NAG Library Function Document

nag_mv_canon_corr (g03adc)

1 Purpose

nag_mv_canon_corr (g03adc) performs canonical correlation analysis upon input data matrices.

2 Specification

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

void nag_mv_canon_corr (Integer n, Integer m, const double z[], Integer tdz,
                        const Integer isz[], Integer nx, Integer ny, const double wt[],
                        double e[], Integer tde, Integer *ncv, double cvx[], Integer tdcvx,
                        double cvy[], Integer tdcvy, double tol, NagError *fail)
```

3 Description

Let there be two sets of variables, \( x \) and \( y \). For a sample of \( n \) observations on \( n_x \) variables in a data matrix \( X \) and \( n_y \) variables in a data matrix \( Y \), canonical correlation analysis seeks to find a small number of linear combinations of each set of variables in order to explain or summarise the relationships between them. The variables thus formed are known as canonical variates.

Let the variance-covariance matrix of the two datasets be

\[
\begin{pmatrix}
S_{xx} & S_{xy} \\
S_{yx} & S_{yy}
\end{pmatrix}
\]

and let

\[
\Sigma = S_{yy}^{-1}S_{yx}S_{xx}^{-1}S_{xy}
\]

then the canonical correlations can be calculated from the eigenvalues of the matrix \( \Sigma \). However, nag_mv_canon_corr (g03adc) calculates the canonical correlations by means of a singular value decomposition (SVD) of a matrix \( V \). If the rank of the data matrix \( X \) is \( k_x \) and the rank of the data matrix \( Y \) is \( k_y \), and both \( X \) and \( Y \) have had variable (column) means subtracted, then the \( k_x \) by \( k_y \) matrix \( V \) is given by:

\[
V = Q_x^T Q_y,
\]

where \( Q_x \) is the first \( k_x \) rows of the orthogonal matrix \( Q \) either from the \( QR \) decomposition of \( X \) if \( X \) is of full column rank, i.e., \( k_x = n_x \):

\[
X = Q_x R_x
\]

or from the SVD of \( X \) if \( k_x < n_x \):

\[
X = Q_x D_x P_x^T.
\]

Similarly \( Q_y \) is the first \( k_y \) rows of the orthogonal matrix \( Q \) either from the \( QR \) decomposition of \( Y \) if \( Y \) is of full column rank, i.e., \( k_y = n_y \):

\[
Y = Q_y R_y
\]

or from the SVD of \( Y \) if \( k_y < n_y \):

\[
Y = Q_y D_y P_y^T.
\]

Let the SVD of \( V \) be:
\[ V = U_z \Delta U_y^T \]
then the nonzero elements of the diagonal matrix \( \Delta, \delta_i \), for \( i = 1, 2, \ldots, l \), are the \( l \) canonical correlations associated with the \( l \) canonical variates, where \( l = \min(k_x, k_y) \).

The eigenvalues, \( \lambda_i^2 \), of the matrix \( \Sigma \) are given by:

\[ \lambda_i^2 = \frac{\delta_i^2}{1 + \delta_i^2}. \]

The value of \( \pi_i = \frac{\lambda_i^2}{\sum \lambda_i^2} \) gives the proportion of variation explained by the \( i \)th canonical variate. The values of the \( \pi_i \) 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 \( \chi^2 \) statistic:

\[ \left( n - \frac{1}{2}(k_x + k_y + 3) \right) \sum_{j=i+1}^l \log \left( 1 + \lambda_j^2 \right) \]

can be used. This is asymptotically distributed as a \( \chi^2 \) distribution with \( (k_x - i)(k_y - i) \) degrees of freedom. If the test for \( i = k_{\text{min}} \) is not significant, then the remaining tests for \( i > k_{\text{min}} \) should be ignored.

The loadings for the canonical variates are calculated from the matrices \( U_z \) and \( U_y \) respectively. These matrices are scaled so that the canonical variates have unit variance.

4 References

5 Arguments
1: \( n \) – Integer  
   \( \text{Input} \)  
   \( \text{On entry:} \) the number of observations, \( n \).  
   \( \text{Constraint:} \ n > nx + ny. \)

2: \( m \) – Integer  
   \( \text{Input} \)  
   \( \text{On entry:} \) the total number of variables, \( m \).  
   \( \text{Constraint:} \ m \geq nx + ny. \)

3: \( z[n \times \text{tdz}] \) – const double  
   \( \text{Input} \)  
   \( \text{On entry:} \) \( z[(i - 1) \times \text{tdz} + j - 1] \) must contain the \( i \)th observation for the \( j \)th variable, for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \).  
