Paper Number SA-15

Tips and Strategies for Mixed Modeling with SAS/STAT® Procedures

Kathleen Kiernan, Jill Tao, and Phil Gibbs, SAS Institute Inc., Cary, NC, USA

ABSTRACT

Inherently, mixed modeling with SAS/STAT® procedures, such as GLIMMIX, MIXED, and NLMIXED is computationally intensive. Therefore, considerable memory and CPU time can be required. The default algorithms in these procedures might fail to converge for some data sets and models. This paper provides recommendations for circumventing memory problems and reducing execution times for your mixed modeling analyses. This paper also shows how the new HPMIXED procedure can be beneficial for certain situations, as with large sparse mixed models. Lastly, the discussion focuses on the best way to interpret and address common notes, warnings, and error messages that can occur with the estimation of mixed models in SAS software.

INTRODUCTION

Over the past 20 years, mixed-modeling methodology has expanded to many areas of statistical applications. Initially, the mixed-model capabilities in the SAS System depended on the MIXED procedure. Subsequently, the NLMIXED, HPMIXED, and GLIMMIX procedures were added. SAS/STAT software is a fully integrated component of the SAS System. PROC GLIMMIX and PROC MIXED are two of the most popular procedures in SAS/STAT software that fit mixed models. Most of the questions that SAS customers have about any of these mixed-modeling procedures can be categorized into the following areas:

- providing recommendations for improving performance
- providing methods for obtaining convergence
- providing explanations of various notes, warnings, or error messages in the SAS log

This paper addresses these specific areas. It also provides recommendations for standard practices that have shown significant benefit when using PROC GLIMMIX, PROC MIXED, and PROC NLMIXED.

There are three sections in this paper. The first section provides tips on how to make programs more efficient by reducing memory and execution time. The second section provides suggestions for troubleshooting convergence problems. The last section includes a brief discussion of some of the commonly reported notes, warnings, and errors that are reported in the SAS log for a mixed model analysis using PROC GLIMMIX, PROC MIXED, or PROC NLMIXED.

SECTION 1: IMPROVING PERFORMANCE

The GLIMMIX, MIXED, and NLMIXED procedures are computationally intensive, and execution times can be long. A model might be resource intensive (requiring a large amount of memory or time) for a variety of reasons:

- The input data set is large.
- There are many levels associated with the variables in the CLASS statement that might subsequently be used in the MODEL, RANDOM, or REPEATED statements.
- The model is complex.
- Certain options are specified in the PROC, MODEL, RANDOM, or REPEATED statements.

If you have a model that encounters an out of memory error or takes too long to run, the following suggestions might be helpful.

MAKE CHANGES TO THE RUNNING ENVIRONMENT

The following changes to the running environment are good practices to implement:

- To maximize available memory on your system, close all unnecessary applications when running your program.
- If your program generates a large number of results tables, use the ODS NORESULTS statement to prevent the tracking of output objects in the Results window. For example, this suggestion is useful when using BY processing with a large number of BY groups or when using a macro to run the procedure many times.

- Submit programs in batch mode rather than interactively.
- In UNIX environments, do not set the MEMSIZE value to `0`. Instead, specify a reasonable value that is less than the UNIX server's physical memory.

USE EFFICIENT CODING TECHNIQUES

Some mixed models can be expressed in different but mathematically equivalent ways with either PROC GLIMMIX or PROC MIXED statements. Alternative specifications of statements lead to equivalent statistical models, but the data processing and estimation phase can be quite different, depending on the syntax of your program statements and options. For example, using the SUBJECT= option in the RANDOM or REPEATED statement affects data processing and estimation. Also keep in mind that certain options, such as the EMPIRICAL option in the PROC MIXED statement, are available only when the data are processed by subject.

PROCESS BY SUBJECTS

When one or more random effects has many levels, for example, 1000 or more, the computations can become resource intensive. For example, you submit the following code:

```
proc mixed;
   class a b;
   model y=a;
   random b;
run;
```

This code might take a long time to run or might result in an out of memory error if variable *B* has many levels and you use the default option DDFM=CONTAINMENT to compute the denominator degrees of freedom.

Below are some alternative specifications of the model that are statistically equivalent but numerically more efficient.

Specification of a SUBJECT= Effect

Using the SUBJECT= option enables the procedure to process the model by subjects. This typically takes less time and memory. Here is an example:

```
proc mixed;
   class a b;
   model y=a;
   random intercept / subject=b;
run;
```

If variable *B* is a numeric variable, or if it can be easily recoded as a numeric variable, then you can further improve the efficiency of the preceding model. You can sort your data by the SUBJECT= effect *B* and remove *B* from the CLASS statement. Here is an example:

```
proc sort;
   by b;
run;

proc mixed;
   class a;
   model y=a;
   random intercept / subject=b;
run;
```

Alternatively, for a random intercept model, the equivalent model can be specified using the REPEATED statement rather than the RANDOM statement. The REPEATED statement is less memory intensive than the RANDOM statement, especially when there are many levels of the SUBJECT= effect. In PROC GLIMMIX, you specify a repeated structure by adding the _RESIDUAL_ or RSIDE keyword to the RANDOM statement. Here is an example:

```
random _residual_ /subject=b type=cs;
```

That being said, you can rewrite the random intercept model shown previously using an equivalent REPEATED statement in PROC MIXED as follows:

```
proc mixed;
   class a b;
   model y=a;
   repeated / subject=b type=cs;
run;
```

Here is an example using PROC GLIMMIX:

```
proc glimmix;
   class a b;
   model y=a/dist=normal;
   random _residual_ / subject=b type=cs;
run;
```

Similar to using the SUBJECT= option in the RANDOM statement, you can further improve the efficiency of the preceding example by sorting your data by the SUBJECT= effect *B* and removing *B* from the CLASS statement as follows:

```
proc sort;
   by b;
run;

proc mixed;
   class a;
   model y=a;
   repeated / subject=b type=cs;
run;
```

Here is the equivalent coding using PROC GLIMMIX:

```
proc glimmix;
   class a;
   model y=a/dist=normal;
   random _residual_ / subject=b type=cs;
run;
```

Note that the R-side random effect with TYPE=CS only is equivalent to the random intercept model when the distribution is normal and the **G** matrix is positive definite.

