# Computational Statistics II

## Unit B.1: Optimal scaling & adaptive Metropolis

**Tommaso Rigon**

**University of Milano-Bicocca**

Ph.D. in Economics, Statistics and Data Science

# Unit B.1

## Main concepts

- Optimal scaling for Metropolis-Hastings algorithm
- Metropolis-within-Gibbs algorithm
- Adaptive MCMC

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

## Main references


- Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. *Statistical Science*, **32**(1), 64–87.
- Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, **16**(4), 351–367.
- Roberts, G. O. and Rosenthal, J. S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, **18**(2), 349–367.

# Random walk Metropolis-Hastings

- Let us consider a **random walk** Metropolis-Hastings (RWM) algorithm and let $\boldsymbol{\theta} = \boldsymbol{\theta}^{(r)}$ be the current status of the chain.
- It is called "random walk" because we sample $\boldsymbol{\theta}^*$ from a **Gaussian proposal** distribution

$$(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}) \sim \mathrm{N}_p(\boldsymbol{\theta}^{(r)}, \boldsymbol{S}), \qquad \text{implying that} \qquad q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) = q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*).$$

- In this special case the **acceptance probability** simplifies and we get

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

- This Gaussian proposal distribution is a sensible choice especially whenever the support of $\boldsymbol{\theta}$ is unbounded.

# Optimal choice of the proposal distribution

- Among all the possible proposal densities for $q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)$, we restrict our focus on multivariate Gaussians centered on $\boldsymbol{\theta}^*$.
- Despite this important simplification, choosing the covariance matrix $\boldsymbol{S}$ remains a difficult task and crucially affects the performance.
- In the **univariate / bivariate cases**, one could tune the variance of the proposal distribution $\boldsymbol{S}$ by **trial and error** and with some patience.
- Unfortunately, whenever the parameter's dimension is large, the "manual" elicitation of the matrix $\boldsymbol{S}$ is almost impossible.
- **Key question**. Can we identify an ideal covariance matrix $\boldsymbol{S}$ that is optimal in some sense? Can we "estimate" it from the data?

# Asymptotic variance

- For an arbitrary squared-integrable function $g(\cdot) : \mathbb{R}^p \to \mathbb{R}$, let us consider the Monte Carlo estimator

$$\hat{\eta}_g = \frac{1}{R} \sum_{r=1}^{R} g(\boldsymbol{\theta}^{(r)}),$$

  for the posterior expectation $\eta_g = \mathbb{E}\{g(\boldsymbol{\theta}) \mid \boldsymbol{X}\}$.

- If a central limit theorem holds, we have that

$$\sqrt{R}\frac{\hat{\eta}_g - \eta_g}{\sigma_g} \xrightarrow{d} Z, \qquad Z \sim \mathrm{N}(0, 1),$$

  where $\sigma_g^2$ is the so-called **asymptotic variance** of the MCMC algorithm.

- Intuitively, we seek a covariance matrix $\boldsymbol{S}$ that minimizes the asymptotic variance $\sigma_g^2$.

- Other measures of "optimality" can be defined, but it can be shown they are all equivalent asymptotically (for large values of $p$).

# Asymptotic variance

- Let $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$ be a Markov chain, with $\boldsymbol{\theta}^{(0)} \sim \pi(\boldsymbol{\theta} \mid \boldsymbol{X})$ being a sample from the stationary distribution.

- The asymptotic variance can be written as follows

$$\sigma_g^2 = \text{var}\{g(\boldsymbol{\theta}^{(0)}) \mid \boldsymbol{X}\}\, \tau_g = \text{var}\{g(\boldsymbol{\theta}^{(0)}) \mid \boldsymbol{X}\} \left[1 + 2 \sum_{r=1}^{\infty} \text{Corr}\{g(\boldsymbol{\theta}^{(0)}), g(\boldsymbol{\theta}^{(r)})\}\right].$$

- The quantity $\tau_g$ is sometimes called **integrated autocorrelation time** and measures the **loss of efficiency** with respect to independent (iid) sampling ($\tau_g = 1$).

