NAG Library Function Document

nag_rank_ci_1var (g07eac)

1 Purpose

nag_rank_ci_1var (g07eac) computes a rank based (nonparametric) estimate and confidence interval for the location argument of a single population.

2 Specification

```
#include <nag.h>
#include <nagg07.h>
void nag_rank_ci_1var (Nag_RCIMethod method, Integer n, const double x[],
     double clevel, double *theta, double *thetal, double *thetau,
     double *estcl, double *wlower, double *wupper, NagError *fail)
```

3 Description

Consider a vector of independent observations, $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ with unknown common symmetric density $f(x_i - \theta)$. nag_rank_ci_1var (g07eac) computes the Hodges–Lehmann location estimator (see Lehmann (1975)) of the centre of symmetry θ, together with an associated confidence interval. The Hodges–Lehmann estimate is defined as

$$\hat{\theta} = \text{median}\left\{\frac{x_i + x_j}{2}, 1 \leq i \leq j \leq n\right\}.$$

Let $m = (n(n+1))/2$ and let a_k, for $k = 1, 2, \ldots, m$ denote the m ordered averages $\left(x_i + x_j\right)/2$ for $1 \leq i \leq j \leq n$. Then

if m is odd, $\hat{\theta} = a_k$ where $k = (m+1)/2$;

if m is even, $\hat{\theta} = (a_k + a_{k+1})/2$ where $k = m/2$.

This estimator arises from inverting the one-sample Wilcoxon signed-rank test statistic, $W(x - \theta_0)$, for testing the hypothesis that $\theta = \theta_0$. Effectively $W(x - \theta_0)$ is a monotonically decreasing step function of θ_0 with

$$\text{mean } (W) = \mu = \frac{n(n+1)}{4},$$

$$\text{var } (W) = \sigma^2 = \frac{n(n+1)(2n+1)}{24}.$$

The estimate $\hat{\theta}$ is the solution to the equation $W\left(x - \hat{\theta}\right) = \mu$; two methods are available for solving this equation. These methods avoid the computation of all the ordered averages a_k; this is because for large n both the storage requirements and the computation time would be excessive.

The first is an exact method based on a set partitioning procedure on the set of all ordered averages $\left(x_i + x_j\right)/2$ for $i \leq j$. This is based on the algorithm proposed by Monahan (1984).

The second is an iterative algorithm, based on the Illinois method which is a modification of the *regula falsi* method, see McKean and Ryan (1977). This algorithm has proved suitable for the function $W(x - \theta_0)$ which is asymptotically linear as a function of θ_0.

The confidence interval limits are also based on the inversion of the Wilcoxon test statistic.

Given a desired percentage for the confidence interval, $1 - \alpha$, expressed as a proportion between 0 and 1, initial estimates for the lower and upper confidence limits of the Wilcoxon statistic are found from

$$W_l = \mu - 0.5 + \left(\sigma\Phi^{-1}(\alpha/2)\right)$$

and

$$W_u = \mu + 0.5 + \left(\sigma\Phi^{-1}(1-\alpha/2)\right),$$

where Φ^{-1} is the inverse cumulative Normal distribution function.

W_l and W_u are rounded to the nearest integer values. These estimates are then refined using an exact method if $n \leq 80$, and a Normal approximation otherwise, to find W_l and W_u satisfying

$$P(W \leq W_l) \leq \alpha/2$$
$$P(W \leq W_l + 1) > \alpha/2$$

and

$$P(W \geq W_u) \leq \alpha/2$$
$$P(W \geq W_u - 1) > \alpha/2.$$

Let $W_u = m - k$; then $\theta_l = a_{k+1}$. This is the largest value θ_l such that $W(x - \theta_l) = W_u$.

Let $W_l = k$; then $\theta_u = a_{m-k}$. This is the smallest value θ_u such that $W(x - \theta_u) = W_l$.

As in the case of $\hat{\theta}$, these equations may be solved using either the exact or the iterative methods to find the values θ_l and θ_u.

Then (θ_l, θ_u) is the confidence interval for θ. The confidence interval is thus defined by those values of θ_0 such that the null hypothesis, $\theta = \theta_0$, is not rejected by the Wilcoxon signed-rank test at the $(100 \times \alpha)\%$ level.

4 References

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

Marazzi A (1987) Subroutines for robust estimation of location and scale in ROBETH *Cah. Rech. Doc. IUMSP, No. 3 ROB 1* Institut Universitaire de Médecine Sociale et Préventive, Lausanne

McKean J W and Ryan T A (1977) Algorithm 516: An algorithm for obtaining confidence intervals and point estimates based on ranks in the two-sample location problem *ACM Trans. Math. Software* **10** 183–185

Monahan J F (1984) Algorithm 616: Fast computation of the Hodges–Lehman location estimator *ACM Trans. Math. Software* **10** 265–270

5 Arguments

1: **method** – Nag_RCIMethod *Input*

On entry: specifies the method to be used.

method = Nag_RCI_Exact
The exact algorithm is used.

method = Nag_RCI_Approx
The iterative algorithm is used.

