Markov Chain Monte Carlo methods

Víctor Sancibrián[1]

CEMFI

February 20, 2023

[1]These notes build on previous versions by Martín Almuzara and Micole De Vera.

Bayesian inference

Let $\theta \in \Theta \subset \mathbb{R}^k$ with posterior distribution

$$p(\theta|y) = \frac{\boldsymbol{f(y|\theta)p(\theta)}}{\int_\Theta f(y|\theta)p(\theta)d\theta}.$$

Bayesian inference on θ requires the computation of features of $p(\theta|y)$ taking the general form

$$\mathrm{E}[h(\theta)|y] = \int_\Theta h(\theta)p(\theta|y)d\theta$$

for a given function h, e.g. $h(\theta) = \theta$ to compute the posterior mean.

- Often computing $p(\theta|y)$ is intractable and we can only evaluate $\boldsymbol{f(y|\theta)p(\theta)}$

Even though we cannot integrate $p(\theta|y)$, can we *sample* from it?

- If so, a sufficiently large sample

$$\theta^{(1)}, \ldots, \theta^{(M)} \overset{a}{\sim} p(\theta|y)$$

  would provide estimates

$$\frac{1}{M} \sum_{m=1}^{M} h\left(\theta^{(m)}\right) \approx \mathrm{E}[h(\theta)|y]$$

- MCMC methods produce $\theta^{(1)}, \ldots, \theta^{(M)}$ as a *chain* where $\theta^{(m)}$ only depends on $\theta^{(m-1)}$ (Markov property)
  - This is an ergodic Markov chain with stationary distribution $p(\theta|y)$
  - The ergodic theorem ensures

$$\frac{1}{M} \sum_{m=1}^{M} h\left(\theta^{(m)}\right) \xrightarrow{p} \mathrm{E}[h(\theta)|y], \quad \text{as } M \to \infty$$

  - Only requires the ability to compute $\boldsymbol{f(y|\theta)p(\theta)}$

Outline

Intro to MCMC methods

Markov chains (I)

A Markov chain with continuous state space Θ is a stochastic process $\{\theta^{(t)}\}_{t \geq 0}$ that is fully characterized by

1. Initial condition density function p_0
2. Markov property

$$\mathrm{P}\left[\theta^{(t)} \in B \middle| \boldsymbol{\theta^{(t-1)}, \ldots, \theta^{(0)}}\right] = \mathrm{P}\left[\theta^{(t)} \in B \middle| \boldsymbol{\theta^{(t-1)}}\right]$$

3. Transition kernel κ

$$\mathrm{P}\left[\theta^{(t)} \in B \middle| \theta^{(t-1)}\right] = \int_B \kappa\left(\theta | \theta^{(t-1)}\right) d\theta$$

Markov chains (II)

Important properties Markov chains *might* satisfy are

- Existence of a unique *stationary* distribution p such that

$$p(\theta) = \int_{\Theta} \kappa\left(\theta|\theta'\right) p\left(\theta'\right) d\theta' \quad \Longleftrightarrow \quad \theta \sim p \quad \text{if} \quad \theta' \sim p$$

- Convergence to the stationary distribution for every density p_0 (i.e. ergodicity)

$$\lim_{t\to\infty} ||p_t - p||_{TV} = 0$$

- Approximation by averages via the ergodic theorem

$$\frac{1}{T}\sum_{t=1}^{T} h\left(\theta^{(t)}\right) \xrightarrow{p} \mathrm{E}[h(\theta)]\,, \quad \text{as } T \to \infty$$

MCMC methods

An MCMC method for $p(\cdot|y)$ is any method producing an ergodic Markov chain $\{\theta^{(t)}\}_{t\geq 0}$ whose **stationary distribution is $p(\cdot|y)$**.

- These methods differ in the way they construct transition kernels $\kappa(\cdot|\theta)$ with stationary distribution $p(\cdot|y)$
  1. Metropolis-Hastings algorithm
  2. Gibbs sampling
- Given an initial value $\theta^{(0)}$, a Markov chain $\{\theta^{(t)}\}_{t\geq 0}$ is then generated via κ
  - Ergodicity ensures that the starting value is–in principle–unimportant

