nag_mv_cluster_indicator (g03ejc)

1. Purpose

nag_mv_cluster_indicator (g03ejc) computes a cluster indicator variable from the results of
nag_mv_hierar_cluster_analysis (g03ecc).

2. Specification

```c
#include <nag.h>
#include <nagg03.h>

void nag_mv_cluster_indicator(Integer n, double cd[], Integer iord[],
                              double dord[], Integer *k, double *dlevel, Integer ic[],
                              NagError *fail)
```

3. Description

Given a distance or dissimilarity matrix for \( n \) objects, cluster analysis aims to group the \( n \) objects
into a number of more or less homogeneous groups or clusters. With agglomerative clustering
methods (see nag_mv_hierar_cluster_analysis (g03ecc)), a hierarchical tree is produced by starting
with \( n \) clusters each with a single object and then at each of \( n - 1 \) stages, merging two clusters
to form a larger cluster until all objects are in a single cluster. nag_mv_cluster_indicator takes the
information from the tree and produces the clusters that exist at a given distance. This is equivalent
to taking the dendrogram (see nag_mv_dendrogram (g03ehc)) and drawing a line across at a given
distance to produce clusters.

As an alternative to giving the distance at which clusters are required, the user can specify the
number of clusters required and nag_mv_cluster_indicator will compute the corresponding distance.
However, it may not be possible to compute the number of clusters required due to ties in the
distance matrix.

If there are \( k \) clusters then the indicator variable will assign a value between 1 and \( k \) to each object
to indicate to which cluster it belongs. Object 1 always belongs to cluster 1.

4. Parameters

\( n \)

Input: the number of objects, \( n \).

Constraint: \( n \geq 2 \).

\( cd[n-1] \)

Input: the clustering distances in increasing order as returned by nag_mv_hierar_cluster_analysis
(g03ecc).

Constraint: \( cd[i] \geq cd[i-1] \) for \( i = 1, 2, \ldots, n-2 \).

\( iord[n] \)

Input: the objects in the dendrogram order as returned by nag_mv_hierar_cluster_analysis
(g03ecc).

\( dord[n] \)

Input: the clustering distances corresponding to the order in \( iord \).

\( k \)

Input: indicates if a specified number of clusters is required.

If \( k > 0 \), then nag_mv_cluster_indicator (g03ejc) will attempt to find \( k \) clusters.

If \( k \leq 0 \), then nag_mv_cluster_indicator (g03ejc) will find the clusters based on the
distance given in \( dlevel \).

Constraint: \( k \leq n \).

Output: the number of clusters produced, \( k \).
NAG C Library Manual

dlevel
Input: if \( k \leq 0 \), then \( \text{dlevel} \) must contain the distance at which clusters are produced. Otherwise \( \text{dlevel} \) need not be set.
Constraint: if \( k \leq 0 \) then \( \text{dlevel} > 0.0 \).
Output: if \( k > 0 \) on entry, then \( \text{dlevel} \) contains the distance at which the required number of clusters are found. Otherwise \( \text{dlevel} \) remains unchanged.

\( \text{ic}[n] \)
Output: \( \text{ic}[i - 1] \) indicates to which of \( k \) clusters the \( i \)th object belongs, for \( i = 1, 2, \ldots, n \).

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

\textbf{NE\_INT\_ARG\_LT}
On entry, \( n \) must not be less than 2: \( n = \langle \text{value} \rangle \).

\textbf{NE\_2\_INT\_ARG\_GT}
On entry, \( k = \langle \text{value} \rangle \) while \( n = \langle \text{value} \rangle \).
These parameters must satisfy \( k \leq n \).

\textbf{NE\_REAL\_INT}
On entry, \( \text{dlevel} = \langle \text{value} \rangle \), \( k = \langle \text{value} \rangle \).
Constraint: \( k \leq 0 \) and \( \text{dlevel} > 0.0 \).

