nag_mv_discrim_mahaldist (g03dbc)

1. Purpose

nag_mv_discrim_mahaldist (g03dbc) computes Mahalanobis squared distances for group or pooled variance-covariance matrices. It is intended for use after nag_mv_discrim (g03dac).

2. Specification

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

void nag_mv_discrim_mahaldist(Nag_GroupCovars equal, Nag_MahalDist mode,
            Integer nvar, Integer ng, double gmean[], Integer tdg, double gc[],
            Integer nobs, Integer m, Integer isx[], double x[], Integer tdx,
            double d[], Integer tdd, NagError *fail)
```

3. Description

Consider p variables observed on n_g populations or groups. Let $\bar{x}_j$ be the sample mean and S_j the within-group variance-covariance matrix for the jth group and let x_k be the kth sample point in a data set. A measure of the distance of the point from the jth population or group is given by the Mahalanobis distance, D^2_{kj}:

$$D^2_{kj} = (x_k - \bar{x}_j)^T S_j^{-1}(x_k - \bar{x}_j).$$

If the pooled estimated of the variance-covariance matrix S is used rather than the within-group variance-covariance matrices, then the distance is:

$$D^2_{kj} = (x_k - \bar{x}_j)^T S^{-1}(x_k - \bar{x}_j).$$

Instead of using the variance-covariance matrices S and S_j, nag_mv_discrim_mahaldist uses the upper triangular matrices R and R_j supplied by nag_mv_discrim (g03dac) such that $S = R^T R$ and $S_j = R_j^T R_j$. D^2_{kj} can then be calculated as $z^T z$ where $R_j z = (x_k - \bar{x}_j)$ or $Rz = (x_k - \bar{x}_j)$ as appropriate.

A particular case is when the distance between the group or population means is to be estimated. The Mahalanobis distance between the ith and jth groups is:

$$D^2_{ij} = (\bar{x}_i - \bar{x}_j)^T S_j^{-1}(\bar{x}_i - \bar{x}_j)$$

or

$$D^2_{ij} = (\bar{x}_i - \bar{x}_j)^T S^{-1}(\bar{x}_i - \bar{x}_j).$$

Note: $D^2_{jj} = 0$ and that in the case when the pooled variance-covariance matrix is used $D^2_{ij} = D^2_{ji}$ so in this case only the lower triangular values of D^2_{ij}, $i > j$, are computed.

4. Parameters

equal

Input: indicates whether or not the within-group variance-covariance matrices are assumed to be equal and the pooled variance-covariance matrix used.

If **equal** = **Nag_EqualCovar** the within-group variance-covariance matrices are assumed equal and the matrix R stored in the first $p(p+1)/2$ elements of **gc** is used.

If **equal** = **Nag_NotEqualCovar** the within-group variance-covariance matrices are assumed to be unequal and the matrices R_j, for $j = 1, 2, \ldots, n_g$, stored in the remainder of **gc** are used.

Constraint: **equal** = **Nag_EqualCovar** or **Nag_NotEqualCovar**.

mode

Input: indicates whether distances from sample points are to be calculated or distances between the group means.

If **mode** = **Nag_SamplePoints** the distances between the sample points given in **x** and the group means are calculated.

If **mode** = **Nag_GroupMeans** the distances between the group means will be calculated.

Constraint: **mode** = **Nag_SamplePoints** or **Nag_GroupMeans**.

nvar

Input: the number of variables, p, in the variance-covariance matrices as specified to nag_mv_discrim (g03dac).

Constraint: **nvar** ≥ 1.

ng

Input: the number of groups, n_g.

