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

```c
#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

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

where $\beta$ is a vector of unknown coefficients for the covariates $z$ and $\alpha$ is a constant term. If the observations come from different strata or groups, $\alpha$ would vary from strata to strata. If the observed outcomes are independent then the $y$s 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

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

where $S_i$ is the set of observations in the $i$th 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 $i$th stratum. In the general case of $c_i$ cases per strata then the full conditional likelihood is

$$
L = \prod_{i=1}^{n_s} \frac{\exp(s_i^T \beta)}{\sum_{l \in C_i} \exp(s_l^T \beta)},
$$

where $s_i$ is the sum of the vectors of covariates for the cases in the $i$th 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 $i$th 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


5 Arguments

1: order – Nag_OrderType

\( \text{On entry:} \) 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

\( \text{On entry:} n \), the number of observations.

Constraint: \( n \geq 2 \).

3: \( m \) – Integer

\( \text{On entry:} \) the number of covariates in array \( z \).

Constraint: \( m \geq 1 \).

4: \( ns \) – Integer

\( \text{On entry:} \) the number of strata, \( n_s \).

Constraint: \( ns \geq 1 \).

5: \( z[\text{dim}] \) – const double

Note: the dimension, \( \text{dim} \), of the array \( z \) must be at least

\[ \max(1, pdz \times m) \text{ when } order = \text{Nag_ColMajor}; \]
\[ \max(1, n \times pdz) \text{ when } order = \text{Nag_RowMajor}. \]
The \((i,j)\)th element of the matrix \(Z\) is stored in
\[
Z[(j - 1) \times \text{pdz} + i - 1] \quad \text{when order = Nag_ColMajor;}
\]
\[
Z[(i - 1) \times \text{pdz} + j - 1] \quad \text{when order = Nag_RowMajor.}
\]

On entry: the \(i\)th row must contain the covariates which are associated with the \(i\)th observation.

6: \text{pdz} – Integer  
\text{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, \text{pdz} \geq n;
if order = Nag_RowMajor, \text{pdz} \geq m.

7: \text{isz[m]} – const Integer  
\text{Input}

On entry: indicates which subset of covariates are to be included in the model.
If \(\text{isz}[j - 1] \geq 1\), the \(j\)th covariate is included in the model.
If \(\text{isz}[j - 1] = 0\), the \(j\)th covariate is excluded from the model and not referenced.
Constraint: \(\text{isz}[j - 1] \geq 0\) and at least one value must be nonzero.

8: \(p\) – Integer  
\text{Input}

On entry: \(p\), the number of covariates included in the model as indicated by isz.
Constraint: \(p \geq 1\) and \(p =\) number of nonzero values of isz.

9: \text{ic[n]} – const Integer  
\text{Input}

On entry: indicates whether the \(i\)th observation is a case or a control.
If \(\text{ic}[i - 1] = 0\), indicates that the \(i\)th observation is a case.
If \(\text{ic}[i - 1] = 1\), indicates that the \(i\)th observation is a control.
Constraint: \(\text{ic}[i - 1] = 0\) or \(1\), for \(i = 1, 2, \ldots, n\).

10: \text{isi[n]} – const Integer  
\text{Input}

On entry: stratum indicators which also allow data points to be excluded from the analysis.
If \(\text{isi}[i - 1] = k\), indicates that the \(i\)th observation is from the \(k\)th stratum, where \(k = 1, 2, \ldots, ns\).
If \(\text{isi}[i - 1] = 0\), indicates that the \(i\)th observation is to be omitted from the analysis.
Constraint: \(0 \leq \text{isi}[i - 1] \leq \text{ns}\) and more than \(p\) values of \(\text{isi}[i - 1] > 0\), for \(i = 1, 2, \ldots, n\).

11: \text{dev} – double *  
\text{Output}

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

12: \text{b[p]} – double  
\text{Input/Output}

On entry: initial estimates of the covariate coefficient arguments \(\beta\). \(\text{b}[j - 1]\) must contain the initial estimate of the coefficient of the covariate in \(Z\) corresponding to the \(j\)th nonzero value of isz.

Suggested value: in many cases an initial value of zero for \(\text{b}[j - 1]\) may be used. For another suggestion see Section 9.

