THE UNIVERSITY OF MINNESOTA

Statistics 5401 December 9, 2005

Cluster Analysis in MacAnova

MacAnova function **cluster()** performs a hierarchical cluster analysis of objects by sequential agglomeration of objects into larger and larger clusters.

There are several agglomerating methods available, the default being *average linkage*. Others include single linkage, complete linkage, and Ward's method.

cluster() can work with any of the following types of information:

1. A n by p data matrix $\mathbf{X}$ whose rows represent values for p variables describing an object.
2. A n by n matrix $\mathbf{D} = [d_{ij}]$ of *dissimilarity* measures or distances between n objects.
2. A n by n matrix $\mathbf{S} = [s_{ij}]$ of *similarity* measures.

For obvious reasons, **cluster()** requires $n \geq 2$.

By default, **cluster()** prints a *class table* and a crude *dendrogram* or cluster *tree*. The class table is a table of cluster membership with one line per case or object. The join points of the dendrogam are labelled with the value of the *criterion*, the distance or similarity between the clusters merged at that level.

There are several optional arguments, indicated by "**...**" in argument lists.

Data matrix input

cluster(x, ...), where x contains data matrix $\mathbf{X}$ first standardizes the columns of x so as to remove dependence on the scales of the variables. It subtracts column means and divides each column by the standard deviation. To suppress standardization, use **cluster(x, standard:F, ...)**.

Let $\mathbf{x}_1'$, $\mathbf{x}_2'$,..., $\mathbf{x}_n'$ be the rows of the standardized matrix (rows of $\mathbf{X}$ if not standardized). Then for every $i \neq j$, **cluster()** computes squared Euclidian distance $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 = \sum_{1\leq k\leq p}(x_{ki}-x_{kj})^2$ between $\mathbf{x}_i$ and $\mathbf{x}_j$. **cluster()** then uses a hierarchical agglomerating algorithm to form clusters using these distances. The objects being clustered are the cases corresponding to the rows of $\mathbf{X}$.

Dissimilarity or similarity matrix input

cluster(dissim:d), where d is an n by n matrix with $n \geq 2$, uses the elements **d[i,j]**, with **j > i** of the *upper* triangle half of d as a *dissimilarity* measure or distance. Each row (and column) of d corresponds to an object being clustered, and **d[i,j]** is the dissimilarity or distance between objects **i** and **j**.

Computationally, matrix **d** is treated as if it were Euclidian distance. This means, that if **d[i,j]** = $\sqrt{\{\|\mathbf{x}_i - \mathbf{x}_j\|^2\}}$, then **cluster(dissim:d)** produces the same clustering as **cluster(x,standard:F, ...)**.

cluster(similar:s, ...), where **s** is an n by n matrix with $n \geq 2$, uses the upper triangle of s as a *similarity* matrix. That is, **s[i,j]** is a measure of the similarity between objects **i** and **j**.

cluster(similar:s) is computationally equivalent to **cluster(dissim:d))** where **d** is computed by **d <- 2*max(vector(s))-s**. The use of a similarity matrix is particularly useful when the objects being clustered are variables rather than cases. In that case you might use r_{ij} or $|r_{ij}|$ to measure the similarity of variables X_i and X_j, where r_{ij} is their sample correlation. When **s[i,j]** is r_{ij}, **2*(max(vector(s))-s)** is the matrix of Euclidean distances $\|\tilde{\mathbf{X}}_i - \tilde{\mathbf{X}}_j\|$ between the vectors of standardized variables.

Optional keywords used in **cluster()**

You control the behavior of **cluster()** using keywords, as summarized in this table:

Keyword	*Possible values*	*Default value*	*Meaning*
nclust	2 ≤ integer ≤ nrows(x)	min(9,nrows(x))	Number of clusters to summarize
method	"average","complete", "single", "median", "ward", "mcquitty", "centroid"	"average"	Agglomerative clustering method
keep	"classes", "crit", "distance", "all"	None	Items to be returned as value
print	F T	F with **keep** T without **keep**	Stops printing Forces printing
tree	F T	F with **keep** T without **keep**	No dendrogram Forces dendrogram
classes	F T	F with **keep** T without **keep**	No class table Forces class table
standard	F	T	Standardize data before clustering
reorder	T	F	Reorder rows of printed class table

Saving results

By default, the only action of **cluster()** is to print results; it returns **NULL** as a value. You use keyword **keep** to ensure that some of the results are saved in a MacAnova variable. The value of **keep** should be one of **"classes"**, **"crit"**, **"distance"**, or **"all"**, or a vector of more than one, for example, **keep:vector("crit","classes")**.

keep:"all" is equivalent to **keep:vector("crit", "classes", "distance")**.

When you use **keep** to save just one item, **cluster()** returns the item as a vector or matrix. When you save more than one item, **cluster()** returns a structure with components **distances**, **classes** and/or **criterion**.

When you use **keep**, printed output is normally suppressed. You can still force printing of the class table or the dendrogram by **class:T** or **tree:T**, respectively; **print:T** forces both to be printed.

Reordering rows of the class table

You can using `reorder:T` to change the order the rows of the printed class table so that objects (cases) in the same cluster are together. This does *not* affect the ordering of rows in the class table that is returned as a value when `keep:"class"` is an argument.

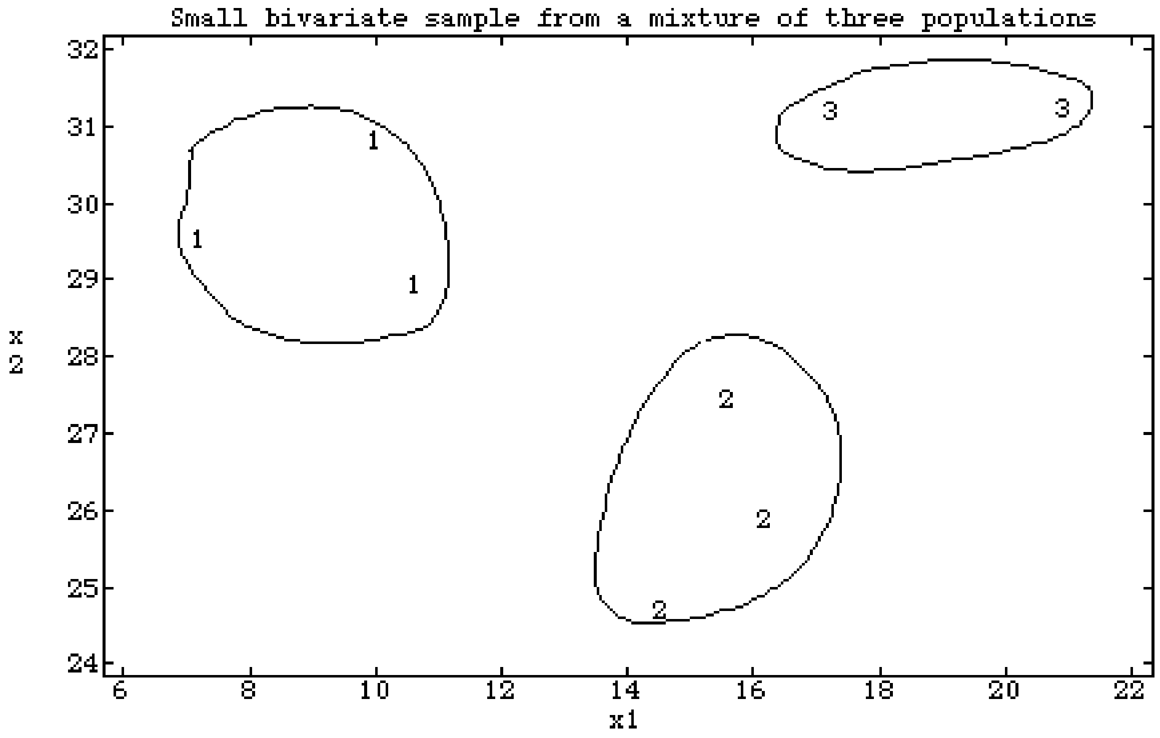
A simple example

The output from `cluster()` is not easy to understand, and is probably best illustrated by its use with a small set of data whose structure is clear. The data is generated as random samples of size 3, 3, and 2 from bivariate normal populations with means $\boldsymbol{\mu}_1$ = [10, 30]', $\boldsymbol{\mu}_2$ = [15, 28]', and $\boldsymbol{\mu}_3$ = [20, 31]', and variance matrices $\boldsymbol{\Sigma}_1 = 4\mathbf{I}_2$, $\boldsymbol{\Sigma}_2$ = diag[1^2, 1.5^2], and $\boldsymbol{\Sigma}_3$ = diag[1.5^2, 0.5^2]. I reorder rows of x using vector `permute` as a subscript so that data from different populations are scrambled.

```
Cmd> setseeds(67871,32211)# done to allow recreation of results
Cmd> x1 <- vector(10+2*rnorm(3),15+rnorm(3),20+1.5*rnorm(2))
Cmd> x2 <- vector(30+2*rnorm(3),28+1.5*rnorm(3),31+.5*rnorm(2))
Cmd> permute <- vector(5,3,2,8,6,1,7,4) # selector to mix up data
Cmd> x1 <- x1[permute]; x2 <- x2[permute]
Cmd> groups <- vector(1,1,1,2,2,2,3,3)[permute] # group ID
Cmd> x <- hconcat(x1,x2) # bivariate data matrix
Cmd> hconcat(groups,x) # population number and bivariate data
(1,1)           2      15.606      27.451
(2,1)           1      7.2295       29.53
(3,1)           1      9.9958      30.821
(4,1)           3      17.241       31.21
(5,1)           2      16.212      25.889
(6,1)           1      10.644      28.937
(7,1)           3      20.954      31.244
(8,1)           2      14.528      24.695
```

