AN ALGORITHM FOR THE COMPUTATION OF GENERALIZED LIKELIHOOD OR SELF-CRITIC. (U) RENSSELAER POLYTECHNIC INST TROY NY SCHOOL OF MANAGEMENT T A DELAHANTY ET AL.

UNCLASSIFIED 1986 WP-37-06-P26 ARO-19072.20-MA F/G 9/2 NL
AN ALGORITHM FOR THE COMPUTATION OF GENERALIZED LIKELIHOOD OR SELF-CRITICAL ESTIMATORS FOR BINARY DATA

by

T. A. Delahanty
and
A. S. Paulson

School of Management
Rensselaer Polytechnic Institute
Troy, NY 12180-3590
(518)266-6586

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

Note: This paper is not to be quoted without the permission of the author(s).
For a list of Working Papers available from the School of Management, please contact Sheila Chao, School of Management.
An Algorithm for the Computation of Generalized Likelihood or Self-Critical Estimators for Binary Data

by

T.A. Delehanty*

and

A.S. Paulson*
Rensselaer Polytechnic Institute

1. Purpose
2. Specification (FORTRAN)
3. Description
4. Numerical Method
5. Parameters
6. Error Indicators and Warnings
7. Auxiliary Routines
8. References
9. Storage
10. Precision, Machine Dependent Constants
11. Further Comments
12. Example
1. **Purpose**

Subroutine **BINARY** is an implementation of the self-critical estimation procedure of Paulson, Presser and Lawrence (1983). The logistic, Gaussian or Type I extreme value distribution may be selected as tolerance distribution. Estimates are expressed in location-scale form on entry and exit, but results may be printed out in regression form, location-scale form, or in both forms. (see Section 3 for a description of the two parametrizations). It is possible to hold location parameters constant during the estimation procedure. Estimation is accomplished by a Newton-Raphson method.

2. **Specification (FORTRAN)**

```fortran
SUBROUTINE BINARY (N, IX, X, IA, NPAR, ISUB, ISTART, IDEP, IDIST, C, RELTOL, ABSTOL, MAXIT, IPRINT, IFLAG, BETA, XLOG, ICOV, COV, LMEM, MEMORY, IFAULT)
```

3. **Description**

The routine is applicable to the following modeling situation: For \( i = 1, \ldots, n \) (\( n \) is the sample size), let \( v_i \) be the stress variable, \( a_i \) a zero-one indicator of withstand or failure, and \( X_i = (x_{i1}, \ldots, x_{ip})^T \) a column vector of covariates. A constant is incorporated as a covariate identically equal to unity. The case \( p=0 \) is possible, but should be rare.

The tolerance distribution and density, \( F(v_i) \) and \( f(v_i) \) respectively, depend on the covariates, a scale parameter \( \sigma \), and a vector of location parameters \( \beta = (\beta_1, \ldots, \beta_p)^T \) as follows, where
\[ u_1 = \frac{v_1 - \beta'X_1}{\sigma} \]

- logistic, \[ F(v_1) = \frac{\exp(u_1)}{1 + \exp(u_1)} \]
\[ f(v_1) = \frac{\exp(u_1)}{\sigma |(1 + \exp(u_1))^2} \]
- Gaussian, \[ F(v_1) = \Phi(u_1) \]
\[ f(v_1) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{1}{2} u_1^2\right) \]
- extreme value, \[ F(v_1) = 1 - \exp(-\exp(u_1)) \]
\[ f(v_1) = \frac{1}{\sigma} \exp(u_1 - \exp(u_1)) \]

In principle, it is possible for \( \sigma \) to be negative, but this contradicts the physical notion of a stress variable.

In the method of maximum likelihood, parameters \( \theta \) are estimated by maximizing the log likelihood,
\[ L(\theta) = \sum_{i=1}^{n} \left[ a_i \log F(v_i) + (1-a_i) \log S(v_i) \right] \]
where \( S = 1 - F \). The self-critical procedure depends on a user-specified quantity \( c \), and estimates \( \theta \) by solving the system
\[ \sum_{i=1}^{n} f^c(v_1) \left[ a_i \frac{\partial}{\partial \theta} \log F(v_1) + (1-a_i) \frac{\partial}{\partial \theta} \log S(v_1) \right] = 0. \]
When \( c = 0 \), maximum likelihood estimates are obtained. As \( c \) increases from zero, increasingly robust estimators result. The sensitivity of parameter estimates to departures from model assumptions can be examined by starting with \( c = 0 \) and refitting the model for increasing values of \( c \). Negative values of \( c \) seem less useful.
4. **Numerical Method**

For the purpose of estimation, the routine (internally) reparameterizes the problem in a "regression" form. In this parameterization, $P(\cdot)$ depends on

$$v_i = \alpha v_i + \mu'X_i$$

instead of

$$u_i = \frac{v_i - \beta'X_i}{\sigma}.$$  

The reparameterization offers two advantages:

1) Computation of the necessary partial derivatives is simple;

2) For maximum likelihood estimation with a logistic tolerance distribution, it can be shown that the regression parameterization results in a concave maximization problem. We anticipate that it will have fairly good properties in the more complicated cases.

It has a disadvantage in terms of potential ill-conditioning of the Hessian (or Jacobian) matrix, so that the input data should be sensibly scaled (see Section 11). Parameters are assumed to be expressed in the easier to understand location-scale form on entry, and are transformed back to location-scale form on exit, so the user need not worry about details of reparameterization.

Let

$$S_{\theta_i} = f^C(v_i)[a_1 \frac{\partial}{\partial \theta} \log F(v_i) + (1-a_1) \frac{\partial}{\partial \theta} \log S(v_i)].$$

The gradient used in Newton-Raphson iteration has $s^{th}$ component

$$g_s = n^{-1} \sum_{i=1}^{n} S_{\theta_i},$$
and the Hessian has $(\theta, \theta')$ component $S_{\theta \theta'} = n^{-1} \sum_{i=1}^{n} \frac{\partial}{\partial \theta} \frac{\partial}{\partial \theta'} S_{\theta_i},$

(the scaling by $n^{-1}$ should be helpful when $n$ is large).
When \( c = 0 \), the estimated asymptotic covariance matrix of the estimators is 
\[ n^{-1} (-H)^{-1} \], while when \( c \neq 0 \) it is 
\[ n^{-1} H^{-1} V H^{-1} , \] 
where \( V \) has \((0, \theta')\) component 
\[ V_{0\theta} = n^{-1} \sum_{i=1}^{n} S_{0i} S_{0i}' . \] 
The estimated asymptotic covariance matrix is expressed in location-scale form by pre- and post-multiplying it by the Jacobian of the underlying parameter transformation.

Newton-Raphson iteration converges when 
\[ \max_{1} \left| e^{(k)} - e^{(k-1)} \right| < ABSTOL \]
and 
\[ \max_{1} \left| \frac{e^{(k)}}{e^{(k-1)}} \right| < RELTOL , \]
where superscripts represent iteration numbers and \( ABSTOL \) and \( RELTOL \) are user-supplied. The user also specifies a maximum allowable number \( MAXIT \) of iterations.

Solution of linear equations and matrix inversion is accomplished by subroutines \( \text{DECOMP} \) and \( \text{SOLVE} \), taken from Forsythe, Malcolm and Moler (1977). These routines are of high numerical quality, and provide an estimate of the condition number of the input matrix, which is often useful. In the interest of portability, iterative improvement is not used.

5. Parameters

5.1 Input Parameters

\( N \) - INTEGER
Sample size. Unchanged on exit.

\( IX \) - INTEGER
Row dimension of data matrix \( X \). Unchanged on exit.
X - REAL array of DIMENSION (IX, q), where q > N. Data matrix containing the dependent (stress) variable and all covariates. Each column corresponds to one observation. If parameters are held fixed, values of the dependent variable will be changed, but their input values restored. Unchanged on exit.

IA - INTEGER array of DIMENSION N. Withstand or failure is indicated for observation I according as IA(I) = 0 or IA(I) ≠ 0. Unchanged on exit.

NPAR - INTEGER. Number of parameters in the model. Unchanged on exit.

ISUB - INTEGER array of DIMENSION (NPAR). Indicates rows of data matrix X corresponding to parameters (the scale parameter is taken to correspond to the dependent variable). For instance, if ISUB(3) = 5, the third parameter corresponds to the fifth row of X. Unchanged on exit.

ISTART - INTEGER array of DIMENSION (NPAR). Indicates the status of parameters in the model, and whether a starting value is to be supplied. If ISTART(I) is equal to
0, BETA(I) is to be estimated, and its input value is to be disregarded;
1, BETA(I) is to be estimated, and its input value is to be used as starting value;
2, BETA(I) is to be held constant at its input value.
(See Section 11.2 for comments on starting values; the scale of parameter is not allowed to be held constant.) Unchanged on exit.

IDEP - INTEGER. Row of dependent (stress) variable in X. Unchanged on exit.

IDIST - INTEGER. Indicates the tolerance distribution desired. If IDIST equals
1, logistic distribution will be used;
2, Gaussian distribution will be used;
3, extreme value distribution will be used.
Unchanged on exit.

RELTOl - DOUBLE PRECISION.
Relative convergence tolerance for Newton-Raphson iteration (see Section 4).
Unchanged on exit.

MAXIT - INTEGER.
Maximum allowable number of Newton-Raphson iteration.
Unchanged on exit.

IPRINT - INTEGER.
Output unit number. If IPRINT < 0, no output will be produced. If IPRINT > 0, standard output will be produced on logical output unit IPRINT.
Unchanged on exit.

IFLAG - INTEGER.
Indicator for form of output. If IFLAG < 0, the standard output summary will be in regression form. If IFLAG = 0, output summaries will be produced for both regression and location-scale parameterizations. If IFLAG > 0, the standard output summary will be in location-scale form. (See Section 11.3).
Unchanged on exit.

5.2 Input/Output (and associated dimension) Parameters.

BETA - DOUBLE PRECISION array of DIMENSION (NPAR).
On entry, contains starting values as specified by ISTART.
On exit, contains parameter estimates, in location scale form.

XLOGL - DOUBLE PRECISION.
On exit, contains the log likelihood if c = 0 (maximum likelihood estimation), and zero otherwise.

ICOV - INTEGER.
Row dimension for COV.
Unchanged on exit.

COV - DOUBLE PRECISION array of DIMENSION (ICOV, q), where q >= NPAR.
On exit, estimated asymptotic covariance information for the parameters. The diagonal contains standard errors, the strict lower triangle correlations, and the strict upper triangle covariances. If a parameter is held constant, all corresponding entries are zero. The covariance matrix will be for the location-scale parameterization rules IFLAG < 0, when it will be in regression form. See Section 11.3.
5.3 Workspace (and associated dimension) parameters

**LMEM** - INTEGER.
Length of work array MEMORY, as declared in calling program unit.
LMEM = NPAR + 4*NACT*(1+NACT) + 2*max(NFIX, 2*NACT), where
NACT is the number of parameters estimated and NFIX = NPAR-NACT
is the number of parameters held constant.
Unchanged on exit.