If the SUBJECT= variable is a numeric variable, you can improve the performance of a repeated measures analysis in PROC MIXED or PROC GLIMMIX by sorting the data by the SUBJECT= effect and removing it from the CLASS statement.

If you have more than one random effect, and if there is a common effect in all the effects appearing in the RANDOM statement, you can "factor out" that common effect and specify it as the SUBJECT= effect. This creates a block-diagonal **G** matrix and enables PROC MIXED and PROC GLIMMIX to process the model by subjects. This approach is typically faster and requires less memory. For example, consider the following GLIMMIX step:

```
proc glimmix;
   class a b c;
   model y=a b / ddfm=satterth;
   random c a*c b*c;
run;
```

You can improve the efficiency of this analysis. Because variable *C* appears in all effects in the first RANDOM statement, it can be factored out and used as the SUBJECT= effect in the second RANDOM statement as follows:

```
proc glimmix;
   class a b c;
   model y=a b / ddfm=satterth;
   random int a b/subject=c;
run;
```

The data processing and estimation in the MIXED or GLIMMIX procedure is a little more complicated when you have multiple RANDOM statements. Both procedures will process the model by subjects if each RANDOM statement has a SUBJECT= effect and if the SUBJECT= effects are nested within each other. If possible, use the same nested

SUBJECT= effects in all RANDOM and REPEATED statements in PROC MIXED. The equivalent specification using the same nested effects also applies to PROC GLIMMIX with RANDOM _RESIDUAL_ statements. The following example shows a nested SUBJECT= effect for a hierarchical linear model (HLM) in PROC GLIMMIX:

```
proc glimmix data=hlm3;
   class doctor hospital patient;
   model y(event="1") = x1 x2 / dist=binary link=logit solution;
   random int x1 x2 / solution type=un subject=hospital;
   random int x1 x2 / solution type=un  subject=doctor(hospital);
   random int x1 x2 / solution type=un   subject=patient(doctor hospital);
run;
```

For more information, see "Processing by Subjects" in the "Details" section of the *SAS/STAT® 9.3 User's Guide: The GLIMMIX Procedure* (SAS Institute Inc. 2012).

Specification of a Nested SUBJECT= Effect

The next example demonstrates how to recode the input data set to take advantage of the nested SUBJECT= effects so you can estimate large HLM models in a reasonable amount of time.

For example, the data below for a 3-level HLM model shows that students are nested within schools and schools are nested within districts. *x1* represents a covariate, and *y* represents the dependent variable. A partial data set is shown below.

district	uschool	x1	y
1	1	19.4	32.4
1	1	19.3	28.3
1	1	19.2	31.9
1	2	18.8	18.9
1	2	18.4	30.3
1	2	18.2	21.3
1	3	19.7	24.8
1	3	20.4	38.6
1	3	20.2	27.4
1	4	19.3	36.1
1	4	21.7	32.4
1	4	21.5	35.6
2	5	21.5	31.3
2	5	18.7	28.5
2	5	19.9	20.1
2	6	18.9	31.8
2	6	18.7	25.1
2	6	19.5	32.9
2	7	18.7	18.3
2	7	19.4	27.4
2	7	19.5	26.2
2	8	20.9	28.6
2	8	21.1	29.9
2	8	20.0	36.4

Here is a possible model for these data:

```
proc mixed data=test;
   class district uschool;
   model y=x1 / ddfm=bw;
   random int / subject=district;
   random int / subject=uschool(district);
run;
```

This PROC MIXED step represents the program statements that are used to model a typical hierarchical linear model. When you use the SUBJECT= effect and specify *USCHOOL* nested within *DISTRICT*, the model is able to process by subjects and use resources more efficiently. However, note that in these data, each school has a unique value of *USCHOOL*. Therefore, the *USCHOOL* variable is not "truly" nested within *DISTRICT*. You can further improve the efficiency of this model by "recoding" each school so it is truly nested within a district. For example, within district 1 there are schools 1, 2, 3, and 4. For district 2, renumber schools to be 1, 2, 3, and 4, as well (instead of 5, 6,

7, and 8). Therefore, a school can be uniquely identified by a combination of DISTRICT and SCHOOL values. Note that school 1 within district 1 is different from school 1 within district 2.

The following sample program shows how to change the coding for *USCHOOL* with a new variable *SCHOOL*:

```
proc sort data=test;
   by district uschool;
run;

data test2;
   set test;
   by district uschool;
      if first.district then school=0;
      if first.uschool then school+1;
run;

proc print data=test2;
   var district uschool school x1 y;
run;
```

The new data set shows observations for the first two districts:

district	uschool	school	x1	y
1	1	1	19.4	32.4
1	1	1	19.3	28.3
1	1	1	19.2	31.9
1	2	2	18.8	18.9
1	2	2	18.4	30.3
1	2	2	18.2	21.3
1	3	3	19.7	24.8
1	3	3	20.4	38.6
1	3	3	20.2	27.4
1	4	4	19.3	36.1
1	4	4	21.7	32.4
1	4	4	21.5	35.6
2	5	1	20.5	29.0
2	5	1	20.5	35.4
2	5	1	21.5	31.3
2	6	2	19.2	27.3
2	6	2	20.2	37.8
2	6	2	18.9	31.8
2	7	3	18.7	18.3
2	7	3	19.4	27.4
2	7	3	19.5	26.2
2	8	4	19.1	33.2
2	8	4	17.9	35.8
2	8	4	20.9	28.6

The same model with this new variable *SCHOOL* processes much faster and requires less memory:

```
proc mixed data=test2;
   class district school;
   model y=x1 / ddfm=bw;
   random int / subject=district;
   random int / subject=school(district);
run;
```

SAS Note 37057 “How large of a hierarchical linear model (HLM) can PROC MIXED fit?” (SAS Institute Inc. 2009) discusses related information about efficient specification of HLMs.

