

NAG Library Function Document

nag_multi_students_t (g01hdc)

1 Purpose

nag_multi_students_t (g01hdc) returns a probability associated with a multivariate Student's t -distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>

double nag_multi_students_t (Integer n, const Nag_TailProbability tail[],
    const double a[], const double b[], double nu, const double delta[],
    Nag_Boolean iscov, double rc[], Integer pdrc, double epsabs,
    double epsrel, Integer numsub, Integer nsampl, Integer fmax,
    double *errest, NagError *fail)
```

3 Description

A random vector $x \in \mathbb{R}^n$ that follows a Student's t -distribution with ν degrees of freedom and covariance matrix Σ has density:

$$\frac{\Gamma((\nu + n)/2)}{\Gamma(\nu/2)\nu^{n/2}\pi^{n/2}|\Sigma|^{1/2}\left[1 + \frac{1}{\nu}x^T\Sigma^{-1}x\right]^{(\nu+n)/2}},$$

and probability p given by:

$$p = \frac{\Gamma((\nu + n)/2)}{\Gamma(\nu/2)\sqrt{|\Sigma|}(\pi\nu)^n} \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_n}^{b_n} (1 + x^T\Sigma^{-1}x/\nu)^{-(\nu+n)/2} dx.$$

The method of calculation depends on the dimension n and degrees of freedom ν . The method of Dunnet and Sobel is used in the bivariate case if ν is a whole number. A Plackett transform followed by quadrature method is adopted in other bivariate cases and trivariate cases. In dimensions higher than three a number theoretic approach to evaluating multidimensional integrals is adopted.

Error estimates are supplied as the published accuracy in the Dunnet and Sobel case, a Monte-Carlo standard error for multidimensional integrals, and otherwise the quadrature error estimate.

A parameter δ allows for non-central probabilities. The number theoretic method is used if any δ is nonzero.

In cases other than the central bivariate with whole ν , nag_multi_students_t (g01hdc) attempts to evaluate probabilities within a requested accuracy $\max(\epsilon_a, \epsilon_r \times I)$, for an approximate integral value I , absolute accuracy ϵ_a and relative accuracy ϵ_r .

4 References

Dunnet C W and Sobel M (1954) A bivariate generalization of Student's t -distribution, with tables for certain special cases *Biometrika* **41** 153–169

Genz A and Bretz F (2002) Methods for the computation of multivariate t -probabilities *Journal of Computational and Graphical Statistics* (**11**) 950–971

5 Arguments

- 1: **n** – Integer *Input*
On entry: n , the number of dimensions.
Constraint: $1 < \mathbf{n} \leq 1000$.
- 2: **tail[n]** – const Nag_TailProbability *Input*
On entry: defines the calculated probability, set **tail**[$i - 1$] to:
tail[$i - 1$] = Nag_LowerTail
 If the i th lower limit a_i is negative infinity.
tail[$i - 1$] = Nag_UpperTail
 If the i th upper limit b_i is infinity.
tail[$i - 1$] = Nag_Central
 If both a_i and b_i are finite.
Constraint: **tail**[$i - 1$] = Nag_LowerTail, Nag_UpperTail or Nag_Central, for $i = 1, 2, \dots, \mathbf{n}$.
- 3: **a[n]** – const double *Input*
On entry: a_i , for $i = 1, 2, \dots, n$, the lower integral limits of the calculation.
 If **tail**[$i - 1$] = Nag_LowerTail, **a**[$i - 1$] is not referenced and the i th lower limit of integration is $-\infty$.
- 4: **b[n]** – const double *Input*
On entry: b_i , for $i = 1, 2, \dots, n$, the upper integral limits of the calculation.
 If **tail**[$i - 1$] = Nag_UpperTail, **b**[$i - 1$] is not referenced and the i th upper limit of integration is ∞ .
Constraint: if **tail**[$i - 1$] = Nag_Central, **b**[$i - 1$] $>$ **a**[$i - 1$].
- 5: **nu** – double *Input*
On entry: ν , the degrees of freedom.
