# nag_mv_distance_mat (g03eac)

## 1. Purpose

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

## 2. Specification

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

void nag_mv_distance_mat(Nag_MatUpdate update, Nag_DistanceType dist,
            Nag_VarScaleType scale, Integer n, Integer m, double x[],
            Integer tdx, 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:

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

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.

(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: $s_i = \sqrt{\sum_{j=1}^{n}(x_{ji} - \bar{x})^2/(n-1)}$
(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 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 Chapter g02 and Chapter g11 respectively.

nag_mv_distance_mat returns the strictly lower triangle of the distance matrix.

## 4. Parameters

**update**

Input: indicates whether or not an existing matrix is to be updated.

If **update** = **Nag_MatUp**, the matrix $D$ is updated and distances are added to $D$.
If **update** = **Nag_NoMatUp**, the matrix $D$ is initialized to zero before the distances are added to $D$.

Constraint: **update** = **Nag_MatUp** or **Nag_NoMatUp**.

**dist**

Input: indicates which type of distances are computed.

If **dist** = **Nag_DistAbs**, absolute distances.
If **dist** = **Nag_DistEuclid**, Euclidean distances.
If **dist** = **Nag_DistSquared**, Euclidean squared distances.

Constraint: **dist** = **Nag_DistAbs**, **Nag_DistEuclid** or **Nag_DistSquared**.

**scale**

Input: indicates the standardization of the variables to be used.

If **scale** = **Nag_VarScaleStd**, standard deviation.

If **scale** = **Nag_VarScaleRange**, range.

If **scale** = **Nag_VarScaleUser**, standardizations given in array S.

If **scale** = **Nag_NoVarScale**, unscaled.

Constraint: **scale** = **Nag_VarScaleStd**, **Nag_VarScaleRange**, **Nag_VarScaleUser** or **Nag_NoVarScale**.

**n**

Input: the number of observations, $n$.

Constraint: $\mathbf{n} \geq 2$.

**m**

Input: the total number of variables in array **x**.

Constraint: $\mathbf{m} > 0$.

**x[n][tdx]**

Input: $\mathbf{x}[i-1][j-1]$ must contain the value of the $j$th variable for the $i$th object, for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, \mathbf{m}$.

**tdx**

Input: the last dimension of the array **x** as declared in the calling program.
Constraint: $\mathbf{tdx} \geq \mathbf{m}$.

**isx[m]**

Input: $\mathbf{isx}[j-1]$ indicates whether or not the $j$th variable in **x** is to be included in the distance computations.

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

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

**s[m]**

Input: if **scale** = **Nag_VarScaleUser** and $\mathbf{isx}[j-1] > 0$ then $\mathbf{s}[j-1]$ must contain the scaling for variable $j$, for $j = 1, 2, \ldots, \mathbf{m}$.

Constraint: if **scale** = **Nag_VarScaleUser** and $\mathbf{isx}[j-1] > 0$ then $\mathbf{s}[j-1] > 0.0$, for $j = 1, 2, \ldots, \mathbf{m}$.

Output: if **scale** = **Nag_VarScaleStd** and $\mathbf{isx}[j-1] > 0$ then $\mathbf{s}[j-1]$ contains the standard deviation of the variable in the $j$th column of **x**. If **scale** = **Nag_VarScaleRange** and $\mathbf{isx}[j-1] > 0$ then $\mathbf{s}[j-1]$ contains the range of the variable in the $j$th column of **x**. If **scale** = **Nag_NoVarScale** and $\mathbf{isx}[j-1] > 0$ then $\mathbf{s}[j-1] = 1.0$ and if **scale** = **Nag_VarScaleUser** then **s** is unchanged.

**d[n∗(n−1)/2]**

Input: if **update** = **Nag_MatUp** then **d** must contain the strictly lower triangle of the distance matrix $D$ to be updated. $D$ 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 **update** = **Nag_MatUp** then $\mathbf{d}[j-1] \geq 0.0$, for $j = 1, 2, \ldots, n(n-1)/2$.

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

**fail**

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

## 5. Error Indications and Warnings

**NE_BAD_PARAM**

On entry, parameter **dist** had an illegal value.
On entry, parameter **update** had an illegal value.
On entry, parameter **scale** had an illegal value.

**NE_INT_ARG_LT**
On entry, **n** must not be less than 2: **n** = ⟨*value*⟩.

**NE_INT_ARG_LE**
On entry, **m** must not be less than or equal to 0: **m** = ⟨*value*⟩.

**NE_2_INT_ARG_LT**
On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩.
These parameters must satisfy **tdx** $\geq$ **m**.