CHOOSE OPTIONS THAT CAN IMPROVE EFFICIENCY

Certain options can be more resource intensive than other options. Being flexible to using other options or specifications can improve performance of mixed models. Here are some examples:

- The DDFM= option in the MODEL statement specifies the estimation method for the denominator degrees of freedom, some of which can be resource intensive. Using the DDFM=RESIDUAL or DDFM=BW option in the MODEL statement can reduce the amount of memory that is required to process your model. The DDFM=KR option is more memory intensive than the DDFM=SATTERTH option. DDFM=SATTERTH and DDFM=BW can be faster than DDFM=CONTAINMENT. If you have a random effect with many levels, use the DDFM=BW option in the MODEL statement to eliminate the time that is required for the default DDFM=CONTAINMENT method. For large models in PROC MIXED, the memory requirement can be exponentially reduced by using the DDFM= RESIDUAL and NOTEST options in the MODEL statement.
- The different types of covariance structures that are available in the REPEATED and the RANDOM statements can affect the memory requirements for estimating a model. Generally speaking, more complex covariance structures, such as TYPE=UN or the spatial structures, are more resource intensive than the simpler covariance structures. You might want to use a simpler covariance structure to reduce the memory or time requirements. However, you should check that the simpler TYPE= structure is supported by the data or expert knowledge. In particular, you might have to consider the following alternatives:
  - The TYPE=UN option in the REPEATED statement in PROC MIXED or the RANDOM _RESIDUAL_ statement in PROC GLIMMIX might not be appropriate if subjects have many repeated measurements. The TYPE=UN covariance structure requires estimation of a large number of parameters and might require excessive amounts of memory. For example, if a subject has 20 repeated measures, TYPE=UN estimates (20 * 21)/2 = 210 variance-covariance parameters, and that might be too many. You might want to consider simpler alternative structures such as TYPE=TOEP.
  - The TYPE=UN option in the RANDOM statement, as shown below, is the common structure for a random coefficients or hierarchical linear model in PROC MIXED or in PROC GLIMMIX.

```
proc mixed;
   class trt;
   model y=trt x1 x2 x3 x4;
   random intercept x1 x2 x3 x4/subject=site type=un;
run;
```

    However, TYPE=UN might not be appropriate for random coefficients or hierarchical models that have many covariates in the RANDOM statement. Again, the TYPE=UN structure requires estimation of a large number of parameters and might require excessive memory. Consider the simpler TYPE=VC structure.
- Keep in mind that if you are estimating a random intercept model (one random effect), then TYPE=UN is not necessary. Changing TYPE=UN to TYPE=VC, as shown below, should give the same model without potential problems:

```
proc mixed;
   class trt;
   model y=trt;
   random intercept/subject=site type=vc;
run;
```

- The GROUP=*effec*t option in the RANDOM or the REPEATED statement will estimate a covariance parameter for each unique level of the GROUP=*effect* option and requires more computational resources in both memory and time.
- Using the NOBOUND option in the PROC MIXED or PARMS statement requires more resources due to changes in the computational algorithm when there are negative variance components.
- The NOCLPRINT option in the PROC MIXED or GLIMMIX statement suppresses the CLASS Level Information table and can help reduce memory and execution time.
- If you add the NOTEST option in MODEL statement in PROC MIXED, the running time could be exponentially reduced for some models.
- Consider fitting your model without any of these options first:
  - ODS graphics
  - the SOLUTION options in the MODEL and RANDOM statements in either PROC MIXED or PROC GLIMMIX
  - the OUTP=, OUTPM=, and the INFLUENCE options in the MODEL statement in PROC MIXED
  - the OUTPUT statement in PROC GLIMMIX

- In MIXED and GLIMMIX you can specify known values for the covariance parameters by using the HOLD= or NOITER option in the PARMS statement or the GDATA= option in the RANDOM statement in PROC MIXED. This eliminates the need for iteration.
- In the GLIMMIX and NLMIXED procedures, when the number of random-effect levels is large, the amount of computation that is required in the adaptive quadrature approximation can be overwhelming. Reducing the QPOINTS does reduce the amount of computation; it also reduces the quality of the approximation to the marginal likelihood. Too few quadrature points might lead to unreliable results for certain likelihood.

For nonlinear mixed models, consider using the following options in the PROC NLMIXED statement to improve the computational speed:

- QPOINTS=1—this option invokes the Laplace approximation and typically makes PROC NLMIXED run much faster.
- METHOD=FIRO—this option uses the first-order method to approximate the integral of the likelihood over the random effects. It is fast and might be appropriate when the conditional distribution of the response variable is normal.

CHOOSE AN ALTERNATIVE PROCEDURE

For linear mixed models with thousands of levels for the fixed or random effects, or for linear mixed models with hierarchically nested fixed or random effects with hundreds or thousands of levels at each level of the hierarchy, you can use PROC HPMIXED rather than PROC MIXED.

The HPMIXED procedure is specifically designed to cope with estimation problems that involve a large number of fixed effects, a large number of random effects, or a large number of observations. The HPMIXED procedure handles only a subset of the analyses of the MIXED procedure. However, you can use the HPMIXED procedure to accelerate your MIXED procedure analyses for large problems. The idea is to use PROC HPMIXED to maximize the likelihood and produce parameter estimates more quickly than just using PROC MIXED. You can then pass these parameter estimates to PROC MIXED for further analysis that is not available within PROC HPMIXED, as discussed next in the analysis of cross-classified models.

Cross-classified models are commonly estimated using either PROC MIXED or PROC GLIMMIX. Cross-classified models are real resource hogs because the typical syntax includes both SUBJECT= effects in the CLASS statement. In addition, the subject effects are not nested, thereby, preventing the model from processing by subjects. One approach to improve efficiency is to sort your data by the subject effect that has the most levels and remove it from the CLASS statement, provided the SUBJECT=*effect* is a numeric variable. Leave everything else in your code alone. If this approach does not improve efficiency, you can use PROC HPMIXED to obtain the estimates for the covariance parameters, and then use a subsequent PROC MIXED step using those particular covariance parameters.

