1 Purpose

nag_sum_sqs_update (g02btc) updates the sample means and sums of squares and cross-products, or sums of squares and cross-products of deviations about the mean, for a new observation. The data may be weighted.

2 Specification

```c
void nag_sum_sqs_update (Nag_SumSquare mean, Integer m, double wt,
const double x[], Integer incx, double *sw, double xbar[], double c[],
NagError *fail)
```

3 Description

nag_sum_sqs_update (g02btc) is an adaptation of West’s WV2 algorithm; see West (1979). This routine updates the weighted means of variables and weighted sums of squares and cross-products or weighted sums of squares and cross-products of deviations about the mean for observations on $m$ variables $X_j$, for $j = 1, 2, \ldots, m$. For the first $i-1$ observations let the mean of the $j$th variable be $\bar{x}_j(i-1)$, the cross-product about the mean for the $j$th and $k$th variables be $c_{jk}(i-1)$ and the sum of weights be $W_{i-1}$. These are updated by the $i$th observation, $x_{ij}$, for $j = 1, 2, \ldots, m$, with weight $w_i$ as follows:

$$W_i = W_{i-1} + w_i, \quad \bar{x}_j(i) = \bar{x}_j(i-1) + \frac{w_i}{W_i} (x_{ij} - \bar{x}_j(i-1)), \quad j = 1, 2, \ldots, m$$

and

$$c_{jk}(i) = c_{jk}(i-1) + \frac{w_i}{W_i} (x_{ij} - \bar{x}_j(i-1))(x_{ik} - \bar{x}_k(i-1))W_{i-1}, \quad j = 1, 2, \ldots, m; \quad k = j, j+1, 2, \ldots, m.$$ 

The algorithm is initialised by taking $\bar{x}_j(1) = x_{1j}$, the first observation and $c_{ij}(1) = 0.0$.

For the unweighted case $w_i = 1$ and $W_i = i$ for all $i$.

4 References


5 Parameters

1. `mean` – Nag_SumSquare

   **Input**

   On entry: indicates whether nag_sum_sqs_update (g02btc) is to calculate sums of squares and cross-products, or sums of squares and cross-products of deviations about the mean.

   If `mean = Nag_AboutMean`, the sums of squares and cross-products of deviations about the mean are calculated.

   If `mean = Nag_AboutZero`, the sums of squares and cross-products are calculated.

   **Constraint:** `mean = Nag_AboutMean` or `Nag_AboutZero`.
2: \( m \) – Integer
   \( On \ entry: \) the number, \( m \), of variables.
   \( Constraint: \ m \geq 1. \)

3: \( wt \) – double
   \( On \ entry: \) the weight to use for the current observation, \( w_i \).
   For unweighted means and cross-products set \( wt = 1.0 \). The use of a suitable negative value of \( wt \), e.g., \(-w_i\), will have the effect of deleting the observation.

4: \( x[dim] \) – const double
   \( Note: \) the dimension, \( dim \), of the array \( x \) must be at least \( m \times incx \).
   \( On \ entry: \) \( x[(j-1)incx] \) must contain the value of the \( j \)th variable for the current observation, \( j = 1,2,\ldots,m. \)

5: \( incx \) – Integer
   \( On \ entry: \) the increment of \( x \).
   \( Constraint: \ incx > 0. \)

6: \( sw \) – double *
   \( Input/Output \)
   \( On \ entry: \) the sum of weights for the previous observations, \( W_{i-1} \).
   If \( sw = 0.0 \), the update procedure is initialised.
   If \( sw + wt = 0.0 \), then all elements of \( xbar \) and \( c \) are set to zero.
   \( Constraint: \ sw \geq 0.0 \) and \( sw + wt \geq 0.0. \)
   \( On \ exit: \) \( sw \) contains the updated sum of weights, \( W_i \).

7: \( xbar[m] \) – double
   \( Input/Output \)
   \( On \ entry: \) \( xbar[j-1] \) must contain the weighted mean of the \( j \)th variable for the previous \((i-1)\) observations, \( \bar{x}_j(i-1) \), for \( j = 1,2,\ldots,m. \)
   \( On \ exit: \) \( xbar[j-1] \) contains the weighted mean of the \( j \)th variable, \( \bar{x}_j(i) \), for \( j = 1,2,\ldots,m. \)

8: \( c[dim] \) – double
   \( Input/Output \)
   \( Note: \) the dimension, \( dim \), of the array \( c \) must be at least \((m \times m + m)/2\).
   \( On \ entry: \) if \( sw \neq 0.0 \), \( c \) must contain the upper triangular part of the matrix of weighted sums of squares and cross-products or weighted sums of squares and cross-products of deviations about the mean. It is stored packed form by column, i.e., the cross-product between the \( j \)th and \( k \)th variable, \( k \geq j \), is stored in \( c[k \times (k-1)/2 + j - 1] \).
   \( On \ exit: \) the update sums of squares and cross-products stored as on input.