- When $\tau_g = 1$, the MCMC algorithm is "optimal" and there is no autocorrelation.

- Rarely, one could obtain $\tau_g < 1$, which is indeed an improvement over iid sampling.

- The `effectiveSize` **R** command produces an estimate of $R\tau_g^{-1}$ from the empirical samples of the chain.

# Optimal scaling

- The relationship between the matrix $\boldsymbol{S}$ and the asymptotic variance $\sigma_g^2$ is unclear. In addition, the variance $\sigma_g^2$ depends on the chosen function $g(\cdot)$.

- Let us initially assume that the posterior distribution has the following form

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{X}) = \prod_{j=1}^{p} f(\theta_j), \qquad \text{var}(\boldsymbol{\theta} \mid \boldsymbol{X}) = \sigma^2 I_p$$

  meaning that the components of $\boldsymbol{\theta}$ are independent and identically distributed from some density $f$.

- Moreover, we consider the following proposal distribution

$$(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}) \sim \mathrm{N}_p(\boldsymbol{\theta}^{(r)}, s_p^2 I_p), \qquad s_p^2 = \ell^2/p.$$

- In this simplified setting, we seek an **optimal scaling** value for $\ell^2$.

# Diffusion processes

- This problem simplifies remarkably in the **asymptotic** regime, as $p$ diverges.
- Let us define a continuous-time stochastic process from the first component $\theta_1$ of the $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$, namely:

$$Z^{(t)} = \theta_1^{([tp])}$$

  where $[\,\cdot\,]$ denotes the integer part function.
- That is, $Z^{(t)}$ is a **speeded-up** continuous-time version of the original algorithm, parametrized to make jumps every $p^{-1}$ time units.
- We need some smoothness conditions on the density $f$ (Roberts et al., 1997), and in particular we assume that

$$\mathcal{I} = \mathbb{E}\left[\left\{\frac{f'(\theta_1)}{f(\theta_1)}\right\}^2\right] < \infty,$$

  is well-defined. The quantity $\mathcal{I}$ equals $\sigma^{-2}$ in the Gaussian case.

# Diffusion processes

## Theorem

Let $B^{(t)}$ be the standard Brownian motion and let $\Phi(\cdot)$ be the cdf of a standard Gaussian. The continuous-time stochastic process $Z$ weakly converges to

$$Z \xrightarrow{d} W, \qquad p \to \infty,$$

where $W$ is a diffusion process satisfying the stochastic differential equation

$$\mathrm{d}W^{(t)} = h(\ell)^{1/2}\mathrm{d}B^{(t)} + \frac{h(\ell)\nabla \log f(W^{(t)})}{2}\mathrm{d}t,$$

where the **speed of the diffusion** is

$$h(\ell) = \ell^2\, 2\Phi\left(-\frac{\mathcal{I}^{1/2}\ell}{2}\right).$$

- **Note**. All the involved quantities have a clear interpretation in terms of the original RWM algorithm.

# Speed of the diffusion $h(\ell)$

- The speed of the diffusion $h(\ell)$ is strictly related to the asymptotic variance of the MCMC algorithm.
- Recall that we aim at finding an **optimal value** for $\ell$ that minimizes the autocorrelation.
- In the first place, note that for small $\epsilon > 0$, it holds that

$$\text{Corr}\{g(W^{(0)}), g(W^{(\epsilon)})\} \approx 1 - \epsilon B_g h(\ell),$$

  where $B_g$ is a constant not depending on $\ell$.
- Let $\tau_g(\ell)$ be the integrated autocorrelation of the RWM. Then, for large $p$ it holds that

$$\tau_g(\ell)^{-1} \approx h(\ell)\, e_g\, p^{-1},$$

  where $e_g > 0$ is some constant depending only on $g(\cdot)$.
- **Remark**. The maximization of the diffusion speed is equivalent to the minimization of the autocorrelation for any function $g(\cdot)$.

