Paper 334-2011

The RANDOM Statement and More: Moving On with PROC MCMC

Fang Chen, SAS Institute Inc, Cary, NC

ABSTRACT

The MCMC procedure, first released in SAS/STAT® 9.2, provides a flexible environment for fitting a wide range of Bayesian statistical models. Key enhancements in SAS/STAT 9.22 and 9.3 offer additional functionality and improved performance. The RANDOM statement provides a convenient way to specify linear and nonlinear random-effects models along with substantially improved performance. The MCMC procedure also supports multivariate distributions, such as the multivariate normal and inverse-Wishart distributions, and implements conjugate sampling methods when appropriate to improve sampling speed.

This paper describes key enhancements of PROC MCMC in SAS/STAT 9.22 and 9.3 and illustrates the use of these enhancements with examples.

INTRODUCTION

The MCMC procedure is a Bayesian sampling procedure based on Markov chain Monte Carlo methods. First released in SAS/STAT 9.2, PROC MCMC accommodates a broad range of Bayesian statistical models, and its main sampling mechanism is a self-tuned random walk Metropolis algorithm. You can use the MCMC procedure to fit linear or nonlinear models that take various forms (for example, a multilevel model in which the parameters have a nonlinear relationship with the response variable and the likelihood takes on a nonstandard form). For an overview of the MCMC procedure, see Chen (2009).

This paper focuses on two key enhancements to PROC MCMC in SAS/STAT 9.3: the RANDOM statement and multivariate distributions. It also illustrates the usage of the PREDDIST statement (available in SAS/STAT 9.22) for posterior prediction and explains the newly implemented conjugate sampling algorithms that are available in SAS/STAT 9.3.

USING PROC MCMC

To use PROC MCMC, you specify the model parameters, the prior distributions, and the likelihood function (the conditional distribution of the response variable given the parameters and covariates). The prior and likelihood function jointly define a posterior distribution, which becomes the objective function that PROC MCMC uses in simulation.

The simplest call to the MCMC procedure has the following form:

PROC MCMC ***options*****;**
  PARMS; declare model parameters
  PRIOR; declare prior distributions, $\log(\pi(\theta))$
  Programming statements; / **MODEL;** declare the likelihood function, $\log(f(y_i|\theta))$, for each observation
  Run;

The *options* control simulation setup, posterior calculation, convergence diagnostics, and plotting. The MCMC procedure enables you to work with either standard distributions (normal, Poisson, truncated gamma, and so on) or construct a general distribution using SAS programming statements. If you are working with a large number of parameters, you can use multiple PARMS statements to place them in blocks. Each block of parameters is updated in the Metropolis algorithm conditionally in every iteration.

By default, PROC MCMC assumes that all observations are independent and calculates the posterior density (on the logarithm scale) as

$$\log(p(\theta|\mathbf{y})) = \log(\pi(\theta)) + \sum_{i=1}^{n} \log(f(y_i|\theta)) + C$$

where θ can be a multidimensional parameter vector, $f(y_i|\theta)$ is the likelihood function for a single observation in the data set, n is the sample size, and C is a constant that can be ignored.

In addition to the essential statements, you can use the BEGINCNST/ENDCNST and BEGINNODATA/ENDNODATA statements to reduce unnecessary evaluation and shorten simulation time:

BEGINCNST;
Programming statements;
ENDCNST;
} declare constant terms

BEGINNODATA;
Programming statements;
ENDNODATA;
} calculate functions of parameters

The BEGINCNST and ENDCNST statements define a block within which PROC MCMC processes the programming statements only during the setup stage of the simulation. Enclose within the BEGINCNST and ENDCNST statements any programming statements that you want to perform only once (such as statements that define a constant or import data set variables into arrays). Similarly, the BEGINNODATA and ENDNODATA statements define a block within which PROC MCMC processes the programming statements without stepping through the entire data set. These are best used to reduce unnecessary observation-level computations. Enclose within the BEGINNODATA and ENDNODATA statements any computations that are identical for every observation (such as transformation of parameters).

RANDOM-EFFECTS MODELS

One of the most frequently used statistical model is the random-effects model. Suppose you are interested in modeling data that consist of groups of populations. These clusters of subjects (such as students in different schools) share a response (for example, a testing score), some common characteristic measures (for example, age and gender), and cluster-specific characteristics (for example, different teaching methods, such as Montessori versus traditional). You can use the following generic setup of a random-effects model to model this type of data:

$$Y_{ij} = \alpha \cdot X_{ij} + \beta_j + \epsilon_{ij}, \quad j = 1 \cdots J, \quad i = 1 \cdots n_j \tag{1}$$

where Y_{ij} is the response value of the ith subject in the jth cluster, J is the total number of clusters, and n_j is the total number of subjects in the jth cluster. The parameter α is the shared regression coefficient for covariate X_{ij}, and ϵ_{ij} are independent and identically distributed (i.i.d.) errors from a common distribution. The β_j (the random effects) are the varying intercepts that are used to model the cluster-specific effect. Often it is assumed that β_j arise from the same distribution,

$$\beta_j \sim \pi(\theta) \tag{2}$$

where θ are the hyperparameters.

If the error term ϵ_{ij} is assumed to have a normal distribution, then model (1) becomes a linear random-effects model. If you choose to model the response variable in a different approach, such as

$$E(Y_{ij}) = g(\mu \cdot X_{ij} + \beta_j)$$

where the expected value of the response is related to the regression through some functional transformation, the result is either a generalized linear model (if Y_{ij} is assumed to arise from the exponential family and $g(\cdot)$ is a one-to-one monotone transformation) or a nonlinear random-effects model (which accommodates an even wider range of distributions and flexible transformations).

Since there is no limit on the number or levels of clusters that you can include in the model, random-effects models provide a natural and flexible tool to measure either individual- or cluster-level dynamics. The first part of the paper focuses on explaining the use of the RANDOM statement in fitting random-effects models.

It is not the intent of this paper to provide a comprehensive overview of literature on Bayesian random-effects models. Interested readers can refer to papers and text books such as Lindley and Smith (1972), Box and Tiao (1973, Chapter 5), Zeger and Karim (1991), Gelman and Hill (2007), and Congdon (2010), and references within.

MULTIVARIATE DISTRIBUTIONS

Often you encounter data that involve multiple response variables and for which you want to model the multiple outcomes jointly (on the observational level) to capture the data variability across dimensions. One example is multivariate normal regression; another example is the multinomial model used in repeated measurement study. In other cases,

you might be interested in specifying a multivariate prior distribution for multiple random effects with an unstructured covariance matrix.

In previous SAS/STAT releases, specifying a multivariate distribution in PROC MCMC required the use of the **general** function and programming statements to construct the logarithm of the density. And PROC MCMC could not efficiently sample constrained multivariate parameters (for example, the covariance matrix that must be positive definite) using the random walk Metropolis algorithm. The second part of the paper focuses on the usage of the multivariate distributions in PROC MCMC and discusses newly implemented conjugate sampling algorithms that improve the performance of PROC MCMC.

THE RANDOM STATEMENT

The RANDOM statement simplifies the specification of the random-effects parameters—the β_j in Equation (2)—in an MCMC procedure program. PROC MCMC assumes a priori that the random-effects parameters within a cluster share the same prior distribution $\pi(\theta)$ and that they are independent of each other. You can have multiple random effects in a model with each having its own prior distribution; this amounts to using multiple RANDOM statements. Once a random effect is defined in the program, it can enter the model on the observational level, regardless of the type of the model (linear or nonlinear) that is being used in the program.

The syntax of the the RANDOM statement in PROC MCMC is similar to the syntax of the RANDOM statement in the NLMIXED procedure. The statement defines the random effects, which you can indicate using an arbitrary name as long as it does not conflict with the names for model parameters or data set variables that are used in the program. The statement has the following syntax:

RANDOM *random-effect* $\sim$ *distribution* **SUBJECT**=*variable* < *options* > **;**

The random effect defined by the statement can be either a single effect or an array of effects. You must specify a prior distribution for the random effect. Not all distributions that are supported in the PRIOR statement or the MODEL statement can be used in the RANDOM statement. Table 1 lists a set of prior distributions that can be used. Unlike their counterparts in the PRIOR statements, these distributions do not take optional lower- or upper-bound specifications.