**MEMORY** - INTEGER array of DIMENSION (LMEM).
Used as workspace.

5.4 Diagnostic Parameter

**IFAULT** - INTEGER.
Unless the routine finds an error or gives a warning, IFAULT
will be 0 on exit. See Section 6.

6. Error Indications and Warnings

Errors or warnings specified by the routine:

**IFAILT < 0** If IFAULT = -I, I > 0, then an arithmetic exception was about to
occur on observation I, while computing partial derivatives. This
failure should be rare. If the data have been sensibly scaled, and
the starting values are not too bad (see Sections 11.1-11.2), the
probable cause is a data error. The routine stops as soon as the
error is detected, and output parameter values are not of interest.

**IFAILT > 0** Input parameter outside expected range. This failure will occur if,
on entry, N < 1, NPAR < 1, IX < NPAR, ICOV < NPAR, RELTOL ≤ 0,
ABSTOL < 0, MAXIT < 1, IDIST < 1, IDIST > 3, IFLAG ≤ 0 and
IPRINT ≤ 0 (see Section 11.3), ISTART(I) < 0 or ISTART(I) > 2 for
some I, ISTART(I) = 2 for all I, ISUB(I) ≠ IDEP for all I, or
ISUB(I) = IDEP and ISTART(I) = 2 for some I. (The last restriction
means that the scale parameter cannot be held constant.) The
routine stops without doing any calculations.

**IFAILT = 2** Insufficient workspace. This failure will occur if, on entry,
LMEM < NPAR + 4*NACT*(1+NACT) + 2*max(NFIX, 2*NACT), where
NACT is the number of parameters estimated (ISTART(I) < 2), and
NFIX = NPAR - NACT is the number of parameters held constant
(ISTART(I) = 2). The routine stops without doing any calculations.

**IFAILT = 3** The Hessian matrix has become numerically singular during Newton-
Raphson iteration. The routine transforms estimates to location-
scale form, restores values of the dependent variable if parameters
were held fixed, and stops. Output parameter values are not
generally of interest.
The Newton-Raphson iteration did not converge to within RELTOL and ABSTOL in the specified MAXIT iterations. The routine transforms estimates to location-scale form, restores values of the dependent variable if parameters were held fixed, and stops. Output parameter values are not generally of interest.

7. Auxiliary Routines

SUBROUTINE PREPAR(IX, N, X, NPAR, BETA, SUB, ISTART, NACT, IACT, IMAP, PAR, NFIX, IFIXED, IDEP, WORK)

Prepares for iteration by setting up some indexing and working arrays, subtracting the effects of fixed parameters from the dependent variable, and setting initial active parameter values in regression form.

SUBROUTINE NEWTON(C, MAXIT, RELTOL, ABSTOL, DIFFER, XLOGL, NACT, IACT, PAR, IPIVOT, HESS, HESFAC, N, IX, X, IA, IPRINT, LWORX, WORK, IDAULT)

Carries out the Newton-Raphson iteration.

SUBROUTINE VARIAB(C, XLOGL, REGRES, DIFFER, VROOT, NACT, IACT, IMAP, PAR, IPIVOT, HESS, HESFAC, ICOV, NPAR, COV, DET, N, IX, X, IA, WORK)

Computes the estimated asymptotic covariance matrix, transforms it to location-scale form if requested, and computes correlations and standard errors.

SUBROUTINE EXPOST(REGRES, IDEP, NPAR, BETA, ISTART, NACT, PAR, IMAP, N, IX, X, NFIX, IFIXED, WORK)

Sets the output vector BETA, transforms to location-scale form if requested. If location-scale form is requested and parameters were held fixed, restore initial values of dependent variable.

SUBROUTINE RESULT(C, REGRES, IPRI, NT, D, IDEP, NACT, NFIX, XLOGL, NPAR, BETA, ISUB, ISTART, ICOV, COV, DET)

Produces a standard output summary on logical output unit IPRINT.

The following routines are specific to particular tolerance distributions, and are declared EXTERNAL in BINARY. The first three, prefixed D, are passed to NEWTON and VARIAB. The next three, prefixed V, are passed to VARIAB.
SUBROUTINE DLLOG3N(C, OBJECT, NACT, IACT, PAR, GRAD, HESS, N, IX, X, IA, IPFAULT)
SUBROUTINE DGAUUN ( )
SUBROUTINE DZXVBN ( )

Compute gradient and Hessian for logistic, Gaussian and extreme value models, resp.

SUBROUTINE VLOGBN(C, NACT, IACT, PAR, V, N, IX, X, IA)
SUBROUTINE VGAUUN ( )
SUBROUTINE VXUBN ( )

Compute V factor of estimated asymptotic covariance matrix for logistic, Gaussian and extreme value models, resp.

The following procedures perform general numerical tasks. They have been included in the interest of portability, although equivalents exist on many computer systems.

SUBROUTINE DECOMP(NDIM, N, A, UL, COND, IPUT, WORK)
SUBROUTINE SOLVE(NDIM, N, A, B, X, IPUT)

DECOMP decomposes a matrix into LU factors and estimates its condition. SOLVE solves a linear system, using the results of DECOMP. These routines are in Forsythe, Malcolm and Moler (1977), but the present versions include an extra argument so that A and B need not be overwritten.

SUBROUTINE DNXMULT(A, IA, N1, B, IB, N2, C, IC, N3, WORK, LWORK, IFLAG, IPFAULT)

Double precision matrix multiplication -
A(N1 x N3) = B(N1 x N2) * C(N2 x N3), where B is overwritten if IFLAG < 0 and c is overwritten if IFLAG > 0.

DOUBLE PRECISION FUNCTION ALNORM (X, UPPER)
Algorithm as 66 (Hill, 1973) to compute tail areas of the standard Normal curve.

DOUBLE PRECISION FUNCTION RMILLS(X)
Computes Z(x)/Q(x), the reciprocal of Mills' ratio, where x is a standard Normal variate. This procedure is based on procedures by Hill (1973) and Adams (1969).
8. References


9. Storage

There are no internally declared arrays.

10. Precision, Machine Dependent Constants

The routine was developed on an IBM computers. The data matrix is single precision to conserve storage. To convert to single precision, take the following steps, with the exception noted below:

1) Change all DOUBLE PRECISION declarations to REAL;

2) Replace references to double precision FORTRAN library functions
   with single precision versions, e.g., EXP replaces DEXP;

3) Replace double precision constants by their single precision versions, e.g., 1.0 replaces 1.000.

Note: In routines PREPAR and EXPOST, the dependent variable is adjusted for the effect of fixed parameters. These adjustments must be calculated in double precision
to avoid a loss of significant digits in \( X \) which could affect subsequent computations.

Procedures DLOGBN, VLOGBN, DGAUBN, VAGUBN, DEXUBN, VEXUBN, DECOMP, ALNORM and RMILLS use machine-dependent constants whose values are set in DATA statements. These constants may have to be altered for some computers. They are pointed out by comments in the program units.

11. Further Comments

11.1 Scaling, conditioning of Hessian matrix

The user should be aware of potential problems of ill-conditioning. Because a regression-type parameterization is used, calculation of the Hessian matrix involves operations similar to the formation of \( X^T X \), where \( X \) is the data matrix. Unfortunately, the resulting Hessian is often rather ill-conditioned. If IPRINT > 0, an estimate of the Hessian's condition number is printed out at each iteration. Roughly speaking, a condition number in excess of \( 10^7 \) is worrisome, although condition numbers in excess of \( 10^{10} \) can be tolerated when working in double precision on an IBM 3081. The user can and should avoid potentially excessive ill-conditioning by scaling the data matrix \( X \) before calling the routine.

Although it is not known how to optimally scale a problem, it seems that a nearly ideal scaling will be achieved if nonconstant variables are transformed to the range \([-1, 1]\), centered at zero. However, such precise scaling is often tedious. It seems most important to "equilibrate" the data matrix so that all variables have roughly the same (moderate) magnitude. Such equilibration is often simply accomplished by dividing by suitable powers of 10, e.g., expressing voltages in megavolts instead of kilovolts. Centering the variables will further reduce the condition number. One often centers covariates anyway, so that the "intercept"
parameter has a clear interpretation.

11.2. Starting values

Since Newton-Raphson procedure is employed, starting values are important. If no starting values are supplied by the user (ISTART(I) = 0 for all I), the routine will use unity for the scale parameter and zero for all location parameters. These starting values will not be acceptable unless the dependent variable has been centered and scaled. It is recommended that the user employ the mean and standard deviation of the dependent variable as starting values for the "intercept" parameter (if any) and scale parameter, respectively. If the model is sequentially refit with different values of $c$, it is recommended that estimates from the most recent call be used as starting values for the next call.

11.3 Output Flags

Because the routine will generally be called sequentially with different values of $c$, it is most convenient to always express entry and exit parameter values in the same form. The location-scale form is used, because that form affords the clearest interpretation. An inconsistency arises of IFLAG $<$ 0, for then the output estimates are in location-scale form, but the covariance matrix is in regression form. Thus, if the user wants to make separate use of the output parameters of BINARY, the routine should be called with IFLAG $>$ 0.

It is recommended that the routine be called with IPRINT $>$ 0. If IFLAG $<$ 0, the requirement IPRINT $>$ 0 is enforced. (When IFLAG $<$ 0, the requirement is in keeping with the inconsistency mentioned above.) The standard output summary should
be sufficient for most applications. For the maximum flexibility in output and interpretation of results, call BINARY with IPRINT > 0, IFLAG = 0.

