NAG Library Function Document

nag_regsn_ridge_opt (g02kac)

1 Purpose
nag_regsn_ridge_opt (g02kac) calculates a ridge regression, optimizing the ridge parameter according to one of four prediction error criteria.

2 Specification
#include <nag.h>
#include <nagg02.h>

void nag_regsn_ridge_opt (Nag_OrderType order, Integer n, Integer m,
const double x[], Integer pdx, const Integer isx[], Integer ip,
const double tau, const double y[], double *h, Nag_PredictError opt,
Integer *niter, double tol, double *nep, Nag_EstimatesOption orig,
double b[], double vif[], double res[], double *rss, Integer *df,
Nag_OptionLOO optloo, double perr[], NagError *fail)

3 Description
A linear model has the form:
\[ y = c + X\beta + \epsilon, \]
where
- \( y \) is an \( n \) by 1 matrix of values of a dependent variable;
- \( c \) is a scalar intercept term;
- \( X \) is an \( n \) by \( m \) matrix of values of independent variables;
- \( \beta \) is an \( m \) by 1 matrix of unknown values of parameters;
- \( \epsilon \) is an \( n \) by 1 matrix of unknown random errors such that variance of \( \epsilon = \sigma^2 I. \)

Let \( \tilde{X} \) be the mean-centred \( X \) and \( \tilde{y} \) the mean-centred \( y \). Furthermore, \( \tilde{X} \) is scaled such that the diagonal elements of the cross product matrix \( \tilde{X}^T \tilde{X} \) are one. The linear model now takes the form:
\[ \tilde{y} = \tilde{X}\tilde{\beta} + \epsilon. \]
Ridge regression estimates the parameters \( \tilde{\beta} \) in a penalised least squares sense by finding the \( \tilde{b} \) that minimizes
\[ \| \tilde{X}\tilde{b} - \tilde{y} \|^2 + h\| \tilde{b} \|^2, \quad h > 0, \]
where \( \| \cdot \| \) denotes the \( \ell_2 \)-norm and \( h \) is a scalar regularization or ridge parameter. For a given value of \( h \), the parameter estimates \( \tilde{b} \) are found by evaluating
\[ \tilde{b} = (\tilde{X}^T\tilde{X} + hI)^{-1}\tilde{X}^T\tilde{y}. \]
Note that if \( h = 0 \) the ridge regression solution is equivalent to the ordinary least squares solution.

Rather than calculate the inverse of \( (\tilde{X}^T\tilde{X} + hI) \) directly, nag_regsn_ridge_opt (g02kac) uses the singular value decomposition (SVD) of \( \tilde{X} \). After decomposing \( \tilde{X} \) into \( UDV^T \) where \( U \) and \( V \) are orthogonal matrices and \( D \) is a diagonal matrix, the parameter estimates become
\[ \tilde{b} = V(D^T D + hI)^{-1}D^T \tilde{y}. \]
A consequence of introducing the ridge parameter is that the effective number of parameters, $\gamma$, in the model is given by the sum of diagonal elements of $D^T D (D^T D + hI)^{-1}$, see Moody (1992) for details.

Any multi-collinearity in the design matrix $X$ may be highlighted by calculating the variance inflation factors for the fitted model. The $j$th variance inflation factor, $v_j$, is a scaled version of the multiple correlation coefficient between independent variable $j$ and the other independent variables, $R_j$, and is given by

$$v_j = \frac{1}{1 - R_j^2}, \quad j = 1, 2, \ldots, m.$$

The $m$ variance inflation factors are calculated as the diagonal elements of the matrix

$$(\hat{X}^T \hat{X} + hI)^{-1} \hat{X}^T \hat{X} (\hat{X}^T \hat{X} + hI)^{-1},$$

which, using the SVD of $\hat{X}$, is equivalent to the diagonal elements of the matrix

$$V (D^T D + hI)^{-1} D^T D (D^T D + hI)^{-1} V^T.$$

Although parameter estimates $\tilde{b}$ are calculated by using $\hat{X}$, it is usual to report the parameter estimates $b$ associated with $X$. These are calculated from $\tilde{b}$, and the means and scalings of $X$. Optionally, either $\tilde{b}$ or $b$ may be calculated.

