NAG Library Function Document

nag_wilcoxon_test (g08agc)

1 Purpose

nag_wilcoxon_test (g08agc) performs the Wilcoxon signed rank test on a single sample of size n.

2 Specification

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

```
void nag_wilcoxon_test (Integer n, const double x[], double median,
     Nag_TailProbability tail, Nag_IncSignZeros zeros, double *w, double *z,
     double *p, Integer *non_zero, NagError *fail)
```

3 Description

The Wilcoxon one sample signed rank test may be used to test whether a particular sample came from a population with a specified median. It is assumed that the population distribution is symmetric. The data consist of a single sample of n observations denoted by $x_1, x_2, \ldots, x_n$. This sample may arise from the difference between pairs of observations from two matched samples of equal size taken from two populations, in which case the test may be used to test whether the median of the first population is the same as that of the second population.

The hypothesis under test, H_0, often called the null hypothesis, is that the median is equal to some given value (X_{med}), and this is to be tested against an alternative hypothesis H_1 which is

H_1 : population median $\neq X_{med}$; or

H_1 : population median $> X_{med}$; or

H_1 : population median $< X_{med}$,

using a two tailed, upper tailed or lower tailed probability respectively. You select the alternative hypothesis by choosing the appropriate tail probability to be computed (see the description of argument **tail** in Section 5).

The Wilcoxon test differs from the Sign test (see nag_sign_test (g08aac)) in that the magnitude of the scores is taken into account, rather than simply the direction of such scores.

The test procedure is as follows:

(a) For each x_i, for $i = 1, 2, \ldots, n$, the signed difference $d_i = x_i - X_{med}$ is found, where X_{med} is a given test value for the median of the sample.

(b) The absolute differences $|d_i|$ are ranked with rank r_i and any tied values of $|d_i|$ are assigned the average of the tied ranks. You may choose whether or not to ignore any cases where $d_i = 0$ by removing them before or after ranking (see the description of the argument **zeros** in Section 5).

(c) The number of nonzero d_i's is found.

(d) To each rank is affixed the sign of the d_i to which it corresponds. Let $s_i = \text{sign}(d_i) r_i$.

(e) The sum of the positive-signed ranks, $W = \sum_{s_i > 0} s_i = \sum_{i=1}^{n} \max(s_i, 0.0)$, is calculated.

nag_wilcoxon_test (g08agc) returns:

(a) The test statistic W;

(b) The number n_1 of nonzero d_i's;

(c) The approximate Normal test statistic z, where

$$z = \frac{\left(W - \frac{n_1(n_1+1)}{4}\right) - \text{sign}\left(W - \frac{n_1(n_1+1)}{4}\right) \times \frac{1}{2}}{\sqrt{\frac{1}{4}\sum_{i=1}^{n} s_i^2}}$$

(d) The tail probability, p, corresponding to W, depending on the choice of the alternative hypothesis, H_1.

If $n_1 \leq 80$, p is computed exactly; otherwise, an approximation to p is returned based on an approximate Normal statistic corrected for continuity according to the tail specified.

The value of p can be used to perform a significance test on the median against the alternative hypothesis. Let α be the size of the significance test (that is, α is the probability of rejecting H_0 when H_0 is true). If $p < \alpha$ then the null hypothesis is rejected. Typically α might be 0.05 or 0.01.

4 References

Conover W J (1980) *Practical Nonparametric Statistics* Wiley

Neumann N (1988) Some procedures for calculating the distributions of elementary nonparametric teststatistics *Statistical Software Newsletter* **14(3)** 120–126

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

5 Arguments

1: **n** – Integer *Input*

On entry: the size of the sample, n.

Constraint: $\mathbf{n} \geq 1$.

2: **x**[**n**] – const double *Input*

On entry: the sample observations, $x_1, x_2, \ldots, x_n$.

3: **median** – double *Input*

On entry: the median test value, X_{med}.

4: **tail** – Nag_TailProbability *Input*

On entry: indicates the choice of tail probability, and hence the alternative hypothesis.

tail = Nag_TwoTail
A two tailed probability is calculated and the alternative hypothesis is H_1: population median $\neq \mathrm{X}_{med}$.

tail = Nag_UpperTail
An upper tailed probability is calculated and the alternative hypothesis is H_1: population median $> \mathrm{X}_{med}$.

tail = Nag_LowerTail
A lower tailed probability is calculated and the alternative hypothesis is H_1: population median $< \mathrm{X}_{med}$.

