

## NAG Library Function Document

### nag\_shapiro\_wilk\_test (g01ddc)

#### 1 Purpose

nag\_shapiro\_wilk\_test (g01ddc) calculates Shapiro and Wilk's  $W$  statistic and its significance level for testing Normality.

#### 2 Specification

```
#include <nag.h>
#include <nagg01.h>

void nag_shapiro_wilk_test (Integer n, const double x[],
    Nag_Boolean calc_wts, double a[], double *w, double *pw, NagError *fail)
```

#### 3 Description

nag\_shapiro\_wilk\_test (g01ddc) calculates Shapiro and Wilk's  $W$  statistic and its significance level for any sample size between 3 and 5000. It is an adaptation of the Applied Statistics Algorithm AS R94, see Royston (1995). The full description of the theory behind this algorithm is given in Royston (1992).

Given a set of observations  $x_1, x_2, \dots, x_n$  sorted into either ascending or descending order (nag\_double\_sort (m01cac) may be used to sort the data) this function calculates the value of Shapiro and Wilk's  $W$  statistic defined as:

$$W = \frac{\left( \sum_{i=1}^n a_i x_i \right)^2}{\sum_{i=1}^n (x_i - \bar{x})^2},$$

where  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$  is the sample mean and  $a_i$ , for  $i = 1, 2, \dots, n$ , are a set of 'weights' whose values depend only on the sample size  $n$ .

On exit, the values of  $a_i$ , for  $i = 1, 2, \dots, n$ , are only of interest should you wish to call the function again to calculate  $w$  and its significance level for a different sample of the same size.

It is recommended that the function is used in conjunction with a Normal ( $Q - Q$ ) plot of the data. Function nag\_normal\_scores\_exact (g01dac) can be used to obtain the required Normal scores.

#### 4 References

- Royston J P (1982) Algorithm AS 181: the  $W$  test for normality *Appl. Statist.* **31** 176–180
- Royston J P (1986) A remark on AS 181: the  $W$  test for normality *Appl. Statist.* **35** 232–234
- Royston J P (1992) Approximating the Shapiro–Wilk's  $W$  test for non-normality *Statistics & Computing* **2** 117–119
- Royston J P (1995) A remark on AS R94: A remark on Algorithm AS 181: the  $W$  test for normality *Appl. Statist.* **44(4)** 547–551

## 5 Arguments

- 1: **n** – Integer *Input*  
*On entry:*  $n$ , the sample size.  
*Constraint:*  $3 \leq n \leq 5000$ .
- 2: **x[n]** – const double *Input*  
*On entry:* the ordered sample values,  $x_i$ , for  $i = 1, 2, \dots, n$ .
- 3: **calc\_wts** – Nag\_Boolean *Input*  
*On entry:* must be set to Nag\_TRUE if you wish nag\_shapiro\_wilk\_test (g01ddc) to calculate the elements of **a**.  
**calc\_wts** should be set to Nag\_FALSE if you have saved the values in **a** from a previous call to nag\_shapiro\_wilk\_test (g01ddc).  
 If in doubt, set **calc\_wts** equal to Nag\_TRUE.
- 4: **a[n]** – double *Input/Output*  
*On entry:* if **calc\_wts** has been set to Nag\_FALSE then before entry **a** must contain the  $n$  weights as calculated in a previous call to nag\_shapiro\_wilk\_test (g01ddc), otherwise **a** need not be set.  
*On exit:* the  $n$  weights required to calculate **w**.
- 5: **w** – double \* *Output*  
*On exit:* the value of the statistic, **w**.
- 6: **pw** – double \* *Output*  
*On exit:* the significance level of **w**.
- 7: **fail** – NagError \* *Input/Output*  
 The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALL\_ELEMENTS\_EQUAL

On entry, all elements of **x** are equal.

### 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  $\langle value \rangle$  had an illegal value.

### NE\_INT\_ARG\_GT

On entry,  $n = \langle value \rangle$ .  
 Constraint:  $n \leq \langle value \rangle$ .

### NE\_INT\_ARG\_LT

On entry,  $n = \langle value \rangle$ .  
 Constraint:  $n \geq 3$ .

**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\_NON\_MONOTONIC**

On entry, elements of  $\mathbf{x}$  not in order.  $\mathbf{x}[\langle value \rangle] = \langle value \rangle$ ,  $\mathbf{x}[\langle value \rangle] = \langle value \rangle$ ,  
 $\mathbf{x}[\langle value \rangle] = \langle value \rangle$ .

**7 Accuracy**

There may be a loss of significant figures for large  $n$ .

**8 Parallelism and Performance**

Not applicable.

