

NAG Library Function Document

nag_tabulate_stats (g11bac)

1 Purpose

nag_tabulate_stats (g11bac) computes a table from a set of classification factors using a selected statistic.

2 Specification

```
#include <nag.h>
#include <nagg11.h>

void nag_tabulate_stats (Nag_TableStats stat, Nag_TableUpdate update,
    Nag_Weightstype weight, Integer n, Integer nfac, const Integer sf[],
    const Integer lfac[], const Integer factor[], Integer tdf,
    const double y[], const double wt[], double table[], Integer maxt,
    Integer *ncells, Integer *ndim, Integer idim[], Integer count[],
    double comm_ar[], NagError *fail)
```

3 Description

A dataset may include both classification variables and general variables. The classification variables, known as factors, take a small number of values known as levels. For example, the factor sex would have the levels male and female. These can be coded as 1 and 2 respectively. Given several factors, a multi-way table can be constructed such that each cell of the table represents one level from each factor. For example, the two factors sex and habitat, habitat having three levels: inner-city, suburban and rural, define the 2 by 3 contingency table:

| Sex | Habitat | | |
|--------|------------|----------|-------|
| | Inner-city | Suburban | Rural |
| Male | | | |
| Female | | | |

For each cell statistics can be computed. If a third variable in the dataset was age, then for each cell the average age could be computed:

| Sex | Habitat | | |
|--------|------------|----------|-------|
| | Inner-city | Suburban | Rural |
| Male | 25.5 | 30.3 | 35.6 |
| Female | 23.2 | 29.1 | 30.4 |

That is the average age for all observations for males living in rural areas is 35.6. Other statistics can also be computed: the number of observations, the total, the variance, the largest value and the smallest value.

nag_tabulate_stats (g11bac) computes a table for one of the selected statistics. The factors have to be coded with levels 1,2,... Weights can be used to eliminate values from the calculations, e.g., if they

represent ‘missing values’. There is also the facility to update an existing table with the addition of new observations.

4 References

John J A and Quenouille M H (1977) *Experiments: Design and Analysis* Griffin

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* (3rd Edition) Griffin

West D H D (1979) Updating mean and variance estimates: An improved method *Comm. ACM* **22** 532–555

5 Arguments

1: **stat** – Nag_TableStats *Input*

On entry: indicates which statistic is to be computed for the table cells.

stat = Nag_TableStatsNObs

The number of observations for each cell.

stat = Nag_TableStatsTotal

The total for the variable in **y** for each cell.

stat = Nag_TableStatsAv

The average (mean) for the variable in **y** for each cell.

stat = Nag_TableStatsVar

The variance for the variable in **y** for each cell.

stat = Nag_TableStatsLarge

The largest value for the variable in **y** for each cell.

stat = Nag_TableStatsSmall

The smallest value for the variable in **y** for each cell.

Constraint: **stat** = Nag_TableStatsNObs, Nag_TableStatsTotal, Nag_TableStatsAv, Nag_TableStatsVar, Nag_TableStatsLarge or Nag_TableStatsSmall.

2: **update** – Nag_TableUpdate *Input*

On entry: indicates if an existing table is to be updated by further observation.

update = Nag_TableUpdateI

The table cells will be initialized to zero before tabulations take place.

update = Nag_TableUpdateU

The table input in **table** will be updated. The arguments **ncells**, **table**, **count** and **comm_ar** must remain unchanged from the previous call to nag_tabulate_stats (g11bac).

Constraint: **update** = Nag_TableUpdateI or Nag_TableUpdateU.

3: **weight** – Nag_Weightstype *Input*

On entry: indicates if weights are to be used.

weight = Nag_NoWeights

Weights are not used and unit weights are assumed.

weight = Nag_Weights or Nag_Weightsvar

Weights are used and must be supplied in **wt**. The only difference between **weight** = Nag_Weights and **weight** = Nag_Weightsvar is if the variance is computed.

weight = Nag_Weights

The divisor for the variance is the sum of the weights minus one and if **weight** = Nag_Weightsvar, the divisor is the number of observations with nonzero weights

minus one. The former is useful if the weights represent the frequency of the observed values.

If **stat** = Nag_TableStatsTotal or Nag_TableStatsAv, the weighted total or mean is computed respectively.

If **stat** = Nag_TableStatsNObs, Nag_TableStatsLarge or Nag_TableStatsSmall the only effect of weights is to eliminate values with zero weights from the computations.

Constraint: **weight** = Nag_NoWeights, Nag_Weightsvar or Nag_Weights.

