Using the IML Procedure for Iteratively Reweighted Multiple Linear Regression with a Block-diagonal Covariance Matrix

Donald H. MacQueen, Lawrence Livermore National Laboratory

Abstract

We have been using SAS/IML® software to model the magnitude of a seismic event using signals from one to four seismic stations. Geophysical properties of the earth through which the shock waves travel introduce a correlation between the recorded signals for each event. Thus up to four correlated dependent variables are recorded for each value of the independent variable. We estimate the dependence of the measured seismic signals on the source using weighted linear regression with a block-diagonal covariance matrix in which the blocks are not all the same dimension and the covariances need to be estimated. PROC IML is a useful and succinct programming tool for this problem.

The basic definition of the magnitude of an earthquake is \( m = \log(A) + \lambda_0 \) where \( A \) is the amplitude of the signal recorded by the seismograph and \( \lambda_0 \) is a correction for the distance between the earthquake and the seismograph. This definition has also been applied to man-made seismic events (for example, underground nuclear explosions, jet aircraft crashes, high explosive blasts used in tunnel drilling) for the purpose of relating the amount of energy (\( w \)) released by the event. A simple linear model is often used:

\[
M = \mathbf{X} B + \mathbf{E}
\]

where \( M \) is now a \( 1 \times n_j \) vector of individual station magnitudes, \( \mathbf{X} \) is the design matrix, \( \mathbf{B} \) is a \( p \times 1 \) vector of parameters, \( \mathbf{E} \) is a \( 1 \times n_j \) vector of errors from the model, and \( \mathbf{E} \) is a \( 1 \times n_j \) block diagonal matrix of station covariances. \( \Sigma \) is block diagonal because errors between events are uncorrelated while errors within one event are correlated. For our particular problem the maximum value of \( n_j \) is 4, but in general this need not be so. Table 1 illustrates this model by writing out part of the data for a design with four intercepts and one slope.

Note that in Table 1 the first two events have all four stations reporting, so they have 4x4 blocks in the covariance matrix. In contrast, events 3 and 5 have three stations each, but a different three stations, so they each have a different 3x3 submatrix of the full 4x4 matrix. Finally, event 4 has only one station with data so it has a 1x1 matrix. All events with 4 stations have the same 4x4 block. There are \( 15 \times 15 \) possible submatrices, any of which may or may not appear in a given set of data.

The normal equations for this model can be written in one of two
equivalent ways:

\[ X' \Sigma^{-1} X B = X' \Sigma^{-1} M \]

where \( M, X, B, \) and \( \Sigma (\Sigma_i \times \Sigma_i) \) are from the full model as defined above, or

\[
\left( \sum_i X_i \Sigma_i^{-1} X_i \right) B = \sum_i X_i \Sigma_i^{-1} M_i
\]

where the matrices are defined as follows:

- \( X_i \) vector of station magnitudes for the \( i \)th event,
- \( X_i \) design matrix for the \( i \)th event,
- \( E_i \) vector of station errors for the \( i \)th event, and
- \( B \) parameter vector.

If the covariances are known, the normal equations can easily be solved using PROCC IML. However, the covariances are not known for our data, so we have devised an iterative procedure to estimate the covariances along with the parameters. The PROCC IML source code is given below, along with code to generate an example dataset.

After reading the data from a dataset into matrices initial parameters are estimated using \( \beta = 1 \). Then \( \beta \) is estimated from the residuals and the model is refit using the new \( \Sigma \). These steps are continued until the changes in the estimates of \( B \) and \( \Sigma \) are less than a user specified value.

This method also produces a weighted, opposed to simple, average magnitude estimate, with the weights depending on which subset of the four stations recorded the event. If we now define \( \Sigma \) to be the 4x4 matrix of estimated covariances for events recording all four stations then the equation for the vector of weights for a particular combination of stations is

\[
W_c = \left( \Sigma_c^{-1} J \right)
\]

where \( J \) is a vector of 1's and \( \Sigma_c \) is the submatrix of \( \Sigma \) corresponding to the combination of stations.

We are also interested in the reverse estimation problem. In the case of simple linear regression \( m=\alpha+b \times (x=\log(w)) \) seismologists often summarize the results of the reverse estimation with the value of 1.96 times the residual standard deviation divided by the slope, converted to a multiplicative factor \( F \). For our model the equations for this are:

\[
\Sigma = \Sigma^{-1} \Sigma^{-1} \\
W_c = \left( \Sigma_c^{-1} J \right)
\]

\[
V_c = \left( J \Sigma_c^{-1} J \right)^{-1}
\]

\[
F_c = 10^{(1.96/\sqrt{n})}
\]

where the subscript \( c \) indicates a sub-matrix corresponding to a subset of stations recording an event, and \( S \) is a diagonal matrix containing the slopes.

