Metropolis-Hastings

Sergio P.E.

11/9/2021

Algoritmo de Metropolis-Hastings

- El algoritmo de Metropolis-Hastings (MH) es una estrategia simple de muestreo *a posteriori*
- Iniciar la cadena en un valor razonable $\boldsymbol{\theta}_1$

en el r-ésimo valor de la cadena

- Sea $\boldsymbol{\theta} = \boldsymbol{\theta}^{(r)}$ el estado actual de la cadena
- Muestrear $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ de una distribución propuesta $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$.
- Calcular la probabilidad de aceptación, definida como

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

- Con probabilidad α, actualizar el estado de la cadena y hacer $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$.

MH con caminata aleatoria

- Considérese el algoritmo Metrolis-Hastings con caminata aleatoria (RWM) y sea $\boldsymbol{\theta} = \boldsymbol{\theta}^r$ el estado actual de la cadena
- Se llama "caminata aleatoria" porque se muestrea $\boldsymbol{\theta}^*$ de una Propuesta gussiana

$$q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}) \sim \mathrm{N}_p\left(\boldsymbol{\theta}^{(r)}, \mathbf{S}\right), \quad \text{lo que implica que} \quad q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) = q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)$$

- En este caso particular la probabilidad de aceptación se simplifia

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

- Esta distribución propuesta gaussiana es una elección sensata, especialmente cuando el soporte de $\boldsymbol{\theta}$ no está acotado
- A pesar de esta importante simplificación, la elección de la matriz de covarianza $\mathbf{S}$ sigue siendo una tarea difícil y afecta de manera crucial el rendimiento.
- Pregunta clave. ¿Podemos identificar una matriz de covarianza ideal $\mathbf{S}$ que sea óptima en alguno sentido? ¿Podemos "estimarla" a partir de los datos?

Varianza asintótica

- Para una función de cuadrado integrable $g(\cdot) : \mathbb{R}^p \to \mathbb{R}$ considérese el estimador Monte Carlo

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

de la esperanza *a posteriori* $\mu_g = \mathbb{E}\{g(\boldsymbol{\theta}) \mid \mathbf{X}\}$.

- Si ocurre el TCL, se tiene que
$$\sqrt{R}\frac{\hat{\mu}_g - \mu_g}{\sigma_g} \xrightarrow{d} \mathrm{N}(0,1),$$
donde σ_g^2 es la varianza asintótica del algoritmo MCMC.
- Intuitivamente, se busca una matrix de covarianza $\mathbf{S}$ que minimiza la varianza asintótica σ_g^2.
- Sea $\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots$ una cadena de Markov, con $\boldsymbol{\theta}^{(0)} \sim \pi(\boldsymbol{\theta} \mid \mathbf{X})$ siendo una muestra de la distribución estacionaria.
- La varianza asintótica puede escribirse como
$$\sigma_g^2 = \mathrm{var}\left\{g(\boldsymbol{\theta}^{(0)|\mathbf{X}})\right\}\tau_g = \mathrm{var}\left\{g(\boldsymbol{\theta}^{(0)|\mathbf{X}})\right\}\left[1 + 2\sum_{r=1}^{\infty}\mathrm{Corr}\left\{g(\boldsymbol{\theta})^{(0)}, g(\boldsymbol{\theta})^{(r)}\right\}\right]$$
- La cantidad τ_g es llamada tiempo de autocorrelación integrado (IAT) y mide la pérdida de eficiencia con respecto al muestreo *iid* ($\tau_g = 1$)
- Cuando $\tau_g = 1$, el algoritmo MCMC es "óptimo" y no hay autocorrelación.
- El comando **R** `effectiveSize` produce un estimador de $R\tau_g^{-1}$ con las muestras empíricas de la cadena.

