Linear Regression: nagdmc_linear_reg

Purpose

nagdmc_linear_reg computes a regression model with p parameters.

Declaration

```
#include <nagdmc.h>
void nagdmc_linear_reg(long rec1, long nvar, long nrec, long dblk, double data[],
                       void (*dfun)(long, long, double [], char *, int *), char *comm,
                       long chunksize, long nxvar, long xvar[], long yvar,
                       long iwts, double *R2, double *rms, long *df, double b[],
                       double se[], double cov[], double model[], double eps,
                       int *info);
```

Parameters

1: **rec1** – `long` *Input*

On entry: the index in the data of the first data record used in the analysis.

Constraint: **rec1** ≥ 0.

2: **nvar** – `long` *Input*

On entry: the number of variables in the data.

Constraint: **nvar** > 1.

3: **nrec** – `long` *Input*

On entry: the number of consecutive records, beginning at **rec1**, used in the analysis.

Constraint: **nrec** > 1.

4: **dblk** – `long` *Input*

On entry: the total number of records in the data block.

Constraint: **dblk** $\geq$ **rec1** + **nrec**.

5: **data**[**dblk** $*$ **nvar**] – `double` *Input*

On entry: the data values for the jth variable (for $j = 0, 1, \ldots,$ **nvar**-1) are stored in **data**[$i*$**nvar**$+j$], for $i = 0, 1, \ldots,$ **dblk** $- 1$. When the data function is used, **data** is not referenced.

6: **dfun** – function supplied by user *External Procedure*

On entry: the pointer to a data function supplied by the user.

Constraint: if **dfun** is a valid pointer, **data** must be 0.

The specification of **dfun** is:

```
void dfun(long irec, long chunksize, double x[], char *comm, int *ierr)
```

1: **irec** – `long` *Input*

On entry: the index in the data of the first record returned.

2: **chunksize** – `long` *Input*

On entry: the number of consecutive records returned.

3: **x**[**chunksize**$*$**nvar**] – `double` *Output*

On exit: data values for the jth variable (for $j = 0, 1, \ldots,$ **nvar** $- 1$) must be returned in **x**[$i *$ **nvar** $+ j$], for $i = 0, 1, \ldots,$ **chunksize** $- 1$.

4: **comm** – `char *` *Input*

On entry: a communication parameter allowing additional information to be passed to **dfun**. This parameter is passed 'as is' through the calling function.

5: **ierr** – `int *` *Output*

On exit: if the value pointed to by **ierr** on return is greater than 100, the NAG DMC function will terminate immediately and **info** will point to this value.

7: **comm** – `char *` *Input*

On entry: a communication parameter allowing additional information to be passed to **dfun**. This parameter is passed 'as is' through the calling function.

8: **chunksize** – `long` *Input*

On entry: if the data function is used, the function inputs no more than **chunksize** data records at a time; otherwise **chunksize** is not referenced.

Constraint: if **dfun** $\neq 0$, **chunksize** ≥ 1.

9: **nxvar** – `long` *Input*

On entry: the number of independent variables. If **nxvar** $= 0$ then all variables in the data, excluding **yvar** and (if **iwts** ≥ 0) **iwts**, are treated as independent variables.

Constraint: $0 \leq$ **nxvar** $<$ **nvar**.

10: **xvar**[**nxvar**] – `long` *Input*

On entry: the indices indicating the position in **data** in which values of the independent variables are stored. If **nxvar** $= 0$ then **xvar** must be 0, and the indices of independent variables are given by $j = 0, 1, \ldots,$ **nvar** $- 1$; $j \neq$ **yvar** and $j \neq$ **iwts**.

Constraints: if **nxvar** > 0, $0 \leq$ **xvar**$[i] <$ **nvar**, for $i = 0, 1, \ldots,$ **nxvar** $- 1$; otherwise **xvar** must be 0.

11: **yvar** – `long` *Input*

On entry: the index in **data** in which values of the dependent variable are stored.

Constraints: $0 \leq$ **yvar** $<$ **nvar**; if **nxvar** > 0, **yvar** $\neq$ **xvar**$[i]$, for $i = 0, 1, \ldots,$ **nxvar** $- 1$.

12: **iwts** – `long` *Input*

On entry: if **iwts** $= -1$, no weights are used; otherwise **iwts** is the index in **data** in which the weights are stored.

Constraints: $-1 \leq$ **iwts** $<$ **nvar**; **iwts** $\neq$ **yvar**; and if **nxvar** > 0, **iwts** $\neq$ **xvar**$[i]$, for $i = 0, 1, \ldots,$ **nxvar** $- 1$.

13: **R2** – `double *` *Output*

On exit: the R^2-value for the fitted model.

14: **rms** – `double *` *Output*

On exit: the residual mean square for the fitted model.

15: **df** – `long *` *Output*

On exit: the degrees of freedom for the residual mean square.

