NAG Library Function Document

nag_rank_regsn_censored (g08rbc)

1 Purpose

nag_rank_regsn_censored (g08rbc) calculates the parameter estimates, score statistics and their variance-covariance matrices for the linear model using a likelihood based on the ranks of the observations when some of the observations may be right-censored.

2 Specification

```
#include <nag.h>
#include <nagg08.h>
void nag_rank_regsn_censored (Nag_OrderType order, Integer ns,
     const Integer nv[], const double y[], Integer p, const double x[],
     Integer pdx, const Integer icen[], double gamma, Integer nmax,
     double tol, double prvr[], Integer pdprvr, Integer irank[],
     double zin[], double eta[], double vapvec[], double parest[],
     NagError *fail)
```

3 Description

Analysis of data can be made by replacing observations by their ranks. The analysis produces inference for the regression model where the location parameters of the observations, θ_i, for $i = 1, 2, \ldots, n$, are related by $\theta = X\beta$. Here X is an n by p matrix of explanatory variables and β is a vector of p unknown regression parameters. The observations are replaced by their ranks and an approximation, based on Taylor's series expansion, made to the rank marginal likelihood. For details of the approximation see Pettitt (1982).

An observation is said to be right-censored if we can only observe Y_j^* with $Y_j^* \leq Y_j$. We rank censored and uncensored observations as follows. Suppose we can observe Y_j, for $j = 1, 2, \ldots, n$, directly but Y_j^*, for $j = n+1, \ldots, q$ and $n \leq q$, are censored on the right. We define the rank r_j of Y_j, for $j = 1, 2, \ldots, n$, in the usual way; r_j equals i if and only if Y_j is the ith smallest amongst the $Y_1, Y_2, \ldots, Y_n$. The right-censored Y_j^*, for $j = n+1, n+2, \ldots, q$, has rank r_j if and only if Y_j^* lies in the interval $\left[Y_{(r_j)}, Y_{(r_j+1)}\right]$, with $Y_0 = -\infty$, $Y_{(n+1)} = +\infty$ and $Y_{(1)} < \cdots < Y_{(n)}$ the ordered Y_j, for $j = 1, 2, \ldots, n$.

The distribution of the Y is assumed to be of the following form. Let $F_L(y) = e^y/(1+e^y)$, the logistic distribution function, and consider the distribution function $F_\gamma(y)$ defined by $1 - F_\gamma = [1 - F_L(y)]^{1/\gamma}$. This distribution function can be thought of as either the distribution function of the minimum, $X_{1,\gamma}$, of a random sample of size γ^{-1} from the logistic distribution, or as the $F_\gamma(y - \log\gamma)$ being the distribution function of a random variable having the F-distribution with 2 and $2\gamma^{-1}$ degrees of freedom. This family of generalized logistic distribution functions $\left[F_\gamma(.); 0 \leq \gamma < \infty\right]$ naturally links the symmetric logistic distribution $(\gamma = 1)$ with the skew extreme value distribution ($\lim\gamma \to 0$) and with the limiting negative exponential distribution ($\lim\gamma \to \infty$). For this family explicit results are available for right-censored data. See Pettitt (1983) for details.

Let l_R denote the logarithm of the rank marginal likelihood of the observations and define the $q \times 1$ vector a by $a = l_R'(\theta = 0)$, and let the q by q diagonal matrix B and q by q symmetric matrix A be given by $B - A = -l_R''(\theta = 0)$. Then various statistics can be found from the analysis.

(a) The score statistic $X^{\mathrm{T}}a$. This statistic is used to test the hypothesis $H_0 : \beta = 0$ (see (e)).

(b) The estimated variance-covariance matrix of the score statistic in (a).

(c) The estimate $\hat{\beta}_R = MX^{\mathrm{T}}a$.

(d) The estimated variance-covariance matrix $M = (X^{\mathrm{T}}(B-A)X)^{-1}$ of the estimate $\hat{\beta}_R$.

(e) The χ^2 statistic $Q = \hat{\beta}_R M^{-1}\ \hat{\beta}_r = a^{\mathrm{T}}X(X^{\mathrm{T}}(B-A)X)^{-1}X^{\mathrm{T}}a$, used to test $H_0 : \beta = 0$. Under H_0, Q has an approximate χ^2-distribution with p degrees of freedom.

(f) The standard errors $M_{ii}^{1/2}$ of the estimates given in (c).