Escalado óptimo

- Asúmase inicialmente que la distribución a posteriori es de la forma
$$\pi(\boldsymbol{\theta} \mid \mathbf{X}) = \prod_{j=1}^{p} f(\theta_j), \quad \mathrm{var}\,(\boldsymbol{\theta} \mid \mathbf{X}) = \sigma^2 I_p$$
lo que significa que los componentes de $\boldsymbol{\theta}$ son independientes e idénticamente distribuidos con alguna función de densidad f.
- Además, considérese la distrbución propuesta
$$\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}\right) \sim \mathrm{N}\left(\boldsymbol{\theta}^{(r)}, s_p^2 I_p\right), \quad s_p^2 = l^2/p$$
- En este contexto simplificado, se busca el valor del parámetro l^2 para obtener el escalado óptimo.
- Sea θ_1 el primer componente de $\boldsymbol{\theta}$. El valor óptimo de escalado es
$$l_{opt} \approx \frac{2.38}{\mathcal{I}^{1/2}}$$
donde
$$\mathcal{I} = \mathbb{E}\left[\left\{\frac{f'(\theta_1)}{f(\theta_1)}\right\}^2\right]$$
es la matriz de información de Fisher.
- La tasa de aceptación, evaluada en l_{opt}, es
$$A(l_{opt}) = 2\Phi\left(-\frac{\mathcal{I}^{1/2} l_{opt}}{2}\right) = 0.234,$$
lo que corresponde a la tasa de aceptación óptima.
- Lo anterior sugiere que la propuesta es $\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(r)}\right) \sim \mathrm{N}\left(\boldsymbol{\theta}^{(r)}, s_p^2 I_p\right)$, para valores grandes de p, con
$$s_p = 2.38^2 (p\mathcal{I})^{-1},$$
donde $\mathcal{I}$ debe ser estimado. Si f es una densidad gaussiana con varianza σ^2, entonces $s_p = 2.38^2\sigma^2 p^{-1}$.
- Estos resultados son asintóticos y requiren que los componentes de $\boldsymbol{\theta}$ sean independientes.
- En la práctica, con $p \approx 5$ la tasa de aceptación óptima es cercana a 0.234.

- Cuando $p = 1$ la tasa de aceptación es aproximadamente 0.44
- Extensión. Si la distribución *a posteriori* es normal con matriz de covarianza $p \times p$ Σ,

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

- El procedimiento es óptimo con p grande, aunque requiere el conocimiento de Σ.

Detalles computacionales

- Recordar que en cada paso del algoritmo es necesario generar una muestra de la distribución normal $\mathrm{N}(0, \mathbf{S})$.
- El uso de funciones integradas en **R** como `rmvnorm` y `mvrnorm` conduce a un aumento considerable del tiempo de computación.
- Para obtener una muestra de $V \sim \mathrm{N}(\mu, \mathbf{S})$ solo es necesario calcular

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

  donde $\mathbf{Z} \sim \mathrm{N}(0, I_p)$ normal estándar y $\mathbf{A}$ es una matrix $p \times p$ tal que $\mathbf{A}\mathbf{A}^T = \mathbf{S}$.
- Entonces, no es necesario calcular $\mathbf{A}$ en cada paso, y esto puede hacerse antes de correr el MCMC.

Regresión binaria

- Sea $\mathbf{y} = (y_1, \ldots, y_n)^T$ un vector respuestas binarias observadas
- Sea $\mathbf{X}$ la matriz diseño correspondiente cuya fila genérica es $\mathbf{x}_i = (1, x_{i2}, \ldots, x_{ip})^T$ para $i = 1, \ldots, n$. Suponer que todos los predictores están estandarizados.
- Considerar el GLM tal que

$$(y_i \mid \pi_i) \overset{ind}{\sim} \mathbf{Bern}(\pi_i), \quad \pi_i = g(\eta_i), \quad \eta_i = \beta_1 x_{xi1} + \ldots + \beta_p x_{ip},$$

  donde $g(\cdot)$ es la inversa de la transformación logit o la cdf de la normal estándar. En lo que sigue trataremos el caso de regresión logística.
- Se estimará el parámetro vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ usando RWM.
- Se asigna una *a priori* vaga centrada en 0, $\boldsymbol{\beta} \sim \mathrm{N}_p(0, 100 I_p)$.

