NAG Library Function Document

nag_glm_est_func (g02gnc)

1 Purpose

nag_glm_est_func (g02gnc) gives the estimate of an estimable function along with its standard error from the results from fitting a generalized linear model.

2 Specification

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

```
void nag_glm_est_func (Integer ip, Integer rank, const double b[],
     const double cov[], const double v[], Integer tdv, const double f[],
     Nag_Boolean *est, double *stat, double *sestat, double *z, double tol,
     NagError *fail)
```

3 Description

nag_glm_est_func (g02gnc) computes the estimates of an estimable function for a general linear regression model which is not of full rank. It is intended for use after a call to nag_glm_normal (g02gac), nag_glm_binomial (g02gbc), nag_glm_poisson (g02gcc) or nag_glm_gamma (g02gdc). An estimable function is a linear combination of the arguments such that it has a unique estimate. For a full rank model all linear combinations of arguments are estimable.

In the case of a model not of full rank the functions use a singular value decomposition (SVD) to find the parameter estimates, $\hat{\beta}$, and their variance-covariance matrix. Given the upper triangular matrix R obtained from the QR decomposition of the independent variables the SVD gives:

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

where D is a k by k diagonal matrix with nonzero diagonal elements, k being the rank of R, and Q_* and P are p by p orthogonal matrices. This leads to a solution:

$$\hat{\beta} = P_1 D^{-1} Q_{*_1}^{\mathrm{T}} c_1$$

P_1 being the first k columns of P, i.e., $P = (P_1 P_0)$; Q_{*_1} being the first k columns of Q_* and c_1 being the first p elements of c.

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

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

as given by nag_glm_normal (g02gac), nag_glm_binomial (g02gbc), nag_glm_poisson (g02gcc) and nag_glm_gamma (g02gdc).

A linear function of the arguments, $F = f^{\mathrm{T}}\beta$, can be tested to see if it is estimable by computing $\zeta = P_0^{\mathrm{T}} f$. If ζ is zero, then the function is estimable, if not, the function is not estimable. In practice $|\zeta|$ is tested against some small quantity η.

Given that F is estimable it can be estimated by $f^{\mathrm{T}}\hat{\beta}$ and its standard error calculated from the variance-covariance matrix of $\hat{\beta}$, C_β, as

$$\mathrm{se}(F) = \sqrt{f^{\mathrm{T}} C_\beta f}$$

Also a z statistic:

$$z = \frac{f^{\mathrm{T}} \hat{\beta}}{\mathrm{se}(F)},$$

can be computed. The distribution of z will be approximately Normal.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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

Searle S R (1971) *Linear Models* Wiley

5 Arguments

1: **ip** – Integer *Input*

On entry: the number of terms in the linear model, p.

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

2: **rank** – Integer *Input*

On entry: the rank of the independent variables, k.

Constraint: $1 \leq \mathbf{rank} \leq \mathbf{ip}$.

3: **b**[**ip**] – const double *Input*

On entry: the **ip** values of the estimates of the arguments of the model, $\hat{\beta}$.

4: **cov**[$\mathbf{ip} \times (\mathbf{ip} + \mathbf{1})/\mathbf{2}$] – const double *Input*

On entry: 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 $j = i, \ldots, \mathbf{ip} - 1$.

5: **v**[$\mathbf{ip} \times \mathbf{tdv}$] – const double *Input*

Note: the (i, j)th element of the matrix V is stored in $\mathbf{v}[(i-1) \times \mathbf{tdv} + j - 1]$.

On entry: **v** as returned by nag_glm_normal (g02gac), nag_glm_binomial (g02gbc), nag_glm_poisson (g02gcc) and nag_glm_gamma (g02gdc).

6: **tdv** – Integer *Input*

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

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

tdv should be as supplied to nag_glm_normal (g02gac), nag_glm_binomial (g02gbc), nag_glm_poisson (g02gcc) or nag_glm_gamma (g02gdc).

7: **f**[**ip**] – const double *Input*

On entry: the linear function to be estimated, f.

8: **est** – Nag_Boolean * *Output*

On exit: **est** indicates if the function was estimable.

est = Nag_TRUE
The function is estimable.

est = Nag_FALSE
The function is not estimable and **stat**, **sestat** and **z** are not set.

9: **stat** – double * *Output*

On exit: if **est** = Nag_TRUE, **stat** contains the estimate of the function, $f^{\mathrm{T}}\hat{\beta}$.

10: **sestat** – double * *Output*

On exit: if **est** = Nag_TRUE, **sestat** contains the standard error of the estimate of the function, $\mathrm{se}(F)$.

11: **z** – double * *Output*

On exit: if **est** = Nag_TRUE, **z** contains the z statistic for the test of the function being equal to zero.

12: **tol** – double *Input*

On entry: **tol** is the tolerance value used in the check for estimability, η.

If **tol** ≤ 0.0, then $\sqrt{\textit{machine precision}}$ is used instead.

13: **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_GT

On entry, **ip** = ⟨*value*⟩ while **rank** = ⟨*value*⟩. These arguments must satisfy **rank** $\leq$ **ip**.

NE_2_INT_ARG_LT

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

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

On entry, **ip** = ⟨*value*⟩.
Constraint: **ip** ≥ 1.

On entry, **rank** = ⟨*value*⟩.
Constraint: **rank** ≥ 1.

NE_RANK_EQ_IP

On entry, **rank** = **ip**. In this case, the boolean variable **est** is returned as Nag_TRUE and all statistics are calculated.

NE_STDES_ZERO

sestat, the standard error of the estimate of the function, $\mathrm{se}(F) = 0.0$; probably due to rounding error or due to incorrectly specified input values of **cov** and **f**.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

nag_glm_est_func (g02gnc) is not threaded in any implementation.

