

NAG Library Function Document

nag_lars (g02mac)

1 Purpose

nag_lars (g02mac) performs Least Angle Regression (LARS), forward stagewise linear regression or Least Absolute Shrinkage and Selection Operator (LASSO).

2 Specification

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

void nag_lars (Nag_LARSModelType mtype, Nag_LARSPreProcess pred,
               Nag_LARSPreProcess prey, Integer n, Integer m, const double d[],
               Integer pdd, const Integer isx[], const double y[], Integer mnstep,
               Integer *ip, Integer *nstep, double b[], Integer pdb, double fitsum[],
               const double ropt[], Integer lropt, NagError *fail)
```

3 Description

nag_lars (g02mac) implements the LARS algorithm of Efron *et al.* (2004) as well as the modifications needed to perform forward stagewise linear regression and fit LASSO and positive LASSO models.

Given a vector of n observed values, $y = \{y_i : i = 1, 2, \dots, n\}$ and an $n \times p$ design matrix X , where the j th column of X , denoted x_j , is a vector of length n representing the j th independent variable x_j ,

standardized such that $\sum_{i=1}^n x_{ij} = 0$, and $\sum_{i=1}^n x_{ij}^2 = 1$ and a set of model parameters β to be estimated from the observed values, the LARS algorithm can be summarised as:

1. Set $k = 1$ and all coefficients to zero, that is $\beta = 0$.
2. Find the variable most correlated with y , say x_{j_1} . Add x_{j_1} to the ‘most correlated’ set \mathcal{A} . If $p = 1$ go to 8.
3. Take the largest possible step in the direction of x_{j_1} (i.e., increase the magnitude of β_{j_1}) until some other variable, say x_{j_2} , has the same correlation with the current residual, $y - x_{j_1}\beta_{j_1}$.
4. Increment k and add x_{j_k} to \mathcal{A} .
5. If $|\mathcal{A}| = p$ go to 8.
6. Proceed in the ‘least angle direction’, that is, the direction which is equiangular between all variables in \mathcal{A} , altering the magnitude of the parameter estimates of those variables in \mathcal{A} , until the k th variable, x_{j_k} , has the same correlation with the current residual.
7. Go to 4.
8. Let $K = k$.

As well as being a model selection process in its own right, with a small number of modifications the LARS algorithm can be used to fit the LASSO model of Tibshirani (1996), a positive LASSO model, where the independent variables enter the model in their defined direction (i.e., $\beta_{kj} \geq 0$), forward stagewise linear regression (Hastie *et al.* (2001)) and forward selection (Weisberg (1985)). Details of the required modifications in each of these cases are given in Efron *et al.* (2004).

The LASSO model of Tibshirani (1996) is given by

$$\underset{\alpha, \beta_k \in \mathbb{R}^p}{\text{minimize}} \|y - \alpha - X^T \beta_k\|^2 \quad \text{subject to} \quad \|\beta_k\|_1 \leq t_k$$

for all values of t_k , where $\alpha = \bar{y} = n^{-1} \sum_{i=1}^n y_i$. The positive LASSO model is the same as the standard LASSO model, given above, with the added constraint that

$$\beta_{kj} \geq 0, \quad j = 1, 2, \dots, p.$$

Unlike the standard LARS algorithm, when fitting either of the LASSO models, variables can be dropped as well as added to the set \mathcal{A} . Therefore the total number of steps K is no longer bounded by p .

Forward stagewise linear regression is an iterative procedure of the form:

1. Initialize $k = 1$ and the vector of residuals $r_0 = y - \alpha$.
2. For each $j = 1, 2, \dots, p$ calculate $c_j = x_j^T r_{k-1}$. The value c_j is therefore proportional to the correlation between the j th independent variable and the vector of previous residual values, r_k .
3. Calculate $j_k = \underset{j}{\operatorname{argmax}} |c_j|$, the value of j with the largest absolute value of c_j .
4. If $|c_{j_k}| < \epsilon$ then go to 7.
5. Update the residual values, with

$$r_k = r_{k-1} + \delta \operatorname{sign}(c_{j_k}) x_{j_k}$$

