment of cluster structure. In Figure 9(b), we assess the clusters using a glyph array, again ordered using the BAR-serrated dendrogram. Glyph color shows cluster membership. Clusters relate closely to the kiln sites: the light green cluster is comprised of samples from the two sites in Wales, the blue cluster consists of the samples from Gloucester, and the dark red cluster contains the pottery pieces from the two New Forest locations. Glyphs from each of the three clusters have a distinctive
shape, but it is not easy to summarize the cluster differences based on Figure 9(b). For this we turn to parallel coordinate plots (PCPs).

Figure 10 shows PCPs of the pottery data. Case color indicates cluster membership as in Figure 9(b) and variables are in arbitrary order, also as in the glyphs of Figure 9(b). In Figure 10(a), panels show poor separation of the three clusters, and so in (b) we reordered the variables to assist in summarizing the cluster differences. For this we use a simple merit measure, which rewards PCP panels showing good separation between clusters.

Figure 9. Pot clustering. Pot order from BAR-sorted dendrogram. In (a) dark to light colors represent low to high Euclidean distances. In (b) glyph color signifies cluster membership.

Figure 10. The variables in the PCP in (a) are ordered arbitrarily. The variables in the PCP in (b) are ordered using a DendSer-LPL seriation. Line color signifies cluster membership.
Let $c_{ij}^r$ denote the centroid of cluster $r$ for variables $i$ and $j$. A PCP panel representing variables $(i, j)$ has good cluster separation if the distance between centroids as measured by

$$m_{ij} = \sum_{1 \leq r,s \leq k} ||c_{ij}^r - c_{ij}^s||$$

is large, where $|| \cdot ||$ denotes Euclidean distance and $k$ is the number of clusters. Here, $m_{ij}$ measures the importance of the $(i, j)$th panel. Our dendrogram seriation strategy requires seriation weights $w_{ij}$ where importance decreases with weight size, thus we use $w_{ij} = \max_{i,j} \{m_{ij}\} - m_{ij}$. An average linkage dendrogram based on $w_{ij}$ is rearranged with the LPL cost function of Equation (2). Note that in this setting, the seriation weights measure similarity rather than dissimilarity between the cluster centroids. While this may seem counterintuitive, we are simply using dendrogram seriation to find a permutation minimizing a cost function, in this instance LPL. The only requirement for the cost function is that $w_{ij} = w_{ji}$.

The LPL cost function in this context should pick out low-weight (and therefore important) panels while also arranging the panels in order of increasing weight (i.e., decreasing importance). Given that people generally read from left to right, it follows that people are also likely to examine a PCP from left to right (or top to bottom, depending on the orientation of the PCP). Placing the most important panels at the beginning of the PCP allows the analyst to immediately see features of interest in the data.

The PCP in Figure 10(b) emphasizes cluster separation. Panels showing high separation of the regions (clusters) appear at the beginning and panels showing less separation are at the end, with $m_{ij}$ scores (Equation (6)) about constant for the first three panels, and decreasing monotonically thereafter. Also, the first variable in Figure 10(b) shows clear separation between the three clusters, while the next four variables separate two of the three clusters only. It is evident that pots from Hampshire (shown in dark red) contain the lowest levels of MgO, Fe$_2$O$_3$, K$_2$O, MnO, and CaO. Pots from Wales (light green) contain the highest levels of MgO, K$_2$O and MnO, and the lowest levels of Al$_2$O$_3$, while Gloucester pots (blue) have the highest levels of Fe$_2$O$_3$ and CaO, and medium levels of MgO, K$_2$O, and MnO. The arbitrarily ordered PCP in Figure 10(a) requires much closer inspection to extract this information.

### 4.2 Fibroblast Data

Iyer et al. (1999) conducted an experiment on human fibroblast cells to determine their importance in wound repair. The resulting dataset consists of 517 genes whose expression level was measured 12 times in a 24-hr time period, following the reintroduction of a serum. Following Eisen et al. (1998), each gene is normalized relative to time 0 and transformed to a log scale. A hierarchical clustering of the genes is then used to find groups of genes that behave similarly. Dissimilarity between gene pairs is measured by 1–Pearson correlation and Ward’s linkage is used.

Figure 11 shows heatmaps of the correlation matrix, with genes ordered using two different dendrogram seriation schemes. The first heatmap (a) uses an ordering given by the procedure suggested by Eisen et al. (1998), that is, leaf sorting with total gene expression as
weights. The second heatmap Figure 11(b) uses the permutation of genes given by DendSer with BAR cost, and this results in a far smoother heatmap. The small dark regions in the top right and bottom left corners hint at a circular structure. This pattern was not visible using PL or ARc as cost functions.

