NAG Library Function Document

nag_anderson_darling_uniform_prob (g08cjc)

1 Purpose

nag_anderson_darling_uniform_prob (g08cjc) calculates the Anderson–Darling goodness-of-fit test statistic and its probability for the case of standard uniformly distributed data.

2 Specification

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

```
void nag_anderson_darling_uniform_prob (Integer n, Nag_Boolean issort,
     double y[], double *a2, double *p, NagError *fail)
```

3 Description

Calculates the Anderson–Darling test statistic A^2 (see nag_anderson_darling_stat (g08chc)) and its upper tail probability by using the approximation method of Marsaglia and Marsaglia (2004) for the case of uniformly distributed data.

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

Marsaglia G and Marsaglia J (2004) Evaluating the Anderson–Darling distribution *J. Statist. Software* **9(2)**

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: **a2** – double * *Output*

On exit: A^2, the Anderson–Darling test statistic.

5: **p** – double * *Output*

On exit: p, the upper tail probability for A^2.

6: **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_BOUND

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

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} > 1$.

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.

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_NOT_INCREASING

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

7 Accuracy

Probabilities greater than approximately 0.09 are accurate to five decimal places; lower value probabilities are accurate to six decimal places.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example calculates the A^2 statistic and its p-value for uniform data obtained by transforming exponential variates.

10.1 Program Text

```
/* nag_anderson_darling_uniform_prob (g08cjc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg08.h>

int main(void)
{
  /* Scalars */
  Integer       exit_status = 0, i, n;
  double        a2, mu, p;
  /* Arrays */
  double        *x = 0, *y = 0;
  /* Nag types */
  Nag_Boolean   issort;
  NagError      fail;

  printf("%s\n\n",
         "nag_anderson_darling_uniform_prob (g08cjc) Example Program Results");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read number of observations and parameter value */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT "", &n);
#else
  scanf("%"NAG_IFMT "", &n);
#endif
#ifdef _WIN32
  scanf_s("%lf", &mu);
#else
  scanf("%lf", &mu);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* 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++)
  {
#ifdef _WIN32
      scanf_s("%lf", x+i);
#else
      scanf("%lf", x+i);
#endif
  }
```

```
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

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

  /* Let nag_anderson_darling_uniform_prob (g08cjc) sort the uniform variates */
  issort = Nag_FALSE;

  /* Calculate the Anderson-Darling goodness-of-fit test statistic and its
     probability for the case of uniformly distributed data */
  INIT_FAIL(fail);
  /* nag_anderson_darling_uniform_prob (g08cjc) */
  nag_anderson_darling_uniform_prob(n, issort, y, &a2, &p, &fail);

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

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

  return exit_status;
}
```

10.2 Program Data

```
nag_anderson_darling_uniform_prob (g08cjc) Example Program Data
26 1.65 :: n, mu
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_uniform_prob (g08cjc) Example Program Results

 H0: data from exponential distribution with mean 1.650000
 Test statistic, A-squared:  0.182982
 Upper tail probability:     0.994487
```