NAG Library Function Document

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

```
#include <nag.h>
#include <nagg03.h>
void nag_mv_cluster_indicator (Integer n, const double cd[],
      const Integer iord[], const 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 (g03ejc) 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, you can specify the number of clusters required and nag_mv_cluster_indicator (g03ejc) 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   References

Everitt B S (1974) *Cluster Analysis* Heinemann

Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

5   Arguments

1:    **n** – Integer   *Input*

On entry: the number of objects, n.

Constraint: **n** ≥ 2.

2:    **cd**[**n** − **1**] – const double   *Input*

On entry: 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** $- 2$.

3:     **iord**[**n**] – const Integer   *Input*

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

4:     **dord**[**n**] – const double   *Input*

On entry: the clustering distances corresponding to the order in **iord**.

5:     **k** – Integer *   *Input/Output*

On entry: indicates if a specified number of clusters is required.

k > 0
    nag_mv_cluster_indicator (g03ejc) will attempt to find **k** clusters.

k ≤ 0
    nag_mv_cluster_indicator (g03ejc) will find the clusters based on the distance given in **dlevel**.

Constraint: **k** ≤ **n**.

On exit: the number of clusters produced, k.

6:     **dlevel** – double *   *Input/Output*

On entry: if **k** ≤ 0, then **dlevel** must contain the distance at which clusters are produced. Otherwise **dlevel** need not be set.

Constraint: if **k** ≤ 0, **dlevel** > 0.0.

On exit: if **k** > 0 on entry, then **dlevel** contains the distance at which the required number of clusters are found. Otherwise **dlevel** remains unchanged.

7:     **ic**[**n**] – Integer   *Output*

On exit: **ic**[$i − 1$] indicates to which of k clusters the ith object belongs, for $i = 1, 2, \ldots, n$.

8:     **fail** – NagError *   *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6   Error Indicators and Warnings

NE_2_INT_ARG_GT

On entry, **k** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These arguments must satisfy **k** ≤ **n**.

NE_CLUSTER

The precise number of clusters requested is not possible because of tied clustering distances. The actual number of clusters produced is ⟨*value*⟩.

NE_INCOMP_ARRAYS

Arrays **cd** and **dord** are not compatible.

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 2.

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 contact NAG for assistance.

NE_NOT_INCREASING

The sequence **cd** is not increasing:
$\mathbf{cd}[\langle value\rangle] = \langle value\rangle$, $\mathbf{cd}[\langle value\rangle] = \langle value\rangle$.

NE_REAL_INT

On entry, **dlevel** $= \langle value\rangle$, $\mathbf{k} = \langle value\rangle$.
Constraint: $\mathbf{k} \leq 0$ and **dlevel** > 0.0.

NW_2_INT

On exit, $\mathbf{k} = \langle value\rangle$, $\mathbf{n} = \langle value\rangle$.
Trivial solution returned.

NW_INT

On exit, $\mathbf{k} = 1$.
Trivial solution returned.

NW_REAL_REALARR

On entry, **dlevel** $= \langle value\rangle$, $\mathbf{cd}[\langle value\rangle] = \langle value\rangle$.
Trivial solution returned.

7 Accuracy

The accuracy will depend upon the accuracy of the distances in **cd** and **dord** (see nag_mv_hierar_cluster_analysis (g03ecc)).

8 Parallelism and Performance

nag_mv_cluster_indicator (g03ejc) is not threaded in any implementation.

9 Further Comments

A fixed number of clusters can be found using the non-hierarchical method used in nag_mv_kmeans_cluster_analysis (g03efc).

10 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 (g03ejc) finds two clusters and the results are printed.

10.1 Program Text

```
/* nag_mv_cluster_indicator (g03ejc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */
```

```
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>

#define X(I, J) x[(I) *tdx + J]
int main(void)
{
  Integer exit_status = 0, i, *ic = 0, *ilc = 0, *iord = 0, *isx = 0;
  Integer *iuc = 0;
  Integer j, k, m, n, nsym, tdx;
  NagError fail;
  Nag_ClusterMethod method;
  Nag_DistanceType dist;
  Nag_MatUpdate update;
  Nag_VarScaleType scale;
  char nag_enum_arg[40];
  char **c = 0, name[40][3];
  double *cd = 0, *d = 0, dlevel, dmin_, *dord = 0, dstep, *s = 0;
  double *x = 0, ydist;

  INIT_FAIL(fail);

  printf("nag_mv_cluster_indicator (g03ejc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &n);
#else
  scanf("%" NAG_IFMT "", &n);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &m);
#else
  scanf("%" NAG_IFMT "", &m);
#endif
  if (n >= 2 && m >= 1) {
    if (!(cd = NAG_ALLOC(n - 1, double)) ||
        !(d = NAG_ALLOC(n * (n - 1) / 2, double)) ||
        !(dord = NAG_ALLOC(n, double)) ||
        !(s = NAG_ALLOC(m, double)) ||
        !(x = NAG_ALLOC((n) * (m), double)) ||
        !(ic = NAG_ALLOC(n, Integer)) ||
        !(ilc = NAG_ALLOC(n - 1, Integer)) ||
        !(iord = NAG_ALLOC(n, Integer)) ||
        !(isx = NAG_ALLOC(m, Integer)) || !(iuc = NAG_ALLOC(n - 1, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tdx = m;
  }
  else {
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
  }
#ifdef _WIN32
  scanf_s("%39s%*[^\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n] ", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
```

