Paper SAS5601-2016

Fitting Your Favorite Mixed Models with PROC MCMC

Fang Chen, Gordon Brown, and Maura Stokes, SAS Institute Inc.

Abstract

The popular MIXED, GLIMMIX, and NLMIXED procedures in SAS/STAT® software fit linear, generalized linear, and nonlinear mixed models, respectively. These procedures take the classical approach of maximizing the likelihood function to estimate model parameters. The flexible MCMC procedure in SAS/STAT can fit these same models by taking a Bayesian approach. Instead of maximizing the likelihood function, PROC MCMC draws samples (using a variety of sampling algorithms) to approximate the posterior distributions of model parameters. Similar to the mixed modeling procedures, PROC MCMC provides estimation, inference, and prediction.

This paper describes how to use the MCMC procedure to fit Bayesian mixed models and compares the Bayesian approach to how the classical models would be fit with the familiar mixed modeling procedures. Several examples illustrate the approach in practice.

Introduction

Random-effects models are frequently used in modern data analysis, and the classical approach plays a prominent role in their estimation and inference. In SAS/STAT software, the MIXED, GLIMMIX, and NLMIXED mixed modeling procedures provide a powerful suite that handles a wide range of models. With advances in Bayesian methodology and Bayesian computation methods, the Bayesian paradigm for modeling random-effects model is gaining momentum. SAS® users are increasingly interested in fitting Bayesian random-effects models using the general-purpose simulation-based MCMC procedure. This paper explains how to use the MCMC procedure to fit the mixed models that are commonly fit by using the SAS/STAT mixed modeling procedures.

The linear mixed-effects model is an extension of ordinary linear regression, in which a random-effects term is added to the predictor to represent and model variability among different units (also known as clusters). This class of models can be fit by PROC MIXED. The generalized mixed linear model relates the linear predictor to (typically the mean of) the response variable via a monotone link function, with the response assumed to rise from a member of the exponential family of distributions. This class of models can be fit by PROC GLIMMIX. The nonlinear mixed-effects model, the most general of the three types, relaxes the modeling assumption even further by permitting a nonlinear relationship between effects (both fixed and random) and the response. This class of models can be fit by PROC NLMIXED.

In a classical (frequentist) approach, the fixed-effects parameters are considered fixed with an unknown mean, and the random effects are treated as unobserved latent variables. The primary interest focuses on the fixed effects, and estimation is achieved by maximizing the marginal likelihood of the fixed-effects parameter while integrating out the random effects (Davidian and Giltinan 1995; Vonesh, Chinchilli, and Pu 1996). Typically, asymptotic normality is assumed in inference, based on the integrated likelihood approach.

Bayesian analysis treats all unknown quantities in a statistical model, including both the fixed and random effects, as random variables. The Bayesian paradigm for inference is based on combining various sources of information, such as the prior distribution and information from the data (typically the likelihood). The Bayesian approach offers an efficient and natural solution for data that have a complex structure, especially when a hierarchy is involved. There is no modeling limitation, and you can develop a random-effects model to arbitrary depth to accurately capture the complexity. Bayesian estimation is achieved by repeatedly sampling from the posterior distribution, most often by using the Markov chain Monte Carlo approach (Gelfand et al. 1990), and using these samples to estimate posterior marginal distributions in quantities of interest (or using them for further inference).

Based on linear model theory and the normality assumption, PROC MIXED provides very close approximation to fully Bayesian solutions that use noninformative prior distributions. PROC MCMC offers certain advantages over PROC MIXED; for example, PROC MCMC can fit models that have different prior distributions. In generalized linear and nonlinear mixed-effects models, the Bayesian approach is based on the exact posterior distribution, and inference does not use the plug-in method or rely on large-sample theory or on the normality assumption on the random effects.

These attributes can lead to flexibility in modeling capability and more precise estimates. For some models, such as nested and nonnested models that require high-dimensional and computationally costly integral approximation methods in mixed procedures, using PROC MCMC can be faster. But in many cases, the sampling-based PROC MCMC runs slower than the mixed modeling procedures. The dimension of the regression problem can also hinder convergence in PROC MCMC.

This paper is organized as follows. The section "Notation" introduces mixed-models notation that is used throughout the paper. The sections "Comparing PROC MCMC to the Mixed Modeling Procedures" and "Selecting Prior Distributions" provide some background information. Then the paper turns to a simple model ("Example: Fixed-Effects Model") and then builds on that example to cover more complex cases, such as random-effects models, repeated measures models, models with interactions, models with different covariance types, and generalized linear mixed-effects models. The section "Postprocessing" discusses how to obtain linear combinations of parameters and predictions. The last few sections provide systematic details about coding classification variables and converting continuous and categorical random effects, and they provide references on distributions, links, and group.

Notation

A linear mixed model is written as

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon} \\ \boldsymbol{\gamma} &\sim N(\mathbf{0}, \mathbf{G}) \\ \boldsymbol{\epsilon} &\sim N(\mathbf{0}, \mathbf{R}) \end{aligned}$$

where $\boldsymbol{\beta}$ denotes fixed-effects parameters whose elements are unknown constants, $\boldsymbol{\gamma}$ denotes random effects that have a multivariate normal prior centering at 0 and a covariance matrix $\mathbf{G}$, and $\boldsymbol{\epsilon}$ are the noise, with covariance $\mathbf{R}$. The MIXED procedure refers to $\mathbf{G}$ and $\mathbf{R}$ as covariance matrices for the random effects (G-side) and random errors (R-side), respectively.

In generalized linear mixed models, the linear predictor $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma}$ relates to the mean of the data via a monotone link function: $\mathrm{E}[\mathbf{Y}|\boldsymbol{\gamma}] = g^{-1}(\boldsymbol{\eta}) = g^{-1}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma})$, where $g^{-1}(\cdot)$ is the inverse link function and where the conditional distribution of the data, given $\boldsymbol{\gamma}$, is a member of the exponential family of distributions, including the normal distribution. The distribution of $\boldsymbol{\gamma}$ is assumed to be normal in PROC GLIMMIX. In a nonlinear mixed model, neither the linearity nor the exponential family assumptions are assumed. The fixed effects and the random effects can enter the conditional likelihood nonlinearly. As in PROC MIXED and PROC GLIMMIX, the random effects $\boldsymbol{\gamma}$ can have only normal distributions in PROC NLMIXED.

The inclusion of the R-side covariance matrix in the GLIMMIX or NLMIXED procedures is not straightforward because of nonlinearity and nonnormally distributed data. Estimating the R-side covariance matrix requires linearization of the mixed models.

Comparing PROC MCMC to the Mixed Modeling Procedures

The syntax of PROC MCMC is more similar to PROC NLMIXED than to PROC MIXED and PROC GLIMMIX. You can use succinct syntax to specify complex models in PROC MIXED and PROC GLIMMIX because these procedures do a huge amount of work behind the scenes. They internally create and keep track of many parts of a model—such as the complete design matrix ($\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma}$) and different types of the covariance matrices (both G-side and R-side)—and they provide estimation and inference capabilities. In contrast, when you use PROC MCMC, you are responsible for defining all aspects of the statistical model. For example, to specify a regression model in PROC MCMC, you need to provide all columns of the design matrix (including categorical variables that are quietly handled by the CLASS statement in PROC MIXED and PROC GLIMMIX), the coefficients, and each parameter's prior distribution. To include random effects, you name the effects, their prior distribution, and the desired covariance matrix, and you construct the $\mathbf{Z}\boldsymbol{\gamma}$ part of the model.

One objective of this paper is to present how different syntax in PROC MIXED and PROC GLIMMIX can lead to nuances in statistical models and what you can do to translate these models in order to carry out similar types of inference and prediction in the Bayesian paradigm.

The following steps provide guidance for how you can use PROC MCMC to perform a Bayesian analogy of the analysis that a mixed modeling procedure produces:

1. Determine the likelihood function, depending on the mixed modeling procedure:
   - PROC MIXED: either the normal or the multivariate normal distribution
   - PROC GLIMMIX: depends on the DIST= specification and the LINK= specification (see the sections "Supported Distributions" and "Links")
   - PROC NLMIXED: use the same specification in the MODEL statement
2. Determine the covariates and model parameters. See "Example: Fixed-Effects Model" and the section "Classification Variables" for guidance and suggestions about handling categorical variables. You can start with noninformative prior distributions on model parameters (see the section "Selecting Prior Distributions").
3. Define the random effects in the model, including the following:
   - Create a name for the random effect. (This is a variable name that you choose.)
   - Determine the subject variable. (This subject effect is usually the same as the subject effect used in other mixed modeling procedures, although interaction or nested effects require additional steps.)
   - Determine the dimension of the effects. The random intercept is univariate; random effects that have covariates are multidimensional.
   - Choose a prior distribution. (Use normal for univariate random effects or multivariate normal for multidimensional random effects. You want to specify the prior with mean zero.)
   - Choose a type of covariance (see "Example: Models with Different Covariance Types").
   - Model repeated measurements if needed (see "Example: Repeated-Measures Model").
4. Identify additional modeling requirements. For examples, see the section "GROUP= Option" if the GROUP= option is required, and see the section "DIST=BYOBS" for joint modeling.
5. Run MCMC. Check for convergence and obtain posterior estimates.
6. Include postfitting prediction or inference that corresponds to the LSMEANS, CONTRAST, OUTPUT, or similar statement. See the section "Postprocessing" for more information.

PROC MCMC is a versatile procedure that can fit linear or nonlinear models, nested or non-nested hierarchical models that have arbitrary depth, models that have built-in or nonstandard distributions, models that have multiple response variables, and models that have missing data. You can also use internal optimization routines to find posterior modes, use the PREDDIST statement to perform posterior prediction, and use the UDS statement to incorporate your own sampling algorithm for a subset of the parameters. After the simulation has completed, you can subsequently use any SAS procedure or the DATA step to analyze the posterior sample. For more information, see Chen (2009), Chen (2011), Chen (2013), or the chapter "The MCMC Procedure" in *SAS/STAT 14.1 User's Guide*.