In an actual cluster analysis, only x1 and x2 would be available, not the group number.

```
Cmd> plot(x1,x2,symbols:groups,ymin:24,ymax:32,xmin:6,xmax:22,\
  title:"Small bivariate sample from a mixture of three populations")
```



```
Cmd> xs <- standardize(x) # standardize x

Cmd> describe(xs,mean:T,stddev:T) # Check standardization
component: mean
(1) -3.3827e-16 -7.2164e-16
component: stddev
(1)          1           1

Cmd> # Create matrix to hold Euclidean distances

Cmd> n <- nrows(x); d <- dmat(n,0) # n by n matrix of 0

Cmd> # Fill it with distance

Cmd> for(i,1,n-1){ # loops over 1 <= i < j <= 8
        for(j,i+1,n){
            d[i,j] <- d[j,i] <- sqrt(sum(vector(xs[i,]-xs[j,])^2));;
        }
    }

Cmd> print(format:"7.3f",d)
d: Matrix of distances among the cases for standardized data
(1,1)   0.000   2.052   1.845   1.551   0.641   1.260   1.936   1.132
(2,1)   2.052   0.000   0.807   2.340   2.485   0.801   3.148   2.536
(3,1)   1.845   0.807   0.000   1.629   2.418   0.770   2.459   2.658
(4,1)   1.551   2.340   1.629   0.000   2.147   1.735   0.831   2.683
(5,1)   0.641   2.485   2.418   2.147   0.000   1.746   2.396   0.610
(6,1)   1.260   0.801   0.770   1.735   1.746   0.000   2.486   1.910
(7,1)   1.936   3.148   2.459   0.831   2.396   2.486   0.000   2.995
(8,1)   1.132   2.536   2.658   2.683   0.610   1.910   2.995   0.000
Pop #    2       1       1       3       2       1       3       2
```

```
Cmd> cluster(x) # cluster using Average Linkage (default)
 Case  Number of Clusters
 No.   2  3  4  5  6  7  8
 ---- -- -- -- -- -- -- --
   1   1  1  1  1  1  1  1
   2   1  3  3  3  3  3  3
   3   1  3  3  3  6  6  6
   4   2  2  2  2  2  2  2     Class  table
   5   1  1  4  4  4  4  4
   6   1  3  3  3  6  7  7
   7   2  2  2  5  5  5  5
   8   1  1  4  4  4  4  8
    Criterion        Dendrogram
                 +
      2.3442     +-----------------+
      2.1452     +--------+        |
     0.91973     +--+     |        |
     0.83119     |  |     |        +--+
     0.80388     |  |     +--+     |  |
     0.76965     |  |     |  +--+  |  |
     0.60954     |  +--+  |  |  |  |  |
 Cluster No.     1  4  8  3  6  7  2  5
                 Clusters 1 to 8 (Top 7 levels of hierarchy).
                 Clustering method: Average linkage
                 Distance: Euclidean (standardized)
```

Note the argument to `cluster()` is x, not xs, because `cluster()` automatically standardizes. `cluster(xs,standard:F)` would produce the same output.

The class table

The first part of the output is the *class table*, a table of cluster membership. Each row corresponds to a case. Each column corresponds to a stage in the agglomerative algorithm, with earlier stages at the right. The column heading is the number of clusters at that stage.

The entry in the row labeled i and the column labeled j is the ID number of the cluster to which case i is assigned at the stage at which there are exactly j clusters, that is at stage n - j = 8 - j in the merging process. Stage 0 corresponds to the start when there are n = 8 "clusters" of size 1. Stage n-1 = 7 is the final stage at which there is j = 1 cluster of size n. It is *not* in the class table, but if it were, it would consist of n 1's. It is represented by the topmost "+" in the dendrogram.

When two clusters merge, the ID of the merged cluster is the minimum of the ID's of the two clusters. Because there are only 8 cases in this example, the last column, with heading "8" because there are 8 clusters, corresponds to stage 0, before any agglomeration, with ID's 1 through 8 corresponding to each case in a non-obvious order. If I had used **nclust:7**, say, as an argument, the last column would have heading "7" and would correspond to stage 1, the stage with 7 clusters after the first merge.

Relationship of case number and cluster ID

In interpreting the dendrogram, a common mistake is to interpret cluster ID number as if it were related to case number. This is not correct. There is no obvious correspondence between the case number and the ID, but there is a logic to the numbering.

Cluster ID numbers are assigned *after* the clustering process is completed in such a way that at each stage, the cluster with the highest ID (equal to the number of clusters at that stage) is merged with another cluster. This can be seen clearly in the following step by step listing of the clusters. The numbers in {...} are the case numbers in each cluster.

Clus ID	1	2	3	4	5	6	7	8
Stage 0	{1}	{4}	{2}	{5}	{7}	{3}	{6}	{8}
Stage 1	{1}	{4}	{2}	{5,8}	{7}	{3}	{6}	
Stage 2	{1}	{4}	{2}	{5,8}	{7}	{3,6}		
Stage 3	{1}	{4}	{2,3,6}	{5,8}	{7}			
Stage 4	{1}	{4,7}	{2,3,6}	{5,8}				
Stage 5	{1,5,8}	{4,7}	{2,3,6}					
Stage 6	{1,2,3,5,6,8}	{4,7}						
Stage 7	{1,2,3,4,5,6,7,8}							

Initially (at stage 0) there are 8 clusters, each consisting of a single case. To get to stage 1, cases 5 and 8 (clusters **4** and **8**) merged to form a new cluster **4** with no cluster labelled **8** remaining. At the next step, going from 7 to 6 clusters, cases 3 and 6 (clusters **6** and **7**) merged to form a new cluster **6** with no cluster labelled **7** remaining. A step 3, a partition of the data into 5 clusters is accomplished by merging case 2 (cluster **3**) and cases 3 and 6 (cluster **6**) to form a new cluster **3** consisting of cases 2, 3, and 6. And so on until the final stage (not included in the printed class table) when there is one cluster consisting of all the cases. The way I am counting the stages, there are n - k clusters at stage k.

Because these data are artificial, we know in advance that the "correct" number of clusters is 3. The partition of cases when there are 3 clusters

(Stage 5 above) shows that **cluster()** has correctly grouped the cases, with clusters **1**, **2**, and **3** corresponding exactly to the initial groups 2, 3, and 1. It would have been a meaningless coincidence if the cluster numbers had matched the group numbers.

Interpreting the dendrogram

The dendrogram below the class table summarizes the final steps of agglomeration in graphical form. Agglomeration starts at the bottom when there are 8 (in general, **nclust**) clusters. The numbering on the bottom corresponds to the cluster ID numbers in the right most column in the class table, *not to case numbers*.

As you move up in the dendrogram, each level corresponds to a merging event as agglomeration goes to the next stage. The printed value of the criterion is either the dissimilarity or similarity between the clusters merged at that level. Thus .60954 (see the table of distance between cases above) rounds to .610 = $\|\mathbf{x}_5-\mathbf{x}_8\|$, the distance between clusters **4** = {5} and **8** = {8}. Here $\mathbf{x}_i$ is a vector of standardized variables.

