# nag_glm_binomial (g02gbc)

## 1. Purpose

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

## 2. Specification

```
#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, $y_i$, from a binomial distribution:

$$\binom{t}{y}\pi^y(1-\pi)^{t-y}.$$

(b) $X$, a set of $p$ independent variables for each observation, $x_1, x_2, \ldots, x_p$.

(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:

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

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, $w$,

$$w = \left(\tau\frac{d\eta}{d\mu}\right)^2 \text{ where } \tau = \sqrt{\frac{t}{\mu(t-\mu)}}$$

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_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:

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

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

$$\hat{\beta} = P_1 D^{-1} \begin{pmatrix} Q_* & 0 \\ 0 & I \end{pmatrix} Q^T w^{\frac{1}{2}} z$$

$P_1$ 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:

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

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

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_glm_binomial also computes the deviance residuals, $r$:

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

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, $o$:

$$\eta = o + \sum \beta_j x_j.$$

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 nag_glm_tran_model (g02gkc) after using 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 nag_glm_est_func (g02gnc).

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

$$P^* = \begin{pmatrix} D^{-1}P_1^T \\ P_0^T \end{pmatrix}.$$

## 4. Parameters

**link**

Input: indicates which link function is to be used.
If **link** = **Nag_Logistic**, then a logistic link is used.
If **link** = **Nag_Probit**, then a probit link is used.
If **link** = **Nag_Compl**, then a complementary log-log link is used.
Constraint: **link** = **Nag_Logistic**, **Nag_Probit** or **Nag_Compl**.

**mean**

Input: indicates if a mean term is to be included.
If **mean** = **Nag_MeanInclude**, a mean term, (intercept), will be included in the model.
If **mean** = **Nag_MeanZero**, the model will pass through the origin, zero point.
Constraint: **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 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 **mean** = **Nag_MeanInclude**, then exactly **ip** $-1$ values of **sx** must be $> 0$.
if **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**$[i-1] \leq$ **binom_t**$[i-1]$, for $i = 1, 2, \ldots, n$.

**binom_t[n]**

Input: the binomial denominator, $t$.
Constraint: **binom_t**$[i] \geq 0.0$, for $i = 1, 2, \ldots, n$.

**wt[n]**

Input: if weighted estimates are required then **wt** must contain the weights to be used with the model, $\omega_i$. Otherwise **wt** must be supplied as the null pointer, (double *)0.
If $\mathbf{wt}[i-1] = 0.0$, then the $i$th observation is not included in the model, in which case the effective number of observations is the number of observations with positive weights.
If **wt** = null pointer, then the effective number of observations is $n$.
Constraint: **wt** = null pointer or $\mathbf{wt}[i-1] \geq 0.0$, for $i = 1, 2, \ldots, n$.

**offset[n]**

Input: if an offset is required then **offset** must contain the values of the offset $o$. Otherwise **offset** must be supplied as the null pointer, (double *)0.

**dev**

Output: the deviance for the fitted model.

**df**

Output: the degrees of freedom associated with the deviance for the fitted model.

**b[ip]**

Output: $\mathbf{b}[i-1]$, $i = 1, \ldots,$**ip** contains the estimates of the parameters of the generalized linear model, $\hat{\beta}$.
If **mean** = **Nag_MeanInclude**, then $\mathbf{b}[0]$ will contain the estimate of the mean parameter and $\mathbf{b}[i]$ will contain the coefficient of the variable contained in column $j$ of **x**, where $\mathbf{sx}[j-1]$ is the $i$th positive value in the array **sx**.
If **mean** = **Nag_MeanZero**, then $\mathbf{b}[i-1]$ will contain the coefficient of the variable contained in column $j$ of **x**, where $\mathbf{sx}[j-1]$ is the $i$th positive value in the array **sx**.

**rank**

Output: the rank of the independent variables.
If the model is of full rank, then **rank** = **ip**.
If the model is not of full rank, then **rank** is an estimate of the rank of the independent variables. **rank** is calculated as the number of singular values greater than **eps** $\times$ (largest singular value). It is possible for the SVD to be carried out but **rank** to be returned as **ip**.

**se[ip]**

Output: the standard errors of the linear parameters.
$\mathbf{se}[i-1]$ contains the standard error of the parameter estimate in $\mathbf{b}[i-1]$, for $i = 1, 2, \ldots,$ **ip**.

**cov[ip∗(ip+1)/2]**

Output: the $\mathbf{ip} \times (\mathbf{ip}+1)/2$ elements of **cov** contain the upper triangular part of the variance-covariance matrix of the **ip** parameter estimates given in **b**. They are stored packed by column, i.e., the covariance between the parameter estimate given in $\mathbf{b}[i]$ and the parameter estimate given in $\mathbf{b}[j]$, $j \geq i$, is stored in $\mathbf{cov}[j(j+1)/2+i]$, for $i = 0, 1, \ldots, \mathbf{ip}-1$ and $j = i, i+1, \ldots, \mathbf{ip}-1$.

