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 \( p_1, p_2, \ldots, p_k \) respectively.

2 Specification

```c
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 \( x_{ij} \) for \( j = 1, 2, \ldots, k \) and \( i = 1, 2, \ldots, n \), from a multinomial distribution with \( m \) trials and \( k \) outcomes, where the probability of \( x_{ij} = I_j \) for each \( j = 1, 2, \ldots, k \) is

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

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 \( j = 1, 2, \ldots, k \). 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

5 Parameters

1: order – Nag_OrderType

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

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 ≤ mode ≤ 3.

3: m – Integer

On entry: the number of trials, m, of the multinomial distribution.

Constraint: m ≥ 0.

4: k – Integer

On entry: the number of possible outcomes, k, of the multinomial distribution.

Constraint: k ≥ 2.

5: p[k] – const double

On entry: contains the probabilities p[j], for j = 1, 2, ..., k, of the k possible outcomes of the
multinomial distribution.

Constraint: 0.0 ≤ p[j] ≤ 1.0 and \( \sum_{j=1}^{k} p[j] = 1.0 \).

6: n – Integer

On entry: the number, n, of pseudo-random numbers to be generated.

Constraint: n ≥ 1.

7: x[dim] – Integer

Note: the dimension, dim, of the array x must be at least max(1, pdx × k) when
order = Nag_ColMajor and at least max(1, pdx × n) when order = Nag_RowMajor.

If order = Nag_ColMajor, the (i, j)th element of the matrix X is stored in x[(j - 1) × pdx + i - 1] and
if order = Nag_RowMajor, the (i, j)th element of the matrix X is stored in x[(i - 1) × 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** ≥ **n**;

if **order** = Nag_RowMajor, **pdx** ≥ **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√**m** × **p_max**(1 − **p_max**)/2 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** = ⟨value⟩.

Constraint: 0 ≤ **mode** ≤ 3.

On entry, **m** = ⟨value⟩.

Constraint: **m** ≥ 0.

On entry, **k** = ⟨value⟩.

Constraint: **k** ≥ 2.

On entry, **n** = ⟨value⟩.

Constraint: **n** ≥ 1.

On entry, **pdx** = ⟨value⟩.

Constraint: **pdx** > 0.

**NE_INT_2**

On entry, **pdx** = ⟨value⟩, **n** = ⟨value⟩.

Constraint: **pdx** ≥ **n**.

On entry, **pdx** = ⟨value⟩, **k** = ⟨value⟩.

Constraint: **pdx** ≥ **k**.

**NE_BAD_PARAM**

On entry, p[i − 1] < 0.0 or p[i − 1] > 1.0 where: i = ⟨value⟩ and p[i − 1] = ⟨value⟩.
NE_PREV_CALL
max(p[i − 1]) or m is not the same as when r was set up in a previous call. Previous value of
max(p[i − 1]) = \langle value\rangle, max(p[i − 1]) = \langle value\rangle. Previous value of m = \langle value\rangle, m = \langle value\rangle.

NE_REAL
On entry, the sum of p[i − 1], i = 1, ..., 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.
 */
#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.
%s
", 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**

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