NAG Library Function Document

nag_deviates_f_vector (g01tdc)

1 Purpose

nag_deviates_f_vector (g01tdc) returns a number of deviates associated with given probabilities of the F or variance-ratio distribution with real degrees of freedom.

2 Specification

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

```
void nag_deviates_f_vector (Integer ltail, const Nag_TailProbability tail[],
     Integer lp, const double p[], Integer ldf1, const double df1[],
     Integer ldf2, const double df2[], double f[], Integer ivalid[],
     NagError *fail)
```

3 Description

The deviate, f_{p_i}, associated with the lower tail probability, p_i, of the F-distribution with degrees of freedom u_i and v_i is defined as the solution to

$$P\left(F_i \leq f_{p_i} : u_i, v_i\right) = p_i = \frac{u_i^{\frac{1}{2}u_i} v_i^{\frac{1}{2}v_i} \Gamma\left(\frac{u_i+v_i}{2}\right)}{\Gamma\left(\frac{u_i}{2}\right)\Gamma\left(\frac{v_i}{2}\right)} \int_0^{f_{p_i}} F_i^{\frac{1}{2}(u_i-2)} (v_i + u_i F_i)^{-\frac{1}{2}(u_i+v_i)}\, dF_i,$$

where $u_i, v_i > 0$; $0 \leq f_{p_i} < \infty$.

The value of f_{p_i} is computed by means of a transformation to a beta distribution, $P_{i\beta_i}\left(B_i \leq \beta_i : a_i, b_i\right)$:

$$P\left(F_i \leq f_{p_i} : u_i, v_i\right) = P_{i\beta_i}\left(B_i \leq \frac{u_i f_{p_i}}{u_i f_{p_i} + v_i} : u_i/2, v_i/2\right)$$

and using a call to nag_deviates_beta_vector (g01tec).

For very large values of both u_i and v_i, greater than 10^5, a Normal approximation is used. If only one of u_i or v_i is greater than 10^5 then a χ^2 approximation is used; see Abramowitz and Stegun (1972).

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

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 $i = 1, 2, \ldots, \max($**ltail**, **lp**, **ldf1**, **ldf2**$)$:

tail$[j]$ = Nag_LowerTail
The lower tail probability, i.e., $p_i = P\left(F_i \leq f_{p_i} : u_i, v_i\right)$.

tail$[j]$ = Nag_UpperTail
The upper tail probability, i.e., $p_i = P\left(F_i \geq f_{p_i} : u_i, v_i\right)$.

Constraint: **tail**$[j - 1]$ = Nag_LowerTail or Nag_UpperTail, 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 F-distribution as defined by **tail** with $p_i =$ **p**$[j]$, $j = (i - 1) \bmod$ **lp**.

Constraints:

if **tail**$[k]$ = Nag_LowerTail, $0.0 \leq$ **p**$[j] < 1.0$;
otherwise $0.0 <$ **p**$[j] \leq 1.0$.

Where $k = (i - 1) \bmod$ **ltail** and $j = (i - 1) \bmod$ **lp**.

5: **ldf1** – Integer *Input*

On entry: the length of the array **df1**.

Constraint: **ldf1** > 0.

6: **df1**[**ldf1**] – const double *Input*

On entry: u_i, the degrees of freedom of the numerator variance with $u_i =$ **df1**$[j]$, $j = (i - 1) \bmod$ **ldf1**.

Constraint: **df1**$[j - 1] > 0.0$, for $j = 1, 2, \ldots,$ **ldf1**.

7: **ldf2** – Integer *Input*

On entry: the length of the array **df2**.

Constraint: **ldf2** > 0.

8: **df2**[**ldf2**] – const double *Input*

On entry: v_i, the degrees of freedom of the denominator variance with $v_i =$ **df2**$[j]$, $j = (i - 1) \bmod$ **ldf2**.

Constraint: **df2**$[j - 1] > 0.0$, for $j = 1, 2, \ldots,$ **ldf2**.

9: **f**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **f** must be at least $\max($**ltail**, **lp**, **ldf1**, **ldf2**$)$.

