Simulation-based Linear Mixed-effects Models with Stan

Yingqi Jing

July 22, 2025

Contents

List of Figures

List of Tables

In this blog, I will provide a step-by-step instruction of how we can generate data from a mixed effect model and recover the parameters of the model with the simulated dataset. This simulation-based experiment can help us better understand the structure and generative process of the multilevel model with correlated random intercepts and slopes. To proceed, I will first illustrate the general form of mixed effect models, and generate data based on a given set of design matrices and parameters (X, β, Z, b). In the end, I will set a Bayesian model to estimate the parameters on the simulated data via `stan`.

1 Matrix form of the mixed effect regression model

The general form of a mixed effect model can be illustrated in the following way:

$$y \sim X\beta + Zb + \epsilon$$

where y is a vector of dependent variables, and the two design matrices, X and Z, are related to fixed effects and random effects, respectively. The parameters of fixed effects are captured by β, and the random effects (e.g., random intercepts and slopes) are specified in b, which is generated from a Multivariate Normal (MN) distribution with mean 0 and variance-covariance matrix Σ. The random errors are indicated by ϵ.

To be more specific, I will expand the matrix form and provide a detailed description of the process for generating the fixed effects, random effects and response. As an example, I will assume a continuous predictor x and correlated random intercepts and slopes. This can also be expressed by the `lmer` style syntax as:

$$y \sim 1 + x + (x|group)$$

- Step 1: generate fixed effects $(X\beta)$

$$X\beta = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \dots & \dots \\ \dots & \dots \\ 1 & 9 \\ 1 & 10 \\ \dots & \dots \\ 1 & 1 \\ 1 & 2 \\ \dots & \dots \\ \dots & \dots \\ 1 & 9 \\ 1 & 10 \end{bmatrix} \times \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \\ \dots \\ \dots \\ 42 \\ 46 \\ \dots \\ 10 \\ 14 \\ \dots \\ \dots \\ 42 \\ 46 \end{bmatrix}$$

The design matrix X consists of two columns, where the first column of 1 is a dummy column related to the intercept and the second column is the predictor x ranging from 1 to 10 for each group. The parameters of β represent the intercept and slope at the population level.

- Step 2: generate random effects (Zb)

  Before simulating the correlated random intercepts and slopes, we first need to prepare the variance-covariance matrix Σ, which is created by multiplying a diagonal matrix, correlation matrix (R) and a diagonal matrix.

$$\begin{aligned} b &\sim \mathcal{N}(0, \Sigma) \\ \Sigma &= \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \\ &= \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \underbrace{\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}}_{R} \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \end{aligned}$$

With the variance-covariance matrix, we can generate the parameters b for random intercepts and slopes. The resulting matrix b includes two columns, corresponds to random intercepts and slopes, respectively. Each row of the matrix b indicates the group-specific random effects.

To generate the outcome of random effects, we need to make the design matrix Z and random effect parameter b in the following format. Of course, you do not need to stick with this

interleaved format of random intercepts. There are also some other ways for generating the random outcomes (see section 2 in the R script).

$$Zb = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 \\ 1 & 7 & 0 & 0 & 0 & 0 \\ 1 & 8 & 0 & 0 & 0 & 0 \\ 1 & 9 & 0 & 0 & 0 & 0 \\ 1 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 0 & 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 5 & 0 & 0 \\ 0 & 0 & 1 & 6 & 0 & 0 \\ 0 & 0 & 1 & 7 & 0 & 0 \\ 0 & 0 & 1 & 8 & 0 & 0 \\ 0 & 0 & 1 & 9 & 0 & 0 \\ 0 & 0 & 1 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 0 & 1 & 10 \end{bmatrix} \times \begin{bmatrix} Intercept_{g_1} \\ Slope_{g_1} \\ Intercept_{g_2} \\ Slope_{g_2} \\ Intercept_{g_3} \\ Slope_{g_3} \end{bmatrix} = \begin{bmatrix} 2.9 \\ 4.8 \\ 6.8 \\ 8.7 \\ 10.6 \\ 12.5 \\ 14.4 \\ 16.4 \\ 18.3 \\ 20.2 \\ -6.3 \\ -8.4 \\ -10.5 \\ -12.6 \\ -14.7 \\ -16.8 \\ -18.9 \\ -21 \\ -23 \\ -25 \\ 1.7 \\ 2.8 \\ 4 \\ 5.1 \\ 6.2 \\ 7.3 \\ 8.5 \\ 9.6 \\ 10.7 \\ 11.8 \end{bmatrix}$$

