NAG Library Function Document

nag_condl_logistic (g11cac)

1 Purpose

nag_condl_logistic (g11cac) returns parameter estimates for the conditional logistic analysis of stratified data, for example, data from case-control studies and survival analyses.

2 Specification

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

void nag_condl_logistic (Nag_OrderType order, Integer n, Integer m,
     Integer ns, const double z[], Integer pdz, const Integer isz[],
     Integer p, const Integer ic[], const Integer isi[], double *dev,
     double b[], double se[], double sc[], double cov[], Integer nca[],
     Integer nct[], double tol, Integer maxit, Integer iprint,
     const char *outfile, NagError *fail)
```

3 Description

In the analysis of binary data, the logistic model is commonly used. This relates the probability of one of the outcomes, say $y = 1$, to p explanatory variates or covariates by

$$\text{Prob}(y = 1) = \frac{\exp(\alpha + z^{\mathrm{T}}\beta)}{1 + \exp(\alpha + z^{\mathrm{T}}\beta)},$$

where β is a vector of unknown coefficients for the covariates z and α is a constant term. If the observations come from different strata or groups, α would vary from strata to strata. If the observed outcomes are independent then the ys follow a Bernoulli distribution, i.e., a binomial distribution with sample size one and the model can be fitted as a generalized linear model with binomial errors.

In some situations the number of observations for which $y = 1$ may not be independent. For example, in epidemiological research, case-control studies are widely used in which one or more observed cases are matched with one or more controls. The matching is based on fixed characteristics such as age and sex, and is designed to eliminate the effect of such characteristics in order to more accurately determine the effect of other variables. Each case-control group can be considered as a stratum. In this type of study the binomial model is not appropriate, except if the strata are large, and a conditional logistic model is used. This considers the probability of the cases having the observed vectors of covariates given the set of vectors of covariates in the strata. In the situation of one case per stratum, the conditional likelihood for n_{s} strata can be written as

$$L = \prod_{i=1}^{n_s} \frac{\exp\left(z_i^{\mathrm{T}}\beta\right)}{\left[\sum_{l \in S_i} \exp\left(z_l^{\mathrm{T}}\beta\right)\right]}, \tag{1}$$

where S_i is the set of observations in the ith stratum, with associated vectors of covariates z_l, $l \in S_i$, and z_i is the vector of covariates of the case in the ith stratum. In the general case of c_i cases per strata then the full conditional likelihood is

$$L = \prod_{i=1}^{n_s} \frac{\exp\left(s_i^{\mathrm{T}}\beta\right)}{\left[\sum_{l \in C_i} \exp\left(s_l^{\mathrm{T}}\beta\right)\right]}, \tag{2}$$

where s_i is the sum of the vectors of covariates for the cases in the ith stratum and s_l, $l \in C_i$ refer to the sum of vectors of covariates for all distinct sets of c_i observations drawn from the ith stratum. The conditional likelihood can be maximized by a Newton–Raphson procedure. The covariances of the parameter estimates can be estimated from the inverse of the matrix of second derivatives of the logarithm of the conditional likelihood, while the first derivatives provide the score function, $U_j(\beta)$, for $j = 1, 2, \ldots, p$, which can be used for testing the significance of arguments.

If the strata are not small, C_i can be large so to improve the speed of computation, the algorithm in Howard (1972) and described by Krailo and Pike (1984) is used.

A second situation in which the above conditional likelihood arises is in fitting Cox's proportional hazard model (see nag_surviv_cox_model (g12bac)) in which the strata refer to the risk sets for each failure time and where the failures are cases. When ties are present in the data nag_surviv_cox_model (g12bac) uses an approximation. For an exact estimate, the data can be expanded using nag_surviv_risk_sets (g12zac) to create the risk sets/strata and nag_condl_logistic (g11cac) used.

4 References

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