Constraint: **nu** $>$ 0.0.
- 6: **delta[n]** – const double *Input*
On entry: **delta**[$i - 1$] the noncentrality parameter for the i th dimension, for $i = 1, 2, \dots, \mathbf{n}$; set **delta**[$i - 1$] = 0 for the central probability.
- 7: **iscov** – Nag_Boolean *Input*
On entry: set **iscov** = Nag_TRUE if the covariance matrix is supplied and **iscov** = Nag_FALSE if the correlation matrix is supplied.
- 8: **rc[n × pdrc]** – double *Input/Output*
Note: the (i, j) th element of the matrix is stored in **rc**[($i - 1$) × **pdrc** + $j - 1$].
On entry: the lower triangle of either the covariance matrix (if **iscov** = Nag_TRUE) or the correlation matrix (if **iscov** = Nag_FALSE). In either case the array elements corresponding to the upper triangle of the matrix need not be set.
On exit: the strict upper triangle of **rc** contains the correlation matrix used in the calculations.

- 9: **pdrc** – Integer *Input*
On entry: the stride separating matrix column elements in the array **rc**.
Constraint: **pdrc** \geq **n**.
- 10: **epsabs** – double *Input*
On entry: ϵ_a , the absolute accuracy requested in the approximation. If **epsabs** is negative, the absolute value is used.
Suggested value: 0.0.
- 11: **epsrel** – double *Input*
On entry: ϵ_r , the relative accuracy requested in the approximation. If **epsrel** is negative, the absolute value is used.
Suggested value: 0.001.
- 12: **numsub** – Integer *Input*
On entry: if quadrature is used, the number of sub-intervals used by the quadrature algorithm; otherwise **numsub** is not referenced.
Suggested value: 350.
Constraint: if referenced, **numsub** $>$ 0.
- 13: **nsampl** – Integer *Input*
On entry: if quadrature is used, **nsampl** is not referenced; otherwise **nsampl** is the number of samples used to estimate the error in the approximation.
Suggested value: 8
Constraint: if referenced, **nsampl** $>$ 0.
- 14: **fmax** – Integer *Input*
On entry: if a number theoretic approach is used, the maximum number of evaluations for each integrand function.
Suggested value: $1000 \times \mathbf{n}$
Constraint: if referenced, **fmax** \geq 1.
- 15: **errest** – double * *Output*
On exit: an estimate of the error in the calculated probability.
- 16: **fail** – NagError * *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.

NE_ARRAY_SIZE

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

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

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

Constraint: **fmax** ≥ 1 .

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

Constraint: $1 < \mathbf{n} \leq 1000$.

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

Constraint: **nsampl** ≥ 1 .

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

Constraint: **numsub** ≥ 1 .

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_ARRAY

On entry, the information supplied in **rc** is invalid.

NE_REAL

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

Constraint: degrees of freedom **nu** > 0.0 .

NE_REAL_2

On entry, $k = \langle value \rangle$.

Constraint: $\mathbf{b}[k - 1] > \mathbf{a}[k - 1]$ for a central probability.

7 Accuracy

An estimate of the error in the calculation is given by the value of **errest** on exit.

8 Parallelism and Performance

nag_multi_students_t (g01hdc) is not threaded by NAG in any implementation.

nag_multi_students_t (g01hdc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example prints two probabilities from the Student's t -distribution.

10.1 Program Text