Datos PIMA

?MASS::Pima.tr

Diabetes in Pima Indian Women

Description

A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to World Health Organization criteria. The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases. We used the 532 complete records after dropping the (mainly missing) data on serum insulin.

```
Pima <- rbind(MASS::Pima.tr, MASS::Pima.te)
y <- as.numeric(Pima$type == "Yes") # Respuesta binaria
X <- model.matrix(type ~ . - 1, data = Pima) # Matriz diseño
X <- cbind(1, scale(X)) # Estandarizar las covariables y agregar el intercepto
```

```
# Modelo Logit (MV)
fit_logit <- glm(type ~ X - 1, family = binomial(link = "logit"), data = Pima)
summary(fit_logit)
```

```
##
## Call:
## glm(formula = type ~ X - 1, family = binomial(link = "logit"),
##     data = Pima)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0100  -0.6613  -0.3692   0.6433   2.4795
##
## Coefficients:
##        Estimate Std. Error z value Pr(>|z|)
## X      -0.99003    0.12276  -8.065 7.35e-16 ***
## Xnpreg  0.40578    0.14488   2.801 0.005097 **
## Xglu    1.09493    0.13157   8.322  < 2e-16 ***
## Xbp    -0.09473    0.12696  -0.746 0.455602
## Xskin   0.07129    0.15533   0.459 0.646242
## Xbmi    0.56892    0.16057   3.543 0.000395 ***
## Xped    0.45091    0.12543   3.595 0.000324 ***
## Xage    0.28383    0.15066   1.884 0.059581 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 737.51  on 532  degrees of freedom
## Residual deviance: 466.32  on 524  degrees of freedom
## AIC: 482.32
##
## Number of Fisher Scoring iterations: 5
```

Log-Verosimiltud

En primer lugar se codifica la log-verosimilitud y la log *a posteriori*

```
# Log-verosimlitud del modelo de regresión logística
loglik <- function(beta, y, X) {
  eta <- c(X %*% beta)
  sum(y * eta - log(1 + exp(eta)))
}

# Log-a posteriori
logpost <- function(beta, y, X) {
  loglik(beta, y, X) + sum(dnorm(beta, 0, 10, log = T))
}
```

MH con caminata aleatoria (MRW)

```
# R:= número de muestras
# burn_in:= número de muestras elimindas
# S:=Matriz de covarianzas de la propuesta normal
RMH <- function(R, burn_in, y, X, S) {
  p <- ncol(X)
  out <- matrix(0, R, p) # Inicializar una matriz vacia para almacer la salida
```

```
  beta <- rep(0, p) # Valores iniciales
  logp <- logpost(beta, y, X)

  # Eigen-descomposición (puede ser Cholesky)
  eig <- eigen(S, symmetric = TRUE)
  A1 <- t(eig$vectors) * sqrt(eig$values)

  # Inicio del Gibbs sampler
  for (r in 1:(burn_in + R)) {
    beta_new <- beta + c(matrix(rnorm(p), 1, p) %*% A1)
    logp_new <- logpost(beta_new, y, X)
    alpha <- min(1, exp(logp_new - logp))
    if (runif(1) < alpha) {
      logp <- logp_new
      beta <- beta_new # Aceptar el valor
    }
    # Almacenar los valores después del periodo burn-in
    if (r > burn_in) {
      out[r - burn_in, ] <- beta
    }
  }
  out
}
```

MH preliminar

Para ejecutar el algoritmo, es ncesario especificar la matriz de covarianza $\mathbf{S}$ de la distribución propuesta. Como primera aproximación se prueba $\mathbf{S} = \texttt{diag}(10^{-3}, \ldots, 10^{-3})$. Esta elección conduce a una cadena altamente autocorrelacionada. Se exploran especificaciones alternativas para $\mathbf{S}$. más adelante.

```
library(coda)
R <- 30000 # Numero de muestras retenidas
burn_in <- 30000 # Burn-in
set.seed(123)

# Matriz de covarianzas de la propuesta
S <- diag(1e-3, ncol(X))

# Correr el MCMC
start.time <- Sys.time()
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S)) # Convertir la matriz en un objeto "coda"
end.time <- Sys.time()
time_in_sec <- as.numeric(end.time - start.time)
time_in_sec
```

```
## [1] 3.632289
```

```
# Diagnóstico
summary(effectiveSize(fit_MCMC)) # Tamaño de muestra efectivo
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   174.9   205.0   258.5   259.6   320.7   333.1
```

