1 Purpose

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

2 Specification

```c
#include <nag.h>
#include <nagged1.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:

<table>
<thead>
<tr>
<th></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 average age could be computed:

<table>
<thead>
<tr>
<th></th>
<th>Habitat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inner-city</td>
</tr>
<tr>
<td>Male</td>
<td>25.5</td>
</tr>
<tr>
<td>Female</td>
<td>23.2</td>
</tr>
</tbody>
</table>

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

5 Arguments
1: stat – Nag_TableStats
   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
   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
   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 \( \text{stat} = \text{Nag\_TableStatsTotal} \) or \( \text{Nag\_TableStatsAv} \), the weighted total or mean is computed respectively.

If \( \text{stat} = \text{Nag\_TableStatsNObs} \), \( \text{Nag\_TableStatsLarge} \) or \( \text{Nag\_TableStatsSmall} \) the only effect of weights is to eliminate values with zero weights from the computations.

**Constraint:** \( \text{weight} = \text{Nag\_NoWeights}, \text{Nag\_Weightsvar} \) or \( \text{Nag\_Weights} \).

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 \( \text{factor} \).  
**Constraint:** \( nfac \geq 1 \).

6: \( sf[nfac] \) – const Integer  
**Input**  
*On entry:* indicates which factors in \( \text{factor} \) are to be used in the tabulation. If \( sf[i - 1] > 0 \) the \( i \)th factor in \( \text{factor} \) 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 \( \text{factor} \).  
**Constraint:** if \( sf[i - 1] > 0 \), \( lfac[i - 1] \geq 2 \), for \( i = 1, 2, \ldots, nfac \).

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

9: \( tdf \) – Integer  
**Input**  
*On entry:* the stride separating matrix column elements in the array \( \text{factor} \).  
**Constraint:** \( tdf \geq nfac \).

10: \( y[n] \) – const double  
**Input**  
*On entry:* the variable to be tabulated.  
If \( \text{stat} = \text{Nag\_TableStatsNObs} \), \( y \) is not referenced.

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

12: \( \text{table}[\text{maxt}] \) – double  
**Input/Output**  
*On entry:* if \( \text{update} = \text{Nag\_TableUpdateU} \), \( \text{table} \) must be unchanged from the previous call to \( \text{nag\_tabulate\_stats} \) (g11bac), otherwise \( \text{table} \) 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 ≥ 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 *
    Input/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 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT
   On entry, tdf = (value) while nfac = (value). These arguments must satisfy tdf ≥ nfac.

NE_2_INT_ARRAY_CONS
   On entry, sf[(value)] = (value) while lfac[0] = (value).
   Constraint: if sf[i] > 0, lfac[i] ≥ 2 for i = 0, 1, ..., nfac.

NE_2D_1D_INT_ARRAYS_CONS
   On entry, factor[(value)] × tdf + (value) = (value) while lfac[0] = (value).
   Constraint: factor[(i) × tdf + j] ≤ lfac[j], for i = 0, 1, ..., n − 1 and j = 0, 1, ..., nfac − 1.
NE_2D_INT_ARRAYCONS
On entry, $\text{factor}[(\text{value}) \times \text{tdf} + (\text{value})] = (\text{value})$.
Constraint: $\text{factor}[(i \times \text{tdf} + j) \geq 1$, for $i = 0, 1, \ldots, n - 1$ and $j = 0, 1, \ldots, \text{nfac} - 1$.

NE_ALLOC_FAIL
Dynamic memory allocation failed.

NE_BAD_PARAM
On entry, argument $\text{stat}$ had an illegal value.
On entry, argument $\text{update}$ had an illegal value.
On entry, argument $\text{weight}$ had an illegal value.

NE_G11BA_CHANGED
$\text{update} = \text{Nag\_TableUpdateU}$ and at least one of $\text{ncells}$, $\text{table}$, $\text{comm\_ar}$ or $\text{count}$ have been changed since previous call to nag_tabulate_stats (g11bac).

NE_INT_ARG_LT
On entry, $n = (\text{value})$.
Constraint: $n \geq 2$.
On entry, $\text{nfac} = (\text{value})$.
Constraint: $\text{nfac} \geq 1$.

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, $\text{maxt}$ is too small.

NE_REAL_ARRAYCONS
On entry, $\text{wt}[(\text{value})] = (\text{value})$.
Constraint: if $\text{weight} = \text{Nag\_Weights}$ or $\text{Nag\_Weightsvar}$, $\text{wt}[i] \geq 0.0$.

NE_VAR_DIV
$\text{stat} = \text{Nag\_TableStatsVar}$ and the divisor for the variance $\leq 0.0$.

NE_WT_ARGS
The $\text{wt}$ array argument must not be NULL when the $\text{weight}$ argument indicates weights.

7 Accuracy
Only applicable when $\text{stat} = \text{Nag\_TableStatsVar}$. In this case a one pass algorithm is used as described by West (1979).

8 Parallelism and Performance
Not applicable.
9 Further Comments

The tables created by \texttt{nag_tabulate_stats} (g11bac) and stored in \texttt{table}, \texttt{count} and, depending on \texttt{stat}, also in \texttt{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 \( m \)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 \( j = 1, 2, \ldots, n-1 \) 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

