Computational Statistics II

Unit A.1: Metropolis-Hastings and Gibbs sampling

Tommaso Rigon

University of Milano-Bicocca

Ph.D. in Economics, Statistics and Data Science

Unit A.1

Main concepts

- Markov Chain Monte Carlo (MCMC)
- The Metropolis–Hastings algorithm
- The Gibbs sampling algorithm
- Writing clean and efficient **R** code

- Associated **R** code is available on the website of the course

Main references

- Robert, C. P., and Casella, G. (2004). Monte Carlo Statistical Methods. Springer.
- Roberts, G. O., and Rosenthal, J. S. (2004). General state space Markov chains and MCMC algorithms. Probability Surveys, 1(1), 20–71.
- Tierney, L. (1994). Markov chains for exploring posterior distributions. Annals of Statistics, 22(4), 1701-176.

Bayesian computations

- Over the past 30 years, Markov Chain Monte Carlo methods (MCMC) methods have **revolutionized** Bayesian statistics.
- Bayesian computational statistics is nowadays a lively and mature research field, compared to the early days. Still, there are several open questions.
- The ISBA bullettin (2011). *What are the open problems in Bayesian statistics?*
- **Alan Gelfand** (ISBA bullettin, 2011): *"Arguably the biggest challenge is in computation. If MCMC is no longer viable for the problems people want to address, then what is the role of INLA, of variational methods, of ABC approaches?"*
- Link: `https://www.stat.berkeley.edu/~aldous/157/Papers/Bayesian_open_problems.pdf`

Bayesian inference (recap)

- Let $\boldsymbol{X}$ be the data, following some distribution $\pi(\boldsymbol{X} \mid \boldsymbol{\theta})$, i.e. the **likelihood**, with $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$ being an unknown set of parameters.
- Let $\pi(\boldsymbol{\theta})$ be the **prior distribution** associated to $\boldsymbol{\theta}$.
- In Bayesian analysis, inference is based on the **posterior distribution** for $\boldsymbol{\theta}$, defined as

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{X}) = \frac{\pi(\boldsymbol{\theta})\pi(\boldsymbol{X} \mid \boldsymbol{\theta})}{\int_\Theta \pi(\boldsymbol{\theta})\pi(\boldsymbol{X} \mid \boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}.$$

- **Key issue**: the **normalizing constant**, i.e. the above integral, is often **intractable** $\implies$ no analytical solutions, beyond conjugate cases.
- Numerical approximations of $\int_\Theta \pi(\boldsymbol{\theta})\pi(\boldsymbol{X} \mid \boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$ are highly unstable, especially in high dimensions $\implies$ the `integrate` **R** function will not work in most cases.

Bayesian inference using sampling

- **Key solution**. It is sometimes possible to **sample** from the posterior distribution even without knowing the normalizing constant.
- If we can get **random samples** $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}$ from the posterior distribution, then we can approximate any functional of interest, i.e.

$$\mathbb{E}(g(\boldsymbol{\theta}) \mid \boldsymbol{X}) \approx \frac{1}{R} \sum_{r=1}^{R} g(\boldsymbol{\theta}^{(r)}).$$

- If $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}$ were **independent** samples from the posterior distribution, this approximation would be called **Monte Carlo integration**.
- Monte Carlo integration is justified by the **law of large numbers**.
- In this course, we will consider samples $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(R)}$ that are **dependent** and follow a Markov Chain $\implies$ Markov Chain Monte Carlo (MCMC).

A short introduction to Markov chains

Markov chains

- A sequence $\boldsymbol{Y}^{(0)}, \boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(R)}$ of random elements is a **Markov chain** if

$$\mathbb{P}(\boldsymbol{Y}^{(r+1)} \in A \mid \boldsymbol{y}^{(0)}, \ldots, \boldsymbol{y}^{(r)}) = \mathbb{P}(\boldsymbol{Y}^{(r+1)} \in A \mid \boldsymbol{y}^{(r)}).$$

- In other words, the conditional distribution of $\boldsymbol{Y}^{(r+1)}$ given $\boldsymbol{y}^{(0)}, \ldots, \boldsymbol{y}^{(r)}$ is the same as the conditional distribution of $\boldsymbol{Y}^{(r+1)}$ given $\boldsymbol{y}^{(r)}$, called **transition kernel**.

- Given an initial condition $\boldsymbol{y}^{(0)}$ a Markov chain is fully characterized by its transition kernel, which we assume does not depend on r (**homogeneity**).

- In continuous cases, the transition kernel is identified by a **conditional density**, denoted with

$$k(\boldsymbol{y}^{(r+1)} \mid \boldsymbol{y}^{(r)}).$$

- When the sample space is finite, the transition kernel is a **transition matrix**, say P.

Example: autoregressive process $\mathrm{AR}(1)$

- **Autoregressive** processes provide a simple illustration of **Markov chains** on continuous state-space.
- Let $Y^{(0)} \sim N(30, 1)$ and let us define

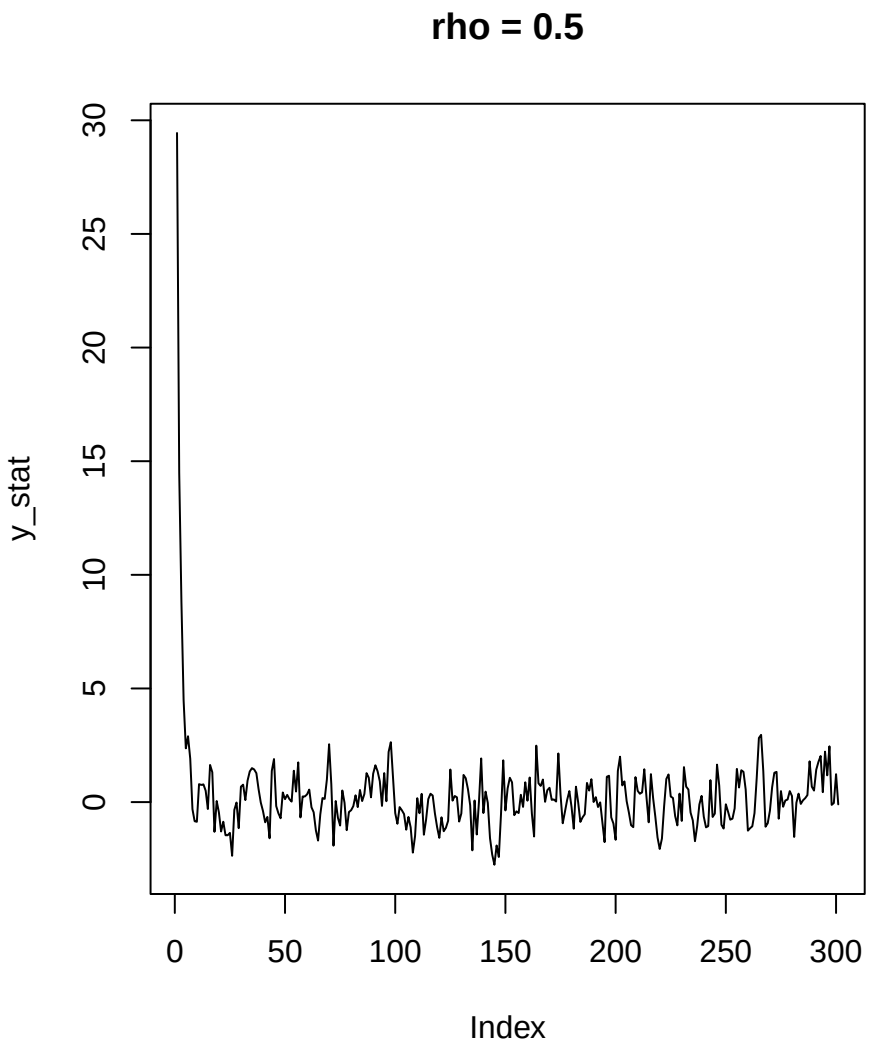
$$Y^{(r)} = \rho Y^{(r-1)} + \epsilon^{(r)}, \qquad \rho \in \mathbb{R},$$