- Step 3: generate response by combining fixed, random outcomes and errors

  Now we can generate the response y by combining fixed, random outcomes and errors $\epsilon \sim \mathcal{N}(0, 1)$. This response will serve as the observations in the Bayesian hierarchical

model.

$$y = X\beta + Zb = \begin{bmatrix} 10 \\ 14 \\ 18 \\ 22 \\ 26 \\ 30 \\ 34 \\ 38 \\ 42 \\ 46 \\ 10 \\ 14 \\ 18 \\ 22 \\ 26 \\ 30 \\ 34 \\ 38 \\ 42 \\ 46 \\ 10 \\ 14 \\ 18 \\ 22 \\ 26 \\ 30 \\ 34 \\ 38 \\ 42 \\ 46 \end{bmatrix} + \begin{bmatrix} 2.9 \\ 4.8 \\ 6.8 \\ 8.7 \\ 10.6 \\ 12.5 \\ 14.4 \\ 16.4 \\ 18.3 \\ 20.2 \\ -6.3 \\ -8.4 \\ -10.5 \\ -12.6 \\ -14.7 \\ -16.8 \\ -18.9 \\ -21 \\ -23 \\ -25 \\ 1.7 \\ 2.8 \\ 4 \\ 5.1 \\ 6.2 \\ 7.3 \\ 8.5 \\ 9.6 \\ 10.7 \\ 11.8 \end{bmatrix} + \epsilon$$

2 R script for simulating the data

To make it easier, I also include the R scripts for you to generate the data based on the matrix format described above. Each block of scripts is corresponding to the previous steps for creating fixed, random and response outcomes.

- Step 1: generate fixed effects ($X\beta$)

```r
N_group = 3 # number of groups
N_elements_group = 10 # number of elements in each group
df = data.frame(intercept = 1,
                x = 1:N_elements_group,
                group_id= rep(1:N_group, each = N_elements_group),
                y = 0)
X = as.matrix(df[, c("intercept", "x")]) # design matrix X
fixed_outcome = X %*% c(6, 4) # X*beta
head(fixed_outcome)
```

```
     [,1]
[1,]   10
[2,]   14
[3,]   18
[4,]   22
[5,]   26
[6,]   30
```

- Step 2: generate random effects (Zb)

  It is worth noting here that the two design matrices, X and Z, are the same in our case, and we can thus generate the random outcome by using elementwise multiplication of X and a reformated b.

```r
rho = 0.7 # correlation coefficient
R = matrix(c(1, rho, rho, 1), ncol = 2) # R
sig1 = 3 # sd of random intercepts
sig2 = 2  # sd of random slopes
diag_mat_tau = matrix(c(sig1, 0, 0, sig2), ncol = 2) # diagonal matrix
Sigma = diag_mat_tau %*% R %*% diag_mat_tau # VCV
random_eff = mvrnorm(N_group, c(0, 0), Sigma)
head(random_eff)

b = random_eff[rep(seq_len(nrow(random_eff)), each = N_elements_group), ]
# Note: X and Z are the same in our case, and we use elementwise matrix multiplication
random_outcome = rowSums(X * b)
head(random_outcome)
```

```
            [,1]      [,2]
[1,] -6.69117740 -2.730157
[2,] -0.02339289 -0.824961
[3,]  4.73638182  1.248729
```

```
[1]  -9.421335 -12.151492 -14.881650 -17.611807 -20.341964 -23.072122
```

