

# Advanced Hierarchical Modeling with the MCMC Procedure

Fang Chen and Maura Stokes, SAS Institute Inc.

## Abstract

Hierarchical models, also known as random-effects models, are widely used for data that consist of collections of units and are hierarchically structured. Bayesian methods offer flexibility in modeling assumptions that enable you to develop models that capture the complex nature of real-world data. These flexible modeling techniques include choice of likelihood functions or prior distributions, regression structure, multiple levels of observational units, and so on.

This paper shows how you can fit these complex, multilevel hierarchical models by using the MCMC procedure in SAS/STAT<sup>®</sup> software. PROC MCMC easily handles models that go beyond the single-level random-effects model, which typically assumes the normal distribution for the random effects and estimates regression coefficients. This paper shows how you can use PROC MCMC to fit hierarchical models that have varying degrees of complexity, from frequently encountered conditional independent models to more involved cases of modeling intricate interdependence. Examples include multilevel models for single and multiple outcomes, nested and non-nested models, autoregressive models, and Cox regression models with frailty. Also discussed are repeated measurement models, latent class models, spatial models, and models with nonnormal random-effects prior distributions.

## Introduction

Hierarchical models, also known as multilevel models or random-effects models, are particularly useful for analyzing data that are structured in groups. The groups can be natural clusters, demographic units, spatial neighbors, medical studies, individual subjects, and so on. Each group is assumed to have its own parameters that are different but related. A hierarchical model has the flexibility to capture and analyze complex data structure and the ability to account for and estimate effects from different groups. One of the strengths of the MCMC procedure is its ability to fit many types of hierarchical models that require a range of data and modeling assumptions. This paper demonstrates how you can use PROC MCMC to model cluster-based data.

To learn the basics of PROC MCMC, see Chen (2009), Chen (2011), and Chen (2013). Collectively, the three papers introduce the essential usage of PROC MCMC, including fitting fixed-effects models and simple random-effects models, handling missing data, and carrying out inferences and predictions. Chen, Brown, and Stokes (2016) offers guidance on how you can use PROC MCMC to perform Bayesian analysis in models that are often fitted using mixed modeling procedures (such as the MIXED and GLIMMIX procedures). Chen, Brown, and Stokes (2016) covers key elements in handling mixed models, such as coding categorical variables, translating RANDOM and REPEATED statements, and fitting models that have various covariance types. This paper goes beyond Chen, Brown, and Stokes (2016) to examine a wider class of statistical models.

Hierarchical models is the subject of a large amount of literature and years of continued study by many statisticians. For overviews on this topic, see the textbooks by Congdon (2010) and Gelman and Hill (2007), which include many other references. This paper provides only a basic coverage of hierarchical models and aims to provide guidance on how to use PROC MCMC to fit these models that have different levels of complexity and modeling assumptions. The paper is organized as follows.

The section “[Multilevel Random-Effects Model](#)” demonstrates how to fit multilevel models that have the most commonly encountered assumptions: the data are assumed to be conditionally independent given the random effects, and the units of random effects across groups are conditionally independent given their shared hyperparameters. Such assumptions are used in a wide range of applications in which cluster samples are drawn according to a hierarchy, which can be nested (such as students within classrooms) or non-nested (such as patients who go to different clinics). This section uses a series of examples to show how to use PROC MCMC to fit variations of multilevel random-effects models, including models that have a single response or multiple responses and models that use nonnormal priors. The section also covers inferences and predictions.

The section “[Dynamic and Autoregressive Models](#)” describes fitting models in which some of the conditional independence assumptions are relaxed. Here, the random-effects parameters are assumed to be correlated, and the

correlation has an autoregressive nature. A dynamic linear model is used in this section to demonstrate PROC MCMC's capability to handle this class of models, which require specific structured correlation among the random effects. The approach described is general and can be extended to all latent variable models that have similar Markov structures.

The section “[Random-Effects Models for Data with Complex Dependence](#)” discusses modeling data that have a more complex structure: the observations are dependent on each other and the random effects within the same group are also dependent on each other. In many situations, prespecified and static correlation structures are not sufficient to satisfy all the modeling requirements. Dependencies can be data-specific and arbitrary. This section examines this situation in more depth and uses Cox regression with frailty as an example. The general idea of modeling such data can be extended to other applications, such as network meta-analysis.

The section “[Further Applications](#)” includes illustrative references that are intended to provide guidelines for handling common situations that arise from hierarchical modeling. Additional material describes how to use PROC MCMC to incorporate non-nested random effects, model repeated observations, account for heteroscedasticity, handle longitudinal data, or include spatial structure.

## Multilevel Random-Effects Model

This section describes how to use PROC MCMC to fit a multilevel random-effects models. In this class of models, the data,  $y_{ij}$  (which represents  $i$ th observation in the  $j$ th group), are assumed to be conditionally independent given the random effects ( $\theta_j$ ) or other random effects in the model that appear higher than the data in the hierarchy. Further, the units of the random effects,  $\theta_j$ , are assumed to be conditionally independent of each other given their shared hyperparameters. This type of data occurs most frequently in practice and is the type of data that can be handled in PROC MCMC with ease—by default, both the MODEL and RANDOM statements assume this type of conditional independence on the random variables (in the data or random-effects parameters).

The following data were collected from a two-arm vaccine trial for safety and immunogenicity. A total of 296 subjects were divided into two groups, with 132 subjects in the control arm and 148 subjects in the treatment arm. The outcome is the count of adverse experiences (such as fatigue, crying, insomnia, cough, and so on). Forty types of adverse experience (AE) were categorized into eight body systems (BdSys). For more information about this **TwoArms** data set, see Mehrotra and Heyse (2004).

The first few records of the **TwoArms** data set are shown as follows:

```
data TwoArms;
  nt = 148;
  nc = 132;
  format term $28.;
  input AE BdSys yt yc term $;
datalines;
  1 1 57 40 Astenia/fatigue
  2 1 34 26 Fever
  3 1 2 0 Infection-fungal
  4 1 3 1 Infection-viral
  ... more lines ...
```

The **term** variable is the name of the illness or the adverse experience. The variable **AE** is the numeric coding of the **term** variable. These subject variables are equivalent in specifying a model in PROC MCMC, and you can use them interchangeably in a RANDOM statement. The **BdSys** represents the body systems. The treatment outcome is **yt**, and the control outcome is **yc**. The variables **nc** and **nt** are the number of subjects in the control and treatment groups, respectively. The total number of subjects are fixed in this example, but they can vary in other scenarios.

This section illustrates how to use PROC MCMC to fit different types of a multilevel hierarchical random-effects model. The text and model specifications draw from Berry and Berry (2004).

## Multilevel Model for Single Response

First consider a model that has a single response variable,  $\mathbf{yc}$ . This variable is modeled using a binomial likelihood,

$$\mathbf{yc}_{bj} \sim \text{binomial}(\mathbf{nt}, c_{bj})$$

where  $b = 1, \dots, 8$  labels the body system (**BdSys**) and  $j = 1, \dots, k_b$  labels the adverse experience within the body system  $b$ . The probability  $c_{bj}$  is a logistic transformation of the **AE**-level random effects  $\gamma_{bj}$ :

$$c_{bj} = \text{logit}(\gamma_{bj})$$

The first-level random effects  $\gamma_{bj}$  have a normal prior distribution:

$$\gamma_{bj} \sim \text{normal}(\mu_{\gamma b}, \sigma_{\gamma}^2)$$

The second level is the body system, and this stage models the hyperparameters of  $\gamma_{bj}$ . The hyperparameters include the **BdSys**-level random effects  $\mu_{\gamma b}$  and parameter  $\sigma_{\gamma}^2$ . They have the following prior distributions, where  $b = 1, \dots, 8$ .

$$\mu_{\gamma b} \sim \text{normal}(\mu_{\gamma 0}, \tau_{\gamma 0}^2)$$

$$\sigma_{\gamma}^2 \sim \text{IG}(\alpha_{\sigma\gamma}, \beta_{\sigma\gamma})$$

The third level models the hyper-hyperparameters (parameters of the hyperprior distributions) and have the following specifications:

$$\mu_{\gamma 0} \sim \text{normal}(\mu_{\gamma 00}, \tau_{\gamma 00}^2)$$

$$\tau_{\gamma 0}^2 \sim \text{IG}(\alpha_{\tau\gamma}, \beta_{\tau\gamma})$$

In PROC MCMC, you use RANDOM statements to specify the random effects of different levels (the  $\gamma_{bj}$  and  $\mu_{\gamma b}$  parameters). In addition, the RANDOM statement supports nested hierarchy, which means that a random effect in the higher hierarchy (for example, the **BdSys**-level  $\mu_{\gamma b}$ ) can be hyperparameters to the random effects in the lower hierarchy (for example, the **AE**-level  $\gamma_{bj}$ ).

