NAG Toolbox

nag_stat_inv_cdf_beta_vector (g01te)

1 Purpose

nag_stat_inv_cdf_beta_vector (g01te) returns a number of deviates associated with given probabilities of the beta distribution.

2 Syntax

```
[beta, ivalid, ifail] = nag_stat_inv_cdf_beta_vector(tail, p, a, b, 'ltail',
ltail, 'lp', lp, 'la', la, 'lb', lb, 'tol', tol)
```

```
[beta, ivalid, ifail] = g01te(tail, p, a, b, 'ltail', ltail, 'lp', lp, 'la', la,
'lb', lb, 'tol', tol)
```

3 Description

The deviate, β_{p_i}, associated with the lower tail probability, p_i, of the beta distribution with parameters a_i and b_i is defined as the solution to

$$P\left(B_i \leq \beta_{p_i} : a_i, b_i\right) = p_i = \frac{\Gamma(a_i + b_i)}{\Gamma(a_i)\Gamma(b_i)} \int_0^{\beta_{p_i}} {B_i}^{a_i - 1} (1 - B_i)^{b_i - 1} \, dB_i, \quad 0 \leq \beta_{p_i} \leq 1; \; a_i, b_i > 0.$$

The algorithm is a modified version of the Newton–Raphson method, following closely that of Cran *et al.* (1977).

An initial approximation, β_{i0}, to β_{p_i} is found (see Cran *et al.* (1977)), and the Newton–Raphson iteration

$$\beta_k = \beta_{k-1} - \frac{f_i(\beta_{k-1})}{f_i'(\beta_{k-1})},$$

where $f_i(\beta_k) = P(B_i \leq \beta_k : a_i, b_i) - p_i$ is used, with modifications to ensure that β_k remains in the range $(0, 1)$.

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

Cran G W, Martin K J and Thomas G E (1977) Algorithm AS 109. Inverse of the incomplete beta function ratio *Appl. Statist.* **26** 111–114

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

5 Parameters

5.1 Compulsory Input Parameters

1: **tail**(**ltail**) – CHARACTER(1) array

Indicates which tail the supplied probabilities represent. For $j = ((i - 1) \bmod \mathbf{ltail}) + 1$, for $i = 1, 2, \ldots, \max(\mathbf{ltail}, \mathbf{lp}, \mathbf{la}, \mathbf{lb})$:

tail(j) = 'L'
The lower tail probability, i.e., $p_i = P\left(B_i \leq \beta_{p_i} : a_i, b_i\right)$.

tail(j) = 'U'
The upper tail probability, i.e., $p_i = P\left(B_i \geq \beta_{p_i} : a_i, b_i\right)$.

Constraint: **tail**(j) = 'L' or 'U', for $j = 1, 2, \ldots,$ **ltail**.

2: **p**(**lp**) – REAL (KIND=nag_wp) array

p_i, the probability of the required beta distribution as defined by **tail** with $p_i = \mathbf{p}(j)$, $j = ((i-1) \bmod \mathbf{lp}) + 1$.

Constraint: $0.0 \leq \mathbf{p}(j) \leq 1.0$, for $j = 1, 2, \ldots,$ **lp**.

3: **a**(**la**) – REAL (KIND=nag_wp) array

a_i, the first parameter of the required beta distribution with $a_i = \mathbf{a}(j)$, $j = ((i-1) \bmod \mathbf{la}) + 1$.

Constraint: $0.0 < \mathbf{a}(j) \leq 10^6$, for $j = 1, 2, \ldots,$ **la**.

4: **b**(**lb**) – REAL (KIND=nag_wp) array

b_i, the second parameter of the required beta distribution with $b_i = \mathbf{b}(j)$, $j = ((i-1) \bmod \mathbf{lb}) + 1$.

Constraint: $0.0 < \mathbf{b}(j) \leq 10^6$, for $j = 1, 2, \ldots,$ **lb**.