9 Further Comments

The value of estimable functions is independent of the solution chosen from the many possible solutions. While nag_glm_est_func (g02gnc) may be used to estimate functions of the arguments of the model as computed by nag_glm_tran_model (g02gkc), β_c, these must be expressed in terms of the original arguments, β. The relation between the two sets of arguments may not be straightforward.

10 Example

A loglinear model is fitted to a 3 by 5 contingency table by nag_glm_poisson (g02gcc). The model consists of terms for for rows and columns. The table is:

141	67	114	79	39
131	66	143	72	35
36	14	38	28	16

The number of functions to be tested is read in, then the linear functions themselves are read in and tested with nag_glm_est_func (g02gnc). The results of nag_glm_est_func (g02gnc) are printed.

10.1 Program Text

```
/* nag_glm_est_func (g02gnc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

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

#define X(I, J) x[(I) *tdx + J]
int main(void)
{
  Nag_Boolean est;
  Integer exit_status = 0, i, ip, j, m, max_iter, n, nestfn, print_iter, rank;
  Integer *sx = 0, tdv, tdx;
  NagError fail;
  double dev, df, eps, ex_power, sestat, stat, tol, z;
  double *b = 0, *cov = 0, *f = 0, *se = 0, *v = 0, *wtptr, *x = 0;
  double *y = 0;

  INIT_FAIL(fail);

  printf("nag_glm_est_func (g02gnc) Example Program Results\n");
  /* Skip heading in data file */
```

```
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT "", &n, &m, &print_iter);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT "", &n, &m, &print_iter);
#endif

  if (n >= 2 && m >= 1) {
    if (!(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;
  }
  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
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &ip);
#else
  scanf("%" NAG_IFMT "", &ip);
#endif

  if (!(b = NAG_ALLOC(ip, double)) ||
      !(f = NAG_ALLOC(ip, double)) ||
      !(v = NAG_ALLOC(n * (ip + 6), double)) ||
      !(cov = NAG_ALLOC(ip * (ip + 1) / 2, double)) ||
      !(se = NAG_ALLOC(ip, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  tdv = ip + 6;

  /* Set control parameters */
  max_iter = 10;
  tol = 5e-5;
  eps = 1e-6;
  ex_power = 0.0;

  /* Fit Log-linear model using nag_glm_poisson (g02gcc) */
```

```
  /* nag_glm_poisson (g02gcc).
   * Fits a generalized linear model with Poisson errors
   */
  nag_glm_poisson(Nag_Log, Nag_MeanInclude, n, x, tdx,
                  m, sx, ip, y, wtptr, (double *) 0, 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) {
    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");
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &nestfn);
#else
    scanf("%" NAG_IFMT "", &nestfn);
#endif
    for (i = 1; i <= nestfn; ++i) {
      for (j = 0; j < ip; ++j)
#ifdef _WIN32
        scanf_s("%lf", &f[j]);
#else
        scanf("%lf", &f[j]);
#endif

      /* nag_glm_est_func (g02gnc).
       * Estimable function and the standard error of a
       * generalized linear model
       */
      nag_glm_est_func(ip, rank, b, cov, v,
                       tdv, f, &est, &stat, &sestat, &z, tol, &fail);

      if (fail.code != NE_NOERROR && fail.code != NE_RANK_EQ_IP) {
        printf("Error from nag_glm_est_func (g02gnc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
      }

      printf("\n");
      printf("Function %" NAG_IFMT "\n\n", i);
      for (j = 0; j < ip; ++j)
        printf("%8.2f%c", f[j], (j % 5 == 4 || j == ip - 1) ? '\n' : ' ');
      printf("\n");
      if (est)
        printf("stat = %10.4f  sestat = %10.4f   z = %10.4f\n",
               stat, sestat, z);
      else
        printf("Function not estimable\n");
    }
  }
  else {
    printf("Error from nag_glm_poisson (g02gcc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(sx);
  NAG_FREE(b);
  NAG_FREE(f);
  NAG_FREE(v);
  NAG_FREE(cov);
  NAG_FREE(se);
  return exit_status;
}
```

10.2 Program Data

```
nag_glm_est_func (g02gnc) Example Program Data
15 8 0
1.0 0.0 0.0 1.0 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  67.
1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 114.
1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0  79.
1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0  39.
0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 131.
0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0  66.
0.0 1.0 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 1.0 0.0  72.
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0  35.
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0  36.
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0  14.
0.0 0.0 1.0 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  28.
0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0  16.
1   1   1   1   1   1   1   1   9
3
1.0  1.0  0.0 0.0  1.0 0.0 0.0 0.0 0.0
0.0  1.0 -1.0 0.0  0.0 0.0 0.0 0.0 0.0
0.0  1.0  0.0 0.0  0.0 0.0 0.0 0.0 0.0
```

10.3 Program Results

```
nag_glm_est_func (g02gnc) Example Program Results

Deviance =    9.0379e+00
Degrees of freedom = 8.0

       Estimate     Standard error

        2.5977        0.0258
        1.2619        0.0438
        1.2777        0.0436
        0.0580        0.0668
        1.0307        0.0551
        0.2910        0.0732
        0.9876        0.0559
        0.4880        0.0675
       -0.1996        0.0904


Function 1

    1.00     1.00     0.00     0.00     1.00
    0.00     0.00     0.00     0.00

stat =     4.8903  sestat =     0.0674   z =    72.5934

Function 2

    0.00     1.00    -1.00     0.00     0.00
    0.00     0.00     0.00     0.00

stat =    -0.0158  sestat =     0.0672   z =    -0.2350

Function 3

    0.00     1.00     0.00     0.00     0.00
    0.00     0.00     0.00     0.00

Function not estimable
```