Paper 256-2009

All the Cows in Canada: Massive Mixed Modeling with the HPMIXED Procedure in SAS® 9.2

Tianlin Wang and Randy Tobias, SAS Institute Inc., Cary NC

ABSTRACT

Linear mixed models enable you to borrow strength for estimation, testing, and prediction by linking observations through a covariance model. Many applications of mixed models lead to computational problems that can only be described as huge. An example is the prediction of breeding value for a large population of livestock with a covariance model that links animals through their pedigrees. The HPMIXED procedure fits large, sparse linear models by using a number of specialized high-performance techniques, which include sparse matrix technology and algorithms for solving the mixed model equations and for optimization. These techniques are tailored to very large systems. They can result in significant savings of both memory and CPU resources and can actually make feasible the fitting of some models that would be infeasible with other approaches. This paper introduces applications of large mixed models, discusses the specialized techniques of the HPMIXED procedure to handle them, and demonstrates the utility of the procedure with examples from agriculture and genomics.

INTRODUCTION

Linear statistical models are widely used to understand and predict a great variety of phenomena in science, business, and industry, from disease prevalence to customer behavior to manufacturing systems. Linear models with components whose effects are random, which essentially represent a sample from a hypothetical population, are called *mixed models* (Graybill 1976; Searle, Casella, and McCulloch 1992). The fundamental role of random effects in mixed models is to link observations that have the same level of the effect with a covariance term, enabling you to borrow strength from related observations for estimation, prediction, and testing. For example, in animal genetic evaluation, yields (say, milk production records from a dairy cow) are categorized by animal, species, season, farm, and so forth; commonality in any one of these categories indicates relatedness that can be exploited to sharpen predictions for individual animals.

SAS/STAT® software provides a number of specialized features for analyzing data with mixed models, including the MIXED, GLIMMIX, and NLMIXED procedures. These features provide a rich toolset for a wide variety of problems, small to large. However, it is common for applications of mixed models to go beyond large, becoming what can only be described as huge. For example, in the animal breeding situation discussed above, data for thousands or even millions of animals might be involved, across many different farms and species. Each farm or species, and probably their interaction as well, is associated with additional fundamental equations that need to be solved in order to fit the model. For a very large problem like this, without special techniques the mixed model equations might be too large to fit into computer memory, much less be solvable in a reasonable amount of time.

The special techniques required to fit huge linear mixed models are the province of the new HPMIXED procedure, which is experimental in SAS/STAT 9.2 software. The fundamental approach is to use *sparse matrix techniques*—specialized linear algebra methods for matrices with many zeros. The HPMIXED procedure also employs specialized high-performance algorithms for optimization, tailored to very large problems. The procedure complements the MIXED procedure and other SAS/STAT procedures for linear modeling by supporting a subset of the models that can be fit with these more versatile tools, while providing considerably better performance for this subset in terms of memory requirements and computational speed.

This paper discusses sparse matrix techniques, present specialized high-performance algorithms for optimization that use these techniques, give an overview of the HPMIXED procedure which incorporates these techniques and algorithms, and demonstrate the utility of the procedure with examples from agriculture and genomics.

COMPUTATION

Mixed Model Equations and the Likelihood

This section discusses the general form of the equations involved in fitting mixed models, indicating where special techniques can be employed for large problems. To begin, recall that a (regular) linear model for how a vector of responses $\mathbf{y}$ relates to a design matrix $\mathbf{X}$ can be written in the following general form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1}$$

where the coefficients $\boldsymbol{\beta}$ are unknown parameters to be estimated and $\boldsymbol{\epsilon}$ is the random error. Assuming the elements of $\boldsymbol{\epsilon}$ are independent and identitically distributed with variance σ^2, the preceding equation can be written equivalently in terms of the mean and variance of $\mathbf{y}$:

$$\mathrm{E}(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta},\ \mathrm{Var}(\mathbf{y}) = \sigma^2\mathbf{I} \tag{2}$$

where $\mathbf{I}$ is the identity matrix.

In this context a general linear *mixed* model can be viewed as having the same form but distinguishing between fixed and random effects:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon} \tag{3}$$

where $\boldsymbol{\beta}$ is fixed as in equation (1) but $\mathbf{u}$ is assumed to be random, with mean $\mathbf{0}$ and variance $\mathbf{G}$, so that

$$\mathrm{E}(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta},\ \mathrm{Var}(\mathbf{y}) = \mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \sigma^2\mathbf{I} \tag{4}$$

For example, in a multicenter clinical trial for a new drug, the different dose levels are usually modeled using fixed effects, but different clinics are modeled using random effects. The columns of $\mathbf{Z}$ are dummy variables that correspond to the different clinics, making $\mathbf{Z}$ potentially large but also quite sparse; that is, most elements of $\mathbf{Z}$ are 0. The impact of the random clinic effect is to model covariance between observations collected by the same clinic.

In order to fit a mixed model, first note that for given $\mathbf{G}$ and σ^2 fixed effects are estimated and random effects are predicted by solving the so-called *mixed model equations* (MME):

$$\begin{bmatrix} \mathbf{X}'\mathbf{X}/\sigma^2 & \mathbf{X}'\mathbf{Z}/\sigma^2 \\ \mathbf{Z}'\mathbf{X}/\sigma^2 & \mathbf{Z}'\mathbf{Z}/\sigma^2 + \mathbf{G}^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y}/\sigma^2 \\ \mathbf{Z}'\mathbf{y}/\sigma^2 \end{bmatrix} \tag{5}$$