Selecting Prior Distributions

In most mixed-effects models, including normal and generalized linear models, flat priors on the regression coefficients ($\boldsymbol{\beta}$) are considered to be noninformative. The flat prior assigns equal likelihood for all possible values of the parameter, $\pi(\boldsymbol{\beta}) \propto 1$. You use the general distribution to specify a flat prior in PROC MCMC as follows[1]:

```
prior Beta: ~ general(0);
```

In addition to the flat prior, a normal prior that has very large variance is also considered to be noninformative, or weakly-informative. The following statement specifices such a prior in PROC MCMC:

```
prior Beta: ~ normal(0, var=1e6);
```

[1]The flat prior that has support on the entire axis is not one of the standard distributions (such as the normal or the beta distribution) that PROC MCMC supports. You use the general distribution to specify a nonstandard prior distribution for a continuous random variable. The general distribution takes an argument that is the expression for the logarithm of the distribution. In the case of the flat prior, $\log(1) = 0$. Hence you use 0 as the input argument to the general distribution.

The noninformativeness of this normal prior depends on the variance value being sufficiently larger than the posterior variances of all the **Beta** parameters.

A noninformative prior on the variance parameter σ_y^2 in a linear model is a uniform prior on the log scale: $\pi(\log(\sigma_y^2)) \propto 1$. This is also known as Jeffreys' prior in a normal model. Another common choice is a uniform prior on the variance: $\pi(\sigma^2) \propto 1$. In either case, because you usually have enough data to estimate the variance parameter, posterior analysis is not as sensitive to the choice of the uniform prior, whether it is on the original scale or on the log scale.

From the point of view of sampling efficiency, you can use an inverse-gamma prior on σ_y^2. PROC MCMC recognizes this conjugate scenario, and a conjugate sampler is more efficient than a Metropolis-based sampler, which is used if a uniform prior is used. If mixing of the scale parameter becomes an issue, you can use the SLICE option in the PARMS statement that declares the variance parameter. This tells PROC MCMC to use the slice sampler to draw the posterior distribution of that variance parameter. A slice sampler can often improve mixing (Neal 2003).

Selection of the prior on the variance component parameter (denoted as σ_u^2 here) is not as straightforward, and there have been many suggestions about how to select it (Box and Tiao 1973; Chaloner 1987; Gelman 2006). It is important to point out that a uniform distribution on the log scale, $\pi(\sigma_u^2) \propto 1$, leads to an improper posterior distribution and should be avoided in all cases. Some frequently suggested priors and the PROC MCMC statements that specify them include the following:

- $\pi(\sigma_u) \propto 1$

  ```
  prior Sigma ~ general(0, lower=0);
  ```

  where σ_u is the standard deviation. Or $\pi(\sigma_u) \sim \text{uniform}(0, d)$, where d is a predetermined upper bound.
- $\pi(\sigma_u^2) \propto 1$

  ```
  prior S2 ~ general(0, lower=0);
  ```

- $\pi(\sigma_u^2) \sim$ igamma(shape=ϵ, scale=ϵ), with small values of ϵ:

  ```
  prior S2 ~ igamma(shape=0.01, scale=0.01);
  ```

- half-t or half-Cauchy distributions on σ_u:

  ```
  prior Sigma ~ t(0, var=10, df=3, lower=0);
  ```

Box and Tiao (1973) suggest a joint noninformative prior on σ_y^2, σ_u^2 in a balanced mixed model,

$$\pi(\sigma_y^2, \sigma_u^2) \propto \frac{1}{\sigma_y^2(\sigma_y^2 + n\sigma_u^2)}$$

where n is the sample size. You can use the general distribution to specify this joint prior distribution. For more detailed discussions on selection of priors on the variance component parameter, see Daniels (1999) and Gelman (2006).

When you have multivariate random effects, you have a covariance Σ_u instead of a scale variance σ_u^2. An inverse-Wishart prior is desirable in PROC MCMC, which uses a conjugate sampler for the covariance matrix. An inverse-Wishart prior with a small degrees of freedom (interpreted as prior sample size) and small diagonal values of the scale matrix (interpreted as sums of squares of multivariate normal data, XX^T) is considered to be weakly informative. The following statements specify a two-dimensional covariance matrix **Sig_u** and its inverse Wishart prior in PROC MCMC:

```
array Sig_u[2,2];
array S[2,2] (1 0 0 1);
prior Sig_u ~ iwish(3, S);
```

Example: Fixed-Effects Model

This section describes how to use PROC MCMC to fit a fixed-effects model. It demonstrates how you transfer information in the CLASS and MODEL statements of PROC MIXED to PROC MCMC. The data were collected by a pharmaceutical company that examined the effects of three drugs on respiratory capacity of patients with asthma. Treatments involved a standard drug (A), a test drug (C), and a placebo (P). Patients received each treatment on different days. The forced expiratory volume (FEV) was measured hourly for eight hours following treatment, and a baseline FEV was also recorded. Analysis in this section is based on the scenario that each patient received all three treatments on different visits. For more information about this data set (**fev1**), see Littell et al. (2006, Chapter 5).

The first few records of the **fev1** data set are shown as follows. The variables **Fev11h** through **Fev18h** represent the measurements for hours 1 through 18, and **Basefev1** is the baseline measurement.

```
data fev1;
   input Patient Basefev1 Fev11h Fev12h Fev13h Fev14h Fev15h Fev16h Fev17h Fev18h Drug $;
datalines;
 201  2.46  2.68 2.76 2.50 2.30 2.14 2.40 2.33 2.20 a
 202  3.50  3.95 3.65 2.93 2.53 3.04 3.37 3.14 2.62 a
 203  1.96  2.28 2.34 2.29 2.43 2.06 2.18 2.28 2.29 a
 204  3.44  4.08 3.87 3.79 3.30 3.80 3.24 2.98 2.91 a
 205  2.80  4.09 3.90 3.54 3.35 3.15 3.23 3.46 3.27 a
 206  2.36  3.79 3.97 3.78 3.69 3.31 2.83 2.72 3.00 a
 207  1.77  3.82 3.44 3.46 3.02 2.98 3.10 2.79 2.88 a

   ... more lines ...
```

This example uses the first hourly measurement, **Fev11h**, as the response and uses the fixed-effect term **Drug** and continuous variable **Basefev1** as explanatory variables. The regression model is represented as

$$\begin{aligned} \textbf{Fev11h}_i &= \mathbf{X}_i \boldsymbol{\beta} + \epsilon_i \\ \epsilon_i &\sim N(0, \sigma^2) \end{aligned}$$

where $i = 1, \ldots, 72$ in the data set.

You use the following statements to fit a linear regression in PROC MIXED:

```
proc mixed data=fev1;
   class Drug;
   model Fev11h=Drug Basefev1/solution;
run;
```

The results are displayed in Figure 1 (parameter estimates and their standard errors) and Figure 2 (variance estimate) for the regression model:

Figure 1 PROC MIXED Parameter Estimates

Solution for Fixed Effects

Effect	Drug	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept		0.4691	0.2966	68	1.58	0.1185
Drug	a	0.6448	0.1376	68	4.69	<.0001
Drug	c	0.8630	0.1375	68	6.27	<.0001
Drug	p	0	.	.	.	.
Basefev1		0.8900	0.1063	68	8.37	<.0001

Figure 2 PROC MIXED Parameter of Sample Variance

Covariance Parameter Estimates	
Cov Parm	**Estimate**
Residual	0.2269

Next, consider fitting this model with PROC MCMC following the steps listed previously. Because PROC MCMC does not support a CLASS statement, you need to create indicator variables for categorical effects, such as **Drug**. There are several possible approaches; see the section "Classification Variables" for more information. The following DATA step creates indicator variables **Da** and **Dc** to represent the **Drug** effect in the design matrix and stores the variables in the **fev2** data set:

```
data fev2;
   set fev1;
   if Drug='a' then Da=1; else Da=0;
   if Drug='c' then Dc=1; else Dc=0;
run;
```

Begin model construction by specifying the likelihood function, which is the normal distribution. You specify it by using the following MODEL statement with response variable **Fev11h**:

```
model Fev11h ~ normal(Mu, var=S2);
```

The symbols **Mu** and **S2** are variables you create to represent the mean and variance of this normal model. The **Mu** variable is defined as a regression mean by the following programming statement:

```
Mu = B0 + B1*Da + B2*Dc + B3*Basefev1;
```

In this statement, **B0**, **B1**, **B2**, and **B3** are the regression parameters, and **Da** and **Dc** are the covariates (indicator variables) for the categorical variable **Drug**.

There are five parameters in the model, which you declare by using the following PARMS statement, where **B0**–**B3** means **B0**, **B1**, **B2**, and **B3**:

```
parms B0-B3 S2;
```

Next, their prior distributions are specified in the PRIOR statements:

```
prior B: ~ normal(0, var=1e6);
prior S2 ~ igamma(0.01, scale = 0.01);
```

The colon (:) after **B** is a shorthand for all symbols that begin with letter B. A normal prior with large variance is assigned to all regression parameters, and an inverse gamma prior is assigned to the variance. An inverse gamma prior is selected here for computational efficiency because PROC MCMC then uses a conjugate sampler to update the parameter.

Combined, the following PROC MCMC statements request a Bayesian regression analysis:

```
proc mcmc data=fev2 seed=5235 nmc=20000 outpost=outFixed;
   parms B0-B3 S2;
   prior B: ~ normal(0, var=1e6);
   prior S2 ~ igamma(0.01, scale = 0.01);
   Mu = B0 + B1*Da + B2*Dc + B3*Basefev1;
   model Fev11h ~ normal(Mu,var=S2);
run;
```

The SEED= option ensures Markov chain reproducibility, the NMC= option requests a simulation size of 20,000, and the OUTPOST= option saves all the simulated draws in the **outFixed** data set. Before conducting any inference, you must check the convergence diagnostics for all the parameters (for more information, see the chapter "Introduction to Bayesian Analysis Procedures" in *SAS/STAT 14.1 User's Guide*). The trace plots (samples versus the simulation index) can be very useful in assessing convergence. The trace plot for **B1** in Figure 3 shows good mixing of the Markov chain, which traverses the parameter space efficiently and has relatively constant mean and variance over iterations. The trace plot shows no indication of nonconvergence; trace plots for other parameters are similar but not shown.

