

# NAG Library Function Document

## nag\_mv\_discrim (g03dac)

### 1 Purpose

nag\_mv\_discrim (g03dac) computes a test statistic for the equality of within-group covariance matrices and also computes matrices for use in discriminant analysis.

### 2 Specification

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

void nag_mv_discrim (Integer n, Integer m, const double x[], Integer tdx,
  const Integer isx[], Integer nvar, const Integer ing[], Integer ng,
  const double wt[], Integer nig[], double gmean[], Integer tdg,
  double det[], double gc[], double *stat, double *df, double *sig,
  NagError *fail)
```

### 3 Description

Let a sample of  $n$  observations on  $p$  variables come from  $n_g$  groups with  $n_j$  observations in the  $j$ th group and  $\sum n_j = n$ . If the data is assumed to follow a multivariate Normal distribution with the variance-covariance matrix of the  $j$ th group  $\Sigma_j$ , then to test for equality of the variance-covariance matrices between groups, that is,  $\Sigma_1 = \Sigma_2 = \dots = \Sigma_{n_g} = \Sigma$ , the following likelihood-ratio test statistic,  $G$ , can be used;

$$G = C \left\{ (n - n_g) \log |S| - \sum_{j=1}^{n_g} (n_j - 1) \log |S_j| \right\},$$

where

$$C = 1 - \frac{2p^2 + 3p - 1}{6(p+1)(n_g - 1)} \left( \sum_{j=1}^{n_g} \frac{1}{(n_j - 1)} - \frac{1}{(n - n_g)} \right),$$

and  $S_j$  are the within-group variance-covariance matrices and  $S$  is the pooled variance-covariance matrix given by

$$S = \frac{\sum_{j=1}^{n_g} (n_j - 1) S_j}{(n - n_g)}.$$

For large  $n$ ,  $G$  is approximately distributed as a  $\chi^2$  variable with  $\frac{1}{2}p(p+1)(n_g - 1)$  degrees of freedom, see Morrison (1967) for further comments. If weights are used, then  $S$  and  $S_j$  are the weighted pooled and within-group variance-covariance matrices and  $n$  is the effective number of observations, that is, the sum of the weights.

Instead of calculating the within-group variance-covariance matrices and then computing their determinants in order to calculate the test statistic, nag\_mv\_discrim (g03dac) uses a  $QR$  decomposition. The group means are subtracted from the data and then for each group, a  $QR$  decomposition is computed to give an upper triangular matrix  $R_j^*$ . This matrix can be scaled to give a matrix  $R_j$  such that  $S_j = R_j^T R_j$ . The pooled  $R$  matrix is then computed from the  $R_j$  matrices. The values of  $|S|$  and the  $|S_j|$  can then be calculated from the diagonal elements of  $R$  and the  $R_j$ .

This approach means that the Mahalanobis squared distances for a vector observation  $x$  can be computed as  $z^T z$ , where  $R_j z = (x - \bar{x}_j)$ ,  $\bar{x}_j$  being the vector of means of the  $j$ th group. These distances can be

calculated by `nag_mv_discrim_mahaldist` (g03dbc). The distances are used in discriminant analysis and `nag_mv_discrim_group` (g03dcc) uses the results of `nag_mv_discrim` (g03dac) to perform several different types of discriminant analysis. The differences between the discriminant methods are, in part, due to whether or not the within-group variance-covariance matrices are equal.

## 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

Morrison D F (1967) *Multivariate Statistical Methods* McGraw-Hill

## 5 Arguments

- 1: **n** – Integer *Input*  
*On entry:* the number of observations,  $n$ .  
*Constraint:*  $n \geq 1$ .
- 2: **m** – Integer *Input*  
*On entry:* the number of variables in the data array **x**.  
*Constraint:*  $m \geq \mathbf{nvar}$ .
- 3: **x**[ $n \times \mathbf{tdx}$ ] – const double *Input*  
*On entry:* **x**[( $k - 1$ )  $\times$   $\mathbf{tdx} + l - 1$ ] must contain the  $k$ th observation for the  $l$ th variable, for  $k = 1, 2, \dots, n$  and  $l = 1, 2, \dots, m$ .
- 4: **tdx** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **x**.  
*Constraint:*  $\mathbf{tdx} \geq m$ .
- 5: **isx**[**m**] – const Integer *Input*  
*On entry:* **isx**[ $l - 1$ ] indicates whether or not the  $l$ th variable in **x** is to be included in the variance-covariance matrices.  
 If **isx**[ $l - 1$ ]  $> 0$  the  $l$ th variable is included, for  $l = 1, 2, \dots, m$ ; otherwise it is not referenced.  
*Constraint:* **isx**[ $l - 1$ ]  $> 0$  for **nvar** values of  $l$ .
- 6: **nvar** – Integer *Input*  
*On entry:* the number of variables in the variance-covariance matrices,  $p$ .  
*Constraint:*  $\mathbf{nvar} \geq 1$ .
- 7: **ing**[**n**] – const Integer *Input*  
*On entry:* **ing**[ $k - 1$ ] indicates to which group the  $k$ th observation belongs, for  $k = 1, 2, \dots, n$ .  
*Constraint:*  $1 \leq \mathbf{ing}[k - 1] \leq \mathbf{ng}$ , for  $k = 1, 2, \dots, n$   
 The values of **ing** must be such that each group has at least **nvar** members

- 8: **ng** – Integer *Input*  
