

## NAG Library Function Document

### nag\_prob\_chi\_sq\_vector (g01scc)

#### 1 Purpose

nag\_prob\_chi\_sq\_vector (g01scc) returns a number of lower or upper tail probabilities for the  $\chi^2$ -distribution with real degrees of freedom.

#### 2 Specification

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

void nag_prob_chi_sq_vector (Integer ltail,
                             const Nag_TailProbability tail[], Integer lx, const double x[],
                             Integer ldf, const double df[], double p[], Integer ivalid[],
                             NagError *fail)
```

#### 3 Description

The lower tail probability for the  $\chi^2$ -distribution with  $\nu_i$  degrees of freedom,  $P = (X_i \leq x_i : \nu_i)$  is defined by:

$$P = (X_i \leq x_i : \nu_i) = \frac{1}{2^{\nu_i/2} \Gamma(\nu_i/2)} \int_{0.0}^{x_i} X_i^{\nu_i/2-1} e^{-X_i/2} dX_i, \quad x_i \geq 0, \nu_i > 0.$$

To calculate  $P = (X_i \leq x_i : \nu_i)$  a transformation of a gamma distribution is employed, i.e., a  $\chi^2$ -distribution with  $\nu_i$  degrees of freedom is equal to a gamma distribution with scale parameter 2 and shape parameter  $\nu_i/2$ .

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 whether the lower or upper tail probabilities are required. For  $j = (i - 1) \bmod \mathbf{ltail}$ , for  $i = 1, 2, \dots, \max(\mathbf{ltail}, \mathbf{lx}, \mathbf{ldf})$ :  
**tail[j]** = Nag\_LowerTail  
The lower tail probability is returned, i.e.,  $p_i = P(X_i \leq x_i : \nu_i)$ .

**tail**[*j*] = Nag\_UpperTail

The upper tail probability is returned, i.e.,  $p_i = P(X_i \geq x_i : \nu_i)$ .

*Constraint:* **tail**[*j* - 1] = Nag\_LowerTail or Nag\_UpperTail, for  $j = 1, 2, \dots, \mathbf{ltail}$ .

- 3: **lx** – Integer *Input*  
*On entry:* the length of the array **x**.  
*Constraint:* **lx** > 0.
- 4: **x**[**lx**] – const double *Input*  
*On entry:*  $x_i$ , the values of the  $\chi^2$  variates with  $\nu_i$  degrees of freedom with  $x_i = \mathbf{x}[j]$ ,  $j = (i - 1) \bmod \mathbf{lx}$ .  
*Constraint:*  $\mathbf{x}[j - 1] \geq 0.0$ , for  $j = 1, 2, \dots, \mathbf{lx}$ .
- 5: **ldf** – Integer *Input*  
*On entry:* the length of the array **df**.  
*Constraint:* **ldf** > 0.
- 6: **df**[**ldf**] – const double *Input*  
*On entry:*  $\nu_i$ , the degrees of freedom of the  $\chi^2$ -distribution with  $\nu_i = \mathbf{df}[j]$ ,  $j = (i - 1) \bmod \mathbf{ldf}$ .  
*Constraint:* **df**[*j* - 1] > 0.0, for  $j = 1, 2, \dots, \mathbf{ldf}$ .
- 7: **p**[*dim*] – double *Output*  
**Note:** the dimension, *dim*, of the array **p** must be at least  $\max(\mathbf{ltail}, \mathbf{ldf}, \mathbf{lx})$ .  
*On exit:*  $p_i$ , the probabilities for the  $\chi^2$  distribution.
- 8: **ivalid**[*dim*] – Integer *Output*  
**Note:** the dimension, *dim*, of the array **ivalid** must be at least  $\max(\mathbf{ltail}, \mathbf{ldf}, \mathbf{lx})$ .  
*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,  $x_i < 0.0$ .  
**ivalid**[*i* - 1] = 3  
 On entry,  $\nu_i \leq 0.0$ .  
**ivalid**[*i* - 1] = 4  
 The solution has failed to converge while calculating the gamma variate. The result returned should represent an approximation to the solution.
- 9: **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: **ldf** > 0.

On entry, array size =  $\langle value \rangle$ .

Constraint: **itail** > 0.

On entry, array size =  $\langle value \rangle$ .

Constraint: **ix** > 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\_INVALID

On entry, at least one value of **x**, **df** or **tail** was invalid, or the solution failed to converge.

Check **invalid** for more information.

## 7 Accuracy

A relative accuracy of five significant figures is obtained in most cases.

## 8 Parallelism and Performance

nag\_prob\_chi\_sq\_vector (g01scc) is not threaded in any implementation.

## 9 Further Comments

For higher accuracy the transformation described in Section 3 may be used with a direct call to nag\_incomplete\_gamma (s14bac).

## 10 Example

Values from various  $\chi^2$ -distributions are read, the lower tail probabilities calculated, and all these values printed out.

## 10.1 Program Text

```

/* nag_prob_chi_sq_vector (g01scc) 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, lx, ldf, i, lout;
    Integer *ivalid = 0;
    Integer exit_status = 0;

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

    /* Double scalar and array declarations */
    double *x = 0, *df = 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_chi_sq_vector (g01scc) 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] ", &lx);

```

```

#else
    scanf("%" NAG_IFMT "%*[\n] ", &lx);
#endif
    if (!(x = NAG_ALLOC(lx, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 0; i < lx; i++)
#ifdef _WIN32
        scanf_s("%lf", &x[i]);
#else
        scanf("%lf", &x[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(lx, 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_chi_sq_vector(ltail, tail, lx, x, ldf, df, p, ivalid, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_prob_chi_sq_vector (g01scc).\n%s\n", fail.message);
        exit_status = 1;
        if (fail.code != NW_INVALID)
            goto END;
    }

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

    /* Display results */
    for (i = 0; i < lout; i++)
        printf(" %15s %6.3f %6.1f %6.4f %3" NAG_IFMT "\n",
            nag_enum_value_to_name(tail[i % ltail]), x[i % lx], df[i % ldf],
            p[i], ivalid[i]);

```

```

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

  return (exit_status);
}

```

## 10.2 Program Data

```

nag_prob_chi_sq_vector (g01scc) Example Program Data
1                                     :: ltail
Nag_LowerTail                       :: tail
3                                     :: lx
8.26 6.2 55.76                       :: x
3                                     :: ldf
20.0 7.5 45.0                         :: df

```

## 10.3 Program Results

nag\_prob\_chi\_sq\_vector (g01scc) Example Program Results

| tail          | x      | df   | p      | ivalid |
|---------------|--------|------|--------|--------|
| Nag_LowerTail | 8.260  | 20.0 | 0.0100 | 0      |
| Nag_LowerTail | 6.200  | 7.5  | 0.4279 | 0      |
| Nag_LowerTail | 55.760 | 45.0 | 0.8694 | 0      |