NAG Library Function Document

nag_glm_normal (g02gac)

1 Purpose

nag_glm_normal (g02gac) fits a generalized linear model with normal errors.

2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_glm_normal (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 *scale, double ex_power, double *rss, 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 Normal errors consists of the following elements:

(a) a set of n observations, y_i, from a Normal distribution with probability density function:

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right),$$

where μ is the mean and σ^2 is the variance.

(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, η, and the mean of the distribution, μ, i.e., $\eta = g(\mu)$. 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: $\eta = \sqrt{\mu}$,

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

(e) a measure of fit, the residual sum of squares $= \sum (y_i - \hat{\mu}_i)^2$

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,

$$w = \left(\frac{d\eta}{d\mu}\right)^2.$$

At each iteration an approximation to the estimate of β, $\hat{\beta}$, is found by the weighted least squares regression of z on X with weights w.

nag_glm_normal (g02gac) 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 a n by p column orthogonal matrix.

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

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

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

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

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

This gives the solution

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

P_1 being the first k columns of P, i.e., $P = (P_1P_0)$.

The iterations are continued until there is only a small change in the residual sum of squares.

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 residual sum of squares, in particular comparing the difference in residual sums of squares between nested models, i.e., when one model is a sub-model of the other.

Let RSS_f be the residual sum of squares for the full model with degrees of freedom ν_f and let RSS_s be the residual sum of squares for the sub-model with degrees of freedom ν_s then:

$$F = \frac{\left(RSS_s - RSS_f\right)/\left(\nu_s - \nu_f\right)}{RSS_f/\nu_f},$$

has, approximately, a F-distribution with $\left(\nu_s - \nu_f\right)$, ν_f degrees of freedom.

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

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

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

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 ith diagonal elements of H, h_i, give a measure of the influence of the ith values of the independent variables on the fitted regression model. These are sometimes known as leverages.

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

nag_glm_normal (g02gac) also computes the residuals, r:

$$r_i = y_i - \hat{\mu}_i$$

An option allows prior weights, ω_i to be used, this gives a model with:

$$\sigma_i^2 = \frac{\sigma^2}{\omega_i}.$$

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 arguments. These solutions can be obtained by using nag_glm_tran_model (g02gkc) after using nag_glm_normal (g02gac). Only certain linear combinations of the arguments will have unique estimates; these are known as estimable functions and 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^*:

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

4 References

Cook R D and Weisberg S (1982) *Residuals and Influence in Regression* Chapman and Hall

McCullagh P and Nelder J A (1983) *Generalized Linear Models* Chapman and Hall

5 Arguments

1: **link** – Nag_Link *Input*

On entry: indicates which link function is to be used.

link = Nag_Expo
An exponent link is used.

link = Nag_Iden
An identity link is used. You are advised not to use nag_glm_normal (g02gac) with an identity link as nag_regsn_mult_linear (g02dac) provides a more efficient way of fitting such a model.

link = Nag_Log
A log link is used.

link = Nag_Sqrt
A square root link is used.

link = Nag_Reci
A reciprocal link is used.

Constraint: **link** = Nag_Expo, Nag_Iden, Nag_Log, Nag_Sqrt or Nag_Reci.

2: **mean** – Nag_IncludeMean *Input*

On entry: indicates if a mean term is to be included.

mean = Nag_MeanInclude
A mean term, (intercept), will be included in the model.

mean = Nag_MeanZero
The model will pass through the origin, zero point.

Constraint: **mean** = Nag_MeanInclude or Nag_MeanZero.

3: **n** – Integer *Input*

On entry: the number of observations, n.

Constraint: $\mathbf{n} \geq 2$.

4: **x**[**n** × **tdx**] – const double *Input*

On entry: $\mathbf{x}[(i-1) \times \mathbf{tdx} + j - 1]$ must contain the ith observation for the jth independent variable, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \mathbf{m}$.

5: **tdx** – Integer *Input*

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

Constraint: $\mathbf{tdx} \geq \mathbf{m}$.

6: **m** – Integer *Input*

On entry: the total number of independent variables.

Constraint: $\mathbf{m} \geq 1$.

7: **sx**[**m**] – const Integer *Input*

On entry: indicates which independent variables are to be included in the model. If $\mathbf{sx}[j-1] > 0$, then the variable contained in the jth column of **x** is included in the regression model.

Constraints:

$\mathbf{sx}[j-1] \geq 0$, for $j = 1, 2, \ldots, \mathbf{m}$;
if **mean** = Nag_MeanInclude, then exactly $\mathbf{ip} - 1$ values of **sx** must be > 0;
if **mean** = 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: $\mathbf{ip} > 0$.