Figure 3 Trace Plot for B1

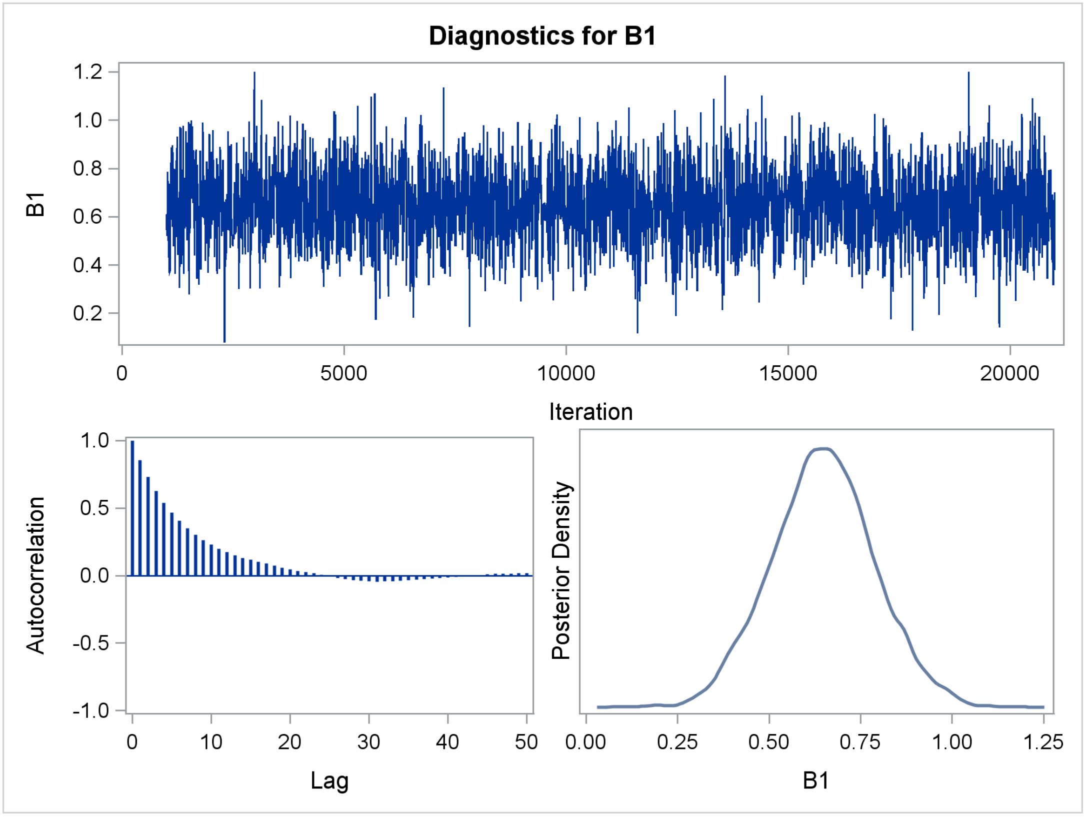

Figure 4 displays the posterior means and standard deviations from PROC MCMC.

Figure 4 PROC MCMC Posterior Summaries

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
B0	20000	0.4752	0.3010	-0.1000	1.0784
B1	20000	0.6476	0.1404	0.3751	0.9251
B2	20000	0.8623	0.1416	0.5912	1.1383
B3	20000	0.8874	0.1077	0.6804	1.1027
S2	20000	0.2338	0.0412	0.1606	0.3183

The posterior estimates are very similar to the point estimates and standard errors that are produced by PROC MIXED (Figure 1 and Figure 2). This result is expected because the prior distributions for the parameters used in the Bayesian analysis are relatively noninformative.

Example: Mixed-Effects Models

This section describes how to use PROC MCMC to fit a mixed-effects model to the **fev1** data set. A mixed-effects model extends the fixed-effects model by including a random-effects term as

$$\begin{aligned} \textbf{Fev11h}_i &= X_i \boldsymbol{\beta} + \gamma_i + \epsilon_i \\ \gamma_i &\sim N(0, \sigma_g^2) \\ \epsilon_i &\sim N(0, \sigma^2) \end{aligned}$$

where γ_i are the **Patient**-level random effects ($i = 1, \ldots, 24$).

You can fit a random-intercepts model in PROC MIXED by adding a RANDOM statement that specifies **Int** as the random effect and **Patient** as the subject:

```
proc mixed data=fev1;
   class Drug Patient;
   model Fev11h=Drug Basefev1/solution;
   random int/subject=Patient;
run;
```

The output for the covariance parameters from PROC MIXED are presented in Figure 5.

Figure 5 PROC MIXED Covariance Parameters

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
Intercept	Patient	0.1475
Residual		0.08654

You use a RANDOM statement in PROC MCMC to specify the random effects, and you use the SUBJECT= option to specify the subject. There is an additional variance parameter, σ_g^2, in the model, which you declare in a PARMS statement, and whose prior you assign in a PRIOR statement. The following statements use PROC MCMC to fit a random-intercepts model:

```
proc mcmc data=fev2 seed=5235 nmc=20000 outpost=outMixed;
   parms B0-B3 S2;
   parms S2g 1;
   prior B: ~ normal(0, var=1e6);
   prior S2 ~ igamma(0.01, scale = 0.01);
   prior S2g ~ general(0, lower=0);
   random Gamma ~ normal(0,var=S2g) subject=Patient;
   Mu = B0 + B1*Da + B2*Dc + B3*Basefev1 + Gamma;
   model Fev11h ~ normal(Mu,var=S2);
run;
```

The second PARMS statement declares the variance parameter **S2g** and assigns it an initial value of 1. The prior distribution for **S2g** is a flat prior on the positive values. The RANDOM statement declares **Gamma** as the random effect, which enters the regression mean function (**Mu**) linearly. Unlike PROC MIXED or PROC GLIMMIX (which include random effects in the model by default), PROC MCMC does not include them automatically, so you must explicitly add the random effects to the regression model.

The posterior summaries from PROC MCMC are presented in Figure 6. Including the random intercept changes the standard deviations for the parameters, but it has little impact on the posterior means.

Figure 6 PROC MCMC Posterior Summaries

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
B0	20000	0.8471	0.4001	0.0548	1.6072
B1	20000	0.6557	0.0894	0.4814	0.8374
B2	20000	0.8692	0.0878	0.6973	1.0446
B3	20000	0.7457	0.1471	0.4776	1.0590
S2	20000	0.0913	0.0201	0.0560	0.1314
S2g	20000	0.1834	0.0783	0.0553	0.3264

The residual variance estimate (**S2**) and the random intercept variance (**S2g**) from PROC MCMC (not shown here) are close to the PROC MIXED estimates that are displayed in Figure 5.

By default, PROC MCMC does not output posterior summary statistics for the random-effects parameters. You can request the display of random-effects parameters by using the MONITOR= option in the RANDOM statement as follows:

```
random Gamma ~ normal(0,var=S2g) subject=Patient monitor=(Gamma);
```

This statement outputs all **Gamma** random-effects parameters (24 in this example). This is similar to the SOLUTION option in the RANDOM statement in PROC MIXED, which produces the solution for the random-effects parameters. You can also display a subset of the parameters:

```
random Gamma ~ normal(0,var=S2g) subject=Patient monitor=(1 to 3);
```

Regardless of whether they are monitored, all the posterior samples are saved by default to the **outMixed** data set, which is specified in the PROC MCMC statement. You can use postprocessing macros, such as the %SUMINT macro to compute summary statistics or the %ESS macro to compute effective sample sizes of the Markov simulation.[2] The following statements compute summary statistics and effective sample sizes for all the random-effects parameters:

```
%sumint(data=outMixed, var=Gamma:);
%ess(data=outMixed, var=Gamma:);
```

You can also use the %TADPlot autocall macro to display convergence and the %Cater autocall macro to create side-by-side comparison plots of the random effects.

Example: Repeated-Measures Model

The examples in this section show you how to perform a repeated measures analysis in PROC MCMC. Based on the previous random-effects model, denote $\mathbf{Y}_{ij} = \{Y_{ij1}, \ldots, Y_{ijK}\}$ to be all K repeated response values from subject i taking treatment j, and denote $\mathbf{X}_{ij}$ to be the vector of covariates from that subject. The following is a linear random-effects model with covariance errors:

$$\begin{aligned}\mathbf{Y}_{ij} &= \mathbf{X}_{ij}\boldsymbol{\beta} + \gamma_i + \boldsymbol{\epsilon}_{ij}\\ \gamma_i &\sim N(0, \sigma_g^2)\\ \boldsymbol{\epsilon}_{ij} &\sim N(0, \mathbf{R})\end{aligned}$$

The REPEATED statement in PROC MIXED specifies R-side covariance parameters, which model dependency between observations. The same functionality is handled in PROC GLIMMIX by the RANDOM statement and the _RESIDUAL_ keyword.

To fit this repeated measurement model in PROC MIXED, you need a data set that has been rolled out so that each response is in a separate observation. The following DATA step creates data set **fev1uni** accordingly:

```
data fev1uni; set fev1;
   array f{8} fev1:;
   do hour = 1 to 8; fev = f{hour}; output; end;
   drop fev1:;
   rename fev=fev1;
run;
```

The following PROC MIXED statements request a repeated measurements model that has unstructured covariance:

```
proc mixed data=fev1uni;
   class Drug Hour Patient;
   model Fev1=Drug Hour Basefev1 / solution;
   random int/subject=Patient s;
   repeated /subject=patient*drug type=un;
run;
```

These statements also include the categorical **Hour** variable in the regression model. For each subject i on treatment j, the eight repeated measures are represented by the following model, where **Hour**$_k$ for $k = 1, \ldots, 7$ are indicator

[2]The %SUMINT and %ESS macros are SAS autocall macros. For a complete list of MCMC-related postprocessing macros and their usages, see the chapter "The MCMC Procedure" in *SAS/STAT 14.1 User's Guide*.

variables (which take a value of 1) and **Hour**$_8 = 0$:

$$\begin{aligned}
\textbf{Fev11h}_{ij1} &= \beta_0 + \beta_1 \cdot \textbf{Drug}_a + \beta_2 \cdot \textbf{Drug}_c + \beta_3 \cdot \textbf{Basefev1} + \beta_{h1} \cdot \textbf{Hour}_1 + \gamma_i + \epsilon_i \\
&\vdots \\
\textbf{Fev17h}_{ij7} &= \beta_0 + \beta_1 \cdot \textbf{Drug}_a + \beta_2 \cdot \textbf{Drug}_c + \beta_3 \cdot \textbf{Basefev1} + \beta_{h7} \cdot \textbf{Hour}_7 + \gamma_i + \epsilon_i \\
\textbf{Fev18h}_{ij8} &= \beta_0 + \beta_1 \cdot \textbf{Drug}_a + \beta_2 \cdot \textbf{Drug}_c + \beta_3 \cdot \textbf{Basefev1} + \gamma_i + \epsilon_i
\end{aligned}$$

PROC MCMC assumes that observations are independent and offers no option that enables you to directly model residual errors among observations within subjects. To model within-subjects observation covariance structures, you want to work with an input data set that stores all repeated measurements from a subject in one row (such as **fev1**). When the data are stored in that format, you can specify a multivariate normal (MVN) distribution in the MODEL statement to model all repeated measurements within the same subject.

