Paper SAS1919-2015

Advanced Techniques for Fitting Mixed Models Using SAS/STAT® Software

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

ABSTRACT

Fitting mixed models to complicated data, such as data that include multiple sources of variation, can be a daunting task. SAS/STAT® software offers several procedures and approaches for fitting mixed models. This paper provides guidance on how to overcome obstacles that commonly occur when you fit mixed models using the MIXED and GLIMMIX procedures. Examples are used to showcase procedure options and programming techniques that can help you overcome difficult data and modeling situations.

INTRODUCTION

SAS/STAT software features several powerful and complex procedures for fitting mixed models. With this complexity can come confusion as to how to best handle difficult modeling issues. This paper illustrates some of the most common difficulties in fitting a mixed model by presenting a series of examples that work through critical issues in deciding how to set up a model. Topics include handling multiple levels of subjects, working with tests of simple effects and effect slices, performing additional post-hoc testing after the model is estimated, and handling predicted probabilities in a generalized linear mixed model.

SPECIFYING COMPLEX MIXED MODELS

Mixed models involve the modeling of random effects, correlated errors, or both. Questions often arise in mixed modeling when you use PROC MIXED, whether you are analyzing data from a simple randomized complete block design, a complex crossover design, or a repeated measures design. For example, what is the appropriate specification for the SUBJECT= effect in the RANDOM or REPEATED statement? When should you specify the repeated effect in the REPEATED statement? How do you model the data when measurements take on more than one experimental unit? The following sections discuss these issues and present examples.

DETERMINING WHETHER TO USE A NESTED, CROSSED, OR SINGLE SUBJECT= EFFECT

When you have data from a randomized complete block design, the specification of the SUBJECT= effect in the RANDOM statement is straightforward—it is typically the blocking variable in your data. However, when the data is more complicated (for example, in hierarchical linear models, split-plot designs, cross-over designs, or other more complex situations), the specification of the appropriate SUBJECT= effect can be a source of confusion. Particularly, should you specify the SUBJECT= effect using a single variable, a nested effect, or a crossed effect? The following example illustrates the specification of the appropriate SUBJECT= effect.

In this example, data is collected from two hospitals. In each hospital, four physicians are randomly selected and measurements for X and Y are obtained from different patients for that physician. You want to model Y as a linear model of X, accounting for the correlations among multiple observations within a physician. Table 1 shows the collected data for this example.

Hospital	Physician	X	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	20.3
1	2	18.2	21.3
1	3	19.7	24.8

(table continued)

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
1	4	20.8	30.7
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	16.4

Table 1. The Hospital Data Set

For illustration purposes, the Hospital effect is ignored at first. The following PROC MIXED program fits a linear mixed model with X as the independent variable, and Physician as a random effect:

```
proc mixed data=hospital;
   class physician;
   model y=x / ddfm=kr s;
   random int / subject=physician;
run;
```

In This Example:

- The SUBJECT=PHYSICIAN option in the RANDOM statement fits the random Physician effect.
- The values for Physician uniquely identify the levels of the physicians. Therefore, no nesting is necessary. If you nest Physician within Hospital [that is, you specify SUBJECT=PHYSICIAN(HOSPITAL) in the RANDOM statement], you obtain identical results.

Now suppose you also want to model Hospital as a random effect. Specifying a second RANDOM statement to the model above adds that new random term to the model:

```
proc mixed data=hospital;
   class hospital physician;
   model y=x / ddfm=kr s;
   random int / subject=hospital;
   random int / subject=physician;
run;
```

There is a computational advantage if you rewrite your second RANDOM statement by nesting Physician within Hospital as the SUBJECT= effect:

```
proc mixed data=hospital;
   class hospital physician;
   model y=x / ddfm=kr s;
   random int / subject=hospital;
   random int / subject=physician(hospital);
run;
```

With this model specification, HOSPITAL appears as common term in the SUBJECT= effects of the two RANDOM statements. This enables the V matrix to be processed by subject, and it is more computationally efficient. Therefore, it can be beneficial to nest your subjects even when nesting is not necessary.

As a side note, the default denominator degrees-of-freedom estimation method, DDFM=CONTAIN, is a syntax-dependent estimation method. This method might produce different degrees-of-freedom estimates when the SUBJECT= effect is specified differently. That change in degrees of freedom can result in different *p*-values for some of the fixed effects. For example, you might obtain different estimates for the denominator degrees of freedom for some fixed effects when you SUBJECT=PHYSICIAN or SUBJECT=PHYSICIAN(HOSPITAL). Other DDFM= options are not syntax-dependent and therefore do not have this issue. Generally speaking, the nesting specification of the subject effect might produce more reasonable estimates of the denominator degrees of freedom.