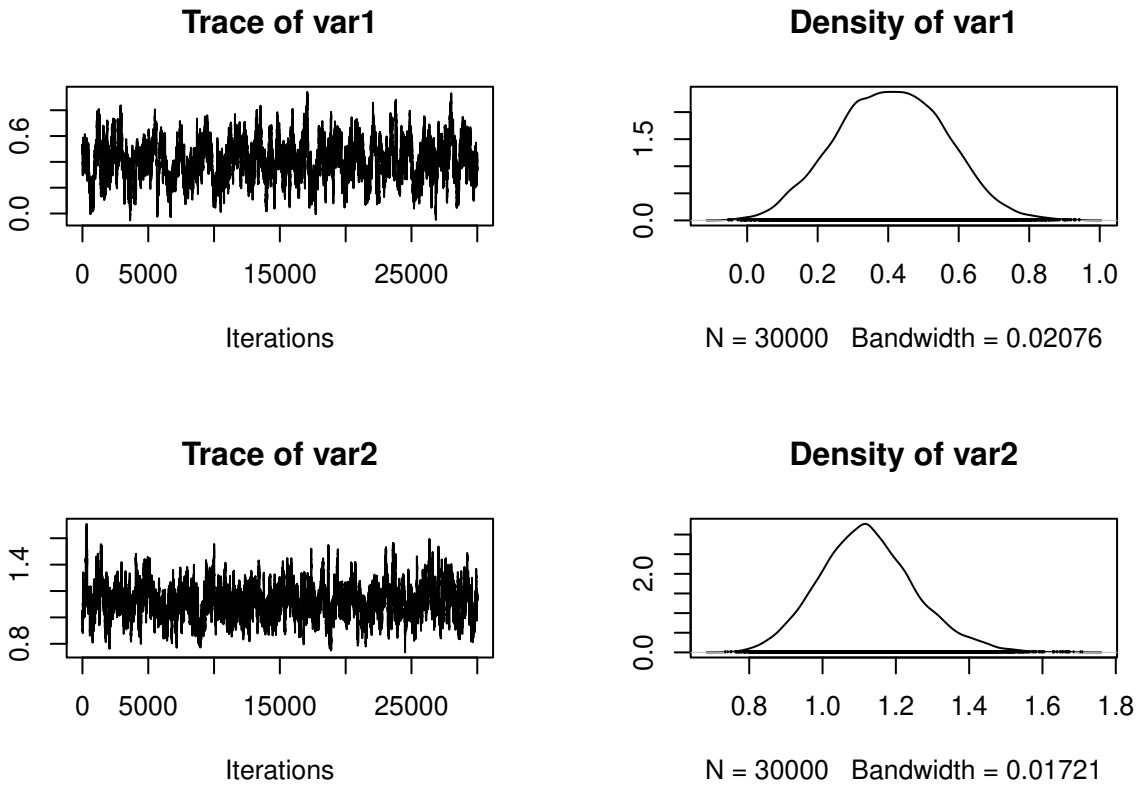
```
summary(R / effectiveSize(fit_MCMC)) # Tiempo de autocorrelación integrado
```

```
##    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)) # Tasa de aceptación
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7191  0.7191  0.7191  0.7191  0.7191  0.7191
```

```
# Traza de 3 parámetros
plot(fit_MCMC[, 2:3])
```



Aproximación de la matriz de covarianzas a posteriori

- Se sabe que una selección apropiada de $\mathbf{S}$ debe estar basada en la matriz de covarianzas *a postetiori* Σ.
- La verdadera Σ es desconocida y debe utilizarse una aproximación
- Una posibilidad está basada en una aproximación cudrática de la función de verosimilitud evaluada en el EMV
- En el caso de regresión logística

$$\hat{\Sigma} = \left(\mathbf{X}^T \hat{H} \mathbf{X}\right)^{-1}, \quad \hat{H} = \texttt{diag}\left\{\hat{\pi}_1(1-\hat{\pi}_1), \ldots, \hat{\pi}_n(1-\hat{\pi}_n)\right\},$$

