

**nag\_binomial\_dist (g01bjc)****1. Purpose**

**nag\_binomial\_dist (g01bjc)** returns the lower tail, upper tail and point probabilities associated with a Binomial distribution.

**2. Specification**

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

void nag_binomial_dist(Integer n, double p, Integer k, double *plek,
                       double *pgtk, double *peqk, NagError *fail)
```

**3. Description**

Let  $X$  denote a random variable having a Binomial distribution with parameters  $n$  and  $p$  ( $n \geq 0$  and  $0 < p < 1$ ). Then

$$\text{Prob}\{X = k\} = \binom{n}{k} p^k (1-p)^{n-k}, \text{ for } k = 0, 1, \dots, n.$$

The mean of the distribution is  $np$  and the variance is  $np(1-p)$ .

This routine computes for given  $n$ ,  $p$  and  $k$  the probabilities:

$$\begin{aligned} \mathbf{plek} &= \text{Prob}\{X \leq k\} \\ \mathbf{pgtk} &= \text{Prob}\{X > k\} \\ \mathbf{peqk} &= \text{Prob}\{X = k\}. \end{aligned}$$

The method is similar to the method for the Poisson distribution described in Knüsel (1986).

**4. Parameters****n**

Input: the parameter  $n$  of the Binomial distribution.  
Constraint:  $\mathbf{n} \geq 0$ .

**p**

Input: the parameter  $p$  of the Binomial distribution.  
Constraint:  $0.0 < \mathbf{p} < 1.0$ .

**k**

Input: the integer  $k$  which defines the required probabilities.  
Constraint:  $0 \leq \mathbf{k} \leq \mathbf{n}$ .

**plek**

Output: the lower tail probability,  $\text{Prob}\{X \leq k\}$ .

**pgtk**

Output: the upper tail probability,  $\text{Prob}\{X > k\}$ .

**peqk**

Output: the point probability,  $\text{Prob}\{X = k\}$ .

**fail**

The NAG error parameter, see the Essential Introduction to the NAG C Library.

**5. Error Indications and Warnings****NE\_INT\_ARG\_LT**

On entry,  $\mathbf{n}$  must not be less than 0:  $\mathbf{n} = \langle \text{value} \rangle$ .

On entry,  $\mathbf{k}$  must not be less than 0:  $\mathbf{k} = \langle \text{value} \rangle$ .

**NE\_2\_INT\_ARG\_GT**

On entry,  $k = \langle value \rangle$  while  $n = \langle value \rangle$ . These parameters must satisfy  $k \leq n$ .

**NE\_ARG\_TOO\_LARGE**

On entry,  $n$  is too large to be represented exactly as a double precision number.

**NE\_REAL\_ARG\_LE**

On entry,  $p$  must not be less than or equal to 0.0:  $p = \langle value \rangle$ .

**NE\_REAL\_ARG\_GE**

On entry,  $p$  must not be greater than or equal to 1.0:  $p = \langle value \rangle$ .

**NE\_VARIANCE\_TOO\_LARGE**

On entry, the variance ( $= np(1-p)$ ) exceeds  $10^6$ .

**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 consult NAG for assistance.

**6. Further Comments**

The time taken by the routine depends on the variance ( $= np(1-p)$ ) and on  $k$ . For given variance, the time is greatest when  $k \approx np$  (= the mean), and is then approximately proportional to the square-root of the variance.

**6.1. Accuracy**

Results are correct to a relative accuracy of at least  $10^{-6}$  on machines with a precision of 9 or more decimal digits, and to a relative accuracy of at least  $10^{-3}$  on machines of lower precision (provided that the results do not underflow to zero).

**6.2. References**

Knüsel L (1986) Computation of the Chi-square and Poisson Distribution. *SIAM J. Sci. Statist. Comput.* **7** 1022–1036.

**7. See Also**

nag\_poisson\_dist (g01bkc)  
nag\_hypergeom\_dist (g01blc)

**8. Example**

This example program reads values of  $n$  and  $p$  from a data file until end-of-file is reached, and prints the corresponding probabilities.

**8.1. Program Text**

```

/* nag_binomial_dist(g01bjc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 *
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nagg01.h>

main()
{
    double plek, peqk, pgtk;
    double p;

    Integer k, n;

```

```

Vprintf("g01bjc Example Program Results\n");
/*      Skip heading in data file */
Vscanf("%*[^\\n] ");

Vprintf("\\n");
Vprintf("  n      p      k      plek      pgtk      peqk\\n\\n");

while ((scanf("%ld %lf %ld%*[^\\n]", &n, &p, &k)) != EOF)
  {
  g01bjc(n, p, k, &plek, &pgtk, &peqk, NAGERR_DEFAULT);
  Vprintf("%5ld%8.3f%5ld%10.5f%10.5f%10.5f\\n",n, p, k, plek, pgtk, peqk);
  }

  exit(EXIT_SUCCESS);
}

```

## 8.2. Program Data

```

g01bjc Example Program Data
4 0.50  2 : n, p, k
19 0.44 13
100 0.75 67
2000 0.33 700

```

## 8.3. Program Results

```

g01bjc Example Program Results

```

n	p	k	plek	pgtk	peqk
4	0.500	2	0.68750	0.31250	0.37500
19	0.440	13	0.99138	0.00862	0.01939
100	0.750	67	0.04460	0.95540	0.01700
2000	0.330	700	0.97251	0.02749	0.00312

---