NAG Library Function Document

nag_mv_canon_var (g03acc)

1 Purpose

nag_mv_canon_var (g03acc) performs a canonical variate (canonical discrimination) analysis.

2 Specification

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

```
void nag_mv_canon_var (Nag_Weightstype weight, Integer n, Integer m,
     const double x[], Integer tdx, const Integer isx[], Integer nx,
     const Integer ing[], Integer ng, const double wt[], Integer nig[],
     double cvm[], Integer tdcvm, double e[], Integer tde, Integer *ncv,
     double cvx[], Integer tdcvx, double tol, Integer *irankx,
     NagError *fail)
```

3 Description

Let a sample of n observations on n_x variables in a data matrix come from n_g groups with $n_1, n_2, \ldots, n_{n_g}$ observations in each group, $\sum n_i = n$. Canonical variate analysis finds the linear combination of the n_x variables that maximizes the ratio of between-group to within-group variation. The variables formed, the canonical variates can then be used to discriminate between groups.

The canonical variates can be calculated from the eigenvectors of the within-group sums of squares and cross-products matrix. However, nag_mv_canon_var (g03acc) calculates the canonical variates by means of a singular value decomposition (SVD) of a matrix V. Let the data matrix with variable (column) means subtracted be X, and let its rank be k; then the k by $\left(n_g - 1\right)$ matrix V is given by:

$V = Q_X^{\mathrm{T}} Q_g$, where Q_g is an n by $\left(n_g - 1\right)$ orthogonal matrix that defines the groups and Q_X is the first k rows of the orthogonal matrix Q either from the QR decomposition of X:

$$X = QR$$

if X is of full column rank, i.e., $k = n_x$, else from the SVD of X:

$$X = QDP^{\mathrm{T}}.$$

Let the SVD of V be:

$$V = U_x \Delta U_g^{\mathrm{T}}$$

then the nonzero elements of the diagonal matrix Δ, δ_i, for $i = 1, 2, \ldots, l$, are the l canonical correlations associated with the l canonical variates, where $l = \min\left(k, n_g\right)$.

The eigenvalues, λ_i^2, of the within-group sums of squares matrix are given by:

$$\lambda_i^2 = \frac{\delta_i^2}{1 - \delta_i^2}.$$

and the value of $\pi_i = \lambda_i^2 / \sum \lambda_i^2$ gives the proportion of variation explained by the ith canonical variate. The values of the π_i's give an indication as to how many canonical variates are needed to adequately describe the data, i.e., the dimensionality of the problem.

To test for a significant dimensionality greater than i the χ^2 statistic:

$$\left(n-1-n_g-\frac{1}{2}(k-n_g)\right)\sum_{j=i+1}^{l}\log\left(1+\lambda_j^2\right)$$

can be used. This is asymptotically distributed as a χ^2 distribution with $(k-i)\left(n_g-1-i\right)$ degrees of freedom. If the test for $i=h$ is not significant, then the remaining tests for $i>h$ should be ignored.

The loadings for the canonical variates are calculated from the matrix U_x. This matrix is scaled so that the canonical variates have unit within group variance.

In addition to the canonical variates loadings the means for each canonical variate are calculated for each group.

Weights can be used with the analysis, in which case the weighted means are subtracted from each column and then each row is scaled by an amount $\sqrt{w_i}$, where w_i is the weight for the ith observation (row).

4 References

Chatfield C and Collins A J (1980) *Introduction to Multivariate Analysis* Chapman and Hall

Gnanadesikan R (1977) *Methods for Statistical Data Analysis of Multivariate Observations* Wiley

Hammarling S (1985) The singular value decomposition in multivariate statistics *SIGNUM Newsl.* **20(3)** 2–25

Kendall M G and Stuart A (1979) *The Advanced Theory of Statistics (3 Volumes)* (4th Edition) Griffin

5 Arguments

1: **weight** – Nag_Weightstype *Input*

On entry: indicates the type of weights to be used in the analysis.

weight = Nag_NoWeights
No weights are used.

weight = Nag_Weightsfreq
The weights are treated as frequencies and the effective number of observations is the sum of the weights.

weight = Nag_Weightsvar
The weights are treated as being inversely proportional to the variance of the observations and the effective number of observations is the number of observations with nonzero weights.

