

## NAG Library Function Document

### 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, const double gmean[], Integer tdg,
    const double gc[], Integer nob, Integer m, const Integer isx[],
    const 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  $j$ th group and let  $x_k$  be the  $k$ th sample point in a dataset. A measure of the distance of the point from the  $j$ th population or group is given by the Mahalanobis distance,  $D_{kj}^2$ :

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

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

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

Instead of using the variance-covariance matrices  $S$  and  $S_j$ , nag\_mv\_discrim\_mahaldist (g03dbc) uses the upper triangular matrices  $R$  and  $R_j$  supplied by nag\_mv\_discrim (g03dac) such that  $S = R^T R$  and  $S_j = R_j^T R_j$ .  $D_{kj}^2$  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  $i$ th and  $j$ th groups is:

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

or

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

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

#### 4 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)* (3rd Edition) Griffin

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

## 5 Arguments

- 1: **equal** – Nag\_GroupCovars *Input*  
*On entry:* indicates whether or not the within-group variance-covariance matrices are assumed to be equal and the pooled variance-covariance matrix used.  
**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.  
**equal** = Nag\_NotEqualCovar  
 The within-group variance-covariance matrices are assumed to be unequal and the matrices  $R_j$ , for  $j = 1, 2, \dots, n_g$ , stored in the remainder of **gc** are used.  
*Constraint:* **equal** = Nag\_EqualCovar or Nag\_NotEqualCovar.
- 2: **mode** – Nag\_MahalDist *Input*  
*On entry:* indicates whether distances from sample points are to be calculated or distances between the group means.  
**mode** = Nag\_SamplePoints  
 The distances between the sample points given in **x** and the group means are calculated.  
**mode** = Nag\_GroupMeans  
 The distances between the group means will be calculated.  
*Constraint:* **mode** = Nag\_SamplePoints or Nag\_GroupMeans.
- 3: **nvar** – Integer *Input*  
*On entry:* the number of variables,  $p$ , in the variance-covariance matrices as specified to nag\_mv\_discrim (g03dac).  
*Constraint:* **nvar**  $\geq 1$ .
- 4: **ng** – Integer *Input*  
*On entry:* the number of groups,  $n_g$ .  
*Constraint:* **ng**  $\geq 2$ .
- 5: **gmean**[**ng**  $\times$  **tdg**] – const double *Input*  
**Note:** the  $(i, j)$ th element of the matrix is stored in **gmean**[( $i - 1$ )  $\times$  **tdg** +  $j - 1$ ].  
*On entry:* the  $j$ th row of **gmean** contains the means of the  $p$  selected variables for the  $j$ th group, for  $j = 1, 2, \dots, n_g$ . These are returned by nag\_mv\_discrim (g03dac).
- 6: **tdg** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **gmean**.  
*Constraint:* **tdg**  $\geq$  **nvar**.
- 7: **gc**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **gc** must be at least  $(\mathbf{ng} + 1) \times \mathbf{nvar} \times (\mathbf{nvar} + 1)/2$ .  
*On entry:* 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  $j = 1, 2, \dots, n_g$ .

8: **nobs** – Integer *Input*

*On entry:* if **mode** = Nag\_SamplePoints the number of sample points in  $\mathbf{x}$  for which distances are to be calculated.

If **mode** = Nag\_GroupMeans, **nobs** is not referenced.

*Constraint:* if **mode** = Nag\_SamplePoints, **nobs**  $\geq 1$ .

9: **m** – Integer *Input*

*On entry:* if **mode** = Nag\_SamplePoints the number of variables in the data array  $\mathbf{x}$ .

If **mode** = Nag\_GroupMeans, then **m** is not referenced.

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

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

*On entry:* if **mode** = Nag\_SamplePoints, **isx[l-1]** indicates if the  $l$ th variable in  $\mathbf{x}$  is to be included in the distance calculations. If **isx[l-1]**  $> 0$ , the  $l$ th variable is included, for  $l = 1, 2, \dots, \mathbf{m}$ ; otherwise the  $l$ th 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  $l$ .

11: **x[nobs  $\times$  tdx]** – const double *Input*

*On entry:* if **mode** = Nag\_SamplePoints, the  $k$ th row of  $\mathbf{x}$  must contain  $x_k$ . That is, **x[(k-1)  $\times$  tdx + l - 1]** must contain the  $k$ th sample value for the  $l$ th variable for  $k = 1, 2, \dots, \mathbf{nobs}$  and  $l = 1, 2, \dots, \mathbf{m}$ . Otherwise  $\mathbf{x}$  is not referenced and may be set to the NULL pointer (double \*)0.

