NAG Library Function Document

nag_mv_distance_mat (g03eac)

1 Purpose

nag_mv_distance_mat (g03eac) computes a distance (dissimilarity) matrix.

2 Specification

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

void nag_mv_distance_mat (Nag_MatUpdate update, Nag_DistanceType dist,
                        Nag_VarScaleType scale, Integer n, Integer m, const double x[],
                        Integer tdx, const Integer isx[], double s[], double d[],
                        NagError *fail)
```

3 Description

Given \( n \) objects, a distance or dissimilarity matrix, is a symmetric matrix with zero diagonal elements such that the \( ij \)th element represents how far apart or how dissimilar the \( i \)th and \( j \)th objects are.

Let \( X \) be an \( n \) by \( p \) data matrix of observations of \( p \) variables on \( n \) objects, then the distance between object \( j \) and object \( k \), \( d_{jk} \), can be defined as:

\[
d_{jk} = \left( \frac{\sum_{i=1}^{p} D(x_{ji} / s_i, x_{ki} / s_i)}{n} \right)^{\alpha},
\]

where \( x_{ji} \) and \( x_{ki} \) are the \((j, i)\)th and \((k, i)\)th elements of \( X \), \( s_i \) is a standardization for the \( i \)th variable and \( D(u, v) \) is a suitable function. Three functions are provided in nag_mv_distance_mat (g03eac):

(a) Euclidean distance: \( D(u, v) = (u - v)^2 \) and \( \alpha = \frac{1}{2} \).
(b) Euclidean squared distance: \( D(u, v) = (u - v)^2 \) and \( \alpha = 1 \).
(c) Absolute distance (city block metric): \( D(u, v) = |u - v| \) and \( \alpha = 1 \).

Three standardizations are available:

1. Standard deviation: \( s_i = \sqrt{\frac{\sum_{j=1}^{n} (x_{ji} - \bar{x})^2}{(n - 1)}} \)
2. Range: \( s_i = \max(x_{1i}, x_{2i}, \ldots, x_{ni}) - \min(x_{1i}, x_{2i}, \ldots, x_{ni}) \)
3. User-supplied values of \( s_i \).

In addition to the above distances there are a large number of other dissimilarity measures, particularly for dichotomous variables (see Krzanowski (1990) and Everitt (1974)). For the dichotomous case these measures are simple to compute and can, if suitable scaling is used, be combined with the distances computed by nag_mv_distance_mat (g03eac) using the updating option.

Dissimilarity measures for variables can be based on the correlation coefficient for continuous variables and contingency table statistics for dichotomous data, see the g02 Chapter Introduction and the g11 Chapter Introduction respectively.

nag_mv_distance_mat (g03eac) returns the strictly lower triangle of the distance matrix.
4 References

5 Arguments
1: \textit{update} – \texttt{Nag MatUpdate} \textit{Input}
   \textit{On entry}: indicates whether or not an existing matrix is to be updated.
   \texttt{update} = \texttt{Nag MatUp}
   The matrix $D$ is updated and distances are added to $D$.
   \texttt{update} = \texttt{Nag NoMatUp}
   The matrix $D$ is initialized to zero before the distances are added to $D$.
   \textit{Constraint}: \texttt{update} = \texttt{Nag MatUp} or \texttt{Nag NoMatUp}.

2: \textit{dist} – \texttt{Nag DistanceType} \textit{Input}
   \textit{On entry}: indicates which type of distances are computed.
   \texttt{dist} = \texttt{Nag DistAbs}
   Absolute distances.
   \texttt{dist} = \texttt{Nag DistEuclid}
   Euclidean distances.
   \texttt{dist} = \texttt{Nag DistSquared}
   Euclidean squared distances.
   \textit{Constraint}: \texttt{dist} = \texttt{Nag DistAbs}, \texttt{Nag DistEuclid} or \texttt{Nag DistSquared}.

3: \textit{scale} – \texttt{Nag VarScaleType} \textit{Input}
   \textit{On entry}: indicates the standardization of the variables to be used.
   \texttt{scale} = \texttt{Nag VarScaleStd}
   Standard deviation.
   \texttt{scale} = \texttt{Nag VarScaleRange}
   Range.
   \texttt{scale} = \texttt{Nag VarScaleUser}
   Standardizations given in array $S$.
   \texttt{scale} = \texttt{Nag NoVarScale}
   Unscaled.
   \textit{Constraint}: \texttt{scale} = \texttt{Nag VarScaleStd}, \texttt{Nag VarScaleRange}, \texttt{Nag VarScaleUser} or \texttt{Nag NoVarScale}.

4: \textit{n} – Integer \textit{Input}
   \textit{On entry}: $n$, the number of observations.
   \textit{Constraint}: $n \geq 2$.

5: \textit{m} – Integer \textit{Input}
   \textit{On entry}: the total number of variables in array $x$.
   \textit{Constraint}: $m > 0$. 

\texttt{g03eac} \quad NAG Library Manual
6: \( \mathbf{x}[\mathbf{n} \times \mathbf{tdx}] \) – const double

*Input*

On entry: \( \mathbf{x}[(i-1) \times \mathbf{tdx} + j-1] \) must contain the value of the \( j \)th variable for the \( i \)th object, for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \).

7: \( \mathbf{tdx} \) – Integer

*Input*

On entry: the stride separating matrix column elements in the array \( \mathbf{x} \).

Constraint: \( \mathbf{tdx} \geq m \).