```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, 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 (g11bac) 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, _countof(nag_enum_arg));
    #else
    scanf("%39s", nag_enum_arg);
    #endif

    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    #ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
    #else
    scanf("%39s", nag_enum_arg);
    #endif

    stat = (Nag_TableStats) nag_enum_name_to_value(nag_enum_arg);
    INIT_FAIL(fail);
    printf("nag_tabulate_stats (g11bac) Example Program Results\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n"]);
    #else
    scanf("%*[\n"]);
    #endif
    return exit_status;
}
```

This program uses \texttt{nag_tabulate_stats} (g11bac) to compute the table of treatment means for yield over blocks. It first reads the data, then sets the \texttt{stat} parameter to \texttt{Nag_TableStats} to compute the table. Finally, it prints the results.
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)
    {
    #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)
    {
    #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


Mark 25
g11bac

Mark 25
g11bac.7
/* nag_tabulate_stats (g11bac).
 * Computes multiway table from set of classification
 * factors using selected statistic
 */

gen_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 (g11bac).\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;
items = 0;
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 (g11bac) 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
1363 5 8
2113 5 0
2213 4 0
2312 0 3
2123 9 7
2223 5 6
2322 9 8
2133 8 2
2233 7 6
2333 5 5
2144 1 8
2243 8 7
2343 7 9
2154 3 2
2253 3 9
2352 9 3
2163 2 2
2264 1 7
2363 4 2
311 8 2
3212 9 7
3311 3 3
3123 0 6
3223 5 2
3323 6 1
3132 2 0
3233 3 3
3332 7 0
3143 8 8
3243 7 9
3342 7 4
3153 3 6
3253 0 7
3352 6 6
3163 8 9
3263 3 3
3363 5 3
3360 1 1

10.3 Program Results

nag_tabulate_stats (g11bac) Example Program Results

Table

<table>
<thead>
<tr>
<th></th>
<th>235.33(3)</th>
<th>342.67(3)</th>
<th>309.33(3)</th>
<th>395.00(3)</th>
<th>373.33(3)</th>
<th>350.00(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>332.67(3)</td>
<td>341.67(3)</td>
<td>370.33(3)</td>
<td>370.33(3)</td>
<td>326.67(3)</td>
<td>381.00(3)</td>
<td></td>
</tr>
<tr>
<td>196.33(3)</td>
<td>332.67(3)</td>
<td>320.33(3)</td>
<td>338.00(3)</td>
<td>292.33(3)</td>
<td>351.00(3)</td>
<td></td>
</tr>
</tbody>
</table>