(g) Approximate z-statistics, i.e., $Z_i = \hat{\beta}_{R_i}/se\left(\hat{\beta}_{R_i}\right)$ for testing $H_0 : \beta_i = 0$. For $i = 1, 2, \ldots, n$, Z_i has an approximate $N(0,1)$ distribution.

In many situations, more than one sample of observations will be available. In this case we assume the model,

$$h_k(Y_k) = X_k^{\mathrm{T}}\beta + e_k, \quad k = 1, 2, \ldots, \mathbf{ns},$$

where **ns** is the number of samples. In an obvious manner, Y_k and X_k are the vector of observations and the design matrix for the kth sample respectively. Note that the arbitrary transformation h_k can be assumed different for each sample since observations are ranked within the sample.

The earlier analysis can be extended to give a combined estimate of β as $\hat{\beta} = Dd$, where

$$D^{-1} = \sum_{k=1}^{\mathbf{ns}} X^{\mathrm{T}}(B_k - A_k)X_k$$

and

$$d = \sum_{k=1}^{\mathbf{ns}} X_k^{\mathrm{T}} a_k,$$

with a_k, B_k and A_k defined as a, B and A above but for the kth sample.

The remaining statistics are calculated as for the one sample case.

4 References

Kalbfleisch J D and Prentice R L (1980) *The Statistical Analysis of Failure Time Data* Wiley

Pettitt A N (1982) Inference for the linear model using a likelihood based on ranks *J. Roy. Statist. Soc. Ser. B* **44** 234–243

Pettitt A N (1983) Approximate methods using ranks for regression with censored data *Biometrika* **70** 121–132

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **ns** – Integer *Input*

On entry: the number of samples.

Constraint: **ns** ≥ 1.

3: **nv**[**ns**] – const Integer *Input*

On entry: the number of observations in the ith sample, for $i = 1, 2, \ldots,$ **ns**.

Constraint: $\mathbf{nv}[i-1] \geq 1$, for $i = 1, 2, \ldots,$ **ns**.

4: $\mathbf{y}[dim]$ – const double *Input*

Note: the dimension, *dim*, of the array **y** must be at least $\left(\sum_{i=1}^{\mathbf{ns}} \mathbf{nv}[i-1]\right)$.

On entry: the observations in each sample. Specifically, $\mathbf{y}[\sum_{k=1}^{i-1} \mathbf{nv}[k-1] + j - 1]$ must contain the jth observation in the ith sample.

5: **p** – Integer *Input*

On entry: the number of parameters to be fitted.

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

6: $\mathbf{x}[dim]$ – const double *Input*

Note: the dimension, *dim*, of the array **x** must be at least

$\max(1, \mathbf{pdx} \times \mathbf{p})$ when **order** = Nag_ColMajor;
$\max\left(1, \left(\sum_{i=1}^{\mathbf{ns}} \mathbf{nv}[i-1]\right) \times \mathbf{pdx}\right)$ when **order** = Nag_RowMajor.

Where $\mathbf{X}(i, j)$ appears in this document, it refers to the array element

$\mathbf{x}[(j-1) \times \mathbf{pdx} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1]$ when **order** = Nag_RowMajor.

On entry: the design matrices for each sample. Specifically, $\mathbf{X}\left(\sum_{k=1}^{i-1} \mathbf{nv}[k-1] + j, l\right)$ must contain the value of the lth explanatory variable for the jth observations in the ith sample.

Constraint: **x** must not contain a column with all elements equal.

7: **pdx** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **x**.

Constraints:

if **order** = Nag_ColMajor, $\mathbf{pdx} \geq \left(\sum_{i=1}^{\mathbf{ns}} \mathbf{nv}[i-1]\right)$;
if **order** = Nag_RowMajor, $\mathbf{pdx} \geq \mathbf{p}$.

8: **icen**[dim] – const Integer *Input*

Note: the dimension, *dim*, of the array **icen** must be at least $\left(\sum_{i=1}^{\mathbf{ns}} \mathbf{nv}[i-1]\right)$.

On entry: defines the censoring variable for the observations in **y**.

$\mathbf{icen}[i-1] = 0$
If $\mathbf{y}[i-1]$ is uncensored.

icen$[i-1] = 1$
If **y**$[i-1]$ is censored.

