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

```c
#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 \( \nu \) degrees of freedom and covariance matrix \( \Sigma \) has density:

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

and probability \( p \) given by:

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

The method of calculation depends on the dimension \( n \) and degrees of freedom \( \nu \). The method of Dunnet and Sobel is used in the bivariate case if \( \nu \) 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 \( \delta \) allows for non-central probabilities. The number theoretic method is used if any \( \delta \) is nonzero.

In cases other than the central bivariate with whole \( \nu \), 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 \( \epsilon_a \) and relative accuracy \( \epsilon_r \).

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

5 Arguments

1: \( n \) – Integer \hspace{1cm} \text{Input}

\text{On entry:} \ n, \ \text{the number of dimensions.}

\text{Constraint:} \ 1 < n \leq 1000.

2: \( \text{tail}[n] \) – const Nag_TailProbability \hspace{1cm} \text{Input}

\text{On entry:} \ \text{defines the calculated probability, set} \ \text{tail}[i - 1] \ \text{to:}

- \( \text{tail}[i - 1] = \text{Nag_LowerTail} \)
  - If the \( i \)th lower limit \( a_i \) is negative infinity.
- \( \text{tail}[i - 1] = \text{Nag_UpperTail} \)
  - If the \( i \)th upper limit \( b_i \) is infinity.
- \( \text{tail}[i - 1] = \text{Nag_Central} \)
  - If both \( a_i \) and \( b_i \) are finite.

\text{Constraint:} \ \text{tail}[i - 1] = \text{Nag_LowerTail}, \text{Nag_UpperTail} \ \text{or} \ \text{Nag_Central}, \ \text{for} \ i = 1, 2, \ldots, n.

3: \( a[n] \) – const double \hspace{1cm} \text{Input}

\text{On entry:} \ a_i, \ \text{for} \ i = 1, 2, \ldots, n, \ \text{the lower integral limits of the calculation.}

\text{If} \ \text{tail}[i - 1] = \text{Nag_LowerTail}, \ a[i - 1] \ \text{is not referenced and the} \ i \ \text{th lower limit of integration is} -\infty.

4: \( b[n] \) – const double \hspace{1cm} \text{Input}

\text{On entry:} \ b_i, \ \text{for} \ i = 1, 2, \ldots, n, \ \text{the upper integral limits of the calculation.}

\text{If} \ \text{tail}[i - 1] = \text{Nag_UpperTail}, \ b[i - 1] \ \text{is not referenced and the} \ i \ \text{th upper limit of integration is} \infty.

\text{Constraint:} \ \text{if} \ \text{tail}[i - 1] = \text{Nag_Central}, \ b[i - 1] > a[i - 1].

5: \( \nu \) – double \hspace{1cm} \text{Input}

\text{On entry:} \ \nu, \ \text{the degrees of freedom.}

\text{Constraint:} \ \nu > 0.0.

6: \( \text{delta}[n] \) – const double \hspace{1cm} \text{Input}

\text{On entry:} \ \text{delta}[i - 1] \ \text{the noncentrality parameter for the} \ i \ \text{th dimension, for} \ i = 1, 2, \ldots, n; \ \text{set} \ \text{delta}[i - 1] = 0 \ \text{for the central probability.}

7: \( \text{iscov} \) – Nag_Boolean \hspace{1cm} \text{Input}

\text{On entry:} \ \text{set} \ \text{iscov} = \text{Nag_TRUE} \ \text{if the covariance matrix is supplied and} \ \text{iscov} = \text{Nag_FALSE} \ \text{if the correlation matrix is supplied.}

8: \( \text{rc}[n \times \text{pdrc}] \) – double \hspace{1cm} \text{Input/Output}

\text{Note:} \ \text{the} \ (i, j) \ \text{th element of the matrix is stored in} \ \text{rc}[(i - 1) \times \text{pdrc} + j - 1].

\text{On entry:} \ \text{the lower triangle of either the covariance matrix (if} \ \text{iscov} = \text{Nag_TRUE} \ \text{or the correlation matrix (if} \ \text{iscov} = \text{Nag_FALSE). In either case the array elements corresponding to the upper triangle of the matrix need not be set.}

\text{On exit:} \ \text{the strict upper triangle of} \ \text{rc} \ \text{contains the correlation matrix used in the calculations.}
9:  pdrc – Integer
    On entry: the stride separating matrix column elements in the array rc.
    Constraint: pdrc ≥ n.

10:  epsabs – double
     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
     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
     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
     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
     On entry: if a number theoretic approach is used, the maximum number of evaluations for each integrand function.
     Suggested value: 1000 × n
     Constraint: if referenced, fmax ≥ 1.

15:  errest – double *
     On exit: an estimate of the error in the calculated probability.

16:  fail – NagError *
     The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL
    Dynamic memory allocation failed.
    See Section 3.2.1.2 in the Essential Introduction for further information.

NE_ARRAY_SIZE
    On entry, pdrc = ⟨value⟩ and n = ⟨value⟩.
    Constraint: pdrc ≥ n.
**NE_BAD_PARAM**

On entry, argument \(<value>\) had an illegal value.

**NE_INT**

On entry, \(f_{\text{max}} = \langle value \rangle\).
Constraint: \(f_{\text{max}} \geq 1\).

On entry, \(n = \langle value \rangle\).
Constraint: \(1 < n \leq 1000\).

On entry, \(nsampl = \langle value \rangle\).
Constraint: \(nsampl \geq 1\).

On entry, \(numsub = \langle value \rangle\).
Constraint: \(numsub \geq 1\).

**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.

An unexpected error has been triggered by this function. Please contact NAG. See Section 3.6.6 in the Essential Introduction for further information.

**NE_INVALID_ARRAY**

On entry, the information supplied in \(rc\) is invalid.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

**NE_REAL**

On entry, \(nu = \langle value \rangle\).
Constraint: degrees of freedom \(nu > 0\).

**NE_REAL_2**

On entry, \(k = \langle value \rangle\).
Constraint: \(b[k-1] > a[k-1]\) 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**

\texttt{nag\_multi\_students\_t} (\texttt{g01hdc}) is not threaded by NAG in any implementation.

\texttt{nag\_multi\_students\_t} (\texttt{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 X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also 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