The Metropolis-Hastings algorithm

The MH algorithm

- Requirements
  - $p(\theta|y) \propto \boldsymbol{f(y|\theta)p(\theta)}$
  - A chosen *proposal distribution* $\boldsymbol{q(\cdot|\theta)}$ to draw *candidates* θ^*
- Algorithm: given M and an initial value $\theta^{(0)}$, for $m = 1, \ldots, M$
  1. Draw $\theta^* \sim \boldsymbol{q}\left(\cdot|\boldsymbol{\theta}^{(m-1)}\right)$
  2. Set

$$\theta^{(m)} = \begin{cases} \theta^*, & \text{w.p. } \rho\left(\theta^*, \theta^{(m-1)}\right), \\ \theta^{(m-1)}, & \text{w.p. } 1 - \rho\left(\theta^*, \theta^{(m-1)}\right), \end{cases}$$

where

$$\rho\left(\theta^*, \theta^{(m-1)}\right) = \min\left\{ \frac{\boldsymbol{f}\left(\boldsymbol{y}|\boldsymbol{\theta}^*\right)\boldsymbol{p}\left(\boldsymbol{\theta}^*\right)\boldsymbol{q}\left(\boldsymbol{\theta}^{(m-1)}|\boldsymbol{\theta}^*\right)}{\boldsymbol{f}\left(\boldsymbol{y}|\boldsymbol{\theta}^{(m-1)}\right)\boldsymbol{p}\left(\boldsymbol{\theta}^{(m-1)}\right)\boldsymbol{q}\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(m-1)}\right)}, 1 \right\}$$

Implementation details

- Choosing $q(\cdot|\theta)$
  - Intuitively, need it to draw candidates in the high density regions of Θ often enough
  - A popular choice is a (symmetric) random walk proposal

$$\theta^* = \theta^{(m-1)} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \Sigma)$$

  - In practice, one tries several proposal distributions and monitors rejection rates

- Choosing $\theta^{(0)}$
  - Matters for approximations since in practice M is finite
  - *Burn-in* (discarding the first M_0 draws) is a popular choice

- *Thinning* (only retaining every d-th iteration to reduce autocorrelation) is useful when memory-constrained

Example: linear regression with known variance

- Likelihood model

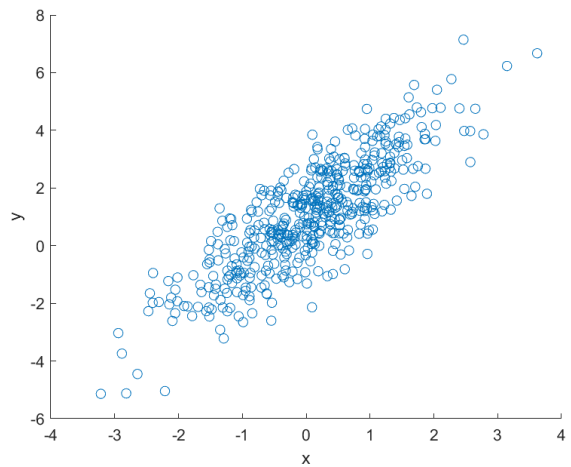
$$y_i | x_i; \beta_0, \beta_1 \sim \mathcal{N}\left(\beta_0 + \beta_1 x_i, 1\right)$$

- Prior

$$\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix}\right)$$

- Proposal density

$$\begin{pmatrix} \beta_0^* \\ \beta_1^* \end{pmatrix} = \begin{pmatrix} \beta_0^{(m-1)} \\ \beta_1^{(m-1)} \end{pmatrix} + \varepsilon, \quad \varepsilon \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}\right)$$



