NAG Library Function Document

nag_mv_discrim_group (g03dcc)

1 Purpose

nag_mv_discrim_group (g03dcc) allocates observations to groups according to selected rules. It is intended for use after nag_mv_discrim (g03dac).

2 Specification

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

```
void nag_mv_discrim_group (Nag_DiscrimMethod type, Nag_GroupCovars equal,
     Nag_PriorProbability priors, Integer nvar, Integer ng,
     const Integer nig[], const double gmean[], Integer tdg,
     const double gc[], const double det[], Integer nobs, Integer m,
     const Integer isx[], const double x[], Integer tdx, double prior[],
     double p[], Integer tdp, Integer iag[], Nag_Boolean atiq, double ati[],
     NagError *fail)
```

3 Description

Discriminant analysis is concerned with the allocation of observations to groups using information from other observations whose group membership is known, X_t; these are called the training set. 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; these are calculated from a training set of n observations with n_j observations in the jth group, and let x_k be the kth observation from the set of observations to be allocated to the n_g groups. The observation can be allocated to a group according to a selected rule. The allocation rule or discriminant function will be based on the distance of the observation from an estimate of the location of the groups, usually the group means. A measure of the distance of the observation from the jth group mean is given by the Mahalanobis distance, D^2_{kj}:

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

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

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

Instead of using the variance-covariance matrices S and S_j, nag_mv_discrim_group (g03dcc) uses the upper triangular matrices R and R_j supplied by nag_mv_discrim (g03dac) such that $S = R^{\mathrm{T}}R$ and $S_j = R_j^{\mathrm{T}}R_j$. D^2_{kj} can then be calculated as $z^{\mathrm{T}}z$ where $R_jz = \left(x_k - \bar{x}_j\right)$ or $Rz = \left(x_k - \bar{x}_j\right)$ as appropriate.

In addition to the distances, a set of prior probabilities of group membership, π_j, for $j = 1, 2, \ldots, n_g$, may be used, with $\sum \pi_j = 1$. The prior probabilities reflect your view as to the likelihood of the observations coming from the different groups. Two common cases for prior probabilities are $\pi_1 = \pi_2 = \cdots = \pi_{n_g}$, that is, equal prior probabilities, and $\pi_j = n_j/n$, for $j = 1, 2, \ldots, n_g$, that is, prior probabilities proportional to the number of observations in the groups in the training set.

nag_mv_discrim_group (g03dcc) uses one of four allocation rules. In all four rules the p variables are assumed to follow a multivariate Normal distribution with mean μ_j and variance-covariance matrix Σ_j if the observation comes from the jth group. The different rules depend on whether or not the within-group variance-covariance matrices are assumed equal, i.e., $\Sigma_1 = \Sigma_2 = \cdots = \Sigma_{n_g}$, and whether a predictive or estimative approach is used. If $p\left(x_k \mid \mu_j, \Sigma_j\right)$ is the probability of observing the observation x_k from group j, then the posterior probability of belonging to group j is:

$$p\big(j \mid x_k\mu_j, \Sigma_j\big) \propto p\big(x_k \mid \mu_j, \Sigma_j\big)\pi_j. \tag{3}$$

In the estimative approach, the arguments μ_j and Σ_j in (3) are replaced by their estimates calculated from X_t. In the predictive approach, a non-informative prior distribution is used for the arguments and a posterior distribution for the arguments, $p\big(\mu_j, \Sigma_j \mid X_t\big)$, is found. A predictive distribution is then obtained by integrating $p\big(j \mid x_k, \mu_j, \Sigma_j\big)p\big(\mu_j, \Sigma_j \mid X\big)$ over the argument space. This predictive distribution then replaces $p\big(x_k \mid \mu_j, \Sigma_j\big)$ in (3). See Aitchison and Dunsmore (1975), Aitchison *et al.* (1977) and Moran and Murphy (1979) for further details.

The observation is allocated to the group with the highest posterior probability. Denoting the posterior probabilities, $p\big(j \mid x_k, \mu_j, \Sigma_j\big)$, by q_j, the four allocation rules are:

(i) Estimative with equal variance-covariance matrices – Linear Discrimination.

$$\log{(q)}_j \propto -\frac{1}{2}D_{kj}^2 + \log\pi_j$$

(ii) Estimative with unequal variance-covariance matrices – Quadratic Discrimination.

$$\log{(q)}_j \propto -\frac{1}{2}D_{kj}^2 + \log\pi_j - \frac{1}{2}\log\left|S_j\right|$$

(iii) Predictive with equal variance-covariance matrices.

$$q_j^{-1} \propto \big((n_j+1)/n_j\big)^{p/2}\Big\{1 + \big[n_j/\big((n-n_g)(n_j+1)\big)\big]D_{kj}^2\Big\}^{\left(n+1-n_g\right)/2}$$