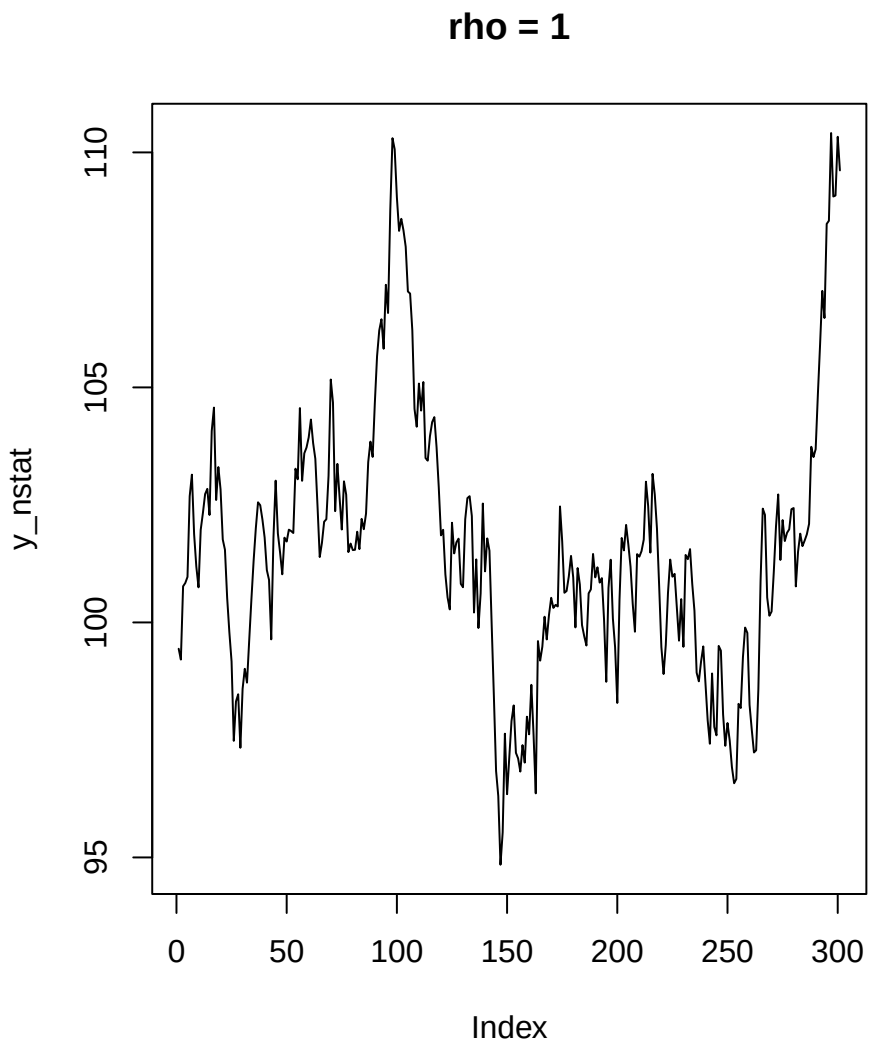
  with the error terms $\epsilon^{(r)}$ being iid according to a standard Gaussian $\mathrm{N}(0, 1)$.
- Then the sequence of $Y^{(r)}$ forms indeed a Markov chain and the **transition density** is such that

$$(y^{(r)} \mid y^{(r-1)}) \sim \mathrm{N}(\rho y^{(r-1)}, 1).$$

- If the parameter $|\rho| < 1$ then the Markov chain has a more "**stable**" behaviour.

Example: autoregressive processes $\mathrm{AR}(1)$





Invariant distribution

- An increased level of stability of a Markov chain occurs when the latter admits an **invariant** or **stationary** probability distribution.
- A probability density $\pi(\boldsymbol{y})$ is invariant for a Markov chain with kernel k if

$$\pi(\boldsymbol{y}^*) = \int k(\boldsymbol{y}^* \mid \boldsymbol{y})\pi(\boldsymbol{y})\mathrm{d}\boldsymbol{y}.$$

- This is to say that the **marginal** distributions of $\boldsymbol{Y}^{(r)}$ and $\boldsymbol{Y}^{(r+1)}$ are the same and are equal to $\pi(\boldsymbol{y})$, although $\boldsymbol{Y}^{(r)}$ and $\boldsymbol{Y}^{(r+1)}$ remain **dependent**.
- Roughly speaking, if a Markov chain admits a stationary distribution + some technical conditions, then for R large enough the chain "stabilizes" around the invariant law.
- In the previous $\mathrm{AR}(1)$ example the stationary distribution is $\mathrm{N}(0, 1/(1-\rho^2))$.

Invariant distribution

- Not every Markov chain admits a stationary law. However, the Markov chains constructed for sampling purposes should always converge to an invariant distribution.
- Indeed, in Markov Chain Monte Carlo the stationary distribution $\pi(\boldsymbol{y})$ represents the **target density** from which we wish to simulate.
- Then, we will make use of the following approximation

$$\int g(\boldsymbol{y})\pi(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \approx \frac{1}{R}\sum_{r=1}^{R} g(\boldsymbol{y}^{(r)}),$$

  where $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(R)}$ are generated according to a Markov chain, with $\boldsymbol{y}^{(0)} \sim \pi(\boldsymbol{y})$.
- How to construct a Markov chain that converges to the desired density $\pi(\boldsymbol{y})$?
- Before delving into this key problem, let us briefly review the assumptions under which this approximation is a reasonable one.

Regularity conditions

- We will consider Markov chains that are **irreducible**, **aperiodic**, and **Harris recurrent**.
- A rigorous presentation of these properties is beyond the aims of this course, so we offer here only a brief description in the **discrete case** to help the intuition.
- For a more detailed treatment, see Chapter 6 of Robert and Casella (2004).
- **Irreducibility**. The chain is irreducible if it does not "get stuck" in a local region of the sample space. In the discrete case the chain is irreducible if all states are connected.
- **Aperiodicity**. The chain is aperiodic if it does not has any deterministic cycle.
- **Harris recurrent**. The chain is (Harris) recurrent if it visits any region of the sample space "sufficiently often".

Irreducibility

- The aforementioned properties are easy to formalize in the **discrete** setting, namely when the values of the Markov chain are $Y^{(r)} \in \{1, 2, \dots\}$.
- The **first passage time** is the first r for which the chain is equal to j, namely:

$$\tau_j = \inf\{r \geq 1 : Y^{(r)} = j\},$$

  where by convention we let $\tau_j = \infty$ if $Y^{(r)} \neq j$ for every $r \geq 1$.
- Moreover, let us denote the **probability of return** to j in a finite number of step, starting from j'

$$\mathbb{P}(\tau_j < \infty \mid y^{(0)} = j').$$

- Hence, the chain is **irreducible** if $\mathbb{P}(\tau_j < \infty \mid y^{(0)} = j') > 0$ for all $j, j' \in \mathbb{N}$.

Aperiodicity

- Consider the two-state chain with **transition matrix**

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

- The two-step ahead transition matrix is $P^2 = I$, so $P^{2r} = I$ and $P^{2r+1} = P$ for all $r \geq 1$.
- Hence, due to periodicity this chain is **failing to converge** anywhere.
- In the discrete case, we call a state j **aperiodic** if the set

$$\{r \geq 1 : [P^r]_{jj} > 0\}$$

has no common divisor other than 1.

- A chain is **aperiodic** if all its states are aperiodic.

Harris recurrence

- Informally, a state j of an **irreducible** Markov chain is recurrent when it is visited by the chain "infinitely often".
- More formally, in the discrete setting a state $j \in \mathbb{N}$ is **recurrent** if and only if

$$\mathbb{P}(\tau_j < \infty \mid y^{(0)} = j) = \mathbb{P}(Y^{(r)} = j \text{ for infinitely many } r \mid y^{(0)} = j) = 1.$$

- The above a definition, with the necessary adjustments, is actually a **sufficient condition** for recurrence in the continuous case.
- Indeed, in the continuous case recurrence is defined in terms of the **average number of passages** on a Borel set, which must be divergent.
- The stronger "**Harris**" recurrence condition is mostly needed to fix measure-theoretic pathologies.

Invariant measures

- A Markov chain that is aperiodic and Harris recurrent displays a quite stable behaviour, so one may wonder if it admits an **invariant** distribution.
- In general, the answer is no: the Gaussian random walk is an example.
- Indeed, we call **Harris positive** a Markov chain which is Harris recurrent and admits an invariant probability distribution.
- In the discrete case, this occurs if and only if $\mathbb{E}(\tau_j \mid y^{(0)} = j) < \infty$.
- However, something can be said about the existence of invariant measures in general.

Theorem

If $(\boldsymbol{Y}^{(r)})_{r \geq 1}$ is a recurrent chain, there exists an invariant σ-finite measure which is unique up to a multiplicative factor.

- Unfortunately, such an invariant measure is not necessarily a probability measure!

Reversibility and detailed balance

- What follows is a popular **sufficient condition** to ensure a recurrent chain is also **positive recurrent**. That is, it admits an invariant probability distribution.
- Interestingly enough, such a condition has also a quite intuitive interpretation.
- We call a Markov chain $(\boldsymbol{Y}^{(r)})_{r\geq 1}$ **reversible** if the distribution of $\boldsymbol{Y}^{(r)}$ conditionally on $\boldsymbol{Y}^{(r+1)}$ is the same as the distribution of $\boldsymbol{Y}^{(r+1)}$ conditionally on $\boldsymbol{Y}^{(r)}$.
- A Markov chain $(\boldsymbol{Y}^{(r)})_{r\geq 1}$ with transition kernel k satisfies the **detailed balance** condition if there exists a function f such that

$$k(\boldsymbol{y} \mid \boldsymbol{y}^*) f(\boldsymbol{y}) = k(\boldsymbol{y}^* \mid \boldsymbol{y}) f(\boldsymbol{y}^*).$$

Theorem

If $(\boldsymbol{Y}^{(r)})_{r\geq 1}$ satisfies the detailed balance condition with π a probability density function, then π is the invariant (stationary) density and the chain is reversible.

