NAG Library Function Document

nag_corr_cov (g02bxc)

1 Purpose

nag_corr_cov (g02bxc) calculates the Pearson product-moment correlation coefficients and the variance-covariance matrix for a set of data. Weights may be used.

2 Specification

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

```
void nag_corr_cov (Integer n, Integer m, const double x[], Integer tdx,
     const Integer sx[], const double wt[], double *sw, double wmean[],
     double std[], double r[], Integer tdr, double v[], Integer tdv,
     NagError *fail)
```

3 Description

For n observations on m variables the one-pass algorithm of West (1979) as implemented in nag_sum_sqs (g02buc) is used to compute the means, the standard deviations, the variance-covariance matrix, and the Pearson product-moment correlation matrix for p selected variables. Suitables weights may be used to indicate multiple observations and to remove missing values. The quantities are defined by:

(a) The means

$$\bar{x}_j = \frac{\sum_{i=1}^{n} w_i x_{ij}}{\sum_{i=1}^{n} w_i} \quad j = 1, \ldots, p$$

(b) The variance-covariance matrix

$$C_{jk} = \frac{\sum_{i=1}^{n} w_i (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)}{\sum_{i=1}^{n} w_i - 1} \quad j, k = 1, \ldots, p$$

(c) The standard deviations

$$s_j = \sqrt{C_{jj}} \quad j = 1, \ldots, p$$

(d) The Pearson product-moment correlation coefficients

$$R_{jk} = \frac{C_{jk}}{\sqrt{C_{jj} C_{kk}}} \quad j, k = 1, \ldots, p$$

where x_{ij} is the value of the ith observation on the jth variable and w_i is the weight for the ith observation which will be 1 in the unweighted case.

Note that the denominator for the variance-covariance is $\sum_{i=1}^{n} w_i - 1$, so the weights should be scaled so that the sum of weights reflects the true sample size.

4 References

Chan T F, Golub G H and Leveque R J (1982) *Updating Formulae and a Pairwise Algorithm for Computing Sample Variances* Compstat, Physica-Verlag

West D H D (1979) Updating mean and variance estimates: An improved method *Comm. ACM* **22** 532–555

5 Arguments

1: **n** – Integer *Input*

On entry: the number of observations in the dataset, n.

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

2: **m** – Integer *Input*

On entry: the total number of variables, m.

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

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

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

4: **tdx** – Integer *Input*

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

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

5: $\mathbf{sx}[\mathbf{m}]$ – const Integer *Input*

On entry: indicates which p variables to include in the analysis.

$\mathbf{sx}[j-1] > 0$
The jth variable is to be included.

$\mathbf{sx}[j-1] = 0$
The jth variable is not to be included.

sx is set to **NULL**
All variables are included in the analysis, i.e., $p = m$.

Constraint: $\mathbf{sx}[i] \geq 0$, for $i = 1, 2, \ldots, m$.

6: $\mathbf{wt}[\mathbf{n}]$ – const double *Input*

On entry: w, the optional frequency weighting for each observation, with $\mathbf{wt}[i-1] = w_i$. Usually w_i will be an integral value corresponding to the number of observations associated with the ith data value, or zero if the ith data value is to be ignored. If **wt** is **NULL** then w_i is set to 1 for all i.

Constraint: if **wt** is not **NULL**, $\sum_{i=1}^{\mathbf{n}} \mathbf{wt}[i-1] > 1.0$, $\mathbf{wt}[i-1] \geq 0.0$, for $i = 1, 2, \ldots, \mathbf{n}$.

7: **sw** – double * *Output*

On exit: the sum of weights if **wt** is not **NULL**, otherwise **sw** contains the number of observations, n.

8: $\mathbf{wmean}[\mathbf{m}]$ – double *Output*

On exit: the sample means. $\mathbf{wmean}[j-1]$ contains the mean for the jth variable.

9: **std**[**m**] – double *Output*

On exit: the standard deviations. **std**[$j-1$] contains the standard deviation for the jth variable.

10: **r**[**m** $\times$ **tdr**] – double *Output*

On exit: the matrix of Pearson product-moment correlation coefficients. **r**[$(j-1) \times$ **tdr** $+ k - 1$] contains the correlation between variables j and k, for $j, k = 1, \ldots, p$.

11: **tdr** – Integer *Input*

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

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

12: **v**[**m** $\times$ **tdv**] – double *Output*

On exit: the variance-covariance matrix. **v**[$(j-1) \times$ **tdv** $+ k - 1$] contains the covariance between variables j and k, for $j, k = 1, \ldots, p$.

13: **tdv** – Integer *Input*

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

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

14: **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, **tdr** $= \langle value \rangle$ while **m** $= \langle value \rangle$.
The arguments must satisfy **tdr** $\geq$ **m**.

