ANALYZING LARGE DATA SETS USING PROC SSCP AND GLM5

James D. Hosking and Ronald W. Helms, University of North Carolina at Chapel Hill

Since the GLM5 macros operate within PROC MATRIX, they are subject to the limitations of PROC MATRIX as well as benefiting from its capabilities as a language for statistical programming. One such limitation is the requirement that the matrices on which it is to operate be small enough to be kept in core. In fact, the practical upper limit on the size of a matrix is 32,000 elements since many PROC MATRIX operators and functions will not work with larger matrices. Many practical linear models problems involve data sets larger than this.

A second limitation of PROC MATRIX is the fact that the arithmetic operators and functions in SAS do not permit one to take advantage of the symmetry of $Z^T Z$ in computing the same of squares and cross products. If the ratio of the number of observations to the number of variables is large enough, computing only one triangle of $Z^T Z$ would reduce computational costs substantially.

Both of these issues became quite salient to us last year, when we wanted to use GLM5 to analyze a data set with over 15,000 observations and 150 columns in $Z$. Fetching $Z$ into core and computing both triangles of $Z^T Z$ would have required over 37,500K and would have meant performing 179,000,000 (179 million) unnecessary double precision multiplications (and the same number of unnecessary double precision additions) when computing $Z^T Z$.

In this paper, I will discuss the facilities provided in the GLM5 system for dealing with large data sets. A data set can be large in two ways. First, it can have a large number of observations; that is $Z$ can have a large number of rows. Second, it can have many variables; that is $Z$ can have a large number of columns. The first section of this paper discusses techniques for dealing with problems caused by a large number of observations. The second section suggests ways to deal with the problem of a large number of variables.

Procedures for a Large Number of Observations

In order to analyze data sets containing large numbers of observations, we wrote a SAS procedure, PROC SSCP. It computes a sum of squares and cross products matrix and outputs it as a SAS data set. Since PROC SSCP has only one observation in memory at a time, its core requirements are modest. For the analysis mentioned above, less than 200K was required by PROC SSCP. Only the upper triangle of $Z^T Z$ is computed. The procedure is written in PDL and is quite fast. The output data set created by PROC SSCP has the same structure as the matrix _SSCP which is produced by the GLM5 module SAVESSCP. Figure 1 illustrates this structure in detail. Basically, the first four observations (rows) in the data set contain various parameters used by GLM5, and the remaining observations are the (uncorrected) sum of squares and cross products matrix. The procedure has a number of other options, which are detailed in the procedure user's guide. PROC SSCP is distributed by us, as part of the GLM5 system. Once its load modules have been put in a library, using it is quite simple. That library is given the ddname SASLIB in your JCL, which the SAS supervisor automatically scans when a requested procedure is not in the main SAS library.

An example of the use of PROC SSCP is shown in Figure 2. The input data set, containing the independent and dependent variables is named Z. The output data set will be named SSCP1, and stored in the file referenced by the ddname MYFILE.

To use the data set with GLM5, first FETCH the data set into memory, naming it _SSCP_ and using the COLNAME parameter in the fetch statement to read the variable names into a matrix named _SSCPN M_. The LINK to GETSS creates from _SSCP_ and _SSCPNM_ the matrices needed by the other modules. Then use the other GLM5 modules just as if you had a small data set and had used MAKESS rather than PROC SSCP.

PROC SSCP provides the simplest method for creating the input needed by GLM5, since the data set it produces is exactly the same as that produced by the SAVESSCP module. It is possible however, to use other SAS procedures, such as PROC CORR or PROC SYSSREG to produce the necessary input matrices.

For example, Figure 3 shows a job in which PROC CORR is used to produce the same results gotten with PROC SSCP in Figure 2. In Figure 3, the input matrices required by FITMODEL are created separately, because this is simpler than trying to create _SSCP_. After they have been created, the SAVESSCP module is used to form _SSCP_ and _SSCPNM_ from them. These are output to disk in the example. In later runs, the data set _MYFILE_.SSCP1 could be used exactly as it was in Figure 2. PROC SYSSREG could be used in much the same way as PROC CORR was.

We have run comparisons of PROC CORR and PROC SSCP on data sets with between 100 and 10,000 observations and between 3 and 200 variables. Typically, PROC SSCP is much faster than PROC CORR and requires somewhat less core. For example, with an input data set containing 220 variables and 250 observations, PROC SSCP used 308K and 73 seconds of CPU time while PROC CORR used 402K and 366 seconds of CPU time.

Not only is it feasible to use GLM5 to analyze data sets with an extremely large number of observations, as the number gets very large GLM5 will usually have important cost advantages over PROC CLM and PROC ANOVA. With large data sets, the cost of computing $Z^T Z$ can be substantial.

In our 16,000 observation, 150 variable problem,
computing $Z^2$ required over 25 minutes of CPU time on a model 360/75. By using PROC SSCP, and the GETSS module this computation can be done once, and stored as a SAS data set. This stored SSCP matrix may then be analyzed repeatedly using GLMMS. With GLM or ANOVA, in contrast, $X^T X$ must be recomputed each time analyses are to be performed. In the analysis of large complex data sets, it is frequently the case that the examination of an initial analysis will generate interesting post-hoc hypotheses. Using GLMMS, such hypotheses can be investigated without the expense of recomputing $X^T X$.