Convergence to equilibrium

- From now on, we will always make use of the **aperiodicity** and **Harris positivity** properties, assuming the existence of a stationary probability density π.
- The following result establishes that a chain converges in **total variation** to its invariant measures as $r \to \infty$.
- Importantly, this occurs regardless the initial conditions $\boldsymbol{Y}^{(0)} \sim \pi_0$.

Theorem

Let the Markov chain $(\boldsymbol{Y}^{(r)})_{r \geq 1}$ be aperiodic and Harris positive, with $\boldsymbol{Y}_0 \sim \pi_0$. Moreover let π_r be the marginal probability density of $Y^{(r)}$. Then

$$\lim_{r \to \infty} |\pi_r(\boldsymbol{y}) - \pi(\boldsymbol{y})|_{\text{TV}} = 0.$$

Furthermore $|\pi_r(\boldsymbol{y}) - \pi(\boldsymbol{y})|_{\text{TV}}$ is decreasing in r.

Ergodic theorem

- The **Ergodic Theorem** is essentially the equivalent of the law of large numbers for Markov chains. It is the main justification for the usage of MCMC methods.
- What follows is a slightly simplified version of it, which is amenable for our purposes.
- Again, the following result holds irrespectively on the initial conditions $\boldsymbol{Y}^{(0)} \sim \pi_0$.

Theorem (Ergodic Theorem)

Let the Markov chain $(\boldsymbol{Y}^{(r)})_{r \geq 1}$ be Harris positive with stationary distribution π. Let the function g be integrable w.r.t. to π. Then

$$\frac{1}{R} \sum_{r=1}^{R} g(\boldsymbol{Y}^{(r)}) \longrightarrow \int g(\boldsymbol{y}) \pi(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}, \qquad R \to \infty,$$

almost surely.

Practical considerations I

- **Sampling** the path of a Markov chain is straightforward from the definition.
- We firstly simulate $\boldsymbol{Y}^{(0)} \sim \pi_0$. Then we simulate the subsequent values $(\boldsymbol{Y}^{(r+1)} \mid \boldsymbol{Y}^{(r)})$ according to the transition kernel k, assuming it is easy to do so.
- If a Markov chain has a **stationary distribution** π, then simulating from a Markov chain leads to a practical strategy for simulating from π as well.
- Because of the previous results, the distribution π_r of $\boldsymbol{Y}^{(r)}$ will eventually **converge** to the stationary law π we wish to simulate.
- Thus, $\boldsymbol{Y}^{(B)}$ for $B > 0$ large enough can be regarded as a sample from π. Moreover, the subsequent values can be also regarded as samples from π, the invariant distribution.

Practical considerations II

- The values $\boldsymbol{Y}^{(1)}, \boldsymbol{Y}^{(2)}, \ldots, \boldsymbol{Y}^{(B)}$ represent the so-called **burn-in** period, namely the values the chain needs to reach convergence.
- The burn-in values should be **discarded**. The choice of B is not always easy in practice.
- Hence, the approximations of functionals of interest are based on the values

$$\int g(\boldsymbol{y})\pi(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \approx \frac{1}{R-B}\sum_{r=B+1}^{R} g(\boldsymbol{y}^{(r)}),$$

  which once again we emphasize it relies on the **Ergodic Theorem**.
- What we are still missing are some practical Markov chains algorithms that indeed target a specific stationary distribution.

The Metropolis Hastings algorithm

Metropolis-Hastings algorithm I

- We finally introduce our first first Markov Chain Monte Carlo MCMC method: the **Metropolis-Hastings** algorithm (MH).
- This idea goes back to Metropolis et al. (1953) and Hastings (1970).
- Like the acceptance-rejection algorithm, the MH is based on proposing values sampled from an **instrumental proposal** distribution.
- The proposed values are then accepted with a certain probability that reflects how likely it is that they are from the target density $\pi(\boldsymbol{y})$.
- Under mild conditions, this ensures that the chain will converge to the target density $\pi(\boldsymbol{y})$, which is shown to be the stationary distribution.

Metropolis-Hastings algorithm II

- Set the first value of the chain $\boldsymbol{y}^{(0)}$ to some (reasonable) value.

At the rth value of the chain

- Let $\boldsymbol{y} = \boldsymbol{y}^{(r)}$ be the current status of the chain.
- Sample $\boldsymbol{y}^*$ from a **proposal distribution** $q(\boldsymbol{y}^* \mid \boldsymbol{y})$.
- Compute the **acceptance probability**, defined as

$$\alpha(\boldsymbol{y}^*, \boldsymbol{y}) = \min\left\{1, \frac{\pi(\boldsymbol{y}^*)}{\pi(\boldsymbol{y})}\frac{q(\boldsymbol{y} \mid \boldsymbol{y}^*)}{q(\boldsymbol{y}^* \mid \boldsymbol{y})}\right\} = \min\left\{1, \frac{\tilde{\pi}(\boldsymbol{y}^*)}{\tilde{\pi}(\boldsymbol{y})}\frac{q(\boldsymbol{y} \mid \boldsymbol{y}^*)}{q(\boldsymbol{y}^* \mid \boldsymbol{y})}\right\}.$$

- With probability $\alpha = \alpha(\boldsymbol{y}^*, \boldsymbol{y})$, **update the status** of the chain and set $\boldsymbol{y} \leftarrow \boldsymbol{y}^*$.

- **Key result**. We do not need to know the normalizing constant K of $\pi(\boldsymbol{y}) = K\tilde{\pi}(\boldsymbol{y})$, because it simplifies in the above ratio.

Detailed balance and reversibility of the MH

- The **transition kernel** of the MH algorithm is therefore the following "mixture"

$$k(\boldsymbol{y}^* \mid \boldsymbol{y}) = \alpha(\boldsymbol{y}^*, \boldsymbol{y}) q(\boldsymbol{y}^* \mid \boldsymbol{y}) + \delta_{\boldsymbol{y}}(\boldsymbol{y}^*) \int q(\boldsymbol{s} \mid \boldsymbol{y}) \{1 - \alpha(\boldsymbol{s} \mid \boldsymbol{y})\} \mathrm{d}\boldsymbol{s},$$

  where $\delta_{\boldsymbol{y}}(\boldsymbol{y}^*)$ is a point mass at $\boldsymbol{y}$.

- **Exercise I**. Using the definition of the acceptance probability, verify the following condition:

$$\pi(\boldsymbol{y}) \alpha(\boldsymbol{y}^*, \boldsymbol{y}) q(\boldsymbol{y}^* \mid \boldsymbol{y}) = \pi(\boldsymbol{y}^*) \alpha(\boldsymbol{y}, \boldsymbol{y}^*) q(\boldsymbol{y} \mid \boldsymbol{y}^*).$$

- **Exercise II**. From the above equations, conclude that

$$k(\boldsymbol{y} \mid \boldsymbol{y}^*) \pi(\boldsymbol{y}) = k(\boldsymbol{y}^* \mid \boldsymbol{y}) \pi(\boldsymbol{y}^*),$$

  corresponding to the **detailed balance** condition.

- Hence, $\pi(\boldsymbol{y})$ is the **stationary** law of a MH process and the chain is **reversible**.

Convergence properties

- The existence of an invariant stationary distribution is quite a strong theoretical result.
- However, one should also check for **irreducibility**, **aperiodicity** and **Harris recurrence** of the MH chain.
- This depends on the proposal distribution $q(\boldsymbol{y}^* \mid \boldsymbol{y})$ and the stationary density $\pi(\boldsymbol{y})$, although it is tipically true under very mild conditions.
- Quite general **sufficient conditions** for ergodicity are given in Chapter 7.3.2 of Robert and Casella (2004).
- Failure of MH algorithm typically occurs in presence of a disconnected support for $\pi(\boldsymbol{y})$ and / or if the proposal $q(\boldsymbol{y}^* \mid \boldsymbol{y})$ is not able to explore the support of $\pi(\boldsymbol{y})$.

An important caveat

170 6 Metropolis–Hastings Algorithms

We won't dabble any further into the theory of convergence of MCMC algorithms, relying instead on the guarantee that standard versions of these algorithms such as the Metropolis–Hastings algorithm or the Gibbs sampler are almost always theoretically convergent. Indeed, the real issue with MCMC algorithms is that, despite those convergence guarantees, the practical implementation of those principles may imply a very lengthy convergence time or, worse, may give an impression of convergence while missing some important aspects of f, as discussed in Chapter 8.

