NAG Library Function Document

nag_partial_corr (g02byc)

1 Purpose

nag_partial_corr (g02byc) computes a partial correlation/variance-covariance matrix from a correlation or variance-covariance matrix computed by nag_corr_cov (g02bxc).

2 Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_partial_corr (Integer m, Integer ny, Integer nx, const Integer sz[],
     const double r[], Integer tdr, double p[], Integer tdp, NagError *fail)
```

3 Description

Partial correlation can be used to explore the association between pairs of random variables in the presence of other variables. For three variables, y_1, y_2 and x_3 the partial correlation coefficient between y_1 and y_2 given x_3 is computed as:

$$\frac{r_{12} - r_{13}r_{23}}{\sqrt{\left(1 - r_{13}^2\right)\left(1 - r_{23}^2\right)}},$$

where r_{ij} is the product-moment correlation coefficient between variables with subscripts i and j. The partial correlation coefficient is a measure of the linear association between y_1 and y_2 having eliminated the effect due to both y_1 and y_2 being linearly associated with x_3. That is, it is a measure of association between y_1 and y_2 conditional upon fixed values of x_3. Like the full correlation coefficients the partial correlation coefficient takes a value in the range $(-1, 1)$ with the value 0 indicating no association.

In general, let a set of variables be partitioned into two groups Y and X with n_y variables in Y and n_x variables in X and let the variance-covariance matrix of all $n_y + n_x$ variables be partitioned into,

$$\begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}$$

The variance-covariance of Y conditional on fixed values of the X variables is given by:

$$\Sigma_{y|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}$$

The partial correlation matrix is then computed by standardizing $\Sigma_{y|x}$,

$$\mathrm{diag}\left(\Sigma_{y|x}\right)^{-\frac{1}{2}}\Sigma_{y|x}\mathrm{diag}\left(\Sigma_{y|x}\right)^{-\frac{1}{2}}.$$

To test the hypothesis that a partial correlation is zero under the assumption that the data has an approximately Normal distribution a test similar to the test for the full correlation coefficient can be used. If r is the computed partial correlation coefficient then the appropriate t statistic is

$$r\sqrt{\frac{n - n_x - 2}{1 - r^2}}$$

which has approximately a Student's t-distribution with $n - n_x - 2$ degrees of freedom, where n is the number of observations from which the full correlation coefficients were computed.

4 References

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

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

Osborn J F (1979) *Statistical Exercises in Medical Research* Blackwell

Snedecor G W and Cochran W G (1967) *Statistical Methods* Iowa State University Press

5 Arguments

1: **m** – Integer *Input*

On entry: the number of variables in the variance-covariance/correlation matrix given in **r**.

Constraint: $\mathbf{m} \geq 3$.

2: **ny** – Integer *Input*

On entry: the number of Y variables, n_y, for which partial correlation coefficients are to be computed.

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

3: **nx** – Integer *Input*

On entry: the number of X variables, n_x, which are to be considered as fixed.

Constraints:

$\mathbf{nx} \geq 1$;
$\mathbf{ny} + \mathbf{nx} \leq \mathbf{m}$.

4: **sz**[**m**] – const Integer *Input*

On entry: indicates which variables belong to set X and Y.

$\mathbf{sz}(i) < 0$
The ith variable is a Y variable, for $i = 1, 2, \ldots, \mathbf{m}$.

$\mathbf{sz}(i) > 0$
The ith variable is a X variable.

$\mathbf{sz}(i) = 0$
