NAG C Library Function Document

nag_rngs_neg_bin (g05mcc)

1 Purpose

nag_rngs_neg_bin (g05mcc) generates a vector of pseudo-random integers from the discrete negative binomial distribution with parameter $m$ and probability $p$ of success at a trial.

2 Specification

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

3 Description

nag_rngs_neg_bin (g05mcc) generates $n$ integers $x_i$ from a discrete negative binomial distribution, where the probability of $x_i = I$ ($I$ successes before $m$ failures) is

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

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_neg_bin (g05mcc) with the same parameter value can then use this reference vector to generate further variates.

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_neg_bin (g05mcc).

4 References


5 Parameters

1: \texttt{mode} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} a code for selecting the operation to be performed by the function:

- \texttt{mode} = 0
  - Set up reference vector only.

- \texttt{mode} = 1
  - Generate variates using reference vector set up in a prior call to nag_rngs_neg_bin (g05mcc).

- \texttt{mode} = 2
  - Set up reference vector and generate variates.

- \texttt{mode} = 3
  - Generate variates without using the reference vector.

\textit{Constraint:} $0 \leq \texttt{mode} \leq 3$.

2: \texttt{m} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of failures, $m$, of the distribution.

\textit{Constraint:} $m \geq 0$. 
3:  \( p \) – double
   
   \textit{Input}
   
   \textit{On entry:} the parameter \( p \) of the negative binomial distribution representing the probability of success at a single trial.
   
   \textit{Constraint:} \( 0.0 \leq p < 1.0 \).

4:  \( n \) – Integer
   
   \textit{Input}
   
   \textit{On entry:} the number, \( n \), of pseudo-random numbers to be generated.
   
   \textit{Constraint:} \( n \geq 1 \).

5:  \( x[n] \) – Integer
   
   \textit{Output}
   
   \textit{On exit:} the \( n \) pseudo-random numbers from the specified negative binomial distribution.

6:  \( igen \) – Integer
   
   \textit{Input}
   
   \textit{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 \texttt{nag_rngs_init_repeatable} (g05kbc) or \texttt{nag_rngs_init_nonrepeatable} (g05kcc).

7:  \( iseed[4] \) – Integer
   
   \textit{Input/Output}
   
   \textit{On entry:} contains values which define the current state of the selected generator.
   
   \textit{On exit:} contains updated values defining the new state of the selected generator.

8:  \( r[dim] \) – double
   
   \textit{Input/Output}
   
   \textit{Note:} the dimension, \( dim \), of the array \( r \) must be at least \( 20 + (20 \sqrt{m \times p} + 30 \times p)/(1 - p) \) when \( mode < 3 \) and at least 1 otherwise.
   
   \textit{On exit:} the reference vector.

9:  \( fail \) – NagError *
   
   \textit{Input/Output}
   
   The NAG error parameter (see the Essential Introduction).

6  \textbf{Error Indicators and Warnings}

\textbf{NE_INT}

\textit{On entry,} \( \text{mode} = \langle \text{value} \rangle \).

\textit{Constraint:} \( 0 \leq \text{mode} \leq 3 \).

\textit{On entry,} \( m = \langle \text{value} \rangle \).

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

\textit{On entry,} \( n = \langle \text{value} \rangle \).

\textit{Constraint:} \( n \geq 1 \).

\textbf{NE_DIM_INFEASIBLE}

\( p \) and \( m \) are such that the reference vector length would exceed integer range. We recommend setting \( mode = 3 \).

\textbf{NE_PREV_CALL}

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

\textbf{NE_REAL}

\textit{On entry,} \( p < 0.0 \) or \( p \geq 1.0 \): \( p = \langle \text{value} \rangle \).
NE_BAD_PARAM

On entry, parameter \(h\) value \(i\) 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

None.

9 Example

The example program prints five pseudo-random integers from a negative binomial distribution with parameters \(m = 60\) and \(p = 0.999\), generated by a single call to nag_rngs_neg_bin (g05mcc), after initialisation by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

/* nag_rngs_neg_bin(g05mcc) Example Program. */
* Copyright 2001 Numerical Algorithms Group.
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

int main(void)
{
    /* Scalars */
    double p;
    Integer i, igen, m, n, nr;
    Integer exit_status=0;
    NagError fail;
    /* Arrays */
    double *r;
    Integer *x;
    Integer iseed[4];

    INIT_FAIL(fail);
    Vprintf("g05mcc Example Program Results\n\n");
    nr = 1;
    n = 20;

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

    /* Set the distribution parameter P */
    p = 0.999;

    // Code to generate random integers
    // using the negative binomial distribution
    // with parameters (m, p)

    // Print the generated integers
    Vprintf("\n\nGenerated integers:\n\n");
    Vprintf("\n\n\n");
    Vprintf("\n\n");
m = 60;
/* 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 = 3 because P close to 1 */
g05mcc(3, m, p, n, x, igen, iseed, r, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g05mcc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
for (i = 0; i < n; ++i)
{
    Vprintf("\%12ld", x[i]);
    Vprintf("\n");
}

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

9.2 Program Data

None.

9.3 Program Results

<table>
<thead>
<tr>
<th>g05mcc Example Program Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>49822</td>
</tr>
<tr>
<td>73300</td>
</tr>
<tr>
<td>57798</td>
</tr>
<tr>
<td>64791</td>
</tr>
<tr>
<td>73197</td>
</tr>
<tr>
<td>58394</td>
</tr>
<tr>
<td>61279</td>
</tr>
<tr>
<td>57579</td>
</tr>
<tr>
<td>56346</td>
</tr>
<tr>
<td>70705</td>
</tr>
<tr>
<td>59782</td>
</tr>
<tr>
<td>58087</td>
</tr>
<tr>
<td>61213</td>
</tr>
<tr>
<td>81333</td>
</tr>
<tr>
<td>47030</td>
</tr>
<tr>
<td>57082</td>
</tr>
<tr>
<td>67922</td>
</tr>
<tr>
<td>56991</td>
</tr>
<tr>
<td>46611</td>
</tr>
<tr>
<td>62691</td>
</tr>
</tbody>
</table>