- Step 3: generate response by combining fixed, random outcomes and errors

```
df_final = df %>%
  mutate(fixed_outcome = fixed_outcome,
         random_outcome = random_outcome,
         y = fixed_outcome + random_outcome + rnorm(nrow(df), 0, 1))
head(df_final)
```

```
  intercept x group_id         y fixed_outcome random_outcome
1         1 1        1 0.2480045             10      -9.421335
2         1 2        1 2.1291830             14     -12.151492
3         1 3        1 3.2603544             18     -14.881650
4         1 4        1 4.3852543             22     -17.611807
5         1 5        1 6.2501398             26     -20.341964
6         1 6        1 5.9388941             30     -23.072122
```

3 Fit the linear mixed effect regression model with stan

With the simulated dataset, we can try to recover the parameters of the hierarchical model with correlated random intercepts and slopes. Here I am using `stan` to build the model and run the analysis via NUTS sampler. The structure of the model can be summarised below. Note that I didn't decompose the correlation matrix (R) via Cholesky factorization (see this blog about Multivariate Normal distribution and Cholesky decomposition). Instead, I am using a LKJ correlation prior for R.

3.1 Form

$$
\begin{aligned}
y &\sim \mathcal{N}(\mu_n, \sigma^2) \\
\mu_n &= \underbrace{\alpha_0 + \beta_0 x_n}_{fixed\ \ effects} + \underbrace{\alpha_{j[n]} + \beta_{k[n]} x_n}_{random\ \ effects} \\
\begin{pmatrix} \alpha_j \\ \beta_k \end{pmatrix} &\sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma \right) \\
\Sigma &= \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \\
&= \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \underbrace{\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}}_{R} \begin{pmatrix} \sigma_\alpha & 0 \\ 0 & \sigma_\beta \end{pmatrix} \\
R &\sim \mathcal{LKJ}corr(2.0) \\
\alpha_0 &\sim \mathcal{N}(0, 10) \\
\beta_0 &\sim \mathcal{N}(0, 10) \\
\alpha_j &\sim \mathcal{N}(0, 10) \\
\beta_k &\sim \mathcal{N}(0, 10) \\
\sigma &\sim \mathcal{N}(0, 5) \\
\sigma_\alpha &\sim \mathcal{N}(0, 5) \\
\sigma_\beta &\sim \mathcal{N}(0, 5)
\end{aligned}
$$

3.2 Model

3.3 Run the analysis

```
ls_final = list(N = nrow(df_final),
                x = as.vector(df_final$x),
                y = as.vector(df_final$y),
                N_group = length(unique(df_final$group_id)),
                group_id = df_final$group_id)
lmer_md = stan_model(file = "./stan_model/lmer_prior_R.stan")
fit_lmer_stan = sampling(lmer_md, data = ls_final,
                         chains = 2, iter = 3000, cores = 2)
```

