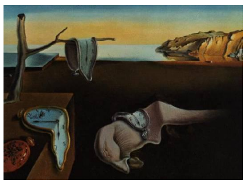

Journal of Statistical Software

June 2022, Volume 103, Issue 7. *doi: 10.18637/jss.v103.i07*

Hierarchical Clustering with Contiguity Constraint in R

Guillaume Guénard
Université de Montréal

Pierre Legendre
Université de Montréal

Abstract

This article presents a new implementation of hierarchical clustering for the R language that allows one to apply spatial or temporal contiguity constraints during the clustering process. The need for contiguity constraint arises, for instance, when one wants to partition a map into different domains of similar physical conditions, identify discontinuities in time series, group regional administrative units with respect to their performance, and so on. To increase computation efficiency, we programmed the core functions in plain C. The result is a new R function, `constr.hclust`, which is distributed in package **adespatial**.

The program implements the general agglomerative hierarchical clustering algorithm described by Lance and Williams (1966; 1967), with the particularity of allowing only clusters that are contiguous in geographic space or along time to fuse at any given step. Contiguity can be defined with respect to space or time. Information about spatial contiguity is provided by a connection network among sites, with edges describing the links between connected sites. Clustering with a temporal contiguity constraint is also known as chronological clustering. Information on temporal contiguity can be implicitly provided as the rank positions of observations in the time series. The implementation was mirrored on that found in the hierarchical clustering function `hclust` of the standard R package **stats** (R Core Team 2022). We transcribed that function from Fortran to C and added the functionality to apply constraints when running the function.

The implementation is efficient. It is limited mainly by input/output access as massive amounts of memory are potentially needed to store copies of the dissimilarity matrix and update its elements when analyzing large problems. We provided R computer code for plotting results for numbers of clusters.

Keywords: R, `hclust`, constrained clustering, space, chronological clustering, Lance and Williams algorithm.

1. Introduction: Constrained clustering in R

Hierarchical cluster analysis consists in assigning elements to clusters as a function of their similarity (or equivalently, their dissimilarity; from here, we will only mention the latter; Legendre and Legendre 2012). Simply put, its purpose is to assemble objects described by variables into the groups where they most likely belong, or to highlight disparities among them by estimating breakpoints where the differences among objects are the strongest.

Situations arise where application of the dissimilarity criterion alone does not lead to a satisfactory clustering solution (reviewed by Murtagh 1985). In many studies, there sometimes are well grounded reasons to force the clusters to be composed of contiguous sites (Legendre and Legendre 2012). For instance, one may wish to relate the results of clustering to causal factors that are geographically-located and known to be spatially autocorrelated (e.g., geological variables). One may also want to delineate ecological regions, administrative units, or resource distribution networks. Another goal may be to cluster sites based on environmental conditions, constrained by spatial contiguity, for the purpose of designing a stratified sampling campaign. Also, scientists may be interested in testing the hypothesis that sites form patches for the variables under study; this can be done by comparing unconstrained and constrained clustering solutions using the Rand (1971) index. Other examples where clustering result comparisons may be helpful for testing hypotheses are given by De Soete, Carroll, and DeSarbo (1987) in fields such as molecular evolution, psycholinguistics, cognitive psychology, and the evolution of languages. When observations form a time series, one may want to segment it with respect to the dissimilarities among the observations while taking their positions along the series into account in order to identify breakpoints representing important changes along the series. Finally, one may want to partition a set of geographic sites on a map into local clusters whose members are also contiguous in space, or the pixels of a geographic raster into groups of various shapes made of pixels carrying similar information. When undertaking such goals with regular, unconstrained clustering algorithms, clusters may end up being spatially disjoint because of irregularities in the spatial distributions of data values. Also, geographical singletons may become included in clusters that are geographically remote. This may be inappropriate when the objective of the analysis is to delineate geographically compact areas for management or administrative purposes. Applying a constraint of contiguity to the clustering procedure helps in solving these problems.

Constrained hierarchical clustering produces solutions that are generally less variable than those of their unconstrained counterparts. By reducing the space of possible solutions, it forces different clustering strategies to yield compact groups of largely similar elements (Legendre, Dallot, and Legendre 1985). In research involving field data, observations are often made during a short time at several points on a surface. Observations repeated in time can also be made at a single site or on a single object; they produce a time series. In such circumstances, grouping observations that are also found in the same geographic area, or in the same time interval, provided that they are also similar enough to the other observations in the cluster, often produces more interesting results than not paying respect to the temporal or spatial relationships among the elements selected by the survey design.

In the present paper, we will focus specifically on hierarchical agglomerative clustering. Here, we will describe software developed to perform hierarchical agglomerative clustering on the basis of a dissimilarity matrix while applying contiguity constraints defined by a list of edges linking pairs of points (sometimes called "links" in layman's term). The software is written in plain C with an R interface and a `plot` method to display results. It is available from package **adespatial** (Dray, Bauman, Blanchet, Borcard, Clappe, Guénard, Jombart, Larocque, Legendre, Madi, and Wagner 2022), which is available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=adespatial`. It works along with existing software available in R to manipulate spatial and temporal data, estimate contiguity, and process clusters, thereby providing users with a tool chain to carry out all aspects of the analysis.

2. Models and software

Hierarchical agglomerative clustering consists in iteratively merging pairs of elements (from singletons onward to clusters involving more and more observations) that are, at any given point of the clustering process, the least dissimilar, until all elements are found in a single, large cluster. Whereas least dissimilarity is a straightforward criterion for choosing which elements are next to be merged, there are multiple ways of recalculating the dissimilarities of the newly formed clusters with respect to the other elements that remain to be clustered. Many approaches have been proposed, each carrying its own assumptions, calculation time, and allowing for adaptations to analyses devoted to answer particular types of questions.

A general theory of hierarchical clustering was developed by Lance and Williams (1966, 1967). These authors showed that all potential combinatorial strategies can be obtained by using a single linear equation to recalculate dissimilarities following the merging of two elements. The software described in the present paper extends Lance and Williams general approach to hierarchical clustering with spatial or temporal contiguity constraint. The algorithm with constraint implemented in our R function was described conceptually by Legendre and Legendre (2012), Section 13.3.2; it is illustrated in Figure 1. We began the development of our constrained clustering function by transcribing function `hclust` of the **stats** R package from its original implementation into a new function in plain C. The `hclust` function uses Fortran code originally contributed to `STATLIB` by F. Murtagh (Department of Computer Science, University of Huddersfield). We also integrated more recent code enhancement featured in R package **flashClust** by Langfelder and Horvath (2012), which consists in caching each observation's nearest neighbor indices and associated dissimilarities and updating them only when necessary. That approach notably enhanced computation speed during unconstrained clustering. We then programmed a new version of that function to apply constraints to the clustering process and wrapped the resulting C functions into R computer code. We used a dissimilarity caching approach similar to that proposed by Langfelder and Horvath (2012) to enhance computation speed for the constrained clustering case.

2.1. The Lance and Williams algorithm

The general theory of hierarchical agglomerative clustering developed by Lance and Williams allows one to implement all combinatorial strategies using a single four-parameter linear equation to calculate $d_{k,h}$, the dissimilarity between any given element k and a newly formed cluster h obtained after merging clusters i and j as follows:

$$d_{k,h} \quad = \quad \alpha_i d_{i,k} + \alpha_j d_{j,k} + \beta d_{i,j} + \gamma |d_{i,k} - d_{j,k}|, \tag{1}$$

where $d_{i,k}$, $d_{j,k}$, and $d_{i,j}$ are the dissimilarities between elements i and k, j and k, and i and j, respectively, and α_i, α_j, β, and γ are the four parameters of the equation whose values

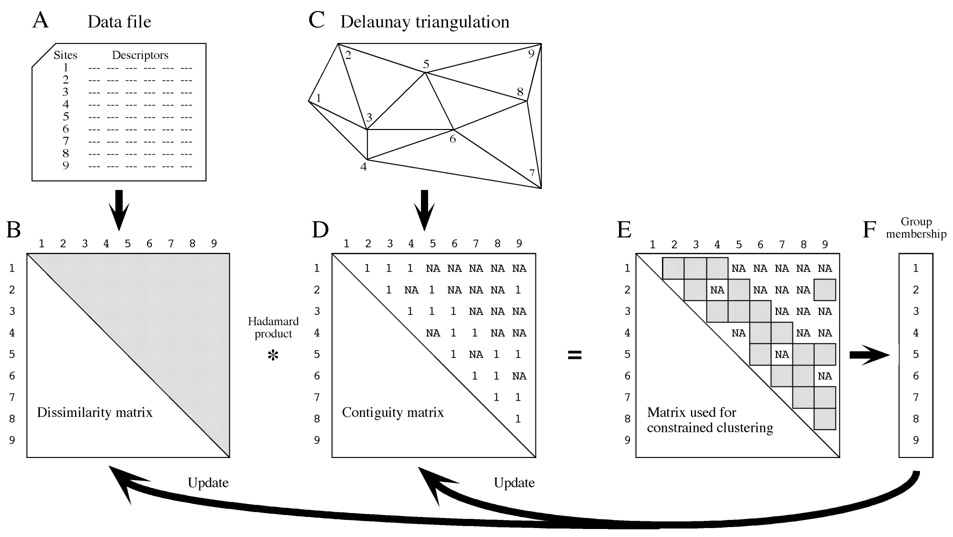

Figure 1: Summary of the spatially-constrained clustering procedure applicable to all combinatorial methods. Dissimilarities in panel B are computed from the data in panel A. Spatial connections (edges in graph C) are translated into a contiguity matrix in **D**, where NAs (NA means "not available" in R and other computer languages) indicate absence of connections. The Hadamard product $\mathbf{B} * \mathbf{C}$ produces NAs in matrix **E**, identifying unconnected pairs of sites, which will not be allowed to cluster. Objects are labelled 1 to 9 in the example. At the start of the clustering process, each object is in a separate group in vector **F**. The grey square showing the lowest dissimilarity, in matrix **E**, determines the next fusion of objects or groups. The group membership vector **F** is then updated by attributing the same identifier to the two objects or groups that have clustered. Matrices **B** and **D** are also updated to account for that fusion. Matrix **E** is recomputed, ready for the next fusion of objects or groups. Modified from Figure 13.25 of Legendre and Legendre (2012).

implement the different clustering strategies. These parameters may be constants or some functions of the number of elements in the clusters (Table 1).