*On entry:* the number of groups,  $n_g$ .  
*Constraint:*  $\mathbf{ng} \geq 2$ .
- 9: **wt[n]** – const double *Input*  
*On entry:* the elements of **wt** must contain the weights to be used in the analysis and the effective number of observations for a group is the sum of the weights of the observations in that group. If  $\mathbf{wt}[k-1] = 0.0$  then the  $k$ th observation is excluded from the calculations.  
 If weights are not provided then **wt** must be set to **NULL** and the effective number of observations for a group is the number of observations in that group.  
*Constraints:*  
     if **wt** is not **NULL**,  $\mathbf{wt}[k-1] \geq 0.0$ , for  $k = 1, 2, \dots, n$ ;  
     the effective number of observations for each group must be greater than 1.
- 10: **nig[ng]** – Integer *Output*  
*On exit:* **nig[j-1]** contains the number of observations in the  $j$ th group, for  $j = 1, 2, \dots, n_g$ .
- 11: **gmean[ng × tdg]** – double *Output*  
**Note:** the  $(i, j)$ th element of the matrix is stored in **gmean** $[(i-1) \times \mathbf{tdg} + j-1]$ .  
*On exit:* the  $j$ th row of **gmean** contains the means of the  $p$  selected variables for the  $j$ th group, for  $j = 1, 2, \dots, n_g$ .
- 12: **tdg** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **gmean**.  
*Constraint:*  $\mathbf{tdg} \geq \mathbf{nvar}$ .
- 13: **det[ng]** – double *Output*  
*On exit:* the logarithm of the determinants of the within-group variance-covariance matrices.
- 14: **gc[dim]** – double *Output*  
**Note:** the dimension,  $dim$ , of the array **gc** must be at least  $(\mathbf{ng} + 1) \times \mathbf{nvar} \times (\mathbf{nvar} + 1)/2$ .  
*On exit:* the first  $p(p+1)/2$  elements of **gc** contain  $R$  and the remaining  $n_g$  blocks of  $p(p+1)/2$  elements contain the  $R_j$  matrices. All are stored in packed form by columns.
- 15: **stat** – double \* *Output*  
*On exit:* the likelihood-ratio test static,  $G$ .
- 16: **df** – double \* *Output*  
*On exit:* the degrees of freedom for the distribution of  $G$ .
- 17: **sig** – double \* *Output*  
*On exit:* the significance level for  $G$ .
- 18: **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,  $\mathbf{m} = \langle \text{value} \rangle$  while  $\mathbf{nvar} = \langle \text{value} \rangle$ . These arguments must satisfy  $\mathbf{m} \geq \mathbf{nvar}$ .

On entry,  $\mathbf{tdg} = \langle \text{value} \rangle$  while  $\mathbf{nvar} = \langle \text{value} \rangle$ . These arguments must satisfy  $\mathbf{tdg} \geq \mathbf{nvar}$ .

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\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_GROUP\_OBSERV

On entry, group  $\langle \text{value} \rangle$  has  $\langle \text{value} \rangle$  effective observations.

Constraint: in each group the effective number of observations must be  $\geq 1$ .

### NE\_GROUP\_VAR

On entry, group  $\langle \text{value} \rangle$  has  $\langle \text{value} \rangle$  members, while  $\mathbf{nvar} = \langle \text{value} \rangle$ .

Constraint: number of members in each group  $\geq \mathbf{nvar}$ .

### NE\_GROUP\_VAR\_RANK

The variables in group  $\langle \text{value} \rangle$  are not of full rank.

### NE\_INT\_ARG\_LT

On entry,  $\mathbf{n} = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{n} \geq 1$ .

On entry,  $\mathbf{ng} = \langle \text{value} \rangle$ .

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

On entry,  $\mathbf{nvar} = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{nvar} \geq 1$ .

### NE\_INTARR\_INT

On entry,  $\mathbf{ing}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ ,  $\mathbf{ng} = \langle \text{value} \rangle$ .

Constraint:  $1 \leq \mathbf{ing}[i-1] \leq \mathbf{ng}$ , for  $i = 1, 2, \dots, n$ .

### 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\_NEG\_WEIGHT\_ELEMENT

On entry,  $\mathbf{wt}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .

Constraint: when referenced, all elements of  $\mathbf{wt}$  must be non-negative.

### NE\_VAR\_INCL\_INDICATED

The number of variables,  $\mathbf{nvar}$  in the analysis =  $\langle \text{value} \rangle$ , while number of variables included in the analysis via array  $\mathbf{isx} = \langle \text{value} \rangle$ . Constraint: these two numbers must be the same.

### NE\_VAR\_RANK

The variables are not of full rank.

## 7 Accuracy

The accuracy is dependent on the accuracy of the computation of the  $QR$  decomposition.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

The time will be approximately proportional to  $np^2$ .

## 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 statistics computed by nag\_mv\_discrim (g03dac). The printed results show that there is evidence that the within-group variance-covariance matrices are not equal.

### 10.1 Program Text