12: **tdx** – Integer *Input*

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

*Constraint:* **tdx**  $\geq$   $\max(1, \mathbf{m})$ .

13: **d[dim1  $\times$  tdd]** – double *Output*

*On exit:* the squared distances.

If **mode** = Nag\_SamplePoints, **d[(k-1)  $\times$  tdd + j - 1]** contains the squared distance of the  $k$ th sample point from the  $j$ th group mean,  $D_{kj}^2$ , for  $k = 1, 2, \dots, \mathbf{nobs}$  and  $j = 1, 2, \dots, n_g$ .

If **mode** = Nag\_GroupMeans and **equal** = Nag\_NotEqualCovar, **d[(i-1)  $\times$  tdd + j - 1]** contains the squared distance between the  $i$ th mean and the  $j$ th mean,  $D_{ij}^2$ , for  $i = 1, 2, \dots, n_g$  and  $j = 1, 2, \dots, i-1, i+1, \dots, n_g$ . The elements **d[(i-1)  $\times$  tdd + i - 1]** are not referenced, for  $i = 1, 2, \dots, n_g$ .

If **mode** = Nag\_GroupMeans and **equal** = Nag\_EqualCovar, **d[(i-1)  $\times$  tdd + j - 1]** contains the squared distance between the  $i$ th mean and the  $j$ th mean,  $D_{ij}^2$ , for  $i = 1, 2, \dots, n_g$  and  $j = 1, 2, \dots, i-1$ . Since  $D_{ij} = D_{ji}$  the elements **d[(i-1)  $\times$  tdd + j - 1]** are not referenced, for  $i = 1, 2, \dots, n_g$  and  $j = i, \dots, n_g$ .

- 14: **tdd** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **d**.  
*Constraint:* **tdd**  $\geq$  **ng**.
- 15: **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\_ENUM\_CONS

On entry, **m** =  $\langle value \rangle$  while **nvar** =  $\langle value \rangle$  and **mode** = Nag\_SamplePoints. These arguments must satisfy **m**  $\geq$  **nvar** when **mode** = Nag\_SamplePoints.

On entry, **tdx** =  $\langle value \rangle$  while **m** =  $\langle value \rangle$  and **mode** = Nag\_SamplePoints. These arguments must satisfy **tdx**  $\geq$  max(1,**m**) when **mode** = Nag\_SamplePoints.

### NE\_2\_INT\_ARG\_LT

On entry, **tdd** =  $\langle value \rangle$  while **ng** =  $\langle value \rangle$ . These arguments must satisfy **tdd**  $\geq$  **ng**.

On entry, **tdg** =  $\langle value \rangle$  while **nvar** =  $\langle value \rangle$ . These arguments must satisfy **tdg**  $\geq$  **nvar**.

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_BAD\_PARAM

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

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

### 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\_INT\_ARG\_ENUM\_CONS

On entry, **nobs** =  $\langle value \rangle$  while **mode** = Nag\_SamplePoints. These arguments must satisfy **nobs**  $\geq$  1 when **mode** = Nag\_SamplePoints.

### NE\_INT\_ARG\_LT

On entry, **ng** =  $\langle value \rangle$ .  
 Constraint: **ng**  $\geq$  2.

On entry, **nvar** =  $\langle value \rangle$ .  
 Constraint: **nvar**  $\geq$  1.

### 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\_VAR\_INCL\_COND**

The number of variables, **nvar** in the analysis =  $\langle value \rangle$ , while number of variables included in the analysis via array **isx** =  $\langle value \rangle$ .

Constraint: These two numbers must be the same when **mode** = Nag\_SamplePoints.

**7 Accuracy**

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

**8 Parallelism and Performance**

Not applicable.

**9 Further Comments**

If the distances are to be used for discrimination, see also nag\_mv\_discrim\_group (g03dcc).

**10 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.

**10.1 Program Text**

```

/* nag_mv_discrim_mahaldist (g03dbc) 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 D(I, J)      d[(I) *tdd + J]
#define GMEAN(I, J) gmean[(I) *tdgmean + J]
#define X(I, J)      x[(I) *tdx + J]

int main(void)
{
    Integer      i, j, m, n, ng, nobs, nvar, tdd, tdgmean, tdx;
    Integer      exit_status = 0, *ing = 0, *isx = 0, *nig = 0;
    char         nag_enum_arg[40];
    double       df, sig, stat;
    double       *d = 0, *det = 0, *gc = 0, *gmean = 0, *wt = 0;
    double       *wtptr = 0, *x = 0;
    Nag_GroupCovars equal;
    Nag_Boolean  weight;
    Nag_Error    fail;

    INIT_FAIL(fail);