The pieces of these equations show up repeatedly throughout the paper, calculated with various vectors taking the role of $\mathbf{y}$. So for notational convenience MME is rewritten as

$$\mathbf{C} \times \mathrm{sol}(\mathbf{y}) = \mathrm{rhs}(\mathbf{y}) \tag{6}$$

where

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}'\mathbf{X}/\sigma^2 & \mathbf{X}'\mathbf{Z}/\sigma^2 \\ \mathbf{Z}'\mathbf{X}/\sigma^2 & \mathbf{Z}'\mathbf{Z}/\sigma^2 + \mathbf{G}^{-1} \end{bmatrix} \tag{7}$$

$$\mathrm{rhs}(\mathbf{y}) = \begin{bmatrix} \mathbf{X}'\mathbf{y}/\sigma^2 \\ \mathbf{Z}'\mathbf{y}/\sigma^2 \end{bmatrix} \tag{8}$$

and $\mathrm{sol}(\mathbf{y})$ is a solution to equation (6). Calculations that involve $\mathbf{C}$, $\mathrm{sol}(.)$, and $\mathrm{rhs}(.)$ play a fundamental role in analyzing the mixed model. For example, the residual maximum likelihood (REML) method for analyzing mixed models (Harville 1977) depends on the residual negative log-likelihood function with respect to all the unknown parameters $\boldsymbol{\theta}$ in $\mathbf{G}$ and σ^2, which can be expressed as

$$L(\boldsymbol{\theta}, \sigma^2) = K + \log|\mathbf{C}| + \log|\mathbf{G}| + \mathbf{y}'\mathbf{P}\mathbf{y} \tag{9}$$

where K is a constant that depends on the problem and

$$\mathbf{P} = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-}\mathbf{X}'\mathbf{V}^{-1}$$

is the matrix that transforms observations into residuals. The term $\log|\mathbf{C}|$ can be computed as a biproduct of solving the MME in addition to the quadratic form $\mathbf{y}'\mathbf{P}\mathbf{y}$. This is because for any two vectors $\mathbf{a}$ and $\mathbf{b}$, the bilinear form $\mathbf{a}'\mathbf{P}\mathbf{b}$ can be computed as

$$\mathbf{a}'\mathbf{P}\mathbf{b} = \frac{1}{\sigma^2}\mathbf{a}'\mathbf{b} - \mathrm{sol}(\mathbf{a}) \times \mathrm{rhs}(\mathbf{b}) \tag{10}$$

(see Johnson and Thompson [1995] for a proof). General terms that have this bilinear form are key calculations in the average information iterative method of optimization, discussed in the section "Average Information Matrix" on page 4. For now, the preceding fact is used to re-express the log likelihood in a form that explicitly depends on solving the MME:

$$L(\boldsymbol{\theta}, \sigma^2) = K + \log|\mathbf{C}| + \log|\mathbf{G}| + \left(\frac{1}{\sigma^2}\mathbf{y}'\mathbf{y} - \mathrm{sol}(\mathbf{y}) \times \mathrm{rhs}\ (\mathbf{y})\right) \tag{11}$$

As this form makes clear, solving the MME is the key to analyzing the mixed model.

Note that **C** is a square symmetric matrix with dimensions equal to the sum of the number of fixed and random parameters. Thus, if any fixed or random effect has many levels (as in the example of the multicenter clinical trial mentioned previously), **C** can be quite large. With more than a few hundred rows and columns, direct (dense) calculations that involve **C** become computationally difficult, requiring far too much memory or CPU time or both. With more than a thousand or so rows and columns, direct calculations become infeasible. The specialized techniques of HPMIXED consist mainly of efficiently calculating $L(\boldsymbol{\theta}, \sigma^2)$ and its derivatives, employing sophisticated sparse matrix technology to store **C** and to solve the MME. These techniques are discussed in the section "Sparse Matrix Storage and Computation" on page 3.

Measures to Speed Up the MIXED Procedure

The MIXED procedure fits models with the same general form as discussed in the last section, and for large models you can try several measures to speed up the calculations. For example:

- Determine whether it is possible to factor out a common effect from the random effects and use it in a SUBJECT= option. This technique reduces memory requirements dramatically when the subject has hundreds or thousands of levels.
- Use the NOTEST option. Tests on fixed effects with many levels can be computationally costly and are often not useful.
- Use the residual degree-of-freedom method to quickly determine denominator degrees of freedom. The containment method for degrees-of-freedom calculations (which is the default for the MIXED procedure) requires large storage of dense matrices. The Kenward-Roger method (Kenward and Roger 1997) provides accurate standard errors and p-values in mixed models, particularly when sample sizes are small, but it can require large amounts of memory and computing time depending on the model structure and user options.

These measures can extend the range of problems that PROC MIXED can handle, but there is still an upper limit to the models that can be fit by its general approach. This is because the procedure works by directly storing and manipulating dense matrices on the order of **C** in the MME. Although this approach allows MIXED to fit mixed models of great generality, the approach is fundamentally restricted by the sheer size of such matrices for large models. For this reason, a key technique in the HPMIXED procedure is the use of sparse matrix methods, which is discussed in the following section.

Sparse Matrix Storage and Computation

Sparse matrices are ones that have many zero elements. Linear algebra techniques offer two useful aspects specifically tailored to handling sparse matrices. First, sparse matrix storage significantly reduces the amount of space necessary to store the matrix, essentially by storing only the nonzero elements and their positions. Second, sparse matrix operations involve only the nonzero elements, significantly reducing computational time. This section discusses these two aspects and applies them to solving the MME for analyzing a mixed model.

Usually, computer programs represent an $N \times M$ matrix in a dense form as an array of size NM, making row-wise and column-wise arithmetic operations particularly efficient to compute. However, if many of these matrix elements are zeros, then correspondingly many of these operations are unnecessary or trivial. Sparse matrix techniques exploit this fact by representing a matrix not as a complete array, but as a set of nonzero elements and their locations (row and column) within the matrix. Sparse matrix techniques require special methods for finding and operating on only nonzero matrix elements. Sparse techniques are more efficient if the dense form contains enough zero element operations to make these special methods worthwhile.

To give an idea of the computational gains that sparse techniques afford, suppose that you need to multiply two $N \times N$ matrices, each with only n nonzero elements. Storing these matrices in the usual, dense form requires an amount of memory proportional to N^2, and the simplest way of multiplying them requires a number of arithmetic operations proportional to N^3. In this case, storage is said to be $O(N^2)$ and computation to be $O(N^3)$. On the other hand, a sparse approach is $O(n)$ for storage and $O(Nn)$ for computation. If n is much smaller than N^2, then both of these reductions can make the difference between a feasible and an infeasible computational task.