The following example demonstrates the code for a cross-classified model using PROC HPMIXED with a subsequent PROC MIXED step. The following HPMIXED code demonstrates computing the same cross-classified model as using PROC MIXED. You can use the ODS OUTPUT COVPARMS= data set from PROC HPMIXED to accelerate your MIXED or GLIMMIX procedure analyses for large problems. Then you pass these covariance parameter estimates to PROC GLIMMIX and PROC MIXED for some further analysis that is not available within PROC HPMIXED. Here is the first step:

```
      /* Output the covariance parameters to an ODS output data set. */
   proc hpmixed data=pupcross noclprint;
      class ethnicity pschool sschool;
      model achiev = ethnicity pupsex pupses  /solution;
      test pupsex pupses;
      random intercept / subject=sschool;
      random intercept / subject=pschool;
      lsmeans ethnicity;
      ods output covparms=covpe;
   run;
```

PROC HPMIXED does not support the COVTEST option or provide Tukey's multiple comparisons tests for the LSMEANS statement. Therefore, the second part of this cross-classified example shows the syntax of the PARMS statement that is using the covariance parameters from PROC HPMIXED and requests the COVTEST option to obtain tests on the covariance parameter estimates and Tukey's multiple comparison tests for the LSMEANS:

```
proc mixed data=pupcross noclprint covtest;
   class ethnicity pschool sschool;
   model achiev = ethnicity pupsex pupses  / solution ;
   random intercept / subject=sschool;
   random intercept / subject=pschool;
   parms/pdata=covpe noiter;
   lsmeans ethnicity/adjust=tukey;
run;
```

Furthermore, consider using item stores and PROC PLM to separate common post-processing tasks, such as testing for treatment differences under a fitted model. With this approach, a numerically expensive model-fitting technique can be applied once to produce a source item store. The PLM procedure can then be called multiple times and the results of the fitted model can be analyzed without incurring the model fitting expenditure again. Here is an example:

```
proc glimmix data=multicenter;
   class center group;
   model sideeffect/n = group / solution;
   random intercept / subject=center;
   store gmx;
run;

proc plm source=gmx;
   lsmeans group / cl ilink;
run;
```

Using item stores and PROC PLM enables you to perform separate common post-processing tasks, CONTRASTS, ESTIMATES, LSMEANS, and LSMESTIMATES for a mixed-model analysis.

SECTION II: TROUBLE SHOOTING CONVERGENCE FAILURES IN MIXED MODELS

If you have practical experience estimating mixed models, then you have occasionally encountered problems with convergence. Such problems are both puzzling and exasperating. Most researchers do not know why certain models and certain data sets lead to convergence difficulties. And for those who do understand the causes of the problem, it is often unclear whether and how the problem can be fixed. This section provides insight on why numerical algorithms for estimation of the mixed models sometimes fail to converge. A number of possible circumventions also are offered for consideration.

If you experience convergence problems, consider these points:

- Always examine the iteration history to see whether progress is being made toward convergence. It is not rare that a convergence failure is reported when the optimization actually converges. This might happen when the default convergence criteria are too tight for the problem. Therefore, you might need to set your own criteria to obtain convergence. Sometimes the parameter estimates have a small magnitude and the default relative convergence criteria cannot be satisfied. Therefore, you might try to use absolute convergence criteria to obtain convergence.
- Try simpler model first. If you are fitting a complex model, it always helps to reduce the complexity of the model. It can help you narrow down the issues too.
- Try different TYPE= covariance structures that are supported by the data or expert knowledge.
- Verify that the model is not misspecified or overspecified. A misspecified or overspecified model is one of the common causes of convergence issues. Always check your model specifications.
- Use the PARMS statement. The PARMS statement lets you input initial values for the covariance parameters and performs a grid search over the likelihood surface. Sometimes the default starting values might not work for your data, and using the PARMS statement to specify a different set of starting values can help when using the GLIMMIX, MIXED, and NLMIXED procedures. The PARMS statement is particularly useful for PROC NLMIXED, where the default starting value is 1 for all parameters. It is important to specify your own reasonable starting values in PROC NLMIXED.
- Rescale the effects in the model. Sometimes the Newton-Raphson algorithm does not perform well when two of the covariance parameters are on a different scale—that is, when they are several orders of magnitude apart. This is because the Hessian matrix is processed jointly for the two parameters, and elements of it corresponding to one of the parameters can become close to internal tolerances in PROC MIXED. In this case, you can improve stability by rescaling the effects in the model so that the covariance parameters are on the same scale.

- Rescale the data values. Data that are extremely large or extremely small can adversely affect results because of the internal tolerances in PROC MIXED. Rescaling the data can improve stability.
- Use a different METHOD= option, such as LAPLACE, in PROC GLIMMIX, if possible.
- Try various options to tune the convergence and other aspects of the optimization process.

  In PROC GLIMMIX, you might want to try the following:

  - Try various options, such as the INITGLM option, in the PROC GLIMMIX statement to control the outer iterations.
  - Try various options, such as TECH=NRRIDG or TECH=NEWRAP, in NLOPTIONS statements. This is especially useful for Binary, Binomial, Negative Binomial, and Poisson distributions.
  - Increase the number of optimizations using the MAXOPT= option in the PROC GLIMMIX statement
  - Increase the number of iterations using the MAXITER= option in the NLOPTIONS statement in PROC GLIMMIX.

  In PROC MIXED, you might want to try the following:

  - Tune the MAXITER= and MAXFUNC= options in the PROC MIXED statement.
  - Use the NOPROFILE= or NOBOUND= options in the PROC MIXED statement.
  - Try the CONVF= or CONVG= option as a convergence criterion in the PROC MIXED statement.
- Use the following strategies for convergence issues encountered in PROC NLMIXED:
  - Sort the data set by the SUBJECT= effect. PROC NLMIXED does not have the CLASS statement. You must sort your data by the SUBJECT= effect first.
  - Verify that there are no spelling errors for your parameters or variables. If PROC NLMIXED cannot recognize a name because it is not in your data set, the procedure will treat it as a new parameter. This can cause convergence issues.
  - Verify that your parameters do not have the same name as one of the variables in your data set. If so, PROC NLMIXED will treat the parameter as the variable with the same name and use its values in your data set for the parameter. This can cause convergence issues.
  - Try different parameterizations for variance-covariance parameters. Section III includes examples of different parameterizations for nonlinear-mixed models.

SECTION III: UNDERSTANDING COMMON NOTES, WARNINGS, OR ERROR MESSAGES IN SAS LOGS FOR MIXED MODELS

This section of the paper describes some of the most common messages in SAS logs that have been reported to the Analytics team within the Technical Support Division of SAS. Some of these messages are formally documented in SAS notes, which are searchable on the SAS Customer Support Web site (support.sas.com).

Be aware that the exact cause for some of the various notes, warnings, and error messages might vary based on the data set, program statement, and procedure that are being used. It is also possible that the same issue might generate different notes, warnings, and error messages. Common strategies might circumvent various notes, warnings, and error messages. Therefore, do not be afraid to try several strategies.

