

NAG Library Function Document

nag_rand_matrix_multi_students_t (g05ryc)

1 Purpose

nag_rand_matrix_multi_students_t (g05ryc) sets up a reference vector and generates an array of pseudorandom numbers from a multivariate Student's t distribution with ν degrees of freedom, mean vector a and covariance matrix $\frac{\nu}{\nu-2}C$.

2 Specification

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

void nag_rand_matrix_multi_students_t (Nag_OrderType order, Nag_ModeRNG mode,
    Integer n, Integer df, Integer m, const double xmu[], const double c[],
    Integer pdc, double r[], Integer lr, Integer state[], double x[],
    Integer pdx, NagError *fail)
```

3 Description

When the covariance matrix is nonsingular (i.e., strictly positive definite), the distribution has probability density function

$$f(x) = \frac{\Gamma\left(\frac{\nu+m}{2}\right)}{(\pi\nu)^{m/2} \Gamma(\nu/2) |C|^{1/2}} \left[1 + \frac{(x-a)^T C^{-1} (x-a)}{\nu} \right]^{-\frac{(\nu+m)}{2}}$$

where m is the number of dimensions, ν is the degrees of freedom, a is the vector of means, x is the vector of positions and $\frac{\nu}{\nu-2}C$ is the covariance matrix.

The function returns the value

$$x = a + \sqrt{\frac{\nu}{s}} z$$

where z is generated by nag_rand_normal (g05skc) from a Normal distribution with mean zero and covariance matrix C and s is generated by nag_rand_chi_sq (g05sdc) from a χ^2 -distribution with ν degrees of freedom.

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_matrix_multi_students_t (g05ryc).

4 References

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

Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, Oxford

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

- 2: **mode** – Nag_ModeRNG *Input*
On entry: a code for selecting the operation to be performed by the function.
mode = Nag_InitializeReference
Set up reference vector only.
mode = Nag_GenerateFromReference
Generate variates using reference vector set up in a prior call to nag_rand_matrix_multi_students_t (g05ryc).
mode = Nag_InitializeAndGenerate
Set up reference vector and generate variates.
Constraint: **mode** = Nag_InitializeReference, Nag_GenerateFromReference or Nag_InitializeAndGenerate.
- 3: **n** – Integer *Input*
On entry: n , the number of random variates required.
Constraint: $n \geq 0$.
- 4: **df** – Integer *Input*
On entry: ν , the number of degrees of freedom of the distribution.
Constraint: $df \geq 3$.
- 5: **m** – Integer *Input*
On entry: m , the number of dimensions of the distribution.
Constraint: $m > 0$.
- 6: **xmu[m]** – const double *Input*
On entry: a , the vector of means of the distribution.
- 7: **c[dim]** – const double *Input*
Note: the dimension, dim , of the array **c** must be at least $pd\mathbf{c} \times \mathbf{m}$.
The (i, j) th element of the matrix C is stored in
 $\mathbf{c}[(j-1) \times pd\mathbf{c} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{c}[(i-1) \times pd\mathbf{c} + j - 1]$ when **order** = Nag_RowMajor.
On entry: matrix which, along with **df**, defines the covariance of the distribution. Only the upper triangle need be set.
Constraint: **c** must be positive semidefinite to *machine precision*.
- 8: **pd\mathbf{c}** – Integer *Input*
On entry: the stride separating row or column elements (depending on the value of **order**) in the array **c**.
Constraint: $pd\mathbf{c} \geq \mathbf{m}$.
- 9: **r[lr]** – double *Input/Output*
On entry: if **mode** = Nag_GenerateFromReference, the reference vector as set up by nag_rand_matrix_multi_students_t (g05ryc) in a previous call with **mode** = Nag_InitializeReference or Nag_InitializeAndGenerate.
On exit: if **mode** = Nag_InitializeReference or Nag_InitializeAndGenerate, the reference vector that can be used in subsequent calls to nag_rand_matrix_multi_students_t (g05ryc) with **mode** = Nag_GenerateFromReference.

- 10: **lr** – Integer *Input*
On entry: the dimension of the array **r**. If **mode** = Nag_GenerateFromReference, it must be the same as the value of **lr** specified in the prior call to nag_rand_matrix_multi_students_t (g05ryc) with **mode** = Nag_InitializeReference or Nag_InitializeAndGenerate.
