NAG Library Function Document

nag_cov_to_corr (g02bwc)

1 Purpose

nag_cov_to_corr (g02bwc) calculates a matrix of Pearson product-moment correlation coefficients from sums of squares and cross-products of deviations about the mean.

2 Specification

```c
#include <nag.h>
#include <nagg02.h>
void nag_cov_to_corr (Integer m, double r[], NagError *fail)
```

3 Description

nag_cov_to_corr (g02bwc) calculates a matrix of Pearson product-moment correlation coefficients from sums of squares and cross-products about the mean for observations on \( m \) variables which can be computed by a single call to nag_sum_sqs (g02buc) or a series of calls to nag_sum_sqs_update (g02btc). The sums of squares and cross-products are stored in an array packed by column and are overwritten by the correlation coefficients.

Let \( c_{jk} \) be the cross-product of deviations from the mean, for \( j = 1, 2, \ldots, m \) and \( k = j, \ldots, m \), then the product-moment correlation coefficient, \( r_{jk} \) is given by

\[
r_{jk} = \frac{c_{jk}}{\sqrt{c_{jj}c_{kk}}}
\]

4 References

None.

5 Arguments

1: \( m \) – Integer

*Input*

On entry: \( m \), the number of variables.

Constraint: \( m \geq 1 \).

2: \( r[(m \times m + m)/2] \) – double

*Input/Output*

On entry: contains the upper triangular part of the sums of squares and cross-products matrix of deviations from the mean. These are stored packed by column, i.e., the cross-product between variable \( j \) and \( k \), \( k \geq j \), is stored in \( r[(k \times (k - 1)/2 + j) - 1] \).


These are stored packed by column corresponding to the input cross-products.

3: \( fail \) – NagError *

*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).
6 Error Indicators and Warnings

NE_ALLOC_FAIL
   Dynamic memory allocation failed.
   See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
   On entry, argument <value> had an illegal value.

NE_INT
   On entry, m = <value>.
   Constraint: m ≥ 1.

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.
   An unexpected error has been triggered by this function. Please contact NAG.
   See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
   Your licence key may have expired or may not have been installed correctly.
   See Section 3.6.5 in the Essential Introduction for further information.

NE_ZERO_VARIANCE
   On entry, a variable has zero variance.

7 Accuracy

The accuracy of nag_cov_to_corr (g02bwc) is entirely dependent upon the accuracy of the elements of
array r.

8 Parallelism and Performance

Not applicable.

9 Further Comments

nag_cov_to_corr (g02bwc) may also be used to calculate the correlations between parameter estimates
from the variance-covariance matrix of the parameter estimates as is given by several functions in this
chapter.

10 Example

A program to calculate the correlation matrix from raw data. The sum of squares and cross-products
about the mean are calculated from the raw data by a call to nag_sum_sq (g02bce). The correlation
matrix is then calculated from these values.
10.1 Program Text

/* nag_cov_to_corr (g02bwc) Example Program. *
* Copyright 2014 Numerical Algorithms Group.
* Mark 7, 2002. */

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

int main(void) {
  /* Arrays */
  char nag_enum_mean[40], nag_enum_weight[40];
  double *c = 0, *wmean = 0, *wt = 0, *x = 0;
  double *wtptr = 0;
  /* Scalars */
  double sw;
  Integer exit_status, j, k, m, n, pdx;
  Nag_OrderType order;
  Nag_SumSquare mean;
  Nag_Boolean weight;
  NagError fail;

  #ifdef NAG_LOAD_FP
  /* The following line is needed to force the Microsoft linker
  to load floating point support */
  float force_loading_of_ms_float_support = 0;
  #endif /* NAG_LOAD_FP */

  #ifdef NAG_COLUMN_MAJOR
  #define X(I, J) x[(J-1)*pdx + I-1]
  order = Nag_ColMajor;
  #else
  #define X(I, J) x[(I-1)*pdx+J-1]
  order = Nag_RowMajor;
  #endif

  INIT_FAIL(fail);
  exit_status = 0;
  printf("nag_cov_to_corr (g02bwc) Example Program Results\n");

