NAG Library Function Document

nag_prob_students_t_vector (g01sbc)

1 Purpose

nag_prob_students_t_vector (g01sbc) returns a number of one or two tail probabilities for the Student’s
$t$-distribution with real degrees of freedom.

2 Specification

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

void nag_prob_students_t_vector (Integer ltail,
     const Nag_TailProbability tail[],
     Integer lt,
     const double t[],
     Integer ldf,
     const double df[],
     double p[],
     Integer ivalid[],
     NagError *fail)
```

3 Description

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

$$P(T_i \leq t_i : \nu_i) = \frac{\Gamma\left((\nu_i + 1)/2\right)}{\sqrt{\pi\nu_i}} \int_{-\infty}^{t_i} \left[1 + \frac{T_i^2}{\nu_i}\right]^{-\left(\nu_i + 1\right)/2} \, dT_i, \quad \nu_i \geq 1.$$ 

Computationally, there are two situations:

(i) when $\nu_i < 20$, a transformation of the beta distribution, $P_{\beta_i}(B_i \leq \beta_i : a_i, b_i)$ is used

$$P(T_i \leq t_i : \nu_i) = \frac{1}{2}P_{\beta_i}\left(B_i \leq \frac{\nu_i}{\nu_i + t_i^2} : \nu_i/2, \frac{1}{2}\right) \quad \text{when } t_i < 0.0$$

or

$$P(T_i \leq t_i : \nu_i) = \frac{1}{2} + \frac{1}{2}P_{\beta_i}\left(B_i \geq \frac{\nu_i}{\nu_i + t_i^2} : \nu_i/2, \frac{1}{2}\right) \quad \text{when } t_i > 0.0;$$

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

The input arrays to this function are designed to allow maximum flexibility in the supply of vector
arguments by re-using elements of any arrays that are shorter than the total number of evaluations
required. See Section 2.6 in the g01 Chapter Introduction for further information.

4 References

Publications


5 Arguments

1: \( \text{ltail} \) – Integer  
\( \text{Input} \)

\textit{On entry:} the length of the array \( \text{tail} \).

\textit{Constraint:} \( \text{ltail} > 0 \).

2: \( \text{tail[ltail]} \) – const Nag_TailProbability  
\( \text{Input} \)

\textit{On entry:} indicates which tail the returned probabilities should represent. For \( j = (i - 1) \mod \text{ltail} \), for \( i = 1, 2, \ldots, \max(\text{ltail}, \text{lt}, \text{ldf}) \):

\( \text{tail}[j] = \text{Nag_LowerTail} \)
\( \text{The lower tail probability is returned, i.e., } p_i = P(T_i \leq t_i : \nu_i). \)

\( \text{tail}[j] = \text{Nag_UpperTail} \)
\( \text{The upper tail probability is returned, i.e., } p_i = P(T_i \geq t_i : \nu_i). \)

\( \text{tail}[j] = \text{Nag_TwoTailConfid} \)
\( \text{The two tail (confidence interval) probability is returned, i.e., } p_i = P(T_i \leq |t_i| : \nu_i) - P(T_i \leq -|t_i| : \nu_i). \)

\( \text{tail}[j] = \text{Nag_TwoTailSignif} \)
\( \text{The two tail (significance level) probability is returned, i.e., } p_i = P(T_i \geq |t_i| : \nu_i) + P(T_i \leq -|t_i| : \nu_i). \)

\textit{Constraint:} \( \text{tail}[j - 1] = \text{Nag_LowerTail}, \text{Nag_UpperTail}, \text{Nag_TwoTailConfid} \) or \( \text{Nag_TwoTailSignif} \), for \( j = 1, 2, \ldots, \text{ltail} \).

3: \( \text{lt} \) – Integer  
\( \text{Input} \)

\textit{On entry:} the length of the array \( t \).

\textit{Constraint:} \( \text{lt} > 0 \).

4: \( \text{t[lt]} \) – const double  
\( \text{Input} \)

\textit{On entry:} \( t_i \), the values of the Student’s \( t \) variates with \( t_i = t[j], j = (i - 1) \mod \text{lt} \).

5: \( \text{ldf} \) – Integer  
\( \text{Input} \)

\textit{On entry:} the length of the array \( df \).

\textit{Constraint:} \( \text{ldf} > 0 \).

6: \( \text{df[ldf]} \) – const double  
\( \text{Input} \)

\textit{On entry:} \( \nu_i \), the degrees of freedom of the Student’s \( t \)-distribution with \( \nu_i = df[j], j = (i - 1) \mod \text{ldf} \).

\textit{Constraint:} \( df[j - 1] \geq 1.0 \), for \( j = 1, 2, \ldots, \text{ldf} \).

7: \( \text{p[dim]} \) – double  
\( \text{Output} \)

\textit{Note:} the dimension, \( \text{dim} \), of the array \( p \) must be at least \( \max(\text{ltail}, \text{lt}, \text{ldf}) \).

\textit{On exit:} \( p_i \), the probabilities for the Student’s \( t \) distribution.

8: \( \text{invalid[dim]} \) – Integer  
\( \text{Output} \)

\textit{Note:} the dimension, \( \text{dim} \), of the array \( \text{invalid} \) must be at least \( \max(\text{ltail}, \text{lt}, \text{ldf}) \).

\textit{On exit:} \( \text{invalid}[i - 1] \) indicates any errors with the input arguments, with \( \text{invalid}[i - 1] = 0 \)
\( \text{No error.} \)
ivalid\[i – 1\] = 1
On entry, invalid value supplied in tail when calculating \( p_i \).
ivalid\[i – 1\] = 2
On entry, \( \nu_i < 1.0 \).

9: 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, array size = \( \langle \text{value} \rangle \).
Constraint: \( \text{ldf} > 0 \).
On entry, array size = \( \langle \text{value} \rangle \).
Constraint: \( \text{lt} > 0 \).
On entry, array size = \( \langle \text{value} \rangle \).
Constraint: \( \text{ltail} > 0 \).

NE_BAD_PARAM
On entry, argument \( \langle \text{value} \rangle \) 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 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.

NW_IVALID
On entry, at least one value of tail or df was invalid.
Check invalid for more information.

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_vector (g01sec). 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.

10.1 Program Text