Table 1 Valid Distributions in the RANDOM Statement

Distribution	Definition
beta(α,β)	Beta distribution with shape parameters α and β
binary(p)	Binary distribution with probability of success p
gamma(a, **scale**|**iscale=** λ)	Gamma distribution with shape a and scale or inverse-scale λ
igamma(a, **scale**|**iscales=**λ)	Inverse-gamma distribution with shape a and scale or inverse-scale λ
normal(μ, **sd**|**var**|**prec=** λ)	Normal distribution with mean μ and standard deviation or variance or precision λ
mvn($\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$)	Multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

You must specify the SUBJECT= option, which declares a data set variable that indicates the random effects' cluster membership. The SUBJECT= variable can be either a numeric variable or character literal, and it does not need to be sorted.

During the setup stage of the simulation, PROC MCMC first determines the number of random-effects parameters, which is the number of unique values in the SUBJECT= variable. Then PROC MCMC creates the random-effects parameters and updates them conditionally in the simulation. Using the SUBJECT= option bypasses the need to account for the number of clusters in the data set variable ahead of the time.

The MCMC procedure assumes that all random-effects parameters defined by the same RANDOM statement are conditionally independent of each other, given the data and other parameters in the model. Conditional independence is not assumed between random-effects parameters that are defined by different RANDOM statements. This conditional independence assumption enables PROC MCMC to identify observations that are uniquely associated with each random-effects parameter and use that information to construct the object function (the logarithm of the conditional posterior density function) for every random-effects parameter.

For continuous random effects, PROC MCMC uses a single-dimensional random walk Metropolis algorithm based on a normal proposal distribution to draw the posterior samples. For discrete random effects, such as random effects that have a binary prior, PROC MCMC uses an inverse cumulative distribution function sampling method. The objective function used in the Metropolis algorithm typically involves only a fraction of the input data set, which reduces the computational cost and shortens the simulation time. In SAS/STAT 9.3, the MCMC procedure requires only one pass

through the data set to update all random-effects parameters from one RANDOM statement. Previous versions required the number of passes through the data set to be equal to the number of unique PARMS statements that are used to define the random-effects parameters.

Hyperparameters in the prior distribution of a random effect can be model parameters, data set variables, or functions of both. However, the hyperparameters must remain constant within each subject group—a meaningful model cannot center a random-effects parameter at two different mean values.

Although multiple RANDOM statements are supported, the hyperparameters of one random effect cannot be other random effects that are declared in a different RANDOM statement. For example, the following statements are not allowed because the random effect g appears in the distribution for the random effect u:

```
random g ~ normal(0,var=s2g) subject=day;
random u ~ normal(g,var=s2u) subject=year;
```

You use the INITIAL= option to specify initial values of the random-effects parameters, the MONITOR= option to output analysis for selected parameters, and the NAMESUFFIX= option to specify how the names of the random-effects parameters are internally created. More details can be found in the section "RANDOM Statement" in the chapter "The MCMC Procedure" in *SAS/STAT 9.3 User's Guide*.

LOGISTIC RANDOM-EFFECTS MODEL

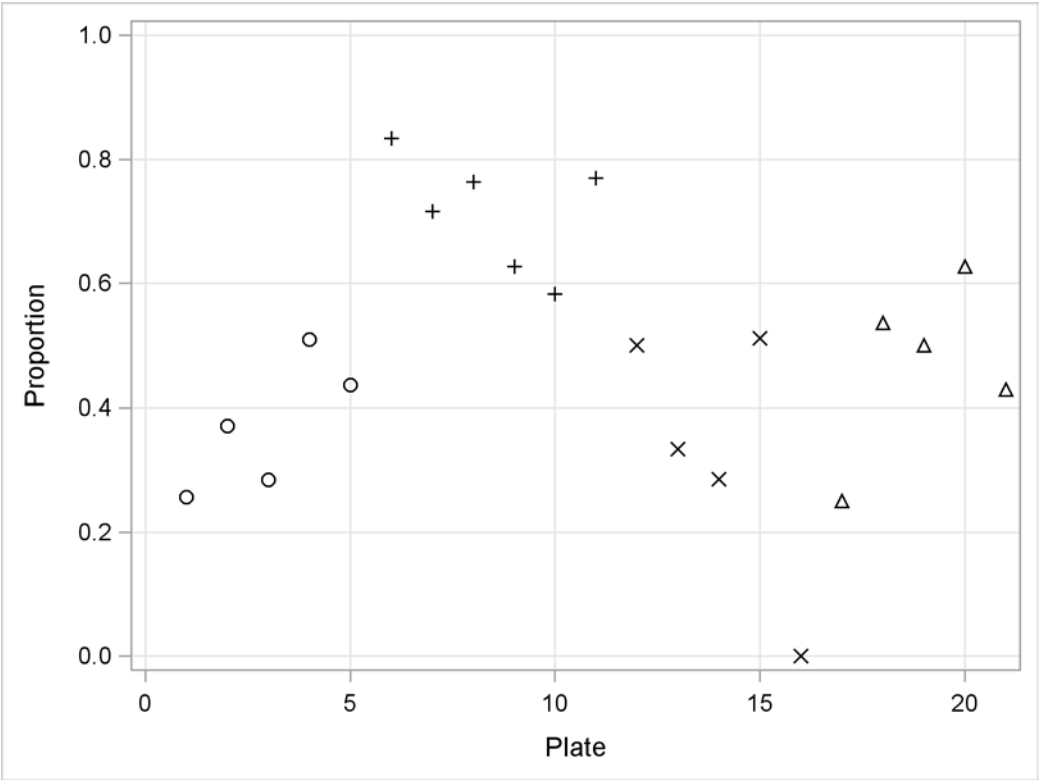
Consider the seeds data set from Crowder (1978) that was also analyzed by Breslow and Clayton (1993). The data set is a 2×2 factorial layout with two types of seeds, *O. aegyptiaca 75* and *O. aegyptiaca 73*, and two root extracts, *bean* and *cucumber*. The experiment included five or six replicates for each combination of seeds and root extracts. In each batch, the number of germinated seeds (variable r) and total number of seeds (variable n) are recorded. The independent variables are seed and extract.

The following statements create the data set:

```
data seeds;
   input r n seed extract @@;
   ind = _N_;
   datalines;
10  39  0  0    23  62  0  0    23  81  0  0    26  51  0  0
17  39  0  0     5   6  0  1    53  74  0  1    55  72  0  1
32  51  0  1    46  79  0  1    10  13  0  1     8  16  1  0
10  30  1  0     8  28  1  0    23  45  1  0     0   4  1  0
 3  12  1  1    22  41  1  1    15  30  1  1    32  51  1  1
 3   7  1  1
;
```

Output 1 shows the success probabilities of germinated seeds for the 21 experiments. The symbol "$\circ$" represents results from the experiments that use *O. aegyptiaca 75* and *bean*, "$+$" represents *O. aegyptiaca 75* and *cucumber*, "$\times$" represents *O. aegyptiaca 73* and *bean*, and "$\triangle$" represents *O. aegyptiaca 73* and *cucumber*.

Output 1 Success Probabilities of Germination



Crowder (1978) noted heterogeneity of the proportions between replicates. In other words, the amount of variability within each batch exceeds that allowed by a binomial model. To account for excessive variation, Breslow and Clayton (1993) suggested a generalized linear random-effects model with observational-level random effects.

The likelihood is a binomial distribution:

$$r_i \sim \text{binomial}(n_i, p_i)$$

A logit function links the covariates of each observation (seed and extract) to the probability of success,

$$\begin{aligned} \mu_i &= \beta_0 + \beta_1 \cdot \text{seed}_i + \beta_2 \cdot \text{extract}_i + \beta_3 \cdot \text{seed}_i \cdot \text{extract}_i \\ p_i &= \text{logistic}(\mu_i + \delta_i) \end{aligned}$$

where δ_i is assumed to be an i.i.d. random effect with a normal prior:

$$\delta_i \sim \text{normal}(0, \text{var} = \sigma^2)$$

The four β regression coefficients and the standard deviation σ^2 in the random effects are model parameters; they are given priors as follows:

$$\begin{aligned} \pi(\beta_0, \beta_1, \beta_2, \beta_3) &\propto \text{normal}(0, \text{sd} = 1000) \\ \sigma^2 &\sim \text{igamma}(\text{shape} = 0.01, \text{scale} = 0.01) \end{aligned}$$

The following statements fit the model:

```
proc mcmc data=seeds outpost=postout seed=332786 nmc=20000
   stats=(summary intervals) diag=none;
   parms beta0 0 beta1 0 beta2 0 beta3 0 s2 1;
   prior beta: ~ normal(0, sd=1000);
   prior s2 ~ igamma(0.01, s=0.01);
   w = beta0 + beta1*seed + beta2*extract + beta3*seed*extract;
   random delta ~ normal(0, var=s2) subject=ind;
   pi = logistic(w + delta);
   model r ~ binomial(n = n, p = pi);
run;
```

The PROC MCMC statement specifies the input and output data sets, sets a seed for the random number generator, and requests a large simulation size. The procedure computes the posterior summary and interval statistics, and turns off all the convergence diagnostics. The PARMS statement declares the model parameters and assigns initial values. There are four regression coefficients (beta0, beta1, beta2, and beta3) and a variance parameter (s2). The PRIOR statements specify the prior distributions for β and σ^2.