   Both \( x \) and \( y \) variables are to be included in \( z \), the indicator array, \( \text{isz} \), being used to assign the variables in \( z \) to the \( x \) or \( y \) sets as appropriate.

4: \( \text{tdz} \) – Integer  
   \( \text{Input} \)  
   \( \text{On entry:} \) the stride separating matrix column elements in the array \( z \).  
   \( \text{Constraint:} \ \text{tdz} \geq m. \)
5: \textbf{isz[m]} – const Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \textbf{isz} \{j - 1\} indicates whether or not the \textit{j}th variable is to be included in the analysis and to which set of variables it belongs.

If \textbf{isz} \{j - 1\} > 0, then the variable contained in the \textit{j}th column of \textbf{z} is included as an \textit{x} variable in the analysis.

If \textbf{isz} \{j - 1\} < 0, then the variable contained in the \textit{j}th column of \textbf{z} is included as a \textit{y} variable in the analysis.

If \textbf{isz} \{j - 1\} = 0, then the variable contained in the \textit{j}th column of \textbf{z} is not included in the analysis.

\textit{Constraint:} only \textit{nx} elements of \textbf{isz} can be > 0 and only \textit{ny} elements of \textbf{isz} can be < 0.

6: \textbf{nx} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of \textit{x} variables in the analysis, \textit{n_x}.

\textit{Constraint:} \textbf{nx} ≥ 1.

7: \textbf{ny} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of \textit{y} variables in the analysis, \textit{n_y}.

\textit{Constraint:} \textbf{ny} ≥ 1.

8: \textbf{wt[n]} – const double \hspace{1cm} \textit{Input}

\textit{On entry:} the elements of \textbf{wt} must contain the weights to be used in the analysis. The effective number of observations is the sum of the weights. If \textbf{wt} \{i - 1\} = 0.0 then the \textit{i}th observation is not included in the analysis.

If weights are not provided then \textbf{wt} must be set to NULL and the effective number of observations is \textit{n}.

\textit{Constraints:}

if \textbf{wt} is not NULL, \textbf{wt} \{i - 1\} ≥ 0.0, for \textit{i} = 1, 2, \ldots, \textit{n};

\[\sum_{i=1}^{n} \textbf{wt} \{i - 1\} \geq \textbf{nx} + \textbf{ny} + 1.\]

9: \textbf{e[\min(nx, ny) × tde]} – double \hspace{1cm} \textit{Output}

\textit{On exit:} the statistics of the canonical variate analysis. \textbf{e} \{(i - 1) \times \text{tde}\}, the canonical correlations, \(\delta_i\), for \textit{i} = 1, 2, \ldots, \textit{l}.

\textbf{e} \{(i - 1) \times \text{tde} + 1\}, the eigenvalues of \(\Sigma\), \(\lambda_i^2\), for \textit{i} = 1, 2, \ldots, \textit{l}.

\textbf{e} \{(i - 1) \times \text{tde} + 2\}, the proportion of variation explained by the \textit{i}th canonical variate, for \textit{i} = 1, 2, \ldots, \textit{l}.

\textbf{e} \{(i - 1) \times \text{tde} + 3\}, the \(\chi^2\) statistic for the \textit{i}th canonical variate, for \textit{i} = 1, 2, \ldots, \textit{l}.

\textbf{e} \{(i - 1) \times \text{tde} + 4\}, the degrees of freedom for \(\chi^2\) statistic for the \textit{i}th canonical variate, for \textit{i} = 1, 2, \ldots, \textit{l}.

\textbf{e} \{(i - 1) \times \text{tde} + 5\}, the significance level for the \(\chi^2\) statistic for the \textit{i}th canonical variate, for \textit{i} = 1, 2, \ldots, \textit{l}.

10: \textbf{tde} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating matrix column elements in the array \textbf{e}.

\textit{Constraint:} \textbf{tde} ≥ 6.

11: \textbf{ncv} – Integer \ast \hspace{1cm} \textit{Output}

\textit{On exit:} the number of canonical correlations, \textit{l}. This will be the minimum of the rank of \textit{X} and the rank of \textit{Y}.
12: \texttt{cvx[nx \times tdcvx]} – double

\textit{Output}

\textit{On exit}: the canonical variate loadings for the \textit{x} variables. \texttt{cvx[(i - 1) \times tdcvx + j - 1]} contains the loading coefficient for the \textit{i}th \textit{x} variable on the \textit{j}th canonical variate.

13: \texttt{tdcvx} – Integer

\textit{Input}

\textit{On entry}: the stride separating matrix column elements in the array \texttt{cvx}.

\textit{Constraint}: \texttt{tdcvx} \geq \texttt{min(nx,ny)}.

14: \texttt{cvy[ny \times tdcvy]} – double

