

NAG C Library Function Document

nag_rngs_sample (g05nbc)

1 Purpose

nag_rngs_sample (g05nbc) selects a pseudo-random sample without replacement from an integer vector.

2 Specification

```
void nag_rngs_sample (const Integer ipop [], Integer n, Integer isampl [], Integer m,
                    Integer igen, Integer iseed [], NagError *fail)
```

3 Description

nag_rngs_sample (g05nbc) selects m elements from a population vector **ipop** of length n and places them in a sample vector **isampl**. Their order in **ipop** will be preserved in **isampl**. Each of the $\binom{n}{m}$ possible combinations of elements of **isampl** may be regarded as being equally probable.

For moderate or large values of n (greater than 75 say), it is theoretically impossible that all combinations of size m may occur, unless m is near 1 or near n . This is because $\binom{n}{m}$ exceeds the cycle length of nag_rngs_basic (g05kac) for all valid values of **igen**. For practical purposes this is irrelevant, as the time taken to generate all possible combinations is many millenia.

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_sample (g05nbc).

4 References

Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* (3rd Edition) Griffin

5 Parameters

- | | | |
|----|---|---------------|
| 1: | ipop [n] – const Integer | <i>Input</i> |
| | <i>On entry:</i> the population to be sampled. | |
| 2: | n – Integer | <i>Input</i> |
| | <i>On entry:</i> the number of elements in the population vector to be sampled. | |
| | <i>Constraint:</i> $\mathbf{n} \geq 1$. | |
| 3: | isampl [m] – Integer | <i>Output</i> |
| | <i>On entry:</i> the selected sample. | |
| 4: | m – Integer | <i>Input</i> |
| | <i>On entry:</i> the sample size. | |
| | <i>Constraint:</i> $1 \leq \mathbf{m} \leq \mathbf{n}$. | |

- 5: **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).
- 6: **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.
- 7: **fail** – NagError * *Input/Output*
The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = $\langle value \rangle$.
Constraint: **n** \geq 1.

NE_INT_2

On entry, **m** < 1 or **m** > **n**: **m** = $\langle value \rangle$, **n** = $\langle value \rangle$.

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

The time taken by `nag_rngs_sample` (g05nbc) is of order n .

In order to sample other kinds of vectors, or matrices of higher dimension, the following technique may be used:

- (a) set **ipop**[i] = i , for $i = 1, 2, \dots, n$;
- (b) use `nag_rngs_sample` (g05nbc) to take a sample from **ipop** and put it into **isampl**;
- (c) use the contents of **isampl** as a set of indices to access the relevant vector or matrix.

In order to divide a population into several groups, `nag_rngs_permute` (g05nac) is more efficient.

9 Example

In the example program random samples of size 1, 2, ..., 8 are selected from a vector containing the first eight positive integers in ascending order. The samples are generated and printed for each sample size by a call to `nag_rngs_sample` (g05nbc) after initialisation by `nag_rngs_init_repeatable` (g05kbc).

9.1 Program Text

```

/* nag_rngs_sample(g05nbc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

int main(void)
{
    /* Scalars */
    Integer i, igen, k, m, n;
    Integer exit_status=0;
    NagError fail;

    /* Arrays */
    Integer *ipop=0, *isampl=0;
    Integer iseed[4];

    INIT_FAIL(fail);
    Vprintf("g05nbc Example Program Results\n\n");

    n = 8;
    /* Allocate memory */
    if ( !(ipop = NAG_ALLOC(n, Integer)) ||
        !(isampl = NAG_ALLOC(n, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Initialise the seed to a repeatable sequence */
    iseed[0] = 1762543;
    iseed[1] = 9324783;
    iseed[2] = 1542344;
    iseed[3] = 742355;
    /* igen identifies the stream. */
    igen = 1;
    g05kbc(&igen, iseed);

    Vprintf(" Samples from the first %lld integers\n", n);
    Vprintf("\n");
    Vprintf(" Sample size      Values\n");
    for (i = 0; i < n; ++i)
        ipop[i] = i+1;
        for (m = 1; m <= n; ++m)
        {
            g05nbc(ipop, n, isampl, m, igen, iseed, &fail);
            if (fail.code != NE_NOERROR)
            {
                Vprintf("Error from g05nbc.\n%s\n", fail.message);
                exit_status = 1;
                goto END;
            }
            Vprintf("%6ld          ", m);
            for (k = 0; k < m; ++k)
            {
                Vprintf("%3ld%s", isampl[k], (k+1)%8 == 0 || k == m-1 ? "\n": " ");
            }
        }
    }
    END:
    if (ipop) NAG_FREE(ipop);
    if (isampl) NAG_FREE(isampl);
    return exit_status;
}

```

}

9.2 Program Data

None.

9.3 Program Results

g05nbc Example Program Results

Samples from the first 8 integers

Sample size	Values							
1	3							
2	2	5						
3	3	4	8					
4	2	4	5	6				
5	1	2	4	6	8			
6	1	2	3	5	7	8		
7	1	2	3	4	5	6	8	
8	1	2	3	4	5	6	7	8