- This snapshot is taken from Chapter 6 of the textbook Robert, C. P., and Casella, G. (2009). *Introducing Monte Carlo methods with R*. Springer.
- In this notation f is the stationary distribution.

Example: Gaussian distribution

- Suppose we wish to simulate from a Gaussian distribution $\mathrm{N}(\mu, \sigma^2)$ using a MH algorithm, whose density is $\pi(y)$.
- This is obviously a **toy example**, because in practice one would just use `rnorm`.
- For the proposal distribution $q(y^* \mid y)$, we can use a **uniform random walk**, namely

$$y^* = y + u, \qquad u \sim \mathrm{Unif}(-\epsilon, \epsilon).$$

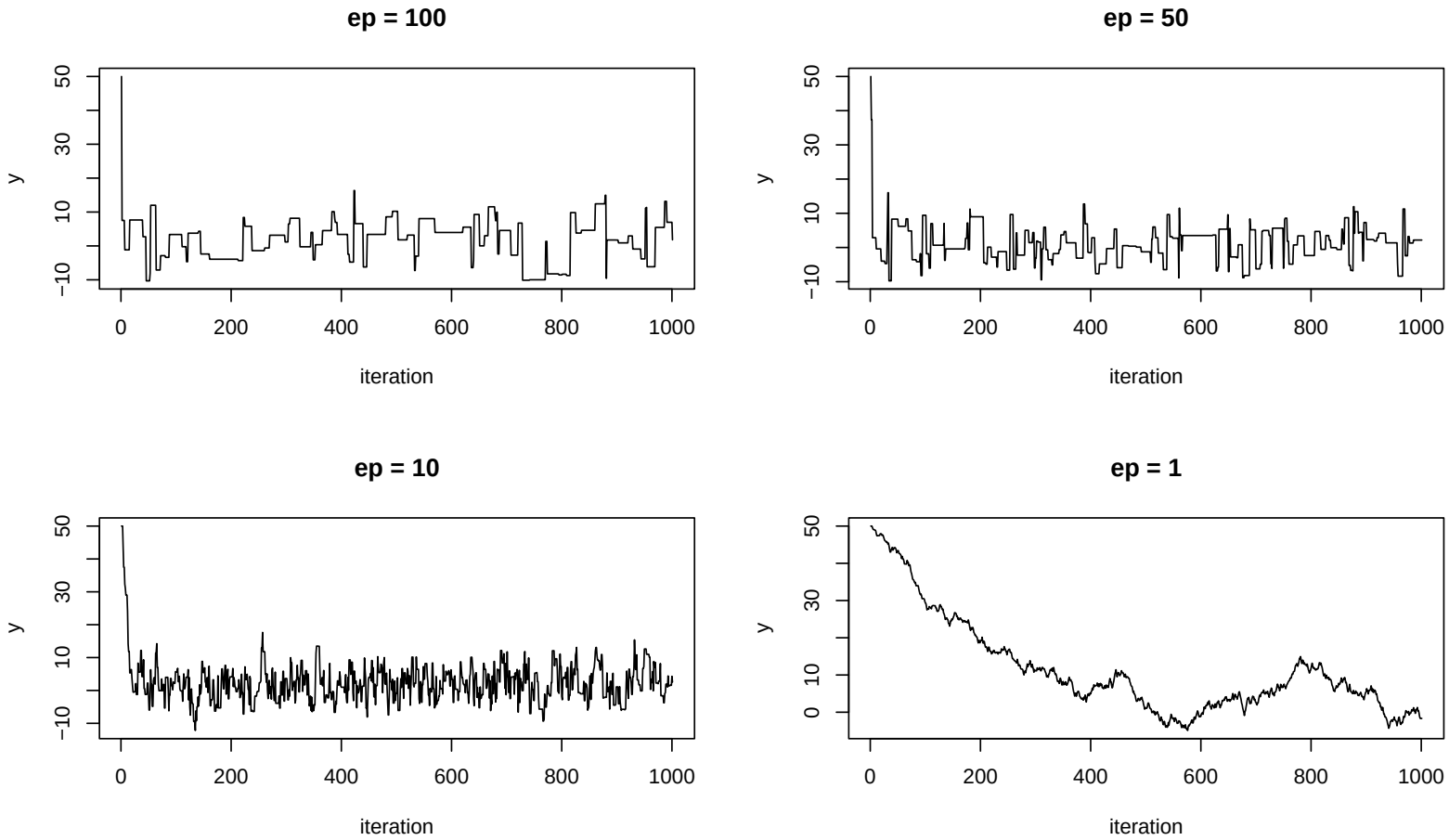
  The choice of $\epsilon > 0$ will have an impact on the algorithm, as we shall see.
- Random walks are **symmetric** proposals distributions, so $q(y^* \mid y) = q(y \mid y^*)$.
- This means the acceptance probability α is equal to

$$\alpha(y^*, y) = \min\left\{1, \frac{\pi(y^*)}{\pi(y)}\right\}.$$

Example: Gaussian distribution

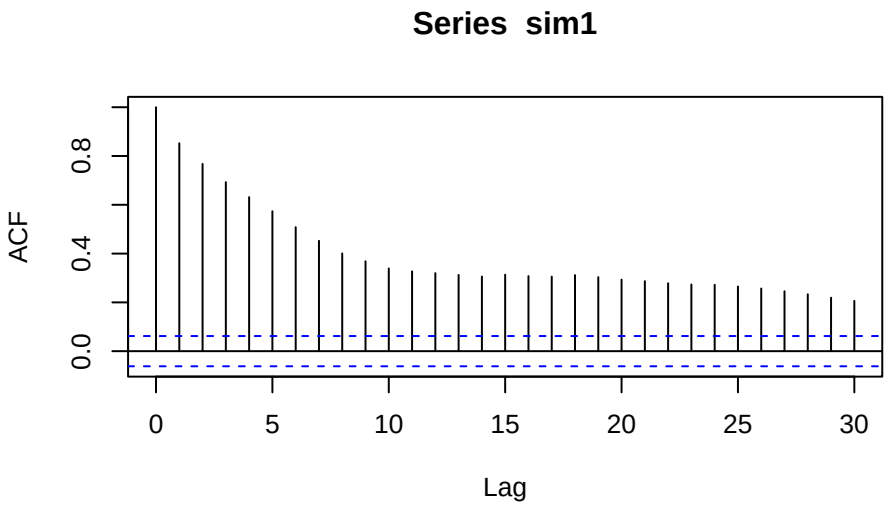
```r
norm_mcmc <- function(R, mu, sig, ep, x0) {
    # Initialization
    out <- numeric(R + 1)
    out[1] <- x0
    # Beginning of the chain
    x <- x0
    # Metropolis algorithm
    for(r in 1:R){
        # Proposed values
        xs     <- x + runif(1, -ep, ep)
        # Acceptance probability
        alpha  <- min(dnorm(xs, mu, sig) / dnorm(x, mu, sig), 1)
        # Acceptance / rejection step
        accept <- rbinom(1, size = 1, prob = alpha)
        if(accept == 1) {
            x <- xs
        }
        out[r + 1] <- x
    }
    out
}
```