To present the constrained Lance and Williams algorithm, we created a simple example with only six observation sites spread on a map and connected by a seven-edge network (Figure 2). That data set involves a single response variable from which the Euclidean distance matrix among sites was calculated (Table 2).

To implement the constrained hierarchical clustering in an efficient way, we provided the information about contiguity as an edge list instead of a binary connection matrix such as in Figure 1 (panel D). It takes the form of a two-column integer matrix **L**. Each row of **L** represents an edge, with the two values referencing the vertices (singletons or clusters) that are linked. The algorithm operates from list **L** and the dissimilarity matrix (Table 2). It first identifies the edge with the smallest dissimilarity, ignoring any non-contiguous pairs of points. In our example, that edge is `L5`. The vertices at both ends of the edge are merged at the level of dissimilarity observed between them (here 0.9). In a non-constrained hierarchical clustering, the first elements to be merged would have been `1` and `6` as they are the pair with the smallest dissimilarity (0.1) in the full dissimilarity matrix. However, they are the most geographically distant sites and will only merge much later in the constrained clustering

Method	α_i	α_j	β	γ
Single-linkage	1/2	1/2	0	−1/2
Complete-linkage	1/2	1/2	0	1/2
UPGMA	$\frac{n_i}{n_h}$	$\frac{n_j}{n_h}$	0	0
WPGMA	1/2	1/2	0	0
UPGMC	$\frac{n_i}{n_h}$	$\frac{n_j}{n_h}$	$-\frac{n_i n_j}{n_h^2}$	0
WPGMC	1/2	1/2	−1/4	0
Ward's min. variance	$\frac{n_i+n_k}{n_h+n_k}$	$\frac{n_j+n_k}{n_h+n_k}$	$-\frac{n_k}{n_h+n_k}$	0
Flexible	$\frac{1-\beta}{2}$	$\frac{1-\beta}{2}$	$-1 \leq \beta < 1$	0

Table 1: Common hierarchical clustering strategies are obtained by assigning specific values to the four-parameter Lance and Williams model (Equation 1), which is used when calculating the dissimilarity between any given element k and a newly formed cluster h made by merging elements i and j. Here, n_i, n_j, and n_k refer to the number of observations in clustering elements i, j, and k, respectively whereas $n_h = n_i + n_j$.

Distances

	1	**2**	**3**	**4**	**5**
2	1.3				
3	3.6	4.9			
4	1.5	2.8	2.1		
5	0.6	1.9	3.0	0.9	
6	0.1	1.2	3.7	1.6	0.7

Links

	From	To	Dissimilarity
L1	1	2	1.3
L2	1	3	3.6
L3	2	3	4.9
L4	3	4	2.1
L5	4	5	0.9
L6	3	6	3.7
L7	4	6	1.6

Table 2: Dissimilarities between the six elements used as an example and the edges involved in their contiguity relationships.

procedure. The new cluster formed is referenced with the smallest of the two indices (here 4) and all references in $\mathbf{L}$ to the largest of the two indices are changed to the smallest one (Table 3). The updated dissimilarities $d_{k,h}$ with new cluster $h = \{4, 5\}$ are obtained for elements $k \in \{1, 2, 3, 6\}$ as follows (Table 3, Figure 3A, B):

$$\begin{aligned}
d_{1,h} &= \frac{1+1}{2+1} d_{1,4} + \frac{1+1}{2+1} d_{1,5} - \frac{1}{2+1} d_{4,5} + 0|d_{1,4} - d_{1,5}| = 1.10,\\
d_{2,h} &= \frac{1+1}{2+1} d_{2,4} + \frac{1+1}{2+1} d_{2,5} - \frac{1}{2+1} d_{4,5} + 0|d_{2,4} - d_{2,5}| = 2.83,\\
d_{3,h} &= \frac{1+1}{2+1} d_{3,4} + \frac{1+1}{2+1} d_{3,5} - \frac{1}{2+1} d_{4,5} + 0|d_{3,4} - d_{3,5}| = 3.10, \text{ and}\\
d_{6,h} &= \frac{1+1}{2+1} d_{6,4} + \frac{1+1}{2+1} d_{6,5} - \frac{1}{2+1} d_{4,5} + 0|d_{6,4} - d_{6,5}| = 1.23.
\end{aligned}$$

It is noteworthy here that $d_{1,h}$ and $d_{2,h}$ are recalculated, whereas neither vertices 1 or 2 could merge with cluster h at the next step. This calculation is nevertheless mandatory because the updates are incremental, proceeding at any give step on the basis of the updates performed at all previous steps.

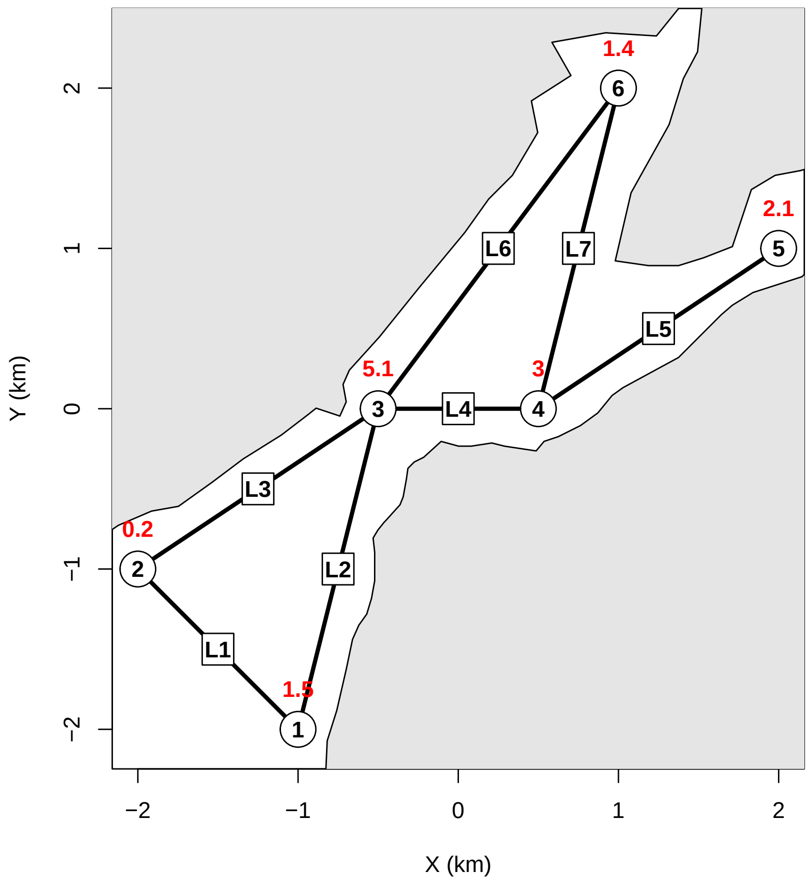

Figure 2: Fictive map designed to illustrate the constrained clustering algorithm. Grey areas represent obstacles defining the limits of the fictional study area where observations were made at six sites (vertices labelled 1 through 6). They are connected by seven edges (labelled L1 through L7) between pairs of neighboring sites. There are no edges 1-4, 1-5, 1-6, and 5-6 because these vertices are separated by obstacles. Edges 2-4, 2-5, and 3-5 were omitted because they would be redundant with combinations of shorter edges between immediate neighbors. A single response variable is used in this example; its values are shown in red atop each observation point. Clustering will proceed using the matrix of Euclidean distances among sites computed from that variable. It will be constrained by the seven edges.

In the next clustering step, vertex **6** merges at dissimilarity of 1.23 (in Table 3) with the cluster formed previously and now referenced with index **4**. The updated dissimilarities $d_{k,h}$ with cluster $h = \{4, 6\}$ are obtained for elements $k \in \{1, 2, 3\}$ as follows (Table 4, Figure 3C):

$$d_{1,h} = \frac{2+1}{3+1}d_{1,4} + \frac{1+1}{3+1}d_{1,6} - \frac{1}{3+1}d_{4,6} + 0|d_{1,4} - d_{1,6}| = 0.57,$$
$$d_{2,h} = \frac{2+1}{3+1}d_{2,4} + \frac{1+1}{3+1}d_{2,6} - \frac{1}{3+1}d_{4,6} + 0|d_{2,4} - d_{2,6}| = 2.42, \text{ and}$$
$$d_{3,h} = \frac{2+1}{3+1}d_{3,4} + \frac{1+1}{3+1}d_{3,6} - \frac{1}{3+1}d_{4,6} + 0|d_{3,4} - d_{3,6}| = 3.87.$$

Distances

	1	**2**	**3**	**4**	**5**
2	1.30				
3	3.60	4.90			
4	1.10	2.83	3.10		
5	—	—	—	—	
6	0.10	1.20	3.70	1.23	—

Links

	From	To	Dissimilarity
L1	1	2	1.30
L2	1	3	3.60
L3	2	3	4.90
L4	3	4	3.10
L5	4	4	—
L6	3	6	3.70
L7	4	6	1.23

Table 3: Updated dissimilarities and edge list after the first two vertices have been merged. The cluster resulting from merging two vertices is referenced using the smallest of the two indices and the data associated with the largest index are discarded. An edge with both ends having the same index indicate that it is internal to a cluster (i.e., the vertices at both of its ends are members of the same cluster) and that it can be disregarded by subsequent agglomerative steps.

