NAG Library Routine Document

G08EAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G08EAF performs a runs up (or a runs down) test on a sequence of observations.

2 Specification

```
SUBROUTINE G08EAF (CL, N, X, M, MAXR, NRUNS, NCOUNT, EX, COV, LDCOV,       &
                   CHI, DF, PROB, WRK, LWRK, IFAIL)

INTEGER            N, M, MAXR, NRUNS, NCOUNT(MAXR), LDCOV, LWRK, IFAIL
REAL (KIND=nag_wp) X(N), EX(MAXR), COV(LDCOV,MAXR), CHI, DF, PROB,         &
                   WRK(LWRK)
CHARACTER(1)       CL
```

3 Description

Runs tests may be used to investigate for trends in a sequence of observations. G08EAF computes statistics for the runs up test. If the runs down test is desired then each observation must be multiplied by -1 before G08EAF is called with the modified vector of observations. G08EAF may be used in two different modes:

(i) a single call to G08EAF which computes all test statistics after counting the runs;

(ii) multiple calls to G08EAF with the final test statistics only being computed in the last call.

The second mode is necessary if all the data do not fit into the memory. See argument CL in Section 5 for details on how to invoke each mode.

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}. G08EAF 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 unless the sequence is part of an initial or intermediate call to G08EAF (i.e., unless there is another call to G08EAF to follow) in which case the unfinished run is used together with the beginning of the next sequence of numbers input to G08EAF in the next call. The following is a trivial example.

Suppose we called G08EAF twice with the following two sequences:

(0.20 0.40 0.45 0.40 0.15 0.75 0.95 0.23) and

(0.27 0.40 0.25 0.10 0.34 0.39 0.61 0.12).

Then after the second call G08EAF would have counted the runs up of the following lengths:

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

When the counting of runs is complete G08EAF 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, 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 G08EAF 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: CL – CHARACTER(1) *Input*

On entry: must specify the type of call to G08EAF.

CL = 'S'
This is the one and only call to G08EAF (single call mode). All data are to be input at once. All test statistics are computed after the counting of runs is complete.

CL = 'F'
This is the first call to the routine. All initializations are carried out and the counting of runs begins. The final test statistics are not computed since further calls will be made to G08EAF.

CL = 'I'
This is an intermediate call during which the counts of runs are updated. The final test statistics are not computed since further calls will be made to G08EAF.

CL = 'L'
This is the last call to G08EAF. The test statistics are computed after the final counting of runs is completed.

Constraint: CL = 'S', 'F', 'I' or 'L'.

2: N – INTEGER *Input*

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

Constraints:

if CL = 'S', N $\geq$ 3;
otherwise N $\geq$ 1.

3: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: the sequence of observations.

4: M – INTEGER *Input*

On entry: the maximum number of runs to be sought. If $M \leq 0$ then no limit is placed on the number of runs that are found.

M must not be changed between calls to G08EAF.

Constraint: if $M \leq N$, CL = 'S'.

5: MAXR – 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.

MAXR must not be changed between calls to G08EAF.

Constraint: $MAXR \geq 1$ and if CL = 'S', $MAXR < N$.

6: NRUNS – INTEGER *Input/Output*

On entry: if CL = 'S' or 'F', NRUNS need not be set.

If CL = 'I' or 'L', NRUNS must contain the value returned by the previous call to G08EAF.

On exit: the number of runs actually found.

7: NCOUNT(MAXR) – INTEGER array *Input/Output*

On entry: if CL = 'S' or 'F', NCOUNT need not be set.

If CL = 'I' or 'L', NCOUNT must contain the values returned by the previous call to G08EAF.

On exit: the counts of runs of the different lengths, c_i, for $i = 1, 2, \ldots, r$.

8: EX(MAXR) – REAL (KIND=nag_wp) array *Output*

On exit: if CL = 'S' or 'L', (i.e., if it is the final exit) then EX contains the expected values of the counts, e_i, for $i = 1, 2, \ldots, r$.

Otherwise the elements of EX are not set.

9: COV(LDCOV, MAXR) – REAL (KIND=nag_wp) array *Output*

On exit: if CL = 'S' or 'L' (i.e., if it is the final exit) then COV contains the covariance matrix of the counts, Σ_c.

Otherwise the elements of COV are not set.

10: LDCOV – INTEGER *Input*

On entry: the first dimension of the array COV as declared in the (sub)program from which G08EAF is called.

Constraint: LDCOV $\geq$ MAXR.

11: CHI – REAL (KIND=nag_wp) *Output*

On exit: if CL = 'S' or 'L' (i.e., if it is the final exit) then CHI contains the approximate χ^2 test statistic, X^2.