Cox D R and Hinkley D V (1974) *Theoretical Statistics* Chapman and Hall

Howard S (1972) Remark on the paper by Cox, D R (1972): Regression methods *J. R. Statist. Soc.* **B 34** and life tables 187–220

Krailo M D and Pike M C (1984) Algorithm AS 196. Conditional multivariate logistic analysis of stratified case-control studies *Appl. Statist.* **33** 95–103

Smith P G, Pike M C, Hill P, Breslow N E and Day N E (1981) Algorithm AS 162. Multivariate conditional logistic analysis of stratum-matched case-control studies *Appl. Statist.* **30** 190–197

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 observations.

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, n_s.

Constraint: **ns** ≥ 1.

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

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

max(1, **pdz** × **m**) when **order** = Nag_ColMajor;
max(1, **n** × **pdz**) when **order** = Nag_RowMajor.

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

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

On entry: the ith row must contain the covariates which are associated with the ith observation.

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.

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

If **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: **p** – Integer *Input*

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

Constraint: **p** ≥ 1 and **p** = number of nonzero values of **isz** .

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

On entry: indicates whether the ith observation is a case or a control.

If **ic**$[i-1] = 0$, indicates that the ith observation is a case.

If **ic**$[i-1] = 1$, indicates that the ith observation is a control.

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

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

On entry: stratum indicators which also allow data points to be excluded from the analysis.

If **isi**$[i-1] = k$, indicates that the ith observation is from the kth stratum, where $k = 1, 2, \ldots,$ **ns**.

If **isi**$[i-1] = 0$, indicates that the ith observation is to be omitted from the analysis.

Constraint: $0 \leq$ **isi**$[i-1] \leq$ **ns** and more than **p** values of **isi**$[i-1] > 0$, for $i = 1, 2, \ldots,$ **n**.

11: **dev** – double * *Output*

On exit: the deviance, that is, $-2 \times$, (maximized log marginal likelihood).

12: **b**[**p**] – double *Input/Output*

On entry: initial estimates of the covariate coefficient arguments β. **b**$[j-1]$ must contain the initial estimate of the coefficent of the covariate in **z** corresponding to the jth nonzero value of **isz**.

Suggested value: in many cases an initial value of zero for **b**$[j-1]$ may be used. For another suggestion see Section 9.

On exit: **b**$[j-1]$ contains the estimate $\hat{\beta}_i$ of the coefficient of the covariate stored in the ith column of **z** where i is the jth nonzero value in the array **isz**.

13: **se**[**p**] – double *Output*

On exit: **se**$[j-1]$ is the asymptotic standard error of the estimate contained in **b**$[j-1]$ and score function in **sc**$[j-1]$, for $j = 1, 2, \ldots,$ **p**.

14: **sc**[**p**] – double *Output*

On exit: **sc**$[j]$ is the value of the score function $U_j(\beta)$ for the estimate contained in **b**$[j-1]$.

15: **cov**[$\mathbf{p} \times (\mathbf{p}+\mathbf{1})/\mathbf{2}$] – double *Output*

On exit: the variance-covariance matrix of the parameter estimates in **b** stored in packed form by column, i.e., the covariance between the parameter estimates given in **b**$[i-1]$ and **b**$[j-1]$, $j \geq i$, is given in **cov**$[j(j-1)/2+i]$.

16: **nca**[**ns**] – Integer *Output*

On exit: **nca**$[i-1]$ contains the number of cases in the ith stratum, for $i=1,2,\ldots,$**ns**.

17: **nct**[**ns**] – Integer *Output*

On exit: **nct**$[i-1]$ contains the number of controls in the ith stratum, for $i=1,2,\ldots,$**ns**.

18: **tol** – double *Input*

On entry: indicates the accuracy required for the estimation. Convergence is assumed when the decrease in deviance is less than **tol** $\times$ $(1.0+\text{CurrentDeviance})$. This corresponds approximately to an absolute accuracy if the deviance is small and a relative accuracy if the deviance is large.