\textit{Output}

\textit{On exit}: the canonical variate loadings for the \textit{y} variables. \texttt{cvy[(i - 1) \times tdcvy + j - 1]} contains the loading coefficient for the \textit{i}th \textit{y} variable on the \textit{j}th canonical variate.

15: \texttt{tdcvy} – Integer

\textit{Input}

\textit{On entry}: the stride separating matrix column elements in the array \texttt{cvy}.

\textit{Constraint}: \texttt{tdcvy} \geq \texttt{min(nx,ny)}.

16: \texttt{tol} – double

\textit{Input}

\textit{On entry}: the value of \texttt{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 \texttt{tol} the stricter the criterion for selecting the singular value decomposition. If a non-negative value of \texttt{tol} less than \textit{machine precision} is entered, then the square root of \textit{machine precision} is used instead.

\textit{Constraint}: \texttt{tol} \geq 0.0.

17: \texttt{fail} – NagError *

\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \ Error Indicators and Warnings

\textbf{NE_2_INT_ARG_LT}

\textit{On entry}, \texttt{tdz} = \texttt{value} while \texttt{m} = \texttt{value}. These arguments must satisfy \texttt{tdz} \geq \texttt{m}.

\textbf{NE_3_INT_ARG_CONS}

\textit{On entry}, \texttt{m} = \texttt{value}, \texttt{nx} = \texttt{value} and \texttt{ny} = \texttt{value}. These arguments must satisfy \texttt{m} \geq \texttt{nx} + \texttt{ny}.

\textit{On entry}, \texttt{n} = \texttt{value}, \texttt{nx} = \texttt{value} and \texttt{ny} = \texttt{value}. These arguments must satisfy \texttt{n} > \texttt{nx} + \texttt{ny}.

\textit{On entry}, \texttt{tdcvx} = \texttt{value}, \texttt{nx} = \texttt{value} and \texttt{ny} = \texttt{value}. These arguments must satisfy \texttt{tdcvx} \geq \texttt{min(nx,ny)}.

\textit{On entry}, \texttt{tdcvy} = \texttt{value}, \texttt{nx} = \texttt{value} and \texttt{ny} = \texttt{value}. These arguments must satisfy \texttt{tdcvy} \geq \texttt{min(nx,ny)}.

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.

\textbf{NE_CANON_CORR_1}

A canonical correlation is equal to one. This will happen if the \textit{x} and \textit{y} variables are perfectly correlated.
NE_INT_ARG_LT
On entry, \( nx = \langle\text{value}\rangle \).
Constraint: \( nx \geq 1 \).
On entry, \( ny = \langle\text{value}\rangle \).
Constraint: \( ny \geq 1 \).
On entry, \( tde = \langle\text{value}\rangle \).
Constraint: \( tde \geq 6 \).

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_MAT_RANK_ZERO
The rank of the \( X \) matrix or the rank of the \( Y \) matrix is zero. This will happen if all the \( x \) and \( y \) variables are constants.

NE_NEG_WEIGHT_ELEMENT
On entry, \( wt = \langle\text{value}\rangle \). 
Constraint: when referenced, all elements of \( wt \) must be non-negative.

NE_OBSERV_LT_VAR
With weighted data, the effective number of observations given by the sum of weights \( = \langle\text{value}\rangle \), while number of variables included in the analysis, \( nx + ny = \langle\text{value}\rangle \).
Constraint: Effective number of observations \( \geq nx + ny + 1 \).

NE_REAL_ARG_LT
On entry, \( tol \) must not be less than 0.0: \( tol = \langle\text{value}\rangle \).

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 \( = \langle\text{value}\rangle \), while the number of \( x \) variables included in the analysis via array \( isz = \langle\text{value}\rangle \).
Constraint: these two numbers must be the same.
The number of variables, \( ny \) in the analysis \( = \langle\text{value}\rangle \), while the number of \( y \) variables included in the analysis via array \( isz = \langle\text{value}\rangle \).
Constraint: these two numbers must be the same.

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, \( \text{nag_mv_canon_corr (g03adc)} \) should be less affected by ill conditioned problems.

8 Parallelism and Performance
Not applicable.

9 Further Comments
None.
10 Example

A sample of nine observations with two variables in each set is read in. The second and third variables are $x$ variables while the first and last are $y$ variables. Canonical variate analysis is performed and the results printed.

10.1 Program Text