Distances

	1	**2**	**3**	**4**	**5**
2	1.30				
3	3.60	4.90			
4	0.57	2.42	3.87		
5	—	—	—	—	
6	—	—	—	—	—

Links

	From	To	Dissimilarity
L1	1	2	1.30
L2	1	3	3.60
L3	2	3	4.90
L4	3	4	3.87
L5	4	4	—
L6	3	4	3.87
L7	4	4	—

Table 4: Updated dissimilarities and edge list after the second step, where a third vertex (6) merged with the cluster formed previously. The resulting cluster is still referenced as 4 and, 6 is discarded.

We note that after updating the edge list, edge L6, which was previously going from **3** to **6**, is now going from **3** to **4** because all references to **6** have been changed to **4**. The third step of the process involves merging vertices **1** and **2** at dissimilarity of 1.30 (in Table 4), forming a new cluster that will be referenced as **1**. The updated dissimilarities $d_{k,h}$ with cluster $h = \{1, 2\}$ are obtained for elements $k \in \{3, 4\}$ as follows (Table 5, Figure 3D):

$$d_{3,h} = \frac{1+1}{2+1}d_{3,1} + \frac{1+1}{2+1}d_{3,2} - \frac{1}{2+1}d_{1,2} + 0|d_{3,1} - d_{3,2}| = 5.23 \text{ and}$$
$$d_{4,h} = \frac{1+3}{2+3}d_{4,1} + \frac{1+3}{2+3}d_{4,2} - \frac{3}{2+3}d_{1,2} + 0|d_{4,1} - d_{4,2}| = 1.61.$$

In the fourth step, clusters 1 and 4 are the least dissimilar but cannot merge because vertex 3 still sits between them. Vertex 3 is less dissimilar to cluster 4 than it is to cluster 1 and thus it merges with the former at dissimilarity of 3.87. The updated dissimilarities $d_{k,h}$ with

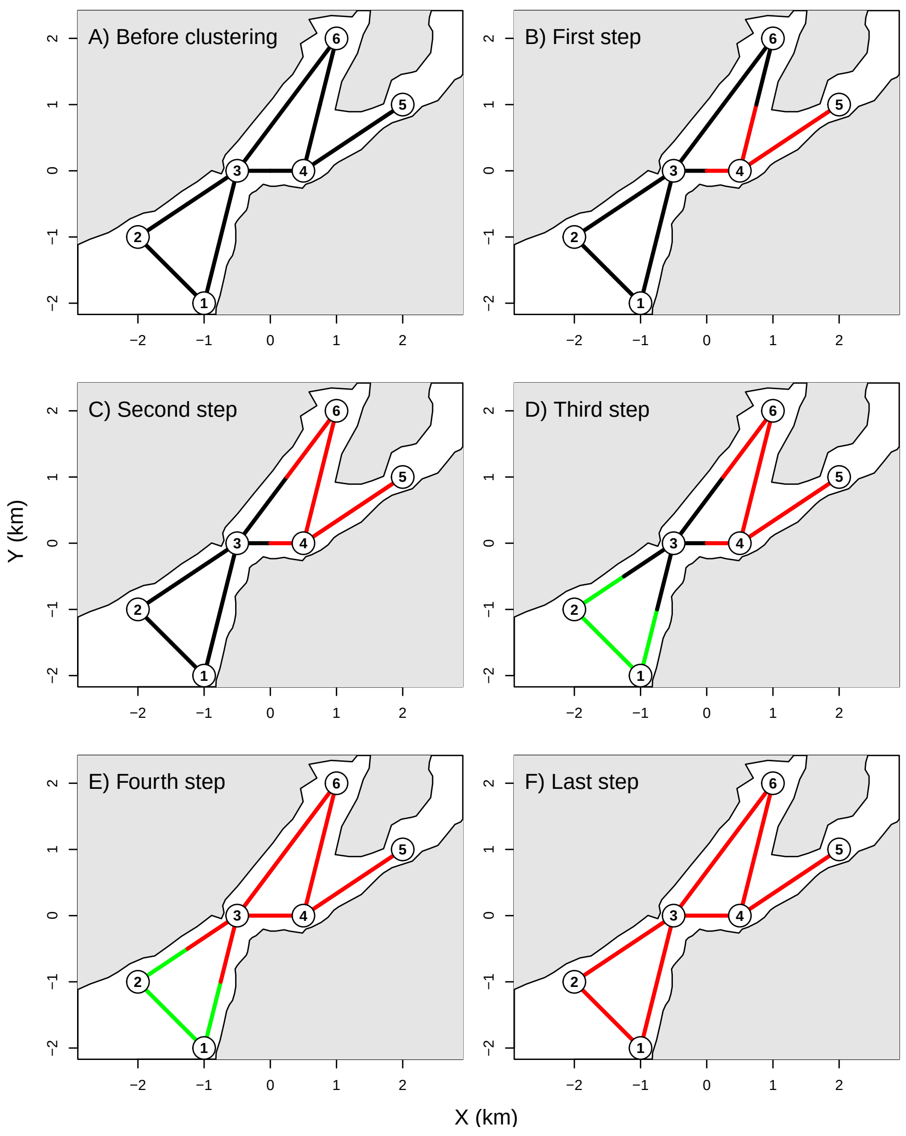

Figure 3: The exemplary scenario contained six vertices at the beginning of the process (panel A). The five agglomerative constrained clustering steps are shown in panels B to F.

cluster $h = \{3, 4\}$ are obtained for element $k \in \{1\}$ as follows (Table 6, Figure 3E):

$$d_{1,h} = \frac{1+2}{4+2}d_{1,3} + \frac{3+2}{4+2}d_{1,4} - \frac{2}{4+2}d_{3,4} + 0|d_{1,3} - d_{1,4}| = 2.67.$$

Finally, the two clusters generated by the constrained clustering procedure will merge at dissimilarity of 2.67, completing the clustering process (Figure 3F). It is noteworthy that this dissimilarity value is smaller than that observed at the previous step. This will produce a level reversal if the resulting tree is displayed as a dendrogram (Figure 4). All sorting strategies except complete-linkage may produce reversals in constrained clustering (Murtagh 1985; Kaufman and Rousseeuw 2005).

Distances

	1	**2**	**3**	**4**	**5**
2	—				
3	5.23	—			
4	1.61	—	3.87		
5	—	—	—	—	
6	—	—	—	—	—

Links

	From	To	Dissimilarity
L1	1	1	—
L2	1	3	5.23
L3	1	3	5.23
L4	3	4	3.87
L5	4	4	—
L6	3	4	3.87
L7	4	4	—

Table 5: Updated dissimilarities and edge list after the third step, where a new cluster is formed by merging 1 and 2.

Distances

	1	**2**	**3**	**4**	**5**
2	—				
3	2.67	—			
4	—	—	—		
5	—	—	—	—	
6	—	—	—	—	—

Links

	From	To	Dissimilarity
L1	1	1	—
L2	1	3	2.67
L3	1	3	2.67
L4	3	3	—
L5	3	3	—
L6	3	3	—
L7	3	3	—

Table 6: Updated dissimilarities and edge list after the fourth step, where 3 merged with 4, forming a new four-element cluster now referenced as 3 (the smallest index).

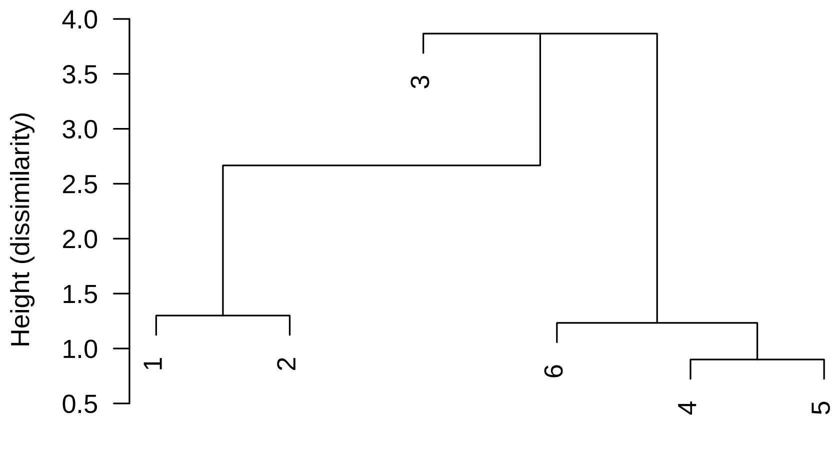

Figure 4: Results of constrained clustering often contain branch reversals due to fact that merging at a given step may occur at a smaller dissimilarity than that of the previous step because of the constraint. A dendrograms is not the most adequate way of displaying constrained clustering results.