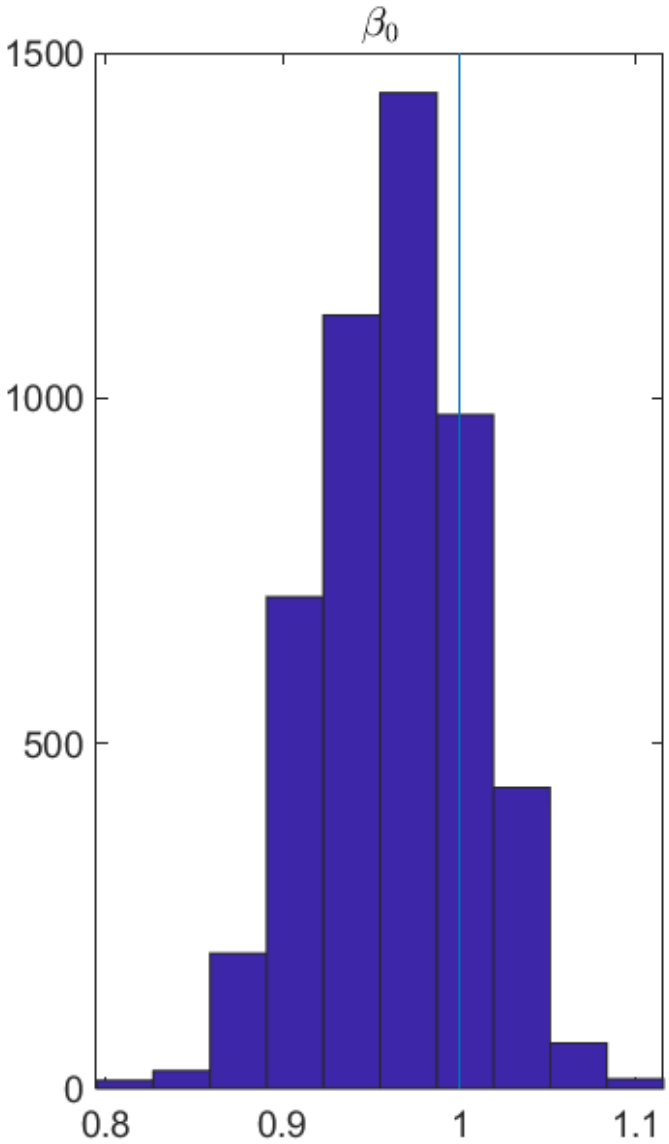
Example: code (I)

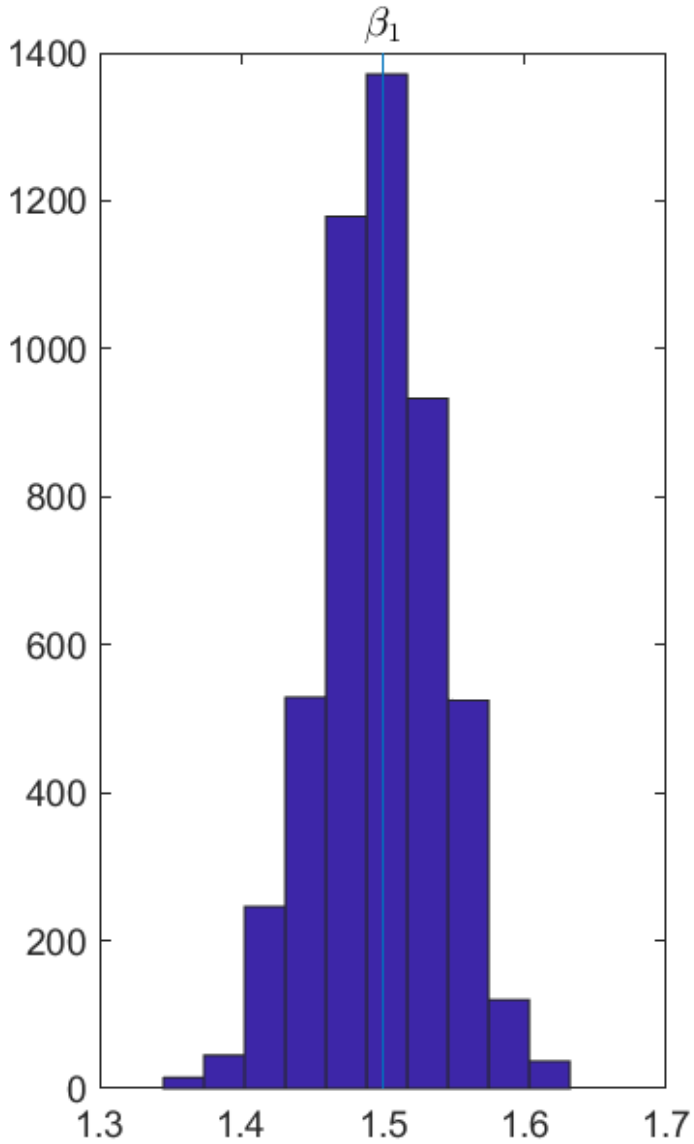
```
function val = llikelihood(y, x, params)
    b0 = params(1);
    b1 = params(2);

    % Get predictions
    pred = b0 + b1 * x;
    indiv_like = normpdf(y, pred, 1);
    indiv_ll = log(indiv_like);
    val = sum(indiv_ll);
end

function val = lprior(params)
    b0 = params(1);
    b1 = params(2);

    % Prior on b0;
    b0_prior = normpdf(b0, 1, 10);
    b1_prior = normpdf(b1, 1, 5);

    % Prior
    val = log(b0_prior) + log(b1_prior);
end

function val = unnorm_lpost(y, x, params)
    val = llikelihood(y, x, params) + lprior(params);
end
```