\textbf{NE\_NOT\_INCREASING}
The sequence \( \text{cd} \) is not increasing:
\( \text{cd}[\langle \text{value} \rangle] = \langle \text{value} \rangle, \text{cd}[\langle \text{value} \rangle] = \langle \text{value} \rangle \).

\textbf{NW\_REAL\_REALARR}
On entry, \( \text{dlevel} = \langle \text{value} \rangle \), \( \text{cd}[\langle \text{value} \rangle] = \langle \text{value} \rangle \).
Trivial solution returned.

\textbf{NW\_INT}
On exit, \( k = 1 \).
Trivial solution returned.

\textbf{NW\_2\_INT}
On exit, \( k = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \).
Trivial solution returned.

\textbf{NE\_INCOMP\_ARRAYS}
Arrays \( \text{cd} \) and \( \text{dord} \) are not compatible.

\textbf{NE\_CLUSTER}
The precise number of clusters requested is not possible because of tied clustering distances. The actual number of clusters produced is \( \langle \text{value} \rangle \).

\textbf{NE\_INTERNAL\_ERROR}
An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

6. Further Comments
A fixed number of clusters can be found using the non-hierarchical method used in \( \text{nag\_mv\_kmeans\_cluster\_analysis (g03efc)} \).

6.1. Accuracy
The accuracy will depend upon the accuracy of the distances in \( \text{cd} \) and \( \text{dord} \) (see \( \text{nag\_mv\_hierar\_cluster\_analysis (g03ecc)} \)).

6.2. References
7. See Also

nag_mv_kmeans_cluster_analysis (g03efc)
nag_mv_hierar_cluster_analysis (g03ecc)

8. Example

Data consisting of three variables on five objects are input. Euclidean squared distances are computed using nag_mv_distance_mat (g03eac) and median clustering performed using nag_mv_hierar_cluster_analysis (g03ecc). A dendrogram is produced by nag_mv_dendrogram (g03ehc) and printed. nag_mv_cluster_indicator finds two clusters and the results are printed.

8.1. Program Text

```c
/* nag_mv_cluster (g03ejc) Example Program.
 * Mark 6 revised, 2000.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>

#define NMAX 10
#define MMAX 10

main()
{
  double cd[NMAX-1], d[NMAX*(NMAX-1)/2], dord[NMAX],
  s[MMAX], x[MMAX][MMAX];
  double dmin_,
  double dstep, ydist;
  double dlevel;
  Integer ic[NMAX], ilc[NMAX-1], iord[NMAX], isx[MMAX],
  iuc[NMAX-1];
  Integer nsym;
  Integer i, j, k;
  Integer m, n;
  Integer int_method;
  Integer tdx=MMAX;
  char ***c = 0;
  char name[NMAX][3];
  char char_dist[2];
  char char_scale[2];
  char char_update[2];
  Nag_ClusterMethod method;
  Nag_MatUpdate update;
  Nag_DistanceType dist;
  Nag_VarScaleType scale;

  Vprintf("g03ejc Example Program Results\n\n");
  /* Skip heading in data file */
  Vscanf("%*[^n\n]");
  Vscanf("%ld", &n);
  Vscanf("%ld", &m);
  if (n <= NMAX && m <= MMAX)
    {
      Vscanf("%ld", &int_method);
      if (int_method == 1)
```
method = Nag_SingleLink;
else if (int_method == 2)
    method = Nag_CompleteLink;
else if (int_method == 3)
    method = Nag_GroupAverage;
else if (int_method == 4)
    method = Nag_Centroid;
else if (int_method == 5)
    method = Nag_Median;
else
    method = Nag_MinVariance;

Vscanf("%s",char_update);
if (*char_update == 'U')
    update = Nag_MatUp;
else
    update = Nag_NoMatUp;

Vscanf("%s",char_dist);
if (*char_dist == 'A')
    dist = Nag_DistAbs;
else if (*char_dist == 'E')
    dist = Nag_DistEuclid;
else
    dist = Nag_DistSquared;