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

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_INT_ARG_LE

On entry, **n** must be greater than 1: **n** $= \langle value \rangle$.

NE_INT_ARG_LT

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

NE_NEG_SX

On entry, at least one element of **sx** is negative.

NE_NEG_WEIGHT

On entry, at least one of the weights is negative.

NE_POS_SX

On entry, no element of **sx** is positive.

NE_SW_LT_ONE

On entry, the sum of weights is less than 1.0.

NE_VAR_EQ_ZERO

A variable has zero variance.
At least one variable has zero variance. In this case **v** and **std** are as calculated, but **r** will contain zero for any correlation involving a variable with zero variance.

7 Accuracy

For a discussion of the accuracy of the one pass algorithm see Chan *et al.* (1982) and West (1979).

8 Parallelism and Performance

Not applicable.

9 Further Comments

Correlation coefficients based on ranks can be computed using nag_ken_spe_corr_coeff (g02brc).

10 Example

A program to calculate the means, standard deviations, variance-covariance matrix and a matrix of Pearson product-moment correlation coefficients for a set of 3 observations of 3 variables.

10.1 Program Text

```
/* nag_corr_cov (g02bxc) Example Program.
 *
 * Copyright 1992 Numerical Algorithms Group.
 *
 * Mark 3, 1992.
 * Mark 8 revised, 2004.
 */

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

#define X(I, J) x[(I) *tdx + J]
#define R(I, J) r[(I) *tdr + J]
#define V(I, J) v[(I) *tdv + J]
int main(void)
{
  Integer  exit_status = 0, i, j, m, n, tdr, tdv, tdx, test;
  NagError fail;
  char     w;
  double   *r = 0, *std = 0, sw, *v = 0, *wmean = 0, *wt = 0, *wtptr, *x = 0;

  INIT_FAIL(fail);

  printf("nag_corr_cov (g02bxc) Example Program Results\n");

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

  test = 0;
  while ((scanf("%ld%ld %c", &m, &n, &w) != EOF))
    {
      if (m >= 1 && n >= 1)
        {
          if (!(x = NAG_ALLOC(n*m, double)) ||
```

```
        !(r = NAG_ALLOC(m*m, double)) ||
        !(v = NAG_ALLOC(m*m, double)) ||
        !(wt = NAG_ALLOC(n, double)) ||
        !(wmean = NAG_ALLOC(m, double)) ||
        !(std = NAG_ALLOC(m, double)))
      {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
      }
    tdx = m;
    tdr = m;
    tdv = m;
  }
else
  {
    printf("Invalid m or n.\n");
    exit_status = 1;
    return exit_status;
  }
for (i = 0; i < n; i++)
  scanf("%lf", &wt[i]);
for (i = 0; i < n; i++)
  for (j = 0; j < m; j++)
    scanf("%lf", &X(i, j));

if (w == 'w')
  wtptr = wt;
else
  wtptr = (double *) 0;

/* nag_corr_cov (g02bxc).
 * Product-moment correlation, unweighted/weighted
 * correlation and covariance matrix, allows variables to be
 * disregarded
 */
nag_corr_cov(n, m, x, tdx, (Integer *) 0, wtptr, &sw, wmean, std,
             r, tdr, v, tdv, &fail);
if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_corr_cov (g02bxc).\n%s\n",
            fail.message);
    exit_status = 1;
    goto END;
  }


if (wtptr)
  printf("\nCase %ld --- Using weights\n", ++test);
else
  printf("\nCase %ld --- Not using weights\n", ++test);

printf("\nInput data\n");
for (i = 0; i < n; i++)
  printf("%6.1f%6.1f%6.1f%6.1f\n",
          X(i, 0), X(i, 1), X(i, 2), wt[i]);

printf("\n");
printf("Sample means.\n");
for (i = 0; i < m; i++)
  printf("%6.1f\n", wmean[i]);
printf("\nStandard deviation.\n");
for (i = 0; i < m; i++)
  printf("%6.1f\n", std[i]);

printf("\nCorrelation matrix.\n");
for (i = 0; i < m; i++)
  {
    for (j = 0; j < m; j++)
      printf("  %7.4f  ", R(i, j));
    printf("\n");
```

```
        }

      printf("\nVariance matrix.\n");
      for (i = 0; i < m; i++)
        {
          for (j = 0; j < m; j++)
            printf("  %7.3f  ", V(i, j));
          printf("\n");
        }
      printf("\nSum of weights %6.1f\n", sw);
 END:
      NAG_FREE(x);
      NAG_FREE(r);
      NAG_FREE(v);
      NAG_FREE(wt);
      NAG_FREE(wmean);
      NAG_FREE(std);
    }
  return exit_status;
}
```