12. Example

The following program illustrates the use of BINARY, and the standard output produced when IPRINT > 0 and IFLAG = 0.
SUBROUTINE BINARY(N, IX, X, IA, NPAR, ISUB, ISTART, IDEP, IDIST, C, RELTOL, ABSTOL, MAXIT, IPRINT, IFLAG, BETA, XLOGL, 1COV, COV, LMEM, MEMORY, IFault)

FAILURE CODES -

IFault = 1 - INPUT ERROR
IFault = 2 - INSUFFICIENT WORKSPACE SUPPLIED
IFault = 3 - HESSIAN MATRIX IS NUMERICALLY SINGULAR
IFault = 4 - NO CONVERGENCE IN MAXIT ITERATIONS
IFault = -1 - AN EXCEPTION WAS ABOUT TO OCCUR WHILE PROCESSING THE I TH OBSERVATION
NFAIL = NPAR - NACT
C CHECK IF WORKSPACE SIZE IS ADEQUATE, ALLOCATE IT
IFault = 2
IND = 2 - NACT
IND1 = IND * NACT
IF (LMAXI.T. NPAR + 2*NACT + IND + 2*IND1 + MAXO(2*NFAIL,2*IND)) RETURN
C PREPARE FOR ESTIMATION - SET UP SUBSCRIPT ARRAYS AND
C STARTING VALUES FOR ACTIVE PARAMETERS, SUBTRACT EFFECT OF
C FIXED PARAMETERS FROM DEPENDENT VARIABLE
CALL PREPAR(NX, N, XP, NPAR, PTHRM, ISUB, ISTART, NACT,
1 MEMORY(MIAC), MEMORY(MPUN), MEMORY(MP), NFAIL,
2 MEMORY(NFAILX), IDEP, MEMORY(NWORK))
C NEWTON-RAPHSON ITERATION WITH EXTERNAL ROUTINE PASSED FOR
C FIRST AND SECOND PARTIALS
IF (IDIST .EQ. 1) CALL NEWTON(C, NACT, RELTOL, ABSTOL, DLOGBN,
1 XLOGL, NACT, MEMORY(MIAC), MEMORY(MP), MEMORY(MIPV),
2 MEMORY(MP), MEMORY(MIAC), N, IX, X, IA, IPRINT, 2*NACT,
3 MEMORY(MWORK), IFault)
IF (IDIST .EQ. 2) CALL NEWTON(C, NACT, RELTOL, ABSTOL, DLOGBN,
1 XLOGL, NACT, MEMORY(MIAC), MEMORY(MP), MEMORY(MIPV),
2 MEMORY(MP), MEMORY(MIAC), N, IX, X, IA, IPRINT, 2*NACT,
3 MEMORY(MWORK), IFault)
IF (IDIST .EQ. 3) CALL NEWTON(C, NACT, RELTOL, ABSTOL, DLOGBN,
1 XLOGL, NACT, MEMORY(MIAC), MEMORY(MP), MEMORY(MIPV),
2 MEMORY(MP), MEMORY(MIAC), N, IX, X, IA, IPRINT, 2*NACT,
3 MEMORY(MWORK), IFault)
C ERROR HANDLING IF NEWTON-RAPHSON PROCEDURE FAILS.
ON ERROR, PRINT OUT MESSAGE IF IPRINT GT. 0, SET UP
THE FINAL CALL TO EXPOSTI()
IF (IIFault .EQ. 0) GO TO 30
REGRES = FALSE
IF (IPrint LE. 0) GO TO 50
IF (IIFault GT. 0) GO TO 20
IND = -IIFault
WRITE (IPrint,60) IND
GO TO 50
20 IIFault = IIFault + 1
IF (IIFault LE. 2) WRITE (IPrint,70)
GO TO 50
C COMPUTE APPROXIMATE ASYMPTOTIC COVARIANCE MATRIX.
C REGRES IS A FLAG FOR WHETHER OR NOT TO SET UP OUTPUT IN
C REGRESSION FORM; STATEMENT 30, THE FIRST BRANCH POINT,
C SETS UP REGRES FOR FIRST OUTPUT (IF REGRESSION AND
C LOCATION-SCALE BOTH REQUESTED, REGRESSION GOES FIRST).
C STATEMENT 40, THE SECOND BRANCH POINT, IS FOR REPEAT
C
CALL (ALWAYS LOCATION-SCALE)
30 REGRES * IFLAG LE 0
40 IF (IDIST EQ 1) CALL VARIAB(C, XLOGN, REGRES, DLOGBN, VLOGBN, ILOGBN, ILOGB)
1 NACT, MEMORY(MIAC), MEMORY(MIMAP), MEMORY(MIPAR),
2 MEMORY(MIPVT), MEMORY(MHES), MEMORY(MHFA), ICOV, NPAR, COV, IBOUND)
3 DET, N, IX, XI, IA, MEMORY(MIWORK)
1 NACT, MEMORY(MIAC), MEMORY(MIMAP), MEMORY(MIPAR),
2 MEMORY(MIPVT), MEMORY(MHES), MEMORY(MHFA), ICOV, NPAR, COV, IBOUND)
3 DET, N, IX, XI, IA, MEMORY(MIWORK)
1 NACT, MEMORY(MIAC), MEMORY(MIMAP), MEMORY(MIPAR),
2 MEMORY(MIPVT), MEMORY(MHES), MEMORY(MHFA), ICOV, NPAR, COV, IBOUND)
3 DET, N, IX, XI, IA, MEMORY(MIWORK)
50 CALL EXPOST(REGRES, IDEP, NPAR, BETA, ISTART, NACT, MEMORY(MIPAR),
1 MEMORY(MIPMAP), N, IX, X, NFIX, MEMORY(MIFIX), MEMORY(MIWORK))
C IF REGRES = NOT REGRES IF ( NOT REGRES AND IFLAG EQ 0) GO TO 40
C IF ONLY REGRESSION FORM OUTPUT WAS REQUESTED, AN EXTRA
C CALL TO EXPOST() IS NEEDED TO RESTORE LOCATION-SCALE FORM
C IF (IFLAG LT 0) CALL EXPOST(REGRES, IDEP, NPAR, BETA, ISTART,
1 NACT, MEMORY(MIPAR), MEMORY(MIPMAP), N, IX, X, NFIX,
2 MEMORY(MIFIX), MEMORY(MIWORK))
C 60 FORMAT ('OFailure - exception on observation no.', F15.0)
70 FORMAT ('OFailure - hessian matrix is numerically singular')
80 FORMAT ('OWarning - no convergence in', I4, ' iterations - possible
1E FAILURE')
C PREPARE FOR ANALYSIS BY SETTING UP THE FOLLOWING ARRAYS -
C IACT(*) - X(*) rows for active parameters
C IFIX(*) - X(*) rows for fixed parameters
C IMAP(*) - positions in BETAS(*) of active parameters
C PAR(*) - initial values of active parameters in
C REGRESSION FORM
C ALSO SUBTRACT EFFECTS OF FIXED PARAMETERS FROM DEPENDENT
C VARIABLE, USING WORK(*)
C NOTE - DEPENDENT VARIABLE (SCALE) PARAMETER IS ALWAYS
C PLACED FIRST IN PAR(*), FOR CONVENIENCE LATER
C SUBROUTINE PREPAR(I, N, X, NP, BETA, ISUB, ISTART, NACT, IACT,
C 1 IBOUND, PAR, NFIX, IFIXED, IBOUND, DWORK)
C ARGUMENTS

PREP0001
PREP0002
PREP0003
PREP0004
PREP0005
PREP0006
PREP0007
PREP0008
PREP0009
PREP0010
PREP0011
PREP0012
PREP0013
PREP0014
PREP0015
INTEGER IX, N, NPAR, ISUB(NPAR), ISTART(NPAR), NACT, IACT(NACT), 
1 IMAP(NACT), NFIX, IFIXED(1), IDEF 
PREP0016
REAL X(IX,N) 
PREP0017
DOUBLE PRECISION BETA(NPAR), PAR(NPAR), WORK(1) 
PREP0018
LOCAL SCALARS 
PREP0019
INTEGER IN1, IND1, LAC'T, ISCALE 
PREP0020
DOUBLE PRECISION TEMP, ZERO, ONE 
PREP0021
DATA ZERO, ONE /0.000, 1.000/ 
PREP0022
C 
C SET UP IACT(*), IFIXED(*), IMAP(*) 
C LACT = 0 
PREP0023
IND = 0 
PREP0024
ISCALE = 0 
PREP0025
DO 20 I = 1, NPAR 
PREP0026
IF (ISTART(I) .EQ. 2) GO TO 10 
PREP0027
LACT = LACT + 1 
PREP0028
IMAP(LACT) = I 
PREP0029
IND1 = ISUB(I) 
PREP0030
IACT(LACT) = IND1 
PREP0031
IF (IND1 .EQ. IDEF) ISCALE = 1 
PREP0032
GO TO 20 
PREP0033
10 IND = IND + 1 
PREP0034
IFIXED(IND) = ISUB(I) 
PREP0035
WORK(IND) = BETA(I) 
PREP0036
PREP0037
PREP0038
PREP0039
20 CONTINUE 
PREP0040
C 
C SWITCH PLACES SO SCALE IS FIRST ACTIVE PARAMETER 
C IND = IACT(1) 
PREP0041
IACT(1) = IACT(ISCALE) 
PREP0042
IACT(ISCALE) = IND 
PREP0043
IND = IMAP(1) 
PREP0044
IMAP(1) = IMAP(ISCALE) 
PREP0045
IMAP(ISCALE) = IND 
PREP0046
C 
C MAIN LOOP OVER SAMPLE IF PARAMETERS ARE FIXED, TO ADJUST 
C DEPENDENT VARIABLE (TO BE RETURN ON EXIT FROM BINARY()) 
C IF (NFX).EQ.0) GO TO 50 
C DO 40 I = 1, N 
PREP0047
C 
C THE FOLLOWING INNER PRODUCT MUST BE ACCUMULATED IN DOUBLE 
C PRECISION ... 
C TEMP = DBLE(X(IDCIP,1)) 
C DO 30 J = 1, NFX 
C IND = IFIXED(J) 
C X(IDCIP,1) = TEMP - WORK(J) * X(IND,1) 
C 30 CONTINUE 
C 40 CONTINUE 
C C SET STARTING VALUES 
C 50 DO GO I = 1, NPAR 
C PAR(I) = ZERO 
PREP0048
C 60 TEMP = ONE 
PREP0049
C IND = IMAP(1) 
PREP0050
IF (ISTART(IND) .EQ. 1 AND BETA(IND) .NE. ZERO) TEMP = BETA(IND) 
PREP0051
PAR(I) = ONE / TEMP 
PREP0052
IF (NACT .EQ. 1) RETURN 
PREP0053
DO 70 I = 2, NACT 
PREP0054
IND = IMAP(I) 
PREP0055
PREP0056
PREP0057
PREP0058
PREP0059
PREP0060
PREP0061
PREP0062
PREP0063
PREP0064
PREP0065
PREP0066
PREP0067
PREP0068
PREP0069
PREP0070
PREP0071
PREP0072
PREP0073
PREP0074
PREP0075
IF (ISTRT(IND) EQ. 1) PAR(I) = -BETA(IND) / TEMP
70 CONTINUE
C
RETURN
END

