1 Purpose

nag_glm_poisson (g02gcc) fits a generalized linear model with Poisson errors.

2 Specification

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

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

3 Description

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

(a) a set of \( n \) observations, \( y_i \), from a Poisson distribution:

\[
\frac{\mu^y e^{-\mu}}{y!}
\]

(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 between the linear predictor, \( \eta \), and the mean of the distribution, \( \mu \), \( \eta = g(\mu) \). The possible link functions are:

(i) exponent link: \( \eta = \mu^a \), for a constant \( a \),

(ii) identity link: \( \eta = \mu \),

(iii) log link: \( \eta = \log \mu \),

(iv) square root link: \( \eta = \sqrt{\mu} \),

(e) reciprocal link: \( \eta = \frac{1}{\mu} \).

(f) a measure of fit, the deviance:

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

The linear arguments 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, \( w \),
\[ w = \left( \tau \frac{d\mu}{d\mu} \right)^2, \] where \( \tau = \sqrt{\mu} \).

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 \( w \).

\text{nag_glm_poisson (g02gcc)} finds a QR decomposition of \( w^2 X \), i.e., \( w^2 X = QR \) 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:
\[ R \hat{\beta} = Q^T w^2 z \]

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 nonzero diagonal elements, \( k \) being the rank of \( R \) and \( w^2 X \). This gives the solution
\[ \hat{\beta} = P_1 D^{-1} \begin{pmatrix} Q_* & 0 \\ 0 & 1 \end{pmatrix} Q^T w^2 z \]

\( P_1 \) 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 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 arguments estimates, \( \hat{\beta} \), are asymptotically Normally distributed with variance-covariance matrix:
\[ C = R^{-1} R^{-T} \] 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 \( H w^2 z \) 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 \( \hat{\mu} = g^{-1}(\hat{\eta}) \).

\text{nag_glm_poisson (g02gcc)} 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, \( o \).
If the model is not of full rank the solution given will be only one of the possible solutions. Other estimates may be obtained by applying constraints to the arguments. These solutions can be obtained by using nag_glm_tran_model (g02gkc) after using nag_glm_poisson (g02gcc).

Only certain linear combinations of the arguments 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 $P^*$:

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

The generalized linear model with Poisson errors can be used to model contingency table data, see Cook and Weisberg (1982) and McCullagh and Nelder (1983).

4 References
Plackett R L (1974) The Analysis of Categorical Data Griffin

5 Arguments

1: \textbf{link} – Nag_Link  
\textit{Input} 
\textit{On entry}: indicates which link function is to be used.  
\textbf{link} = Nag_Expo  
An exponent link is used.
\textbf{link} = Nag_Iden  
An identity link is used.
\textbf{link} = Nag_Log  
A log link is used.
\textbf{link} = Nag_Sqrt  
A square root link is used.
\textbf{link} = Nag_Reci  
A reciprocal link is used.  
\textit{Constraint}: \textbf{link} = Nag_Expo, Nag_Iden, Nag_Log, Nag_Sqrt or Nag_Reci.

2: \textbf{mean} – Nag_IncludeMean  
\textit{Input} 
\textit{On entry}: indicates if a mean term is to be included.  
\textbf{mean} = Nag_MeanInclude  
A mean term, (intercept), will be included in the model.
\textbf{mean} = Nag_MeanZero  
The model will pass through the origin, zero point.  
\textit{Constraint}: \textbf{mean} = Nag_MeanInclude or Nag_MeanZero.

3: \textbf{n} – Integer  
\textit{Input} 
\textit{On entry}: the number of observations, $n$.  
\textit{Constraint}: $n \geq 2$.  

Mark 25
4: \( x[n \times tdx] \) – const double

*Input*

*On entry:* \( x[(i-1) \times tdx + j-1] \) must contain the \( i \)th observation for the \( j \)th independent variable, for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \).

5: \( tdx \) – Integer

*Input*

*On entry:* the stride separating matrix column elements in the array \( x \).

