NAG Library Function Document
nag_rand_binomial (g05tac)

1 Purpose
nag_rand_binomial (g05tac) generates a vector of pseudorandom integers from the discrete binomial distribution with parameters $m$ and $p$.

2 Specification

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

void nag_rand_binomial (Nag_ModeRng mode, Integer n, Integer m, double p,
           double r[], Integer lr, Integer state[], Integer x[], NagError *fail)
```

3 Description

nag_rand_binomial (g05tac) generates $n$ integers $x_i$ from a discrete binomial distribution, where the probability of $x_i = I$ is

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

where $m \geq 0$ 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_rand_binomial (g05tac) with the same parameter values can then use this reference vector to generate further variates.

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_binomial (g05tac).

4 References


5 Arguments

1: mode – Nag_ModeRng
   
   Input

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

   mode = Nag_IntegerizeReference
   Set up reference vector only.

   mode = Nag_GenerateFromReference
   Generate variates using reference vector set up in a prior call to nag_rand_binomial (g05tac).

   mode = Nag_IntegerizeAndGenerate
   Set up reference vector and generate variates.
Given:

\( \text{mode} = \text{Nag\_Generate\_Without\_Reference} \)

Generate variates without using the reference vector.

\( \text{Constraint: mode} = \text{Nag\_Initialize\_Reference}, \text{Nag\_Generate\_From\_Reference}, \text{Nag\_Initialize\_And\_Generate} \) or \( \text{Nag\_Generate\_Without\_Reference}. \)

2: \( n \) – Integer

On entry: \( n \), the number of pseudorandom numbers to be generated.

\( \text{Constraint: } n \geq 0. \)

3: \( m \) – Integer

On entry: \( m \), the number of trials of the distribution.

\( \text{Constraint: } m \geq 0. \)

4: \( p \) – double

On entry: \( p \), the probability of success of the binomial distribution.

\( \text{Constraint: } 0.0 \leq p \leq 1.0. \)

5: \( r[\text{lr}] \) – double

\( \text{Communication Array} \)

On entry: if \( \text{mode} = \text{Nag\_Generate\_From\_Reference} \), the reference vector from the previous call to \text{g05tac}. If \( \text{mode} = \text{Nag\_Generate\_Without\_Reference} \), \( r \) is not referenced and may be \text{NULL}.

On exit: if \( \text{mode} \neq \text{Nag\_Generate\_Without\_Reference} \), the reference vector.

6: \( \text{lr} \) – Integer

On entry: the dimension of the array \( r \).

\( \text{Suggested value:} \)

\[ \text{if } \text{mode} \neq \text{Nag\_Generate\_Without\_Reference}, \text{lr} = 22 + 20 \times \sqrt{m \times p \times (1 - p)}; \]

otherwise \( \text{lr} = 1. \)

\( \text{Constraints:} \)

\[ \text{if } \text{mode} = \text{Nag\_Initialize\_Reference} \text{ or } \text{Nag\_Initialize\_And\_Generate}, \]

\[ \text{lr} > \min(m, \text{int}[m \times p + 7.15 \times \sqrt{m \times p \times (1 - p)} + 1]) \]

\[ - \max(0, \text{int}[m \times p - 7.15 \times \sqrt{m \times p \times (1 - p)} - 7.15]) + 8 ; \]

\text{if } \text{mode} = \text{Nag\_Generate\_From\_Reference}, \text{lr} \text{ must remain unchanged from the previous call to } \text{g05tac}. \)

7: \( \text{state}[\text{dim}] \) – Integer

\( \text{Communication Array} \)

\( \text{Note:} \) the dimension, \( \text{dim} \), 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 \text{g05kfc} or \text{g05kge}.

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

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

8: \( x[n] \) – Integer

On exit: the \( n \) pseudorandom numbers from the specified binomial distribution.

9: \( \text{fail} \) – NagError *

\( \text{Input/Output} \)

The NAG error argument (see Section 3.6 in the Essential Introduction).
6 Error Indicators and Warnings

NE_ALLOC_FAIL
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
On entry, argument ⟨value⟩ had an illegal value.

NE_INT
On entry, lr is too small when mode = Nag_InitializeReference or Nag_InitializeAndGenerate:
lr = ⟨value⟩, minimum length required = ⟨value⟩.
On entry, m = ⟨value⟩.
Constraint: m ≥ 0.
On entry, n = ⟨value⟩.
Constraint: n ≥ 0.

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 the Essential Introduction 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 the Essential Introduction for further information.

NE_PREV_CALL
p or m is not the same as when r was set up in a previous call.
Previous value of p = ⟨value⟩ and p = ⟨value⟩.
Previous value of m = ⟨value⟩ and m = ⟨value⟩.

NE_REAL
On entry, p = ⟨value⟩.
Constraint: 0.0 ≤ p ≤ 1.0.

NE_REF_VEC
On entry, some of the elements of the array r have been corrupted or have not been initialized.

7 Accuracy
Not applicable.

8 Parallelism and Performance
nag_rand_binomial (g05tac) 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

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_rand_binomial (g05tac), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

```c
/* nag_rand_binomial (g05tac) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 9, 2009.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, lr, lstate;
    Integer *state = 0, *x = 0;
    /* NAG structures */
    NagError fail;
    Nag_ModeRNG mode;
    /* Double scalar and array declarations */
    double *r = 0;
    /* Set the distribution parameters */
    double p = 0.8e0;
    Integer m = 6000;
    /* Set the sample size */
    Integer n = 20;
    /* Choose the base generator */
    Naq_BaseRNG genid = Naq_Basic;
    Integer subid = 0;
    /* Set the seed */
    Integer seed[] = { 1762543 };
    Integer lseed = 1;
    /* Initialise the error structure */
    INIT_FAIL(fail);
    printf("nag_rand_binomial (g05tac) Example Program Results\n\n");
    /* Get the length of the state array */
    lstate = -1;
    nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    ....
```

```
```
/* Calculate the size of the reference vector */
lr = 22 + 20 * sqrt(m * p * (1 - p));

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

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

/* Generate the variates, initialising the reference vector
at the same time */
mode = Nag_InitializeAndGenerate;
nag_rand_binomial(mode, n, m, p, r, lr, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_binomial (g05tac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Display the variates */
for (i = 0; i < n; i++)
    printf(" %12"NAG_IFMT"\n", x[i]);

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

10.2 Program Data
None.

10.3 Program Results
nag_rand_binomial (g05tac) Example Program Results

4811
4761
4821
4826
4761
4800
4791
4825
4800
4714
4749
4780
4810