



Contiguity-Constrained Hierarchical Agglomerative Clustering Using SAS

Anthony Recchia
University of Illinois

Abstract

Hierarchical clustering is one of the most basic methods for partitioning a set of objects into clusters of similar objects. In standard clustering analysis, every pair of objects or clusters is eligible to be joined during each iteration of the clustering algorithm. However, there are circumstances under which it would be preferable to limit this eligibility. One obvious situation is when the objects of interest are geographical regions which should only be allowed to merge when they are contiguous. The goal here is to demonstrate a SAS macro that will perform the agglomerative version of hierarchical clustering while providing the user with an intuitive means of imposing a contiguity constraint.

Keywords: contiguity constraint, hierarchical clustering, SAS.

1. Introduction

Clustering is a major branch of multivariate statistical analysis. Many of the software packages currently in use can perform a variety of different types of clustering analyses. SAS (SAS Institute Inc. 2003), for example, has clustering procedures that implement many of the usual techniques.

In standard clustering analysis, every pair of objects or clusters is eligible to be joined during each iteration of the clustering algorithm. However, there are circumstances under which it would be preferable to limit this eligibility. One obvious situation is when the objects of interest are geographical regions which should only be allowed to merge when they are contiguous. Such constraints have applications in the earth and social sciences and in image processing. They are also of use in the insurance industry for the development of rating factors based upon territorial characteristics.

Our goal here is to demonstrate a SAS macro that will perform one specific type of clustering analysis, hierarchical agglomerative clustering, in either a constrained or unconstrained

manner. The macro is meant to mimic the look and feel of SAS's hierarchical clustering procedure, PROC CLUSTER, while providing the user with an intuitive means of imposing a contiguity constraint.

2. Hierarchical agglomerative clustering

2.1. Unconstrained clustering

Clustering analysis involves partitioning a set $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of n objects into clusters of similar objects. Perhaps the simplest clustering method is hierarchical agglomerative clustering. For this method, each object begins in its own cluster. Proceeding iteratively, the most “similar” (or, equivalently, the least “dissimilar” or least “distant”) pair of clusters is joined one at a time until all of the objects are contained in a single cluster. There are many variations on this method, each distinguished by how dissimilarities between objects and between clusters are defined.

Initial dissimilarities between pairs of objects are determined by measurements on one or more variables. For each pair $(\mathbf{x}_i, \mathbf{x}_j)$ of objects, a dissimilarity value $d(\mathbf{x}_i, \mathbf{x}_j)$ is calculated in some way based upon these measurements. The dissimilarity function $d(\cdot, \cdot)$ can be any function that satisfies the properties $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$, $d(\mathbf{x}_i, \mathbf{x}_i) = 0$, and $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$ for all i and j . Once calculated, these dissimilarities are assembled into a dissimilarity matrix whose (i, j) entry is the dissimilarity between \mathbf{x}_i and \mathbf{x}_j .

During the first iteration of the clustering algorithm, the least dissimilar pair of objects are located with the initial dissimilarity matrix and joined. As the algorithm proceeds and clusters are joined to form larger clusters, new dissimilarity values must be calculated between newly formed clusters and any remaining clusters. There are many methods by which this updating can be performed, and those that are available in our SAS macro are discussed in the next section. At the end of each iteration of the algorithm, the particular update method that was chosen by the user is applied to the dissimilarity matrix to reflect the joining that took place during that iteration.

Thus, standard hierarchical agglomerative clustering analysis proceeds according to the following scheme.

1. Construct the initial dissimilarity matrix.
2. Locate the smallest value in the dissimilarity matrix.
3. Join the corresponding pair of clusters.
 - (a) Remove their entries from the dissimilarity matrix.
 - (b) Calculate dissimilarities between the new cluster and each remaining cluster.
 - (c) Add these new entries to the dissimilarity matrix.
4. Repeat steps 2. and 3. until there is only one cluster remaining.

This is the scheme that our SAS macro will follow when performing an unconstrained analysis. Such an analysis will require the user to provide a dissimilarity matrix as an input data set.