To illustrate sparse matrix storage, let the symmetric matrix **C** be the matrix of mixed model equations of size 5×5.

$$\mathbf{C} = \begin{bmatrix} 8.0 & 0 & 0 & 2.0 & 0 \\ 0 & 4.0 & 3.0 & 0 & 0 \\ 0 & 3.0 & 5.0 & 0 & 0 \\ 2.0 & 0 & 0 & 7.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 \end{bmatrix}$$

There are 15 elements in the upper triangle of **C**, but 8 of them are zeros. The row and column indices and the values of the 7 nonzero elements are listed as follows:

i	1	1	2	2	3	4	5
j	1	4	2	3	3	4	5
C_{ij}	8.0	2.0	4.0	3.0	5.0	7.0	9.0

Because the sparse form requires both positions and values to be stored, it does not lead to a significant reduction in memory use for this small example. However, as the size of the matrix grows, the sparse memory requirements grow only linearly with the number of nonzero elements, rather than quadratically with the size of the matrix.

Sparse matrix techniques are potentially useful for many linear models, but they are particularly apt for mixed models. In mixed models, sparse matrices usually arise from a large number of levels for random effects. For example, in the example of the multicenter clinical trial mentioned previously, if the trial involves 5 doses and 1500 clinics, then the MME has 1,505 equations, but only about 4 nonzero elements per equation. Storing just the upper triangle of these equations in a dense form requires $(1 + 1505) \times 1505/2 = 1,133,265$ elements, but the sparse form requires less than 1% of that amount of memory.

When applicable, sparse matrix techniques can provide huge efficiency improvements in terms of both memory and computation time. However, the key when working with sparse matrices is to make sure that you *only and always* work with matrices that are indeed sparse. As soon as a dense matrix pops up, it dominates the memory and CPU requirements for the analysis. For example, although the matrix **C** in the MME is usually very sparse, its generalized inverse $\mathbf{C}^-$ is not. For this reason, the likelihood and derivative calculations for REML must avoid computing $\mathbf{C}^-$ directly. Toward this end, two further techniques employed by the HPMIXED procedure are sparse Cholesky factorization and partial sparse inversion.

A *Cholesky factorization* of a symmetric, nonnegative definite matrix **C** computes the unique upper triangular matrix **L** such that $\mathbf{L}'\mathbf{L} = \mathbf{C}$. Cholesky factorization is often employed in matrix analysis, because complicated computations for **C** are often equivalent to much simpler computations in terms of **L**. For example, the determinant of **C** is equal to just the square of the product of the diagonal elements of **L**. However, as with matrix inversion, the direct Cholesky factor of **C** is not usually sparse and thus must be avoided. The HPMIXED procedure employs the sparse Cholesky factorization algorithm (George and Liu 1981) to reorder the rows and columns of **C** so that its Cholesky factor *is* sparse.

Nonlinear optimization of the REML log likelihood requires, approximately, first derivatives (to indicate the direction of steepest ascent) and an information matrix (to say how far to move in that direction). In addition to sparse matrix techniques, the HPMIXED procedure uses an alternative information matrix that admits more scope for sparse techniques than standard techniques. This so-called average information method is discussed in the following section.

Average Information Matrix

The HPMIXED procedure fits mixed models by using a modified form of the iterative Newton-Raphson method for minimizing the negative log-likelihood function $L(\boldsymbol{\theta}, \sigma^2)$ with respect to the parameters $\boldsymbol{\theta}$. In a sense, the modification is the average of strict Newton-Raphson iteration and an alternative form called Fisher scoring. This average information method is particularly suited to sparse matrix processing, as discussed in the following.

Newton-Raphson and Fisher scoring are alternative iterative methods for maximum likelihoods, having the following general characterization. A single iteration of Newton-Raphson involves updating a current estimate for the parameters $\boldsymbol{\theta}^{\text{curr}}$ to obtain a new estimate $\boldsymbol{\theta}^{\text{new}}$ according to the formula

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{curr}} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{\text{curr}})\mathbf{g}(\boldsymbol{\theta}^{\text{curr}}) \tag{12}$$

where $\mathbf{g}(\boldsymbol{\theta}) = \frac{\partial L}{\partial \boldsymbol{\theta}}$ is the first derivative vector of L and $\mathbf{H}(\boldsymbol{\theta}) = \{\frac{\partial^2 L}{\partial \theta_i \partial \theta_j}\}$ is the second derivative (Hessian matrix). The matrix $-\mathbf{H}(\boldsymbol{\theta}^{\text{curr}})$ is also known as the *observed information matrix*; at convergence (when $\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{curr}}$), its inverse gives an approximate variance-covariance matrix for the resulting parameter estimates. The Fisher scoring method

is an alternative to this strict form of Newton-Raphson, replacing the observed information matrix $-\mathbf{H}(\boldsymbol{\theta}^{\text{curr}})$ with its expectation $-\text{E}(\mathbf{H}(\boldsymbol{\theta}^{\text{curr}}))$. The expected information matrix can be more stable or easier to compute than the observed one, which is why Fisher scoring is often preferred to strict Newton-Raphson.

The *average information* (AI) method (Johnson and Thompson 1995) involves iterative updates as with Newton-Raphson and Fisher scoring. But instead of using either the observed or expected information matrix, it uses their average $-(\mathbf{H}(\boldsymbol{\theta}^{\text{curr}}) + \text{E}(\mathbf{H}(\boldsymbol{\theta}^{\text{curr}})))/2$. AI can be expected to share some of the stability benefits of Fisher scoring, but its real beauty for mixed models is the computational efficiency it affords. Both the observed and expected information matrices for $L(\boldsymbol{\theta}, \sigma^2)$ involve a certain term that is complicated and difficult to compute. When they are averaged, this term drops out, so that the average information matrix with respect to scalar variance components θ_i and θ_j has the form (Harville 1977):

$$\text{AI}(i,j) = \mathbf{y}'\mathbf{P}\frac{\partial \mathbf{V}}{\partial \theta_i}\mathbf{P}\frac{\partial \mathbf{V}}{\partial \theta_j}\mathbf{P}\mathbf{y} \tag{13}$$

Defining the vector

$$\mathbf{f}(\theta_j) = \frac{\partial \mathbf{V}}{\partial \theta_j}\mathbf{P}\mathbf{y}$$

the average information matrix can be re-expressed as

$$\text{AI}(i,j) = \mathbf{f}(\theta_i)'\mathbf{P}\mathbf{f}(\theta_j)$$

As noted in the discussion about equation (10), this bilinear form can be efficiently computed by solving the mixed model equations with $\mathbf{y}$ replaced by $\mathbf{f}(\theta_i)$.