*Constraint:* \( tdx \ge m \).

6: \( m \) – Integer

*Input*

*On entry:* the total number of independent variables.

*Constraint:* \( m \ge 1 \).

7: \( sx[m] \) – const Integer

*Input*

*On entry:* 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:*

\[
sx[j-1] \ge 0, \text{ for } j = 1, 2, \ldots, m;
\]

if \( \text{mean} = \text{Nag\_MeanInclude} \), then exactly \( ip - 1 \) values of \( sx \) must be \( > 0 \);

if \( \text{mean} = \text{Nag\_MeanZero} \), then exactly \( ip \) values of \( sx \) must be \( > 0 \).

8: \( ip \) – Integer

*Input*

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

*Constraint:* \( ip > 0 \).

9: \( y[n] \) – const double

*Input*

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

*Constraint:* \( y[i-1] \ge 0, \text{ for } i = 1, 2, \ldots, n \).

10: \( wt[n] \) – const double

*Input*

*On entry:* if weighted estimates are required, then \( wt \) must contain the weights to be used. Otherwise \( wt \) need not be defined and may be set to NULL.

If \( wt[i-1] = 0.0 \), 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 \( wt \) is NULL, then the effective number of observations is \( n \).

*Constraint:* \( wt \) is NULL or \( wt[i-1] \ge 0.0, \text{ for } i = 1, 2, \ldots, n \).

11: \( offset[n] \) – const double

*Input*

*On entry:* if an offset is required then \( offset \) must contain the values of the offset \( o \). Otherwise \( offset \) must be supplied as NULL.

12: \( ex\_power \) – double

*Input*

*On entry:* if \( link = \text{Nag\_Expo} \) then \( ex\_power \) must contain the power \( a \) of the exponential.

If \( link \neq \text{Nag\_Expo} \), \( ex\_power \) is not referenced.

*Constraint:* If \( link = \text{Nag\_Expo} \), \( ex\_power \neq 0.0 \).
**g02 – Correlation and Regression Analysis**

13: **dev** – double

*Output*  
*On exit:* the deviance for the fitted model.

14: **df** – double

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

15: **b[i]** – double

*Output*  
*On exit:* the estimates of the arguments of the generalized linear model, \( \hat{\beta} \).

If \( \text{mean} = \text{Nag}_\text{MeanInclude} \), then \( b[0] \) will contain the estimate of the mean argument and \( b[i] \)
will contain the coefficient of the variable contained in column \( j \) of \( x \), where \( sx[j-1] \) is the \( i \)th positive value in the array \( sx \).

If \( \text{mean} = \text{Nag}_\text{MeanZero} \), then \( b[i-1] \) will contain the coefficient of the variable contained in column \( j \) of \( x \), where \( sx[j-1] \) is the \( i \)th positive value in the array \( sx \).

16: **rank** – Integer

*Output*  
*On exit:* the rank of the independent variables.

If the model is of full rank, then \( \text{rank} = \text{ip} \).

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

17: **se[i]** – double

*Output*  
*On exit:* the standard errors of the linear arguments.  
\( \text{se}[i-1] \) contains the standard error of the parameter estimate in \( b[i-1] \), for \( i = 1, 2, \ldots, \text{ip} \).

18: **cov[ip \times (ip + 1)/2]** – double

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

19: **v[n \times \text{tdv}]** – double

*Output*  
*On exit:* auxiliary information on the fitted model.  
\( v[(i-1) \times \text{tdv}] \), contains the linear predictor value, \( \eta_i \), for \( i = 1, 2, \ldots, n \).  
\( v[(i-1) \times \text{tdv} + 1] \), contains the fitted value, \( \hat{\mu}_i \), for \( i = 1, 2, \ldots, n \).  
\( v[(i-1) \times \text{tdv} + 2] \), contains the variance standardization, \( \tau_i \), for \( i = 1, 2, \ldots, n \).  
\( v[(i-1) \times \text{tdv} + 3] \), contains the working weight, \( w_i \), for \( i = 1, 2, \ldots, n \).  
\( v[(i-1) \times \text{tdv} + 4] \), contains the deviance residual, \( r_i \), for \( i = 1, 2, \ldots, n \).  
\( v[(i-1) \times \text{tdv} + 5] \), contains the leverage, \( h_i \), for \( i = 1, 2, \ldots, n \).  
\( v[(i-1) \times \text{tdv} + j - 1] \), for \( j = 7, 8, \ldots, \text{ip} + 6 \), contains the results of the \( QR \) decomposition or the singular value decomposition.

