NAG C Library Function Document

nag_rngs_hypergeometric (g05mlc)

1 Purpose

nag_rngs_hypergeometric (g05mlc) generates a vector of pseudo-random integers from the discrete hypergeometric distribution of the number of specified items in a sample of size \( l \), taken from a population of size \( n \) with \( m \) specified items in it.

2 Specification

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

3 Description

nag_rngs_hypergeometric (g05mlc) generates \( n \) integers \( x_i \) from a discrete hypergeometric distribution with mean \( \lambda \), where the probability of \( x_i = I \) is

\[
P(i = I) = \frac{l!m!(n-l)!(n-m)!}{I!(l-I)!(m-I)!(n-m-l+I)!n!}
\]

if \( I = \max(0, m + l - n), \ldots, \min(l, m) \),

\[ P(i = I) = 0 \]

otherwise.

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_hypergeometric (g05mlc) with the same parameter values can then use this reference vector to generate further variates. The reference array is generated by a recurrence relation if \( lm(n - l)(n - m) < 50n^3 \), otherwise Stirling’s approximation is used.

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_hypergeometric (g05mlc).

4 References


5 Parameters

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_hypergeometric (g05mlc).

mode = 2

Set up reference vector and generate variates.
mode = 3

Generate variates without using the reference vector.

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

2: **ns** – Integer  
*Input*

*On entry:* the sample size, $l$, of the hypergeometric distribution.

Constraint: $0 \leq \text{ns} \leq \text{np}$.

3: **np** – Integer  
*Input*

*On entry:* the population size, $n$, of the hypergeometric distribution.

Constraint: $\text{np} \geq 0$.

4: **m** – Integer  
*Input*

*On entry:* the number of specified items, $m$, of the hypergeometric distribution.

Constraint: $0 \leq \text{m} \leq \text{np}$.

5: **n** – Integer  
*Input*

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

Constraint: $n \geq 1$.

6: **x[n]** – Integer  
*Output*

*On exit:* the $n$ pseudo-random numbers from the specified hypergeometric distribution.

7: **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).

8: **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.

9: **r[dim]** – double  
*Input/Output*

*Note:* the dimension, $dim$, of the array $r$ must be at least

$20 + \sqrt{(\text{ns} \times \text{m} \times (\text{np} - \text{m}) \times (\text{np} - \text{ns})}/n^3$ when $\text{mode} < 3$ and at least 1 otherwise.

*On exit:* the reference vector.

10: **fail** – NagError *  
*Input/Output*

The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_INT**

*On entry,* $\text{mode} = \{\text{value}\}$.

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

*On entry,* $\text{np} = \{\text{value}\}$.

Constraint: $\text{np} \geq 0$. 
On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 1 \).

**NE_INT_2**

On entry, \( ns > np \) or \( ns < 0 \): \( ns = \langle \text{value} \rangle \), \( np = \langle \text{value} \rangle \).
On entry, \( m > np \) or \( m < 0 \): \( m = \langle \text{value} \rangle \), \( np = \langle \text{value} \rangle \).

**NE_PREV_CALL**

\( ns \) or \( np \) or \( m \) is not the same as when \( r \) was set up in a previous call or the data in \( r \) has been corrupted.

**NE_BAD_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) 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

None.

### 9 Example

The example program prints 20 pseudo-random integers from a hypergeometric distribution with \( l = 500 \), \( m = 900 \) and \( n = 1000 \), generated by a single call to nag_rgs_hypergeometric (g05mlc), after initialisation by nag_rgs_init_repeatable (g05kbc).

#### 9.1 Program Text

```c
/* nag_rgs_hypergeometric(g05mlc) 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, m, np, ns, n, nr;
    Integer exit_status=0;
    NagError fail;

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

    INIT_FAIL(fail);
    Vprintf("g05mlc Example Program Results\n\n");
    n = 20;
    nr = 2200;
```
/* 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 parameters NS, NP, M */
ns = 500;
m = 900;
np = 1000;
/* 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 */
g05mlc(2, ns, np, m, n, x, igen, iseed, r, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g05mlc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
for (i = 0; i < n; ++i)
{
    Vprintf("%12ld\n", x[i]);
}

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

9.2 Program Data
None.

9.3 Program Results

g05mlc Example Program Results

<table>
<thead>
<tr>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>444</td>
</tr>
<tr>
<td>458</td>
</tr>
<tr>
<td>449</td>
</tr>
<tr>
<td>453</td>
</tr>
<tr>
<td>458</td>
</tr>
<tr>
<td>449</td>
</tr>
<tr>
<td>451</td>
</tr>
<tr>
<td>449</td>
</tr>
<tr>
<td>448</td>
</tr>
<tr>
<td>456</td>
</tr>
<tr>
<td>450</td>
</tr>
<tr>
<td>449</td>
</tr>
<tr>
<td>451</td>
</tr>
<tr>
<td>462</td>
</tr>
<tr>
<td>442</td>
</tr>
<tr>
<td>448</td>
</tr>
<tr>
<td>455</td>
</tr>
<tr>
<td>448</td>
</tr>
<tr>
<td>441</td>
</tr>
<tr>
<td>452</td>
</tr>
</tbody>
</table>