NAG Library Function Document

nag_prod_limit_surviv_fn (g12aac)

1 Purpose
nag_prod_limit_surviv_fn (g12aac) computes the Kaplan–Meier, (or product-limit), estimates of survival probabilities for a sample of failure times.

2 Specification
#include <nag.h>
#include <nagg12.h>
void nag_prod_limit_surviv_fn (Integer n, const double t[], const Integer ic[], const Integer freq[], Integer *nd, double tp[], double p[], double psig[], NagError *fail)

3 Description
A survivor function, \( S(t) \), is the probability of surviving to at least time \( t \) with \( S(t) = 1 - F(t) \), where \( F(t) \) is the cumulative distribution function of the failure times. The Kaplan–Meier or product limit estimator provides an estimate of \( S(t) \), \( \hat{S}(t) \), from sample of failure times which may be progressively right-censored.

Let \( t_i, i = 1, 2, \ldots, n_d \), be the ordered distinct failure times for the sample of observed failure/censored times, and let the number of observations in the sample that have not failed by time \( t_i \) be \( n_i \). If a failure and a loss (censored observation) occur at the same time \( t_i \), then the failure is treated as if it had occurred slightly before time \( t_i \) and the loss as if it had occurred slightly after \( t_i \).

The Kaplan–Meier estimate of the survival probabilities is a step function which in the interval \( t_i \) to \( t_{i+1} \) is given by

\[
\hat{S}(t) = \prod_{j=1}^{i} \left( \frac{n_j - d_j}{n_j} \right)
\]

where \( d_j \) is the number of failures occurring at time \( t_j \).

nag_prod_limit_surviv_fn (g12aac) computes the Kaplan–Meier estimates and the corresponding estimates of the variances, \( \text{vár}(\hat{S}(t)) \), using Greenwood’s formula,

\[
\text{vár}(\hat{S}(t)) = \hat{S}(t)^2 \sum_{j=1}^{i} \frac{d_j}{n_j(n_j - d_j)}.
\]

4 References

5 Arguments
1: n – Integer

\[ \text{Input} \]

On entry: the number of failure and censored times given in t.

Constraint: \( n \geq 2 \).

Mark 25

g12aac.1
2: \( \text{t}[n] \) – const double

*Input*

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

3: \( \text{ic}[n] \) – const Integer

*Input*

*On entry:* \( \text{ic}[i-1] \) contains the censoring code of the \( i \)th observation, for \( i = 1, 2, \ldots, n \).

- \( \text{ic}[i-1] = 0 \)
  The \( i \)th observation is a failure time.

- \( \text{ic}[i-1] = 1 \)
  The \( i \)th observation is right-censored.

*Constraint:* \( \text{ic}[i-1] = 0 \) or 1, for \( i = 1, 2, \ldots, n \).

4: \( \text{freq}[n] \) – const Integer

*Input*

*On entry:* indicates whether frequencies are provided for each failure and censored time point. If frequencies are provided then \( \text{freq} \) must be dimensioned at least \( n \). If the failure and censored times are to be considered as single observations, i.e., a frequency of 1 is to be assumed then \( \text{freq} \) must be set to NULL.

*Constraint:* either \( \text{freq} = (\text{Integer})*0 \) or \( \text{freq}[i-1] \geq 0 \), for \( i = 1, 2, \ldots, n \).

5: \( \text{nd} \) – Integer *

*Output*

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

6: \( \text{tp}[n] \) – double

*Output*

*On exit:* \( \text{tp}[i-1] \) contains the \( i \)th ordered distinct failure time, \( t_i \), for \( i = 1, 2, \ldots, n_d \).

7: \( \text{p}[n] \) – double

*Output*

*On exit:* \( \text{p}[i-1] \) contains the Kaplan–Meier estimate of the survival probability, \( \hat{S}(t) \), for time \( \text{tp}[i-1] \), for \( i = 1, 2, \ldots, n_d \).

8: \( \text{psig}[n] \) – double

*Output*

*On exit:* \( \text{psig}[i-1] \) contains an estimate of the standard deviation of \( p[i-1] \), for \( i = 1, 2, \ldots, n_d \).

9: \( \text{fail} \) – 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_INT_ARG_LT**

On entry, \( n = \langle \text{value} \rangle \).

*Constraint:* \( n \geq 2 \).

