NAG C Library Function Document

nag_runs_test (g08eac)

1 Purpose

nag_runs_test (g08eac) performs a runs up (or a runs down) test on a sequence of observations.

2 Specification

```
#include <nag.h>
#include <nagg08.h>

void nag_runs_test (Integer n, const double x[], Integer max_run, Integer *nruns,
      double *chi, double *df, double *prob, NagError *fail)
```

3 Description

Runs tests may be used to investigate for trends in a sequence of observations. nag_runs_test (g08eac) computes statistics for the runs up test. If the runs down test is desired then each observation must be multiplied by -1 before nag_runs_test (g08eac) is called with the modified vector of observations.

A run up is a sequence of numbers in increasing order. A run up ends at x_k when $x_k > x_{k+1}$ and the new run then begins at x_{k+1}. nag_runs_test (g08eac) counts the number of runs up of different lengths. Let c_i denote the number of runs of length i, for $i = 1, 2, \ldots, r - 1$. The number of runs of length r or greater is then denoted by c_r. An unfinished run at the end of a sequence is not counted. The following is a trivial example.

Suppose we called nag_runs_test (g08eac) with the following sequence:

0.20 0.40 0.45 0.40 0.15 0.75 0.95 0.23 0.27 0.40 0.25 0.10 0.34 0.39 0.61 0.12.

Then nag_runs_test (g08eac) would have counted the runs up of the following lengths:

3, 1, 3, 3, 1, and 4.

When the counting of runs is complete nag_runs_test (g08eac) computes the expected values and covariances of the counts, c_i. For the details of the method used see Knuth (1981). An approximate χ^2 statistic with r degrees of freedom is computed, where

$$X^2 = (c - \mu_c)^{\mathrm{T}} \Sigma_c^{-1} (c - \mu_c)$$

where

c is the vector of counts, c_i, for $i = 1, 2, \ldots, r$,
μ_c is the vector of expected values, e_i, for $i = 1, 2, \ldots, r$, where e_i is the expected value for c_i under the null hypothesis of randomness, and
Σ_c is the covariance matrix of c under the null hypothesis.

The use of the χ^2 distribution as an approximation to the exact distribution of the test statistic improves as the expected values increase.

The user may specify the total number of runs to be found. If the specified number of runs is found before the end of a sequence nag_runs_test (g08eac) will exit before counting any further runs. The number of runs actually counted and used to compute the test statistic is returned via **nruns**.

4 References

Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

Morgan B J T (1984) *Elements of Simulation* Chapman and Hall

Ripley B D (1987) *Stochastic Simulation* Wiley

5 Arguments

1: **n** – Integer *Input*

On entry: the length of the current sequence of observations, n.

Constraint: $\mathbf{n} \geq 3$.

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

On entry: the sequence of observations.

3: **max_run** – Integer *Input*

On entry: the length of the longest run for which tabulation is desired, r. That is, all runs with length greater than or equal to r are counted together.

Constraint: **max_run** ≥ 1 and **max_run** $<$ **n**.

4: **nruns** – Integer * *Output*

On exit: the number of runs actually found.

5: **chi** – double * *Output*

On exit: contains the approximate χ^2 test statistic, X^2.

6: **df** – double * *Output*

On exit: contains the degrees of freedom of the χ^2 statistic.

7: **prob** – double * *Output*

On exit: contains the upper tail probability corresponding to the χ^2 test statistic, i.e., the significance level.

8: **fail** – NagError * *Input/Output*

The NAG error parameter, see the Essential Introduction.

6 Error Indicators and Warnings

NE_2_INT_ARG_GE

On entry, **max_run** $= \langle value \rangle$ while **n** $= \langle value \rangle$. These parameters must satisfy **max_run** $<$ **n**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_G08EA_COVAR

Internally computed covariance matrix is not positive-definite. This may be because the value of **max_run** is too large relative to the full length of the series. Thus the approximate χ^2 test statistic cannot be computed.

NE_G08EA_RUNS

The number of runs requested were not found. All statistics are still computed and the information returned may still be of use.

NE_G08EA_RUNS_LENGTH

The total length of the runs found is less than **max_run**.

NE_G08EA_TIE

There is a tie in the sequence of observations.

NE_INT_ARG_LT

On entry, **max_run** must not be less than 1: **max_run** = ⟨*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

The computations are believed to be stable. The computation of **prob** given the values of **chi** and **df** will obtain a relative accuracy of five significant figures for most cases.

8 Further Comments

The time taken by nag_runs_test (g08eac) increases with the number of observations n.

9 Example

The following program performs a runs up test on 10000 pseudo-random numbers taken from a uniform distribution $U(0,1)$, generated by nag_random_continuous_uniform (g05cac). All runs of length 6 or more are counted together.

9.1 Program Text

```
/* nag_runs_test (g08eac) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 *
 * Mark 8 revised, 2004
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>
#include <nagg08.h>

int main(void)
{
  Integer exit_status=0, igen = 0, init, iseed[] = {0, 0, 0, 0}, max_run, n;
  Integer nruns;
  NagError fail;
  double chi, df, enda, endb, p, *x=0;

  INIT_FAIL(fail);
  Vprintf("nag_runs_test (g08eac) Example Program Results\n");

  n = 10000;
  if (!(x = NAG_ALLOC(n, double)))
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  init = 0;
  /* nag_rngs_init_repeatable (g05kbc).
   * Initialize seeds of a given generator for random number
   * generating functions (that pass seeds explicitly) to give
   * a repeatable sequence
   */
  nag_rngs_init_repeatable(&igen, iseed);
  enda = 0.0;
  endb = 1.0;
  /* nag_rngs_uniform (g05lgc).
   * Generates a vector of random numbers from a uniform
   * distribution, seeds and generator number passed
   * explicitly
   */
  nag_rngs_uniform(enda, endb, n, x, igen, iseed, NAGERR_DEFAULT);
  max_run = 6;
  /* nag_runs_test (g08eac).
   * Performs the runs up or runs down test for randomness
   */
  nag_runs_test(n, x, max_run, &nruns, &chi, &df, &p, &fail);

  if (fail.code == NE_NOERROR || fail.code == NE_G08EA_RUNS)
    {
      Vprintf("\n");
      Vprintf("%s%10ld\n", "Total number of runs found = ", nruns);
      if (fail.code == NE_G08EA_RUNS)
        Vprintf("%s\n", " ** Note : the number of runs requested were not "
                "found.");
      Vprintf("\n");
      Vprintf("%s%10.4f\n", "Chisq = ", chi);
      Vprintf("%s%8.2f\n", "DF    = ", df);
      Vprintf("%s%10.4f\n", "Prob  = ", p);
    }
  else
    {
      Vprintf("Error from nag_runs_test (g08eac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  if (x) NAG_FREE(x);
  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
nag_runs_test (g08eac) Example Program Results

Total number of runs found =       4970

Chisq =     9.7612
DF    =     6.00
Prob  =     0.1351
```