10.2 Program Data

```
nag_corr_cov (g02bxc) Example Program Data
3 3 w
  9.1231   3.7011   4.5230
  0.9310   0.0900   0.8870
  0.0009   0.0099   0.0999
  0.1300   1.3070   0.3700

3 3 w
  0.1300   1.3070   0.3700
  9.1231   3.7011   4.5230
  0.9310   0.0900   0.8870
  0.0009   0.0099   0.0999

3 3 u
  0.717   9.370   0.013
  1.119   0.133   9.700
 11.100  23.510  11.117
  0.900   9.013   8.710

3 3 w
  0.717  19.370   0.013
  1.119   0.133   9.700
 11.100  23.510  11.117
  0.900   9.013  78.710

3 3 u
  0.717  19.370   0.013
  1.119   0.133   9.700
 11.100   3.510  13.117
  0.900   0.013  78.710

3 3 w
  0.717  19.370   0.913
  1.119   0.133   9.700
 17.100  93.510  13.117
 30.900   0.013  78.710
```

10.3 Program Results

```
nag_corr_cov (g02bxc) Example Program Results

Case 1 --- Using weights

Input data
   0.9   0.1   0.9   9.1
   0.0   0.0   0.1   3.7
   0.1   1.3   0.4   4.5
```

```
Sample means.
   0.5
   0.4
   0.6

Standard deviation.
   0.4
   0.6
   0.3

Correlation matrix.
   1.0000    -0.4932     0.9839
  -0.4932     1.0000    -0.3298
   0.9839    -0.3298     1.0000

Variance matrix.
    0.197     -0.123      0.149
   -0.123      0.316     -0.063
    0.149     -0.063      0.117

Sum of weights   17.3

Case 2 --- Using weights

Input data
   9.1   3.7   4.5   0.1
   0.9   0.1   0.9   1.3
   0.0   0.0   0.1   0.4

Sample means.
   1.3
   0.3
   1.0

Standard deviation.
   3.3
   1.4
   1.5

Correlation matrix.
   1.0000     0.9908     0.9903
   0.9908     1.0000     0.9624
   0.9903     0.9624     1.0000

Variance matrix.
   10.851      4.582      5.044
    4.582      1.971      2.089
    5.044      2.089      2.391

Sum of weights    1.8

Case 3 --- Not using weights

Input data
   1.1   0.1   9.7   0.7
  11.1  23.5  11.1   9.4
   0.9   9.0   8.7   0.0

Sample means.
   4.4
  10.9
   9.8

Standard deviation.
   5.8
  11.8
   1.2

Correlation matrix.
   1.0000     0.9193     0.9200
```

```
   0.9193     1.0000     0.6915
   0.9200     0.6915     1.0000

Variance matrix.
   33.951     63.208      6.485
   63.208    139.250      9.871
    6.485      9.871      1.464

Sum of weights    3.0

Case 4 --- Using weights

Input data
   1.1   0.1   9.7   0.7
  11.1  23.5  11.1  19.4
   0.9   9.0  78.7   0.0

Sample means.
  10.7
  22.7
  11.1

Standard deviation.
   1.9
   4.5
   1.8

Correlation matrix.
   1.0000     0.9985     0.0173
   0.9985     1.0000     0.0716
   0.0173     0.0716     1.0000

Variance matrix.
    3.672      8.538      0.059
    8.538     19.909      0.570
    0.059      0.570      3.185

Sum of weights   20.1

Case 5 --- Not using weights

Input data
   1.1   0.1   9.7   0.7
  11.1   3.5  13.1  19.4
   0.9   0.0  78.7   0.0

Sample means.
   4.4
   1.2
  33.8

Standard deviation.
   5.8
   2.0
  38.9

Correlation matrix.
   1.0000     0.9999    -0.4781
   0.9999     1.0000    -0.4881
  -0.4781    -0.4881     1.0000

Variance matrix.
   33.951     11.567    -108.343
   11.567      3.941    -37.687
  -108.343    -37.687    1512.750

Sum of weights    3.0

Case 6 --- Using weights

Input data
```

```
   1.1   0.1   9.7   0.7
  17.1  93.5  13.1  19.4
  30.9   0.0  78.7   0.9

Sample means.
  17.2
  86.3
  15.9

Standard deviation.
   4.2
  25.6
  13.7

Correlation matrix.
   1.0000    -0.0461     0.7426
  -0.0461     1.0000    -0.7033
   0.7426    -0.7033     1.0000

Variance matrix.
   17.846     -4.989     43.123
   -4.989    656.407   -247.692
   43.123   -247.692    188.970

Sum of weights   21.0
```