The method can adopt one of four criteria to minimize while calculating a suitable value for $h$:

(a) Generalized cross-validation (GCV):

$$\frac{ns}{(n - \gamma)^2};$$

(b) Unbiased estimate of variance (UEV):

$$\frac{s}{n - \gamma};$$

(c) Future prediction error (FPE):

$$\frac{1}{n} \left( s + \frac{2\gamma s}{n - \gamma} \right);$$

(d) Bayesian information criterion (BIC):

$$\frac{1}{n} \left( s + \frac{\log (n) \gamma s}{n - \gamma} \right);$$

where $s$ is the sum of squares of residuals. However, the function returns all four of the above prediction errors regardless of the one selected to minimize the ridge parameter, $h$. Furthermore, the function will optionally return the leave-one-out cross-validation (LOOCV) prediction error.

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

\textit{On entry:} the \texttt{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 \texttt{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \texttt{order} = Nag_RowMajor or Nag_ColMajor.

2: \( n \) – Integer

\textit{On entry:} \( n \), the number of observations.

\textit{Constraint:} \( n > 1 \).

3: \( m \) – Integer

\textit{On entry:} the number of independent variables available in the data matrix \( X \).

\textit{Constraint:} \( m \leq n \).

4: \( x[\text{dim}] \) – const double

\textit{Note:} the dimension, \( \text{dim} \), of the array \( x \) must be at least

\[ \max(1, \text{pdx} \times m) \text{ when } \text{order} = \text{Nag_ColMajor}; \]
\[ \max(1, n \times \text{pdx}) \text{ when } \text{order} = \text{Nag_RowMajor}. \]

The \( (i, j) \)th element of the matrix \( X \) is stored in

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

\textit{On entry:} the values of independent variables in the data matrix \( X \).

5: \( \text{pdx} \) – Integer

\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) in the array \( x \).

\textit{Constraints:}

\[ \text{if } \text{order} = \text{Nag_ColMajor}, \text{pdx} \geq n; \]
\[ \text{if } \text{order} = \text{Nag_RowMajor}, \text{pdx} \geq m. \]

6: \( \text{isx}[m] \) – const Integer

\textit{On entry:} indicates which \( m \) independent variables are included in the model.

\( \text{isx}[j - 1] = 1 \)

The \( j \)th variable in \( x \) will be included in the model.

\( \text{isx}[j - 1] = 0 \)

Variable \( j \) is excluded.

\textit{Constraint:} \texttt{isx}[j - 1] = 0 or 1, for \( j = 1, 2, \ldots, m \).

7: \( \text{ip} \) – Integer

\textit{On entry:} \( m \), the number of independent variables in the model.

\textit{Constraints:}

\[ 1 \leq \text{ip} \leq m; \]

Exactly \( \text{ip} \) elements of \texttt{isx} must be equal to 1.
8:  \text{tau} – double \quad Input

\textit{On entry:} singular values less than \text{tau} of the SVD of the data matrix \( X \) will be set equal to zero.
\textit{Suggested value:} \text{tau} = 0.0
\textit{Constraint:} \text{tau} \geq 0.0.

9:  \text{y[n]} – const double \quad Input

\textit{On entry:} the \( n \) values of the dependent variable \( y \).

10: \text{h} – double * \quad Input/Output

\textit{On entry:} an initial value for the ridge regression parameter \( h \); used as a starting point for the optimization.
\textit{Constraint:} \text{h} > 0.0.
\textit{On exit:} \text{h} is the optimized value of the ridge regression parameter \( h \).

11: \text{opt} – Nag_PredictError \quad Input

\textit{On entry:} the measure of prediction error used to optimize the ridge regression parameter \( h \). The value of \text{opt} must be set equal to one of:
\begin{itemize}
  \item \text{opt} = Nag_GCV
    Generalized cross-validation (GCV);
  \item \text{opt} = Nag_UEV
    Unbiased estimate of variance (UEV)
  \item \text{opt} = Nag_FPE
    Future prediction error (FPE)
  \item \text{opt} = Nag_BIC
    Bayesian information criterion (BIC).
\end{itemize}
\textit{Constraint:} \text{opt} = \text{Nag_GCV}, \text{Nag_UEV}, \text{Nag_FPE} or \text{Nag_BIC}.