NOTES

Notes in the log often are informative messages regarding the respective procedure. You cannot always tell if you need to make necessary changes in your program or if the note is just an informative bulletin. Some of the common notes that occur in the SAS log are listed below.

NOTE: Convergence criteria met.

NOTE: Estimated G matrix is not positive definite.

This note occurs when the value for any of the estimated Covariance Parameter Estimates is 0 as shown in the following table:

Covariance Parameter Estimates	
Cov Parm	**Estimate**
Lab	0.3175
Temp*Lab	0
Batch(Temp*Lab)	2.0737
Residual	0.6026

This note explaining that the estimated $\mathbf{G}$ matrix is not positive definite indicates that either procedure converged to a solution where the variance of the random effect is 0. Both PROC GLIMMIX and PROC MIXED apply a lower boundary of 0 to all variance components in the model. If the boundary was removed, the actual value of the variance component could become negative.

This note is telling you that there was not enough variation in the response to attribute any variation to the random effect, controlling for everything else in the model. Although there is nothing inherently wrong with the results when this note occurs, some statisticians would tell you to remove the corresponding random effect from the model.

Some statisticians would recommend that random effects are part of the model that has a component to it that is essential to the design structure. However, you should keep the term in the model even if it is 0. If the variance component is the block effect, you might keep it in the model because, in this way, the model reflects the design and data collection. Removing it does not make sense. One of the advantages of using PROC GLIMMIX or PROC MIXED is that you will get the same answers with or without the zeroed term in the RANDOM statement.

Often, when the estimated $\mathbf{G}$ matrix is not positive definite, the SAS log might also include any of the following additional notes:

NOTE: Asymptotic variance matrix of covariance parameter estimates has been found to be singular and a generalized inverse was used. Covariance parameters with zero variance do not contribute to degrees of freedom computed by DDFM=KENWARDROGER.

This note means that if the value for the estimate for the covariance parameter is 0 then the degrees for the fixed effects should remain unchanged if you remove the associated random effect from the model.

NOTE: Convergence criteria met but final hessian is not positive definite.

The reasons for why this note occurs can vary. The most common reasons for this note relates to scaling issues or misspecified or overspecified models.

A nonpositive definite Hessian matrix can indicate a surface saddlepoint or linear dependencies in the parameters. If the MIXED procedure has converged to a saddlepoint, then the final solution given by PROC MIXED is not the optimal solution. To get around this problem, run the model again using different starting values. Try adding a PARMS statement to your PROC MIXED code. Either use the OLS option to specify ordinary least squares starting values (versus the default MIVQUE0 values), or specify your own starting values in the PARMS statement. The syntax for the PARMS statement with the OLS option is simple, as shown below:

```
parms  / ols;
```

If you still get this note with new starting values, then there are probably linear dependencies in the parameters. This means that the model you want to run is too complex for your data to support. Try simplifying the covariance structure and model until you no longer receive this note.

Sometimes this note can be caused by a scaling issue. Try to rescale your data or reparameterize your model.

Similar notes can occur for PROC GLIMMIX and PROC NLMIXED. Similar strategies also apply to these procedures. In addition, with PROC GLIMMIX, you can add the INITGLM option in the PROC GLIMMIX statement to specify a different set of starting values. In PROC NLMIXED, you might double check your programs to ensure that there are no spelling errors. Detailed circumventions for PROC NLMIXED are discussed later in this section. ***In any case, if the final Hessian is not positive definite, subsequent results should be interpreted with caution.***

NOTE: An infinite likelihood is assumed in iteration 0 because of a nonpositive definite estimated R matrix for Subject x.

PROC MIXED requires that you have one observation per repeated effect per subject. This note occurs when your PROC MIXED step includes a REPEATED statement and the input data set has multiple observations at a given time

point for that particular subject. The following examples illustrate two common situations in which you have multiple observations per repeated effect per subject:

1. You have a data mistake.

   In this first data example, a simulated data set with 20 subjects in each of 3 groups with 3 repeated measures on each subject are used to demonstrate a repeated-measures analysis with a spatial-correlation structure. Here is the sample code and the resulting notes from the log:

```
139  proc mixed data=test;
140     class group id;
141     model y=group;
142     repeated / type=sp(pow)(time) subject=id(group);
143  run;

NOTE: An infinite likelihood is assumed in iteration 0 because of a
      nonpositive definite
      estimated R matrix for id(group) 10 2.
NOTE: PROCEDURE MIXED used (Total process time):
      real time           0.01 seconds
      cpu time            0.03 seconds
```

   Upon examination, the simulated data set indeed shows that there is a repeat observation for subject 10 in group 2 as shown with the following observations of the printed data set:

Obs	Y	Group	ID	Time
13	4.40865	2	9	1
14	4.06241	2	9	2
15	5.45901	2	9	3
16	2.32084	2	10	1
17	5.59659	2	10	2
18	8.99619	2	10	2
19	6.04762	2	10	3
20	3.83093	2	11	1
21	5.22728	2	11	2
22	8.39027	2	11	3

2. There is a misspecified subject effect or repeated effect.

   The SUBJECT= effect in the REPEATED statement is used to identify the subjects from which the repeated measures were obtained. Sometimes this subject effect needs to be a combination of several variables. For example, in your study each patient is randomly assigned to each of the treatment combinations, and at each treatment combination you obtain repeated measures over several time points. The first few observations of your data might look similar to the following:

patient	trt1	trt2	time	resp
1	1	1	1	4.3
1	1	1	2	3.8
1	1	1	3	3.1
1	1	1	4	2.7
1	1	2	1	4.3
1	1	2	2	3.4
1	1	2	3	3.2
1	1	2	4	3.5
1	2	1	1	3.0
1	2	1	2	3.3
1	2	1	3	2.5
1	2	1	4	3.4
1	2	2	1	3.3
1	2	2	2	3.2
1	2	2	3	3.0
1	2	2	4	3.2
2	1	1	1	2.6
2	1	1	2	3.0
2	1	1	3	3.2
2	1	1	4	4.1
2	1	2	1	4.3
2	1	2	2	3.8
2	1	2	3	4.2
2	1	2	4	4.1
2	2	1	1	3.9
2	2	1	2	4.2
2	2	1	3	3.5
2	2	1	4	3.7
2	2	2	1	3.3
2	2	2	2	2.8
2	2	2	3	2.8
2	2	2	4	3.2