At the next level up, the criterion = .76965 = $\|\mathbf{x}_3-\mathbf{x}_6\|$, the distance between clusters **6** = {3} and **7** = {6}. Up an additional level, .80388 is (because we are using the average linkage method) the square root of the average squared distances between cases in cluster **3** = {2} and cases in cluster **6** = {3, 6}, that is

$$\{(\|\mathbf{x}_2-\mathbf{x}_3\|^2 + \|\mathbf{x}_2-\mathbf{x}_6\|^2)/2\}^{1/2} = \sqrt{}\{(.807^2 + .801^2)/2\} = 0.804 \approx .80388.$$

The top level at which the final two clusters (**1** and **3**) are merged into 1, has criterion value 2.3442 is the square root of the average squared distances between cases in cluster {1,2,3,5,6,8} and those in {4,7}.

```
Cmd> J1 <- vector(1,2,3,5,6,8); J2 <- vector(4,7) # selectors

Cmd> sqrt(sum(vector(d[J1,J2]^2))/12)
(1)       2.3442
```

When **order:T** is an argument, the printed class table is reordered so that cases in the same cluster are adjacent. Here is **cluster()** output when **reorder:T** is used:

```
Cmd> cluster(x,reorder:T)
 Case  Number of Clusters
 No.   2  3  4  5  6  7  8
 ---- -- -- -- -- -- -- --
   1   1  1  1  1  1  1  1
   5   1  1  4  4  4  4  4
   8   1  1  4  4  4  4  8
   2   1  3  3  3  3  3  3     Reordered  class  table
   3   1  3  3  3  6  6  6
   6   1  3  3  3  6  7  7
   4   2  2  2  2  2  2  2
   7   2  2  2  5  5  5  5

    Criterion
                 +
       2.3442    +-----------------+
       2.1452    +--------+        |
      0.91973    +--+     |        |
      0.83119    |  |     |        +--+
      0.80388    |  |     +--+     |  |
      0.76965    |  |     |  +--+  |  |
      0.60954    |  +--+  |  |  |  |  |
  Cluster No.    1  4  8  3  6  7  2  5
                 Clusters 1 to 8 (Top 7 levels of hierarchy).
                 Clustering method: Average linkage
                 Distance: Euclidian (standardized)
```

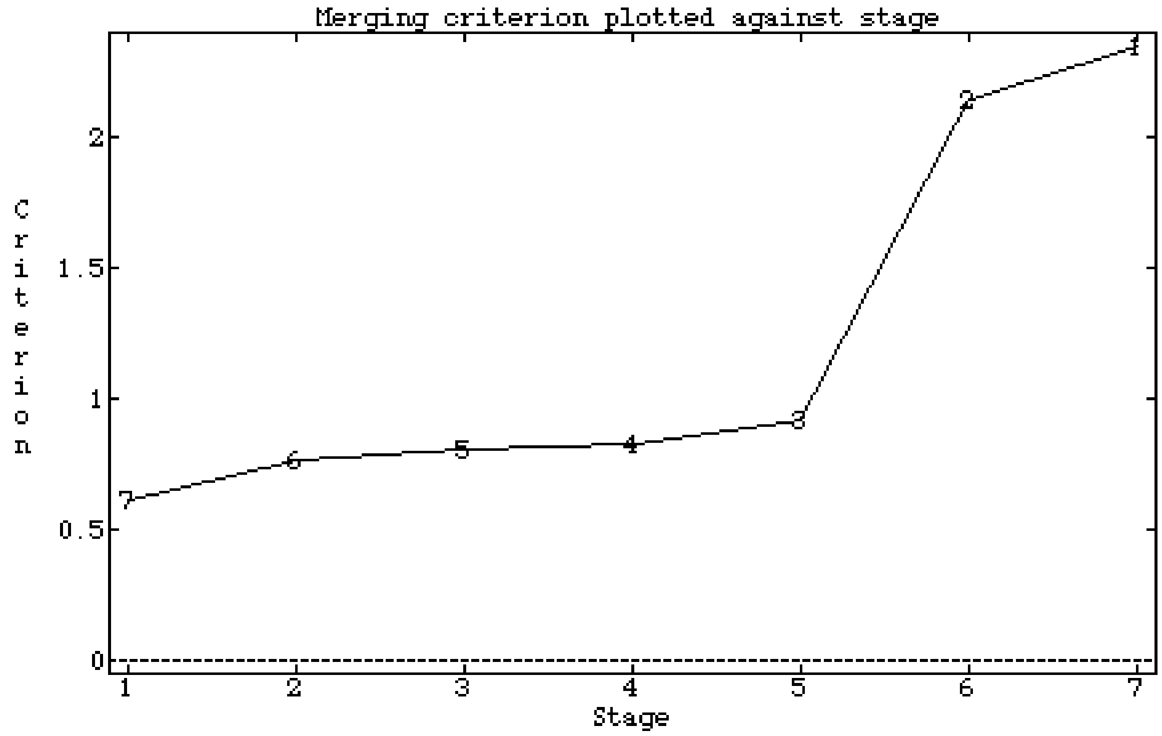
The dendrogram is unchanged, but the rows of the class table have been reordered so that, at every stage, all the cases in each cluster are adjacent.

Graphical clues to the number of clusters

The values of the criterion offer clues as to how many clusters you actually may have. Using `keep:"crit"`, you can save these values for use in making various plots. I used `reverse()` to order them from first stage (bottom of tree) to last (top of tree) so they are increasing. The last value is the distance between the last two clusters that are merged.

```
Cmd> criterion <- reverse(cluster(x,keep:"crit")); criterion
(1)     0.60954     0.76965     0.80388     0.83119     0.91973
(6)      2.1452      2.3442
```

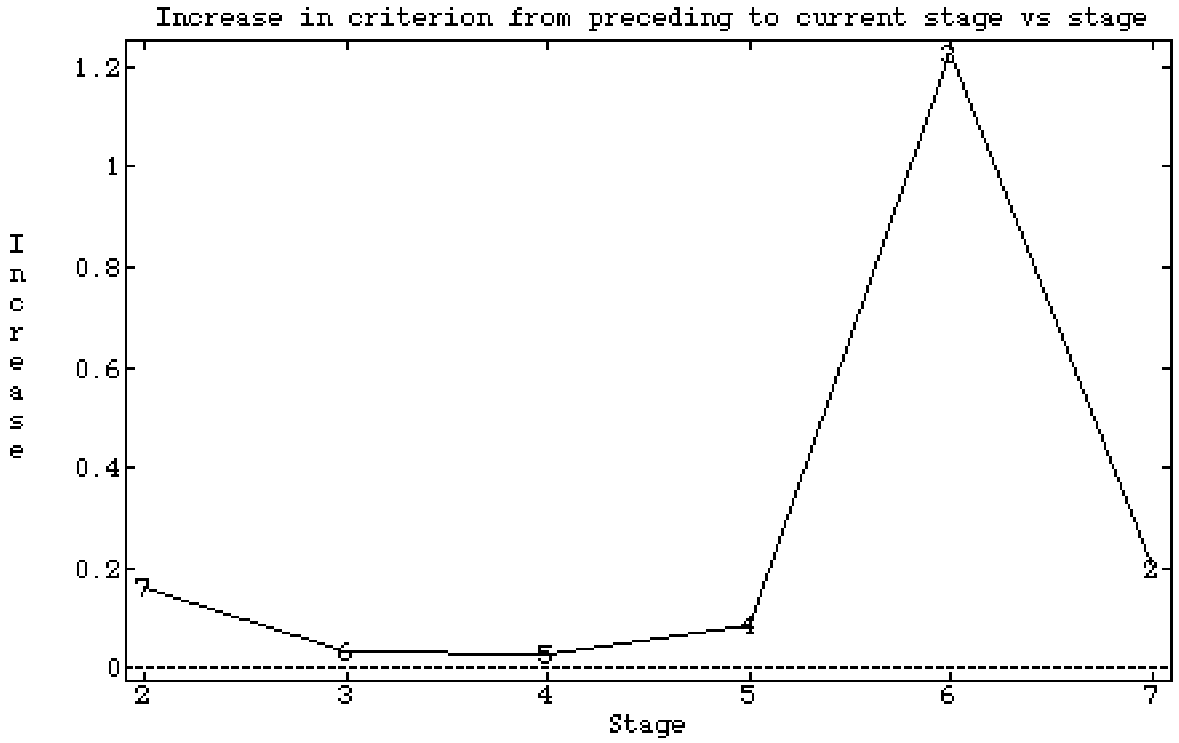
```
Cmd> stages <- run(7) # stage or merge number,
Cmd> clustersleft <- 8 - stages # number of clusters at stage
Cmd> lineplot(stages,criterion,symbols:clustersleft,ymin:0,\
       xlab:"Stage", ylab:"Criterion",\
       title:"Merging criterion plotted against stage")
```



Stage i is the result of the ith merge which transitions from 8 - i + 1 clusters to 8 - i clusters. The points are labelled with the number of clusters after the merging (8 - i). There is a substantial jump in the value of the criterion (distance between clusters) when we go from the "correct" number of clusters (3) at stage 5, to too few clusters (2) at stage 6. This is the sort of evidence that you are looking for in attempting to decide on how many clusters there are.

