NAG Library Function Document

nag_surviv_risk_sets (g12zac)

1 Purpose

nag_surviv_risk_sets (g12zac) creates the risk sets associated with the Cox proportional hazards model for fixed covariates.

2 Specification

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

void nag_surviv_risk_sets (Nag_OrderType order, Integer n, Integer m,
     Integer ns, const double z[], Integer pdz, const Integer isz[],
     Integer ip, const double t[], const Integer ic[], const Integer isi[],
     Integer *num, Integer ixs[], Integer *nxs, double x[], Integer mxn,
     Integer id[], Integer *nd, double tp[], Integer irs[], NagError *fail)
```

3 Description

The Cox proportional hazards model (see Cox (1972)) relates the time to an event, usually death or failure, to a number of explanatory variables known as covariates. Some of the observations may be right-censored, that is, the exact time to failure is not known, only that it is greater than a known time.

Let t_i, for $i=1,2,\ldots,n$, be the failure time or censored time for the ith observation with the vector of p covariates z_i. The covariance matrix Z is constructed so that it contains n rows with the ith row containing the p covariates z_i. It is assumed that censoring and failure mechanisms are independent. The hazard function, $\lambda(t,z)$, is the probability that an individual with covariates z fails at time t given that the individual survived up to time t. In the Cox proportional hazards model, $\lambda(t,z)$ is of the form

$$\lambda(t,z)=\lambda_0(t)\exp\left(z^{\mathrm{T}}\beta\right),$$

where λ_0 is the base-line hazard function, an unspecified function of time, and β is a vector of unknown arguments. As λ_0 is unknown, the arguments β are estimated using the conditional or marginal likelihood. This involves considering the covariate values of all subjects that are at risk at the time when a failure occurs. The probability that the subject that failed had their observed set of covariate values is computed.

The risk set at a failure time consists of those subjects that fail or are censored at that time and those who survive beyond that time. As risk sets are computed for every distinct failure time, it should be noted that the combined risk sets may be considerably larger than the original data. If the data can be considered as coming from different strata such that λ_0 varies from strata to strata but β remains constant, then nag_surviv_risk_sets (g12zac) will return a factor that indicates to which risk set/strata each member of the risk sets belongs rather than just to which risk set.

Given the risk sets the Cox proportional hazards model can then be fitted using a Poisson generalized linear model (nag_glm_poisson (g02gcc) with nag_dummy_vars (g04eac) to compute dummy variables) using Breslow's approximation for ties (see Breslow (1974)). This will give the same fit as nag_surviv_cox_model (g12bac). If the exact treatment of ties in discrete time is required, as given by Cox (1972), then the model is fitted as a conditional logistic model using nag_condl_logistic (g11cac).

4 References

Breslow N E (1974) Covariate analysis of censored survival data *Biometrics* **30** 89–99

Cox D R (1972) Regression models in life tables (with discussion) *J. Roy. Statist. Soc. Ser. B* **34** 187–220

Gross A J and Clark V A (1975) *Survival Distributions: Reliability Applications in the Biomedical Sciences* Wiley

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **n** – Integer *Input*

On entry: n, the number of data points.

Constraint: **n** ≥ 2.

3: **m** – Integer *Input*

On entry: the number of covariates in array **z**.

Constraint: **m** ≥ 1.

4: **ns** – Integer *Input*

On entry: the number of strata. If **ns** > 0 then the stratum for each observation must be supplied in **isi**.

Constraint: **ns** ≥ 0.

5: **z**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **z** must be at least

$\max(1, \mathbf{pdz} \times \mathbf{m})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{n} \times \mathbf{pdz})$ when **order** = Nag_RowMajor.

The (i, j)th element of the matrix Z is stored in

$\mathbf{z}[(j-1) \times \mathbf{pdz} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{z}[(i-1) \times \mathbf{pdz} + j - 1]$ when **order** = Nag_RowMajor.

On entry: must contain the n covariates in column or row major order.

6: **pdz** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **z**.

Constraints:

if **order** = Nag_ColMajor, **pdz** $\geq$ **n**;
if **order** = Nag_RowMajor, **pdz** $\geq$ **m**.

7: **isz**[**m**] – const Integer *Input*

On entry: indicates which subset of covariates are to be included in the model.

isz$[j-1] \geq 1$
The jth covariate is included in the model.