Of course, this functionality is included only for convenience and produces essentially the same output as `PROC CLUSTER` itself.

2.2. Clustering with a contiguity constraint

One approach to imposing a contiguity constraint is to form a contiguity matrix whose (i, j) entry is equal to 1 if \mathbf{x}_i and \mathbf{x}_j are contiguous and is equal to 0 otherwise (Gordon 1996). This matrix would then be consulted during each iteration of the clustering algorithm to ensure that only contiguous clusters are considered for joining. The least dissimilar pair of contiguous clusters would then be joined, and the contiguity matrix would be updated to reflect the fact that any cluster which was contiguous with either of the clusters that were just joined must now be contiguous with the newly formed cluster.

The SAS macro discussed here uses a slightly different approach. Instead of a contiguity matrix, the macro uses a list of all of the contiguous pairs of objects. This has a couple of advantages. First, such a list is more straightforward to create for the user who is likely to have to input the pairs by hand. And second, since a contiguity list does not contain the discontinuous pairs of objects, this approach eliminates the need to search the contiguity matrix for its nonzero entries during each iteration of the clustering algorithm.

Thus, contiguity-constrained hierarchical agglomerative clustering analysis proceeds according to the following scheme.

1. Construct the initial dissimilarity matrix and contiguity list.
2. Locate the smallest value among the contiguous pairs in the dissimilarity matrix.
3. Join the corresponding pair of clusters.
 - (a) Remove their entries from the dissimilarity matrix.
 - (b) Calculate dissimilarities between the new cluster and each remaining cluster.
 - (c) Add these new entries to the dissimilarity matrix.
4. Update the contiguity list.
5. Repeat steps 2. through 4. until there are no more contiguous pairs of clusters remaining.

This is the scheme that our SAS macro will follow when performing a constrained analysis. Such an analysis will require the user to provide a dissimilarity matrix and a contiguity list as input data sets.

3. Updating the dissimilarity matrix

Given the initial dissimilarities $d(\mathbf{x}_i, \mathbf{x}_j)$ between the objects of interest, we must be able to calculate new dissimilarities between disjoint subsets H and M of these objects in order to update the dissimilarity matrix. If, for instance, an iteration of the clustering algorithm were to join clusters K and L to form cluster M , then the task of updating the dissimilarity matrix consists of first removing all dissimilarities involving either K or L and then appending new dissimilarities D_{HM} between M and any remaining cluster H . The method by which

Method	Description	Formula for D_{HM} where $M = K \cup L$
AVERAGE	Average linkage method	$\frac{n_K D_{HK} + n_L D_{HL}}{n_K + n_L}$
CENTROID	Centroid method	$\frac{n_K D_{HK} + n_L D_{HL}}{n_K + n_L} - \frac{n_K n_L D_{KL}}{(n_K + n_L)^2}$
COMPLETE	Complete linkage method	$\max(D_{HK}, D_{HL})$
FLEXIBLE	Flexible-beta method	$(1 - \beta) \frac{D_{HK} + D_{HL}}{2} + \beta D_{KL}$
MCQUITTY	McQuitty's similarity method	$\frac{D_{HK} + D_{HL}}{2}$
MEDIAN	Median method	$\frac{D_{HK} + D_{HL}}{2} - \frac{D_{KL}}{4}$
SINGLE	Single linkage method	$\min(D_{HK}, D_{HL})$
WARD	Ward's minimum variance method	$\frac{(n_H + n_K) D_{HK} + (n_H + n_L) D_{HL} - n_H D_{KL}}{n_H + n_K + n_L}$

Table 1: Clustering methods for the macro `contigclust`.

these new dissimilarities are calculated is chosen ahead of time by the user and essentially distinguishes one hierarchical agglomerative clustering method from another.

The update methods that are implemented in our SAS macro are described in Table 1. Here, the numbers n_H , n_K , and n_L refer to the sizes of the clusters H , K , and L , respectively. The user specifies the clustering method that they wish to use by supplying one of the names from the first column of the table as the value of the macro argument `Method`. Then if they are using the `FLEXIBLE` method, they supply the value of the parameter β in the Flexible-Beta update formula using the argument `Beta`. For more information on the interpretations and behavior of these methods, see the SAS documentation for `PROC CLUSTER`.