In summary, when the values for the subject variable uniquely identify the levels, nesting is not necessary. However, nesting the effects can be beneficial for performance improvement and to appropriately estimate the default denominator degrees of freedom.

Now suppose the values for Physician change, as indicated in Table 2.

Hospital	Physician	X	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	20.3
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
1	4	20.8	30.7
2	1	21.5	31.3
2	1	18.7	28.5
2	1	19.9	20.1
2	2	18.9	31.8
2	2	18.7	25.1

(table continued)

2	2	19.5	32.9
2	3	18.7	18.3
2	3	19.4	27.4
2	3	19.5	26.2
2	4	20.9	28.6
2	4	21.1	29.9
2	4	20.0	16.4

Table 2. Physicians That Are Not Uniquely Numbered

The values for Physician do not uniquely identify the physicians anymore. The information for Hospital must be considered as well to identify the appropriate physician. You must nest Physician within Hospital to identify the total of eight unique physicians, as shown in this RANDOM statement:

```
random int / subject=physician(hospital);
```

In this case, nesting is necessary. If you do not nest the effects, the physicians are not identified appropriately. As a result, unexpected results are produced.

Whether you nest or cross two effects, the same level for the subjects is identified. Therefore, it typically does not matter whether you nest or cross the two factors for the SUBJECT= effect in the RANDOM statement or in the REPEATED statement.

IDENTIFYING MULTIPLE SUBJECT EFFECTS

In the following example, a pharmaceutical company wants to examine the effects of a new drug on the respiratory ability of asthma patients. The company evaluates the performance of the new drug (a) against a placebo (p) and a control (c). Each of the 24 patients in the study received the three treatments in a random order, with four patients randomized to each of the six treatment sequences. Each patient made three visits to a clinic, with sufficient time between visits to eliminate the consideration of a carry-over effect. At each visit to the clinic, a baseline, Basefev1, is measured immediately before the drug is administered, and Fev1 was measured hourly for four hours following the treatment.

Table 3 contains a partial listing of the data.

Patient	Sequence	Basefev1	Drug	Period	Hour	Fev1
201	B	2.14	p	1	1	2.36
201	B	2.14	p	1	2	2.36
201	B	2.14	p	1	3	2.28
201	B	2.14	p	1	4	2.35
201	B	2.30	c	2	1	3.41
201	B	2.30	c	2	2	3.48
201	B	2.30	c	2	3	3.41
201	B	2.30	c	2	4	3.49
201	B	2.46	a	3	1	2.68
201	B	2.46	a	3	2	2.76
201	B	2.46	a	3	3	2.50
201	B	2.46	a	3	4	2.30
202	E	2.91	c	1	1	3.92
202	E	2.91	c	1	2	4.02
202	E	2.91	c	1	3	4.04
202	E	2.91	c	1	4	3.64
202	E	3.37	p	2	1	3.03

Table 3. Partial Listing of Fev1uni Data

The analysis of this data is complex because there are two experimental units involved in the study. The first experimental unit is the individual patient. Observations from the same patient should be correlated. The second experimental unit is the application of a drug on a patient. The four hourly observations that are taken after the application of each drug should also be correlated. But how do you model correlations in this type of data?

You can analyze this data in several ways. One approach is to use the RANDOM and REPEATED statements in PROC MIXED. The RANDOM statement models the correlations among the observations within the same patient. Then you can use the REPEATED statement to further model the correlations among the repeated measures over the four hourly measurements on a specific treatment that is applied to a patient. You can think of this approach as modeling the crossover part of the data in the RANDOM statement, and modeling the repeated measures part of the data in the REPEATED statement. The REPEATED statement is used to model the correlation of observations on the smallest experimental unit. The key here is to identify the subject effect appropriately. The four hourly measurements were obtained for each patient for a specific drug (or during a specific period). The subject effect, then, is the combination of patient and drug (or patient and period). In PROC MIXED, you can use the option SUBJECT=PATIENT*DRUG (or SUBJECT=PATIENT*PERIOD) to identify the appropriate subjects for the REPEATED statement.

Using this approach, the code for PROC MIXED is as follows:

```
proc mixed data=fev1uni;
   class drug hour patient sequence period;
   model fev1=drug hour drug*hour basefev1 sequence period / ddfm=kr;
   random int / subject=patient;
   repeated hour / subject=patient*drug type=ar(1);
run;
```

In This Example:

- The RANDOM statement models the correlations of the observations within the same patient.
- The REPEATED statement models the correlations among the four repeated measures for each patient for each drug.
- As discussed previously, you might nest the factors instead of crossing for the SUBJECT= effect in the REPEATED statement.
- While the first-order autoregressive structure is used in the example, other covariance structures can be used.