Figure 12 shows heatmaps of the gene expression data, where the genes (i.e., rows) are ordered with DendSer-BAR, as in Figure 11(b). A visual inspection of the dendrogram suggests 10 clusters; the resulting cluster membership is shown in the legend on the left. From the cluster mean expressions shown in Figure 12(b), there appears to be a smooth transition between the clusters. This smooth transition made it easier to observe global relationships between the clusters. We note that the gene ordering of Figure 11(a) places the first five clusters in order A,B,D,C,E and the second five in order F, J, I, G, and H, which does not lead to a smooth transition between the clusters.

4.3 Sleep Data

Based on Bertin’s (1983) concept of “diagonalization,” Hurley (2004) proposed seriating variables in scatterplot matrices so that interesting panels were placed close to the main diagonal. The logic behind this is that interesting panels are placed in a prominent position, making it easier for the analyst to observe features of interest in the data. Seriation with LPL cost makes interesting panels even more prominent by positioning them close to the main diagonal and in addition, positioning the most interesting panels in the top-left of the scatterplot matrix, thus drawing them to the viewer’s attention.

Outliers are one of the first features to look for when analyzing data and scatterplot matrices are a convenient visualization tool for revealing bivariate outliers in data. Figure 13 shows a scatterplot matrix of the Sleep data (Allison and Cicchetti 1976), which contains 10 measurements on 62 mammals. The outlying scagnostic index (Wilkinson, Anand, and Grossman 2005) was used to award each scatterplot a value \( m_{ij} \in [0, 1] \) measuring the degree of outlyingness present. In Figure 13, the top 20% of panels have a dark background, the next 30% have a mid color background, which the bottom 50% have a light background.
Figure 12. Fibroblast data with genes are ordered as in Figure 11(b). The legend on the left of (a) shows cluster membership.

Also, the order of variables in the scatterplot matrix was chosen to give prominence to panels with a high degree of outlyingness, placing them in the top-left corner and/or close to the main diagonal. Specifically, an average-linkage dendrogram with seriation weights $w_{ij} = 1 - m_{ij}$ was rearranged with DendSer-LPL.

In Figure 13, panels with the most extreme outliers appear in the top-left of the scatterplot matrix, making it easier to see that the variables brain weight (Br) and body weight (Bw) contain two extreme outliers (Asian and African elephant) and the variable life (L) contains one, less extreme, outlier (Human). These outliers are not as clearly observed from the arbitrarily ordered scatterplot matrix. For these data, dendrogram seriation with cost functions PL, ARc, and BAR or the GW-method used by Hurley (2004) for seriating
scatterplot matrices, all produce similar results, placing panels with a high degree of outlyingness close to the main diagonal, but in the middle of the display.

5. DISCUSSION

Visualizations of statistical data benefit from systematic ordering of data objects to highlight features and structure. In this article, we focused on seriation via dendrograms and described DendSer, a general-purpose dendrogram seriation algorithm. We also showed in Sections 3.2 and 3.3 that DendSer compares favorably to other seriation algorithms.

We introduced the concept of node operator to describe the permutations evaluated by DendSer at each node. Typically, other dendrogram seriation algorithms use node translation $T_0$, node reflection $R_0$, or the subnode rotation $R_1$ employed by Gruvaeus and Wainer (1972). In Section 3.1, we showed that node translation $T_0$ gives good results for ARc cost, but $R_0$ and $R_1$ are not recommended. We also introduced some new node operators two of which, $C_0$ and $R_{01}$ give good results for all four cost functions examined.

DendSer produces seriations appropriate to different visualization settings via the choice of cost function and seriation weights. For example, in Section 4 we demonstrated the use of seriation in producing visualizations highlighting cluster separation and the presence of outliers. We presented two new seriation cost functions, namely LPL and BAR. LPL is useful for importance sorting of objects, which is relevant for parallel coordinate displays and scatterplot matrices. We recommend BAR for seriating heatmap visualizations. It
offers a convenient compromise between cost functions ARc and PL which impose too much smoothness in the case of ARc and too little in the case of PL.

For seriations consistent with groupings produced by nonhierarchical clustering algorithms, a two-stage process first seriating objects within groups, then seriating the groups themselves, may be employed. Hahsler and Hornik (2011) give many examples of this approach.