(iv) Predictive with unequal variance-covariance matrices

$$q_j^{-1} \propto C\Big\{\Big(\Big(n_j^2-1\Big)/n_j\Big)\left|S_j\right|\Big\}^{p/2}\Big\{1 + \Big(n_j/\Big(n_j^2-1\Big)\Big)D_{kj}^2\Big\}^{n_j/2}$$

where

$$C = \frac{\Gamma\left(\frac{1}{2}(n_j-p)\right)}{\Gamma\left(\frac{1}{2}n_j\right)}$$

In the above the appropriate value of D_{kj}^2 from (1) or (2) is used. The values of the q_j are standardized so that,

$$\sum_{j=1}^{n_g} q_j = 1.$$

Moran and Murphy (1979) show the similarity between the predictive methods and methods based upon likelihood ratio tests.

In addition to allocating the observation to a group, nag_mv_discrim_group (g03dcc) computes an atypicality index, $I_j(x_k)$. This represents the probability of obtaining an observation more typical of group j than the observed x_k (see Aitchison and Dunsmore (1975) and Aitchison *et al.* (1977)). The atypicality index is computed as:

$$I_j(x_k) = P\bigg(B \leq z : \frac{1}{2}p, \frac{1}{2}(n_j-d)\bigg)$$

where $P(B \leq \beta : a, b)$ is the lower tail probability from a beta distribution where, for unequal within-group variance-covariance matrices,

$$z = D_{kj}^2/\Big(D_{kj}^2 + \Big(n_j^2-1\Big)/n_j\Big),$$

and for equal within-group variance-covariance matrices,

$$z = D_{kj}^2/\Big(D_{kj}^2 + (n-n_g)(n_j-1)/n_j\Big).$$

If $I_j(x_k)$ is close to 1 for all groups it indicates that the observation may come from a grouping not represented in the training set. Moran and Murphy (1979) provide a frequentist interpretation of $I_j(x_k)$.

4 References

Aitchison J and Dunsmore I R (1975) *Statistical Prediction Analysis* Cambridge

Aitchison J, Habbema J D F and Kay J W (1977) A critical comparison of two methods of statistical discrimination *Appl. Statist.* **26** 15–25

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

Moran M A and Murphy B J (1979) A closer look at two alternative methods of statistical discrimination *Appl. Statist.* **28** 223–232

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

5 Arguments

1: **type** – Nag_DiscrimMethod *Input*

On entry: indicates whether the estimative or predictive approach is to be used.

type = Nag_DiscrimEstimate
The estimative approach is used.

type = Nag_DiscrimPredict
The predictive approach is used.

Constraint: **type** = Nag_DiscrimEstimate or Nag_DiscrimPredict.

2: **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_i, for $i = 1, 2, \ldots, n_g$, stored in the remainder of **gc** are used.

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

3: **priors** – Nag_PriorProbability *Input*

On entry: indicates the form of the prior probabilities to be used.

priors = Nag_EqualPrior
Equal prior probabilities are used.

priors = Nag_GroupSizePrior
Prior probabilities proportional to the group sizes in the training set, n_j, are used.

priors = Nag_UserPrior
The prior probabilities are input in **prior**.

Constraint: **priors** = Nag_EqualPrior, Nag_GroupSizePrior or Nag_UserPrior.

4: **nvar** – Integer *Input*

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

Constraint: **nvar** ≥ 1.

5: **ng** – Integer *Input*

On entry: the number of groups, n_g.

Constraint: **ng** ≥ 2.

6: **nig**[**ng**] – const Integer *Input*

On entry: the number of observations in each group training set, n_j.

Constraints:

if **equal** = Nag_EqualCovar, **nig**$[j-1] > 0$ and $\sum_{j=1}^{n_g}$**nig**$[j-1] >$ **ng** + **nvar**, for $j = 1, 2, \ldots, n_g$;
if **equal** = Nag_NotEqualCovar, **nig**$[j-1] >$ **nvar**, for $j = 1, 2, \ldots, n_g$.

7: **gmean**[**ng** × **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 jth row of **gmean** contains the means of the p variables for the jth group, for $j = 1, 2, \ldots, n_j$. These are returned by nag_mv_discrim (g03dac).

8: **tdg** – Integer *Input*

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

Constraint: **tdg** $\geq$ **nvar**.

9: **gc**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **gc** must be at least $($**ng** $+ 1) \times$ **nvar** $\times$ $($**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 must be $\neq 0.0$;
if **equal** = Nag_NotEqualCovar, the diagonal elements of the R_j must be $\neq 0.0$, for $j = 1, 2, \ldots, n_g$.

10: **det**[**ng**] – const double *Input*

On entry: if **equal** = Nag_NotEqualCovar, the logarithms of the determinants of the within-group variance-covariance matrices as returned by nag_mv_discrim (g03dac). Otherwise **det** is not referenced.