12: \text{niter} – Integer * \quad Input/Output

\textit{On entry:} the maximum number of iterations allowed to optimize the ridge regression parameter \( h \).
\textit{Constraint:} \text{niter} \geq 1.
\textit{On exit:} the number of iterations used to optimize the ridge regression parameter \( h \) within \text{tol}.

13: \text{tol} – double \quad Input

\textit{On entry:} iterations of the ridge regression parameter \( h \) will halt when consecutive values of \( h \) lie within \text{tol}.
\textit{Constraint:} \text{tol} > 0.0.

14: \text{nep} – double * \quad Output

\textit{On exit:} the number of effective parameters, \( \gamma \), in the model.

15: \text{orig} – Nag_EstimatesOption \quad Input

\textit{On entry:} if \text{orig} = Nag_EstimatesOrig, the parameter estimates \( b \) are calculated for the original data; otherwise \text{orig} = Nag_EstimatesStand and the parameter estimates \( \tilde{b} \) are calculated for the standardized data.
\textit{Constraint:} \text{orig} = \text{Nag_EstimatesOrig} or \text{Nag_EstimatesStand}. 

\textit{Note:}\ \text{nep} \leq \text{niter}. 

\textit{Note:}\ \text{tau} \leq \text{tol}.
On exit: contains the intercept and parameter estimates for the fitted ridge regression model in the order indicated by isx. The first element of b contains the estimate for the intercept; b[j] contains the parameter estimate for the jth independent variable in the model, for j = 1, 2, ..., ip.

On exit: the variance inflation factors in the order indicated by isx. For the jth independent variable in the model, vif[j - 1] is the value of v_j, for j = 1, 2, ..., ip.

On exit: res[i - 1] is the value of the i-th residual for the fitted ridge regression model, for i = 1, 2, ..., n.

On exit: the sum of squares of residual values.

On exit: the degrees of freedom for the residual sum of squares rss.

On entry: if optloo = Nag_WantLOO, the leave-one-out cross-validation estimate of prediction error is calculated; otherwise no such estimate is calculated and optloo = Nag_NoLOO.

Constraint: optloo = Nag_NoLOO or Nag_WantLOO.

On exit: the first four elements contain, in this order, the measures of prediction error: GCV, UEV, FPE and BIC. If optloo = Nag_WantLOO, perr[4] is the LOOCV estimate of prediction error; otherwise perr[4] is not referenced.

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_CONS
On entry, ip = ⟨value⟩; m = ⟨value⟩.
Constraint: 1 ≤ ip ≤ m.

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 ⟨value⟩ had an illegal value.

NE_INT
On entry, n = ⟨value⟩.
Constraint: n > 1.
On entry, \texttt{niter} = \langle value\rangle.
Constraint: \texttt{niter} \geq 1.

On entry, \texttt{pdx} = \langle value\rangle.
Constraint: \texttt{pdx} > 0.

\textbf{NE\_INT\_2}

On entry, \texttt{m} = \langle value\rangle and \texttt{n} = \langle value\rangle.
Constraint: \texttt{m} \leq \texttt{n}.

On entry, \texttt{pdx} = \langle value\rangle; \texttt{n} = \langle value\rangle.
Constraint: \texttt{pdx} \geq \texttt{n}.

On entry, \texttt{pdx} = \langle value\rangle and \texttt{m} = \langle value\rangle.
Constraint: \texttt{pdx} \geq \texttt{m}.

\textbf{NE\_INT\_ARG\_CONS}

On entry, \texttt{ip} = \langle value\rangle.
Constraint: \text{sum(\texttt{isx})} = \texttt{ip}.

\textbf{NE\_INT\_ARRAY\_VAL\_1\_OR\_2}

On entry, \texttt{isx[\langle value\rangle]} = \langle value\rangle.
Constraint: \texttt{isx[j - 1]} = 0 or 1.

\textbf{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.

\textbf{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.

\textbf{NE\_REAL}

On entry, \texttt{h} = \langle value\rangle.
Constraint: \texttt{h} > 0.0.

On entry, \texttt{tau} = \langle value\rangle.
Constraint: \texttt{tau} \geq 0.0.

On entry, \texttt{tol} = \langle value\rangle.
Constraint: \texttt{tol} > 0.0.

\textbf{NE\_SVD\_FAIL}

SVD failed to converge.