C**************************************************************
C NEWTON-RAPHSON ITERATION
C**************************************************************
C
SUBROUTINE NEWTON(C, MAXIT, RELTOL, ABSTOL, DIFFER, XLOGL, NACT, NEWTO11
1 IACT, PAR, IPIVOT, HESS, HESFAC, N, IX, IA, IPRINT, NEWTO12
2 LWORK, WORK, IFault)
C
INTEGER MAXIT, NACT, IACT(NACT), IPIVOT(NACT), N, IX, IA(N).
C
REAL X(IX,N)
C
DOUBLE PRECISION C, RELTOL, ABSTOL, XLOGL, PAR(NACT),
1 HESS(NACT,NACT), HESFAC(NACT,NACT), WORK(LWORK)
C
LOCAL SCALARS
C
INTEGER ITER, IND
C
DOUBLE PRECISION GNORM, COND, ABSCERR, RELERR, TEMP, XNEW, ZERO.
C
1 IF (ONE .EQ. 0) WRITE (IPRT,50) C, MAXIT, RELTOL, ABSTOL
2 DATA ZERO, ONE /O 0,0, 1 0,0/
C
1 IFault = 1
2 IF (LWORK LT 2*NACT) RETURN
3 IFault = 0
4 IND = NACT + 1
5 ITER = 0
6 IF (IPRT GT 0) WRITE (IPRT,50) C, MAXIT, RELTOL, ABSTOL
C
7 LOOPING POINT FOR ITERATION - THE FIRST NACT LOCATIONS OF
8 WORK(*) ARE USED FOR GRADIENT/INCREMENT, WHILE THE NEXT
9 NACT LOCATIONS ARE WORKSPACE FOR DECOMP
C
10 ITER = ITER + 1
11 CALL DIFFER(C, XLOGL, NACT, IACT, PAR, WORK(1), HESS, N, IX, X,
12 IA, IFault)
13 IF (IFault LT 0) RETURN
C
14 FACTOR THE HESSIAN INTO L * U, CHECK IF SINGULAR
15 CALL DECOMP(NACT, NACT, HESS, HESFAC, COND, IPIVOT, WORK(IND))
16 IF (COND + ONE NE COND) GO TO 20
17 IFault = 2
18 RETURN
C
19 SOLVE THE SYSTEM HESS * INCREMENT = -GRADIENT.
20 GNORM = ZERO
21 DO 30 I = 1, NACT
22 TEMP = WORK(1)
23 WORK(1) = TEMP
24 TEMP = WORK(1)
25 WORK(1) = TEMP
26 GNORM + GNORM + TEMP * TEMP
27 DO 30 I = 1, NACT
28 CONTINUE
29 GNORM = DSQRT(GNORM)
30 CONTINUE

C**************************************************************
CALL SOLVE(NACT, NACT, HESFAC, WORK, WORK, IPIVOT)

C INCREMENT PARAMETERS, COMPUTE CONVERGENCE CRITERIA
RELERR = ZERO
ABSERR = ZERO
DO 40 I = 1, NACT
   TEMP = WORK(I)
   ABSERR = OMAX1(ABSERR, DABS(TEMP))
   XNEW = PAR(I) + TEMP
   PAR(I) = XNEW
   IF (ONE + XNEW .LE. ONE) TEMP = TEMP / XNEW
   RELERR = OMAX1(RELERR, DABS(TEMP))
40 CONTINUE
IF (IPRINT GT 0) WRITE (IPRINT, 60) IER, GNORM, RELERR, ABSERR
ICOND
C CHECK CONVERGENCE, SET FAILURE CODE IF NONE
IF (RELERR .LT. RELTOL AND ABSERR .LT. ABSTOL) RETURN
IF (ITER .LT. MAXIT) GO TO 10
IFault = 3
C
50 FORMAT (' 1 C MAX ITNS REL TOLER ABS TOLER'/11.3, 19., 1D11.3, 19.
1 2011 2)
60 FORMAT (' 0 ITN GRAD NORM REL CHANGE ABS CHANGE'/15. 3D12. 2, 1.
1 ' ESTIMATED CONDITION NUMBER OF HESSIAN IS', 10.2)
RETURN
END