**NE_INTARR**
On entry, isx[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **isx**$[i-1] > 0$, for at least one $i, i = 1, 2, \ldots,$**m**.

**NE_REALARR**
On entry, **d**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **d**$[i-1] \geq 0$, $i = 1, 2, \ldots,$**n**$*$(**n**$-1)/2$ when **update** = **Nag_MatUp**.
On entry, **s**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **s**$[j-1] > 0$, $j = 1, 2, \ldots,$**m** when **scale** = **Nag_VarScaleUser** and **isx**$[j-1] > 0$.

**NE_IDEN_ELEM_COND**
On entry, **scale** = **Nag_VarScaleRange** or **scale** = **Nag_VarScaleStd**, and **x**$[i-1][j-1]$ = **x**$[i][j-1]$, for $i = 1, 2, \ldots, n-1$, for some $j$ with **isx**$[i-1] > 0$.

**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

nag_mv_hierar_cluster_analysis (g03ecc) can be used to perform cluster analysis on the computed distance matrix.

### 6.1. Accuracy

The computations are believed to be stable.

### 6.2. References

Everitt B S (1974) *Cluster Analysis* Heinemann.
Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press.

## 7. See Also

nag_mv_hierar_cluster_analysis (g03ecc)
nag_mv_prin_coord_analysis (g03fac)

## 8. Example

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.

### 8.1. Program Text

```
/* nag_mv_distance_mat (g03eac) Example Program.
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 *
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>
```

```
#define NMAX 10
#define MMAX 10

main()
{
  double d[NMAX*(NMAX-1)/2], s[MMAX], x[NMAX][MMAX];

  Integer isx[MMAX];
  Integer i, j, m, n;
  Integer tdx=MMAX;

  char char_scale[2];
  char char_update[2];
  char char_dist[2];

  Nag_MatUpdate update;
  Nag_DistanceType dist;
  Nag_VarScaleType scale;

  Vprintf("g03eac Example Program Results\n\n");

  /*  Skip heading in data file */
  Vscanf("%*[^\n]");

  Vscanf("%ld",&n);
  Vscanf("%ld",&m);
  if (n <= NMAX && m <= MMAX)
    {
      Vscanf("%s",char_update);
      Vscanf("%s",char_dist);
      Vscanf("%s",char_scale);
      for (j = 0; j < n; ++j)
        {
          for (i = 0; i < m; ++i)
            Vscanf("%lf",&x[j][i]);
        }
      for (i = 0; i < m; ++i)
        Vscanf("%ld",&isx[i]);
      for (i = 0; i < m; ++i)
        Vscanf("%lf",&s[i]);

      /*  Compute the distance matrix */
      if (*char_update == 'U')
        update = Nag_MatUp;
      else if (*char_update == 'I')
        update = Nag_NoMatUp;

      if (*char_dist == 'A')
        dist = Nag_DistAbs;
      else if (*char_dist == 'E')
        dist = Nag_DistEuclid;
      else if (*char_dist == 'S')
        dist = Nag_DistSquared;

      if (*char_scale == 'S')
        scale = Nag_VarScaleStd;
      else if (*char_scale == 'R')
        scale = Nag_VarScaleRange;
      else if (*char_scale == 'G')
        scale = Nag_VarScaleUser;
      else if (*char_scale == 'U')
        scale = Nag_NoVarScale;

      g03eac(update, dist, scale, n, m, (double *)x, tdx, isx, s, d, NAGERR_DEFAULT);

      /*  Print the distance matrix */

      Vprintf("\n");
      Vprintf(" Distance Matrix ");
      Vprintf("\n");
```

```
        Vprintf("\n");
        Vprintf("    %s\n","   1       2       3       4");
        Vprintf("\n");
        for (i = 2; i <= n; ++i)
          {
            Vprintf("%2ld     ",i);
            for (j=(i-1)*(i-2)/2+1; j<=i*(i - 1)/2; ++j)
              Vprintf("%5.2f   ",d[j-1]);
            Vprintf("\n");
          }
        exit(EXIT_SUCCESS);
      }
    else
      {
        Vprintf("Incorrect input value of n or m.\n");
        exit(EXIT_FAILURE);
      }
}
```

### 8.2. Program Data

```
g03eac Example Program Data
5 3
I S U
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
```

### 8.3. Program Results

```
g03eac Example Program Results


 Distance Matrix

       1       2       3       4

 2      1.00
 3     29.00   26.00
 4     50.00   49.00    5.00
 5     50.00   53.00   13.00    4.00
```