Example: code (II)

```
% MH Parameters
burn = 5000;
M = 5000;

chain = NaN(2, burn + M);
chain(:, 1) = [1; 1];
accept = NaN(1, burn + M);

for m = 2:(burn + M)
    % Proposal
    proposal = chain(:, m - 1) + mvnrnd([0; 0], [0.01, 0;
                                                 0, 0.01])';

    % Acceptance probability
    rho = exp(unnorm_lpost(y, x, proposal) - unnorm_lpost(y, x, chain(:, m - 1)));
    rho = min(1, rho);

    % Update
    if rand(1) <= rho
        chain(:, m) = proposal;
        accept(m) = 1;
    else
        chain(:, m) = chain(:, m - 1);
        accept(m) = 0;
    end
end

% Acceptance ratio
mean(accept(:, burn+1:end))
```

Example: posteriors





Gibbs sampling

Gibbs sampling

- Requirements
  - Access to *conditional* posterior distributions
    - Partition θ into (θ_1, θ_2)
    - Denote conditional posteriors as $p(\theta_1|\theta_2; y)$ and $p(\theta_2|\theta_1; y)$
- Algorithm: given M and an initial value $\theta_1^{(0)}$, for $m = 1, \ldots, M$
  1. Draw $\theta_2^{(m)} \sim p(\cdot|\theta_1^{(m-1)}; y)$
  2. Draw $\theta_1^{(m)} \sim p(\cdot|\theta_2^{(m)}; y)$
- Generalizes to a partition $\theta = (\theta_1, \ldots, \theta_d)$

Example: linear regression with independent priors

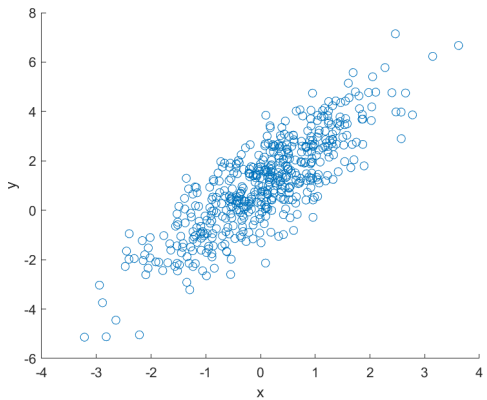
- Likelihood model $\beta = (\beta_0, \beta_1)'$

$$y_i | X_i; \beta \sim \mathcal{N}(X_i \beta, \sigma^2)$$

- Priors

$$\beta \sim \mathcal{N}(\beta_0, \Sigma_0)$$
$$\sigma^2 \sim \text{Inv-Gamma}(a_0, b_0)$$

where $a_0 > 0$ and $b_0 > 0$ are the shape and scale parameters



- Conditional posteriors

$$\beta|\sigma^2; y \sim \mathcal{N}(\beta_1, \Sigma_1)$$
$$\sigma^2|\beta; y \sim \text{Inv-Gamma}(a_1, b_1)$$

with

$$\Sigma_1 = \left(\Sigma_0^{-1} + \frac{1}{\sigma^2} X'X\right)^{-1}$$
$$\beta_1 = \Sigma_1 \left(\Sigma_0^{-1}\beta_0 + \left(\frac{1}{\sigma^2} X'X\right)\hat{\beta}\right)$$
$$\hat{\beta} = (X'X)\, X'y$$
$$a_1 = \frac{N}{2} + a_0$$
$$b_1 = \left(\frac{1}{b_0} + \frac{1}{2}(y - X\beta)'(y - X\beta)\right)^{-1}$$