- 4: **n** – Integer *Input*
On entry: the number of observations.
Constraint: $n \geq 2$.

- 5: **nfac** – Integer *Input*
On entry: the number of classifying factors in **factor**.
Constraint: $nfac \geq 1$.

- 6: **sf[nfac]** – const Integer *Input*
On entry: indicates which factors in **factor** are to be used in the tabulation.
If $sf[i - 1] > 0$ the i th factor in **factor** is included in the tabulation.
Note that if $sf[i - 1] \leq 0$ for $i = 1, 2, \dots, nfac$ then the statistic for the whole sample is calculated and returned in a 1 by 1 table.

- 7: **lfac[nfac]** – const Integer *Input*
On entry: the number of levels of the classifying factors in **factor**.
Constraint: if $sf[i - 1] > 0$, $lfac[i - 1] \geq 2$, for $i = 1, 2, \dots, nfac$.

- 8: **factor[n × tdf]** – const Integer *Input*
On entry: the **nfac** coded classification factors for the **n** observations.
Constraint: $1 \leq \text{factor}[(i - 1) \times \text{tdf} + j - 1] \leq \text{lfac}[j - 1]$, for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, nfac$.

- 9: **tdf** – Integer *Input*
On entry: the stride separating matrix column elements in the array **factor**.
Constraint: $tdf \geq nfac$.

- 10: **y[n]** – const double *Input*
On entry: the variable to be tabulated.
If **stat** = Nag_TableStatsNObs, **y** is not referenced.

- 11: **wt[n]** – const double *Input*
On entry: if **weight** = Nag_Weights or Nag_Weightsvar, **wt** must contain the **n** weights. Otherwise **wt** is not referenced and can be set to null, (double *)0.
Constraint: if **weight** = Nag_Weights or Nag_Weightsvar, $wt[i - 1] \geq 0.0$, for $i = 1, 2, \dots, n$.

- 12: **table[maxt]** – double *Input/Output*
On entry: if **update** = Nag_TableUpdateU, **table** must be unchanged from the previous call to nag_tabulate_stats (g11bac), otherwise **table** need not be set.

On exit: the computed table. The **ncells** cells of the table are stored so that for any two factors the index relating to the factor referred to later in **lfac** and **factor** changes faster. For further details see Section 9.

13: **maxt** – Integer *Input*

On entry: the maximum size of the table to be computed.

Constraint: **maxt** \geq product of the levels of the factors included in the tabulation.

14: **ncells** – Integer * *Input/Output*

On entry: if **update** = Nag_TableUpdateU, **ncells** must be unchanged from the previous call to nag_tabulate_stats (g11bac), otherwise **ncells** need not be set.

On exit: the number of cells in the table.

15: **ndim** – Integer * *Output*

On exit: the number of factors defining the table.

16: **idim[nfac]** – Integer *Output*

On exit: the first **ndim** elements contain the number of levels for the factors defining the table.

17: **count[maxt]** – Integer *Input/Output*

On entry: if **update** = Nag_TableUpdateU, **count** must be unchanged from the previous call to nag_tabulate_stats (g11bac), otherwise **count** need not be set.

On exit: a table containing the number of observations contributing to each cell of the table, stored identically to **table**. Note if **stat** = Nag_TableStatsNObs this is the same as is returned in **table**.

18: **comm_ar[*]** – double *Input/Output*

On entry: if **update** = Nag_TableUpdateU, **comm_ar** must be unchanged from the previous call to nag_tabulate_stats (g11bac), otherwise **comm_ar** need not be set.

On exit: if **stat** = Nag_TableStatsAv or Nag_TableStatsVar, the first **ncells** values hold the table containing the sum of the weights for the observations contributing to each cell, stored identically to **table**. If **stat** = Nag_TableStatsVar, then the second set of **ncells** values hold the table of cell means. Otherwise **comm_ar** is not referenced.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **tdf** = $\langle value \rangle$ while **nfac** = $\langle value \rangle$. These arguments must satisfy **tdf** \geq **nfac**.

NE_2_INT_ARRAY_CONS

On entry, **sf**[$\langle value \rangle$] = $\langle value \rangle$ while **lfac**[0] = $\langle value \rangle$.

Constraint: if **sf**[i] > 0 , **lfac**[i] ≥ 2 for $i = 0, 1, \dots, \mathbf{nfac}$.

NE_2D_1D_INT_ARRAYS_CONS

On entry, **factor**[($\langle value \rangle$) \times **tdf** + $\langle value \rangle$] = $\langle value \rangle$ while **lfac**[0] = $\langle value \rangle$.

