

## 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,  $\theta_i$ , for  $i = 1, 2, \dots, n$ , are related by  $\theta = X\beta$ . Here  $X$  is an  $n$  by  $p$  matrix of explanatory variables and  $\beta$  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  $j = 1, 2, \dots, n$ , directly but  $Y_j^*$ , for  $j = n + 1, \dots, q$  and  $n \leq q$ , are censored on the right. We define the rank  $r_j$  of  $Y_j$ , for  $j = 1, 2, \dots, n$ , in the usual way;  $r_j$  equals  $i$  if and only if  $Y_j$  is the  $i$ th smallest amongst the  $Y_1, Y_2, \dots, Y_n$ . The right-censored  $Y_j^*$ , for  $j = n + 1, n + 2, \dots, q$ , has rank  $r_j$  if and only if  $Y_j^*$  lies in the interval  $[Y_{(r_j)}, Y_{(r_j+1)}]$ , with  $Y_0 = -\infty$ ,  $Y_{(n+1)} = +\infty$  and  $Y_{(1)} < \dots < Y_{(n)}$  the ordered  $Y_j$ , for  $j = 1, 2, \dots, n$ .

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  $\gamma^{-1}$  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  $[F_\gamma(\cdot); 0 \leq \gamma < \infty]$  naturally links the symmetric logistic distribution ( $\gamma = 1$ ) with the skew extreme value distribution ( $\lim \gamma \rightarrow 0$ ) and with the limiting negative exponential distribution ( $\lim \gamma \rightarrow \infty$ ). 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.

- The score statistic  $X^T a$ . This statistic is used to test the hypothesis  $H_0 : \beta = 0$  (see (e)).
- The estimated variance-covariance matrix of the score statistic in (a).
- The estimate  $\hat{\beta}_R = M X^T a$ .

- (d) The estimated variance-covariance matrix  $M = (X^T(B - A)X)^{-1}$  of the estimate  $\hat{\beta}_R$ .
- (e) The  $\chi^2$  statistic  $Q = \hat{\beta}_R M^{-1} \hat{\beta}_R = a^T X (X^T(B - A)X)^{-1} X^T a$ , used to test  $H_0 : \beta = 0$ . Under  $H_0$ ,  $Q$  has an approximate  $\chi^2$ -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.,  $Z_i = \hat{\beta}_{R_i} / se(\hat{\beta}_{R_i})$  for testing  $H_0 : \beta_i = 0$ . For  $i = 1, 2, \dots, n$ ,  $Z_i$  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,

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

where  $\mathbf{ns}$  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  $k$ th 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  $\beta$  as  $\hat{\beta} = Dd$ , where

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

and

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

with  $a_k$ ,  $B_k$  and  $A_k$  defined as  $a$ ,  $B$  and  $A$  above but for the  $k$ th 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 3.2.1.3 in the Essential Introduction 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**  $\geq$  1.

3: **nv[ns]** – const Integer *Input*  
*On entry:* the number of observations in the  $i$ th sample, for  $i = 1, 2, \dots, \mathbf{ns}$ .  
*Constraint:*  $\mathbf{nv}[i - 1] \geq 1$ , for  $i = 1, 2, \dots, \mathbf{ns}$ .

4: **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  $j$ th observation in the  $i$ th sample.

5: **p** – Integer *Input*  
*On entry:* the number of parameters to be fitted.  
*Constraint:*  $\mathbf{p} \geq 1$ .

6: **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

$$\begin{aligned} &\mathbf{x}[(j - 1) \times \mathbf{pdx} + i - 1] \text{ when } \mathbf{order} = \text{Nag\_ColMajor}; \\ &\mathbf{x}[(i - 1) \times \mathbf{pdx} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}. \end{aligned}$$

*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  $l$ th explanatory variable for the  $j$ th observations in the  $i$ th 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:*

$$\begin{aligned} &\text{if } \mathbf{order} = \text{Nag\_ColMajor}, \mathbf{pdx} \geq \left(\sum_{i=1}^{\mathbf{ns}} \mathbf{nv}[i - 1]\right); \\ &\text{if } \mathbf{order} = \text{Nag\_RowMajor}, \mathbf{pdx} \geq \mathbf{p}. \end{aligned}$$

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**.

$$\begin{aligned} &\mathbf{icen}[i - 1] = 0 \\ &\text{If } \mathbf{y}[i - 1] \text{ is uncensored.} \end{aligned}$$

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

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

9: **gamma** – double *Input*

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

*Constraint:* **gamma**  $\geq 0.0$ .

10: **nmax** – Integer *Input*

*On entry:* the value of the largest sample size.

*Constraint:* **nmax** =  $\max_{1 \leq i \leq \mathbf{ns}} (\mathbf{nv}[i - 1])$  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  $|Y_i - Y_j| < \mathbf{tol}$ .

*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)  $\times$  **pdprvr** + *i* - 1] when **order** = Nag\_ColMajor;  
**prvr**[(*i* - 1)  $\times$  **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 *i*th and *j*th score statistics. For  $1 \leq j \leq i \leq \mathbf{p} - 1$ , **PRVR**(*i* + 1, *j*) contains an estimate of the covariance between the *i*th and *j*th 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  $g(\cdot)$  of the order statistics.

- 16: **eta**[**nmax**] – double *Output*  
*On exit:* for the one sample case, **eta** contains the expected values of the function  $g(\cdot)$  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  $g(\cdot)$  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  $\chi^2$  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 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.  
 See Section 3.2.1.2 in the Essential Introduction for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_INT

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

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

On entry, **pdprvr** =  $\langle value \rangle$ .  
 Constraint: **pdprvr** > 0.

On entry, **pdx** =  $\langle value \rangle$ .  
 Constraint: **pdx** > 0.

### NE\_INT\_2

On entry, **nmax** =  $\langle value \rangle$  and **p** =  $\langle value \rangle$ .  
 Constraint: **nmax** > **p**.

On entry, **pdprvr** =  $\langle value \rangle$  and **p** =  $\langle value \rangle$ .  
 Constraint: **pdprvr** ≥ **p**.

On entry, **pdprvr** =  $\langle value \rangle$  and **p** =  $\langle value \rangle$ .  
 Constraint: **pdprvr** ≥ **p** + 1.

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

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

**NE\_INT\_ARRAY**

On entry,  $\mathbf{nv}[\langle value \rangle] = \langle value \rangle$ .  
 Constraint:  $\mathbf{nv}[i - 1] \geq 1$ , for  $i = 1, 2, \dots, \mathbf{ns}$ .

**NE\_INT\_ARRAY\_ELEM\_CONS**

On entry  $M = \langle value \rangle$ .  
 Constraint:  $M$  elements of array  $\mathbf{icen} = 0$  or  $1$ .  
 On entry  $M = \langle value \rangle$ .  
 Constraint:  $M$  elements of array  $\mathbf{nv} > 0$ .

**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 the Essential Introduction for further information.

**NE\_MAT\_ILL\_DEFINED**

The matrix  $X^T(B - A)X$  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 the Essential Introduction for further information.

**NE\_OBSERVATIONS**

All the observations were adjudged to be tied.

**NE\_REAL**

On entry,  $\mathbf{gamma} = \langle value \rangle$ .  
 Constraint:  $\mathbf{gamma} \geq 0.0$ .  
 On entry,  $\mathbf{tol} = \langle value \rangle$ .  
 Constraint:  $\mathbf{tol} > 0.0$ .

**NE\_REAL\_ARRAY\_ELEM\_CONS**

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

**NE\_SAMPLE**

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

**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' Note for 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.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#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
#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

```

---