You mistakenly specify the SUBJECT= effect in the REPEATED statement for these data as follows:

```
proc mixed data=test;
   class patient trt1 trt2 time;
   model resp=trt1 trt2 trt1*trt2 time;
   random int trt1 trt2 / subject=patient;
   repeated time / subject=patient type=ar(1);
run;
```

You would then see the following note in the log:

```
NOTE: An infinite likelihood is assumed in iteration 0 because of a nonpositive
      definite estimated R matrix for patient 1.
```

The note occurs because you have more than one observation per repeated effect (time) and subject (patient). You need to define your SUBJECT= effect so that it corresponds to the experimental unit on which the repeated measures over different time points were obtained, as follows:

```
proc mixed data=test;
   class patient trt1 trt2 time;
   model resp=trt1 trt2 trt1*trt2 time;
   random int trt1 trt2 / subject=patient;
   repeated time / subject=patient*trt1*trt2 type=ar(1);
run;
```

Now the subjects are correctly identified as patient*trt1*trt2. You have one observation per time point per subject, and the program runs with no problems.

One common study for this type of situation is a crossover design with repeated measures. Be sure to specify the subject effect appropriately for these type of data.

NOTE: A linear combination of covariance parameters is confounded with the residual variance.

This note suggests that you have overspecified your model. For example, the message is generated when each level of an interaction effect has a single observation and you define that interaction as a random effect. For a normal response variable in PROC MIXED or PROC GLIMMIX, you cannot specify a term as a random effect that is equivalent to the residual variance. This is typically the reason why the note is generated and can be circumvented by removing the random effect.

NOTE: At least one element of the (projected) gradient is greater than 1e-3.

You often see this note for PROC GLIMMIX and PROC NLMIXED.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm will converge if ABSGCONV < 1E5, FCONV < 10-FDigits, or GCONV < 1E8.

The GCONV convergence criterion in the NLMIXED or GLIMMIX procedure is a relative gradient criterion. If the relative change in the gradient values between two consecutive iterations is smaller than the cut off value, the convergence criterion is met and the procedure stops the iterations. The gradient value measures the rate of change in the response versus change in the associated parameter estimate. Even though the relative change in the gradient value is sufficiently small, the gradient value itself might not be small. When the value is greater than 1e-3, you see this note in the log. This note is not typically a concern if the gradient values are still reasonably small, for example, less than `0.01`. However, if you are worried about the possibly negative implications of this message, you might want to continue the estimation until the max gradient is sufficiently small. This can be done by simply setting the GCONV criterion to `0`. In PROC GLIMMIX, you use the following statement:

```
nloptions gconv=0;
```

In PROC NLMIXED, you add the GCONV=0 option to the PROC NLMIXED statement:

```
proc nlmixed gconv=0;
```

Setting the GCONV criterion to `0` will cause the procedure to continue for a few more iterations until the relative change in function value criterion (FCONV) is satisfied. Often times the estimates and their standard errors do not change much. Thus, there are no negative implications for having stopped with the default GCONV criterion in these cases.

The gradient values are automatically displayed in the PROC NLMIXED output. In PROC GLIMMIX, however, you would need the SOLUTION or S option in the MODEL statement to see the gradient values in the parameter estimates table.

WARNINGS

Warning messages are typically more serious indications of a problematic optimization process with your model for your data. Some of the commonly seen warnings are presented in this section.

WARNING: Stopped because of too many likelihood evaluations.

This warning is typically related to bad starting values or a model that is over-specified.

There are several recommendations to try to circumvent this warning. Here are some examples:

- Make sure the covariates and the response variable are on the same scale. Rescale if necessary.
- Try different starting values with the PARMS statement.
- Check that you do not have multiple observations at a given time point.
- Remove one or more of the random effects.
- Remove complex covariance structures such as TYPE=UN.
- Double check your model to be sure it is appropriate for your data.

WARNING: MIVQUE0 estimate of profiled variance is linearly related to other covariance parameters.

This warning is generated under several circumstances. In one situation, the PROC GLIMMIX step includes this statement:

```
random _residual_ / subject=id type=cs;
```

However, the subgroups had only one record per ID. If you specify TYPE=VC for these subgroups, you obtain the same results without the warning.

This particular warning can also occur with PROC GLIMMX models where the random effect is equivalent to residual variance. In PROC MIXED you might see this message:

```
NOTE: A linear combination of covariance parameters is confounded with the residual
      variance.
```

The warning "`MIVQUE0 estimate of profiled variance is linearly related to other covariance parameters`" also can indicate that your model might be over specified. Try to respecify your model so it is appropriate for your data.

WARNING: Pseudo-likelihood update fails in outer iteration xx.

There are many reasons for why an outer iteration might fail. Typically, it happens while PROC GLIMMIX is trying to iterate on the recently created pseudo-data set. That is, following an update of the parameter estimates, the procedure creates new pseudo-data. The algorithm cannot continue when the new pseudo-data are combined with the new parameter estimates.

This particular warning is data and model dependent. Sometimes using a likelihood-based estimation method, such as METHOD=QUAD or METHOD=LAPLACE, instead of the pseudo-likelihood based method might help. General strategies, such as rescaling your data, using the PARMS statement to supply your starting values, respecifying your model, using the appropriate options for the defined model, and so on, might help with this situation.

WARNING: The initial estimates did not yield a valid objective function.

PROC GLIMMIX fits a generalized linear model to obtain the initial fixed-effect estimates. Sometimes, the initial estimates might not yield a valid objective function for the generalized linear mixed model. To address this situation, you might increase the value specified in the INITITER= option in the PROC GLIMMIX statement. This option defines the maximum number of iterations for the optimization when the generalized linear model is fit initially. The default value is `4`. With this value of `4`, the optimization might have stopped before it converged. The resulting estimates were used as the starting values in the GLIMMIX model optimization, and apparently they were not good enough and did not yield a valid objective function value. Increasing the INITITER= value might help.

In some cases with the increased value of the INITITER= option, you might be able to circumvent the warning about the initial estimates not yielding a valid objective function, but you might encounter a new warning or error message, such as "`NEWRAP cannot improve the objective function`". Discussions about this message are provided later in this section. As always, rescaling and respecifying your model might help resolve this warning and error message all at once.