11: **nobs** – Integer *Input*

On entry: the number of observations in **x** which are to be allocated.

Constraint: **nobs** ≥ 1.

12: **m** – Integer *Input*

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

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

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

On entry: **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 $l = 1, 2, \ldots, \mathbf{m}$; otherwise the lth variable is not referenced.

Constraint: **isx**$[l-1] > 0$ for **nvar** values of l.

14: **x**[**nobs** × **tdx**] – const double *Input*

On entry: $\mathbf{x}[(k-1) \times \mathbf{tdx} + l - 1]$ must contain the kth observation for the lth variable, for $k = 1, 2, \ldots, \mathbf{nobs}$ and $l = 1, 2, \ldots, \mathbf{m}$.

15: **tdx** – Integer *Input*

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

Constraint: $\mathbf{tdx} \geq \mathbf{m}$.

16: **prior**[**ng**] – double *Input/Output*

On entry: if **priors** = Nag_UserPrior the prior probabilities for the n_g groups.

Constraint: if **priors** = Nag_UserPrior, **prior**$[j-1] > 0.0$ and $\left|1 - \sum_{j=1}^{n_g} \mathbf{prior}[j-1]\right| \leq 10 \times$ ***machine precision***, for $j = 1, 2, \ldots, n_g$.

On exit: if **priors** = Nag_GroupSizePrior, the computed prior probabilities in proportion to group sizes for the n_g groups.

If **priors** = Nag_UserPrior, the input prior probabilities will be unchanged.

If **priors** = Nag_EqualPrior, **prior** is not set.

17: **p**[**nobs** × **tdp**] – double *Output*

On exit: $\mathbf{p}[(k-1) \times \mathbf{tdp} + j - 1]$ contains the posterior probability p_{kj} for allocating the kth observation to the jth group , for $k = 1, 2, \ldots, \mathbf{nobs}$ and $j = 1, 2, \ldots, n_g$.

18: **tdp** – Integer *Input*

On entry: the stride separating matrix column elements in the arrays **p**, **ati**.

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

19: **iag**[**nobs**] – Integer *Output*

On exit: the groups to which the observations have been allocated.

20: **atiq** – Nag_Boolean *Input*

On entry: **atiq** must be Nag_TRUE if atypicality indices are required. If **atiq** is Nag_FALSE, the array **ati** is not set.

21: **ati**[**nobs** × **tdp**] – double *Output*

On exit: if **atiq** is Nag_TRUE, $\mathbf{ati}[(k-1) \times \mathbf{tdp} + j - 1]$ will contain the atypicality index for the kth observation with respect to the jth group, for $k = 1, 2, \ldots, \mathbf{nobs}$ and $j = 1, 2, \ldots, n_g$. If **atiq** is Nag_FALSE, **ati** is not set.

22: **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, **m** $= \langle value \rangle$ while **nvar** $= \langle value \rangle$. These arguments must satisfy **m** $\geq$ **nvar**.

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

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

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

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

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

On entry, argument **type** 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_GROUP_SUM

On entry, the $\sum_{j=1}^{\mathbf{ng}} \mathbf{nig}[j-1] = \langle value \rangle$, **ng** $= \langle value \rangle$, **nvar** $= \langle value \rangle$.
Constraint: $\sum_{j=1}^{\mathbf{ng}} \mathbf{nig}[j-1] > \mathbf{ng} + \mathbf{nvar}$ when **equal** = Nag_EqualCovar.

NE_INT_ARG_LT

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

On entry, **nobs** $= \langle value \rangle$.
Constraint: **nobs** ≥ 1.

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

NE_INTARR

On entry, $\mathbf{nig}[\langle value \rangle] = \langle value \rangle$.
Constraint: $\mathbf{nig}[i-1] > 0$, for $i = 1, 2, \ldots, \mathbf{ng}$, when **equal** = Nag_EqualCovar.

NE_INTARR_INT

On entry, $\mathbf{nig}[\langle value \rangle] = \langle value \rangle$, **nvar** $= \langle value \rangle$.
Constraint: $\mathbf{nig}[i-1] > \mathbf{nvar}$, $i = 1, 2, \ldots, \mathbf{ng}$ when **equal** = Nag_NotEqualCovar.

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_PRIOR_SUM

On entry, $\sum_{j=1}^{\mathbf{ng}} \mathbf{prior}[j-1] = \langle value \rangle$.
Constraint: $\sum_{j=1}^{\mathbf{ng}} \mathbf{prior}[j-1]$ must be within $10\times$ ***machine precision*** of 1 when **priors** $=$ Nag_UserPrior.