It should be noted that there are several reasonable ways in which contiguity between clusters can be handled during the update process. For our macro, once two clusters H and M are deemed to be contiguous with one another, each pair of objects where one object is from H and the other is from M is implicitly treated as contiguous regardless of whether or not those objects were contiguous in the first place. However, it might be more appropriate for methods like Single Linkage to use a stronger approach when calculating the dissimilarity between two clusters. One such approach would only consider pairs of objects from H and M that were contiguous at the beginning of the clustering algorithm when the dissimilarity matrix is updated (Murtagh 1985). Of course, this would require that the original contiguity list and dissimilarity matrix be consulted throughout the entire algorithm, increasing the amounts of computer memory and processing time that would be needed. Because of this, we have employed the simpler technique here.

4. Using the macro `contigclust`

The SAS macro is called `contigclust` and is included with this article in the file `contigclustmacro.sas`. The arguments taken by the macro are summarized in Table 2.

In the following sections, we will discuss these arguments in greater detail. Since there are

Argument	Description	Usage
DistFile	The name of the data set containing the dissimilarity matrix	Required
IdVar	The name of the variable in the DistFile data set that identifies the object that corresponds to each row of the dissimilarity matrix	Required unless the values of the variables in the contiguity list are row numbers of the dissimilarity matrix
ContigFile	The name of the data set containing the contiguity list	Optional; an unconstrained analysis is performed if this is left blank
ContigVars	The names of the two variables in the ContigFile data set that give the pairs of contiguous objects	Required unless this data set only contains two variables
Method	The name of the clustering method to be used	Required
Beta	The value of the parameter β when the method is FLEXIBLE	Defaults to -0.25 if left blank
MaxSize	The maximum number of objects that a single cluster is allowed to contain	Optional
Options	The names of any options separated by spaces	Optional
OutHist	The name of the output data set that will contain the clustering history	Optional
OutTree	The name of the output data set for drawing tree diagrams	Optional

Table 2: Arguments for the macro `contigclust`.

quite a few points to keep in mind, the macro performs extensive error checking in the hopes that any problems will be discovered before the clustering algorithm begins. When an error is detected, the macro will cancel execution and print a description in the SAS log that is meant to guide the user to the error's source.

4.1. Input data sets

The user begins their call to the `contigclust` macro by inputting the name of the data set containing the dissimilarity matrix via the **DistFile** argument. The dimensions of this data set should either be $n \times (n+1)$ or $n \times n$, depending upon whether or not it includes a separate variable for identifying the objects to which each row corresponds. If it does include such a variable, then that variable's name must be supplied to the argument **IdVar**. It is important that this data set contains no other variables besides the dissimilarity variables and **IdVar** variable since the macro assumes that this is the case when it attempts to determine where the dissimilarities are stored in the data set.

It is fortunate for our purposes that SAS includes a procedure called `PROC DISTANCE` whose purpose is to create dissimilarity matrices out of raw data. It provides a large assortment of methods for computing the initial dissimilarities and is capable of handling categorical

and ordinal variables as well as regular quantitative variables. There are two aspects of this procedure that one will want to keep in mind when using it conjunction with our macro. First, if the raw data contains an appropriate `IdVar` variable, the user can include it in the dissimilarity matrix by placing its name in a `COPY` or `ID` statement in the `PROC DISTANCE` step. And second, `PROC DISTANCE` will by default produce a lower triangular matrix instead of a full square matrix in order to take advantage of the symmetry of dissimilarity matrices. Since the macro will only consider the values below the diagonal of the dissimilarity matrix anyway, this default behavior need not be changed. For more information, see the SAS documentation for `PROC DISTANCE`.

If an unconstrained analysis is desired, then the next two arguments to the macro can be omitted. Otherwise for a constrained analysis, the user must next input the name of the data set containing the contiguity list via the `ContigFile` argument. This data set should contain a row for each pair of contiguous objects and two variables giving the members of the pairs. If there are more than two variables in this data set, then the names of the two variables that contain the object pairs must be provided as a space-separated list to the argument `ContigVars`.