C******************************************************************************
C COMPUTATIONS FOR VARIABILITY OF THE ESTIMATORS
C ESTIMATE ASYMPOTIC COVARIANCE MATRIX, TRANSFORM IT FROM
C REGRESSION TO LOCATION-SCALE FORM, COMPUTE ASYMPOTIC
C CORRELATIONS AND STD. ERRORS
C******************************************************************************
SUBROUTINE VARIAB(C, XLGL, REGRES, DIFFER, VROUT, NACT, IACT,
1 IMAP, PAR, IPIVOT, HESS, HESFAC, ICOV, NPAR, COV, DET,
2 N, IX, X, IA, WORK)
C ARGUMENTS - DIFFER AND VROUT ARE SUBROUTINES
C LOGICAL REGRES
C INTEGER IACT(NACT), IMAP(NACT), IPIVOT(NACT), ICOV, NPAR, N
C REAL IX,IX(N)
C DOUBLE PRECISION C, XLGL, PAR(NACT), HESS(NACT), NACT,
1 HESFAC(NACT), COV(ICOV,NPAR), DET, WORK(NACT)
C LOCAL SCALARS
C LOGICAL FLAG
C INTEGER INDX, INDX, INDX2
C DOUBLE PRECISION ALPHA, ALPHA2, TEMP, TEMPL, ZERON, ONE, SMALL
1 #10018
C MACHINE-DEPENDENT CONSTANT - SMALL SET SO THAT DEXP(X)
1 #10020
C WILL CAUSE EXCEPTION IF X LT SMALL
C DATA ZERO, ONE, SMALL /0 0 00 1 000 -180 000/
C CALL HESSIAN AT OPTIMAL POINT AND FACTOR IT
C THIS MAY BE WASTEFUL IN SOME CASES, BUT IT AVOIDS
C SOME LOGICAL COMPLICATIONS; WORK(*) USED AS SCRATCH
C THE HESSIAN IS ASSUMED NONSINGULAR, AS IT SHOULD BE IF
C THIS POINT IS REACHED, FACTORIZATION TO HESFAC(*),
C CALL DIFFER(C, XLGL, NACT, IACT, PAR, WORK, HESS, N, IX, X, IA,
1 INDX)
C CALL DECOMP(NACT, NACT, HESS, HESFAC, TEMP, IPIVOT, WORK)
C INVERT NEGATIVE OF HESSIAN, USING IPIVOT(*) AND
C FACTORIZATION IN HESFACT(*). PLACE INVERSE IN HESS(*).
DO 20 J = 1, NACT
   DO 10 I = 1, NACT
      HESS(I, J) = ZERO
      HESS(J, J) = ONE
   CALL SOLVE(NACT, NACT, HESAC, HESS(I, J), HESS(J, J), IPIVOT)
20 CONTINUE
C SECTION FOR ROBUST ANALYSIS -
C PLACE PRODUCT (H. INVERSE) * V * (H. INVERSE) IN HESS(*).
C FIRST COMPUTE VI(*).
C CALL VROUT(C, NACT, IACT, PAR, COV, N, IX, X, IA)
C MULTIPLY HESS * COV, OVERWRITING COV(*).*
C CALL DMMLT(COV, NACT, NACT, HESS, NACT, COV, NACT, NACT, COV, NACT, NACT, F800, NACT, 1, IND)
C MULTIPLY COV * HESS, OVERWRITING HESS(*).
C CALL DMMLT(HESS, NACT, NACT, COV, NACT, NACT, HESS, NACT, 1, WORK, NACT, 1, IND)
C ALL VALUES OF C - IF LOCATION-SCALE FORM, TRANSFORM
30 IF (REGRS) GO TO 80
   ALPHA = PAR(I)
   ALPHA2 = ALPHA * ALPHA
   FLAG = NACT Eq 1
C LEFT MULTIPLY HESS(*) BY JACOBIAN, RESULT TO HESFAC(*).
   DO 50 J = 1, NACT
      TEMP = HESS(I, J) / ALPHA2
   HESFAC(I, J) = -TEMP
         IF (FLAG) GO TO 50
      DO 40 I = 2, NACT
50 CONTINUE
C RIGHT MULTIPLY HESFAC(*) BY TRANSPOSE OF JACOBIAN.
C RESULT TO HESS(*).
   DO 70 I = 1, NACT
      TEMP = HESFAC(I, 1) / ALPHA2
   HESS(I, J) = -TEMP
      IF (FLAG) GO TO 70
      DO 60 J = 2, NACT
60 CONTINUE
70 CONTINUE
C BOTH PARAMETERIZATIONS - DIVIDE COVARIANCE MATRIX BY
C SAMPLE SIZE
80 TEMP = OBLE(FLOAT(N))
   DO 90 J = 1, NACT
      DO 90 I = 1, J
         HESS(I, J) = HESS(I, J) / TEMP
90 CONTINUE
C FIND DETERMINANT OF COVARIANCE MATRIX - FACTOR IT.
C RESETING IPIVOT(*) AND HESFAC(*).
VARI0034
VARI0035
VARI0036
VARI0037
VARI0038
VARI0039
VARI0040
VARI0041
VARI0042
VARI0043
VARI0044
VARI0045
VARI0046
VARI0047
VARI0048
VARI0049
VARI0050
VARI0051
VARI0052
VARI0053
VARI0054
VARI0055
VARI0056
VARI0057
VARI0058
VARI0059
VARI0060
VARI0061
VARI0062
VARI0063
VARI0064
VARI0065
VARI0066
VARI0067
VARI0068
VARI0069
VARI0070
VARI0071
VARI0072
VARI0073
VARI0074
VARI0075
VARI0076
VARI0077
VARI0078
VARI0079
VARI0080
VARI0081
VARI0082
VARI0083
VARI0084
VARI0085
VARI0086
VARI0087
VARI0088
VARI0089
VARI0090
VARI0091
VARI0092
VARI0093
CALL DECOMP(NACT, NACT, HESS, HESFAC, TEMP, IPIVOT, WORK)
DET = ZERO
IF (TEMP .EQ. TEMP) GO TO 110
IND = IPIVOT(NACT)
DO 100 I = 1, NACT
TEMP1 = HESFAC(I,NACT)
IF (TEMP1 LT ZERO) IND = -IND
DET = DET * DLOGDABS(TEMP1)
100 CONTINUE
C TAUHE THE NACT ROOT OF DETERMINANT SET DET TO ZERO
C IF IT UNDERFLOWS OR IS NEGATIVE
DET = DET / DBLE(FLOAT(NACT))
IF (DET LT SMALL OR IND LT 0) DET = ZERO
IF (DET GE SMALL AND IND GT 0) DET = DEXP(DET)
C MOVE CONTENTS OF HESS(*) TO UPPER TRIANGLE OF COV(*,*)
110 DO 120 J = 1, NPAR
DO 120 I = 1, NPAR
120 COV(I,J) = ZERO
DO 130 I = 1, NACT
IND = IMAP(I)
DO 130 J = 1, NACT
IND1 = IMAP(J)
IND2 = MINO(IND,IND1)
IND1 = MAXO(IND,IND1)
COV(IND2,IND1) = HESS(I,J)
130 CONTINUE
C PUT 510 ERRORS ON DIAG OF COV(*,*) CORRELATIONS BELOW
DO 150 I = 1, NPAR
150 TEMP = COV(I,I)
IF (TEMP .EQ. ZERO) GO TO 150
TEMP = DSORT(TEMP)
COV(I,I) = TEMP
IF (I .EQ. 1) GO TO 150
IND = 1 - 1
DO 140 J = 1, IND
 TEMP1 = COV(J,J)
 IF (TEMP1 NE ZERO) COV(I,J) = COV(J,I) / (TEMP*TEMP1)
140 CONTINUE
150 CONTINUE
C RETURN
END
C***********************************************************************
C EX POST ADJUSTMENTS BEFORE EXIT
C PUT OPTIMAL PARAMETERS IN BETA(*) IF LOCATION SCALE
C FORM, TRANSFORM THE PARAMETERS AND RESTORE DEPENDENT
C VARIABLE TO INITIAL VALUES IF PARAMETERS WERE FIXED
C***********************************************************************
SUBROUTINE EXPOST(REGRES, IDEP, NPAR, BETA, ISTART, NACT, PAR, IMA
P, IX, X, NFIX, IFIXED, WORK)
1 C LOCAL SCALARS
INTEGER IND
DOUBLE PRECISION TEMP, ONE
DATA ONE /1.0D0/
C
C INSERT WORKING PARAMETERS PAR(*) IN BETA(*), IN THE
C SUITABLE FORM.
C TEMP = PAR(I)
IND = IMAP(I)
IF (REGRES) BETA(IND) = TEMP
IF ( NOT REGRES) BETA(IND) = ONE / TEMP
IF (NACT EQ 1) GO TO 20
DO 10 I = 2, NACT
IND = IMAP(I)
IF (REGRES) BETA(IND) = PAR(I)
IF ( NOT REGRES) BETA(IND) = -PAR(1) / TEMP
10 CONTINUE
C
C IF LOCATION-SCALE FORM AND PARAMETERS WERE HELD FIXED,
C RESTORE DEPENDENT VARIABLE
C 20 IF (REGRES OR NFIX .EQ. 0) RETURN
IND = 0
DO 30 I = 1, NPAR
IF (ISTART(I) NE 2) GO TO 30
IND = IND + 1
WORK(IND) = BETA(I)
30 CONTINUE
DO 50 I = 1, N
C
C THE FOLLOWING INNER PRODUCT MUST BE ACCUMULATED IN DOUBLE
C PRECISION
TEMP = DBLE(X(IDEP,1))
DO 40 J = 1, NFIX
IND = IFIXED(J)
TEMP = TEMP + WORK(J) * X(IND,1)
40 CONTINUE
X(IDEP,1) = SNGL(TMP)
C
50 CONTINUE
C
RETURN
END
C
C---------------------------------------------------------------------
C PRINT OUT TYPICAL OUTPUT SUMMARY ON OUTPUT UNIT IPRINT
C
C SUBROUTINE RESULTIC, REGRES, IPRINT, IDIST, N, IDEP, NACT, NFIX,
C XLOGL, NPAR, BETA, ISUB, ISTART, ICID, COV, DET
C
C ARGUMENTS
C LOGICAL REGRES
C INTEGER IPRINT, IDIST, N, IDEP, NACT, NFIX, NPAR, ISUB(NPAR),
C ISTART(NPAR), ICID
C DOUBLE PRECISION C, XLOGL, BETA(NPAR), COV(ICID, NPAR), DET
C LOCAL SCALARS
C LOGICAL MLE
C INTEGER IND
C DOUBLE PRECISION TSTAT, ZERO
C DATA ZERO /0.0D0/
C
C BASIC HEADINGS
MLE = 'C EQ ZERO
IF (MLE) WRITE (IPRINT, '401')
C     FAILURE CODE - IFault = -1 - AN EXCEPTION WAS ABOUT TO
C     OCCUR WHILE PROCESSING THE I TH OBSERVATION
C**************************************************************************
C     SUBROUTINE DLOGBN(C, OBJECT, NACT, IACT, PAR, GRAD, HESS, N, IX,
C                1 X, IA, IFault)
C**************************************************************************
C     ARGUMENTS
C     INTEGER NACT, IACT(NACT), N, IX, IA(N), IFault
C     REAL X(IX,N)
C     DOUBLE PRECISION C, OBJECT, PAR(NACT), GRAD(NACT), HESS(N,NACT)
C     LOCAL SCALARS
C     LOGICAL MLE
C     INTEGER IND
C     DOUBLE PRECISION EXTRA, F, S, G, FC, EITHER, DDENS, XJ1, ZERO,
C                1 ONE, BIG
C     MACHINE-DEPENDENT CONSTANT - BIG ROUGHLY CHOSEN SO THAT
C     DEXP(X) WILL CAUSE EXCEPTION IF /X/ .GT. BIG
C     DATA ZERO, ONE, BIG /0.000, 1.000, 174.000/