NE_REALARR

On entry, $\mathbf{prior}[\langle value \rangle] = \langle value \rangle$.
Constraint: $\mathbf{prior}[j-1] > 0$, $j = 1, 2, \ldots, \mathbf{ng}$ when **priors** $=$ Nag_UserPrior.

NE_VAR_INCL_INDICATED

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.

7 Accuracy

The accuracy of the returned posterior probabilities will depend on the accuracy of the input R or R_j matrices. The atypicality index should be accurate to four significant places.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The distances D_{kj}^2 can be computed using nag_mv_discrim_mahaldist (g03dbc) if other forms of discrimination are required.

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 allocations made using the predictive approach and under the assumption that the within-group covariance matrices are not equal. The posterior probabilities of group membership, q_j, and the atypicality index are printed along with the allocated group. The atypicality index shows that observations 5 and 6 do not seem to be typical of the three types present in the initial 21 observations.

10.1 Program Text

```
/* nag_mv_discrim_group (g03dcc) 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 ATI(I, J)   ati[(I) *tdati + J]
#define GMEAN(I, J) gmean[(I) *tdgmean + J]
#define P(I, J)     p[(I) *tdp + J]
#define X(I, J)     x[(I) *tdx + J]
```

```
int main(void)
{
  Integer           exit_status = 0, i, *iag = 0, *ing = 0, *isx = 0, j, m, n,
                    ng, *nig = 0, nobs;
  Integer           nvar, tdati, tdgmean, tdp, tdx;
  double            *ati = 0, *det = 0, df, *gc = 0, *gmean = 0, *p = 0;
  double            *prior = 0, sig, stat, *wt = 0, *wtptr = 0, *x = 0;
  char              nag_enum_arg[40];
  Nag_Boolean       atiq = Nag_TRUE, weight;
  Nag_DiscrimMethod type;
  Nag_GroupCovars   equal;
  NagError          fail;

  INIT_FAIL(fail);

  printf("nag_mv_discrim_group (g03dcc) 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)) ||
          !(prior = NAG_ALLOC(ng, 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;
        }
```

```
      tdati = ng;
      tdgmean = nvar;
      tdp = ng;
      tdx = m;
    }
  else
    {
      printf("Invalid n or nvar 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);
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  type = (Nag_DiscrimMethod) nag_enum_name_to_value(nag_enum_arg);
  if (nobs >= 1)
    {
      if (!(ati = NAG_ALLOC((nobs)*(ng), double)) ||
          !(p = NAG_ALLOC((nobs)*(ng), double)) ||
          !(iag = NAG_ALLOC(nobs, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tdati = ng;
      tdp = 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_group (g03dcc).
       * Allocates observations to groups, following
       * nag_mv_discrim (g03dac)
       */
      nag_mv_discrim_group(type, equal, Nag_EqualPrior, nvar, ng, nig, gmean,
                           tdgmean, gc, det, nobs, m, isx, x, tdx, prior, p,
                           tdp, iag, atiq, ati, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_mv_discrim_group (g03dcc).\n%s\n",
                 fail.message);
          exit_status = 1;
          goto END;
        }

      printf("\n");
      printf("    Obs       Posterior        Allocated ");
      printf("      Atypicality ");
      printf("\n");
      printf("             probabilities    to group      index ");
      printf("\n");
      printf("\n");
      for (i = 0; i < nobs; ++i)
        {
          printf(" %6"NAG_IFMT"     ", i+1);
          for (j = 0; j < ng; ++j)
            {
              printf("%6.3f", P(i, j));
            }
          printf("  %6"NAG_IFMT"    ", iag[i]);
          for (j = 0; j < ng; ++j)
            {
```

```
              printf("%6.3f", ATI(i, j));
            }
          printf("\n");
        }
    }

 END:
  NAG_FREE(ati);
  NAG_FREE(det);
  NAG_FREE(gc);
  NAG_FREE(gmean);
  NAG_FREE(p);
  NAG_FREE(prior);
  NAG_FREE(wt);
  NAG_FREE(x);
  NAG_FREE(iag);
  NAG_FREE(ing);
  NAG_FREE(isx);
  NAG_FREE(nig);
  return exit_status;
}
```

10.2 Program Data

```
nag_mv_discrim_group (g03dcc) 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 Nag_DiscrimPredict
  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_group (g03dcc) Example Program Results


    Obs       Posterior        Allocated       Atypicality
              probabilities    to group      index

      1      0.094 0.905 0.002       2     0.596 0.254 0.975
```

```
     2      0.005 0.168 0.827      3      0.952 0.836 0.018
     3      0.019 0.920 0.062      2      0.954 0.797 0.912
     4      0.697 0.303 0.000      1      0.207 0.860 0.993
     5      0.317 0.013 0.670      3      0.991 1.000 0.984
     6      0.032 0.366 0.601      3      0.981 0.978 0.887
```