The symbol w calculates the regression mean, and the RANDOM statement specifies the random effect, with a normal prior distribution, centered at 0 with variance σ^2 (s2). The SUBJECT= option indicates the group index for the random-effects parameters. The symbol pi is the logit transformation. The MODEL statement specifies the response variable r as a binomial distribution with parameters n and pi.

Output 2 shows part of the output from this analysis:

Output 2 Model Parameters Information

Parameters

Block	Parameter	Sampling Method	Initial Value	Prior Distribution
1	s2	Conjugate	1.0000	igamma(0.01, s=0.01)
2	beta0	N-Metropolis	0	normal(0, sd=1000)
	beta1		0	normal(0, sd=1000)
	beta2		0	normal(0, sd=1000)
	beta3		0	normal(0, sd=1000)

The "Parameters" table lists the sampling block information, the name of the model parameters, sampling algorithms used, initial values, and their prior distributions. The conjugate sampling algorithm (see the section "CONJUGATE SAMPLING") is used to draw the posterior samples of s2 and random walk Metropolis for the regression parameters.

Output 3 Random-Effects Parameters Information

Random Effects Parameters

Parameter	Subject	Levels	Prior Distribution
delta	ind	21	normal(0, var=s2)

The "Random Effects Parameters" table in Output 3 lists the name of the random effect, the subject variable, and the number of distinct levels in the subject variable. The total number of random-effects parameters in this model is 21. The prior is a normal distribution that centers at 0 and has a variance parameter s2.

Output 4 lists the posterior summary statistics of all the model parameters.

Output 4 Posterior Summary and Interval Statistics of the Model Parameters

The MCMC Procedure

Posterior Summaries

Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
beta0	20000	−0.5546	0.2037	−0.6844	−0.5497	−0.4301
beta1	20000	0.0679	0.3241	−0.1455	0.0666	0.2892
beta2	20000	1.3640	0.2934	1.1744	1.3539	1.5429
beta3	20000	−0.8237	0.4644	−1.1209	−0.8215	−0.5130
s2	20000	0.1192	0.1023	0.0482	0.0929	0.1609

Posterior Intervals

Parameter	Alpha	Equal-Tail Interval		HPD Interval	
beta0	0.050	−0.9597	−0.1310	−0.9514	−0.1283
beta1	0.050	−0.5719	0.6872	−0.5719	0.6835
beta2	0.050	0.8070	1.9912	0.7707	1.9445
beta3	0.050	−1.7778	0.1026	−1.6803	0.1620
s2	0.050	0.00811	0.3829	0.00190	0.3139

The "Posterior Summaries" and "Posterior Intervals" tables in Output 4 list the posterior summary and interval statistics. You can see a rather significant regression coefficient estimate for extract, and a less significant effect for seed and its interaction with extract. By default, PROC MCMC does not output analysis for the random-effects parameters. That is why you do not see posterior summary statistics of delta in the "Posterior Summaries" table. Use the MONITOR= option to output relevant analyses, as illustrated in the following statement:

```
random delta ~ normal(0, var=s2) subject=ind monitor=(delta);
```

Output 5 lists the posterior summary statistics of all 21 random-effects parameters.

Output 5 Posterior Summary and Interval Statistics of the Random-Effects Parameters

The MCMC Procedure

Posterior Summaries

Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
delta_1	20000	−0.2169	0.2684	−0.3695	−0.1812	−0.0333
delta_2	20000	0.0138	0.2369	−0.1253	0.00907	0.1496
delta_3	20000	−0.2188	0.2437	−0.3711	−0.2005	−0.0523
delta_4	20000	0.3052	0.2717	0.1164	0.2791	0.4687
delta_5	20000	0.1309	0.2587	−0.0352	0.1083	0.2797
delta_6	20000	0.0793	0.3297	−0.1145	0.0424	0.2586
delta_7	20000	0.0639	0.2356	−0.0825	0.0526	0.2024
delta_8	20000	0.1959	0.2450	0.0338	0.1746	0.3449
delta_9	20000	−0.1519	0.2567	−0.2964	−0.1238	0.00920
delta_10	20000	−0.2843	0.2559	−0.4356	−0.2543	−0.1016
delta_11	20000	0.0939	0.2955	−0.0855	0.0756	0.2573
delta_12	20000	0.1478	0.2994	−0.0535	0.1213	0.3189
delta_13	20000	−0.0794	0.2662	−0.2397	−0.0693	0.0789
delta_14	20000	−0.1431	0.2774	−0.3009	−0.1130	0.0262
delta_15	20000	0.2719	0.2877	0.0688	0.2339	0.4404
delta_16	20000	−0.1566	0.3326	−0.3336	−0.1227	0.0479
delta_17	20000	−0.2471	0.3276	−0.4348	−0.1886	−0.0218
delta_18	20000	0.0473	0.2535	−0.0981	0.0405	0.1969
delta_19	20000	−0.00976	0.2646	−0.1592	−0.00728	0.1378
delta_20	20000	0.2486	0.2761	0.0448	0.2211	0.4179
delta_21	20000	−0.0508	0.3084	−0.2212	−0.0317	0.1260

You can use the following syntax to select a subset of the random-effects parameters:

```
random delta ~ normal(0, var=s2) subject=ind monitor=(delta_1 delta_5-delta_9);
```

If you want to keep track of functions of the random-effects parameters (such as the variable pi for each subject), you need to allocate an array, use programming statements to store the functional values in that array, and use the MONITOR= option in the PROC MCMC statement to output analyses for or save the posterior samples of these array values. A call of **`monitor=(pi)`** does not work because PROC MCMC does not know which of the pi values you want

to monitor. During the simulation, PROC MCMC internally replaces the variable delta with the appropriate random-effects parameter at each observation as it steps through the data set. Therefore, at each iteration, there are 21 different values of pi. In order to monitor all of them, you need to allocate an array and store every value.

The following program illustrates such a usage:

```
proc mcmc data=seeds outpost=postout seed=332786 nmc=20000
   stats=(summary intervals) diag=none monitor=(pi_ary);
   array pi_ary[21];
   parms beta0 0 beta1 0 beta2 0 beta3 0 s2 1;
   prior beta: ~ normal(0, sd=1000);
   prior s2 ~ igamma(0.01, s=0.01);
   w = beta0 + beta1*seed + beta2*extract + beta3*seed*extract;
   random delta ~ normal(0, var=s2) subject=ind;
   pi = logistic(w + delta);
   pi_ary[ind] = pi;
   model r ~ binomial(n = n, p = pi);
run;
```

The ARRAY statement allocates an array variable pi_ary of size 21, the number of random-effects parameters. In the programming statements, **`pi_ary[ind] = pi`** stores the value of pi in the ind-th cell for each observation. For this statement to work, the subject index variable ind must have an integer value (1, 2, 3, and so on). If the SUBJECT= variable in the RANDOM statement for delta is a character variable, then you must create a matching integer index in the input data set and use the numeric index variable in the array. Lastly, the MONITOR= option in the PROC MCMC statement monitors all elements of pi_ary.

