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

```
#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 ν_i degrees of freedom, $P(T_i \leq t_i : \nu_i)$ is defined by:

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

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

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

or

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

(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

Abramowitz M and Stegun I A (1972) *Handbook of Mathematical Functions* (3rd Edition) Dover Publications

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworth

Hill G W (1970) Student's t-distribution *Comm. ACM* **13(10)** 617–619

5 Arguments

1: **ltail** – Integer *Input*

On entry: the length of the array **tail**.

Constraint: **ltail** > 0.

2: **tail**[**ltail**] – const Nag_TailProbability *Input*

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

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

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

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

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

Constraint: **tail**$[j - 1]$ = Nag_LowerTail, Nag_UpperTail, Nag_TwoTailConfid or Nag_TwoTailSignif, for $j = 1, 2, \ldots,$ **ltail**.

3: **lt** – Integer *Input*

On entry: the length of the array **t**.

Constraint: **lt** > 0.

4: **t**[**lt**] – const double *Input*

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

5: **ldf** – Integer *Input*

On entry: the length of the array **df**.

Constraint: **ldf** > 0.

6: **df**[**ldf**] – const double *Input*

On entry: ν_i, the degrees of freedom of the Student's t-distribution with $\nu_i = \mathbf{df}[j]$, $j = (i - 1)$ mod **ldf**.

Constraint: **df**$[j - 1] \geq 1.0$, for $j = 1, 2, \ldots,$ **ldf**.

7: **p**[*dim*] – double *Output*

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

On exit: p_i, the probabilities for the Student's t distribution.

8: **ivalid**[*dim*] – Integer *Output*

Note: the dimension, *dim*, of the array **ivalid** must be at least $\max(\mathbf{ltail}, \mathbf{lt}, \mathbf{ldf})$.

On exit: **ivalid**$[i - 1]$ indicates any errors with the input arguments, with

ivalid$[i - 1] = 0$
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 * *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, array size = ⟨*value*⟩.
Constraint: **ldf** > 0.

On entry, array size = ⟨*value*⟩.
Constraint: **lt** > 0.

On entry, array size = ⟨*value*⟩.
Constraint: **ltail** > 0.

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.

NW_IVALID

On entry, at least one value of **tail** or **df** was invalid.
Check **ivalid** 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.

9 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.

10 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

```
/* nag_prob_students_t_vector (g01sbc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.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*/
  scanf("%*[^\n] ");

  /* Read in the input vectors */
  scanf("%ld%*[^\n] ", &ltail);
  if (!(tail = NAG_ALLOC(ltail, Nag_TailProbability))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ltail; i++) {
    scanf("%39s", ctail);
    tail[i] = (Nag_TailProbability) nag_enum_name_to_value(ctail);
  }
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &lt);
  if (!(t = NAG_ALLOC(lt, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lt; i++)
    scanf("%lf", &t[i]);
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &ldf);
  if (!(df = NAG_ALLOC(ldf, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ldf; i++)
    scanf("%lf", &df[i]);
  scanf("%*[^\n] ");


  /* Allocate memory for output */
  lout = MAX(ltail,MAX(lt,ldf));
  if (!(p = NAG_ALLOC(lout, double)) ||
      !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure\n");
    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("         tail           t         df        p     ivalid\n");
  printf("-----------------------------------------------------------\n");

  /* Display results */
  for (i = 0; i < lout; i++)
    printf(" %17s    %6.3f    %6.1f    %6.4f    %3ld\n",
           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                                                                       :: ldf
20.0                                                                    :: df
```

10.3 Program Results

```
nag_prob_students_t_vector (g01sbc) Example Program Results

          tail           t         df        p     ivalid
-----------------------------------------------------------
     Nag_LowerTail     0.850      20.0    0.7973      0
 Nag_TwoTailSignif     0.850      20.0    0.4054      0
 Nag_TwoTailConfid     0.850      20.0    0.5946      0
     Nag_UpperTail     0.850      20.0    0.2027      0
```