NAG C Library Function Document

nag_chi_sq_goodness_of_fit_test (g08cgc)

1 Purpose

nag_chi_sq_goodness_of_fit_test (g08cgc) computes the test statistic for the χ^2 goodness of fit test for data with a chosen number of class intervals.

2 Specification

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

void nag_chi_sq_goodness_of_fit_test (Integer nclass, const Integer ifreq[],
      const double cint[], Nag_Distributions dist, const double par[], Integer npest,
      const double prob[], double *chisq, double *p, Integer *ndf, double eval[],
      double chisqi[], NagError *fail)
```

3 Description

The χ^2 goodness of fit test performed by nag_chi_sq_goodness_of_fit_test (g08cgc) is used to test the null hypothesis that a random sample arises from a specified distribution against the alternative hypothesis that the sample does not arise from the specified distribution.

Given a sample of size n, denoted by $x_1, x_2, \ldots, x_n$, drawn from a random variable X, and that the data have been grouped into k classes,

$$\begin{array}{l} x \leq c_1, \\ c_{i-1} < x \leq c_i, \quad i = 2, 3, \ldots, k-1, \\ x > c_{k-1}, \end{array}$$

then the χ^2 goodness of fit test statistic is defined by:

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

where O_i is the observed frequency of the ith class, and E_i is the expected frequency of the ith class.

The expected frequencies are computed as

$$E_i = p_i \times n,$$

where p_i is the probability that X lies in the ith class, that is

$$\begin{array}{l} p_1 = P(X \leq c_1), \\ p_i = P(c_{i-1} < X \leq c_i), \quad i = 2, 3, \ldots, k-1, \\ p_k = P(X > c_{k-1}). \end{array}$$

These probabilities are either taken from a common probability distribution or are supplied by the user. The available probability distributions within this function are:

Normal distribution with mean μ, variance σ^2;

uniform distribution on the interval $[a, b]$;

exponential distribution with probability density function $pdf = \lambda e^{-\lambda x}$;

χ^2 distribution with f degrees of freedom; and

gamma distribution with $pdf = \frac{x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)\beta^\alpha}$.

The user must supply the frequencies and classes. Given a set of data and classes the frequencies may be calculated using nag_frequency_table (g01aec).

nag_chi_sq_goodness_of_fit_test (g08cgc) returns the χ^2 test statistic, X^2, together with its degrees of freedom and the upper tail probability from the χ^2 distribution associated with the test statistic. Note that the use of the χ^2 distribution as an approximation to the distribution of the test statistic improves as the expected values in each class increase.

4 References

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

Kendall M G and Stuart A (1973) *The Advanced Theory of Statistics (Volume 2)* (3rd Edition) Griffin

Siegel S (1956) *Non-parametric Statistics for the Behavioral Sciences* McGraw–Hill

5 Arguments

1: **nclass** – Integer *Input*

On entry: the number of classes, k, into which the data is divided.

Constraint: **nclass** ≥ 2.

2: **ifreq**[**nclass**] – const Integer *Input*

On entry: **ifreq**$[i-1]$ must specify the frequency of the ith class, O_i, for $i = 1, 2, \ldots, k$.

Constraint: **ifreq**$[i-1] \geq 0$, for $i = 1, 2, \ldots, k$.

3: **cint**[**nclass** − **1**] – const double *Input*

On entry: **cint**$[i-1]$ must specify the upper boundary value for the ith class, for $i = 1, 2, \ldots, k-1$.

Constraints:

cint$[0] <$ **cint**$[1] < \cdots <$ **cint**[**nclass** $- 2$];
For the exponential, gamma and χ^2 distributions **cint**$[0] \geq 0.0$.

4: **dist** – Nag_Distributions *Input*

On entry: indicates for which distribution the test is to be carried out.

dist = **Nag_Normal**

The Normal distribution is used.

dist = **Nag_Uniform**

The uniform distribution is used.

dist = **Nag_Exponential**

The exponential distribution is used.

dist = **Nag_ChiSquare**

The χ^2 distribution is used.

dist = **Nag_Gamma**

The gamma distribution is used.

dist = **Nag_UserProb**

The user must supply the class probabilities in the array **prob**.

Constraint: **dist** = **Nag_Normal**, **Nag_Uniform**, **Nag_Exponential**, **Nag_ChiSquare**, **Nag_Gamma** or **Nag_UserProb**.

5: **par**[**2**] – const double *Input*

On entry: **par** must contain the parameters of the distribution which is being tested. If the user supplies the probabilities (i.e., **dist** = **Nag_UserProb**) the array **par** is not referenced.

If a Normal distribution is used then **par**[0] and **par**[1] must contain the mean, μ, and the variance, σ^2, respectively.

If a uniform distribution is used then **par**[0] and **par**[1] must contain the boundaries a and b respectively.

