NAG Library Function Document

nag_tabulate_percentile (g11bbc)

1 Purpose

nag_tabulate_percentile (g11bbc) computes a table from a set of classification factors using a given percentile or quantile, for example the median.

2 Specification

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

```
void nag_tabulate_percentile (Nag_TabulateVar type, Integer n, Integer nfac,
     const Integer sf[], const Integer lfac[], const Integer factor[],
     Integer tdf, double percnt, const double y[], const double wt[],
     double table[], Integer maxt, Integer *ncells, Integer *ndim,
     Integer idim[], Integer count[], 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 median age could be computed:

Sex	**Habitat**		
	Inner-city	Suburban	Rural
Male	24	31	37
Female	21.5	28.5	33

That is the median age for all observations for males living in rural areas is 37. The median being the 50% quantile. Other quantiles can also be computed: the p percent quantile or percentile, q_p, is the estimate of the value such that p percent of observations are less than q_p. This is calculated in two different ways depending on whether the tabulated variable is continuous or discrete. Let there be m values in a cell and let $y_{(1)}, y_{(2)}, \ldots, y_{(m)}$ be the values for that cell sorted into ascending order. Also, associated with each value there is a weight, $w_{(1)}, w_{(2)}, \ldots, w_{(m)}$, which could represent the observed frequency for that value, with $W_j = \sum_{i=1}^{j} w_{(i)}$ and $W_j' = \sum_{i=1}^{j} w_{(i)} - \frac{1}{2} w_{(j)}$. For the p percentile let $p_w = (p/100)W_m$ and $p_w' = (p/100)W_m'$ then the percentiles for the two cases are as given below.

If the variable is discrete, that is takes only a limited number of (usually integer) values then the percentile is defined as:

$$\begin{array}{ll} y_{(j)} & \text{if } W_{j-1} < p_W < W_j \\ \frac{y_{(j+1)}+y_{(j)}}{2} & \text{if } p_w = W_j \end{array}$$

If the data is continuous then the quantiles are estimated by linear interpolation.

$$\begin{array}{ll} y_{(1)} & \text{if } p'_w \leq W'_1 \\ (1-f)y_{(j-1)} + fy_{(j)} & \text{if } W'_{j-1} < p'_w \leq W'_j \\ y_{(m)} & \text{if } p'_w > W'_m \end{array}$$

where $f = \left(p'_w - W'_{j-1}\right) / \left(W'_j - W'_{j-1}\right)$.

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

5 Arguments

1: **type** – Nag_TabulateVar *Input*

On entry: indicates whether the variable to be tabulated is discrete or continuous.

type = Nag_TabulateVarDiscr
The percentiles are computed for a discrete variable.

type = Nag_TabulateVarCont
The percentiles are computed for a continuous variable using linear interpolation.

Constraint: **type** = Nag_TabulateVarDiscr or Nag_TabulateVarCont.

2: **n** – Integer *Input*

On entry: the number of observations.

Constraint: $\mathbf{n} \geq 2$.

3: **nfac** – Integer *Input*

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

Constraint: $\mathbf{nfac} \geq 1$.

4: **sf**[**nfac**] – const Integer *Input*

On entry: indicates which factors in **factor** are to be used in the tabulation.

If $\mathbf{sf}[i-1] > 0$ the ith factor in **factor** is included in the tabulation.

Note that if $\mathbf{sf}[i-1] \leq 0$, for $i = 1, 2, \ldots, \mathbf{nfac}$ then the statistic for the whole sample is calculated and returned in a 1 by 1 table.

5: **lfac**[**nfac**] – const Integer *Input*

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

Constraint: if $\mathbf{sf}[i-1] > 0$, $\mathbf{lfac}[i-1] \geq 2$, for $i = 1, 2, \ldots, \mathbf{nfac}$.

6: **factor**[**n** × **tdf**] – const Integer *Input*

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

Constraint: if $\mathbf{sf}[i-1] > 0$, $1 \leq \mathbf{factor}[(i-1) \times \mathbf{tdf} + j - 1] \leq \mathbf{lfac}[j-1]$, for $i = 1, 2, \ldots, \mathbf{n}$ and $j = 1, 2, \ldots, \mathbf{nfac}$.

7: **tdf** – Integer *Input*

On entry: the stride separating matrix column elements in the array **factor**.

Constraint: **tdf** $\geq$ **nfac**.

8: **percnt** – double *Input*

On entry: the percentile to be tabulated, p.

Constraint: $0.0 <$ **percnt** < 100.0.

9: **y**[**n**] – const double *Input*

On entry: the variable to be tabulated.

10: **wt**[**n**] – const double *Input*

On entry: **wt** must contain the **n** weights. Otherwise **wt** must be set to **NULL**.

Constraint: **wt**$[i-1] \geq 0.0$, for $i = 1, 2, \ldots,$ **n**.

11: **table**[**maxt**] – double *Output*

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 occurring later in **lfac** and **factor** changes faster. For further details see Section 9.

12: **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.

13: **ncells** – Integer * *Output*

On exit: the number of cells in the table.

14: **ndim** – Integer * *Output*

On exit: the number of factors defining the table.

15: **idim**[**nfac**] – Integer *Output*

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

16: **count**[**maxt**] – Integer *Output*

On exit: a table containing the number of observations contributing to each cell of the table, stored identically to **table**.

17: **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** = ⟨*value*⟩ while **nfac** = ⟨*value*⟩. These arguments must satisfy **tdf** $\geq$ **nfac**.

NE_2_INT_ARRAY_CONS

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

NE_2D_1D_INT_ARRAYS_CONS

On entry, **factor**$[(\langle value\rangle) \times$ **tdf** $+ \langle value\rangle] = \langle value\rangle$ while **lfac**$[\langle value\rangle] = \langle value\rangle$.
Constraint: **factor**$[(i) \times$ **tdf** $+ j] \leq$ **lfac**$[j]$, for $i = 0, 1, \ldots, n - 1$ and $j = 0, 1, \ldots,$ **nfac** $- 1$.

NE_2D_INT_ARRAY_CONS

On entry, **factor**$[(\langle value\rangle) \times$ **tdf** $+ \langle value\rangle] = \langle value\rangle$.
Constraint: **factor**$[(i) \times$ **tdf** $+ j] \geq 1$, for $i = 0, 1, \ldots, n - 1$ and $j = 0, 1, \ldots,$ **nfac** $- 1$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_CELL_EMPTY

At least one cell is empty.

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

On entry, **percnt** $= \langle value\rangle$.
Constraint: $0.0 <$ **percnt** < 100.0.

NE_REAL_ARRAY_CONS

On entry, **wt**$[\langle value\rangle] = \langle value\rangle$.
Constraint: **wt**$[i] \geq 0$, for $i = 0, 1, \ldots, n - 1$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_tabulate_percentile (g11bbc) is not threaded in any implementation.

9 Further Comments

The tables created by nag_tabulate_percentile (g11bbc) and stored in **table** and **count** 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 $c_n = 1$.

10 Example

The data, given by John and Quenouille (1977), are 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, and the 3 by 6 table of treatment medians for yield over blocks is computed and printed.

10.1 Program Text