```c
/* nag_prob_students_t_vector (g01sbc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011.
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdbin.h>
#include <nagg01.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer ltail, lt, ldf, i, lout;
    Integer *ivalid = 0;
    Integer exit_status = 0;
    /* NAG structures */
    NagError fail;
    Nag_TailProbability *tail = 0;
    /* Double scalar and array declarations */
    double *t = 0, *df = 0, *p = 0;
    /* Character scalar and array declarations */
    char ctail[40];
    /* Initialise the error structure to print out any error messages */
    INIT_FAIL(fail);
    printf("nag_prob_students_t_vector (g01sbc) Example Program Results\n\n");
    /* Skip heading in data file*/
    #ifdef _WIN32
        scanf_s("%*[\n ] ");
    #else
        scanf("%*[\n ] ");
    #endif
    /* Read in the input vectors */
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n ] ", &ltail);
    #else
    scanf("%"NAG_IFMT"%*[\n ] ", &ltail);
    #endif
    if (!(tail = NAG_ALLOC(ltail, Nag_TailProbability))) {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 0; i < ltail; i++) {
        #ifdef _WIN32
        scanf_s("%39s", ctail, _countof(ctail));
        #else
```
```c
scanf("%39s", ctail);
#endif
tail[i] = (Nag_TailProbability) nag_enum_name_to_value(ctail);
#endif
#ifdef _WIN32
scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[\n] ", &lt);
#else
scanf("%"NAG_IFMT"%*[\n] ", &lt);
#endif
if (!(t = NAG_ALLOC(lt, double))) {
    printf("Allocation failure
");
    exit_status = -1;
    goto END;
}
for (i = 0; i < lt; i++)
#ifdef _WIN32
    scanf_s("%lf", &t[i]);
#else
    scanf("%lf", &t[i]);
#endif
#ifdef _WIN32
scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif
#ifdef _WIN32
scanf_s("%"NAG_IFMT"%*[\n] ", &ldf);
#else
scanf("%"NAG_IFMT"%*[\n] ", &ldf);
#endif
if (!(df = NAG_ALLOC(ldf, double))) {
    printf("Allocation failure
");
    exit_status = -1;
    goto END;
}
for (i = 0; i < ldf; i++)
#ifdef _WIN32
    scanf_s("%lf", &df[i]);
#else
    scanf("%lf", &df[i]);
#endif
#ifdef _WIN32
scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif
/* Allocate memory for output */
lout = MAX(ltail, MAX(lt, ldf));
if (!(p = NAG_ALLOC(lout, double)) ||
    !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure
");
    exit_status = -1;
    goto END;
}
/* Calculate probability */
nag_prob_students_t_vector(ltail, tail, lt, t, ldf, df, p, ivalid, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_prob_students_t_vector (g01sbc).\n%s\n", fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID) goto END;
}
```
} /* Display title */
printf(" %7s %6.3f %6.1f %6.4f %3"NAG_IFMT"
",
    nag_enum_value_to_name(tail[i%ltail]), t[i%lt], df[i%ldf],
    p[i], ivalid[i]);
END:
    NAG_FREE(tail);
    NAG_FREE(t);
    NAG_FREE(df);
    NAG_FREE(p);
    NAG_FREE(ivalid);
    return(exit_status);
}

10.2 Program Data

nag_prob_students_t_vector (g01sbc) Example Program Data
4 :: ltail
Nag_LowerTail Nag_TwoTailSignif Nag_TwoTailConfid Nag_UpperTail :: tail
1 :: lt
0.85 :: t
1.0 :: ldf
20.0 :: df

10.3 Program Results

nag_prob_students_t_vector (g01sbc) Example Program Results

<table>
<thead>
<tr>
<th>tail</th>
<th>t</th>
<th>df</th>
<th>p</th>
<th>ivalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nag_LowerTail</td>
<td>0.85</td>
<td>20.0</td>
<td>0.7973</td>
<td>0</td>
</tr>
<tr>
<td>Nag_TwoTailSignif</td>
<td>0.85</td>
<td>20.0</td>
<td>0.4054</td>
<td>0</td>
</tr>
<tr>
<td>Nag_TwoTailConfid</td>
<td>0.85</td>
<td>20.0</td>
<td>0.5946</td>
<td>0</td>
</tr>
<tr>
<td>Nag_UpperTail</td>
<td>0.85</td>
<td>20.0</td>
<td>0.2027</td>
<td>0</td>
</tr>
</tbody>
</table>