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

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 variables \(j = 1, 2, \ldots, m; k = j, j + 1, \ldots, m,\)
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 Parameters

1:  
\(m\) – Integer

Input

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

Constraint: \(m \geq 1\).

2:  
\(r[\text{dim}]\) – double

Input/Output

Note: the dimension, \(\text{dim}\), of the array \(r\) must be at least \((m \times m + m)/2\).

On entry: \(r\) 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 \(r(k \times (k - 1)/2 + j)\).


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

3:  
\(\text{fail}\) – NagError *

Input/Output

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, \(m = \langle\text{value}\rangle\).

Constraint: \(m \geq 1\).
NE_ZERO_VARIANCE
On entry, a variable has zero variance.

NE_BAD_PARAM
On entry, parameter (value) had an illegal value.

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 consult NAG for assistance.

7 Accuracy
The accuracy of this routine is entirely dependent upon the accuracy of the elements of array r.

8 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 routines in this chapter.

9 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_sqs (g02buc). The correlation matrix is then calculated from these values.

9.1 Program Text
/* nag_cov_to_corr (g02bwc) Example Program. */
/* Copyright 2002 Numerical Algorithms Group. */
/* Mark 7, 2002. */
#include <stdio.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double sw;
    Integer exit_status, j, k, m, n, pdx;
    NagError fail;
    Nag_OrderType order;
    Nag_SumSquare mean_enum;

    /* Arrays */
    char mean[2], weight[2];
    double *c=0, *wmean=0, *wt=0, *x=0;
    double *wtptr=0;

    #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

    ...
#include< stdio.h >
#include< string.h >

int main()
{
    double mean[0], weight[0];
    long m, n;

    // Read input
    scanf("%ld%ld%*\[\n\] ", m, n);

    // Allocate memory
    double *c, *wmean, *wt, *x;
    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)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    // Read weights
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &wt[j-1]);

    // Read data
    for (j = 1; j <= n; ++j)
        for (k = 1; k <= m; ++k)
            Vscanf("%lf", &x[j,k]);

    // Process mean
    if (mean[0] == 'M')
        mean_enum = Nag_AboutMean;
    else if (mean[0] == 'Z')
        mean_enum = Nag_AboutZero;
    else
    {
        Vprintf("Incorrect value for mean\n");
        exit_status = -1;
        goto END;
    }

    // Process weights
    wtptr = wt;

    // Calculate sums of squares and cross-products
    g02buc(order, mean_enum, n, m, x, pdx, wtptr, &sw, wmean, c, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from g02buc.\n\n", fail.message);
        exit_status = 1;
        goto END;
    }

    // Calculate correlation matrix
    g02bwc(m, c, &fail);

    // Print correlation matrix
    if (fail.code == NE_NOERROR)
    {
        Vprintf("\n");
        x04ccc(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, c,
               "Correlation matrix", 0, &fail);
        if (fail.code != NE_NOERROR)
        {
            Vprintf("Error from x04ccc.\n\n", fail.message);
        }
    }
    else
    {
        Vprintf("\n");
    }

    return 0;
}

#include< stdio.h >
#include< string.h >

int main()
{
    double mean[0], weight[0];
    long m, n;

    // Read input
    scanf("%ld%ld%*\[\n\] ", m, n);

    // Allocate memory
    double *c, *wmean, *wt, *x;
    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)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    // Read weights
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &wt[j-1]);

    // Read data
    for (j = 1; j <= n; ++j)
        for (k = 1; k <= m; ++k)
            Vscanf("%lf", &x[j,k]);

    // Process mean
    if (mean[0] == 'M')
        mean_enum = Nag_AboutMean;
    else if (mean[0] == 'Z')
        mean_enum = Nag_AboutZero;
    else
    {
        Vprintf("Incorrect value for mean\n");
        exit_status = -1;
        goto END;
    }

    // Process weights
    wtptr = wt;

    // Calculate sums of squares and cross-products
    g02buc(order, mean_enum, n, m, x, pdx, wtptr, &sw, wmean, c, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from g02buc.\n\n", fail.message);
        exit_status = 1;
        goto END;
    }

    // Calculate correlation matrix
    g02bwc(m, c, &fail);

    // Print correlation matrix
    if (fail.code == NE_NOERROR)
    {
        Vprintf("\n");
        x04ccc(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, c,
               "Correlation matrix", 0, &fail);
        if (fail.code != NE_NOERROR)
        {
            Vprintf("Error from x04ccc.\n\n", fail.message);
        }
    }
    else
    {
        Vprintf("\n");
    }

    return 0;
}
```c
exit_status = 1;
goto END;
}
}
else if (fail.code == NE_ZERO_VARIANCE)
{
    Vprintf("\n\nNOTE: some variances are zero\n\n");
    x04ccc(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, c,
        "Correlation matrix", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from x04ccc.\n\ns\n", fail.message);
        exit_status = 1;
        goto END;
    }
    else
    {
        Vprintf("Error from g02bwc.\n\ns\n", fail.message);
        exit_status = 1;
        goto END;
    }
}
else
{
    Vprintf("Error from g02bwc.\n\ns\n", fail.message);
    exit_status = 1;
    goto END;
}
if (c) NAG_FREE(c);
if (wmean) NAG_FREE(wmean);
if (wt) NAG_FREE(wt);
if (x) NAG_FREE(x);

END:
if (c) NAG_FREE(c);
if (wmean) NAG_FREE(wmean);
if (wt) NAG_FREE(wt);
if (x) NAG_FREE(x);
return exit_status;

9.2 Program Data

9.2.1 g02bwc Example Program Data

'M' 'W' 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

9.3 Program Results

9.3.1 g02bwc Example Program Results

Correlation matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>0.9908</td>
<td>0.9903</td>
</tr>
<tr>
<td>2</td>
<td>1.0000</td>
<td>0.9624</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```