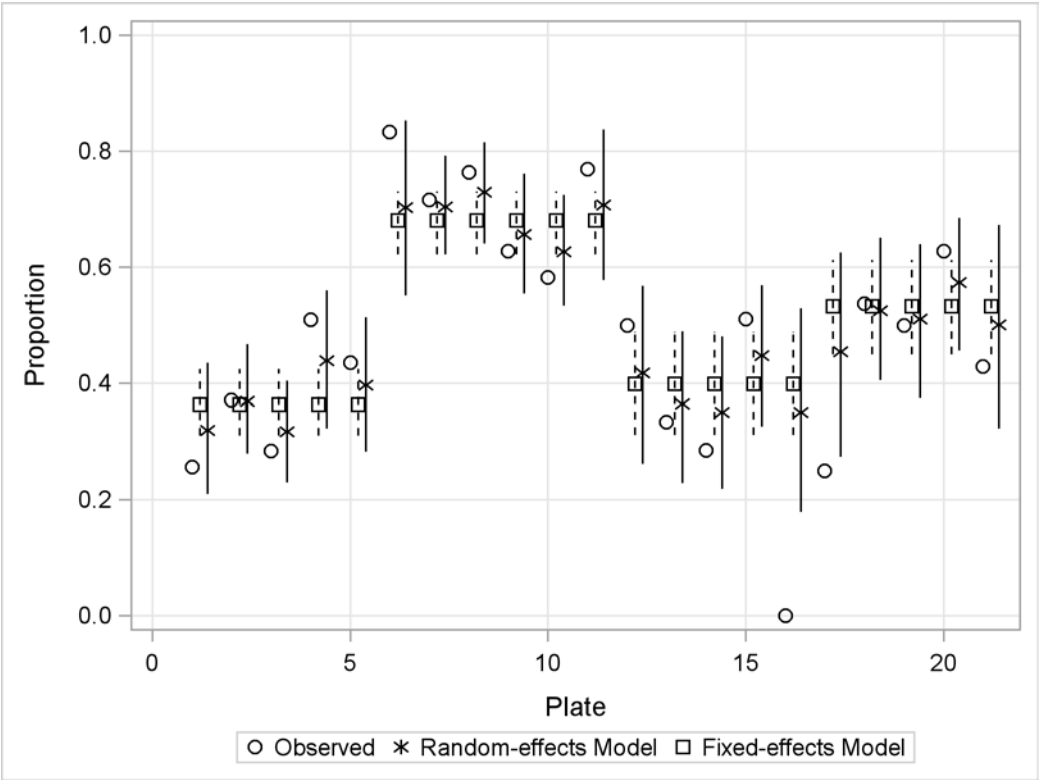
Output 6 shows the observed proportion and the fitted values from two different models: a logistic regression without the random effects and the logistic random-effects model. The fitting of a logistic regression is relatively straightforward—if you take out the RANDOM statement and the prior for s2 in the previous program, you are left with a program that fits a logistic regression with fixed-effects parameters only.

The following program shows how to fit a fixed-effects logistic regression:

```
proc mcmc data=seeds outpost=postout1 seed=332786 nmc=20000
   stats=(summary intervals) diag=none;
   ods output postsummaries = ps postintervals=pi;
   parms beta0 0 beta1 0 beta2 0 beta3 0;
   prior beta: ~ normal(0, sd=1000);
   w = beta0 + beta1*seed + beta2*extract + beta3*seed*extract;
   pi = logistic(w);
   model r ~ binomial(n = n, p = pi);
run;
```

Output 6 compares observed values ($\circ$) and fitted values from two different models: logistic regression ($\square$) and random-effects logistic regression ($*$). The vertical bars are the 95% highest posterior density intervals of the two models.

Output 6 Comparing Observed and Fitted Values from Two Different Models



Output 6 indicates that a logistic regression with only fixed-effects parameters fails to capture all the within-group variability. The model predicts a constant success probability for all experiments that use the same seed and root extract. On the other hand, the random-effects model can capture this variability, and there is also a clear shrinkage effect of the random-effects estimates towards the group means (the $*$ are all closer to the $\square$ than to the $\circ$).

REGENERATING DIAGNOSTICS PLOTS

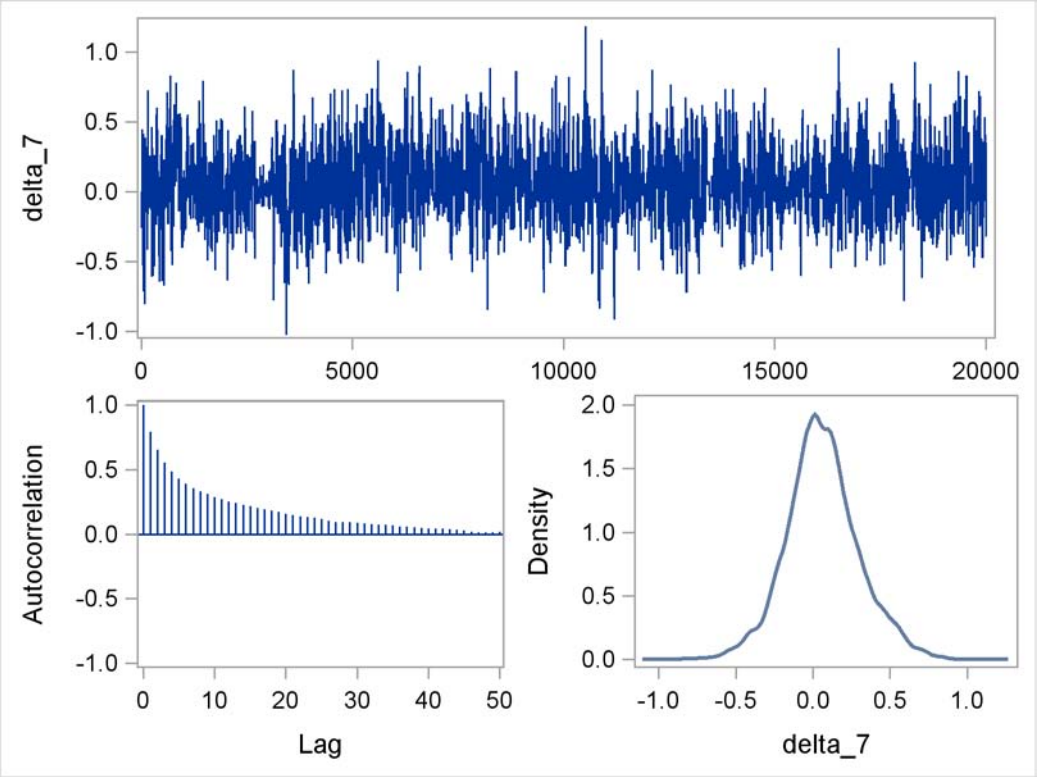
If you want to examine the convergence diagnostics of some of the random-effects parameters, you can use the MONITOR= option in the RANDOM statement. Then PROC MCMC produces a visual display of the posterior samples. If you forgot to include the MONITOR= option in the initial program run, you can use the %TADPlot autocall macro after the fact to regenerate the same display.

Autocall macros are macros that are defined in libraries that SAS supplies; they are automatically included so that you can use them without having to define or include them in your programs. The %TADPlot macro requires you to specify two arguments: an input data set (such as the output data set from a PROC MCMC run) and a list of variables that you want to plot. The macro generates the following three default diagnostics plots in a panel for every variable on the list: the trace plot, the autocorrelation plot, and the kernel density plot.

The following statement calls the %TADPlot macro and generates a diagnostics plot for the random-effects parameter delta_7, which is shown in Output 7:

```
%tadplot(data=postout, var=delta_7);
```

Output 7 Regenerated Diagnostics Plots for delta_7



The diagnostics plot indicates good mixing for the random-effects parameter delta_7.

The following statement shows how to create convergence diagnostics plots for all the random-effects parameters:

```
%tadplot(data=postout, var=delta_:);
```

The colon (:) is a shorthand for all variables that start with delta_.

CATERPILLAR PLOTS

The caterpillar plot is a side-by-side bar plot of the 95% equal-tail posterior intervals for multiple parameters. Typically, you use the plot to visualize and compare the posterior distributions of a large number of parameters, where it could be confusing or even difficult to dissect differences if you choose to overlay the kernel densities over each other. Random-effects models tend to have large numbers of random-effects parameters, and you can use the %CATER autocall macro to create a caterpillar plot of these parameters.

