NAG Library Routine Document

G01SLF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G01SLF returns a number of the lower tail, upper tail and point probabilities for the hypergeometric distribution.

2 Specification

```
SUBROUTINE G01SLF (LN, N, LL, L, LM, M, LK, K, PLEK, PGTK, PEQK, IVALID,      &
                   IFAIL)
INTEGER            LN, N(LN), LL, L(LL), LM, M(LM), LK, K(LK), IVALID(*),     &
                   IFAIL
REAL (KIND=nag_wp) PLEK(*), PGTK(*), PEQK(*)
```

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 $= \frac{l_i m_i}{n_i}$, and the variance $= \frac{l_i m_i (n_i - l_i)(n_i - m_i)}{n_i^2 (n_i - 1)}$.

G01SLF 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 Knüsel (1986) for the Poisson distribution.

The input arrays to this routine are designed to allow maximum flexibility in the supply of vector parameters 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 Parameters

1: LN – INTEGER *Input*

On entry: the length of the array N

Constraint: LN > 0.

2: N(LN) – INTEGER array *Input*

On entry: n_i, the parameter of the hypergeometric distribution with $n_i = \mathrm{N}(j)$, $j = ((i-1) \bmod \mathrm{LN}) + 1$, for $i = 1, 2, \ldots, \max(\mathrm{LN}, \mathrm{LL}, \mathrm{LM}, \mathrm{LK})$.

Constraint: $\mathrm{N}(j) \geq 0$, for $j = 1, 2, \ldots, \mathrm{LN}$.

3: LL – INTEGER *Input*

On entry: the length of the array L

Constraint: $\mathrm{LL} > 0$.

4: L(LL) – INTEGER array *Input*

On entry: l_i, the parameter of the hypergeometric distribution with $l_i = \mathrm{L}(j)$, $j = ((i-1) \bmod \mathrm{LL}) + 1$.

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

5: LM – INTEGER *Input*

On entry: the length of the array M

Constraint: $\mathrm{LM} > 0$.

6: M(LM) – INTEGER array *Input*

On entry: m_i, the parameter of the hypergeometric distribution with $m_i = \mathrm{M}(j)$, $j = ((i-1) \bmod \mathrm{LM}) + 1$.

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

7: LK – INTEGER *Input*

On entry: the length of the array K

Constraint: $\mathrm{LK} > 0$.

8: K(LK) – INTEGER array *Input*

On entry: k_i, the integer which defines the required probabilities with $k_i = \mathrm{K}(j)$, $j = ((i-1) \bmod \mathrm{LK}) + 1$.

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

9: PLEK(*) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array PLEK must be at least $\max(\mathrm{LN}, \mathrm{LL}, \mathrm{LM}, \mathrm{LK})$.

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

10: PGTK(*) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array PGTK must be at least $\max(\mathrm{LN}, \mathrm{LL}, \mathrm{LM}, \mathrm{LK})$.

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

11: PEQK(*) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array PEQK must be at least $\max(\mathrm{LN}, \mathrm{LL}, \mathrm{LM}, \mathrm{LK})$.

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

12: IVALID(∗) – INTEGER array *Output*

Note: the dimension of the array IVALID must be at least max(LN, LL, LM, LK).

On exit: IVALID(i) indicates any errors with the input arguments, with

IVALID(i) = 0
No error.

IVALID(i) = 1

On entry, $n_i < 0$.

IVALID(i) = 2

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

IVALID(i) = 3

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

IVALID(i) = 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) = 5

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

IVALID(i) = 6

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

13: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 1

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.

IFAIL = 2

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

IFAIL $= 3$

On entry, array size $= \langle value \rangle$.
Constraint: LL > 0.

IFAIL $= 4$

On entry, array size $= \langle value \rangle$.
Constraint: LM > 0.

IFAIL $= 5$

On entry, array size $= \langle value \rangle$.
Constraint: LK > 0.

IFAIL $= -999$

Dynamic memory allocation failed.

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 Further Comments

The time taken by G01SLF 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.

9 Example

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

9.1 Program Text

```
    Program g01slfe
!     G01SLF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: g01slf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, lk, ll, lm, ln, lout
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: peqk(:), pgtk(:), plek(:)
      Integer, Allocatable             :: ivalid(:), k(:), l(:), m(:), n(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, mod, repeat
!     .. Executable Statements ..
      Write (nout,*) 'G01SLF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in the input vectors
      Read (nin,*) ln
      Allocate (n(ln))
      Read (nin,*) n(1:ln)
```

```
      Read (nin,*) ll
      Allocate (l(ll))
      Read (nin,*) l(1:ll)

      Read (nin,*) lm
      Allocate (m(lm))
      Read (nin,*) m(1:lm)

      Read (nin,*) lk
      Allocate (k(lk))
      Read (nin,*) k(1:lk)

!     Allocate memory for output
      lout = max(ln,ll,lm,lk)
      Allocate (pgtk(lout),plek(lout),peqk(lout),ivalid(lout))

!     Calculate probability
      ifail = -1
      Call g01slf(ln,n,ll,l,lm,m,lk,k,plek,pgtk,peqk,ivalid,ifail)

      If (ifail==0 .Or. ifail==1) Then
!       Display titles
        Write (nout,*) '    N         L         M         K    &
          &   PLEK      PGTK      PEQK    IVALID'
        Write (nout,*) repeat('-',78)

!       Display results
        Do i = 1, lout
          Write (nout,99999) n(mod(i-1,ln)+1), l(mod(i-1,ll)+1), &
            m(mod(i-1,lm)+1), k(mod(i-1,lk)+1), plek(i), pgtk(i), peqk(i), &
            ivalid(i)
        End Do
      End If

99999 Format (1X,4(I6,4X),3(F6.3,4X),I3)
    End Program g01slfe
```

9.2 Program Data

```
G01SLF 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
```

9.3 Program Results

```
 G01SLF 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
```