Estimation of Models—Applied

Theodore Dounias

11/5/2018

MCMC Estimation Processes for Multilevel Models

In statistical science a Markov Chain is a sequence of random variables whose value depends on the value of the exact previous random variable. In mathematical terms, this would be a sequence $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, ..., \theta^{(t)}$ where $\mathbb{P}(\Theta = \theta^{(t)}|\theta^{(n)}) = \mathbb{P}(\Theta = \theta^{(t)})$ for $n \in [1, t-2]$ but $\mathbb{P}(\Theta = \theta^{(t)}|\theta^{(n)})$ is dependent on $\theta^{(n)}$ for $n = t-1$. A Markov Chain Monte Carlo simulation uses Bayesian estimation to update each sequential estimate of θ, leading it to converge to the true value being estimated [@gelman_data_2006].

Multilevel models can be estimated using MCMC sampling. Indicatively, this appendix presents the construction and coding of two types of MCMC samplers based on the Gibbs algorithm and Metropolis-Hastings algorithm. The code and mathematical derivations are adapted to my models from Gelman and Hill (2006).

Gibbs Sampler for the County Models

The Gibbs algorithm works as follows:

1. Choose a number of parallel simulation runs (chains). This number should be relatively low. In this example it is set to 3.
2. For each chain do the following:
   (a) Initialize vector of parameters $\Theta^{(0)} = \{\theta_1^{(0)}\, \theta_2^{(0)}\, ..., \theta_n^{(0)}\}$
   (b) Choose a number of iterations. For each iteration update every parameter in vector $\Theta^{(n_{iteration})}$, based on the values of vector $\Theta^{(n_{iteration}-1)}$.
3. Evaluate convergence between the chains.

If convergence is poor, repeat for more iterations, or follow diagnostic procedures. These are not specified here, but Gelman and Hill provide a good overview [@gelman_data_2006; @gelman_bayesian_2003].

County Model 1 (Only Random County Effects)

A basic multilevel model with only group-level intercept mixed effects can be written as follows:

$$y_i \ \sim \ N(a_{j[i]}, \sigma_y^2),\ i \in [1, n] a_j \ \sim \ N(\mu_\alpha, \sigma_\alpha^2),\ j \in [1, J]$$

This specification is slightly different from that presented in Chapter 2. Here $\alpha_{j[i]}$ is the coefficient for the group j that individual i belongs to, σ_y, σ_α the variances of the individual and group level distributions respectively, and μ_α the mean of the group-level distribution. In the case of the most basic county-level model estimated in my thesis (County Model 1), $n = 704$ and $J = 64$. Using Maximum Likelihood Estimation, and given that:

$$\alpha_j|y, \mu_\alpha, \sigma_y, \sigma_\alpha \ \sim \ N(\hat{\alpha_j}, V_j) \tag{1}$$

we can obtain estimates:

$$\hat{\alpha_j} = \frac{\frac{n_{[j]}}{\sigma_y^2}\bar{y}_{[j]} + \frac{1}{\sigma^2\alpha}}{\frac{n_{[j]}}{\sigma_y^2} + \frac{1}{\sigma^2\alpha}}, \qquad V_j = \frac{1}{\frac{n_{[j]}}{\sigma_y^2} + \frac{1}{\sigma^2\alpha}}, \tag{2}$$

where $n_{[j]}$ is the number of observations for group j, and $\bar{y}_{[j]}$ is the mean response for group j. Using these estimates and the common MLE estimates for variance and mean in a normal distribution, it is possible to construct a Gibbs sampler for model coefficients and errors. Step 2(b) in the Gibbs sampler would then be:

1. Estimate $a_j,\ j \in [1, J]$ using equations (1), (2).
2. Estimate μ_α by drawing from $N(\frac{1}{J}\sum_1^J \alpha_j, \sigma_\alpha^2/J)$ using the previous values estimated in step 1.
3. Estimate σ_y^2 as $\frac{\frac{1}{n}\sum_1^n (y_i - \alpha_{j[i]})^2}{X^2_{n-1}}$ where X^2_{n-1} is a draw from a χ^2 distribution with $n-1$ degrees of freedom.
4. Estimate σ_α^2 as $\frac{\frac{1}{J}\sum_1^J (\alpha_j - \mu_\alpha)^2}{X^2_{J-1}}$ where X^2_{n-1} is a draw from a χ^2 distribution with $J-1$ degrees of freedom.