Constraint: **method** = Nag_RCI_Exact or Nag_RCI_Approx.

2: **n** – Integer *Input*

On entry: n, the sample size.

Constraint: **n** ≥ 2.

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

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

4: **clevel** – double *Input*

On entry: the confidence interval desired.

For example, for a 95% confidence interval set **clevel** $= 0.95$.

Constraint: $0.0 <$ **clevel** < 1.0.

5: **theta** – double * *Output*

On exit: the estimate of the location, $\hat{\theta}$.

6: **thetal** – double * *Output*

On exit: the estimate of the lower limit of the confidence interval, θ_l.

7: **thetau** – double * *Output*

On exit: the estimate of the upper limit of the confidence interval, θ_u.

8: **estcl** – double * *Output*

On exit: an estimate of the actual percentage confidence of the interval found, as a proportion between $(0.0, 1.0)$.

9: **wlower** – double * *Output*

On exit: the upper value of the Wilcoxon test statistic, W_u, corresponding to the lower limit of the confidence interval.

10: **wupper** – double * *Output*

On exit: the lower value of the Wilcoxon test statistic, W_l, corresponding to the upper limit of the confidence interval.

11: **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_CONVERGENCE

Warning. The iterative procedure to find an estimate of the lower confidence point had not converged in 100 iterations.

Warning. The iterative procedure to find an estimate of Theta had not converged in 100 iterations.

Warning. The iterative procedure to find an estimate of the upper confidence point had not converged in 100 iterations.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 2$.

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.

NE_REAL

On entry, **clevel** is out of range: **clevel** $= \langle value \rangle$.

NE_SAMPLE_IDEN

Not enough information to compute an interval estimate since the whole sample is identical. The common value is returned in **theta**, **thetal** and **thetau**.

7 Accuracy

nag_rank_ci_1var (g07eac) should produce results accurate to five significant figures in the width of the confidence interval; that is the error for any one of the three estimates should be less than $0.00001 \times (\mathbf{thetau} - \mathbf{thetal})$.

8 Parallelism and Performance

nag_rank_ci_1var (g07eac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken increases with the sample size n.

10 Example

The following program calculates a 95% confidence interval for θ, a measure of symmetry of the sample of 50 observations.

10.1 Program Text

```
/* nag_rank_ci_1var (g07eac) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg07.h>

int main(void)
{

  /* Scalars */
  double   clevel, estcl, theta, thetal, thetau, wlower, wupper;
  Integer  exit_status, i, n;
  NagError fail;
```

```
  /* Arrays */
  double   *x = 0;

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_rank_ci_1var (g07eac) Example Program Results\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &n);

  /* Allocate memory */
  if (!(x = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (i = 1; i <= n; ++i)
    scanf("%lf", &x[i - 1]);
  scanf("%*[^\n] ");
  scanf("%lf%*[^\n] ", &clevel);

  /* nag_rank_ci_1var (g07eac).
   * Robust confidence intervals, one-sample
   */
  nag_rank_ci_1var(Nag_RCI_Exact, n, x, clevel, &theta, &thetal, &thetau,
                   &estcl, &wlower, &wupper, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rank_ci_1var (g07eac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n");
  printf(" Location estimator     Confidence Interval\n");
  printf("\n");
  printf("%10.4f            ( %6.4f , %6.4f )\n", theta, thetal,
          thetau);
  printf("\n");
  printf(" Corresponding Wilcoxon statistics\n");
  printf("\n");
  printf(" Lower : %8.2f\n", wlower);
  printf(" Upper : %8.2f\n", wupper);

 END:
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_rank_ci_1var (g07eac) Example Program Data
 40
 -0.23   0.35  -0.77   0.35   0.27   -0.72   0.08  -0.40  -0.76   0.45
  0.73   0.74   0.83  -0.87   0.21    0.29  -0.91  -0.04   0.82  -0.38
 -0.31   0.24  -0.47  -0.68  -0.77   -0.86  -0.59   0.73   0.39  -0.44
  0.63  -0.22  -0.07  -0.43  -0.21   -0.31   0.64  -1.00  -0.86  -0.73
  0.95
```

10.3 Program Results

```
nag_rank_ci_1var (g07eac) Example Program Results

 Location estimator     Confidence Interval

   -0.1300            ( -0.3300 , 0.0350 )

 Corresponding Wilcoxon statistics

 Lower :   556.00
 Upper :   264.00
```