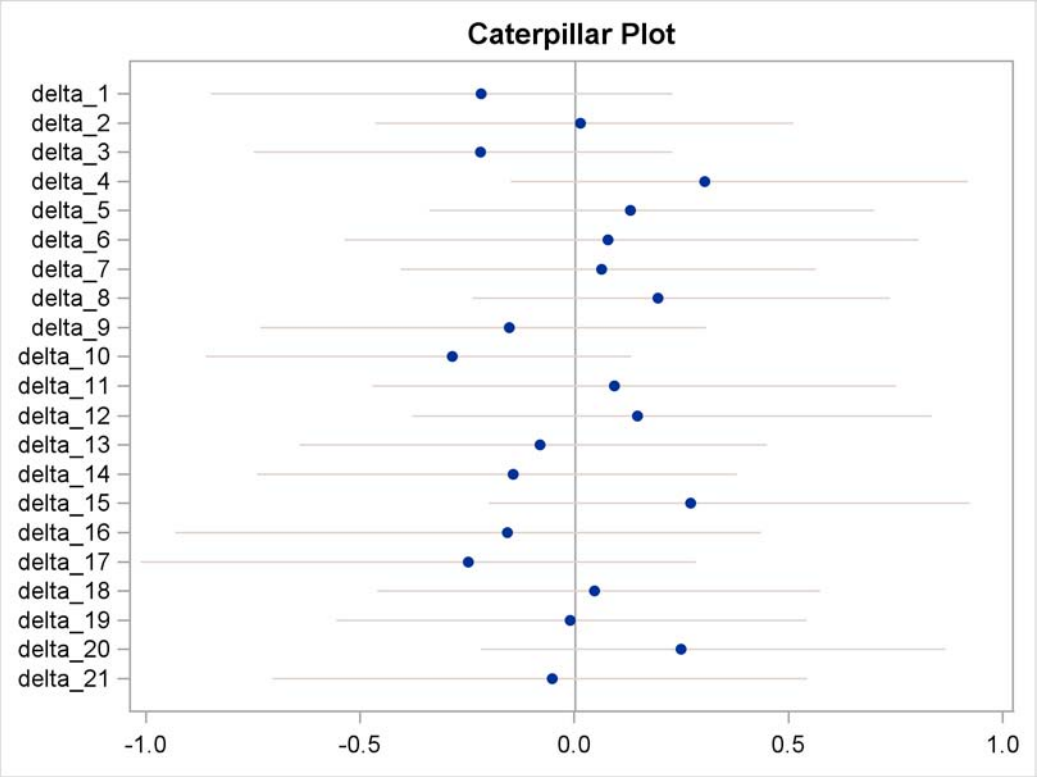
The %CATER macro's syntax is identical to that of the %TADplot macro: you specify the name of the input data set and a list of variable names.

The following statement takes the output from the random-effects analysis and generates a caterpillar plot of all the random-effects parameters:

```
%CATER(data=postout, var=delta:);
```

Output 8 is a caterpillar plot of the random-effects parameters delta_1–delta_21.

Output 8 Caterpillar Plot of the Random-Effects Parameters



Dots on the caterpillar plots are the posterior mean estimates for each of the parameters, and the vertical line in the middle of the graph is the overall mean, which is very close to 0. The figure shows that all random-effects parameters have posterior intervals that cover 0 and indicates that these parameter estimates do not significantly deviate from 0. However, this does not imply that a fixed-effects model fits the data as well as the random-effects model. The random-effects model accounts for variation between clusters (on the observational level in this analysis), something that a fixed-effects model cannot achieve. You can see this effect clearly in Output 6: on one hand, all the 95% predictive probabilities from the random-effects model cover the predicted mean from the logistic regression with only fixed effects (random effects that are close to 0); on the other hand, the random-effects model produces more suitable prediction intervals that vary across observations and account for overdispersion.

MULTIVARIATE DISTRIBUTIONS

A new feature of the MCMC procedure in SAS/STAT 9.3 is that you can specify array-based parameters and multivariate distributions in the prior or likelihood function. This section presents an overview of supported distributions and basic steps to set up a multivariate analysis.

Table 2 lists the multivariate distributions that you can use in PROC MCMC. All distributions are supported in the MODEL statement, meaning that you can use them to construct your likelihood function. All except the multinomial distribution are supported in the PRIOR and HYPERPRIOR statements. The restriction for the PRIOR and HYPERPRIOR statements implies that you cannot directly specify a multinomial prior distribution for a parameter. You can still declare a multinomial prior by using the **`general`** function and SAS programming statements that use the **`logmpdfmultinom`** function.

Table 2 Multivariate Distributions

Distribution	Definition
dirichlet(α)	Dirichlet distribution with parameter vector α, which must be a one-dimensional array of length greater than 1
iwish(ν, S)	Inverse Wishart distribution with ν degrees of freedom and symmetric positive definite scale array S
mvn(μ, Σ)	Multivariate normal distribution with mean μ and covariance Σ
multinom(p)	Multinomial distribution with probability vector p

You can apply a multivariate distribution only on either an array-based parameter (in the PRIOR or HYPERPRIOR statement) or an array that stores data set variables (in the MODEL statement).

The following example shows required steps for doing a multivariate analysis. Suppose that you are interested in estimating the mean and covariance of multivariate data using the following multivariate normal model:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \sim \text{MVN}\left(\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}\right)$$

You use the following independent prior on μ and Σ:

$$\begin{aligned} \mu &\sim \text{MVN}\left(\mu_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_0 = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix}\right) \\ \Sigma &\sim \text{iWishart}\left(\nu = 2, S = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) \end{aligned}$$

The following statements describe the steps needed to specify multivariate distributions in PROC MCMC:

```
array data[2] x1 x2;
array mu[2];
array Sigma[2,2];
array mu0[2] (0 0);
array Sigma0[2,2] (100 0 0 100);
array S[2,2] (1 0 0 1);
parm mu Sigma;
prior mu ~ mvn(mu0, Sigma0);
prior Sigma ~ iwish(2, S);
model data ~ mvn(mu, Sigma);
```

The first three ARRAY statements declare the data array (which stores the response variables x1 and x2), the mu array (mean), and the Sigma array (covariance). These arrays are the random variables used in the PRIOR and MODEL statements. The next three ARRAY statements declare hyperparameters (mu0, Sigma0, and S) which are all constant arrays. The parentheses ( ) after the ARRAY statement assign constant values to the arrays. The PARMS statement declare mu and Sigma to be model parameters. The PRIOR statement specifies the desired prior distributions, and the MODEL statement specifies the multivariate normal likelihood.

When applicable, PROC MCMC updates the multivariate parameters via conjugate sampling (see the section "CONJUGATE SAMPLING"). In cases where conjugacy does not apply, either a random walk Metropolis algorithm or an independent Metropolis algorithm is used to draw samples. The sampling algorithm information can be found in the "Parameters" table which is part of the analysis output.

CONJUGATE SAMPLING

Conjugate prior is a family of prior distributions in which the prior and the posterior distributions are of the same family of distributions. For example, if you model an i.i.d. random variable y_i using a normal likelihood with known variance σ^2,

$$y_i \sim \text{normal}(\mu, \sigma^2)$$

a normal prior on μ,

$$\mu \sim \text{normal}(\mu_0, \sigma_0^2)$$

is a conjugate prior because the posterior distribution of μ is also a normal distribution given y $= \{y_i\}$, σ^2, μ_0, and σ_0^2,

$$\mu|\text{y} \sim \text{normal}\left(\left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)^{-1}\cdot\left(\frac{\mu_0}{\sigma_0^2}+\frac{n\cdot\bar{y}}{\sigma^2}\right),\left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)^{-1}\right)$$

where n is the sample size and $\bar{y}$ is the sample avarage.

Conjugate sampling is efficient because the Markov chain can obtain samples from the target distribution directly. When appropriate, PROC MCMC uses conjugate sampling methods to draw conditional posterior samples. Table 3 lists scenarios that lead to conjugate sampling in PROC MCMC.

Table 3 Conjugate Sampling in PROC MCMC

Family of Distributions	Parameter	Prior
Normal with known μ	Variance σ^2	Inverse gamma family
Normal with known μ	Precision τ	Gamma family
Normal with known scale parameter (σ^2, σ, or τ)	Mean μ	Normal
Multivariate normal with known Σ	Mean $\boldsymbol{\mu}$	Multivariate normal
Multivariate normal with known $\boldsymbol{\mu}$	Covariance Σ	Inverse Wishart
Multinomial	$\boldsymbol{p}$	Dirichlet
Binomial/binary	p	Beta
Poisson	λ	Gamma family

In most, but not necessarily all, cases the distribution shown in the Family column in Table 3 refers to the likelihood function. This distribution is conditional on the parameter of interest, and it can appear in any level of the hierarchical model, including in the random-effects level.

PROC MCMC can detect conjugacy only if the model parameter (not a function or a transformation of the model parameter) is used in the prior and family distributions. For example, the following program leads to a conjugate sampler being used on the parameter mu:

```
parm mu;
prior mu ~ n(0, sd=1000);
model y ~ n(mu, var=s2);
```

However, if you modify the program slightly in the following way, although the conjugacy still holds in theory, PROC MCMC cannot detect conjugacy on mu because the parameter enters the normal likelihood function through the symbol w:

```
parm mu;
prior mu ~ n(0, sd=1000);
w = mu;
model y ~ n(w, var=s2);
```

In this case, PROC MCMC resorts to the default sampling algorithm, which is a random walk Metropolis based on a normal kernel.

Similarly, the following statements also prevent PROC MCMC from detecting conjugacy on the parameter mu:

```
parm mu;
prior mu ~ n(0, sd=1000);
model y ~ n(mu + 2, var=s2);
```

In a normal family, an often-used conjugate prior on the variance σ^2 is

```
igamma(shape=0.001, scale=0.001)
```

An often-used conjugate prior on the precision τ is

```
gamma(shape=0.001, iscale=0.001)
```

Exercise caution in using the **igamma** and **gamma** distributions because PROC MCMC supports both **scale** and **iscale** parameterizations in these distributions.