A better way to view this same information is to plot the *changes* in the criteria between successive stages. A large change indicates the merging of two clusters that are quite far apart compared to previous merges and hence possibly ought not be merged.

```
Cmd> J <- stages[-1]; changes <- criterion[J] - criterion[J-1]
Cmd> lineplot(J,changes,symbols:clustersleft[-1]+1,ymin:0,\
    xlab:"Stage", ylab:"Increase", title:\
"Increase in criterion from preceding to current stage vs stage")
```



The quantity plotted is the increase between the criterion achieved at a given stage and the criterion achieved at the preceding stage. There is a large change between the criterion at stage 5 (8 - 5 = 3 clusters) to stage 6 (8 - 6 = 2 clusters), indicating the merger of two clusters that were relatively far apart compared to the distance between clusters merged at previous stages. The labels of the plotted values are the number of clusters before the merge. Thus, the fact that the peak is labeled 3 suggests three clusters might be appropriate.

Effect of specifying a smaller number of clusters

Often only the output corresponding to the last few stages is of interest. When **nclust:m** is an argument, the bottom of the dendrogram and the right most column of the class table correspond to stage n - m when there are m clusters. Nothing is printed about stages with more than m clusters.

```
Cmd> cluster(x,nclust:4)
 Case  Number of Clusters
 No.   2  3  4
 ---- -- -- --
   1   1  1  1
   2   1  3  3
   3   1  3  3
   4   2  2  2
   5   1  1  4
   6   1  3  3
   7   2  2  2
   8   1  1  4
    Criterion
                +
       2.3442  +--------+
       2.1452  +-----+  |
      0.91973  +--+  |  |
  Cluster No.  1  4  3  2
               Clusters 1 to 4 (Top 3 levels of hierarchy).
               Clustering method: Average linkage
               Distance: Euclidian (standardized)
```

The three columns of the class table are the same as the first three columns of the full class table, and the dendrogram gives the structure of the top few lines of the full dendrogram, with empty columns squeezed out. That is, cluster **4** is already the merger of clusters **5**, and **8**, cluster **3** is already the merger of cluster **3**, **6**, and **7**, and cluster **2** is already the merger of original clusters **2**, and **5**.

nclust:m also affects what is returned when you use keyword **keep**. The class table and criterion vector pertain only to the results from stage n - m on.

```
Cmd> cluster(x,nclust:4,keep:vector("class","crit"))
component: classes
(1,1)          1           1           1
(2,1)          1           3           3
(3,1)          1           3           3
(4,1)          2           2           2
(5,1)          1           1           4
(6,1)          1           3           3
(7,1)          2           2           2
(8,1)          1           1           4
component: criterion
(1)      2.3442      2.1452     0.91973
```

Cluster analysis of a larger data set

Here is an application of cluster to the data in file cbspots.txt, transformed to $\log_{10}(1 + y)$.

```
Cmd> spots <- read("","spots") # read from cbspots.txt
spots          50     20 format labels
) Density measurements on 19 identifiable spots on each of 50 electro-
) phoretic gels, each spot corresponding to a particular (probably
) unknown) protein.
)
) The data in each row was derived from the blood of a rat subjected to
) a treatment expected to affect its thyroid hormones.
) There were 10 treatments in all, including a control (treatment 2).
)
) Col. 1: trt = treatment number (1-10)
) Col. 2-20: density measurements on spots 1-19.
)
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
)
Read from file "TP1:Stat5401:Data:cbspots.txt"
Cmd> groups <- vector(spots[,1],labels:NULL)
Cmd> print(format:"2.0f",Case_No:run(50),Group_No:groups)
Case_No:
 (1)  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
(23) 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
(45) 45 46 47 48 49 50
Group_No:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3
(23)  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8  9  9
(45)  9  9 10 10 10 10
Cmd> y <- log10(1 + spots[,-1]) # transform data
```

Here is output from cluster(). I have added a column to the class table giving the actual group membership (**bold**).

The sample size is 50 which means that the complete dendrogram would have 50 roots and the full class table would be 50 by 49. We are interested in the final stages so I used nclust:20.

```
Cmd> cluster(y,nclust:20,reorder:T) # cases will be reordered
 Case  Number of Clusters
 No.    2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 ---- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
   1    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
   2    1  1  1  1  1  1  8  8  8  8  8  8  8  8  8  8  8  8  8  1
   4    1  1  1  1  1  1  8  8  8  8  8  8  8  8  8  8  8  8 20  1
   3    1  1  1  1  1  1  8  8  8 11 11 11 11 11 11 11 11 11 11  1
   5    2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
   6    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
   7    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
   8    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
   9    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  10    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  11    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  12    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  13    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  14    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  15    2  2  2  2  2  2  2  2  2  2  2  2  2 15 15 15 15 15 15  2
  43    2  2  2  2  2  2  2  2  2  2 12 12 12 12 12 12 12 12 12  9
  44    2  2  2  2  2  2  2  2  2  2 12 12 12 12 12 12 12 12 12  9
  45    2  2  2  2  2  2  2  2  2  2 12 12 12 12 12 12 12 12 12  9
  46    2  2  2  2  2  2  2  2  2  2 12 12 12 12 12 12 12 12 12  9
  47    2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9 10
  48    2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9 10
  49    2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9 10
  50    2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9 10
  23    2  2  2  2  2  7  7  7  7  7  7  7  7  7  7  7  7  7  7  4
  25    2  2  2  2  2  7  7  7  7  7  7  7  7  7  7  7  7  7  7  4
  24    2  2  2  2  2  7  7  7  7  7  7  7 14 14 14 14 14 14 14  4
  38    2  2  2  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  8
  39    2  2  2  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  8
  40    2  2  2  5  5  5  5  5  5  5  5  5  5  5 16 16 16 16 16  8
  41    2  2  2  5  5  5  5  5  5  5  5  5  5  5 16 16 16 16 16  8
  42    2  2  2  5  5  5  5  5  5  5  5  5  5  5 16 16 16 16 16  8
  16    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3
  17    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3
  18    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3
  19    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3
  21    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3
  29    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4 17 17 17 17  5
  30    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4 17 17 17 17  5
  31    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4 17 17 17 17  5
  32    2  2  4  4  4  4  4  4  4  4  4  4  4  4  4 17 17 17 17  5
  20    2  2  4  4  4  4  4  4  4  4  4 13 13 13 13 13 13 13 13  3
  22    2  2  4  4  4  4  4  4  4  4  4 13 13 13 13 13 13 13 13  3
  26    2  2  4  4  4  4  4  4 10 10 10 10 10 10 10 10 10 10 10  5
  27    2  2  4  4  4  4  4  4 10 10 10 10 10 10 10 10 10 10 10  5
  37    2  2  4  4  4  4  4  4 10 10 10 10 10 10 10 10 10 10 10  7
  36    2  2  4  4  4  4  4  4 10 10 10 10 10 10 10 10 10 19 19  7
  33    2  2  4  4  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  35    2  2  4  4  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  34    2  2  4  4  6  6  6  6  6  6  6  6  6  6  6  6 18 18 18  6
  28    2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  5
```

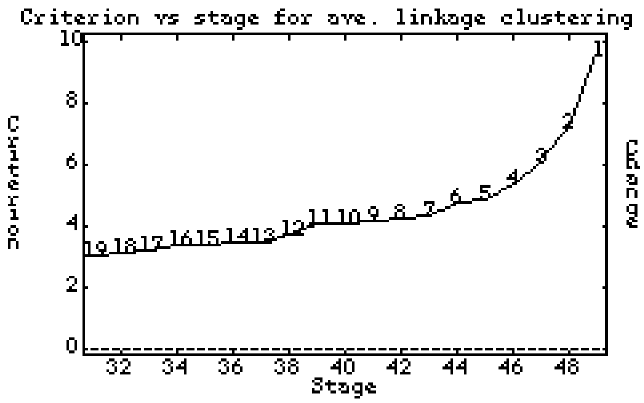
```
Criterion
            +
    9.5263  +-----------+
    7.2292  |           +---------------------------------------------+
    6.1127  |           +-----------------------+                    |
    5.3389  |           +-----------------+     |                    |
    4.8421  |           |                 |     +--------------+     |
    4.7849  |           +-----------+     |     |              |     |
    4.3116  +--+        |           |     |     |              |     |
    4.2362  |  |        +--------+  |     |     |              |     |
    4.1678  |  |        |        |  |     |     +--------+     |     |
    4.1118  |  +-----+  |        |  |     |     |        |     |     |
     4.041  |  |     |  +-----+  |  |     |     |        |     |     |
    3.7605  |  |     |  |     |  |  |     |     +-----+  |     |     |
    3.4842  |  |     |  |     |  |  +--+  |     |     |  |     |     |
    3.4541  |  |     |  +--+  |  |  |  |  |     |     |  |     |     |
    3.3835  |  |     |  |  |  |  |  |  |  +--+  |     |  |     |     |
    3.3675  |  |     |  |  |  |  |  |  |  |  |  +--+  |  |     |     |
    3.1754  |  |     |  |  |  |  |  |  |  |  |  |  |  |  |     +--+  |
     3.084  |  |     |  |  |  |  |  |  |  |  |  |  |  +--+  |  |  |
    3.0093  |  +--+  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
Cluster No. 1  8 20 11  2 15 12  9  7 14  5 16  4 17 13 10 19  6 18  3
            Clusters 1 to 20 (Top 19 levels of hierarchy).
            Clustering method: Average linkage
            Distance: Euclidian (standardized)
```