SUPPLEMENTARY MATERIALS

**DendSer package** The methods described in this article are available in package *DendSer* (Hurley and Earle 2013).

**Applications** Code for the comparisons of Section 3.3 and applications in Section 4 is provided in *DendSer* package demos.

**Extra figures** The file extrafigs.pdf contains figures supporting some statements made in the text of this article.

- **Figure A** shows cost reductions for LPL and BAR, as given in Figure 4 for PL and ARc costs.
- **Figure B** shows average number of DendSer iterations, averaged over HSAUR2 dendrograms, for various DendSer settings and cost functions PL, LPL, ARc, and BAR.
- **Figure C** Boxplots show differences in PL cost reduction for the default setting $R_{01}$-GW-down with 27 other settings, by dataset and linkage.
- **Figures D, E, F** As Figure C, but for cost functions LPL, ARc, and BAR.
- **Figure G** shows heatmaps, corresponding to the seriations of Figure 8.

ACKNOWLEDGMENTS

Both authors supported by a Research Frontiers Grant from Science Foundation Ireland (Grant No.: 06/RFP/MAT006).

[Received April 2013. Revised September 2013.]

REFERENCES

Allison, T., and Cicchetti, D. V. (1976), “Sleep in Mammals: Ecological and Constitutional Correlates,” *Science*, 194, 732–734. [20]

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999), “Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays,” *Proceedings of the National Academy of Sciences*, 96, 6745–6750. [3,5,10]

Bar-Joseph, Z., Gifford, D. K., and Jaakkola, T. S. (2001), “Fast Optimal Leaf Ordering for Hierarchical Clustering,” *Bioinformatics*, 17, S22–S29. [3,5,8,12]

Bertin, J. (1983), *Semiology of Graphics: Diagrams, Networks, Maps*, Madison, WI: University of Wisconsin Press. [1,20]

Brusco, M., Köhn, H.-F., and Stahl, S. (2008), “Heuristic Implementation of Dynamic Programming for Matrix Permutation Problems in Combinatorial Data Analysis,” *Psychometrika*, 73, 503–522. [2,13]
Chae, M., and Chen, J. J. (2011), “Reordering Hierarchical Tree Based on Bilateral Symmetric Distance,” *PLoS One*, 6, e22546. [5,9,10]

Chen, C. H. (2002), “Generalized Association Plots: Information Visualization Via Iteratively Generated Correlation Matrices,” *Statistica Sinica*, 12, 7–29. [2,13]

Degerman, R. (1982), “Ordered Binary Trees Constructed Through an Application of Kendall’s Tau,” *Psychometrika*, 47, 523–527. [3,5,10]

Earle, D. (2010), “Dendrogram Seriation in Data Visualisation: Algorithms and Applications,” Dissertation, National University of Ireland Maynooth. Available at [http://eprints.nuim.ie/2442/][3]

Earle, D., and Hurley, C. B. (2011), “A New Flexible Dendrogram Seriation Algorithm for Data Visualisation,” in *Proceedings of ISI*. Available at [http://isi2011.congressplanner.eu/pdfs/650118.pdf][3]

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998), “Cluster Analysis and Display of Genome-Wide Expression Patterns,” *Proceedings of the National Academy of Sciences*, 95, 14863–14868. [2,3,5,9,10,19]

Everitt, B. S., and Hothorn, T. (2013), *HSAUR2: A Handbook of Statistical Analyses Using R* (2nd ed.). Available at [http://CRAN.R-project.org/package=HSAUR2][11,12]

Forina, M., Lanteri, S., Casale, M., and Concepción Cerrato Oliveros, M. (2007), “A New Algorithm for Seriation and its Use in Similarity Dendrograms,” *Chemometrics and Intelligent Laboratory Systems*, 87, 262–274. [5,6,8,10]

Friendly, M., and Kwan, E. (2003), “Effect Ordering for Data Displays,” *Computational Statistics & Data Analysis*, 43, 509–539. [1,2]

Gale, N., Halperin, W., and Costanzo, C. (1984), “Unclassed Matrix Shading and Optimal Ordering in Hierarchical Cluster Analysis,” *Journal of Classification*, 1, 75–92. [3,5,8,9,10]

Gruvaeus, G., and Wainer, H. (1972), “Two Additions to Hierarchical Cluster Analysis,” *British Journal of Mathematical and Statistical Psychology*, 25, 200–206. [3,10,16,22]

Gutin, G., and Punnen, A. (2002), *The Traveling Salesman Problem and Its Variations*, Dordrecht: Kluwer. [8]