WARNING: The final Hessian matrix is full rank but has at least one negative eigenvalue. Second-order optimality condition violated.

This warning typically appears for PROC NLMIXED. Although strategies for handling PROC NLMIXED convergence issues are provided later in this section, rescaling often helps for this particular situation. Scaling is one of the common causes of convergence issues in mixed models. For example, if the values for one of your variables are in thousands and other values for other variables are between `0` and `1`, you are likely to encounter this type of warning. Rescale the data, for example, divide the variable by 1000. This often helps with the convergence.

Sometimes, if the model parameters are several orders of magnitude apart, you might want to rescale the data or reparameterize the model. For example, if *a1* is estimated to be `150`, *s2u* is estimated to be `0.004`, you might want to reparameterize the model. A common approach is to reparameterize the variance estimates or use a Cholesky-root reparameterization, as shown below:

- Use a standard deviation rather than the variance. Your RANDOM statement is likely to change in this way:

  ```
  random u ~ normal(0, sd*sd) subject=id;
  ```

  In this statement, `sd` is the standard deviation for the random effect `u`.

- Use other parameterizations, such as the log of the standard deviation, as the parameter. Here is an example:

  ```
  parms logsigu=0.3;
  random u ~ normal(0, exp(2*logsigu)) subject=id;
  ```

  It can be shown that `exp(2*log of the standard deviation)` is the variance.

- For an unstructured covariance matrix for random effects, use Cholesky-root reparameterization. That is, if $\mathbf{A}$ is a *p x p* positive definite matrix, you can find an upper triangular matrix such that $\mathbf{A}= \mathbf{T}'\mathbf{T}$, so that $\mathbf{T}$ is a type of square root of $\mathbf{A}$. For a 2 x 2 matrix, it can be shown the following:

$$T' = \begin{bmatrix} t_1 & 0 \\ t_{12} & t_2 \end{bmatrix}$$

$$T'T = \begin{bmatrix} t_1 & 0 \\ t_{12} & t_2 \end{bmatrix}\begin{bmatrix} t_1 & t_{12} \\ 0 & t_2 \end{bmatrix} = \begin{bmatrix} {t_1}^2 & t_1 t_{12} \\ t_1 t_{12} & {t_{12}}^2 + {t_2}^2 \end{bmatrix} = \begin{bmatrix} {\sigma_1}^2 & \sigma_{12} \\ \sigma_{12} & {\sigma_2}^2 \end{bmatrix} = A$$

  Therefore, you can reparameterize the unstructured matrix with the Cholesky root. The following example program illustrates the use of Cholesky-root reparameterization:

```
proc nlmixed data=test;
   parms b0=-2.4 b1=3.6 t1=0.2 t12=-1 t2=0.6;
   eta=b0+b1*trt+u0+u1*trt;
   p=1/(1+exp(-eta));
   model y ~ binomial(n, p);
   s2u0 = t1*t1;
   su01 = t1*t12;
   s2u1 = t12*t12 + t2*t2;
   random u0 u1 ~ normal([0, 0], [s2u0, su01, s2u1]) subject=clinic;
   estimate 'intercept variance s2u0' t1*t1;
   estimate 'slope variance s2u1' t12*t12 + t2*t2;
   estimate 'covariance su01' t1*t12;
run;
```

  In the preceding example program, the variances and covariance in the UN structure are not parameterized directly. Instead, *t1*, *t12*, and *t2* are used as the model parameters. Based on the mathematical relationship between the UN and CHOL structures, you can compute the variances and the covariance in the UN structure accordingly. Cholesky-root reparameterizations generally have better numerical behaviors than the UN structure, and are useful when the program failed to converge or was experiencing a long run time.

  Other than rescaling your data or parameters, you might also want to double check your model specification. In some cases, a misspelled variable or parameter name could cause this warning or other error messages, which indicate problems with convergence.

ERRORS

Error message are the most severe indications of numerical challenges in fitting your model to your data. Some of the common error messages are summarized below.

ERROR: NEWRAP cannot improve the objective function.

This message indicates that the full likelihood optimization routine is in an area where several iterations produced the same estimates. This usually happens when the model is complicated or the data set is huge, especially with multinomial models where some levels of the response are relatively rarely observed, leading to separation.

You might try the following strategies for resolving this message in PROC GLIMMIX:

- Use different estimation methods such as the METHOD=QUAD, METHOD=LAPLACE, or METHOD=RMPL options in the PROC GLIMMIX statement.
- Specify initial starting values in the PARMS statement.
- Check the scales of your variables to see whether they differ in magnitude. Rescale if necessary.
- Add the INITGLM option in the PROC GLIMMIX statement.
- Increase the maximum number of optimizations from the default of 20 to 50 by using the MAXOPT= option in the PROC GLIMMIX statement.

ERROR: QUANEW Optimization cannot be completed.
NOTE: Optimization routine cannot improve the function value.

ERROR: Quadrature accuracy of 0.000100 could not be achieved with 31 points. The achieved accuracy was 1.000000.

These two messages often occur with PROC NLMIXED. They can also happen with PROC GLIMMIX, and the text can vary depending on the optimization routine that is used. These messages are data and model dependent. Unfortunately, there is not one strategy that always works for these types of situations. General strategies as described previously in this section should help, as well as some additional information outlined below specifically for PROC NLMIXED.

1. Make sure your data is sorted by the SUBJECT variable.

   PROC NLMIXED does not have the CLASS statement. It is important to sort your data by the SUBJECT= effect before running PROC NLMIXED.

2. Make sure there are no spelling errors for your parameters or variables.

   If PROC NLMIXED cannot recognize a name that is not in your data set nor defined in the PARMS statement, the procedure will treat the name as a new parameter. It can often cause convergence issues. Double check the spelling of your variables and parameters!

3. Make sure the parameter does not have the same name as one of the variables in your data set.

   If you name one of your parameters the same name as one of your variables in the data, PROC NLMIXED will treat it as the variable name and use the values for that variable in your data set. This can cause convergence issues.

4. Make sure you use the PARMS statement to specify reasonable starting values.

   Starting values are very important in PROC NLMIXED modeling. Unreasonable starting values can have an adverse effect on the optimization process, and therefore cause convergence problems. The default starting value is 1 for all parameters, and this is often not appropriate. You need to make sure you specify reasonable starting values for PROC NLMIXED models.