Another approach is to model the two repeated effects using a Kronecker product covariance structure in the REPEATED statement in PROC MIXED. The experimental unit is Patient, and there are two repeated effects for each patient: Drug and Hour. There are three Kronecker product covariance structures available in PROC MIXED: UN@UN, UN@AR(1), and UN@CS. These structures were developed for multivariate repeated measures data, and they are useful in modeling two repeated effects for a single SUBJECT= effect. Below is the sample program:

```
proc mixed data=fev1uni;
   class drug hour patient sequence period;
   model fev1=drug hour drug*hour basefev1 sequence period;
   repeated drug hour / subject=patient type=un@ar(1) r;
run;
```

The Kronecker product structure is similar to combining the two covariance structures UN and AR(1). The covariance matrix R appears as follows when you have two levels for drug and three levels for hour. In this case, UN is a 2x2 matrix and AR(1) is a 3x3 matrix. The resulting matrix is a 6x6 matrix.

$$\begin{bmatrix} \sigma^2_1 & \sigma_{12} \\ \sigma_{21} & \sigma^2_2 \end{bmatrix} \otimes \begin{bmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \sigma^2_1 & \sigma^2_1\rho & \sigma^2_1\rho^2 & \sigma_{12} & \sigma_{12}\rho & \sigma_{12}\rho^2 \\ \sigma^2_1\rho & \sigma^2_1 & \sigma^2_1\rho & \sigma_{12}\rho & \sigma_{12} & \sigma_{12}\rho \\ \sigma^2_1\rho^2 & \sigma^2_1\rho & \sigma^2_1 & \sigma_{12}\rho^2 & \sigma_{12}\rho & \sigma_{12} \\ \sigma_{21} & \sigma_{21}\rho & \sigma_{21}\rho^2 & \sigma^2_2 & \sigma^2_2\rho & \sigma^2_2\rho^2 \\ \sigma_{21}\rho & \sigma_{21} & \sigma_{21}\rho & \sigma^2_2\rho & \sigma^2_2 & \sigma^2_2\rho \\ \sigma_{21}\rho^2 & \sigma_{21}\rho & \sigma_{21} & \sigma^2_2\rho^2 & \sigma^2_2\rho & \sigma^2_2 \end{bmatrix}$$

The Fev1uni data example actually has 3 drugs and 4 hours, so the resulting Kronecker product covariance matrix is a 12x12 matrix. The R option in the REPEATED statement displays this covariance matrix for the first patient.

Even though the two approaches generate different results, both models provide an appropriate analysis. You might want to examine the fit statistics from both models to see which one fits the data better. You might also consider the interpretation of these two models and choose the one that makes the most sense for your data.

UNDERSTANDING WHEN THE REPEATED EFFECT IS NECESSARY

The *repeated effect* (Hour in the following statement) is optional from the syntax perspective.

```
repeated hour / subject=patient*drug type=ar(1);
```

The Hour effect indicates how PROC MIXED should order the observations for a given subject. If no repeated effect is listed, as shown in the following statement, then PROC MIXED assumes that the observations for a given subject are listed in the appropriate order within the input data and that there is no missing data in the repeated measures.

```
repeated / subject=patient*drug type=ar(1);
```

However, when you have missing observations at some time points and those missing observations are omitted from the data, you might obtain different results with and without the repeated effect for certain covariance structures.

The following example shows how this issue might arise. The data are listed by Block, but not all values of Time are available for all blocks.

```
data temp;
   input Block Time Y;
   datalines;
1   1  56
1   2  41
1   4  36
1   5  24
1   6  41
1   7  50
```

(code continued)

```
1   8  39
1   9  35
2   1  30
2   2  25
2   3  36
2   4  28
2   6  36
2   7  40
2   8  33
2   9  30
3   1  32
3   2  24
3   3  31
3   4  27
3   5  45
3   6  32
3   7  23
3   8  15
3   9  19
;
run;
```

In the Temp data set shown above, there are a total of nine values for Time, as observed in Block 3. However, not all nine levels of Time are available for Block 1 and Block 2. For Block 1, the Time variable is missing a value of 3; for Block 2, the Time variable is missing a value of 5. One possible model you can fit to this data is as follows:

```
proc mixed data=temp;
   class Block Time;
   model Y=Time;
   repeated / subject=Block type=ar(1);
run;
```

Without Time listed as the repeated effect in the REPEATED statement, PROC MIXED does not know which variable to use to identify the multiple observations for each block. Therefore, PROC MIXED uses the rank order of the observations to identify the multiple observations within each block. As a result, the third observation for Block 1 is considered the third time measurement for Block 1 (rather than Time 4 for that block). This misinterpretation carries forward for the remaining observations in that block. In the end, the procedure assumes that Block 1 has no observation for the ninth measurement, whereas, in fact, Time 3 is missing in the data set. A similar problem occurs for Block 2. A quick solution is to include the repeated effect Time in the REPEATED statement, as shown below:

```
proc mixed data=temp;
   class Block Time;
   model Y=Time;
   repeated Time / subject=Block type=ar(1);
run;
```

When you include Time in the REPEATED statement, there is no misinterpretation of the time values for the repeated measures within a block because the values for Time are used to identify the repeated measurements.

