NAG Library Function Document

nag_prob_poisson_vector (g01skc)

1 Purpose

nag_prob_poisson_vector (g01skc) returns a number of the lower tail, upper tail and point probabilities for the Poisson distribution.

2 Specification

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

```
void nag_prob_poisson_vector (Integer ll, const double l[], Integer lk,
     const Integer k[], double plek[], double pgtk[], double peqk[],
     Integer ivalid[], NagError *fail)
```

3 Description

Let $X = \{X_i : i = 1, 2, \ldots, m\}$ denote a vector of random variables each having a Poisson distribution with parameter λ_i (> 0). Then

$$\text{Prob}\{X_i = k_i\} = e^{-\lambda_i} \frac{\lambda_i^{k_i}}{k_i!}, \quad k_i = 0, 1, 2, \ldots$$

The mean and variance of each distribution are both equal to λ_i.

nag_prob_poisson_vector (g01skc) computes, for given λ_i and k_i the probabilities: $\text{Prob}\{X_i \leq k_i\}$, $\text{Prob}\{X_i > k_i\}$ and $\text{Prob}\{X_i = k_i\}$ using the algorithm described in Knüsel (1986).

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

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

5 Arguments

1: **ll** – Integer *Input*

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

Constraint: $\textbf{ll} > 0$.

2: **l[ll]** – const double *Input*

On entry: λ_i, the parameter of the Poisson distribution with $\lambda_i = \textbf{l}[j]$, $j = (i - 1) \bmod \textbf{ll}$, for $i = 1, 2, \ldots, \max(\textbf{ll}, \textbf{lk})$.

Constraint: $0.0 < \textbf{l}[j - 1] \leq 10^6$, for $j = 1, 2, \ldots, \textbf{ll}$.

3: **lk** – Integer *Input*

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

Constraint: $\textbf{lk} > 0$.

4: **k**[**lk**] – const Integer *Input*

On entry: k_i, the integer which defines the required probabilities with $k_i = \mathbf{k}[j]$, $j = (i-1) \bmod \mathbf{lk}$.

Constraint: $\mathbf{k}[j-1] \geq 0$, for $j = 1, 2, \ldots, \mathbf{lk}$.

5: **plek**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **plek** must be at least max(**ll**, **lk**).

On exit: $\text{Prob}\{X_i \leq k_i\}$, the lower tail probabilities.

6: **pgtk**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **pgtk** must be at least max(**ll**, **lk**).

On exit: $\text{Prob}\{X_i > k_i\}$, the upper tail probabilities.

7: **peqk**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **peqk** must be at least max(**ll**, **lk**).

On exit: $\text{Prob}\{X_i = k_i\}$, the point probabilities.

8: **ivalid**[*dim*] – Integer *Output*

Note: the dimension, *dim*, of the array **ivalid** must be at least max(**ll**, **lk**).

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, $\lambda_i \leq 0.0$.

ivalid$[i-1] = 2$

On entry, $k_i < 0$.

ivalid$[i-1] = 3$

On entry, $\lambda_i > 10^6$.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_ARRAY_SIZE

On entry, array size = ⟨*value*⟩.
Constraint: **lk** > 0.

On entry, array size = ⟨*value*⟩.
Constraint: **ll** > 0.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ 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.

NW_IVALID

On entry, at least one value of **l** or **k** was invalid.
Check **ivalid** for more information.

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 (provided that the results do not underflow to zero).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_prob_poisson_vector (g01skc) to calculate each probability depends on λ_i and k_i. For given λ_i, the time is greatest when $k_i \approx \lambda_i$, and is then approximately proportional to $\sqrt{\lambda_i}$.

10 Example

This example reads a vector of values for λ and k, and prints the corresponding probabilities.

10.1 Program Text

```
/* nag_prob_poisson_vector (g01skc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
  /* Integer scalar and array declarations */
  Integer lk, ll, i, lout;
  Integer *ivalid = 0, *k = 0;
  Integer exit_status = 0;

  /* NAG structures */
  NagError fail;

  /* Double scalar and array declarations */
  double *peqk = 0, *pgtk = 0, *plek = 0, *l = 0;

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

  printf("nag_prob_poisson_vector (g01skc) Example Program Results\n\n");

  /* Skip heading in data file*/
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &ll);
  if (!(l = NAG_ALLOC(ll, double))) {
    printf("Allocation failure\n");
```

```
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ll; i++)
    scanf("%lf", &l[i]);
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &lk);
  if (!(k = NAG_ALLOC(lk, Integer))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lk; i++)
    scanf("%ld", &k[i]);
  scanf("%*[^\n] ");

  /* Allocate memory for output */
  lout = MAX(ll,lk);
  if (!(peqk = NAG_ALLOC(lout, double)) ||
      !(pgtk = NAG_ALLOC(lout, double)) ||
      !(plek = NAG_ALLOC(lout, double)) ||
      !(ivalid = NAG_ALLOC(lout, Integer))) {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Calculate probability */
  nag_prob_poisson_vector(ll, l, lk, k, plek, pgtk, peqk, ivalid, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_prob_poisson_vector (g01skc).\n%s\n",
            fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID) goto END;
  }

  /* Display title */
  printf("     l         k       plek      pgtk      peqk  ivalid\n");
  printf(" --------------------------------------------------------\n");

  /* Display results */
  for (i = 0; i < lout; i++)
    printf(" %6.2f    %6ld    %6.3f    %6.3f    %6.3f   %3ld\n",
           l[i%ll], k[i%lk], plek[i], pgtk[i], peqk[i], ivalid[i]);

 END:
  NAG_FREE(l);
  NAG_FREE(k);
  NAG_FREE(plek);
  NAG_FREE(pgtk);
  NAG_FREE(peqk);
  NAG_FREE(ivalid);

  return(exit_status);
}
```

10.2 Program Data

```
nag_prob_poisson_vector (g01skc) Example Program Data
4                                                               :: ll
0.75 9.20 34.0 175.0                                            :: l
4                                                               :: lk
3 12 25 175                                                     :: k
```

10.3 Program Results

```
nag_prob_poisson_vector (g01skc) Example Program Results

     l         k       plek      pgtk       peqk  ivalid
 -------------------------------------------------------
   0.75         3      0.993     0.007      0.033      0
   9.20        12      0.861     0.139      0.078      0
  34.00        25      0.067     0.933      0.021      0
 175.00       175      0.520     0.480      0.030      0
```