Although the user will want to be sure that they have entered at least the minimum amount of information into the contiguity list that is necessary for imposing the contiguity constraint, they need not be concerned about entering redundant information. The macro will remove multiple copies of object pairs from the list even if the objects appear in reverse order in some of the pairs; thus, the macro regards the pairs $(\mathbf{x}_i, \mathbf{x}_j)$ and $(\mathbf{x}_j, \mathbf{x}_i)$ as equivalent and will be unaffected by duplicate copies of either of these. Also, contiguity between an object and itself will always be disregarded; thus, the macro will always ignore pairs of the form $(\mathbf{x}_i, \mathbf{x}_i)$.

The macro requires that objects referred to in the contiguity list data set are able to be matched with those referred to in the dissimilarity matrix data set. Therefore, when object names are included in an `IdVar` variable in the `DistFile` data set, the user will need to ensure that those names are entered in exactly the same way in the `ContigFile` data set. Alternatively, if there is no `IdVar` variable in the `DistFile` data set, then the values of the object pair variables in the `ContigFile` data set must be integers corresponding to row numbers of the dissimilarity matrix. Both character and numeric forms of these row numbers are acceptable.

4.2. Setting up the analysis

The user specifies the clustering method with the argument `Method` by supplying to it one of the names listed in the first column of Table 1. If the chosen method is `FLEXIBLE`, then the user can also specify a value for the parameter β in the flexible-beta update formula using the argument `Beta`. This value can be any number and defaults to -0.25 if no value is provided.

After that, the user has the option of inputting commands to further control the clustering analysis. A positive integer provided to the `MaxSize` argument will place a limit on how many objects are allowed to inhabit a single cluster, perhaps to prevent any one cluster from becoming too large early on in the process. The use of this argument will often cause the clustering algorithm to terminate before all of the objects have been joined into one cluster.

Then option keywords can be supplied in a space-separated list to the `Options` argument. These options are summarized in Table 3. First, the `NOPRINT` option is useful when the only form of output that the user wants is one or both of the output data sets.

Option	Description
<code>NOPRINT</code>	Suppresses the display of all output
<code>NOSQUARE</code>	Prevents dissimilarities from being squared with the <code>AVERAGE</code> , <code>CENTROID</code> , <code>MEDIAN</code> , and <code>WARD</code> methods
<code>RSQ</code>	Requests that various statistics be included in output data sets with the <code>AVERAGE</code> , <code>CENTROID</code> , and <code>WARD</code> methods

Table 3: Options for the macro `contigclust`.

When certain update formulas (those for the `AVERAGE`, `CENTROID`, `MEDIAN`, and `WARD` methods) are used that are more naturally defined in terms of a Euclidean coordinate system, `PROC CLUSTER` assumes that the initial dissimilarities in the `DistFile` data set are Euclidean distances and proceeds to square them. So in keeping with the behavior of `PROC CLUSTER`, the `NOSQUARE` option is available for use with these methods to prevent this squaring. This option has no effect when any of the other clustering methods are used.

Lastly, the `RSQ` option requests that five statistics – the root-mean-square standard deviation, R-squared, semipartial R-squared, pseudo F statistic, and pseudo t-squared statistic – be included in any output data sets. These statistics, whose formulas are described in the SAS documentation for `PROC CLUSTER`, are meant to aid the user in selecting the final number of clusters when the `AVERAGE`, `CENTROID`, or `WARD` methods are used. However, their computation involves sums of squares and is based upon the assumption that the initial dissimilarities are Euclidean distances. If, in fact, the initial dissimilarities are squared Euclidean distances, then the user can prevent the macro from squaring them a second time by using the `NOSQUARE` option. The `RSQ` option is ignored when any of the other clustering methods are used and when the user does not request any output data sets.