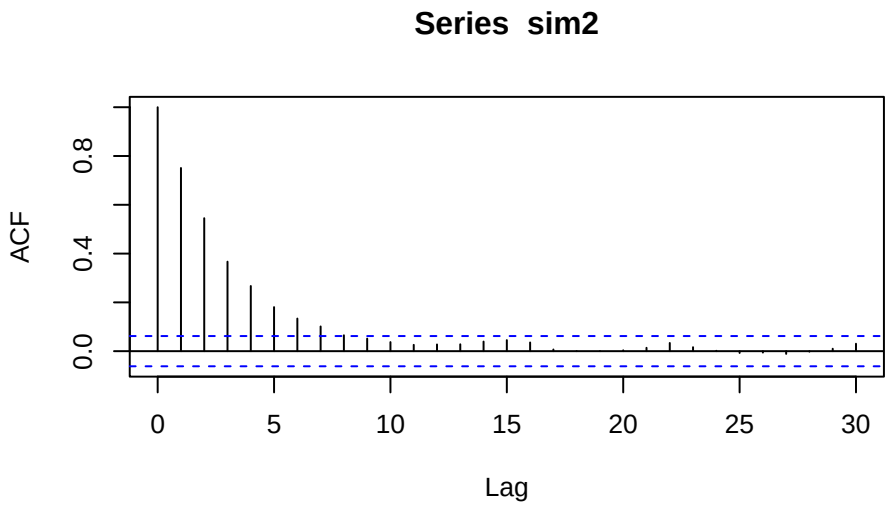
Example: Gaussian distribution

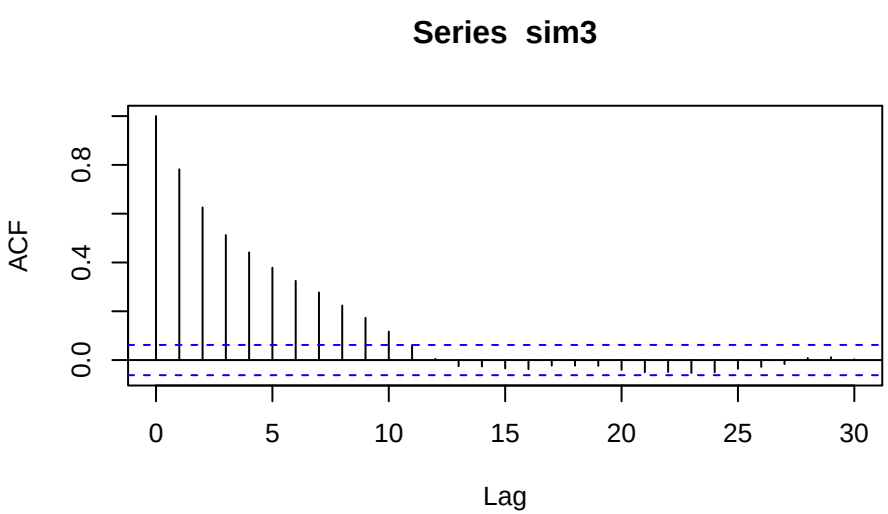


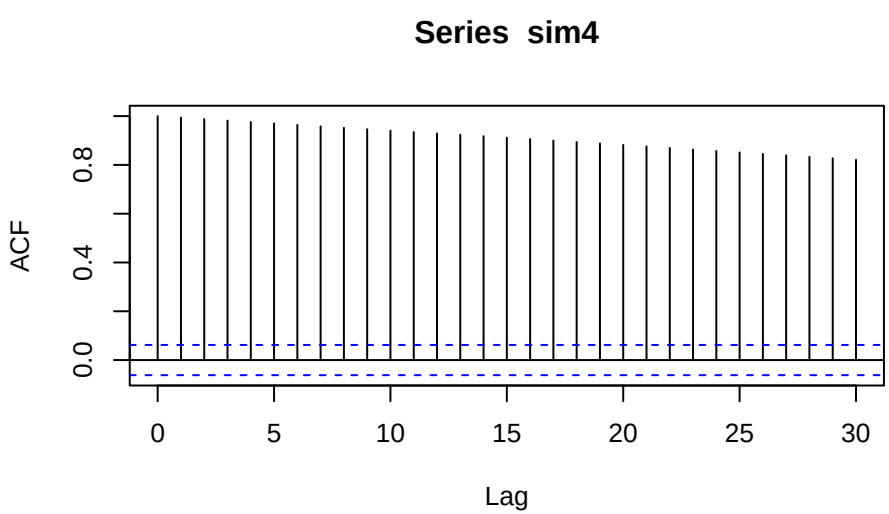
- MH algorithm targeting the stationary density $\mathrm{N}(2, 5^2)$ using the proposal distribution $y^* = y + u$, $u \sim \mathrm{Unif}(-\epsilon, \epsilon)$, with $\epsilon = 100, 50, 10, 1$ (ep).

Example: Gaussian distribution

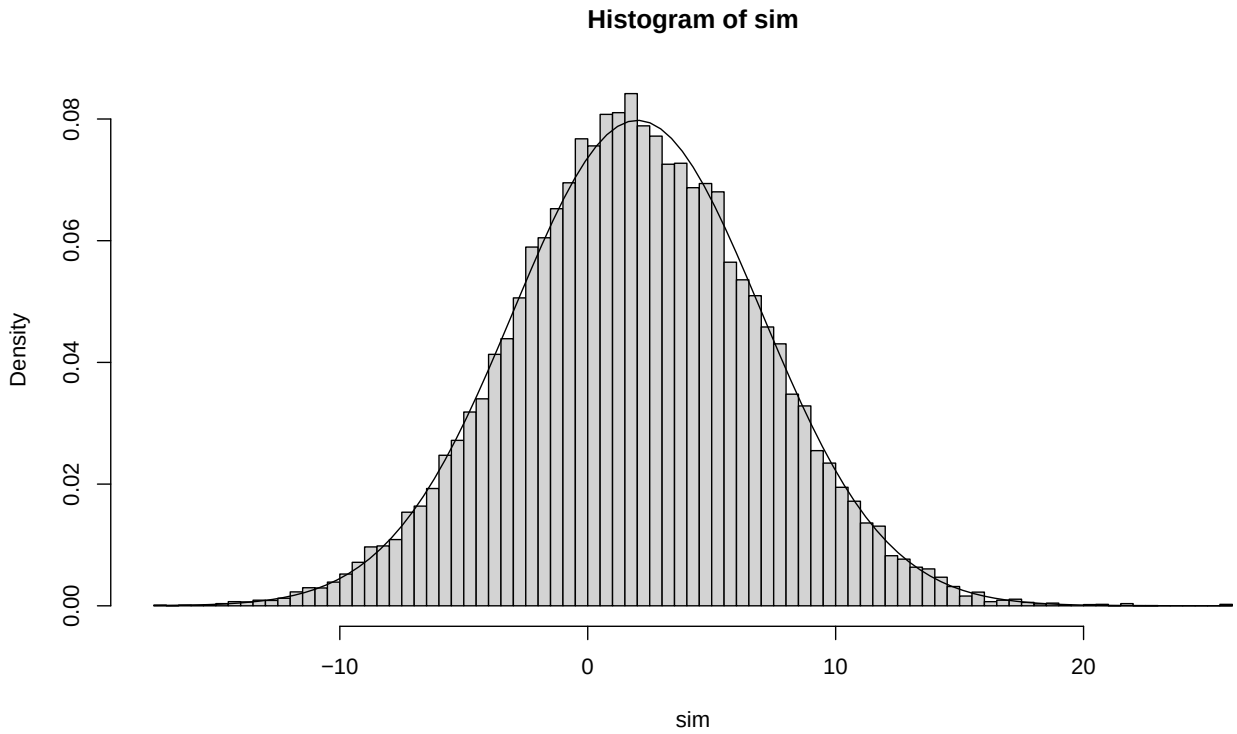








Example: Gaussian distribution




```r
# Simulate the MH chain
sim <- norm_mcmc(50000, mu = 2, sig = 5, ep = 10, x0 = 50)
# Identify a burn-in period
burn_in <- 1:200; sim <- sim[-c(burn_in)]
# Plot the results
hist(sim, breaks = 100, freq = FALSE)
curve(dnorm(x, 2, 5), add = T) # This is usually not known!
```

Hybrid Metropolis-Hastings

- The actual advantage of MCMC over classical sampling methods is actually evident in **high dimensions**. Recall that $\boldsymbol{Y}^{(r)} = (Y_1^{(r)}, \ldots, Y_p^{(r)})$.
- As option is to use the "vanilla" Metropolis-Hastings algorithm. However the proposal distribution is not easy to choose if $p > 2$. To this issue is devoted Unit B.1.
- An alternative is using a "**hybrid**" Metropolis-Hastings algorithm. This scheme is also known as **Metropolis-within-Gibbs**.
- The idea is quite simple: interatively apply a Metropolis-Hastings update to **each coordinate** $Y_j^{(r)}$, according to the proposal distributions $q_j(y_j^* \mid y_j)$.
- Sometimes it is convenient to update block of coordinates rather than univariate components.
- This algorithms is **ergodic** and has **stationary** distribution $\pi(\boldsymbol{y})$, under mild conditions. This should be taken for granted, see e.g. Chapter 10.3.3 of Robert and Casella (2004).

Example: bivariate Gaussian

- Suppose we aim at simulating from a bivariate Gaussian distribution, whose density is

$$\pi(y_1, y_2) = \frac{1}{2\pi\sqrt{(1-\rho^2)}} \exp\left\{-\frac{1}{2(1-\rho^2)}(y_1^2 - 2\rho y_1 y_2 + y_2^2)\right\}.$$

```
# Density of a bivariate Gaussian (up to a proportionality constant)
dbvnorm <- function(x, rho) {
    exp(-(x[1]^2 - 2 * rho * x[1] * x[2] + x[2]^2) / (2 * (1 - rho^2)))
}
```

- For the proposal distributions $q_j(y_j^* \mid y_j)$, we can again use a **uniform random walk**, namely