3.4 Traceplot

```
rstan::traceplot(fit_lmer_stan, pars = c("alpha_0", "beta_0", "random_group"))
```

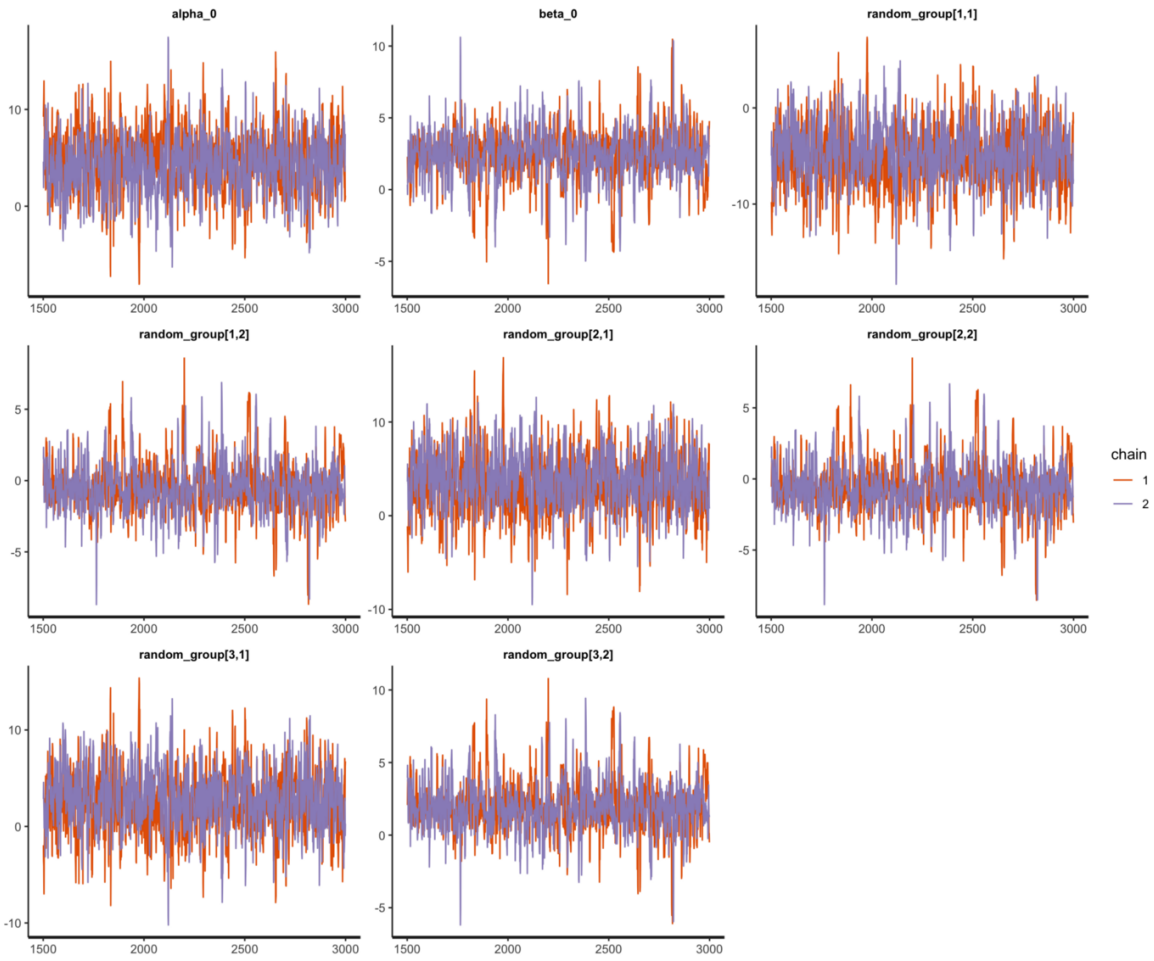

Figure 1: traceplot of posterior parameter estimates

4 Useful references and links

- Nicenboim, Schad and Vasishth (2021) An Introduction to Bayesian Data Analysis for Cognitive Science. Chapter 5 Online.
- Sorensen, Tanner & Shravan Vasishth (2015) Bayesian Linear Mixed Models Using

Stan: A Tutorial for Psychologists, Linguists, and Cognitive Scientists. arXiv Preprint arXiv:1506.06201.

- https://stats.stackexchange.com/questions/488188/what-are-the-steps-to-simulate-data-for-a-linear-model-with-random-slopes-and-ra
- https://yingqijing.medium.com/lkj-correlation-distribution-in-stan-29927b69e9be
- https://yingqijing.medium.com/multivariate-normal-distribution-and-cholesky-decomposition-in-stan-d9244b9aa623
- https://mlisi.xyz/post/bayesian-multilevel-models-r-stan/#fn1