where δ is a small constant and $\operatorname{sign}(c_{j_k}) = -1$ when $c_{j_k} < 0$ and 1 otherwise.

6. Increment k and go to 2.
7. Set $K = k$.

If the largest possible step were to be taken, that is $\delta = |c_{j_k}|$ then forward stagewise linear regression reverts to the standard forward selection method as implemented in `nag_step_regsn` (g02eec).

The LARS procedure results in K models, one for each step of the fitting process. In order to aid in choosing which is the most suitable Efron *et al.* (2004) introduced a C_p -type statistic given by

$$C_p^{(k)} = \frac{\|y - X^T \beta_k\|^2}{\sigma^2} - n + 2\nu_k,$$

where ν_k is the approximate degrees of freedom for the k th step and

$$\sigma^2 = \frac{n - y^T y}{\nu_K}.$$

One way of choosing a model is therefore to take the one with the smallest value of $C_p^{(k)}$.

4 References

- Efron B, Hastie T, Johnstone I and Tibshirani R (2004) Least Angle Regression *The Annals of Statistics* (Volume 32) **2** 407–499
- Hastie T, Tibshirani R and Friedman J (2001) *The Elements of Statistical Learning: Data Mining, Inference and Prediction* Springer (New York)
- Tibshirani R (1996) Regression Shrinkage and Selection via the Lasso *Journal of the Royal Statistics Society, Series B (Methodological)* (Volume 58) **1** 267–288
- Weisberg S (1985) *Applied Linear Regression* Wiley

5 Arguments

- 1: **mtype** – Nag_LARSModelType *Input*
On entry: indicates the type of model to fit.
mtype = Nag_LARS_LAR
 LARS is performed.
mtype = Nag_LARS_ForwardStagewise
 Forward linear stagewise regression is performed.
mtype = Nag_LARS_LASSO
 LASSO model is fit.
mtype = Nag_LARS_PositiveLASSO
 A positive LASSO model is fit.
Constraint: **mtype** = Nag_LARS_LAR, Nag_LARS_ForwardStagewise, Nag_LARS_LASSO or Nag_LARS_PositiveLASSO.
- 2: **pred** – Nag_LARSPreProcess *Input*
On entry: indicates the type of data preprocessing to perform on the independent variables supplied in **d** to comply with the standardized form of the design matrix.
pred = Nag_LARS_None
 No preprocessing is performed.
pred = Nag_LARS_Centered
 Each of the independent variables, x_j , for $j = 1, 2, \dots, p$, are mean centred prior to fitting the model. The means of the independent variables, \bar{x} , are returned in **b**, with $\bar{x}_j = \mathbf{b}[(\mathbf{nstep} + 1) \times \mathbf{pdb} + j - 1]$, for $j = 1, 2, \dots, p$.
pred = Nag_LARS_Normalized
 Each independent variable is normalized, with the j th variable scaled by $1/\sqrt{x_j^T x_j}$. The scaling factor used by variable j is returned in $\mathbf{b}[\mathbf{nstep} \times \mathbf{pdb} + j - 1]$.
pred = Nag_LARS_CenteredNormalized
 As **pred** = Nag_LARS_Centered and Nag_LARS_Normalized, all of the independent variables are mean centred prior to being normalized.
Suggested value: **pred** = Nag_LARS_CenteredNormalized.
Constraint: **pred** = Nag_LARS_None, Nag_LARS_Centered, Nag_LARS_Normalized or Nag_LARS_CenteredNormalized.
- 3: **prey** – Nag_LARSPreProcess *Input*
On entry: indicates the type of data preprocessing to perform on the dependent variable supplied in **y**.
prey = Nag_LARS_None
 No preprocessing is performed, this is equivalent to setting $\alpha = 0$.
prey = Nag_LARS_Centered
 The dependent variable, y , is mean centred prior to fitting the model, so $\alpha = \bar{y}$. Which is equivalent to fitting a non-penalized intercept to the model and the degrees of freedom etc. are adjusted accordingly.
 The value of α used is returned in **fitsum**[**nstep** × 6].
Suggested value: **prey** = Nag_LARS_Centered.
Constraint: **prey** = Nag_LARS_None or Nag_LARS_Centered.

- 4: **n** – Integer *Input*
On entry: n , the number of observations.
Constraint: $n \geq 1$.
- 5: **m** – Integer *Input*
On entry: m , the total number of independent variables.
Constraint: $m \geq 1$.
- 6: **d**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **d** must be at least $\mathbf{pdd} \times \mathbf{m}$.
