NAG Library Function Document

nag_pairs_test (g08ebc)

1 Purpose

nag_pairs_test (g08ebc) performs a pairs test on a sequence of observations in the interval $[0, 1]$.

2 Specification

```
#include <nag.h>
#include <nagg08.h>
void nag_pairs_test (Integer n, const double x[], Integer max_count,
     Integer lag, double *chi, double *df, double *prob, NagError *fail)
```

3 Description

nag_pairs_test (g08ebc) computes the statistics for performing a pairs test which may be used to investigate deviations from randomness in a sequence, $x = \{x_i : i = 1, 2, \ldots, n\}$, of $[0, 1]$ observations.

For a given lag, $l \geq 1$, an m by m matrix, C, of counts is formed as follows. The element c_{jk} of C is the number of pairs (x_i, x_{i+l}) such that

$$\frac{j-1}{m} \leq x_i < \frac{j}{m}$$

$$\frac{k-1}{m} \leq x_{i+l} < \frac{k}{m}$$

where $i = 1, 3, 5, \ldots, n-1$ if $l = 1$, and $i = 1, 2, \ldots, l, 2l+1, 2l+2, \ldots 3l, 4l+1, \ldots, n-l$, if $l > 1$.

Note that all pairs formed are non-overlapping pairs and are thus independent under the assumption of randomness.

Under the assumption that the sequence is random, the expected number of pairs for each class (i.e., each element of the matrix of counts) is the same; that is, the pairs should be uniformly distributed over the unit square $[0, 1]^2$. Thus the expected number of pairs for each class is just the total number of pairs, $\sum_{j,k=1}^{m} c_{jk}$, divided by the number of classes, m^2.

The χ^2 test statistic used to test the hypothesis of randomness is defined as

$$X^2 = \sum_{j,k=1}^{m} \frac{\left(c_{jk} - e\right)^2}{e},$$

where $e = \sum_{j,k=1}^{m} c_{jk}/m^2 =$ expected number of pairs in each class.

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.

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: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: $\mathbf{n} \geq 2$.

2: **x**[**n**] – const double *Input*

On entry: the sequence of observations.

Constraint: $0.0 \leq \mathbf{x}[i-1] \leq 1.0$, for $i = 1, 2, \ldots, n$.

3: **max_count** – Integer *Input*

On entry: m, the size of the matrix of counts.

Constraint: **max_count** ≥ 2.

4: **lag** – Integer *Input*

On entry: l, the lag to be used in choosing pairs.

If $\mathbf{lag} = 1$, then we consider the pairs $(\mathbf{x}[i-1], \mathbf{x}[i])$, for $i = 1, 3, \ldots, n-1$, where n is the number of observations.

If $\mathbf{lag} > 1$, then we consider the pairs $(\mathbf{x}[i-1], \mathbf{x}[i+l-1])$, for $i = 1, 2, \ldots, l, 2l+1, 2l+2, \ldots, 3l, 4l+1, \ldots, n-l$, where n is the number of observations.

Constraint: $1 \leq \mathbf{lag} < \mathbf{n}$.

5: **chi** – double * *Output*

On exit: contains the χ^2 test statistic, X^2, for testing the null hypothesis of randomness.

6: **df** – double * *Output*

On exit: contains the degrees of freedom for the χ^2 statistic.

7: **prob** – double * *Output*

On exit: contains the upper tail probability associated with the χ^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_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.

NE_G08EB_CELL

max_count is too large relative to the number of pairs, therefore the expected value for at least one cell is less than or equal to 5.0.
This implies that the χ^2 distribution may not be a very good approximation to the distribution of test statistic.
max_count = ⟨*value*⟩, number of pairs = ⟨*value*⟩ and expected value = ⟨*value*⟩.
All statistics are returned and may still be of use.

NE_G08EB_PAIRS

No pairs were found. This will occur if the value of **lag** is greater than or equal to the total number of observations.

NE_INT_2

On entry, **lag** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: $1 \leq \mathbf{lag} < \mathbf{n}$.

NE_INT_ARG_LE

On entry, **max_count** = ⟨*value*⟩.
Constraint: **max_count** ≥ 2.

NE_INT_ARG_LT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 2

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.

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

NE_NO_LICENCE

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.

NE_REAL_ARRAY_CONS

On entry, at least one element of **x** is out of range.
Constraint: $0 \leq \mathbf{x}[i-1] \leq 1$, for $i = 1, 2, \ldots, \mathbf{n}$.

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

Not applicable.

9 Further Comments

The time taken by the function increases with the number of observations n.

10 Example

The following program performs the pairs test on 10000 pseudorandom numbers taken from a uniform distribution $U(0,1)$, generated by nag_rand_basic (g05sac). nag_pairs_test (g08ebc) is called with **lag** $= 1$ and **max_count** $= 10$..

10.1 Program Text

```
/* nag_pairs_test (g08ebc) 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     lstate;
  Integer     *state = 0;

  /* NAG structures */
  NagError    fail;

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

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

  /* Set the seed */
  Integer     seed[] = { 438532 };
  Integer     lseed = 1;

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

  /* Set the size of the matrix of counts */
  Integer     max_count = 10;

  /* Set the lag used when choosing pairs */
  Integer     lag = 1;

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

  printf("nag_pairs_test (g08ebc) 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;
```

```
    }

  /* Allocate arrays */
  if (!(x = NAG_ALLOC(n, double)) ||
      !(state = NAG_ALLOC(lstate, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* 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%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Generate vector of n uniform variates between 0.0 and 1.0 */
  nag_rand_basic(n, state, x, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_basic (g05sac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_pairs_test (g08ebc).
   * Performs the pairs (serial) test for randomness
   */
  nag_pairs_test(n, x, max_count, lag, &chi, &df, &p, &fail);

  if (fail.code != NE_NOERROR && fail.code != NE_G08EB_CELL)
    {
      printf("Error from nag_pairs_test (g08ebc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Display the results */
  printf("\n");
  printf("\n");
  printf("%s%10.4f\n", "CHISQ          = ", chi);
  printf("%s%8.2f\n", "DF             = ", df);
  printf("%s%10.4f\n", "Probability    = ", p);
  if (fail.code == NE_G08EB_CELL)
    printf("Error from nag_pairs_test (g08ebc).\n%s\n", fail.message);

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

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_pairs_test (g08ebc) Example Program Results


CHISQ          =    96.7200
DF             =    99.00
Probability    =     0.5461
```