Constraint: **icen**$[i-1] = 0$ or 1, for $i = 1, 2, \ldots, \left(\sum_{i=1}^{\mathbf{ns}} \mathbf{nv}[i-1]\right)$.

9: **gamma** – double *Input*

On entry: the value of the parameter defining the generalized logistic distribution. For **gamma** ≤ 0.0001, the limiting extreme value distribution is assumed.

Constraint: **gamma** ≥ 0.0.

10: **nmax** – Integer *Input*

On entry: the value of the largest sample size.

Constraint: **nmax** $= \max\limits_{1 \leq i \leq \mathbf{ns}} (\mathbf{nv}[i-1])$ and **nmax** $> \mathbf{p}$.

11: **tol** – double *Input*

On entry: the tolerance for judging whether two observations are tied. Thus, observations Y_i and Y_j are adjudged to be tied if $\left|Y_i - Y_j\right| <$ **tol**.

Constraint: **tol** > 0.0.

12: **prvr**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **prvr** must be at least

max$(1, \mathbf{pdprvr} \times \mathbf{p})$ when **order** $=$ Nag_ColMajor;
max$(1, \mathbf{p} + 1 \times \mathbf{pdprvr})$ when **order** $=$ Nag_RowMajor.

Where **PRVR**(i, j) appears in this document, it refers to the array element

prvr$[(j-1) \times \mathbf{pdprvr} + i - 1]$ when **order** $=$ Nag_ColMajor;
prvr$[(i-1) \times \mathbf{pdprvr} + j - 1]$ when **order** $=$ Nag_RowMajor.

On exit: the variance-covariance matrices of the score statistics and the parameter estimates, the former being stored in the upper triangle and the latter in the lower triangle. Thus for $1 \leq i \leq j \leq \mathbf{p}$, **PRVR**$(i, j)$ contains an estimate of the covariance between the ith and jth score statistics. For $1 \leq j \leq i \leq \mathbf{p} - 1$, **PRVR**$(i + 1, j)$ contains an estimate of the covariance between the ith and jth parameter estimates.

13: **pdprvr** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **prvr**.

Constraints:

if **order** $=$ Nag_ColMajor, **pdprvr** $\geq \mathbf{p} + 1$;
if **order** $=$ Nag_RowMajor, **pdprvr** $\geq \mathbf{p}$.

14: **irank**[**nmax**] – Integer *Output*

On exit: for the one sample case, **irank** contains the ranks of the observations.

15: **zin**[**nmax**] – double *Output*

On exit: for the one sample case, **zin** contains the expected values of the function $g(.)$ of the order statistics.

16: **eta**[**nmax**] – double *Output*

On exit: for the one sample case, **eta** contains the expected values of the function $g\prime(.)$ of the order statistics.

17: **vapvec**[**nmax** × (**nmax** + **1**)/**2**] – double *Output*

On exit: for the one sample case, **vapvec** contains the upper triangle of the variance-covariance matrix of the function $g(.)$ of the order statistics stored column-wise.

18: **parest**[**4** × **p** + **1**] – double *Output*

On exit: the statistics calculated by the function.

The first **p** components of **parest** contain the score statistics.

The next **p** elements contain the parameter estimates.

parest[2 × **p**] contains the value of the χ^2 statistic.

The next **p** elements of **parest** contain the standard errors of the parameter estimates.

Finally, the remaining **p** elements of **parest** contain the z-statistics.

19: **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_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

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

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

On entry, **pdprvr** = ⟨*value*⟩.
Constraint: **pdprvr** > 0.

On entry, **pdx** = ⟨*value*⟩.
Constraint: **pdx** > 0.

NE_INT_2

On entry, **nmax** = ⟨*value*⟩ and **p** = ⟨*value*⟩.
Constraint: **nmax** > **p**.

On entry, **pdprvr** = ⟨*value*⟩ and **p** = ⟨*value*⟩.
Constraint: **pdprvr** ≥ **p**.

On entry, **pdprvr** = ⟨*value*⟩ and **p** = ⟨*value*⟩.
Constraint: **pdprvr** ≥ **p** + 1.

On entry, **pdx** $= \langle value \rangle$ and **p** $= \langle value \rangle$.
Constraint: **pdx** $\geq$ **p**.

On entry, **pdx** $= \langle value \rangle$ and sum **nv**$[i-1] = \langle value \rangle$.
Constraint: **pdx** $\geq$ the sum of **nv**$[i-1]$.