Otherwise CHI is not set.

12: DF – REAL (KIND=nag_wp) *Output*

On exit: if CL = 'S' or 'L' (i.e., if it is the final exit) then DF contains the degrees of freedom of the χ^2 statistic.

Otherwise DF is not set.

13: PROB – REAL (KIND=nag_wp) *Output*

On exit: if CL = 'S' or 'L', (i.e., if it is the final exit) then PROB contains the upper tail probability corresponding to the χ^2 test statistic, i.e., the significance level.

Otherwise PROB is not set.

14: WRK(LWRK) – REAL (KIND=nag_wp) array *Workspace*

15: LWRK – INTEGER *Input*

On entry: the dimension of the array WRK as declared in the (sub)program from which G08EAF is called.

Constraint: LWRK $\geq \frac{\text{MAXR}\times(\text{MAXR}+5)}{2} + 1$.

16: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: G08EAF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL = 1

On entry, CL = ⟨*value*⟩.
Constraint: CL = 'S', 'F', 'I' or 'L'.

IFAIL = 2

On entry, N = ⟨*value*⟩.
Constraint: if CL = 'S', N $\geq$ 3, otherwise N $\geq$ 1.

IFAIL = 3

On entry, M = ⟨*value*⟩ and N = ⟨*value*⟩.
Constraint: if CL = 'S', M $\leq$ N.

IFAIL = 4

On entry, MAXR = ⟨*value*⟩.
Constraint: MAXR $\geq$ 1.

On entry, MAXR = ⟨*value*⟩ and N = ⟨*value*⟩.
Constraint: if CL = 'S', MAXR $<$ N.

IFAIL $= 5$

On entry, LDCOV $= \langle value \rangle$ and MAXR $= \langle value \rangle$.
Constraint: LDCOV $\geq$ MAXR.

IFAIL $= 6$

On entry, LWRK $= \langle value \rangle$.
Constraint: LWRK $\geq$ MAXR $\times$ (MAXR $+ 5)/2 + 1 = \langle value \rangle$.

IFAIL $= 7$

There is a tie in the sequence of observations.

IFAIL $= 8$

The total length of the runs found is less than MAXR.
MAXR $= \langle value \rangle$ whereas the total length of all runs is $\langle value \rangle$.

IFAIL $= 9$

The covariance matrix stored in COV is not positive definite, thus the approximate χ^2 test statistic cannot be computed.
This may be because MAXR is too large relative to the length of the full sequence.

IFAIL $= 10$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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 Parallelism and Performance

G08EAF is not thread safe and should not be called from a multithreaded user program. Please see Section 3.12.1 in How to Use the NAG Library and its Documentation for more information on thread safety.

G08EAF 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 routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by G08EAF increases with the number of observations n, and also depends to some extent on whether the call to G08EAF is an only, first, intermediate or last call.

10 Example

The following program performs a runs up test on 500 pseudorandom numbers. G08EAF is called 5 times with 100 observations each time. No limit is placed on the number of runs to be counted. All runs of length 6 or more are counted together.