Constraint: **ng** ≥ 2.

gmean[ng][tdg]

Input: the jth row of **gmean** contains the means of the p selected variables for the jth group, for $j = 1, 2, \ldots, n_g$. These are returned by nag_mv_discrim (g03dac).

tdg

Input: the last dimension of the array **gmean** as declared in the calling program.
Constraint: **tdg** $\geq$ **nvar**.

gc[(ng+1)*nvar*(nvar+1)/2]

Input: the first $p(p+1)/2$ elements of **gc** should contain the upper triangular matrix R and the next n_g blocks of $p(p+1)/2$ elements should contain the upper triangular matrices R_j. All matrices must be stored packed by column. These matrices are returned by nag_mv_discrim (g03dac). If **equal** = **Nag_EqualCovar** only the first $p(p+1)/2$ elements are referenced, if **equal** = **Nag_NotEqualCovar** only the elements $p(p+1)/2$ to $(n_g+1)p(p+1)/2-1$ are referenced.

Constraints:

If **equal** = **Nag_EqualCovar** the diagonal elements of $R \neq 0.0$.
If **equal** = **Nag_NotEqualCovar** the diagonal elements of the $R_j \neq 0.0$, for $j = 1, 2, \ldots,$**ng**.

nobs

Input: if **mode** = **Nag_SamplePoints** the number of sample points in **x** for which distances are to be calculated. If **mode** = **Nag_GroupMeans**, **nobs** is not referenced.

Constraint: if **mode** = **Nag_SamplePoints**, **nobs** ≥ 1.

m

Input: if **mode** = **Nag_SamplePoints** the number of variables in the data array **x**. If **mode** = **Nag_GroupMeans**, then **m** is not referenced.

Constraint: if **mode** = **Nag_SamplePoints**, **m** $\geq$ **nvar**.

isx[m]

Input: if **mode** = **Nag_SamplePoints**, **isx**$[l-1]$ indicates if the lth variable in **x** is to be included in the distance calculations. If **isx**$[l-1] > 0$, the lth variable is included, for $l = 1, 2, \ldots,$ **m**; otherwise the lth variable is not referenced.

If **mode** = **Nag_GroupMeans**, then **isx** is not referenced and may be set to the NULL pointer (Integer *)0.

Constraint: if **mode** = **Nag_SamplePoints**, **isx**$[l-1] > 0$ for **nvar** values of l.

x[nobs][tdx]

Input: if **mode** = **Nag_SamplePoints** the kth row of **x** must contain x_k. That is, **x**$[k-1][l-1]$ must contain the kth sample value for the lth variable for $k = 1, 2, \ldots,$**nobs**; $l = 1, 2, \ldots,$**m**. Otherwise **x** is not referenced and may be set to the NULL pointer (double *)0.

tdx

Input: the last dimension of the array **x** as declared in the calling program.
Constraint: **tdx** $\geq$ max(1, **m**).

d[*dim1*][**tdd**]

Output: the squared distances.

If **mode** = **Nag_SamplePoints**, **d**$[k-1][j-1]$ contains the squared distance of the kth sample point from the jth group mean, D^2_{kj}, for $k = 1, 2, \ldots,$**nobs**; $j = 1, 2, \ldots, n_g$.
If **mode** = **Nag_GroupMeans** and **equal** = **Nag_NotEqualCovar**, **d**$[i-1][j-1]$ contains the squared distance between the ith mean and the jth mean, D^2_{ij}, for $i = 1, 2, \ldots, n_g$; $j = 1, 2, \ldots, i-1, i+1, \ldots, n_g$. The elements **d**$[i-1][i-1]$ are not referenced for $i = 1, 2, \ldots, n_g$.
If **mode** = **Nag_GroupMeans** and **equal** = **Nag_EqualCovar**, **d**$[i-1][j-1]$ contains the squared distance between the ith mean and the jth mean, D^2_{ij}, for $i = 1, 2, \ldots, n_g$; $j = 1, 2, \ldots, i-1$. Since $D_{ij} = D_{ji}$ the elements **d**$[i-1][j-1]$ are not referenced, for $i = 1, 2, \ldots, n_g$; $j = i, i+1, \ldots, n_g$.