```
     * Converts NAG enum member name to value
     */
  method = (Nag_ClusterMethod) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  update = (Nag_MatUpdate) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  dist = (Nag_DistanceType) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s("%39s%*[^\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n] ", nag_enum_arg);
#endif
  scale = (Nag_VarScaleType) nag_enum_name_to_value(nag_enum_arg);

  for (j = 0; j < n; ++j) {
    for (i = 0; i < m; ++i)
#ifdef _WIN32
      scanf_s("%lf", &X(j, i));
#else
      scanf("%lf", &X(j, i));
#endif
#ifdef _WIN32
    scanf_s("%2s", name[j], 3);
#else
    scanf("%2s", name[j]);
#endif
  }
  for (i = 0; i < m; ++i)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &isx[i]);
#else
    scanf("%" NAG_IFMT "", &isx[i]);
#endif
  for (i = 0; i < m; ++i)
#ifdef _WIN32
    scanf_s("%lf", &s[i]);
#else
    scanf("%lf", &s[i]);
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &k);
#else
  scanf("%" NAG_IFMT "", &k);
#endif
#ifdef _WIN32
  scanf_s("%lf", &dlevel);
#else
  scanf("%lf", &dlevel);
#endif

  /* Compute the distance matrix */
  /* nag_mv_distance_mat (g03eac).
   * Compute distance (dissimilarity) matrix
   */
  nag_mv_distance_mat(update, dist, scale, n, m, x, tdx, isx, s, d, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mv_distance_mat (g03eac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Perform clustering */
```

```
  /* nag_mv_hierar_cluster_analysis (g03ecc).
   * Hierarchical cluster analysis
   */
  nag_mv_hierar_cluster_analysis(method, n, d, ilc, iuc, cd, iord, dord,
                                 &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mv_cluster_indicator (g03ejc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nDistance   Clusters Joined\n\n");

  for (i = 0; i < n - 1; ++i) {
    printf("%10.3f      ", cd[i]);
    printf("%3s", name[ilc[i] - 1]);
    printf("%3s", name[iuc[i] - 1]);
    printf("\n");
  }
  /* Produce dendrogram */
  nsym = 20;
  dmin_ = 0.0;
  dstep = cd[n - 2] / (double) nsym;
  /* nag_mv_dendrogram (g03ehc).
   * Construct dendrogram following
   * nag_mv_hierar_cluster_analysis (g03ecc)
   */
  nag_mv_dendrogram(Nag_DendSouth, n, dord, dmin_, dstep, nsym, &c, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mv_dendrogram (g03ehc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  printf("\n");
  printf("Dendrogram ");
  printf("\n");
  printf("\n");
  ydist = cd[n - 2];
  for (i = 0; i < nsym; ++i) {
    if ((i + 1) % 3 == 1) {
      printf("%10.3f%6s", ydist, "");
      printf("%s", c[i]);
      printf("\n");
    }
    else {
      printf("%16s%s", "", c[i]);
      printf("\n");
    }
    ydist -= dstep;
  }
  printf("\n");
  printf("%14s", "");
  for (i = 0; i < n; ++i) {
    printf("%3s", name[iord[i] - 1]);
  }
  printf("\n");
  /* nag_mv_dend_free (g03xzc).
   * Frees memory allocated to the dendrogram array in
   * nag_mv_dendrogram (g03ehc)
   */
  nag_mv_dend_free(&c);
  /* nag_mv_cluster_indicator (g03ejc).
   * Construct clusters following
   * nag_mv_hierar_cluster_analysis (g03ecc)
   */
  nag_mv_cluster_indicator(n, cd, iord, dord, &k, &dlevel, ic, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mv_cluster_indicator (g03ejc).\n%s\n",
           fail.message);
    exit_status = 1;
```

```
    goto END;
  }
  printf("\n%s%2" NAG_IFMT "%s\n\n", "Allocation to ", k, " clusters");
  printf("Object  Cluster\n\n");
  for (i = 0; i < n; ++i) {
    printf("%5s%s%5s", "", name[i], "");
    printf("%" NAG_IFMT "      ", ic[i]);
    printf("\n");
  }
END:
  NAG_FREE(cd);
  NAG_FREE(d);
  NAG_FREE(dord);
  NAG_FREE(s);
  NAG_FREE(x);
  NAG_FREE(ic);
  NAG_FREE(ilc);
  NAG_FREE(iord);
  NAG_FREE(isx);
  NAG_FREE(iuc);
  return exit_status;
}
```

10.2 Program Data

```
nag_mv_cluster_indicator (g03ejc) Example Program Data
5 3
Nag_Median
Nag_NoMatUp Nag_DistSquared Nag_NoVarScale
 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
```

10.3 Program Results

```
nag_mv_cluster_indicator (g03ejc) Example Program Results


Distance    Clusters Joined

     1.000       B  D
     2.000       A  C
     6.500       A  E
    14.125       A  B


Dendrogram

    14.125             -------
                       I     I
                       I     I
    12.006             I     I
                       I     I
                       I     I
     9.887             I     I
                       I     I
                       I     I
     7.769             I     I
                     ---*     I
                     I I      I
     5.650           I I      I
                     I I      I
                     I I      I
     3.531           I I      I
                     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

```
    A      1
    B      2
    C      1
    D      2
    E      1
```