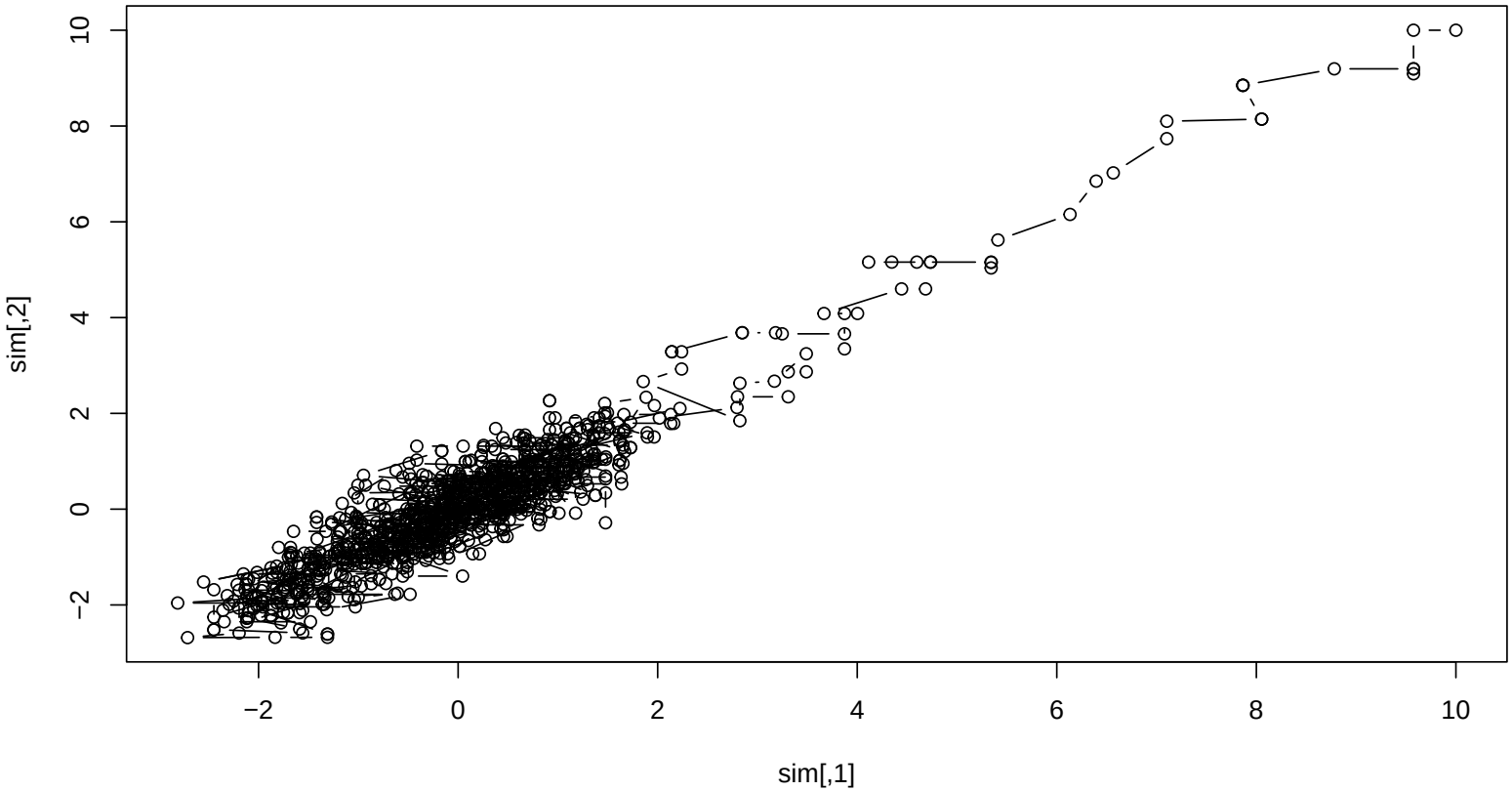
$$y_j^* = y_j + u_j, \qquad u \sim \text{Unif}(-\epsilon_j, \epsilon_j), \qquad j = 1, 2.$$

- As before, the choice of ϵ_j affects the performance of the MH.

Example: bivariate Gaussian

```r
# Hybrid Metropolis (Metropolis-within-Gibbs)
bvnorm_mcmc <- function(R, rho, ep, x0) {
    out <- matrix(0, R + 1, 2)
    out[1, ] <- x0
    x <- x0
    for(r in 1:R){
        for(j in 1:2){
            xs <- x
            xs[j] <- x[j] + runif(1, -ep[j], ep[j])
            alpha <- min(dbvnorm(xs, rho) / dbvnorm(x, rho), 1) # Acceptance probability
            accept <- rbinom(1, size = 1, prob = alpha) # Acceptance / rejection step
            if(accept == 1) {
                x[j] <- xs[j]
            }
        }
        out[r + 1, ] <- x
    }
    out
}
```

Example: bivariate Gaussian



- Hybrid MH algorithm targeting the stationary density of a bivariate normal with correlation $\rho = 0.8$, with starting point $(10, 10)$.

MCMC for Bayesian statistics

Metropolis-Hastings algorithm in Bayesian statistics

- The **Metropolis-Hastings** (MH) algorithm is especially useful for Bayesian inference. In the following, we just rephrase the MH using the Bayesian notation.
- Set the first value of the chain $\boldsymbol{\theta}^{(0)}$ to some (reasonable) value.

At the rth value of the chain

- Let $\boldsymbol{\theta} = \boldsymbol{\theta}^{(r)}$ be the current status of the chain.
- Sample $\boldsymbol{\theta}^*$ from a **proposal distribution** $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$.
- Compute the **acceptance probability**, defined as

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X})} \frac{q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}\right\} = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)\pi(\boldsymbol{X} \mid \boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\pi(\boldsymbol{X} \mid \boldsymbol{\theta})} \frac{q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}\right\}.$$

- With probability α, **update the status** of the chain and set $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$.

Implementation of MCMC

- Here we focus on **practical considerations** concerning the implementation with **R**. Higher performance can be achieved using C++ and the **Rcpp** package (i.e. unit A.2).
- This is far from a comprehensive guide about **R** programming. We will consider a specific model and we will implement the relevant code in **R**.

What about BUGS / JAGS / Stan?

- If the performance is not a concern, Stan-like software is an extremely useful tool for **practitioners** who wish to implement standard Bayesian models.
- Conversely, any non-standard or novel model, i.e. those usually developed by researchers in statistics, may be difficult or even impossible to implement.
- Besides, the "manual" implementation is very useful to **gain insights** about the model itself and it facilitates a lot the **debugging** process.

Example II: Weibull model for censored data

- We consider an example from survival analysis, i.e. the data are **survival times** which may be **censored**.
- In this example, we assume that the survival times are iid random variable following a Weibull distribution Weib(γ, β).
- The observed survival time t_i is either **complete** $(d_i = 1)$ or **right censored** $(d_i = 0)$, meaning that the survival time is higher than the observed t_i.
- The **hazard** and **survival** functions of a Weibull distribution are

$$h(t \mid \gamma, \beta) = \frac{\gamma}{\beta}\left(\frac{t}{\beta}\right)^{\gamma-1}, \quad S(t \mid \gamma, \beta) = \exp\left\{-\left(\frac{t}{\beta}\right)^{\gamma}\right\}.$$

- Recall that the **density** function is obtained as $f(t \mid \gamma, \beta) = h(t \mid \gamma, \beta)S(t \mid \gamma, \beta)$.

Likelihood function

- The **likelihood** for this parametric model, under suitable censorship assumptions, is **proportional** to the following quantity

$$\pi(\mathbf{t}, \mathbf{d} \mid \boldsymbol{\theta}) \propto \prod_{i=1}^{n} h(t_i \mid \gamma, \beta)^{d_i} S(t_i \mid \gamma, \beta) = \prod_{i:d_i=1} f(t_i \mid \gamma, \beta) \prod_{i:d_i=0} S(t_i \mid \gamma, \beta),$$

  with (γ, β) being the parameter vector.

- **Remark** When performing (Bayesian) inference, note that the likelihood is always defined up to an **irrelevant normalizing constant** not depending on the parameters $\boldsymbol{\theta}$.

- These irrelevant constants **can and should be omitted** when performing computations, especially if they are expensive to evaluate.

Bad implementation I (use the log-scale)

- In our experiments, we make use the `stanford2` dataset of the `survival` package.
- In first place, we need to implement the log-likelihood function, say `loglik`.
- The following implementation of the log-likelihood is correct but **numerically unstable**.

```
loglik_inaccurate <- function(t, d, gamma, beta) {
  hazard <- prod((gamma / beta * (t / beta)^(gamma - 1))^d)
  survival <- prod(exp(-(t / beta)^gamma))
  log(hazard * survival)
}

# Evaluate the log-likelihood at the point (0.5, 1000)
loglik_inaccurate(t, d, gamma = 0.5, beta = 1000)
# [1] -Inf
```

- The product of several terms close to 0 leads to numerical inaccuracies $\Longrightarrow$ **use the log-scale** instead.