NE_INT_ARRAY

On entry, **nv**$[\langle value \rangle] = \langle value \rangle$.
Constraint: **nv**$[i-1] \geq 1$, for $i = 1, 2, \ldots,$ **ns**.

NE_INT_ARRAY_ELEM_CONS

On entry $M = \langle value \rangle$.
Constraint: M elements of array **icen** $= 0$ or 1.

On entry $M = \langle value \rangle$.
Constraint: M elements of array **nv** > 0.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_MAT_ILL_DEFINED

The matrix $X^{\mathrm{T}}(B-A)X$ is either singular or non positive definite.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

NE_OBSERVATIONS

All the observations were adjudged to be tied.

NE_REAL

On entry, **gamma** $= \langle value \rangle$.
Constraint: **gamma** ≥ 0.0.

On entry, **tol** $= \langle value \rangle$.
Constraint: **tol** > 0.0.

NE_REAL_ARRAY_ELEM_CONS

On entry, all elements in column $\langle value \rangle$ of **x** are equal to $\langle value \rangle$.

NE_SAMPLE

The largest sample size is $\langle value \rangle$ which is not equal to **nmax**, **nmax** $= \langle value \rangle$.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

nag_rank_regsn_censored (g08rbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_rank_regsn_censored (g08rbc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by nag_rank_regsn_censored (g08rbc) depends on the number of samples, the total number of observations and the number of parameters fitted.

In extreme cases the parameter estimates for certain models can be infinite, although this is unlikely to occur in practice. See Pettitt (1982) for further details.

10 Example

This example fits a regression model to a single sample of 40 observations using just one explanatory variable.

10.1 Program Text

```
/* nag_rank_regsn_censored (g08rbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg08.h>

int main(void)
{

  /* Scalars */
  double gamma, tol;
  Integer exit_status, i, p, j, nmax, ns, nsum;
  Integer pdx, pdprvr;
  NagError fail;
  Nag_OrderType order;

  /* Arrays */
  double *eta = 0, *parest = 0, *prvr = 0, *vapvec = 0, *x = 0;
  double *y = 0, *zin = 0;
  Integer *icen = 0, *irank = 0, *iwa = 0, *nv = 0;

#ifdef NAG_COLUMN_MAJOR
#define X(I, J)      x[(J-1)*pdx + I - 1]
#define PRVR(I, J) prvr[(J-1)*pdprvr + I - 1]
  order = Nag_ColMajor;
#else
#define X(I, J)      x[(I-1)*pdx + J - 1]
#define PRVR(I, J) prvr[(I-1)*pdprvr + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_rank_regsn_censored (g08rbc) Example Program Results\n");
```

```
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read number of samples, number of parameters to be fitted, */
  /* distribution power parameter and tolerance criterion for ties. */

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%lf%lf%*[^\n] ", &ns, &p, &gamma, &tol);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%lf%lf%*[^\n] ", &ns, &p, &gamma, &tol);
#endif
  printf("\n");

  /* Allocate memory to nv only */
  if (!(nv = NAG_ALLOC(ns, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  printf("Number of samples =%2" NAG_IFMT "\n", ns);
  printf("Number of parameters fitted =%2" NAG_IFMT "\n", p);
  printf("Distribution power parameter =%10.5f\n", gamma);

  printf("Tolerance for ties =%10.5f\n", tol);

  printf("\n");
  /* Read the number of observations in each sample */

  for (i = 1; i <= ns; ++i)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &nv[i - 1]);
#else
    scanf("%" NAG_IFMT "", &nv[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  nmax = 0;
  nsum = 0;
  for (i = 1; i <= ns; ++i) {
    nsum += nv[i - 1];
    nmax = MAX(nmax, nv[i - 1]);
  }

  /* Allocate memory */
  if (!(eta = NAG_ALLOC(nmax, double)) ||
      !(parest = NAG_ALLOC(4 * p + 1, double)) ||
      !(prvr = NAG_ALLOC(7 * 6, double)) ||
      !(vapvec = NAG_ALLOC(nmax * (nmax + 1) / 2, double)) ||
      !(x = NAG_ALLOC(nsum * p, double)) ||
      !(y = NAG_ALLOC(nsum, double)) ||
      !(zin = NAG_ALLOC(nmax, double)) ||
      !(icen = NAG_ALLOC(nsum, Integer)) ||
      !(irank = NAG_ALLOC(nmax, Integer)) || !(iwa = NAG_ALLOC(400, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
#ifdef NAG_COLUMN_MAJOR
  pdx = nsum;
  pdprvr = p + 1;
```

```
#else
  pdx = p;
  pdprvr = p;
#endif

  /* Read in observations, design matrix and censoring variable */

  for (i = 1; i <= nsum; ++i) {
#ifdef _WIN32
    scanf_s("%lf", &y[i - 1]);
#else
    scanf("%lf", &y[i - 1]);
#endif

    for (j = 1; j <= p; ++j) {
#ifdef _WIN32
      scanf_s("%lf", &X(i, j));
#else
      scanf("%lf", &X(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &icen[i - 1]);
#else
    scanf("%" NAG_IFMT "", &icen[i - 1]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* nag_rank_regsn_censored (g08rbc).
   * Regression using ranks, right-censored data
   */
  nag_rank_regsn_censored(order, ns, nv, y, p, x, pdx, icen, gamma,
                          nmax, tol, prvr, pdprvr, irank, zin, eta, vapvec,
                          parest, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rank_regsn_censored (g08rbc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("Score statistic\n");

  for (i = 1; i <= p; ++i)
    printf("%9.3f\n", parest[i - 1]);
  printf("\n");

  printf("Covariance matrix of score statistic\n");
  for (j = 1; j <= p; ++j) {
    for (i = 1; i <= j; ++i)
      printf("%9.3f\n", PRVR(i, j));
    printf("\n");
  }

  printf("Parameter estimates\n");
  for (i = 1; i <= p; ++i)
    printf("%9.3f\n", parest[p + i - 1]);
  printf("\n");
  printf("Covariance matrix of parameter estimates\n");
  for (i = 1; i <= p; ++i) {
    for (j = 1; j <= i; ++j)
      printf("%9.3f\n", PRVR(i + 1, j));
    printf("\n");
  }

  printf("Chi-squared statistic =%9.3f with%2" NAG_IFMT " d.f.\n",
```

```
          parest[p * 2], p);
  printf("\n");

  printf("Standard errors of estimates and\n");
  printf("approximate z-statistics\n");

  for (i = 1; i <= p; ++i)
    printf("%9.3f%14.3f\n", parest[2 * p + 1 + i - 1],
           parest[p * 3 + 1 + i - 1]);
END:
  NAG_FREE(eta);
  NAG_FREE(parest);
  NAG_FREE(prvr);
  NAG_FREE(vapvec);
  NAG_FREE(x);
  NAG_FREE(y);
  NAG_FREE(zin);
  NAG_FREE(icen);
  NAG_FREE(irank);
  NAG_FREE(iwa);
  NAG_FREE(nv);

  return exit_status;
}
```