\textbf{NW\_TOO\_MANY\_ITER}

Maximum number of iterations used.

\section{Accuracy}

Not applicable.
8 Parallelism and Performance

nag_regsn_ridge_opt (g02kac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_regsn_ridge_opt (g02kac) 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

nag_regsn_ridge_opt (g02kac) allocates internally \( \max(5 \times (n - 1), 2 \times ip \times ip) + (n + 3) \times ip + n \) elements of double precision storage.

10 Example

This example reads in data from an experiment to model body fat, and a ridge regression is calculated that optimizes GCV prediction error.

10.1 Program Text

/* nag_regsn_ridge_opt (g02kac) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 9, 2009. */

/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>
#include <nagx04.h>

int main(void)
{
  /*Integer scalar and array declarations */
  Integer exit_status = 0;
  Integer df, i, ip, ip1, j, m, n, niter, one = 1;
  Integer *isx = 0;
  /*Double scalar and array declarations */
  double h, nep, rss, tau, tol;
  double *b = 0, *perr = 0, *res = 0, *vif = 0, *x = 0, *y = 0;
  /*Character scalar and array declarations */
  char sopt[40], sorig[40], soptloo[40];
  /*NAG Types */
  Nag_OrderType order;
  Nag_PredictError opt;
  Nag_EstimatesOption orig;
  Nag_OptionLOO optloo;
  NagError fail;

  INIT_FAIL(fail);

  printf("\n",
      "nag_regsn_ridge_opt (g02kac) Example Program Results");
  /* Skip heading in data file*/
  ifndef _WIN32
    scanf_s("%*\n",
      "nag_regsn_ridge_opt (g02kac) Example Program Results");
  /* Skip heading in data file*/
  else
  
  
Mark 25
/* Read in data and check array limits*/
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%f%"NAG_IFMT"%f%"NAG_IFMT"%f%[\n] ",
        &n, &m, &h, sopt, _countof(sopt), &tol, &niter, sorig,
        _countof(sorig), soptloo, _countof(soptloo));
#else
    scanf("%"NAG_IFMT"%f%"NAG_IFMT"%f%"NAG_IFMT"%f%[\n] ",
        &n, &m, &h, sopt, &tol, &niter, sorig, soptloo);
#endif
opt = (Nag_PredictError) nag_enum_name_to_value(sopt);
orig = (Nag_EstimatesOption) nag_enum_name_to_value(sorig);
optloo = (Nag_OptionLOO) nag_enum_name_to_value(soptloo);

#ifdef NAG_COLUMN_MAJOR
    pdx = n;
    #define X(I, J) x[(J-1)*pdx + I-1]
#else
    pdx = m;
    #define X(I, J) x[(I-1)*pdx + J-1]
#endif
if (!(b = NAG_ALLOC(m+1, double)) ||
    !(perr = NAG_ALLOC(5, double)) ||
    !(res = NAG_ALLOC(n, double)) ||
    !(vif = NAG_ALLOC(m, double)) ||
    !(x = NAG_ALLOC(pdx*(order == Nag_RowMajor?n:m, double)) ||
    !(y = NAG_ALLOC(n, double)) ||
    !(isx = NAG_ALLOC(m, Integer))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
for (i = 1; i <= n; i++)
{
    for (j = 1; 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-1]);
    #else
        scanf("%lf ", &y[i-1]);
    #endif
}