```
/* nag_tabulate_percentile (g11bbc) 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, *lfac = 0, *sf = 0;
  double percnt, *table = 0, *wt = 0, *wtptr, *y = 0;
  char nag_enum_arg[40];
  Nag_TabulateVar type;
  Nag_Weightstype weight;
  NagError fail;

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

  INIT_FAIL(fail);

  printf("nag_tabulate_percentile (g11bbc) 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
   */
  type = (Nag_TabulateVar) 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 " %lf", &n, &nfac, &percnt);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT " %lf", &n, &nfac, &percnt);
#endif
  ltmax = 18;
  maxt = ltmax;
  if (!(sf = 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))
      || !(table = NAG_ALLOC(maxt, double))
      || !(wt = NAG_ALLOC(n, 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
    }
    wtptr = wt;
  }
  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
    }
    wtptr = 0;
  }
  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 "", &sf[j - 1]);
#else
    scanf("%" NAG_IFMT "", &sf[j - 1]);
#endif
  tdf = nfac;
  /* nag_tabulate_percentile (g11bbc).
   * Computes multiway table from set of classification
   * factors using given percentile/quantile
   */
  nag_tabulate_percentile(type, n, nfac, sf, lfac, factor, tdf, percnt, y,
                          wtptr, table, maxt, &ncells, &ndim, idim, count,
                          &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_tabulate_percentile (g11bbc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }
  printf("\n");
  printf("%s%4.0f%s\n", "Table for ", percnt, "th percentile");
  printf("\n");
  ncol = idim[ndim - 1];
  nrow = ncells / ncol;
  k = 1;
  for (i = 1; i <= nrow; ++i) {
    for (items = 1, j = k; 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(sf);
  NAG_FREE(lfac);
  NAG_FREE(idim);
  NAG_FREE(factor);
  NAG_FREE(count);
  NAG_FREE(y);
  NAG_FREE(table);
  NAG_FREE(wt);
  return exit_status;
}
```

10.2 Program Data

```
nag_tabulate_percentile (g11bbc) Example Program Data

Nag_TabulateVarCont  Nag_NoWeights  54 3 50.0

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_percentile (g11bbc) Example Program Results

Table for   50th percentile

  226.00( 3)  320.25( 3)  299.50( 3)  385.75( 3)  348.00( 3)  334.75( 3)
  329.25( 3)  343.25( 3)  365.25( 3)  370.50( 3)  327.25( 3)  378.00( 3)
  185.50( 3)  328.75( 3)  319.50( 3)  339.25( 3)  286.25( 3)  350.25( 3)
```