If an exponential distribution is used then **par**[0] must contain the parameter λ. **par**[1] is not used.

If a χ^2 distribution is used then **par**[0] must contain the number of degrees of freedom. **par**[1] is not used.

If a gamma distribution is used **par**[0] and **par**[1] must contain the parameters α and β respectively.

Constraints:

if **dist** = **Nag_Normal**, **par**[1] > 0.0;
if **dist** = **Nag_Uniform**, **par**[0] $<$ **par**[1] and **par**[0] $\leq$ **cint**[0];
par[1] $\geq$ **cint**(**nclass** $- 2$) otherwise;
if **dist** = **Nag_Exponential**, **par**[0] > 0.0;
if **dist** = **Nag_ChiSquare**, **par**[0] > 0.0;
if **dist** = **Nag_Gamma**, **par**[0] and **par**[1] > 0.0.

6: **npest** – Integer *Input*

On entry: the number of estimated parameters of the distribution.

Constraint: $0 \leq$ **npest** $<$ **nclass** $- 1$.

7: **prob**[**nclass**] – const double *Input*

On entry: if the user is supplying the probability distribution (i.e., **dist** = **Nag_UserProb**) then **prob**[$i - 1$] must contain the probability that X lies in the ith class.

If **dist** $\neq$ **Nag_UserProb**, **prob** is not referenced.

Constraint: if **dist** = **Nag_UserProb**, **prob**[$i - 1$] > 0.0 and $\sum_{i=1}^{k}$**prob**[$i - 1$] $= 1.0$, for $i = 1, 2, \ldots, k$.

8: **chisq** – double * *Output*

On exit: the test statistic, X^2, for the χ^2 goodness of fit test.

9: **p** – double * *Output*

On exit: the upper tail probability from the χ^2 distribution associated with the test statistic, X^2, and the number of degrees of freedom.

10: **ndf** – Integer * *Output*

On exit: contains (**nclass** $- 1 -$ **npest**), the degrees of freedom associated with the test.

11: **eval**[**nclass**] – double *Output*

On exit: **eval**[$i - 1$] contains the expected frequency for the ith class, E_i, for $i = 1, 2, \ldots, k$.

12: **chisqi**[**nclass**] – double *Output*

On exit: **chisqi**[$i - 1$] contains the contribution from the ith class to the test statistic, that is $(O_i - E_i)^2/E_i$, for $i = 1, 2, \ldots, k$.

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

The NAG error parameter, see the Essential Introduction.

6 Error Indicators and Warnings

NE_ARRAY_CONS

The contents of array **prob** are not valid.
Constraint: Sum of **prob**$[i-1] = 1$, for $i = 1, 2, \ldots,$ **nclass**, when **dist** = **Nag_UserProb**.

NE_ARRAY_INPUT

On entry, the values provided in **par** are invalid.

NE_BAD_PARAM

On entry, parameter **dist** had an illegal value.

NE_G08CG_CLASS_VAL

This is a warning that expected values for certain classes are less than 1.0. This implies that one cannot be confident that the χ^2 distribution is a good approximation to the distribution of the test statistic.

NE_G08CG_CONV

The solution obtained when calculating the probability for a certain class for the gamma or χ^2 distribution did not converge in 600 iterations. The solution may be an adequate approximation.

NE_G08CG_FREQ

An expected frequency is equal to zero when the observed frequency is not.

NE_INT_2

On entry, **npest** $= \langle value \rangle$, **nclass** $= \langle value \rangle$.
Constraint: $0 \leq$ **npest** $<$ **nclass** $- 1$.

NE_INT_ARG_LT

On entry, **nclass** must not be less than 2: **nclass** $= \langle value \rangle$.

NE_INT_ARRAY_CONS

On entry, **ifreq**$[\langle value \rangle] = \langle value \rangle$.
Constraint: **ifreq**$[i-1] \geq 0$, for $i = 1, 2, \ldots,$ **nclass**.

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.

NE_NOT_STRICTLY_INCREASING

The sequence **cint** is not strictly increasing **cint**$[\langle value \rangle] = \langle value \rangle$, **cint**$[\langle value \rangle - 1] = \langle value \rangle$.

NE_REAL_ARRAY_CONS

On entry, **prob**$[\langle value \rangle] = \langle value \rangle$.
Constraint: **prob**$[i-1] > 0$, for $i = 1, 2, \ldots,$ **nclass**, when **dist** = **Nag_UserProb**.

NE_REAL_ARRAY_ELEM_CONS

On entry, **cint**[0] = ⟨*value*⟩.
Constraint: **cint**[0] ≥ 0.0, if **dist** = **NagExponential**||**NagChiSquare**||**NagGamma**.

7 Accuracy

The computations are believed to be stable.

8 Further Comments

The time taken by nag_chi_sq_goodness_of_fit_test (g08cgc) is dependent both on the distribution chosen and on the number of classes, k.

9 Example

The example program applies the χ^2 goodness of fit test to test whether there is evidence to suggest that a sample of 100 observations generated by nag_random_continuous_uniform_ab (g05dac) do not arise from a uniform distribution $U(0,1)$. The class intervals are calculated such that the interval (0,1) is divided into five equal classes. The frequencies for each class are calculated using nag_frequency_table (g01aec).

9.1 Program Text