4.3. Output data sets

Finally, the user specifies the names of any output data sets that they want the macro to create. The `OutHist` argument names the data set that will contain the clustering history with one row per iteration of the algorithm. The `OutTree` argument names the data set that will contain information for producing tree diagrams with the SAS procedure `PROC TREE`. If no name is provided for one of these data sets, it will not be created. However, the printed cluster history and tree diagram will still appear unless the `NOPRINT` option is specified.

Both output data sets will use the notation CLk where k is a positive integer as a shorthand means of referring to the cluster that was created during the iteration of the algorithm that resulted in there being k clusters (that is, k is the difference between the number of objects and the iteration number). Additionally, if no `IdVar` variable was specified, then the notation OBm where m is a positive integer will be used as a shorthand means of referring to the object that was described by the m th row and column of the initial dissimilarity matrix.

5. Example

We will work through the use of the macro with an example. Raw data as well as code for performing an example analysis are provided with this article in the file

`contigclustexamples.sas`.

Suppose that we are interested in clustering the 50 members of the United States into regions so that states with similar population densities are grouped together. This is exactly the kind of situation in which it would be natural to impose a contiguity constraint while clustering the objects. To that end, we will work with data on state population densities from the U.S. Census Bureau's Census 2000 Summary Files (see <http://factfinder.census.gov/>).

The first step is to create the initial dissimilarity matrix, a task which is easily accomplished with `PROC DISTANCE`. For this analysis, we will calculate the initial dissimilarities as Euclidean distances and identify the states with their full names. The second step is to create the contiguity list for the United States. Since there is no shortcut for doing this, the pairs of contiguous states must simply be entered manually. Code for completing both of these steps is included in the example file, and the resulting data sets are called `statedistmatrix` and `statecontiglist`, respectively. The variable `statename` serves as the identification variable for the dissimilarity matrix data set, and the variables `statename1` and `statename2` give the members of the pairs of contiguous states in the contiguity list data set.

Next, we choose a clustering method. In this case there is no reason to prefer one over another, so for illustrative purposes we will use the flexible-beta method with parameter $\beta = -1$. This instructs the macro to update dissimilarities between clusters using the rather simple formula $D_{HM} = D_{HK} + D_{HL} - D_{KL}$.

Finally, we invoke the clustering macro, supplying to it the names of our two input data sets, the names of the relevant variables in these data sets, and the details about the clustering method. This would be accomplished with the following macro call in SAS.

```
%contigclust(DistFile = statedistmatrix, IdVar = statename,
  ContigFile = statecontiglist, ContigVars = statename1 statename2,
  Method = flexible, Beta = -1);
```

The output of the macro will include a cluster history in table form like the one produced by `PROC CLUSTER` as well as a tree diagram that will display some of the same information graphically. The history is partially reproduced in Table 4 and the tree diagram is displayed in Figure 1. Notice in the tree diagram that the states of Alaska and Hawaii remain in there own clusters throughout the entire process since neither is contiguous with any other states.

Each of these descriptions of the clustering algorithm's iterations will continue until the last possible step when the three connected components of the United States map have all become there own clusters. Of course for applications, one will want to choose some small number of clusters to use for data summarization or further analysis. If we stop at, say, 13 clusters, then the result would look like the map shown in Figure 2. The cluster numbers in that map have been assigned in order of increasing average cluster population density. Code for producing a similar map and others corresponding to different numbers of clusters and different clustering methods is included in the example file.

Since this clustering includes two clusters of ten or more states, it might be reasonable to prevent the algorithm from forming such large clusters. If, for instance, we wanted to limit cluster sizes to 8 states, then we could include a value for the `MaxSize` argument in our previous macro call.

Number of clusters	Clusters joined		Frequency of new cluster	Distance
49	North Dakota	South Dakota	2	0.6
48	Arkansas	Oklahoma	2	1.0
47	Montana	Wyoming	2	1.1
46	Idaho	Nevada	2	2.6
45	Ohio	Pennsylvania	2	3.3
44	Georgia	Tennessee	2	3.4
43	Indiana	Michigan	2	5.5
42	Colorado	Kansas	2	8.6
41	Iowa	Minnesota	2	9.4
40	South Carolina	CL44	3	9.6
39	Utah	CL42	3	11.4
38	CL49	CL47	4	13.5
⋮	⋮	⋮	⋮	⋮
3	CL6	CL4	48	93123.0

Table 4: Example cluster history.

