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

5 Arguments
1: \( n \) – Integer
   \hspace{1cm} Input
   On entry: the number of objects, \( n \).
   Constraint: \( n \geq 2 \).

2: \( cd[n - 1] \) – const double
   \hspace{1cm} 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:  
   \texttt{iord[n]} -- const Integer  
   \textit{Input}
   \textit{On entry:} the objects in the dendrogram order as returned by \texttt{nag_mv_hierar_cluster_analysis (g03ecc)}.

4:  
   \texttt{dord[n]} -- const double  
   \textit{Input}
   \textit{On entry:} the clustering distances corresponding to the order in \texttt{iord}.

5:  
   \texttt{k} -- Integer *  
   \textit{Input/Output}
   \textit{On entry:} indicates if a specified number of clusters is required.
   \texttt{k} > 0
   \textit{nag_mv_cluster_indicator (g03ejc)} will attempt to find \texttt{k} clusters.
   \texttt{k} \leq 0
   \textit{nag_mv_cluster_indicator (g03ejc)} will find the clusters based on the distance given in \texttt{dlevel}.
   \textit{Constraint:} \texttt{k} \leq \texttt{n}.
   \textit{On exit:} the number of clusters produced, \texttt{k}.

6:  
   \texttt{dlevel} -- double *  
   \textit{Input/Output}
   \textit{On entry:} if \texttt{k} \leq 0, then \texttt{dlevel} must contain the distance at which clusters are produced. Otherwise \texttt{dlevel} need not be set.
   \textit{Constraint:} if \texttt{k} \leq 0, \texttt{dlevel} \geq 0.0.
   \textit{On exit:} if \texttt{k} > 0 on entry, then \texttt{dlevel} contains the distance at which the required number of clusters are found. Otherwise \texttt{dlevel} remains unchanged.

7:  
   \texttt{ic[n]} -- Integer  
   \textit{Output}
   \textit{On exit:} \texttt{ic[i-1]} indicates to which of \texttt{k} clusters the \texttt{i}th object belongs, for \texttt{i} = 1, 2, \ldots, \texttt{n}.

8:  
   \texttt{fail} -- NagError *  
   \textit{Input/Output}
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6  \textbf{Error Indicators and Warnings}

\textbf{NE_2_INT_ARG_GT}

On entry, \texttt{k} = \langle value\rangle while \texttt{n} = \langle value\rangle. These arguments must satisfy \texttt{k} \leq \texttt{n}.

\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 value\rangle.

\textbf{NE_INCOMP_ARRAYS}

Arrays \texttt{cd} and \texttt{dord} are not compatible.

\textbf{NE_INT_ARG_LT}

On entry, \texttt{n} = \langle value\rangle.
   \textit{Constraint:} \texttt{n} \geq 2.

\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 contact NAG for assistance.
The sequence cd is not increasing:
\[ cd[\text{value}] = \text{value}, \quad cd[\text{value}] = \text{value}. \]

On entry, dlevel = (value), \( k = (value) \).
Constraint: \( k \leq 0 \) and dlevel > 0.0.

On exit, \( k = (value) \), \( n = (value) \).
Trivial solution returned.

On exit, \( k = 1 \).
Trivial solution returned.

On entry, dlevel = (value), \[ cd[\text{value}] = \text{value}. \]
Trivial solution returned.

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

Not applicable.

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

Data consisting of three variables on five objects are input. Euclidean squared distances are computed using nag_mv_distance_mat (g03ec) 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.

/* nag_mv_cluster_indicator (g03ejc) Example Program.  *
* Copyright 2014 Numerical Algorithms Group.  *
* Mark 5, 1998.  *
* Mark 6 revised, 2000.  *
* Mark 8 revised, 2004.  *
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <g03.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
    #ifdef _WIN32
        scanf_s("%39s%*[\n] ", nag_enum_arg, _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 */

    if (n >= 2 && m >= 1) {
        if (!cd || !d || !dord || !s || !x)
            printf("Allocation failure\n");
        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, _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, _countof(nag_enum_arg));
#else
    scanf("%39s", nag_enum_arg);
#endif
#endif
update = (Nag_MatUpdate) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, _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, _countof(nag_enum_arg));
#else
    scanf("%39s%*[\n] ", nag_enum_arg);
#endif
#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
        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).
        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 dendogram 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("\nDendrogram ");
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]);
}

/* 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",
            fail.message);
    exit_status = 1;
    goto END;
}

printf("%s%2"NAG_IFMT"%s\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
12.006      I     I
     I     I
  9.887      I     I
     I     I
### Allocation to 2 clusters

Object | Cluster
-------|--------
A      | 1      
B      | 2      
C      | 1      
D      | 2      
E      | 1      

---

- Allocation to 2 clusters

<table>
<thead>
<tr>
<th>Value</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.769</td>
<td>A, C</td>
</tr>
<tr>
<td>5.650</td>
<td>B</td>
</tr>
<tr>
<td>3.531</td>
<td>D</td>
</tr>
<tr>
<td>1.412</td>
<td>E</td>
</tr>
</tbody>
</table>