```

```

printf(
    "nag_mv_discrim_mahaldist (g03dbc) Example Program Results\n\n");

/* Skip headings 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("%"NAG_IFMT"", &nvar);
#else
scanf("%"NAG_IFMT"", &nvar);
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"", &ng);
#else
scanf("%"NAG_IFMT"", &ng);
#endif
#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
 */
weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);

if (n >= 1 && nvar >= 1 && m >= nvar && ng >= 2)
{
    if (!(det = NAG_ALLOC(ng, double)) ||
        !(gc = NAG_ALLOC((ng+1)*nvar*(nvar+1)/2, double)) ||
        !(gmean = NAG_ALLOC(ng*nvar, double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(x = NAG_ALLOC(n*m, double)) ||
        !(ing = NAG_ALLOC(n, Integer)) ||
        !(isx = NAG_ALLOC(m, Integer)) ||
        !(nig = NAG_ALLOC(ng, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tdgmean = nvar;
    tdx = m;
}
else
{
    printf("Invalid n or nvar or m or ng.\n");
    exit_status = 1;
    return exit_status;
}
if (weight)
{
    for (i = 0; i < n; ++i)
    {
        for (j = 0; j < m; ++j)
#ifdef _WIN32
scanf_s("%lf", &X(i, j));
#else

```

```

        scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
        scanf_s("%"NAG_IFMT"", &ing[i]);
#else
        scanf("%"NAG_IFMT"", &ing[i]);
#endif
#ifdef _WIN32
        scanf_s("%lf", &wt[i]);
#else
        scanf("%lf", &wt[i]);
#endif
    }
    wtptr = wt;
}
else
{
    for (i = 0; i < n; ++i)
    {
        for (j = 0; j < m; ++j)
#ifdef _WIN32
            scanf_s("%lf", &X(i, j));
#else
            scanf("%lf", &X(i, j));
#endif
#ifdef _WIN32
            scanf_s("%"NAG_IFMT"", &ing[i]);
#else
            scanf("%"NAG_IFMT"", &ing[i]);
#endif
        }
    for (j = 0; j < m; ++j)
#ifdef _WIN32
        scanf_s("%"NAG_IFMT"", &isx[j]);
#else
        scanf("%"NAG_IFMT"", &isx[j]);
#endif
    /* nag_mv_discrim (g03dac).
     * Test for equality of within-group covariance matrices
     */
    nag_mv_discrim(n, m, x, tdx, isx, nvar, ing, ng, wtptr, nig,
                  gmean, tdgmean, det, gc, &stat, &df, &sig, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_mv_discrim (g03dac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"", &nobs);
#else
    scanf("%"NAG_IFMT"", &nobs);
#endif
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s", nag_enum_arg);
#endif
    equal = (Nag_GroupCovars) nag_enum_name_to_value(nag_enum_arg);
    if (nobs >= 1)
    {
        if (!(d = NAG_ALLOC(nobs*ng, double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
        tdd = ng;

        for (i = 0; i < nobs; ++i)

```

```

        {
            for (j = 0; j < m; ++j)
#ifdef _WIN32
                scanf_s("%lf", &X(i, j));
#else
                scanf("%lf", &X(i, j));
#endif
        }

/* nag_mv_discrim_mahaldist (g03dbc).
 * Mahalanobis squared distances, following nag_mv_discrim
 * (g03dac)
 */
nag_mv_discrim_mahaldist(equal, Nag_SamplePoints, nvar, ng, gmean,
                        tdgmean, gc, nobs, m, isx, x, tdx, d, tdd,
                        &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_mv_discrim_mahaldist (g03dbc).\n%s\n",
          fail.message);
    exit_status = 1;
    goto END;
}
printf("\n  Obs          Distances\n\n");
for (i = 0; i < nobs; ++i)
{
    printf(" %3"NAG_IFMT", i+1);
    for (j = 0; j < ng; ++j)
        printf("%10.3f", D(i, j));
    printf("\n");
}
}

END:
NAG_FREE(d);
NAG_FREE(det);
NAG_FREE(gc);
NAG_FREE(gmean);
NAG_FREE(wt);
NAG_FREE(x);
NAG_FREE(ing);
NAG_FREE(isx);
NAG_FREE(nig);

return exit_status;
}

```

## 10.2 Program Data

nag\_mv\_discrim\_mahaldist (g03dbc) Example Program Data

21	2	2	3	Nag_FALSE
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 Nag_NotEqualCovar
1.6292   -0.9163
2.5572    1.6094
2.5649   -0.2231
0.9555   -2.3026
3.4012   -2.3026
3.0204   -0.2231

```

### 10.3 Program Results

nag\_mv\_discrim\_mahaldist (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

---