C**************************************************************************
C     IFault = 0
C     MLE = ONE + C.EQ. ONE
C     EXTRA = ONE / PAR(1)
C     OBJECT = ZERO
C     DO 10 J = 1, NACT
C     GRAD(J) = ZERO
C     DO 10 I = 1, NACT
C     HESS(I,J) = ZERO
C     CONTINUE
C     MAIN LOOP OVER SAMPLE
C     DO 90 I = 1, N
C     FC = ZERO
C     DO 20 J = 1, NACT
C     IND = IACT(J)
C     FC = FC + PAR(J) * X(IND,1)
C     CONTINUE
C     IF (DABS(FC) .GT. BIG) GO TO 110
C     FC = DEXP(FC)
C     S = ONE / FC
C     F = FC / S
C     S = ONE / S
C     G = F * S
C     IF (IA(I) .EQ. 0) GO TO 30
C     EITHER = S
C     XJ1 = F
C     GO TO 40
C     EITHER = -F
C     XJ1 = S
C     IF (MLE) GO TO 50
C     FC = G ** C
C     GO TO 60
C     OBJECT = OBJECT + DLOG(XJ1)
C     FC = ONE
C     LOOP TO INCREMENT GRADIENT AND HESSIAN UPPER TRIANGLE
C     10 EITHER = FC * EITHER
C     G = FC + G
C     FC = C * EITHER
C     DO 80 J = 1, NACT
C     IND = IACT(J)
C     XJ1 = DBLE(X(IND,1))
C     80 CONTINUE
C     20 CONTINUE
C     30 EITHER = -F
C     XJ1 = S
DDENS = XI J * EITHER
GRAD(J) = GRAD(J) + DDENS
DDENS = XI J * (S - F)
IF (J .EQ. 1) DDENS = DDENS + EXTRA
DDENS = DDENS + FC - G * XI J
DO 70 K = J, NACT
   IND = IACT(K)
   HESS(K, J) = HESS(K, J) + X(IND, I) * DDENS
70 CONTINUE
80 CONTINUE
90 CONTINUE
C SCALE GRADIENT AND HESSIAN BY SAMPLE SIZE
EXTRA = DBLE(FLOAT(N))
DO 100 J = 1, NACT
   GRAD(J) = GRAD(J) / EXTRA
DO 100 I = J, NACT
   HESS(I, J) = HESS(I, J) / EXTRA
   HESS(J, I) = HESS(J, I)
100 CONTINUE
IFault = 0
RETURN
C ERROR EXIT
110 IFault = -1
C RETURN
END
C***************************************************************************
C COMPUTE V(*,*) FACTOR FOR ASYMPOTIC COVARIANCE MATRIX, VLOGISTIC BINARY, CALLED ONLY WHEN C NE 0
C***************************************************************************
SUBROUTINE VLOGISTIC(NACT, IACT, PAR, V, N, IX, X, IA)
C ARGUMENTS
C INTEGER NACT, IACT(NACT), N, IX, IA(N)
C REAL X(IX, N)
C DOUBLE PRECISION C, PAR(NACT), V(NACT, NACT)
C LOCAL SCALARS
C INTEGER IND
C DOUBLE PRECISION FC, S, F, EITHER, TEMP, ZERO, ONE, BIG
C MACHINE-DEPENDENT CONSTANT - BIG ROUGHLY CHOSSEN SO THAT
C DEXP(X) WILL CAUSE EXCEPTION IF /X/ .GT. BIG
C DATA ZERO, ONE, BIG /0 000, 1 000, 174 000/
C C SET UPPER TRIANGLE OF VI(*,*) TO 0
C DD 10 J = 1, NACT
C DD 10 I = J, NACT
10 VI(I, J) = ZERO
C MAIN LOOP OVER SAMPLE
C DD 50 I = 1, N
C FC = ZERO
C DD 20 J = 1, NACT
C IND = IACT(J)
C FC = FC + PAR(J) * X(IND, I)
20 CONTINUE
IF (DABS(FC) .GE. BIG) GO TO 50
C FC = DEXP(FC)
S = ONE + FC
F = FC / S
C***************************************************************************
S = ONE / S
FC = (F*S)**C
EITHER = S
IF (IA(I) EQ 0) EITHER = F
EITHER = (EIGHT + FC)**2
C
C INCREMENT TERM OF V(i,j)
DO 40 J = 1, NACT
   TEMP = EITHER*X(IND,I)
   DO 30 K = 1, NACT
      IND = IACT(K)
      VX = VX + V(K,J)*TEMP*X(IND,I)
   30 CONTINUE
40 CONTINUE
C
C DIVIDE V(i,j) BY SAMPLE SIZE, FILL OUT
TEMP = DBLE(FLOAT(N))
DO 60 J = 1, NACT
   DO 60 I = 1, NACT
      VI = VI / TEMP
   60 CONTINUE
C
RETURN
END
C
C FIRST AND SECOND PARTIAL DERIVATIVES FOR BINARY GAUSSIAN,
C REGRESSION PARAMETERIZATION: PARTIALS ARE SCALED BY
C SAMPLE SIZE
C FAILURE CODE - IFAULT = 1: AN EXCEPTION WAS ABOUT TO OCCUR WHILE PROCESSING THE I TH OBSERVATION
C
SUBROUTINE DGAUBN(C, OBJECT, NACT, IACT, PAR, GRAD, HESS, N, IX, IFAULT)
C
C ARGUMENTS
INTEGER NACT, IACT(NACT), N, IX, IA(N), IFAULT
REAL X(IX,N)
DOUBLE PRECISION C, OBJECT, PAR(NACT), GRAD(NACT), HESS(NACT,NACT)
C
C LOCAL SCALARS
LOGICAL MLE
INTEGER IND
DOUBLE PRECISION EXTRA, DOT, DOTMIN, FC, CBY2, RATIO, EITHER,
       HTERM, XJI, DDEMS, ZE0, ONE, TWO, BIG
C
C FUNCTIONS CALLED
DOUBLE PRECISION ALNORM, RMILLS
C
C MACHINE-DEPENDENT CONSTANT - BIG ROUGHLY CHOSEN SO THAT
C DEXP(X) WILL CAUSE EXCEPTION IF /X/ GT BIG
C DATA ZERO, ONE, TWO, BIG /0 000, 1 000, 2 000, 174 000/
C
IFault = 0
MLE = ONE + C EQ ONE
EXTRA = ONE / PAR(1)
CBY2 = C / TWO
OBJECT = ZERO
DO 10 J = 1, NACT
   GRAD(J) = ZERO
10 CONTINUE
DO 50 I = 1, NACT
   HESS(I,J) = ZERO
50 CONTINUE
10 CONTINUE
C       MAIN LOOP OVER SAMPLE
      DO 90 I = 1, N
        DOT = ZERO
      DO 20 J = 1, NACT
        IND = IACT(J)
        DOT = DOT + PAR(J) * X(IND, I)
      CONTINUE
20 CONTINUE
      DOTMIN = -DOT
      IF (IA(I) .EQ. 0) GO TO 30
      RATIO = RMILLS(DOTMIN)
      EITHER = RATIO
      GO TO 40
30 RATIO = RMILLS(DOT)
      DOT = DOTMIN
      EITHER = -RATIO
      IF (MLE) GO TO 50
      XJI = CBY2 * DOT * DOT
      IF (DABS(XJI) .GT. BIG) GO TO 110
      FC = DEXP(-XJI)
      GO TO 60
50 IF (ONE) GO TO 110
      XJI = ALNORM(DOT, FALSE)
      IF (XJI .EQ. ZERO) GO TO 110
      OBJECT = OBJECT + DLOG(XJI)
C       INCREMENT GRADIENT AND HESSIAN
60 EITHER = FC * EITHER
      HTERM = -FC * RATIO * (DOT + RATIO)
      FC = C * EITHER
      DO 80 J = 1, NACT
        IND = IACT(J)
        XJI = DBLE(X(IND, I))
        GRAD(J) = GRAD(J) + XJI * EITHER
        DDENS = XJI * DOTMIN
        IF (J .EQ. 1) DDENS = DDENS + EXTRA
        HDENS = DDENS * FC + XJI * HTERM
        DO 70 K = J, NACT
          IND = IACT(K)
          HESS(K, J) = HESS(K, J) + X(IND, I) * DDENS
        CONTINUE
70 CONTINUE
80 CONTINUE
90 CONTINUE
C       SCALE GRADIENT AND HESSIAN BY SAMPLE SIZE
      EXTRA = DBLE(FLOAT(N))
      DO 100 J = 1, NACT
        GRAD(J) = GRAD(J) / EXTRA
      CONTINUE
    DO 100 I = J, NACT
      HESS(I, J) = HESS(I, J) / EXTRA
    CONTINUE
100 CONTINUE
      IFAULT = 0
      RETURN
C       ERROR EXIT
C         110 IFault = -1
C         RETURN
C* COMPUTE V(.,.*) FACTOR FOR ASYMPTOTIC COVARIANCE MATRIX.
C* GAUSSIAN BINARY, CALLED ONLY WHEN C NE 0
C*************************************************************************
SUBROUTINE VGAUBB(C, NACT, IACT, PAR, V, N, IX, X, IA)
C*************************************************************************
C* ARGUMENTS
C INTEGER NACT, IACT(NACT), N, IX, IA(N)
REAL X(IX,N)
DOUBLE PRECISION C, PAR(NACT), V(NACT,NACT)
C LOCAL SCALARS
INTEGER IND
DOUBLE PRECISION F2C, TEMP, EITHER, ZERO, BIG
C FUNCTION CALLED
DOUBLE PRECISION RMILLS
C MACHINE-DEPENDENT CONSTANT - BIG ROUGHLY CHOSEN SO THAT
C DEXP(X) WILL CAUSE EXCEPTION IF /X/ GT. BIG
DATA ZERO, BIG /O ODO. 174.000/
C C* SET UPPER TRIANGLE OF V(.,.*) TO ZERO
DO 10 J = 1, NACT
  DO 10 I = J, NACT
  10 V(I,J) = ZERO
C C* MAIN LOOP OVER SAMPLE
DO 70 I = 1, N
  F2C = ZERO
  DO 70 J = 1, NACT
    IND = IACT(J)
    F2C = F2C + PAR(J) * X(IND,1)
  70 CONTINUE
  TEMP = C * F2C * F2C
  IF (DABS(TEMP) GT BIG) GO TO 70
  IF (IA(I) EQ 0) GO TO 30
  EITHER = RMILLS(-F2C)
  GO TO 40
  30 EITHER = RMILLS(F2C)
  40 F2C = DEXP(-TEMP) * EITHER * EITHER
  DO 60 J = 1, NACT
    IND = IACT(J)
    TEMP = F2C * X(IND,1)
  60 CONTINUE
  V(K,J) = V(K,J) + TEMP * X(IND,1)
  50 CONTINUE
  60 CONTINUE
  70 CONTINUE
C C* DIVIDE V(.,.*) BY SAMPLE SIZE, FILL OUT
TEMP = DBLE(FLOAT(N))
DO 80 J = 1, NACT
  DO 80 I = J, NACT
    V(I,J) = V(I,J) / TEMP
  80 CONTINUE
C C RETURN
END
C*************************************************************************
C* FIRST AND SECOND PARTIAL DERIVATIVES FOR BINARY EXTREME
C*************************************************************************
C*************************************************************************
**VALUE, REGRESSION PARAMETERIZATION. PARTIALS ARE SCALED**

