NAG C Library Function Document

nag_rngs_2_way_table (g05qdc)

1 Purpose
nag_rngs_2_way_table (g05qdc) generates a random two-way table.

2 Specification

```c
void nag_rngs_2_way_table (Nag_OrderType order, Integer mode, Integer nrow,
                        Integer ncol, const Integer totr[], const Integer totc[], Integer x[],
                        Integer pdx, Integer igen, Integer iseed[], double r[], Integer nr,
                        NagError *fail)
```

3 Description

Given $m$ row totals $R_i$ and $n$ column totals $C_j$ (with $\sum_{i=1}^{m} R_i = \sum_{j=1}^{n} C_j = T$, say), nag_rngs_2_way_table (g05qdc) will generate a pseudorandom two-way table of integers such that the row and column totals are satisfied.

The method used is based on that described by Patefield (1981) which is most efficient when $T$ is large relative to the number of table entries $m \times n$ (i.e., $T > 2mn$). Entries are generated one row at a time and one entry at a time within a row. Each entry is generated using the conditional probability distribution for that entry given the entries in the previous rows and the previous entries in the same row.

A reference vector is used to store computed values that can be reused in the generation of new tables with the same row and column totals. nag_rngs_2_way_table (g05qdc) can be called to simply set up the reference vector, or to generate a two-way table using a reference vector set up in a previous call, or it can combine both functions in a single call.

One of the initialisation functions nag_rngs_init_repeatable (g05kbc) (for a repeatable sequence if computed sequentially) or nag_rngs_init_nonrepeatable (g05kcc) (for a non-repeatable sequence) must be called prior to the first call to nag_rngs_2_way_table (g05qdc).

4 References

Patefield WM (1981) An efficient method of generating $R \times C$ tables with given row and column totals Appl. Stats. 30 91–97

5 Parameters

1:  
`order` – Nag_OrderType  

*Input*  

*On entry*: the `order` parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by `order = Nag_RowMajor`. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

*Constraint*: `order = Nag_RowMajor` or `Nag_ColMajor`.

2:  
`mode` – Integer  

*Input*  

*On entry*: a code for selecting the operation to be performed by the function:
mode = 0
Set up reference vector only.

mode = 1
Generate two-way table using reference vector set up in a prior call to nag_rngs_2_way_table (g05qdc).

mode = 2
Set up reference vector and generate two-way table.

Constraint: 0 ≤ mode ≤ 2.

3: nrow – Integer

On entry: the number of rows in the table, m.

Constraint: nrow ≥ 2.

4: ncol – Integer

On entry: the number of columns in the table, n.

Constraint: ncol ≥ 2.

5: totr[nrow] – const Integer

On entry: the m row totals, Ri, for i = 1, 2, ..., m.

Constraints:

\[ \text{totr}[i] ≥ 0 \] for \( i = 0, 1, \ldots, m - 1 \);
\[ \sum_{i=1}^{m} \text{totr}[i] = \sum_{j=1}^{n} \text{totc}[j]. \]

6: totc[ncol] – const Integer

On entry: the n column totals, Cj, for j = 1, 2, ..., n.

Constraints:

\[ \text{totc}[j] ≥ 0 \] for \( j = 0, 1, \ldots, n - 1 \);
\[ \sum_{j=1}^{n} \text{totc}[j] = \sum_{i=1}^{m} \text{totr}[i]. \]

7: x[dim] – Integer

Note: the dimension, dim, of the array x must be at least \( \max(1, \text{pdx} \times \text{ncol}) \) when order = Nag_ColMajor and at least \( \max(1, \text{pdx} \times \text{nrow}) \) when order = Nag_RowMajor.

Where \( \text{X}(i, j) \) appears in this document, it refers to the array element

\[
\begin{align*}
\text{if order = Nag_ColMajor, } & \text{x}[(j - 1) \times \text{pdx} + i - 1]; \\
\text{if order = Nag_RowMajor, } & \text{x}[(i - 1) \times \text{pdx} + j - 1].
\end{align*}
\]

On exit: a pseudo-random two-way m by n table, X, with element \( \text{X}(i, j) \) containing the \((i, j)\)th entry in the table such that \( \sum_{i=1}^{\text{nrow}} \text{X}(i, j) = \text{totc}[j] \) and \( \sum_{j=1}^{\text{ncol}} \text{X}(i, j) = \text{totr}[i] \)

8: pdx – Integer

On entry: the stride separating matrix row or column elements (depending on the value of order) in the array x.

Constraints:

\[
\begin{align*}
\text{if order = Nag_ColMajor, } & \text{pdx} ≥ \text{nrow}; \\
\text{if order = Nag_RowMajor, } & \text{pdx} ≥ \text{ncol}.
\end{align*}
\]
9:  igen – Integer
    Input
    On entry: must contain the identification number for the generator to be used to return a pseudo-
    random number and should remain unchanged following initialisation by a prior call to one of the
    functions nag_rngs_init-repeatable (g05kbc) or nag_rngs_init-nonrepeatable (g05kcc).

    Input/Output
    On entry: contains values which define the current state of the selected generator.
    On exit: contains updated values defining the new state of the selected generator.