```c
/* nag_multi_students_t (g01hdc) Example Program. */
/* Copyright 2014 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*/
    #ifdef _WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif

    #ifdef _WIN32
    scanf_s("%"NAG_IFMT, &n);
    #else
    scanf("%"NAG_IFMT, &n);
    #endif

    scanf("%30s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));

    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;
```

"g01 – Simple Calculations on Statistical Data"
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) {
    #ifdef _WIN32
        ierr = scanf_s("%*[\n ] ");
    #else
        ierr = scanf("%*[\n ] ");
    #endif
    if (ierr == EOF) break;
    #ifdef _WIN32
        scanf_s("%lf%*[\n ] ", &nu);
    #else
        scanf("%lf%*[\n ] ", &nu);
    #endif
    /* read NAG enum member name as string and convert using *
     * nag_enum_name_to_value (x04nac).
     */
    for (j = 0; j < n; j++) {
        #ifdef _WIN32
            scanf_s("%30s", nag_enum_arg, _countof(nag_enum_arg));
        #else
            scanf("%30s", nag_enum_arg);
        #endif
        tail[j] = (Nag_TailProbability) nag_enum_name_to_value(nag_enum_arg);
    }
    #ifdef _WIN32
        scanf_s("%*[\n ] ");
    #else
        scanf("%*[\n ] ");
    #endif
    for (j = 0; j < n; j++)
    #ifdef _WIN32
        scanf_s("%lf", &a[j]);
    #else
        scanf("%lf", &a[j]);
    #endif
    #ifdef _WIN32
        scanf_s("%*[\n ] ");
    #else
        scanf("%*[\n ] ");
    #endif
    for (j = 0; j < n; j++)
    #ifdef _WIN32
        scanf_s("%lf", &b[j]);
    #else
        scanf("%lf", &b[j]);
    #endif
    #ifdef _WIN32
        scanf_s("%*[\n ] ");
    #else
        scanf("%*[\n ] ");
    #endif
    for (j = 0; j < n; j++)
    #ifdef _WIN32
        scanf_s("%lf", &delta[j]);
    #else
        scanf("%lf", &delta[j]);
    #endif
```c
/* 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", "Error estimate:", errest);
} else {
    printf("nag_multi_students_t (g01hdc) failed.\n\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**

**Example 1**

<table>
<thead>
<tr>
<th>nu</th>
<th>tails</th>
<th>lower bounds</th>
<th>upper bounds</th>
<th>delta</th>
<th>correlation matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>Nag_UpperTail Nag_UpperTail</td>
<td>-0.1 -0.1 -0.1 -0.1 -0.1</td>
<td>888 888 888 888 888</td>
<td>0.0 0.0 0.0 0.0 0.0</td>
<td>0.75 0.75 0.75 0.75 0.75</td>
</tr>
</tbody>
</table>

**Example 2**

<table>
<thead>
<tr>
<th>nu</th>
<th>tails</th>
<th>lower bounds</th>
<th>upper bounds</th>
<th>delta</th>
<th>correlation matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>Nag_LowerTail Nag_LowerTail</td>
<td>Nag_LowerTail Nag_LowerTail</td>
<td>Nag_LowerTail : tails</td>
<td>Nag_LowerTail Nag_LowerTail</td>
<td>Nag_LowerTail : tails</td>
</tr>
</tbody>
</table>
g01hdc

NAG Library Manual

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      |