Constraint: $lr \geq m \times (m + 1) + 2$.
- 11: **state**[*dim*] – Integer *Communication Array*
Note: the actual argument supplied must be the array **state** supplied to the initialization functions nag_rand_init_repeatable (g05kfc) or nag_rand_init_nonrepeatable (g05kgc).
On entry: contains information on the selected base generator and its current state.
On exit: contains updated information on the state of the generator.
- 12: **x**[*dim*] – double *Output*
Note: the dimension, *dim*, of the array **x** must be at least
 $\max(1, \mathbf{pdx} \times \mathbf{m})$ when **order** = Nag_ColMajor;
 $\max(1, \mathbf{n} \times \mathbf{pdx})$ when **order** = Nag_RowMajor.
Where $\mathbf{X}(i, j)$ appears in this document, it refers to the array element
 $\mathbf{x}[(j - 1) \times \mathbf{pdx} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{x}[(i - 1) \times \mathbf{pdx} + j - 1]$ when **order** = Nag_RowMajor.
On exit: the array of pseudorandom multivariate Student's *t* vectors generated by the function, with $\mathbf{X}(i, j)$ holding the *j*th dimension for the *i*th variate.
- 13: **pdx** – Integer *Input*
On entry: the stride used in the array **x**.
Constraints:
if **order** = Nag_ColMajor, $\mathbf{pdx} \geq \mathbf{n}$;
if **order** = Nag_RowMajor, $\mathbf{pdx} \geq \mathbf{m}$.
- 14: **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, **df** = $\langle value \rangle$.

Constraint: $\mathbf{df} \geq 3$.

On entry, **lr** is not large enough, **lr** = $\langle value \rangle$: minimum length required = $\langle value \rangle$.

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

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

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

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

NE_INT_2

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

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

On entry, $\mathbf{pdx} = \langle \text{value} \rangle$ and $\mathbf{m} = \langle \text{value} \rangle$.

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

On entry, $\mathbf{pdx} = \langle \text{value} \rangle$ and $\mathbf{n} = \langle \text{value} \rangle$.

Constraint: $\mathbf{pdx} \geq \mathbf{n}$.

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_INVALID_STATE

On entry, **state** vector has been corrupted or not initialized.

NE_POS_DEF

On entry, the covariance matrix C is not positive semidefinite to *machine precision*.

NE_PREV_CALL

\mathbf{m} is not the same as when \mathbf{r} was set up in a previous call.

Previous value of $\mathbf{m} = \langle \text{value} \rangle$ and $\mathbf{m} = \langle \text{value} \rangle$.

7 Accuracy

Not applicable.

8 Further Comments

The time taken by `nag_rand_matrix_multi_students_t (g05ryc)` is of order nm^3 .

It is recommended that the diagonal elements of C should not differ too widely in order of magnitude. This may be achieved by scaling the variables if necessary. The actual matrix decomposed is $C + E = LL^T$, where E is a diagonal matrix with small positive diagonal elements. This ensures that, even when C is singular, or nearly singular, the Cholesky factor L corresponds to a positive definite covariance matrix that agrees with C within *machine precision*.

9 Example

This example prints ten pseudorandom observations from a multivariate Student's t -distribution with ten degrees of freedom, means vector

$$\begin{bmatrix} 1.0 \\ 2.0 \\ -3.0 \\ 0.0 \end{bmatrix}$$

and \mathbf{c} matrix

$$\begin{bmatrix} 1.69 & 0.39 & -1.86 & 0.07 \\ 0.39 & 98.01 & -7.07 & -0.71 \\ -1.86 & -7.07 & 11.56 & 0.03 \\ 0.07 & -0.71 & 0.03 & 0.01 \end{bmatrix},$$

generated by `nag_rand_matrix_multi_students_t (g05ryc)`. All ten observations are generated by a single call to `nag_rand_matrix_multi_students_t (g05ryc)` with `mode = Nag_InitializeAndGenerate`. The random number generator is initialized by `nag_rand_init_repeatable (g05kfc)`.

9.1 Program Text