The hyperparameters  $\sigma_{\gamma}^2$ ,  $\mu_{\gamma 0}$ , and  $\tau_{\gamma 0}^2$  should be declared in the PARMs statement and assigned prior distributions in the PRIOR statement. You can choose fixed constants for the remaining parameters in the model specification; these include  $\alpha_{\sigma\gamma}$ ,  $\beta_{\sigma\gamma}$ ,  $\mu_{\gamma 00}$ ,  $\tau_{\gamma 00}^2$ ,  $\alpha_{\tau\gamma}$ , and  $\beta_{\tau\gamma}$ .

The following PROC MCMC statements fit this model:

```
proc mcmc data=TwoArms nmc=20000 seed=12707 outpost=OutYc;
  parm sig2_g mu_g0 tau2_g0;
  prior mu_g0 ~ normal(0, sd=100);
  prior sig2_g tau2_g0 ~ igamma(shape=3, scale=1);

  random mu_gb ~ normal(mu_g0, var=tau2_g0) subject=BdSys;
  random gamma_bj ~ normal(mu_gb, var=sig2_g) subject=AE;
  c_bj = logistic(gamma_bj);
  model yc ~ binomial(nc, c_bj);
run;
```

Figure 1 shows the fit of the model.

**Figure 1** PROC MCMC Posterior Summaries for the Control Response

Posterior Summaries and Intervals					
Parameter	N	Standard		95%	
		Mean	Deviation	HPD Interval	
sig2_g	20000	1.9635	0.6116	0.9074	3.1670
mu_g0	20000	-3.8577	0.3626	-4.5668	-3.1338
tau2_g0	20000	0.3790	0.2310	0.0891	0.8044

The mean of the **BdSys**-level random effects **mu\_g0** is a large negative value, indicating almost no body system effects on the binomial probability (the logistic transformation of a large negative value is close to 0). This is not surprising because only a response variable from the control group is used. By default, the posterior summary statistics of the random-effects parameters (**mu\_gb**: and **gamma\_bj**:) are not displayed. The posterior samples of the random-effects parameters are saved automatically to the OUTPOST= data set. You can also use the MONITOR= option in the RANDOM statement to output posterior summary statistics and diagnostics of specific random-effects parameters in a PROC MCMC call.

### Multilevel Model for Multiple Responses

Once you learn how to use PROC MCMC to fit a multilevel model to one response variable, it is relatively straightforward to extend the program to include another response variable (**yt**). The model specification for **yt** is parallel to that for **yc**. The probability of treatment success is denoted as  $t_{bj}$ ,

$$t_{bj} = \text{logit}(\gamma_{bj} + \theta_{bj})$$

where the  $\theta_{bj}$  are the **AE**-level treatment effects (log odds ratios). The rest of the model specification mirrors the hierarchy for the control outcome:

$\theta_{bj} \sim \text{normal}(\mu_{\theta b}, \sigma_{\theta}^2)$	First-level random effects
$\mu_{\theta b} \sim \text{normal}(\mu_{\theta 0}, \tau_{\theta 0}^2)$	Second-level random effects
$\sigma_{\theta}^2 \sim \text{IG}(\alpha_{\sigma\theta}, \beta_{\sigma\theta})$	Second-level hyperparameter
$\mu_{\theta 0} \sim \text{normal}(\mu_{\theta 00}, \tau_{\theta 00}^2)$	Third-level hyper-hyperparameter
$\tau_{\theta 0}^2 \sim \text{IG}(\alpha_{\tau\theta}, \beta_{\tau\theta})$	Third-level hyper-hyperparameter

You can use multiple MODEL statements to specify different likelihood functions for different response variables. Therefore, to extend the previous program, you add one additional MODEL statement for the **yt** variable and two more RANDOM statements to model the nested random effects on the treatment side. The random effect (**gamma\_bj**) is the baseline effect, which enters both likelihood functions; the random effect (**theta\_bj**) is the treatment effect, which enters only the treatment likelihood function.

The following PROC MCMC statements fit the model:

```
proc mcmc data=TwoArms nmc=20000 seed=12707 outpost=outT;
  parm sig2_g mu_g0 tau2_g0 sig2_t mu_t0 tau2_t0;
  prior mu_g0 mu_t0 ~ normal(0, sd=100);
  prior sig2_g tau2_g0 sig2_t tau2_t0 ~ igamma(shape=3, scale=1);

  random mu_gb ~ normal(mu_g0, var=tau2_g0) subject=BdSys;
  random gamma_bj ~ normal(mu_gb, var=sig2_g) subject=AE;

  random mu_tb ~ normal(mu_t0, var=tau2_t0) subject=BdSys;
  random theta_bj ~ normal(mu_tb, var=sig2_t) subject=AE;

  c_bj = logistic(gamma_bj);
  model yc ~ binomial(nc, c_bj);
  t_bj = logistic(gamma_bj + theta_bj);
  model yt ~ binomial(nt, t_bj);
run;
```

Figure 2 shows the fit of the model.

**Figure 2** PROC MCMC Posterior Summaries for Modeling Both Responses

Posterior Summaries and Intervals					
Parameter	N	Standard		95%	
		Mean	Deviation	HPD Interval	
<b>sig2_g</b>	20000	1.5786	0.4168	0.8669	2.4036
<b>mu_g0</b>	20000	-3.7850	0.3346	-4.4652	-3.1454
<b>tau2_g0</b>	20000	0.3737	0.2227	0.0915	0.7922
<b>sig2_t</b>	20000	0.2153	0.0837	0.0822	0.3796
<b>mu_t0</b>	20000	0.3221	0.2539	-0.1758	0.8306
<b>tau2_t0</b>	20000	0.2911	0.1615	0.0885	0.5993

The posterior estimates for parameters in the control group are approximately the same as the analysis that is based on only the control response. The posterior mean of **mu\_t0** is around 0.3, indicating a positive shift in probability of success over the baseline effect.

### Alternate Data Coding

Different data coding leads to different approaches of using PROC MCMC to model data that have multiple responses. In the previous example, the two responses are stored in two separate columns of the input data, with one variable for each response. When the data are stored in this format (a wide data set), you can use multiple MODEL statements for the data, matching one MODEL statement to one response variable. Suppose that instead of binomial distributions, you have a continuous outcome variable and you want to model the response variables by using a multivariate normal distribution. You can still use this approach by storing the variables in a wide data set and using a multivariate normal likelihood to model them jointly in a MODEL statement.

You might encounter data that are stored in a long data set, where one variable is used to represent all response variables. The data set might have an indicator variable that indicates the membership information of each response value. This type of long data set is frequently used in the MIXED and GLIMMIX procedures. These procedures support a single MODEL statement, and you use the REPEATED statement with a SUBJECT= variable to model multiple responses.