Bad implementation II (initialize the output)

- This second coding attempt relies on the log-scale and is indeed numerically much more stable than the previous version.
- However, this implementation is inefficient $\Longrightarrow$ **do not increase objects' dimension**.

```
loglik_inefficient2 <- function(t, d, gamma, beta) {
  n <- length(t) # Sample size
  log_hazards <- NULL
  log_survivals <- NULL

  for (i in 1:n) {
    log_hazards <- c(log_hazards, d[i] * ((gamma - 1) * log(t[i] / beta) + log(gamma / beta)))
    log_survivals <- c(log_survivals, -(t[i] / beta)^gamma)
  }
  sum(log_hazards) + sum(log_survivals)
}

# Evaluate the log-likelihood at the point (0.5, 1000)
loglik_inefficient2(t, d, gamma = 0.5, beta = 1000)
# [1] -873.3299
```

Bad implementation III (avoid `for` loops)

- This third attempt avoids the previous pitfalls but it is still quite inefficient $\Longrightarrow$ **use vectorized code** whenever possible.

```
loglik_inefficient1 <- function(t, d, gamma, beta) {
  n <- length(t) # Sample size
  log_hazards <- numeric(n)
  log_survivals <- numeric(n)

  for (i in 1:n) {
    log_hazards[i] <- d[i] * ((gamma - 1) * log(t[i] / beta) + log(gamma / beta))
    log_survivals[i] <- -(t[i] / beta)^gamma
  }
  sum(log_hazards) + sum(log_survivals)
}

# Evaluate the log-likelihood at the point (0.5, 1000)
loglik_inefficient1(t, d, gamma = 0.5, beta = 1000)
# [1] -873.3299
```

Good implementation

- The following version is both **numerically stable** and **efficient**.

```
loglik <- function(t, d, gamma, beta) {
  log_hazard <- sum(d * ((gamma - 1) * log(t / beta) + log(gamma / beta)))
  log_survival <- sum(-(t / beta)^gamma)
  log_hazard + log_survival
}

# Evaluate the log-likelihood at the point (0.5, 1000)
loglik(t, d, gamma = 0.5, beta = 1000)
# [1] -873.3299
```

- All these versions of `loglik` run in fractions of seconds. However, the `loglik` function must be executed i.e. $\sim 10^5$ times within a MH algorithm.

- Moreover, in more complex models several instances of these inefficiencies add up.

Benchmarking the code

- To understand which function works better, you need to **test its performance**.
- There exist specialized packages to do so, i.e. **R** `rbenchmark` or `microbenchmark`.
- These packages execute the code several times and report the **average execution time**.
- The column "`elapsed`" refer to the overall time (in seconds) over 1000 replications.

```
library(rbenchmark) # Library for performing benchmarking

benchmark(
  loglik1 = loglik(t, d, gamma = 0.5, beta = 1000),
  loglik2 = loglik_inefficient1(t, d, gamma = 0.5, beta = 1000),
  loglik3 = loglik_inefficient2(t, d, gamma = 0.5, beta = 1000),
  columns = c("test", "replications", "elapsed", "relative"),
  replications = 1000
)

#       test replications elapsed relative
#1 loglik1         1000   0.014    1.000
#2 loglik2         1000   0.079    5.643
#3 loglik3         1000   0.412   29.429
```

A matter of style

- **Formatting your code** properly is a healthy programming practice.
- You can refer to `https://style.tidyverse.org` for a comprehensive overview of good practices in **R**.
- Quoting the **tidyverse style guide**: "*Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread*".
- The `styler` **R** package automatically restyles your code for you and it is integrated within **RStudio** as an add-in.

```
# Good
x <- 5

# Bad
x = 5
```

Reparametrizations I

- When performing (Bayesian) inference, the choice of the **parametrization** has strong impacts on computations.
- **General advice**: perform computations on the most convenient parametrization and then transform back the obtained samples.
- As a rule of thumb, you should use parametrizations with **unbounded domains**. This facilitates the choice of proposal distributions and could also **improve the mixing**.
- In our model, the two parameters γ, β are strictly positive. Hence, a common strategy is to consider their logarithm, i.e. $\boldsymbol{\theta} = (\theta_1, \theta_2) = (\log \gamma, \log \beta)$.

To log or not to log?

Roberts, G. O. and Rosenthal, J. S. (2009). *Examples of adaptive MCMC*. Journal of Computational and Graphical Statistics, **18**(2), 349–367.

Reparametrizations & priors

- When reparametrizations are involved, there two possible modeling strategies.
- Choose the prior **before** the reparametrization. In our setting, we could let for example

$$\gamma \sim \text{Ga}(0.1, 0.1), \qquad \beta \sim \text{Ga}(0.1, 0.1).$$

  If you do so, remember to include the **jacobian** of the transformation when considering the transformed posterior!

- Choose the prior **after** the reparametrization. In our setting, we could let for example

$$\theta_1 = \log(\gamma) \sim \text{N}(0, 100), \qquad \theta_2 = \log(\beta) \sim \text{N}(0, 100).$$

  This strategy is simpler as it avoids the extra step of computing the jacobian.

```
logprior <- function(theta) {
  sum(dnorm(theta, 0, sqrt(100), log = TRUE))
}

logpost <- function(t, d, theta) {
  loglik(t, d, exp(theta[1]), exp(theta[2])) + logprior(theta)
}
```

The MH implementation

- Since the space of $\boldsymbol{\theta}$ is unbounded, it is reasonable to select a **Gaussian random walk** as proposal distribution, namely

$$(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) \sim \mathrm{N}_2(\boldsymbol{\theta}, 0.25^2 I_2).$$

  The choice of the variance will be discussed in unit B.1.

- Gaussian random walks are **symmetric** proposals distributions, implying that

$$q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*) = q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}),$$

  which means that their ratio can be simplified $(= 1)$ when computing the acceptance probability α.

- As before, make sure you compute α using the log-scale to avoid numerical instabilities.

- **Remark**. Unfortunately, there is no way to avoid `for` loops, which are highly inefficient $\Longrightarrow$ this justifies the usage of **Rcpp** and **RcppArmadillo**.

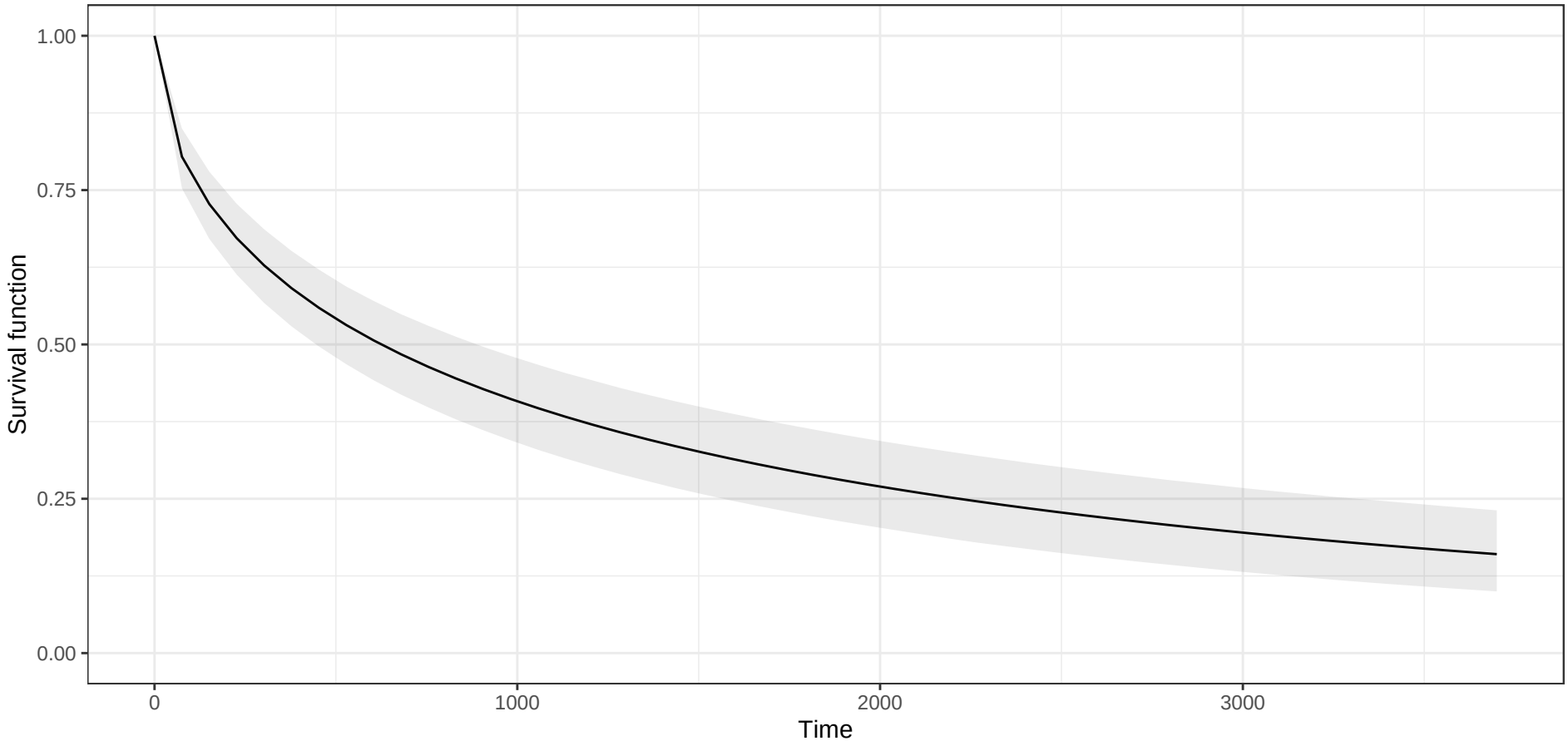
Metropolis-Hastings code

```
RMH <- function(R, burn_in, t, d) {
  out <- matrix(0, R, 2) # Initialize an empty matrix to store the values
  theta <- c(0, 0) # Initial values
  logp <- logpost(t, d, theta) # Log-posterior
  for (r in 1:(burn_in + R)) {
    theta_new <- rnorm(2, mean = theta, sd = 0.25) # Propose a new value
    logp_new <- logpost(t, d, theta_new)
    alpha <- min(1, exp(logp_new - logp))
    if (runif(1) < alpha) {
      theta <- theta_new; logp <- logp_new # Accept the value
    }
    if (r > burn_in) {
      out[r - burn_in, ] <- theta     # Store the values after the burn-in period
    }
  }
  out
}
```

```
# Executing the code
library(tictoc) # Library for "timing" the functions
tic()
fit_MCMC <- RMH(R = 50000, burn_in = 5000, t, d)
toc()
# 0.92 sec elapsed
```