On entry: D , the data, which along with **pred** and **isx**, defines the design matrix X . The i th observation for the j th variable must be supplied in **d**[($j - 1$) \times **pdd** + $i - 1$], for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$.
- 7: **pdd** – Integer *Input*
On entry: the stride separating row elements in the two-dimensional data stored in the array **d**.
Constraint: $\mathbf{pdd} \geq n$.
- 8: **isx**[**m**] – const Integer *Input*
On entry: indicates which independent variables from **d** will be included in the design matrix, X .
 If **isx** is **NULL**, all variables are included in the design matrix.
 Otherwise, for $j = 1, 2, \dots, m$ when **isx**[$j - 1$] must be set as follows:
isx[$j - 1$] = 1
 To indicate that the j th variable, as supplied in **d**, is included in the design matrix;
isx[$j - 1$] = 0
 To indicated that the j th variable, as supplied in **d**, is not included in the design matrix;
 and $p = \sum_{j=1}^m \mathbf{isx}[j - 1]$.
Constraint: **isx**[$j - 1$] = 0 or 1 and at least one value of **isx**[$j - 1$] $\neq 0$, for $j = 1, 2, \dots, m$.
- 9: **y**[**n**] – const double *Input*
On entry: y , the observations on the dependent variable.
- 10: **mnstep** – Integer *Input*
On entry: the maximum number of steps to carry out in the model fitting process.
 If **mtype** = Nag_LARS_LAR, the maximum number of steps the algorithm will take is $\min(p, n)$ if **prey** = Nag_LARS_None, otherwise $\min(p, n - 1)$.
 If **mtype** = Nag_LARS_ForwardStagewise, the maximum number of steps the algorithm will take is likely to be several orders of magnitude more and is no longer bound by p or n .
 If **mtype** = Nag_LARS_LASSO or Nag_LARS_PositiveLASSO, the maximum number of steps the algorithm will take lies somewhere between that of the LARS and forward linear stagewise regression, again it is no longer bound by p or n .
Constraint: **mnstep** ≥ 1 .
- 11: **ip** – Integer * *Output*
On exit: p , number of parameter estimates.

If **isx** is **NULL**, $p = \mathbf{m}$, i.e., the number of variables in **d**.

Otherwise p is the number of nonzero values in **isx**.

12: **nstep** – Integer * Output

On exit: K , the actual number of steps carried out in the model fitting process.

13: **b**[*dim*] – double Output

Note: the dimension, *dim*, of the array **b** must be at least $\mathbf{pdb} \times (\mathbf{mnstep} + 2)$.

On exit: β the parameter estimates, with $\mathbf{b}[(k-1) \times \mathbf{pdb} + j - 1] = \beta_{kj}$, the parameter estimate for the j th variable, $j = 1, 2, \dots, p$ at the k th step of the model fitting process, $k = 1, 2, \dots, \mathbf{nstep}$.

By default, when **pred** = Nag_LARS_Normalized or Nag_LARS_CenteredNormalized the parameter estimates are rescaled prior to being returned. If the parameter estimates are required on the normalized scale, then this can be overridden via **ropt**.

The values held in the remaining part of **b** depend on the type of preprocessing performed.

If **pred** = Nag_LARS_None,

$$\begin{aligned} \mathbf{b}[\mathbf{nstep} \times \mathbf{pdb} + j - 1] &= 1 \\ \mathbf{b}[(\mathbf{nstep} + 1) \times \mathbf{pdb} + j - 1] &= 0 \end{aligned}$$

If **pred** = Nag_LARS_Centered,

$$\begin{aligned} \mathbf{b}[\mathbf{nstep} \times \mathbf{pdb} + j - 1] &= 1 \\ \mathbf{b}[(\mathbf{nstep} + 1) \times \mathbf{pdb} + j - 1] &= \bar{x}_j \end{aligned}$$

If **pred** = Nag_LARS_Normalized,

$$\begin{aligned} \mathbf{b}[\mathbf{nstep} \times \mathbf{pdb} + j - 1] &= 1 / \sqrt{x_j^T x_j} \\ \mathbf{b}[(\mathbf{nstep} + 1) \times \mathbf{pdb} + j - 1] &= 0 \end{aligned}$$

If **pred** = Nag_LARS_CenteredNormalized,

$$\begin{aligned} \mathbf{b}[\mathbf{nstep} \times \mathbf{pdb} + j - 1] &= 1 / \sqrt{(x_j - \bar{x}_j)^T (x_j - \bar{x}_j)} \\ \mathbf{b}[(\mathbf{nstep} + 1) \times \mathbf{pdb} + j - 1] &= \bar{x}_j \end{aligned}$$

for $j = 1, 2, \dots, p$.

14: **pdb** – Integer Input

On entry: the stride separating row elements in the two-dimensional data stored in the array **b**.