isz$[j-1] = 0$
The jth covariate is excluded from the model and not referenced.

Constraint: **isz**$[j-1] \geq 0$ and at least one value must be nonzero.

8: **ip** – Integer *Input*

On entry: p, the number of covariates included in the model as indicated by **isz**.

Constraint: **ip** = the number of nonzero values of **isz**.

9: **t**[**n**] – const double *Input*

On entry: the vector of n failure censoring times.

10: **ic**[**n**] – const Integer *Input*

On entry: the status of the individual at time t given in **t**.

ic$[i-1] = 0$
Indicates that the ith individual has failed at time **t**$[i-1]$.

ic$[i-1] = 1$
Indicates that the ith individual has been censored at time **t**$[i-1]$.

Constraint: **ic**$[i-1] = 0$ or 1, for $i = 1, 2, \ldots,$ **n**.

11: **isi**[*dim*] – const Integer *Input*

Note: the dimension, *dim*, of the array **isi** must be at least

n when **ns** > 0;
1 otherwise.

On entry: if **ns** > 0, the stratum indicators which also allow data points to be excluded from the analysis.

If **ns** $= 0$, **isi** is not referenced.

isi$[i] = k$
Indicates that the ith data point is in the kth stratum, where $k = 1, 2, \ldots,$ **ns**.

isi$[i] = 0$
Indicates that the ith data point is omitted from the analysis.

Constraint: if **ns** > 0, $0 \leq$ **isi**$[i] \leq$ **ns**, for $i = 0, 1, \ldots,$ **n** $- 1$.

12: **num** – Integer * *Output*

On exit: the number of values in the combined risk sets.

13: **ixs**[**mxn**] – Integer *Output*

On exit: the factor giving the risk sets/strata for the data in **x** and **id**.

If **ns** $= 0$ or 1, **ixs**$[i-1] = l$ for members of the lth risk set.

If **ns** > 1, **ixs**$[i-1] = (j-1) \times$ **nd** $+ l$ for the observations in the lth risk set for the jth strata.

14: **nxs** – Integer * *Output*

On exit: the number of levels for the risk sets/strata factor given in **ixs**.

15: **x**[**mxn** × **ip**] – double *Output*

Note: the (i, j)th element of the matrix X is stored in

> **x**$[(j-1) \times$ **mxn** $+ i - 1]$ when **order** = Nag_ColMajor;
> **x**$[(i-1) \times$ **ip** $+ j - 1]$ when **order** = Nag_RowMajor.

On exit: the first **num** rows contain the values of the covariates for the members of the risk sets.

16: **mxn** – Integer *Input*

On entry: the first dimension of the array **x** and the dimension of the arrays **ixs** and **id**.

Constraint: **mxn** must be sufficiently large for the arrays to contain the expanded risk sets. The size will depend on the pattern of failures times and censored times. The minimum value will be returned in **num** unless the function exits with **fail.code** = NE_INT.

17: **id**[**mxn**] – Integer *Output*

On exit: indicates if the member of the risk set given in **x** failed.

id$[i-1] = 1$ if the member of the risk set failed at the time defining the risk set and **id**$[i-1] = 0$ otherwise.

18: **nd** – Integer * *Output*

On exit: the number of distinct failure times, i.e., the number of risk sets.

19: **tp**[**n**] – double *Output*

On exit: **tp**$[i-1]$ contains the ith distinct failure time, for $i = 1, 2, \ldots,$ **nd**.

20: **irs**[**n**] – Integer *Output*

On exit: indicates rows in **x** and elements in **ixs** and **id** corresponding to the risk sets. The first risk set corresponding to failure time **tp**$[0]$ is given by rows 1 to **irs**$[0]$. The lth risk set is given by rows **irs**$[l-2]+1$ to **irs**$[l-1]$, for $l = 1, 2, \ldots,$ **nd**.

21: **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.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, element ⟨*value*⟩ of **ic** is not equal to 0 or 1.

On entry, element ⟨*value*⟩ of **isi** is not valid.

On entry, element ⟨*value*⟩ of **isz** < 0.

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 1.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 2.

On entry, **ns** = ⟨*value*⟩.
Constraint: **ns** ≥ 0.

On entry, **pdz** = ⟨*value*⟩.
Constraint: **pdz** > 0.

NE_INT_2

On entry, **pdz** = ⟨*value*⟩ and **m** = ⟨*value*⟩.
Constraint: **pdz** $\geq$ **m**.

NE_INT_ARRAY_ELEM_CONS

mxn is too small: min value = ⟨*value*⟩.

On entry, there are not **ip** values of **isz** > 0.

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.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

When there are strata present, i.e., **ns** > 1, not all the **nxs** groups may be present.

10 Example

The data are the remission times for two groups of leukemia patients (see page 242 of Gross and Clark (1975)). A dummy variable indicates which group they come from. The risk sets are computed using nag_surviv_risk_sets (g12zac) and the Cox's proportional hazard model is fitted using nag_condl_logistic (g11cac).

10.1 Program Text