  donde $\hat{\pi}_i = \left[1 + \exp\left\{-(x_{i1}\hat{\beta}_1 + \cdots + x_{ip}\hat{\beta}_p)\right\}\right]^{-1}$
- Esta estimación corresponde a la inversa de la información de Fisher evaluada en el EMV.
- Es una variación de la aproximación de Laplace que ignora la información *a priori*.
- Una matriz de covarianza para la propuesta basada en la aproximación de Laplace es $\mathbf{S} = 2.38^2\hat{\Sigma}/p$.
- Esta selección es óptima y el tamaño de muestra efectivo correspondiente es más grande.
- En regresión logística $\hat{\Sigma}$ se puede obtener con la función `vcov` de `R`.

```r
set.seed(123)

# Correr el MCMC
start.time <- Sys.time()

# Se utiliza la aproximación de Laplace de la matriz de covarianzas
fit_logit <- glm(type ~ X - 1, family = binomial(link = "logit"), data = Pima)
p <- ncol(X)
S <- 2.38^2 * vcov(fit_logit) / p
# MCMC
fit_MCMC <- as.mcmc(RMH(R, burn_in, y, X, S)) # Convertit la matriz en un objeto coda
end.time <- Sys.time()

time_in_sec <- as.numeric(end.time - start.time)
time_in_sec
```

```
## [1] 3.583909
```

```r
# Diagnóstico
summary(effectiveSize(fit_MCMC)) # Tamaño de muestra efectivo
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1107    1174    1206    1194    1228    1245
```

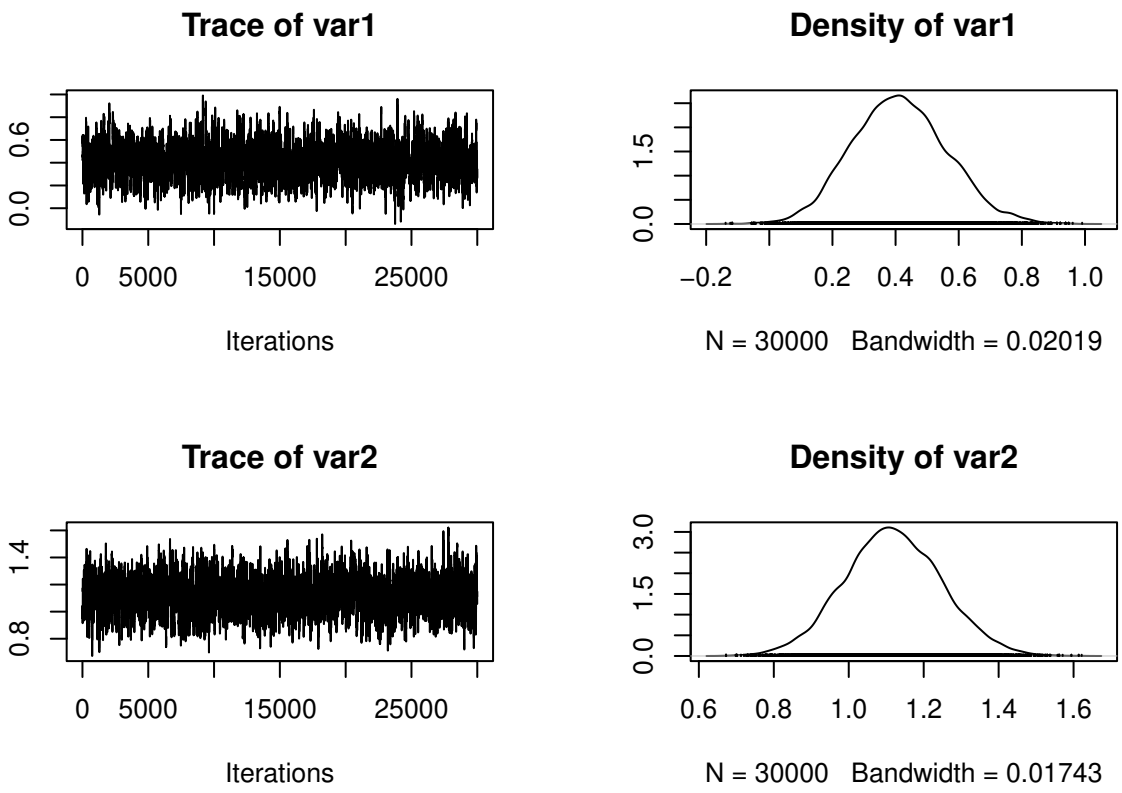
```r
summary(R / effectiveSize(fit_MCMC)) # Tiempo de autocorrelación integrado
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   24.10   24.43   24.87   25.15   25.56   27.10
```

```r
summary(1 - rejectionRate(fit_MCMC)) # Tasa de aceptación
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2746  0.2746  0.2746  0.2746  0.2746  0.2746
```

```r
# Trazas
plot(fit_MCMC[, 2:3])
```