# Speed of diffusion and acceptance rate

- Let us define the acceptance rate of the original $p$-dimensional RWM as

$$A_p(\ell) = \lim_{R \to \infty} \frac{\text{“\# of accepted moves”}}{R},$$

namely the long-term proportion of accepted moves.

- Then, it can be shown that

$$\lim_{p \to \infty} A_p(\ell) = A(\ell) = 2\Phi\left(-\frac{\mathcal{I}^{1/2}\ell}{2}\right).$$

- Moreover, recall the definition of the speed of diffusion

$$h(\ell) = \ell^2\, 2\Phi\left(-\frac{\mathcal{I}^{1/2}\ell}{2}\right) = \ell^2\, A(\ell),$$

implying that the **speed of the diffusion** is strictly related to the **acceptance rate**.

# Important consequences

- Hence, we consider $\ell$ maximizing the speed of diffusion $h(\ell)$, obtaining the **optimal scaling**

$$\ell_{\text{opt}} \approx \frac{2.38}{\mathcal{I}^{1/2}}.$$

- The acceptance rate, evaluated at the optimal scaling, is

$$A(\ell_{\text{opt}}) \approx 0.234,$$

  corresponding to the **optimal acceptance rate**.

- This suggests the following optimal proposal variance for $(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}) \sim \mathrm{N}_p(\boldsymbol{\theta}^{(r)}, s_p^2 I_p)$ for large values of $p$, with

$$s_p^2 = 2.38^2 \, (p \, \mathcal{I})^{-1},$$

  where $\mathcal{I}$ must be estimated/guessed somehow.

- If $f$ is a Gaussian density with variance $\sigma^2$, then we obtain $s_p^2 = 2.38^2 \sigma^2 \, p^{-1}$.

# Getting practical

- These results are asymptotic (large $p$) and require that the posterior distribution has independent components.
- In practice, when $p \approx 5$, the optimal acceptance rate is close to 0.234 based on simulation studies (Gelman et al., 1996).
- When $p = 1$ the optimal acceptance rate is higher and about 0.44.
- **Key extension**. If the posterior distribution is Gaussian with $p \times p$ covariance matrix $\boldsymbol{\Sigma}$, it suffices to translate the components and **rotate the axes** according to $\boldsymbol{\Sigma}$ to make the components iid, leading to

$$(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}) \sim \mathrm{N}_p(\boldsymbol{\theta}^{(r)}, \boldsymbol{S}), \qquad \boldsymbol{S} = 2.38^2 \, \boldsymbol{\Sigma} \, / \, p.$$

- This procedure is **optimal** for large $p$, although it requires the knowledge of $\boldsymbol{\Sigma}$.

# Binary regression

- Let $\mathbf{y} = (y_1, \ldots, y_n)^\intercal$ be a vector of the observed **binary responses**.
- Let $\mathbf{X}$ be the corresponding **design matrix** whose generic row is $\mathbf{x}_i = (1, x_{i2}, \ldots, x_{ip})^\intercal$, for $i = 1, \ldots, n$. All predictors have been **standardized**.
- We consider a generalized linear model such that

$$(y_i \mid \pi_i) \overset{\text{ind}}{\sim} \text{Bern}(\pi_i), \qquad \pi_i = g(\eta_i), \qquad \eta_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

  where $g(\cdot)$ is either the inverse logit transform or the cdf of a standard normal. We focus here on the **logistic regression case**.
- We aim at estimating the parameter vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\intercal$ using RWM.
- We will employ a relatively vague prior centered at 0, namely

$$\boldsymbol{\beta} \sim N_p(0, 100\, I_p).$$

# The Pima Indian dataset

- During the course, we will test several algorithms on the "famous" **Pima Indian dataset**, with $n = 532$ and $p = 8$.
- The purpose of this exercise is mainly to present the implementation of the various MCMC algorithms and show their performance in this specific example.
- **Warning**. The following results should not be generalized to any statistical models nor even to any logistic regression model.
- Depending on the sample size $n$, the dimension of the parameter space $p$, as well as the dependence structure of the predictor, the results may vary significantly.
- Refer to the nice paper by Chopin & Ridgway (2017) for a more comprehensive discussion on this aspect.