You can fit the same model that was described in the section “[Multilevel Model for Multiple Responses](#)” by using a long data set with minimal changes. The following DATA step modifies the **TwoArms** data set by combining the **yt** and **yc** variables into one **y** variable in the **TwoArmsL** data set:

```
data TwoArmsL(drop=nt nc yt yc);
  set TwoArms;
  y = yt; trt = 1; n = 148; output;
  y = yc; trt = 0; n = 132; output;
run;
```

The binary variable **trt** indicates whether an observation belongs to the treatment or the control group. Instead of two MODEL statements for two response variables, you can use DATA step statements in PROC MCMC to construct success probabilities for different observations in the data set and use a single MODEL statement to model the response:

```
p = logistic(gamma_bj + trt * theta_bj);
model y ~ binomial(n, p);
```

The specifications of the random effects and the hyperparameters remain the same, and you can use the following program to fit the same model to the **TwoArmsL** data set:

```
proc mcmc data=TwoArmsL nmc=20000 seed=12707;
  parm sig2_g mu_g0 tau2_g0 sig2_t mu_t0 tau2_t0;
  prior mu_g0 mu_t0 ~ normal(0, sd=100);
  prior sig2_g tau2_g0 sig2_t tau2_t0 ~ igamma(shape=3, scale=1);

  random mu_gb ~ normal(mu_g0, var=tau2_g0) subject=BdSys;
  random gamma_bj ~ normal(mu_gb, var=sig2_g) subject=AE;
```

```

random mu_tb ~ normal(mu_t0, var=tau2_t0) subject=BdSys;
random theta_bj ~ normal(mu_tb, var=sig2_t) subject=AE;

p = logistic(gamma_bj + trt * theta_bj);
model y ~ binomial(n, p);
run;

```

This analysis provides the same posterior estimates as the analysis in the section [“Multilevel Model for Multiple Responses”](#).

### Nonstandard Prior for Random Effects

Previous examples used the normal prior distribution on the random effects  $\theta_{bj}$ . An alternative mixture prior was proposed by Berry and Berry (2004) to indirectly infer the probability that  $\theta_{bj} = 0$ . This modeling enables you to directly infer whether the probability of a patient experiencing a particular adverse effect is the same in both arms of the study. The mixture prior has the following marginal distribution, where  $\alpha_\pi$  and  $\beta_\pi$  are hyperparameters:

$$\begin{aligned}\theta_{bj} &\sim \pi_b I_{[0]} + (1 - \pi_b) \cdot \text{normal}(\mu_{\theta_b}, \sigma_{\theta_b}^2) \\ \pi_b &\sim \text{beta}(\alpha_\pi, \beta_\pi)\end{aligned}$$

You cannot directly specify the mixture prior on  $\theta_{bj}$  in PROC MCMC because PROC MCMC requires that a parameter is either continuous (for example, a normal prior or the general prior) or discrete (for example, a Poisson prior or dgeneral prior). The parameter cannot be continuous sometimes and discrete other times. The solution is to reparametrize the marginal distribution and represent the mixture model by using a latent class model:

$$\begin{aligned}p_b &\sim \text{beta}(\alpha_\pi, \beta_\pi) \\ z_{bj} &\sim \text{binary}(p_b) \\ \pi(\theta_{bj}) &= \begin{cases} \text{normal}(\mu_{\theta_b}, \sigma_{\theta_b}^2) & \text{if } z_{bj} = 0 \\ P(\theta_{bj} = 0) = 1 & \text{if } z_{bj} = 1 \end{cases}\end{aligned}$$

The term  $p_b$  is the mixture probability for each **BdSys**, and  $z_{bj}$  is the latent variable that identifies the mixture component for each observation (that is, either  $\theta_{bj} = 0$  when the vaccine has no effect on adverse experience or  $\theta_{bj} \neq 0$  when the vaccine has an effect). The hyperparameters  $\alpha_\pi$  and  $\beta_\pi$  take on a truncated exponential prior (values greater than 1).

A random-effects model that uses a mixture prior on  $\theta_{bj}$  can be fitted using the following PROC MCMC statements:

```

proc mcmc data=TwoArms nmc=50000 seed=12707 outpost=outM;
parm sig2_g mu_g0 tau2_g0 sig2_t mu_t0 tau2_t0;
prior mu_g0 mu_t0 ~ normal(0, sd=100);
prior sig2_g tau2_g0 sig2_t tau2_t0 ~ igamma(shape=3, scale=1);

random mu_gb ~ normal(mu_g0, var=tau2_g0) subject=BdSys;
random gamma_bj ~ normal(mu_gb, var=sig2_g) subject=AE;

random mu_tb ~ normal(mu_t0, var=tau2_t0) subject=BdSys;
random theta_bj ~ normal(mu_tb, var=sig2_t) subject=AE;

parm a b;
prior a b ~ expon(scale=1, lower=1);
random p ~ beta(a, b) subject=BdSys;
random z ~ binary(p) subject=Term;

c_bj = logistic(gamma_bj);
model yc ~ binomial(nc, c_bj);
t_bj = logistic(gamma_bj + (1 - z) * theta_bj);
model yt ~ binomial(nt, t_bj);
run;

```

The added PARMs and PRIOR statements declare two hyperparameters of the beta prior (for  $p_b$ ) and assign the truncated exponential prior distribution to them. Two RANDOM statements are used to specify the  $p_b$  and the  $z_{bj}$ . In the `t_bj` assignment statement, the `(1-z)` trick is used to zero out the effect of  $\theta_{bj}$  when  $z_{bj} = 1$ .

This parametrization of the mixture prior estimates the probability that the treatment and control have the same effect on adverse experiences. You can estimate the probability of each AE by computing the posterior mean of each of the latent variables  $z_{bj}$ —for example, by using PROC MEANS as follows (results not shown):

```
proc means data=outM mean;
  var z;;
run;
```

This section illustrates how to fit models by using a nonnormal prior on the random effects. The marginal mixture prior distribution is reparameterized using latent variables, and the standard beta and binary prior distributions can be used to specify the same model in the RANDOM statement. In situations where the reparametrization is not possible, you can record the distribution and use the `general` function in the RANDOM statement. The `general` function in the RANDOM statement works the same way as the `general` function in the PRIOR or MODEL statement. Remember to always specify the density on a logarithmic scale, use the INIT= option in the RANDOM statement to set the initial values, and use the `dgeneral` function to specify discrete prior distributions.

### Odds Ratios and Risk Ratios

In many analyses, you want to provide measures such as odds ratios and risk ratios. These measures are functions of the random effects: the  $\theta_{bj}$  are the log odds ratios, and the ratios of the two probabilities,  $c_{bj}/t_{bj}$ , are the relative risk ratios. To carry out posterior inferences, all you need are the posterior samples of all the model parameters. Once you complete a PROC MCMC run, you can compute these quantities by using a DATA step to compute functions of the random effects as follows:

```
data ratios;
  set outm;
  array theta[40] theta;
  array gamma[40] gamma;
  array or[40];
  array rr[40];
  do i = 1 to 40;
    or[i] = exp(theta[i]);
    rr[i] = logistic(gamma[i]) / logistic(theta[i]);
  end;
  keep or: rr:;
run;
```

The `outm` data set is the data set that is specified in the OUTPOST= option from the model that uses the mixture prior on  $\theta_{bj}$ . The first two ARRAY statements store all the random-effects variables in the `theta` and `gamma` arrays. A DO loop computes the odds and risk ratios. To compute these posterior summary statistics, you can use either SAS<sup>®</sup> procedures (such as PROC MEANS) or autocall postprocessing macros (such as %SUMINT).

```
%sumint(data=ratios, var=or1-or4 rr1-rr4);
```

You can also use the %CATER autocall macro to display side-by-side 95% highest posterior density (HPD) intervals of these quantities as follows:

```
%cater(data=ratios, var=or:);
%cater(data=ratios, var=rr:);
```

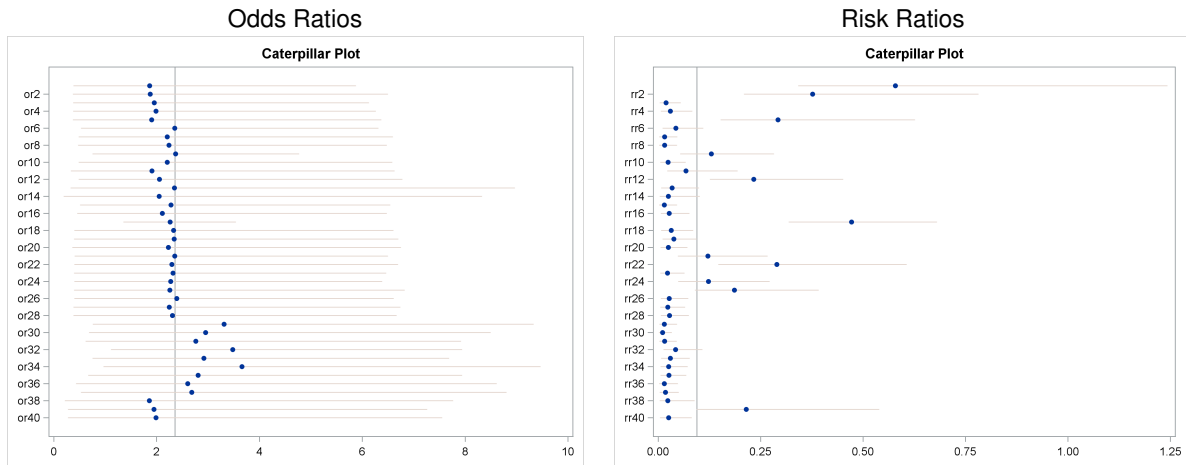
Figure 3 shows posterior estimates of some of the odds ratios and risk ratios.

