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 \(i = 1, 2, \ldots, r\). 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:

\[
0.20 \ 0.40 \ 0.45 \ 0.40 \ 0.15 \ 0.75 \ 0.23 \ 0.27 \ 0.40 \ 0.25 \ 0.10 \ 0.34 \ 0.39 \ 0.61 \ 0.12.
\]

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 \(\chi^2\) statistic with \(r\) degrees of freedom is computed, where

\[
X^2 = (c - \mu_c)^T \Sigma_c^{-1} (c - \mu_c),
\]

where
- \(c\) is the vector of counts, \(c_i\), for \(i = 1, 2, \ldots, r\),
- \(\mu_c\) 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
- \(\Sigma_c\) is the covariance matrix of \(c\) under the null hypothesis.

The use of the \(\chi^2\)-distribution as an approximation to the exact distribution of the test statistic, \(X^2\), improves as the length of the sequence relative to \(m\) increases and hence the expected value, \(e\), increases.

You 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 \texttt{nruns}. 

Mark 25
4 References
Ripley B D (1987) Stochastic Simulation Wiley

5 Arguments
1: n – Integer 
   Input
   On entry: n, the length of the current sequence of observations.
   Constraint: n ≥ 3.

2: x[n] – const double 
   Input
   On entry: the sequence of observations.

3: max_run – Integer 
   Input
   On entry: r, the length of the longest run for which tabulation is desired. That is, all runs with
   length greater than or equal to r are counted together.
   Constraint: 1 ≤ max_run < n.

4: nruns – Integer* 
   Output
   On exit: the number of runs actually found.

5: chi – double* 
   Output
   On exit: contains the approximate $\chi^2$ test statistic, $X^2$.

6: df – double* 
   Output
   On exit: contains the degrees of freedom of the $\chi^2$ statistic.

7: prob – double* 
   Output
   On exit: contains the upper tail probability corresponding to the $\chi^2$ test statistic, i.e., the
   significance level.

8: fail – NagError* 
   Input/Output
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_GE
   On entry, max_run = (value) and n = (value).
   Constraint: max_run < n.

NE_ALLOC_FAIL
   Dynamic memory allocation failed.
   See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
   On entry, argument (value) had an illegal value.
Internally computed covariance matrix is not positive definite. This may be because the value of \texttt{max\_run} is too large relative to the full length of the series. Thus the approximate $\chi^2$ test statistic cannot be computed.

The number of runs requested were not found, only \textit{value} out of the requested \textit{value} where found. All statistics are returned and may still be of use.

The total length of the runs found is less than \texttt{max\_run}. \texttt{max\_run} = \textit{value} whereas the total length of all runs is \textit{value}.

There is a tie in the sequence of observations.

On entry, \texttt{max\_run} = \textit{value}. Constraint: \texttt{max\_run} \geq 1.

On entry, \texttt{n} = \textit{value}. Constraint: \texttt{n} \geq 3.

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.

An unexpected error has been triggered by this function. Please contact NAG. See Section 3.6.6 in the Essential Introduction for further information.

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

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

\texttt{g08\_a\_tietests} is not threaded by NAG in any implementation. \texttt{g08\_a\_tietests} makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

The time taken by \texttt{g08\_a\_tietests} increases with the number of observations \texttt{n}.

\texttt{Mark 25}
10 Example

The following program performs a runs up test on 10000 pseudorandom numbers taken from a uniform distribution $U(0,1)$, generated by nag_rand_uniform (g05sqc). All runs of length 6 or more are counted together.

10.1 Program Text

/* nag_runs_test (g08eac) Example Program. */
/* Copyright 2014 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 scalar and array declarations */
    Integer exit_status = 0;
    Integer nruns, lstate;
    Integer *state = 0;
    /* NAG structures */
    NagError fail;

    /* Double scalar and array declarations */
    double chi, df, p, *x = 0;

    /* Choose the base generator */
    Nag_BaseRNG genid = Nag_Basic;
    Integer subid = 0;

    /* Set the seed */
    Integer seed[] = { 324213 };  
    Integer lseed = 1;

    /* Set the size of the (randomly generated) dataset */
    Integer n = 10000;

    /* Set the the length of the longest run */
    Integer max_run = 6;

    /* Initialise the error structure */
    INIT_FAIL(fail);

    printf("nag_runs_test (g08eac) Example Program Results\n");

    /* Get the length of the state array */
    lstate = -1;
    nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_rand_init_repeatable (g05kfc).\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Allocate arrays */
    if (!(*x = NAG_ALLOC(n, double)) || 
        !(state = NAG_ALLOC(lstate, Integer)))

END:
{  
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/*/ Initialise the generator to a repeatable sequence */
if (fail.code != NE_NOERROR)  
    {
        printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",  
            fail.message);
        exit_status = 1;
        goto END;
    }

/*/ Generate vector of n uniform variates between 0.0 and 1.0 */
if (fail.code == NE_NOERROR || fail.code == NE_G08EA_RUNS)  
    {
        printf("Total number of runs found = %10NAG_IFMT"\n",  
            nruns);
        if (fail.code == NE_G08EA_RUNS)  
            printf("** Note : the number of runs requested were not found.\n");
        printf("\n");
        printf("Chisq = %10.4f\n", chi);
        printf("DF = %8.2f\n", df);
        printf("Prob = %10.4f\n", p);
    }
else  
    {
        printf("Error from nag_runs_test (g08eac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

END:
NAG_FREE(x);
NAG_FREE(state);

return exit_status;

10.2 Program Data
None.

10.3 Program Results

nag_runs_test (g08eac) Example Program Results

Total number of runs found = 5024
Chisq = 1.8717
DF = 6.00
Prob = 0.9311