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

```c
#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 percent, 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:

<table>
<thead>
<tr>
<th>Sex</th>
<th>Habitat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inner-city</td>
</tr>
<tr>
<td>Male</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Sex</th>
<th>Habitat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inner-city</td>
</tr>
<tr>
<td>Male</td>
<td>24</td>
</tr>
<tr>
<td>Female</td>
<td>21.5</td>
</tr>
</tbody>
</table>

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:

Mark 25
\[
\begin{align*}
y(j) &= \frac{y(j) - y(j+1) + y(j)}{2} & \text{if } W_{j-1} < p_w < W_j \\
y(j) &= \text{if } p_w = W_j
\end{align*}
\]

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

\[
\begin{align*}
y(1) &= (1 - f)y(j-1) + fy(j) & \text{if } p'_w \leq W'_l \\
y(m) &= \text{if } W'_{j-1} < p'_w \leq W'_j \\
y(m) &= \text{if } p'_w > W'_m
\end{align*}
\]

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

5 Arguments
1: \texttt{type} – Nag_TabulateVar \hspace{1cm} \textit{Input}
   \hspace{1cm} \textit{On entry}: indicates whether the variable to be tabulated is discrete or continuous.
   \texttt{type} = Nag_TabulateVarDiscr
   \hspace{1cm} \text{The percentiles are computed for a discrete variable.}
   \hspace{1cm} \texttt{type} = Nag_TabulateVarCont
   \hspace{1cm} \text{The percentiles are computed for a continuous variable using linear interpolation.}
   \hspace{1cm} \textit{Constraint}: \texttt{type} = Nag_TabulateVarDiscr or Nag_TabulateVarCont.

2: \texttt{n} – Integer \hspace{1cm} \textit{Input}
   \hspace{1cm} \textit{On entry}: the number of observations.
   \hspace{1cm} \textit{Constraint}: \texttt{n} \geq 2.

3: \texttt{nfac} – Integer \hspace{1cm} \textit{Input}
   \hspace{1cm} \textit{On entry}: the number of classifying factors in \texttt{factor}.
   \hspace{1cm} \textit{Constraint}: \texttt{nfac} \geq 1.

4: \texttt{sf[nfac]} – const Integer \hspace{1cm} \textit{Input}
   \hspace{1cm} \textit{On entry}: indicates which factors in \texttt{factor} are to be used in the tabulation.
   \hspace{1cm} If \texttt{sf}[i - 1] > 0 \text{ the } i\text{th factor in } \texttt{factor} \text{ is included in the tabulation.}
   \hspace{1cm} \text{Note that if } \texttt{sf}[i - 1] \leq 0, \text{ for } i = 1, 2, \ldots, \texttt{nfac} \text{ then the statistic for the whole sample is calculated and returned in a 1 by 1 table.}

5: \texttt{lfac[nfac]} – const Integer \hspace{1cm} \textit{Input}
   \hspace{1cm} \textit{On entry}: the number of levels of the classifying factors in \texttt{factor}.
   \hspace{1cm} \textit{Constraint}: \texttt{sf}[i - 1] > 0, \texttt{lfac}[i - 1] \geq 2, \text{ for } i = 1, 2, \ldots, \texttt{nfac}.

6: \texttt{factor[n \times tdf]} – const Integer \hspace{1cm} \textit{Input}
   \hspace{1cm} \textit{On entry}: the \texttt{nfac} coded classification factors for the \texttt{n} observations.
   \hspace{1cm} \textit{Constraint}: \texttt{sf}[i - 1] > 0, 1 \leq \texttt{factor}[(i - 1) \times \texttt{tdf} + j - 1] \leq \texttt{lfac}[j - 1], \text{ for } i = 1, 2, \ldots, \texttt{n} \text{ and } j = 1, 2, \ldots, \texttt{nfac}.
7:  \texttt{tdf} – Integer  \hspace{1cm} \textit{Input}
On entry: the stride separating matrix column elements in the array \texttt{factor}.
\textit{Constraint:} \( \texttt{tdf} \geq \texttt{nfac} \).

8:  \texttt{percnt} – double  \hspace{1cm} \textit{Input}
On entry: the percentile to be tabulated, \( p \).
\textit{Constraint:} \( 0.0 < \texttt{percnt} < 100.0 \).

9:  \texttt{y[n]} – const double  \hspace{1cm} \textit{Input}
On entry: the variable to be tabulated.

10:  \texttt{wt[n]} – const double  \hspace{1cm} \textit{Input}
On entry: \texttt{wt} must contain the \( n \) weights. Otherwise \texttt{wt} must be set to NULL.
\textit{Constraint:} \( \texttt{wt[i]} \geq 0.0 \), for \( i = 1, 2, \ldots, n \).

11:  \texttt{table[max]} – double  \hspace{1cm} \textit{Output}