Constraint: **weight** = Nag_NoWeights, Nag_Weightsfreq or Nag_Weightsvar.

2: **n** – Integer *Input*

On entry: the number of observations, n.

Constraint: $\mathbf{n} \geq \mathbf{nx} + \mathbf{ng}$.

3: **m** – Integer *Input*

On entry: the total number of variables, m.

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

4: $\mathbf{x}[\mathbf{n} \times \mathbf{tdx}]$ – const double *Input*

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

5: **tdx** – Integer *Input*

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

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

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

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

If **isx**$[j-1] > 0$, then the variable contained in the jth column of **x** is included in the canonical variate analysis, for $j = 1, 2, \ldots, m$.

Constraint: **isx**$[j-1] > 0$ for **nx** values of j.

7: **nx** – Integer *Input*

On entry: the number of variables in the analysis, n_x.

Constraint: **nx** ≥ 1.

8: **ing**[**n**] – const Integer *Input*

On entry: **ing**$[i-1]$ indicates which group the ith observation is in, for $i = 1, 2, \ldots, n$. The effective number of groups is the number of groups with nonzero membership.

Constraint: $1 \leq$ **ing**$[i-1] \leq$ **ng**, for $i = 1, 2, \ldots, n$.

9: **ng** – Integer *Input*

On entry: the number of groups, n_g.

Constraint: **ng** ≥ 2.

10: **wt**[**n**] – const double *Input*

On entry: if **weight** = Nag_Weightsfreq or Nag_Weightsvar then the elements of **wt** must contain the weights to be used in the analysis.

If **wt**$[i-1] = 0.0$ then the ith observation is not included in the analysis.

Constraints:

wt$[i-1] \geq 0.0$, for $i = 1, 2, \ldots, n$;
$\sum_{i=1}^{n}$**wt**$[i-1] \geq$ **nx**+ effective number of groups.

Note: if **weight** = Nag_NoWeights then **wt** is not referenced and may be **NULL**..

11: **nig**[**ng**] – Integer *Output*

On exit: **nig**$[j-1]$ gives the number of observations in group j, for $j = 1, 2, \ldots, n_g$.

12: **cvm**[**ng** $\times$ **tdcvm**] – double *Output*

On exit: **cvm**$[(i-1) \times$ **tdcvm** $+ j - 1]$ contains the mean of the jth canonical variate for the ith group, for $i = 1, 2, \ldots, n_g$ and $j = 1, 2, \ldots, l$; the remaining columns, if any, are used as workspace.

13: **tdcvm** – Integer *Input*

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

Constraint: **tdcvm** $\geq$ **nx**.

14: **e**[**min**(**nx**, **ng** – **1**) $\times$ **tde**] – double *Output*

On exit: the statistics of the canonical variate analysis. **e**$[(i-1) \times$ **tde**$]$, the canonical correlations, δ_i, for $i = 1, 2, \ldots, l$.

$\mathbf{e}[(i-1) \times \mathbf{tde}+1]$, the eigenvalues of the within-group sum of squares matrix, λ_i^2, for $i = 1, 2, \ldots, l$.

$\mathbf{e}[(i-1) \times \mathbf{tde}+2]$, the proportion of variation explained by the ith canonical variate, for $i = 1, 2, \ldots, l$.

$\mathbf{e}[(i-1) \times \mathbf{tde}+3]$, the χ^2 statistic for the ith canonical variate, for $i = 1, 2, \ldots, l$.

$\mathbf{e}[(i-1) \times \mathbf{tde}+4]$, the degrees of freedom for χ^2 statistic for the ith canonical variate, for $i = 1, 2, \ldots, l$.

$\mathbf{e}[(i-1) \times \mathbf{tde}+5]$, the significance level for the χ^2 statistic for the ith canonical variate, for $i = 1, 2, \ldots, l$.

15: **tde** – Integer *Input*

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

Constraint: **tde** ≥ 6.