MCMC adaptativo

- En varios casos, no es posible realizar una estimación $\hat{\Sigma}$ rápida y razonable.
- Una posibilidad es ajustar una matriz $\mathbf{S}$ en cada iteración utilizando las muestras obtenidas previamente.
- Una condición para preservar la ergodicidad de la cadena es que los cambios en $\mathbf{S}$ sean insignificantes cuando $R \to \infty$.
- Una versión del MH adaptativo hace uso de la propuesta siguiente:

$$q_r(\boldsymbol{\beta}^* \mid \boldsymbol{\beta}) = \mathrm{N}(\boldsymbol{\beta}, 2.38^2/p\Sigma_r + \epsilon I_p),$$

  donde Σ_r es la matriz de covarianza de las r muestras previas $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(r)}$.
- La constante $\epsilon > 0$ es una valor pequeño para evitar que la matriz de covarianzas sea singular.
- Ocurre la formula recursiva siguiente:

$$\Sigma_r = \frac{1}{r-1}\sum_{j=1}^{r}\left(\boldsymbol{\beta}^{(j)} - \bar{\boldsymbol{\beta}}^{(r)}\right)\left(\boldsymbol{\beta}^{(j)} - \bar{\boldsymbol{\beta}}^{(r)}\right)^T = \frac{r-2}{r-1}\Sigma_{r-1} + \frac{1}{r}\left(\boldsymbol{\beta}^{(r)} - \bar{\boldsymbol{\beta}}^{(r-1)}\right)\left(\boldsymbol{\beta}^{(r)} - \bar{\boldsymbol{\beta}}^{(r-1)}\right)^T$$

  donde $\bar{\boldsymbol{\beta}}^{(r)} = (r-1)/r\bar{\boldsymbol{\beta}}^{(r-1)} + \boldsymbol{\beta}^{(r)}/r$ es la media aritmetica de los primeros r valores.
- Se obtienen resultados similares al MH basado en la aproximación de Laplace en términos de tamaño de muestra efectivo.
- No obstante, el tiempo de computo es más grande porque es necesario calcular $\mathbf{S}$ en cada iteración.

Código para el MH adaptativo

```
# R:= número de muestras
# burn_in:= número de muestras elimindas
# S:=Matriz de covarianzas de la propuesta normal
```

```r
RMH_Adaptive <- function(R, burn_in, y, X) {
  p <- ncol(X)
  out <- matrix(0, R, p) # Iniciar una matriz vacía para almacenar los valores
  beta <- rep(0, p) # Valores iniciales
  logp <- logpost(beta, y, X)
  epsilon <- 1e-6

  # Initial matrix S
  S <- diag(epsilon, p)
  Sigma_r <- diag(0, p)
  mu_r <- beta

  for (r in 1:(burn_in + R)) {

    # Actualización de la matriz de covarianzas
    if(r > 1){
      Sigma_r <- (r - 2) / (r - 1) * Sigma_r + tcrossprod(beta - mu_r) / r
      mu_r <- (r - 1) / r * mu_r + beta / r
      S <- 2.38^2 * Sigma_r / p + diag(epsilon, p)
    }

    # Eigen-descomposición
    eig <- eigen(S, symmetric = TRUE)
    A1 <- t(eig$vectors) * sqrt(eig$values)

    beta_new <- beta + c(matrix(rnorm(p), 1, p) %*% A1)
    logp_new <- logpost(beta_new, y, X)
    alpha <- min(1, exp(logp_new - logp))
    if (runif(1) < alpha) {
      logp <- logp_new
      beta <- beta_new # Aceptar valor
    }

    # Guardar valores después de perido burn-in
    if (r > burn_in) {
      out[r - burn_in, ] <- beta
    }
  }
  out
}
```

```r
set.seed(123)

# Correr el MCMC
start.time <- Sys.time()
fit_MCMC <- as.mcmc(RMH_Adaptive(R = R, burn_in = burn_in, y, X))
end.time <- Sys.time()

time_in_sec <- as.numeric(end.time - start.time)
time_in_sec
```

```
## [1] 9.175268
```

```r
# Diagnóstico
summary(effectiveSize(fit_MCMC)) # Tamaño de muestra efectivo
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   856.7   905.9  1124.5  1110.9  1269.2  1412.6
```