If the model is not of full rank, i.e., \( \text{rank} < \text{ip} \), then the first \( \text{ip} \) rows of columns 7 to \( \text{ip} + 6 \) contain the \( P^* \) matrix.

20: **tdv** – Integer

*Input*  
*On entry:* the stride separating matrix column elements in the array \( v \).  
*Constraint:* \( \text{tdv} \geq \text{ip} + 6 \).
21: tol – double  
*Input*

*On entry:* 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 \( \text{tol} \times (1.0 + \text{Current Deviance}) \). This is approximately an absolute precision if the deviance is small and a relative precision if the deviance is large.

If \( 0.0 \leq \text{tol} < \text{machine precision} \), then the function will use \( 10 \times \text{machine precision} \).

*Constraint:* \( \text{tol} \geq 0.0 \).

22: max_iter – Integer  
*Input*

*On entry:* the maximum number of iterations for the iterative weighted least squares.

If \( \text{max_iter} = 0 \), then a default value of 10 is used.

*Constraint:* \( \text{max_iter} \geq 0 \).

23: print_iter – Integer  
*Input*

*On entry:* indicates if the printing of information on the iterations is required and the rate at which printing is produced.

\( \text{print_iter} \leq 0 \)  
- There is no printing.

\( \text{print_iter} > 0 \)  
- 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.

24: outfile – const char *  
*Input*

*On entry:* a null terminated character string giving the name of the file to which results should be printed. If outfile is NULL or an empty string then the stdout stream is used. Note that the file will be opened in the append mode.

25: eps – double  
*Input*

*On entry:* 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 \( 0.0 \leq \text{eps} < \text{machine precision} \), then the function will use *machine precision* instead.

*Constraint:* \( \text{eps} \geq 0.0 \).

26: fail – NagError *  
*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_2_INT_ARG_LT**  
On entry, tdv = \( \langle \text{value} \rangle \) while ip = \( \langle \text{value} \rangle \). These arguments must satisfy \( \text{tdv} \geq \text{ip} + 6 \).

On entry, tdx = \( \langle \text{value} \rangle \) while m = \( \langle \text{value} \rangle \). These arguments must satisfy \( \text{tdx} \geq \text{m} \).

**NE_ALLOC_FAIL**  
Dynamic memory allocation failed.
NE_BAD_PARAM
  On entry, argument link had an illegal value.
  On entry, argument mean had an illegal value.

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

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

NE_IP_INCOMP SX
  Argument ip is incompatible with mean and sx.

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_NOT_APPEND_FILE
  Cannot open file ⟨string⟩ for appending.

NE_NOT_CLOSE_FILE
  Cannot close file ⟨string⟩.

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 you should check how the deviance has changed during iterations.

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

NE_REAL_ENUM_ARG_CONS
  On entry, ex_power = 0.0, link = Nag_Expo. These arguments must satisfy link = Nag_Expo and ex_power ≠ 0.0.

NE_SVD_NOT_CONV
  The singular value decomposition has failed to converge.
A fitted value is at a boundary, i.e., $\mu = 0.0$. This may occur if there are $y$ values of 0.0 and the model is too complex for the data. The model should be reformulated with, perhaps, some observations dropped.

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

The accuracy is determined by $\text{tol}$ as described in Section 5. As the adjusted deviance is a function of $\log \mu$ the accuracy of the $\beta$'s will be a function of $\text{tol}$. $\text{tol}$ should therefore be set to a smaller value than the accuracy required for $\beta$.