Constraint: *dim1* must be $\geq$ **nobs** if **mode** = **Nag_SamplePoints**, otherwise *dim1* must be $\geq$ **ng**.

tdd

Input: the last dimension of the array **dd** as declared in the calling program.
Constraint: **tdd** $\geq$ **ng**.

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 **equal** had an illegal value.
On entry, parameter **mode** had an illegal value.

NE_INT_ARG_LT

On entry, **nvar** must not be less than 1: **nvar** = ⟨*value*⟩.
On entry, **ng** must not be less than 2: **ng** = ⟨*value*⟩.

NE_2_INT_ARG_LT

On entry, **tdg** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩.
These parameters must satisfy **tdg** $\geq$ **nvar**.
On entry, **tdd** = ⟨*value*⟩ while **ng** = ⟨*value*⟩.
These parameters must satisfy **tdd** $\geq$ **ng**.

NE_INT_ARG_ENUM_CONS

On entry, **nobs** = ⟨*value*⟩ while **mode** = **Nag_SamplePoints**.
These parameters must satisfy **nobs** $\geq$ 1 when **mode** = **Nag_SamplePoints**.

NE_2_INT_ARG_ENUM_CONS

On entry, **m** = ⟨*value*⟩ while **nvar** = ⟨*value*⟩ and **mode** = **Nag_SamplePoints**.
These parameters must satisfy **m** $\geq$ **nvar** when **mode** = **Nag_SamplePoints**.

On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩ and **mode** = **Nag_SamplePoints**.
These parameters must satisfy **tdx** $\geq$ max(1,**m**) when **mode** = **Nag_SamplePoints**.

NE_VAR_INCL_COND

The number of variables, **nvar** in the analysis = ⟨*value*⟩, while number of variables included in the analysis via array **isx** = ⟨*value*⟩.
Constraint: These two numbers must be the same when **mode** = **Nag_SamplePoints**.

NE_DIAG_0_COND

A diagonal element of R is zero when **equal** = **Nag_EqualCovar**.

NE_DIAG_0_J_COND

A diagonal element of R is zero for some j, when **equal** = **Nag_NotEqualCovar**.

NE_ALLOC_FAIL
Memory allocation failed.

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

If the distances are to be used for discrimination, see also nag_mv_discrim_group (g03dcc).

6.1. Accuracy

The accuracy will depend upon the accuracy of the input R or R_j matrices.

6.2. References

Aitchison J and Dunsmore I R (1975) *Statistical Prediction Analysis* Cambridge.
Kendall M G and Stuart A (1976) *The Advanced Theory of Statistics (Volume 3)* Griffin (3rd Edition).
Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press.

7. See Also

nag_mv_discrim (g03dac)
nag_mv_discrim_group (g03dcc)

8. Example

The data, taken from Aitchison and Dunsmore (1975), is concerned with the diagnosis of three 'types' of Cushing's syndrome. The variables are the logarithms of the urinary excretion rates (mg/24hr) of two steroid metabolites. Observations for a total of 21 patients are input and the group means and R matrices are computed by nag_mv_discrim (g03dac). A further six observations of unknown type are input, and the distances from the group means of the 21 patients of known type are computed under the assumption that the within-group variance-covariance matrices are not equal. These results are printed and indicate that the first four are close to one of the groups while observations 5 and 6 are some distance from any group.

8.1. Program Text