8: \( \mathbf{iss}[\mathbf{m}] \) – const Integer

*Input*

On entry: \( \mathbf{iss}[j-1] \) indicates whether or not the \( j \)th variable in \( \mathbf{x} \) is to be included in the distance computations.

If \( \mathbf{iss}[j-1] > 0 \) the \( j \)th variable is included, for \( j = 1, 2, \ldots, m \); otherwise it is not referenced.

Constraint: \( \mathbf{iss}[j-1] > 0 \) for at least one \( j \), for \( j = 1, 2, \ldots, m \).

9: \( \mathbf{s}[\mathbf{m}] \) – double

*Input/Output*

On entry: if \( \mathbf{scale} = \text{Nag\_VarScaleUser} \) and \( \mathbf{iss}[j-1] > 0 \) then \( \mathbf{s}[j-1] \) must contain the scaling for variable \( j \), for \( j = 1, 2, \ldots, m \).

Constraint: if \( \mathbf{scale} = \text{Nag\_VarScaleUser} \) and \( \mathbf{iss}[j-1] > 0 \), \( \mathbf{s}[j-1] > 0.0 \), for \( j = 1, 2, \ldots, m \).

On exit: if \( \mathbf{scale} = \text{Nag\_VarScaleStd} \) and \( \mathbf{iss}[j-1] > 0 \) then \( \mathbf{s}[j-1] \) contains the standard deviation of the variable in the \( j \)th column of \( \mathbf{x} \).

If \( \mathbf{scale} = \text{Nag\_VarScaleRange} \) and \( \mathbf{iss}[j-1] > 0 \) then \( \mathbf{s}[j-1] \) contains the range of the variable in the \( j \)th column of \( \mathbf{x} \).

If \( \mathbf{scale} = \text{Nag\_NoVarScale} \) and \( \mathbf{iss}[j-1] > 0 \) then \( \mathbf{s}[j-1] = 1.0 \) and if \( \mathbf{scale} = \text{Nag\_VarScaleUser} \) then \( \mathbf{s} \) is unchanged.

10: \( \mathbf{d}[\mathbf{n} \times (\mathbf{n} - 1)/2] \) – double

*Input/Output*

On entry: if \( \mathbf{update} = \text{Nag\_MatUp} \) then \( \mathbf{d} \) must contain the strictly lower triangle of the distance matrix \( \mathbf{D} \) to be updated. \( \mathbf{D} \) must be stored packed by rows, i.e., \( \mathbf{d}[(i-1)(i-2)/2 + j-1], i > j \) must contain \( d_{ij} \).

Constraint: if \( \mathbf{update} = \text{Nag\_MatUp} \), \( \mathbf{d}[j-1] \geq 0.0 \), for \( j = 1, 2, \ldots, n(n-1)/2 \).

On exit: the strictly lower triangle of the distance matrix \( \mathbf{D} \) stored packed by rows, i.e., \( d_{ij} \) is contained in \( \mathbf{d}[(i-1)(i-2)/2 + j-1], i > j \).

11: \( \mathbf{fail} \) – NagError

*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

**NE\_2\_INT\_ARG\_LT**

On entry, \( \mathbf{tdx} = \langle \text{value} \rangle \) while \( \mathbf{m} = \langle \text{value} \rangle \). These arguments must satisfy \( \mathbf{tdx} \geq \mathbf{m} \).

**NE\_BAD\_PARAM**

On entry, argument \( \mathbf{dist} \) had an illegal value.

On entry, argument \( \mathbf{scale} \) had an illegal value.

On entry, argument \( \mathbf{update} \) had an illegal value.
On entry, \( \text{scale} = \text{Nag}_\text{VarScaleRange} \) or \( \text{scale} = \text{Nag}_\text{VarScaleStd} \), and
\[ x_i[(i-1) \times \text{tdx} + j - 1] = x_i[(i) \times \text{tdx} + j - 1], \quad \text{for } i = 1, 2, \ldots, n - 1, \quad \text{for some } j \text{ with } isx[i-1] > 0. \]

On entry, \( m = \langle \text{value} \rangle \).
Constraint: \( m > 0 \).

On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 2 \).

On entry, \( isx[i-1] = \langle \text{value} \rangle \).
Constraint: \( isx[i-1] > 0, \) for at least one \( i, i = 1, 2, \ldots, m \).

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.

On entry, \( d[i-1] = \langle \text{value} \rangle \).
Constraint: \( d[i-1] \geq 0, \) for \( i = 1, 2, \ldots, n \times (n - 1)/2 \), when \( \text{update} = \text{Nag}_\text{MatUp} \).
On entry, \( s[j-1] = \langle \text{value} \rangle \).
Constraint: \( s[j-1] > 0, \) for \( j = 1, 2, \ldots, m \), when \( \text{scale} = \text{Nag}_\text{VarScaleUser} \) and \( isx[j-1] > 0 \).

The computations are believed to be stable.

Not applicable.

nag\_mv\_hierar\_cluster\_analysis (g03ecc) can be used to perform cluster analysis on the computed distance matrix.

A data matrix of five observations and three variables is read in and a distance matrix is calculated from variables 2 and 3 using squared Euclidean distance with no scaling. This matrix is then printed.

A data matrix of five observations and three variables is read in and a distance matrix is calculated from variables 2 and 3 using squared Euclidean distance with no scaling. This matrix is then printed.

/* nag\_mv\_distance\_mat (g03eac) Example Program. *
* Copyright 2014 Numerical Algorithms Group. *
* Mark 5, 1998. *
* Mark 8 revised, 2004. */

/* Example */

/* Further Comments */

/* Program Text */
```c
#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, *isx = 0, j, m, n, tdx;
    double *d = 0, *s = 0, *x = 0;
    char nag_enum_arg[40];
    Nag_DistanceType dist;
    Nag_MatUpdate update;
    Nag_VarScaleType scale;
    NagError fail;
    INIT_FAIL(fail);