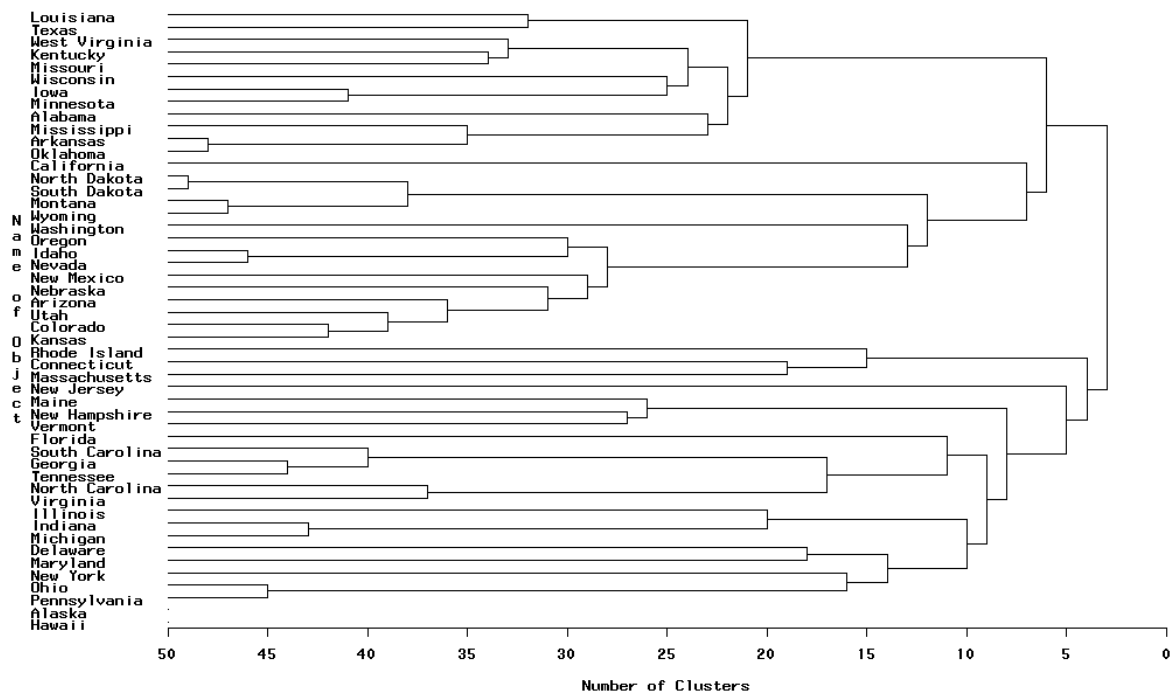


Figure 1: Example tree diagram.