5. Make sure you do not have any scaling issues.

   A scaling problem is one of the common causes of convergence issues in mixed models. Rescaling your data or reparameterizing your model often helps with the convergence issues. Refer to common rescaling and reparameterization strategies described previously in this section for examples and recommendations.

6. Make sure your model is appropriately specified for your data.

   Many convergence issues are caused by an incorrectly specified model, including spelling mistakes in the parameters or variables; an inappropriately defined link function, an inappropriately defined log likelihood, and so on. The circumvention is always data dependent, so examples are not presented in this paper.

Note: Execution error for Observation xxx

ERROR: Signal caught by CMP from PROC.

ERROR: Invalid Operation.

ERROR: Termination due to Floating Point Exception

This note and error messages often happen with PROC NLMIXED. They typically indicate that some operations in your PROC NLMIXED statements might not be appropriate for the observations in your data. To address these issues, you might want to try the following steps:

1. Check the model specifications.

   These messages can be caused by a misspecified model, for example, an inappropriate link function, an incorrectly defined log likelihood function, and so on. Always double check your model.

2. Double check your statements and data.

   The program might be trying to take the log of a nonpositive number, divide by a number practically zero, take the square root of a negative number, and so on. Check your data, specifically for the observation xxx that was singled out in the log. In some cases, even though these invalid operations might look unlikely given your data, you are recommended to write corresponding statements to prevent this from happening. For example, use the if-then statement to only take the log of a positive value, as illustrated in the following statements:

```
if lhf>1e-8 then logl=log(lhf);
else logl=-1e20;
```

   Another common approach for addressing these messages is to program the log likelihood directly, rather than taking the log of the programmed likelihood. For example, for a negative binomial model, you initially compute the log likelihood similar to the following:

```
pdf = gamma(y + 1/k)/(gamma(y + 1)*(gamma(1/k)))*
   (((k*lambda)**y)/(1 + k*lambda)**(y + 1/k));
   if pdf > 1e-8 then loglike = log(pdf);
   else loglike = -1e20;
```

   Instead, it might work better numerically if you compute the log likelihood directly, as illustrated in the following specification:

```
loglike = lgamma( y + 1/k) – lgamma( y + 1) - lgamma (1/k) + y * log(k*lambda)
– (y+1/k)*log(1+k*lambda)
```

   Programming the log likelihood directly can sometimes prevent invalid operations from taking place in the computation of the likelihood function.

ERROR: No Valid Parameter Points were found

You might want to use the PARMS statement to specify a different set of starting values. To find reasonable starting values, you might want to fit a simpler model first. For example, fit a model with no random effects first by using PROC MIXED, PROC LOGISTIC, PROC GENMOD, PROC GLIMMIX, or PROC NLIN. For random effect parameters, you might consider fitting a simpler fixed effect part of the model, so that you can use procedures that do not require starting values. Subject matter knowledge, results from past studies, and literature always serve as excellent sources for determining starting values. If possible, graph your data to get a better sense of the magnitude of the parameters so you can specify reasonable starting values.

This error tends to happen when the data does not support the model, so double check the model specification for your data.

CONCLUSION: SELECTING THE BEST ANALYSIS STRATEGY FOR MIXED MODELS

Mixed models are used in a variety of research situations. Panel data, split plots, strip plots, repeated measures, multi-site clinical trials, hierarchical linear models, random coefficients, and cross classified and analysis of covariance are all special cases of the mixed model that might encounter an out of memory error or long executing times if not coded efficiently.

There are a variety of considerations to be made in order to program a mixed model efficiently. Several strategies were discussed in this paper that you can use to improve the efficiency of the program. Here are some strategies:

- Factor out the common effect and specify it as a SUBJECT= effect.
- Equivalently code DIST=NORMAL models with a random effect and many levels using the RANDOM _RESIDUAL_/TYPE=CS SUBJECT= statement in PROC GLIMMIX or the REPEATED/TYPE=CS SUBJECT= statement in PROC MIXED.
- Process by subjects when possible.
- Try different DDFM= options.
- Consider alternative TYPE= covariance structures that are supported by the data and expert knowledge.
- Define the SUBJECT= effect as a nested effect when appropriate.

- Remove the numeric SUBJECT= effect from the CLASS statement provided that the data is sorted by the SUBJECT= effect.
- Simplify your model as much as possible.
- Consider using PROC HPMIXED.

The final section of the paper provide information about some of the various notes, warnings, and error messages that occur in using PROC GLIMMIX, PROC MIXED, and PROC NLMIXED. Providing recommendations and circumventing various Notes, Warnings, and Error messages are often data and model dependent and should be evaluated on a case-by-case basis. Common strategies include the following:

- Rescale your data so the values are on a similar scale.
- Simplify your model first.
- Respecify your model.
- Use the PARMS statement.
- Use an alternative estimation method.
- Reparameterize your model.

The goal of this paper was to provide recommendations for reducing performance problems and troubleshooting nonconvergence problems, and to discuss common Notes, Warnings, and Error messages when using PROC GLIMMIX, PROC MIXED, and PROC NLMIXED. These recommendations are based on experience supporting these procedures in SAS Technical Support and working with the respective developers of the procedures. The consultants in SAS Technical Support are always happy to help address questions about using SAS software. You can contact Technical Support by phone at 919-677-8008 or electronically by submitting the SAS Technical Support Form online (support.sas.com/ctx/supportform/createForm).

REFERENCES

SAS Institute Inc. 2012. Select Chapters in *SAS/STAT® 9.3 User's Guide*. Cary, NC: SAS Institute Inc.

"The GLIMMIX Procedure." Available at support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_glimmix_a0000001398.htm.

"The HPMIXED Procedure." Available at support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#hpmixed_sect001.htm.

"The MIXED Procedure." Available at support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_mixed_sect001.htm.

"The PLM Procedure." Available at support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_plm_a000000115.htm.

"Processing by Subjects." Available at support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_glimmix_a0000001446.htm.

SAS Institute Inc. 2009. SAS Note 37057 "How large of a hierarchical linear model (HLM) can PROC MIXED fit?". Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/37/057.html.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Kathleen Kiernan
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com

Jill Tao
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com

Phil Gibbs
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com