Estimated survival function

- Posterior mean of the survival function with pointwise 95% credible intervals.



The Gibbs sampling algorithm

Gibbs sampling

- We now introduce another Markov Chain Monte Carlo method: the **Gibbs Sampling**.
- Recall that $\pi(\boldsymbol{\theta} \mid \boldsymbol{X})$ denotes the posterior distribution of $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$ given the data.
- Let us partition the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L)$ into L blocks of parameters. Sometimes we will have as many blocks as parameters, so that $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$.
- Let $\pi(\boldsymbol{\theta}_\ell \mid -)$ be the so-called **full-conditional** of $\boldsymbol{\theta}_\ell$, that is

$$\pi(\boldsymbol{\theta}_\ell \mid -) = \pi(\boldsymbol{\theta}_\ell \mid \boldsymbol{X}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{\ell-1}, \boldsymbol{\theta}_{\ell+1}, \ldots, \boldsymbol{\theta}_L), \quad \ell = 1, \ldots, L,$$

  namely the conditional distribution of $\boldsymbol{\theta}_\ell$ given the **data** and the **other parameters**.
- Repeatedly sampling $\boldsymbol{\theta}_\ell$, for $\ell = 1, \ldots, L$, from the corresponding full conditionals leads to a MCMC algorithm targeting the posterior distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{X})$.

Connection with the hybrid Metropolis-Hastings

- The Gibbs sampler is a **special case** of the hybrid Metropolis-Hastings, in which the **full conditionals** are used as **proposal distribution**.
- The general hybrid MH is indeed often called **Metropolis-within-Gibbs**.
- Suppose that $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$. Then it can be shown that

$$\frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X})} = \frac{\pi(\theta_j^* \mid \boldsymbol{X}, \boldsymbol{\theta}_{-j})}{\pi(\theta_j \mid \boldsymbol{X}, \boldsymbol{\theta}_{-j})}.$$

- In addition, note that the **acceptance probabilities** of the hybrid MH algorithm are

$$\alpha_j = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X})} \frac{q_j(\theta_j \mid \boldsymbol{\theta}^*)}{q_j(\theta_j^* \mid \boldsymbol{\theta})}\right\} = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^* \mid \boldsymbol{X})}{\pi(\boldsymbol{\theta} \mid \boldsymbol{X})} \frac{\pi(\theta_j \mid \boldsymbol{X}, \boldsymbol{\theta}_{-j})}{\pi(\theta_j^* \mid \boldsymbol{X}, \boldsymbol{\theta}_{-j})}\right\} = 1.$$

General considerations

- The **acceptance rate** of the Gibbs sampler is uniformly equal to 1.
- The use of Gibbs-sampler requires the knowledge of the full-conditional distributions, from which we should be able to sample.
- The Gibbs sampling is "automatic", in the sense that there are no tuning parameters that we need to choose, which is both good and bad news.
- Ergodicity and convergence to the posterior stationary distribution are ensured under very mild condition, i.e. requiring the connectedness of the support.
- The **Hammersley-Clifford** theorem implies that a sufficiently regular joint density can be expressed as a function of the full-conditionals.

Example: non-connected support

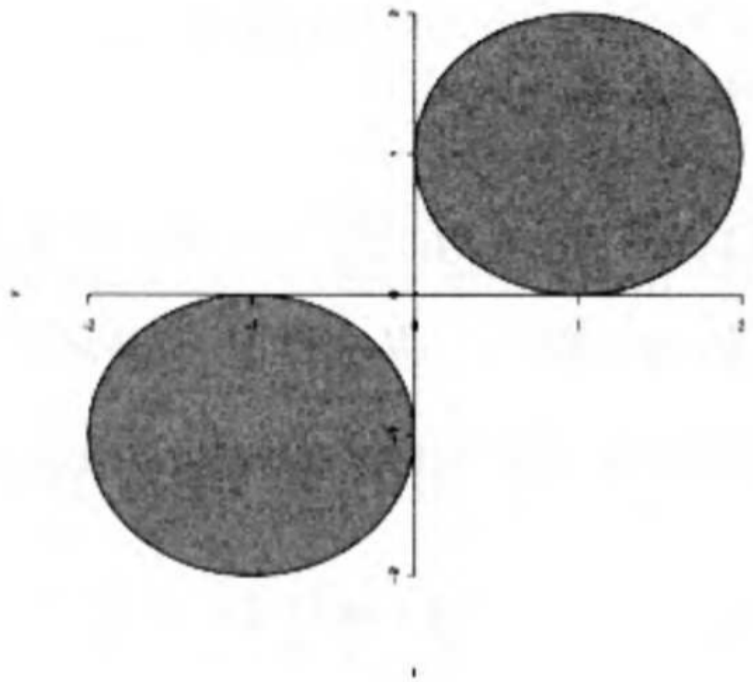


Fig. 10.1. Support of the function $f(x_1, x_2)$ of (10.3)

Example of non-connected support: gray areas have positive probability. Picture taken from Robert and Casella (2004).

Example: conditionally-conjugate Gaussian model

- Let us assume the observations $(x_1, \ldots, x_n)$ are draws from

$$(x_i \mid \mu, \sigma^2) \overset{\text{iid}}{\sim} \mathsf{N}(\mu, \sigma^2), \qquad i = 1, \ldots, n.$$

  with independent priors $\mu \sim \mathsf{N}(\mu_\mu, \sigma^2_\mu)$ and $\sigma^{-2} \sim \mathsf{Ga}(a_\sigma, b_\sigma)$.

- The full conditional distribution for the **mean** μ is:

$$(\mu \mid -) \sim \mathsf{N}\left(\mu_n, \sigma^2_n\right), \qquad \mu_n = \sigma^2_n \left(\frac{\mu_\mu}{\sigma^2_\mu} + \frac{1}{\sigma^2} \sum_{i=1}^{n} x_i \right), \quad \sigma^2_n = \left(\frac{n}{\sigma^2} + \frac{1}{\sigma^2_\mu} \right)^{-1}.$$

- The full conditional distribution for the **precision** σ^{-2} is:

$$(\sigma^{-2} \mid -) \sim \mathsf{Ga}\left(a_n, b_n\right), \qquad a_n = a_\sigma + n/2, \quad b_n = b_\sigma + \frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^2.$$

Example: conditionally-conjugate Gaussian model

```r
gibbs_R <- function(x, mu_mu, sigma2_mu, a_sigma, b_sigma, R, burn_in) {
  # Initialization
  n <- length(x); xbar <- mean(x)
  out <- matrix(0, R, 2)
  # Initial values for mu and sigma
  sigma2 <- var(x); mu <- xbar
  for (r in 1:(burn_in + R)) {
    # Sample mu
    sigma2_n <- 1 / (1 / sigma2_mu + n / sigma2)
    mu_n <- sigma2_n * (mu_mu / sigma2_mu + n / sigma2 * xbar)
    mu <- rnorm(1, mu_n, sqrt(sigma2_n))
    # Sample sigma2
    a_n <- a_sigma + 0.5 * n
    b_n <- b_sigma + 0.5 * sum((x - mu)^2)
    sigma2 <- 1 / rgamma(1, a_n, b_n)
    # Store the values after the burn-in period
    if (r > burn_in) {
      out[r - burn_in, ] <- c(mu, sigma2)
    }
  }
  out
}
```

Example: conditionally-conjugate Gaussian model