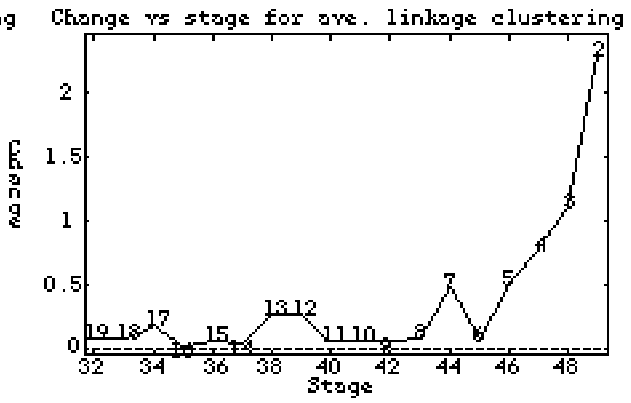
If you carefully examine the reordered class table you can see a good deal of correspondence between clusters found at various stages and the original groups. Both the dendrogram and the class table show that one cluster, **3** = {28}, remains a "singleton" consisting of a single case right up to the final merge from two clusters, distance 9.5263 apart, to one.

Let's see what the values of the criterion might suggest about the numbers of clusters. Macro `cluscritplot()` in file `mvgraphics.mac` eases the job of plotting the criterion and changes in the criterion against stage. You can provide either the vector of criterion values or a structure with a vector component `criterion` as may be returned by `cluster()`.

Here are plots of the criterion and changes of the criterion against stage similar to those above.

```
Cmd> info <- cluster(y,nclust:20,keep:vector("crit","classes"))
Cmd> N <- nrows(y) # sample size
Cmd> cluscritplot(info,N,values:T,\
         title:"Criterion vs stage for ave. linkage clustering")
Cmd> cluscritplot(info,N,change:T,\
         title:"Change vs stage for ave. linkage clustering")
```





The stages start with stage 31 when there are 50 - 31 = 19 clusters. The plot of the criterion itself does not seem very informative in terms of determining an appropriate number of clusters.

The plot of the changes may be more helpful. The quantity plotted at stage i is the change in the criterion (always an increase) found during the merge to 50-i clusters as compared to the criterion found during the merge to 50-i+1 clusters. They are labelled with the number, 50-i+1, of clusters *before* the merge. The small maximum at stage 44 labeled with 7 is weak evidence that the number of clusters is near 7. There is much stronger evidence that the number of clusters is not less that 5 because of the start of a string of rapidly increasing distances at stage 46.

Here is a comparison of the actual groups with the cluster membership when there are 10 clusters, the "correct number."

```
Cmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,9]))
groups:
 (1)   1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3
(23)   4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8  9  9
(45)   9  9 10 10 10 10
Clusters:  Average  linkage  clusters
 (1)   1  8  8  8  2  2  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4
(23)   7  7  7 10 10  3  4  4  4  4  6  6  6 10 10  5  5  5  5  5  2  2
(45)   2  2  9  9  9  9
```

There are 4 underlined clusters (**5**, **6**, **7** and **9**) that coincide with the original groups, (groups 8, 6, 4 and 10). Cluster **2** is made up of groups 2 and 9. Three out of four members of group 1 make up cluster **8**, with the remaining element a singleton (cluster consisting of a single element). As noted above, cluster **3** is a singleton, consists only of case 28. Altogether, average linkage clustering does a pretty good job of recovering the original groups.

Here is similar output from the stage when there are 7 clusters:

```
Cmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,6]))
groups:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3
(23)  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8  9  9
(45)  9  9 10 10 10 10
Clusters:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4
(23)  7  7  7  4  4  3  4  4  4  4  6  6  6  4  4  5  5  5  5  5  2  2
(45)  2  2  2  2  2  2
```

Arguably, the is a more accurate clustering than with 10 clusters. Groups 1, 4, 6 and 8 match clusters **1**, **7**, **6** and **5**. And, except for case in the group 5 that makes up the singleton cluster **3**, each of the other groups is entirely within one cluster, groups 2, 9 and 10 in cluster **2**, and groups 3, 5 and 7 in cluster **5**.

Another way to evaluate the clustering is with a "confusion table", a cross tabulation of the actual groups and the cluster numbers.

```
Cmd> table <- tabs(,groups,info$classes[,9])

Cmd> setlabels(table,structure("G","C")) # labels w row & col numbers

Cmd> print(format:"3.0f",table)
table:  Rows are groups, columns are clusters
      C1  C2  C3  C4  C5  C6  C7  C8  C9 C10
G1     1   0   0   0   0   0   0   3   0   0
G2     0  11   0   0   0   0   0   0   0   0
G3     0   0   0   7   0   0   0   0   0   0
G4     0   0   0   0   0   0   3   0   0   0
G5     0   0   1   4   0   0   0   0   0   2
G6     0   0   0   0   0   3   0   0   0   0
G7     0   0   0   0   0   0   0   0   0   2
G8     0   0   0   0   5   0   0   0   0   0
G9     0   4   0   0   0   0   0   0   0   0
G10    0   0   0   0   0   0   0   0   4   0
```

It's easier to see what's going on if you reorder the groups (rows) so that, as far as possible, the maximum count in a column is on the diagonal. It's a little tricky so I wrote macro `ordertable()` (posted to web page) to attempt it. It does a reasonably good job.

```
Cmd> ordertable <- read("","ordertable")
ordertable   macro dollars
) Macro to reorder rows and columns of a "confusion" table so that
) the diagonal dominates.
)
) Usage:
)  table <- ordertable(groups, clusters)
)  table <- ordertable(groups, classtable, nclust)
)   groups       vector of N positive integers identifying the group
)                each case belongs to
)   clusters     vector of N positive integers identifying the cluster
)                each case belongs to
)   classtable   N by m matrix of positive integers, intended to be
)                the matrix returned cluster(x,method:meth,
)                keep:"classes")
)   nclust       integer, 2 <= nclust <= ncols(classtable) + 1.  This
)                defines a vector clusters = classtable[,nclust-1]
)   table        the reordered g = max(groups) by max(clusters)
)                confusion matrix tabs(,groups, clusters)
)
) The columns of the confusion matrix is first reordered from largest
) cluster to smallest.  Then rows are reordered in stages, so as to
) put the column maxima on the diagonal as far as possible.
)
Read from file "TP1:Macros:Mulvar:ordertable.mac"
```

```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,10))
MATRIX:     Rows are groups, columns are clusters
      C2  C4  C5  C9 C10  C8  C7  C6  C3  C1
G2    11   0   0   0   0   0   0   0   0   0
G3     0   7   0   0   0   0   0   0   0   0
G8     0   0   5   0   0   0   0   0   0   0
G10    0   0   0   4   0   0   0   0   0   0
G7     0   0   0   0   2   0   0   0   0   0
G1     0   0   0   0   0   3   0   0   0   1
G4     0   0   0   0   0   0   3   0   0   0
G6     0   0   0   0   0   0   0   3   0   0
G5     0   4   0   0   2   0   0   0   1   0
G9     4   0   0   0   0   0   0   0   0   0
```

This shows the structure reasonable well. If row i and column i both have non-zero values only where they meet on the diagonal, this indicates a perfect match.

Do the same with the 7 cluster solution.