**v[n][tdv]**

Output: auxiliary information on the fitted model.
$\mathbf{v}[i-1][0]$, contains the linear predictor value, $\eta_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][1]$, contains the fitted value, $\hat{\mu}_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][2]$, contains the variance standardization, $\tau_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][3]$, contains the working weight, $w_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][4]$, contains the deviance residual, $r_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][5]$, contains the leverage, $h_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][j-1]$, for $j = 7, \ldots,$**ip**+6, contains the results of the $QR$ decomposition or the singular value decomposition.

If the model is not of full rank, i.e., **rank** < **ip**, then the first **ip** rows of columns 7 to **ip**+6 contain the $P^*$ matrix.

**tdv**

Input: the second dimension of the array **v** as declared in the function from which nag_glm_binomial is called.
Constraint: **tdv** $\geq$ **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+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** $<$ ***machine precision***, then the function will use $10\times$ ***machine precision***.
Constraint: **tol** $\geq 0.0$.

**max_iter** – Integer

Input: the maximum number of iterations for the iterative weighted least-squares.
If **max_iter** $= 0$, then a default value of 10 is used.
Constraint: **max_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 **print_iter** $\leq 0$, then there is no printing.
If **print_iter** $> 0$, then the following items are printed every **print_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 **outfile = 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 **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 **eps** the stricter the criterion for selecting the singular value decomposition.
If $0.0 \leq$ **eps** $<$ ***machine precision***, then the function will use ***machine precision*** instead.
Constraint: **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 **fail.code** $\neq$ **NE_NOERROR** on exit. Users are therefore advised to supply the **fail** parameter and set **fail.print** = `TRUE`.

## 5. Error Indications and Warnings

**NE_BAD_PARAM**

On entry parameter **link** had an illegal value.
On entry parameter **mean** had an illegal value.

**NE_INT_ARG_LT**

On entry, **n** must not be less than 2: **n** = ⟨*value*⟩.
On entry, **m** must not be less than 1: **m** = ⟨*value*⟩.
On entry, **ip** must not be less than 1: **ip** = ⟨*value*⟩.
On entry, **max_iter** must not be less than 0: **max_iter** = ⟨*value*⟩.
On entry, **sx**[⟨*value*⟩] must not be less than 0: **sx**[⟨*value*⟩] = ⟨*value*⟩.

**NE_REAL_ARG_LT**

On entry, **tol** must not be less than 0.0: **tol** = ⟨*value*⟩.
On entry, **eps** must not be less than 0.0: eps = ⟨*value*⟩.
On entry, **wt**[⟨*value*⟩] must not be less than 0.0: **wt**[⟨*value*⟩] = ⟨*value*⟩.
On entry, **binom_t**[⟨*value*⟩] must not be less than 0.0: **binom_t**[⟨*value*⟩] = ⟨*value*⟩.
On entry, **y**[⟨*value*⟩] must not be less than 0.0: **y**[⟨*value*⟩] = ⟨*value*⟩.

**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 $\beta$ 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 $\log\mu$ the accuracy of the $\hat{\beta}$'s will be a function of **tol**. **tol** should therefore be set to a smaller value than the accuracy required for $\hat{\beta}$.

### 6.2. References

Cook R D and Weisberg S (1982) *Residuals and Influence in Regression.* Chapman and Hall.
Cox D R (1983) *Analysis of Binary Data.* Chapman and Hall.
McCullagh P and Nelder J A (1983) *Generalized Linear Models* Chapman and Hall.

## 7. See Also

nag_glm_normal (g02gac)
nag_glm_poisson (g02gcc)
nag_glm_gamma (g02gdc)
nag_glm_tran_model (g02gkc)
nag_glm_est_func (g02gnc)

## 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

```
/* nag_glm_binomial(g02gbc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <ctype.h>
#include <nagg02.h>

#ifdef NAG_PROTO
static void set_enum(char linkc, Nag_Link *link, char meanc,
                     Nag_IncludeMean *mean);
#else
static void set_enum();
#endif

#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
        {
```

```
          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\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:\
 m = %-3ld while  n = %-3ld\n", m, n);
      exit(EXIT_FAILURE);
    }
  exit(EXIT_SUCCESS);
}

#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

       Estimate     Standard error

       -2.8682        0.1217
       -0.4264        0.1598

     binom     y   fitted value  Residual  Leverage

     516.0   19.0     18.45      0.1296     0.769
     560.0   29.0     30.10     -0.2070     0.422
     293.0   24.0     23.45      0.1178     0.809
```