On exit: f_{p_i}, the deviates for the F-distribution.

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

Note: the dimension, *dim*, of the array **ivalid** must be at least $\max($**ltail**, **lp**, **ldf1**, **ldf2**$)$.

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 f_{p_i}.

ivalid$[i-1] = 2$
On entry, invalid value for p_i.

ivalid$[i-1] = 3$
On entry, $u_i \leq 0.0$,
or $v_i \leq 0.0$.

ivalid$[i-1] = 4$
The solution has not converged. The result should still be a reasonable approximation to the solution.

ivalid$[i-1] = 5$
The value of p_i is too close to 0.0 or 1.0 for the result to be computed. This will only occur when the large sample approximations are used.

11: **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: **ldf1** > 0.

On entry, array size = ⟨*value*⟩.
Constraint: **ldf2** > 0.

On entry, array size = ⟨*value*⟩.
Constraint: **lp** > 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**, **p**, **df1**, **df2** was invalid, or the solution failed to converge. Check **ivalid** for more information.

7 Accuracy

The result should be accurate to five significant digits.

8 Parallelism and Performance

Not applicable.

9 Further Comments

For higher accuracy nag_deviates_beta_vector (g01tec) can be used along with the transformations given in Section 3.

10 Example

This example reads the lower tail probabilities for several F-distributions, and calculates and prints the corresponding deviates.

10.1 Program Text

```
/* nag_deviates_f_vector (g01tdc) 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, lp, ldf1, ldf2, 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, *df1 = 0, *df2 = 0, *f = 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_f_vector (g01tdc) 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] ", &lp);
  if (!(p = NAG_ALLOC(lp, double))) {
    printf("Allocation failure\n");
```

```
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lp; i++)
    scanf("%lf", &p[i]);
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &ldf1);
  if (!(df1 = NAG_ALLOC(ldf1, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ldf1; i++)
    scanf("%lf", &df1[i]);
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &ldf2);
  if (!(df2 = NAG_ALLOC(ldf2, double))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ldf2; i++)
    scanf("%lf", &df2[i]);
  scanf("%*[^\n] ");

  /* Allocate memory for output */
  lout = MAX(ltail,MAX(lp,MAX(ldf1,ldf2)));
  if (!(f = NAG_ALLOC(lout, double)) ||
      !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Calculate probability */
  nag_deviates_f_vector(ltail, tail, lp, p, ldf1, df1, ldf2, df2, f,
                        ivalid, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_deviates_f_vector (g01tdc).\n%s\n",
            fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID) goto END;
  }

  /* Display title */
  printf("       tail          p        df1       df2       f     ivalid\n");
  printf(" ---------------------------------------------------------------\n");

  /* Display results */
  for (i = 0; i < lout; i++)
    printf(" %15s  %6.3f    %6.1f    %6.1f   %7.3f    %3ld\n",
           nag_enum_value_to_name(tail[i%ltail]),p[i%lp], df1[i%ldf1],
           df2[i%ldf2], f[i], ivalid[i]);

 END:
  NAG_FREE(tail);
  NAG_FREE(p);
  NAG_FREE(df1);
  NAG_FREE(df2);
  NAG_FREE(f);
  NAG_FREE(ivalid);

  return(exit_status);
}
```

10.2 Program Data

```
nag_deviates_f_vector (g01tdc) Example Program Data
1                                                                  :: ltail
Nag_LowerTail                                                      :: tail
3                                                                  :: lp
0.984 0.9 0.534                                                    :: p
3                                                                  :: ldf1
10.0 1.0 20.25                                                     :: df1
3                                                                  :: ldf2
25.5 1.0 1.0                                                       :: df2
```

10.3 Program Results

```
nag_deviates_f_vector (g01tdc) Example Program Results

       tail          p        df1       df2        f     ivalid
 ---------------------------------------------------------------
   Nag_LowerTail   0.984      10.0      25.5     2.847       0
   Nag_LowerTail   0.900       1.0       1.0    39.863       0
   Nag_LowerTail   0.534      20.2       1.0     2.498       0
```