NAG Library Function Document

nag_prob_hypergeom_vector (g01slc)

1 Purpose

nag_prob_hypergeom_vector (g01slc) returns a number of the lower tail, upper tail and point probabilities for the hypergeometric distribution.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
void nag_prob_hypergeom_vector (Integer ln, const Integer n[], Integer ll,
     const Integer l[], Integer lm, const Integer m[], Integer lk,
     const Integer k[], double plek[], double pgtk[], double peqk[],
     Integer ivalid[], NagError *fail)
```

3 Description

Let $X = \{X_i : i = 1, 2, \ldots, r\}$ denote a vector of random variables having a hypergeometric distribution with parameters n_i, l_i and m_i. Then

$$\text{Prob}\{X_i = k_i\} = \frac{\binom{m_i}{k_i}\binom{n_i - m_i}{l_i - k_i}}{\binom{n_i}{l_i}},$$

where $\max(0, l_i + m_i - n_i) \leq k_i \leq \min(l_i, m_i)$, $0 \leq l_i \leq n_i$ and $0 \leq m_i \leq n_i$.

The hypergeometric distribution may arise if in a population of size n_i a number m_i are marked. From this population a sample of size l_i is drawn and of these k_i are observed to be marked.

The mean of the distribution $= \dfrac{l_i m_i}{n_i}$, and the variance $= \dfrac{l_i m_i (n_i - l_i)(n_i - m_i)}{{n_i}^2 (n_i - 1)}$.

nag_prob_hypergeom_vector (g01slc) computes for given n_i, l_i, m_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 an algorithm similar to that described in KnÏsel (1986) for the Poisson distribution.

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

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

5 Arguments

1: **ln** – Integer *Input*

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

Constraint: **ln** > 0.

2: **n**[**ln**] – const Integer *Input*

On entry: n_i, the parameter of the hypergeometric distribution with $n_i = \mathbf{n}[j]$, $j = (i-1) \bmod \mathbf{ln}$, for $i = 1, 2, \ldots, \max(\mathbf{ln}, \mathbf{ll}, \mathbf{lm}, \mathbf{lk})$.

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

3: **ll** – Integer *Input*

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

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

4: **l**[**ll**] – const Integer *Input*

On entry: l_i, the parameter of the hypergeometric distribution with $l_i = \mathbf{l}[j]$, $j = (i-1) \bmod \mathbf{ll}$.

Constraint: $0 \leq l_i \leq n_i$.

5: **lm** – Integer *Input*

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

Constraint: $\mathbf{lm} > 0$.

6: **m**[**lm**] – const Integer *Input*

On entry: m_i, the parameter of the hypergeometric distribution with $m_i = \mathbf{m}[j]$, $j = (i-1) \bmod \mathbf{lm}$.

Constraint: $0 \leq m_i \leq n_i$.

7: **lk** – Integer *Input*

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

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

8: **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: $\max(0, l_i + m_i - n_i) \leq k_i \leq \min(l_i, m_i)$.

9: **plek**[dim] – double *Output*

Note: the dimension, *dim*, of the array **plek** must be at least $\max(\mathbf{ln}, \mathbf{ll}, \mathbf{lm}, \mathbf{lk})$.

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

10: **pgtk**[dim] – double *Output*

Note: the dimension, *dim*, of the array **pgtk** must be at least $\max(\mathbf{ln}, \mathbf{ll}, \mathbf{lm}, \mathbf{lk})$.

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

11: **peqk**[dim] – double *Output*

Note: the dimension, *dim*, of the array **peqk** must be at least $\max(\mathbf{ln}, \mathbf{ll}, \mathbf{lm}, \mathbf{lk})$.

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

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

Note: the dimension, *dim*, of the array **ivalid** must be at least $\max(\mathbf{ln}, \mathbf{ll}, \mathbf{lm}, \mathbf{lk})$.

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, $n_i < 0$.

ivalid$[i-1] = 2$

On entry, $l_i < 0$,
or $l_i > n_i$.

ivalid$[i-1] = 3$

On entry, $m_i < 0$,
or $m_i > n_i$.

ivalid$[i-1] = 4$

On entry, $k_i < 0$,
or $k_i > l_i$,
or $k_i > m_i$,
or $k_i < l_i + m_i - n_i$.

ivalid$[i-1] = 5$

On entry, n_i is too large to be represented exactly as a real number.

ivalid$[i-1] = 6$

On entry, the variance (see Section 3) exceeds 10^6.

13: **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 = ⟨*value*⟩.
Constraint: **lk** > 0.

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

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

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

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_IVALID

On entry, at least one value of **n**, **l**, **m** or **k** was invalid, or the variance was too large.
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

nag_prob_hypergeom_vector (g01slc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_prob_hypergeom_vector (g01slc) to calculate each probability depends on the variance (see Section 3) and on k_i. For given variance, the time is greatest when $k_i \approx l_i m_i / n_i$ (= the mean), and is then approximately proportional to the square-root of the variance.

10 Example

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

10.1 Program Text