On exit: the computed table. The \texttt{ncells} cells of the table are stored so that for any two factors the index relating to the factor occurring later in \texttt{lfac} and \texttt{factor} changes faster. For further details see Section 9.

12:  \texttt{maxt} – Integer  \hspace{1cm} \textit{Input}
On entry: the maximum size of the table to be computed.
\textit{Constraint:} \( \texttt{maxt} \geq \text{product of the levels of the factors included in the tabulation} \).

13:  \texttt{ncells} – Integer *  \hspace{1cm} \textit{Output}
On exit: the number of cells in the table.

14:  \texttt{ndim} – Integer *  \hspace{1cm} \textit{Output}
On exit: the number of factors defining the table.

15:  \texttt{idim[nfac]} – Integer  \hspace{1cm} \textit{Output}
On exit: the first \texttt{ndim} elements contain the number of levels for the factors defining the table.

16:  \texttt{count[max]} – Integer  \hspace{1cm} \textit{Output}
On exit: a table containing the number of observations contributing to each cell of the table, stored identically to \texttt{table}.

17:  \texttt{fail} – NagError *  \hspace{1cm} \textit{Input/Output}
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \hspace{1cm} \textbf{Error Indicators and Warnings}

\textbf{NE\_2\_INT\_ARG\_LT}
On entry, \( \texttt{tdf} = \langle \text{value} \rangle \) while \( \texttt{nfac} = \langle \text{value} \rangle \). These arguments must satisfy \( \texttt{tdf} \geq \texttt{nfac} \).

\textbf{NE\_2\_INT\_ARRAY\_CONS}
On entry, \( \texttt{sf[i]} = \langle \text{value} \rangle \) while \( \texttt{lfac[i]} = \langle \text{value} \rangle \). Constraint: if \( \texttt{sf[i]} > 0 \), \( \texttt{lfac[i]} \geq 2 \), for \( i = 0, 1, \ldots, \texttt{nfac} - 1 \).
NE_2D_1D_INT_ARRAYS_CONS
On entry, factor[(value) × tdf + (value)] = (value) while lfac[<value>] = (value).
Constraint: factor[(i) × tdf + j] ≤ lfac[j], for i = 0, 1, ..., n - 1 and j = 0, 1, ..., nfac - 1.

NE_2D_INT_ARRAY_CONS
On entry, factor[(value) × tdf + (value)] = (value).
Constraint: factor[(i) × tdf + j] ≥ 1, for i = 0, 1, ..., n - 1 and j = 0, 1, ..., 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 = <value>.
Constraint: n ≥ 2.

On entry, nfac = <value>.
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, percent = <value>.
Constraint: 0.0 < percent < 100.0.

NE_REAL_ARRAY_CONS
On entry, wt[(value)] = <value>.
Constraint: wt[i] ≥ 0, for i = 0, 1, ..., n - 1.

7 Accuracy
Not applicable.

8 Parallelism and Performance
Not applicable.

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 lk levels, then the cell defined by the levels i_1, i_2, ..., i_n of the factors is stored in nth cell given by:
\[ m = 1 + \sum_{k=1}^{n}(i_k - 1)c_k, \]

where \( c_j = \prod_{k=j}^{n} l_k \), 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

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

int main(void)
{
    Integer exit_status = 0, i, items, j, k, lmax, 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
");
    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[^\n]");
    #else
        scanf("%*[^\n]");
    #endif
    #ifdef _WIN32
        scanf_s("%39s", nag_enum_arg, _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);
    weight = (Nag_Weightstype) nag_enum_name_to_value(nag_enum_arg);
    #if defined _WIN32
        scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s", nag_enum_arg);
    #endif
    /* nag_enum_name_to_value (x04nac). * Converts NAG enum member name to value */
    scanf_s("%NAG_IFMT" %NAG_IFMT" %lf", &n, &nfac, &percnt);
```

Mark 25
if (!(sf = NAG_ALLOC(nfac, Integer))
  || !(lfac = 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\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n");
printf("%s4.0f%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 3 362
### 10.3 Program Results

**nag_tabulate_percentile (g11bbc) Example Program Results**

<table>
<thead>
<tr>
<th>Table for 50th percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>226.00(3)</td>
</tr>
<tr>
<td>329.25(3)</td>
</tr>
<tr>
<td>185.50(3)</td>
</tr>
</tbody>
</table>