**nag_glm_binomial (g02gbc)**

1. **Purpose**

*nag_glm_binomial (g02gbc)* fits a generalized linear model with binomial errors.

2. **Specification**

```c
#include <nag.h>
#include <nagg02.h>

void nag_glm_binomial(Nag_Link link, Nag_IncludeMean mean, Integer n,
                      double x[], Integer tdx, Integer m, Integer sx[],
                      Integer ip, double y[], double binom_t[],
                      double wt[], double offset[],
                      double *dev, double *df, double b[],
                      Integer *rank, double se[],
                      double cov[], double v[], Integer tdv,
                      double tol, Integer max_iter, Integer print_iter,
                      char *outfile, double eps, NagError *fail)
```

3. **Description**

A generalized linear model with binomial errors consists of the following elements:

(a) A set of *n* observations, *y*<sub>i</sub>, from a binomial distribution:

\[ \left( \frac{t}{y} \right) \pi^y (1 - \pi)^{t-y} \]

(b) *X*, a set of *p* independent variables for each observation, *x*<sub>1</sub>, *x*<sub>2</sub>, ..., *x*<sub>p</sub>.

(c) A linear model:

\[ \eta = \sum \beta_j x_j \]

(d) A link function \( \eta = g(\mu) \), linking the linear predictor, \( \eta \) and the mean of the distribution, \( \mu = \pi t \). The possible link functions are:

(i) Logistic link:

\[ \eta = \log \left( \frac{\mu}{t - \mu} \right) \]

(ii) Probit link:

\[ \eta = \Phi^{-1} \left( \frac{\mu}{t} \right) \]

(iii) Complementary log-log link:

\[ \log \left( -\log \left( 1 - \frac{\mu}{t} \right) \right) \]

(e) A measure of fit, the deviance:

\[ \sum_{i=1}^{n} \text{dev}(y_i, \hat{\mu}_i) = \sum_{i=1}^{n} 2 \left\{ y_i \log \left( \frac{y_i}{\hat{\mu}_i} \right) + (t_i - y_i) \log \left( \frac{(t_i - y_i)}{(t_i - \hat{\mu}_i)} \right) \right\} \]

The linear parameters are estimated by iterative weighted least-squares. An adjusted dependent variable, *z*, is formed:

\[ z = \eta + (y - \mu) \frac{d\eta}{d\mu} \]

and a working weight, *w*,

\[ w = \left( \frac{\tau d\eta}{d\mu} \right)^2 \text{ where } \tau = \sqrt{\frac{t}{\mu(t - \mu)}} \]
At each iteration an approximation to the estimate of $\beta$, $\hat{\beta}$ is found by the weighted least-squares regression of $z$ on $X$ with weights $w$.

$nag_{\text{glm_binomial}}$ finds a $QR$ decomposition of $w^{\frac{1}{2}}X$, i.e.,

$$w^{\frac{1}{2}}X = QR$$

where $R$ is a $p$ by $p$ triangular matrix and $Q$ is an $n$ by $p$ column orthogonal matrix.

If $R$ is of full rank then $\hat{\beta}$ is the solution to:

$$R\hat{\beta} = Q^Tw^{\frac{1}{2}}z$$

If $R$ is not of full rank a solution is obtained by means of a singular value decomposition (SVD) of $R$.

$$R = Q_*\begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}P^T,$$

where $D$ is a $k$ by $k$ diagonal matrix with non-zero diagonal elements, $k$ being the rank of $R$ and $w^{\frac{1}{2}}X$.

This gives the solution

$$\hat{\beta} = P_1D^{-1}\begin{pmatrix} Q_* & 0 \\ 0 & I \end{pmatrix}Q^Tw^{\frac{1}{2}}z$$

$P_1$ being the first $k$ columns of $P$, i.e., $P = (P_1 P_0)$.

The iterations are continued until there is only a small change in the deviance.

The initial values for the algorithm are obtained by taking

$$\hat{\eta} = g(y)$$

The fit of the model can be assessed by examining and testing the deviance, in particular, by comparing the difference in deviance between nested models, i.e., when one model is a sub-model of the other. The difference in deviance between two nested models has, asymptotically, a $\chi^2$ distribution with degrees of freedom given by the difference in the degrees of freedom associated with the two deviances.

The parameters estimates, $\hat{\beta}$, are asymptotically Normally distributed with variance-covariance matrix:

$$C = R^{-1}R^{-1T} \text{ in the full rank case, otherwise}$$

$$C = P_1D^{-2}P_1^T$$

The residuals and influence statistics can also be examined.