Constraint: **tol** $\geq 10 \times$ ***machine precision***.

19: **maxit** – Integer *Input*

On entry: the maximum number of iterations required for computing the estimates. If **maxit** is set to 0 then the standard errors, the score functions and the variance-covariance matrix are computed for the input value of β in **b** but β is not updated.

Constraint: **maxit** ≥ 0.

20: **iprint** – Integer *Input*

On entry: indicates if the printing of information on the iterations is required.

iprint ≤ 0
No printing.

iprint ≥ 1
The deviance and the current estimates are printed every **iprint** iterations.

Suggested value: **iprint** $=0$.

21: **outfile** – const char * *Input*

On entry: the name of a file to which diagnostic output will be directed. If **outfile** is **NULL** the diagnostic output will be directed to standard output.

22: **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 ⟨*value*⟩ had an illegal value.

NE_CONVERGENCE

Convergence not achieved in $\langle$*value*$\rangle$ iterations.

NE_INT

On entry, element $\langle$*value*$\rangle$ of **isz** < 0.

On entry, **ic**[$\langle$*value*$\rangle$] $=$ $\langle$*value*$\rangle$.
Constraint: **ic**[i] $= 0$ or 1, for all i.

On entry, **m** $=$ $\langle$*value*$\rangle$.
Constraint: **m** ≥ 1.

On entry, **maxit** $=$ $\langle$*value*$\rangle$.
Constraint: **maxit** ≥ 0.

On entry, **n** $=$ $\langle$*value*$\rangle$.
Constraint: **n** ≥ 2.

On entry, **ns** $=$ $\langle$*value*$\rangle$.
Constraint: **ns** ≥ 1.

On entry, **p** $=$ $\langle$*value*$\rangle$.
Constraint: **p** ≥ 1.

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

On entry, **pdz** $=$ $\langle$*value*$\rangle$.
Constraint: **pdz** > 0.

NE_INT_2

On entry, **isi**[$\langle$*value*$\rangle$] $=$ $\langle$*value*$\rangle$ and **ns** $=$ $\langle$*value*$\rangle$.
Constraint: $0 \leq$ **isi**[$i - 1$] $\leq$ **ns**.

On entry, **pdz** $=$ $\langle$*value*$\rangle$ and **m** $=$ $\langle$*value*$\rangle$.
Constraint: **pdz** $\geq$ **m**.

NE_INT_ARRAY_ELEM_CONS

On entry, there are not **p** 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.

NE_NOT_CLOSE_FILE

Cannot close file $\langle$*value*$\rangle$.

NE_NOT_WRITE_FILE

Cannot open file $\langle$*value*$\rangle$ for writing.

NE_OBSERVATIONS

On entry, too few observations included in model.

NE_OVERFLOW

Overflow in calculations.

NE_REAL

On entry, **tol** $=$ $\langle$*value*$\rangle$.
Constraint: **tol** $\geq 10 \times$ ***machine precision***.

NE_SINGULAR

The matrix of second partial derivatives is singular; computation abandoned.

7 Accuracy

The accuracy is specified by **tol**.

8 Parallelism and Performance

nag_condl_logistic (g11cac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_condl_logistic (g11cac) 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 Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The other models described in Section 3 can be fitted using the generalized linear modelling functions nag_glm_binomial (g02gbc) and nag_glm_poisson (g02gcc).

The case with one case per stratum can be analysed by having a dummy response variable y such that $y = 1$ for a case and $y = 0$ for a control, and fitting a Poisson generalized linear model with a log link and including a factor with a level for each strata. These models can be fitted by using nag_glm_poisson (g02gcc).

nag_condl_logistic (g11cac) uses mean centering, which involves subtracting the means from the covariables prior to computation of any statistics. This helps to minimize the effect of outlying observations and accelerates convergence. In order to reduce the risk of the sums computed by Howard's algorithm becoming too large, the scaling factor described in Krailo and Pike (1984) is used.

If the initial estimates are poor then there may be a problem with overflow in calculating $\exp(\beta^{\mathrm{T}} z_i)$ or there may be non-convergence. Reasonable estimates can often be obtained by fitting an unconditional model.

10 Example

The data was used for illustrative purposes by Smith *et al.* (1981) and consists of two strata and two covariates. The data is input, the model is fitted and the results are printed.

10.1 Program Text