/* Total number of variables.*/
ip = 0;
for (j = 0; j < m; j++)
{
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    for (j = 0; j < m; j++)
        #ifdef _WIN32
            scanf_s("%"NAG_IFMT " , &isx[j]);
        #else
            scanf("%"NAG_IFMT " , &isx[j]);
        #endif
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
}
if (isx[j] == 1)
    ip = ip+1;
}
#endif NAG_COLUMN_MAJOR
pdb = n;
pdres = n;
pdvif = ip;
#else
pdb = one;
pdres = one;
pdvif = one;
#endif
/* Tolerance for setting singular values of H to zero.*/
tau = 0.00e0;
df = 0;
/* Call function.*/
/*
* nag_regsn_ridge_opt (g02kac)
* Ridge regression
*/
nag_regsn_ridge_opt(order, n, m, x, pdx, isx, ip, tau, y, &h, opt, &niter,
    tol, &nep, orig, b, vif, res, &rss, &df, optloo, perr, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_regsn_ridge_opt (g02kac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print results:*/
printf("\n");
printf("%s %10.4f\n", "Value of ridge parameter:", h);
printf("\n");
printf("%s %13.4e\n", "Sum of squares of residuals:", rss);
printf("%s %5"NAG_IFMT"\n", "Degrees of freedom:", df);
printf("%s %10.4f\n", "Number of effective parameters:", nep);
printf("\n");
ipl = ip + 1;
/*
* nag_gen_real_mat_print (x04cac)
* Print real general matrix (easy-to-use)
*/
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, ipl, one,
    b, pdb, "Parameter estimates", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
printf("%s"NAG_IFMT"\n", "Number of iterations:", niter);
printf("\n");
if (opt == Nag_GCV)
    printf("%s", "Ridge parameter minimises GCV");
else if (opt == Nag_UEV)
    printf("%s", "Ridge parameter minimises UEV");
else if (opt == Nag_FPE)
    printf("%s", "Ridge parameter minimises FPE");
else if (opt == Nag_BIC)
    
Mark 25 g02kac.9
printf("%s\n", "Ridge parameter minimises BIC");
}
printf("%n");
printf("%s\n", "Estimated prediction errors:");
printf("%s %10.4f\n", "GCV =", perr[0]);
printf("%s %10.4f\n", "UEV =", perr[1]);
printf("%s %10.4f\n", "FPE =", perr[2]);
printf("%s %10.4f\n", "BIC =", perr[3]);
if (optloo == Nag_WantLOO)
{
    printf("%s %10.4f\n", "LOO CV =", perr[4]);
}
printf("%n");

/*
 * nag_gen_real_mat_print (x04cac)
 * Print real general matrix (easy-to-use)
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, one,
res, pdres, "Residuals", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
go to END;
}
printf("%n");

/*
 * nag_gen_real_mat_print (x04cac)
 * Print real general matrix (easy-to-use)
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, ip, one,
vif, pdvif, "Variance inflation factors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
go to END;
}

END:
NAG_FREE(b);
NAG_FREE(perr);
NAG_FREE(res);
NAG_FREE(vif);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(isx);

return exit_status;
}

10.2 Program Data

nag_regsn_ridge_opt (g02kac) Example Program Data
20 3 0.5 Nag_GCV 1.0e-4 25
Nag_EstimatesStand
Nag_WantLOO : n, m, h, opt, tol, niter, orig, optloo
19.5 43.1 29.1 11.9
24.7 49.8 28.2 22.8
30.7 51.9 37.0 18.7
29.8 54.3 31.1 20.1
19.1 42.2 30.9 12.9
25.6 53.9 23.7 21.7
31.4 58.5 27.6 27.1

NAG Library Manual
10.3 Program Results

nag_regsn_ridge_opt (g02kac) Example Program Results

Value of ridge parameter:    0.0712
Sum of squares of residuals:  1.0917e+02
Degrees of freedom:             16
Number of effective parameters:  2.9059

Parameter estimates
   1   20.1950  
   2    9.7934  
   3    9.9576  
   4   -2.0125  

Number of iterations:     6

Ridge parameter minimises GCV

Estimated prediction errors:
   GCV  =   7.4718  
   UEV  =   6.3862  
   FPE  =   7.3141  
   BIC  =   8.2380  
   LOO CV =   7.5495  

Residuals
   1   -1.9894  
   2    3.5469  
   3   -3.0392  
   4   -3.0309  
   5   -0.1899  
   6   -0.3146  
   7    0.9775  
   8    4.0157  
   9    2.5332  
  10   -2.3560  
  11    0.5446  
  12    2.3989  
  13   -4.0876  
  14    3.2778  
  15    0.2894  
  16    0.7330  
  17   -0.7116  
  18   -0.6092  
  19   -2.9995  
  20    1.0110  

: End of data
1 1 1  
: isx
<table>
<thead>
<tr>
<th></th>
<th>Variance inflation factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2928</td>
</tr>
<tr>
<td>2</td>
<td>0.4162</td>
</tr>
<tr>
<td>3</td>
<td>0.8089</td>
</tr>
</tbody>
</table>