# Computational details

- Recall that at each step of the algorithm, we need a sample from a multivariate Gaussian distribution $N_p(0, \boldsymbol{S})$.
- Albeit tempting, using built-in **R** functions such as `rmvnorm` and `mvrnorm` leads to a sensible **waste of computing time**.
- Indeed, in order to get a sample from $V \sim N_p(\mu, \boldsymbol{S})$, one needs to compute

$$V = \mu + \boldsymbol{A}\boldsymbol{Z},$$

  where $\boldsymbol{Z} \sim N(0, I_p)$ is standard Gaussian and $\boldsymbol{A}$ is a $p \times p$ matrix such that $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{S}$.
- Hence, there is no need to compute $\boldsymbol{A}$ at every step, as this can be done before running the MCMC.

```
A <- chol(S)  # Cholesky decomposition of S (outside the MCMC)
V <- mu + crossprod(A, rnorm(2)) # Sample from V (inside the MCMC)
```

# Naive covariance matrix

- Let us start with a **naive choice** for the proposal covariance $\boldsymbol{S} = \text{diag}\{10^{-3}, \ldots, 10^{-3}\}$.
- Albeit being sub-optimal, this "random" choice of $\boldsymbol{S}$ works decently.

```
# Covariance matrix of the proposal
S <- diag(1e-3, ncol(X))

# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S)) # Convert the matrix into a "coda" object

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   174.9   205.0   258.5   259.6   320.7   333.1

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   90.06   93.56  119.31  122.76  146.43  171.52

summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.7191  0.7191  0.7191  0.7191  0.7191  0.7191
```

# Approximating the posterior covariance matrix

- We know that a sensible choice for $\boldsymbol{S}$ would be based on posterior covariance matrix $\boldsymbol{\Sigma}$.
- The true $\boldsymbol{\Sigma}$ is unknown and therefore we need to rely on some (fast) **approximation**.
- A possibility is based on a **quadratic approximation** of the likelihood function, evaluated at the maximum likelihood estimate.
- This is particularly simple in the logistic regression case (do it as an exercise!) since we can set

$$\hat{\boldsymbol{\Sigma}} = (\boldsymbol{X}^T \hat{\boldsymbol{H}} \boldsymbol{X})^{-1}, \qquad \hat{\boldsymbol{H}} = \text{diag}\{\hat{\pi}_1(1-\hat{\pi}_1), \ldots, \hat{\pi}_n(1-\hat{\pi}_n)\},$$

  where $\hat{\pi}_i = [1 + \exp\{-(x_{i1}\hat{\beta}_{1,\text{ML}} + \cdots + x_{ip}\hat{\beta}_{p,\text{ML}})\}]^{-1}$.
- This estimate $\hat{\boldsymbol{\Sigma}}$ corresponds to the **Fisher information**, evaluated at the MLE.
- This is a variant of the **Laplace approximation** that ignores the prior contribution. For a more general and detailed explanation, refer to Chopin and Ridgway (2017).

