

NAG C Library Function Document

nag_rngs_gen_multinomial (g05mrc)

1 Purpose

nag_rngs_gen_multinomial (g05mrc) generates a sequence of n variates, each consisting of k pseudo-random integers, from the discrete multinomial distribution with k outcomes and m trials, where the outcomes have probabilities p_1, p_2, \dots, p_k respectively.

2 Specification

```
void nag_rngs_gen_multinomial (Nag_OrderType order, Integer mode, Integer m,
    Integer k, const double p[], Integer n, Integer x[], Integer pdx,
    Integer igen, Integer iseed[], double r[], NagError *fail)
```

3 Description

nag_rngs_gen_multinomial (g05mrc) generates a sequence of n groups of k integers $x_{i,j}$ for $j = 1, 2, \dots, k$ and $i = 1, 2, \dots, n$, from a multinomial distribution with m trials and k outcomes, where the probability of $x_{i,j} = I_j$ for each $j = 1, 2, \dots, k$ is

$$P(i_1 = I_1, \dots, i_k = I_k) = \frac{m!}{\prod_{j=1}^k I_j!} \prod_{j=1}^k p_j^{I_j} = \frac{m!}{I_1! I_2! \dots I_k!} p_1^{I_1} p_2^{I_2} \dots p_k^{I_k},$$

where

$$\sum_{j=1}^k p_j = 1 \quad \text{and} \quad \sum_{j=1}^k I_j = m.$$

A single trial can have several outcomes (k , say) and the probability of achieving each outcome is known (p_j , say). After m trials each outcome will have occurred a certain number of times. The k numbers representing the numbers of occurrences for each outcome after m trials is then a single sample from the multinomial distribution defined by the parameters k , m and p_j , for $j = 1, 2, \dots, k$. This function returns n such samples with each sample being stored as a row in a two-dimensional array of integers.

When $k = 2$ this distribution is equivalent to the binomial distribution with parameters m and $p = p_1$ (nag_rngs_binomial (g05mjc)).

The variates can be generated with or without using a search table and index. If a search table is used then it is stored with the index in a reference vector and subsequent calls to nag_rngs_gen_multinomial (g05mrc) with the same parameter values can then use this reference vector to generate further variates. The reference array is only generated for the outcome with greatest probability. The number of successes for the outcome with greatest probability is calculated first as for the binomial distribution (nag_rngs_binomial (g05mjc)); the number of successes for other outcomes are calculated in turn for the remaining reduced multinomial distribution; the number of successes for the final outcome is simply calculated to ensure that the total number of successes is m .

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_gen_multinomial (g05mrc).

4 References

Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

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 variates using reference vector set up in a prior call to nag_rngs_gen_multinomial (g05mrc).
mode = 2
 Set up reference vector and generate variates.
mode = 3
 Generate variates without using the reference vector.
Constraint: $0 \leq \mathbf{mode} \leq 3$.
- 3: **m** – Integer *Input*
On entry: the number of trials, m , of the multinomial distribution.
Constraint: $\mathbf{m} \geq 0$.
- 4: **k** – Integer *Input*
On entry: the number of possible outcomes, k , of the multinomial distribution.
Constraint: $\mathbf{k} \geq 2$.
- 5: **p[k]** – const double *Input*
On entry: contains the probabilities p_j , for $j = 1, 2, \dots, k$, of the k possible outcomes of the multinomial distribution.
Constraint: $0.0 \leq \mathbf{p}[j - 1] \leq 1.0$ and $\sum_{j=1}^k \mathbf{p}[j - 1] = 1.0$.
- 6: **n** – Integer *Input*
On entry: the number, n , of pseudo-random numbers to be generated.
Constraint: $\mathbf{n} \geq 1$.
- 7: **x[dim]** – Integer *Output*
Note: the dimension, dim , of the array **x** must be at least $\max(1, \mathbf{pdx} \times \mathbf{k})$ when **order = Nag_ColMajor** and at least $\max(1, \mathbf{pdx} \times \mathbf{n})$ when **order = Nag_RowMajor**.
 If **order = Nag_ColMajor**, the (i, j) th element of the matrix X is stored in $\mathbf{x}[(j - 1) \times \mathbf{pdx} + i - 1]$ and if **order = Nag_RowMajor**, the (i, j) th element of the matrix X is stored in $\mathbf{x}[(i - 1) \times \mathbf{pdx} + j - 1]$.
On exit: the first n rows of **x** each contain k pseudo-random numbers representing a k -dimensional variate from the specified multinomial distribution.