MULTIVARIATE NORMAL RANDOM-EFFECTS MODEL

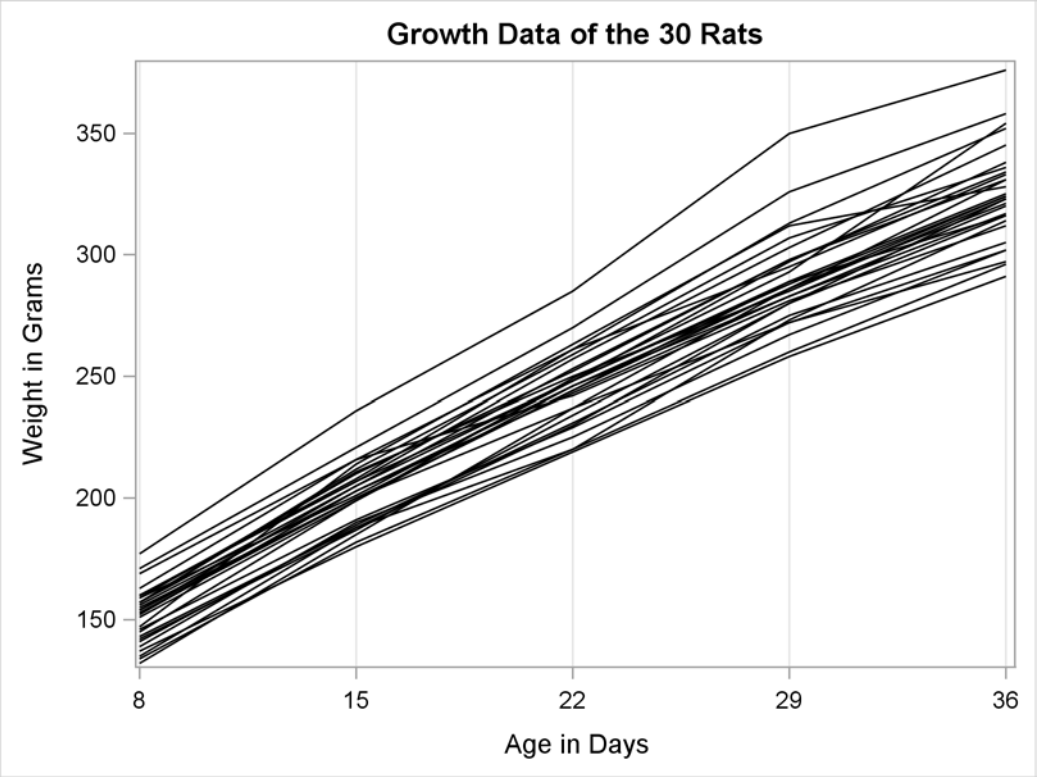
This example illustrates the fitting of a multivariate normal random-effects model. The analysis was carried out by Gelfand et al. (1990) to estimate regression coefficients for the growth of 30 young rats in a control group over a period of five weeks. You can also think of this as a repeated measure random-effects model, where the responses are observed over a period of few weeks.

The following statements create a SAS data set with measurements of Weight, Age (in days), and Subject.

```
data rats;
   array days[5] (8 15 22 29 36);
   input weight @@;
   subject = ceil(_n_/5);
   index = mod(_n_-1, 5) + 1;
   age = days[index];
   drop index days:;
datalines;
151 199 246 283 320 145 199 249 293 354
147 214 263 312 328 155 200 237 272 297
135 188 230 280 323 159 210 252 298 331
141 189 231 275 305 159 201 248 297 338
177 236 285 350 376 134 182 220 260 296
160 208 261 313 352 143 188 220 273 314
154 200 244 289 325 171 221 270 326 358
163 216 242 281 312 160 207 248 288 324
142 187 234 280 316 156 203 243 283 317
157 212 259 307 336 152 203 246 286 321
154 205 253 298 334 139 190 225 267 302
146 191 229 272 302 157 211 250 285 323
132 185 237 286 331 160 207 257 303 345
169 216 261 295 333 157 205 248 289 316
137 180 219 258 291 153 200 244 286 324
;
```

Output 9 shows the growth data of the 30 rats. Each profile is roughly linear, with a different starting weights and slightly varying growth rates.

Output 9 Profiles of the Rats' Growth Data



Gelfand et al. (1990) assume a normal model for the growth rate,

$$\text{Weight}_{ij} \sim \text{normal}\left(\alpha_i + \beta_i \text{Age}_{ij}, \sigma^2\right), \quad i = 1 \cdots 30; j = 1 \cdots 5$$

where i indexes rat (Subject variable), j indexes the time period, Weight_{ij} and Age_{ij} denote the weight and age of the ith rat in week j, and σ^2 is the variance in the normal likelihood. The individual intercept and slope coefficients are modeled as the following:

$$\boldsymbol{\theta}_i = \begin{pmatrix} \alpha_i \\ \beta_i \end{pmatrix} \sim \text{MVN}\left(\boldsymbol{\theta}_c = \begin{pmatrix} \alpha_c \\ \beta_c \end{pmatrix}, \boldsymbol{\Sigma}_c\right), \quad i = 1, \cdots, 30$$

You can use the following independent prior distributions on $\boldsymbol{\theta}_c$, $\boldsymbol{\Sigma}_c$, and σ^2:

$$\begin{aligned} \boldsymbol{\theta}_c &\sim \text{MVN}\left(\boldsymbol{\mu_0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma_0} = \begin{pmatrix} 1000 & 0 \\ 0 & 1000 \end{pmatrix}\right) \\ \boldsymbol{\Sigma}_c &\sim \text{iwishart}\left(\rho = 2, \boldsymbol{S} = \rho \cdot \begin{pmatrix} 0.01 & 0 \\ 0 & 10 \end{pmatrix}\right) \\ \sigma^2 &\sim \text{igamma}\,(\text{shape} = 0.01, \text{scale} = 0.01) \end{aligned}$$

The following statements fit this multivariate normal random-effects model:

```
proc mcmc data=rats nmc=10000 outpost=postout
   seed=17 init=random;
   ods select Parameters REParameters PostSummaries;
   array theta[2] alpha beta;
   array theta_c[2];
   array Sig_c[2,2];
   array mu0[2];
   array Sig0[2,2];
   array S[2,2] (0.02 0 0 20);

   begincnst;
   call zeromatrix(mu0);
   call identity(Sig0);
   call mult(Sig0, 1000, Sig0);
   endcnst;

   parms theta_c Sig_c {121 0 0 0.26} var_y;
   prior theta_c ~ mvn(mu0, Sig0);
   prior Sig_c ~ iwish(2, S);
   prior var_y ~ igamma(0.01, scale=0.01);

   random theta ~ mvn(theta_c, Sig_c) subject=subject
      monitor=(alpha_9 beta_9 alpha_25 beta_25);
   mu = alpha + beta * age;
   model weight ~ normal(mu, var=var_y);
run;
```

The ODS SELECT statement displays information about model parameters, random-effects parameters, and the posterior summary statistics. The ARRAY statements allocate memory space for the multivariate parameters and hyperparameters in the model. The parameters are θ (theta, where the variable name of each element is alpha or beta), θ_c (theta_c), and Σ_c (Sig_c). The hyperparameters are μ_0 (mu0), Σ_0 (Sig0), and S (S).

To assign a constant hyperparameter array, you can either use the parentheses ( )—for example, in the case of S—or you can use programming statements. If you use programming statements to set constant arrays, it is important to enclose the statements within the BEGINCNST/ENDCNST statements to reduce unnecessary computation. In the programming statement, you can either set an array to be 0 (such as the case with the parameter mu0 using **`call zeromatrix`**) or a diagonal matrix (such as the case with the parameter Sig0 using **`call identity`** and **`call mult`** subroutine calls).

The PARMS statement declares model parameters and assigns initial values to Sig_c using braces { }. The other two parameters (theta_c and theta_c) are left uninitialized, and PROC MCMC generates random values from their prior distribution at the beginning of the simulation. The PRIOR statements specify the prior distributions. The RANDOM statement defines an array random effect theta and specifies a multivariate normal prior distribution. The SUBJECT= option indicates cluster membership for each of the random-effects parameters. The MONITOR= option monitors the individual intercept and slope coefficients of subjects 9 and 25.