Constraint: **factor**[(i) \times **tdf** + j] \leq **lfac**[j], for $i = 0, 1, \dots, \mathbf{n} - 1$ and $j = 0, 1, \dots, \mathbf{nfac} - 1$.

NE_2D_INT_ARRAY_CONS

On entry, **factor** $[(\langle value \rangle) \times \mathbf{tdf} + \langle value \rangle] = \langle value \rangle$.

Constraint: **factor** $[(i) \times \mathbf{tdf} + j] \geq 1$, for $i = 0, 1, \dots, \mathbf{n} - 1$ and $j = 0, 1, \dots, \mathbf{nfac} - 1$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument **stat** had an illegal value.

On entry, argument **update** had an illegal value.

On entry, argument **weight** had an illegal value.

NE_G11BA_CHANGED

update = Nag_TableUpdateU and at least one of **ncells**, **table**, **comm_ar** or **count** have been changed since previous call to nag_tabulate_stats (g11bac).

NE_INT_ARG_LT

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

Constraint: **n** ≥ 2 .

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

Constraint: **nfac** ≥ 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_MAXT

The maximum size of the table to be computed, **maxt** is too small.

NE_REAL_ARRAY_CONS

On entry, **wt** $[\langle value \rangle] = \langle value \rangle$.

Constraint: if **weight** = Nag_Weights or Nag_Weightsvar, **wt** $[i] \geq 0.0$.

NE_VAR_DIV

stat = Nag_TableStatsVar and the divisor for the variance ≤ 0.0 .

NE_WT_ARGS

The **wt** array argument must not be **NULL** when the **weight** argument indicates weights.

7 Accuracy

Only applicable when **stat** = Nag_TableStatsVar. In this case a one pass algorithm is used as described by West (1979).

8 Parallelism and Performance

nag_tabulate_stats (g11bac) is not threaded in any implementation.

9 Further Comments

The tables created by `nag_tabulate_stats` (`g1lbac`) and stored in **table**, **count** and, depending on **stat**, also in **comm_ar** are stored in the following way. Let there be n factors defining the table with factor k having l_k levels, then the cell defined by the levels i_1, i_2, \dots, i_n of the factors is stored in m th cell given by:

$$m = 1 + \sum_{k=1}^n \{(i_k - 1)c_k\},$$

where $c_j = \prod_{k=j+1}^n l_k$, for $j = 1, 2, \dots, n-1$ and $c_n = 1$.

10 Example

The data, given by John and Quenouille (1977), is for a 3 by 6 factorial experiment in 3 blocks of 18 units. The data is input in the order: blocks, factor with 3 levels, factor with 6 levels, yield. The 3 by 6 table of treatment means for yield over blocks is computed and printed.

10.1 Program Text

```
/* nag_tabulate_stats (g1lbac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg11.h>

int main(void)
{
    Integer exit_status = 0, i, items, j, k, ltmax, maxt, n, ncells;
    Integer ncol, ndim, nfac, nrow, tdf;
    Integer *count = 0, *factor = 0, *idim = 0, *isf = 0, *lfac = 0;
    double *comm_ar = 0, *table = 0, *wt = 0, *y = 0;
    char nag_enum_arg[40];
    Nag_TableStats stat;
    Nag_Weightstype weight;
    NagError fail;

#define FACTOR(I, J) factor[((I) - 1)*nfac + (J) - 1]

    INIT_FAIL(fail);

    printf("nag_tabulate_stats (g1lbac) Example Program Results\n");

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s", nag_enum_arg);
#endif
    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    stat = (Nag_TableStats) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
```

```

    scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s", nag_enum_arg);
#endif
    weight = (Nag_Weightstype) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " %" NAG_IFMT " ", &n, &nfac);
#else
    scanf("%" NAG_IFMT " %" NAG_IFMT " ", &n, &nfac);
#endif

    ltmax = 18;
    maxt = ltmax;
    if (!(isf = NAG_ALLOC(nfac, Integer))
        || !(lfac = NAG_ALLOC(nfac, Integer))
        || !(idim = NAG_ALLOC(nfac, Integer))
        || !(factor = NAG_ALLOC(n * nfac, Integer))
        || !(count = NAG_ALLOC(maxt, Integer))
        || !(y = NAG_ALLOC(n, double))
        || !(wt = NAG_ALLOC(n, double))
        || !(table = NAG_ALLOC(maxt, double))
        || !(comm_ar = NAG_ALLOC(2 * maxt, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    if (weight == Nag_Weights || weight == Nag_Weightsvar) {
        for (i = 1; i <= n; ++i) {
            for (j = 1; j <= nfac; ++j)
#ifdef _WIN32
                scanf_s("%" NAG_IFMT "", &FACTOR(i, j));
#else
                scanf("%" NAG_IFMT "", &FACTOR(i, j));
#endif
#ifdef _WIN32
                scanf_s("%lf %lf", &y[i - 1], &wt[i - 1]);
#else
                scanf("%lf %lf", &y[i - 1], &wt[i - 1]);
#endif
            }
        }
    }
    else {
        for (i = 1; i <= n; ++i) {
            for (j = 1; j <= nfac; ++j)
#ifdef _WIN32
                scanf_s("%" NAG_IFMT "", &FACTOR(i, j));
#else
                scanf("%" NAG_IFMT "", &FACTOR(i, j));
#endif
#ifdef _WIN32
                scanf_s("%lf", &y[i - 1]);
#else
                scanf("%lf", &y[i - 1]);
#endif
            }
        }
        for (j = 1; j <= nfac; ++j)
#ifdef _WIN32
            scanf_s("%" NAG_IFMT "", &lfac[j - 1]);
#else
            scanf("%" NAG_IFMT "", &lfac[j - 1]);
#endif
        for (j = 1; j <= nfac; ++j)
#ifdef _WIN32
            scanf_s("%" NAG_IFMT "", &isf[j - 1]);
#else
            scanf("%" NAG_IFMT "", &isf[j - 1]);
#endif
        tdf = 3;
    }