For certain common variance models, the form of the vector $\mathbf{f}(\theta_i)$ can also be expressed as a vector of bilinear forms and thus can be calculated by solving the MME. For example, the $\mathbf{f}(\theta_i)$ for the random effect $\mathbf{u}_i$ in a linear mixed model can be expressed as

$$\mathbf{f}(\theta_i) = \mathbf{Z}_i\mathbf{Z}_i'\mathbf{P}\mathbf{y} = \frac{1}{\sigma^2}\mathbf{Z}_i\hat{\mathbf{u}}_i \tag{14}$$

The upshot is that the average information matrix can be computed by solving the mixed model equations $q+1$ times for a model with q simple random effects, without the need to store and invert any really large matrices. Combined with sparse matrix techniques, the average information algorithm achieves a significant reduction in the memory and CPU requirements for large models.

SYNTAX AND FEATURES OF THE HPMIXED PROCEDURE

This section briefly presents the statements and options in the HPMIXED procedure and introduce its basic features. Since PROC HPMIXED shares much syntax with PROC MIXED, the focus is mainly on the differences. For complete documentation of all statements and options for the HPMIXED procedure, see the section "Syntax: HPMIXED Procedure" in the *SAS/STAT User's Guide.*

The HPMIXED procedure shares with the MIXED procedure many of the common mixed modeling statements such as CLASS, MODEL, RANDOM, CONTRAST, ESTIMATE, and LSMEANS. The syntax for these statements is the same in both procedures, except that PROC HPMIXED supports only a subset of options associated with each statement. For example, PROC HPMIXED currently offers only two covariance types: TYPE=VC (the default) for simple variance components, and TYPE=CHOL for modeling unstructured covariance matrices in terms of the Cholesky parameterization. The subset of functionality implemented is chosen so that these two covariance types can be efficiently implemented with the techniques introduced in the previous two sections.

Following are some differences between PROC HPMIXED and PROC MIXED:

- PROC HPMIXED does not support the PROC MIXED REPEATED statement, because PROC HPMIXED offers only one type of residual effect.
- PROC HPMIXED does not compute Type III linear tests unless you request precisely the ones you want by using the TEST statement. PROC MIXED computes these tests for all fixed effects by default. Type III tests can be computationally intensive to compute in large models and of questionable value for effects with very many levels.

- PROC HPMIXED offers the NLOPTIONS statement for specifying and tuning the optimization method. PROC MIXED offers no such statement.
- PROC HPMIXED has an OUTPUT statement for computing predicted values, residuals, and other statistics on individual observations. PROC MIXED offers no such statement.

The following examples show highlights of this syntax in action.

EXAMPLES

This section demonstrates the utility of the procedure with examples from agriculture and genomics. Note that the input and output of PROC HPMIXED are not really the focus of these examples, because you could use pretty much the same syntax to get the same results from other SAS/STAT mixed modeling tools (for example, PROC MIXED or PROC GLIMMIX). Rather, the point is to identify mixed models that are particularly apt for fitting with the HPMIXED procedure.

What makes a mixed model a good candidate for PROC HPMIXED? The short answer is that it is one with random effects that have many levels. You also have to use one of the two variance types that the HPMIXED procedure can handle: VC (simple variance components) and CHOL (Cholesky parameterization). Finally, the analysis must involve only those features of mixed modeling that can be performed using the sparse techniques of the HPMIXED procedure.

Ranking Cows Based on Random-Effect Coefficients

The title of this paper promises an application to large scale animal breeding problems. One example of such a problem is to compute the *estimated breeding value* (EBV) for each animal in a large population, with the goal of selecting animals with relatively high EBVs for breeding. An EBV is modeled as the predicted coefficient of the random effect for each animal. Thus if there are many animals to analyze, the HPMIXED procedure is the right choice of tool. The MIXED and GLIMMIX procedures can potentially compute EBVs, but PROC HPMIXED is particularly suited for the large, sparse matrix calculations involved.

The data for this problem are generated by simulation. Suppose there are 100 animals on each of 100 farms—hardly "all the cows in Canada," but about the number of cows in a large county. Furthermore, suppose the animals are categorized into five species. Finally, suppose there are about 40 observations of the response variable *Yield* made per animal. For example, in dairy cows *Yield* might be the pounds of milk produced per day. The following DATA step simulates such data.

```
%let NFarm = 100;
%let NAnimal = %eval(&NFarm*100);
%let Va = 4.0;
%let Ve = 8.0;
data Sim;
   keep Species Farm Animal Yield;
   array BV{&NAnimal};
   array AnimalSpecies{&NAnimal};
   array AnimalFarm{&NAnimal};
   do i = 1 to &NAnimal;
      BV           {i} = sqrt(&Va)*rannor(12345);
      AnimalSpecies{i} = 1 + int(   5  *ranuni(12345));
      AnimalFarm   {i} = 1 + int(&NFarm*ranuni(12345));
      end;
   do i = 1 to 40*&NAnimal;
      Animal  = 1 + int(&NAnimal*ranuni(12345));
      Species = AnimalSpecies{Animal};
      Farm    = AnimalFarm   {Animal};
      Yield   = 1 + Species
                  + Farm
                  + BV{Animal}
                  + sqrt(&Ve)*rannor(12345);
      output;
      end;
run;
```

In this simulation, the true breeding value for each animal has a variance component of 4.0, while the level of background variance is 8.0.

The effect of *Animal* is to be modeled as a simple random effect, inducing a constant correlation between the approximately 40 observed values of *Yield* for each animal; the EBV is the best linear unbiased predictor (BLUP) for each animal's random coefficient. In contrast, *Species* and the interaction between *Species* and *Farm* are typically modeled as fixed effects. The following HPMIXED statements to fit such a model are just like what you would use to fit the model in the MIXED or GLIMMIX procedures.

```
ods listing close;
proc hpmixed data=Sim;
   class Species Farm Animal;
   model Yield = Species Farm*Species;
   random Animal/cl;
   ods output SolutionR=EBV;
run;
ods listing;

proc sort data=EBV;
   by descending estimate;
proc print data=EBV(obs=10) noobs;
   var Animal Estimate StdErrPred Lower Upper;
run;
```

Note the CL option in the RANDOM statement. This requests that the BLUPs for the random effect of *Animal* be computed, along with 95% confidence limits (by default). Instead of printing these 10,000 coefficients, the listing destination is closed and the ODS OUTPUT statement saves the BLUPs to a data set named EBV. Finally, the EBV data set is sorted, and the 10 animals with the highest EBVs are displayed.