```

/* nag_rand_matrix_multi_students_t (g05ryc) Example Program.
 *
 * Copyright 2008, 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>

#define X(I, J) x[(order == Nag_ColMajor)?(J*pdx + I):(I*pdx + J)]
#define C(I, J) c[(order == Nag_ColMajor)?(J*pdx + I):(I*pdx + J)]

int main(void)
{
    /* Integer scalar and array declarations */
    Integer      exit_status = 0;
    Integer      i, j, lstate, lr, x_size;
    Integer      *state = 0;
    Integer      pdx;

    /* NAG structures */
    NagError     fail;
    Nag_ModeRNG  mode;

    /* Double scalar and array declarations */
    double       *r = 0, *x = 0;

    /* Use column major order */
    Nag_OrderType order = Nag_RowMajor;

    /* Set the number of variables and variates */
    Integer      m = 4;
    Integer      n = 10;

    /* Input the covariance matrix */
    double       c[] = { 1.69e0, 0.39e0, -1.86e0, 0.07e0,
                        0.39e0, 98.01e0, -7.07e0, -0.71e0,
                        -1.86e0, -7.07e0, 11.56e0, 0.03e0,
                        0.07e0, -0.71e0, 0.03e0, 0.01e0 };
    Integer      pdx = 4;

    /* Input the means */
    double       xmu[] = { 1.0e0, 2.0e0, -3.0e0, 0.0e0 };

    /* Set the degrees of freedom*/
    Integer      df = 10;

    /* Choose the base generator */
    Nag_BaseRNG  genid = Nag_Basic;
    Integer      subid = 0;

    /* Set the seed */
    Integer      seed[] = { 1762543 };
    Integer      lseed = 1;

    /* Initialise the error structure */
    INIT_FAIL(fail);

    printf("nag_rand_matrix_multi_students_t (g05ryc)  "
           "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;
}

pdx = (order == Nag_ColMajor)?n:m;
x_size = (order == Nag_ColMajor)?pdx * m:pdx * n;

/* Calculate the size of the reference vector */
lr = m*m+m+2;

/* Allocate arrays */
if (!(r = NAG_ALLOC(lr, double)) ||
    !(x = NAG_ALLOC(x_size, double)) ||
    !(state = NAG_ALLOC(lstate, 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;
}

/* Set up reference vector and generate N numbers */
mode = Nag_InitializeAndGenerate;
nag_rand_matrix_multi_students_t(order, mode, n, df, m, xmu, c, pdc, r,
                                  lr, state, x, pdx, &fail);
if (fail.code != NE_NOERROR)
{
    printf(
        "Error from nag_rand_matrix_multi_students_t (g05ryc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Display the variates */
for (i = 0; i < n; i++)
{
    printf(" ");
    for (j = 0; j < m; j++)
        printf("%9.4f%s", X(i, j), (j+1)%10?" ":"\n");
    if (m%10) printf("\n");
}

END:
if (r) NAG_FREE(r);
if (x) NAG_FREE(x);
if (state) NAG_FREE(state);

return exit_status;
}

```

9.2 Program Data

None.

9.3 Program Results

nag_rand_matrix_multi_students_t (g05ryc) Example Program Results

1.4957	-15.6226	-3.8101	0.1294
-1.0827	-6.7473	0.6696	-0.0391
2.1369	6.3861	-5.7413	0.0140
2.2481	-16.0417	-1.0982	0.1641
-0.2550	3.5166	-0.2541	-0.0592
0.9731	-4.3553	-4.4181	0.0043
0.7098	-3.4281	1.1741	0.0586
1.8827	23.2619	1.5140	-0.0704
0.9904	22.7479	0.1811	-0.0893
1.5026	2.7753	-2.2805	-0.0112
