NAG Library Function Document

nag_ranks_and_scores (g01dhc)

1 Purpose

nag_ranks_and_scores (g01dhc) computes the ranks, Normal scores, an approximation to the Normal scores or the exponential scores as requested by you.

2 Specification

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

```
void nag_ranks_and_scores (Nag_Scores scores, Nag_Ties ties, Integer n,
     const double x[], double r[], NagError *fail)
```

3 Description

nag_ranks_and_scores (g01dhc) computes one of the following scores for a sample of observations, $x_1, x_2, \ldots, x_n$.

1. **Rank Scores**

   The ranks are assigned to the data in ascending order, that is the ith observation has score $s_i = k$ if it is the kth smallest observation in the sample.

2. **Normal Scores**

   The Normal scores are the expected values of the Normal order statistics from a sample of size n. If x_i is the kth smallest observation in the sample, then the score for that observation, s_i, is $E(Z_k)$ where Z_k is the kth order statistic in a sample of size n from a standard Normal distribution and E is the expectation operator.

3. **Blom, Tukey and van der Waerden Scores**

   These scores are approximations to the Normal scores. The scores are obtained by evaluating the inverse cumulative Normal distribution function, $\Phi^{-1}(\cdot)$, at the values of the ranks scaled into the interval $(0,1)$ using different scaling transformations.

   The Blom scores use the scaling transformation $\frac{r_i - \frac{3}{8}}{n + \frac{1}{4}}$ for the rank r_i, for $i = 1, 2, \ldots, n$. Thus the Blom score corresponding to the observation x_i is

$$s_i = \Phi^{-1}\left(\frac{r_i - \frac{3}{8}}{n + \frac{1}{4}}\right).$$

   The Tukey scores use the scaling transformation $\frac{r_i - \frac{1}{3}}{n + \frac{1}{3}}$; the Tukey score corresponding to the observation x_i is

$$s_i = \Phi^{-1}\left(\frac{r_i - \frac{1}{3}}{n + \frac{1}{3}}\right).$$

   The van der Waerden scores use the scaling transformation $\frac{r_i}{n+1}$; the van der Waerden score corresponding to the observation x_i is

$$s_i = \Phi^{-1}\left(\frac{r_i}{n + 1}\right).$$

   The van der Waerden scores may be used to carry out the van der Waerden test for testing for differences between several population distributions, see Conover (1980).

4. **Savage Scores**

   The Savage scores are the expected values of the exponential order statistics from a sample of size n. They may be used in a test discussed by Savage (1956) and Lehmann (1975). If x_i is the kth smallest observation in the sample, then the score for that observation is

$$s_i = E(Y_k) = \tfrac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{n-k+1},$$

   where Y_k is the kth order statistic in a sample of size n from a standard exponential distribution and E is the expectation operator.

Ties may be handled in one of five ways. Let $x_{t(i)}$, for $i = 1, 2, \ldots, m$, denote m tied observations, that is $x_{t(1)} = x_{t(2)} = \cdots = x_{t(m)}$ with $t(1) < t(2) < \cdots < t(m)$. If the rank of $x_{t(1)}$ is k, then if ties are ignored the rank of $x_{t(j)}$ will be $k + j - 1$. Let the scores ignoring ties be $s^*_{t(1)}, s^*_{t(2)}, \ldots, s^*_{t(m)}$. Then the scores, $s_{t(i)}$, for $i = 1, 2, \ldots, m$, may be calculated as follows:

–if averages are used, then $s_{t(i)} = \sum_{j=1}^{m} s^*_{t(j)}/m$;

–if the lowest score is used, then $s_{t(i)} = s^*_{t(1)}$;

–if the highest score is used, then $s_{t(i)} = s^*_{t(m)}$;

–if ties are to be broken randomly, then $s_{t(i)} = s^*_{t(I)}$ where $I \in \{\text{random permutation of } 1, 2, \ldots, m\}$;

–if ties are to be ignored, then $s_{t(i)} = s^*_{t(i)}$.