Figure 1 displays the EBVs of the top 10 animals, along with their precision and prediction limits.

Figure 1 Estimated Breeding Values: Top 10 Animals

Animal	Estimate	StdErr Pred	Lower	Upper
5707	7.6412	0.7325	6.2055	9.0768
2489	7.0352	0.6342	5.7921	8.2783
2161	6.8833	0.6347	5.6394	8.1272
3315	6.6600	0.5966	5.4908	7.8293
2958	6.1371	0.5745	5.0111	7.2630
2001	6.0613	0.6626	4.7626	7.3599
5927	6.0179	0.6160	4.8104	7.2253
2278	5.9773	0.6770	4.6504	7.3042
1346	5.9720	0.6696	4.6596	7.2844
9786	5.9701	0.6448	4.7063	7.2339

You can also use PROC MIXED and PROC GLIMMIX to compute EBVs, but the performance of these general mixed modeling procedures for this specialized kind of data and model is quite different from that of PROC HPMIXED. Here is the difference: whereas using a desktop-class PC to fit this model would take PROC MIXED or PROC GLIMMIX *hours* (assuming sufficient memory is available), the HPMIXED procedure runs in *under a minute* using relatively little memory. The MIXED and GLIMMIX procedures are engineered to have good performance properties across a broad class of models and analyses, a class much broader than what PROC HPMIXED can handle. On the other hand, The HPMIXED procedure can have better performance, in terms of both memory and run time, for certain specialized models and analyses, of which the current example is one. The next section explores this comparison between performance for these three procedures in more depth.

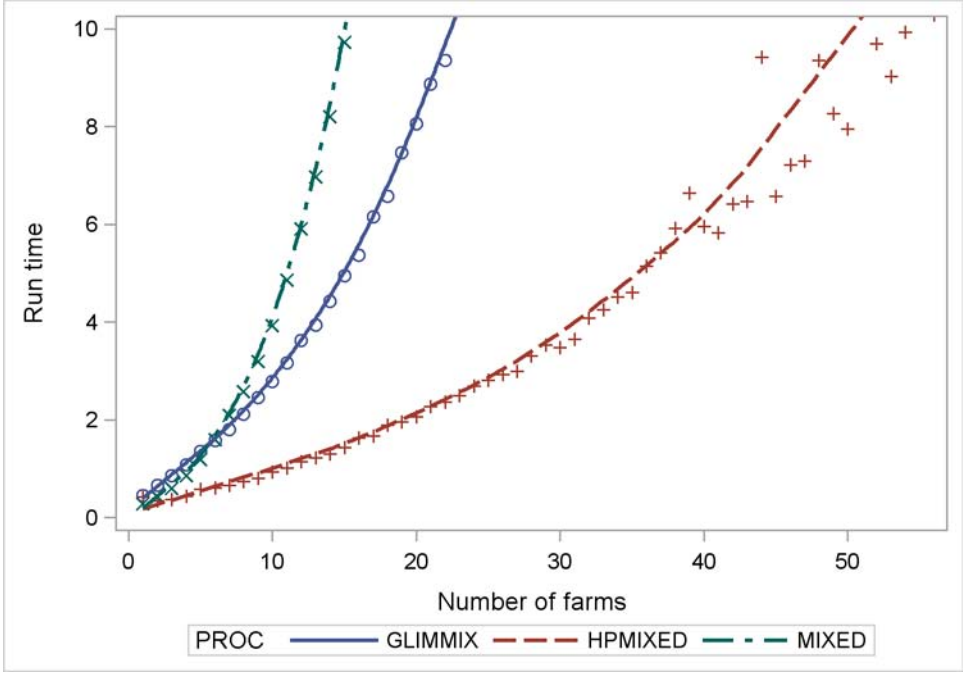
Exploring Relative Performance of Mixed Modeling Procedures

For the animal breeding example discussed in the previous section, equivalent PROC MIXED and PROC GLIMMIX approaches can take many times longer to run. It is difficult to provide numbers that accurately reflect relative performance across a broad variety of machines, because the MIXED and GLIMMIX procedures are sensitive to the speed of disk access for writing to and reading from the utility file that holds the underlying matrices. This example shows one approach for measuring relative performance of these procedures.

The SAS program provided in the appendix defines and runs a macro to perform a time trial for the HPMIXED, MIXED, and GLIMMIX procedures on the animal breeding example discussed in the previous section. Each procedure is run with data simulated on an increasing number of farms until its run time exceeds a specified maximum value *TopTime*,

and the results are plotted. For *TopTime*=10, the resulting plot of procedure run times on a server-class PC for increasing numbers of farms is shown in Figure 2.

