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

```c
#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:

- $\text{plek} = \text{Prob}\{X \leq k\}$
- $\text{pgtk} = \text{Prob}\{X > k\}$
- $\text{peqk} = \text{Prob}\{X = k\}$.

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

4 References


5 Arguments

1: $n$ – Integer

   **Input**

   *On entry:* the parameter $n$ of the binomial distribution.

   *Constraint:* $n \geq 0$.

2: $p$ – double

   **Input**

   *On entry:* the parameter $p$ of the binomial distribution.

   *Constraint:* $0.0 < p < 1.0$.

3: $k$ – Integer

   **Input**

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

   *Constraint:* $0 \leq k \leq n$. 
4: **plek** – double * Output
   On exit: the lower tail probability, \( \text{Prob}\{X \leq k\} \).

5: **pgtk** – double * Output
   On exit: the upper tail probability, \( \text{Prob}\{X > k\} \).

6: **peqk** – double * Output
   On exit: the point probability, \( \text{Prob}\{X = k\} \).

7: **fail** – NagError * Input/Output
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 **Error Indicators and Warnings**

**NE_2_INT_ARG_GT**
   On entry, \( k = \langle\text{value}\rangle \) and \( n = \langle\text{value}\rangle \).
   Constraint: \( k \leq n \) or \( n \geq 0 \).

**NE_ALLOC_FAIL**
   Dynamic memory allocation failed.
   See Section 3.2.1.2 in the Essential Introduction 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 \( \langle\text{value}\rangle \) had an illegal value.

**NE_INT_ARG_LT**
   On entry, \( k = \langle\text{value}\rangle \).
   Constraint: \( k \geq 0 \).
   On entry, \( n = \langle\text{value}\rangle \).
   Constraint: \( n \geq 0 \).

**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 3.6.6 in the Essential Introduction 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 the Essential Introduction for further information.

**NE_REAL_ARG_GE**
   On entry, \( p = \langle\text{value}\rangle \).
   Constraint: \( p < 1.0 \).
NE_REAL_ARG_LE
On entry, p = ⟨value⟩.
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
Not applicable.

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 ≈ 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. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 4, 1996. */

#include <nag.h>
#include <nag_errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <nag.h>
#include <nag_errno.h>
#include <stdio.h>
#include <stdlib.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");
#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\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

<table>
<thead>
<tr>
<th>n</th>
<th>p</th>
<th>k</th>
<th>plek</th>
<th>pgtk</th>
<th>peqk</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.50</td>
<td>2</td>
<td>0.68750</td>
<td>0.31250</td>
<td>0.37500</td>
</tr>
<tr>
<td>19</td>
<td>0.44</td>
<td>13</td>
<td>0.99138</td>
<td>0.00862</td>
<td>0.01939</td>
</tr>
<tr>
<td>100</td>
<td>0.75</td>
<td>67</td>
<td>0.04460</td>
<td>0.95540</td>
<td>0.01700</td>
</tr>
<tr>
<td>2000</td>
<td>0.33</td>
<td>700</td>
<td>0.97251</td>
<td>0.02749</td>
<td>0.00312</td>
</tr>
</tbody>
</table>