

NAG Library Function Document

nag_sign_test (g08aac)

1 Purpose

nag_sign_test (g08aac) performs the Sign test on two related samples of size n .

2 Specification

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

void nag_sign_test (Integer n, const double x[], const double y[],
                   Integer *s, double *p, Integer *non_tied, NagError *fail)
```

3 Description

The Sign test investigates the median difference between pairs of scores from two matched samples of size n , denoted by $\{x_i, y_i\}$, for $i = 1, 2, \dots, n$. The hypothesis under test, H_0 , often called the null hypothesis, is that the medians are the same, and this is to be tested against a one- or two-sided alternative H_1 (see below).

nag_sign_test (g08aac) computes:

- the test statistic S , which is the number of pairs for which $x_i < y_i$;
- the number n_1 of non-tied pairs ($x_i \neq y_i$);
- the lower tail probability p corresponding to S (adjusted to allow the complement $(1 - p)$ to be used in an upper one tailed or a two tailed test). p is the probability of observing a value $\leq S$ if $S < \frac{1}{2}n_1$, or of observing a value $< S$ if $S > \frac{1}{2}n_1$, given that H_0 is true. If $S = \frac{1}{2}n_1$, p is set to 0.5.

Suppose that a significance test of a chosen size α is to be performed (i.e., α is the probability of rejecting H_0 when H_0 is true; typically α is a small quantity such as 0.05 or 0.01). The returned value of p can be used to perform a significance test on the median difference, against various alternative hypotheses H_1 , as follows

- H_1 : median of $x \neq$ median of y . H_0 is rejected if $2 \times \min(p, 1 - p) < \alpha$.
- H_1 : median of $x >$ median of y . H_0 is rejected if $p < \alpha$.
- H_1 : median of $x <$ median of y . H_0 is rejected if $1 - p < \alpha$.

4 References

Siegel S (1956) *Non-parametric Statistics for the Behavioral Sciences* McGraw-Hill

5 Arguments

- n** – Integer *Input*
On entry: n , the size of each sample.
Constraint: $n \geq 1$.
- x[n]** – const double *Input*
- y[n]** – const double *Input*
On entry: $x[i - 1]$ and $y[i - 1]$ must be set to the i th pair of data values, $\{x_i, y_i\}$, for $i = 1, 2, \dots, n$.

- 4: **s** – Integer * *Output*
On exit: the Sign test statistic, S .
- 5: **p** – double * *Output*
On exit: the lower tail probability, p , corresponding to S .
- 6: **non_tied** – Integer * *Output*
On exit: the number of non-tied pairs, n_1 .
- 7: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_INT

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 1 .

7 Accuracy

The tail probability, p , is computed using the relationship between the binomial and beta distributions. For $n_1 < 120$, p should be accurate to at least 4 significant figures, assuming that the machine has a precision of 7 or more digits. For $n_1 \geq 120$, p should be computed with an absolute error of less than 0.005. For further details see nag_prob_beta_dist (g01eec).

8 Parallelism and Performance

nag_sign_test (g08aac) is not threaded in any implementation.

9 Further Comments

The time taken by nag_sign_test (g08aac) is small, and increases with n .

10 Example

This example is taken from page 69 of Siegel (1956). The data relates to ratings of ‘insight into paternal discipline’ for 17 sets of parents, recorded on a scale from 1 to 5.

10.1 Program Text

```
/* nag_sign_test (g08aac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
```

```

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg08.h>

int main(void)
{
    Integer exit_status = 0, i, n, non_tied, s;
    NagError fail;
    double p, *x = 0, *y = 0;

    INIT_FAIL(fail);

    printf("nag_sign_test (g08aac) Example Program Results\n");

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

    n = 17;
    if (!(x = NAG_ALLOC(n, double))
        || !(y = NAG_ALLOC(n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    for (i = 1; i <= n; i++)
#ifdef _WIN32
        scanf_s("%lf", &x[i - 1]);
#else
        scanf("%lf", &x[i - 1]);
#endif

    for (i = 1; i <= n; i++)
#ifdef _WIN32
        scanf_s("%lf", &y[i - 1]);
#else
        scanf("%lf", &y[i - 1]);
#endif

    printf("\n%s\n\n", "Sign test");
    printf("%s\n\n", "Data values");
    for (i = 1; i <= n; i++)
        printf("%3.0f%s", x[i - 1], i % n ? "" : "\n");
    printf("\n");

    for (i = 1; i <= n; i++)
        printf("%3.0f%s", y[i - 1], i % n ? "" : "\n");
    printf("\n");

    /* nag_sign_test (g08aac).
     * Sign test on two paired samples
     */
    nag_sign_test(n, x, y, &s, &p, &non_tied, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_sign_test (g08aac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    printf("%s%5" NAG_IFMT "\n", "Test statistic", s);
    printf("%s%5" NAG_IFMT "\n", "Observations", non_tied);
    printf("%s%5.3f\n", "Lower tail prob.", p);
}

```

```
END:
  NAG_FREE(x);
  NAG_FREE(y);
  return exit_status;
}
```

10.2 Program Data

```
nag_sign_test (g08aac) Example Program Data
 4 4 5 5 3 2 5 3 1 5 5 5 4 5 5 5
 2 3 3 3 3 3 3 3 2 3 2 2 5 2 5 3 1
```

10.3 Program Results

```
nag_sign_test (g08aac) Example Program Results
```

Sign test

Data values

```
 4 4 5 5 3 2 5 3 1 5 5 5 4 5 5 5
 2 3 3 3 3 3 3 3 2 3 2 2 5 2 5 3 1
```

```
Test statistic      3
Observations       14
Lower tail prob. 0.029
```