Not applicable.

None.

A 3 by 5 contingency table given by Plackett (1974) is analysed by fitting terms for rows and columns. The table is:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>67</td>
<td>114</td>
<td>79</td>
<td>39</td>
</tr>
<tr>
<td>131</td>
<td>66</td>
<td>143</td>
<td>72</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>14</td>
<td>38</td>
<td>28</td>
<td>16</td>
</tr>
</tbody>
</table>

```
/* nag_glm_poisson (g02gcc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 4, 1996.
 * Mark 6 revised, 2000.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <ctype.h>
#include <nagg02.h>
#define X(I, J) x[(I) *tdx + J]
#define V(I, J) v[(I) *tdv + J]
int main(void)
{
    Integer exit_status = 0, i, ip, j, m, max_iter, n, print_iter, rank;
    Integer *sx = 0;
    Integer tdx, tdv;
    Nag_IncludeMean mean;
    Nag_Link link;
    Nag_Boolean weight;
    char *nag_enum_arg[40];
    double dev, df, eps, ex_power, tol;
    double *b = 0, *cov = 0, *offsetptr = 0, *se = 0;
```
double *v = 0, *wt = 0, *wtptr, *x = 0, *y = 0;
NagError fail;

INIT_FAIL(fail);

printf("nag_glm_poisson (g02gcc) Example Program Results\n");
/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n"]);
#else
scanf("%*[\n"]);
#endif
#ifdef _WIN32
scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value */
link = (Nag_Link) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s", nag_enum_arg);
#endif
mean = (Nag_IncludeMean) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s", nag_enum_arg);
#endif
weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
scanf("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT", &n, &m, &print_iter);
#else
scanf("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT", &n, &m, &print_iter);
#endif

/* Check and set control parameters */
if (n >= 2 && m >= 1)
{
    if (!(wt = NAG_ALLOC(n, double)) ||
        !(x = NAG_ALLOC(n*m, double)) ||
        !(y = NAG_ALLOC(n, double)) ||
        !(sx = NAG_ALLOC(m, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tdx = m;
}
else
{
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
}
if (weight)
{
    wtptr = wt;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        {
            #ifdef _WIN32
            scanf_s("%lf", &X(i, j));
            #else
            scanf("%lf", &X(i, j));
            #endif
        }
    }
}

Mark 25
```c
scanf_s("%lf%lf", &y[i], &wt[i]);
#else
scanf("%lf%lf", &y[i], &wt[i]);
#endif
}
}
else
{
    wtptr = (double *) 0;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
            #ifdef _WIN32
                scanf_s("%lf", &X(i, j));
            #else
                scanf("%lf", &X(i, j));
            #endif
            #ifdef _WIN32
                scanf_s("%lf", &y[i]);
            #else
                scanf("%lf", &y[i]);
            #endif
    }
    for (j = 0; j < m; j++)
        #ifdef _WIN32
            scanf_s("%"NAG_IFMT"",&sx[j]);
        #else
            scanf("%"NAG_IFMT", &sx[j]);
        #endif
    /* Calculate ip */
    ip = 0;
    for (j = 0; j < m; j++)
    {
        if (sx[j] > 0) ip += 1;
        if (mean == Nag_MeanInclude)
            ip += 1;
        if (link == Nag_Expo)
            #ifdef _WIN32
                scanf_s("%lf", &ex_power);
            #else
                scanf("%lf", &ex_power);
            #endif
        else
            scanf("%lf", &ex_power);
    }
    else
    {
        ex_power = 0.0;
        if (!b || !v || !se || !cov)
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    tdv = ip+6;
    /* Set other control parameters */
    max_iter = 10;
    tol = 5e-5;
    eps = 1e-6;

    /* nag_glm_poisson (g02gcc). */
    /* Fits a generalized linear model with Poisson errors */
    nag_glm_poisson(link, mean, n, x, tdx, m, sx, ip, y,
        wtptr, offsetptr, ex_power, &dev, &df, b, &rank, se, cov,
        v, tdv, tol, max_iter, print_iter,
        ",", eps, &fail);

    if (fail.code == NE_NOERROR || fail.code == NE_LSQ_ITER_NOT_CONV ||
```
fail.code == NE_RANK_CHANGED || fail.code == NE_ZERO_DOF_ERROR) {
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_glm_poisson (g02gcc).\n%s\n", fail.message);
    }
    printf("\nDeviance = %13.4e\n", dev);
    printf("Degrees of freedom = %3.1f\n\n", df);
    printf(" Estimate Standard error\n\n");
    for (i = 0; i < ip; i++)
        printf("%14.4f%14.4f\n", b[i], se[i]);
    printf("\n");
    printf(" y fitted value Residual Leverage\n\n");
    for (i = 0; i < n; ++i)
        { printf("%7.1f%10.2f%12.4f%10.3f\n", y[i], V(i, 1), V(i, 4), V(i, 5));
        }
    }
else {
    printf("Error from nag_glm_poisson (g02gcc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(wt);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(sx);
NAG_FREE(b);
NAG_FREE(v);
NAG_FREE(se);
NAG_FREE(cov);
return exit_status;
}