The ith variable is not included in the computations.

Constraints:

exactly **ny** elements of **sz** must be < 0,

exactly **nx** elements of **sz** must be > 0.

5: **r**[**m** × **tdr**] – const double *Input*

Note: the (i, j)th element of the matrix R is stored in $\mathbf{r}[(i-1) \times \mathbf{tdr} + j - 1]$.

On entry: the variance-covariance or correlation matrix for the **m** variables as given by nag_corr_cov (g02bxc). Only the upper triangle need be given.

Note: the matrix must be a full rank variance-covariance or correlation matrix and so be positive-definite. This condition is not directly checked by the function.

6: **tdr** – Integer *Input*

On entry: the stride separating matrix row elements in **r**.

Constraint: $\mathbf{tdr} \geq \mathbf{m}$

7: **p**[**ny** × **tdp**] – double *Output*

Note: the (i, j)th element of the matrix P is stored in $\mathbf{p}[(i-1) \times \mathbf{tdp} + j - 1]$.

On exit: the strict upper triangle of **p** contains the strict upper triangular part of the n_y by n_y partial correlation matrix. The lower triangle contains the lower triangle of the n_y by n_y partial variance-

covariance matrix if the matrix given in **r** is a variance-covariance matrix. If the matrix given in **r** is a correlation matrix then the variance-covariance matrix is for standardized variables.

8: **tdp** – Integer *Input*

On entry: the stride separating matrix row elements in **p**.

Constraint: **tdp** $\geq$ **ny**

9: **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, **tdp** = ⟨*value*⟩ while **ny** = ⟨*value*⟩. These arguments must satisfy **tdp** $\geq$ **ny**.

On entry, **tdr** = ⟨*value*⟩ while **m** = ⟨*value*⟩. These arguments must satisfy **tdr** $\geq$ **m**.

NE_3_INT_ARG_CONS

On entry, **ny** = ⟨*value*⟩, **nx** = ⟨*value*⟩ and **m** = ⟨*value*⟩. These arguments must satisfy **ny** + **nx** $\leq$ **m**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_NX_SET

On entry, **nx** = ⟨*value*⟩ and there are not exactly **nx** values of **sz** < 0.

NE_BAD_NY_SET

On entry, **ny** = ⟨*value*⟩ and there are not exactly **ny** values of **sz** < 0.
Number of values of **sz** < 0 = ⟨*value*⟩.

NE_COR_MAT_POSDEF

Either a diagonal element of the partial variance-covariance matrix is zero and/or a computed partial correlation coefficient is greater than one. Both indicate that the matrix input in **r** was not positive-definite.

NE_COR_MAT_RANK

On entry, either the variance-covariance matrix or the correlation matrix is not of full rank. Try removing some of the **nx** variables by setting the appropriate elements of **sz** to zero.

NE_INT_ARG_LT

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 3.

On entry, **nx** = ⟨*value*⟩.
Constraint: **nx** ≥ 1.

On entry, **ny** = ⟨*value*⟩.
Constraint: **ny** ≥ 2.

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.

7 Accuracy

nag_partial_corr (g02byc) computes the partial variance-covariance matrix, $\Sigma_{y|x}$, by computing the Cholesky factorization of Σ_{xx}. If Σ_{xx} is not of full rank the computation will fail.

8 Further Comments

Models that represent the linear associations given by partial correlations can be fitted using the multiple regression function nag_regsn_mult_linear (g02dac).

9 Example

Data, given by Osborn (1979), on the number of deaths, smoke $\left(mg/m^3\right)$ and sulphur dioxide (parts/million) during an intense period of fog is input. The correlations are computed using nag_corr_cov (g02bxc) and the partial correlation between deaths and smoke given sulphur dioxide is computed using nag_partial_corr (g02byc).

9.1 Program Text

