NAG Library Function Document

nag_anderson_darling_stat (g08chc)

1 Purpose

nag_anderson_darling_stat (g08chc) calculates the Anderson–Darling goodness-of-fit test statistic.

2 Specification

```
#include <nag.h>
#include <nagg08.h>
double nag_anderson_darling_stat (Integer n, Nag_Boolean issort, double y[],
       NagError *fail)
```

3 Description

Denote by A^2 the Anderson–Darling test statistic for n observations $y_1, y_2, \ldots, y_n$ of a variable Y assumed to be standard uniform and sorted in ascending order, then:

$$A^2 = -n - S;$$

where:

$$S = \sum_{i=1}^{n} \frac{2i-1}{n} [\ln y_i + \ln(1 - y_{n-i+1})].$$

When observations of a random variable X are non-uniformly distributed, the probability integral transformation (PIT):

$$Y = F(X),$$

where F is the cumulative distribution function of the distribution of interest, yields a uniformly distributed random variable Y. The PIT is true only if all parameters of a distribution are known as opposed to estimated; otherwise it is an approximation.

4 References

Anderson T W and Darling D A (1952) Asymptotic theory of certain ‘goodness-of-fit’ criteria based on stochastic processes *Annals of Mathematical Statistics* **23** 193–212

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of observations.

Constraint: **n** > 1.

2: **issort** – Nag_Boolean *Input*

On entry: set **issort** = Nag_TRUE if the observations are sorted in ascending order; otherwise the function will sort the observations.

3: **y**[**n**] – double *Input/Output*

On entry: y_i, for $i = 1, 2, \ldots, n$, the n observations.

On exit: if **issort** = Nag_FALSE, the data sorted in ascending order; otherwise the array is unchanged.

Constraint: if **issort** = Nag_TRUE, the values must be sorted in ascending order. Each y_i must lie in the interval $(0, 1)$.

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

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

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_BOUND

The data in **y** must lie in the interval $(0, 1)$.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** > 1.

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_INCREASING

issort = Nag_TRUE and the data in **y** is not sorted in ascending order.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example calculates the A^2 statistic for data assumed to arise from an exponential distribution with a sample parameter estimate and simulates its p-value using the NAG basic random number generator.

10.1 Program Text

```
/* nag_anderson_darling_stat (g08chc) Example Program.
 *
 * Mark 23 Release. NAG Copyright 2011.
 */
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
```

```
#include <nagg05.h>
#include <nagg08.h>

int main(void)
{
  /* Scalars */
  double        a2, aa2, beta, nupper, p, sa2, sbeta;
  const Integer lseed = 1, subid = -1;
  Integer       exit_status = 0, i, j, k, lstate = 17, n, nsim, n_pseudo;
  /* Arrays */
  double        *x = 0, *xsim = 0, *y = 0;
  Integer       seed[1], state[17];
  /* NAG types */
  Nag_Boolean   issort;
  NagError      fail;

  printf("%s\n\n",
         "nag_anderson_darling_stat (g08chc) Example Program Results");

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

  /* Read number of observations */
  scanf("%"NAG_IFMT "", &n);
  scanf("%*[^\n] ");

  /* Memory allocation */
  if (!(x = NAG_ALLOC(n, double)) ||
      !(y = NAG_ALLOC(n, double)))
  {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
  }

  /* Read observations */
  for (i = 0; i < n; i++)
  {
      scanf("%lf", x+i);
  }
  scanf("%*[^\n] ");

  /* Maximum likelihood estimate of mean */
  for (i = 0, beta = 0.0; i < n; i++)
  {
      beta += x[i];
  }
  beta /= (double)n;

  /* PIT, using exponential CDF with mean beta */
  for (i = 0; i < n; i++)
  {
      y[i]= 1.0 - exp(-x[i]/beta);
  }

  /* Let nag_anderson_darling_stat (g08chc) sort the (approximately)
     uniform variates */
  issort = Nag_FALSE;

  /* Calculate the Anderson-Darling goodness-of-fit test statistic */
  INIT_FAIL(fail);
  /* nag_anderson_darling_stat (g08chc) */
  a2 = nag_anderson_darling_stat(n, issort, y, &fail);

  /* Correction due to estimated mean */
  aa2 = (1.0 + 0.6/(double)n)*a2;

  /* Number of simulations; a suitably high number */
  nsim = 888;

  /* Generate exponential variates using a repeatable seed */
```

```
  n_pseudo = n*nsim;
  if (!(xsim = NAG_ALLOC(n_pseudo, double)))
  {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
  }

  INIT_FAIL(fail);

  /* Initialize NAG's Basic pseudorandom number generator to give a
     repeatable sequence */
  seed[0] = 206033;
  /* nag_rand_init_repeatable (g05kfc) */
  nag_rand_init_repeatable(Nag_Basic, subid, (const Integer*)seed,
                           lseed, state, &lstate, &fail);

  /* Generate a vector of pseudorandom numbers from an exponential
     distribution */
  /* nag_rand_exp (g05sfc) */
  nag_rand_exp(n_pseudo, beta, state, xsim, &fail);

  /* Simulations loop */
  for (j = 0, nupper = 0.0; j < nsim; j++)
  {
      /* Index in the simulated data */
      k = j*n;

      /* Maximum likelihood estimate of mean */
      for (i = 0, sbeta = 0.0; i < n; i++)
      {
          sbeta += xsim[k+i];
      }
      sbeta /= (double)n;

      /* PIT */
      for (i = 0; i < n; i++)
      {
          y[i] = 1.0 - exp(-xsim[k+i]/sbeta);
      }

      /* Calculate A-squared */
      /* nag_anderson_darling_stat (g08chc) */
      sa2 = nag_anderson_darling_stat(n, issort, y, &fail);

      if (sa2 > aa2)
      {
          nupper++;
      }
  }

  /* Simulated upper tail probability value */
  p = nupper/(double)(nsim+1);

  /* Results */
  printf("%s", " H0: data from exponential distribution with mean ");
  printf("%g\n", beta);
  printf("%s", " Test statistic, A-squared: ");
  printf("%6g\n", a2);
  printf("%s", " Upper tail probability:    ");
  printf("%6g\n", p);

  END:
  NAG_FREE(x);
  NAG_FREE(xsim);
  NAG_FREE(y);

  return exit_status;
}
```

10.2 Program Data

```
nag_anderson_darling_stat (g08chc) Example Program Data
26 :: n
0.4782745 1.2858962 1.1163891 2.0410619 2.2648109 0.0833660 1.2527554
0.4031288 0.7808981 0.1977674 3.2539440 1.8113504 1.2279834 3.9178773
1.4494309 0.1358438 1.8061778 6.0441929 0.9671624 3.2035042 0.8067364
0.4179364 3.5351774 0.3975414 0.6120960 0.1332589 :: end of observations
```

10.3 Program Results

```
nag_anderson_darling_stat (g08chc) Example Program Results

 H0: data from exponential distribution with mean 1.52402
 Test statistic, A-squared: 0.161632
 Upper tail probability:    0.979753
```