Figure 2 Comparing Mixed Model Tools for Increasingly Sparse Problems



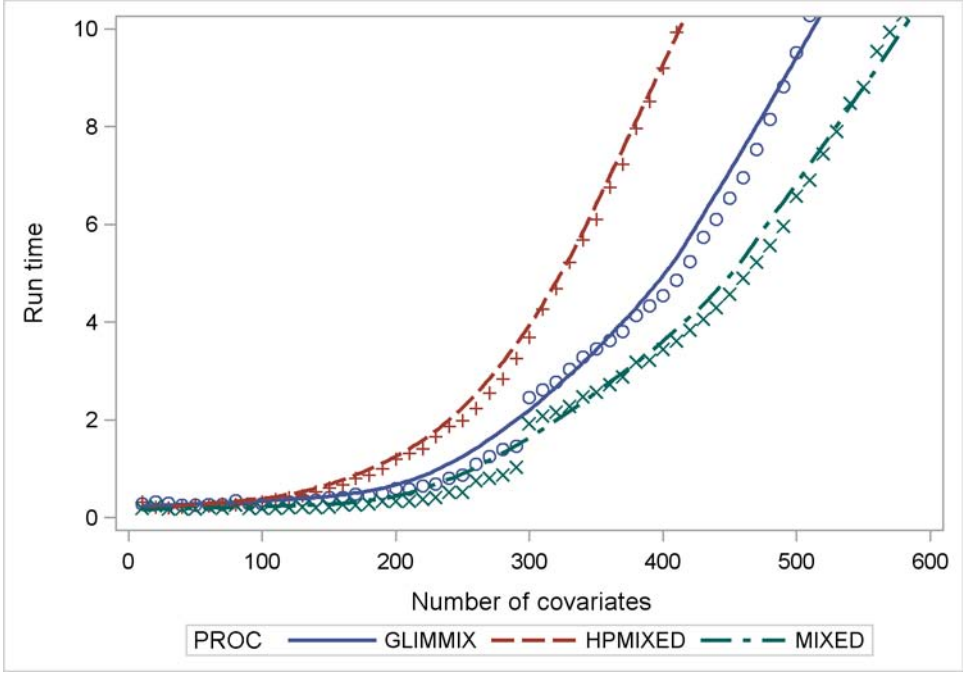
The vertical axis in Figure 2 measures run time. The plot shows that although the performance of the MIXED and GLIMMIX procedures is relatively competitive with the HPMIXED procedure for a few farms, relative performance of both procedures decreases rapidly for more than a dozen or so farms. As far as specific timings go, "your mileage may vary" on a different machine. But relative performance is what counts, and that is expected to be fairly invariant to machine architecture. This example shows that, for the limited class of models to which it applies, the sparse matrix representation that the HPMIXED procedure uses provides better computational performance than a dense representation, in terms of both run time and memory use.

However, it is critical to note that PROC HPMIXED can be quite *inefficient* relative to PROC MIXED and PROC GLIMMIX for models and data that are *not* sparse, because it can take many times longer to invert a large, dense matrix by sparse techniques. For example, Figure 3 shows relative performance of the three procedures for a study like the one above, but fitting a different model, one that involves only an increasing number *nCov* of continuous covariates as shown in the following statements:

```
proc hpmixed;
   model y = x1-x&nCov;
run;
```

The equations for this model are dense, so the sparse methods employed by the HPMIXED procedure are expected to be less efficient. Indeed the plot bears this out.

Figure 3 Comparing Mixed Model Tools for Increasingly Dense Problems



The HPMIXED procedure is competitive with PROC MIXED and PROC GLIMMIX for few covariates. But as the number of covariates increases, PROC HPMIXED rapidly becomes relatively inefficient as the size of the dense fixed-effects matrix increases.

Mixed Model Analysis of Microarray Data

DNA microarrays are a new technology to identify thousands of differentially expressed genes simultaneously. A microarray is the glass slide that has been spotted or "arrayed" with DNA fragments that represent specific genes. The intensity of fluorescence in a spot represents the gene response. These gene responses indicate the associations with disease conditions. The systematic biases that result from experimental factors such as treatments, spot location, arrays, dyes, and various interactions of these effects violate the basic assumption that the experimental subjects are similar across levels of the treatment variables.

Several statistical approaches for identifying differentially expressed genes while adjusting for these systematic biases have been developed. One popular approach is the mixed model approach developed by Wolfinger et al. (2001). This normalization method uses linear mixed models to normalize and then analyze data. The normalization model attempts to correct the systematic biases across all genes. The residuals from the normalization model are then passed to gene-specific models. Notice that the second step is a by-gene analysis. The gene model is analyzed one gene at time. This significantly reduces the computer memory requirement. However, this two-step method is based on the assumption that the systematic biases are independent from the gene effects. When this assumption is untenable, several researchers (Kerr, Martin, and Churchill 2000; Churchill 2002; Littell et al. 2006) propose a single mixed model analysis that combines both the systematic biases and the gene effects. The following linear mixed model is usually considered:

$$
\begin{aligned}
Y_{ijkmnr} \quad = \quad & & \textbf{Fixed Effects} \\
& \mu & \text{Overall mean} \\
+ \quad & \lambda_i & \text{Gene} \\
+ \quad & \tau_j & \text{Treatment} \\
+ \quad & \delta_k & \text{Dye} \\
+ \quad & (\tau\lambda)_{ij} & \text{Treatment-by-gene} \\
+ \quad & (\delta\lambda)_{ik} & \text{Dye-by-gene} \\
& & \\
& & \textbf{Random Effects} \\
+ \quad & a_m & \text{Microarray} \\
+ \quad & (a\lambda)_{im} & \text{Microarray-by-gene} \\
+ \quad & e_{ijkmnr} & \text{Residual noise}
\end{aligned}
$$

This single mixed model method leads to a huge mixed model because thousands or tens of thousands of genes might be in the analysis. The HPMIXED procedure is well suited for this approach.

This example uses a two-channel microarray data publicly available at

http://www.bch.msu.edu/~zacharet/publications/supplementary/ee_dr

to demonstrate the possible usage of PROC HPMIXED in microarray analysis. This data has more than 1.2 million records and 6,769 genes.

The following is a SAS program for the analysis:

```
proc hpmixed data=microarray;
   class marray dye trt gene;
   model log2i = dye trt gene dye*gene trt*gene/ ddfm=residual;
   random marray marray*gene;
run;
```

The "Dimensions" table shown in Figure 4 indicates that this is a very large model. It has 62,830 columns in the **X** matrix and 75,396 columns in the **Z** matrix. The dimension of the resulting MME is 138,226 by 138,226. The upper triangle of the MME has more than 9.5 billion elements among which there are only 1.8 million nonzero elements. It would be computationally very inefficient to fit this model by using dense matrix methods; the sparse matrix approach of the HPMIXED procedure is of critical importance.

Figure 4 Mixed Model Dimensions

Dimensions	
G-side Cov. Parameters	2
R-dide Cov. Parameters	1
Columns in X	62830
Colums in Z	75396
Subjects (Blocks in V)	1

The preceding SAS program produces only the parameter estimates without any Type III tests. It takes seven hours to finish in a desktop with 2GB RAM. Running PROC MIXED with the same model and the same data with the NOTEST option would require approximately 150GB of RAM and would take something on the order of five months on a server-class PC.

