1 Purpose

nag_surviv_logrank (g12abc) calculates the rank statistics, which can include the logrank test, for comparing survival curves.

2 Specification

```c
#include <nag.h>
#include <nagg12.h>

void nag_surviv_logrank (Integer n, const double t[], const Integer ic[],
const Integer grp[], Integer ngrp, const Integer ifreq[],
const double wt[], double *ts, Integer *df, double *p,
const double expt[], const Integer *nd, Integer di[], Integer ni[],
Integer ldn,
NagError *fail)
```

3 Description

A survivor function, \( S(t) \), is the probability of surviving to at least time \( t \). Given a series of \( n \) failure or right-censored times from \( g \) groups nag_surviv_logrank (g12abc) calculates a rank statistic for testing the null hypothesis

\[ H_0 : S_1(t) = S_2(t) = \cdots = S_g(t), \quad t \leq \tau \]

where \( \tau \) is the largest observed time, against the alternative hypothesis

\[ H_1 : \text{at least one of the } S_i(t) \text{ differ, for some } t \leq \tau. \]

Let \( t_i \), for \( i = 1, 2, \ldots, n_d \), denote the list of distinct failure times across all \( g \) groups and \( w_i \) a series of \( n_d \) weights. Let \( d_{ij} \) denote the number of failures at time \( t_i \) in group \( j \) and \( n_{ij} \) denote the number of observations in the group \( j \) that are known to have not failed prior to time \( t_i \), i.e., the size of the risk set for group \( j \) at time \( t_i \). If a censored observation occurs at time \( t_i \) then that observation is treated as if the censoring had occurred slightly after \( t_i \) and therefore the observation is counted as being part of the risk set at time \( t_i \). Finally let

\[
d_i = \sum_{j=1}^{g} d_{ij} \quad \text{and} \quad n_i = \sum_{j=1}^{g} n_{ij}.
\]

The (weighted) number of observed failures in the \( j \)th group, \( O_j \), is therefore given by

\[
O_j = \sum_{i=1}^{n_d} w_i d_{ij}
\]

and the (weighted) number of expected failures in the \( j \)th group, \( E_j \), by

\[
E_j = \sum_{i=1}^{n_d} w_i \frac{n_{ij} d_i}{n_i}.
\]

If \( x \) denotes the vector of differences \( x = (O_1 - E_1, O_2 - E_2, \ldots, O_g - E_g) \) and

\[
V_{jk} = \sum_{i=1}^{n_d} w_i^2 \left( \frac{d_i(n_i - d_i)(n_{ik}I_{jk} - n_{ij}n_{ik})}{n_i^2(n_i - 1)} \right)
\]

where \( I_{jk} = 1 \) if \( j = k \) and 0 otherwise, then the rank statistic, \( T \), is calculated as

\[
T = x V^{-} x^T
\]
where $V^-$ denotes a generalized inverse of the matrix $V$. Under the null hypothesis, $T \sim \chi^2_{\nu}$ where the degrees of freedom, $\nu$, is taken as the rank of the matrix $V$.

4 References


5 Arguments

1: $n$ – Integer

*Input*

On entry: $n$, the number of failure and censored times.

Constraint: $n \geq 2$.

2: $t[n]$ – const double

*Input*

On entry: the observed failure and censored times; these need not be ordered.

Constraint: $t[i - 1] \neq t[j - 1]$ for at least one $i \neq j$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$.

3: $ic[n]$ – const Integer

*Input*

On entry: $ic[i - 1]$ contains the censoring code of the $i$th observation, for $i = 1, 2, \ldots, n$.

$ic[i - 1] = 0$

the $i$th observation is a failure time.

$ic[i - 1] = 1$

the $i$th observation is right-censored.

Constraints:

$ic[i - 1] = 0$ or $1$, for $i = 1, 2, \ldots, n$;

$ic[i - 1] = 0$ for at least one $i$.

4: $grp[n]$ – const Integer

*Input*

On entry: $grp[i - 1]$ contains a flag indicating which group the $i$th observation belongs in, for $i = 1, 2, \ldots, n$.

Constraints:

$1 \leq grp[i - 1] \leq ngrp$, for $i = 1, 2, \ldots, n$;

each group must have at least one observation.

5: $ngrp$ – Integer

*Input*

On entry: $g$, the number of groups.

Constraint: $2 \leq ngrp \leq n$.

6: $ifreq[dim]$ – const Integer

*Input*

Note: the dimension, $dim$, of the array $ifreq$ must be at least $n$, unless $ifreq$ is NULL.
On entry: optionally, the frequency (number of observations) that each entry in \( t \) corresponds to. If ifreq is NULL then each entry in \( t \) is assumed to correspond to a single observation, i.e., a frequency of 1 is assumed.