Procedures for a Large Number of Variables

The second way in which a data set may be large, is to have a large number of columns in $Z$. This may come about because of a large design (many columns in $X$) or because of many dependent variables (many columns in $Y$). This can make the size of $Z^2$ large enough to be important. If, for example, $Z$ has 180 columns, then $Z^2$ will be $180 \times 180$, roughly 254K when fetched into core in PROC MATRIX. We mention 180 as an example, because a $180 \times 180$ matrix has 32,400 elements and is the largest square matrix which PROC MATRIX can process. As a general rule of thumb, the various temporary matrices created by GLMMS will require a total of roughly twice the core used to store $SS$, plus a workspace of roughly 750K would be required to process this matrix. In addition to the storage required for matrices, the code produced when PROC MATRIX compiles the GLMMS macros into a load module must be stored in core. The full GLMMS system consists of about 3,000 PROC MATRIX statements, of which about 1,000 are executable statements which must be compiled and stored in core. This code takes up about 250K. Thus, in order to analyze the largest possible problem, using the full GLMMS system, a region of roughly 1,000K would be required. While this is quite a lot, the comparisons in Table 1 demonstrate that for many large problems, GLMMS requires far less core than PROC GLM, in spite of our inability to use overlays.

On our 360/370 network at the University of North Carolina, 1,000K is available without special arrangement, so we don't view this as a major problem. Certainly, anyone using SAS on a virtual system should have 1,000K available for the analysis of large problems. However, there are ways in which the user who does have limited core available can reduce the region needed to use GLMMS to analyze data sets with a large number of columns in $Z$.

The first thing that the user can do to reduce the core requirements of GLMMS is to bring in to PROC MATRIX only those modules which are to be used. Figure 6 shows the structure of the GLMMS modules. In general, each of the larger modules is in its own separate macro. The macro SYSTEM contains the code needed to initialize the matrices used, and those modules which must always be in core. These mainly perform internal housekeeping functions. The remaining modules can be used independently of each other. Hence, a user with a large data set, who wanted only to fit a linear model and then estimate secondary parameters, could reduce his core requirements by using the code shown in Figure 5. First PROC SSCP is used to create _SSCP_, and then PROC MATRIX is invoked. Rather than using the INITIAL, LINMOD and CONSTRUCT statements which compile all of the modules in GLMMS, the user compiles only SYSTEM and the three macros which he wants to use. Thus PUTOUT, CANCORR, MAEKS, PRINCOV and SAVESSCP are not compiled (and not available for use). This would reduce the core requirements by roughly 75K. By only compiling the macros containing the modules which are needed for a particular run, 50 to 100K can often be saved with no added work by the user. If this is still not enough of a reduction in core, the user can, at the cost of added work and execution time, reduce the size of the GLMMS code even further. We have written the various sections of GLMMS as fairly autonomous modules, which all operate on a small common set of data and parameter matrices. These matrices are those which comprise the SSCP matrix which is produced by SAVESSCP or by PROC SSCP. $SS$ is initially the matrix of uncorrected sums of squares and cross-products for the columns of $Z$. FITMODEL sweeps this matrix, turning it into a matrix containing $\bar{Z}$, $(N-p) \bar{Z}$, and $(\bar{Z}' \bar{Z})^{-1}$. The following are the primary parameter estimates for general linear models. _SUMSQ_ is a vector holding the original sums of squares of the independent and dependent variables. It is used primarily for setting tolerances and in checking for computational accuracy. _PARM_ is a vector which contains $N$, the number of dependent variables, the rank of $Z$, the number of columns of $Z$, not currently in either $X$ or $Y$ and an epsilon which is used to determine when a diagonal element of _SS_ is within rounding error of zero. _VTYPE_ is a status vector which indicates whether a particular column of $Z$ is a dependent variable, an independent variable, or not currently in the model. _VNAME_ is a vector of variable names, and _ECODE_ is a vector of error codes set by the modules to inform later modules that errors have been encountered. All of the statistical modules which build on the results of FITMODEL, such as TESTGLH, CANCORR, and PRINCOV, use only these matrices as input from FITMODEL.

The SAVESSCP module can be used to create a SSCP matrix from _SS_, _VTYPE_, etc., after FITMODEL has been used to fit a linear model. This data set can be output as a SAS data set. In a later job (or a later step within your SAS program) this already swept SSCP matrix can be FETCHed, GETSS can be used to create _SS_, _PARM_, etc., _SSCP_ can be freed and the TESTGLH, CANCORR, or PRINCOV code can then be compiled and used. Thus the user can use multiple PROC MATRIX steps, in successive jobs or in the same job to effectively overlay the GLMMS code. Figure 6 is an example of this technique.

Even when sufficient core is available, using SAVESSCP and GETSS after fitting a model with FITMODEL can be useful with large problems. The work involved in fitting the model can be done once, and secondary parameters can then be estimated.
mated repeatedly in later jobs.