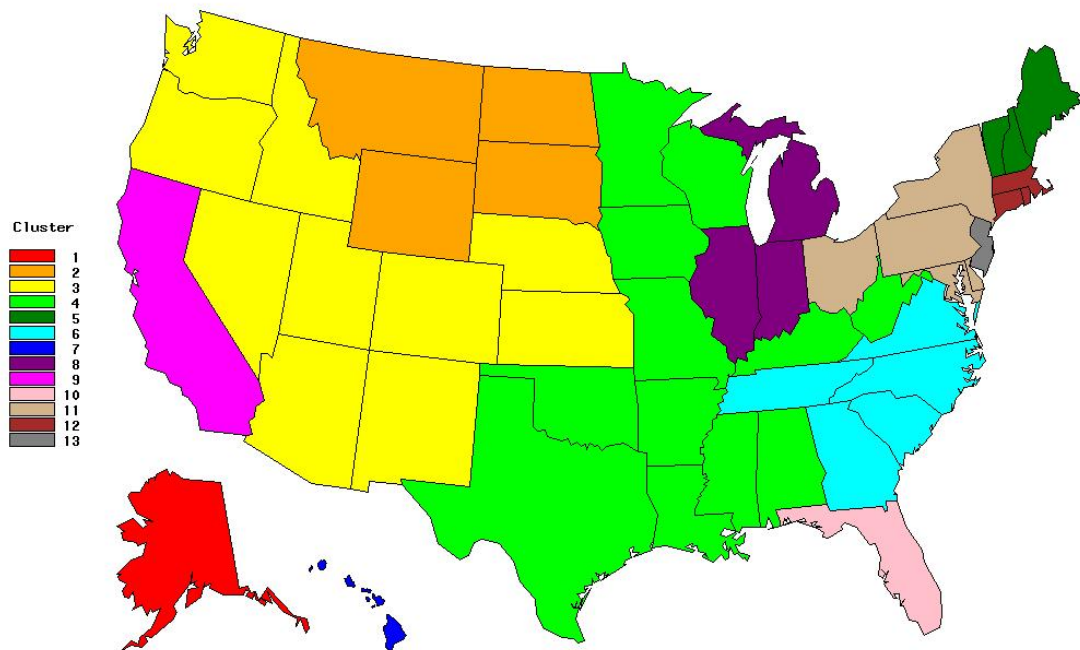


Figure 2: Example clustering with contiguity constraint.

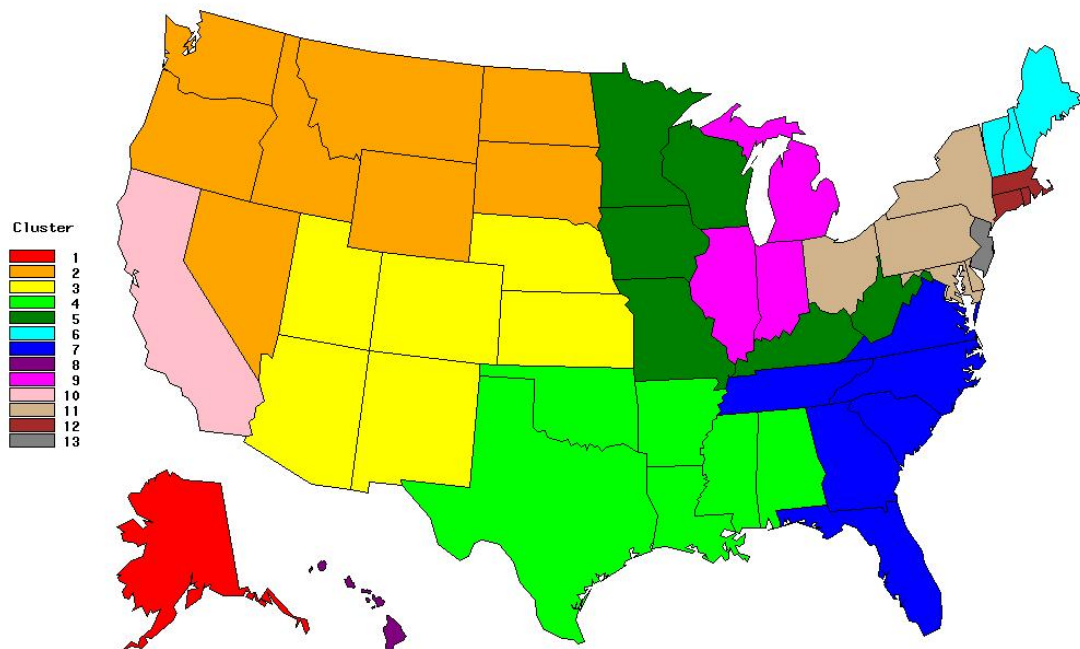


Figure 3: Example clustering with contiguity constraint and cluster size limit.