```
/* nag_chi_sq_goodness_of_fit_test (g08cgc) 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 <nagg01.h>
#include <nagg05.h>
#include <nagg08.h>

int main (void)
{
  Integer exit_status=0, i, iclass, *ifreq=0, init, igen = 0;
  Integer iseed[] = {0, 0, 0, 0}, n, nclass, ndf, npest;
  NagError fail;
  Nag_ClassBoundary class_enum;
  Nag_Distributions cdist_enum;
  char cdist[2];
  double chisq, *chisqi=0, *cint=0, *eval=0, p, *par=0, *prob=0, *x=0, xmax;
  double xmin;

  INIT_FAIL(fail);
  Vprintf("nag_chi_sq_goodness_of_fit_test (g08cgc) Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n]");

  Vscanf("%ld %ld  %s %*[^\n] ", &n, &nclass, cdist);
  if (*cdist ==  'U')
    cdist_enum = Nag_Uniform;
  else if (*cdist == 'N')
    cdist_enum =  Nag_Normal;
  else if (*cdist ==  'G')
    cdist_enum = Nag_Gamma;
  else if (*cdist ==  'C')
    cdist_enum = Nag_ChiSquare;
```

```
  else if (*cdist ==  'E')
    cdist_enum =  Nag_Exponential;
  else if (*cdist ==  'A')
    cdist_enum = Nag_UserProb;
  else
    cdist_enum =  (Nag_Distributions)-999;
  if (!(x = NAG_ALLOC(n, double))
      || !(cint = NAG_ALLOC(nclass-1, double)  )
      || !(par = NAG_ALLOC(2, double))
      || !(ifreq = NAG_ALLOC(nclass, Integer)))
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (i = 1; i <= 2; ++i)
    Vscanf("%lf", &par[i - 1]);
  npest = 0;
  /* Generate random numbers from a uniform distribution */
  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);
  /* nag_rngs_uniform (g05lgc).
   * Generates a vector of random numbers from a uniform
   * distribution, seeds and generator number passed
   * explicitly
   */
  nag_rngs_uniform(par[0], par[1], n, x, igen, iseed, NAGERR_DEFAULT);
  iclass = 0;
  /* Determine suitable intervals */
  if (cdist_enum == Nag_Uniform)
    {
      iclass = 1;
      cint[0] = par[0] + (par[1] - par[0]) / nclass;
      for (i = 2; i <= nclass - 1; ++i)
        cint[i - 1] = cint[i - 2] + (par[1] - par[0]) / nclass;
    }
  if (iclass == 1)
    class_enum = Nag_ClassBoundaryUser;
  else
    class_enum = Nag_ClassBoundaryComp;


  /* nag_frequency_table (g01aec).
   * Frequency table from raw data
   */
  nag_frequency_table(n, x, nclass, class_enum, cint, ifreq, &xmin, &xmax,
                      &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from nag_frequency_table (g01aec).\n%s\n", fail.message);
      return 1;
    }

  if (!(chisqi = NAG_ALLOC(nclass, double))
      || !(eval = NAG_ALLOC(nclass, double))
      || !(prob = NAG_ALLOC(nclass, double)))
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  /* nag_chi_sq_goodness_of_fit_test (g08cgc).
   * Performs the chi^2 goodness of fit test, for standard
   * continuous distributions
   */
  nag_chi_sq_goodness_of_fit_test(nclass, ifreq, cint, cdist_enum, par, npest,
```

```
                                  prob, &chisq, &p, &ndf, eval, chisqi, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from nag_chi_sq_goodness_of_fit_test (g08cgc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
  Vprintf("\n");
  Vprintf("%s%10.4f\n", "Chi-squared test statistic   = ", chisq);
  Vprintf("%s%5ld\n", "Degrees of freedom.          = ", ndf);
  Vprintf("%s%10.4f\n", "Significance level           = ", p);
  Vprintf("\n");
  Vprintf("%s\n", "The contributions to the test statistic are :-");
  for (i = 1; i <= nclass; ++i)
    Vprintf("%10.4f\n", chisqi[i - 1]);
 END:
  if (x) NAG_FREE(x);
  if (cint) NAG_FREE(cint);
  if (par) NAG_FREE(par);
  if (ifreq) NAG_FREE(ifreq);
  if (chisqi) NAG_FREE(chisqi);
  if (eval) NAG_FREE(eval);
  if (prob) NAG_FREE(prob);
  return exit_status;
}
```

9.2 Program Data

```
nag_chi_sq_goodness_of_fit_test (g08cgc) Example Program Data.
100 5  U      :n  nclass cdist
0.0 1.0        :par[0] par[2]
```

9.3 Program Results

```
nag_chi_sq_goodness_of_fit_test (g08cgc) Example Program Results

Chi-squared test statistic   =     3.3000
Degrees of freedom.          =     4
Significance level           =     0.5089

The contributions to the test statistic are :-
    1.8000
    0.8000
    0.2000
    0.0500
    0.4500
```