Constraint: if ifreq is not NULL, ifreq[\( i - 1 \)] \( \geq 0 \), for \( i = 1, 2, \ldots, n \).

7: \( wt[dim] \) – const double  

Note: the dimension, \( dim \), of the array \( wt \) must be at least ldn, unless \( wt \) is NULL.

On entry: optionally, the \( n_d \) weights, \( w_i \), where \( n_d \) is the number of distinct failure times. If \( wt \) is NULL then \( w_i = 1 \) for all \( i \).

Constraint: if \( wt \) is not NULL, \( wt[i - 1] \geq 0.0 \), for \( i = 1, 2, \ldots, n_d \).

8: \( ts \) – double *  

On exit: \( T \), the test statistic.

9: \( df \) – Integer *  

On exit: \( \nu \), the degrees of freedom.

10: \( p \) – double *  

On exit: \( P(\bar{X} \geq T) \), when \( \bar{X} \sim \chi^2_t \), i.e., the probability associated with \( ts \).

11: \( obsd[agr] \) – double  

On exit: \( O_i \), the observed number of failures in each group.

12: \( ext[agr] \) – double  

On exit: \( E_i \), the expected number of failures in each group.

13: \( nd \) – Integer *  

On exit: \( n_d \), the number of distinct failure times.

14: \( di[ldn] \) – Integer  

On exit: the first \( nd \) elements of \( di \) contain \( d_i \), the number of failures, across all groups, at time \( t_i \).

15: \( ni[ldn] \) – Integer  

On exit: the first \( nd \) elements of \( ni \) contain \( n_i \), the size of the risk set, across all groups, at time \( t_i \).

16: \( ldn \) – Integer  

On entry: the size of arrays \( di \) and \( ni \). As \( n_d \leq n \), if \( n_d \) is not known a priori then a value of \( n \) can safely be used for \( ldn \).

Constraint: \( ldn \geq n_d \), the number of unique failure times.

17: \( fail \) – NagError *  

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 \( \langle \text{value} \rangle \) had an illegal value.

**NE_GROUP_OBSERV**
On entry, group \( \langle \text{value} \rangle \) has no observations.

**NE_INT**
On entry, \( \text{ldn} = \langle \text{value} \rangle \).
Constraint: \( \text{ldn} \geq \langle \text{value} \rangle \).

On entry, \( \text{n} = \langle \text{value} \rangle \).
Constraint: \( \text{n} \geq 2 \).

**NE_INT_2**
On entry, \( \text{ngrp} = \langle \text{value} \rangle \) and \( \text{n} = \langle \text{value} \rangle \).
Constraint: \( 2 \leq \text{ngrp} \leq \text{n} \).

**NE_INT_ARRAY**
On entry, \( \text{grp}[\langle \text{value} \rangle] = \langle \text{value} \rangle \) and \( \text{ngrp} = \langle \text{value} \rangle \).
Constraint: \( 1 \leq \text{grp}[i - 1] \leq \text{ngrp} \).

**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_INVALID_CENSOR_CODE**
On entry, \( \text{ic}[\langle \text{value} \rangle] = \langle \text{value} \rangle \).
Constraint: \( \text{ic}[i - 1] = 0 \) or \( 1 \).

**NE_INVALID_FREQ**
On entry, \( \text{ifreq}[\langle \text{value} \rangle] = \langle \text{value} \rangle \).
Constraint: \( \text{ifreq}[i - 1] \geq 0 \).

**NE_NEG_WEIGHT**
On entry, \( \text{wt}[\langle \text{value} \rangle] = \langle \text{value} \rangle \).
Constraint: \( \text{wt}[i - 1] \geq 0.0 \).

**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_OBSERVATIONS**
On entry, all observations are censored.

**NE_TIME_SERIES_IDEN**
On entry, all the times in \( \text{t} \) are the same.

**NE_ZERO_DF**
The degrees of freedom are zero.
7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_surviv_logrank (g12abc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_surviv_logrank (g12abc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

The use of different weights in the formula given in Section 3 leads to different rank statistics being calculated. The logrank test has \( w_i = 1 \), for all \( i \), which is the equivalent of calling nag_surviv_logrank (g12abc) when \( wt \) is NULL. Other rank statistics include Wilcoxon (\( w_i = n_i \)), Tarone–Ware (\( w_i = \sqrt{n_i} \)) and Peto–Peto (\( w_i = \frac{S(t_i)}{n_i} \)) amongst others.

Calculation of any test, other than the logrank test, will probably require nag_surviv_logrank (g12abc) to be called twice, once to calculate the values of \( n_i \) and \( d_i \) to facilitate in the computation of the required weights, and once to calculate the test statistic itself.