Constraint: $\mathbf{pdb} \geq p$, where p is the number of parameter estimates as described in **ip**.

15: **fitsum**[$6 \times (\mathbf{mnstep} + 1)$] – double Output

On exit: summaries of the model fitting process. When $k = 1, 2, \dots, \mathbf{nstep}$,

fitsum[($k-1$) \times 6]

$\|\beta_k\|_1$, the sum of the absolute values of the parameter estimates for the k th step of the modelling fitting process. If **pred** = Nag_LARS_Normalized or Nag_LARS_CenteredNormalized, the scaled parameter estimates are used in the summation.

fitsum[($k-1$) \times 6 + 1]

RSS_k , the residual sums of squares for the k th step, where $\text{RSS}_k = \|y - X^T \beta_k\|^2$.

fitsum[($k-1$) \times 6 + 2]

ν_k , approximate degrees of freedom for the k th step.

fitsum[($k-1$) \times 6 + 3]

$C_p^{(k)}$, a C_p -type statistic for the k th step, where $C_p^{(k)} = \frac{\text{RSS}_k}{\sigma^2} - n + 2\nu_k$.

fitsum $[(k-1) \times 6 + 4]$

\hat{C}_k , correlation between the residual at step $k-1$ and the most correlated variable not yet in the active set \mathcal{A} , where the residual at step 0 is y .

fitsum $[(k-1) \times 6 + 5]$

$\hat{\gamma}_k$, the step size used at step k .

In addition

fitsum $[\text{nstep} \times 6]$

α , with $\alpha = \bar{y}$ if **prey** = Nag_LARS_Centered and 0 otherwise.

fitsum $[\text{nstep} \times 6 + 1]$

RSS_0 , the residual sums of squares for the null model, where $\text{RSS}_0 = y^T y$ when **prey** = Nag_LARS_None and $\text{RSS}_0 = (y - \bar{y})^T (y - \bar{y})$ otherwise.

fitsum $[\text{nstep} \times 6 + 2]$

ν_0 , the degrees of freedom for the null model, where $\nu_0 = 0$ if **prey** = Nag_LARS_None and $\nu_0 = 1$ otherwise.

fitsum $[\text{nstep} \times 6 + 3]$

$C_p^{(0)}$, a C_p -type statistic for the null model, where $C_p^{(0)} = \frac{\text{RSS}_0}{\sigma^2} - n + 2\nu_0$.

fitsum $[\text{nstep} \times 6 + 4]$

σ^2 , where $\sigma^2 = \frac{n - \text{RSS}_K}{\nu_K}$ and $K = \text{nstep}$.

Although the C_p statistics described above are returned when **fail.code** = NW_LIMIT_REACHED they may not be meaningful due to the estimate σ^2 not being based on the saturated model.

16: **ropt** $[\text{lropt}]$ – const double

Input

On entry: optional parameters to control various aspects of the LARS algorithm.

The default value will be used for **ropt** $[i-1]$ if **lropt** $< i$, therefore setting **lropt** = 0 will use the default values for all optional arguments and **ropt** need not be set and may be **NULL**. The default value will also be used if an invalid value is supplied for a particular argument, for example, setting **ropt** $[i-1] = -1$ will use the default value for argument i .

ropt $[0]$

The minimum step size that will be taken.

Default is $100 \times \text{eps}$ is used, where *eps* is the *machine precision* returned by nag_machine_precision (X02AJC).

ropt $[1]$

General tolerance, used amongst other things, for comparing correlations.

Default is **ropt** $[0]$.

ropt $[2]$

If set to 1, parameter estimates are rescaled before being returned.

If set to 0, no rescaling is performed.

This argument has no effect when **pred** = Nag_LARS_None or Nag_LARS_Centered.

Default is for the parameter estimates to be rescaled.

ropt $[3]$

If set to 1, it is assumed that the model contains an intercept during the model fitting process and when calculating the degrees of freedom.

If set to 0, no intercept is assumed.

This has no effect on the amount of preprocessing performed on **y**.

Default is to treat the model as having an intercept when **prey** = Nag_LARS_Centered and as not having an intercept when **prey** = Nag_LARS_None.

ropt[4]

As implemented, the LARS algorithm can either work directly with y and X , or it can work with the cross-product matrices, $X^T y$ and $X^T X$. In most cases it is more efficient to work with the cross-product matrices. This flag allows you direct control over which method is used, however, the default value will usually be the best choice.