**BY SAMPLE SIZE**

**FAILURE CODE - IFAULT = -1 - AN EXCEPTION WOULD HAVE OCCURRED WHEN PROCESSING THE I TH OBSERVATION**

**SUBROUTINE DEXVBN(C, OBJECT, NACT, IACT, PAR, GRAD, Hess, N, IX, IFAULT)**

**1**

**ARGUMENTS**

**INTEGER NACT, IACT(NACT), N, IX, IA(N), IFAULT**

**REAL X(IX,N)**

**DOUBLE PRECISION C, OBJECT, PAR(NACT), GRAD(NACT), Hess(NACT,NACT)**

**LOCAL SCALARS**

**LOGICAL MLE**

**INTEGER IND**

**DOUBLE PRECISION EXTRA, DOT, EXPON, DENFAC, EXPEXP, FC, EITHER, HTERM, ZERO, ONE, BIG, BIG1**

**IF ((/X/ GT BIG1) .NE. 1.00D0) IF X .GT. BIG1**

**MACHINE-DEPENDENT CONSTANTS - BIG ROUGHLY CHOSEN SO THAT DEXP(X) WILL CAUSE EXCEPTION IF /X/ GT. BIG, BIG1 SUCH**

**DATA ZERO, ONE, BIG, BIG1 /O 0 0 1.00D0, 174.00D0, -36 200/**

**IFault = 0**

**MLE = ONE + C EQ ONE**

**EXTRA = ONE / PAR(1)**

**OBJECT = ZERO**

**DO 10 J = 1, NACT**

**GRAD(J) = ZERO**

**DO 10 I = J, NACT**

**Hess(I,J) = ZERO**

**10 CONTINUE**

**MAIN LOOP OVER SAMPLE**

**DO 90 I = 1, N**

**DOT = ZERO**

**DO 20 J = 1, NACT**

**IND = IACT(J)**

**DOT = DOT + PAR(J) * X(IND,1)**

**20 CONTINUE**

**IF (DABS(DOT) GE BIG) GO TO 110**

**DEXP = DEXP(DOT)**

**DENFAC = ONE - EXPON**

**IF (IA(1) EQ. 0) GO TO 40**

**IF (EXON .GT. BIG OR (DOT .GE. BIG1) .GO TO 110**

**EXPES = DEXP(EXPSON)**

**DOT = ONE - ONE / EXPS**

**FC = EXPON / EXPS**

**EITHER = FC / DOT**

**HTERM = EITHER * (DENFAC - EITHER)**

**IF (MLE) GO TO 30**

**FC = FC ** C**

**GO TO 60**

**30 FC = ONE**

**OBJECT = OBJECT + DLOG(DOT)**

**GO TO 60**

**NON FAILURE TERM, IA(1) .EQ 0**

**EITHER = EXPON**

**HTERM = EITHER**

**40 CONTINUE**
IF (MLE) GO TO 50
FC = C + (DOT - EXPON)
IF (DABS(FC) GE BIG) GO TO 110
FC = DEXP(FC)
GO TO 60
50 FC = ONE
OBJ = OBJ + EITHER
C
C LOOP TO INCREMENT THE DERIVATIVES
60 EITHER = FC * EITHER
HTERM = FC * HTERM
FC = C + EITHER
DO 80 J = 1, NACT
IND = IACT(J)
DOT = DBLE(X(IND,1))
GRAD(J) = GRAD(J) / DOT + EITHER
EXPON = DOT + DNPAC
IF (J EQ 1) EXPON = EXPON + EXTRA
EXPON = FC * EXPON + DOT + HTERM
DO 70 K = J, NACT
IND = IACT(K)
HESS(K,J) = HESS(K,J) + X(IND,1) * EXPON
70 CONTINUE
80 CONTINUE
90 CONTINUE
C
SCALE PARTIALS BY SAMPLE SIZE, NORMAL EXIT
EXTRA = DBLE(FLOAT(N))
DO 100 J = 1, NACT
GRAD(J) = GRAD(J) / EXTRA
DO 100 I = J + 1, NACT
HESS(I,J) = HESS(I,J) / EXTRA
HESS(J,J) = HESS(J,J)
100 CONTINUE
RETURN
C
ERROR EXIT - EXCEPTIONS
110 IFault = 1
C
RETURN
END
C
******************************************************************************
C COMPUTE V( *, *) FACTOR FOR ASYMPTOTIC COVARIANCE MATRIX.
C BINARY EXTREME VALUE CALLED ONLY WHEN C = 0
C
C SUBROUTINE VEXVBNC(NACT, IACT, PAR, V, N, IX, X, IA)
C ARGUMENTS
C INTEGER NACT, IACT(NACT), N, IX, IA(N)
C REAL X(IX,N)
C DOUBLE PRECISION C, PAR(NACT), V(NACT, NACT)
C
C LOCAL SCALARS
C INTEGER IND
C DOUBLE PRECISION DOT, EXPON, FC, ZERO, ONE, BIG, BIG1
C MACHINDEPENDENT CONSTANTS BIG ROUGHLY CHOSEN SO THAT
C DEXP(X) WILL CAUSE EXCEPTION IF /X/ GT. BIG, BIG1 SUCH
C THAT 1.0D0 / DEXP(DEXP(X)) NE 1.0D0 IF X GT BIG1
C DATA ZERO, ONE, BIG, BIG1 /O/ 0.0D0, 1.0D0, 174.0D0, -36 200/
C SET UPPER TRIANGLE OF VI( *, *) TO ZERO
C
C DO 10 J = 1, NACT
C
DO 10 I = J, NACT
10 V(I,J) = ZERO
C
C MAIN LOOP OVER SAMPLE
DO 70 I = 1, N
DOT = ZERO
DO 20 J = 1, NACT
IND = IACT(J)
DOT = DOT + PAR(J) * X(IND,1)
20 CONTINUE
IF (DABS(DOT) GE BIG) GO TO 70
EXPON = DEXP(DOT)
IF (I(A(I)) EQ 0) GO TO 30
IF (DOT LT BIG) GO TO 70
FC = DEXP(EXPON)
DOT = ONE - ONE / FC
FC = EXPON / FC
EXPON = FC / DOT
FC = EXPON - FC ** C
GO TO 40
30 FC = C * (DOT - EXPON)
IF (DABS(FC) GT BIG) GO TO 70
FC = EXPON - DEXP(FC)
C
C SUMMERS FOR V(*,*)
40 FC = FC + FC
DO 60 J = 1, NACT
IND = IACT(J)
DOT = FC * X(IND,1)
60 DO 50 K = J, NACT
IND = IACT(K)
V(K,J) = V(K,J) + DOT * X(IND,1)
50 CONTINUE
60 CONTINUE
70 CONTINUE
C
C DIVIDE V(*,*) BY SAMPLE SIZE. FILL OUT
DOT = DBLE(FLOAT(N))
DO 80 J = 1, NACT
DO 80 I = J, NACT
V(I,J) = V(I,J) / DOT
80 CONTINUE!
C
RETURN
END
C
C DOUBLE PRECISION MATRIX MULTIPLICATION
C X(N1 BY N3) * Y(N1 BY N2) = Z(N2 BY N3)
C THREE OPTIONS -
C IFLAG = 0 - X, Y, AND Z ARE DISTINCT
C IFLAG LT 0 - X, Y OVERLAP, CORNER OF Y OVERWRITTEN
C IFLAG GT 0 - X, Z OVERLAP, CORNER OF Z OVERWRITTEN
C
C SUBROUTINE DMXMLT(X, IX, N1, Y, IY, N2, Z, IZ, N3, WORK, LWORK, IFLAG, IFault)
C
C ARGUMENTS
C INTEGER IX, N1, IY, N2, IZ, N3, LWORK, IFLAG, IFault
C DOUBLE PRECISION X(IX,N3), Y(IY,N2), Z(IZ,N3), WORK(LWORK)
C LOCAL SCALARS
C
C DMXMO001 DMXMO002 DMXMO003 DMXMO004 DMXMO005 DMXMO006 DMXMO007 DMXMO008 DMXMO009 DMXMO010 DMXMO011 DMXMO012 DMXMO013 DMXMO014
DOUBLE PRECISION TEMP, ZERO
DATA ZERO /0 000/
C
C ERROR EXITS
IF AULT = 1
IF (MINO(N1,N2,N3) LT 1 OR MINO(I1,X,IV) LT N1 OR IZ LT N)
1 N2) RETURN
IF AULT = 2
IF (IFLAG LT 0 AND (N3 GT N2 OR LWORK LT 1))
1 RETURN
IF AULT = 3
IF (IFLAG LT 0 AND (N1 GT IZ OR LWORK LT N1))
1 RETURN
IF AULT = 0
IF (IFLAG NE 0) GO TO 30
C
C STRAIGHTFORWARD, NO OVERWRITING
DO 20 I = 1, N1
DO 20 J = 1, N3
TEMP = ZERO
DO 10 K = 1, N2
10 TEMP = TEMP + Y(I,K) * Z(K,J)
X(I,J) = TEMP
20 CONTINUE
RETURN
C
C CORNER OF MATRIX Z IS OVERWRITTEN
DO 30 I = 1, N1
DO 30 J = 1, N3
TEMP = ZERO
DO 20 K = 1, N2
20 TEMP = TEMP + Y(I,K) * Z(K,J)
WORK(I) = TEMP
30 CONTINUE
RETURN
C
C CORNER OF MATRIX Y IS OVERWRITTEN
DO 80 I = 1, N1
DO 80 J = 1, N3
TEMP = ZERO
DO 70 K = 1, N2
70 TEMP = TEMP + Y(I,K) * Z(K,J)
WORK(J) = TEMP
80 CONTINUE
RETURN
C
RETURN
END
C
C DOUBLE PRECISION VERSION OF ALGORITHM AS 66. APPLIED
C STATISTICS (1973), VOL. 22, NO. 3
C EVALUATES THE TAIL AREA OF THE STANDARD NORMAL CURVE
C FROM X TO INFINITY IF UPPER IS TRUE OR FROM
C MINUS INFINITY TO X IF UPPER IS FALSE.
C*******
DOUBLE PRECISION FUNCTION ALNORM(X,UPPER)
C
ARGUMENTS
LOGICAL UPPER
DOUBLE PRECISION X
C
LOCAL SCALARS
LOGICAL UP
DOUBLE PRECISION LTONE, UTZERO, ZERO, HALF, ONE, CON, Z, Y
C
MACHINE-DEPENDENT CONSTANTS - LTONE = (N + 9) / 3, WHERE
C
N IS NO. OF DECIMAL DIGITS IN DOUBLE PRECISION NUMBER,
C
UTZERO SUCH THAT DEXP(-X*X/2.000) WILL CAUSE AN EXCEPTION
C
IF X GT UTZERO
C
-CONSTANTS IN EXPRESSIONS ARE AS IN AS 66.
DATA LTONE, UTZERO, ZERO, HALF, ONE, CON /8.000, 18.65DO, 0.000, 1.0DO, 1.0DO, 1.28DO/
C
UP * UPPER
Z = X
IF (Z GE ZERO) GO TO 10
UP * NOT UP
Z = -Z
10 IF (Z LE LTONE OR UP AND Z .LE. UTZERO) GO TO 20
ALNORM = ZERO
GO TO 40
20 Y = HALF + Z
IF (Z .GT. CON) GO TO 30
ALNORM = HALF - Z
C
ALNORM = Z
C
ALNORM = 0.39998752438504D0 - DEXP(-Y) / (Z - Z + 3.8052D-8 + 1.0600615302DD0 / (Z - 3.805D-8) - 1.9861591364D0 / (Z - 4.8365912803D0 - 15.3150974510D0 / (Z + 4.74238092427D0 + 3.078833034D0) / (Z + 3.49001941701DO)))))
GO TO 40
30 ALNORM = 0.39998752438504D0 - DEXP(-Y) / (Z - Z + 3.8052D-8 + 1.0600615302DD0 / (Z - 3.805D-8) - 1.9861591364D0 / (Z - 4.8365912803D0 - 15.3150974510D0 / (Z + 4.74238092427D0 + 3.078833034D0) / (Z + 3.49001941701DO)))))
GO TO 40
40 IF ( .NOT. UP) ALNORM = ONE - ALNORM
C
RETURN
END
C*******
RECIPROCAL OF MILLS RATIO, Z(X) / Q(X), X A STANDARD NORMAL
C
VARIATE, BASED ON FUNCTION ALNORM(), ALGORITHM AS 66.
C
DOUBLE PRECISION FUNCTION RMILLS(X)
C
ARGUMENTS
DOUBLE PRECISION X
C
LOCAL SCALARS
LOGICAL UP
DOUBLE PRECISION Z, Y, LTONE, UTZERO, ZERO, HALF, ONE, CON, FP1, FP2
C
MACHINE-DEPENDENT CONSTANTS - LTONE = (N + 9) / 3, WHERE
C
N IS NO. OF DECIMAL DIGITS IN DOUBLE PRECISION NUMBER,
C
UTZERO SUCH THAT DEXP(-X*X/2.000) WILL CAUSE AN EXCEPTION
C
IF X LT UTZERO
C
-CONSTANTS IN EXPRESSIONS ARE AS IN AS 66.
DATA LTONE, UTZERO, ZERO, HALF, ONE, CON, FP1, FP2 /8.000,
C
RMIL0001
RMIL0002
RMIL0003
RMIL0004
RMIL0005
RMIL0006
RMIL0007
RMIL0008
RMIL0009
RMIL0010
RMIL0011
RMIL0012
RMIL0013
RMIL0014
RMIL0015
RMIL0016
RMIL0017
RMIL0018
RMIL0019
RMIL0020
RMIL0021
RMIL0022
RMIL0023
RMIL0024
RMIL0025
RMIL0026
RMIL0027
RMIL0028
RMIL0029
RMIL0030
RMIL0031
RMIL0032
RMIL0033
RMIL0034
RMIL0035
RMIL0036
RMIL0037
RMIL0038
RMIL0039
RMIL0040
RMIL0041
RMIL0042
RMIL0043
RMIL0044
RMIL0045
RMIL0046
RMIL0047
RMIL0048
RMIL0049
RMIL0050
RMIL0051
RMIL0052
RMIL0053
RMIL0054
RMIL0055
RMIL0056
RMIL0057
RMIL0058
RMIL0059
RMIL0060
RMIL0061
RMIL0062
RMIL0063
RMIL0064
RMIL0065
RMIL0066
RMIL0067
RMIL0068
RMIL0069
RMIL0070
RMIL0071
RMIL0072
RMIL0073
RMIL0074
RMIL0075
RMIL0076
RMIL0077
RMIL0078
RMIL0079
RMIL0080
RMIL0081
RMIL0082
RMIL0083
RMIL0084
RMIL0085
RMIL0086
RMIL0087
RMIL0088
RMIL0089
RMIL0090
RMIL0091
RMIL0092
RMIL0093
RMIL0094
RMIL0095
RMIL0096
RMIL0097
RMIL0098
RMIL0099
RMIL0100
RMIL0101
RMIL0102
RMIL0103
RMIL0104
RMIL0105
RMIL0106
RMIL0107
RMIL0108
RMIL0109
RMIL0110
RMIL0111
RMIL0112
RMIL0113
RMIL0114
RMIL0115
RMIL0116
RMIL0117
RMIL0118
1. \texttt{z = 186500, 0 000, 0 500, 1 000, 1 2800, 7 988456082865360-1, RMIL0019}
2. \texttt{z = 3 98942280401432680-1, RMIL0020}

\begin{verbatim}
C TRIVIAL CASES
IF (X NE ZERO) GO TO 10
RMILLS + FP11
RETURN
10 IF (X GT ZERO) GO TO 20
RMILLS + ZERO
RETURN