5.2 Optional Input Parameters

1: **ltail** – INTEGER

Default: the dimension of the array **tail**.

The length of the array **tail**.

Constraint: **ltail** > 0.

2: **lp** – INTEGER

Default: the dimension of the array **p**.

The length of the array **p**.

Constraint: **lp** > 0.

3: **la** – INTEGER

Default: the dimension of the array **a**.

The length of the array **a**.

Constraint: **la** > 0.

4: **lb** – INTEGER

Default: the dimension of the array **b**.

The length of the array **b**.

Constraint: **lb** > 0.

5: **tol** – REAL (KIND=nag_wp)

Default: 0.0

The relative accuracy required by you in the results. If nag_stat_inv_cdf_beta_vector (g01te) is entered with **tol** greater than or equal to 1.0 or less than $10 \times$ ***machine precision*** (see nag_machine_precision (x02aj)), then the value of $10 \times$ ***machine precision*** is used instead.

5.3 Output Parameters

1: **beta**(:) – REAL (KIND=nag_wp) array

The dimension of the array **beta** will be max(**ltail**, **lp**, **la**, **lb**)

β_{p_i}, the deviates for the beta distribution.

2: **ivalid**(:) – INTEGER array

The dimension of the array **ivalid** will be max(**ltail**, **lp**, **la**, **lb**)

ivalid(i) indicates any errors with the input arguments, with

ivalid(i) $= 0$
No error.

ivalid(i) $= 1$

On entry, invalid value supplied in **tail** when calculating β_{p_i}.

ivalid(i) $= 2$

On entry, $p_i < 0.0$,
or $p_i > 1.0$.

ivalid(i) $= 3$

On entry, $a_i \leq 0.0$,
or $a_i > 10^6$,
or $b_i \leq 0.0$,
or $b_i > 10^6$.

ivalid(i) $= 4$
The solution has not converged but the result should be a reasonable approximation to the solution.

ivalid(i) $= 5$
Requested accuracy not achieved when calculating the beta probability. The result should be a reasonable approximation to the correct solution.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Note: nag_stat_inv_cdf_beta_vector (g01te) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail $= 1$ (*warning*)

On entry, at least one value of **tail**, **p**, **a**, or **b** was invalid, or the solution failed to converge. Check **ivalid** for more information.

ifail $= 2$

Constraint: **ltail** > 0.

ifail $= 3$

Constraint: **lp** > 0.

ifail $= 4$

Constraint: **la** > 0.

ifail $= 5$

Constraint: **lb** > 0.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The required precision, given by **tol**, should be achieved in most circumstances.

8 Further Comments

The typical timing will be several times that of nag_stat_prob_beta_vector (g01se) and will be very dependent on the input argument values. See nag_stat_prob_beta_vector (g01se) for further comments on timings.

9 Example

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

9.1 Program Text

```
function g01te_example

fprintf('g01te example results\n\n');

tail = {'L'};
p = [0.5; 0.99; 0.25];
a = [1.0; 1.5; 20.0];
b = [2.0; 1.5; 10.0];

[x, ivalid, ifail] = g01te( ...
                            tail, p, a, b);

fprintf('  tail  probability    a         b    deviate    ivalid\n');
ltail = numel(tail);
lp    = numel(p);
la    = numel(a);
lb    = numel(b);
len   = max ([ltail, lp, la, lb]);
for i=0:len-1
  fprintf('%5s%9.4f%10.3f%10.3f%10.4f%8d\n', tail{mod(i, ltail)+1}, ...
          p(mod(i,lp)+1), a(mod(i,la)+1), b(mod(i,lb)+1), x(i+1), ivalid(i+1));
end
```

9.2 Program Results

```
    g01te example results

 tail  probability    a         b     deviate    ivalid
   L   0.5000      1.000     2.000    0.2929       0
   L   0.9900      1.500     1.500    0.9672       0
   L   0.2500     20.000    10.000    0.6105       0
```