NAG C 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 data set 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 data set 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 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 Parameters

1: **stat** – Nag_TableStats *Input*

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

If **stat**`[]` = **Nag_TableStatsNObs**, the number of observations for each cell.

If **stat**[] = **Nag_TableStatsTotal**, the total for the variable in **y**[] for each cell.
If **stat**[] = **Nag_TableStatsAv**, the average (mean) for the variable in **y**[] for each cell.
If **stat**[] = **Nag_TableStatsVar**, the variance for the variable in **y**[] for each cell.
If **stat**[] = **Nag_TableStatsLarge**, the largest value for the variable in **y**[] for each cell.
If **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.
If **update**[] = **Nag_TableUpdateI**, the table cells will be initialised to zero before tabulations take place.
If **update**[] = **Nag_TableUpdateU**, the table input in **table**[] will be updated. The parameters
ncells[], **table**[], **count**[] and **comm_ar**[] must remain unchanged from the previous call to nag_tabulate_stats.

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

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

On entry: indicates if weights are to be used.
If **weight**[] = **Nag_NoWeights**, weights are not used and unit weights are assumed.
If **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.
If **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 non-zero 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**[] ≥ 2.

5: **nfac** – Integer *Input*

On entry: the number of classifying factors in **factor**[].

Constraint: **nfac**[] ≥ 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 ith factor in **factor**[] is included in the tabulation.

Note that if **sf**[]$[i-1] \leq 0$ for $i = 1, 2, \ldots,$**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 $i = 1, 2, \ldots,$**nfac**[].

8: **factor[n][tdf]** – const Integer *Input*

On entry: the **nfac**[] coded classification factors for the **n**[] observations.

Constraint: $1 \leq$ **factor**[]$[i-1][j-1] \leq$ **lfac**[]$[j-1]$ for $i = 1, 2, \ldots,$**n**[]; $j = 1, 2, \ldots,$**nfac**[].

9: **tdf** – Integer *Input*

On entry: the second dimension of the array **factor**[]# as declared in the function from which nag_tabulate_stats is called.

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 $i = 1, 2, \ldots,$**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, 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 6.

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, 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, 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[*dim1*]** – double *Input/Output*

Note: the dimension, *dim1*, of the array **comm_ar**[] must be at least **ncells**[] if **stat**[] = **Nag_TableStatsAv**, at least 2×**ncells**[] if **stat**[] = **Nag_TableStatsVar**. **comm_ar**[] can be set to null, (double*) 0 otherwise.

On entry: if **update**[] = **Nag_TableUpdateU**, **comm_ar**[] must be unchanged from the previous call to nag_tabulate_stats, 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 parameter (see the Essential Introduction).

5 Error Indicators and Warnings

NE_INT_ARG_LT

On entry, **n**[] must not be less than 2: **n**[] = <*value*>.
On entry, **nfac**[] must not be less than 1: **nfac**[] = <*value*>.

NE_2_INT_ARG_LT

On entry, **tdf**[] = <*value*> while **nfac**[] = <*value*>.
These parameters must satisfy **tdf**[] ≥ **nfac**[].

NE_BAD_PARAM

On entry, parameter **stat**[] had an illegal value.
On entry, parameter **weight**[] had an illegal value.
On entry, parameter **update**[] had an illegal value.

NE_WT_ARGS

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

NE_REAL_ARRAY_CONS

On entry, **wt**[][<*value*>] = <*value*>.
Constraint: if **weight**[] = **Nag_Weights** or **Nag_Weightsvar**, **wt**[][i] $\geq$ 0.0.

NE_2_INT_ARRAY_CONS

On entry, **sf**[][<*value*>] = <*value*> while **lfac**[][0] = <*value*>.
Constraint: if **sf**[][i] $>$ 0, **lfac**[][i] $\geq$ 2 for $i = 0, 1, \ldots,$**nfac**[].

NE_2D_INT_ARRAY_CONS

On entry, **factor**[][<*value*>][<*value*>] = <*value*>.
Constraint: **factor**[][i][j] $\geq$ 1 for $i = 0, 1, \ldots,$**n**[]−1; $j = 0, 1, \ldots,$**nfac**[]−1.

NE_2D_1D_INT_ARRAYS_CONS

On entry, **factor**[][<*value*>][<*value*>] = <*value*> while **lfac**[][0] = <*value*>.
Constraint: **factor**[][i][j] $\leq$ **lfac**[][j] for $i = 0, 1, \ldots,$**n**[]−1; $j = 0, 1, \ldots,$**nfac**[]−1.