- 8: **pdx** – Integer *Input*
On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **x**.
Constraints:
 if **order** = **Nag_ColMajor**, **pdx** \geq **n**;
 if **order** = **Nag_RowMajor**, **pdx** \geq **k**.
- 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).
- 10: **iseed**[4] – Integer *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**[*dim*] – double *Input/Output*
Note: the dimension, *dim*, of the array **r** must be at least $22 + 20\sqrt{\mathbf{m} \times p_{max}(1 - p_{max})}$ when **mode** $<$ 3 and at least 1 otherwise.
On exit: the reference vector.
- 12: **fail** – NagError * *Input/Output*
 The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

Constraint: $0 \leq \mathbf{mode} \leq 3$.

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

Constraint: **m** \geq 0.

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

Constraint: **k** \geq 2.

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

Constraint: **n** \geq 1.

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

Constraint: **pdx** $>$ 0.

NE_INT_2

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

Constraint: **pdx** \geq **n**.

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

Constraint: **pdx** \geq **k**.

NE_BAD_PARAM

On entry, **p**[*i* - 1] $<$ 0.0 or **p**[*i* - 1] $>$ 1.0 where: *i* = $\langle value \rangle$ and **p**[*i* - 1] = $\langle value \rangle$.

NE_PREV_CALL

$\max(\mathbf{p}[i-1])$ or \mathbf{m} is not the same as when \mathbf{r} was set up in a previous call. Previous value of $\max(\mathbf{p}[i-1]) = \langle value \rangle$, $\max(\mathbf{p}[i-1]) = \langle value \rangle$. Previous value of $\mathbf{m} = \langle value \rangle$, $\mathbf{m} = \langle value \rangle$.

NE_REAL

On entry, the sum of $\mathbf{p}[i-1]$, $i = 1, \dots, \mathbf{k}$ is not unity. The difference from unity in the summation is: $\langle value \rangle$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ 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

Not applicable.

8 Further Comments

Only the reference vector for one outcome can be set up because the conditional distributions cannot be known in advance of the generation of variates. The outcome with greatest probability of success is chosen for the reference vector because it will have the greatest spread of likely values.

9 Example

The example program prints 20 pseudo-random k -dimensional variates from a multinomial distribution with $k = 4$, $m = 6000$, $p_1 = 0.08$, $p_2 = 0.1$, $p_3 = 0.8$ and $p_4 = 0.02$, generated by a single call to `nag_rngs_gen_multinomial` (g05mrc), after initialisation by `nag_rngs_init_repeatable` (g05kbc).

9.1 Program Text

```

/* nag_rngs_gen_multinomial(g05mrc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

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

    /* Arrays */
    double *p=0, *r=0;
    Integer *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("g05mrc Example Program Results\n\n");
k = 4;
n = 20;
nr = 16007;

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

#ifdef NAG_COLUMN_MAJOR
pdx = n;
#else
pdx = k;
#endif

/* Set the distribution parameters P and M */
p[0] = 0.08;
p[1] = 0.1;
p[2] = 0.8;
p[3] = 0.02;
m = 6000;
/* Initialise the seed to a repeatable sequence */
iseed[0] = 1762543;
iseed[1] = 9324783;
iseed[2] = 42344;
iseed[3] = 742355;
/* igen identifies the stream. */
igen = 1;
g05kbc(&igen, iseed);

/* Choose MODE = 2 */
g05mrc(order, 2, m, k, p, n, x, pdx, igen, iseed, r, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g05mrc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= k; ++j)
    {
        Vprintf("%12ld%s", X(i,j), j%10 == 0 || j == 4 ? "\n": " ");
    }
}
END:
if (p) NAG_FREE(p);
if (r) NAG_FREE(r);
if (x) NAG_FREE(x);
return exit_status;
}

```

9.2 Program Data

None.

9.3 Program Results

g05mrc Example Program Results

503	615	4758	124
452	536	4851	161
488	581	4793	138
443	624	4820	113
471	554	4851	124
480	609	4795	116
487	568	4807	138
473	609	4792	126
516	580	4787	117
459	582	4842	117
499	582	4801	118
489	594	4794	123
486	597	4806	111
454	543	4878	125
526	599	4745	130
512	574	4790	124
477	582	4832	109
476	615	4789	120
461	654	4743	142
476	595	4812	117