Constraint: **tail** = Nag_TwoTail, Nag_UpperTail or Nag_LowerTail.

5: **zeros** – Nag_IncSignZeros *Input*

On entry: indicates whether or not to include the cases where $d_i = 0.0$ in the ranking of the d_i's.

zeros = Nag_IncSignZerosY
All $d_i = 0.0$ are included when ranking.

zeros = Nag_IncSignZerosN
All $d_i = 0.0$, are ignored, that is all cases where $d_i = 0.0$ are removed before ranking.

Constraint: **zeros** = Nag_IncSignZerosY or Nag_IncSignZerosN.

6: **w** – double * *Output*

On exit: the Wilcoxon rank sum statistic, W, being the sum of the positive ranks.

7: **z** – double * *Output*

On exit: the approximate Normal test statistic, z, as described in Section 3.

8: **p** – double * *Output*

On exit: the tail probability, p, as specified by the argument **tail**.

9: **non_zero** – Integer * *Output*

On exit: the number of nonzero d_i's, n_1.

10: **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.

NE_BAD_PARAM

On entry, argument **tail** had an illegal value.

On entry, argument **zeros** had an illegal value.

NE_G08AG_SAMP_IDEN

The whole sample is identical to the given median test value.

NE_INT_ARG_LT

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

7 Accuracy

The approximation used to calculate p when $n_1 > 80$ will return a value with a relative error of less than 10 percent for most cases. The error may increase for cases where there are a large number of ties in the sample.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_wilcoxon_test (g08agc) increases with n_1, until $n_1 > 80$, from which point on the approximation is used. The time decreases significantly at this point and increases again modestly with n_1 for $n_1 > 80$.

10 Example

The example program performs the Wilcoxon signed rank test on two matched samples of size 8, taken from two populations. The distribution of the differences between pairs of observations from the two populations is assumed to be symmetric. The test is used to test whether the medians of the two distributions of the populations are equal or not. The test statistic, the approximate Normal statistic and the two tailed probability are computed and printed.

10.1 Program Text

```
/* nag_wilcoxon_test (g08agc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 6, 2000.
 */

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

int main(void)
{
  Integer  exit_status = 0, i, n, non_zero;
  NagError fail;
  double   *data = 0, median, p, w, *x = 0, *y = 0, z;

  INIT_FAIL(fail);

  printf("nag_wilcoxon_test (g08agc) Example Program Results");

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

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &n);
#else
  scanf("%"NAG_IFMT"", &n);
#endif
  if (!(x = NAG_ALLOC(n, double))
     || !(y = NAG_ALLOC(n, double))
     || !(data = 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\n");
  printf("%s\n", "Wilcoxon one sample signed ranks test");
  printf("\n");
  printf("%s", "Data values\n");
  for (i = 1; i <= n; ++i)
    printf("%5.1f%s", x[i - 1], i%8?"":"\n");
  for (i = 1; i <= n; ++i)
    printf("%5.1f%s", y[i - 1], i%8?"":"\n");

  for (i = 1; i <= n; ++i)
    data[i - 1] = x[i - 1] - y[i - 1];
  median = 0.0;
  /* nag_wilcoxon_test (g08agc).
   * Performs the Wilcoxon one-sample (matched pairs) signed
   * rank test
   */
  nag_wilcoxon_test(n, data, median, Nag_TwoTail, Nag_IncSignZerosN, &w, &z,
                    &p, &non_zero, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_wilcoxon_test (g08agc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n\n");
  printf("%s%8.4f\n", "Test statistic            = ", w);
  printf("%s%8.4f\n", "Normalized test statistic = ", z);
  printf("%s%8"NAG_IFMT"\n", "Degrees of freedom        = ", non_zero);
  printf("%s%8.4f\n", "Two tail probability      = ", p);

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

10.2 Program Data

```
nag_wilcoxon_test (g08agc) Example Program Data
 8
 82 69 73 43 58 56 76 65
 63 42 74 37 51 43 80 62
```

10.3 Program Results

```
nag_wilcoxon_test (g08agc) Example Program Results

Wilcoxon one sample signed ranks test

Data values
 82.0 69.0 73.0 43.0 58.0 56.0 76.0 65.0
 63.0 42.0 74.0 37.0 51.0 43.0 80.0 62.0


Test statistic            =  32.0000
Normalized test statistic =   1.8904
Degrees of freedom        =        8
Two tail probability      =   0.0547
```