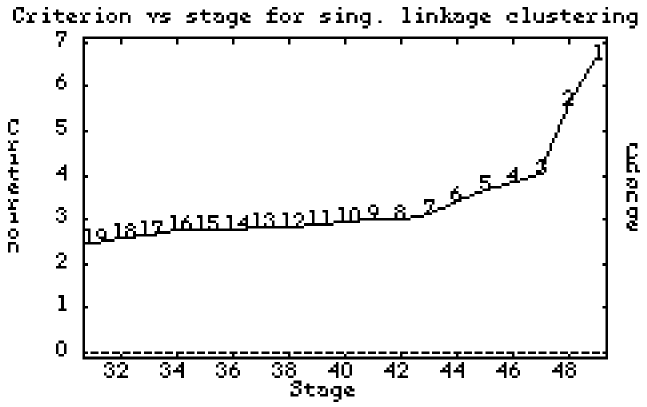
```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,7))
MATRIX:     Rows  are  groups,  columns  are  clusters
      C2  C4  C5  C1  C6  C7  C3
G2    11   0   0   0   0   0   0
G3     0   7   0   0   0   0   0
G8     0   0   5   0   0   0   0
G1     0   0   0   4   0   0   0
G6     0   0   0   0   3   0   0
G4     0   0   0   0   0   3   0
G5     0   6   0   0   0   0   1
G7     0   2   0   0   0   0   0
G10    4   0   0   0   0   0   0
G9     4   0   0   0   0   0   0
```

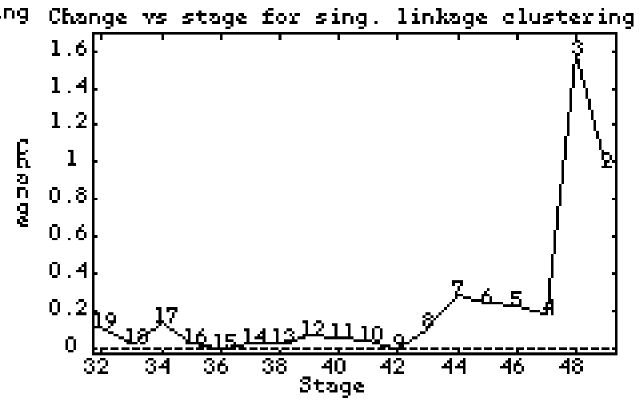
Use of single and complete linkage methods

Let's look at two other agglomerative cluster methods applied to these data, single linkage and complete linkage.

```
Cmd> info <- cluster(y,method:"single",nclust:20,tree:T,\
keep:vector("crit","classes")) #tree:T means print dendrogram
 Criterion  Single linkage dendrogram
            +
    6.6162  +-----------+
    5.6339  |           +--------------------------------------------+
    4.0571  +--------+  |                                            |
      3.87  +--+     |  |                                            |
    3.6466  |  |     |  +--------------------------+                 |
    3.4067  |  |     |  +-----------------------+  |                 |
    3.1208  |  |     |  +--------------+        |  |                 |
    3.0093  |  +--+  |  |              |        |  |                 |
    3.0056  |  |  |  |  +-----------+  |        |  |                 |
    2.9679  |  |  |  |  +--------+  |  |        |  |                 |
    2.9129  |  |  |  |  |        |  |  |        |  +--+              |
    2.8421  |  |  |  |  +--+     |  |  |        |  |  |              |
    2.8172  |  |  |  |  |  |     |  |  |        |  |  +-----+        |
    2.7926  |  |  |  |  |  |     |  |  |        |  |  +--+  |        |
    2.7916  |  |  |  |  |  |     |  |  +-----+  |  |  |  |  |        |
    2.7668  |  |  |  |  |  |     |  |  |     |  |  |  |  |  +-----+  |
    2.6297  |  |  |  |  |  +--+  |  |  |     |  |  |  |  |  |     |  |
    2.5984  |  |  |  |  |  |  |  |  |  |     |  |  |  |  |  +--+  |  |
    2.4827  |  |  |  |  |  |  |  |  |  +--+  |  |  |  |  |  |  |  |  |
Cluster No. 1  5  9  4  2 13 18 11 10  8 20 16  7  6 12 15 14 19 17  3
            Clusters 1 to 20 (Top 19 levels of hierarchy).
            Clustering method: Single linkage
            Distance: Euclidian (standardized)
```

```
   Cmd> cluscritplot(info,N,values:T,\
           title:"Criterion vs stage for sing. linkage clustering")

   Cmd> cluscritplot(info,N,change:T,\
           title:"Change vs stage for sing. linkage clustering")
```





There is no clear identification of the number of clusters although 8 (where change starts to increase), or 7 (biggest change before final push) might be a an informed guess.

```
   Cmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,9]))
   groups:
    (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3
   (23)  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8  9  9
   (45)  9  9 10 10 10 10
   Clusters:   Single  linkage  clusters
    (1)  1  5  4  9  2  2  2  2  2  2  2  2  2  2  2  6  6  6  6  6  6  6
   (23) 10  7 10  6  6  3  6  6  6  6  6  6  6  6  6  8  8  8  8  8  2  2
   (45)  2  2  2  2  2  2
```

There is only a single cluster, **8**, which coincides with an actual group. Five clusters (**1**, **3**, **4**, **5** and **7**) are singletons. Single linkage often yields lots of singletons.

Here are reordered confusion matrices for the 10 and 7 cluster solutions.

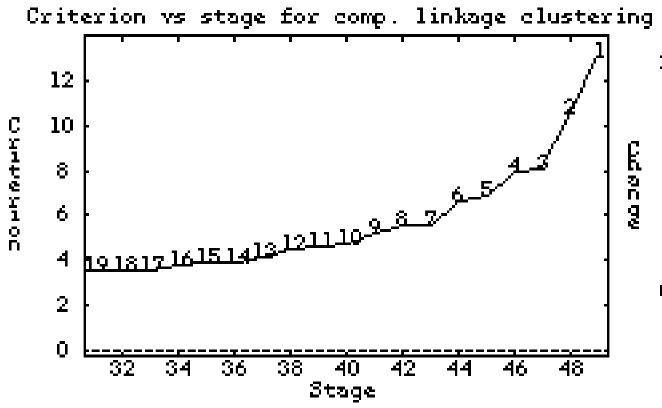
```
   Cmd> print(format:"3.0f",ordertable(groups,info$classes,10))
   MATRIX:
         C2   C6   C8  C10   C7   C9   C5   C3   C1   C4
   G2    11    0    0    0    0    0    0    0    0    0
   G3     0    7    0    0    0    0    0    0    0    0
   G8     0    0    5    0    0    0    0    0    0    0
   G4     0    0    0    2    1    0    0    0    0    0
   G7     0    2    0    0    0    0    0    0    0    0
   G1     0    0    0    0    0    1    1    0    1    1
   G6     0    3    0    0    0    0    0    0    0    0
   G5     0    6    0    0    0    0    0    1    0    0
   G10    4    0    0    0    0    0    0    0    0    0
   G9     4    0    0    0    0    0    0    0    0    0
```

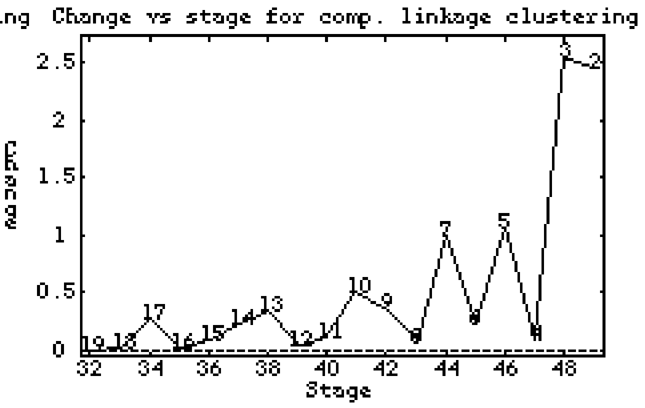
```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,7))
MATRIX:
     C2  C6  C5  C1  C4  C3  C7
G2   11   0   0   0   0   0   0
G3    0   7   0   0   0   0   0
G1    0   0   2   1   1   0   0
G9    4   0   0   0   0   0   0
G5    0   6   0   0   0   1   0
G10   4   0   0   0   0   0   0
G4    2   0   0   0   0   0   1
G6    0   3   0   0   0   0   0
G8    5   0   0   0   0   0   0
G7    0   2   0   0   0   0   0
```