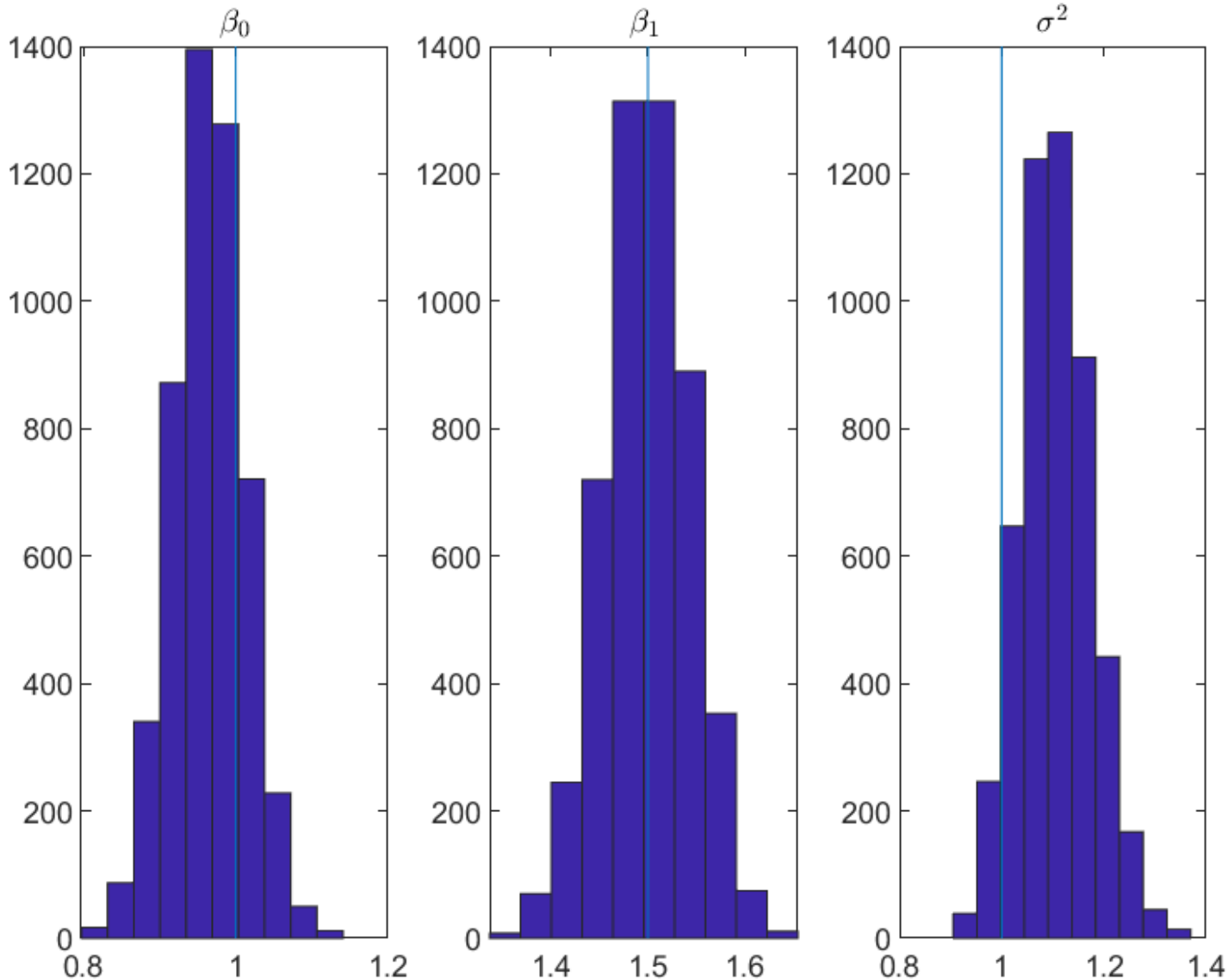
Example: code (I)

```
% Prior hyperparameters
beta0 = [1; 1];
Sigma0 = [2, 0; 0, 2];
a0 = 1;
b0 = 1;

% OLS coefficient
beta_ols = (X' * X) \ X' * y;
```

Example: code (II)

```
% Gibbs Sampler
burn = 5000;
M = 5000;
chain = NaN(3, burn + M);

chain(1:2, 1) = beta_ols;

for m = 2:(burn + M)
    % Draw sigma_sq conditional on beta
    a1 = (N / 2) + a0;
    b1 = (1 / b0) + 0.5 * (y - X * chain(1:2, m - 1))' * (y - X * chain(1:2, m - 1));
    b1 = 1 / b1;
    chain(3, m) = 1 / gamrnd(a1, b1);

    % Draw beta conditional on sigma_sq
    Sigma1 = pinv(pinv(Sigma0) + X' * X / chain(3, m));
    beta1 = Sigma1 * (pinv(Sigma0) * beta0 + X' * X * beta_ols / chain(3, m));
    chain(1:2, m) = mvnrnd(beta1, Sigma1)';
end
```

Example: posteriors



Discussion

Discussion

- Assessing convergence
  - Some recommendations
    - Monitor rejection rates (if applicable)
    - Check sensitivity to starting values and tune-in parameters
    - Plot!
  - Diagnostics can only reliably be used to determine *lack of convergence*
- Improper posteriors
- Further topics
  - Other algorithms: slice sampling, Metropolis-within-Gibbs...
  - Geweke (2004)