NAG Library Routine Document

G02BZF

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

G02BZF combines two sets of sample means and sums of squares and cross-products matrices. It is designed to be used in conjunction with G02BUF to allow large datasets to be summarised.

2 Specification

```
SUBROUTINE G02BZF (MEAN, M, XSW, XMEAN, XC, YSW, YMEAN, YC, IFAIL)

INTEGER             M, IFAIL
REAL (KIND=nag_wp)  XSW, XMEAN(M), XC((M*M+M)/2), YSW, YMEAN(M),              &
                    YC((M*M+M)/2)
CHARACTER(1)        MEAN
```

3 Description

Let X and Y denote two sets of data, each with m variables and n_x and n_y observations respectively. Let μ_x denote the (optionally weighted) vector of m means for the first dataset and C_x denote either the sums of squares and cross-products of deviations from μ_x

$$C_x = \left(X - e\mu_x^{\mathrm{T}}\right)^{\mathrm{T}} D_x \left(X - e\mu_x^{\mathrm{T}}\right)$$

or the sums of squares and cross-products, in which case

$$C_x = X^{\mathrm{T}} D_x X$$

where e is a vector of n ones and D_x is a diagonal matrix of (optional) weights, with the sum of the diagonal elements of D_x equal to W_x. Similarly, let μ_y, C_y and W_y denote the same quantities for the second dataset.

Given $\mu_x, \mu_y, C_x, C_y, W_x$ and W_y G02BZF calculates μ_z, C_z and W_z as if a dataset Z, with m variables and $n_x + n_y$ observations were supplied to G02BUF, with Z constructed as

$$Z = \begin{pmatrix} X \\ Y \end{pmatrix}.$$

G02BZF has been designed to combine the results from two calls to G02BUF allowing large datasets, or cases where all the data is not available at the same time, to be summarised.

4 References

Bennett J, Pebay P, Roe D and Thompson D (2009) Numerically stable, single-pass, parallel statistics algorithms *Proceedings of IEEE International Conference on Cluster Computing*

5 Parameters

1: MEAN – CHARACTER(1) *Input*

On entry: indicates whether the matrices supplied in XC and YC are sums of squares and cross-products, or sums of squares and cross-products of deviations about the mean.

MEAN = 'M'
Sums of squares and cross-products of deviations about the mean have been supplied.

MEAN = 'Z'
Sums of squares and cross-products have been supplied.

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

2: M – INTEGER *Input*

On entry: m, the number of variables.

Constraint: M ≥ 1.

3: XSW – REAL (KIND=nag_wp) *Input/Output*

On entry: W_x, the sum of weights, from the first set of data, X. If the data is unweighted then this will be the number of observations in the first dataset.

On exit: W_z, the sum of weights, from the combined dataset, Z. If both datasets are unweighted then this will be the number of observations in the combined dataset.

Constraint: XSW ≥ 0.

4: XMEAN(M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: μ_x, the sample means for the first set of data, X.

On exit: μ_z, the sample means for the combined data, Z.

5: XC($(\mathrm{M} \times \mathrm{M} + \mathrm{M})/2$) – REAL (KIND=nag_wp) array *Input/Output*

On entry: C_x, the sums of squares and cross-products matrix for the first set of data, X, as returned by G02BUF.

G02BUF, returns this matrix packed by columns, i.e., the cross-product between the jth and kth variable, $k \geq j$, is stored in XC$(k \times (k-1)/2 + j)$.

No check is made that C_x is a valid cross-products matrix.

On exit: C_z, the sums of squares and cross-products matrix for the combined dataset, Z.

This matrix is again stored packed by columns.

6: YSW – REAL (KIND=nag_wp) *Input*

On entry: W_y, the sum of weights, from the second set of data, Y. If the data is unweighted then this will be the number of observations in the second dataset.

Constraint: YSW ≥ 0.

7: YMEAN(M) – REAL (KIND=nag_wp) array *Input*

On entry: μ_y, the sample means for the second set of data, Y.

8: YC($(\mathrm{M} \times \mathrm{M} + \mathrm{M})/2$) – REAL (KIND=nag_wp) array *Input*

On entry: C_y, the sums of squares and cross-products matrix for the second set of data, Y, as returned by G02BUF.

G02BUF, returns this matrix packed by columns, i.e., the cross-product between the jth and kth variable, $k \geq j$, is stored in YC$(k \times (k-1)/2 + j)$.

No check is made that C_y is a valid cross-products matrix.

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 $= 11$

On entry, MEAN $= \langle value \rangle$.
Constraint: MEAN = 'M' or 'Z'.

IFAIL $= 21$

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

IFAIL $= 31$

On entry, XSW $= \langle value \rangle$.
Constraint: XSW ≥ 0.0.

IFAIL $= 61$

On entry, YSW $= \langle value \rangle$.
Constraint: YSW ≥ 0.0.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example illustrates the use of G02BZF by dividing a dataset into three blocks of 4, 5 and 3 observations respectively. Each block of data is summarised using G02BUF and then the three summaries combined using G02BZF.