The results from running the two models differ because of the handling of the repeated measures within each block. The following partial output from the two models illustrates where the results might be different:

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
AR(1)	Block	0.5935
Residual		111.25

Output 1. Covariance Parameter Estimates Table from the Model without the REPEATED Effect

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
AR(1)	Block	0.5873
Residual		110.52

Output 2. Covariance Parameter Estimates Table from the Model with the REPEATED Effect

The results from Output 2 are consistent with what you expect.

You can omit the repeated effect that precedes the slash in the REPEATED statement only when you have one of the following conditions:

- complete data for each subject.
- missing response values that are explicitly coded in your data set. That is, if you have 10 subjects, each observed at 5 time points, then you must have 50 observations in your data set. Missing measurements are recorded using missing values in the response variable.

If you have missing values in the explanatory variables of your model, then you must include the repeated effect in the REPEATED statement to preserve the correct spacing of the measurements for a given subject.

The general recommendation is to specify the repeated effect in the REPEATED statement in PROC MIXED, regardless of whether you have missing time points. By including the repeated effect in the REPEATED statement, PROC MIXED has a clear instruction on how to handle the multiple measurements within a subject.

All of the discussions and examples thus far used PROC MIXED. The concepts also apply to PROC GLIMMIX with regard to specifying the SUBJECT= effects. A couple of points are worth mentioning.

- Kronecker product covariance structures are not available in PROC GLIMMIX.
- There is no REPEATED statement in PROC GLIMMIX. Instead, you use the RANDOM statement with appropriate keywords to fit a model that is similar to the REPEATED statement in PROC MIXED. For example, suppose you have this REPEATED statement in PROC MIXED:

```
repeated / subject=Block type=ar(1);
```

(list continued)

You can replace that statement with this RANDOM statement in PROC GLIMMIX:

```
random _residual_ / subject=Block type=ar(1);
```

Or, suppose you have the following REPEATED statement in PROC MIXED with a repeated effect of Time:

```
repeated Time / subject=Block type=ar(1);
```

You can replace that statement with the following RANDOM statement in PROC GLIMMIX:

```
random Time / subject=Block type=ar(1) residual;
```

POSTFITTING ANALYSIS IN MIXED MODELS WITH THE PLM PROCEDURE

Most researchers recognize that estimation of mixed models is computationally intensive, requiring lots of memory and processing time. If you consider using item stores and the PLM procedure to separate common postprocessing tasks, you might be able to save considerable time. With this approach, you can apply a numerically expensive model-fitting technique once to fit the model and to store the statistics from this fit into 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. PROC PLM contains many useful statements for estimation, inference, and graphical displays. In addition, PROC PLM offers the following statements that are helpful for displaying and filtering analysis contents and results.

- **FILTER:** Filters the results of the PLM procedure.
- **SHOW:** Displays the contents of the source item store.
- **WHERE:** Selects a subset of BY groups from the source item store to which to apply the PROC PLM statements.

Using item stores and PROC PLM enables you to separate common postprocessing tasks (for example, contrasts, estimates, LS-means, and estimates among LS-means) from the fitting of your mixed model.

GETTING STARTED WITH AN ITEM STORE

The STORE statement requests that a SAS procedure save the context and results of the statistical analysis into an item store. An *item store* is an internal binary file that cannot be modified by the user. You can process the contents of an item store with PROC PLM and use the item store to produce further postprocessing results for the statistical analysis. The item-store name is a one- or two-level SAS name, similar to the name structure that is used for SAS data sets. If you specify a one-level name, the item store resides in the Work library, and it is deleted at the end of the SAS session. In order for model results to be available in another SAS session, you need to specify a two-level name in the following form: *library.member-name*. Be aware that item stores are not portable across computing platforms. For example, you cannot use an item store that is produced in UNIX operating environments in Microsoft Windows operating environments. However, item stores are portable across SAS releases within a platform.

The following three examples demonstrate using item stores and PROC PLM for some commonly used postprocessing tasks from the fitting of your mixed model. The first example investigates an interaction and demonstrates tests of simple effects. The second example illustrates scoring. The final example computes treatment differences and differences in event probabilities for a binomial logistic regression model.

INVESTIGATING AN INTERACTION USING THE PLM PROCEDURE

This first example demonstrates using PROC PLM to investigate an interaction with the EFFECTPLOT and SLICE statements.