```
/* nag_mv_discrim_mahaldist (g03dbc) 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 21
#define MMAX 2
#define GPMAX 3

main()
{
  double d[NMAX][GPMAX], det[GPMAX],
  gc[(GPMAX+1)*MMAX*(MMAX+1)/2], gmean[GPMAX][MMAX],
  wt[NMAX], x[NMAX][MMAX];
  double stat;
  double df;
  double sig;
  double *wtptr=0;
```

```
  Integer nobs, nvar;
  Integer  ing[NMAX], isx[MMAX], nig[GPMAX];
  Integer i, j, m, n;
  Integer ng;
  Integer tdd=GPMAX, tdgmean=MMAX, tdx=MMAX;

  char char_equal[2];
  char weight[2];

  Nag_GroupCovars equal;

  Vprintf("g03dbc Example Program Results\n\n");

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

  Vscanf("%ld",&n);
  Vscanf("%ld",&m);
  Vscanf("%ld",&nvar);
  Vscanf("%ld",&ng);
  Vscanf("%s",weight);

  if (n <= NMAX && m <= MMAX)
    {
      if (*weight == 'W')
        {
          for (i = 0; i < n; ++i)
            {
              for (j = 0; j < m; ++j)
                Vscanf("%lf",&x[i][j]);
              Vscanf("%ld",&ing[i]);
              Vscanf("%lf",&wt[i]);
            }
          wtptr = wt;
        }
      else
        {
          for (i = 0; i < n; ++i)
            {
              for (j = 0; j < m; ++j)
                Vscanf("%lf",&x[i][j]);
              Vscanf("%ld",&ing[i]);
            }
        }
      for (j = 0; j < m; ++j)
        Vscanf("%ld",&isx[j]);
      g03dac(n, m, (double *)x, tdx, isx, nvar, ing, ng, wtptr, nig,
             (double *)gmean, tdgmean, det, gc, &stat, &df, &sig, NAGERR_DEFAULT);
      Vscanf("%ld",&nobs);
      Vscanf("%s",char_equal);
      if (nobs <= NMAX)
        {
          for (i = 0; i < nobs; ++i)
            {
              for (j = 0; j < m; ++j)
                Vscanf("%lf",&x[i][j]);
            }
          if (*char_equal == 'E')
            {
              equal = Nag_EqualCovar;
            }
          else if (*char_equal == 'U')
            {
              equal = Nag_NotEqualCovar;
            }

          g03dbc(equal, Nag_SamplePoints, nvar, ng, (double *)gmean, tdgmean, gc,
                 nobs, m, isx, (double *)x, tdx, (double *)d, tdd, NAGERR_DEFAULT);
          Vprintf("\n   Obs          Distances\n\n");
          for (i = 0; i < nobs; ++i)
```

```
              {
                Vprintf(" %3ld",i+1);
                for (j = 0; j < ng; ++j)
                  Vprintf("%10.3f",d[i][j]);
                Vprintf("\n");
              }
          }
        exit(EXIT_SUCCESS);
      }
    else
      {
        Vprintf("Incorrect input value of n or m.\n");
        exit(EXIT_FAILURE);
      }
}
```

8.2. Program Data

```
g03dbc Example Program Data
  21 2 2 3 U
  1.1314    2.4596    1
  1.0986    0.2624    1
  0.6419   -2.3026    1
  1.3350   -3.2189    1
  1.4110    0.0953    1
  0.6419   -0.9163    1
  2.1163    0.0000    2
  1.3350   -1.6094    2
  1.3610   -0.5108    2
  2.0541    0.1823    2
  2.2083   -0.5108    2
  2.7344    1.2809    2
  2.0412    0.4700    2
  1.8718   -0.9163    2
  1.7405   -0.9163    2
  2.6101    0.4700    2
  2.3224    1.8563    3
  2.2192    2.0669    3
  2.2618    1.1314    3
  3.9853    0.9163    3
  2.7600    2.0281    3
   1         1
   6 U
  1.6292   -0.9163
  2.5572    1.6094
  2.5649   -0.2231
  0.9555   -2.3026
  3.4012   -2.3026
  3.0204   -0.2231
```

8.3. Program Results

```
g03dbc Example Program Results


   Obs          Distances

   1     3.339     0.752    50.928
   2    20.777     5.656     0.060
   3    21.363     4.841    19.498
   4     0.718     6.280   124.732
   5    55.000    88.860    71.785
   6    36.170    15.785    15.749
```