11:  r[+nr] – double
    Input/Output
    On exit: the reference vector.

12:  nr – Integer
    Input
    On entry: the dimension of the array r as declared in the function from which
    nag_mgs_2-way_table (g05qdc) is called.
    Constraint: nr \geq \sum_{i=1}^{nrow} \text{totr}[i] + 4.

13:  fail – NagError *
    Input/Output
    The NAG error parameter (see the Essential Introduction).

6  Error Indicators and Warnings

**NE_INT**

On entry, nrow = (value).
Constraint: nrow \geq 2.

On entry, pdx = (value).
Constraint: pdx > 0.

On entry, nr not large enough, nr = (value). Minimum length required = (value).
On entry, mode = (value).
Constraint: 0 \leq mode \leq 2.

**NE_INT_2**

On entry, pdx = (value), nrow = (value).
Constraint: pdx \geq nrow.

On entry, pdx = (value), ncol = (value).
Constraint: pdx \geq ncol.

On entry, nrow < 2 or ncol < 2: nrow = (value), ncol = (value).

**NE_PREV_CALL**

nrow or ncol is not the same as when r was set up in a previous call. Previous value of
nrow = (value), nrow = (value). Previous value of ncol = (value), ncol = (value).

**NE_REAL_ARRAY_ELEM_CONS**

On entry, totc has at least one negative element.
On entry, totr has at least one negative element.

**NE_REAL_ARRAYS_SUM**

On entry, the arrays totr and totc do not sum to the same total: totr array total is (value) totc array
total is (value).
NE_ALLOC_FAIL
Memory allocation failed.

NE_BAD_PARAM
On entry, parameter (value) had an illegal value.

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.

7 Accuracy
None.

8 Further Comments
None.

9 Example
Following initialisation of the pseudo-random number generator by a call to nag_rngs_init_repeatable (g05kbc), a 4 by 3 two-way table, with row totals of 9, 11, 7 and 23 respectively, and column totals of 16, 17 and 17 respectively, is generated and printed.

9.1 Program Text
/* nag_rngs_2_way_table(g05qdc) Example Program. *
 * Copyright 2001 Numerical Algorithms Group.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

int main(void)
{
    /* Scalars */
    Integer i, igen, j, rctot;
    Integer exit_status=0;
    Integer nrow, ncol, nr;
    Integer pdx;
    NagError fail;
    Nag_OrderType order;

    /* Arrays */
    double *r=0;
    Integer *totc=0, *totr=0, *x=0;
    Integer iseed[4];

    #ifdef NAG_COLUMN_MAJOR
    #define X(I,J) x[(J-1)*pdx+I-1]
    order = Nag_ColMajor;
    #else
    #define X(I,J) x[(I-1)*pdx+J-1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("g05qdc Example Program Results\n\n");

    return exit_status;
}
nrow = 4;
ncol = 3;

nr = 60;

/* Allocate memory */
if ( !(r = NAG_ALLOC(nr, double)) || 
    !(totc = NAG_ALLOC(ncol, Integer)) || 
    !(totr = NAG_ALLOC(nrow, Integer)) || 
    !(x = NAG_ALLOC(nrow * ncol, Integer)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

#ifdef NAG_COLUMN_MAJOR
pdx = nrow;
#else
pdx = ncol;
#endif

/* Set the table row and column totals */
totr[0] = 9;
totr[1] = 11;
totr[2] = 7;
totr[3] = 23;
totc[0] = 16;
totc[1] = 17;
totc[2] = 17;
rctot = 50;

/* igen identifies the stream. */
igen = 1;
/* Initialise the seed to a repeatable sequence */
iseed[0] = 1762543;
iseed[1] = 9324783;
iseed[2] = 42344;
iseed[3] = 742355;
g05kbc(&igen, iseed);

/* Choose MODE = 2 */
g05qdc(order, 2, nrow, ncol, totr, totc, x, pdx, igen, iseed, r, nr, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g05qdc.\n", fail.message);
    exit_status = 1;
    goto END;
}
for (i = 1; i <= nrow; ++i)
{
    Vprintf("%lds", "");
    for (j = 1; j <= ncol; ++j)
    {
        Vprintf("%4ld %s", X(i,j), j%3 == 0 ?" |":" ");
    }
    Vprintf("%5ld\n", totr[i - 1]);
}
Vprintf(" ---------------+-----
");
Vprintf("%lds", "");
for (j = 1; j <= ncol; ++j)
{
    Vprintf("%4ld %s", totc[j - 1], j%3 == 0 ?" |":" ");
}
Vprintf("%5ld\n", rctot);

END:
if (r) NAG_FREE(r);
if (totc) NAG_FREE(totc);
if (totr) NAG_FREE(totr);
if (x) NAG_FREE(x);
g05qdc

    return exit_status;
}

9.2 Program Data

None.

9.3 Program Results

g05qdc Example Program Results

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>6</td>
<td>23</td>
</tr>
</tbody>
</table>

---------------

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>17</td>
<td>17</td>
<td>50</td>
</tr>
</tbody>
</table>