NAG Library Function Document
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, 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 ijth element represents how far apart or how dissimilar the ith and jth 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:

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

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 ith 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: $s_i = \sqrt{\sum_{j=1}^{n}\left(x_{ji} - \bar{x}\right)^2/(n-1)}$
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

Everitt B S (1974) *Cluster Analysis* Heinemann

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

5 Arguments

1: **update** – Nag_MatUpdate *Input*

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

update = Nag_MatUp
The matrix D is updated and distances are added to D.

update = Nag_NoMatUp
The matrix D is initialized to zero before the distances are added to D.

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

2: **dist** – Nag_DistanceType *Input*

On entry: indicates which type of distances are computed.

dist = Nag_DistAbs
Absolute distances.

dist = Nag_DistEuclid
Euclidean distances.

dist = Nag_DistSquared
Euclidean squared distances.

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

3: **scale** – Nag_VarScaleType *Input*

On entry: indicates the standardization of the variables to be used.

scale = Nag_VarScaleStd
Standard deviation.

scale = Nag_VarScaleRange
Range.

scale = Nag_VarScaleUser
Standardizations given in array S.

scale = Nag_NoVarScale
Unscaled.

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

4: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: **n** ≥ 2.

5: **m** – Integer *Input*

On entry: the total number of variables in array **x**.

Constraint: **m** > 0.

6: **x**[**n** × **tdx**] – const double *Input*

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

7: **tdx** – Integer *Input*

On entry: the stride separating matrix column elements in the array **x**.

Constraint: **tdx** $\geq$ **m**.

8: **isx**[**m**] – const Integer *Input*

On entry: **isx**$[j-1]$ indicates whether or not the jth variable in **x** is to be included in the distance computations.

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

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

9: **s**[**m**] – double *Input/Output*

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

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

On exit: if **scale** = Nag_VarScaleStd and **isx**$[j-1] > 0$ then **s**$[j-1]$ contains the standard deviation of the variable in the jth column of **x**.

If **scale** = Nag_VarScaleRange and **isx**$[j-1] > 0$ then **s**$[j-1]$ contains the range of the variable in the jth column of **x**.

If **scale** = Nag_NoVarScale and **isx**$[j-1] > 0$ then **s**$[j-1] = 1.0$ and if **scale** = Nag_VarScaleUser then **s** is unchanged.

10: **d**[**n** × (**n** − **1**)/**2**] – double *Input/Output*

On entry: 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., **d**$[(i-1)(i-2)/2 + j - 1]$, $i > j$ must contain d_{ij}.

Constraint: if **update** = Nag_MatUp, **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 D stored packed by rows, i.e., d_{ij} is contained in **d**$[(i-1)(i-2)/2 + j - 1]$, $i > j$.

11: **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, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩. These arguments must satisfy **tdx** $\geq$ **m**.

NE_BAD_PARAM

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

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

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

NE_IDEN_ELEM_COND

On entry, **scale** = Nag_VarScaleRange or **scale** = Nag_VarScaleStd, and $\mathbf{x}[(i-1) \times \mathbf{tdx} + j - 1] = \mathbf{x}[(i) \times \mathbf{tdx} + j - 1]$, for $i = 1, 2, \ldots, n-1$, for some j with $\mathbf{isx}[i-1] > 0$.

NE_INT_ARG_LE

On entry, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} > 0$.

NE_INT_ARG_LT

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

NE_INTARR

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

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: $\mathbf{d}[i-1] \geq 0$, for $i = 1, 2, \ldots, \mathbf{n} \times (\mathbf{n} - 1)/2$, when **update** = Nag_MatUp.

On entry, $\mathbf{s}[\langle value \rangle] = \langle value \rangle$.
Constraint: $\mathbf{s}[j-1] > 0$, for $j = 1, 2, \ldots, \mathbf{m}$, when **scale** = Nag_VarScaleUser and $\mathbf{isx}[j-1] > 0$.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

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

10 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.

10.1 Program Text

```
/* nag_mv_distance_mat (g03eac) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 * Mark 8 revised, 2004.
 *
 */
```

```
#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);
```

```
#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("%lf", &s[i]);
#else
    scanf("%lf", &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

       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
```