16: **b**$[p]$ – `double` *Output*

On exit: the parameter estimates. **b**$[0]$ is the mean parameter. **b**$[i]$ is the coefficient of the ith variable included in the model, for $i = 1, 2, \ldots, p - 1$. If **nxvar** > 0 then the order the independent variables are added to the model is defined by **xvar**, otherwise the order is defined by indices in the data.

17: **se**$[p]$ – `double` *Output*

On exit: the standard errors of the parameters in **b**.

18: **cov**$[p * (p + 1)/2]$ – `double` *Output*

On exit: the first $p * (p + 1)/2$ elements of **cov** contain the upper triangular part of the variance-covariance matrix of the p parameters 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 $i = 0, 1, \ldots, p - 1$ and $j = i, i + 1, \ldots, p - 1$.

19: **model**[$(3 * p * (p + 1))/2 +$ **nvar** $+ 14$] – `double` *Output*

On exit: if not 0, information on the fitted model for use in the functions described in 'See Also'.

20: **eps** – `double` *Input*

On entry: the value of the criterion used for model pruning. If **eps** $= 0.0$, a default value of $1e^{-10}$ is used.

Constraint: **eps** ≥ 0.0.

21: **info** – `int *` *Output*

On exit: **info** gives information on the success of the function call:

0: the function successfully completed its task.

i; $i = 1, 2, \ldots, 4, 6, 8, 9, \ldots, 12, 20$: the specification of the ith formal parameter was incorrect.

57: there are no degrees of freedom for the error estimates.

58: the fit is exact, no error estimates.

98: there is an underlying computational problem (this is an unlikely error exit).

99: the function failed to allocate enough memory.

> 100: an error occurred in a function specified by the user.

Notation

nrec	the number of data records, n.
nxvar	the number of independent variables, $p - 1$.
xvar	the independent variables that take the values in X (except the mean).
yvar	the dependent variable that takes the values in y.
iwts	if **iwts** ≥ 0, **iwts** is the index in the data that defines the weights, W.
b	the parameter estimates, $\hat{\beta}$.

Description

The general multiple linear regression model is defined by

$$y = X\beta + \varepsilon,$$

where y is a vector of n observations on the dependent variable,
X is an n by p matrix of the independent variables of column rank k,
β is a vector of length p of unknown parameters,
and ε is a vector of length n of unknown, Normally distributed, random errors such that var $\varepsilon = V\sigma^2$, where V is a known diagonal matrix of size n.

In most 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 assumed in the NAG DMC.

If $V = I$, the identity matrix, then least squares estimation is used. If $V \neq I$, then for a given weight matrix $W \propto V^{-1}$, weighted least squares estimation is used.

The least squares estimates $\hat{\beta}$ of the parameters β minimize $(y - X\beta)^T(y - X\beta)$ while the weighted least squares estimates minimize $(y - X\beta)^T W(y - X\beta)$.

NAG DMC uses a QR decomposition of X (or $W^{1/2}X$ in the weighted case), i.e.,

$$X = QR^* \text{ (or } W^{1/2}X = QR^*),$$

where $R^* = \begin{pmatrix} R \\ 0 \end{pmatrix}$ and R is a p by p upper triangular matrix and Q is an n by n orthogonal matrix.
If R is of full rank, $\hat{\beta}$ is the solution to

$$R\hat{\beta} = c_1,$$

where $c = Q^T y$ (or $Q^T W^{1/2} y$) and c_1 is the first p elements of c. 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 non-zero diagonal elements, k being the rank of R, and Q_* and P are p by p orthogonal matrices. This gives the solution

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

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

The fit of the model can be examined by considering the residuals. The residuals are defined as $r_i = y_i - \hat{y}_i$ in the case of an unweighted analysis and as $r_i = \sqrt{w_i}(y_i - \hat{y}_i)$ in a weighted analysis. In both cases $\hat{y} = X\hat{\beta}$ and corresponds to the fitted values. These fitted values can be written as Hy for an n by n matrix H. The ith diagonal element of H, h_i, gives a measure of the influence of the ith value of the independent variables on the fitted regression model. The values h_i are sometimes known as leverages.

References and Further Reading

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

Draper N R and Smith H (1985) *Applied Regression Analysis* (2nd Edition) Wiley.

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

Hammarling S (1985) The singular value decomposition in multivariate statistics *SIGNUM Newsl.* **20 (3)** 2-25.

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

Searle S R (1971) *Linear Models* Wiley.

See Also

nagdmc_basic_reg	simplified version of **nagdmc_linear_reg** using a restricted set of parameters.
nagdmc_extr_reg	computes fitted values, residuals and leverages for a regression.
nagdmc_load_reg	reads a linear regression model from a binary file.
nagdmc_predict_reg	computes predictions given a fitted regression model.
nagdmc_save_reg	writes a linear regression model to a binary file.
nagdmc_stepwise_reg	stepwise linear regression with Normal errors.
linear_reg_ex.c	the example calling program.