The data set that is analyzed in this task contains data from Littell, Freund, and Spector (1991). Subjects in the study participated in one of three different weightlifting programs, and their strength was measured once every other day for two weeks after they began the program. The first program increased the number of repetitions as the subject became stronger (RI); the second program increased the amount of weight as subjects became stronger (WI); and the subjects in the third program did not participate in weightlifting (CONT). The objective of this analysis is to investigate the effect that each weightlifting program has on increasing strength over time.

The code for this analysis is as follows:

```
proc mixed data=split;
   class program time subject;
   model strength=program time program*time;
   repeated / subject=subject(program) type=cs;
   store mixres;
run;
```

The Type 3 tests on the fixed effects show a significant test for the Program*Time effect.

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
Program	2	54	3.07	0.0548
Time	6	324	7.43	<.0001
Program*Time	12	324	2.99	0.0005

Output 3. Type 3 Tests of Fixed Effects

Suppose you want to plot the Program*Time interaction and explore the tests for simple effects within Program*Time. You might use the SLICE and SLICEDIFF options in the LSMEANS statement in PROC GLIMMIX or the SLICE statement in PROC MIXED. However, using the SLICE statement in PROC PLM saves you the cost of refitting the model, as shown in the following example:

```
ods graphics on;

proc plm restore=mixres;
   effectplot interaction(sliceby=program);
   slice Program*Time / sliceby=Time diff adjust=tukey;
run;
```

PROC PLM takes the model results from the MIXRES (the source item store that is specified in the RESTORE= option), and it enables you to produce effect plot for the Program*Time interaction. The EFFECTPLOT statement is not available in PROC MIXED directly. However, storing the model results and running those results through PROC PLM adds the following graph to your mixed-model arsenal.

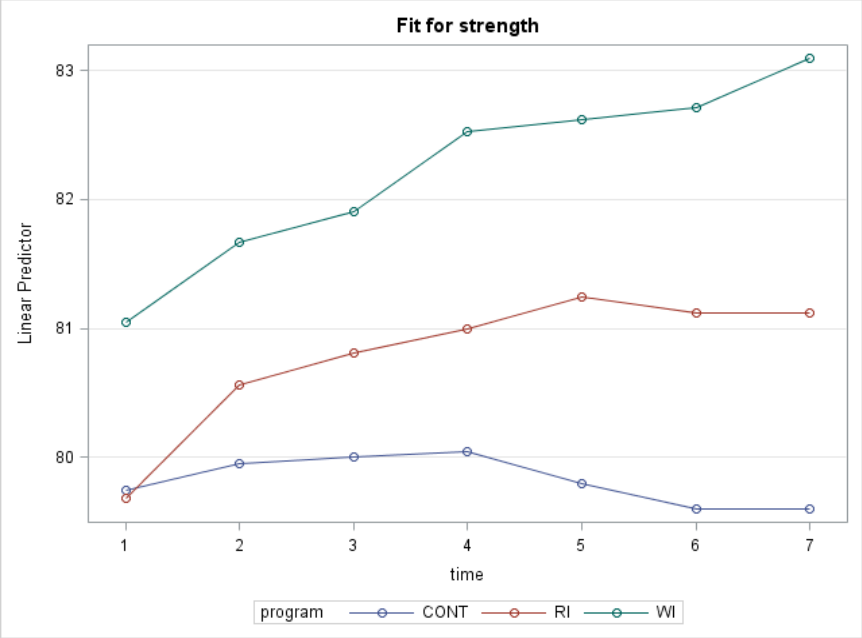

Figure 1. Effect Plot for the Interaction of Program*Time

The SLICE statement outputs the tests of simple effects, testing for Program differences within each level of Time. The results for Time 6 are shown below in Output 4.

F Test for Program*Time Least Squares Means Slice

Slice	DF	Num DF	Den DF	F Value	Pr > F
Time 6	2	324		4.60	0.0107

Simple Differences of Program*Time Least Squares Means
Adjustment for Multiple Comparisons: Tukey-Kramer

Slice	Program	_Program	Estimate	Standard Error	DF	t Value	Pr > |t|	Adj P
Time 6	CONT	RI	-1.5250	1.1023	324	-1.38	0.1675	0.3507
Time 6	CONT	WI	-3.1143	1.0268	324	-3.03	0.0026	0.0074
Time 6	RI	WI	-1.5893	1.0906	324	-1.46	0.1460	0.3130

Output 4. Results from Using the SLICE Statement in PROC PLM

The *F* test shows that there are significant differences among the programs at Time 6. The sliced differences show that the CONT program is different from WI program. PROC PLM also outputs a diffogram of the program simple effect comparisons at each time. Figure 2 shows this comparison at time 6.