# Laplace covariance matrix

- Let us use a covariance matrix based on the Laplace approximation $\boldsymbol{S} = 2.38^2 \, \hat{\boldsymbol{\Sigma}} \, / \, p$.
- This choice for $\boldsymbol{S}$ is almost optimal, and the effective sample size is much higher.

```
# Covariance matrix is selected using a Laplace approximation
fit_logit <- glm(type ~ X - 1, family = binomial(link = "logit"), data = Pima)
S <- 2.38^2 * vcov(fit_logit) / ncol(X) # The desired matrix is extracted using vcov

# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S))

summary(effectiveSize(fit_MCMC)) # Effective sample size
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   1107    1174    1206    1194    1228    1245

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  24.10   24.43   24.87   25.15   25.56   27.10

summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.2746  0.2746  0.2746  0.2746  0.2746  0.2746
```

# Adaptive MCMC

- In several cases, it is not possible to come up with a fast and reasonable estimate $\hat{\Sigma}$.
- Hence, a possibility is tuning the covariance matrix $\boldsymbol{S}$ **on the fly**, namely using the previously obtained samples.
- **Warning**. This is no longer a MH algorithm; therefore, the chain is not necessarily converging to the correct stationary distribution or converging.
- However, in many cases **ergodicity** of the chain is **preserved** as long as we adaptively tune $\boldsymbol{S}$ in a reasonable manner.
- The key condition is called **diminishing adaptation**, which essentially means that the changes in $\boldsymbol{S}$ are negligible as $R \to \infty$; see Roberts & Rosenthal (2009).

# Adaptive Metropolis

- An example of adaptive MCMC is the so-called **adaptive Metropolis** (AM) algorithm.
- We implement here a **version** of the AM which makes use of the following proposal distribution

$$q_r(\beta^* \mid \beta) \sim N(\beta,\, 2.38^2/p\, \boldsymbol{\Sigma}_r + \epsilon I_p),$$

  where $\boldsymbol{\Sigma}_r$ is the covariance matrix of the previously $r$ sampled values $\beta^{(1)}, \ldots, \beta^{(r)}$.
- The constant $\epsilon > 0$ is some small value that avoid degeneracies. We will use $\epsilon = 10^{-6}$.
- Moreover, note that the following **recursive formula** holds true:

$$\boldsymbol{\Sigma}_r = \frac{1}{r-1}\sum_{j=1}^{r}(\beta^{(j)} - \bar{\beta}^{(r)})(\beta^{(j)} - \bar{\beta}^{(r)})^\intercal = \frac{r-2}{r-1}\boldsymbol{\Sigma}_{r-1} + \frac{1}{r}(\beta^{(r)} - \bar{\beta}^{(r-1)})(\beta^{(r)} - \bar{\beta}^{(r-1)})^\intercal.$$

  where $\bar{\beta}^{(r)} = (r-1)/r\bar{\beta}^{(r-1)} + \beta^{(r)}/r$ is the arithmetic means of the first $r$ values.
- Several **variants** of this scheme exist, but the core idea is trying to estimate $\boldsymbol{\Sigma}$.

# Adaptive Metropolis

- We obtain results that are comparable to the MH based on the Laplace approximation in terms of effective sample size.
- However, the **computing time** is much **higher**, because we need to decompose $\boldsymbol{S}$ at each iteration.