The MVN distribution can be applied only to array-based variables. Therefore, you must first store all response variables (**Fev11h** through **Fev18h**) in an array of size 8, as in the following statements:

```
array y[8] Fev11h Fev12h Fev13h Fev14h Fev15h Fev16h Fev17h Fev18h;
model Y ~ mvn(Mu, Cov);
```

In these statements, **Mu** is an eight-dimensional array whose elements correspond to each of the regression means, and **Cov** is the R-side 8-by-8 covariance matrix.

The following statements fit a Bayesian random-effects model that has repeated measurements in PROC MCMC:

```
proc mcmc data=fev2 seed=5235 nmc=20000 outpost=OutRep monitor=(B0-B3 H1-H7 S2g);
   array Y[8] Fev11h Fev12h Fev13h Fev14h Fev15h Fev16h Fev17h Fev18h;
   array Mu[8];
   array Cov[8,8];
   array S[8,8];
   array H[8] H1-H7 0;

   parms B0-B3 Cov;
   parms H1-H7;
   parms S2g 1;
   prior B: H: ~ normal(0, var=1e6);
   prior Cov ~ iwish(8,S);
   prior S2g ~ general(0, lower=0);

   begincnst;
      call identity(S);
   endcnst;
   random Gamma ~ normal(0,var=S2g) subject=Patient;
   Mn = B0 + B1*Da + B2*Dc + B3*Basefev1 + Gamma;
   call addmatrix(H, Mn, Mu);
   model Y ~ mvn(Mu, Cov);
run;
```

The array **Y** stores the eight response variables, and the **Mu** and **Cov** arrays are the means and covariance matrices, respectively, of the likelihood function. The **S** array is the inverse Wishart scale matrix (which is set to be the identity matrix in the preceding statements), and the **H** array stores the regression coefficients for the **Hour** indicator variables. The last element in the **H** array is zero, which corresponds to eliminating the last level of the **Hour** variable.

The BEGINCNST and ENDCNST statements define a block within which PROC MCMC processes the programming statements only during the setup stage of the simulation. Statements that do not need to be evaluated during the simulation should be put within this block. The CALL ADDMATRIX routine performs an elementwise addition, adding **Mn** to each element in the array **H**, and then stores the result in **Mu**. This is the same operation as the following loop:

```
do i = 1 to 8;
   Mu[i] = H[i] + Mn;
end;
```

Figure 7 displays the posterior summaries for these parameters. The posterior estimates are close to those produced by PROC MIXED.

Figure 7 PROC MCMC Posterior Summary

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
B0	20000	0.8791	0.3918	0.1146	1.6415
B1	20000	0.3757	0.0836	0.2105	0.5316
B2	20000	0.5624	0.0864	0.3898	0.7304
B3	20000	0.6269	0.1409	0.3358	0.8835
H1	20000	0.4677	0.0618	0.3533	0.5929
H2	20000	0.4420	0.0560	0.3305	0.5517
H3	20000	0.3555	0.0512	0.2586	0.4523
H4	20000	0.2557	0.0514	0.1511	0.3514
H5	20000	0.1580	0.0409	0.0813	0.2387
H6	20000	0.0948	0.0356	0.0281	0.1661
H7	20000	0.0128	0.0414	-0.0665	0.0968
S2g	20000	0.2209	0.0879	0.0870	0.3983

Example: Model with Interactions

This section illustrates how to include interaction terms in your model. One approach for adding interaction terms is to construct the design matrix; see the section "Classification Variables" for an example. Another approach is described here, where a repeated measurement model (which is described in "Example: Repeated-Measures Model") takes on a cross product of two main effects, **Drug** and **Hour**.

For the ith patient on treatment j at hour k, you have the following regression model:

$$\begin{aligned}\textbf{Fev1kh}_{ijk} &= \beta_0 + \beta_1 \cdot \textbf{Drug}_a + \beta_2 \cdot \textbf{Drug}_c + \beta_3 \cdot \textbf{Basefev1} \\ &\quad + \beta_{hk} \cdot \textbf{Hour}_k + \beta_{hak} \cdot \textbf{Drug}_a \cdot \textbf{Hour}_k + \beta_{hck} \cdot \textbf{Drug}_c \cdot \textbf{Hour}_k + \gamma_i + \epsilon_i\end{aligned}$$

This model introduces 14 more parameters, $\beta_{ha1} \ldots \beta_{ha7}$ and $\beta_{hc1} \ldots \beta_{hc7}$, where $\beta_{ha8} = \beta_{hc8} = 0$ to avoid singularity.

The following PROC MIXED call adds **Drug** $\times$ **Hour** to the repeated measurement model:

```
proc mixed data=fev1uni;
   class Drug Hour Patient;
   model Fev1=Drug|Hour Basefev1/solution;
   random int/subject=Patient;
   repeated /subject=Patient(Drug) type=un r;
 run;
```

To translate the same model to PROC MCMC, each mean (`mu[i]`) of the **Fev1kh** response requires two additional terms, $\beta_{hak} \cdot$ **Da** and $\beta_{hbk} \cdot$ **Dc**, where **Da** and **Dc** are indicator variables for **Drug**. Again, the model takes advantage of a fact that is intrinsic to the design matrix for this model: **Hour**$_k$ is 1 for each k.

Similar to how the **H** array is added in the repeated measurement model without interaction terms, you allocate two additional arrays that store the β_{hak} and β_{hbk} parameters.

The following statements fit a Bayesian repeated measurement random-effects model with main effects and interactions:

```
proc mcmc data=fev2 seed=5235 nmc=20000 monitor=(ha1-ha7 hc1-hc7) outpost=OutInter;
   array Y[8] Fev11h Fev12h Fev13h Fev14h Fev15h Fev16h Fev17h Fev18h;
   array Mu[8];
   array Cov[8,8];
   array S[8,8];
```

```
      array H[8] H1-H7 0;
      array Ha[8] Ha1-Ha7 0;
      array Hc[8] Hc1-Hc7 0;

      parms B0-B3 cov;
      parms H1-H7;
      parms Ha1-Ha7 Hc1-Hc7;
      parms S2g 1;
      prior B: H: ~ normal(0, var=1e6);
      prior Cov ~ iwish(8,S);
      prior S2g ~ general(0, lower=0);

      begincnst;
         call identity(S);
      endcnst;
      random Gamma ~ normal(0,var=S2g) subject=Patient;
      Mn = B0 + B1*Da + B2*Dc + B3*Basefev1 + Gamma;
      do i = 1 to 8;
         Mu[i] = Mn + H[i] + Da*Ha[i] + Dc*Hc[i];
      end;
      model Y ~ mvn(Mu, Cov);
   run;
```

Two ARRAY statements create arrays of the new β parameters, and the DO loop adds the interaction terms to the regression means. The MONITOR= option displays posterior estimates of β_{hak} and β_{hck} parameters, as shown in Figure 8.

Figure 8 PROC MCMC Posterior Summaries

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
Ha1	20000	0.5566	0.1335	0.3055	0.8234
Ha2	20000	0.3849	0.1317	0.1182	0.6325
Ha3	20000	0.1877	0.1148	-0.0504	0.3913
Ha4	20000	0.0738	0.1175	-0.1611	0.2820
Ha5	20000	0.1743	0.0973	-0.0312	0.3675
Ha6	20000	0.0428	0.0848	-0.1158	0.2104
Ha7	20000	-0.0418	0.0913	-0.2117	0.1344
Hc1	20000	0.6224	0.1199	0.3976	0.8691
Hc2	20000	0.4657	0.1153	0.2449	0.7158
Hc3	20000	0.4354	0.1110	0.2214	0.6521
Hc4	20000	0.3206	0.1125	0.1147	0.5396
Hc5	20000	0.2010	0.0949	0.0283	0.3694
Hc6	20000	0.00384	0.0879	-0.1565	0.1809
Hc7	20000	-0.0717	0.1018	-0.2708	0.1276

In Figure 8, the highest posterior density (HPD) intervals that are associated with the interaction parameters largely do not contain 0, implying that the **Drug** $\times$ **Hour** interaction is meaningful.

One advantage of simulation-based Bayesian computation is that you can use posterior samples to estimate many probabilities of interest, including joint probabilities. Suppose you are interested in estimating the probability that all interaction terms are positive—that is, Pr($\beta_{hak} > 0, \beta_{hck} > 0$|Data) for all k. All you need to do is to step through the posterior samples and count the proportion of times that these positive interaction terms occur. This is demonstrated in the following DATA step:

```
   data prob;
      set OutInter;
      array Beta[14] Ha1-Ha7 Hc1-Hc7;
      Positive = 1;
      do i = 1 to 14;
         Positive = Positive*(Beta[i] > 0);
      end;
   run;
   %sumint(data=prob, var=positive);
```

The DATA step counts the number of times, out of 20,000 draws, that all **Ha** and **Hc** samples are positive. The ratio is the expected probability that such a joint event takes place, and you can use the %SUMINT postprocessing macro to compute the statistics. The summary statistics are shown in Figure 9. The estimated joint probability is around 7%.

Figure 9 Estimate of Pr($\beta_{hak} > 0, \beta_{hck} > 0$|Data) for All k

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
positive	20000	0.0707	0.2562	0	1.0000

Example: Models with Different Covariance Types

Previous sections illustrated how to use the MVN distribution to fit data that have unstructured covariance matrices. An inverse Wishart prior on the covariance matrix ensures sampling efficiency because PROC MCMC uses the conjugate sampler to draw the unstructured covariance. In PROC MCMC, you can use a covariance matrix with an inverse Wishart prior to model either the G-side covariance on the random effects (in the RANDOM statement) or the R-side covariance matrix on the data (in the MODEL statement). However, PROC GLIMMIX and PROC MIXED support a wide range of covariance structures. This section describes how you can work with different types of covariance structures in PROC MCMC.

First-Order Autoregressive Covariance

AR(1) is the first-order autoregressive covariance structure with the following specification:

$$\sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$$

For a particular row (r) and column (c), the element in the covariance matrix $\mathbf{\Sigma}$ is defined as $\sigma^2\rho^{(|r-c|)}$. There are only two parameters in this model, σ^2 and ρ.