```
/* nag_partial_corr (g02byc) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 */

#include <stdio.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define X(I, J) x[((I) -1)*m + ((J) -1)]
#define R(I, J) r[((I) -1)*m + ((J) -1)]
int main(int argc, char *argv[])
{
  FILE     *fpin, *fpout;

  Integer  exit_status = 0, j, k, m, n, nx, ny, *sz = 0;
  NagError fail;
  double   *r = 0, *std = 0, sw, *v, *x = 0, *xbar = 0;

  INIT_FAIL(fail);

  /* Check for command-line IO options */
  fpin = nag_example_file_io(argc, argv, "-data", NULL);
  fpout = nag_example_file_io(argc, argv, "-results", NULL);
  fprintf(fpout, "nag_partial_corr (g02byc) Example Program Results\n");

  /* Skip heading in data file */
  fscanf(fpin, "%*[^\n]");

  fscanf(fpin, "%ld %ld", &n, &m);
  if (!(r = NAG_ALLOC(m*m, double))
     || !(std = NAG_ALLOC(m, double))
     || !(v = NAG_ALLOC(m*m, double))
     || !(x = NAG_ALLOC(n*m, double))
     || !(xbar = NAG_ALLOC(m, double))
     || !(sz = NAG_ALLOC(m, Integer)))
    {
      fprintf(fpout, "Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (j = 1; j <= n; ++j)
    for (k = 1; k <= m; ++k)
```

```
      fscanf(fpin, "%lf", &X(j, k));

  /* Calculate correlation matrix */
  /* nag_corr_cov (g02bxc).
   * Product-moment correlation, unweighted/weighted
   * correlation and covariance matrix, allows variables to be
   * disregarded
   */
  nag_corr_cov(n, m, x, m, 0, 0, &sw, xbar, std, r, m, v, m,
               &fail);
  if (fail.code == NE_NOERROR)
    {
      /* Print the correlation matrix */
      fprintf(fpout, "\nCorrelation Matrix\n\n");
      for (j = 1; j <= m; j++)
        {
          for (k = 1; k <= m; k++)
            if (j > k)
              fprintf(fpout, "%11s", "");
            else
              fprintf(fpout, "%7.4f%4s", R(j, k), "");
          fprintf(fpout, "\n");
        }

      fscanf(fpin, "%ld %ld", &ny, &nx);
      for (j = 1; j <= m; ++j)
        fscanf(fpin, "%ld", &sz[j - 1]);

      /* Calculate partial correlation matrix */
      /* nag_partial_corr (g02byc).
       * Computes partial correlation/variance-covariance matrix
       * from correlation/variance-covariance matrix computed by
       * nag_corr_cov (g02bxc)
       */
      nag_partial_corr(m, ny, nx, sz, v, m, r, m, &fail);
      if (fail.code != NE_NOERROR)
        {
          fprintf(fpout, "Error from nag_partial_corr (g02byc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      /* Print partial correlation matrix */
      fprintf(fpout, "\n");
      fprintf(fpout, "\nPartial Correlation Matrix\n\n");
      for (j = 1; j <= ny; j++)
        {
          for (k = 1; k <= ny; k++)
            {
              if (j > k)
                fprintf(fpout, "%11s", "");
              else if (j == k)
                fprintf(fpout, "%7.4f%4s", 1.0, "");
              else
                fprintf(fpout, "%7.4f%4s", R(j, k), "");
            }
          fprintf(fpout, "\n");
        }
    }
  else
    {
      fprintf(fpout, "Error from nag_corr_cov (g02bxc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
END:
  if (fpin != stdin) fclose(fpin);
  if (fpout != stdout) fclose(fpout);
  if (r) NAG_FREE(r);
  if (std) NAG_FREE(std);
  if (v) NAG_FREE(v);
```

```
  if (x) NAG_FREE(x);
  if (xbar) NAG_FREE(xbar);
  if (sz) NAG_FREE(sz);
  return exit_status;
}
```

9.2 Program Data

```
nag_partial_corr (g02byc) Example Program Data
15 3
112 0.30 0.09
140 0.49 0.16
143 0.61 0.22
120 0.49 0.14
196 2.64 0.75
294 3.45 0.86
513 4.46 1.34
518 4.46 1.34
430 1.22 0.47
274 1.22 0.47
255 0.32 0.22
236 0.29 0.23
256 0.50 0.26
222 0.32 0.16
213 0.32 0.16

 2  1
-1 -1  1
```

9.3 Program Results

```
nag_partial_corr (g02byc) Example Program Results

Correlation Matrix

 1.0000     0.7560     0.8309
            1.0000     0.9876
                       1.0000


Partial Correlation Matrix

 1.0000    -0.7381
            1.0000
```