1 Purpose

\texttt{nag\_chi\_sq\_goodness\_of\_fit\_test (g08cgc)} computes the test statistic for the \( \chi^2 \) goodness-of-fit test for data with a chosen number of class intervals.

2 Specification

\begin{verbatim}
#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)
\end{verbatim}

3 Description

The \( \chi^2 \) goodness-of-fit test performed by \texttt{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,

\[
x \leq c_1,
\]
\[
c_{i-1} < x \leq c_i, \quad i = 2, 3, \ldots, k - 1,
\]
\[
x > c_{k-1},
\]

then the \( \chi^2 \) 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

\[
p_1 = P(X \leq c_1),
\]
\[
p_i = P(c_{i-1} < X \leq c_i), \quad i = 2, 3, \ldots, k - 1,
\]
\[
p_k = P(X > c_{k-1}).
\]

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 \( \mu \), variance \( \sigma^2 \);
- uniform distribution on the interval \([a, b] \);
- exponential distribution with probability density function \( pdf = \lambda e^{-\lambda x} \);
- \( \chi^2 \) distribution with \( f \) degrees of freedom; and
- gamma distribution with \( pdf = \frac{x^{\alpha-1}e^{-x/\beta}}{\Gamma(\alpha)} \).
You must supply the frequencies and classes. Given a set of data and classes the frequencies may be calculated using \texttt{nag\_frequency\_table (g01aec)}.

\texttt{\texttt{nag\_chi\_sq\_goodness\_of\_fit\_test (g08cgc)}} returns the $\chi^2$ test statistic, $X^2$, together with its degrees of freedom and the upper tail probability from the $\chi^2$ distribution associated with the test statistic. Note that the use of the $\chi^2$ 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) \textit{Practical Nonparametric Statistics} Wiley


5 Arguments

1: \texttt{nclass} – Integer 
\hspace{1cm} \textit{Input}
\hspace{1cm} \textit{On entry}: the number of classes, $k$, into which the data is divided.
\hspace{1cm} \textit{Constraint}: $\texttt{nclass} \geq 2$.

2: \texttt{ifreq[nclass]} – const Integer 
\hspace{1cm} \textit{Input}
\hspace{1cm} \textit{On entry}: \texttt{ifreq[i-1]} must specify the frequency of the $i$th class, $O_i$, for $i = 1, 2, \ldots, k$.
\hspace{1cm} \textit{Constraint}: $\texttt{ifreq[i-1]} \geq 0$, for $i = 1, 2, \ldots, k$.

3: \texttt{cint[nclass-1]} – const double 
\hspace{1cm} \textit{Input}
\hspace{1cm} \textit{On entry}: \texttt{cint[i-1]} must specify the upper boundary value for the $i$th class, for $i = 1, 2, \ldots, k-1$.
\hspace{1cm} \textit{Constraints}:
\hspace{2cm} $\texttt{cint[0]} < \texttt{cint[1]} < \cdots < \texttt{cint[nclass-2]}$;
\hspace{2cm} For the exponential, gamma and $\chi^2$ distributions $\texttt{cint[0]} \geq 0.0$.

4: \texttt{dist} – \texttt{Nag\_Distributions} 
\hspace{1cm} \textit{Input}
\hspace{1cm} \textit{On entry}: indicates for which distribution the test is to be carried out.
\hspace{1cm} \texttt{dist} = \texttt{Nag\_Normal} 
\hspace{1cm} The Normal distribution is used.
\hspace{1cm} \texttt{dist} = \texttt{Nag\_Uniform} 
\hspace{1cm} The uniform distribution is used.
\hspace{1cm} \texttt{dist} = \texttt{Nag\_Exponential} 
\hspace{1cm} The exponential distribution is used.
\hspace{1cm} \texttt{dist} = \texttt{Nag\_ChiSquare} 
\hspace{1cm} The $\chi^2$ distribution is used.
\hspace{1cm} \texttt{dist} = \texttt{Nag\_Gamma} 
\hspace{1cm} The gamma distribution is used.
\hspace{1cm} \texttt{dist} = \texttt{Nag\_UserProb} 
\hspace{1cm} You must supply the class probabilities in the array \texttt{prob}.
\hspace{1cm} \textit{Constraint}: \texttt{dist} = \texttt{Nag\_Normal}, \texttt{Nag\_Uniform}, \texttt{Nag\_Exponential}, \texttt{Nag\_ChiSquare}, \texttt{Nag\_Gamma} or \texttt{Nag\_UserProb}.
5:  \texttt{par[2]} – const double \hspace{1cm} \textit{Input}

\textit{On entry:} \texttt{par} must contain the arguments of the distribution which is being tested. If you supply the probabilities (i.e., \texttt{dist} = \texttt{Nag_UserProb}) the array \texttt{par} is not referenced.

If a Normal distribution is used then \texttt{par[0]} and \texttt{par[1]} must contain the mean, \(\mu\), and the variance, \(\sigma^2\), respectively.

If a uniform distribution is used then \texttt{par[0]} and \texttt{par[1]} must contain the boundaries \(a\) and \(b\) respectively.

If an exponential distribution is used then \texttt{par[0]} must contain the argument \(\lambda\). \texttt{par[1]} is not used.

If a \(\chi^2\) distribution is used then \texttt{par[0]} must contain the number of degrees of freedom. \texttt{par[1]} is not used.