**Figure 3** Odds Ratios and Risk Ratios

Posterior Summaries and Intervals					
Parameter	N	Mean	Standard	95%	
			Deviation	HPD Interval	
or1	50000	1.8641	1.7311	0.0610	4.2343
or2	50000	1.8759	1.7882	0.0845	4.7591
or3	50000	1.9554	1.7965	0.1137	4.7548
or4	50000	1.9875	2.3829	0.1046	4.7447
rr1	50000	0.5787	0.3299	0.2896	1.0027
rr2	50000	0.3768	0.2020	0.1781	0.6535
rr3	50000	0.0192	0.0189	0.00154	0.0455
rr4	50000	0.0298	0.0261	0.00328	0.0674

Figure 4 displays the odds ratios and relative risk ratios that are estimated from the model.

**Figure 4** Odds Ratios and Relative Risk Ratios



You can also compute functions of random-effects parameters directly in a PROC MCMC program, using a similar idea as in the DATA step. For example, you can insert the following four lines in a multilevel model program in PROC MCMC:

```
array or[40];
array rr[40];
or[ae] = exp(theta_bj);
rr[ae] = c_bt / t_bj;
```

In each of the simulation iterations, PROC MCMC steps through the input data set and computes the odds ratios and risk ratios based on every random-effects parameter. You add the `monitor=(or rr)` option in the PROC MCMC statement to save the transformations of the random variables. The results will be the same as those produced by the DATA step calculations, but you will have added some overhead to the PROC MCMC call. See Chen (2011) for a more detailed discussion about monitoring functions of random-effects parameters in a PROC MCMC call.

### Predictions

With hierarchical models, you might be interested in two different types of predictive distribution: the first is a new observation that corresponds to an existing random effect (for example, the number of positive outcomes in a new vaccine trial among subjects who report diarrhea in body system 2), and the second is a new observation that corresponds to new or unknown random effects (for example, the number of positive outcomes in a new vaccine trial). The first type of prediction is conditioned on known random effects because you are inferring from a particular unit; the second type of prediction is conditioned on the marginal distribution, where all the uncertainties with respect to the random effects have been integrated out.

You can use the PREDDIST statement in PROC MCMC to perform both types of predictions.



The following statements generate two prediction data sets, **pred1** and **pred2**, which correspond to the two types of prediction scenarios:

```

data pred1;
  nt = 100;
  nc = 100;
  format term $10.;
  input AE BdSys term $;
datalines;
  9 2 Diarrhea
;

data pred2;
  nt = 100;
  nc = 100;
;

```

The **pred1** data set represents the scenario of a future two-arm trial in which each arm has 100 subjects, who all report **AE=9** (Diarrhea) in **BdSys=2**. The **pred2** data set contains only the number of subjects in the trials, with no information about adverse experience or body systems.

You can add the following two statements to previous PROC MCMC programs, and the procedure draws posterior predictive samples of the treatment and control outcomes accordingly:

```

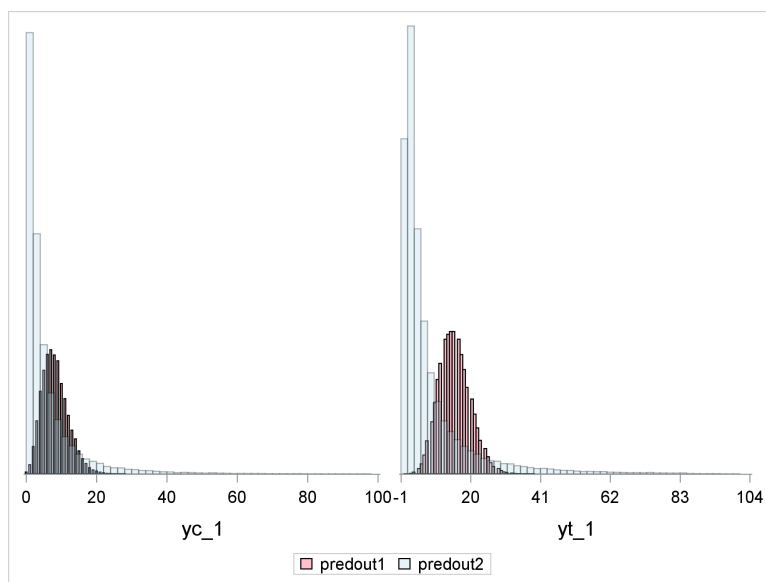
preddist covariates=pred1 outpred=predout1 stats=brief;
preddist covariates=pred2 outpred=predout2 stats=brief;

```

In predictions based on **pred1**, PROC MCMC draws new response values from a predictive distribution that is conditioned on the posterior samples of the random-effects parameters for **AE=9** and **BdSys=2**. In predictions based on **pred2**, PROC MCMC first generates the random-effects parameters from their marginal distributions (also conditional on the posterior samples of other model parameters), and then uses the newly sampled random-effects parameters in drawing samples from the predictive distributions for the response variables.

Figure 5 shows posterior histograms of predicted outcome in the two-arm trials. The variable **yc\_1** (left panel) is from the control arm, and the variable **yt\_1** (right panel) is from the treatment arm. The darker histograms display the expected outcomes from 100 subjects that have group information; the light histograms display the expected outcomes from 100 new subjects.

**Figure 5** Comparing Two Predictions



## Subject Identifiers in a Multilevel Model

The coding of the subject identifier variables is important when you use PROC MCMC to fit a multilevel random-effects model. There are different ways of representing subject levels, and each would entail different syntax specification. In the **TwoArms** analysis, the variable **AE** is the symptom identifier, which has a unique value for every symptom. You use **AE** as a SUBJECT= variable in the symptom-level random effects, and that specification results in 40 unique random-effects parameters for that effect.

In some data sets, the coding scheme can be different. For example, **AE** can be coded as repeats within the **BdSys** level as follows:

```
AE BdSys
  1  1
  2  1
  3  1
...
  1  2
  2  2
  3  2
  4  2
...
```

Many SAS users might be familiar with this type of data representation. The corresponding syntax for specifying nested models that is used in mixed modeling procedures (such as PROC GLIMMIX) is SUBJECT=AE(BdSys). This syntax ensures that there are 40 unique **AE**-level random effects. Because the RANDOM statement in PROC MCMC does not support nested-effects syntax, you must preprocess the input data set in order to fit the same nested hierarchical model. One simple solution is to use the DATA step to concatenate the **AE** and the **BdSys** variables:

```
UniqAE = trim(AE)||'_'||trim(BdSys);
```

You then use the new subject variable in the RANDOM statement for the **AE**-level random effects:

```
random theta_bj ~ normal(mu_tb, var=sig2_t) subject=UniqAE;
```

## Summary of Analysis of Multilevel Random-Effects models

This section illustrates how to use PROC MCMC to fit the most frequently encountered multilevel hierarchical models, with a conditional independence assumption imposed on both the data and the random effects. This type of model is easy to fit in PROC MCMC because both the MODEL statement and the PRIOR statement assume such conditional independence. The examples demonstrate how you can fit a four-level hierarchical model, and you can extend these examples to more complex scenarios. PROC MCMC does not restrict the number of random effects that you can specify, where or how they enter a model, or how many response variables that you can include. The flexibility in the syntax ensures that you can easily account for covariates information and use a combination of the RANDOM and MODEL statements with programming statements to represent your model of interest to any level of detail.