```
/* nag_condl_logistic (g11cac) Example Program.
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

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

int main(void)
{
  /* Scalars */
  double        dev, tol;
  Integer       iprint, exit_status, i, p, j, m, maxit, n, ns, pdz;
  NagError      fail;
  Nag_OrderType order;

  /* Arrays */
  double        *b = 0, *cov = 0, *sc = 0, *se = 0, *z = 0;
  Integer       *ic = 0, *isi = 0, *isz = 0, *nca = 0, *nct = 0;

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

  INIT_FAIL(fail);

  exit_status = 0;

  printf("nag_condl_logistic (g11cac) Example Program Results\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%ld%ld%ld%*[^\n] ", &n, &m,
         &ns, &maxit);

  pdz = n;
  iprint = 0;
  tol = 1e-5;

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

  if (order == Nag_ColMajor)
    pdz = n;
  else
    pdz = m;

  for (i = 1; i <= n; ++i)
    {
      scanf("%ld%ld", &isi[i-1], &ic[i-1]);
      for (j = 1; j <= m; ++j)
        scanf("%lf", &Z(i, j));
      scanf("%*[^\n] ");
    }

  for (j = 1; j <= m; ++j)
    scanf("%ld", &isz[j-1]);
  scanf("%ld%*[^\n] ", &p);

  /* Allocate other arrays */
  if (!(b = NAG_ALLOC(p, double)) ||
      !(cov = NAG_ALLOC(p*(p+1)/2, double)) ||
      !(sc = NAG_ALLOC(p, double)) ||
      !(se = NAG_ALLOC(p, double)) ||
      !(nca = NAG_ALLOC(ns, Integer)) ||
      !(nct = NAG_ALLOC(ns, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (j = 1; j <= m; ++j)
```

```
    scanf("%lf", &b[j-1]);
  scanf("%*[^\n] ");

  /* nag_condl_logistic (g11cac).
   * Returns parameter estimates for the conditional analysis
   * of stratified data
   */
  nag_condl_logistic(order, n, m, ns, z, pdz, isz, p, ic, isi, &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(" Deviance = %13.4e\n", dev);
  printf("\n");

  printf(" Strata     No. Cases   No. Controls\n");
  printf("\n");
  for (i = 1; i <= ns; ++i)
    printf("   %3ld          %2ld           %2ld"
            "           \n", i, nca[i-1], nct[i-1]);
  printf("\n");

  printf(" Parameter      Estimate       Standard Error\n");
  printf("\n");
  for (i = 1; i <= p; ++i)
    printf("%5ld          %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(z);
  NAG_FREE(ic);
  NAG_FREE(isi);
  NAG_FREE(isz);
  NAG_FREE(nca);
  NAG_FREE(nct);

  return exit_status;
}
```

10.2 Program Data

```
nag_condl_logistic (g11cac) Example Program Data

7 2 2 10

1 0 0 1
1 0 1 2
1 1 0 1
1 1 1 3
2 0 0 1
2 1 1 0
2 1 0 2

    1 1 2

0.0 0.0
```

10.3 Program Results

```
nag_condl_logistic (g11cac) Example Program Results

 Deviance =    5.4749e+00

 Strata     No. Cases   No. Controls

     1           2           2
     2           1           2

 Parameter      Estimate       Standard Error

    1          -0.5223            1.3901
    2          -0.2674            0.8473
```