The estimated linear predictor $\hat{\eta} = X\hat{\beta}$, can be written as $Hw^{\frac{1}{2}}z$ for an $n$ by $n$ matrix $H$. The $i$th diagonal elements of $H$, $h_i$, give a measure of the influence of the $i$th values of the independent variables on the fitted regression model. These are known as leverages.

The fitted values are given by $\hat{\mu} = g^{-1}(\hat{\eta})$.

$nag_{\text{glm_binomial}}$ also computes the deviance residuals, $r$:

$$r_i = \text{sign}(y_i - \hat{\mu}_i)\sqrt{\text{dev}(y_i, \hat{\mu}_i)}.$$

An option allows prior weights to be used with the model.

In many linear regression models the first term is taken as a mean term or an intercept, i.e., $x_{i,1} = 1$, for $i = 1, 2, \ldots, n$. This is provided as an option.

Often only some of the possible independent variables are included in a model; the facility to select variables to be included in the model is provided.

If part of the linear predictor can be represented by a variable with a known coefficient then this can be included in the model by using an offset, $o$:

$$\eta = o + \sum \beta_jx_j.$$
If the model is not of full rank the solution given will be only one of the possible solutions. Other estimates be may be obtained by applying constraints to the parameters. These solutions can be obtained by using \texttt{nag glm tran model (g02gkc)} after using \texttt{nag glm binomial}. Only certain linear combinations of the parameters will have unique estimates, these are known as estimable functions, these can be estimated and tested using \texttt{nag glm est func (g02gnc)}.

Details of the SVD, are made available, in the form of the matrix $P^*$:

$$P^* = \begin{pmatrix} D^{-1}P_T^T \\ P_0^T \\ \end{pmatrix}.$$  

4. Parameters

**link**

Input: indicates which link function is to be used.

- If \texttt{link = Nag.Logistic}, then a logistic link is used.
- If \texttt{link = Nag.Probit}, then a probit link is used.
- If \texttt{link = Nag.Compl}, then a complementary log-log link is used.

Constraint: \texttt{link = Nag.Logistic, Nag.Probit or Nag.Compl}.

**mean**

Input: indicates if a mean term is to be included.

- If \texttt{mean = Nag.MeanInclude}, a mean term, (intercept), will be included in the model.
- If \texttt{mean = Nag.MeanZero}, the model will pass through the origin, zero point.

Constraint: \texttt{mean = Nag.MeanInclude or Nag.MeanZero}.

**n**

Input: the number of observations, \( n \).

Constraint: \( n \geq 2 \).

**x[n][tdx]**

Input: \( x[i-1][j-1] \) must contain the \( i \)th observation for the \( j \)th independent variable, for \( i = 1, 2, \ldots, n; j = 1, 2, \ldots, m \).

**tdx**

Input: the second dimension of the array \( x \) as declared in the function from which \texttt{nag glm binomial} is called.

Constraint: \( tdx \geq m \).

**m**

Input: the total number of independent variables.

Constraint: \( m \geq 1 \).

**sx[m]**

Input: indicates which independent variables are to be included in the model.

- If \( sx[j-1] > 0 \), then the variable contained in the \( j \)th column of \( x \) is included in the regression model.

Constraints: \( sx[j-1] \geq 0 \), for \( j = 1, 2, \ldots, m \).

- If \texttt{mean = Nag.MeanInclude}, then exactly \( ip - 1 \) values of \( sx \) must be \( > 0 \).
- If \texttt{mean = Nag.MeanZero}, then exactly \( ip \) values of \( sx \) must be \( > 0 \).

**ip**

Input: the number \( p \) of independent variables in the model, including the mean or intercept if present.

Constraint: \( ip > 0 \).

**y[n]**

Input: observations on the dependent variable, \( y_i \), for \( i = 1, 2, \ldots, n \).

Constraint: \( 0.0 \leq y[i-1] \leq \text{binom.t}[i-1] \), for \( i = 1, 2, \ldots, n \).

**binom.t[n]**

Input: the binomial denominator, \( t \).