If **ropt**[4] = 1, y and X are worked with directly.

If **ropt**[4] = 0, the cross-product matrices are used.

Default is 1 when $p \geq 500$ and $n < p$ and 0 otherwise.

Constraints:

ropt[0] > *machine precision*;

ropt[1] > *machine precision*;

ropt[2] = 0 or 1;

ropt[3] = 0 or 1;

ropt[4] = 0 or 1.

17: **lropt** – Integer

Input

On entry: length of the options array **ropt**.

Constraint: $0 \leq \text{lropt} \leq 5$.

18: **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 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_ARRAY_SIZE

On entry, **lropt** = $\langle \text{value} \rangle$.

Constraint: $0 \leq \text{lropt} \leq 5$.

On entry, **pdb** = $\langle \text{value} \rangle$ and **m** = $\langle \text{value} \rangle$.

Constraint: if **isx** is **NULL** then **pdb** \geq **m**.

On entry, **pdb** = $\langle \text{value} \rangle$ and $p = \langle \text{value} \rangle$.

Constraint: if **isx** is not **NULL** then **pdb** $\geq p$.

On entry, **pdd** = $\langle \text{value} \rangle$ and **n** = $\langle \text{value} \rangle$.

Constraint: **pdd** \geq **n**.

NE_BAD_PARAM

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

NE_INT

On entry, **m** = $\langle \text{value} \rangle$.

Constraint: **m** ≥ 1 .

On entry, $\mathbf{n} = \langle \text{value} \rangle$.
 Constraint: $\mathbf{n} \geq 1$.

NE_INT_ARRAY

On entry, all values of \mathbf{isx} are zero.
 Constraint: at least one value of \mathbf{isx} must be nonzero.

On entry, $\mathbf{isx}[\langle \text{value} \rangle] = \langle \text{value} \rangle$.
 Constraint: $\mathbf{isx}[i] = 0$ or 1 for all i .

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_MAX_STEP

On entry, $\mathbf{mnstep} = \langle \text{value} \rangle$.
 Constraint: $\mathbf{mnstep} \geq 1$.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
 See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NW_LIMIT_REACHED

Fitting process did not finish in \mathbf{mnstep} steps. Try increasing the size of \mathbf{mnstep} and supplying larger output arrays.
 All output is returned as documented, up to step \mathbf{mnstep} , however, σ and the C_p statistics may not be meaningful.

NW_OVERFLOW_WARN

$\nu_K = n$, therefore σ has been set to a large value. Output is returned as documented.
 σ^2 is approximately zero and hence the C_p -type criterion cannot be calculated. All other output is returned as documented.

NW_POTENTIAL_PROBLEM

Degenerate model, no variables added and $\mathbf{nstep} = 0$. Output is returned as documented.

7 Accuracy

Not applicable.

8 Further Comments

`nag_lars` (g02mac) returns the parameter estimates at various points along the solution path of a LARS, LASSO or stagewise regression analysis. If the solution is required at a different set of points, for example when performing cross-validation, then `nag_lars_param` (g02mcc) can be used.

For datasets with a large number of observations, n , it may be impractical to store the full X matrix in memory in one go. In such instances the cross-product matrices $X^T y$ and $X^T X$ can be calculated, using for example, multiple calls to `nag_sum_sqs` (g02buc) and `nag_sum_sqs_combine` (g02bzc), and `nag_lars_xtx` (g02mbc) called to perform the analysis.

The amount of workspace used by `nag_lars` (g02mac) depends on whether the cross-product matrices are being used internally (as controlled by `ropt`). If the cross-product matrices are being used then

nag_lars (g02mac) internally allocates approximately $2p^2 + 4p + \max(np)$ elements of real storage compared to $p^2 + 3p + \max(np) + 2n + n \times p$ elements when X and y are used directly. In both cases approximately $5p$ elements of integer storage are also used. If a forward linear stagewise analysis is performed then an additional $p^2 + 5p$ elements of real storage are required.

9 Example

This example performs a LARS on a simulated dataset with 20 observations and 6 independent variables.

9.1 Program Text