In summary, we have found that virtually any size design which can be analyzed by SAS linear models procedures such as PROC GLM and PROC ANOVA can be analyzed with GLM5. For some of these GLM5 offers significant savings in time and core required. For some of these GLM5 offers significant savings in time and core required.

Figure 1. The structure of an _SSCP_ matrix.

//BIGMEM2 JOB
// EXEC SAS
// MYFILE DD DSN=my.personal.SAS.disk.file,
// DISP=OLD;
//SYSIN DD *
TITLE GLM5 ANALYSIS OF A LARGE DATA SET;
TITLE2 USING PROC CORR;
{code to create Z}
PROC CORR DATA=Z NOCORR SSCP NOMISS OUT=SSCP1;
VARIABLES list of variables;
* CREATE DATA SETS CONTAINING _SS_ AND _N_;
DATA SS(KEEP= list of vars in input data set)
  N(KEEP= ANY ONE VARIABLE);
SET SSCP1;
IF _TYPE_='N' THEN OUTPUT N;
IF _TYPE_='SSCP' THEN OUTPUT SS;
PROC MATRIX;
INITIAL;
* CREATE INPUTS FOR FITMODEL;
FETCH _SS_ DATA=SS COLNAME=_VNAME_;
_VTYPE_ =J(1,NCOL(_SS_),0);
_SUMSQ_ =VTYPE=ECODE=(_SS_;);
_ECODE_ =J(20,1,0);
FETCH N DATA=N;
_PARM_ =J(1,0,0)|NCOL(_SS_);(0);
* PARM IS: (N NCOL(Y) RANK(X) NCOL(Z) DELTA);
* USE SAVESSCP TO CREATE A _SSCP_ MATRIX;
LINK SAVESSCP;
* STORE IT FOR REUSE LATER;
OUTPUT _SSCP_ OUT=MYFILE.SSCP1 COLNAME=_SSCPNM_;
FREE _SSCP_ SSCPNM_;
* NOW CONTINUE WITH FITMODEL, TESTGLH, ETC;
Figure 3. Using PROC CORR to perform the computations shown in Figure 2.
Figure 4. Structure and core requirements of GLMM modules.

```plaintext
PROC SSCP DATA=BIG OUT=SSCP1;
PROC MATRIX;
  _SYSTEM_ * SYSTEM SUBROUTINES;
  _GETS_ * GETSS MODULE;
  _FMOD_ * FITMODEL MODULE;
  _TEST_ * TESTGLH MODULE;
  FETCH _SSCP_DATA=SSCP1 COLNAME=_SSCPNM_;
  LINK GETSS;
  INDVARS=list of independent variable names;
  DEPVARS=list of dependent variable names;
  LINK FITMODEL;
  [statements defining C, U, and G]
  LINK TESTGLH;
Figure 5. Reducing GLMM core requirements by compiling only necessary macros.

PROC SSCP DATA=ONE OUT=SSCPP1;
PROC MATRIX; * FIT THE LINEAR MODEL HERE;
  _SYSTEM_ _GETSS_
  FETCH _SSCP_DATA=SSCPP1 COLNAME=_SSCPNM_;
  LINK GETSS;
  FREE _SSCP_ _SSCPNM_;
  _FMOD_ * COMPILIE FITMODEL;
  INDVARS=list of independent variables;
  DEPVARS=list of dependent variables;
  LINK FITMODEL;
  _SAVESS_ * COMPILIE SAVESSCP;
  LINK SAVESSCP;
  OUTPUT _SSCP_OUT=SSCPFIT COLNAME=_SSCPNM_;
  PROC MATRIX; * ESTIMATE SECONDARY PARAMETERS;
  _SYSTEM_ _GETSS_ _TEST_
  FETCH _SSCP_DATA=SSCPFIT COLNAME=_SSCPNM_;
  LINK GETSS;
  FREE _SSCP_ _SSCPNM_;
  [statements defining C, U, and G]
  LINK TESTGLH;
Figure 6. Using multiple PROC MATRIX steps to overlay GLMM source code.
```
TABLE 1
Comparison of the Core Requirements of GLMM5 and PROC GLM for Large Complete Factorial Designs

<table>
<thead>
<tr>
<th>Design (('# cells')</th>
<th>Core Required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GLMM5 &amp;</td>
</tr>
<tr>
<td></td>
<td>PROC SSCP</td>
</tr>
<tr>
<td></td>
<td>PROC GLM</td>
</tr>
<tr>
<td>CRF-3 (3)</td>
<td>232K</td>
</tr>
<tr>
<td>CRF-3x3x3 (27)</td>
<td>262K</td>
</tr>
<tr>
<td>CRF-3x3x3x2 (54)</td>
<td>302K</td>
</tr>
<tr>
<td>CRF-3x3x3x2x2 (108)</td>
<td>654K</td>
</tr>
</tbody>
</table>

1All designs had four dependent variables and 216 subjects. All GLMM5 modules were compiled, including those not needed for the analyses.

PROC GLM could not perform the analysis in 1000K, this figure is the amount necessary as computed and printed by PROC GLM.