

# NAG Library Function Document

## nag\_mv\_hierar\_cluster\_analysis (g03ecc)

### 1 Purpose

nag\_mv\_hierar\_cluster\_analysis (g03ecc) performs hierarchical cluster analysis.

### 2 Specification

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

void nag_mv_hierar_cluster_analysis (Nag_ClusterMethod method, Integer n,
    double d[], Integer ilc[], Integer iuc[], double cd[], Integer iord[],
    double dord[], NagError *fail)
```

### 3 Description

Given a distance or dissimilarity matrix for  $n$  objects (see nag\_mv\_distance\_mat (g03eac)), cluster analysis aims to group the  $n$  objects into a number of more or less homogeneous groups or clusters. With agglomerative clustering methods, 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. This process may be represented by a dendrogram (see nag\_mv\_dendrogram (g03ehc)).

At each stage, the clusters that are nearest are merged, methods differ as to how the distance between the new cluster and other clusters are computed. For three clusters  $i$ ,  $j$  and  $k$  let  $n_i$ ,  $n_j$  and  $n_k$  be the number of objects in each cluster and let  $d_{ij}$ ,  $d_{ik}$  and  $d_{jk}$  be the distances between the clusters. Let clusters  $j$  and  $k$  be merged to give cluster  $jk$ , then the distance from cluster  $i$  to cluster  $jk$ ,  $d_{i,jk}$  can be computed in the following ways:

1. Single link or nearest neighbour:  $d_{i,jk} = \min(d_{ij}, d_{ik})$ .
2. Complete link or furthest neighbour:  $d_{i,jk} = \max(d_{ij}, d_{ik})$ .
3. Group average:  $d_{i,jk} = \frac{n_j}{n_j+n_k}d_{ij} + \frac{n_k}{n_j+n_k}d_{ik}$ .
4. Centroid:  $d_{i,jk} = \frac{n_j}{n_j+n_k}d_{ij} + \frac{n_k}{n_j+n_k}d_{ik} - \frac{n_j n_k}{(n_j+n_k)^2}d_{jk}$ .
5. Median:  $d_{i,jk} = \frac{1}{2}d_{ij} + \frac{1}{2}d_{ik} - \frac{1}{4}d_{jk}$ .
6. Minimum variance:  $d_{i,jk} = \{(n_i + n_j)d_{ij} + (n_i + n_k)d_{ik} - n_i d_{jk}\} / (n_i + n_j + n_k)$ .

For further details see Everitt (1974) or Krzanowski (1990).

If the clusters are numbered  $1, 2, \dots, n$  then, for convenience, if clusters  $j$  and  $k$ ,  $j < k$ , merge then the new cluster will be referred to as cluster  $j$ . Information on the clustering history is given by the values of  $j$ ,  $k$  and  $d_{jk}$  for each of the  $n - 1$  clustering steps. In order to produce a dendrogram, the ordering of the objects such that the clusters that merge are adjacent is required. This ordering is computed so that the first element is 1. The associated distances with this ordering are also computed.

### 4 References

Everitt B S (1974) *Cluster Analysis* Heinemann

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

## 5 Arguments

- 1: **method** – Nag\_ClusterMethod *Input*  
*On entry:* indicates which clustering.  
**method** = Nag\_SingleLink  
 Single link.  
**method** = Nag\_CompleteLink  
 Complete link.  
**method** = Nag\_GroupAverage  
 Group average.  
**method** = Nag\_Centroid  
 Centroid.  
**method** = Nag\_Median  
 Median.  
**method** = Nag\_MinVariance  
 Minimum variance.  
*Constraint:* **method** = Nag\_SingleLink, Nag\_CompleteLink, Nag\_GroupAverage, Nag\_Centroid, Nag\_Median or Nag\_MinVariance.
- 2: **n** – Integer *Input*  
*On entry:* the number of objects,  $n$ .  
*Constraint:*  $n \geq 2$ .
- 3: **d**[ $n \times (n - 1) / 2$ ] – double *Input/Output*  
*On entry:* the strictly lower triangle of the distance matrix.  $D$  must be stored packed by rows, i.e., **d**[( $i - 1$ )( $i - 2$ )/2 +  $j - 1$ ],  $i > j$  must contain  $d_{ij}$ .  
*On exit:* is overwritten.  
*Constraint:* **d**[ $i - 1$ ]  $\geq 0.0$ , for  $i = 1, 2, \dots, n(n - 1) / 2$ .
- 4: **ilc**[ $n - 1$ ] – Integer *Output*  
*On exit:* **ilc**[ $l - 1$ ] contains the number,  $j$ , of the cluster merged with cluster  $k$  (see **iuc**),  $j < k$ , at step  $l$ , for  $l = 1, 2, \dots, n - 1$ .
- 5: **iuc**[ $n - 1$ ] – Integer *Output*  
*On exit:* **iuc**[ $l - 1$ ] contains the number,  $k$ , of the cluster merged with cluster  $j$ ,  $j < k$ , at step  $l$ , for  $l = 1, 2, \dots, n - 1$ .
- 6: **cd**[ $n - 1$ ] – double *Output*  
*On exit:* **cd**[ $l - 1$ ] contains the distance  $d_{jk}$ , between clusters  $j$  and  $k$ ,  $j < k$ , merged at step  $l$ , for  $l = 1, 2, \dots, n - 1$ .
- 7: **iord**[ $n$ ] – Integer *Output*  
*On exit:* the objects in dendrogram order.
- 8: **dord**[ $n$ ] – double *Output*  
*On exit:* the clustering distances corresponding to the order in **iord**. **dord**[ $l - 1$ ] contains the distance at which cluster **iord**[ $l - 1$ ] and **iord**[ $l$ ] merge, for  $l = 1, 2, \dots, n - 1$ . **dord**[ $n - 1$ ] contains the maximum distance.

9: **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\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_BAD\_PARAM

On entry, argument **method** had an illegal value.

### NE\_DENDROGRAM

A true dendrogram cannot be formed because the distances at which clusters have merged are not increasing for all steps, i.e.,  $\mathbf{cd}[i - 1] < \mathbf{cd}[i - 2]$  for some  $i = 2, 3, \dots, n - 1$ . This can occur for the **method** = Nag\_Centroid and **method** = Nag\_Median methods.

### NE\_INT\_ARG\_LT

On entry, **n** =  $\langle value \rangle$ .  
Constraint: **n**  $\geq 2$ .

### 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\_REALARR

On entry,  $\mathbf{d}[\langle value \rangle] = \langle value \rangle$ .  
Constraint:  $\mathbf{d}[i - 1] \geq 0.0$ , for  $i = 1, 2, \dots, n \times (n - 1)/2$ .

## 7 Accuracy

For methods other than **method** = Nag\_SingleLink or Nag\_CompleteLink, slight rounding errors may occur in the calculations of the updated distances. These would not normally significantly affect the results, however there may be an effect if distances are (almost) equal.

If at a stage, two distances  $d_{ij}$  and  $d_{kl}$ ,  $i < k$  or  $i = k$  and  $j < l$ , are equal then clusters  $k$  and  $l$  will be merged rather than clusters  $i$  and  $j$ . For single link clustering this choice will only affect the order of the objects in the dendrogram. However, for other methods the choice of  $kl$  rather than  $ij$  may affect the shape of the dendrogram. If either of the distances  $d_{ij}$  or  $d_{kl}$  are affected by rounding errors then their equality, and hence the dendrogram, may be affected.

## 8 Parallelism and Performance

nag\_mv\_hierar\_cluster\_analysis (g03ecc) is not threaded in any implementation.

## 9 Further Comments

The dendrogram may be formed using nag\_mv\_dendrogram (g03ehc). Groupings based on the clusters formed at a given distance can be computed using nag\_mv\_cluster\_indicator (g03ejc).

## 10 Example

Data consisting of three variables on five objects are read in. Euclidean squared distances based on two variables are computed using `nag_mv_distance_mat` (g03eac), the objects are clustered using `nag_mv_hierar_cluster_analysis` (g03ecc) and the dendrogram computed using `nag_mv_dendrogram` (g03ehc). The dendrogram is then printed.

### 10.1 Program Text