10.2 Program Data
nag_glm_poisson (g02gcc) Example Program Data
Nag_Log Nag_MeanInclude Nag_FALSE 15 8 0
1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 141.
1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 67.
1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 114.
1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 79.
1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 39.
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 131.
0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 66.
0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 143.
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 72.
0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 35.
0.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 36.
0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 14.
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 38.
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 28.
0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 16.
1 1 1 1 1 1 1 1 1 1

10.3 Program Results
nag_glm_poisson (g02gcc) Example Program Results
Deviance = 9.0379e+00
Degrees of freedom = 8.0

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Standard error</th>
</tr>
</thead>
</table>

Mark 25 g02gcc.11
<table>
<thead>
<tr>
<th>y</th>
<th>fitted value</th>
<th>Residual</th>
<th>Leverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>141.0</td>
<td>132.99</td>
<td>0.6875</td>
<td>0.604</td>
</tr>
<tr>
<td>67.0</td>
<td>63.47</td>
<td>0.4386</td>
<td>0.514</td>
</tr>
<tr>
<td>114.0</td>
<td>127.38</td>
<td>-1.2072</td>
<td>0.596</td>
</tr>
<tr>
<td>79.0</td>
<td>77.29</td>
<td>0.1936</td>
<td>0.532</td>
</tr>
<tr>
<td>39.0</td>
<td>38.86</td>
<td>0.0222</td>
<td>0.482</td>
</tr>
<tr>
<td>131.0</td>
<td>135.11</td>
<td>-0.3553</td>
<td>0.608</td>
</tr>
<tr>
<td>66.0</td>
<td>64.48</td>
<td>0.1881</td>
<td>0.520</td>
</tr>
<tr>
<td>143.0</td>
<td>129.41</td>
<td>1.1749</td>
<td>0.601</td>
</tr>
<tr>
<td>72.0</td>
<td>78.52</td>
<td>-0.7465</td>
<td>0.537</td>
</tr>
<tr>
<td>35.0</td>
<td>39.48</td>
<td>-0.7271</td>
<td>0.488</td>
</tr>
<tr>
<td>36.0</td>
<td>39.90</td>
<td>-0.6276</td>
<td>0.393</td>
</tr>
<tr>
<td>14.0</td>
<td>19.04</td>
<td>-1.2131</td>
<td>0.255</td>
</tr>
<tr>
<td>38.0</td>
<td>38.21</td>
<td>-0.0346</td>
<td>0.382</td>
</tr>
<tr>
<td>28.0</td>
<td>23.19</td>
<td>0.9675</td>
<td>0.282</td>
</tr>
<tr>
<td>16.0</td>
<td>11.66</td>
<td>1.2028</td>
<td>0.206</td>
</tr>
</tbody>
</table>