## Dynamic and Autoregressive Models

This section demonstrates how you can fit models in which some of the conditional independence assumptions are lifted. By default, a RANDOM statement assumes that all the random-effects parameters are independent of each other, *a priori*, conditional on the shared hyperparameters. Most commonly,  $\theta_j \sim \text{normal}(\mu, \sigma^2)$  and the hyperparameters are not functions of other random-effects parameters  $\theta_k$  for  $k \neq j$ . This assumption is often violated in models that involve a time variable. Common examples include autoregressive models or dynamic models. This section illustrates how to fit a hierarchical model when the prior of the random-effects parameter depends on other random-effects parameters.

The basic model described in this section is a dynamic linear model that models a univariate response. The idea of how to fit such model can be extracted and extended to other required scenarios of a similar autoregressive nature, including multivariate data. In its most basic representation, a dynamic linear model can be written as two models (an observation equation and a process equation),

$$\begin{aligned}y_t &= F_t z_t + v_t, & v_t &\sim \text{normal}(0, \sigma_v^2) \\z_t &= G_t z_{t-1} + \omega_t, & \omega_t &\sim \text{normal}(0, \sigma_\omega^2)\end{aligned}$$

where  $y_t$  are observations that are indexed by time  $t = 1, \dots, n$ , and  $F_t$  is the observation operator, which acts as a linear transformation function on the latent (unobserved) variable  $z_t$ . The latent variable  $z_t$  evolves according to the transition operator  $G_t$ , and it depends on hidden variables in the past. The errors are assumed to be independent. All the scale parameters can be generalized to matrix parameters, and there can be multiple time dependencies.

The dynamic linear model can be seen as a hierarchical model that has three levels: the first level is the likelihood function on the data  $y_t$ ; the second level is on the unknown states  $z_t$ , and the process equation specifies the dynamic part of the prior distribution; and the third level models the uncertainty that is related to the hyperparameters of the  $z_t$ . In PROC MCMC, you use the MODEL statement to specify the observation equation for the data, the RANDOM statement to specify the prior for the latent variables, and the PRIOR statement to specify the prior distributions of the hyperparameters to the random effects.

This type of prior distribution on the  $z_t$  requires access to lag values of random-effects parameters according to the time variable. Suppose that the effect variable is  $\mathbf{z}$  (the effect name in a RANDOM statement). Use the following rules to construct symbols to access lag or lead variables of  $\mathbf{z}$ :

$\mathbf{z}.Li$ : the  $i$ th lag of the variable  $\mathbf{z}$ , which looks back to the past.

$\mathbf{z}.Ni$ : the  $i$ th lead of the variable  $\mathbf{z}$ , which looks forward to the future.

This construction is allowed for random variables that are associated with an index. For random-effects parameters, the index corresponds to the SUBJECT= variable of the statement. Random-effects parameters from a RANDOM statement are indexed according to the order of the unique subject values. When a lag or a lead variable of order  $k$  is used in a PROC MCMC program, the variable is resolved according to the underlying index.

For example, the following RANDOM statement specifies a first-order Markov dependence in the random effect **theta**, which is indexed by the subject variable **time**:

```
random theta ~ normal(theta.l1, sd=1) subject=time;
```

This statement corresponds to the following prior:

$$\theta_t \sim \text{normal}(\theta_{t-1}, \text{sd}=1)$$

At each observation, PROC MCMC fills in the symbol **theta** with the random-effects parameter  $\theta_t$  that belongs to the current cluster  $t$ . To fill in the symbol **theta.l1**, the procedure looks back and finds a lag-1 random-effects parameter,  $\theta_{t-1}$ , from the last cluster  $t-1$ . As the procedure moves forward in the input data set, these two symbols are constantly updated as appropriate.

When the index is out of range, such as  $t-1$  when  $t$  is 1, PROC MCMC fills in the missing state from the ICOND= option in the RANDOM statement. For more information about the initial conditions, see the chapter "The MCMC Procedure" in *SAS/STAT User's Guide*.

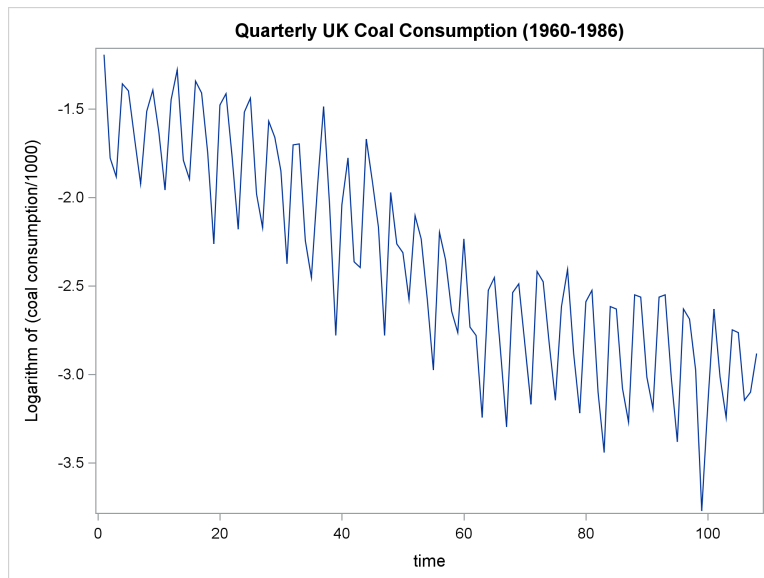
The following example fits a time-varying coefficients model to observed quarterly coal consumption in the United Kingdom for the years 1960–1986. It is based on an example that is published in Congdon (2003, p. 207), which in turn is based on a model and data that are published in Harvey (1989). The following statements create the data set **UKcoal**, which contains the variables **Coal**, **Y**, **Year**, **Quarter**, and **T**. The variable **Coal** measures quarterly coal consumption. **Y** is a variance-stabilizing transformation of **Coal**; specifically, it is the natural logarithm of **Coal** divided by 1,000. The variables **Year** and **Quarter** represent the years and quarters of the observations. The variable **T** indexes the observations.

```
data UKcoal;
  input coal year quarter @@;
  t=_N_;
  Y=log(coal/1000);
  datalines;
303 1960 1 169 1960 2 152 1960 3 257 1960 4
247 1961 1 189 1961 2 146 1961 3 220 1961 4
248 1962 1 195 1962 2 141 1962 3 235 1962 4
278 1963 1 167 1963 2 150 1963 3 261 1963 4
244 1964 1 174 1964 2 104 1964 3 228 1964 4
```

... more lines ...

Figure 6 displays the log-transformed coal consumption time series. It is reasonable to assume homogeneity of the variance of the series.

**Figure 6** Quarterly UK Coal Consumption (1960–1986) on the Logarithmic Scale



The following specification describes a smoothing model that appears in Congdon (2003, p. 207) for a time series that has seasonal effects and a secular trend:

$y_t \sim \text{normal}(z_t, \sigma_y^2)$	Observational model
$z_t = \mu_t + s_t$	Underlying trend after allowing for seasonal effects
$\mu_t \sim \text{normal}(\mu_{t-1} + \alpha_{t-1}, \sigma_\mu^2)$	Evolution in mean
$\alpha_{t-1} \sim \text{normal}(\phi\alpha_{t-2}, \sigma_\alpha^2)$	Increments in mean
$s_t \sim \text{normal}(-s_{t-1} - s_{t-2} - s_{t-3}, \sigma_s^2)$	Seasonality

The first expression is the observational model. The mean of the response (second expression) variable consists of a trend ( $\mu_t$ ) plus seasonal effects ( $s_t$ ). Both  $\mu_t$  and  $s_t$  depend on their past effects, in addition to the autoregressive drifting term  $\alpha_{t-1}$ .

To simplify the presentation without loss of generality, the initial conditions for  $s_t$  and  $\alpha_t$  are assumed to be zero when the index is out of range, and the initial condition for  $\mu_t$  (named  $\mu_0$ ) is treated as unknown and is estimated in the program. The normal prior with large variance is used on  $\phi$  and  $\mu_0$ , and the inverse gamma prior is used on the four variance terms.