On exit: \(\text{b}[j - 1]\) contains the estimate \(\hat{\beta}_j\) of the coefficient of the covariate stored in the \(i\)th column of \(Z\) where \(i\) is the \(j\)th 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[p \times (p + 1)/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 $i$th stratum, for `i = 1, 2, \ldots, ns`.

17: `nct[ns]` – Integer

*Output*

On exit: `nct[i - 1]` contains the number of controls in the $i$th 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

$\frac{\text{CurrentDeviance}}{\text{tol} \times (1.0 + \text{CurrentDeviance})}$

This corresponds approximately to an absolute accuracy if the deviance is small and a relative accuracy if the deviance is large.

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

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 $\beta$ in `b` but $\beta$ is not updated.

*Constraint: $\text{maxit} \geq 0.$*

20: `iprint` – Integer

*Input*

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

`iprint \leq 0`

No printing.

`iprint \geq 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.
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_CONVERGENCE
Convergence not achieved in ⟨value⟩ iterations.

NE_INT
On entry, element ⟨value⟩ of isz < 0.
On entry, ic[⟨value⟩] = ⟨value⟩.
Constraint: ic[i] = 0 or 1, for all i.
On entry, m = ⟨value⟩.
Constraint: m ≥ 1.
On entry, maxit = ⟨value⟩.
Constraint: maxit ≥ 0.
On entry, n = ⟨value⟩.
Constraint: n ≥ 2.
On entry, ns = ⟨value⟩.
Constraint: ns ≥ 1.
On entry, p = ⟨value⟩.
Constraint: p ≥ 1.
On entry, pdz < n: pdz = ⟨value⟩.
On entry, pdz = ⟨value⟩.
Constraint: pdz > 0.

NE_INT_2
On entry, isi[⟨value⟩] = ⟨value⟩ and ns = ⟨value⟩.
Constraint: 0 ≤ isi[i - 1] ≤ ns.
On entry, pdz = ⟨value⟩ and m = ⟨value⟩.
Constraint: pdz ≥ 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.
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.
NE_NOT_CLOSE_FILE
Cannot close file \(\text{value}\).

NE_NOT_WRITE_FILE
Cannot open file \(\text{value}\) for writing.

NE_OBSERVATIONS
On entry, too few observations included in model.

NE_OVERFLOW
Overflow in calculations.

NE_REAL
On entry, \(\text{tol} = \text{value}\).
Constraint: \(\text{tol} \geq 10 \times \text{machine precision}\).

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

7 Accuracy
The accuracy is specified by \(\text{tol}\).

8 Parallelism and Performance
\text{nag\_condl\_logistic (g11cac)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
\text{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 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 other models described in Section 3 can be fitted using the generalized linear modelling functions \text{nag\_glm\_binomial (g02gbc)} and \text{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 \text{nag\_glm\_poisson (g02gcc)}.

\text{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 \(\exp(\beta^Tz_i)\) 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 2014 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;

    INIT_FAIL(fail);
    exit_status = 0;
    printf("nag_condl_logistic (g11cac) Example Program Results
");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
    scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &m, 
      &ns, &maxit);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &m, 
      &ns, &maxit);
    #endif
    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))
    

Mark 25  g11cac.7
if (order == Nag_ColMajor)
    pdz = n;
else
    pdz = m;

for (i = 1; i <= n; ++i)
{
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT"%"NAG_IFMT", &isi[i-1], &ic[i-1]);
    #else
        scanf("%"NAG_IFMT"%"NAG_IFMT", &isi[i-1], &ic[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("%*\n");
        #else
            scanf("%*\n");
        #endif
    }

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

/* 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)
    #ifdef _WIN32
        scanf_s("%lf", &b[j-1]);
    #else
        scanf("%lf", &b[j-1]);
    #endif
    #ifdef _WIN32
        scanf("%*\n");
    #else
        scanf("%*\n");
    #endif

/* nag_condl_logistic (gllcac). *
* Returns parameter estimates for the conditional analysis */
gll – Contingency Table 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 (gllcac).\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(" %3"NAG_IFMT" %2"NAG_IFMT" %2"NAG_IFMT"
         " \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("%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(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 (gllcac) 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 (g1lcac) Example Program Results

Deviance = 5.4749e+00

<table>
<thead>
<tr>
<th>Strata</th>
<th>No. Cases</th>
<th>No. Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5223</td>
<td>1.3901</td>
</tr>
<tr>
<td>2</td>
<td>-0.2674</td>
<td>0.8473</td>
</tr>
</tbody>
</table>