In evaluating the results of this model, the correlation matrix, the weights, and \( F_c \) are of particular interest.

Source code

The source code consists of two files. The first, called tst.sas, creates a sample dataset and then fits a model using the iterative method. Examining the structure of this dataset should help the reader to prepare his or her own data. The second file, called icv.sas, contains the source code for the iterative method.

```
/*-------------------------------*/
tst.sas
Create example dataset TST for PROC IML module ICV

Run ICV to estimate the parameters

Data with 5 dependent variables

True model with 5 intercepts and a common slope

\[
\begin{align*}
y_1 &= 1 + 3x \\
y_2 &= 2 + 3x \\
y_3 &= 3 + 3x \\
y_4 &= 4 + 3x \\
y_5 &= 5 + 3x
\end{align*}
\]

To fit a model with 5 slopes, change the values of \( b_1-b_5 \), and change the value of desvars to \( il-15 \) \( sl-s5 \)

```
data tst;
length y x il-iS sl-s5 8 subc $ 5;
array iep[S] iI-iS,
array slp[S] sl-s5,
array a[S] aI-aS,
array b[S] b1-b5,
array usej[S] j1-j5;
do i=1 to 5,
a[i]=i,
b[i]=3,
ep[i]=i;
end;

*10 events with a some or all of the dependent variables;
do i=1 to 10;
  nev+i;
x=10*ranuni(0);
  if ranuni(0)>0.25 then do;
    usej[i]=0;
    usej[i]=1;
  end;