Hahsler, M., Buchta, C., and Hornik, K. (2013), “Infrastructure for the Traveling Salesperson Problem,” *Journal of Statistical Software*, 23, 1–21. [12]

——— (2011), “Dissimilarity Plots: A Visual Exploration Tool for Partitional Clustering,” *Journal of Computational and Graphical Statistics*, 10, 335–354. [23]

Hahsler, M., Hornik, K., and Buchta, C. (2008), “Getting Things in Order: An Introduction to the R Package Seriation.” *Journal of Statistical Software*, 25, 1–34. [2,7,12]

Hubert, L. (1974), “Some Applications of Graph Theory and Related Nonmetric Techniques to Problems of Approximate Seriation: The Case of Symmetric Proximity Measures,” *British Journal of Mathematical and Statistical Psychology*, 27, 133–153. [8]

Hubert, L., Arabie, P., and Meulman, J. (2001), *Combinatorial Data Analysis*, Philadelphia, PA: Society for Industrial and Applied Mathematics. [2,4,9]

Hurley, C. B. (2004), “Clustering Visualizations of Multidimensional Data,” *Journal of Computational and Graphical Statistics*, 13, 788–806. [1,2,3,8,20,21]

——— (2012), “gclus: Clustering Graphics,” available at [http://CRAN.R-project.org/package=gclus][12]

Iyer, V. R., Eisen, M. B., Ross, D. T., Schuler, G., Moore, T., Lee, J. C. F., Trent, J. M., Staudt, L. M., Hudson, J., Boguski, M. S., Lashkari, D., Shalon, D., Botstein, D., and Brown, P. O. (1999), “The Transcriptional Program in the Response of Human Fibroblasts to Serum,” *Science*, 283, 83–87. [19]

Kendall, D. G. (1971), “Seriation From Abundance Matrices,” in *Mathematics in the Archaeological and Historical Sciences*, eds. F. Hodson, D. Kendall, and P. Tautu, Edinburgh: Edinburgh University Press, pp. 215–252. [1]
Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., and Shmoys, D. B. (eds.) (1985), The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, New York: Wiley-Interscience. [2]

Liiv, I. (2010), “Seriation and Matrix Reordering Methods: An Historical Overview,” Statistical Analysis and Data Mining, 3, 70–91. [2]

Morris, S. A., Asnake, B., and Yen, G. G. (2003), “Dendrogram Seriation Using Simulated Annealing,” Information Visualization, 2, 95–104. [5,8,10]

Petrie, W. M. (1899), “Sequences in Prehistoric Remains,” Journal of the Anthropological Institute, 29, 295–301. [1]

R Core Team (2013), “R: A Language and Environment for Statistical Computing,” Available at http://www.R-project.org. [2,10]

Robinson, W. S. (1951), “A Method for Chronologically Ordering Archaeological Deposits,” American Antiquity, 16, 293–301. [1,8]

Tien, Y.-J., Lee, Y.-S., Wu, H.-M., and Chen, C.-H. (2008), “Methods for Simultaneously Identifying Coherent Local Clusters With Smooth Global Patterns in Gene Expression Profiles,” BMC Bioinformatics, 9, 155. [5,9,10]

Tubb, A., Parker, A. J., and Nickless, G. (1980), “The Analysis of Romano-British Pottery by Atomic Absorption Spectrophotometry,” Archaeometry, 22, 153–171. [17]

Wilkinson, L. (2005), The Grammar of Graphics (Statistics and Computing), Secaucus, NJ: Springer-Verlag. [16]

Wilkinson, L., Anand, A., and Grossman, R. (2005), “Graph-Theoretic Scagnostics,” in Proceedings of the 2005 IEEE Symposium on Information Visualization, Washington, DC: IEEE Computer Society, INFOVIS ’05, 157–164. [20]

Wilkinson, L., and Friendly, M. (2009), “The History of the Cluster Heat Map,” The American Statistician, 63, 179–184. [2]

Wishart, D. (1999), “ClustanGraphics 3 Interactive Graphics for Cluster Analysis,” in Classification in the Information Age: Studies in Classification, Data Analysis and Knowledge Organization, ed. W. Gaul and H. Locarek-Junge, Berlin: Springer, pp. 268–275. [2,3,4,5,8,10]

Wu, H.-M., Tien, Y.-J., and Chen, C.-H. (2010), “GAP: A Graphical Environment for Matrix Visualization and Cluster Analysis,” Computational Statistics & Data Analysis, 54, 767–778. [5,10]