9: \( fail \) – NagError *
   \( Input/Output \)
   The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT
   \( On \ entry, \ incx = \langle value \rangle. \)
   \( Constraint: \ incx \geq 1. \)
   \( On \ entry, \ m = \langle value \rangle. \)
   \( Constraint: \ m \geq 1. \)
NE_REAL
On entry, sw = \langle value\rangle.
Constraint: sw \geq 0.0.

NE_SUM_WEIGHT
On entry, (sw + wt) < 0.0: (sw + wt) = \langle value\rangle.

NE_BAD_PARAM
On entry, parameter \langle value\rangle 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
For a detailed discussion of the accuracy of this method see Chan \textit{et al.} (1982) and West (1979).

8 Further Comments
\begin{itemize}
\item nag_sum_sq_update (g02btc) may be used to update the results returned by nag_sum_sq (g02buc).
\item nag_cov_to_corr (g02bwc) may be used to calculate the correlation matrix from the matrix of sums of
squares and cross-products of deviations about the mean.
\end{itemize}

9 Example
A program to calculate the means, the required sums of squares and cross-products matrix, and the
variance matrix for a set of 3 observations of 3 variables.

9.1 Program Text
\begin{verbatim}
/* nag_sum_sq_update (g02btc) Example Program. */
/* Copyright 2002 Numerical Algorithms Group. */
/* Mark 7, 2002. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nag_string.h>
#include <nagf06.h>
#include <nagg02.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  double alpha, sw, wt;
  Integer exit_status, i, j, m, mm, n, nprint, incx;
  NagError fail;
  Nag_SumSquare mean_enum;

  /* Arrays */
  char mean[2];
  double *c=0, *v=0, *x=0, *xbar=0;
  INIT_FAIL(fail);
  exit_status = 0;
  Vprintf("g02btc Example Program Results\n");

  return exit_status;
}
\end{verbatim}
/* Skip heading in data file */
Vscanf("%*[\n] ");

incx = 1;
while (scanf("' %ls '1d%ld%ld%ld%*[\n]", mean, &m, &n, &nprint) != EOF)
{
/* Allocate memory */
if ( !(c = NAG_ALLOC((m*m+m)/2, double)) ||
    !(v = NAG_ALLOC((m*m+m)/2, double)) ||
    !(x = NAG_ALLOC(m*incx, double)) ||
    !(xbar = NAG_ALLOC(m, double)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
sw = 0.0;
for (i = 1; i <= n; ++i)
{
    Vscanf("%lf", &wt);
    for (j = 1; j <= m; ++j)
        Vscanf("%lf", &x[j - 1]);
    Vscanf("%*[\n] ");
    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;
    }
    /* Calculate the sums of squares and cross-products matrix */
g02btc(mean_enum, m, wt, x, incx, &sw, xbar, c, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from g02btc.\n\n", fail.message);
        exit_status = 1;
        goto END;
    }
    if (i % nprint == 0 || i == n)
    {
        Vprintf("\n");
        Vprintf("---------------------------------------------\n");
        Vprintf("Observation: %4ld Weight = %13.4f\n", i, wt);
        Vprintf("---------------------------------------------\n");
        Vprintf("\n");
        Vprintf("Means\n");
        for (j = 1; j <= m; ++j)
            Vprintf("%14.4f%s", xbar[j - 1], j%4 == 0 || j == m ? "\n": "");
        Vprintf("\n");
    } /* Print the sums of squares and cross products matrix */
    x04ccc(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, c,
    "Sums of squares and cross-products", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from x04ccc.\n\n", fail.message);
        exit_status = 1;
        goto END;
    }
    if (sw > 1.0)
}
{ /* Calculate the variance matrix */
alpha = 1.0 / (sw - 1.0);
mm = m * (m + 1) / 2;
f06fdc(mm, alpha, c, 1, v, 1);
/* Print the variance matrix */
Vprintf("\n");
x04ccc(Nag_ColMajor, Nag_Upper, Nag_NonUnitDiag, m, v,
"Variance matrix", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04ccc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
}
if (c) NAG_FREE(c);
if (v) NAG_FREE(v);
if (x) NAG_FREE(x);
if (xbar) NAG_FREE(xbar);

END:
if (c) NAG_FREE(c);
if (v) NAG_FREE(v);
if (x) NAG_FREE(x);
if (xbar) NAG_FREE(xbar);
return exit_status;
}

9.2 Program Data

g02btc Example Program Data
'M' 3 3 3
0.1300 9.1231 3.7011 4.5230
1.3070 0.9310 0.0900 0.8870
0.3700 0.0099 0.0999

9.3 Program Results

g02btc Example Program Results
---------------------------------------------
Observation: 3 Weight = 0.3700
---------------------------------------------
Means
1.3299 0.3334 0.9874

Sums of squares and cross-products
1 2 3
1 8.7569 3.6978 4.0707
2 1.5905 1.6861
3 1.9297

Variance matrix
1 2 3
1 10.8512 4.5822 5.0443
2 1.9709 2.0893
3 2.3912

[g02btc.5 (last)]