

NAG Library Function Document

nag_rngs_binomial (g05mjc)

1 Purpose

nag_rngs_binomial (g05mjc) generates a vector of pseudorandom integers from the discrete binomial distribution with parameters m and p .

2 Specification

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

```
void nag_rngs_binomial (Integer mode, Integer m, double p, Integer n, Integer x[],
    Integer igen, Integer iseed[], double r[], NagError *fail)
```

3 Description

nag_rngs_binomial (g05mjc) generates n integers x_i from a discrete binomial distribution, where the probability of $x_i = I$ is

$$P(x_i = I) = \frac{m!}{I!(m-I)!} p^I \times (1-p)^{m-I}, \quad I = 0, 1, \dots, m,$$

where $0 \leq m$ and $0 \leq p \leq 1$. This represents the probability of achieving I successes in m trials when the probability of success at a single trial is p .

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_binomial (g05mjc) with the same parameter values can then use this reference vector to generate further variates.

One of the initialization 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_binomial (g05mjc).

4 References

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* (3rd Edition) Griffin
 Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

5 Arguments

1: **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_binomial (g05mjc).

mode = 2

Set up reference vector and generate variates.

mode = 3

Generate variates without using the reference vector.

Constraint: **mode** = 0, 1, 2 or 3.

- 2: **m** – Integer *Input*
On entry: m , the number of trials of the distribution.
Constraint: $\mathbf{m} \geq 0$.
- 3: **p** – double *Input*
On entry: p , the probability of success of the binomial distribution.
Constraint: $0.0 \leq \mathbf{p} \leq 1.0$.
- 4: **n** – Integer *Input*
On entry: n , the number of pseudorandom numbers to be generated.
Constraint: $\mathbf{n} \geq 1$.
- 5: **x[n]** – Integer *Output*
On exit: the n pseudorandom numbers from the specified binomial distribution.
- 6: **igen** – Integer *Input*
On entry: must contain the identification number for the generator to be used to return a pseudorandom number and should remain unchanged following initialization by a prior call to `nag_rngs_init_repeatable` (g05kbc) or `nag_rngs_init_nonrepeatable` (g05kcc).
- 7: **iseed**[4] – Integer *Communication Array*
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.
- 8: **r**[*dim*] – double *Communication Array*
Note: the dimension, *dim*, of the array **r** must be at least
 $22 + 20\sqrt{\mathbf{m} \times \mathbf{p}(1 - \mathbf{p})}$ when **mode** = 3;
 1 otherwise.
On entry: if **mode** = 1, the reference vector from the previous call to `nag_rngs_binomial` (g05mjc).
 If **mode** = 3, **r** is not referenced by `nag_rngs_binomial` (g05mjc).
On exit: the reference vector.
- 9: **fail** – NagError * *Input/Output*
 The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

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

NE_INT

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

Constraint: $\mathbf{m} \geq 0$.

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

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

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

Constraint: **mode** = 0, 1, 2 or 3.

On entry, $\mathbf{n} = \langle \text{value} \rangle$.
 Constraint: $\mathbf{n} \geq 1$.

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_PREV_CALL

\mathbf{p} or \mathbf{m} is not the same as when \mathbf{r} was set up in a previous call. Previous value of $\mathbf{p} = \langle \text{value} \rangle$, $\mathbf{p} = \langle \text{value} \rangle$. Previous value of $\mathbf{m} = \langle \text{value} \rangle$, $\mathbf{p} = \langle \text{value} \rangle$.

NE_REAL

On entry, $\mathbf{p} < 0.0$ or $\mathbf{p} > 1.0$: $\mathbf{p} = \langle \text{value} \rangle$.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example prints 20 pseudorandom integers from a binomial distribution with parameters $m = 6000$ and $p = 0.8$, generated by a single call to `nag_rngs_binomial` (g05mjc), after initialization by `nag_rngs_init_repeatable` (g05kbc).

9.1 Program Text

```

/* nag_rngs_binomial (g05mjc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagg05.h>

int main(int argc, char *argv[])
{
    FILE      *fpout;
    /* Scalars */
    double    p;
    Integer   i, igen, m, n, nr;
    Integer   exit_status = 0;
    NagError  fail;

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

    INIT_FAIL(fail);

    /* Check for command-line IO options */
    fpout = nag_example_file_io(argc, argv, "--results", NULL);
    fprintf(fpout, "nag_rngs_binomial (g05mjc) Example Program Results\n\n");

```

```

n = 20;
nr = 6007;

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

/* Set the distribution parameters P and M */
p = 0.8;
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;
/* nag_rngs_init_repeatable (g05kbc).
 * Initialize seeds of a given generator for random number
 * generating functions (that pass seeds explicitly) to give
 * a repeatable sequence
 */
nag_rngs_init_repeatable(&igen, iseed);

/* Choose MODE = 2 */
/* nag_rngs_binomial (g05mjc).
 * Generates a vector of random integers from a binomial
 * distribution, seeds and generator number passed
 * explicitly
 */
nag_rngs_binomial(2, m, p, n, x, igen, iseed, r, &fail);
if (fail.code != NE_NOERROR)
{
    fprintf(fpout, "Error from nag_rngs_binomial (g05mjc).\n%s\n",
            fail.message);
    exit_status = 1;
    goto END;
}
for (i = 0; i < n; ++i)
{
    fprintf(fpout, "%12ld\n", x[i]);
}
END:
if (fpout != stdout) fclose(fpout);
if (r) NAG_FREE(r);
if (x) NAG_FREE(x);
return exit_status;
}

```

9.2 Program Data

None.

9.3 Program Results

nag_rngs_binomial (g05mjc) Example Program Results

```

4758
4851
4793
4820
4851
4795
4807
4792

```

4787
4842
4801
4794
4806
4878
4745
4790
4832
4789
4743
4812