As the preceding discussion indicates, for genomic applications where only parameter estimates and BLUPs are required, the HPMIXED procedure makes it possible to analyze many more genes via the single mixed model approach. However, there are other types of post-fit analyses for which the HPMIXED procedure still has limitations. For example, Type III tests, least squares means, and tests of contrasts all still require the HPMIXED procedure to construct and manipulate dense forms of the MME, and this leads to limitations in how many genes can be analyzed using these methods. For example, if you want to perform a Type III test of treatment effect on the preceding data, you are still out of luck; with a server-class PC, the HPMIXED procedure runs out of memory for Type III tests with much more than

1,000 genes. Still, it runs much faster than PROC MIXED for that number of genes. With 1,000 genes, HPMIXED takes about a half-hour to fit the model and test the treatment effect, but PROC MIXED would require about a day and a half.

FUTURE WORK

Currently, the HPMIXED procedure enables mixed modeling for only a small but important subset of the variance types available in the MIXED and GLIMMIX procedures. One focus of future work for the procedure will be to extend these types. Among the types offered by the GLIMMIX and MIXED procedures, the two most urgent and best suited to PROC HPMIXED's high-performance techniques are compound symmetry (CS) and first-order autoregressive (AR(1)). Also, a general pedigree relationship variance type (Henderson 1984) is useful for mixed model applications that involve animals or plant species, where the model includes components of correlation between organisms that are related by genetic inheritance.

As seen in the EXAMPLES section, although the HPMIXED procedure is efficient for fitting the model, it still has a performance bottleneck when it comes to post-fit analysis—primarily, Type III tests. This is because the calculation of the Type III $\mathbf{L}$ matrices for testing $H_0 : \mathbf{L}'\beta = 0$ still requires the dense form of the MME. It should be possible to reformulate this calculation to use the sparse form of the MME. Such a reformulation can be expected to dramatically reduce memory requirements and run time.

Another direction for future work on PROC HPMIXED will be to further expand the already large models that the procedure can handle. In certain scientific applications, researchers need to solve the MME just once for given parameter values, but the size of the MME is in the millions or even tens of millions. For example, to evaluate breeding of dairy cattle, the national genetic evaluation program (VanRaden et al. 2004) solves the MME to compute the best linear unbiased prediction (BLUP) to rank animals based on future genetic merit. Really huge mixed model problems such as this call for techniques beyond sparse matrix methods, such as computing BLUPs by using the conjugate gradient algorithm (Shewchuk 1994) or the iteration-on-data technique (Tsuruta, Misztal, and Stranden 2001).

CONCLUSION

The HPMIXED procedure is specifically designed to cope with mixed model estimation problems that involve fixed or random effects with many levels, or that have a large number of observations. It uses a number of specialized high-performance techniques, including sparse matrix technology and optimization algorithms that are tailored to very large systems. The HPMIXED procedure complements the MIXED procedure and other SAS/STAT procedures for mixed modeling by supporting a subset of their models but enabling considerably better performance for that subset in terms of memory requirements and computational speed for large problems.

When should you use the HPMIXED procedure? In general, you should consider it when some effects (either fixed or random) have very many levels; on present-day (2009) desktops, "very many" is about a thousand or more. It is difficult to diagnose the procedure's potential utility more precisely, but in general if you are analyzing a mixed model with PROC MIXED and you find that it either runs out of memory or takes too long to complete, try using PROC HPMIXED instead.

REFERENCES

Churchill, G. A. (2002), "Fundamentals of Experimental Design for cDNA Microarray," *Nature Genetics,* 32, 490–495.

George, J. A. and Liu, J. W. (1981), *Computer Solutions of Large Sparse Positive Definite Systems,* Englewood Cliffs, NJ: Prentice-Hall.

Graybill, F. A. (1976), *Theory and Applications of the Linear Model,* North Scituate, MA: Duxbury Press.

Harville, D. A. (1977), "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems," *Journal of the American Statistical Association*, 72, 320–338.

Henderson, C. R. (1984), *Applications of Linear Models in Animal Breeding*, University of Guelph.

Johnson, D. L. and Thompson, R. (1995), "Restricted Maximum Likelihood Estimation of Variance Components for Univariate Animal Models Using Sparse Matrix Techniques and Average Information," *Journal of Dairy Science,* 78, 449–456.

Kenward, M. G. and Roger, J. H. (1997), "Small Sample Inference for Fixed Effects from Restricted Maximum Likelihood," *Biometrics,* 53, 983–997.

Kerr, M. K., Martin, M., and Churchill, G. A. (2000), "Analysis of Variance for Gene Expression Microarray Data," *Journal of Computational Biology,* 7, 819–837.

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. (2006), *SAS for Mixed Models,* Second Edition, Cary, NC: SAS Press.

Searle, S. R., Casella, G., and McCulloch, C. E. (1992), *Variance Components,* New York: John Wiley & Sons.

Shewchuk, J. R. (1994), *An Introduction to the Conjugate Gradient Method without the Agonizing Pain,* Technical Report, Carnegie Mellon University, Pittsburgh, PA.

Tsuruta, S., I. Misztal, and I. Stranden (2001) "Use of the Preconditioned Conjugate Gradient Algorithm as a Generic Solver for Mixed-Model Equations in Animal Breeding Applications," *Journal of Animal Science,* 79, 1166–1172.

VanRaden, P. M., Sanders, A. H., Tooker, M. E., Miller, R. H., Norman, H. D., Kuhn, M. T., and Wiggans, G. R. (2004), "Development of a National Genetic Evaluation for Cow Fertility," *Journal of Dairy Science,* 87, 2285–2292.

Wolfinger, R. D., Gibson, G., Wolfinger, E., Bennett, L., Hamadeh, H., Bushel, P., Afshari, C., and Paules, R. S. (2001), "Assessing Genes Significance from cDNA Microarray Expression Data via Mixed Models," *Journal of Computational Biology,* 8, 625–637.