  #ifdef _WIN32
  scanf_s("%39s %39s %*[\n]", nag_enum_mean, nag_enum_weight, &m, &n);
  #else
  scanf("%39s %39s %*[\n]", nag_enum_mean, nag_enum_weight, &m, &n);
  #endif

  #ifdef _WIN32
  while (scanf_s("%39s %39s %*[\n]", nag_enum_mean, nag_enum_weight, &m, &n) != EOF) {
  #else
  while (scanf("%39s %39s %*[\n]", nag_enum_mean, nag_enum_weight, &m, &n) != EOF) {
  #endif
  /* nag_enum_name_to_value (x04nac).
  * Converts NAG enum member name to value */
  mean = (Nag_SumSquare) nag_enum_name_to_value(nag_enum_mean);
  weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_weight);
/* Allocate memory */
if (!(c = NAG_ALLOC((m*(m+1))/2, double)) || 
!(wmean = NAG_ALLOC(m, double)) ||
!(wt = NAG_ALLOC(n, double)) ||
!(x = NAG_ALLOC(n * m, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
#endif NAG_COLUMN_MAJOR
pdx = n;
#else
    pdx = m;
#endif
for (j = 1; j <= n; ++j)
    #ifdef _WIN32
        scanf_s("%lf", &wt[j-1]);
    #else
        scanf("%lf", &wt[j-1]);
    #endif
    #ifdef _WIN32
        scanf_s("%*[^
]");
    #else
        scanf("%*[^
]");
    #endif
    for (j = 1; j <= n; ++j)
    {
        for (k = 1; k <= m; ++k)
            #ifdef _WIN32
                scanf_s("%lf", &X(j, k));
            #else
                scanf("%lf", &X(j, k));
            #endif
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
        if (weight)
            wtptr = wt;
    /* Calculate the sums of squares and cross-products matrix */
    /* nag_sum_sqs (g02buc). */
    /* Computes a weighted sum of squares matrix */
    nag_sum_sqs(order, mean, n, m, x, pdx, wtptr, &sw, wmean, c, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_sum_sqs (g02buc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    /* Calculate the correlation matrix */
    /* nag_cov_to_corr (g02bwc). */
    /* Computes a correlation matrix from a sum of squares */
    /* matrix */
    nag_cov_to_corr(m, c, &fail);
    /* Print the correlation matrix */
    if (fail.code == NE_NOERROR)
    {
        printf("\n");
    /* nag_pack_real_mat_print (x04ccc). */
    /* Print real packed triangular matrix (easy-to-use)*/
fflush(stdout);
nag_pack_real_mat_print(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, c, "Correlation matrix", 0, &fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_pack_real_mat_print (x04ccc).\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
else if (fail.code == NE_ZERO_VARIANCE)
{
  printf("\n");  
  printf("NOTE: some variances are zero\n\n");  
  /* nag_pack_real_mat_print (x04ccc), see above. */  
  fflush(stdout);
  nag_pack_real_mat_print(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, c, "Correlation matrix", 0, &fail);
  if (fail.code != NE_NOERROR)
  {
    printf("Error from nag_pack_real_mat_print (x04ccc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
}
else
{
  printf("Error from nag_cov_to_corr (g02bwc).\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

NAG_FREE(c);
NAG_FREE(wmean);
NAG_FREE(wt);
NAG_FREE(x);

END:
NAG_FREE(c);
NAG_FREE(wmean);
NAG_FREE(wt);
NAG_FREE(x);

return exit_status;

10.2 Program Data
nag_cov_to_corr (g02bwc) Example Program Data
  Nag_AboutMean Nag_TRUE 3 3
  0.1300  1.3070  0.3700
  9.1231  3.7011  4.5230
  0.9310  0.0900  0.8870
  0.0009  0.0099  0.0999
10.3 Program Results

nag_cov_to_corr (g02bwc) Example Program Results

Correlation matrix

\[
\begin{array}{ccc}
1 & 2 & 3 \\
1 & 1.0000 & 0.9908 & 0.9903 \\
2 & 1.0000 & 0.9624 & \\
3 & & 1.0000 & \\
\end{array}
\]