You use the following statements to fit an AR(1) covariance to the R-side of a model in PROC MIXED:

```
   proc mixed data=fev1uni;
      class Drug Hour Patient;
      model Fev1=Drug|Hour /solution;
      random int/subject=Patient;
      repeated Hour/subject=Patient(Drug) type=ar(1);
   run;
```

To translate the model, use the MVNAR distribution in PROC MCMC. The MVNAR distribution is a multivariate normal distribution that has an AR(1) covariance matrix. The distribution takes on three parameters: an array mean vector, a variance parameter (σ^2), and a correlation parameter (ρ). Instead of specifying an inverse Wishart prior on a covariance matrix, you define two additional parameters and assign priors accordingly.

The following PROC MCMC statements fit a repeated-measures model that has an AR(1) covariance structure:

```
   proc mcmc data=fev2 seed=5235 nmc=20000 monitor=(s2 rho) outpost=OutAR1;
      array Y[8] Fev11h Fev12h Fev13h Fev14h Fev15h Fev16h Fev17h Fev18h;
      array Mu[8];
      array H[8] H1-H7 0;
      array Ha[8] Ha1-Ha7 0;
      array Hc[8] Hc1-Hc7 0;
      parms B0-B3 S2;
      parms H1-H7;
      parms Ha1-Ha7 Hc1-Hc7;
      parms S2g 1;
      parms Rho;
      prior B: H: ~ normal(0, var=1e6);
      prior S2g ~ general(0, lower=0);
      prior S2 ~ igamma(shape=0.01, scale=0.01);
      prior Rho ~ uniform(-1, 1);

      random Gamma ~ normal(0,var=S2g) subject=Patient;
      Mn = B0 + B1*Da + B2*Dc + B3*Basefev1 + Gamma;
      do i = 1 to 8;
         Mu[i] = Mn + H[i] + Da*Ha[i] + Dc*Hc[i];
      end;
      model Y ~ mvnar(Mu, var=S2, rho=Rho);
   run;
```

The **Rho** parameter has a uniform prior between –1 and 1, which ensures stationarity in the AR(1) covariance structure.

Figure 10 shows the posterior summary for the covariance parameters from PROC MCMC for the AR(1) model.

Figure 10 PROC MCMC AR(1) Posterior Estimates

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
S2	20000	0.1133	0.0117	0.0919	0.1369
Rho	20000	0.6591	0.0349	0.5843	0.7210

Figure 11 displays the covariance parameters from PROC MIXED. The AR(1) parameter in PROC MIXED represents the correlation and is similar to **Rho** from PROC MCMC.

Figure 11 PROC MIXED AR(1) Covariance Parameters

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
Intercept	Patient	0.3831
AR(1)	Patient(Drug)	0.6629
Residual		0.1130

Compound Symmetry Type

PROC MCMC does not provide built-in distributions for other covariance types, so you need to use programming statements to construct the covariance structure. This section shows how to work with compound symmetry (CS).

A compound symmetry covariance matrix has two parameters, σ and ϕ, which are specified as follows:

$$\mathrm{Cov}\left[y_i, y_j\right] = \begin{cases} \phi + \sigma & i = j \\ \sigma & i \neq j \end{cases}$$

You use the MVN distribution to specify a covariance matrix, and the elements of the covariance matrix must be filled in according to the covariance specification.

Use the following PROC MCMC statements to request the repeated-measures model with a CS covariance structure:

```
proc mcmc data=fev2 seed=5235 nmc=20000 monitor=(sigma phi)
    outpost=OutAR1 propcov=quanew;
   array Y[8] Fev11h Fev12h Fev13h Fev14h Fev15h Fev16h Fev17h Fev18h;
   array Mu[8];
   array H[8] H1-H7 0;
   array Ha[8] Ha1-Ha7 0;
   array Hc[8] Hc1-Hc7 0;
   array Cov[8,8];

   parms B0-B3;
   parms H1-H7;
   parms Ha1-Ha7 Hc1-Hc7;
   parms S2g 1 Sigma 1 Phi 1 / slice;
   prior B: H: ~ normal(0, var=1e6);
   prior S2g Sigma Phi ~ general(0, lower=0);

   random Gamma ~ normal(0,var=S2g) subject=Patient;
   Mn = B0 + B1*Da + B2*Dc + B3*Basefev1 + Gamma;
   do i = 1 to 8;
      Mu[i] = Mn + H[i] + Da*Ha[i] + Dc*Hc[i];
   end;

   beginnodata;
      do i = 1 to 8;
         do j = 1 to 8;
            Cov[i,j] = Sigma + Phi * (i = j);
         end;
      end;
   endnodata;
   model Y ~ mvn(Mu, Cov);
run;
```

The SLICE option invokes the slice sampler (Neal 2003) on the three variance parameters in the model. Although the slice sampler is relatively computationally intensive, it can often improve mixing, especially on truncated parameters such as the variance parameters. The BEGINNODATA and ENDNODATA statement block limits redundant calculation of the **Cov** matrix at each simulation step; PROC MCMC invokes this double DO loop calculation only twice while it steps through the input data set, at the first and the last observation.

Figure 12 shows the posterior summaries for the covariance parameters. The relatively large value for **Sigma** compared to **Phi** indicates a high level of within-subject correlation.

Figure 12 PROC MCMC CS Posterior Summary

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
Sigma	20000	0.0613	0.0158	0.0324	0.0920
Phi	20000	0.0637	0.00418	0.0558	0.0719

Generally speaking, you need to use programming statements in PROC MCMC to construct the covariance matrix for most covariance types that are supported in PROC MIXED and PROC GLIMMIX. Some covariance matrix types can result in a large number of parameters, so you want to pay close attention to the specification of prior distributions for these parameters.

Example: Generalized Linear Mixed-Effects Model

This section illustrates how to fit a generalized linear mixed-effects model by using PROC MCMC. McCullagh and Nelder (1989, Chapter 14.5) describe a mating experiment that involves two geographically isolated populations of mountain dusky salamanders. One goal of the experiment was to determine whether barriers to interbreeding have evolved as a result of the geographical isolation of the populations. The experiment involved 40 animals: 20 rough butt

(R) and 20 whiteside (W) salamanders, each group having equal numbers of males and females. The animals were grouped into two sets of R males, two sets of R females, two sets of W males, and two sets of W females, so that each set consisted of five salamanders. Each set was mated against one rough butt and one whiteside set, creating eight crossings. Within the pairings of sets, each female was paired to three male animals.

The following statements show the first few observations of the input data set **Salamander**:

```
data Salamander;
   input day fpop$ fnum mpop$ mnum mating;
   datalines;
4  rb  1 rb  1 1
4  rb  2 rb  5 1
4  rb  3 rb  2 1
4  rb  4 rb  4 1
4  rb  5 rb  3 1
4  rb  6 ws  9 1
4  rb  7 ws  8 0

   ... more lines ...
```

The first observation indicates that rough butt female 1 was paired in the laboratory on day 4 of the experiment with rough butt male 1, and the pair mated. On the same day, rough butt female 7 was paired with whiteside male 8, but the pairing did not result in mating of the animals.

This model can be fitted using a logistic regression with two random effects:

$$p_i = \frac{1}{1+\exp(-(\beta_0 + \beta_1 \cdot \textbf{Fpop}_{\text{rb}} + \beta_2 \cdot \textbf{Mpop}_{\text{rb}} + \beta_3 \cdot \textbf{Fpop}_{\text{rb}} \cdot \textbf{Mpop}_{\text{rb}} + \gamma_f + \gamma_m))}$$
$$\gamma_f \sim N(0, \sigma_f^2)$$
$$\gamma_m \sim N(0, \sigma_m^2)$$

This represents a logistic regression model that has main effects, an interaction, one female group effect (for $f = 1, \ldots, 20$), and one male group effect (for $m = 1, \ldots, 20$). You can use the following statements to fit the model in PROC GLIMMIX:

```
proc glimmix data=salamander;
   class Fpop Fnum Mpop Mnum;
   model Mating(event='1') = Fpop|Mpop / dist=binary solution;
   random int/subject=Fpop*Fnum;
   random int/subject=Mpop*Mnum;
run;
```

To fit a generalized linear model in PROC MCMC, you first specify the likelihood and the link function. See the sections "Supported Distributions" and "Links" for a complete list of distributions and links that are supported in PROC GLIMMIX and their corresponding coding in PROC MCMC. In this example, the likelihood function is binary and the link function is logit.

The RANDOM statements in PROC GLIMMIX take on two interaction effects: `Fpop*Fnum` and `Mpop*Mnum`. The SUBJECT= option in the RANDOM statement in PROC MCMC does not support a syntax for interaction effects. To fit the same subjects in PROC MCMC, you need to create a new subject variable in the input data set (see the section "Random Statement"). The following DATA step concatenates the **Fpop** and the **Fnum** variables to create a new subject, and it does the same thing with the **Mpop** and **Mnum** variables:

```
data salamander;
   set salamander;
   Fpop_fnum = compress(trim(Fpop)||'_'||put(Fnum, 2.));
   Mpop_mnum = compress(trim(Mpop)||'_'||put(Mnum, 2.));
run;
```

The following PROC MCMC statements fit a logistic mixed-effects model that has two random effects:

```
proc mcmc data=salamander nmc=30000 seed=5235 outpost=outgmx propcov=quanew;
   parms B0-B3;
   parms S2f 1 S2m 1;
   prior B: ~ normal(0, var=1e6);
   prior S2f S2m ~ general(0, lower=0);
   Mu = B0 + B1*(Fpop = 'rb') + B2*(Mpop = 'rb') + B3*(Fpop = 'rb')*(Mpop = 'rb') ;
   random F ~ normal(0, var=S2f) subject=Fpop_fnum;
   random M ~ normal(0, var=S2m) subject=Mpop_mnum;
   P=logistic(Mu + M + F);
   model Mating ~ binary(P);
run;
```

In the PROC MCMC program, the categorical variables **Fpop** and **Mpop** are coded using programming statements. This step could have been done in a preprocessing DATA step, as in previous examples.

Figure 13 displays the posterior means and standard deviations from PROC MCMC.

Figure 13 PROC MCMC Posterior Summaries

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
B0	30000	1.6498	1.1139	-0.3642	4.0476
B1	30000	-0.4213	1.2468	-2.6640	2.3282
B2	30000	-3.9596	1.4113	-6.8399	-1.3843
B3	30000	4.6274	1.5107	1.5322	7.4726
S2f	30000	5.5312	4.1934	0.7139	13.5365
S2m	30000	2.1601	2.2810	0.0368	5.9147

Postprocessing

After you fit a model in PROC GLIMMIX, you are often interested in performing postprocessing inference or prediction. You would use the LSMEANS, ESTIMATE, CONTRAST, and other statements to construct linear combinations of model parameters for the purpose of performing hypothesis testing. Or you could use the OUTPUT statement to do prediction, based on the fitted model. This section illustrates how you can use output from PROC MCMC to carry out equivalent inference and prediction.