Vscanf("%s",char_scale);
if (*char_scale == 'S')
    scale = Nag_VarScaleStd;
else if (*char_scale == 'R')
    scale = Nag_VarScaleRange;
else if (*char_scale == 'G')
    scale = Nag_VarScaleUser;
else
    scale = Nag_NoVarScale;

for (j = 0; j < n; ++j)
    for (i = 0; i < m; ++i)
        Vscanf("%lf",&x[j][i]);
    Vscanf("%s",name[j]);

for (i = 0; i < m; ++i)
    Vscanf("%ld",&isx[i]);
for (i = 0; i < m; ++i)
    Vscanf("%lf",&s[i]);

Vscanf("%ld",&k);
Vscanf("%lf",&dlevel);

/* Compute the distance matrix */
g03exc(update, dist, scale, n, m, (double *)x, tdx, isx, s, d, NAGERR_DEFAULT);

/* Perform clustering */
g03ecc(method, n, d, ilc, iuc, cd, iord, dord, NAGERR_DEFAULT);

Vprintf("\nDistance Clusters Joined\n\n");
for (i = 0; i < n-1; ++i)
{  
    Vprintf("%10.3f ",cd[i]);
    Vprintf("%3s",name[ilc[i]-1]);
    Vprintf("%3s",name[iuc[i]-1]);
    Vprintf("\n");
}  
/* Produce dendrogram */
nsym = 20;
dmin_ = 0.0;
dstep = cd[n - 2] / (double) nsym;
g03ehc(Nag_DendSouth, n, dord, dmin_, dstep, nsym, &c, NAGERR_DEFAULT);
Vprintf("\n");
Vprintf("Dendrogram ");
Vprintf("\n");
ydist = cd[n - 2];
for (i = 0; i < nsym; ++i)
{
    if ((i+1) % 3 == 1)
    {
        Vprintf("%10.3f%6s",ydist," ");
        Vprintf("%s",c[i]);
        Vprintf("\n");
    }
    else
    {
        Vprintf("%16s%s"," ",c[i]);
        Vprintf("\n");
    }
    ydist -= dstep;
}
Vprintf("\n");
Vprintf("%14s"," ");
for (i = 0; i < n; ++i)
{
    Vprintf("%3s",name[iord[i]-1]);
}
Vprintf("\n");
g03xzc(&c);
g03ejc(n, cd, iord, dord, &k, &dlevel, ic, NAGERR_DEFAULT);
Vprintf("%14s"," ");
Vprintf("%2ld%16s
\n"," Allocation to ",k," clusters");
Vprintf("Object Cluster\n\n");
for (i = 0; i < n; ++i)
{
    Vprintf("%5s%5s"," ",name[i]," ");
    Vprintf("%ld ",ic[i]);
    Vprintf("\n");
}
exit(EXIT_SUCCESS);
}
else
{
    Vprintf("Incorrect input value of n or m.\n");
    exit(EXIT_FAILURE);
}

8.2. Program Data

g03ejc Example Program Data

5 3
I S U
1 5.0 2.0 A
2 1.0 1.0 B
3 4.0 3.0 C
4 1.0 2.0 D
5 5.0 0.0 E
0 1 1
1.0 1.0 1.0
2 0.0
8.3. Program Results

g03ejc Example Program Results

Distance Clusters Joined

<table>
<thead>
<tr>
<th>Distance</th>
<th>Clusters Joined</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>B D</td>
</tr>
<tr>
<td>2.000</td>
<td>A C</td>
</tr>
<tr>
<td>6.500</td>
<td>A E</td>
</tr>
<tr>
<td>14.125</td>
<td>A B</td>
</tr>
</tbody>
</table>

Dendrogram

14.125 -------
  I I
  I I
12.006 I I
  I I
9.887  I I
  I I
7.769  I I
  ---* I
  I I
5.650  I I
  I I
3.531  I I
  ---* I
  I I
1.412  I I I ----*
  I I I I I
A C E B D

Allocation to 2 clusters

Object Cluster

<table>
<thead>
<tr>
<th>Object</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
</tbody>
</table>