```
/* nag_lars (g02mac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer i, j, k, ip, ldb, ldd, m, mnstep, n, nstep, lropt;
    Integer *isx = 0;
    Integer exit_status = 0;

    /* NAG structures and types */
    NagError fail;
    Nag_LARSModelType mtype;
    Nag_LARSPreProcess pred, prey;

    /* Double scalar and array declarations */
    double *b = 0, *d = 0, *fitsum = 0, *y = 0, *ropt = 0;

    /* Character scalar and array declarations */
    char cmttype[40], cpred[40], cprey[40];

    /* Initialize the error structure */
    INIT_FAIL(fail);

    printf("nag_lars (g02mac) Example Program Results\n\n");

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

    /* Read in the problem size */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &m);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &m);
#endif

    /* Read in the model specification */
#ifdef _WIN32
    scanf_s("%39s%39s%39s%" NAG_IFMT "%*[\n] ", cmttype,
            (unsigned)_countof(cmttype), cpred,
            (unsigned)_countof(cpred), cprey,
```

```

        (unsigned)_countof(cprey), &mnstep);
#else
    scanf("%39s%39s%39s" NAG_IFMT "%*[\n] ", cmttype, cpred, cprey, &mnstep);
#endif
    mtype = (Nag_LARSModelType) nag_enum_name_to_value(cmttype);
    pred = (Nag_LARSPreProcess) nag_enum_name_to_value(cpred);
    prey = (Nag_LARSPreProcess) nag_enum_name_to_value(cprey);

    /* Using all variables */
    isx = 0;

    /* Optional arguments (using defaults) */
    lropt = 0;
    ropt = 0;

    /* Allocate memory for the data */
    ldd = n;
    if (!(y = NAG_ALLOC(n, double)) || !(d = NAG_ALLOC(ldd * m, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read in the data */
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
#ifdef _WIN32
            scanf_s("%lf", &d[j * ldd + i]);
#else
            scanf("%lf", &d[j * ldd + i]);
#endif
        }
#ifdef _WIN32
        scanf_s("%lf", &y[i]);
#else
        scanf("%lf", &y[i]);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

    /* Allocate output arrays */
    ldb = m;
    if (!(b = NAG_ALLOC(ldb * (mnstep + 2), double)) ||
        !(fitsum = NAG_ALLOC(6 * (mnstep + 1), double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Call nag_lars (g02mac) to fit the model */
    nag_lars(mtype, pred, prey, n, m, d, ldd, isx, y, mnstep, &ip, &nstep,
        b, ldb, fitsum, ropt, lropt, &fail);
    if (fail.code != NE_NOERROR) {
        if (fail.code != NW_OVERFLOW_WARN && fail.code != NW_POTENTIAL_PROBLEM
            && fail.code != NW_LIMIT_REACHED) {
            printf("Error from nag_lars (g02mac).\n%s\n", fail.message);
            exit_status = 11;
            goto END;
        }
        else {
            printf("Warning from nag_lars (g02mac).\n%s\n", fail.message);
            exit_status = 2;
        }
    }
}

```