10.1 Program Text

```
    Program g08eafe

!     G08EAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g08eaf, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: chi, df, prob
      Integer                          :: i, ifail, ldcov, lwrk, m, maxr, n,   &
                                          nruns, nsamp, pn
      Character (1)                    :: cl
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: cov(:,:), ex(:), wrk(:), x(:)
      Integer, Allocatable             :: ncount(:)
!     .. Executable Statements ..
      Write (nout,*) 'G08EAF Example Program Results'
      Write (nout,*)

!     Skip main heading in data file
      Read (nin,*)

!     Read in number of samples
      Read (nin,*) nsamp, m, maxr

      ldcov = maxr
      lwrk = maxr*(maxr+5)/2 + 1
      Allocate (ncount(maxr),cov(ldcov,maxr),ex(maxr),wrk(lwrk),x(1))

      If (nsamp==1) Then
        cl = 'S'
      Else
        cl = 'F'
      End If

      pn = 0
      Do i = 1, nsamp
!       Skip run heading in data file
        Read (nin,*)

!       Read in sample size
        Read (nin,*) n

        If (n>pn) Then
!         Reallocate X if required
```

```
          Deallocate (x)
          Allocate (x(n))
          pn = n
        End If

!       Read in the sample
        Read (nin,*) x(1:n)

!       Process the sample
        ifail = -1
        Call g08eaf(cl,n,x,m,maxr,nruns,ncount,ex,cov,ldcov,chi,df,prob,wrk,   &
          lwrk,ifail)
        If (ifail/=0 .And. ifail/=10) Then
          Go To 100
        End If

!       Adjust CL for intermediate calls
        If (i<nsamp-1) Then
          cl = 'I'
        Else
          cl = 'L'
        End If

      End Do

!     Display results
      Write (nout,99999) 'Total number of runs found = ', nruns
      If (ifail==10) Then
        Write (nout,*)                                                        &
          ' ** Note : the number of runs requested were not found.'
      End If
      Write (nout,*)
      Write (nout,*) '                              Count'
      Write (nout,*)                                                          &
        '          1        2        3        4        5       >5'
      Write (nout,99998) ncount(1:maxr)
      Write (nout,*)
      Write (nout,*) '                              Expect'
      Write (nout,*)                                                          &
        '          1        2        3        4        5       >5'
      Write (nout,99997) ex(1:maxr)
      Write (nout,*)
      Flush (nout)
      ifail = 0
      Call x04caf('General',' ',maxr,maxr,cov,ldcov,'Covariance matrix',ifail)
      Write (nout,*)
      Write (nout,99996) 'Chisq = ', chi
      Write (nout,99995) 'DF    = ', df
      Write (nout,99996) 'Prob  = ', prob

100   Continue

99999 Format (1X,A,I10)
99998 Format (3X,6I9)
99997 Format (3X,6F9.1)
99996 Format (1X,A,F10.4)
99995 Format (1X,A,F7.1)
    End Program g08eafe
```

