

NAG Library Function Document

nag_rand_2_way_table (g05pzc)

1 Purpose

nag_rand_2_way_table (g05pzc) generates a random two-way table.

2 Specification

```
#include <nag.h>
#include <nagg05.h>

void nag_rand_2_way_table (Nag_ModeRNG mode, Integer nrow, Integer ncol,
    const Integer totr[], const Integer totc[], double r[], Integer lr,
    Integer state[], Integer x[], Integer pdx, 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_rand_2_way_table (g05pzc) 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_rand_2_way_table (g05pzc) 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 initialization functions nag_rand_init_repeatable (g05kfc) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeatable (g05kgc) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_2_way_table (g05pzc).

4 References

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

5 Arguments

1: **mode** – Nag_ModeRNG *Input*

On entry: a code for selecting the operation to be performed by the function.

mode = Nag_InitializeReference
Set up reference vector only.

mode = Nag_GenerateFromReference
Generate two-way table using reference vector set up in a prior call to nag_rand_2_way_table (g05pzc).

mode = Nag_InitializeAndGenerate

Set up reference vector and generate two-way table.

C o n s t r a i n t: **mode** = Nag_InitializeReference, Nag_GenerateFromReference o r Nag_InitializeAndGenerate.

2: **nrow** – Integer *Input*

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

Constraint: **nrow** ≥ 2 .

3: **ncol** – Integer *Input*

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

Constraint: **ncol** ≥ 2 .

4: **totr**[**nrow**] – const Integer *Input*

On entry: the m row totals, R_i , for $i = 1, 2, \dots, m$.

Constraints:

$$\mathbf{totr}[i - 1] \geq 0, \text{ for } i = 1, 2, \dots, m;$$

$$\sum_{i=1}^m \mathbf{totr}[i - 1] = \sum_{j=1}^n \mathbf{totc}[j - 1];$$

$$\sum_i \mathbf{totr}[i - 1] > 0, \text{ for } i = 1, 2, \dots, m.$$

5: **totc**[**ncol**] – const Integer *Input*

On entry: the n column totals, C_j , for $j = 1, 2, \dots, n$.

Constraints:

$$\mathbf{totc}[j - 1] \geq 0, \text{ for } j = 1, 2, \dots, n;$$

$$\sum_{j=1}^n \mathbf{totc}[j - 1] = \sum_{i=1}^m \mathbf{totr}[i - 1].$$

6: **r**[**lr**] – double *Communication Array*

On entry: if **mode** = Nag_GenerateFromReference, the reference vector from the previous call to nag_rand_2_way_table (g05pzc).

On exit: the reference vector.

7: **lr** – Integer *Input*

On entry: the dimension of the array **r**.

Constraint: **lr** $\geq \sum_{i=1}^m \mathbf{totr}[i - 1] + 5$.

8: **state**[*dim*] – Integer *Communication Array*

Note: the dimension, *dim*, of this array is dictated by the requirements of associated functions that must have been previously called. This array **MUST** be the same array passed as argument **state** in the previous call to nag_rand_init_repeatable (g05kfc) or nag_rand_init_nonrepeatable (g05kgc).

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

- 9: **x**[**nrow** × **pdx**] – Integer *Output*
On exit: if **mode** = Nag_GenerateFromReference or Nag_InitializeAndGenerate, a pseudorandom two-way m by n table, X , with element $\mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1]$ containing the (i, j) th entry in the table such that $\sum_{i=1}^m \mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1] = \mathbf{totc}[j - 1]$ and $\sum_{j=1}^n \mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1] = \mathbf{totr}[i - 1]$
- 10: **pdx** – Integer *Input*
On entry: the stride separating matrix column elements in the array **x**.
Constraint: **pdx** ≥ **ncol**.
- 11: **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_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **lr** is not large enough, **lr** = $\langle value \rangle$: minimum length required = $\langle value \rangle$.

On entry, **ncol** = $\langle value \rangle$.

Constraint: **ncol** ≥ 2.

On entry, **nrow** = $\langle value \rangle$.

Constraint: **nrow** ≥ 2.

NE_INT_2

On entry, **pdx** = $\langle value \rangle$ and **ncol** = $\langle value \rangle$.

Constraint: **pdx** ≥ **ncol**.

NE_INT_ARRAY

On entry, at least one element of **totr** is negative or **totr** sums to zero.

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

NE_INT_ARRAY_2

On entry, the arrays **totr** and **totc** do not sum to the same total: **totr** array total is $\langle value \rangle$, **totc** array total is $\langle value \rangle$.

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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_INVALID_STATE

On entry, **state** vector has been corrupted or not initialized.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

NE_PREV_CALL

nrow or **ncol** is not the same as when **r** was set up in a previous call.
Previous value of **nrow** = $\langle value \rangle$ and **nrow** = $\langle value \rangle$.
Previous value of **ncol** = $\langle value \rangle$ and **ncol** = $\langle value \rangle$.

7 Accuracy

None.

8 Parallelism and Performance

nag_rand_2_way_table (g05pzc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

Following initialization of the pseudorandom number generator by a call to nag_rand_init_repeatable (g05kfc), this example generates and prints 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.

10.1 Program Text

