NAG C Library Function Document

nag_normal_scores_var (g01dcc)

1 Purpose

nag_normal_scores_var (g01dcc) computes an approximation to the variance-covariance matrix of an ordered set of independent observations from a Normal distribution with mean 0.0 and standard deviation 1.0.

2 Specification

```c
void nag_normal_scores_var (Integer n, double exp1, double exp2, double sumssq,
                          double vec[], NagError *fail)
```

3 Description

nag_normal_scores_var (g01dcc) is an adaptation of the Applied Statistics Algorithm AS 128, see Davis and Stephens (1978). An approximation to the variance-covariance matrix, $V$, using a Taylor series expansion of the Normal distribution function is discussed in David and Johnson (1954).

However, convergence is slow for extreme variances and covariances. The present function uses the David–Johnson approximation to provide an initial approximation and improves upon it by use of the following identities for the matrix.

For a sample of size $n$, let $m_i$ be the expected value of the $i$th largest order statistic, then:

(a) for any $i = 1, 2, \ldots, n$, $\sum_{j=1}^{n} V_{ij} = 1$

(b) $V_{12} = V_{11} + m_n^2 - m_{n-1}^2 - 1$

(c) the trace of $V$ is $tr(V) = n - \sum_{i=1}^{n} m_i^2$

(d) $V_{ij} = V_{ji} = V_{rs} = V_{sr}$ where $r = n + 1 - i$, $s = n + 1 - j$ and $i, j = 1, 2, \ldots, n$. Note that only the upper triangle of the matrix is calculated and returned column-wise in vector form.

4 References


5 Parameters

1:  
   `n` – Integer
   
   **Input**
   
   *On entry:* the sample size, $n$.
   
   *Constraint*: $n \geq 1$.

2:  
   `exp1` – double
   
   **Input**
   
   *On entry:* the expected value of the largest Normal order statistic, $m_n$, from a sample of size $n$.

3:  
   `exp2` – double
   
   **Input**
   
   *On entry:* the expected value of the second largest Normal order statistic, $m_{n-1}$, from a sample of size $n$. 

[NP3645/7]
4:  sumssq  – double
    Input
    On entry: the sum of squares of the expected values of the Normal order statistics from a sample of size n.

5:  vec[dim]  – double
    Output
    Note: the dimension, dim, of the array vec must be at least n × (n + 1)/2.
    On exit: the upper triangle of the n by n variance-covariance matrix packed by column. Thus element $V_{ij}$ is stored in vec[$i + j × (j - 1)/2 - 1$], for $1 \leq i \leq j \leq n$.

6:  fail – NagError *
    Input/Output
    The NAG error parameter (see the Essential Introduction).

6  Error Indicators and Warnings

NE_INT
    On entry, n = (value).
    Constraint: n ≥ 1.

NE_BAD_PARAM
    On entry, parameter (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 consult NAG for assistance.

7  Accuracy

For n ≤ 20, where comparison with the exact values can be made, the maximum error is less than 0.0001.

8  Further Comments

The time taken by nag_normal_scores_var (g01dcc) is approximately proportional to $n^2$.

The arguments exp1 ($= mn$), exp2 ($= mn - 1$) and sumssq ($= \sum_{j=1}^{n} m_j^2$) may be found from the expected values of the Normal order statistics obtained from nag_normal_scores_exact (g01dac).

9  Example

A program to compute the variance-covariance matrix for a sample of size 6. nag_normal_scores_exact (g01dac) is called to provide values for exp1, exp2 and sumssq.

9.1  Program Text

/* nag_normal_scores_var (g01dcc) Example Program. *
 * * Copyright 2001 Numerical Algorithms Group. *
 * * Mark 7, 2001. */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
/* Scalars */
  double errest, etol, exp1, exp2, sumssq;
  Integer exit_status, i, ifail, j, k, n, vec_elem;
  NagError fail;

/* Arrays */
  double *pp=0, *vec=0;

Vprintf("g01dcc Example Program Results\n");
INIT_FAIL(fail);
etol = 1e-4;
exit_status = 0;
n = 6;

/* Allocate memory */
if ( !(pp = NAG_ALLOC(n, double)) ||
    !(vec = NAG_ALLOC(n*(n+1)/2, double)) )
{
  Vprintf("Allocation failure\n");
  exit_status = -1;
  goto END;
}
g01dac(n, pp, etol, &errest, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from g01dac.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

exp1 = pp[5];
exp2 = pp[4];
sumssq = 0.0;
for (i = 1; i <= 6; ++i)
  sumssq += pp[i - 1] * pp[i - 1];
ifail = 0;
g01dcc(n, exp1, exp2, sumssq, vec, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from g01dac.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

Vprintf("Sample size = %2ld\n
", n);
Vprintf("Variance-covariance matrix\n");
k = 1;
for (j = 1; j <= n; ++j)
{
  vec_elem = 1;
  for (i = k; i <= k + j - 1; ++i)
  {
    Vprintf("%8.4f%s", vec[i - 1], vec_elem%6 == 0 ?"\n":" ");
    vec_elem++;
  }
  Vprintf("\n");
k += j;
}

END:
if (pp) NAG_FREE(pp);
if (vec) NAG_FREE(vec);
return exit_status;
}
9.2 Program Data

None.

9.3 Program Results

g01dcc Example Program Results

Sample size = 6

Variance-covariance matrix
0.4159
0.2085 0.2796
0.1394 0.1889 0.2462
0.1025 0.1397 0.1834 0.2462
0.0774 0.1060 0.1397 0.1889 0.2796
0.0563 0.0774 0.1025 0.1394 0.2085 0.4159