The tree resulting from constrained clustering can be cut at any level to obtain partitions with anywhere from two to n clusters, where n is the number of vertices. That process is fast because obtaining the tree, which is the most computationally intensive task, has already

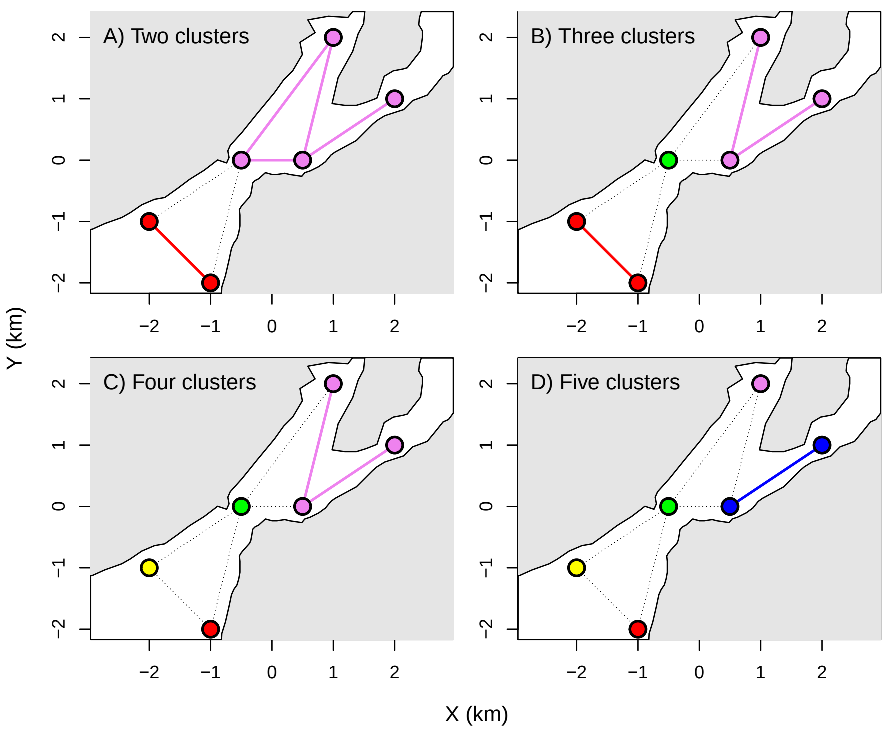

Figure 5: Partitioning of the tree obtained from the spatially constrained clustering of the example into two, three, four, and five groups. Because the number of elements was very small, groups often contained a single observation point. That situation will generally not occur when the number of observations is larger.

been done; obtaining a suite of partitions with different numbers of groups is a light task in comparison. For instance, we found the task of cutting the tree to be ≈ 20 times shorter than performing the clustering when n was a few hundreds; the factor increased to ≈ 100 with n in the few thousands. It is thus quick to represent multiple partitions into different numbers of clusters on a map or along a time series (e.g. Figure 5).

Whereas an unconstrained, hierarchical agglomerative clustering algorithm based on the Lance and Williams approach would iterate, unhampered by contiguity, through each and every pair of points or clusters, to find the pair with the smallest dissimilarity in the original or updated dissimilarity matrix, a constrained algorithm only needs to consider the dissimilarities corresponding to contiguous pairs. This makes a constrained hierarchical agglomerative clustering generally faster procedure than its unconstrained counterpart, provided that the former is implemented in code that takes advantage of the smaller number of dissimilarities that is has to consider to proceed.

The linear equations underlying the different classificatory sorting strategies outlined in Lance and Williams (1966, 1967, Table 1) were implemented in a modular manner in the C computer code, as a set of updating functions that are referenced by addresses using a function pointer. We used that approach to allow one to implement alternative sorting strategies, e.g., by changing parameter values or adding more linear equations, with minimum programming effort. Since the constrained agglomerative clustering program described here also includes features to perform unconstrained hierarchical agglomerative clustering identical to that performed by function `hclust` from the standard R package **stats**, any alternative sorting strategy added to it would also become available to perform regular, unconstrained hierarchical agglomerative clustering without further programming effort; see the C language code and comments for instructions about how to add new updating functions.

3. Illustrations

The constrained hierarchical agglomerative clustering program described in this paper is available in R package **adespatial** (Dray *et al.* 2022). For software developers, a development package specific to the features discussed in the present article and called **constr.hclust** (Legendre and Guénard 2020) is available from GitHub at `https://github.com/guenardg/constr.hclust` or can be downloaded from `https://numericalecology.com/Rcode/`.

Let us begin these illustrations by loading the necessary R packages **adespatial**, **magrittr** (Bache and Wickham 2022), **spdep** (Bivand, Pebesma, and Gomez-Rubio 2013), and **vegan** (Oksanen *et al.* 2022):

```
R> library("adespatial")
R> library("magrittr")
R> library("spdep")
R> library("vegan")
```

Package **adespatial** implements the new constrained hierarchical agglomerative clustering function `constr.hclust`; **magrittr** will provide us with the forward pipe operator `%>%`, which allows us to pass results among chains of function calls and avoid parenthesis embedding; **spdep** will allow us to obtain a neighborhood graph using Delauney triangulation later in the examples, and package **vegan** contains the data set for the first example.

Function `constr.hclust`, which carries on the constrained hierarchical agglomerative clustering methods, has a total of seven arguments, namely:

`d` A dissimilarity matrix of class '`dist`'.

`method` The agglomeration method to be used (Table 1).

`links` A list of edges (or links) connecting the points.

`coords` The coordinates of the observations (data rows) in the dissimilarity matrix `d`, which are used for data plotting purposes.

`beta` The beta parameter for beta-flexible clustering.

`chron` A boolean indicating whether a chronological clustering should be calculated.

`members` `NULL` or a vector with length size of `d`.

As mentioned earlier, function `constr.hclust` was developed by using function `hclust` from R package **stats** as a template. Consequently, its first two arguments correspond to those of that function. The two functions have an argument `d`, which is mandatory and used to pass a dissimilarity matrix and an argument `method` to specify the sorting strategy to be used. Argument `method` can be given the same textual values as for function `hclust`, but has `"ward.D2"` as its default value, whereas `hclust`'s default value for this argument is `"complete"`. The classificatory strategies associated with the different values of argument `method` are listed in Table 1. All other arguments can be omitted. Argument `links` allows the user to pass an edge list, argument `coords` to pass the geographic Cartesian coordinates of the clustered elements, argument `beta` to pass the β parameter used in flexible clustering, and argument `chron` to trigger chronological clustering. Argument `members` has the same function as in `hclust` and allows one to resume clustering computation from an among-cluster dissimilarity matrix with vector `members` giving the number of observations in each of the clusters. The default value, `NULL`, corresponds to the general case where `d` is a dissimilarity matrix among observations.

β-flexible clustering is a form of agglomerative clustering that was described by Lance and Williams in the two papers where they described the general algorithm for agglomerative clustering (Lance and Williams 1966, 1967). This clustering strategy always produces cophenetic matrices that are ultrametric in the case of unconstrained clustering; the corresponding dendrograms are without reversals, which are often found in agglomerative centroid methods. Reversals can be obtained in constrained clustering even with β-flexible clustering.

When argument `links` is omitted and argument `chron` is `FALSE` (the default), function `constr.hclust` performs in an identical way to `hclust` (notwithstanding the different default value for argument `method`), with β-flexible clustering as an additional functionality not found in `hclust`. When `links` is provided, a constrained clustering is performed, whereas when `chron = TRUE`, a chronological clustering, where the constraint is simply the sequence of observations, is calculated (in that case, argument `links`, if not omitted, is not used). Information from argument `coords` is stored into the object returned by `constr.hclust`, ready to be used later for displaying the results on a map from the constrained or chronological clustering.

For constrained hierarchical clustering, it is commonly assumed that the graph described by the `links` is entirely connected (i.e., it involves no disjoint set of vertices). If this assumption is not met, the process described previously cannot merge all clusters into a single entity. If function `constr.hclust` is provided a graph that is not entirely connected, it will merge the disjoint sets two-by-two at the end of the process, while returning a missing value (`NA`) as the dissimilarity at which disconnected clusters have been merged. This behavior was necessary to remain consistent with '`hclust-class`' objects, which expect $n-1$ merging events (where n is the number of observation), while also remaining consistent with the theory of constrained clustering, which stipulates that only connected entities are allowed to merge. Function `constr.hclust` issues a warning to the user upon encountering disconnected clusters.

3.1. First application example: Borcard's Oribatid mite data

As a first illustration, we will use constrained clustering to find the ecological boundaries in the Oribatid mite microfauna data sampled in a peat moss mat. This data set was originally described by Borcard, Legendre, and Drapeau (1992) and Borcard and Legendre (1994). It originates from a faunistic survey performed on the peat mat surrounding Lac Geai, Québec, Canada (WGS84 geodesic coordinates: +45.995424; −73.993691). A total of 70 peat moss cores were sampled in a plot 2.5 m wide along the forest edge by 10 m in length toward the open lake water. Core numbering starts at the edge of the forest and ends at the water edge. For each core, Oribatid mite specimens (Phylum: Arthropoda, Class: Arachnida, Subclass: Acari) were identified into 35 morphospecies (i.e., taxonomic species identification based entirely on morphological differences from related species) and counted.

Let us begin the example by loading the necessary data sets files, from package **vegan**, as follows:

```
R> data("mite", package = "vegan")
R> data("mite.xy", package = "vegan")
```

These files contain:

`mite` The number of individuals from each morphospecies in each core (or site).

`mite.xy` When standing on the edge of the forest and looking towards the lake, the x (short side) and y (long side of the study area) coordinates of the core samples with respect to the lower left corner of the plot.

