

NAG 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, \dots, x_n , drawn from a random variable X , and that the data have been grouped into k classes,

$$\begin{aligned} x &\leq c_1, \\ c_{i-1} &< x \leq c_i, \quad i = 2, 3, \dots, k-1, \\ x &> c_{k-1}, \end{aligned}$$

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 i th class, and E_i is the expected frequency of the i th class.

The expected frequencies are computed as

$$E_i = p_i \times n,$$

where p_i is the probability that X lies in the i th class, that is

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

These probabilities are either taken from a common probability distribution or are supplied by you. 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 $pdf = \frac{x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)\beta^\alpha}$.

You 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 i th class, O_i , for $i = 1, 2, \dots, k$.
Constraint: **ifreq** $[i - 1] \geq 0$, for $i = 1, 2, \dots, k$.
- 3: **cint[nclass - 1]** – const double *Input*
On entry: **cint** $[i - 1]$ must specify the upper boundary value for the i th class, for $i = 1, 2, \dots, k - 1$.
Constraints:
cint $[0] < \mathbf{cint}[1] < \dots < \mathbf{cint}[\mathbf{nclass} - 2]$;
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
You 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 arguments of the distribution which is being tested. If you supply 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 argument 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 arguments α and β respectively.
Constraints:
 if **dist** = Nag_Normal, **par[1]** > 0.0;
 if **dist** = Nag_Uniform, **par[0]** < **par[1]** and **par[0]** ≤ **cint[0]**;
 otherwise **par[1]** ≥ **cint**(**nclass** – 2);
 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 arguments of the distribution.
Constraint: $0 \leq \mathbf{npest} < \mathbf{nclass} - 1$.
- 7: **prob[nclass]** – const double *Input*
On entry: if you are supplying the probability distribution (i.e., **dist** = Nag_UserProb) then **prob[i – 1]** must contain the probability that X lies in the i th class.
 If **dist** ≠ Nag_UserProb, **prob** is not referenced.
Constraint: if **dist** = Nag_UserProb, **prob[i – 1]** > 0.0 and $\sum_{i=1}^k \mathbf{prob}[i - 1] = 1.0$, for $i = 1, 2, \dots, k$.
- 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 i th class, E_i , for $i = 1, 2, \dots, k$.
- 12: **chisqi[nclass]** – double *Output*
On exit: **chisqi[i – 1]** contains the contribution from the i th class to the test statistic, that is $(O_i - E_i)^2 / E_i$, for $i = 1, 2, \dots, k$.

13: **fail** – NagError *

Input/Output

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ARRAY_CONS

The contents of array **prob** are not valid.

Constraint: Sum of **prob** $[i - 1] = 1$, for $i = 1, 2, \dots, \mathbf{nclass}$, when **dist** = Nag_UserProb.

NE_ARRAY_INPUT

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

NE_BAD_PARAM

On entry, argument **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: $0 \leq \mathbf{npest} < \mathbf{nclass} - 1$.

NE_INT_ARG_LT

On entry, **nclass** = $\langle value \rangle$.

Constraint: **nclass** ≥ 2 .

NE_INT_ARRAY_CONS

On entry, **ifreq** $[\langle value \rangle] = \langle value \rangle$.

Constraint: **ifreq** $[i - 1] \geq 0$, for $i = 1, 2, \dots, \mathbf{nclass}$.

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 contact 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 $i = 1, 2, \dots, \mathbf{nclass}$, when **dist** = Nag_UserProb.

NE_REAL_ARRAY_ELEM_CONS

On entry, **cint**[0] = $\langle value \rangle$.

Constraint: **cint**[0] ≥ 0.0 , if **dist** = Nag_Exponential||Nag_ChiSquare||Nag_Gamma.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

nag_chi_sq_goodness_of_fit_test (g08cgc) is not threaded in any implementation.

9 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 .

10 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_rand_uniform (g05sqc) 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).

10.1 Program Text

```
/* nag_chi_sq_goodness_of_fit_test (g08cgc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>
#include <nagg05.h>
#include <nagg08.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0, i, n, nclass, ndf, npest, lstate;
    Integer *ifreq = 0, *state = 0;

    /* NAG structures */
    Nag_ClassBoundary classb;
    Nag_Distributions cdist;
    NagError fail;

    /* Double scalar and array declarations */
    double chisq, *chisqi = 0, *cint = 0, *eval = 0, p, *par = 0;
    double *prob = 0, *x = 0, xmax, xmin;

    /* Character array declarations */
    char nag_enum_arg[40];

    /* Choose the base generator */
    Nag_BaseRNG genid = Nag_Basic;
```

```

Integer subid = 0;

/* Set the seed */
Integer seed[] = { 1762543 };
Integer lseed = 1;

INIT_FAIL(fail);

printf("nag_chi_sq_goodness_of_fit_test (g08cgc) 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%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " %" NAG_IFMT " %39s %*[\n]", &n, &nclass,
        nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%" NAG_IFMT " %" NAG_IFMT " %39s %*[\n] ", &n, &nclass,
        nag_enum_arg);
#endif

/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
cdist = (Nag_Distributions) nag_enum_name_to_value(nag_enum_arg);

if (!(x = NAG_ALLOC(n, double))
    || !(state = NAG_ALLOC(lstate, Integer))
    || !(cint = NAG_ALLOC(nclass - 1, double))
    || !(par = NAG_ALLOC(2, double))
    || !(ifreq = NAG_ALLOC(nclass, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

for (i = 1; i <= 2; ++i)
#ifdef _WIN32
    scanf_s("%lf", &par[i - 1]);
#else
    scanf("%lf", &par[i - 1]);
#endif
npest = 0;

/* Initialize the generator to a repeatable sequence */
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%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Generate random numbers from a uniform distribution */
nag_rand_uniform(n, par[0], par[1], state, x, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_uniform (g05sqc).\n%s\n", fail.message);
}

```

```

    return 1;
}

classb = Nag_ClassBoundaryComp;
/* Determine suitable intervals */
if (cdist == Nag_Uniform) {
    classb = Nag_ClassBoundaryUser;
    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;
}

/* nag_frequency_table (g01aec).
 * Frequency table from raw data
 */
nag_frequency_table(n, x, nclass, classb, cint, ifreq, &xmin, &xmax, &fail);
if (fail.code != NE_NOERROR) {
    printf("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)))
{
    printf("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, par, npest,
                                prob, &chisq, &p, &ndf, eval, chisqi,
                                &fail);

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

```

10.2 Program Data

```

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

```

10.3 Program Results

nag_chi_sq_goodness_of_fit_test (g08cgc) Example Program Results

```
Chi-squared test statistic   =    4.0000
Degrees of freedom.         =     4
Significance level           =    0.4060
```

The contributions to the test statistic are :-

```
1.8000
1.2500
0.4500
0.0500
0.4500
```