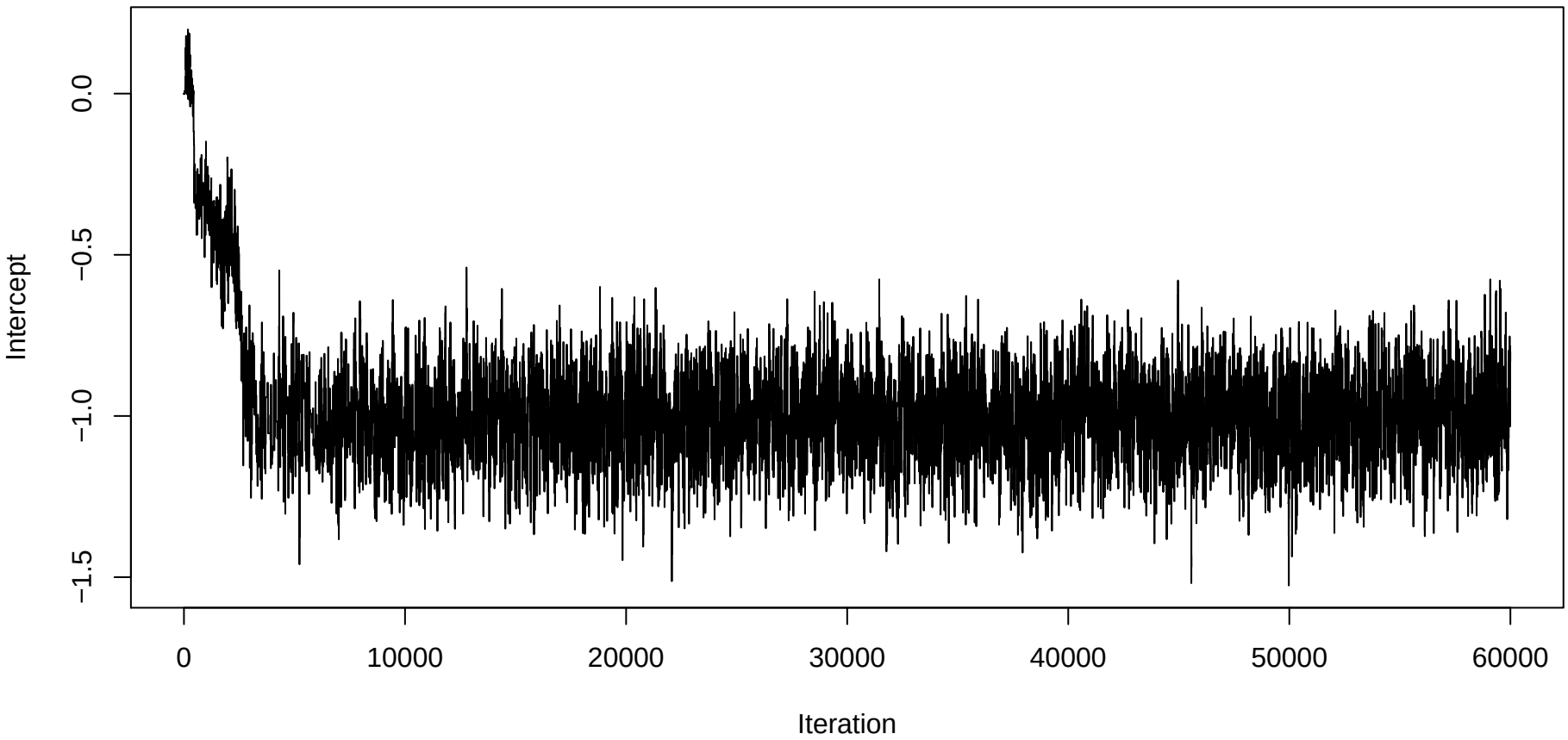
```
# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH_Adaptive(R = R, burn_in = burn_in, y, X))

# summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   856.7   905.9  1124.5  1110.9  1269.2  1412.6

# summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   21.24   23.65   26.69   27.89   33.12   35.02

# summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  0.1907  0.1907  0.1907  0.1907  0.1907  0.1907
```

# Traceplot of $\beta_1$, including the burn-in



# Metropolis-within-Gibbs (recap)

- The **Metropolis-within-Gibbs** algorithm is an MCMC algorithm that combines the MH and the Gibbs sampling algorithms.
- Let $\pi(\theta_j \mid -)$ be the so-called **full-conditional** of $\theta_j$, that is

$$\pi(\theta_j \mid -) = \pi(\theta_j \mid \boldsymbol{X}, \theta_1, \ldots, \theta_{j-1}, \theta_{j+1}, \ldots, \theta_p), \quad j = 1, \ldots, p,$$

  namely the conditional distribution of $\theta_j$ given the data and the other parameters.
- In the Metropolis-within-Gibbs, we proceed as in a standard Gibbs sampling, but instead of drawing from the full conditional $\pi(\theta_j \mid -)$, we conduct a **Metropolis step**.
- We propose a value from $q(\theta_j^* \mid \theta_j)$, typically a univariate Gaussian random walk, that we accept/reject in the usual manner.
- This means that at each step of the chain, some parameters are updated, others are not. This produces **local moves** rather than **global moves**.