* Create example dataset TST for PROC IML module ICV

```
1515
*Several events with all 5 dependent variables:
do i=1 to 20;
  nev+i;
  sub=12345; subc=left(sub);
  x=10*ranuni(0);
do j=1 to 5;
do k=1 to 5;
  if j=k then icp[k]=1;
  else icp[k]=0;
  if j=k then slp[k]=x;
  else slp[k]=0;
  end;
  y=a[j]+b[j]*x+.15*ranuni(0);
  output;
  end;
  end;
  end;

drop i k al-a5 b1-b5;
run;

proc sort data=tst;
  by nev j;
run;
proc print data=tst;
  var subc y x i1-i5 sl-s5i;
run;
proc iml;
%inc icv;

dset={tst};
depvar={y};
desvars={i1 i2 i3 i4 i5 s1 s2 s3 s4 s5};
bveci={6 6 6 6 6};
subvar={subc};
maxiter=25;
maxrel=.0001;
run icv;

/* ----------------------------------
   End of tst.sas
   ----------------------------------*/

/* ----------------------------------
   icv.sas
   IVC consists of 6 modules
   prep preparations
   fitmod estimates parameters
   newcov estimates covariance
   loglike calculates log-likelihood
   wts calculates weights
   icv controls the entire process
   Before running icv, 6 iml matrices must have values
   dset a SAS dataset, sorted by dependent variable index
   depvar the dependent variable in tst
desvars the design variables in tst
   subvar character variable in tst identifying the combination of dependent variables of the current case
   bveci a vector of indicies indicating which elements of the parameter vector are to be considered slopes for the reverse regression. The number of elements in bveci is the same as the maximum covariance block size.
   maxiter the maximum number of iterations
   maxrel the convergence criteria. Convergence is declared when relative change is less than maxrel

For example,
  dset=tst;
depvar=y;
desvars={i1 i2 i3 i4 i5 s1 s2 s3 s4 s5};
bveci={6 6 6 6 6};
subvar={subc};
maxiter=25;
maxrel=.0001;

/* ----------------------------------
   module prep: preparations
   ----------------------------------*/
start prep;
call execute('use ',dset,';');
read all var depvar into fully;
read all var desvars into fullx;
read all var subvar into fullsub;
call execute('close ',dset,';');

npts=row{fully};
subs=unique(fullsub);
nsub=nrow(subs);
npar=ncol(desvars);
ncrnb=repeat(.,nsub,1);
do j=1 to ncrnb;
  atmp[ ,j]=substr(subs,j,l);
  end;
subm=um(atmp);
ncsub=length(subs);

/* Find number with each combination;
do i=1 to nsub;
  isub=loc(fullsub=subs[i1]);
  cmb=subs[i1];
  ncbi=nrow(fully[i1])/length(cmb);
  end;
ntot=ncbi;

subm=substr(atmp);
csub=length(subs);

*twopi=2*pi;
wts=twopi/exp(twopi); lli=repeat(.,nsub,1);
do j=1 to nc;
  atmp[j]=substr(subs,j,1);
  end;

subm=substr(atmp);
csub=length(subs);

* Find number with each combination;
do i=1 to nsub;
  isub=loc(fullsub=subs[i1]);
  cmb=subs[i1];
  ncbi=nrow(fully[i1])/length(cmb);
  end;
ntot=ncbi;

cov=inv(nc);
oldcov=cov;
bft=repeat(1,1,npar);
twopi=2*pi; ili=repeat(.,nsub,1);
module fitmod: fit the model
-------------------------------------*/
start fitmod;
oldcov=cov;
oldbf=bft;
xpx=repeat(0,npar,npar);
xy=repeat(0,npar,1);  ypy=0;
do i=1 to nsub;
    isub=loc(fullsub=subs[i]);
    cmb=subs[i];
    icmb=subm[i,jcmb];
    covsub=cov[icmb,icmb];
    invcov=inv(covsub);  
    y=fully[isub];
x=xfullx[isub,];
do iev=1 to nmb[i];
    evt=(iev-1)*ncsub[i]+iev*ncsub[i];
    xpx=xpx+x[evt,]*invcov*x[evt,];  
    xpy=xpy+x[evt,j']*invcov*y[evt];
    ypy=ypy+ y[evt]*invcov*y[evt];
end;
end;
fullrank=0;
hrmxpx=hermite(xpx);
if hrmxpx=1(nrow(xpx)) then do;
    fullrank=1;
    b=;solve(xpx,xpy);
b=bf;
    relbf=abs((b-bf)/bf);  
    maxrelbf=max(relbf);
end;
finish;

module newcov: estimate the covariance matrix
-----------------------------------------------------*/
start newcov;
yhat=xfullx*bf;
rsd=yhat-fully;
    ic=loc(length(subs)=nc);
isub=loc(fullsub=subs[ic]);
r=rss(isub,;
    n4=nrow(r)/nc;
    rs=shape(r,n4,nc);
    cov=rs'*rs/n4;
    dgnl=inv(diag(cov)###.5);
corr=dgnl*cov*dgnl;
    relcov=abs((cov-oldcov)/cov);
    maxrelcov=max(relcov);
finish;

module wts: calculate the weights, residual sigma, reverse regression sigma and F
----------------------------------------------------------------------------------------*/
start wts;
    rowsum=repeat(0,nsub,1);
    wsig=repeat(.,nsub,1);
    ysig=repeat(.,nsub,1);
    wts=repeat(.,nsub,nc);
bvec=repeat(.,nc,1);
do i=1 to ncol(bvec);
bvec[i]=bft[bvec[i]];
end;
wcov=inv(diag(bvec))*cov*inv(diag(bvec));
do i=1 to nsub;
    cmb=subs[i];
    icmb=subm[i,jcmb];
    covsub=cov[icmb,icmb];
    invcov=inv(covsub);
    jvec=j(ncsub[i,1,1,1]);
    yvar=1/(jvec'*invcov*jvec);
    ysig[i]=sqrt(yvar);
    wvar=1/(jvec'*invcovw*jvec);
    wsig[i]=sqrt(wvar);
    wts[i,icmb]=jvec'*invcovw*wvar;
    rowsum[i]=sum(wts[i,]);
end;
f=(10)**(1.96*wsig);
finish;

module loglike: log-likelihood assuming multi-variate normal distribution
----------------------------------------------------------------------------------------*/
start loglike;
    /*------------------------------------
    Use oldcov, which is the covariance matrix
    which was used to fit the model. This lets
    the loglikelihood after the first estimate
    of the parameters correspond to a non-
    weighted fit.
    --------------------------------------*/
do i=1 to nsub;
    isub=loc(fullsub=subs[i]);
    cmb=subs[i];
    icmb=subm[i,jcmb];
    covsub=oldcov[icmb,icmb];
    invcov=inv(covsub);  

if nc>9 then
dlab=concat('y',char(1:9,1,0))||concat('y',char(10:nc,2,0))
end;

if nc<10 then
dlab=concat('y',char(1:nc,1,0))
end;

if notc="=" then
print '------------------ Results ------------------'
print 'Parameters';
print bft [c=desvars];
print 'Correlation matrix';
print corr [r=dlab c=dlab format=5.3];
print 'Sigma of residuals, sigma reverse regression,' 'F for each combination, N of each combination,' 'weights, sum of weights';
print subs [c=slab]
ysig [c=ylab]
wsig [c=wlab]
f [c=flab]
ncmb [c=nlab format=8.0]
wt [[c=dlab format=5.3]];
print ntot 'Events' npts 'Points' npar 'Parameters' ' Block size' nc [format=4.0];
print 'Log-likelihood' loglk;
print '------------------'/'
finish;

The author may be reached at:
Donald H. MacQueen
Lawrence Livermore National Laboratory
7000 East Ave., L-307
Livermore, CA 94550
Electronic mail: macq@miguel.lnl.gov

Acknowledgements
SAS/IML is a registered trademark of SAS Institute Inc., Cary, NC, USA