10.2 Program Data

```
nag_rank_regsn_censored (g08rbc) Example Program Data
 1 1 0.00001 0.00001
 40
 143.0 0.0 0 164.0 0.0 0 188.0 0.0 0 188.0 0.0 0 190.0 0.0 0
 192.0 0.0 0 206.0 0.0 0 209.0 0.0 0 213.0 0.0 0 216.0 0.0 0
 220.0 0.0 0 227.0 0.0 0 230.0 0.0 0 234.0 0.0 0 246.0 0.0 0
 265.0 0.0 0 304.0 0.0 0 216.0 0.0 1 244.0 0.0 1 142.0 1.0 0
 156.0 1.0 0 163.0 1.0 0 198.0 1.0 0 205.0 1.0 0 232.0 1.0 0
 232.0 1.0 0 233.0 1.0 0 233.0 1.0 0 233.0 1.0 0 233.0 1.0 0
 239.0 1.0 0 240.0 1.0 0 261.0 1.0 0 280.0 1.0 0 280.0 1.0 0
 296.0 1.0 0 296.0 1.0 0 323.0 1.0 0 204.0 1.0 1 344.0 1.0 1
```

10.3 Program Results

```
nag_rank_regsn_censored (g08rbc) Example Program Results

Number of samples = 1
Number of parameters fitted = 1
Distribution power parameter =   0.00001
Tolerance for ties =   0.00001

Score statistic
    4.584

Covariance matrix of score statistic
    7.653

Parameter estimates
    0.599

Covariance matrix of parameter estimates
    0.131

Chi-squared statistic =    2.746 with 1 d.f.

Standard errors of estimates and
approximate z-statistics
    0.361         1.657
```