4 References

Blom G (1958) *Statistical Estimates and Transformed Beta-variables* Wiley

Conover W J (1980) *Practical Nonparametric Statistics* Wiley

Lehmann E L (1975) *Nonparametrics: Statistical Methods Based on Ranks* Holden–Day

Savage I R (1956) Contributions to the theory of rank order statistics – the two-sample case *Ann. Math. Statist.* **27** 590–615

Tukey J W (1962) The future of data analysis *Ann. Math. Statist.* **33** 1–67

5 Arguments

1: **scores** – Nag_Scores *Input*

On entry: indicates which of the following scores are required.

scores = Nag_RankScores
The ranks.

scores = Nag_NormalScores
The Normal scores, that is the expected value of the Normal order statistics.

scores = Nag_BlomScores
The Blom version of the Normal scores.

scores = Nag_TukeyScores
The Tukey version of the Normal scores.

scores = Nag_WaerdenScores
The van der Waerden version of the Normal scores.

scores = Nag_SavageScores
The Savage scores, that is the expected value of the exponential order statistics.

Constraint: **scores** = Nag_RankScores, Nag_NormalScores, Nag_BlomScores, Nag_TukeyScores, Nag_WaerdenScores or Nag_SavageScores.

2: **ties** – Nag_Ties *Input*

On entry: indicates which of the following methods is to be used to assign scores to tied observations.

ties = Nag_AverageTies
The average of the scores for tied observations is used.

ties = Nag_LowestTies
The lowest score in the group of ties is used.

ties = Nag_HighestTies
The highest score in the group of ties is used.

ties = Nag_RandomTies
The repeatable random number generator is used to randomly untie any group of tied observations.

ties = Nag_IgnoreTies
Any ties are ignored, that is the scores are assigned to tied observations in the order that they appear in the data.

Constraint: **ties** = Nag_AverageTies, Nag_LowestTies, Nag_HighestTies, Nag_RandomTies or Nag_IgnoreTies.

3: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: **n** ≥ 1.

4: **x**[**n**] – const double *Input*

On entry: the sample of observations, x_i, for $i = 1, 2, \ldots, n$.

5: **r**[**n**] – double *Output*

On exit: contains the scores, s_i, for $i = 1, 2, \ldots, n$, as specified by **scores**.

6: **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_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

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.

7 Accuracy

For **scores** = Nag_RankScores, the results should be accurate to ***machine precision***.

For **scores** = Nag_SavageScores, the results should be accurate to a small multiple of ***machine precision***.

For **scores** = Nag_NormalScores, the results should have a relative accuracy of at least $\max(100 \times \epsilon, 10^{-8})$ where ϵ is the ***machine precision***.

For **scores** = Nag_BlomScores, Nag_TukeyScores or Nag_WaerdenScores, the results should have a relative accuracy of at least $\max(10 \times \epsilon, 10^{-12})$.

8 Parallelism and Performance

Not applicable.

9 Further Comments

If more accurate Normal scores are required nag_normal_scores_exact (g01dac) should be used with appropriate settings for the input argument **etol**.

10 Example

This example computes and prints the Savage scores for a sample of five observations. The average of the scores of any tied observations is used.

10.1 Program Text

```
/* nag_ranks_and_scores (g01dhc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 * Mark 8 revised, 2004.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{

  Integer  exit_status = 0, i, n;
  NagError fail;
  double   *r = 0, *x = 0;

  INIT_FAIL(fail);

  printf("nag_ranks_and_scores (g01dhc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld ", &n);
  if (n >= 1)
    {
      if (!(r = NAG_ALLOC(n, double)) ||
          !(x = NAG_ALLOC(n, double)))
```

```
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }
  for (i = 1; i <= n; ++i)
    scanf("%lf ", &x[i - 1]);

  /* nag_ranks_and_scores (g01dhc).
   * Ranks, Normal scores, approximate Normal scores or
   * exponential (Savage) scores
   */
  nag_ranks_and_scores(Nag_SavageScores, Nag_AverageTies, n, x, r,
                       &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_ranks_and_scores (g01dhc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("The Savage Scores : \n");
  printf("  (Average scores are used for tied observations)\n\n");
  for (i = 1; i <= n; ++i)
    printf("%10.4f\n", r[i - 1]);
 END:
  NAG_FREE(r);
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_ranks_and_scores (g01dhc) Example Program Data
5
2 0 2 2 0
```

10.3 Program Results

```
nag_ranks_and_scores (g01dhc) Example Program Results

The Savage Scores :
  (Average scores are used for tied observations)

    1.4500
    0.3250
    1.4500
    1.4500
    0.3250
```