# Metropolis-within-Gibbs

- We use random walk proposals $(\theta_j^* \mid \theta_j) \sim N_1(\theta_j, s_j^2)$, for $j = 1, \ldots, p$.
- In this first experiment we set $s_1^2 = \cdots = s_p^2 = 10^{-4}$.
- These results are **unacceptable**. We need a better specification for the variances $s_j^2$.

```
p <- ncol(X) # Dimension of the parameter space
se <- sqrt(rep(1e-4, p)) # Standard deviations of the proposal distributions

# Running the MCMC (R = 30000, burn_in = 30000)
fit_MCMC <- as.mcmc(RMH_Gibbs(R = R, burn_in = burn_in, y, X, se))

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   27.02   36.43   37.37   37.57   40.58   44.21
summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   678.6   740.1   802.8   814.8   824.1  1110.1
summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.9682  0.9685  0.9697  0.9698  0.9710  0.9719
```

# Adaptive Metropolis-within-Gibbs

- In order to get a better mixing, we could **adaptively** choose the variances $s_j^2$ as in Roberts and Rosenthal (2009).
- Since the updates are univariate, we can rely on a more direct adaptive approach targeting the **optimal acceptance rate**, which is 0.44.
- Every 50 iterations (a batch), the algorithm increases or decreases the standard errors $s_j$ according to the fraction of accepted values among the 50 batch values.
- It is convenient to work in the **logarithmic scale**, to facilitate the exploration of the space of suitable values.
- If the fraction of accepted values for the $j$th component is **higher**/**lower** than 0.44, then we **increase**/**decrease** the corresponding $\log s_j$ by the quantity $\min\{0.01, 1/\sqrt{r}\}$.
- Note that the **diminishing adaptation** condition is satisfied, as the correction is vanishing as $r$ (the number of iterations) increases.

# Adaptive Metropolis-within-Gibbs

- These results are comparable with the other suitably tuned MH approaches.
- However, the computing time is much higher.
- Note that the overall acceptance rates are close to 0.44.

```
fit_MCMC <- as.mcmc(RMH_Gibbs_Adaptive(R = R, burn_in = burn_in, y, X)) #

summary(effectiveSize(fit_MCMC)) # Effective sample size (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   653.2   733.1  1021.5  1009.3  1293.6  1373.3

summary(R / effectiveSize(fit_MCMC)) # Integrated autocorrelation time (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   21.84   23.19   31.43   32.76   41.07   45.93

summary(1 - rejectionRate(fit_MCMC)) # Acceptance rate (beta)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.4451  0.4472  0.4479  0.4483  0.4494  0.4517
```

# Summary of the results

- The following table compare the **average** results. Here, ESS represents the estimated and average **effective sample size**.
- Among these competitors, the Laplace MH seems preferable.
- Note that we could sensibly speed up these results by using **Rcpp**!

| | Seconds | ESS | ESS / Sec. | Acceptance rate |
|---|---|---|---|---|
| Vanilla MH | 1.89 | 259.58 | 137.60 | 0.72 |
| Laplace MH | 1.77 | 1194.42 | 676.49 | 0.27 |
| AM | 4.88 | 1110.90 | 227.45 | 0.19 |
| Metropolis-within-Gibbs | 11.95 | 37.57 | 3.14 | 0.97 |
| Ad. Metropolis-within-Gibbs | 11.95 | 1009.32 | 84.48 | 0.45 |