The following statements use PROC MCMC to fit the time-varying coefficients model:

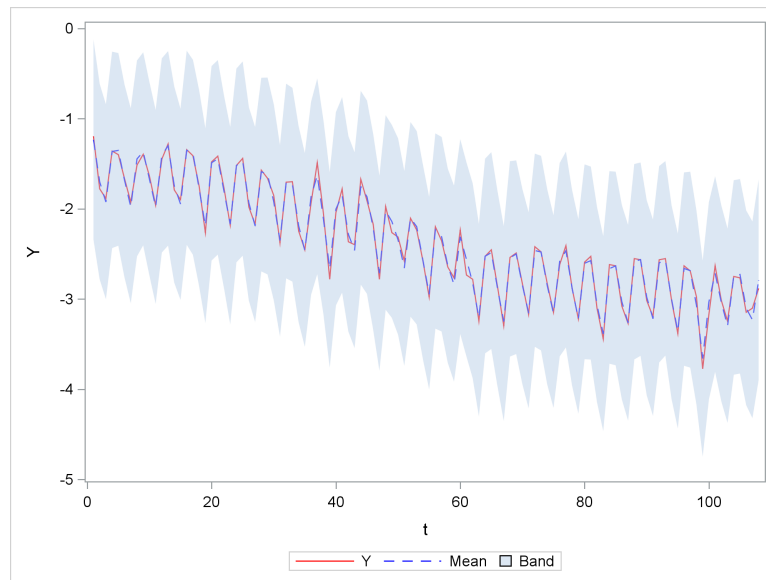
```
proc mcmc data=UKcoal nmc=50000 seed=112701 outpost=CoalOut propcov=quanew;
  parms phi mu0;
  parms sig2_a sig2_s sig2_m sig2_y;
  prior phi mu0 ~ normal(0,var=100);
  prior sig2: ~ igamma(shape = 3/10, scale = 10/3);
  random alpha ~ normal(phi*alpha.l1, var=sig2_a) subject=t;
  random s ~ normal(-s.l1-s.l2-s.l3, var=sig2_s) subject=quarter;
  random mu ~ normal(mu.l1 + alpha.l1, var=sig2_m) subject=t icond=(mu0);
  z = mu + s;
  model y ~ normal(z,var=sig2_y);
  preddist outpost=TVCoutpred;
```

```
ods output PredSumInt=TVCPredSumInt;
run;
```

The three RANDOM statements specify the three autoregressive latent variables, in  $\alpha_t$ ,  $s_t$ , and  $\mu_t$ . The lag variables enable you to look back and access different lag values of these variables, as specified in the statistical model. The PREDDIST statement requests sampling of the response variables. Because no data set is provided in the COVARIATES= option, PROC MCMC uses the default **UKcoal** data set for prediction. In this situation, the predicted samples of the fitted values are predicted.

Figure 7 shows the overlay of the predicted mean with observed data. The solid red line represents the original in-sample response variable, the dashed blue line represents the mean of the predictive distribution, and the light-blue shaded area represents the 95% HPD interval.

**Figure 7** Prediction of the Dynamic Linear Model



The predicted mean values follow the observed values closely (the solid and dashed lines are almost indistinguishable). You can also use PROC MCMC to make out-of-range predictions. For more information, see the chapter “The MCMC Procedure” in *SAS/STAT User’s Guide*.

## Random-Effects Models for Data with Complex Dependence

Although conditional independent models and Markov-structured models are pervasive in practice, some analyses require modeling assumptions to be further relaxed in order to capture the complex dependency among the data. In many such models—including Cox regression models, item response models, discrete choice models, network meta-analysis models, and so on—the likelihood function of an observation depends on other observations in an arbitrary case-specific fashion. To further complicate the modeling task, you often must build a hierarchical model on top in order to account for and infer cluster-level effects. This section illustrates how you can use PROC MCMC to fit a random-effects model that has complex dependence.

This example uses a leukemia data set from Freireich et al. (1963). This study includes 21 pairs of patient; within each pair, one patient receives the drug 6-mercaptopurine (6-MP), and the other patients receives a placebo. The following DATA step generates the data set **leuk**:

```
data leuk;
  input time event x id;
datalines;
  1 1 -0.5 1
  1 1 -0.5 2
  2 1 -0.5 3
  2 1 -0.5 4
  3 1 -0.5 5
```

```

4 1 -0.5 6
4 1 -0.5 7
... more lines ...

```

The **time** variable is the failure time, **event** is the censoring indicator variable, **x** is a covariate, and **ID** is the pairing indicator. The likelihood function used in this analysis is the Breslow likelihood function, which is generally expressed as

$$L(\beta) = \prod_{i=1}^n \left[ \prod_{j=1}^{d_i} \frac{\exp(\beta' \mathbf{Z}_j(t_i))}{\sum_{l \in \mathcal{R}_i} \exp(\beta' \mathbf{Z}_l(t_i))} \right]^{v_i}$$

where  $\beta$  is the vector of parameters,  $n$  is the number of patients,  $t_i$  is the  $i$ th time,  $\mathbf{Z}_l(t)$  is the vector of explanatory variables,  $d_i$  is the multiplicity of failures at  $t_i$ ,  $\mathcal{R}_i$  is the risk set for the  $i$ th time  $t_i$ , and  $v_i$  is the censoring indicator. The covariates  $\mathbf{Z}_l(t_i)$  can depend on the survival time but are assumed to be time-independent here. To extend the Breslow likelihood to the frailty model, you add a random effects  $\gamma_i$  to the regression mean of  $\beta' \mathbf{Z}_j(t_i)$ .

## Cox Models

First consider fitting a Cox regression without the frailty. Because the Breslow likelihood function is not a standard distribution, you must use the **general** function to specify the likelihood, on a logarithmic scale. The numerator involves a regression model and is easy to write as **beta \* x**, where **beta** is the regression coefficient. The denominator requires observations from the risk set, which involves patients who have equal or longer **time** values than the current observation. This is where the data dependency comes in—the likelihood value for any observation depends on other records. One solution is to transpose the data in such a way that each row of the data (for an individual patient) contains information from all patients who are in the individual patient's risk set. Although it is possible to implement this approach, doing so requires substantial preprocessing of the input data set, is prone to programming mistakes, and can be difficult to generalize. Another solution is to reformulate the survival model as a Poisson counting process and fit a Poisson model (Clayton 1991; Spiegelhalter et al. 1996). This solution also requires substantial preprocessing of the data.

The general approach taken here is to specify the Breslow likelihood function as it is and find a way to construct the likelihood for each observation by including records from other rows of the input data set. First, you use the **read\_array** function to save the data (for example, all **time** values) in an array. To construct a likelihood for each observation, you compare the current **time** value to all survival **time** values, identify patients in the risk set, and accumulate the denominator term.

The following statements illustrate how to use PROC MCMC to fit a Cox regression model:

```

proc mcmc data=leuk outpost=outCox nmc=20000 seed=176;
  array tAry[1] / nosym;
  array xAry[1 ] / nosym;
  begincnst;
  rc = read_array("leuk", tAry, "time");
  rc = read_array("leuk", xAry, "x");
  endcnst;

  parms beta 0;
  prior beta ~ normal(0, prec=1e-6);

  if(event eq 0) then
    loglike = 0;
  else do;
    b = beta * x;
    S = 0;
    do i = 1 to 42;
      if time <= tAry[i] then do;
        bc = beta * xAry[i];
        S = S + exp(bc);
      end;
    end;
  end;
end;

```

```

        loglike = (b - log(S));
    end;
    model general(loglike);
run;

```

The first two ARRAY statements allocate two dynamic arrays: the **tAry** array is used to store all the **time** values, and the **xAry** is used to store all the **x** values. The two **read\_array** function calls put all **time** values and **x** values from the **leuk** data set in the **tAry** and **xAry** arrays, respectively. These two statements are wrapped by the **BEGINCNST** and **ENDCNST** statements to reduce unnecessary iterationwise function calls. After the execution of these two statements, the **tAry** and **xAry** arrays are each 42 elements long.

Once you have access to the entire vectors of **time** and **x** values, the calculation of the log-likelihood function becomes straightforward. When the **event** value is equal to 0, the log-likelihood value is 0. Otherwise, you use a DO loop to compare the current **time** value with all **tAry** values. For patients who have longer survival times, you add the **exp(beta \* xAry[i])** term to the cumulated denominator symbol **S**. By the end of the loop, you compute the Breslow log likelihood.