LSMEANS and Related Statements

The LSMEANS statement computes least squares means of fixed effects, $\mathbf{L}\widehat{\boldsymbol{\beta}}$, where $\widehat{\boldsymbol{\beta}}$ is the maximum likelihood estimate of the regression coefficients that are produced by PROC GLIMMIX. In the Bayesian paradigm, instead of using plug-in methods, you compute linear transformations of the sampled $\boldsymbol{\beta}$ draws in order to estimate the posterior distribution of $\mathbf{L}\boldsymbol{\beta}$ given the data. If you know what the coefficient matrix **L** is, you can use it in a DATA step. If not, you can call PROC GLIMMIX, output the **L** matrix to a data set, and then proceed.

The following statements generate the least squares means for the **Fpop** $\times$ **Mpop** interaction from PROC GLIMMIX:

```
ods output Glimmix.LSMeans.Coef=lsmcoef;
proc glimmix data=salamander;
   class Fpop Fnum Mpop Mnum;
   model Mating(event='1') = Fpop|Mpop / dist=binary solution;
   random int/subject=Fpop*Fnum;
   random int/subject=Mpop*Mnum;
   lsmeans Fpop*Mpop/ilink e;
run;
```

The E option in the LSMEANS statement produces a least squares means coefficient matrix **L**, which is stored in the **lsmcoef** data set. You want to clean up the **L** matrix first by getting rid of reference cells and transposing the data set as follows:

```
data tmp;
   set lsmcoef;
   if Fpop='ws' or Mpop='ws' then delete;
proc transpose data=tmp out=lsmcoefT(drop=_name_);
   var Row:;
   id Effect;
run;
```

The resulting coefficient matrix is shown in Figure 14.

Figure 14 Coefficient Matrix **L**

Obs	Intercept	fpop	mpop	fpop_mpop
1	1	1	1	1
2	1	1	0	0
3	1	0	1	0
4	1	0	0	0

The following statements step through the **outgmx** data set, compute the $\mathbf{L}\boldsymbol{\beta}$ product for every draw of the fixed-effects parameters, and store the products in the **Lbeta1** to **Lbeta4** variables:

```
data BayesLSMeans;
   set outgmx;
   array Beta[4] B0 B1 B2 B3;
   do i = 1 to 4;
     set lsmcoefT point=i;
     array L[*] Intercept Fpop Mpop Fpop_mpop;
     array LBeta[4];
     LBeta[i] = 0;
     do j = 1 to 4;
        LBeta[i] = LBeta[i] + L[j] * Beta[j];
     end;
   end;
   output;
   keep lbeta:;
run;
```

You use the following postprocessing %SUMINT macro to compute the posterior summaries and intervals of these least squares means estimates:

```
%SUMINT(data=BayesLSMeans, var=Lbeta:, alpha=0.05);
```

Figure 15 shows posterior estimates for the Bayesian least squares means.

Figure 15 Posterior-Adjusted Treatment Means

Posterior Summaries and Intervals

Parameter	N	Mean	Standard Deviation	95% HPD Interval	
LBeta1	30000	1.8964	1.0515	0.0347	4.1060
LBeta2	30000	1.2286	0.9424	-0.6256	3.0823
LBeta3	30000	-2.3098	1.2685	-4.8492	-0.0197
LBeta4	30000	1.6498	1.1139	-0.3642	4.0476

You can also include the $\mathbf{L}\boldsymbol{\beta}$ computation in a PROC MCMC program, which could be easier because PROC MCMC supports a number of call functions for array-based computations. For a complete list of matrix functions in PROC MCMC, see the chapter “The MCMC Procedure” in *SAS/STAT 14.1 User’s Guide*. You read the **lsmcoefT** data set in an array and use the CALL MULT routine to compute the statistics.

To compute $\mathbf{L}\boldsymbol{\beta}$, you would add the following statements to the PROC MCMC program that was introduced in “Example:

Generalized Linear Mixed-Effects Model":

```
array Beta[4] B0-B3;
array L[1] / nosymbols;
array LBeta[4];
begincnst;
  rc = read_array('lsmcoefT', L);
endcnst;
beginnodata;
  call mult(L, Beta, LBeta);
endnodata;
```

The READ_ARRAY function stores the **lsmcoefT** data set in a dynamic **L** array. This function is called within the BEGINCNST and ENDCNST block. The CALL MULT routine computes $\mathbf{L}\boldsymbol{\beta}$ and stores the result in the **Lbeta** array. You also want to specify MONITOR=(LBeta) in the PROC MCMC statement to output the linear combinations. This produces results that are identical to those shown in Figure 15.

The GLIMMIX and the MIXED procedures support other statements (for example, the LSMESTIMATE statement) that provide mechanisms for obtaining custom hypothesis tests, which are linear combinations of the fixed-effects parameters. You can follow a similar approach: use the E option in any statement that performs custom hypothesis testing to obtain the **L** matrix, use the DATA step to postprocess the posterior draws or to compute the statistics inside a PROC MCMC call, and monitor the output.

Predictions

In PROC GLIMMIX, you use the OUTPUT statement to create a data set that contains predicted values and residual diagnostics, which are based on plug-in estimates of the fixed and random effects. You can request a linear predictor ($\widehat{\eta} = \mathbf{x}'\widehat{\boldsymbol{\beta}} + \mathbf{z}'\widehat{\boldsymbol{\gamma}}$), a marginal linear predictor ($\widehat{\eta}_m = \mathbf{x}'\widehat{\boldsymbol{\beta}}$), a predicted mean ($g^{-1}(\widehat{\eta})$), a marginal mean ($g^{-1}(\widehat{\eta}_m)$), and respective standard errors. For example, the following statements output the predicted mean and standard errors, and save them to the **gmxout** data set:

```
proc glimmix data=salamander;
   class Fpop Fnum Mpop Mnum;
   model Mating(event='1') = Fpop|Mpop / dist=binary solution;
   random int/subject=Fpop*Fnum;
   random int/subject=Mpop*Mnum;
   output out=gmxout pred(ilink) stderr(ilink);
run;
```

You use the PREDDIST statement in PROC MCMC to draw samples from a posterior predictive distribution, $f(\mathbf{Y}_{\text{pred}}|\mathbf{Y}, \mathbf{X})$, which is a marginal distribution whose parameters are integrated out with respect to the posterior distribution. Instead of one predicted mean and one standard error for each data point, you get many samples from a predictive distribution for each data point. Adding the following statement to any PROC MCMC program in this paper enables you to obtain predictive draws for each observation:

```
preddict outpred=mcmc_out;
```

The **mcmc_out** data set has as many variables as the input data set has rows, where each variable represents the posterior draws for a corresponding response value. By default, the PREDDIST statement produces draws for the fitted values. You can either request summary statistics in the PREDDIST statement or use any of the postprocessing summary macros to compute relevant posterior statistics.

The PREDDIST statement cannot be used to compute predicted statistics, such as the linear predictor $\eta = \mathbf{x}'\boldsymbol{\beta} + \mathbf{z}'\boldsymbol{\gamma}$. To compute η or similar statistics, you can use the DATA step or SAS/IML® software. This computation can be based either on the posterior mean estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ or on the posterior samples of these parameters. The former approach produces point estimates, and the latter approach produces distributions of the predicted values. Another option is to include the calculation of the predicted statistics in PROC MCMC statements and use the MONITOR= option to output the linear predictors.

There are a few ways to use PROC GLIMMIX to compute predicted values for new observations. One of the most commonly used approaches is to add observations to the input data set that you use to fit the model. New observations should have missing response values, which will be ignored by PROC GLIMMIX when it fits the model. The OUTPUT statement produces predicted values for every observation, including those that have missing responses.

In PROC MCMC, you can use the same approach of augmenting the input data set to obtain predictive samples. Alternatively, you can save the new observations in a different data set and use the PREDDIST statement in PROC MCMC. Suppose the new observations are saved in the **Pred** data set. The following statement requests prediction based on the new covariates in PROC MCMC:

```
preddist outpred=mcmc_out covariates=Pred;
```

The **mcmc_out** data set is as wide as the **pred** data set is tall; each variable represents predictive draws from a corresponding observation in the prediction data set.

Classification Variables

This section describes how to handle classification (categorical) variables in PROC MCMC. Because PROC MCMC does not support a CLASS statement, you must handle these variables manually in one of the two ways: The first is a preprocessing approach, in which you modify the input data set to create a design matrix (which contains indicator variables) and use the new design matrix as the input data set to a PROC MCMC call. The second is an in-procedure approach, in which you use a feature of the RANDOM statement to handle categorical variables. These approaches are described in the following subsections. Both approaches use a **Class** data set that contains a categorical variable, **Cluster**, which has four levels and is generated by the following DATA step:

```
data Class;
   array c[4] $ ('A' 'B' 'C' 'D');
   call streaminit(1207);
   do i = 1 to 100;
      ci = rand('table', 0.25, 0.25, 0.25, 0.25);
      cluster = c[ci];
      y = rand('normal');
      center = mod(i, 3);
      output;
   end;
   drop c1-c4 ci;
run;
```

Preprocessing Approach

If you know how to fit a model using PROC GLIMMIX, you can use the OUTDESIGN= option to create the design matrix and use that design matrix as the input data set to PROC MCMC. For example, the following statements use PROC GLIMMIX to produce the design matrix of a one-level random-effects model:

```
proc glimmix data=class outdesign(names)=InputMCMC nofit;
   class Cluster Center;
   model Y = Cluster;
   random intercept / subject=Center;
run;
```

The OUTDESIGN= option outputs the **X** and **Z** matrices to the **InputMCMC** data set. The NOFIT option creates the design matrix without fitting the model. The **InputMCMC** data set includes the following variables:

```
OBS _X1 _X2 _X3 _X4 _X5 _Z1
  1  1   0   1   0   0   1
  2  1   1   0   0   0   1
  3  1   1   0   0   0   1
...
```

You can use these covariates in PROC MCMC. The design matrix that PROC GLIMMIX produces includes all levels. If you want to treat one (for example, **_X5**) as the baseline and exclude it in the regression model in order to avoid singularity, you can use the following statements:

```
parms B1-B4;
prior B: ~ n(0, sd=100);
random Gamma ~ n(0, var=s2) subject = Center;
Eta = B1 + B2*_X2 + B3*_X3 + B4 * _X4 + Gamma;
```

In these statements, **Gamma** is the intercept random effect, **Eta** is the regression mean, and **S2** a variance parameter. The variables **_X1** and **_Z1** are 1's (intercept terms) that are omitted in the PROC MCMC program.