Output 10 Parameters and Random-Effects Parameters Information Tables

The MCMC Procedure

Parameters

Block	Parameter	Array Index	Sampling Method	Initial Value	Prior Distribution
1	theta_c1		Conjugate	-4.5834	MVNormal(mu0, Sig0)
	theta_c2			5.7930	
2	Sig_c1	[1,1]	Conjugate	121.0	iWishart(2, S)
	Sig_c2	[1,2]		0	
	Sig_c3	[2,1]		0	
	Sig_c4	[2,2]		0.2600	
3	var_y		Conjugate	2806714	igamma(0.01, scale=0.01)

Random Effects Parameters

Parameter	Subject	Levels	Prior Distribution
theta	subject	30	MVNormal(theta_c, Sig_c)

Output 10 displays the "Parameters" and "Random Effects Parameters" information tables. The Parameter column in the "Parameters" table lists element names of array parameters, as opposed to the names of array parameters. For example, you see theta_c1 and theta_c2 instead of theta. There is a new Array Index column in the same table, which lists the index reference of the elements in the 2-by-2 array parameters (Sig_c). The sampling algorithms used in this simulation for model parameters are all conjugate method, and the last two columns of the table list initial values and prior distributions. Note that the variable var_y is given an unusually large initial value. Scale parameters that are in the far tail of the distribution could lead to computational problems, such as convergence issues. You can always start the Markov chain at a more reasonable place if you think that the chain is not converging well. The "Random-Effects Parameters" table shows that the total number of random-effects parameters is 30.

Output 11 Multivariate Normal Random-Effects Model

The MCMC Procedure

Posterior Summaries

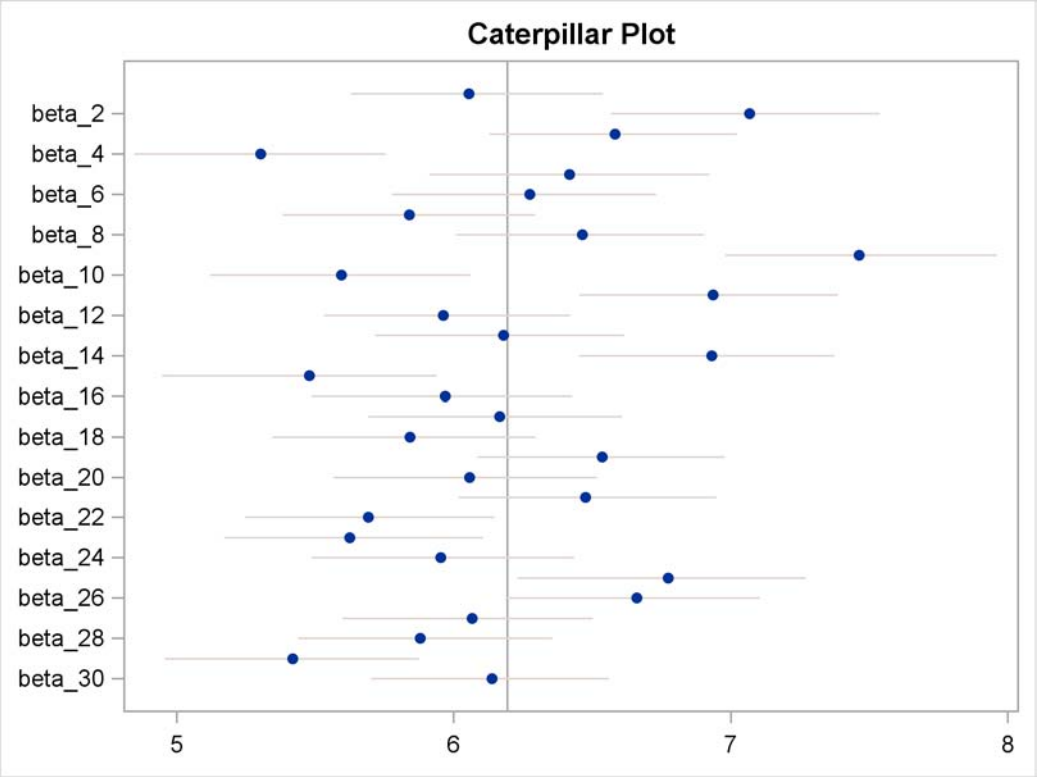
Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
theta_c1	10000	105.9	2.2768	104.4	105.9	107.5
theta_c2	10000	6.2006	0.1933	6.0730	6.2005	6.3268
Sig_c1	10000	110.0	45.0691	79.1442	103.2	133.4
Sig_c2	10000	-1.3895	2.2853	-2.6703	-1.2236	0.0374
Sig_c3	10000	-1.3895	2.2853	-2.6703	-1.2236	0.0374
Sig_c4	10000	1.0523	0.2983	0.8409	1.0008	1.2080
var_y	10000	37.4037	5.7093	33.4374	36.7871	40.8374
alpha_9	10000	119.4	5.8922	115.5	119.4	123.3
alpha_25	10000	86.6763	6.2424	82.6208	86.5522	90.8919
beta_9	10000	7.4628	0.2491	7.2932	7.4622	7.6230
beta_25	10000	6.7747	0.2633	6.5920	6.7855	6.9507

Output 11 displays posterior summary statistics for model parameters and the random-effects parameters for subjects 9 and 25. The posterior estimates Sig_c indicate that the random slopes and random intercepts are negatively correlated, meaning that heavier rats at the beginning of the trial tend to experience slower weight gain over the course of the time. This claim is supported in Output 9, which shows that a number of rats have growth curves that cross each other. Output 11 also shows a substantial difference in the intercepts and growth rates between the rats 9 and 25.

A quick %CATER macro call on the random intercepts (Output 12) shows uneven growth rates across the group.

```
%cater(data=postout, var=beta_:);
```

Output 12 Caterpillar Plot of the Random Intercepts



Rat number 9 seems to have the fastest growth rate and rat number 4 has the slowest. This result illustrates another advantage of the random-effects models: a fixed-effects model focuses on the coefficient effects after averaging subject-wise or cluster-wise differences, whereas a random-effects model can help identify groups that have unusual features, such as extremely high or low growth rates among the rats.

POSTERIOR PREDICTION SIMULATION

Suppose that you are interested in drawing samples from the posterior predictive distribution of the multivariate random-effects model on subjects 4 and 9 (the rats that have the slowest and fastest growth rate) at a future time (Age = 43 days). You can use the PREDDIST statement, which is available in SAS/STAT 9.22 and later.

The PREDDIST statement creates a new SAS data set that contains random samples from the posterior predictive distribution of the response variable. The posterior predictive distribution is the distribution of unobserved observations (prediction) conditional on the observed data. Let $\mathbf{y}$ be the observed data, $\mathbf{X}$ be the covariates, θ be the parameter, and $\mathbf{y}_{\text{pred}}$ be the unobserved data. The posterior predictive distribution is defined to be the following:

$$\begin{aligned} p(\mathbf{y}_{\text{pred}}|\mathbf{y},\mathbf{X}) &= \int p(\mathbf{y}_{\text{pred}},\theta|\mathbf{y},\mathbf{X})d\theta \\ &= \int p(\mathbf{y}_{\text{pred}}|\theta,\mathbf{y},\mathbf{X})p(\theta|\mathbf{y},\mathbf{X})d\theta \end{aligned}$$

With the assumption that the observed and unobserved data are conditionally independent given θ, the posterior predictive distribution can be further simplified as the following:

$$p(\mathbf{y}_{\text{pred}}|\mathbf{y},\mathbf{X}) = \int p(\mathbf{y}_{\text{pred}}|\theta)p(\theta|\mathbf{y},\mathbf{X})d\theta$$

The posterior predictive distribution is an integral of the likelihood function $p(\mathbf{y}_{\text{pred}}|\theta)$ with respect to the posterior distribution $p(\theta|\mathbf{y})$. The PREDDIST statement generates samples from a posterior predictive distribution based on draws from the posterior distribution of θ.

The PREDDIST statement works only on response variables that have standard distributions, and it does not support either the GENERAL or DGENERAL function. You can specify multiple PREDDIST statements.

The statement has the following syntax:

PREDDIST OUTPRED=*SAS-data-set* <**COVARIATES**=*SAS-data-set*> <**NSIM**=*n*>
<**STATISTICS**=*options*> **;**

The OUTPRED= option creates an output data set that stores the posterior predictive samples. The COVARIATES= option names the SAS data set that contains the sets of explanatory variable values for which the predictions are established. The NSIM= option specifies the number of simulated predicted values. And the STATISTICS= option specifies options for calculating posterior statistics.