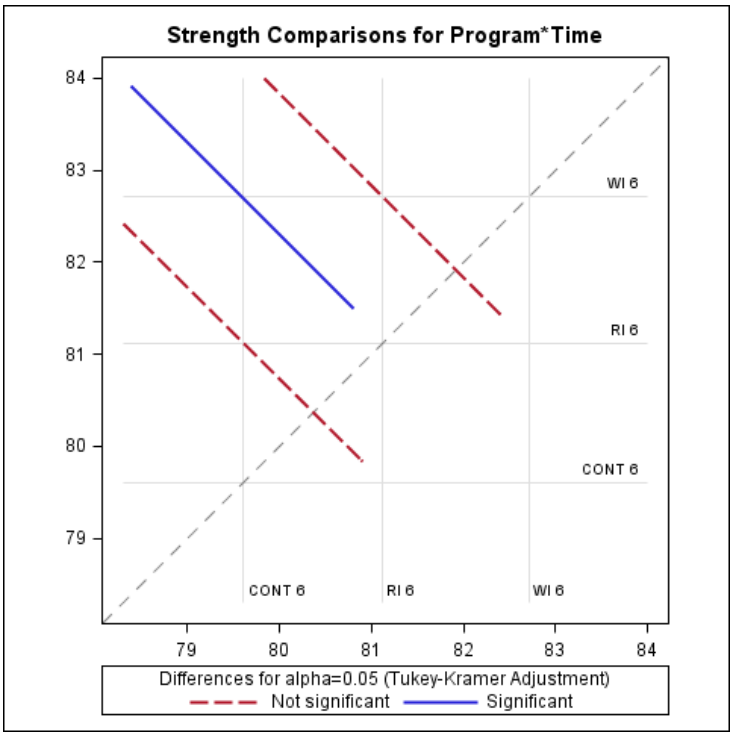

Figure 2. Diffogram Results for Program Comparisons at Time 6

SCORING MIXED MODELS IN THE PLM PROCEDURE

You can also use PROC PLM to score new data by using the SCORE statement. However, there is an important limitation of this statement. As mentioned before, one of the advantages of using PROC PLM is that you no longer need access to the input data set to produce postfitting analyses. You only need the item store to produce these results. To score a mixed model that involves both fixed and random effects ($x\beta+z\gamma$), the scoring process needs access to the full data set. The full data set is required in order to produce the standard errors of the predicted values whenever random effects are included in the model. Because PROC PLM only has access to the item store and not to the full data set, scoring can only be performed using the fixed-effects portion of the model. That is, scoring results are only performed for fixed effects ($x\beta$), not for both fixed and random effects ($x\beta+z\gamma$).

The following example from Gotway and Stroup (1997) analyzes data from an agronomic field trial. The researchers counted the number of damaged plants (Y) and the total number of plants (N) that were growing in each level of Entry. They identified the plants by the Block number and the Entry number within each block, respectively. Table 4 shows the first few observations in the data set Plants:

Y	N	Block	Entry
2	8	1	14
1	9	1	16
9	13	1	7
9	9	1	6
2	9	1	13
7	14	1	15

Table 4. Partial Observations in Data Set Plants

The responses are correlated because observations within a block share the same level of the random effect, Block. The following statements request that PROC GLIMMIX fit a generalized linear mixed model (GLMM) with a random block effect.

```
proc glimmix data=Plants method=quad;
   class block entry(ref='1');
   model y/n = entry / solution;
   random int / subject=block;
   store gmxplants;
run;
```

In this Example:

- The REF= option in the CLASS statement defines the first level, Entry 1, as the reference group rather than the default last level, Entry 16.
- The METHOD=QUAD option requests the quadrature estimation method for this model.
- The STORE statement saves the model results to the item store GMXPLANTS.

The Class Level Information table in Output 5 lists the levels of the variable that are specified in the CLASS statement above. The output also shows the ordering of the levels.

Class Level Information

Class	Levels	Values
Block	4	1 2 3 4
Entry	16	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1

Output 5. Class Level Information Table

Now suppose that you want to score a couple of new observations. The new observations are from Block 5 and Block 6, which are not included in the original study. Also suppose that the predicted values for Entry 13 and 6 are of particular interest. The new observations, stored in data set New, are shown below.

Block	Entry	Y	N
5	13	.	.
6	6	.	.

Table 5. Scoring Data Set New with New Blocks

Because there is no information about blocks 5 and 6 from the original data set, the predicted probability for these two observations can only come from fixed effects (xβ) from the original model. Using the SCORE statement in PROC PLM works fine in this case. Be aware that because Entry is treated as a classification fixed effect, the values for Entry in the data set New must be one of the existing levels in the original data set. Otherwise, if the level for Entry is a value that does not appear in the original data set (for example, 20), then no predicted values are computed because of missing information for the new entry in the fit model. However, this issue does not apply if Entry is a continuous variable in the model.