```c
/* nag_mv_canon_corr (g03adc) 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 <nag03.h>
#define Z(I, J) z[(I)*tdz + J]
#define CVX(I, J) cvx[(I)*tdcvx + J]
#define CVY(I, J) cvy[(I)*tdcvy + J]
#define E(I, J) e[(I)*tde + J]

int main(void)
{
    Integer exit_status = 0, i, *isz = 0, j, m, n, ncv, nx, ny, tdcvx, tdcvy,
    tde = 6;
    Integer tdz;
    NagError fail;
    char weight[2];
    double *cvx = 0, *cvy = 0, *e = 0, tol, *wt = 0, *wtptr, *z = 0;

    INIT_FAIL(fail);

    printf("nag_mv_canon_corr (g03adc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[^
\]");
    #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"", &ny);
    #else
        scanf("%"NAG_IFMT"", &ny);
    #endif
    #ifdef _WIN32
        scanf_s("%ls", weight, _countof(weight));
    #else
```
scanf("%s", weight);
#endif

if (nx >= 1 && ny >= 1 && n > nx*ny && m >= nx*ny)
{
    if (!(z = NAG_ALLOC(n*m, double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(isz = NAG_ALLOC(m, Integer)) ||
        !(cvx = NAG_ALLOC(nx*(MIN(nx, ny)), double)) ||
        !(cvy = NAG_ALLOC(ny*(MIN(nx, ny)), double)) ||
        !(e = NAG_ALLOC((MIN(nx, ny))*6, double))
    )
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tdz = m;
    tdcvx = MIN(nx, ny);
    tdcvy = MIN(nx, ny);
    tde = 6;
}
else
{
    printf("Invalid nx or ny or n or m.\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", &Z(i, j));
        #else
            scanf("%lf", &Z(i, j));
        #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", &Z(i, j));
        #else
            scanf("%lf", &Z(i, j));
        #endif
    }
    wtptr = 0;
}
for (j = 0; j < m; ++j)
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"", &isz[j]);
#else
    scanf("%"NAG_IFMT"", &isz[j]);
#endif
tol = 1e-6;
/* nag_mv_canon_corr (g03adc).
  * Canonical correlation analysis
  */
nag_mv_canon_corr(n, m, z, tdz, isz, nx, ny, wtptr, e, tde, &ncv, cvx, tdcvx, cvy, tdcvy, tol, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_mv_canon_corr (g03adc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("%s\n" "\n\n\n\n%s%2"NAG_IFMT"%s%2"NAG_IFMT"\n\n\n", "Rank of x = ", nx,
" Rank of y = ", ny);
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("%n\n" "\n\n\n\n%\n\n\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("%n\n\n\n\n%\n\n\nCanonical coefficients for y\n");
for (i = 0; i < ny; ++i)
{
    for (j = 0; j < ncv; ++j)
        printf("%9.4f", CVY(i, j));
    printf("\n");
}
END:
NAG_FREE(z);
NAG_FREE(wt);
NAG_FREE(isz);
NAG_FREE(cvx);
NAG_FREE(cvy);
NAG_FREE(e);
return exit_status;

10.2 Program Data
nag_mv_canon_corr (g03adc) Example Program Data
9 4 2 2 U
80.0 58.4 14.0 21.0
75.0 59.2 15.0 27.0
78.0 60.3 15.0 27.0
75.0 57.4 13.0 22.0
79.0 59.5 14.0 26.0
78.0 58.1 14.5 26.0
75.0 58.0 12.5 23.0
64.0 55.5 11.0 22.0
80.0 59.2 12.5 22.0
-1 1 1 1

10.3 Program Results
nag_mv_canon_corr (g03adc) Example Program Results

Rank of x = 2 Rank of y = 2

Canonical Eigenvalues Percentage Chisq DF Sig

-
<table>
<thead>
<tr>
<th>correlations</th>
<th>variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9570</td>
<td>10.8916</td>
</tr>
<tr>
<td>0.3624</td>
<td>0.1512</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Canonical coefficients for x</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.4261</td>
</tr>
<tr>
<td>1.0337</td>
</tr>
<tr>
<td>-0.3444</td>
</tr>
<tr>
<td>-1.1136</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Canonical coefficients for y</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.1415</td>
</tr>
<tr>
<td>0.1504</td>
</tr>
<tr>
<td>-0.2384</td>
</tr>
<tr>
<td>-0.3424</td>
</tr>
</tbody>
</table>