### Cox Models with Frailty

This section extends the Cox model to incorporate frailty. The leukemia data arise from a paired design, and you can include the **ID**-specific group effects. Suppose  $\gamma_k$  is the frailty and  $k$  is the clustering index. The Breslow likelihood with frailty has a similar form to what was presented in the section “Cox Models with Frailty”, with the frailty entering the linear predictor as  $\beta'Z_j(t_i) + \gamma_k$ . As in the Cox models, the numerator term is easy to specify and you can add the random effects that belong to the current observation. The denominator is more complex and requires the inclusion of random-effects parameters from other patients in the risk set. Here, the likelihood of each observation depends not only on other observations but also on other random-effects parameters within the same group.

Unfortunately, you cannot use the **RANDOM** statement to specify a random effect and add the effect to the likelihood computation. The random effect (in the **RANDOM** statement) is matched to the current observation, and the only way to access random-effects parameters from other clusters is through the lag or lead variable specification. But the Cox frailty model requires that you have an arbitrary index to all other random-effects parameters, which is not achievable using only the effect in the **RANDOM** statement.

The solution is to use the **PARMS** statements to declare random-effects parameters from the **ID** cluster as parameters in the model and use the **PRIOR** statement to specify their prior distributions. Once you declare all random-effects parameters, you can use any subsets of these parameters to construct the log-likelihood function.

Suppose the random-effects parameters follow a normal distribution:

$$\gamma_i \sim \text{normal}(0, \text{var}=\sigma_\gamma^2)$$

You can use the following statements to specify a Cox regression with frailty in **PROC MCMC**:

```

proc mcmc data=leuk outpost=outFrailty nmc=50000 seed=176;
    array tAry[1] / nosym;
    array xAry[1 ] / nosym;
    array idAry[1] / nosym;
    begincnst;
    rc = read_array("leuk", tAry, "time");
    rc = read_array("leuk", xAry, "x");
    rc = read_array("leuk", idAry, "ID");
    endcnst;

    parms beta 0;
    prior beta ~ normal(0, prec=1e-6);
    array gamma[21];
    parms gamma1-gamma7;
    parms gamma8-gamma14;
    parms gamma15-gamma21;
    parms sig2 .1;
    prior gamma:~normal(0, v=sig2);
    prior sig2~igamma(1, scale=1);

```

```

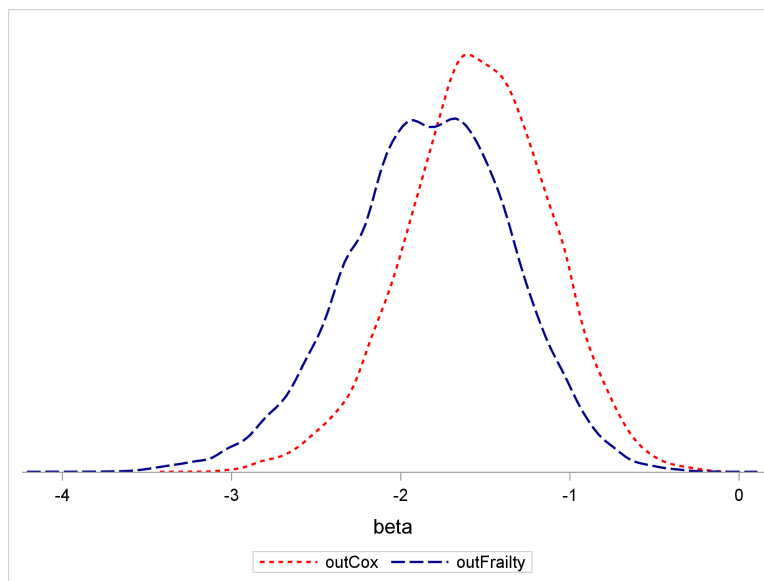
if(event eq 0) then
  loglike = 0;
else do;
  b = beta * x + gamma[id];
  S = 0;
  do i = 1 to 42;
    if time <= tAry[i] then do;
      tID = idAry[i];
      bc = beta * xAry[i] + gamma[tID];
      S = S + exp(bc);
    end;
  end;
  loglike = (b - log(S));
end;
model general(loglike);
run;

```

The **idAry** array stores all the **ID** values in the **leuk** data set. This array is used the same way as the covariate **xAry** array is used in constructing the denominator of the Breslow likelihood function: the elements of the **idAry** provide the right index of the random-effects parameters  $\theta_j$  for patients in each of the risk sets. The additional PARMs statements declare the 21 **gamma** random-effects parameters and their variance parameter **sig2**. All 21 **gamma** parameters are stored in the **gamma** array for ease of indexing. In the construction of the likelihood, the **gamma** random effect for the current observation is added to the linear predictor **beta \* x**; in the DO loop, and each patient in the risk set has a corresponding random effect **gamma[tID]** added to the **bc** term. The **tID** variable is the subject **ID** for that patient.

Figure 8 compares the two posterior densities of the **beta** parameter, one from the Cox model and one from the frailty model.

Figure 8 Cox Model versus Frailty Model



You probably have noticed some programming inefficiency in the construction of the denominator piece in the previous PROC MCMC program: the DO loop compares all 42 patients' survival time with the current observation, and some of these comparisons are clearly not needed. To reduce redundancy, you can first sort the input data set according to descending survival time. The risk set for each patient consists of observations that are above this patient in the sorted data set plus a few whose survival time is the same. In this way, you can stop the DO loop much earlier for many of the patients in the data set and increase the efficiency of the program. The following program uses a presorted data set to fit a Cox regression. The results are the same as the result that uses unsorted input data set.



```

proc sort data=leuk out=a;
  by Descending Time;
proc freq data=a;
  tables time / out=_freqs;
proc sort data = _freqs;
  by descending time;
run;

data LeukByTime;
  merge a _freqs(drop=percent);
  by descending time;
  ind = _n_;
  retain StopInd;
  if first.time then
    StopInd = _n_ + count - 1;
run;

proc mcmc data=LeukByTime outpost=outs nmc=50000 seed=176;
  array xA[1] / nosym;
  array idA[1] / nosym;
  begincnst;
  rc = read_array("LeukByTime", xA, "x");
  rc = read_array("LeukByTime", idA, "ID");
  endcnst;

  parms beta 0;
  prior beta ~ normal(0, prec=1e-6);

  array gamma[21];
  parms gamma1-gamma7;
  parms gamma8-gamma14;
  parms gamma15-gamma21;
  parms sig2 .1;
  prior gamma:~normal(0, v=sig2);
  prior sig2~igamma(1, scale=1);

  b = beta * x + gamma[id];
  loglike = 0;
  if event eq 1 then do;
    S = 0;
    do i = 1 to StopInd;
      gID = idA[i];
      bc = beta * xA[i] + gamma[gID];
      S = S + exp(bc);
    end;
    loglike = (b-log(S));
  end;
  model general(loglike);
run;

```

## Further Applications

This section describes additional applications and features of PROC MCMC in fitting multilevel random-effects models.

### Non-nested Models

As demonstrated in this paper, you can use multiple RANDOM statements to specify random effects according to different clustering. Because PROC MCMC processes each RANDOM statement independently, the clustering effects are not required to be nested (for example, students nested within classrooms) and the subject variables do not need to be sorted (PROC MCMC supports unsorted subject variables in both numeric or character format). When you have

crossed effects (for example, students taking classes from different teachers), you can specify two separate RANDOM statements in a program.

Suppose that the variable **teacher** is the teacher identifier and the variable **student** is the student identifier. Further, suppose that these identifiers have unique values for every observation, meaning that each student and each teacher have their own names. You can use the following statements to fit a non-nested model in PROC MCMC, where the effects **alpha** and **beta** can enter the model in linear or nonlinear form:

```
random alpha ~ normal(0, var=s2a) subject=teachers;  
random beta  ~ normal(0, var=s2b) subject=students;
```

However, with non-nested models, you cannot force a hierarchy among the random effects. For example, when **students** and **teachers** are non-nested, you cannot specify the following model, where the **alpha** random effect appears in the hyperparameter of the **beta**'s prior distribution:

```
random alpha ~ normal(0, var=s2a) subject=teachers;  
random beta  ~ normal(alpha, var=s2b) subject=students;
```

This type of model is not valid because a random-effects parameter from a student cannot have different prior means.