9: **y**[**n**] – const double *Input*

On entry: observations on the dependent variable, y_i, 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 $\mathbf{wt}[i-1] = 0.0$, then the ith 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 $\mathbf{wt}[i-1] \geq 0.0$, 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: **scale** – double * *Input/Output*

On entry: indicates the scale argument for the model, σ^2. If **scale** $= 0.0$, then the scale argument is estimated using the residual mean square.

On exit: if on input **scale** $= 0.0$, then **scale** contains the estimated value of the scale argument, $\hat{\sigma}^2$. If on input **scale** $\neq 0.0$, then **scale** is unchanged on exit.

Constraint: **scale** ≥ 0.0.

13: **ex_power** – double *Input*

On entry: if **link** = Nag_Expo then **ex_power** must contain the power a of the exponential.

If **link** $\neq$ Nag_Expo, **ex_power** is not referenced.

Constraint: If **link** = Nag_Expo, **ex_power** $\neq 0.0$.

14: **rss** – double * *Output*

On exit: the residual sum of squares for the fitted model.

15: **df** – double * *Output*

On exit: the degrees of freedom associated with the residual sum of squares for the fitted model.

16: **b**[**ip**] – double *Output*

On exit: **b**$[i-1]$, $i = 1, 2, \ldots,$ **ip** contains the estimates of the arguments of the generalized linear model, $\hat{\beta}$.

If **mean** = Nag_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 ith positive value in the array **sx**.

If **mean** = Nag_MeanZero, then **b**$[i-1]$ will contain the coefficient of the variable contained in column j of **x**, where **sx**$[j-1]$ is the ith positive value in the array **sx**.

17: **rank** – Integer * *Output*

On exit: 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**.

18: **se**[**ip**] – double *Output*

On exit: the standard errors of the linear arguments.

se$[i-1]$ contains the standard error of the parameter estimate in **b**$[i-1]$, for $i = 1, 2, \ldots,$ **ip**.

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

On exit: the **ip** $\times$ (**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 **b**$[i]$ and the parameter estimate given in **b**$[j]$, $j \geq i$, is stored in **cov**$[j(j+1)/2 + i]$, for $i = 0, 1, \ldots,$ **ip** $- 1$ and $j = i, \ldots,$ **ip** $- 1$.

20: **v**[**n** $\times$ **tdv**] – double *Output*

On exit: auxiliary information on the fitted model.

v$[(i-1) \times$ **tdv**$]$, contains the linear predictor value, η_i, for $i = 1, 2, \ldots, n$.

v$[(i-1) \times$ **tdv** $+ 1]$, contains the fitted value, $\hat{\mu}_i$, for $i = 1, 2, \ldots, n$.

$\mathbf{v}[(i-1)\times\mathbf{tdv}+2]$, is only included for consistency with other functions. $\mathbf{v}[(i-1)\times\mathbf{tdv}+2]=1.0$, for $i=1,2,\ldots,n$.

$\mathbf{v}[(i-1)\times\mathbf{tdv}+3]$, contains the working weight, w_i, for $i=1,2,\ldots,n$.

$\mathbf{v}[(i-1)\times\mathbf{tdv}+4]$, contains the standardized residual, r_i, for $i=1,2,\ldots,n$.

$\mathbf{v}[(i-1)\times\mathbf{tdv}+5]$, contains the leverage, h_i, for $i=1,2,\ldots,n$.

$\mathbf{v}[(i-1)\times\mathbf{tdv}+j-1]$, for $j=7,8,\ldots,\mathbf{ip}+6$, contains the results of the QR decomposition or the singular value decomposition.

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

21: **tdv** – Integer *Input*

On entry: the stride separating matrix column elements in the array **v**.

Constraint: $\mathbf{tdv}\geq\mathbf{ip}+6$.

22: **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 **tol**$\times$ (1.0+current residual sum of squares). This is approximately an absolute precision if the residual sum of squares is small and a relative precision if the residual sum of squares is large.

If $0.0\leq\mathbf{tol}<$ ***machine precision***, then the function will use $10\times$ ***machine precision***.

Constraint: $\mathbf{tol}\geq 0.0$.

23: **max_iter** – Integer *Input*

On entry: the maximum number of iterations for the iterative weighted least squares. If $\mathbf{max\_iter}=0$, then a default value of 10 is used.

Constraint: $\mathbf{max\_iter}\geq 0$.

24: **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. The following values are available:

print_iter ≤ 0
There is no printing.

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.

25: **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.

26: **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 $0.0 \leq$ **eps** $<$ ***machine precision***, then the function will use ***machine precision*** instead.

Constraint: **eps** ≥ 0.0.