Constraint: \( \text{binom.t}[i] \geq 0.0 \), for \( i = 1, 2, \ldots, n \).
MeanZero
MeanInclude
binomial

\[ y_i = \frac{\exp(\eta_i)}{1 + \exp(\eta_i)} \]

\[ \eta_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \]

\[ \beta_j = \sum_{i=1}^{n} w_i y_i x_{ij} / \sum_{i=1}^{n} w_i x_{ij}^2 \]

\[ \operatorname{dev} = -2 \sum_{i=1}^{n} w_i \log \left( \frac{y_i \exp(\eta_i)}{1 + \exp(\eta_i)} \right) \]

\[ \operatorname{rank} = \min(\operatorname{rank}(\mathbf{x}), n) \]

\[ \operatorname{cov} = \mathbf{V} \mathbf{S} \mathbf{V}^T / \sum_{i=1}^{n} w_i \]

\[ \operatorname{df} = n - \operatorname{rank} \]

\[ \operatorname{se} = \sqrt{\operatorname{diag}(\operatorname{cov})} \]

\[ \operatorname{sdv} = \sqrt{\sum_{i=1}^{n} w_i x_{ij}^2} \]

\[ \operatorname{v}[i-1][0], \text{ contains the linear predictor value, } \eta_i, \text{ for } i = 1, 2, \ldots, n. \]

\[ \operatorname{v}[i-1][1], \text{ contains the fitted value, } \hat{y}_i, \text{ for } i = 1, 2, \ldots, n. \]

\[ \operatorname{v}[i-1][2], \text{ contains the variance standardization, } \tau_i, \text{ for } i = 1, 2, \ldots, n. \]

\[ \operatorname{v}[i-1][3], \text{ contains the working weight, } w_i, \text{ for } i = 1, 2, \ldots, n. \]

\[ \operatorname{v}[i-1][4], \text{ contains the deviance residual, } r_i, \text{ for } i = 1, 2, \ldots, n. \]

\[ \operatorname{v}[i-1][5], \text{ contains the leverage, } h_i, \text{ for } i = 1, 2, \ldots, n. \]

\[ \operatorname{v}[i-1][j-1], \text{ for } j = 7, \ldots, \operatorname{ip} + 6, \text{ contains the results of the QR decomposition or the singular value decomposition.} \]

\[ \operatorname{tdv} \geq \operatorname{ip} + 6. \]
tol
Input: indicates the accuracy required for the fit of the model.
The iterative weighted least-squares procedure is deemed to have converged if the absolute change in deviance between interactions is less than \( tol \times (1.0 + \text{Current Deviance}) \). This is approximately an absolute precision if the deviance is small and a relative precision if the deviance is large.
If \( 0.0 \leq tol < \text{machine precision} \), then the function will use \( 10 \times \text{machine precision} \).
Constraint: \( tol \geq 0.0 \).

max_iter – Integer
Input: the maximum number of iterations for the iterative weighted least-squares.
If \( \text{max}_\text{iter} = 0 \), then a default value of 10 is used.
Constraint: \( \text{max}_\text{iter} \geq 0 \).

print_iter
Input: indicates if the printing of information on the iterations is required and the rate at which printing is produced. The following values are available.
If \( \text{print}_\text{iter} \leq 0 \), then there is no printing.
If \( \text{print}_\text{iter} > 0 \), then the following items are printed every \( \text{print}_\text{iter} \) iterations:
(i) the deviance,
(ii) the current estimates, and
(iii) if the weighted least-squares equations are singular then this is indicated.

outfile
Input: a null terminated character string giving the name of the file to which results should be printed. If \( \text{outfile} = \text{NULL} \) or an empty string then the stdout stream is used. Note that the file will be opened in the append mode.

eps
Input: the value of \( \text{eps} \) is used to decide if the independent variables are of full rank and, if not, what the rank of the independent variables is. The smaller the value of \( \text{eps} \) the stricter the criterion for selecting the singular value decomposition.
If \( 0.0 \leq \text{eps} < \text{machine precision} \), then the function will use \( \text{machine precision} \) instead.
Constraint: \( \text{eps} \geq 0.0 \).

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

For this function the values of output parameters may be useful even if \( \text{fail}.\text{code} \neq \text{NE_NOERROR} \) on exit. Users are therefore advised to supply the \( \text{fail} \) parameter and set \( \text{fail}.\text{print} = \text{TRUE} \).

5. Error Indications and Warnings

NE_BAD_PARAM
On entry parameter \( \text{link} \) had an illegal value.
On entry parameter \( \text{mean} \) had an illegal value.

NE_INT_ARG_LT
On entry, \( n \) must not be less than 2: \( n = \langle \text{value} \rangle \).
On entry, \( m \) must not be less than 1: \( m = \langle \text{value} \rangle \).
On entry, \( \text{ip} \) must not be less than 1: \( \text{ip} = \langle \text{value} \rangle \).
On entry, \( \text{max}_\text{iter} \) must not be less than 0: \( \text{max}_\text{iter} = \langle \text{value} \rangle \).
On entry, \( \text{sx}[(\text{value})] \) must not be less than 0: \( \text{sx}[(\text{value})] = \langle \text{value} \rangle \).

