NAG Library Routine Document

G02BTF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G02BTF 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

```
SUBROUTINE G02BTF (MEAN, M, WT, X, INCX, SW, XBAR, C, IFAIL)
INTEGER             M, INCX, IFAIL
REAL (KIND=nag_wp) WT, X(M*INCX), SW, XBAR(M), C((M*M+M)/2)
CHARACTER(1)        MEAN
```

3 Description

G02BTF 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 jth variable be $\bar{x}_j(i-1)$, the cross-product about the mean for the jth and kth variables be $c_{jk}(i-1)$ and the sum of weights be W_{i-1}. These are updated by the ith observation, x_{ij}, for $j = 1, 2, \ldots, m$, with weight w_i as follows:

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

and

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

The algorithm is initialized 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

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 Parameters

1: MEAN – CHARACTER(1) *Input*

On entry: indicates whether G02BTF is to calculate sums of squares and cross-products, or sums of squares and cross-products of deviations about the mean.

MEAN = 'M'
The sums of squares and cross-products of deviations about the mean are calculated.

MEAN = 'Z'
The sums of squares and cross-products are calculated.

Constraint: MEAN = 'M' or 'Z'.

2: M – INTEGER *Input*

On entry: m, the number of variables.

Constraint: M ≥ 1.

3: WT – REAL (KIND=nag_wp) *Input*

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(M × INCX) – REAL (KIND=nag_wp) array *Input*

On entry: X$((j-1) \times \text{INCX} + 1)$ must contain the value of the jth variable for the current observation, $j = 1, 2, \ldots, m$.

5: INCX – INTEGER *Input*

On entry: the increment of X. Two situations are common.

If INCX = 1, the data values are to be found in consecutive locations in X, i.e., in a column.

If INCX = ldx, for some positive integer ldx, the data values are to be found as a row of an array with first dimension ldx.

Constraint: INCX > 0.

6: SW – REAL (KIND=nag_wp) *Input/Output*

On entry: the sum of weights for the previous observations, W_{i-1}.

SW = 0.0
The update procedure is initialized.

SW + WT = 0.0
All elements of XBAR and C are set to zero.

Constraint: SW ≥ 0.0 and SW + WT ≥ 0.0.

On exit: contains the updated sum of weights, W_i.

7: XBAR(M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if SW = 0.0, XBAR is initialized, otherwise XBAR(j) must contain the weighted mean of the jth variable for the previous $(i-1)$ observations, $\bar{x}_j(i-1)$, for $j = 1, 2, \ldots, m$.

On exit: XBAR(j) contains the weighted mean of the jth variable, $\bar{x}_j(i)$, for $j = 1, 2, \ldots, m$.

8: C((M × M + M)/2) – REAL (KIND=nag_wp) array *Input/Output*

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 jth and kth variable, $k \geq j$, is stored in C$(k \times (k-1)/2 + j)$.

On exit: the update sums of squares and cross-products stored as on input.

9: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, M < 1,
or INCX < 1.

IFAIL $= 2$

On entry, SW < 0.0.

IFAIL $= 3$

On entry, $(\text{SW} + \text{WT}) < 0.0$, the current weight causes the sum of weights to be less than 0.0.

IFAIL $= 4$

On entry, MEAN $\neq$ 'M' or 'Z'.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

For a detailed discussion of the accuracy of this method see Chan *et al.* (1982) and West (1979).

8 Parallelism and Performance

Not applicable.

9 Further Comments

G02BTF may be used to update the results returned by G02BUF.

G02BWF may be used to calculate the correlation matrix from the matrix of sums of squares and cross-products of deviations about the mean and the matrix may be scaled using F06EDF (DSCAL) or F06FDF to produce a variance-covariance matrix.

10 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.

10.1 Program Text

```
    Program g02btfe

!     G02BTF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: g02btf, nag_wp, x04ccf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0_nag_wp
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: alpha, sw, wt
      Integer                          :: i, ifail, incx, lc, m, n, nprint
      Character (1)                    :: mean
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: c(:), v(:), x(:), xbar(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: mod
!     .. Executable Statements ..
      Write (nout,*) 'G02BTF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in problem size
      Read (nin,*) mean, m, n, nprint

      lc = (m*m+m)/2
      Allocate (x(m),xbar(m),c(lc),v(lc))

!     Elements of X are stored consecutively
      incx = 1

!     Loop over each observation individually, updating the sums of squares
!     and cross-product matrix at each iteration
      sw = zero
      i = 0
data_lp: Do
        Read (nin,*,Iostat=ifail) wt, x(1:m)
        If (ifail/=0) Then
!         Finished processing all the data
          Exit data_lp
        End If

        i = i + 1

!       Update the sums of squares and cross-products matrix
        ifail = 0
        Call g02btf(mean,m,wt,x,incx,sw,xbar,c,ifail)

!       Display the results, either at the end or every NPRINT iterations
        If (mod(i,nprint)==0 .Or. i==n) Then
```

```
          Write (nout,*) '-------------------------------------------'
          Write (nout,99999) 'Observation: ', i, '      Weight = ', wt
          Write (nout,*) '-------------------------------------------'
          Write (nout,*)
          Write (nout,*) 'Means'
          Write (nout,99998) xbar(1:m)
          Write (nout,*)
          Flush (nout)
          ifail = 0
          Call x04ccf('Upper','Non-unit',m,c, &
            'Sums of squares and cross-products',ifail)

!         Convert the sums of squares and cross-products to a variance matrix
          If (sw>one) Then
            alpha = one/(sw-one)
            v(1:lc) = alpha*c(1:lc)
            Write (nout,*)
            Flush (nout)
            ifail = 0
            Call x04ccf('Upper','Non-unit',m,v,'Variance matrix',ifail)
          End If
          Write (nout,*)
        End If
      End Do data_lp

99999 Format (1X,A,I4,A,F13.4)
99998 Format (1X,4F14.4)
    End Program g02btfe
```

10.2 Program Data

```
G02BTF 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   0.0009   0.0099   0.0999
```

10.3 Program Results

```
 G02BTF 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
```