NE_MAXT

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

NE_VAR_DIV

stat`[]` = **Nag_TableStatsVar** and the divisor for the variance $\leq$ 0.0.

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.

NE_ALLOC_FAIL

Memory allocation failed.

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.

6 Further Comments

The tables created by nag_tabulate_stats 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, $i_2, \ldots, i_n$ of the factors is stored in mth 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 $j = 1, 2, \ldots, n-1$ and $c_n = 1$.

6.1 Accuracy

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

6.2 References

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

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)* Griffin (3rd Edition)

7 See Also

None.

8 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.

8.1 Program Text

```
/* nag_tabulate_stats (g11bac) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 */
```

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

int main (void)
{
  char stat[2], weight[2];
  double *comm_ar=0, *table=0, *wt=0, *y=0;
  Integer items, i, *count=0, *idim=0, *factor=0, *isf=0;
  Integer j, k, tdf, *lfac=0, ltmax, maxt, n, ncells, ncol, ndim, nfac;
  Integer nrow;
  Integer exit_status=0;
  Nag_TableStats stat_enum;
  Nag_Weightstype weight_enum;
  NagError fail;

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

  INIT_FAIL(fail);
  Vprintf("g11bac Example Program Results\n");

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

  Vscanf(" %s %s %ld %ld ", stat, weight, &n, &nfac);
  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)))
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

      if (*weight == 'W' || *weight == 'V')
        {
          for (i = 1; i <= n; ++i)
            {
              for (j = 1; j <= nfac; ++j)
                  Vscanf("%ld", &FACTOR(i,j));
              Vscanf("%lf %lf", &y[i - 1], &wt[i - 1]);
            }
        }
      else
        {
          for (i = 1; i <= n; ++i)
            {
              for (j = 1; j <= nfac; ++j)
                  Vscanf("%ld", &FACTOR(i,j));
```

```
          Vscanf("%lf", &y[i - 1]);
        }
    }
  for (j = 1; j <= nfac; ++j)
      Vscanf("%ld", &lfac[j - 1]);
  for (j = 1; j <= nfac; ++j)
      Vscanf("%ld", &isf[j - 1]);
  tdf = 3;
  maxt = ltmax;
  if (*stat == 'N')
      stat_enum = Nag_TableStatsNObs;
  else if (*stat == 'T')
      stat_enum = Nag_TableStatsTotal;
  else if (*stat == 'A')
      stat_enum = Nag_TableStatsAv;
  else if (*stat == 'V')
      stat_enum = Nag_TableStatsVar;
  else if (*stat == 'L')
      stat_enum = Nag_TableStatsLarge;
  else if (*stat == 'S')
      stat_enum = Nag_TableStatsSmall;
  else
      stat_enum = (Nag_TableStats)-999;


  if (*weight == 'U')
      weight_enum = Nag_NoWeights;
  else if (*weight == 'W')
      weight_enum = Nag_Weights;
  else if (*weight == 'V')
      weight_enum = Nag_Weightsvar;
  else
      weight_enum = (Nag_Weightstype)-999;




  g11bac(stat_enum, Nag_TableUpdateI, weight_enum, n, nfac, isf, lfac,
         factor, tdf, y, wt,
           table, maxt, &ncells, &ndim, idim, count, comm_ar, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g11bac.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  Vprintf("\n");
  Vprintf("%s\n", " Table");
  Vprintf("\n");
  ncol = idim[ndim - 1];
  nrow = ncells / ncol;
  k = 1;
  items = 0;
  for (i = 1; i <= nrow; ++i)
    {
      for (j = k, items =1; j <= k + ncol - 1; ++j, items++)
          Vprintf("%8.2f(%2ld)%s", table[j - 1],
                 count[j - 1], items%6?"":"\n");
      k += ncol;
```

```
        }
 END:
      if (isf) NAG_FREE(isf);
      if (lfac) NAG_FREE(lfac);
      if (idim) NAG_FREE(idim);
      if (factor) NAG_FREE(factor);
      if (count) NAG_FREE(count);
      if (y) NAG_FREE(y);
      if (wt) NAG_FREE(wt);
      if (table) NAG_FREE(table);
      if (comm_ar) NAG_FREE(comm_ar);
      return exit_status;
}
```

8.2 Program Data

```
g11bac Example Program Data

A  U  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
```

8.3 Program Results

```
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)
```