Here is a complete linkage cluster analysis.

```
Cmd> info <- cluster(y,method:"complete",nclust:20,tree:T,\
        keep:vector("crit","classes"))

 Criterion    Complete  linkage  dendrogram
            +
     13.03  +--------+
    10.578  |        +--------------------+
    8.0279  |        +-----------------+  |
    7.9267  |        |                 |  +--------------------+
    6.8612  |        |                 |  +--------------+     |
    6.6204  |        |                 |  |              |     +-----+
    5.6254  |        |                 |  +--------+     |     |     |
    5.5654  |        +-----------+     |  |        |     |     |     |
     5.196  |        +--------+  |     |  |        |     |     |     |
    4.6902  +--+     |        |  |     |  |        |     |     |     |
    4.5724  |  |     |        |  |     |  +-----+  |     |     |     |
    4.5197  |  |     +--+     |  |     |  |     |  |     |     |     |
    4.1659  |  +--+  |  |     |  |     |  |     |  |     |     |     |
    3.9365  |  |  |  |  |     |  +--+  |  |     |  |     |     |     |
    3.8553  |  |  |  |  |     |  |  |  |  +--+  |  |     |     |     |
    3.8446  |  |  |  |  |     |  |  |  |  |  |  |  |     |     +--+  |
    3.5601  |  |  |  |  |     |  |  |  |  |  |  |  +--+  |     |  |  |
     3.537  |  |  |  |  |     |  |  |  |  |  |  |  |  |  +--+  |  |  |
    3.5343  |  |  |  |  +--+  |  |  |  |  |  |  |  |  |  |  |  |  |  |
Cluster No. 1 11 14  2 13 20 10  9 15  4  3 16 12  8 18  6 19  5 17  7
            Clusters 1 to 20 (Top 19 levels of hierarchy).
            Clustering method: Complete linkage
            Distance: Euclidian (standardized)
```

```
Cmd> cluscritplot(info,N,values:T,\
          title:"Criterion vs stage for comp. linkage clustering")
Cmd> cluscritplot(info,N,change:T,\
          title:"Change vs stage for comp. linkage clustering")
```





```
ΩCmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,9]))
groups:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3
(22)  3  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8
(43)  9  9  9  9 10 10 10 10
Clusters:   Complete  linkage  clusters
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3
(22)  3  8  8  8  5  5  7  3  3  3  3  6  6  6  5  5  4  4  9  9  9
(43) 10 10 10 10  9  9  9  9
```

There are 5 clusters, **1**, **2**, **6**, **8**, and **10**, that coincide with groups. These correspond to groups 1, 2, 6, 4 and 9. There is one singleton, cluster **7** consisting of case 28, in group 5.

```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,10))
      C2  C3  C9  C1  C5 C10  C6  C8  C4  C7
G2    11   0   0   0   0   0   0   0   0   0
G3     0   7   0   0   0   0   0   0   0   0
G10    0   0   4   0   0   0   0   0   0   0
G1     0   0   0   4   0   0   0   0   0   0
G5     0   4   0   0   2   0   0   0   0   1
G9     0   0   0   0   0   4   0   0   0   0
G6     0   0   0   0   0   0   3   0   0   0
G4     0   0   0   0   0   0   0   3   0   0
G8     0   0   3   0   0   0   0   0   2   0
G7     0   0   0   0   2   0   0   0   0   0
```

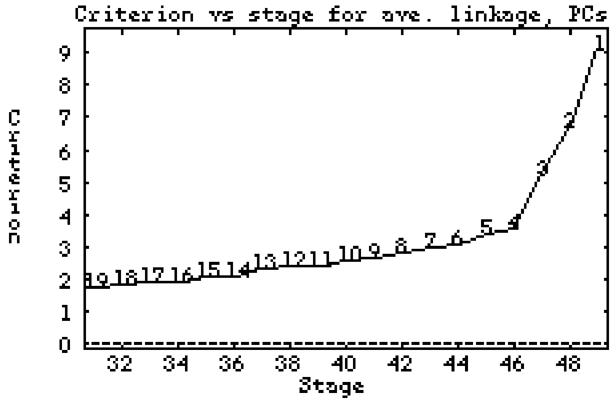
```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,7))
MATRIX:
     C2  C3  C5  C1  C6  C4  C7
G2   11   0   0   0   0   0   0
G3    0   7   0   0   0   0   0
G5    0   4   2   0   0   0   1
G1    0   0   0   4   0   0   0
G6    0   0   0   0   3   0   0
G8    3   0   0   0   0   2   0
G9    4   0   0   0   0   0   0
G4    0   3   0   0   0   0   0
G10   4   0   0   0   0   0   0
G7    0   0   2   0   0   0   0
```

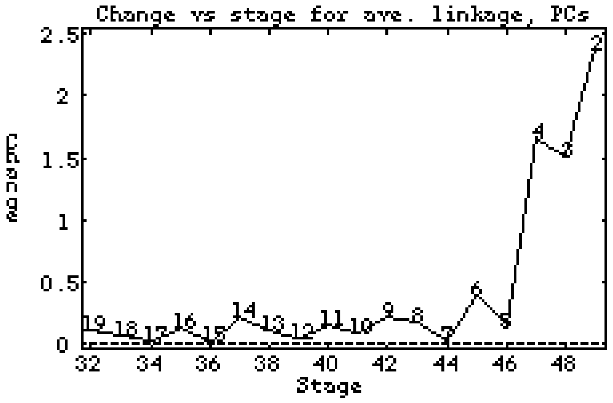
Choice of variables

The choice of variables input to a clustering algorithm can have a profound effect on its success. When you include variables that themselves do not cluster, you may dilute the effect of other variables that do cluster. Any steps that can be taken to select more informative variables will usually pay dividends. One method that is sometimes helpful is to replace the data by the first few principal components. Here we apply average linkage clustering to the first 4 correlation principal components.

```
Cmd> eigs <- eigen(cor(y)); eigs$values # 4 values > 1
 (1)      8.1623      3.7179      1.7374      1.1274     0.89645
 (6)     0.76408     0.56048     0.53438     0.36238      0.2947
(11)     0.20819     0.16422     0.12944     0.10691    0.080667
(16)    0.054977    0.050571     0.02898    0.018455

Cmd> princomps <- standardize(y) %*% eigs$vectors[,run(4)]

Cmd> describe(princomps,var:T) # variances = eigenvalues
(1)      8.1623      3.7179      1.7374      1.1274

Cmd> info <- cluster(princomps,nclust:20,\
        keep:vector("classes","crit"),standard:F) # don't standardize
```

```
Cmd> cluscritplot(info,N,values:T,\
          title:"Criterion vs stage for ave. linkage, PCs")
Cmd> cluscritplot(info,N,change:T,\
          title:"Change vs stage for ave. linkage, PCs")
```





The bump at stage 45 = 50-5, corresponding a change from 6 to 5 clusters, might suggest 6 might be an appropriate number of clusters although that is not clear.

```
Cmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,9]))
groups:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3
(22)  3  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8
(43)  9  9  9  9 10 10 10 10
Clusters: Principal components average linkage clusters
 (1)  1  1  9  1  2  2  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4
(22)  4  7  8  2  5  5  5 10 10 10 10 10 10 10  5  5  3  3  6  6  6
(43)  7  7  7  7  6  6  6  6

Cmd> print(format:"3.0f",ordertable(groups,info$classes,10))
MATRIX:
     C2  C4  C6 C10  C5  C7  C1  C3  C9  C8
G2   11   0   0   0   0   0   0   0   0   0
G3    0   7   0   0   0   0   0   0   0   0
G10   0   0   4   0   0   0   0   0   0   0
G5    0   0   0   4   3   0   0   0   0   0
G7    0   0   0   0   2   0   0   0   0   0
G9    0   0   0   0   0   4   0   0   0   0
G1    0   0   0   0   0   0   3   0   1   0
G8    0   0   3   0   0   0   0   2   0   0
G6    0   0   0   3   0   0   0   0   0   0
G4    1   0   0   0   0   1   0   0   0   1
```

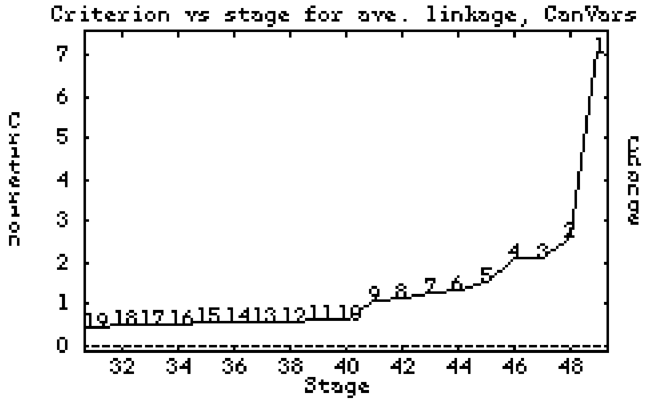
```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,6))
MATRIX:
     C2  C4  C6  C5  C1  C3
G2   11   0   0   0   0   0
G3    0   7   0   0   0   0
G10   0   0   4   0   0   0
G5    0   4   0   3   0   0
G1    0   0   0   0   4   0
G8    0   0   3   0   0   2
G7    0   0   0   2   0   0
G6    0   3   0   0   0   0
G9    4   0   0   0   0   0
G4    2   1   0   0   0   0
```

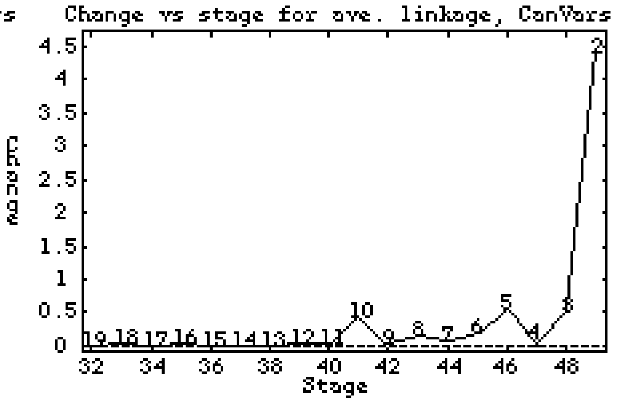
Only two of 10 clusters coincide (**3** and **9**) with groups and 11 out of 12 cases in cluster **2** are in group 2. This does not do as well as complete linkage applied to the original data. There is one singleton, cluster **1**. Principal components has not improved things with these data.