There are many ways of obtaining the edge list. For the sake of this example, we will use a Delauney triangulation with removal of the edges whose lengths are above a certain threshold, which we will set to 1.5 m. First, a neighbour list is obtained from the coordinates using function `tri2nb`, which is then converted to a spatial weight list using function `nb2listw`. The argument `style = "B"` given to `nb2listw` means that the edges will all be assigned a constant weight of 1. Finally, function `listw2sn` is used to convert the spatial weight list into a list of edges. These operations can be performed into a single line of R code as follows:

```
R> mite.edge <-
+    mite.xy %>%
+    tri2nb %>%
+    nb2listw(style = "B") %>%
+    listw2sn
```

The resulting graph had outer edges that were very long (connecting marginal locations by long edges) and fairly parallel to nearby inner edges. To help in removing them, we calculated the length of each edge as the Euclidean distance between the locations at the end of the edges as follows:

```
R> names(mite.edge)[3L] <- "distance"
R> mite.edge$distance <-
+    mite.xy %>%
+    dist %>%
+    as.matrix %>%
+    .[mite.edge[,1L:2L] %>% as.matrix]
```

Here, column `weight` was renamed `distance` and assigned the Euclidean distances. Then, edges with distance > 1.5 m were discarded from the edge list as follows:

```
R> mite.edge %<>% .[.$distance <= 1.5,]
```

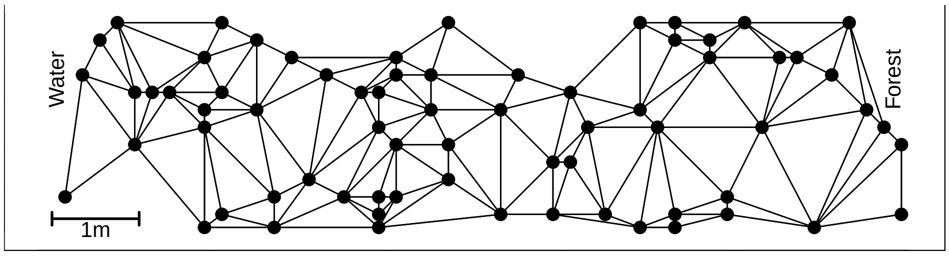

Figure 6: Map of the 2.5 m wide by 10 m long plot of peat mat sampled by Borcard *et al.* (1992), with the spatial contiguity network of edges obtained by Delauney triangulation that was used for the first application example (the plot is rotated 90° in the counterclockwise direction). Core #1 is in the lower-right corner of the map near the forest edge; core #70 is in the lower-left corner near the open water edge of the plot.

An edge list may also be obtained from some underlying theory or process, or by known connections among sites (e.g., hydrographic network, roads, power lines, underground pipe network) resulting in a more or less densely connected network. Any approach can be used as long as any given observation point is indirectly connected to all other points. The upper density limit of a connection network is attained when all observation points are directly connected to all other points; there is then no spatial contiguity constraint.

For this example, let us cluster the cores with respect to their species composition. A suitable dissimilarity metric to analyze species composition (or presence/absence) data should not consider the joint absences (or null abundances) of a species at two sites as an evidence for similarity as much as their joint presences (or abundances > 0), as the Euclidean distance would do; see Legendre and Legendre (2012) for discussions of double-zero symmetric and asymmetric coefficients. To compute the dissimilarity matrix, we used the Hellinger distance, which is a double-zero asymmetrical coefficient appropriate for the analysis of community composition data. It is obtained by computing the Euclidean distance of the square-rooted relative abundance data (`mite$fau`). This distance can be computed directly by function `dist.ldc` of the R package adespatial as follows:

```
R> mite.hel <- mite %>% dist.ldc("hellinger")
```

```
Info -- This coefficient is Euclidean
```

Constrained hierarchical agglomerative clustering is obtained using function `constr.hclust` as follows:

```
R> mite.chclust <- constr.hclust(d = mite.hel, links = mite.edge,
+    coords = mite.xy[, c(2, 1)])
```

Here, clustering is carried out using the dissimilarity matrix `mite.hel`, with `mite.link` providing the edge list and `mite.xy` the site coordinates. The sorting strategy is left to the default value (`"ward.D2"` to obtain Ward agglomerative clustering; that default is different from `hclust`'s default value, which is `"complete"`). A suite of figures showing spatially constrained partitions into two, three, five, and seven groups can be obtained as follows (Figure 7):

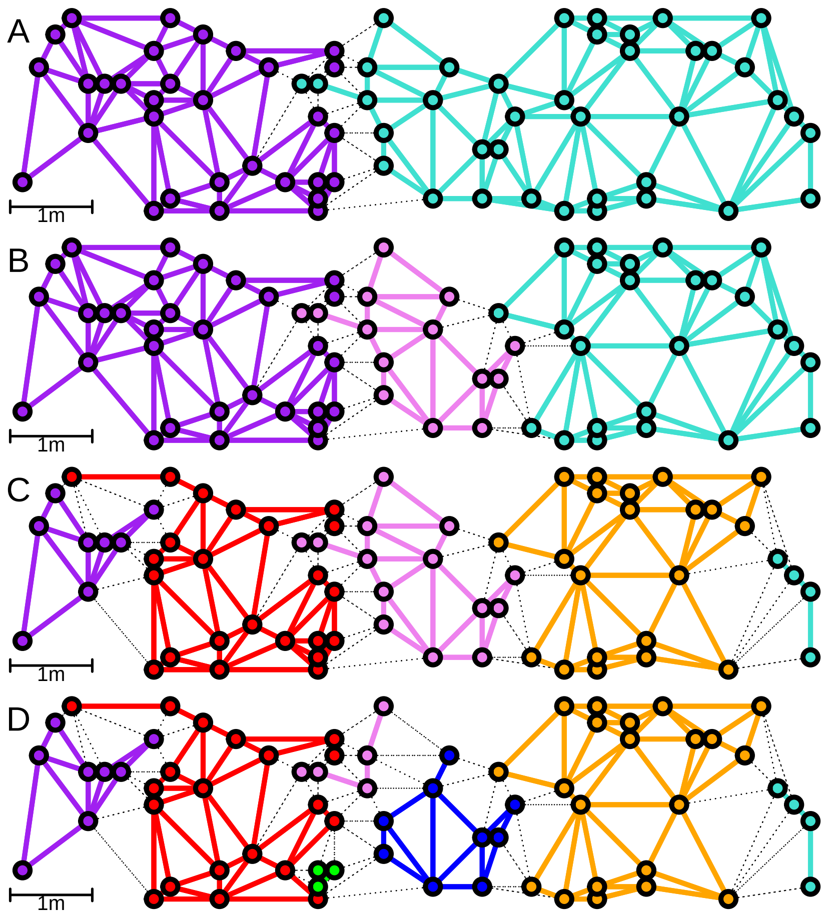

Figure 7: A set of four maps showing partitions of the sites into two (A), three (B), five (C), and seven (D) clusters, with points and segments of different colors. Fine black dotted segments are edges with vertices in different groups.

```
R> par(mfrow = c(4, 1), mar = c(0.5, 0, 0.5, 0))
R> cols <- c("turquoise", "orange", "blue", "violet", "green", "red",
+    "purple")
R> parts <- c(2, 3, 5, 7)
R> for (i in 1L:length(parts)) {
+    plot(NA, xlim = c(10, 0), ylim = c(-0.1, 2.5), xaxs = "i", yaxs = "i",
+      asp = 1, axes = FALSE)
+    arrows(x0 = 9.85, x1 = 8.85, y0 = 0.1, y1 = 0.1, code = 3,
+      length = 0.05, angle = 90, lwd = 2)
+    text(x = 9.35, y = 0, labels = "1m", cex = 1.5)
+    plot(mite.chclust, parts[i], links = TRUE, plot = FALSE,
+      col = cols[round(seq(1, length(cols), length.out = parts[i]))],
+      lwd = 4, cex = 2.5, pch = 21, hybrids = "single", lwd.hyb = 0.25,
+      lty.hyb = 3, xpd = TRUE)
+    text(x = 9.75, y = 2.25, labels = LETTERS[i], cex = 2.5)
+ }
```

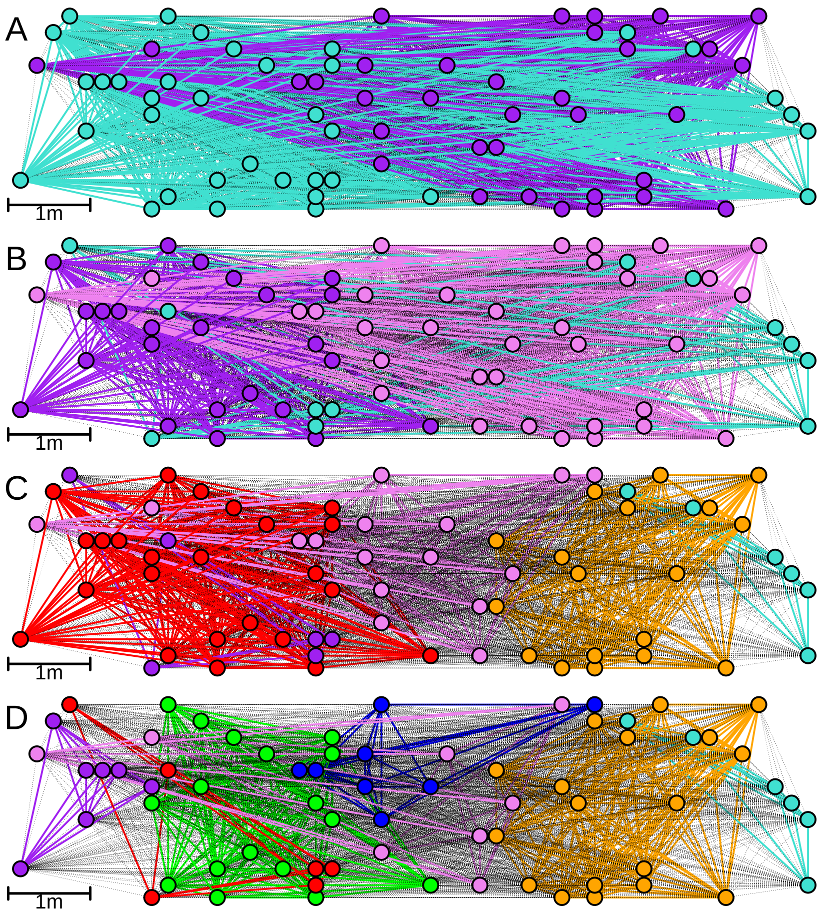

Figure 8: A set of four maps showing partitions of the sites into two (A), three (B), five (C), and seven (D) clusters obtained without a spatial contiguity constraint.