There are other ways to create indicator variables, including using the DATA step (for example, a series of IF ELSE-THEN statements), SAS/IML, or the GLMSELECT procedure or the TRANSREG procedure in SAS/STAT. Some of the procedures create variable names that can be more informative than the names that PROC GLIMMIX provides.

In-Procedure Approach

The in-procedure approach uses the RANDOM statement to create coefficients for categorical variables in a model. In mixed modeling procedures, random effects are interpreted differently than fixed effects are (the fixed effects are parameters, and the random effects are not). But in the Bayesian paradigm, all unknown quantities are parameters, and there is no distinction between the fixed and random effects. In the in-procedure approach, the trick is to use the RANDOM statement to create parameters for the categorical variables.

By default, the RANDOM statement creates a random-effects parameter for every unique value in the SUBJECT= variable. When the SUBJECT= variable is a classification variable that has J categories, the statement creates J parameters, where each observation matches its corresponding parameter. This is identical to creating coefficients of categorical variables, except that the corresponding design matrix is singular with one too many levels. To avoid a singularity, you use the ZERO= option in the RANDOM statement to eliminate one of the levels. Extending the previous example, the following statements treat **Cluster** as a subject in the RANDOM statement:

```
parm B1;
prior B1 ~ n(0, sd=100);
random Gamma ~ n(0, var=S2) subject = Center;
random B ~ n(0, sd=100) subject=Cluster zero='D';
Eta = B1 + B + Gamma;
```

The first RANDOM statement declares the intercept random effect for **Center**. The second RANDOM statement declares a **B** effect, which according to the syntax appears to be a random effect but mathematically is a fixed effect. Because this is a fixed effect disguised in a RANDOM statement, you want to use the same prior here as for other regression coefficients, such as **B1**. In other words, you do not want to treat **B** as a random effect and model its variance hyperparameter. The symbol **B** enters the regression model by itself, without any covariates. As PROC MCMC steps through the **Class** data set, **B** takes either parameter values (when **Cluster = A, B,** or **C**) or a value of zero (when **Cluster = D**). The model that this approach produces is equivalent to the model that is produced by the preprocessing approach in the previous section.

Comparing the Preprocessing and In-Procedure Approaches

There are tradeoffs between the preprocessing and in-procedure approaches for handling categorical variables. The preprocessing approach does the following:

- offers more flexibility in coding a different design matrix
- requires that you declare all $J-1$ coefficients in PARMS statements and write the complete regression equation
- updates coefficients jointly, which can be efficient when parameters are correlated but can be inefficient if the dimension becomes large

The in-procedure approach does the following:

- offers one design matrix coding (0 or 1 indicator variable)
- reduces code writing and can easily handle situations that have large J categories when parameters are correlated

Random Statement

The RANDOM statement in PROC MIXED defines the random effects and the G-side covariance matrix. The RANDOM statement in PROC GLIMMIX can also be used to define the R-side structure (which is achieved by the REPEATED statement in PROC MIXED). The random-effects distribution in the MIXED and GLIMMIX procedures is limited to the normal or the multivariate normal distribution. PROC MCMC is similar to PROC NLMIXED: in the RANDOM statement, you must name the effects, their prior distribution, and the SUBJECT= variable.[3]

The SUBJECT= variable in the RANDOM statement indicates cluster membership and identifies the subjects in the mixed model. All three mixed modeling procedures assume independence across subjects. PROC MCMC does too, but it can also model autoregressive dependence across subjects.[4]

It is important to understand the SUBJECT=*effect* option in the MIXED and GLIMMIX procedures in order to translate models to PROC MCMC. You particularly need to understand how it interacts with random effects (whether they are continuous or categorical) in the PROC MCMC statement.

Continuous Effects

When the random effects in a PROC MIXED or a PROC GLIMMIX program are continuous, translation to PROC MCMC is straightforward. Suppose that you have the following statements in PROC GLIMMIX:

```
   random Intercept Age / subject = Center;
```

In this statement, the subject effect is **Center** and the random effects are **Intercept** and **Age** (**Age** is a continuous variable). The preceding statement corresponds to fitting a two-dimensional random-effects model in PROC MCMC as follows:

```
   array Alpha[2];
   random Alpha ~ mvn(Mu, Sigma) subject=Center;
   Eta = Alpha[1] + Alpha[2]*Age;
```

In these statements, **Alpha** is a two-dimensional array-based random effect, you include programming statements to construct the regression mean **Mu** (which can be an array with values of zeros or computed means), and **Sigma** is the covariance matrix, which can take on different types (see “Example: Models with Different Covariance Types”). Here **Eta** is a placeholder for the linear predictor in the random-effects model. You can also include fixed effects in the calculation if needed. PROC MCMC preprocesses the input data set, determines the number of unique **Center** values (J), and creates J random-effects parameters. As the procedure steps through the input data set, the array **Alpha** is filled with random-effects parameters, depending on the values of **Center**.

Suppose the subject is specified as an interaction in PROC GLIMMIX:

```
   random Intercept Age / subject = Center*County;
```

You must first create a new subject variable in PROC MCMC’s input data set that matches this interaction subject effect. You can do that, for example, by using the DATA step and concatenating the **Center** and the **County** variables:

```
   CenterCounty = trim(Center)||'_'||trim(County);
```

Now you can fit the same model in PROC MCMC:

```
   random Alpha ~ mvn(Mu, Sigma) subject=CenterCounty;
   Eta = Alpha[1] + Alpha[2]*Age;
```

The same approach works for nested subject effects—such as SUBJECT=CENTER(COUNTRY) in PROC GLIMMIX syntax—and you can translate the model by creating a new subject index in PROC MCMC’s input data set.

Categorical Effects

When some of the random effects in a PROC MIXED or a PROC GLIMMIX program are categorical, the procedure assumes a nesting effect in which all random effects in the RANDOM statement are assumed to be nested within the SUBJECT= effect. Here is an example in which **Cluster** and **County** are both categorical variables and **Age** is a continuous variable:

[3]The RANDOM statement in PROC MCMC offers a bit more flexibility in terms of more prior distributions and ability to fit non-nested models, and it handles different types of subject variables (sorted or unsorted).

[4]This dependence across subjects is different from a G-side AR(1) covariance that models within-subject covariance structure.

```
random Intercept Age*Cluster County / subject = Center;
```

You can model categorical random effects by using the same approach as is used in modeling continuous random effects. The approach requires that you know the levels of each categorical variable and that you construct the design matrix in advance.

Alternatively, if you assume independent priors on the random effects, there is a more straightforward way to translate this nesting effect, which is assumed in PROC GLIMMIX. You first rewrite a single RANDOM statement as separate statements (also in PROC GLIMMIX syntax) by absorbing each categorical variable into the SUBJECT= effect, and then translate each statement in PROC MCMC.

The following expansion of statements in PROC GLIMMIX is equivalent to the previous RANDOM statement in PROC GLIMMIX:

```
random Intercept / subject = Center;
random Age       / subject = Center*Cluster;
random Intercept / subject = Center*County;
```

In these statements, each categorical variable is combined with the SUBJECT= effect to form a new SUBJECT= effect. Now translation to PROC MCMC is straightforward. You preprocess the input data set to create two new subject variables, **Centercluster** and **Centercounty**:

```
CenterCluster = trim(Center)||'_'||trim(Cluster);
CenterCounty  = trim(Center)||'_'||trim(County);
```

You can then use the following statements to fit the same model:

```
random A  ~ normal(0, var=S2_a)  subject = Center;
random Cg ~ normal(0, var=S2_cg) subject = CenterCluster;
random Cc ~ normal(0, var=S2_cc) subject = CenterCounty;
Mu = A + Cg*Age + Cc;
```

Here it is assumed that the three random effects are independent and that each one has a separate variance term. Although the random effects are assumed to be independent a priori, the posterior distribution still accounts for correlations among all parties, reflecting correctly what the data inform about the parameters. This type of independent normal prior is often desirable in practice.

Quick Reference

Supported Distributions

PROC MIXED supports only the normal distribution, whereas PROC GLIMMIX supports all members of the exponential family of distributions. This section explains how to replicate PROC GLIMMIX distributions in PROC MCMC (see Table 1 for equivalence). PROC GLIMMIX parameterizes each distribution in such a way that in all cases $\mathbf{E}(\mathbf{Y}) = \boldsymbol{\mu}$ and $\mathrm{var}[y] = \mu + \phi\mu^2$, where $\boldsymbol{\mu} = g^{-1}(\eta)$ and ϕ is the scale parameter. Therefore, the parameterizations for some distributions (such as the beta and gamma) are significantly different from conventional formulations and might be unfamiliar.

The random variable y is denoted as **y** in the PROC MCMC example code, and the scale parameter (shown as **scale** in the PROC GLIMMIX output) is denoted as **phi**.[5]

Table 1 PROC GLIMMIX Distributions and Usage in PROC MCMC

DIST=	MCMC Code	Default Link Function
BETA	See the section "Beta Distribution"	`p = logistic(eta)`
BINARY[6]	`binary(1-p)`	`p = logistic(eta)`
BINOMIAL	`binomial(n, p)`	`p = logistic(eta)`

[5]The exception is the gamma distribution, for which PROC GLIMMIX reports **scale = 1/phi**.

[6]By default, PROC GLIMMIX models $Pr(y = 0)$. You can model $\Pr(y = 1)$ by specifying the EVENT= option in the MODEL statement's response variable options, as in `model y(event='1')=...;`.