NE_REAL_ARG_LT
On entry, \( \text{tol} \) must not be less than 0.0: \( \text{tol} = \langle \text{value} \rangle \).
On entry, \( \text{eps} \) must not be less than 0.0: \( \text{eps} = \langle \text{value} \rangle \).
On entry, \( \text{wt}[(\text{value})] \) must not be less than 0.0: \( \text{wt}[(\text{value})] = \langle \text{value} \rangle \).
On entry, \( \text{binom}_\text{t}[(\text{value})] \) must not be less than 0.0: \( \text{binom}_\text{t}[(\text{value})] = \langle \text{value} \rangle \).
On entry, \( \text{y}[(\text{value})] \) must not be less than 0.0: \( \text{y}[(\text{value})] = \langle \text{value} \rangle \).
NE_2_INT_ARG_LT
On entry tdx = (value) while m = (value). These parameters must satisfy tdx ≥ m.
On entry tdv = (value) while ip = (value). These parameters must satisfy tdv ≥ ip+6.

NE_2_REAL_ARG_GT
On entry y[⟨value⟩] = (value) while binom_t[⟨value⟩] = (value). These parameters must satisfy y[⟨value⟩] ≤ binom_t[⟨value⟩].

NE_IP_INCOMP_SX
Parameter ip is incompatible with parameters mean and sx.

NE_IP_GT_OBSERV
Parameter ip is greater than the effective number of observations.

NE_VALUE_AT_BOUNDARY_B
A fitted value is at a boundary i.e., 0.0 or 1.0. This may occur if there are y values of 0.0 or
binom_t and the model is too complex for the data. The model should be reformulated with,
perhaps, some observations dropped.

NE_ALLOC_FAIL
Memory allocation failed.

NE_SVD_NOT_CONV
The singular value decomposition has failed to converge.

NE_LSQ_ITER_NOT_CONV
The iterative weighted least-squares has failed to converge in max_iter = (value) iterations.
The value of max_iter could be increased but it may be advantageous to examine the
convergence using the print_iter option. This may indicate that the convergence is slow
because the solution is at a boundary in which case it may be better to reformulate the
model.

NE_RANK_CHANGED
The rank of the model has changed during the weighted least-squares iterations. The estimate
for β returned may be reasonable, but the user should check how the deviance has changed
during iterations.

NE_ZERO_DOF_ERROR
The degrees of freedom for error are 0. A saturated model has been fitted.

NE_NOT_APPEND_FILE
Cannot open file ⟨string⟩ for appending.

NE_NOT_CLOSE_FILE
Cannot close file ⟨string⟩.

6. Further Comments

6.1. Accuracy
The accuracy is determined by tol as described in Section 4. As the adjusted deviance is a function
of log µ the accuracy of the β’s will be a function of tol. tol should therefore be set to a smaller
value than the accuracy required for β.

6.2. References

7. See Also
nag_glm_normal (g02gac)
nag_glm_poisson (g02gcc)
nag_glm_gamma (g02gdc)
nag_glm_tran_model (g02gkc)
nag_glm_est_func (g02gmc)
8. Example
A linear trend \((x = -1, 0, 1)\) is fitted to data relating the incidence of carriers of Streptococcus pyogenes to size of tonsils. The data is described in Cox (1983).

8.1. Program Text

```c
#include <nag.h>
#include <stdio.h>
#include <nag_stlidb.h>
#include <ctype.h>
#include <nagg02.h>

#define NMAX 3
#define MMAX 2
#define TDX MMAX
#define TDV MMAX+6

main()
{
    char linkc, meanc, weightc;
    Nag_Link link;
    Nag_IncludeMean mean;
    Integer i, j, m, n, nvar;
    Integer ivar[MMAX];
    double beta[MMAX], binom[NMAX], v[NMAX][TDV], wt[NMAX],
           x[NMAX][MMAX], y[NMAX];
    double *wtptr, *offsetptr=(double *)0;
    Integer max_iter, print_iter;
    double tol, eps;
    Integer rank;
    double df, dev, se[MMAX], cov[MMAX*(MMAX+1)/2];
    static NagError fail;

    Vprintf("g02gbc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^\n]");
    Vscanf(" %c %c %c %ld %ld %ld", &linkc, &meanc, &weightc,
           &n, &m, &print_iter);
    /* Check and set control parameters */
    set_enum(linkc, &link, meanc, &mean);
    if (n<=NMAX && m<MMAX)
    {
        if (toupper(weightc)=='W')
        {
            wtptr = wt;
            for (i=0; i<n; i++)
            {
                for (j=0; j<m; j++)
                    Vscanf("%lf", &x[i][j]);
                Vscanf("%lf%lf%lf", &y[i], &binom[i], &wt[i]);
            }
        }
        else
        {
```
```