To score new observations based on fixed effects (xβ), you can use the following statements:

```
proc plm restore=gmxplants;
   score data=new out=out / ilink;
run;
```

In this example, the SCORE statement applies the contents of the source item store to compute predicted values based only on the fixed effects (xβ) and other observation-wise statistics for a SAS data set. The ILINK option requests the inverse linked predicted values (for this example, the probability of damaged plants).

The predicted probability is shown below in the data set Out.

Obs	Block	Entry	Y	N	Predicted
1	5	13	.	.	0.11428
2	6	6	.	.	0.62801

Output 6. Predicted Probabilities for New Blocks Computed Based on Xβ

Now suppose that you want to obtain predicted values for Block 1 and Block 2. Now the data set New has different observations, as shown in Table 6 below:

Block	Entry	Y	N
1	13	.	.
2	6	.	.

Table 6. Scoring Data Set NEW with Existing Blocks

The levels for Block 1 and Block 2 are are among the existing blocks in the original data set. In this case, the predicted values are computed as a linear combination of fixed and random effects (xβ+zγ). Therefore, the SCORE statement in PROC PLM is not appropriate to use if you want to include the random effect (zγ).

To score mixed models with random effects:

1. Append the data set containing the new observations that you want to score to the input data set.
2. Set the dependent variable to missing for the new observations.
3. Use the OUTPUT statement in PROC GLIMMIX to obtain predicted values.

Because the dependent variable is missing, those observations that are appended from the scoring data set New are not used to estimate the model. However, PROC GLIMMIX does provide predicted values for these observations using both the fixed and random components.

The following example program illustrates this approach of appending new observations to be scored to the existing input data set.

```
data new;
   input Block Entry Y N;
   datalines;
1 13 . .
2 6  . .
;

data all;
   set plants new;
run;
```

(code continued)

```
proc glimmix data=all method=quad;
   class block entry(ref='1');
   model y/n = entry / solution;
   random int / subject=block;
   output out=out pred(blup ilink)=pred;
run;
```

The data set Out contains the predicted probability for all observations in the data set, including the ones appended at the end. The values of Pred are the predicted probabilities calculated using both the fixed and random effects.

Obs	Y	N	Block	Entry	Pred
63	2	13	4	13	0.11488
64	9	13	4	8	0.45804
65	.	.	1	13	0.11466
66	.	.	2	6	0.62789

Output 7. Predicted Probabilities for Existing Blocks That Are Computed Based on Xβ+Zγ

Thus far, the scoring examples demonstrated PROC GLIMMIX and PROC PLM. However, the concepts also apply to PROC MIXED. To obtain predicted values that include the random effects in PROC MIXED, specify the OUTP= option in the MODEL statement.

COMPUTING DIFFERENCES IN LS-MEANS AND DIFFERENCES IN PROBABILITIES

In the previous section, the PROC GLIMMIX code for the Plants data is as follows:

```
proc glimmix data=Plants method=quad;
   class block entry(ref='1');
   model y/n=entry / solution;
   random intercept/subject=block;
   store gmxplants;
run;
```

Recall that the Block variable has four levels, and the Entry variable has sixteen levels. To compute the LS-means for Entry, you can use the following PROC PLM statements to conduct the postfitting analysis rather than incurring the cost of refitting the model in PROC GLIMMIX:

```
proc plm restore=gmxplants;
   lsmeans entry/ilink adj=tukey;
   filter adjp <0.001;
run;
```

In This Example:

- The LSMEANS statement computes the LS-means on the logit scale for the 16 levels for Entry.
- The ILINK option in the LSMEANS statement reports the LS-means on the scale of the mean, which is in predicted probabilities for this model.
- The ADJ=TUKEY option requests a Tukey-Kramer multiple comparison adjustment for the *p*-values for the differences of LS-means (difference in the logits). The ADJ=TUKEY option implies the PDIFF option, which produces the pairwise difference in the logits for the levels of Entry.

(list continued)

- Given 16 levels for Entry, there are 120 pairs of LS-means differences. Use the FILTER statement to subset those LS-means differences in which the adjusted *p*-value is significant (less than 0.001).

Output 8 displays those filtered LS-means differences on the logits in which the adjusted *p*-value is significant (less than 0.001).

Differences of entry Least Squares Means
Adjustment for Multiple Comparisons: Tukey-Kramer

Entry	_Entry	Estimate	Error	DF	t Value	Pr > |t|	Adj P
2	13	3.3564	0.6420	45	5.23	<.0001	0.0004
2	16	2.7749	0.5141	45	5.40	<.0001	0.0002
5	13	3.3472	0.6520	45	5.13	<.0001	0.0006
5	16	2.7657	0.5275	45	5.24	<.0001	0.0004
13	1	-3.5519	0.6598	45	-5.38	<.0001	0.0003
16	1	-2.9704	0.5384	45	-5.52	<.0001	0.0002

Output 8. Differences of LS-Means for the Entry Variable

The difference between Entry 2 and Entry 13 on the logit scale is 3.3564. If you want to obtain the difference in the event probability between Entry 2 and Entry 13, that is not currently available in PROC GLIMMIX. Furthermore, exponentiating that logit difference through an ESTIMATE statement in PROC GLIMMIX does not yield the difference in the event probabilities.