If a gamma distribution is used \texttt{par[0]} and \texttt{par[1]} must contain the arguments \(\alpha\) and \(\beta\) respectively.

\textit{Constraints:}
- if \texttt{dist} = \texttt{Nag_Normal}, \texttt{par[1]} > 0.0;
- if \texttt{dist} = \texttt{Nag_Uniform}, \texttt{par[0]} < \texttt{par[1]} and \texttt{par[0]} \leq \texttt{cint[0]};
- otherwise \texttt{par[1]} \geq \texttt{cint(nclass - 2)};
- if \texttt{dist} = \texttt{Nag_Exponential}, \texttt{par[0]} > 0.0;
- if \texttt{dist} = \texttt{Nag_ChiSquare}, \texttt{par[0]} > 0.0;
- if \texttt{dist} = \texttt{Nag_Gamma}, \texttt{par[0]} and \texttt{par[1]} > 0.0.

6:  \texttt{npest} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the number of estimated arguments of the distribution.

\textit{Constraint:} 0 \leq \texttt{npest} < \texttt{nclass} - 1.

7:  \texttt{prob[nclass]} – const double \hspace{1cm} \textit{Input}

\textit{On entry:} if you are supplying the probability distribution (i.e., \texttt{dist} = \texttt{Nag_UserProb}) then \texttt{prob[i - 1]} must contain the probability that \(X\) lies in the \(i\)th class.

If \texttt{dist} \neq \texttt{Nag_UserProb}, \texttt{prob} is not referenced.

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

8:  \texttt{chisq} – double * \hspace{1cm} \textit{Output}

\textit{On exit:} the test statistic, \(X^2\), for the \(\chi^2\) goodness-of-fit test.

9:  \texttt{p} – double * \hspace{1cm} \textit{Output}

\textit{On exit:} the upper tail probability from the \(\chi^2\) distribution associated with the test statistic, \(X^2\), and the number of degrees of freedom.

10:  \texttt{ndf} – Integer * \hspace{1cm} \textit{Output}

\textit{On exit:} contains (\texttt{nclass} - 1 - \texttt{npest}), the degrees of freedom associated with the test.

11:  \texttt{eval[nclass]} – double \hspace{1cm} \textit{Output}

\textit{On exit:} \texttt{eval[i - 1]} contains the expected frequency for the \(i\)th class, \(E_i\), for \(i = 1, 2, \ldots, k\).

12:  \texttt{chisqi[nclass]} – double \hspace{1cm} \textit{Output}

\textit{On exit:} \texttt{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 \(i = 1, 2, \ldots, k\).
13: \texttt{fail} – NagError*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

\textbf{NE\_ARRAY\_CONS}

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

\textbf{NE\_ARRAY\_INPUT}

On entry, the values provided in \texttt{par} are invalid.

\textbf{NE\_BAD\_PARAM}

On entry, argument \texttt{dist} had an illegal value.

\textbf{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 $\chi^2$ distribution is a good approximation to the distribution of the test statistic.

\textbf{NE\_G08CG\_CONV}

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

\textbf{NE\_G08CG\_FREQ}

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

\textbf{NE\_INT\_2}

On entry, $\texttt{npest} = \langle\text{value}\rangle$, $\texttt{nclass} = \langle\text{value}\rangle$.
Constraint: $0 \leq \texttt{npest} < \texttt{nclass} - 1$.

\textbf{NE\_INT\_ARG\_LT}

On entry, $\texttt{nclass} = \langle\text{value}\rangle$.
Constraint: $\texttt{nclass} \geq 2$.

\textbf{NE\_INT\_ARRAY\_CONS}

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

\textbf{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.

\textbf{NE\_NOT\_STRICTLY\_INCREASING}

The sequence \texttt{cint} is not strictly increasing $\texttt{cint}[\langle\text{value}\rangle] = \langle\text{value}\rangle$, $\texttt{cint}[\langle\text{value}\rangle - 1] = \langle\text{value}\rangle$.

\textbf{NE\_REAL\_ARRAY\_CONS}

On entry, $\texttt{prob}[\langle\text{value}\rangle] = \langle\text{value}\rangle$.
Constraint: $\texttt{prob}[i - 1] > 0$, for $i = 1, 2, \ldots, \texttt{nclass}$, when \texttt{dist} = Nag_UserProb.
NE_REAL_ARRAY_ELEM_CONS
On entry, cint[0] = <value>.
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
Not applicable.

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 \( \chi^2 \) 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.
 * Copyright 2014 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 scalar and array declarations */
    Integer exit_status = 0, i, n, nclass, ndf, npest, lstate;
    Integer *ifreq = 0, *state = 0;

    /* NAG structures */
    Nag_ClassBoundary class;
    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\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("%"NAGIFORM "%"NAGIFORM %39s %*[\n] ",  
   &n, &nclass, 
   nag_enum_arg, _countof(nag_enum_arg));
#else
   scanf("%"NAGIFORM "%"NAGIFORM %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;

/* Initialise 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\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\n%s\n", fail.message);
        return 1;
    }

class = Nag_ClassBoundaryComp;
/* Determine suitable intervals */
if (cdist == Nag_Uniform)
    {
        class = 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 */
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 */
if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_chi_sq_goodness_of_fit_test (g08cgc).\n\n%s\n", fail.message);
        return 1;
    }

for (i = 1; i <= nclass; ++i)
    printf("%.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;

Mark 25

Mark 25
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