```

/* Display the parameter estimates */
printf(" Step ");
for (i = 0; i < MAX(ip - 2, 0) * 5; i++)
    printf(" ");
printf(" Parameter Estimate\n ");
for (i = 0; i < 5 + ip * 10; i++)
    printf("-");
printf("\n");
for (k = 0; k < nstep; k++) {
    printf(" %3" NAG_IFMT " ", k + 1);
    for (j = 0; j < ip; j++) {
        printf(" %9.3f", b[k * ldb + j]);
    }
    printf("\n");
}
printf("\n");
printf(" alpha: %9.3f\n", fitsum[6 * nstep]);
printf("\n");
printf(" Step      Sum      RSS      df      Cp      Ck      Step Size\n ");
for (i = 0; i < 64; i++)
    printf("-");
printf("\n");
for (k = 0; k < nstep; k++) {
    printf(" %3" NAG_IFMT " %9.3f %9.3f %6.0f %9.3f %9.3f %9.3f\n",
        k + 1, fitsum[k * 6], fitsum[k * 6 + 1], fitsum[k * 6 + 2],
        fitsum[k * 6 + 3], fitsum[k * 6 + 4], fitsum[k * 6 + 5]);
}
printf("\n");
printf(" sigma^2: %9.3f\n", fitsum[nstep * 6 + 4]);

END:
NAG_FREE(y);
NAG_FREE(d);
NAG_FREE(b);
NAG_FREE(fitsum);
NAG_FREE(ropt);

return (exit_status);
}

```

9.2 Program Data

```

nag_lars (g02mac) Example Program Data
20 6 :: n,m
Nag_LARS_LAR Nag_LARS_CenteredNormalized
Nag_LARS_Centered 6 :: mtype,pred,prey,mnstep
10.28 1.77 9.69 15.58 8.23 10.44 -46.47
9.08 8.99 11.53 6.57 15.89 12.58 -35.80
17.98 13.10 1.04 10.45 10.12 16.68 -129.22
14.82 13.79 12.23 7.00 8.14 7.79 -42.44
17.53 9.41 6.24 3.75 13.12 17.08 -73.51
7.78 10.38 9.83 2.58 10.13 4.25 -26.61
11.95 21.71 8.83 11.00 12.59 10.52 -63.90
14.60 10.09 -2.70 9.89 14.67 6.49 -76.73
3.63 9.07 12.59 14.09 9.06 8.19 -32.64
6.35 9.79 9.40 12.79 8.38 16.79 -83.29
4.66 3.55 16.82 13.83 21.39 13.88 -16.31
8.32 14.04 17.17 7.93 7.39 -1.09 -5.82
10.86 13.68 5.75 10.44 10.36 10.06 -47.75
4.76 4.92 17.83 2.90 7.58 11.97 18.38
5.05 10.41 9.89 9.04 7.90 13.12 -54.71
5.41 9.32 5.27 15.53 5.06 19.84 -55.62
9.77 2.37 9.54 20.23 9.33 8.82 -45.28
14.28 4.34 14.23 14.95 18.16 11.03 -22.76
10.17 6.80 3.17 8.57 16.07 15.93 -104.32
5.39 2.67 6.37 13.56 10.68 7.35 -55.94 :: End of d, y

```

9.3 Program Results

nag_lars (g02mac) Example Program Results

Step	Parameter Estimate					
1	0.000	0.000	3.125	0.000	0.000	0.000
2	0.000	0.000	3.792	0.000	0.000	-0.713
3	-0.446	0.000	3.998	0.000	0.000	-1.151
4	-0.628	-0.295	4.098	0.000	0.000	-1.466
5	-1.060	-1.056	4.110	-0.864	0.000	-1.948
6	-1.073	-1.132	4.118	-0.935	-0.059	-1.981

alpha: -50.037

Step	Sum	RSS	df	Cp	Ck	Step Size
1	72.446	8929.855	2	13.355	123.227	72.446
2	103.385	6404.701	3	7.054	50.781	24.841
3	126.243	5258.247	4	5.286	30.836	16.225
4	145.277	4657.051	5	5.309	19.319	11.587
5	198.223	3959.401	6	5.016	12.266	24.520
6	203.529	3954.571	7	7.000	0.910	2.198

sigma^2: 304.198

This example plot shows the regression coefficients (β_k) plotted against the scaled absolute sum of the parameter estimates ($\|\beta_k\|_1$).

