NAG Library Function Document

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

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

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

nag_binomial_dist (g01bjc) computes for given n, p and k the probabilities:

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

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

4 References

KnÏsel L (1986) Computation of the chi-square and Poisson distribution *SIAM J. Sci. Statist. Comput.* **7** 1022–1036

5 Arguments

1: **n** – Integer *Input*

On entry: the parameter n of the binomial distribution.

Constraint: $\mathbf{n} \geq 0$.

2: **p** – double *Input*

On entry: the parameter p of the binomial distribution.

Constraint: $0.0 < \mathbf{p} < 1.0$.

3: **k** – Integer *Input*

On entry: the integer k which defines the required probabilities.

Constraint: $0 \leq \mathbf{k} \leq \mathbf{n}$.

4: **plek** – double * *Output*

On exit: the lower tail probability, $\mathrm{Prob}\{X \le k\}$.

5: **pgtk** – double * *Output*

On exit: the upper tail probability, $\mathrm{Prob}\{X > k\}$.

6: **peqk** – double * *Output*

On exit: the point probability, $\mathrm{Prob}\{X = k\}$.

7: **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_2_INT_ARG_GT

On entry, **k** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **k** $\le$ or **n**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_ARG_TOO_LARGE

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

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT_ARG_LT

On entry, **k** $= \langle value \rangle$.
Constraint: **k** ≥ 0.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

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 2.7.6 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 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_REAL_ARG_GE

On entry, **p** $= \langle value \rangle$.
Constraint: **p** < 1.0.

NE_REAL_ARG_LE

On entry, **p** $= \langle value \rangle$.
Constraint: **p** > 0.0.

NE_VARIANCE_TOO_LARGE

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

7 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).

8 Parallelism and Performance

nag_binomial_dist (g01bjc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_binomial_dist (g01bjc) 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.

10 Example

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

10.1 Program Text

```
/* nag_binomial_dist (g01bjc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

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

int main(void)
{
  Integer exit_status = 0;
  Integer k, n;
  double plek, peqk, pgtk;
  double p;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_binomial_dist (g01bjc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
```

```
  printf("\n");
  printf("   n     p      k     plek      pgtk      peqk\n\n");

#ifdef _WIN32
  while ((scanf_s("%" NAG_IFMT " %lf %" NAG_IFMT "%*[^\n]", &n, &p, &k)) !=
         EOF)
#else
  while ((scanf("%" NAG_IFMT " %lf %" NAG_IFMT "%*[^\n]", &n, &p, &k)) != EOF)
#endif
  {
    /* nag_binomial_dist (g01bjc).
     * Binomial distribution function
     */
    nag_binomial_dist(n, p, k, &plek, &pgtk, &peqk, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_binomial_dist (g01bjc)\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
    printf("%5" NAG_IFMT "%8.3f%5" NAG_IFMT "%10.5f%10.5f%10.5f\n", n, p, k,
           plek, pgtk, peqk);
  }

END:
  return exit_status;
}
```

10.2 Program Data

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

10.3 Program Results

```
nag_binomial_dist (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
```