```
/* nag_surviv_risk_sets (g12zac) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 * Mark 7b revised, 2004.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg11.h>
```

```
#include <nagg12.h>

int main(void)
{
  /* Scalars */
  double        dev, tol;
  Integer       exit_status, i, ip, iprint, j, lisi, m,
                maxit, mxn, n, nd, ns, num, nxs, pdx, pdz;
  NagError      fail;
  Nag_OrderType order;

  /* Arrays */
  double        *b = 0, *cov = 0, *sc = 0, *se = 0, *t = 0, *tp = 0,
  *x = 0, *z = 0;
  Integer       *ic = 0, *id = 0, *irs = 0, *isi = 0, *isz = 0, *ixs = 0,
  *nca = 0, *nct = 0;

#ifdef NAG_COLUMN_MAJOR
#define Z(I, J) z[(J-1)*pdz + I - 1]
#define X(I, J) x[(J-1)*pdx + I - 1]
  order = Nag_ColMajor;
#else
#define Z(I, J) z[(I-1)*pdz + J - 1]
#define X(I, J) x[(I-1)*pdx + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  exit_status = 0;

  printf("nag_surviv_risk_sets (g12zac) Example Program Results\n");

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

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ",
          &n, &m, &ns, &maxit, &iprint);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ",
          &n, &m, &ns, &maxit, &iprint);
#endif

  /* Allocate arrays t, z, ic and isi */
  if (ns > 0)
    lisi = n;
  else
    lisi = 1;
  if (!(t = NAG_ALLOC(n, double)) ||
      !(z = NAG_ALLOC(n * n, double)) ||
      !(ic = NAG_ALLOC(n, Integer)) ||
      !(isi = NAG_ALLOC(lisi, Integer)) ||
      !(isz = NAG_ALLOC(m, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  if (order == Nag_ColMajor)
    {
      pdz = n;
    }
  else
    {
      pdz = m;
```

```
    }

  if (ns > 0)
    {
      for (i = 1; i <= n; ++i)
        {
#ifdef _WIN32
          scanf_s("%lf", &t[i-1]);
#else
          scanf("%lf", &t[i-1]);
#endif
          for (j = 1; j <= m; ++j)
#ifdef _WIN32
            scanf_s("%lf", &Z(i, j));
#else
            scanf("%lf", &Z(i, j));
#endif
#ifdef _WIN32
          scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &ic[i-1], &isi[i-1]);
#else
          scanf("%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &ic[i-1], &isi[i-1]);
#endif
        }
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
#ifdef _WIN32
          scanf_s("%lf", &t[i-1]);
#else
          scanf("%lf", &t[i-1]);
#endif
          for (j = 1; j <= m; ++j)
#ifdef _WIN32
            scanf_s("%lf", &Z(i, j));
#else
            scanf("%lf", &Z(i, j));
#endif
#ifdef _WIN32
          scanf_s("%"NAG_IFMT"%*[^\n] ", &ic[i-1]);
#else
          scanf("%"NAG_IFMT"%*[^\n] ", &ic[i-1]);
#endif
        }
    }

  for (i = 1; i <= m; ++i)
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"", &isz[i-1]);
#else
    scanf("%"NAG_IFMT"", &isz[i-1]);
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ", &ip);
#else
  scanf("%"NAG_IFMT"%*[^\n] ", &ip);
#endif