### Longitudinal Models

Longitudinal data refer to observations that are collected in studies in which individuals are measured repeatedly over time. See Chen (2013) for examples of how to fit variance components models with normal or multivariate normal likelihood. In that paper, it is assumed that, for a given individual, observations over time are independent given the variance or covariance parameters.

More commonly, individual-level observations over time can be assumed to be dependent (the response at time  $t$  might depend on the response at time  $t-1$ ), the outcome variables can be continuous or discrete (categorical or count variables), some responses might be missing, and the responses can have different numbers of repeats (unbalanced). You can fit these different types of model in PROC MCMC. The blueprint is to arrange all longitudinal data from a subject in a row and use programming statements to account for and model all subject-level dependencies.

To illustrate, suppose that the following three individuals have repeated measurements over the course of a few weeks:

- Individual 1 has measurements in weeks 1, 2, and 3
- Individual 2 has measurements in weeks 2, 3, 4, 6, and 8
- Individual 3 has measurements in weeks 5, 6, and 7

The data are stored as the following:

ind	y	week
1	2	1
1	5	2
1	5	3
2	4	2
2	5	3
2	1	4
2	3	6
2	9	8
3	4	5
3	2	6
4	6	7

You first want to transpose the response variable **y** by **ind** so that all responses from each individual are in the same row. Second, you want to make sure that the number of column variables matches the maximum time points in the data set (eight in this example). Because not every individual has the same number of responses, you will have missing values (.) in the data set.

Following is the representation of the response values from the three individuals with all responses stored in each record:

```

ind  Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8
  1   2  5  5  .  .  .  .  .
  2   .  4  5  1  .  3  .  9
  3   .  .  .  .  4  2  6  .

```

It is assumed here that the corresponding covariates (which translate to eight **x** variables for each covariate in the data set) are not missing. The covariates are not shown here.

In PROC MCMC, you use the ARRAY statements to read in all response variables from each record:

```

array y[8] y1-y8;
array t[8] 1-8;
array x[8] x1-x8;

```

For simplicity, times are assumed to be 1 to 8 here. But you can have **t1–t8** in your input data set, where each variable can represent a time point on a different scale.

Using the array **y** to store all the responses for a subject means that you can model the longitudinal part of the data in any way you see fit. For example, you can model lag dependency (for example, **y[i]** depends on **y[i–1]**), include random effects, impute missing responses (assuming that you have nonmissing covariates at these time points), and so on. You can either use standard distributions in separate MODEL statements to model the likelihood or construct a nonstandard likelihood function and specify it by using the **general** function.

In short, with complex longitudinal data that have unbalanced responses from each individual, organizing the input data set in a wide format gives you the most flexibility in handling different scenarios. You can start with a model that has only fixed effects and build up the hierarchy one level at a time to incorporate different random effects.

### Heteroscedasticity

Heteroscedasticity occurs when data do not exhibit constant variance. One strategy is to model the variance as a function of other variables. Assume the normal model,

$$Y_i \sim \text{normal}(\mu_i, \sigma_j^2)$$

where  $\mu_i$  is the mean and  $\sigma_j^2$  can depend on other data set variables, parameters, or random-effects parameters:

$$\sigma_j^2 = \exp(\beta_0 + \beta_1 \cdot \text{age}_j + \beta_2 \cdot \nu_j)$$

The following statements adjust heteroscedasticity with the covariate **age** and random effects **nu**:

```

model nu ~ normal(nu0, var=s2_nu) subject=cluster;
s2 = exp(b0 + b1 * age + b2 * nu);
model y ~ normal(mu, var=s2);

```

### Spatial Prior

The RANDOM statement supports a conditional autoregressive Gaussian prior. You can use it to model random effects that are spatially correlated. The autoregressive normal prior has the following density,

$$\theta_i | \theta_{-i} \sim \text{normal} \left( \sum_{j \in N(i)} \theta_j / m_i, s^2 \right)$$

where  $N(i)$  is the set of neighbors of area  $i$ ,  $m_i$  is the total number of neighbors of area  $i$ , and  $s$  is the standard deviation of the normal distribution.

Suppose that you are interested in modeling spatial correlation on the county level and that **county** is the county identifier in the data set. You can save the adjacency information in the following way in a SAS data set:

```

county num  ID1      ID2      ID3      ID4      ID5      ID6      ID7
Wake      6  Johnston Franklin Granville Durham Chatham Harnett .
Durham    5   Wake    Granville Person   Orange Chatham  .  .
...

```

The **county** variable indicates the county name. The **num** variable indicates the number of neighbors that each county has. The **ID** variables are the neighboring county names, which must be in the same format as the **county** variable. The **ID** variables share the same prefix and are followed by numbers 1, 2, and so on. In this data set, the maximum number of neighbors in any county in the data set is 7. A period (.) is used in the **ID** columns when a county has fewer than the maximum number of neighbors.

The following statement specifies a spatial prior on the random effects **a** on the **county**-level:

```
random a ~ normalcar(neighbors=ID, num=num, var=s2a) subject=county;
```

Here **county** is the subject variable, and the NEIGHBORS= option specifies the prefix to be used for data variables that contain the neighbor county names. PROC MCMC processes all **ID** variables and forms a prior that accounts for the desired spatial correlation among the random-effects parameters. The random effects **a** can then enter the model in any way you see fit.

## Conclusion

You can use PROC MCMC to fit a variety of Bayesian hierarchical models. The direct usage of MODEL and RANDOM statements assume conditional independence, which is the most prevalent assumption held in modeling clustered data. The procedure does not limit the number of random effects that you can include nor does it place a requirement on their nested hierarchy. Hence, you can fit many types of similar models that share the conditional independence assumption. If you want to model an autoregressive type of correlation structure, you can use the lag and lead variables in one-dimensional cases, or you can use the spatial prior to account for two-dimensional correlation between the random effects. In the most complex case, where you have complex dependence both at the data level and the random-effects level, you can use the **read\_array** function to store data in arrays and declare random-effects parameters in the PARMS statement. This approach gives you flexibility in using random indexing to account for these dependencies in your model.

Since its first release in SAS/STAT 9.2, the MCMC procedure has expanded considerably in its ability to fit complex and multilevel random-effects models. This paper summarizes some of these features. Development of PROC MCMC is an ongoing effort, and future development aims to further improve its performance and enhance its functionality.

## REFERENCES

- Berry, S. M., and Berry, D. A. (2004). "Accounting for Multiplicities in Assessing Drug Safety: A Three-Level Hierarchical Mixture Model." *Biometrics* 60:418–426.
- 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>.
- Chen, F., Brown, G., and Stokes, M. (2016). "Fitting Your Favorite Mixed Models with PROC MCMC." In *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. <https://support.sas.com/resources/papers/proceedings16/SAS5601-2016.pdf>.
- Clayton, D. G. (1991). "A Monte Carlo Method for Bayesian Inference in Frailty Models." *Biometrics* 47:467–485.
- Congdon, P. (2003). *Applied Bayesian Modeling*. New York: John Wiley & Sons.
- Congdon, P. (2010). *Applied Bayesian Hierarchical Methods*. Boca Raton, FL: Chapman & Hall/CRC.
- Freireich, E. J., Gehan, E., Frei, E., III, Schroeder, L. R., Wolman, I. J., Anbari, R., Burgert, E. O., Mills, S. D., Pinkel, D., Selawry, O. S., Moon, J. H., Gendel, B. R., Spurr, C. L., Storrs, R., Haurani, F., B., H., and Lee, S. (1963).

“The Effect of 6-Mercaptopurine on the Duration of Steroid-Induced Remissions in Acute Leukemia: A Model for Evaluation of Other Potentially Useful Therapy.” *Blood Journal* 21:699–716.

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

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge: Cambridge University Press.

Mehrotra, D. V., and Heyse, J. F. (2004). “Use of the False Discovery Rate for Evaluating Clinical Safety Data.” *Statistical Methods in Medical Research* 13:227–238.

Spiegelhalter, D. J., Thomas, A., Best, N. G., and Gilks, W. R. (1996). “BUGS Examples, Volume 1.” Version 0.5 (version ii).

## **RECOMMENDED READING**

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>.

## **CONTACT INFORMATION**

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](mailto:fangk.chen@sas.com)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.