```
/* nag_prob_hypergeom_vector (g01slc) 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 ln, ll, lm, lk, i, lout;
  Integer *ivalid = 0, *n = 0, *l = 0, *m = 0, *k = 0;
  Integer exit_status = 0;

  /* NAG structures */
  NagError fail;

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

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

  printf("nag_prob_hypergeom_vector (g01slc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &ln);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &ln);
#endif
  if (!(n = NAG_ALLOC(ln, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ln; i++)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &n[i]);
#else
    scanf("%" NAG_IFMT "", &n[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &ll);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &ll);
#endif
  if (!(l = NAG_ALLOC(ll, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < ll; i++)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &l[i]);
#else
    scanf("%" NAG_IFMT "", &l[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &lm);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &lm);
#endif
  if (!(m = NAG_ALLOC(lm, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lm; i++)
```

```
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &m[i]);
#else
    scanf("%" NAG_IFMT "", &m[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &lk);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &lk);
#endif
  if (!(k = NAG_ALLOC(lk, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  for (i = 0; i < lk; i++)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &k[i]);
#else
    scanf("%" NAG_IFMT "", &k[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Allocate memory for output */
  lout = MAX(ln, MAX(ll, MAX(lm, 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_hypergeom_vector(ln, n, ll, l, lm, m, lk, k,
                            plek, pgtk, peqk, ivalid, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_prob_hypergeom_vector (g01slc).\n%s\n",
           fail.message);
    exit_status = 1;
    if (fail.code != NW_IVALID)
      goto END;
  }

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

  /* Display results */
  for (i = 0; i < lout; i++) {
    printf(" %6" NAG_IFMT "    %6" NAG_IFMT "    %6" NAG_IFMT "    %6"
           NAG_IFMT "", n[i % ln], l[i % ll], m[i % lm], k[i % lk]);
    printf("    %6.3f    %6.3f    %6.3f   %3" NAG_IFMT "\n", plek[i], pgtk[i],
           peqk[i], ivalid[i]);
  }
```

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

  return (exit_status);
}
```

10.2 Program Data

```
nag_prob_hypergeom_vector (g01slc) Example Program Data
4                                                                    :: ln
10 40 155 1000                                                       :: n
4                                                                    :: ll
2 10 35 444                                                          :: l
4                                                                    :: lm
5 3 122 500                                                          :: m
4                                                                    :: lk
1 2 22 220                                                           :: k
```

10.3 Program Results

```
nag_prob_hypergeom_vector (g01slc) Example Program Results

     n         l         m         k       plek      pgtk      peqk  ivalid
 ------------------------------------------------------------------------
    10         2         5         1     0.778     0.222     0.556     0
    40        10         3         2     0.988     0.012     0.137     0
   155        35       122        22     0.011     0.989     0.008     0
  1000       444       500       220     0.424     0.576     0.049     0
```