While each step here seems relatively intuitive, the derivations behind some of the details (like the chi-squared distribution) are complex MLE processes and beyond the scope of this thesis. The R code for this algorithm is as follows:

```
## Gibbs sampler in R
a.update <- function(){
  a.new <- rep (NA, J)
  for (j in 1:J){
    n.j <- sum (model_dt$county==cnt_vec[j])
    y.bar.j <- mean (model_dt$turnout[model_dt$county==cnt_vec[j]])
    a.hat.j <- ((n.j/sigma.y^2)*y.bar.j + (1/sigma.a^2)*mu.a)/
               (n.j/sigma.y^2 + 1/sigma.a^2)
    V.a.j <- 1/(n.j/sigma.y^2 + 1/sigma.a^2)
    a.new[j] <- rnorm (1, a.hat.j, sqrt(V.a.j))
  }
  return (a.new)
}
mu.a.update <- function(){
  mu.a.new <- rnorm (1, mean(a), sigma.a/sqrt(J))
  return (mu.a.new)
}
sigma.y.update <- function(){
  sigma.y.new <- sqrt(sum((model_dt$turnout-a[model_dt$county])^2)/rchisq(1,703))
  return (sigma.y.new)
}
sigma.a.update <- function(){
  sigma.a.new <- sqrt(sum((a-mu.a)^2)/rchisq(1,J-1))
  return (sigma.a.new)
}

J <- 64
n.chains <- 3
n.iter <- 1000
sims <- array (NA, c(n.iter, n.chains, J+3))
dimnames (sims) <- list (NULL, NULL, c (paste ("a[", 1:J, "]", sep=""), "mu.a",
   "sigma.y", "sigma.a"))
```

```
for (m in 1:n.chains){
  mu.a <- rnorm (1, mean(model_dt$turnout), sd(model_dt$turnout))
  sigma.y <- runif (1, 0, sd(model_dt$turnout))
  sigma.a <- runif (1, 0, sd(model_dt$turnout))
  for (t in 1:n.iter){
    a <- a.update ()
    mu.a <- mu.a.update ()
    sigma.y <- sigma.y.update ()
    sigma.a <- sigma.a.update ()
    sims[t,m,] <- c (a, mu.a, sigma.y, sigma.a)
  }
}
```

Table 1: Gibbs sampler results for County Model 1

Calculated from...	mu.a	sigma.y	sigma.a
Sampler	0.4684	0.2	0.04019
Model	0.469	0.199	0.039

As is obvious from Table, the Gibbs sampler produces values very similar to the ones given by an *R* call of Model 1.

County Model 2 (Random County Effects and County-Level Predictors)

With slight changes from the previous model the following is the mathematical expression for a mixed effects model with group-level predictors:

$$y_i \sim N(a_{j[i]}, \sigma_y^2),\ i \in [1,n] a_j \sim N(U_j\gamma, \sigma_\alpha^2),\ j \in [1,J],$$

where U_j is a vector of predictor values for group j, and γ a vector of group-level coefficients, with the rest of the parameters having the same designation as previously. Bear in mind that the second of the previous expressions can also be written as:

$$\alpha_j = U_j\gamma + \eta_j, \quad \eta_j \sim N(0, \sigma_\alpha^2) \tag{3}$$

Updating the estimates used previously, it is again possible to construct a Gibbs sampler for model coefficients and errors. Step 2(b) in the Gibbs sampler in this case is:

1. Estimate $a_j,\ j \in [1,J]$. Start by calculating $y_i^{temp} = y_i - U_{j[i]}\gamma$. Then calculate an estimate $\hat{\eta}_j$ and variance matrix V_j from equations (1), (2), by replacing $\hat{\alpha}_j$ with $\hat{\eta}_j$ and y with y^{temp}. Use $\eta_j \sim N(\hat{\eta}_j, V_j)$ to draw errors η_j and then use (3) to estimate α_j for $j \in [1,J]$.
2. Estimate γ by first regressing α by predictor matrix U to obtain $\hat{\gamma}$ and variance matrix V_γ. Then use distribution $\gamma_j \sim N(\hat{\gamma}_j, V_j)$ to obtain estimates for vector γ.
3. Estimate σ_y^2 as $\frac{\frac{1}{n}\sum_1^n (y_i - \alpha_{j[i]})^2}{X_{n-1}^2}$ where X_{n-1}^2 is a draw from a χ^2 distribution with $n-1$ degrees of freedom.
4. Estimate σ_α^2 as $\frac{\frac{1}{J}\sum_1^J (\alpha_j - U_j\gamma)^2}{X_{J-1}^2}$ where X_{n-1}^2 is a draw from a χ^2 distribution with $J-1$ degrees of freedom.

County Model 2, as presented in Chapter 4, includes two county-level predictors: percentage of white residents and percentage of urban population; this means that $U = \{x^{\%white}, x^{\%urban}\}$. Keeping this in mind the following code estimates the coefficients and standard errors for Model 2:

```
## Gibbs sampler for a multilevel model with county predictors

a.update <- function(){
  y.temp <- y - U[county]%*%g
  eta.new <- rep (NA, J)
  for (j in 1:J){
    n.j <- sum (county==j)
    y.bar.j <- mean (y.temp[county==j])
    eta.hat.j <- ((n.j/sigma.y^2)*y.bar.j/
                  (n.j/sigma.y^2 + 1/sigma.a^2))
    V.eta.j <- 1/(n.j/sigma.y^2 + 1/sigma.a^2)
    eta.new[j] <- rnorm (1, eta.hat.j, sqrt(V.eta.j))
  }
  a.new <- U%*%g + eta.new
  return (a.new)
}

g.update <- function(){
  lm.0 <- lm (a ~ U)
  g.new <- sim (lm.0, n.sims=1)
  return (g.new)
}
sigma.y.update <- function(){
  sigma.y.new <- sqrt(sum((y-a[county]-X%*%b)^2)/rchisq(1,n-1))
  return (sigma.y.new)
}
sigma.a.update <- function(){
  sigma.a.new <- sqrt(sum((a-U%*%g)^2)/rchisq(1,J-1))
  return (sigma.a.new)
}
```