NAG Library Function Document

nag_sum_sqs_update (g02btc)

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

```
#include <nag.h>
#include <nagg02.h>
```

```
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 function 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 Arguments

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.

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

mean $=$ Nag_AboutZero
The sums of squares and cross-products are calculated.

Constraint: **mean** $=$ Nag_AboutMean or Nag_AboutZero.

2: **m** – Integer *Input*

On entry: m, the number of variables.

Constraint: **m** ≥ 1.

3: **wt** – double *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**] – const double *Input*

On entry: **x**$[(j-1) \times$ **incx**$]$ 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**.

Constraint: **incx** > 0.

6: **sw** – double * *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**] – double *Input/Output*

On entry: if **sw** $= 0.0$, **xbar** is initialized, otherwise **xbar**$[j-1]$ 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**$[j-1]$ contains the weighted mean of the jth variable, $\bar{x}_j(i)$, for $j = 1, 2, \ldots, m$.

8: **c**[(**m** × **m** + **m**)/**2**] – double *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**$[k \times (k-1)/2 + j - 1]$.

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

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

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

On entry, **m** $= \langle value \rangle$.
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.

NE_REAL

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

NE_SUM_WEIGHT

On entry, $(\mathbf{sw} + \mathbf{wt}) = \langle value \rangle$.
Constraint: $(\mathbf{sw} + \mathbf{wt}) \geq 0.0$.

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

nag_sum_sqs_update (g02btc) may be used to update the results returned by nag_sum_sqs (g02buc).

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 .

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

```
/* nag_sum_sqs_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 <nagf16.h>
```

```
#include <nagg02.h>
#include <nagx04.h>

int main(void)
{
  /* Arrays */
  char          nag_enum_arg[40];
  double        *c = 0, *v = 0, *x = 0, *xbar = 0;
  /* Scalars */
  double        alpha, sw, wt;
  Integer       exit_status, i, j, m, mm, n, nprint, incx;
  Nag_SumSquare mean;
  NagError      fail;

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_sum_sqs_update (g02btc) Example Program Results\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");

  incx = 1;
  while (scanf("%39s %ld%ld%ld%*[^\n]",
               nag_enum_arg, &m, &n, &nprint) != EOF)
    {
      /* nag_enum_name_to_value (x04nac).
       * Converts NAG enum member name to value
       */
      mean = (Nag_SumSquare) nag_enum_name_to_value(nag_enum_arg);
      /* 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)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      sw = 0.0;
      for (i = 1; i <= n; ++i)
        {
          scanf("%lf", &wt);
          for (j = 1; j <= m; ++j)
            scanf("%lf", &x[j - 1]);
          scanf("%*[^\n] ");

          /* Calculate the sums of squares and cross-products matrix */
          /* nag_sum_sqs_update (g02btc).
           * Update a weighted sum of squares matrix with a new
           * observation
           */
          nag_sum_sqs_update(mean, m, wt, x, incx, &sw, xbar, c, &fail);

          if (fail.code != NE_NOERROR)
            {
              printf("Error from nag_sum_sqs_update (g02btc).\n%s\n",
                      fail.message);
              exit_status = 1;
              goto END;
            }

          if (i % nprint == 0 || i == n)
            {
              printf("\n");
              printf("---------------------------------------------\n");
              printf("Observation: %4ld      Weight = %13.4f\n",
                      i, wt);
              printf("\n");
              printf("---------------------------------------------\n");
              printf("\n");

              printf("Means\n");
              for (j = 1; j <= m; ++j)
                printf("%14.4f%s", xbar[j - 1],
                        j%4 == 0 || j == m?"\n":" ");
              printf("\n");

              /* Print the sums of squares and cross products matrix */
              /* 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,
                                      "Sums of squares and cross-products",
                                      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;
                }
              if (sw > 1.0)
                {
                  /* Calculate the variance matrix */
                  alpha = 1.0 / (sw - 1.0);
                  mm = m * (m + 1) / 2;
                  /* v[] = alpha*c[] using
                   * nag_daxpby (f16ecc)
                   * Multiply real vector by scalar, preserving input vector
                   */
                  nag_daxpby(mm, alpha, c, 1, 0.0, v, 1, &fail);

                  /* Print the variance matrix */
                  printf("\n");
                  /* nag_pack_real_mat_print (x04ccc), see above. */
                  fflush(stdout);
                  nag_pack_real_mat_print(Nag_ColMajor, Nag_Upper,
                                          Nag_NonUnitDiag, m, v,
                                          "Variance 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;
                    }
                }
            }
        }

      NAG_FREE(c);
      NAG_FREE(v);
      NAG_FREE(x);
      NAG_FREE(xbar);
    }

 END:
  NAG_FREE(c);
  NAG_FREE(v);
  NAG_FREE(x);
  NAG_FREE(xbar);

  return exit_status;
}
```

10.2 Program Data

```
nag_sum_sqs_update (g02btc) Example Program Data
  Nag_AboutMean 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

```
nag_sum_sqs_update (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
```