NAG Library Function Document

nag_anderson_darling_normal_prob (g08ckc)

1 Purpose

nag_anderson_darling_normal_prob (g08ckc) calculates the Anderson–Darling goodness-of-fit test statistic and its probability for the case of a fully-unspecified Normal distribution.

2 Specification

```
#include <nag.h>
#include <nagg08.h>
void nag_anderson_darling_normal_prob (Integer n, Nag_Boolean issort,
     const double y[], double *ybar, double *yvar, double *a2, double *aa2,
     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 for the small sample correction:

$$\text{Adjusted } A^2 = A^2\left(1 + 0.75/n + 2.25/n^2\right),$$

for n observations.

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

Stephens M A and D'Agostino R B (1986) *Goodness-of-Fit Techniques* Marcel Dekker, New York

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**] – const double *Input*

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

Constraint: if **issort** = Nag_TRUE, the values must be sorted in ascending order.

4: **ybar** – double * *Output*

On exit: the maximum likelihood estimate of mean.

5: **yvar** – double * *Output*

On exit: the maximum likelihood estimate of variance.

6: **a2** – double * *Output*

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

7: **aa2** – double * *Output*

On exit: the adjusted A^2.

8: **p** – double * *Output*

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

9: **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_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.

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 are calculated using piecewise polynomial approximations to values estimated by simulation.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example calculates the A^2 statistics for data assumed to arise from a fully-unspecified Normal distribution and the p-value.

10.1 Program Text

```
/* nag_anderson_darling_normal_prob (g08ckc) 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)
{
  Integer        exit_status = 0;
  /* Scalars */
  double         a2, aa2, p, ybar, yvar;
  Integer        i, n;
  /* Arrays */
  double         *y = 0;
  /* Nag types */
  Nag_Boolean    issort;
  NagError       fail;

  printf("%s\n\n",
         "nag_anderson_darling_normal_prob (g08ckc) Example Program Results");

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

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

  /* Memory allocation */
  if (!(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", y+i);
#else
      scanf("%lf", y+i);
```

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

  /* Let nag_anderson_darling_normal_prob (g08ckc) sort the data */
  issort = Nag_FALSE;

  /* Calculate the Anderson-Darling goodness-of-fit test statistic and its
     probability for the case of a fully-unspecified Normal distribution */
  INIT_FAIL(fail);
  /* nag_anderson_darling_normal_prob (g08ckc) */
  nag_anderson_darling_normal_prob(n, issort, (const double *)y, &ybar, &yvar,
                                   &a2, &aa2, &p, &fail);

  /* Results */
  printf("%s ", "H0: data from Normal distribution with mean");
  printf("%6g ", ybar);
  printf("%s ", "and variance");
  printf("%6g\n", yvar);
  printf("%s", " Test statistic, A-squared: ");
  printf("%6g\n", a2);
  printf("%s", " Adjusted A-squared:        ");
  printf("%6g\n", aa2);
  printf("%s", " Upper tail probability:    ");
  printf("%6g\n", p);

 END:
  NAG_FREE(y);

  return exit_status;
}
```

10.2 Program Data

```
nag_anderson_darling_normal_prob (g08ckc) Example Program Data
26 :: n
 0.3131132  0.2520412  1.5788841  1.4416712 -0.8246043 -1.6466685
 0.7943184  1.2874915 -0.8347250  0.3352505  0.9434467  2.1099520
-0.2801654 -0.7843009  0.6218187  2.0963809  1.7170403 -0.1350142
 0.7982763 -0.2980977  1.2283043  1.5576090 -0.4828757  2.6070754
 0.1213996  0.1431621 :: end of observations
```

10.3 Program Results

```
nag_anderson_darling_normal_prob (g08ckc) Example Program Results

H0: data from Normal distribution with mean 0.563876 and variance 1.1386
 Test statistic, A-squared: 0.165956
 Adjusted A-squared:        0.171296
 Upper tail probability:    0.931155
```