**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.

**NE_INVALID_CENSOR_CODE**

On entry, \( \text{ic} \langle \text{value} \rangle \) = \( \langle \text{value} \rangle \). The censor code for an observation must be either 0 or 1.
On entry, \( \text{freq}[(\text{value})] = (\text{value}) \). The value of frequency for an observation must be \( \geq 0 \).

## 7 Accuracy
The computations are believed to be stable.

## 8 Parallelism and Performance
Not applicable.

## 9 Further Comments
If there are no censored observations, \( \hat{S}(t) \), reduces to the ordinary binomial estimate of the probability of survival at time \( t \).

## 10 Example
The remission times for a set of 21 leukaemia patients at 18 distinct time points are read in and the Kaplan–Meier estimate computed and printed. For further details see page 242 of Gross and Clark (1975).

### 10.1 Program Text

```c
/* nag_prod_limit_surviv_fn (g12aac) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 4, 1996.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg12.h>
int main(void)
{
    Integer exit_status = 0, i, *ic = 0, *ifreq = 0, n, nd;
    NagError fail;
    double *p = 0, *psig = 0, *t = 0, *tp = 0;
    INIT_FAIL(fail);
    printf("nag_prod_limit_surviv_fn (g12aac) Example Program Results\n");
    /* 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 (n >= 2)
    {
        if (!((psig = NAG_ALLOC(n, double)) ||
```
p = NAG_ALLOC(n, double) ||
t = NAG_ALLOC(n, double) ||
(tp = NAG_ALLOC(n, double)) ||
(ifreq = NAG_ALLOC(n, Integer)) ||
(ic = NAG_ALLOC(n, Integer))
{
printf("Allocation failure\n");
exit_status = -1;
go to END;
}
else
{
printf("Invalid n.\n");
exit_status = 1;
return exit_status;
}
for (i = 0; i < n; ++i)
#else _WIN32
scanf_s("%lf %"NAG_IFMT" %"NAG_IFMT" ", &t[i], &ic[i], &ifreq[i]);
#else
scanf("%lf %"NAG_IFMT" %"NAG_IFMT" ", &t[i], &ic[i], &ifreq[i]);
#endif
/* nag_prod_limit_surviv_fn (g12aac).
 * Computes Kaplan-Meier (product-limit) estimates of
 * survival probabilities
 */
if (fail.code != NE_NOERROR)
{
printf("Error from nag_prod_limit_surviv_fn (g12aac).\n", fail.message);
exit_status = 1;
go to END;
}
printf("\n Time Survival Standard probability deviation
Time Survival Standard probability deviation

for (i = 0; i < nd; ++i)
printf("%6.1f%10.3f %10.3f
", tp[i], p[i], psig[i]);
END:
NAG_FREE(psig);
NAG_FREE(p);
NAG_FREE(t);
NAG_FREE(tp);
NAG_FREE(ifreq);
NAG_FREE(ic);
return exit_status;
}

10.2 Program Data

nag_prod_limit_surviv_fn (g12aac) Example Program Data
18
6.0 1 1 6.0 0 3 7.0 0 1 9.0 1 1 10.0 0 1 10.0 1 1
11.0 1 1 13.0 0 1 16.0 0 1 17.0 1 1 19.0 1 1 20.0 1 1
22.0 0 1 1 23.0 0 1 1 25.0 1 1 32.0 1 2 34.0 1 1 35.0 1 1

10.3 Program Results

nag_prod_limit_surviv_fn (g12aac) Example Program Results

<table>
<thead>
<tr>
<th>Time</th>
<th>Survival</th>
<th>Standard probability deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>0.857</td>
<td>0.076</td>
</tr>
<tr>
<td>7.0</td>
<td>0.807</td>
<td>0.087</td>
</tr>
<tr>
<td>Value</td>
<td>Column 1</td>
<td>Column 2</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>10.0</td>
<td>0.753</td>
<td>0.096</td>
</tr>
<tr>
<td>13.0</td>
<td>0.690</td>
<td>0.107</td>
</tr>
<tr>
<td>16.0</td>
<td>0.627</td>
<td>0.114</td>
</tr>
<tr>
<td>22.0</td>
<td>0.538</td>
<td>0.128</td>
</tr>
<tr>
<td>23.0</td>
<td>0.448</td>
<td>0.135</td>
</tr>
</tbody>
</table>