27: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **tdv** $= \langle value \rangle$ while **ip** $= \langle value \rangle$. These arguments must satisfy **tdv** $\geq$ **ip** $+ 6$.

On entry, **tdx** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **tdx** $\geq$ **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** $= \langle value \rangle$.
Constraint: **ip** ≥ 1.

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

On entry, **max_iter** must not be less than 0: **max_iter** $= \langle value \rangle$.

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

On entry, **sx**$[\langle value \rangle]$ must not be less than 0: **sx**$[\langle value \rangle] = \langle value \rangle$.

NE_IP_GT_OBSERV

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

NE_IP_INCOMP_SX

Parameter **ip** is incompatible with arguments **mean** and **sx**.

NE_LSQ_ITER_NOT_CONV

The iterative weighted least squares has failed to converge in **max_iter** $= \langle value \rangle$ 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 $\langle string \rangle$ for appending.

NE_NOT_CLOSE_FILE

Cannot close file $\langle string \rangle$.

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** $= \langle value \rangle$.

On entry, **scale** must not be less than 0.0: **scale** $= \langle value \rangle$.

On entry, **tol** must not be less than 0.0: **tol** $= \langle value \rangle$.

On entry, **wt**$[\langle value \rangle]$ must not be less than 0.0: **wt**$[\langle value \rangle] = \langle value \rangle$.

NE_REAL_ENUM_ARG_CONS

On entry, **ex_power** $= 0.0$, **link** $=$ Nag_Expo. These arguments must satisfy **link** $=$ Nag_Expo and **ex_power** $\neq 0.0$.

NE_SVD_NOT_CONV

The singular value decomposition has failed to converge.

NE_VALUE_AT_BOUNDARY_A

A fitted value is at a boundary. This will only occur with **link** $=$ Nag_Expo, Nag_Log or Nag_Reci. This may occur if there are small values of y and the model is not suitable for the data. The model should be reformulated with, perhaps, some observations dropped.

NE_ZERO_DOF_ERROR

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

7 Accuracy

The accuracy is determined by **tol** as described in Section 5. As the residual sum of squares is a function of μ^2 the accuracy of the $\hat{\beta}$'s will depend on the link used and may be of the order $\sqrt{\textbf{tol}}$.

8 Parallelism and Performance

nag_glm_normal (g02gac) is not threaded in any implementation.

9 Further Comments

None.

10 Example

The model:

$$y = \frac{1}{\beta_1 + \beta_2 x} + \epsilon$$

for a sample of five observations.

10.1 Program Text

```
/* nag_glm_normal (g02gac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

#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 tdv, tdx;
  Nag_IncludeMean mean;
  Nag_Link link;
  Nag_Boolean weight;
  char nag_enum_arg[40];
  double df, eps, ex_power, rss, scale, tol;
  double *b = 0, *cov = 0, *offsetptr = 0, *se = 0, *v = 0, *wt = 0;
  double *wtptr, *x = 0, *y = 0;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_glm_normal (g02gac) 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, (unsigned)_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, (unsigned)_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, (unsigned)_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_s("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT " %lf", &n, &m,
          &print_iter, &scale);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT " %lf", &n, &m, &print_iter,
        &scale);
```

```
#endif

  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
#ifdef _WIN32
      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
```

```
    ex_power = 0.0;

  if (!(b = NAG_ALLOC(ip, double)) ||
      !(v = NAG_ALLOC(n * (ip + 6), double)) ||
      !(se = NAG_ALLOC(ip, double)) ||
      !(cov = NAG_ALLOC(ip * (ip + 1) / 2, double)))
  {
    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_normal (g02gac).
   * Fits a generalized linear model with Normal errors
   */
  nag_glm_normal(link, mean, n, x, tdx, m, sx, ip, y, wtptr, offsetptr,
                 &scale, ex_power, &rss, &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_normal (g02gac).\n%s\n", fail.message);
    }
    printf("\nResidual sum of squares = %13.4e\n", rss);
    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_normal (g02gac).\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_normal (g02gac) Example Program Data
Nag_Reci Nag_MeanInclude Nag_FALSE 5 1 0  0.0
1.0 25.0
2.0 10.0
3.0  6.0
4.0  4.0
5.0  3.0
1
```

10.3 Program Results

```
nag_glm_normal (g02gac) Example Program Results

Residual sum of squares =    3.8717e-01
Degrees of freedom = 3.0

       Estimate     Standard error

       -0.0239        0.0028
        0.0638        0.0026

     y   fitted value  Residual   Leverage

   25.0     25.04     -0.0387     0.995
   10.0      9.64      0.3613     0.458
    6.0      5.97      0.0320     0.268
    4.0      4.32     -0.3221     0.167
    3.0      3.39     -0.3878     0.112
```