**9 Further Comments**

The time taken by `nag_shapiro_wilk_test` (g01ddc) depends roughly linearly on the value of  $n$ .

For very small samples the power of the test may not be very high.

The contents of the array  $\mathbf{a}$  should not be modified between calls to `nag_shapiro_wilk_test` (g01ddc) for a given sample size, unless `calc_wts` is reset to `Nag_TRUE` before each call of `nag_shapiro_wilk_test` (g01ddc).

The Shapiro and Wilk's  $W$  test is very sensitive to ties. If the data has been rounded the test can be improved by using Sheppard's correction to adjust the sum of squares about the mean. This produces an adjusted value of  $\mathbf{w}$ ,

$$WA = W \frac{\sum x_{(i)} - \bar{x}^2}{\left\{ \sum_{i=1}^n x_{(i)} = \bar{x}^2 - \frac{n-1}{12} \omega^2 \right\}}$$

where  $\omega$  is the rounding width.  $WA$  can be compared with a standard Normal distribution, but a further approximation is given by Royston (1986).

If  $n > 5000$ , a value for  $\mathbf{w}$  and  $\mathbf{pw}$  is returned, but its accuracy may not be acceptable. See Section 4 for more details.

**10 Example**

This example tests the following two samples (each of size 20) for Normality.

Sample Number	Data
1	0.11, 7.87, 4.61, 10.14, 7.95, 3.14, 0.46, 4.43, 0.21, 4.75, 0.71, 1.52, 3.24, 0.93, 0.42, 4.97, 9.53, 4.55, 0.47, 6.66
2	1.36, 1.14, 2.92, 2.55, 1.46, 1.06, 5.27, -1.11, 3.48, 1.10, 0.88, -0.51, 1.46, 0.52, 6.20, 1.69, 0.08, 3.67, 2.81, 3.49

The elements of **a** are calculated only in the first call of `nag_shapiro_wilk_test` (g01ddc), and are re-used in the second call.

## 10.1 Program Text

```

/* nag_shapiro_wilk_test (g01ddc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 * Mark 8 revised, 2004.
 *
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nagg01.h>
#include <nagm01.h>

int main(void)
{
    /* Local variables */
    Nag_Boolean calwts;
    Integer      exit_status = 0, i, j, n;
    NagError     fail;
    double       *a = 0, pw, w, *x = 0;

    INIT_FAIL(fail);

    printf("nag_shapiro_wilk_test (g01ddc) Example Program Results\n");

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

    calwts = Nag_TRUE;
#ifdef _WIN32
    scanf_s("%"NAG_IFMT" ", &n);
#else
    scanf("%"NAG_IFMT" ", &n);
#endif
    if (n >= 3)
    {
        if (!(a = NAG_ALLOC(n, double)) ||
            !(x = NAG_ALLOC(n, double)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    else
    {
        printf("Invalid n.\n");
        exit_status = 1;
        return exit_status;
    }
    for (j = 1; j <= 2; ++j)
    {
        for (i = 1; i <= n; ++i)
#ifdef _WIN32
            scanf_s("%lf ", &x[i - 1]);
#else
            scanf("%lf ", &x[i - 1]);
#endif
    }
}

```

```

/* nag_double_sort (m01cac).
 * Quicksort of set of values of data type double
 */
nag_double_sort(x, (size_t) n, Nag_Ascending, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_double_sort (m01cac).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}
/* nag_shapiro_wilk_test (g01ddc).
 * Shapiro and Wilk's W test for Normality
 */
nag_shapiro_wilk_test(n, x, calwts, a, &w, &pw, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_shapiro_wilk_test (g01ddc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}

printf("\n For sample number %2"NAG_IFMT", value of W statistic = "
        "%7.4f\n", j, w);
printf("                                Significance level is %8.4f\n",
        pw);
calwts = Nag_FALSE;
}
END:
NAG_FREE(a);
NAG_FREE(x);
return exit_status;
}

```

## 10.2 Program Data

```

nag_shapiro_wilk_test (g01ddc) Example Program Data
20
0.11  7.87  4.61 10.14  7.95  3.14  0.46  4.43  0.21  4.75
0.71  1.52  3.24  0.93  0.42  4.97  9.53  4.55  0.47  6.66
1.36  1.14  2.92  2.55  1.46  1.06  5.27 -1.11  3.48  1.10
0.88 -0.51  1.46  0.52  6.20  1.69  0.08  3.67  2.81  3.49

```

## 10.3 Program Results

```

nag_shapiro_wilk_test (g01ddc) Example Program Results

```

```

For sample number  1, value of W statistic =  0.9005
                   Significance level is  0.0421

For sample number  2, value of W statistic =  0.9590
                   Significance level is  0.5246

```

---