16: **ncv** – Integer * *Output*

On exit: the number of canonical variates, l. This will be the minimum of $n_g - 1$ and the rank of **x**.

17: **cvx**[**nx** $\times$ **tdcvx**] – double *Output*

On exit: the canonical variate loadings. $\mathbf{cvx}[(i-1) \times \mathbf{tdcvx} + j - 1]$ contains the loading coefficient for the ith variable on the jth canonical variate, for $i = 1, 2, \ldots, n_x$ and $j = 1, 2, \ldots, l$; the remaining columns, if any, are used as workspace.

18: **tdcvx** – Integer *Input*

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

Constraint: **tdcvx** $\geq$ **ng** $- 1$.

19: **tol** – double *Input*

On entry: the value of **tol** is used to decide if the variables are of full rank and, if not, what is the rank of the variables. The smaller the value of **tol** the stricter the criterion for selecting the singular value decomposition. If a non-negative value of **tol** less than ***machine precision*** is entered, then the square root of ***machine precision*** is used instead.

Constraint: **tol** ≥ 0.0.

20: **irankx** – Integer * *Output*

On exit: the rank of the dependent variables.

If the variables are of full rank then **irankx** $=$ **nx**.

If the variables are not of full rank then **irankx** is an estimate of the rank of the dependent variables. **irankx** is calculated as the number of singular values greater than **tol**$\times$(largest singular value).

21: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

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

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

On entry, **tdcvx** $= \langle value \rangle$ while **ng** $= \langle value \rangle$. These arguments must satisfy **tdcvx** $\geq$ **ng** $- 1$.

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

NE_3_INT_ARG_CONS

On entry, **n** $= \langle value \rangle$, **nx** $= \langle value \rangle$ and **ng** $= \langle value \rangle$. These arguments must satisfy **n** $\geq$ **nx** $+$ **ng**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_CANON_CORR_1

A canonical correlation is equal to one. This will happen if the variables provide an exact indication as to which group every observation is allocated.

NE_GROUPS

Either the effective number of groups is less than two or the effective number of groups plus the number of variables, **nx** is greater than the effective number of observations.

NE_INT_ARG_LT

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

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

On entry, **tde** $= \langle value \rangle$.
Constraint: **tde** ≥ 6.

NE_INTARR_INT

On entry, **ing**$[\langle value \rangle] = \langle value \rangle$, **ng** $= \langle value \rangle$. Constraint: $1 \leq$ **ing**$[i-1] \leq$ **ng**, for $i = 1, 2, \ldots, n$.

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, **wt**$[\langle value \rangle] = \langle value \rangle$.
Constraint: When referenced, all elements of **wt** must be non-negative.

NE_RANK_ZERO

The rank of the variables is zero. This will happen if all the variables are constants.

NE_REAL_ARG_LT

On entry, **tol** must not be less than 0.0: **tol** $= \langle value \rangle$.

NE_SVD_NOT_CONV

The singular value decomposition has failed to converge. This is an unlikely error exit.

NE_VAR_INCL_INDICATED

The number of variables, **nx** 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.

NE_WT_ARGS

The **wt** array argument must not be **NULL** when the **weight** argument indicates weights.

7 Accuracy

As the computation involves the use of orthogonal matrices and a singular value decomposition rather than the traditional computing of a sum of squares matrix and the use of an eigenvalue decomposition, nag_mv_canon_var (g03acc) should be less affected by ill conditioned problems.

8 Parallelism and Performance

nag_mv_canon_var (g03acc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

A sample of nine observations, each consisting of three variables plus group indicator, is read in. There are three groups. An unweighted canonical variate analysis is performed and the results printed.