    printf("nag_mv_distance_mat (g03eac) 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 (!(d = NAG_ALLOC(n*(n-1)/2, double)) ||
            !(s = NAG_ALLOC(m, double)) ||
            !(x = NAG_ALLOC((n)*(m), double)) ||
            !(isx = NAG_ALLOC(m, 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", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s", nag_enum_arg);
    #endif
    /* nag_enum_name_to_value (x04nac).
        * Converts NAG enum member name to value
        */
    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);

END:
    exit_status = 0;
    return exit_status;
}
```

This code snippet is an example program demonstrating how to use the `nag_mv_distance_mat` function from the `g03eac` library, which is part of the `g03` module for multivariate methods in the NAG C Library.
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, _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
    }

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("%1f", &s[i]);
    #else
        scanf("%1f", &s[i]);
    #endif

/* 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;
}

/* Print the distance matrix */

printf("\n");
printf(" Distance Matrix ");
printf("\n");
printf("\n");
printf(" %s\n", " 1 2 3 4");
printf("\n");
for (i = 2; i <= n; ++i)
{
    printf("%2"NAG_IFMT"", i);
    for (j = (i-1)*(i-2)/2+1; j <= i*(i - 1)/2; ++j)
        printf("%5.2f ", d[j-1]);
    printf("\n");
}

END:
NAG_FREE(d);
NAG_FREE(s);
NAG_FREE(x);
NAG_FREE(isx);

return exit_status;
10.2 Program Data

nag_mv_distance_mat (g03eac) Example Program Data
5 3
Nag_NoMatUp Nag_DistSquared Nag_NoVarScale
1.0 1.0 1.0
2.0 1.0 2.0
3.0 6.0 3.0
4.0 8.0 2.0
5.0 8.0 0.0
0 1 1
1.0 1.0 1.0

10.3 Program Results

nag_mv_distance_mat (g03eac) Example Program Results

Distance Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>29.00</td>
<td>26.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>50.00</td>
<td>49.00</td>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>50.00</td>
<td>53.00</td>
<td>13.00</td>
<td>4.00</td>
</tr>
</tbody>
</table>