```

/* nag_multi_students_t (g01hdc) Example Program.
 *
 * Copyright 2013 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

#define RC(I, J) rc[(I-1)*tdrc + J-1]

int main(void)
{
    /* Integer scalar and array declarations */
    Integer          exit_status = 0;
    Integer          fmax, i, ierr, iscov, j, tdrc, n, nsampl, numsub;

    /* Double scalar and array declarations */
    double           epsabs, epsrel, errest, nu, prob;
    double           *a = 0, *b = 0, *delta = 0, *rc = 0;

    /* NAG structures */
    Nag_TailProbability *tail = 0;
    NagError           fail;

    /* Character scalar and array declarations */
    char             nag_enum_arg[30+1];

    printf("nag_multi_students_t (g01hdc) Example Program Results\n\n");

    /* Skip heading in data file*/
    scanf("%*[\n] ");
    scanf("%"NAG_IFMT, &n);
    scanf("%30s%*[\n] ", nag_enum_arg);

    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    iscov = (Nag_Boolean)nag_enum_name_to_value(nag_enum_arg);

    tdrc = n;
    numsub = 200;
    nsampl = 8;
    fmax = 25000;
    epsabs = 0.0;
    epsrel = 1.0e-3;

    if (!(tail = NAG_ALLOC(n, Nag_TailProbability)) ||
        !(a = NAG_ALLOC(n, double)) ||
        !(b = NAG_ALLOC(n, double)) ||
        !(delta = NAG_ALLOC(n, double)) ||
        !(rc = NAG_ALLOC(tdrc*n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    while (1) {
        ierr = scanf("%*[\n] ");

        if (ierr == EOF) break;
        scanf("%lf%*[\n] ", &nu);
    }
}

```

```

/* read NAG enum member name as string and convert using
 * nag_enum_name_to_value (x04nac).
 */
for (j = 0; j < n; j++) {
    scanf("%30s", nag_enum_arg);
    tail[j] = (Nag_TailProbability) nag_enum_name_to_value(nag_enum_arg);
}
scanf("%*[\n]");

for (j = 0; j < n; j++)
    scanf("%lf", &a[j]);
scanf("%*[\n] ");

for (j = 0; j < n; j++)
    scanf("%lf", &b[j]);
scanf("%*[\n] ");

for (j = 0; j < n; j++)
    scanf("%lf", &delta[j]);
scanf("%*[\n] ");

for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
        scanf("%lf", &RC(i, j));
scanf("%*[\n] ");

/* nag_multi_students_t (g01hdc). Multivariate Student's t probability */
INIT_FAIL(fail);
prob = nag_multi_students_t(n, tail, a, b, nu, delta, iscov, rc, tdrc,
                             epsabs, epsrel, numsub, nsampl, fmax, &errest,
                             &fail);

if (fail.code == NE_NOERROR) {
    printf("%24s%24.7e\n", "Probability: ", prob);
    printf("%24s%24.2e\n\n", "Error estimate:", errest);
} else {
    printf("nag_multi_students_t (g01hdc) failed.\n%s\n", fail.message);
    exit_status = 1;
}
}

END:

NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(delta);
NAG_FREE(rc);
NAG_FREE(tail);

return exit_status;
}

```

10.2 Program Data

```

nag_multi_students_t (g01hdc) Example Program Data
  5                Nag_FALSE          : n, iscov
Example 1
10.0              : nu
Nag_UpperTail Nag_UpperTail
Nag_UpperTail Nag_UpperTail
Nag_UpperTail
-0.1 -0.1 -0.1 -0.1 -0.1 : lower bounds
 888  888  888  888  888 : upper bounds
 0.0  0.0  0.0  0.0  0.0 : delta
 1.0  0.75 0.75 0.75 0.75
 0.75 1.0  0.75 0.75 0.75
 0.75 0.75 1.0  0.75 0.75
 0.75 0.75 0.75 1.0  0.75
 0.75 0.75 0.75 0.75 1.0 : correlation matrix
Example 2

```

```
3.0 : nu
Nag_LowerTail Nag_LowerTail
Nag_LowerTail Nag_LowerTail
Nag_LowerTail : tails
888 888 888 888 888 : lower bounds
-0.1 -0.1 -0.1 -0.1 -0.1 : upper bounds
1.0 2.0 3.0 3.0 3.0 : delta
1.0 0.75 0.75 0.75 0.75
0.75 1.0 0.75 0.75 0.75
0.75 0.75 1.0 0.75 0.75
0.75 0.75 0.75 1.0 0.75
0.75 0.75 0.75 0.75 1.0 : correlation matrix
```

10.3 Program Results

nag_multi_students_t (g01hdc) Example Program Results

```
Probability:          3.0164222e-01
Error estimate:       1.09e-05

Probability:          8.6224881e-05
Error estimate:       7.30e-08
```