```c
wtptr = (double *)0;
for (i=0; i<n; i++)
{
    for (j=0; j<m; j++)
        Vscanf("%lf", &x[i][j]);
    Vscanf("%lf%lf", &y[i], &binom[i]);
}
for (j=0; j<m; j++)
    Vscanf("%ld", &ivar[j]);
/* Calculate nvar */
nvar = 0;
for (i=0; i<m; i++)
    if (ivar[i]>0) nvar += 1;
if (mean == Nag_MeanInclude)
    nvar += 1;
/* Set other control parameters */
max_iter = 10;
tol = 5e-5;
eps = 1e-6;
g02gbc(link, mean, n, (double *)x, (Integer)TDX, m,
ivar, nvar, y, binom, wtptr, offsetptr, &dev, &df, beta, &rank,
se, cov,(double *)v, (Integer)TDV, tol, max_iter, print_iter, "",
eps, &fail);
if (fail.code == NE_NOERROR || fail.code == NE_SVD_NOT_CONV ||
    fail.code == NE_LSQ_ITER_NOT_CONV || fail.code == NE_RANK_CHANGED || fail.code == NE_ZERO_DOF_ERROR)
{
    Vprintf("\nDeviance = %12.4e\n", dev);
    Vprintf("Degrees of freedom = %3.1f\n", df);
    Vprintf(" Estimate Standard error\n\n");
    for (i=0; i<nvar; i++)
        Vprintf("%14.4f%14.4f\n", beta[i], se[i]);
    Vprintf("\n");
    Vprintf(" binom y fitted value Residual Leverage\n\n");
    for (i = 0; i < n; ++i)
    {
        Vprintf("%10.1f%7.1f%10.2f%12.4f%10.3f\n", binom[i], y[i],
            v[i][1], v[i][4], v[i][5]);
    }
}
else
{
    Vprintf("%s\n",fail.message);
    exit(EXIT_FAILURE);
}
else
{
    Vfprintf(stderr, "One or both of m and n are out of range:\n m = %-3ld while n = %-3ld\n", m, n);
    exit(EXIT_FAILURE);
}
exit(EXIT_SUCCESS);
```

```c
#ifdef NAG_PROTO
static void set_enum(char linkc, Nag_Link *link, char meanc,
Nag_IncludeMean *mean)
#else
static void set_enum(linkc, link, meanc, mean)
char linkc;
Nag_Link *link;
char meanc;
Nag_IncludeMean *mean;
#endif
```
{ } if (toupper(linkc) == 'G' || toupper(linkc) == 'P' || toupper(linkc) == 'C') {
    switch (toupper(linkc))
    {
    case ('G'):
        *link = Nag_Logistic;
        break;
    case ('P'):
        *link = Nag_Probit;
        break;
    case ('C'):
        *link = Nag_Compl;
        break;
    default:
    }
}
else
{
    Vfprintf(stderr, "The parameter link has an invalid value: link = %c\n", linkc);
    exit(EXIT_FAILURE);
}
if (toupper(meanc)=='M')
    *mean = Nag_MeanInclude;
else if (toupper(meanc)=='Z')
    *mean = Nag_MeanZero;
else
{
    Vfprintf(stderr, "The parameter mean has an invalid value: mean = %c\n", meanc);
    exit(EXIT_FAILURE);
}
return;

8.2. Program Data

g02gbc Example Program Data
  g m n 3 1 0
  1.0 19. 516.
  0.0 29. 560.
  -1.0 24. 293.
  1

8.3. Program Results

g02gbc Example Program Results

Deviance = 7.3539e-02
Degrees of freedom = 1.0

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Standard error</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.8682</td>
<td>0.1217</td>
</tr>
<tr>
<td>-0.4264</td>
<td>0.1598</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>binom</th>
<th>y</th>
<th>fitted value</th>
<th>Residual</th>
<th>Leverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>516.0</td>
<td>19.0</td>
<td>18.45</td>
<td>0.1296</td>
<td>0.769</td>
</tr>
<tr>
<td>560.0</td>
<td>29.0</td>
<td>30.10</td>
<td>-0.2070</td>
<td>0.422</td>
</tr>
<tr>
<td>293.0</td>
<td>24.0</td>
<td>23.45</td>
<td>0.1178</td>
<td>0.809</td>
</tr>
</tbody>
</table>