```

```

maxt = ltmax;

/* nag_tabulate_stats (g1lbac).
 * Computes multiway table from set of classification
 * factors using selected statistic
 */
nag_tabulate_stats(stat, Nag_TableUpdateI, weight, n, nfac, isf,
                  lfac, factor, tdf, y, wt, table, maxt, &ncells, &ndim,
                  idim, count, comm_ar, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_tabulate_stats (g1lbac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
printf("%s\n", "Table");
printf("\n");
ncol = idim[ndim - 1];
nrow = ncells / ncol;
k = 1;
for (i = 1; i <= nrow; ++i) {
    for (j = k, items = 1; j <= k + ncol - 1; ++j, items++)
        printf("%8.2f(%2" NAG_IFMT ")%s", table[j - 1],
                count[j - 1], items % 6 ? "" : "\n");
    k += ncol;
}

END:
NAG_FREE(isf);
NAG_FREE(lfac);
NAG_FREE(idim);
NAG_FREE(factor);
NAG_FREE(count);
NAG_FREE(y);
NAG_FREE(wt);
NAG_FREE(table);
NAG_FREE(comm_ar);

return exit_status;
}

```

10.2 Program Data

nag_tabulate_stats (g1lbac) Example Program Data

Nag_TableStatsAv Nag_NoWeights 54 3

```

1 1 1 274
1 2 1 361
1 3 1 253
1 1 2 325
1 2 2 317
1 3 2 339
1 1 3 326
1 2 3 402
1 3 3 336
1 1 4 379
1 2 4 345
1 3 4 361
1 1 5 352
1 2 5 334
1 3 5 318
1 1 6 339
1 2 6 393
1 3 6 358
2 1 1 350
2 2 1 340
2 3 1 203
2 1 2 397
2 2 2 356

```



```

2 3 2 298
2 1 3 382
2 2 3 376
2 3 3 355
2 1 4 418
2 2 4 387
2 3 4 379
2 1 5 432
2 2 5 339
2 3 5 293
2 1 6 322
2 2 6 417
2 3 6 342
3 1 1 82
3 2 1 297
3 3 1 133
3 1 2 306
3 2 2 352
3 3 2 361
3 1 3 220
3 2 3 333
3 3 3 270
3 1 4 388
3 2 4 379
3 3 4 274
3 1 5 336
3 2 5 307
3 3 5 266
3 1 6 389
3 2 6 333
3 3 6 353

3 3 6
0 1 1

```

10.3 Program Results

nag_tabulate_stats (g11bac) Example Program Results

Table

| | | | | | |
|------------|------------|------------|------------|------------|------------|
| 235.33(3) | 342.67(3) | 309.33(3) | 395.00(3) | 373.33(3) | 350.00(3) |
| 332.67(3) | 341.67(3) | 370.33(3) | 370.33(3) | 326.67(3) | 381.00(3) |
| 196.33(3) | 332.67(3) | 320.33(3) | 338.00(3) | 292.33(3) | 351.00(3) |