The following programming statements create a data set (WeekSix) that has two observations, with a different SUBJECT value, indicating that you want to make a prediction on individual rat number four and number nine. A fixed AGE value of 43 (days) is used for each observation.

```
data WeekSix;
   age = 43;
   subject = 4; output;
   subject = 9; output;
run;
```

The following statements fit the model described previously in the "MULTIVARIATE NORMAL RANDOM-EFFECTS MODEL" section, request prediction, and save the posterior predictive mean estimates to the data set ps:

```
title 'Multivariate Normal Random-Effects Model';
proc mcmc data=rats nmc=10000 outpost=postout
   seed=17 init=random stats=summary diag=none;
   ods output predsummaries=ps;
   array theta[2] alpha beta;
   array theta_c[2];
   array Sig_c[2,2];
   array mu0[2] (0 0);
   array Sig0[2,2] (1000 0 0 1000);
   array S[2,2] (0.02 0 0 20);

   parms theta_c Sig_c {121 0 0 0.26} var_y;
   prior theta_c ~ mvn(mu0, Sig0);
   prior Sig_c ~ iwish(2, S);
   prior var_y ~ igamma(0.01, scale=0.01);

   random theta ~ mvn(theta_c, Sig_c) subject=subject;
   mu = alpha + beta * age;
   model weight ~ normal(mu, var=var_y);
   preddist outpred=WeekSixPred covariates=WeekSix;
run;
```

The PREDDIST statement invokes the prediction capability, uses the WeekSix data set, and saves the posterior predictive samples to the WeekSixPred data set.

The following statements combines the ps data set that stores posterior predictive values for the two rats at week 6 with their corresponding observations in the original rats data set, and generates a prediction plot using PROC SGPLOT. The result is shown in Output 13:

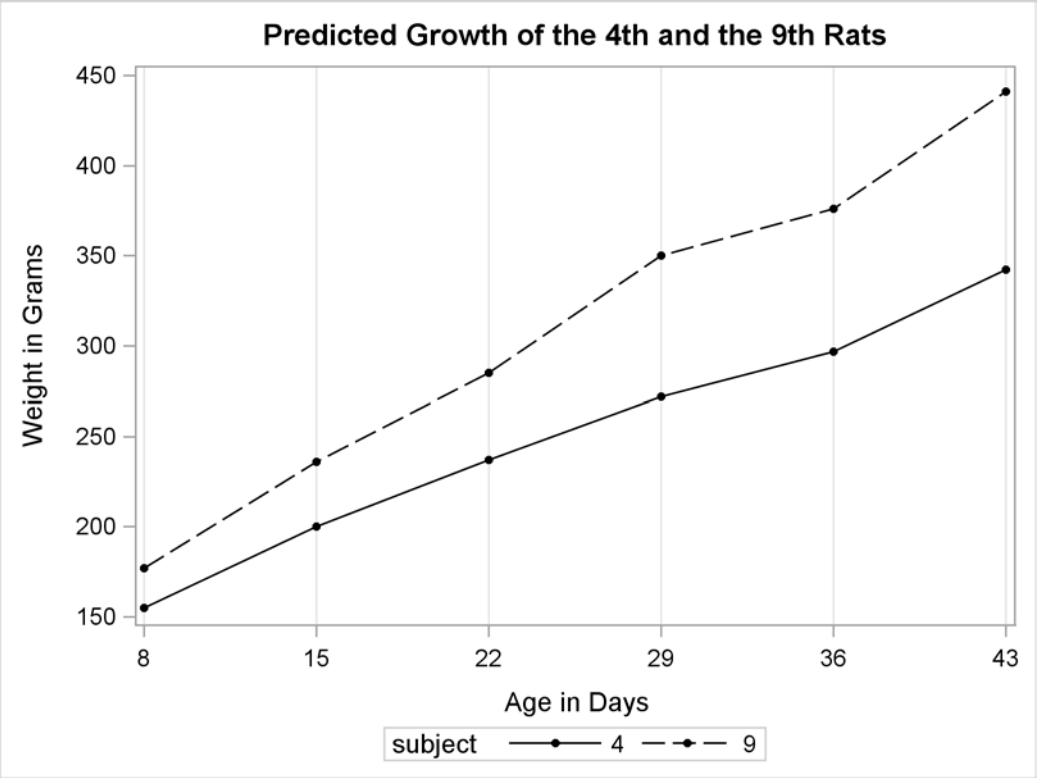
```
data predmn;
   merge ps(rename=(mean=weight)) weeksix;
   keep weight subject age;
run;

data pred;
   set rats(where=(subject in (4 9))) predmn;
run;

title 'Predicted Growth of the 4th and the 9th Rats';
proc sgplot data=pred;
   xaxis label="Age in Days" values=(8 15 22 29 36 43) grid;
   yaxis label="Weight in Grams";
   series x=age y=weight /group=subject markers
        lineattrs=(color=black)
        markerattrs=(color=black symbol=circlefilled size=4);
run;
```

Output 13 shows the growth profile and prediction for the two rats of interest. The first five data points are the observed data, and the sixth point (day 43) is the predicted growth.

Output 13 Plots of Observed Growth and Predicted Growth



As expected, the predicted weight discrepancy in week 6 is larger than that in week 5, due to different random slope estimates for the rats. If you think that there is nonlinearity in the data and that a linear growth model is not sufficient to capture all the variability, you can consider, among other possibilities, a logistic nonlinear mixed model (Lindstrom and Bates 1990 and Pinheiro and Bates 1995). You can fit those types of models by modifying linear regression code in PROC MCMC to accommodate nonlinearity.

CONCLUSION

In previous SAS/STAT releases, the MCMC procedure required more programming for fitting a random-effects model or specifying a multivariate distribution. In some cases (such as models with a large number of random-effects parameters or multivariate analysis that involves covariance matrices), the MCMC procedure might experience convergence problems in addition to sampling difficulties. In SAS/STAT 9.3, PROC MCMC offers major enhancements in both areas.

The RANDOM statement simplifies the specification of the random-effects parameter in a model by eliminating the need to predetermine cluster levels and allocate array memory to accommodate these parameters. The statement also significantly improves the computational and convergence performance, because it requires only a single pass through the data set to update all random-effects parameters from one RANDOM statement.

You can now directly specify multivariate distributions in PROC MCMC. The array-based syntax is intuitive, and it simplifies coding for multivariate analysis. The MCMC procedure implements conjugate sampling algorithms to solve some known sampling difficulties, such as drawing posterior samples from positive definite covariance parameters. Finally, the PREDDIST statement, available in SAS/STAT 9.22 and later, facilitates posterior predictions in a wide range of Bayesian models.

The MCMC procedure continues to be an active and crucial area of SAS/STAT development; future releases of PROC MCMC will provide additional features that improve the performance and usability of the software.

REFERENCES

Box, G. E. P. and Tiao, G. C. (1973), *Bayesian Inference in Statistical Analysis*, Wiley Classics Library Edition, published 1992, New York: John Wiley & Sons.

Breslow, N. E. and Clayton, D. G. (1993), "Approximate Inference in Generalized Linear Mixed Models," *Journal of the American Statistical Association*, 88, 9–25.

Chen, F. (2009), "Bayesian Modeling Using the MCMC Procedure," in *Proceedings of the SAS Global Forum 2009 Conference*, Cary, NC: SAS Institute Inc.

Congdon, P. (2010), *Applied Bayesian Hierarchical Methods*, Chapman & Hall/CRC.

Crowder, M. J. (1978), "Beta-Binomial Anova for Proportions," *Applied Statistics*, 27, 34–37.

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. and Hill, J. (2007), *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge, UK: Cambridge University Press.

Lindley, D. and Smith, A. (1972), "Bayes Estimates for the Linear Model," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 34, 1–41.

Lindstrom, M. J. and Bates, D. M. (1990), "Nonlinear Mixed Effects Models for Repeated Measures Data," *Biometrics*, 46, 673–687.

Pinheiro, J. C. and Bates, D. M. (1995), "Approximations to the Log-Likelihood Function in the Nonlinear Mixed-Effects Model," *Journal of Computational and Graphical Statistics*, 4, 12–35.

Zeger, S. and Karim, M. (1991), "Generalized Linear Models with Random Effects; A Gibbs Sampling Approach," *Journal of the American Statistical Association*, 412, 79–86.

CONTACT INFORMATION

The MCMC procedure requires SAS/STAT 9.2 and later. The PREDDIST statement requires SAS/STAT 9.22 and later. The RANDOM statement, multivariate distributions, and conjugate sampling algorithms require SAS/STAT 9.3. 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 at:

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