The partitioning highlights the spatial variation in mite species composition as one proceeds from the water (on the left) to the forest (on the right). The first partition takes place at about 40% of the water-forest distance (A: first panel from the top down). The second partition (B: second panel) is concomitant with a change in the peat composition (see Borcard and Legendre (1994), Figure 1: the change occurs from the *Sphagn. 1* to the *Sphagn. 2* peat types). The five-group partition (C: third panel) separates the sites closest to the open water, and the four sites closest to the forest edge, from the three central groups of sites. Finally, the seven-group partition (D: fourth panel) shows a group of five sites (with *Sphagn. 1* substrate), and a group of three sites (with bare peat) separating from the central clusters. Clustering without a spatial contiguity constraint (i.e., one where any points is free to cluster with any other point), also with `method = "ward.D2"`, yielded a different result, though the partitions also seem to feature increasing spatial organisation as the number of clusters increases (Figure 8).

The example illustrates the fact that, because of its hierarchical nature, agglomerative clustering always results into subgroups that are embedded into higher-level clusters. In addition, because of the constraint of spatial contiguity, each group formed is geometrically compact since its member sites are connected by contiguity edges.

3.2. Second application example: Tiahura fish transect

As our second example, we will use the Tiahura fish transect data described and analyzed by Galzin and Legendre (1987). It consists of presence/absence data for 280 fish species observed at 22 sites along a 1020 m long coast-to-sea cross-reef transect located in front of the Tiahura village, near the northwestern corner of the high volcanic island of Moorea in French Polynesia (WGS84: -17.4934, -149.8680). Each fish survey site was 50 m long. Species presence/absence data were recorded by a diver trained in underwater fish identification. The transect began on a coral sand beach, followed by a zone of detritic sediments, then a dying reef flat, followed by a zone of coral patches. That relatively flat area ended into a 100 m wide channel, 9 m deep, followed by a barrier reef 490 m wide, which ended in a slightly elevated reef ridge, followed by the outer slope in the Pacific Ocean. The survey was terminated at depth of ≈ 25 m; this is the maximum depth allowing scuba diving for any length of time without having to perform decompression stops. See Galzin and Legendre (1987) for further details about the transect and the survey method.

The data set also includes a data matrix of fish traits (we did not use them for this example) and a matrix of environmental variables observed at the sites. We used the latter along with the clustering results to highlight possible factors affecting changes in population structure. The presence/absence Tiahura fish data set are contained in an R data file, which is loaded as follows:

```
R> data("Tiahura", package = "adespatial")
```

The file contains the following information:

`Tiahura$fish` A 22 sites (rows) $\times$ 280 species (columns) data matrix indicating presence (1) or absence (0) of each fish species at each site.

`Tiahura$species` Names of the 280 fish species.

`Tiahura$trait` Five categorical variables describing behavioral traits of the 280 fish species (not used in this example).

`Tiahura$habitat` Ten environmental variables describing the habitat at the 22 survey sites.

`Tiahura$reef` A data frame with 6 rows (sections) and 3 columns describing the different sections of the transect.

We will perform clustering on dissimilarities obtained as $D = \sqrt{1 - S}$ where S is the Jaccard index of similarity (Legendre and Legendre 2012), which is obtained from function `dist.ldc` found in R package **adespatial** as follows:

```
R> tiah.jac <- Tiahura$fish %>% dist.ldc(method = "jaccard")

Info -- D is Euclidean because the function outputs D[jk] = sqrt(1-S[jk])
```

Because the transect is a linear arrangement of study sites, we are using the argument for chronological clustering (`chron = TRUE`) to automatically create connections between immediate neighbors. This argument dispenses us from explicit edge calculation. We have, of course, to check and make sure that the sites are in the correct order in the data file. The clustering is obtained as follows:

```
R> tiah.chclust <- constr.hclust(d = tiah.jac,
+    coords = Tiahura$habitat[,"distance"], chron = TRUE)
```

The constrained clustering results are displayed as follows (Figure 9):

```
R> par(mfrow = c(3, 1))
R> par(mar = c(3, 6.5, 2, 2))
R> dst <- Tiahura$habitat[, "distance"]
R> plot(NA, xlim = dst %>% range, ylim = c(0.5, 5.5), yaxt = "n",
+    ylab = "Partitions\n\n", xlab = "")
R> parts <- c(2, 3, 5, 7, 12)
R> cols <- c("turquoise", "orange", "chartreuse", "aquamarine", "blue",
+    "violet", "pink" ,"cyan" ,"green", "red", "cornsilk", "purple")
R> for (i in 1L:length(parts)) {
+    tiah.chclust$coords[,"y"] <- i
+    plot(tiah.chclust, parts[i], link = TRUE, lwd = 3, hybrids = "none",
+      lwd.pt = 0.5, cex = 3, pch = 21, plot = FALSE,
+      col = cols[round(seq(1, length(cols), length.out = parts[i]))])
+  }
R> axis(2, at = 1:length(parts), labels = paste(parts, "groups"), las = 1)
R> par(mar = c(4, 6.5, 1, 2))
R> plot(x = dst, y = Tiahura$habitat[,"depth"],
+    ylim = Tiahura$habitat[,"depth"] %>% range %>% max %>% c(-300),
+    las = 1, ylab = "Depth\n(cm)\n", xlab = "", type = "l", lwd = 2)
R> for (i in 1L:nrow(Tiahura$reef)) {
+    abline(v = Tiahura$reef[i,2], lty = 3)
+    abline(v = Tiahura$reef[i,3], lty = 3)
+    if ((Tiahura$reef[i,3] - Tiahura$reef[i,2]) < 100) {
+      text(x = (Tiahura$reef[i,2] + Tiahura$reef[i,3]) / 2, y = 2350,
+        labels = toupper(Tiahura$reef[i,1]), srt = 90, adj = 0)
+    } else {
+      text(x = (Tiahura$reef[i,2] + Tiahura$reef[i,3]) / 2, y = -150,
+        labels = toupper(Tiahura$reef[i,1]))
+    }
+  }
R> par(mar = c(5, 6.5, 0, 2))
R> plot(NA, xlim = dst %>% range, ylim = c(0, 1), las = 1,
+    ylab = "Bottom composition\n(proportions)\n", xlab = "Distance (m)")
R> bot <- cbind(0, Tiahura$habitat[,3:10])
R> for (i in 2:9) bot[,i] <- bot[,i] + bot[,i-1]
R> cols <- c("", "grey75", "brown", "grey25", "green", "purple",
+    "lightgreen", "yellow", "white")
R> for (i in 2:9)
+    polygon(x = c(dst, rev(dst)), y = c(bot[,i], rev(bot[,i-1])) / 50,
+      col = cols[i])
R> text(x = c(44, 365, 707, 538, 957, 111, 965),
+    y = c(0.05, 0.47, 0.37, 0.58, 0.42, 0.80, 0.88),
+    labels = colnames(bot)[2:8], xpd = TRUE)
```

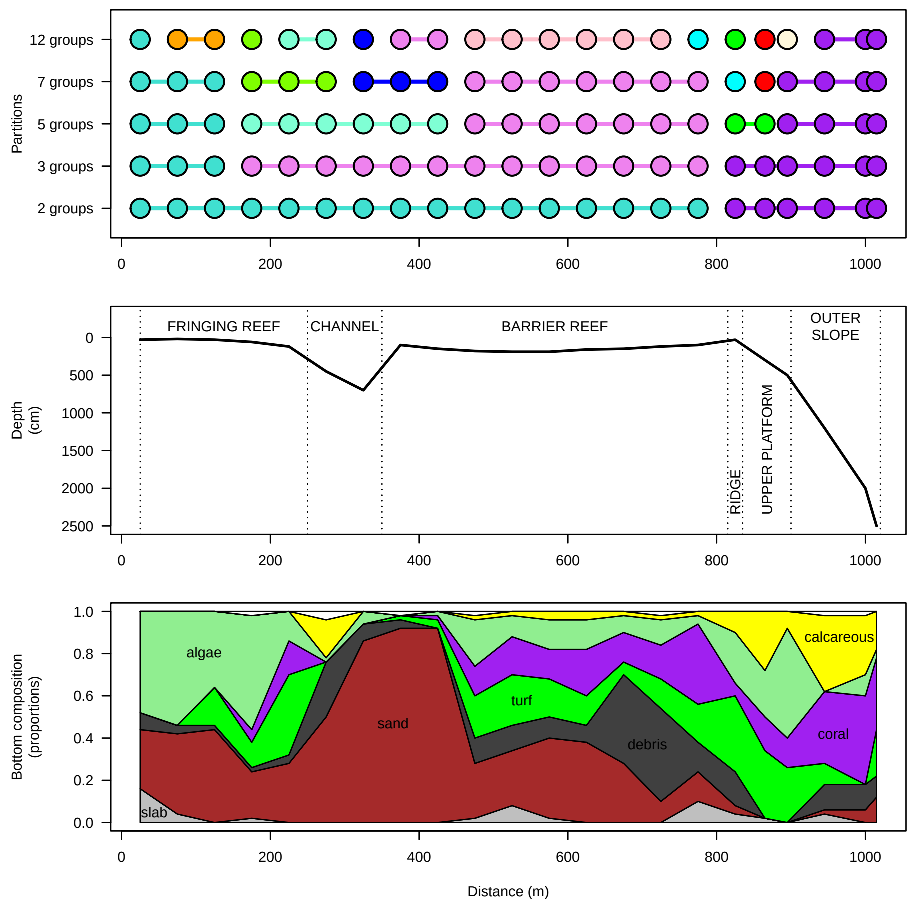

Figure 9: Upper panel: partitions of the Tiahura fish community structure into two, three, five, and seven groups along a cross-reef transect. Partitions were obtained by constrained hierarchical clustering analysis performed on the basis of the Jaccard index and using the minimum variance sorting strategy (Ward, Jr. 1963). Abscissa: distance in m from the beach. Middle panel: water depth as a function of the distance from the shore. Lower panel: substrate composition in each survey site, expressed as the number of times each of 8 substrate types was observed at 50 points, spaced by 1 m, along a 50 m rope stretched close to the bottom of each site; the values were in the range $[0, 50]$. The bottom is classified as stone slab (grey), sand (brown), coral debris (charcoal), turf and dead coral (green), live coral (purple), large algae (light green), calcareous algae (yellow), other substrate (white; large echinoderms: holothuroids and sea stars; sponges, anemones, and alcyonarians).

