NAG Library Function Document

nag_deviates_students_t_vector (g01tbc)

1 Purpose

nag_deviates_students_t_vector (g01tbc) returns a number of deviates associated with given probabilities of Student's t-distribution with real degrees of freedom.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
void nag_deviates_students_t_vector (Integer ltail,
     const Nag_TailProbability tail[], Integer lp, const double p[],
     Integer ldf, const double df[], double t[], Integer ivalid[],
     NagError *fail)
```

3 Description

The deviate, t_{p_i} associated with the lower tail probability, p_i, of the Student's t-distribution with ν_i degrees of freedom is defined as the solution to

$$P\left(T_i < t_{p_i} : \nu_i\right) = p_i = \frac{\Gamma((\nu_i+1)/2)}{\sqrt{\nu_i\pi}\Gamma(\nu_i/2)}\int_{-\infty}^{t_{p_i}}\left(1+\frac{T_i^2}{\nu_i}\right)^{-(\nu_i+1)/2} dT_i, \quad \nu_i \geq 1;\ -\infty < t_{p_i} < \infty.$$

For $\nu_i = 1$ or 2 the integral equation is easily solved for t_{p_i}.

For other values of $\nu_i < 3$ a transformation to the beta distribution is used and the result obtained from nag_deviates_beta (g01fec).

For $\nu_i \geq 3$ an inverse asymptotic expansion of Cornish–Fisher type is used. The algorithm is described by 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

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 supplied probabilities represent. For $j = (i-1) \bmod$ **ltail**, for $i = 1, 2, \ldots, \max(\textbf{ltail}, \textbf{lp}, \textbf{ldf})$:

tail[j] = Nag_LowerTail
The lower tail probability, i.e., $p_i = P\left(T_i \leq t_{p_i} : \nu_i\right)$.

tail$[j]$ = Nag_UpperTail
The upper tail probability, i.e., $p_i = P\left(T_i \geq t_{p_i} : \nu_i\right)$.

tail$[j]$ = Nag_TwoTailConfid
The two tail (confidence interval) probability,
i.e., $p_i = P\left(T_i \leq \left|t_{p_i}\right| : \nu_i\right) - P\left(T_i \leq -\left|t_{p_i}\right| : \nu_i\right)$.

tail$[j]$ = Nag_TwoTailSignif
The two tail (significance level) probability,
i.e., $p_i = P\left(T_i \geq \left|t_{p_i}\right| : \nu_i\right) + P\left(T_i \leq -\left|t_{p_i}\right| : \nu_i\right)$.

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

3: **lp** – Integer *Input*

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

Constraint: **lp** > 0.

4: **p**[**lp**] – const double *Input*

On entry: p_i, the probability of the required Student's t-distribution as defined by **tail** with $p_i =$ **p**$[j]$, $j = (i-1) \bmod$ **lp**.

Constraint: $0.0 <$ **p**$[j-1] < 1.0$, for $j = 1, 2, \ldots,$ **lp**.

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 $\nu_i =$ **df**$[j]$, $j = (i-1) \bmod$ **ldf**.

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

7: **t**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **t** must be at least max(**ltail**, **lp**, **ldf**).

On exit: t_{p_i}, the deviates for the Student's t-distribution.

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

Note: the dimension, *dim*, of the array **ivalid** must be at least max(**ltail**, **lp**, **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 t_{p_i}.

ivalid$[i-1] = 2$
On entry, $p_i \leq 0.0$,
or $p_i \geq 1.0$.

ivalid$[i-1] = 3$
On entry, $\nu_i < 1.0$.

ivalid$[i-1] = 4$
The solution has failed to converge. The result returned should represent an approximation to the solution.

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.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_ARRAY_SIZE

On entry, array size $= \langle value \rangle$.
Constraint: **ldf** > 0.

On entry, array size $= \langle value \rangle$.
Constraint: **lp** > 0.

On entry, array size $= \langle value \rangle$.
Constraint: **ltail** > 0.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ 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**, **p** or **df** was invalid, or the solution failed to converge. Check **ivalid** for more information.

7 Accuracy

The results should be accurate to five significant digits, for most argument values. The error behaviour for various argument values is discussed in Hill (1970).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The value t_{p_i} may be calculated by using a transformation to the beta distribution and calling nag_deviates_beta_vector (g01tec). This function allows you to set the required accuracy.

10 Example

This example reads the probability, the tail that probability represents and the degrees of freedom for a number of Student's t-distributions and computes the corresponding deviates.

10.1 Program Text

```
/* nag_deviates_students_t_vector (g01tbc) Example Program.
 *
 * Copyright 2014 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, lp, ldf, i, lout;
  Integer *ivalid = 0;
  Integer exit_status = 0;

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

  /* Double scalar and array declarations */
  double *p = 0, *df = 0, *t = 0;

  /* Character scalar and array declarations */
  char ctail[40];

  /* Initialise the error structure to print out any error messages */
  INIT_FAIL(fail);

  printf("nag_deviates_students_t_vector (g01tbc) 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
    scanf("%39s", ctail);
#endif
    tail[i] = (Nag_TailProbability) nag_enum_name_to_value(ctail);
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
```

```
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &lp);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &lp);
#endif
  if (!(p = NAG_ALLOC(lp, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lp; i++)
#ifdef _WIN32
    scanf_s("%lf", &p[i]);
#else
    scanf("%lf", &p[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\n");
    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(lp,ldf));
  if (!(t = NAG_ALLOC(lout, double)) ||
      !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Calculate probability */
  nag_deviates_students_t_vector(ltail, tail, lp, p, ldf, df, t, ivalid,
                                 &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_deviates_students_t_vector (g01tbc).\n%s\n",
            fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID) goto END;
  }

  /* Display title */
  printf("          tail           p         df        t     ivalid\n");
  printf("-------------------------------------------------------------\n");
```

```
  /* Display results */
  for (i = 0; i < lout; i++)
    printf(" %17s    %6.3f    %6.1f   %7.4f    %3"NAG_IFMT"\n",
           nag_enum_value_to_name(tail[i%ltail]), p[i%lp], df[i%ldf], t[i],
           ivalid[i]);

 END:
  NAG_FREE(tail);
  NAG_FREE(p);
  NAG_FREE(df);
  NAG_FREE(t);
  NAG_FREE(ivalid);

  return(exit_status);
}
```

10.2 Program Data

```
nag_deviates_students_t_vector (g01tbc) Example Program Data
3                                                                   :: ltail
Nag_TwoTailSignif  Nag_LowerTail  Nag_TwoTailConfid                 :: tail
3                                                                   :: lp
0.01 0.01 0.99                                                      :: p
3                                                                   :: ldf
20.0 7.5 45.0                                                       :: df
```

10.3 Program Results

```
nag_deviates_students_t_vector (g01tbc) Example Program Results

          tail           p         df        t     ivalid
-------------------------------------------------------------
 Nag_TwoTailSignif     0.010      20.0    2.8453      0
     Nag_LowerTail     0.010       7.5   -2.9431      0
 Nag_TwoTailConfid     0.990      45.0    2.6896      0
```