```

/* nag_mv_hierar_cluster_analysis (g03ecc) 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, j, m, n, nsym, tdx;
  Integer *ilc = 0, *iord = 0, *isx = 0, *iuc = 0;
  char **c = 0, name[40][2];
  double dmin_, dstep, ydist;
  double *cd = 0, *d = 0, *dord = 0, *s = 0, *x = 0;
  char nag_enum_arg[40];
  Nag_ClusterMethod method;
  Nag_DistanceType dist;
  Nag_MatUpdate update;
  Nag_VarScaleType scale;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_mv_hierar_cluster_analysis (g03ecc) 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)) ||
        !(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;
}
#endif
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s", 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", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s", 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("%1s", name[j], 2);
#else
            scanf("%1s", 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
    }

    /* 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_hierar_cluster_analysis (g03ecc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}

printf("\n Distance Clusters Joined\n\n");
for (i = 1; i <= n - 1; ++i)
    printf("%10.3f      %3s%3s\n", cd[i - 1], name[ilc[i - 1] - 1],
           name[iuc[i - 1] - 1]);

/* 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 (g03zxc).
 * Frees memory allocated to the dendrogram array in
 * nag_mv_dendrogram (g03ehc)
 */
nag_mv_dend_free(&c);

END:
NAG_FREE(cd);
NAG_FREE(d);

```

```

NAG_FREE(dord);
NAG_FREE(s);
NAG_FREE(x);
NAG_FREE(ilc);
NAG_FREE(iord);
NAG_FREE(isx);
NAG_FREE(iuc);

return exit_status;
}

```

## 10.2 Program Data

```

nag_mv_hierar_cluster_analysis (g03ecc) 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

```

## 10.3 Program Results

```

nag_mv_hierar_cluster_analysis (g03ecc) 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
5.650       I      I
              I      I
              I      I
              I      I
3.531       I      I
              I      I
              I      I
              ---*   I
1.412       I      I      I      ---*
              I      I      I      I      I
              A      C      E      B      D

```

---