```

/* nag_rand_2_way_table (g05pzc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

#define X(I, J) x[I*pdx + J]

int main(void)
{
    /* Integer scalar and array declarations */

```

```

Integer exit_status = 0;
Integer lr, i, j, lstate, rctot, x_size;
Integer *state = 0, *x = 0;
Integer pdx;

/* NAG structures */
NagError fail;
Nag_ModeRNG mode;

/* Double scalar and array declarations */
double *r = 0;

/* Set the size of the table and the row and column totals */
Integer nrow = 3;
Integer ncol = 4;
Integer totr[] = { 16, 17, 17 };
Integer totc[] = { 9, 11, 7, 23 };

/* Choose the base generator */
Nag_BaseRNG genid = Nag_Basic;
Integer subid = 0;

/* Set the seed */
Integer seed[] = { 1762543 };
Integer lseed = 1;

/* Initialize the error structure */
INIT_FAIL(fail);

printf("nag_rand_2_way_table (g05pzc) Example Program Results\n\n");

/* Get the length of the state array */
lstate = -1;
nag_rand_init_repeatabe(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatabe (g05kfc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Calculate the grand total */
for (i = 0, rctot = 0; i < nrow; i++)
    rctot += totr[i];

/* Calculate the size of the reference vector */
lr = 5 + rctot;

pdx = ncol;
x_size = pdx * nrow;

/* Allocate arrays */
if (!(r = NAG_ALLOC(lr, double)) ||
    !(state = NAG_ALLOC(lstate, Integer)) ||
    !(x = NAG_ALLOC(x_size, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Initialize the generator to a repeatable sequence */
nag_rand_init_repeatabe(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatabe (g05kfc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Generate the random table */

```

```

mode = Nag_InitializeAndGenerate;
nag_rand_2_way_table(mode, nrow, ncol, totr, totrc, r, lr, state, x, pdx,
                    &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_2_way_table (g05pzc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Display the results */
for (i = 0; i < nrow; i++) {
    printf(" ");
    for (j = 0; j < ncol; j++)
        printf("%4" NAG_IFMT "%s", X(i, j), (j + 1) % ncol ? " " : " | ");
    printf("%5" NAG_IFMT "", totr[i]);
    printf("\n");
}
printf(" -----+-----\n ");
for (j = 0; j < ncol; j++)
    printf("%4" NAG_IFMT "%s", totrc[j], (j + 1) % ncol ? " " : " | ");
printf("%5" NAG_IFMT "", rctot);
printf("\n");

END:
NAG_FREE(r);
NAG_FREE(state);
NAG_FREE(x);

return exit_status;
}

```

10.2 Program Data

None.

10.3 Program Results

nag_rand_2_way_table (g05pzc) Example Program Results

2	6	2	6		16
4	1	4	8		17
3	4	1	9		17
-----+					-----
9	11	7	23		50