The partition in two groups shows a first shift in community structure at the reef ridge, with the reef flat on one side, and the ridge, upper platform, and outer slope on the other (Figure 9). The three-group partition separates the three sites of the sandy fringing reef near the beach from the other sites of the reef flat. The five-group partition creates two groups on the reef flat, the first one with six sites dominated by sandy substrate, the second with seven sites on a variety of coralline substrate types. The partition into seven groups splits the cluster of six sandy sites into two groups of three sites separated by the deepest point of the channel.

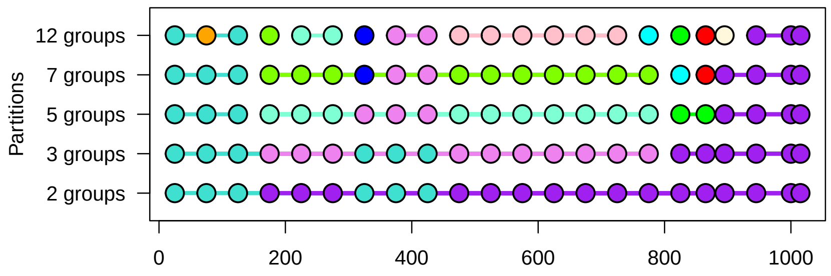

Figure 10: Partitions of the Tiahura fish community structure into two, three, five, and seven groups along a cross-reef transect without the contiguity constraint. Abscissa: distance in meter from the beach.

From that point, the following partitions create finer and finer groups. In the twelve-group partition, for instance, the reef appears to have highly diversified fish assemblages (high β diversity; see next paragraph), except for the six coralline sites on the barrier reef which form a fairly homogeneous block of sites. The reef ridge region, in particular, shows a rapid succession of assemblages. The four sites closest to the ridge (located at site #17) are singletons. The nine sites, from the beach up to (and including) the channel, are also highly β-diversified: the 12-cluster partition recognized six groups, including three singletons, among these nine sites. Performing the analysis without the contiguity constraint, also with `method = "ward.D2"`, yielded a different result (Figure 10), but the 12-cluster partition differed only by the first two points, perhaps because the fish community composition is naturally spatially structured.

β diversity is the variation in the species composition of ecological communities among the study sites. It can be mathematically estimated in different ways. Given a set of sites disseminated throughout a study area, β diversity is high when the sites vary a lot in their species compositions, and it is low when the sites have fairly or completely similar compositions (Anderson *et al.* 2011; Legendre and De Cáceres 2013).

Analysis of the fine partitions can proceed further until all observations are single-observation clusters. By analyzing where (in spatial studies) or when (in temporal studies) clusters split at different levels of the criterion, the analysis may help identify processes structured in space or time that influence the community structure (or other data) used in the analysis.

This example, which involves clustering a series of locations along a transect, features similar computations for clustering as the analysis of multiple observations performed repeatedly at the same location, (i.e., a time series). In such a situation, time-constrained hierarchical agglomerative clustering is referred to as "chronological clustering" and allows one to identify abrupt change points in the time series.

4. Summary and discussion

Constrained hierarchical clustering methods take into account more information than their unconstrained counterparts. For spatial or temporal contiguity, the admissible clusters are those obeying the contiguity relationship. In this paper, we presented a new implementation of constrained hierarchical clustering using the generalized Lance and Williams algorithm in the R language and environment. We also illustrated the usefulness of that kind of analysis in addressing different kinds of problems that arise in science, such as partitioning the elements of a map, or segmenting observations along transects or time series.

The examples used in the present paper do not represent an exhaustive enumeration of what constrained hierarchical clustering can achieve. For instance, constrained hierarchical clustering would be applicable to three-dimensional or spatio-temporal sampling designs (e.g., Planes, Lefèvre, Legendre, and Galzin 1993), provided that a suitable distance metric was used and that the contiguity of the observations was accurately described by an edge list. If these conditions are met, our constrained clustering program will encounter no difficulty in computing a solution.

The program introduced here is geared towards flexibility and meant to be a general-purpose implementation of constrained hierarchical clustering. On the one hand, for specific applications like, for instance, image segmentation, it may not perform as well as specialized software (e.g., Li 2011; Sourati, Brooks, Dy, and Erdogmus 2012; Lou, Yang, and Cao 2015). The latter generally assumes neighboring relationships between contiguous pixels of a square grid (i.e., an image) or contiguity of elements along a time series (i.e., chronological clustering) and implements a single classificatory sorting strategy (or a limited number thereof) for performance reasons. Our function is more flexible and general.

4.1. Performance

Our constrained hierarchical clustering program operates on a copy of the dissimilarity matrix. That copy is updated as clusters are merged during the agglomeration process. The number of non-redundant elements in dissimilarity matrices increases in a quadratic way with sample size. In practice, we found the amount of computation time and storage required to carry out the analysis to grow following roughly a power relationship during benchmark tests (i.e., a linear relationship on a log-log scales; Figure 11), with empirical results consistent with $O(n^2)$ requirements (Murtagh 1985; Langfelder and Horvath 2012). As long as the internal data types remain the same for the R computing environment, the storage requirements *vs.* sample size relationship shown in Figure 11 may hold for various systems (i.e., sets of hardware and software). However, the computation time-sample size relationship should not be expected to be consistent among systems as it is influenced by a broad array of different factors such as the type of hardware used and its configuration, the operating system from which it is used and its configuration, the build version of the R environment, its software dependencies (e.g., the compiler used, compiler options), and so on.

4.2. Reversals

Reversals occur when two groups are merged at a dissimilarity value smaller than the dissimilarity at which any or both of the subgroups were formed (Ferligoj and Batagelj 1982; Murtagh 1985). In unconstrained hierarchical clustering using monotonic methods, groups are merged at consistently larger dissimilarities than the ones at which their subgroups were themselves previously formed; thus monotonic hierarchical clustering methods cannot produce reversals. In addition, complete-linkage clustering cannot produce reversals in constrained hierarchical clustering. For all the other strategies enumerated in Table 1, reversals may occur in constrained hierarchical clustering.

It is not clear to us whether reversals really represent a problem regarding the adequacy of the clustering solution, besides yielding a dendrogram with a somewhat awkward representation. In constrained clustering, the clusters are usually represented on a map or along a transect for spatial constraints, along a time series for temporal constraint, on a phylogenetic tree for

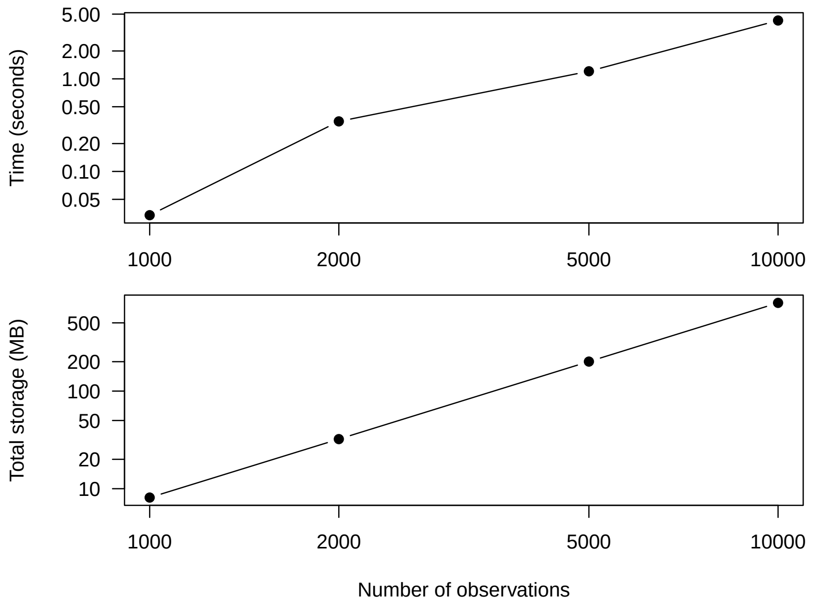

Figure 11: Benchmark results showing the running time and memory usage (on logarithmic scales) for analyzing data sets having different numbers of observations. The data sets were randomly generated and consisted of a normally-distributed variable ($\mu = 0$, $\sigma^2 = 1$) on which the Euclidean distance was calculated to be used as dissimilarities (argument `d` of function `constr.hclust`) and an uniformly-distributed, two-dimensional plot that was used to calculate a Delauney triangulation and obtain the edge list. The clustering strategy (argument `method` of function `constr.hclust`) was set to `"ward.D2"`. Computation time is that used only for the execution of `constr.hclust` whereas total storage was obtained as 2 $\times$ the size of the dissimilarity matrix + the size of the edge list + the size of the results. Hardware description for these benchmarks is an Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (i.e., the exact vendor's description string) with 8041076 kB of RAM and running Linux release 4.15.0-91-generic. The amount of RAM used for calculation and computation time follow relationships with sample size (n) of the form an^b, where a and b are empirically-derived constants that we found to be $a = 8.4082$ byte and $b = 1.9946$ for storage and $a = 4.9182\text{e-}05$ ms and $b = 2.0009$ for running time for the system on which the benchmarks were calculated.

a phylogenetic constraint, or on a network for a network-based constraint. The issue was raised by Murtagh (1985) that the presence of reversals "makes difficult the interpretation of partitions and the definition of similarity between classes". As we previously illustrated, a reversal arises in constrained clustering when observations with similar values are not neighbors in space; they will merge later in the process when the groups to which they pertain finally become neighbors in space. This kind of situation is expected for data sets with repeated spatial patterns such as waves and bumps; this situation is not uncommon in practice. Since the process from where the negative dendrogram edges (corresponding to reversals) occur appears straightforward to us, we cannot conclude that they make the interpretation of partitions or the definition of similarity between classes more complex. Also, since obtaining a given number of partitions is readily done using the order in which the groups merge (i.e., by cutting the tree at a given dissimilarity level), we do not see how the fact that some dendrogram edges are non-positive represents a significant issue during partitioning.