Table 1 *continued*

DIST=	MCMC Code	Default Link Function
EXPONENTIAL	`expon(scale=mu)`	`mu = exp(eta)`
GAMMA	See the section "Gamma Distribution"	`mu = exp(eta)`
GAUSSIAN	`normal(mu, var=)`	`mu = eta`
GEOMETRIC	`general(y*(log(mu) - (y+1)*log(1+mu)))`	`mu = exp(eta)`
INVGAUSS	`wald(mean=mu, iscale=1/phi)`	`mu = sqrt(1/eta)`
LOGNORMAL	`lognormal(mean=mu, var=phi)`	`mu = eta`
MULTINOMIAL	`multinom(p)`	Cumulative logit
NEGBINOMIAL	See the section "Negative Binomial Distribution"	`mu = exp(eta)`
POISSON	`poisson(mu)`	`mu = exp(eta)`
TCENTRAL	`t(mu, var=scale, df=)`	`mu = eta`
BYOBS	See the section "DIST=BYOBS"	

Beta Distribution

PROC GLIMMIX uses the following parameterization for the beta distribution (Ferrari and Cribari-Neto 2004):

$$l(\mu,\phi;y) = \log\left\{\frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)}\right\} + (\mu\phi-1)\log(y) + ((1-\mu)\phi-1)\log(1-y)$$

To specify the same likelihood function in PROC MCMC, you can use the general distribution and the following code:

```
Mu = logistic(Eta);
llike = lgamma(Phi) - lgamma(Mu*Phi) - lgamma((1-Mu)*Phi)
   + (Mu*Phi - 1)*log(Y) + ((1-Mu)*Phi - 1)*log(1 - Y);
model general(llike);
```

The symbol **Eta** is the linear predictor, and **Mu** is used here to be consistent with the likelihood function—it is the probability, which is usually represented by **P**.

Gamma Distribution

PROC GLIMMIX uses the following parameterization for the gamma distribution:

$$l(\mu,\phi;y) = \phi\log\left\{\frac{y\phi}{\mu}\right\} - \frac{y\phi}{\mu} - \log(y) - \log(\Gamma(\phi))$$

The following PROC MCMC statements specify the same likelihood function:

```
Mu = exp(Eta);
llike = Phi*log(Y*Phi / Mu) - Y*Phi / Mu - log(Y) - lgamma(Phi);
model general(llike);
```

PROC GLIMMIX reports **1/phi** as the **scale** parameter in its output tables.

Negative Binomial Distribution

PROC GLIMMIX uses the following parameterization for the negative binomial distribution:

$$l(\mu,\phi;y) = y\log\{\phi\mu\} - (y+1/\phi)\log\{1+\phi\mu\} + \log\left\{\frac{\Gamma(y+1/\phi)}{\Gamma(1/\phi)\Gamma(y+1)}\right\}$$

The following PROC MCMC statements specify the same likelihood function:

```
Mu = exp(Eta);
llike = Y*log(Phi*Mu) - (Y + 1/Phi)*log(1 + Phi*Mu)
   + lgamma(Y + 1/Phi) - lgamma(1/Phi) - lgamma(Y + 1);
model general(llike);
```

DIST=BYOBS

The DIST=BYOBS option in PROC GLIMMIX enables you to model bivariate responses (for example, two response variables, one that has a binary distribution and one that has a Poisson distribution). The following is an example of such a data set:

```
OBS  dist   d patient age response

  1 Binary  B     1    78    0
  2 Poisson P     1    78    9
  3 Binary  B     2    60    0
  4 Poisson P     2    60    4
  5 Binary  B     3    68    1
...
```

Here the **Dist** variable indicates the distribution for each observation, and it is used in the DIST=BYOBS(DIST) option to perform joint modeling of the bivariate outcome, as follows:

```
proc glimmix data=joint;
   class Patient Dist;
   model Response(event='1') = Dist Dist*Age / dist=byobs(Dist) noint;
   random int / subject=Patient;
run;
```

PROC GLIMMIX fits two models (one logistic and one Poisson regression) to the two outcome variables, with separate intercept and age coefficients and a shared random **Patient**-level intercept.

To perform joint modeling in PROC MCMC, you transpose the original data set by the **Dist** variable and create two response variables, **Binresp** and **Poiresp**:

```
                  Bin   Poi
OBS patient age  Resp  Resp

  1     1    78    0     9
  2     2    60    0     4
  3     3    68    1     7
...
```

You can use the following statements in PROC MCMC to fit a joint model:

```
proc mcmc data=wide nmc=10000 seed=12797;
   parms A0 A1 B0 B1 S2;
   prior A: B: ~ n(0, sd=10000);
   prior S2 ~ igamma(shape=3, scale=2);
   random Gamma ~ n(0, var=S2) subject = Patient;
   P = logistic(A0 + A1*Age + Gamma);
   model BinResp ~ binary(P);
   Lambda = exp(B0 + B1*Age + Gamma);
   model PoiResp ~ poisson(Lambda);
run;
```

Each of the two MODEL statements is based on a different regression model, but they share the same random-effects, **Gamma**.

Links

The linear predictor is $\eta = \beta X + \gamma Z$, which is represented by **Eta** in this section. The link functions are $g(\mu) = \eta$, and it follows that $\mu = g^{-1}(\eta)$. Table 2 lists some common link functions that are used in PROC GLIMMIX and equivalent programming statements that should be used in PROC MCMC if you want to fit the same generalized linear mixed-effects model.

Table 2 Link Functions and Usage in PROC MCMC

LINK=	$g(\mu) = \eta =$	$\mu = g^{-1}(\eta)$
CLOGLOG	$\log(-\log(1-\mu))$	`mu = 1 - exp(-exp(eta))`
IDENTITY	μ	`mu = eta`
LOG	$\log(\mu)$	`mu = exp(eta)`
LOGIT	$\log(\mu/(1-\mu))$	`mu = logistic(eta)`
LOGLOG	$-\log(-\log(\mu))$	`mu = exp(-exp(-eta))`
PROBIT	$\Phi^{-1}(\mu)$	`mu = cdf('normal', eta)`
POWER(λ)	$\mu^{\lambda}, \lambda \neq 0$	`mu = eta**(1/lambda)`
POWERMINUS2	$1/\mu^2$	`mu = sqrt(1/eta)`
RECIPROCAL	$1/\mu$	`mu = 1/eta`

Ordinal Data

PROC GLIMMIX uses various cumulative links to model ordinal response data. The input data set is organized in a long format with one response variable (**Y**) and one categorical variable (**C**), which indicates the response ordering. PROC GLIMMIX determines the dimension of the multinomial model by the number of unique values in the **C** variable. To fit an ordinal model in PROC GLIMMIX, you input the category-indicating variable **C** in the MODEL statement and the response variable **Y** in the FREQ statement. You can fit different models by selecting different link functions (such as CUMCLL, CUMLOGIT, CUMLOGLOG, CUMPROBIT, or GLOGIT).

To model ordinal data in PROC MCMC, you must first reshape the input data set into a wide format, in which each record contains all categories of the response, Then you specify the multinomal distribution in the MODEL statement to model the data. For more information, see http://support.sas.com/rnd/app/examples/stat/BayesMulti/stat_webex.pdf .

GROUP= Option

The GROUP= option in a RANDOM statement varies the covariance parameters according to group within each cluster. For example, consider the following statement:

```
random Intercept / subject=Center group=Gender;
```

In this statement, **Gender** takes on two values ('M' and 'F'). The statement estimates one variance for the male group and one for the female group, within groups that are clustered by **Center**.

The following PROC MCMC statements fit the same the model:

```
parms S2m S2f;
prior S2m S2f ~ uniform(0, 100);
if Gender = 'M' then S2 = S2m;
else S2 = S2f;
model Gamma ~ n(0, var=S2) subject = Center;
```

There are two variances: **S2m** for the male group and **S2f** for the female groups. The IF ELSE-THEN programming statements replace the variance parameter **S2** according to the **Gender** values.

Conclusion

This paper provides information about how to use PROC MCMC to perform Bayesian analysis in models that are frequently fitted using SAS/STAT mixed modeling procedures, with a focus on more commonly encountered linear mixed-effects models (often fitted by PROC MIXED) and generalized linear mixed-effects models (often fitted by PROC GLIMMIX). The paper presents some fundamental concepts in making Bayesian inferences (such as the selection of noninformative prior distributions) and covers key elements in translating mixed models into the MCMC procedure (such as coding CLASS variables, converting RANDOM and REPEATED statements, and fitting models with various covariance types). Additional details, such as custom hypothesis testing, predictions on existing and new observations, and information about distributions of errors and link functions, are also presented.

REFERENCES

Box, G. E. P., and Tiao, G. C. (1973). *Bayesian Inference in Statistical Analysis*. New York: John Wiley & Sons.

Chaloner, K. (1987). "A Bayesian Approach to the Estimation of Variance Components for the Unbalanced One-Way Random Model." *Technometrics* 29:323–337.

Chen, F. (2009). "Bayesian Modeling Using the MCMC Procedure." In *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. `http://support.sas.com/resources/papers/proceedings09/257-2009.pdf`.

Chen, F. (2011). "The RANDOM Statement and More: Moving On with PROC MCMC." In *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc. `http://support.sas.com/resources/papers/proceedings11/334-2011.pdf`.

Chen, F. (2013). "Missing No More: Using the MCMC Procedure to Model Missing Data." In *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. `https://support.sas.com/resources/papers/proceedings13/436-2013.pdf`.

Daniels, M. J. (1999). "A Prior for the Variance in Hierarchical Models." *Canadian Journal of Statistics* 27:567–578.

Davidian, M., and Giltinan, D. M. (1995). *Nonlinear Models for Repeated Measurement Data*. New York: Chapman & Hall.

Ferrari, S. L. P., and Cribari-Neto, F. (2004). "Beta Regression for Modelling Rates and Proportions." *Journal of Applied Statistics* 31:799–815.

Gelfand, A. E., Hills, S. E., Racine-Poon, A., and Smith, A. F. M. (1990). "Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling." *Journal of the American Statistical Association* 85:972–985.

Gelman, A. (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis* 1:515–533.

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. (2006). *SAS for Mixed Models*. 2nd ed. Cary, NC: SAS Institute Inc.

McCullagh, P., and Nelder, J. A. (1989). *Generalized Linear Models*. 2nd ed. London: Chapman & Hall.

Neal, R. M. (2003). "Slice Sampling." *Annals of Statistics* 31:705–757.

Vonesh, E. F., Chinchilli, V. M., and Pu, K. (1996). "Goodness-of-Fit in Generalized Nonlinear Mixed-Effects Models." *Biometrics* 52:572–587.

Acknowledgments

The authors would like to thank Dr. Min Zhu and Dr. Tianlin Wang for their valuable comments and suggestions, especially about the SAS mixed modeling procedures.

CONTACT INFORMATION

Complete documentation for the MCMC procedure, in both PDF and HTML format, can be found on the web at `http://support.sas.com/documentation/onlinedoc/stat/indexproc.html`.

You can find additional coding examples at `http://support.sas.com/rnd/app/examples/index.html`.

Your comments and questions are valued and encouraged. Contact the author:

Fang Chen
SAS Institute Inc.
SAS Campus Drive, Cary, NC 27513
E-mail: fangk.chen@sas.com