APPENDIX

The following SAS program defines and runs a macro to perform a time trial for the HPMIXED, MIXED, and GLIMMIX procedures on the animal breeding example discussed in the section "Exploring Relative Performance of Mixed Modeling Procedures" on page 7. The macro measures how much time each procedure takes as the number of farms in the trial is increased. Each procedure is run until its run time exceeds a specified maximum value *TopTime*. Note that the trial can take a long time to run. For this reason, a quick line-printer plot is printed to the listing for each number of farms; if you run the program in the interactive SAS display manager, you can track its progress using these plots. The final product of the macro is a plot of the collected run times for each number of farms and each procedure, along with a LOESS smooth to help you visualize the relationship. See "Exploring Relative Performance of Mixed Modeling Procedures" on page 7 for examples of running this macro.

```
%Macro TimeTrial(TopTime);

   /*
   /  Set up loop over the number of farms.
   /-------------------------------------------------------------*/
   %let NFarmStart   = 1;
   %let NFarmBy      = 1;

   /*
   /  As the run-time for each procedure exceeds the value of
   /  TopTime, it will be dropped from the trial.  At the start,
   /  all three procedures are to be run.
   /-------------------------------------------------------------*/
   %let RunHPMIXED   = 1;
   %let RunGLIMMIX   = 1;
   %let RunMIXED     = 1;

   /*
   /  Loop while at least one procedure's run-time has not exceeded
   /  TopTime.
   /-------------------------------------------------------------*/
   data AllTimes; if (0); run;
   %let NFarm = &NFarmStart;
   %do %while (&RunHPMIXED | &RunMIXED | &RunGLIMMIX);

      %let NAnimal = %eval(&NFarm*100);

      /*
      /  Generate data for this number of farms.
```

```
/*-----------------------------------------------------*/
data Sim;
   keep Species Farm Animal Yield;
   array AnimalEffect{%eval(&NAnimal)};
   array AnimalSpecies{&NAnimal};
   array AnimalFarm{&NAnimal};
   do i = 1 to &NAnimal;
      AnimalEffect {i} = sqrt(4.0)*rannor(1);
      AnimalSpecies{i} = 1 + int(   5  *ranuni(12345));
      AnimalFarm   {i} = 1 + int(&NFarm*ranuni(12345));
      end;
   do i = 1 to %eval(40*&NAnimal);
      Animal  = 1 + int(&NAnimal*ranuni(1));
      Species = AnimalSpecies{Animal};
      Farm    = AnimalFarm   {Animal};
      Yield   = 1 + Species
                  + Farm
                  + AnimalEffect{Animal}
                  + sqrt(8.0)*rannor(1);
      output;
   end;
run;

/*
/  Collect run-times for each procedure.
/-----------------------------------------------------*/
data Time; NFarm = &NFarm; run;

%if (&RunHPMIXED) %then %do;
   %let Start = %sysfunc(datetime());
   ods listing close;
   proc hpmixed data=Sim;
      class Species Farm Animal;
      model Yield = Species Farm*Species;
      random Animal/cl;
      ods output SolutionR=p;
   run;
   ods listing;
   %let Stop = %sysfunc(datetime());
   data Time; merge Time; HPMIXED = &Stop - &Start; run;
   %end;

%if (&RunMIXED) %then %do;
   %let Start = %sysfunc(datetime());
   ods listing close;
   proc mixed data=Sim;
      class Species Farm Animal;
      model Yield = Species Farm*Species
                    / ddfm=residual notest;
      random int / subject = Animal cl;
      ods output SolutionR=p;
   run;
   ods listing;
   %let Stop = %sysfunc(datetime());
   data Time; merge Time; MIXED = &Stop - &Start; run;
   %end;

%if (&RunGLIMMIX) %then %do;
   %let Start = %sysfunc(datetime());
   ods listing close;
   proc glimmix data=Sim;
      class Species Farm Animal;
      model Yield = Species Farm*Species / ddfm=residual;
      random int / subject = Animal cl;
      ods output SolutionR=p;
   run;
   ods listing;
   %let Stop = %sysfunc(datetime());
   data Time; merge Time; GLIMMIX = &Stop - &Start; run;
```

```
      %end;

   data AllTimes; set AllTimes Time;
   run;

   /*
   /  Since the trial can take a long time to run, produce a
   /  quick plot to the listing as we go to say where we are in
   /  the process.
   /-----------------------------------------------------------*/
   proc summary data=AllTimes;
      by NFarm;
      var HPMIXED MIXED GLIMMIX;
      output out=sAllTimes mean = HPMIXED MIXED GLIMMIX;
   data Plot; set sAllTimes;
      if (HPMIXED ^= .) then HPMIXED = min(HPMIXED,&TopTime);
      if (GLIMMIX ^= .) then GLIMMIX = min(GLIMMIX,&TopTime);
      if (MIXED   ^= .) then MIXED   = min(MIXED  ,&TopTime);
   proc plot data=Plot;
      title1 "NFarm = &NFarm";
      plot HPMIXED*NFarm = "H"
           MIXED  *NFarm = "M"
           GLIMMIX*NFarm = "G" / overlay;
   run;
   title1;

   /*
   /  If the median run-time for a PROC over the last three
   /  NFarm's is bigger than TopTime, then drop this procedure
   /  from the rest of the trial.
   /-----------------------------------------------------------*/
   proc summary data=AllTimes;
      where (NFarm >= &NFarm - 3*&NFarmBy);
      var HPMIXED MIXED GLIMMIX;
      output out=LastTimes median = HPMIXED MIXED GLIMMIX;
   data _null_; set LastTimes;
      if (HPMIXED > &TopTime) then call symput('RunHPMIXED','0');
      if (GLIMMIX > &TopTime) then call symput('RunGLIMMIX','0');
      if (MIXED   > &TopTime) then call symput('RunMIXED'  ,'0');
   run;

   %let NFarm = %eval(&NFarm + &NFarmBy);
   %end;

/*
/  Plot the run-times, along with LOESS smooths.
/-------------------------------------------------------------*/
proc transpose data= AllTimes
               out =tAllTimes(rename=(COL1=Time _NAME_=PROC));
   by NFarm;
   var HPMIXED MIXED GLIMMIX;
data tAllTImes; set tAllTimes;
   label Time  = "PROC run-time";
   label PROC  = "PROC";
   label NFarm = "Number of farms";
proc sort data=tAllTimes;
   by PROC NFarm;
proc sgplot data=tAllTimes;
   yaxis max=&TopTime;
   loess y=Time x=NFarm
         / group=PROC interpolation=cubic smooth=0.5;
run;
%mend;

options nonotes;
%TimeTrial(10);
```

CONTACT INFORMATION

Tianlin Wang
SAS Institute Inc.
500 SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-1312
E-mail: Tianlin.Wang@sas.com
Web: http://www.sas.com