C USUAL SITUATION
20 UP = TRUE,
Z * X
IF (Z GE ZERO) GO TO 30
UP = FALSE.
Z * -Z
30 Y = HALF * Z * Z
IF (Z GT CON) GO TO 50

C CENTRAL PORTION - Q LT ABS(X) LT 1.28
RMILLS + Z * (0 3989428044400-0 39990343850400+Y/(Y + 5
1/58854805800-29 21355780800/(Y + 2 6243312167900+48
2/49993069200/(Y + 5 9288572443800)))
RMILLS = Y / (HALF + RMILLS)
RETURN
40 RMILLS = Y / (HALF - RMILLS)
RETURN

C OUTER PORTION - ABS(X) GE 1.28
50 IF (UP OR Z LE LONE) GO TO 60

C SPECIAL CASE - LOWER TAIL AND Q ESSENTIALLY 1
RMILLS + FP12 + DEXP(-Y)
RETURN

C USUAL SITUATION
60 RMILLS - (Z - 3 80520 8 + 1 0000061530200/(Z + 3 980647940-4 + 1)
19861538136400/(Z - 0 1516791663500+5 29330242600/(Z + 4
2938591280800 15 150897245100/(Z + 0 74238092402700+30 78999303400
3(Z + 3 990194701100)))
RMILLS = Y / (ONE - Y/RMILLS)
RETURN
END
\end{verbatim}
C**********************************************************************
C SAMPLE MAIN PROGRAM AND INPUT SUBROUTINE FOR SELF-CRITICAL
C
C BINARY ESTIMATION. IT IS RECOMMENDED THAT BINARY() AND
C ITS AUXILIARY PROCEDURES BE COMPILLED AND PLACED IN AN
C OBJECT CODE LIBRARY, SO THE USER CAN CALL BINARY() FROM
C ARBITRARY PROGRAMS.
C
C**********************************************************************

INTEGER N, IX, IA(750), NPAR, ISUB(10), ISTART(10), IDEP, IDIST
1 MAXIT, IPRINT, IFLAG, ICOV, LMEM, MEMORY(490), IFAULT, NC
2 NMAX, NCMAX, IREAD
3 REAL X(10, 750)
4 DOUBLE PRECISION C(5), RELTOL, ABSTOL, BETA(10), XLOGL, COV(10, 10)

C DIMENSION SPECIFICATIONS FOR ARRAYS.
C THIS MAIN PROGRAM CAN HANDLE UP TO 750 OBSERVATIONS.
C WITH UP TO 10 PARAMETERS AND 5 DIFFERENT VALUES OF
C FOR ESTIMATION
C DATA IX, ICOV, LMEM, NMAX, NCXMAX /10, 10, 490, 750, 5/
C BASIC INFORMATION TO PASS TO BINARY() - IPRINT STANDS
C FOR LOGICAL OUTPUT UNIT 6
C DATA MAXIT, RELTOL, ABSTOL, CPRINT, IFLAG /15, 1.00-7, 1.00-7, 6/
1 0/
2 IREAD IS LOGICAL INPUT UNIT 5
3 DATA IREAD /5/
C SPECIFY EVERYTHING BY HAND IN SUBROUTINE INPUT - ARRAY
C MEMORY(*) PASSED AS WORKSPACE

C CALL INPUT(IX, N, NMAX, X, IA, NPAR, BETA, ISUB, ISTART, NC,
1 0 NCXMAX, C, IDEP, IDIST, MEMORY, IREAD, CPRINT, IFAULT)
3 IF (IFault EQ 0) GO TO 10
4 IF (IFault EQ 1) WRITE (CPRINT, 50) N, NMAX
5 IF (IFault EQ 2) WRITE (CPRINT, 60) NPAR, IX
6 IF (IFault EQ 3) WRITE (CPRINT, 70) NC, NCXMAX
7 WRITE (CPRINT, 80)
8 STOP
C LOOP OVER THE VALUES OF C REQUESTED FOR ESTIMATION
10 DO 40 I = 1, NC
2 CALL BINARY(IN, IX, X, IA, NPAR, ISUB, ISTART, IDEP, IDIST, C(I))
3 RELTOL, ABSTOL, MAXIT, CPRINT, IFLAG, BETA, XLOGL, COV
4 COV, LMEM, MEMORY, IFault
5 IF (IFault EQ 0) GO TO 20
6 WRITE (CPRINT, 90) IFault
7 STOP
8 20 IF (I .GT. 1) GO TO 40
9 C AFTER THE FIRST ESTIMATION, SET ISTART(*) TO 1, SO
10 LATEST ESTIMATES CAN BE USED AS STARTING VALUES
11 DO 30 J = 1, NPAR
12 ISTART(J) = 1
13 30 DO 30 J = 1, NPAR
C 40 CONTINUE
50 FORMAT ('ERROR - SAMPLE SIZE OF', 110, ' EXCEEDS DIMENSION OF', 10)
60 FORMAT ('ERROR - ', 13, ' PARAMETERS EXCEEDS DIMENSION OF', 13)
70 FORMAT ('ERROR - ', 13, ' C VALUES EXCEEDS DIMENSION OF', 13)
80 FORMAT ('ORECOMPILE MAIN PROGRAM WITH NEW DIMENSIONS')

C**********************************************************************
C*----------------------------------------------------------------------
C CRUDE INPUT ROUTINE  SETS ESTIMATION CONTROL PARAMETERS
C EXPLICITLY, INSTEAD OF READING THEM IN
C----------------------------------------------------------------------
SUBROUTINE INPUT(X, NMAX, X, IA, NPAR, BETA, ISUB, ISTART, NC, NCMAX, C, IDEP, IDIST, WORK, IREAD, IPRINT, IFAULT)

C* ARGUMENTS
C
INTEGER IX, N, NMAX, IA(1), NPAR, ISUB(1), ISTART(1), NC, NCMAX,
1 IDEP, IDIST, IPRINT, IFAULT
REAL (IX,1), WORK(1)
DOUBLE PRECISION BETA(1), C(1)

C* LOCAL TYPE DECLARATION
C
LOGICAL FLAG

C SET DETAILS OF PROBLEM SIZE
N = 670
NPAR = 5
NC = 4

C CHECK FOR SIZE ERRORS
IFault = 1
IF (N GT NMAX) RETURN
IFault = 2
IF (NPAR GT IX) RETURN
IFault = 3
IF (NC GT NCMAX) RETURN
IFault = 0

C MODELING DETAILS
C DEPENDENT VARIABLE IN FIRST ROW
IDEP = 1

C CONSTANT IN SECOND ROW
ICONS = 2

C GAUSSIAN TOLERANCE DISTRIBUTION
IDIST = 2

C VALUES OF C FOR SELF-CRITICAL
C(1) = 0 000
C(2) = 0 100
C(3) = 0 200
C(4) = 0 300

C INITIALIZE AVERAGES TO 0
YBAR* = 0 0
YSE = 0 0
DO 10 I = 1, NPAR
10 WORK(1) = 0 0

C LOOP OVER SAMPLE TO READ IN DATA
FLAG = NPAR LE 2
DO 40 I = 1, N
READ (IREAD,20) VC, IA(I), ALPHA, S
20 FORMAT (F10 0, 12, 2F8 0)

C EXPRESS S IN METERS, SCALE VC
VC = ALOG(VC/1000 0)
S = 0 3048 * EXP(S)

C   FILL THE DATA MATRIX - CONSTANT IN SECOND ROW
  X(IDEP,1) = VC
  X(ICONST,1) = 1 0
  X(3,1) = ALPHA
  X(4,1) = S
  X(5,1) = ALPHA * S
C   UPDATE MEAN AND STD ERROR OF DEPENDENT VARIABLE IN A
C          WAY WHICH DOESN'T LOSE SIGNIFICANT FIGURES
  TEMP = VC - YBAR
  YBAR = YBAR + TEMP / FLOAT(I)
  YSE = YSE + TEMP * (VC - YBAR)
C   INCREMENT COVARIATE AVERAGES (3D ROW AND UP)
   IF (FLAG) GO TO 40
   DO 30 J = 3, NPAR
      WORK(J) = WORK(J) + X(J,1)
30   WORK(J) = WORK(J) + X(J,1)
C   40 CONTINUE
C   SET ISUB(*) AND ISTART(*), PROVIDING STARTING VALUES
C          FOR INTERCEPT AND SCALE
   DO 50 I = 1, NPAR
      ISUB(I) = I
      ISTART(I) = 0
50  CONTINUE
  TEMP = FLOAT(N)
  YSE = SORT(YSE/TEMP)
  ISTART(IDEP) = 1
  ISTART(ICONST) = 1
  BETA(IDEP) = DBLE(YSE)
  BETA(ICONST) = DBLE(YBAR)
C   WRITE OUT INFORMATION ON STD. ERROR AND AVERAGES
   WRITE (IPRINT,60) YBAR, YSE
60  FORMAT ('IDEPENDENT (STRESS) VARIABLE'/'OMEAN = ', E14.6,
   'STANDARD DEVIATION = ', E14.6)
   IF (FLAG) RETURN
   WRITE (IPRINT,70)
70  FORMAT ('OCOVARIATES HAVE BEEN CENTERED BY THEIR MEANS'/
   'VARIABLE' , 10X, 'MEAN')
   DO 90 I = 3, NPAR
      WORK(I) = WORK(I) / TEMP
      WRITE (IPRINT,80) ISUB(I), WORK(I)
80  FORMAT ('O', 1B, E14.6)
90  CONTINUE
C   SUBTRACT MEANS FROM COVARIATES
   DO 110 I = 1, N
      DO 100 J = 3, NPAR
         X(J,1) = X(J,1) - WORK(J)
100  CONTINUE
110  CONTINUE
C   RETURN
END
An Algorithm for the Computation of Generalized Likelihood or Self-Critical Estimators for Binary Data

T.A. Delehanty
A.S. Paulson

Rensselaer Polytechnic Institute
Troy, New York 12180

Approved for public release; distribution unlimited.

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

binary data, extreme value distribution, logistic distribution, Gaussian distribution, generalized likelihood, model-critical analysis, regression models, parametric proportional hazard models

This paper describes the computational algorithms for computing generalized likelihood estimators for parametric proportional hazards models.
END
5-87
DTIC