  /* Allocate other arrays for nag_surviv_risk_sets (g12zac) */
  mxn = 1000;

  if (order == Nag_ColMajor)
    {
      pdx = mxn;
    }
  else
    {
      pdx = ip;
    }
```

```
  if (!(cov = NAG_ALLOC(ip*(ip+1)/2, double)) ||
      !(sc = NAG_ALLOC(ip, double)) ||
      !(se = NAG_ALLOC(ip, double)) ||
      !(tp = NAG_ALLOC(n, double)) ||
      !(x = NAG_ALLOC(mxn * ip, double)) ||
      !(id = NAG_ALLOC(mxn, Integer)) ||
      !(irs = NAG_ALLOC(n, Integer)) ||
      !(ixs = NAG_ALLOC(mxn, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* nag_surviv_risk_sets (g12zac).
   * Creates the risk sets associated with the Cox
   * proportional hazards model for fixed covariates
   */
  nag_surviv_risk_sets(order, n, m, ns, z, pdz, isz, ip, t, ic, isi, &num, ixs,
                       &nxs, x, mxn, id, &nd, tp, irs, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_surviv_risk_sets (g12zac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Allocate arrays for nag_condl_logistic (g11cac) */
  if (!(b = NAG_ALLOC(ip, double)) ||
      !(nca = NAG_ALLOC(nxs, Integer)) ||
      !(nct = NAG_ALLOC(nxs, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

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

  tol = 1e-5;
  /* nag_condl_logistic (g11cac).
   * Returns parameter estimates for the conditional analysis
   * of stratified data
   */
  nag_condl_logistic(order, num, ip, nxs, x, pdx, isz, ip, id, ixs, &dev, b,
                     se, sc, cov, nca, nct, tol, maxit, iprint, 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_condl_logistic (g11cac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("\n");
  printf(" Parameter      Estimate       Standard Error\n");
  printf("\n");
  for (i = 1; i <= ip; ++i)
    printf("%5"NAG_IFMT"          %8.4f          %8.4f          \n",
```

```
            i, b[i-1], se[i-1]);

 END:
  NAG_FREE(b);
  NAG_FREE(cov);
  NAG_FREE(sc);
  NAG_FREE(se);
  NAG_FREE(t);
  NAG_FREE(tp);
  NAG_FREE(x);
  NAG_FREE(z);
  NAG_FREE(ic);
  NAG_FREE(id);
  NAG_FREE(irs);
  NAG_FREE(isi);
  NAG_FREE(isz);
  NAG_FREE(ixs);
  NAG_FREE(nca);
  NAG_FREE(nct);

  return exit_status;
}
```

10.2 Program Data

```
nag_surviv_risk_sets (g12zac) Example Program Data

42 1 0 20 0

 1 0 0
 1 0 0
 2 0 0
 2 0 0
 3 0 0
 4 0 0
 4 0 0
 5 0 0
 5 0 0
 8 0 0
 8 0 0
 8 0 0
 8 0 0
11 0 0
11 0 0
12 0 0
12 0 0
15 0 0
17 0 0
22 0 0
23 0 0
 6 1 0
 6 1 0
 6 1 0
 7 1 0
10 1 0
13 1 0
16 1 0
22 1 0
23 1 0
 6 1 1
 9 1 1
10 1 1
11 1 1
17 1 1
19 1 1
20 1 1
25 1 1
32 1 1
32 1 1
34 1 1
```

```
35 1 1

   1    1

0.0 0.0
```

10.3 Program Results

```
nag_surviv_risk_sets (g12zac) Example Program Results

 Parameter      Estimate       Standard Error

    1             1.6282            0.4331
```