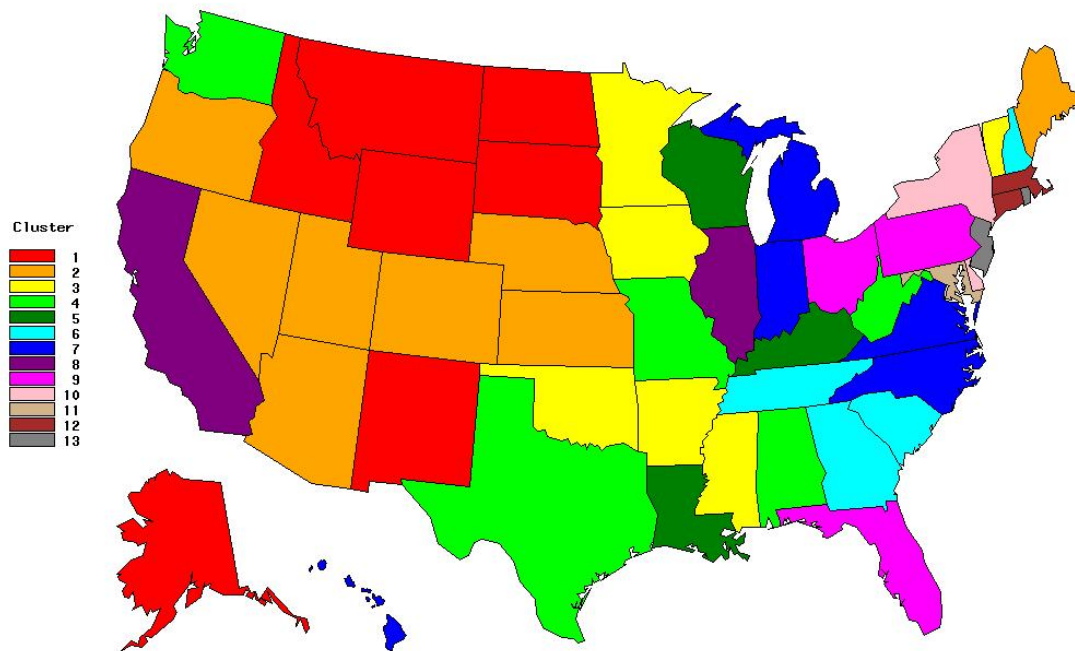


Figure 4: Example clustering without contiguity constraint.

```
%contigclust(DistFile = statedistmatrix, IdVar = statename,
  ContigFile = statecontiglist, ContigVars = statename1 statename2,
  Method = flexible, Beta = -1, MaxSize = 8);
```

With 13 clusters, the resulting map looks like Figure 3. Again, the cluster numbers in the map have been assigned in order of increasing average cluster population density. Notice that while some of the clusters are exactly the same as they were before, the new largest cluster does indeed contain 8 states.

For purposes of comparison, it might also be interesting to look at how the clustering would have turned out without any constraints or limits. This analysis is easy to perform either with PROC CLUSTER or with the following macro call.

```
%contigclust(DistFile = statedistmatrix, IdVar = statename,
  Method = flexible, Beta = -1);
```

In this case, the map looks like Figure 4. Here we see that although some contiguous states have been merged into the same clusters, most of the clusters contain multiple discontinuous regions.

6. Conclusion

We have presented here the SAS macro `contigclust` for performing contiguity-constrained hierarchical agglomerative clustering and illustrated its use by applying it to population den-

sity data for the United States. Future work might include implementation of some of the more sophisticated density-based update methods that are available in PROC CLUSTER.

Acknowledgments

I wish to thank John I. Marden of the Department of Statistics, University of Illinois, for his outstanding instruction in multivariate analysis and Maria E. Muyot of the Department of Statistics, University of Illinois, for her outstanding instruction in SAS programming.

References

- Gordon AD (1996). "A Survey of Constrained Classification." *Computational Statistics & Data Analysis*, **21**, 17–29.
- Murtagh F (1985). "A Survey of Algorithms for Contiguity-Constrained Clustering and Related Problems." *The Computer Journal*, **28**(1), 82–88.
- SAS Institute Inc (2003). *The SAS System, Version 9.1*. Cary, NC. URL <http://www.sas.com/>.

Affiliation:

Anthony Recchia
Department of Statistics
University of Illinois
725 S. Wright St.
Champaign, Illinois 61820, United States of America
E-mail: recchia@illinois.edu