NAG Library Function Document

nag_prob_beta_vector (g01sec)

1 Purpose

nag_prob_beta_vector (g01sec) computes a number of lower or upper tail probabilities for the beta distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
void nag_prob_beta_vector (Integer ltail, const Nag_TailProbability tail[],
     Integer lbeta, const double beta[], Integer la, const double a[],
     Integer lb, const double b[], double p[], Integer ivalid[],
     NagError *fail)
```

3 Description

The lower tail probability, $P\left(B_i \leq \beta_i : a_i, b_i\right)$ is defined by

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

The function $I_{\beta_i}(a_i, b_i)$, also known as the incomplete beta function is calculated using nag_incomplete_beta (s14ccc).

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

Majumder K L and Bhattacharjee G P (1973) Algorithm AS 63. The incomplete beta integral *Appl. Statist.* **22** 409–411

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 whether a lower or upper tail probabilities are required. For $j = (i - 1) \bmod$ **ltail**, for $i = 1, 2, \ldots, \max($**ltail**, **lbeta**, **la**, **lb**$)$:

tail$[j]$ = Nag_LowerTail
The lower tail probability is returned, i.e., $p_i = P\left(B_i \leq \beta_i : a_i, b_i\right)$.

tail$[j]$ = Nag_UpperTail
The upper tail probability is returned, i.e., $p_i = P(B_i \geq \beta_i : a_i, b_i)$.

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

3: **lbeta** – Integer *Input*

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

Constraint: **lbeta** > 0.

4: **beta**[**lbeta**] – const double *Input*

On entry: β_i, the value of the beta variate with $\beta_i =$ **beta**$[j]$, $j = (i-1) \bmod$ **lbeta**.

Constraint: $0.0 \leq$ **beta**$[j-1] \leq 1.0$, for $j = 1, 2, \ldots,$ **lbeta**.

5: **la** – Integer *Input*

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

Constraint: **la** > 0.

6: **a**[**la**] – const double *Input*

On entry: a_i, the first parameter of the required beta distribution with $a_i =$ **a**$[j]$, $j = (i-1) \bmod$ **la**.

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

7: **lb** – Integer *Input*

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

Constraint: **lb** > 0.

8: **b**[**lb**] – const double *Input*

On entry: b_i, the second parameter of the required beta distribution with $b_i =$ **b**$[j]$, $j = (i-1) \bmod$ **lb**.

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

9: **p**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **p** must be at least max(**ltail**, **lbeta**, **la**, **lb**).

On exit: p_i, the probabilities for the beta distribution.

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

Note: the dimension, *dim*, of the array **ivalid** must be at least max(**ltail**, **lbeta**, **la**, **lb**).

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, $\beta_i < 0.0$,
or $\beta_i > 1.0$.

ivalid$[i-1] = 3$

On entry, $a_i \leq 0.0$,
or $b_i \leq 0.0$,

11: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_ARRAY_SIZE

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

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

On entry, array size $= \langle value \rangle$.
Constraint: **lbeta** > 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 How to Use the NAG Library and its Documentation 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 How to Use the NAG Library and its Documentation for further information.

NW_IVALID

On entry, at least one value of **beta**, **a**, **b** or **tail** was invalid.
Check **ivalid** for more information.

7 Accuracy

The accuracy is limited by the error in the incomplete beta function. See Section 7 in nag_incomplete_beta (s14ccc) for further details.

8 Parallelism and Performance

nag_prob_beta_vector (g01sec) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example reads values from a number of beta distributions and computes the associated lower tail probabilities.

10.1 Program Text

```
/* nag_prob_beta_vector (g01sec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
  /* Integer scalar and array declarations */
  Integer ltail, lbeta, la, lb, i, lout;
  Integer *ivalid = 0;
  Integer exit_status = 0;

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

  /* Double scalar and array declarations */
  double *beta = 0, *a = 0, *b = 0, *p = 0;

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

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

  printf("nag_prob_beta_vector (g01sec) 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, (unsigned)_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] ", &lbeta);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &lbeta);
#endif
  if (!(beta = NAG_ALLOC(lbeta, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lbeta; i++)
#ifdef _WIN32
    scanf_s("%lf", &beta[i]);
#else
    scanf("%lf", &beta[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

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

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &lb);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &lb);
#endif
  if (!(b = NAG_ALLOC(lb, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lb; i++)
#ifdef _WIN32
    scanf_s("%lf", &b[i]);
#else
```

```
    scanf("%lf", &b[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

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

  /* Calculate probability */
  nag_prob_beta_vector(ltail, tail, lbeta, beta, la, a, lb, b,
                       p, ivalid, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_prob_beta_vector (g01sec).\n%s\n", fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID)
      goto END;
  }

  /* Display title */
  printf("        tail          beta        a         b        ");
  printf("p     ivalid\n");
  printf(" ------------------------------------------------------");
  printf("--------------\n");

  /* Display results */
  for (i = 0; i < lout; i++)
    printf(" %15s    %6.2f    %6.2f    %6.2f    %6.3f    %3" NAG_IFMT "\n",
           nag_enum_value_to_name(tail[i % ltail]), beta[i % lbeta],
           a[i % la], b[i % lb], p[i], ivalid[i]);

END:
  NAG_FREE(tail);
  NAG_FREE(beta);
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(p);
  NAG_FREE(ivalid);

  return (exit_status);
}
```

10.2 Program Data

```
nag_prob_beta_vector (g01sec) Example Program Data
1                                                                   :: ltail
Nag_LowerTail                                                       :: tail
3                                                                   :: lbeta
0.26 0.75 0.5                                                       :: beta
3                                                                   :: la
1.0 1.5 2.0                                                         :: a
3                                                                   :: lb
2.0 1.5 1.0                                                         :: b
```

10.3 Program Results

```
nag_prob_beta_vector (g01sec) Example Program Results

        tail          beta         a          b         p     ivalid
 ------------------------------------------------------------------
   Nag_LowerTail      0.26      1.00       2.00     0.452       0
   Nag_LowerTail      0.75      1.50       1.50     0.804       0
   Nag_LowerTail      0.50      2.00       1.00     0.250       0
```