To see what is probably the *best* that might possibly be done, I computed the first 4 MANOVA canonical variables from these data and input them to `cluster()`. By design, these variables are the linear combinations of the original variables that best separate the known populations. Of course, this is not possible in any real situation because when you have pre-defined groups there is no need to do cluster analysis.

```
Cmd> groups <- factor(groups) # make sure groups is a factor

Cmd> manova("y=groups",silent:T)

Cmd> eigs <- releigen(SS[2,,],SS[3,,]) # relative eigen stuff

Cmd> z <- y %*% (eigs$vectors[,run(4)]) # compute canonical variables

Cmd> info <- cluster(z,nclust:20,keep:vector("classes","crit"),\
       standard:F) # don't standardize
```

```
Cmd> cluscritplot(info,N,values:T,\
          title:"Criterion vs stage for ave. linkage, CanVars")
Cmd> cluscritplot(info,N,change:T,\
          title:"Change vs stage for ave. linkage, CanVars")
```





Note a clear bump at stage 41 = 50-9 would suggests 10 as the right number of clusters, while another bump at stage 46 = 50 - 4 suggests these might be grouped in 5 "super clusters".

```
Cmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,9]))
groups:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3
(22)  3  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8
(43)  9  9  9  9 10 10 10 10
Clusters: Canonical variables average linkage clusters
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  6  6  6  6  6  6
(22)  6  3  3  3  4  4  4  4  4  4  4  8  8  8 10 10  5  5  5  5  5
(43)  7  7  7  7  9  9  9  9
Cmd> print(format:"3.0f",ordertable(groups,info$classes,10))
MATRIX:
     C2  C4  C6  C5  C7  C9  C1  C3  C8 C10
G2   11   0   0   0   0   0   0   0   0   0
G5    0   7   0   0   0   0   0   0   0   0
G3    0   0   7   0   0   0   0   0   0   0
G8    0   0   0   5   0   0   0   0   0   0
G9    0   0   0   0   4   0   0   0   0   0
G10   0   0   0   0   0   4   0   0   0   0
G1    0   0   0   0   0   0   4   0   0   0
G4    0   0   0   0   0   0   0   3   0   0
G6    0   0   0   0   0   0   0   0   3   0
G7    0   0   0   0   0   0   0   0   0   2
```

Remarkably, the 10 clusters selected exactly correspond to the original groups, numbered a different way. Clearly, using the right variables helps a lot.

```
Cmd> print(format:"3.0f",ordertable(groups,info$classes,5))
MATRIX:
     C2  C4  C3  C5  C1
G2   11   0   0   0   0
G5    0   7   0   0   0
G10   0   0   4   0   0
G8    0   0   0   5   0
G1    0   0   0   0   4
G4    0   0   3   0   0
G3    7   0   0   0   0
G7    0   2   0   0   0
G9    0   0   4   0   0
G6    0   3   0   0   0
```

"Super cluster" **2** consists of groups 2 and 3, **4** consists of groups 5, 6 and 7, **3** consists of groups 10, 3, and 4 and **5** and **6** are groups 8 and 1.

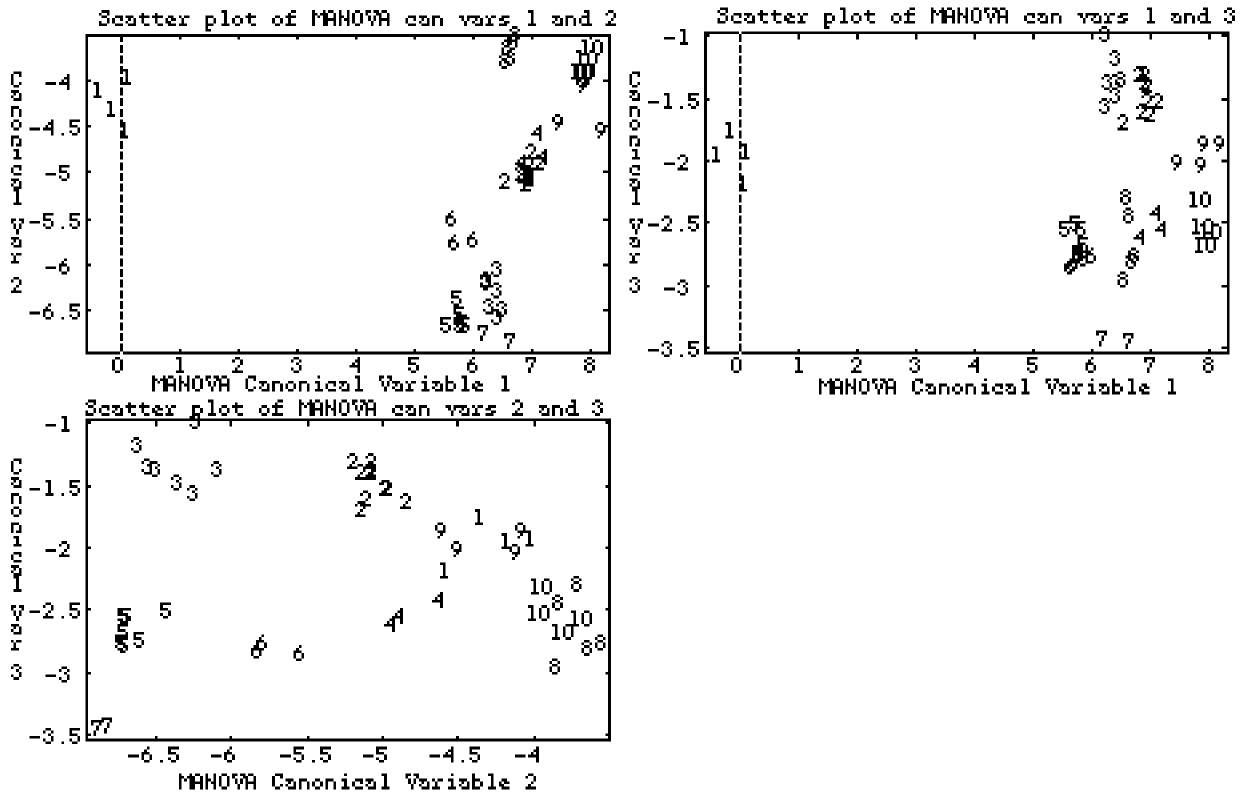
We do just as well with only 3 canonical variables (not shown), but worse with only 2.

```
Cmd> info <- cluster(z[,run(2)],nclust:20,\
        keep:vector("classes","crit"))

Cmd> print(format:"2.0f",groups,Clusters:vector(info$classes[,9]))
groups:
 (1)  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3
(22)  3  4  4  4  5  5  5  5  5  5  5  6  6  6  7  7  8  8  8  8  8
(43)  9  9  9  9 10 10 10 10
Clusters:
 (1)  1  1 10  1  2  2  2  2  2  2  2  2  2  2  2  3  3  8  8  8  8
(22)  3  7  2  2  3  3  3  3  3  3  3  5  5  5  3  3  4  4  4  4  4
(43)  6  6  7  9  6  6  6  6
```

Only two clusters coincide with groups. There are no singletons.

We can gain some idea of how apparent the actual groups are by scatter plots of the first three canonical variables against each other:

```
Cmd> plot(z[,1],z[,2],symbols:groups,\
    title:"Scatter plot of MANOVA can vars 1 and 2",\
    xlab:"MANOVA Canonical Variable 1",ylab:"Canonical Var 2")
Cmd> plot(z[,1],z[,3],symbols:groups,\
    title:"Scatter plot of MANOVA can vars 1 and 3",\
    xlab:"MANOVA Canonical Variable 1",ylab:"Canonical Var 3")
Cmd> plot(z[,2],z[,3],symbols:groups,\
    title:"Scatter plot of MANOVA can vars 2 and 3",\
    xlab:"MANOVA Canonical Variable 2",ylab:"Canonical Var 3")
```



Canonical variable 1 is primarily determined by the distance group 1 is from everything else, but also helps separate the other groups.