Some researchers are not comfortable interpreting the differences of LS-means in terms of logits and prefer to interpret results in terms of the difference in the event probabilities. In order to compute the difference in the event probabilities requires a nonlinear transformation of the predictors and is not currently available in the GLIMMIX procedure. However, the ESTIMATE statement in the NLMIXED procedure can compute the difference in event probabilities for fixed and G-side random effect logistic regression models.

To obtain the difference in event probabilities between two levels of Entry, use PROC NLMIXED to fit the model to your data, then use the appropriate ESTIMATE statement to form the difference of the event probabilities. Prior to SAS /STAT 13.2, in order for PROC NLMIXED to identify the subjects appropriately, the data must be sorted by the SUBJECT= effect. The following statements request that NLMIXED fit this model:

```
proc sort data=plants;
   by block;
run;

proc nlmixed data=plants;
   parms b0=1.5 b2=-0.1 b3=-0.5 b4=-1.5 b5=-0.2 b6=-1 b7=-0.5
         b8=-1.5 b9=-1.5 b10=-0.5 b11=-1.5 b12=-1.5 b13=-4
         b14=-3 b15=-2 b16=-3 s2=0.0001;
   logit=b0+b2*(entry=2)+b3*(entry=3)+b4*(entry=4)+b5*(entry=5)
         +b6*(entry=6)+b7*(entry=7)+b8*(entry=8)+b9*(entry=9)
         +b10*(entry=10)+b11*(entry=11)+b12*(entry=12)+b13*(entry=13)
         +b14*(entry=14)+b15*(entry=15)+b16*(entry=16)+u0;
   p=1/(1+exp(-logit));
```

(code continued)

```
   model y~binomial(n,p);
   random u0 ~ normal(0, s2) subject=block;
   estimate 'prob diff 16-1' 1/(1+exp(-(b0+b16)));
   estimate 'prob diff 2-13' 1/(1+exp(-(b0+b2)))- 1/(1+exp(-(b0+b13)));
run;
```

Output 9 shows the partial results of PROC NLMIXED step.

Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
prob diff 16-1	0.1875	0.05640	3	3.32	0.0449	0.05	0.008022	0.3670
prob diff 2-13	0.6733	0.08062	3	8.35	0.0036	0.05	0.4168	0.9299

Output 9. Differences in Event Probabilities for Entry 2 and Entry 13

In This Example:

- The event probability difference between Entry 16 and Entry 1 is 0.1875 with a 95% confidence interval of (0.008, 0.3670).
- The event probability difference between Entry 2 and Entry 13 is 0.6733 with a 95% confidence interval of (0.4168, 0.9299)

Note that the *p*-values for these probability differences are not the same as the *p*-values for the LS-means comparisons on the logit scale (see Output 8). The results in Output 9 are nonlinear comparisons and the standard errors are estimated using the delta method. Therefore, the results differ as expected.

CONCLUSION

Mixed-model analysis is a complex area of statistics. This paper addressed several areas of confusion in writing code with SAS/STAT software for a mixed model. In addition, the paper addressed the following topics:

- nesting and crossing of SUBJECT= effects
- handling multiple and, potentially, nested SUBJECT= effects in the RANDOM and REPEATED statements
- using the repeated effect in the REPEATED statement
- using PROC PLM for postfitting analysis of a mixed-model interaction
- scoring new data from a mixed model in PROC PLM
- computing differences in event probabilities and confidence intervals using PROC NLMIXED

Using these features should enable you to be more confident in your use of PROC MIXED and PROC GLIMMIX. This paper also enables you to expand your model-fitting toolkit through the use of PROC PLM and PROC NLMIXED.

REFERENCES

Gotway, C. A. and Stroup, W. W. 1997. "A Generalized Linear Model Approach to Spatial Data and Prediction." *Journal of Agricultural, Biological, and Environmental Statistics,* 2, 157–187.

(list continued)

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 Institute Inc.

Littell, R. C., Freund, R. J., and Spector, P. C. 1991. *SAS® System for Linear Models, Third Edition*, Cary, NC: SAS Institute Inc.

ACKNOWLEDGEMENTS

The authors are grateful to Susan Berry of SAS Institute Inc. for her valuable editorial assistance in the preparation of this paper.

CONTACT INFORMATION

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

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

Kathleen Kiernan
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**