10.2 Program Data

```
G08EAF Example Program Data
5  0  6            :: NSAMP,M,MAXR
## Sample 1
100                :: N
0.11389 0.84996 0.84821 0.18431 0.14104 0.03144 0.68013 0.13297 0.27696 0.86743
0.32674 0.87990 0.85580 0.47830 0.75318 0.93643 0.19396 0.31091 0.34956 0.94923
0.18940 0.24715 0.62503 0.50406 0.05686 0.26481 0.68746 0.80387 0.48184 0.25034
0.20141 0.35062 0.58591 0.93407 0.93848 0.98496 0.66180 0.35957 0.71122 0.35875
0.96504 0.60832 0.36569 0.73499 0.25223 0.88296 0.06659 0.78113 0.40016 0.31768
```

```
0.47655 0.15008 0.20608 0.62633 0.62737 0.16400 0.44104 0.56993 0.13178 0.50499
0.44176 0.44385 0.75372 0.82178 0.60227 0.98944 0.33133 0.81067 0.40798 0.71608
0.69306 0.22144 0.47942 0.65697 0.50881 0.25223 0.82373 0.50148 0.65246 0.53275
0.92935 0.13455 0.19901 0.78844 0.14006 0.50600 0.41069 0.49703 0.47858 0.02210
0.91444 0.10784 0.54642 0.63091 0.14419 0.80457 0.51336 0.71451 0.12564 0.88051
## Sample 2
100                 :: N
0.84976 0.63094 0.46109 0.80538 0.62387 0.90670 0.09969 0.67992 0.70503 0.09560
0.69991 0.37616 0.42030 0.23665 0.28771 0.24935 0.94950 0.12008 0.66217 0.20900
0.97026 0.98368 0.80206 0.43918 0.73232 0.03533 0.97995 0.06637 0.54726 0.48530
0.68865 0.94302 0.33718 0.61014 0.70127 0.36827 0.51335 0.24476 0.14203 0.02428
0.73691 0.22192 0.40374 0.85757 0.83335 0.73309 0.05563 0.17332 0.72253 0.43291
0.77476 0.35967 0.94242 0.61337 0.43513 0.80573 0.70630 0.83115 0.24622 0.45445
0.53595 0.31476 0.87968 0.75365 0.86291 0.34051 0.62232 0.16762 0.45506 0.15561
0.76615 0.77421 0.06035 0.72290 0.93712 0.83223 0.40044 0.96575 0.73176 0.27827
0.02174 0.75326 0.82876 0.64979 0.98038 0.61054 0.87742 0.95273 0.39091 0.42146
0.89020 0.08617 0.90953 0.00416 0.70915 0.21123 0.95342 0.19269 0.68252 0.27600
## Sample 3
100                 :: N
0.40629 0.96486 0.66026 0.07134 0.35492 0.34348 0.87164 0.59746 0.43724 0.26730
0.11840 0.04604 0.49037 0.99669 0.32784 0.34772 0.93599 0.95806 0.80635 0.18897
0.60061 0.83359 0.63026 0.14084 0.05323 0.70247 0.28532 0.09572 0.36153 0.50378
0.42679 0.71801 0.51010 0.72090 0.97537 0.29919 0.30059 0.23610 0.25668 0.07510
0.92481 0.65715 0.69686 0.27840 0.20555 0.64015 0.05725 0.25120 0.32288 0.22320
0.16582 0.71466 0.34030 0.55575 0.51468 0.18013 0.74670 0.21455 0.52649 0.47487
0.85805 0.24616 0.11459 0.38690 0.83475 0.83629 0.83754 0.18998 0.46715 0.24162
0.19488 0.03281 0.39291 0.37834 0.97169 0.65229 0.88913 0.53777 0.05780 0.20468
0.33788 0.10130 0.72771 0.31306 0.74279 0.26546 0.37941 0.04878 0.03061 0.52394
0.74104 0.97192 0.04550 0.81382 0.44430 0.32402 0.06791 0.73602 0.22640 0.67260
## Sample 4
100                 :: N
0.46016 0.95901 0.37581 0.45836 0.26220 0.30389 0.46845 0.52940 0.71121 0.89187
0.33346 0.81783 0.07194 0.01163 0.63324 0.69208 0.28685 0.02491 0.97931 0.53225
0.47009 0.12105 0.80291 0.21191 0.74158 0.78269 0.30493 0.06901 0.54152 0.88463
0.60358 0.81066 0.77771 0.74140 0.65465 0.32613 0.42757 0.36584 0.42506 0.39980
0.04686 0.79805 0.53593 0.15562 0.09924 0.68011 0.61072 0.88701 0.56239 0.64343
0.19223 0.07325 0.40971 0.85265 0.27507 0.88884 0.10551 0.62646 0.11055 0.91368
0.58845 0.68942 0.29994 0.30395 0.45696 0.88127 0.38773 0.12028 0.48981 0.28535
0.84174 0.46451 0.17140 0.90827 0.49424 0.29557 0.25788 0.76838 0.19073 0.26051
0.47442 0.03224 0.32034 0.97378 0.43992 0.13338 0.45850 0.02122 0.30482 0.49427
0.89839 0.01770 0.85679 0.90157 0.29537 0.15213 0.21464 0.37237 0.86199 0.60364
## Sample 5
100                 :: N
0.66793 0.00711 0.17970 0.98702 0.50449 0.88105 0.08259 0.77263 0.06050 0.73389
0.86517 0.76088 0.40239 0.50178 0.13811 0.63441 0.91949 0.48518 0.96923 0.08820
0.14556 0.28177 0.99598 0.46908 0.83279 0.26252 0.64987 0.20426 0.41060 0.76120
0.78022 0.44662 0.04918 0.36644 0.62337 0.16849 0.63846 0.41247 0.54464 0.05721
0.79852 0.23048 0.76139 0.22493 0.45640 0.07671 0.96152 0.50771 0.02376 0.49537
0.07095 0.86385 0.71385 0.35192 0.68827 0.49737 0.44847 0.26744 0.46983 0.44270
0.78845 0.72560 0.38886 0.45552 0.45917 0.64241 0.44654 0.42665 0.01122 0.76716
0.01727 0.33687 0.02836 0.48409 0.02777 0.63643 0.59711 0.02880 0.63758 0.56746
0.41342 0.40939 0.61578 0.89186 0.70151 0.38707 0.94021 0.17271 0.27477 0.04308
0.91821 0.97517 0.57249 0.14325 0.46058 0.26434 0.85927 0.77526 0.64717 0.08314
```

10.3 Program Results

```
 G08EAF Example Program Results

 Total number of runs found =        251

                           Count
          1        2        3        4        5       >5
         77      120       39       12        1        2

                           Expect
          1        2        3        4        5       >5
       83.8    104.0     45.6     13.1      2.9      0.6

 Covariance matrix
             1          2          3          4          5          6
```

```
 1     64.2222    -9.8639    -7.4780    -3.5759    -1.1406    -0.3305
 2     -9.8639    70.2942   -24.4639    -9.8092    -2.7386    -0.7103
 3     -7.4780   -24.4639    29.9473    -5.8284    -1.5474    -0.3852
 4     -3.5759    -9.8092    -5.8284    11.0343    -0.5319    -0.1289
 5     -1.1406    -2.7386    -1.5474    -0.5319     2.7169    -0.0318
 6     -0.3305    -0.7103    -0.3852    -0.1289    -0.0318     0.5809

 Chisq =     9.7559
 DF    =     6.0
 Prob  =     0.1353
```