```

/* nag_mv_discrim (g03dac) 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 GMEAN(I, J) gmean[(I) *tdgmean + J]
#define X(I, J) x[(I) *tdx + J]
int main(void)
{
  Integer  exit_status = 0, i, *ing = 0, *isx = 0, j, m, n, ng, *nig = 0, nvar,
           tdgmean;
  Integer  tdx;
  NagError fail;
  char     weight[2];
  double   *det = 0, df, *gc = 0, *gmean = 0, sig, stat, *wt = 0, *wtptr = 0;
  double   *x = 0;

  INIT_FAIL(fail);

  printf("nag_mv_discrim (g03dac) 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("%1s", weight, _countof(weight));
#else
    scanf("%1s", weight);
#endif
    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 == 'W')
    {
        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;
}

printf("\nGroup means\n\n");
for (i = 0; i < ng; ++i)
{
    for (j = 0; j < nvar; ++j)
        printf("%10.4f", GMEAN(i, j));
    printf("\n");
}
printf("\nLOG of determinants\n\n");
for (j = 0; j < ng; ++j)
    printf("%10.4f", det[j]);
printf("\n\n%s%7.4f\n", "stat = ", stat);
printf("%s%7.4f\n", " df = ", df);
printf("%s%7.4f\n", " sig = ", sig);
END:
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 (g03dac) 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
```

### **10.3 Program Results**

nag\_mv\_discrim (g03dac) Example Program Results

Group means

```
1.0433  -0.6034
2.0073  -0.2060
2.7097   1.5998
```

LOG of determinants

```
-0.8273  -3.0460  -2.2877
```

```
stat = 19.2410
df = 6.0000
sig = 0.0038
```

---