10 Example

This example compares the time to death for 51 adults with two different types of recurrent gliomas (brain tumour), astrocytoma and glioblastoma, using a logrank test. For further details on the data see Rostomily et al. (1994).

10.1 Program Text

/* nag_surviv_logrank (g12abc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2011. */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg12.h>

int main(void)
{

/* Integer scalar and array declarations */
Integer i, n, ngrp, lfreq, df, nd, ldn, exit_status;
Integer *ic = 0, *grp = 0, *ifreq = 0, *di = 0, *ni = 0;

/* NAG structures */
NagError fail;

/* Double scalar and array declarations */
double ts, p;
double *t = 0, *obsd = 0, *expt = 0;
/* Performing a logrank test, so no weights needed */
double *wt = 0;
exit_status = 0;
/* Initialise the error structure */
INIT_FAIL(fail);

printf("nag_surviv_logrank (g12abc) Example Program Results\n");

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

/* Read in the problem size */
#ifdef _WIN32
    scanf_s("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT"")[\n ]", &n, &ngrp, &lfreq);
#else
    scanf("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT"")[\n ]", &n, &ngrp, &lfreq);
#endif
ldn = n;

/* Allocate memory to input and output arrays */
if (!((ic = NAG_ALLOC(n, Integer)) ||
    !(grp = NAG_ALLOC(n, Integer)) ||
    !(obsd = NAG_ALLOC(ngrp, double)) ||
    !(expt = NAG_ALLOC(ngrp, double)) ||
    !(di = NAG_ALLOC(ldn, Integer)) ||
    !(ni = NAG_ALLOC(ldn, Integer))))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

if (lfreq > 0) {
    lfreq = n;
    if (!((ifreq = NAG_ALLOC(lfreq, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}

/* Read in the times, censored flag, group information and if supplied the frequencies */
for (i=0; i<n; i++) {
#ifdef _WIN32
    scanf_s("%lf%"NAG_IFMT"%"NAG_IFMT"", &t[i], &ic[i], &grp[i]);
#else
    scanf("%lf%"NAG_IFMT"%"NAG_IFMT"", &t[i], &ic[i], &grp[i]);
#endif
    if (lfreq > 0)
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n ]", &ifreq[i]);
#else
    scanf("%"NAG_IFMT"%*[\n ]", &ifreq[i]);
#endif
}

/* Calculate the logrank statistic using nag_surviv_logrank (g12abc) */
if (fail.code != NE_NOERROR) {
    printf("g12abc\nNAG Library Manual\ng12abc.6 Mark 25\n");
}
printf("Error from nag_surviv_logrank (g12abc).\n%s\n", fail.message);
exit_status = 1;
goto END;
}

/* Display the test information */
printf("\n");
printf(" Observed Expected\n");
for (i = 0; i < ngrp; i++)
    printf(" %-5s %1"NAG_IFMT" %8.2f %8.2f\n", "Group",i+1,obsd[i], expt[i]);
printf("\n");
printf(" No. Unique Failure Times = %3"NAG_IFMT"\n", nd);
printf("\n");
printf(" Test Statistic = %8.4f\n", ts);
printf(" Degrees of Freedom = %3"NAG_IFMT"\n", df);
printf(" p-value = %8.4f\n", p);

END:
NAG_FREE(t);
NAG_FREE(ic);
NAG_FREE(ifreq);
NAG_FREE(wt);
NAG_FREE(grp);
NAG_FREE(obsd);
NAG_FREE(expt);
NAG_FREE(di);
NAG_FREE(ni);

return exit_status;
}

10.2 Program Data

nag_surviv_logrank (g12abc) Example Program Data
51 2 0 :: n, ngrp, ifreq
6.0 0 1
13.0 0 1
21.0 0 1
30.0 0 1
31.0 1 1
37.0 0 1
38.0 0 1
47.0 1 1
49.0 0 1
50.0 0 1
63.0 0 1
79.0 0 1
80.0 1 1
82.0 1 1
82.0 1 1
86.0 0 1
98.0 0 1
149.0 1 1
202.0 0 1
219.0 0 1
10.0 0 2
10.0 0 2
12.0 0 2
13.0 0 2
14.0 0 2
15.0 0 2
16.0 0 2
17.0 0 2
18.0 0 2
20.0 0 2
24.0 0 2
24.0 0 2
25.0 0 2
26.0 0 2
### Program Results

nag_surviv_logrank (g12abc) Example Program Results

<table>
<thead>
<tr>
<th>Group</th>
<th>Observed</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.00</td>
<td>22.48</td>
</tr>
<tr>
<td>2</td>
<td>28.00</td>
<td>19.52</td>
</tr>
</tbody>
</table>

No. Unique Failure Times = 36

Test Statistic = 7.4966
Degrees of Freedom = 1
p-value = 0.0062