The exception of complete linkage comes from the fact that in that case, dissimilarities between the merged groups and other groups is the largest of any or the observations between the groups. Therefore, it is not possible to encounter a smaller similarity later during the clustering process (Ferligoj and Batagelj 1982). One could thus resort to complete linkage clustering when reversals in dendrograms should be avoided at all cost. However, we cannot think of an example where this situation would present itself. Another possibility would be to attempt developing alternative strategies to those shown in Figure 1, which would be designed to avoid reversals.

4.3. Future perspectives and developments

Besides space and time, clustering can be constrained by any network structure related to other source of knowledge. For instance, one could cluster the response of communities, quantified as the shifts in their relative species abundances as the environment changes, as proposed by Legendre (1987), using, for example, the structure of the trophic network as a constraint. Such an analysis would allow one to obtain a suite of partitions of a community into groups with related trophic status and whose responses to environmental changes would be the most consistent. Also, rather than using the classification tree to obtain partitions, it may be possible to devise approaches to use its topological and edge length information in spectral decomposition methods similar to those proposed to model directional spatial processes in ecology (Blanchet, Legendre, and Borcard 2008) or to model phylogenetic variation in species traits (Guénard, Legendre, and Peres-Neto 2013).

We are hoping that the software developed and presented in the present paper will be useful to researchers with a clustering problem where applying a spatial contiguity constraint would be relevant. New clustering strategies may be added in the future. Therefore, we encourage the reader to keep their software up to date and consult the documentation for recent additions that may help them in their analyses. We also encourage anyone with suggestions for improving and enhance the software and its documentation to contact us.

Computational details

The constrained hierarchical clustering analyses featured in this paper were produced with R 4.2.0 and the package **adespatial** 0.3-19. Other calculations and generation of the figures were done with packages **magrittr** 2.0.3, **spdep** 1.2-4, and **vegan** 2.6-2. R and the other packages used are available from CRAN at `https://CRAN.R-project.org/`.

Acknowledgments

We are thankful to early users of the **adespatial** package for their help in testing the completeness of the package's functions as well as to Fionn Murtagh for advise during the early writing of the paper. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) research grant no. 7738 to P. Legendre.

References

Anderson MJ, Crist TO, Chase JM, Vellend M, Inouye BD, Freestone AL, Sanders NJ, Cornell HV, Comita LS, Davies KF, Harrison SP, Kraft NJB, Stegen JC, Swenson NG (2011). "Navigating the Multiple Meanings of β Diversity: A Roadmap for the Practicing Ecologist." *Ecology Letters*, **14**, 19–28. doi:10.1111/j.1461-0248.2010.01552.x.

Bache SM, Wickham H (2022). ***magrittr**: A Forward-Pipe Operator for R*. URL https://CRAN.R-project.org/package=magrittr.

Bivand RS, Pebesma EJ, Gomez-Rubio V (2013). *Applied Spatial Analysis with R*. 2nd edition. Springer-Verlag. doi:10.1007/978-1-4614-7618-4.

Blanchet FG, Legendre P, Borcard D (2008). "Modelling Directional Spatial Processes in Ecological Data." *Ecological Modelling*, **215**, 325–336. doi:10.1016/j.ecolmodel.2008.04.001.

Borcard D, Legendre P (1994). "Environmental Control and Spatial Structure in Ecological Communities: An Example Using Oribatid Mites (Acari, Oribatei)." *Environmental and Ecological Statistics*, **1**(1), 37–61. doi:10.1007/bf00714196.

Borcard D, Legendre P, Drapeau P (1992). "Partialling out the Spatial Component of Ecological Variation." *Ecology*, **73**(3), 1045–1055. doi:10.2307/1940179.

De Soete G, Carroll JD, DeSarbo WS (1987). "Least Squares Algorithms for Constructing Constrained Ultrametric and Additive Tree Representations of Symmetric Proximity Data." *Journal of Classification*, **4**(2), 155–173. doi:10.1007/bf01896984.

Dray S, Bauman D, Blanchet G, Borcard D, Clappe S, Guénard G, Jombart T, Larocque G, Legendre P, Madi N, Wagner HH (2022). ***adespatial**: Multivariate Multiscale Spatial Analysis*. URL https://CRAN.R-project.org/package=adespatial.

Ferligoj A, Batagelj V (1982). "Clustering with Relational Constraint." *Psychometrika*, **47**(4), 413–426. doi:10.1007/bf02293706.

Galzin R, Legendre P (1987). "The Fish Communities of a Coral Reef Transect." *Pacific Science*, **41**(1–4), 158–165.

Guénard G, Legendre P, Peres-Neto P (2013). "Phylogenetic Eigenvector Maps (PEM): A Framework to Model and Predict Species Traits." *Methods in Ecology and Evolution*, **4**, 1120–1131. doi:10.1111/2041-210x.12111.

Kaufman L, Rousseeuw PJ (2005). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons. doi:10.1002/9780470316801.

Lance GN, Williams WT (1966). "A Generalized Sorting Strategy for Computer Classifications." *Nature*, **212**(5058), 218–218. doi:10.1038/212218a0.

Lance GN, Williams WT (1967). "A General Theory of Classificatory Sorting Strategies 1. Hierarchical Systems." *The Computer Journal*, **9**(4), 373–380. doi:10.1093/comjnl/9.4.373.

Langfelder P, Horvath S (2012). "Fast R Functions for Robust Correlations and Hierarchical Clustering." *Journal of Statistical Software*, **46**(1), 1–17. doi:10.18637/jss.v046.i11.

Legendre P (1987). "Constrained Clustering." In P Legendre, L Legendre (eds.), *Developments in Numerical Ecology*, volume G-14 of *NATO ASI Series*, pp. 289–307. Springer-Verlag, Berlin.

Legendre P, Dallot S, Legendre L (1985). "Succession of Species within a Community: Chronological Clustering, with Applications to Marine and Freshwater Zooplankton." *The American Naturalist*, **125**(2), 257–288. doi:10.1086/284340.

Legendre P, De Cáceres M (2013). "Beta Diversity as the Variance of Community Data: Dissimilarity Coefficients and Partitioning." *Ecology Letters*, **16**, 951–963. doi:10.1111/ele.12141.

Legendre P, Guénard G (2020). ***constr.hclust**: Space- and Time-Constrained Clustering Package.* URL https://numericalecology.com/Rcode/.

Legendre P, Legendre L (2012). *Numerical Ecology*. 3rd English edition. Elsevier.

Li J (2011). "Agglomerative Connectivity Constrained Clustering for Image Segmentation." *Statistical Analysis and Data Mining: The ASA Data Science Journal*, **4**(1), 84–99. doi:10.1002/sam.10109.

Lou JY, Yang XL, Cao AZ (2015). "A Spatial Shape Constrained Clustering Method for Mammographic Mass Segmentation." *Computational and Mathematical Methods in Medicine*, **2015**. doi:10.1155/2015/891692.

Murtagh F (1985). "A Survey of Algorithms for Contiguity-Constrained Clustering and Related Problems." *The Computer Journal*, **28**(1), 82–88. doi:10.1093/comjnl/28.1.82.

Oksanen J, Simpson GL, Blanchet FG, Kindt R, Legendre P, Minchin PR, O'Hara RB, Solymos P, Stevens MHH, Szoecs E, Wagner H, Barbour M, Bedward M, Bolker B, Borcard D, Carvalho G, Chirico M, De Cáceres M, Durand S, Evangelista HBA, FitzJohn R, Friendly M, Furneaux B, Hannigan G, Hill MO, Lahti L, McGlinn D, Ouellette MH, Ribeiro Cunha E, Smith T, Stier A, Ter Braak CJF, Weedon J (2022). ***vegan**: Community Ecology Package.* R package version 2.6-2, URL https://CRAN.R-project.org/package=vegan.

Planes S, Lefèvre A, Legendre P, Galzin R (1993). "Spatio-Temporal Variability in Fish Recruitment to a Coral Reef (Moorea, French Polynesia)." *Coral Reefs*, **12**(2), 105–113. doi:10.1007/bf00302110.

Rand WM (1971). "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association*, **66**(336), 846–850. doi:10.1080/01621459.1971.10482356.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Sourati J, Brooks DH, Dy JG, Erdogmus D (2012). "Constrained Spectral Clustering for Image Segmentation." In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6. doi:10.1109/mlsp.2012.6349765.

Ward, Jr JH (1963). "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association*, **58**(301), 236–244. doi:10.1080/01621459.1963.10500845.

Affiliation:

Guillaume Guénard, Pierre Legendre
Département de sciences biologiques
Univesité de Montréal
CP 6128, succrusale centre-ville
Montréal, Québec
H3C 2J7 Canada
E-mail: guillaume.guenard@gmail.com, pierre.legendre@umontreal.ca
URL: https://numericalecology.com/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
June 2022, Volume 103, Issue 7 *Submitted:* 2019-12-30
doi:10.18637/jss.v103.i07 *Accepted:* 2021-08-25