NAG Library Function Document

nag_anderson_darling_exp_prob (g08clc)

1 Purpose

nag_anderson_darling_exp_prob (g08clc) calculates the Anderson–Darling goodness-of-fit test statistic
and its probability for the case of an unspecified exponential distribution.

2 Specification

```c
#include <nag.h>
#include <nagg08.h>
void nag_anderson_darling_exp_prob (Integer n, Nag_Boolean issort,
       const double y[], double *ybar, 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:

$$\text{Adjusted } A^2 = A^2(1 + 0.6/n),$$

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


5 Arguments

1:  
   `n` – Integer
   
   On entry: $n$, the number of observations.
   
   Constraint: $n > 1$.

2:  
   `issort` – Nag_Boolean
   
   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
   
   On entry: $y_i$, for $i = 1, 2, \ldots, n$, the $n$ observations.
   
   Constraint: if `issort = Nag_TRUE`, values must be sorted in ascending order. Each $y_i$ must be
   greater than zero.

4:  
   `ybar` – double *
   
   On exit: the maximum likelihood estimate of mean.
5: \textbf{a2} – double * \\
\textit{On exit: } A^2, \text{ the Anderson–Darling test statistic.}

6: \textbf{aa2} – double * \\
\textit{On exit: } the adjusted A^2.

7: \textbf{p} – double * \\
\textit{On exit: } p, \text{ the upper tail probability for the adjusted } A^2.

8: \textbf{fail} – NagError * \\
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL} \\
Dynamic memory allocation failed. \\
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM} \\
On entry, argument \textit{value} had an illegal value.

\textbf{NE_BOUND} \\
The data in \textit{y} must be greater than zero.

\textbf{NE_INT} \\
On entry, \textit{n} = \textit{value}. \\
Constraint: \textit{n} > 1.

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

\textbf{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.

\textbf{NE_NOT_INCREASING} \\
\textbf{issort } = \text{Nag_TRUE} \text{ and the data in } \textit{y} \text{ 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 an unspecified exponential distribution and calculates the $p$-value.

10.1 Program Text

```c
/* nag_anderson_darling_exp_prob (g08clc) 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, aa2, p, ybar;
    /* Array */
    double *y = 0;
    /* NAG types */
    Nag_Boolean issort;
    NagError fail;

    printf("%s\n", "nag_anderson_darling_exp_prob (g08clc) 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++)
    {
```
```c
#ifndef _WIN32
    scanf_s("%lf", y+i);
#else
    scanf("%lf", y+i);
#endif
#endif

/* Let nag_anderson_darling_exp_prob (g08clc) sort the data */
issort = Nag_FALSE;

/* Calculate the Anderson-Darling goodness-of-fit test statistic and its
   probability for the case of an unspecified exponential distribution */
INIT_FAIL(fail);
/* nag_anderson_darling_exp_prob (g08clc) */
nag_anderson_darling_exp_prob(n, issort, (const double *)y, &ybar, &a2, &aa2,
&p, &fail);

/* Results */
printf("%s ", "H0: data from exponential distribution with mean");
printf("%6g\n", ybar);
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_exp_prob (g08clc) 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_exp_prob (g08clc) Example Program Results

H0: data from exponential distribution with mean 1.52402
    Test statistic, A-squared: 0.161632
    Adjusted A-squared: 0.165362
    Upper tail probability: 0.983115
```