The resulting sums of squares and cross-products matrix is then scaled to obtain the covariance matrix for the whole dataset.

9.1 Program Text

```
    Program g02bzfe
!     G02BZFE Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: dscal, g02buf, g02bzf, nag_wp, x04ccf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
```

```
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: alpha, xsw, ysw
      Integer                          :: b, i, ierr, ifail, lc, ldx, m, n
      Character (1)                    :: mean, weight
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: wt(:), x(:,:), xc(:), xmean(:),      &
                                          yc(:), ymean(:)
!     .. Executable Statements ..
      Write (nout,*) 'G02BZF Example Program Results'
      Write (nout,*)

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

!     Read in the problem defining variables
      Read (nin,*) mean, m

!     Allocate memory for output arrays
      lc = (m*m+m)/2
      Allocate (xmean(m),ymean(m),xc(lc),yc(lc))

!     Loop over each block of data
      b = 0
      Do
!       Read in the number of observations in this block and the weight flag
        Read (nin,*,Iostat=ierr) n, weight
        If (ierr/=0) Exit

!       Keep a running total of the number of blocks of data
        b = b + 1

!       Allocate arrays to hold data and read the current block of data in
        ldx = n
        Allocate (x(ldx,m))
        If (weight=='W' .Or. weight=='w') Then
!         Weighted
          Allocate (wt(n))
          Do i = 1, n
            Read (nin,*) x(i,1:m), wt(i)
          End Do
        Else
!         Unweighted
          Allocate (wt(0))
          Do i = 1, n
            Read (nin,*) x(i,1:m)
          End Do
        End If

!       Summarise this block of data
        If (b==1) Then
!         This is the first block of data, so summarise the data into XMEAN
!         and XC
          ifail = 0
          Call g02buf(mean,weight,n,m,x,ldx,wt,xsw,xmean,xc,ifail)

        Else
!         This is not the first block of data, so summarise the data into
!         YMEAN and YC
          ifail = 0
          Call g02buf(mean,weight,n,m,x,ldx,wt,ysw,ymean,yc,ifail)

!         Update the running summaries
          ifail = 0
          Call g02bzf(mean,m,xsw,xmean,xc,ysw,ymean,yc,ifail)
        End If

        Deallocate (x,wt)
      End Do

!     Display results
```

```
      Write (nout,*) 'Means'
      Write (nout,99999) xmean(1:m)

      Write (nout,*)
      ifail = 0
      Call x04ccf('Upper','Non-unit',m,xc,'Sums of squares and cross-products' &
        ,ifail)

      If (xsw>1.0_nag_wp .And. (mean=='M' .Or. mean=='m')) Then
!       Use DSCAL (F06EDF) to scale the sums of squares and cross-products
!       matrix XC, and so convert it to a covariance matrix
        alpha = 1.0_nag_wp/(xsw-1.0_nag_wp)
        Call dscal(lc,alpha,xc,1)

        Write (nout,*)
        ifail = 0
        Call x04ccf('Upper','Non-unit',m,xc,'Covariance matrix',ifail)
      End If

99999 Format (1X,6F14.4)
    End Program g02bzfe
```

9.2 Program Data

```
G02BZF Example Program Data
M  5                                       :: MEAN,M
4 U                                        :: N,WEIGHT (1st block)
-1.10  4.06  -0.95  8.53 10.41
 1.63 -3.22  -1.15 -1.30  3.78
-2.23 -8.19  -3.50  4.31 -1.11
 0.92  0.33  -1.60  5.80 -1.15             :: End of X for 1st block
5 W                                        :: N,WEIGHT (2nd block)
 2.12  5.00 -11.69 -1.22  2.86 2.00
 4.82 -7.23  -4.67  0.83  3.46 0.89
-0.51 -1.12  -1.76  1.45  0.26 0.32
-4.32  4.89   1.34 -1.12 -2.49 4.19
 0.02 -0.74   0.94 -0.99 -2.61 4.33        :: End of X,WT for 2nd block
3 U                                        :: N,WEIGHT (3rd block)
 1.37  0.00  -0.53 -7.98  3.32
 4.15 -2.81  -4.09 -7.96 -2.13
13.09 -1.43   5.16 -1.83  1.58             :: End of X for 3rd block
```

9.3 Program Results

```
 G02BZF Example Program Results

 Means
        0.4369        0.4929       -1.3387       -0.5684        0.0987

 Sums of squares and cross-products
             1          2          3          4          5
 1    304.5052  -123.7700   -27.1830   -60.7092    83.4830
 2               298.9148   -17.3196    -2.1710     5.2072
 3                          332.1639    -3.9445   -96.9299
 4                                     264.7684    79.6211
 5                                                225.5948

 Covariance matrix
             1          2          3          4          5
 1     17.1746    -6.9808    -1.5332    -3.4241     4.7086
 2                16.8593    -0.9769    -0.1224     0.2937
 3                           18.7346    -0.2225    -5.4670
 4                                      14.9334     4.4908
 5                                                 12.7239
```