10.1 Program Text

```
/* nag_mv_canon_var (g03acc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg03.h>

#define X(I, J)   x[(I) *tdx + J]
#define E(I, J)   e[(I) *tde + J]
#define CVM(I, J) cvm[(I) *tdcvm + J]
#define CVX(I, J) cvx[(I) *tdcvx + J]
int main(void)
{

  Integer exit_status = 0, i, irx, j, m, n, ncv, ng;
  Integer nx, tdcvm, tdcvx, tde, tdx;
  Integer *ing = 0, *isx = 0, *nig = 0;
```

```
  double *cvm = 0, *cvx = 0, *e = 0, tol, *wt = 0, *x = 0;
  char nag_enum_arg[40];
  Nag_Weightstype weight;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_mv_canon_var (g03acc) 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
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &nx);
#else
  scanf("%" NAG_IFMT "", &nx);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &ng);
#else
  scanf("%" NAG_IFMT "", &ng);
#endif
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_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_Weightstype) nag_enum_name_to_value(nag_enum_arg);
  if (n >= nx + ng && m >= nx) {
    if (!(x = NAG_ALLOC(n * m, double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(ing = NAG_ALLOC(n, Integer)) ||
        !(e = NAG_ALLOC((MIN(nx, ng - 1)) * 6, double)) ||
        !(cvm = NAG_ALLOC(ng * nx, double)) ||
        !(cvx = NAG_ALLOC(nx * (ng - 1), double)) ||
        !(nig = NAG_ALLOC(ng, Integer)) || !(isx = NAG_ALLOC(m, Integer))

          )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tdx = m;
    tde = 6;
    tdcvm = nx;
    tdcvx = ng - 1;
  }
  else {
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
  }
  if (weight == Nag_Weightsfreq || weight == Nag_Weightsvar) {
```

```
    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("%lf", &wt[i]);
#else
      scanf("%lf", &wt[i]);
#endif
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &ing[i]);
#else
      scanf("%" NAG_IFMT "", &ing[i]);
#endif
    }
  }
  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

  tol = 1e-6;
  /* nag_mv_canon_var (g03acc).
   * Canonical variate analysis
   */
  nag_mv_canon_var(weight, n, m, x, tdx, isx, nx, ing, ng, wt, nig,
                   cvm, tdcvm, e, tde, &ncv, cvx, tdcvx, tol, &irx, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_mv_canon_var (g03acc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("%s%2" NAG_IFMT "\n\n", "Rank of x = ", irx);
  printf("Canonical    Eigenvalues     Percentage   CHISQ"
         "        DF          SIG \n");
  printf("Correlations                 Variation\n");
  for (i = 0; i < ncv; ++i) {
    for (j = 0; j < 6; ++j)
      printf("%12.4f", E(i, j));
    printf("\n");
  }
  printf("\nCanonical Coefficients for X\n");
  for (i = 0; i < nx; ++i) {
    for (j = 0; j < ncv; ++j)
      printf("%9.4f", CVX(i, j));
    printf("\n");
  }
  printf("\nCanonical variate means\n");
  for (i = 0; i < ng; ++i) {
    for (j = 0; j < ncv; ++j)
```

```
      printf("%9.4f", CVM(i, j));
    printf("\n");
  }

END:
  NAG_FREE(x);
  NAG_FREE(wt);
  NAG_FREE(ing);
  NAG_FREE(e);
  NAG_FREE(cvm);
  NAG_FREE(cvx);
  NAG_FREE(nig);
  NAG_FREE(isx);

  return exit_status;
}
```

10.2 Program Data

```
nag_mv_canon_var (g03acc) Example Program Data
 9 3 3 3 Nag_NoWeights
 13.3 10.6 21.2 1
 13.6 10.2 21.0 2
 14.2 10.7 21.1 3
 13.4  9.4 21.0 1
 13.2  9.6 20.1 2
 13.9 10.4 19.8 3
 12.9 10.0 20.5 1
 12.2  9.9 20.7 2
 13.9 11.0 19.1 3
  1    1    1
```

10.3 Program Results

```
nag_mv_canon_var (g03acc) Example Program Results

Rank of x =  3

Canonical    Eigenvalues     Percentage   CHISQ        DF           SIG
Correlations                 Variation
      0.8826      3.5238      0.9795      7.9032      6.0000      0.2453
      0.2623      0.0739      0.0205      0.3564      2.0000      0.8368

Canonical Coefficients for X
  -1.7070   0.7277
  -1.3481   0.3138
   0.9327   1.2199

Canonical variate means
   0.9841   0.2797
   1.1805  -0.2632
  -2.1646  -0.0164
```