NAG Library Function Document

nag_prob_students_t (g01ebc)

1 Purpose

nag_prob_students_t (g01ebc) returns the lower tail, upper tail or two tail probability for the Student’s t-distribution with real degrees of freedom.

2 Specification

```c
#include <nag.h>
#include <nagg01.h>
double nag_prob_students_t (Nag_TailProbability tail, double t, double df, NagError *fail)
```

3 Description

The lower tail probability for the Student’s t-distribution with \( \nu \) degrees of freedom, \( P(T \leq t : \nu) \) is defined by:

\[
P(T \leq t : \nu) = \frac{\Gamma((\nu + 1)/2)}{\sqrt{\pi \nu \Gamma(\nu/2)}} \int_{-\infty}^{t} \left[ 1 + \frac{T^2}{\nu} \right]^{-(\nu+1)/2} dT, \quad \nu \geq 1.
\]

Computationally, there are two situations:

(i) when \( \nu < 20 \), a transformation of the beta distribution, \( P_\beta(B \leq \beta : a, b) \) is used

\[
P(T \leq t : \nu) = \frac{1}{2} P_\beta \left( B \leq \frac{\nu}{\nu + t^2} : \nu/2, \frac{1}{2} \right) \quad \text{when } t < 0.0
\]

or

\[
P(T \leq t : \nu) = \frac{1}{2} + \frac{1}{2} P_\beta \left( B \geq \frac{\nu}{\nu + t^2} : \nu/2, \frac{1}{2} \right) \quad \text{when } t > 0.0;
\]

(ii) when \( \nu \geq 20 \), an asymptotic normalizing expansion of the Cornish–Fisher type is used to evaluate the probability, see Hill (1970).

4 References


5 Arguments

1: tail - Nag_TailProbability

   On entry: indicates which tail the returned probability should represent.

   tail = Nag_UpperTail
   The upper tail probability is returned, i.e., \( P(T \geq t : \nu) \).

   tail = Nag_TwoTailSignif
   The two tail (significance level) probability is returned, i.e., \( P(T \geq |t| : \nu) + P(T \leq -|t| : \nu) \).
The two tail (confidence interval) probability is returned, i.e., $P(T \leq |t| : \nu) - P(T \leq -|t| : \nu)$.

The lower tail probability is returned, i.e., $P(T \leq t : \nu)$.

**Constraint:** tail = Nag_UpperTail, Nag_TwoTailSignif, Nag_TwoTailConfid or Nag_LowerTail.

2: $t$ – double

*Input*

*On entry:* $t$, the value of the Student’s $t$ variate.

3: $df$ – double

*Input*

*On entry:* $\nu$, the degrees of freedom of the Student’s $t$-distribution.

*Constraint:* $df \geq 1.0$.

4: $fail$ – 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.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument *value* 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 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_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_ARG_LT**

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

*Constraint:* $df \geq 1.0$.

## 7 Accuracy

The computed probability should be accurate to five significant places for reasonable probabilities but there will be some loss of accuracy for very low probabilities (less than $10^{-10}$), see Hastings and Peacock (1975).

## 8 Parallelism and Performance

Not applicable.
Further Comments

The probabilities could also be obtained by using the appropriate transformation to a beta distribution (see Abramowitz and Stegun (1972)) and using nag_prob_beta_dist (g01eec). This function allows you to set the required accuracy.

Example

This example reads values from, and degrees of freedom for Student’s $t$-distributions along with the required tail. The probabilities are calculated and printed until the end of data is reached.

Program Text

```c
/* nag_prob_students_t (g01ebc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * *
 * Mark 4, 1996.
 * Mark 5 revised, 1998.
 * Mark 7 revised, 2001.
 * *
 * #include <nag.h>
 * #include <stdio.h>
 * #include <nag_stdlib.h>
 * #include <nagg01.h>
 *
 int main(void)
 {
 Integer exit_status = 0;
 double df, prob, t;
 int i;
 static Nag_TailProbability tail[4] = { Nag_LowerTail, Nag_UpperTail,
 Nag_TwoTailSignif, Nag_TwoTailConfid };
 static const char *tailmess[] = { "Nag_LowerTail", "Nag_UpperTail",
 "Nag_TwoTailSignif", "Nag_TwoTailConfid" };
 NagError fail;
 INIT_FAIL(fail);
 printf("nag_prob_students_t (g01ebc) Example Program Results\n\n");
 /* Skip heading in data file */
 #ifdef _WIN32
 scanf_s("%*[^
");
#else
 scanf("%*[^
");
#endif
 printf(" t df prob tail \n");
 #ifdef _WIN32
 while (scanf_s("%lf %lf %d\n", &t, &df, &i) != EOF)
#else
 while (scanf("%lf %lf %d\n", &t, &df, &i) != EOF)
#endif
 { /* nag_prob_students_t (g01ebc). *
 * Probabilities for Student’s t-distribution *
 */
 prob = nag_prob_students_t(tail[i], t, df, &fail);
 if (fail.code != NE_NOERROR)
 { printf("Error from nag_prob Students_t (g01ebc).\n\n", fail.message);
 exit_status = 1;
 goto END;
 }
 printf(" %6.3f%8.3f%8.4f %s\n", t, df, prob, tailmess[i]);

 END:
 exit_status = exit_status + fail.code;
 if (exit_status) { exit(-1); } /* fail code or program failure */
 return exit_status;

 END:
 exit_status = exit_status + fail.code;
 if (exit_status) { exit(-1); } /* fail code or program failure */
 return exit_status;
```
10.2 Program Data

nag_prob_students_t (g01ebc) Example Program Data
0.85 20.0 0
0.85 20.0 2
0.85 20.0 3
0.85 20.0 1

10.3 Program Results

nag_prob_students_t (g01ebc) Example Program Results

<table>
<thead>
<tr>
<th>t</th>
<th>df</th>
<th>prob</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.85</td>
<td>20.0</td>
<td>0.7973</td>
<td>Nag_LowerTail</td>
</tr>
<tr>
<td>0.85</td>
<td>20.0</td>
<td>0.4054</td>
<td>Nag_TwoTailSignif</td>
</tr>
<tr>
<td>0.85</td>
<td>20.0</td>
<td>0.5946</td>
<td>Nag_TwoTailConfid</td>
</tr>
<tr>
<td>0.85</td>
<td>20.0</td>
<td>0.2027</td>
<td>Nag_UpperTail</td>
</tr>
</tbody>
</table>