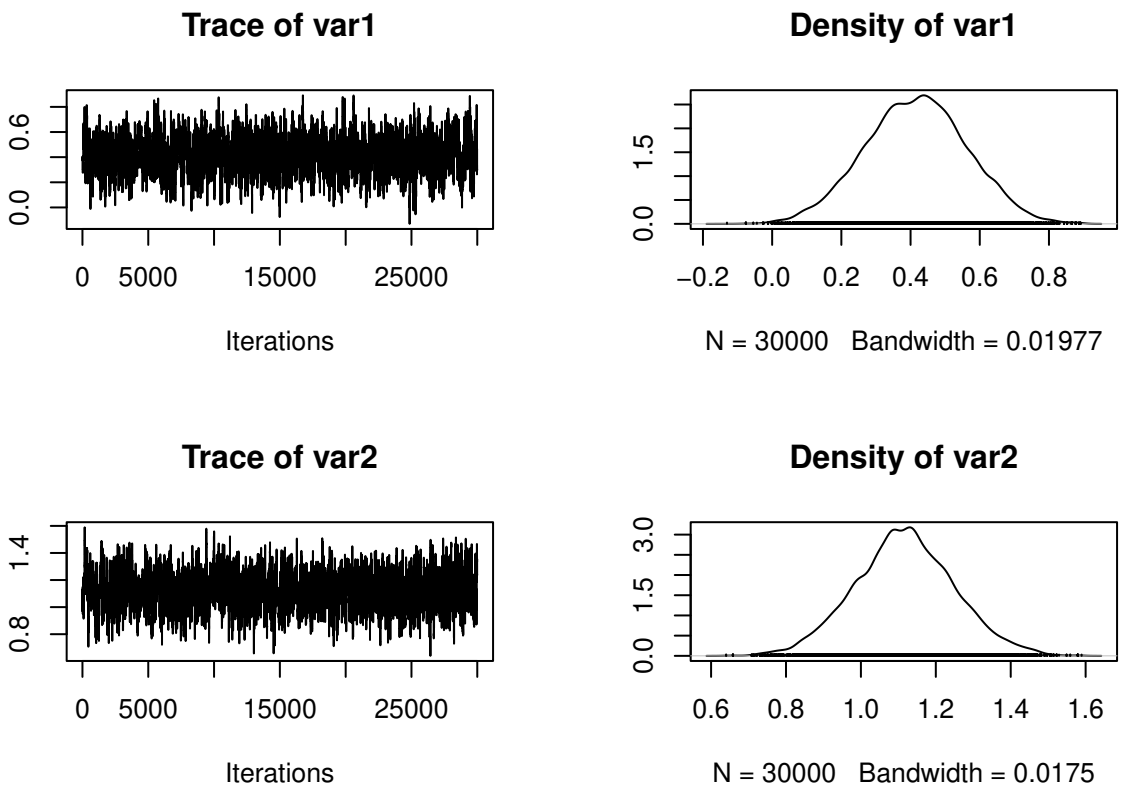
```r
summary(R / effectiveSize(fit_MCMC)) # Tiempo de autocorrelación integrado
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.24   23.65   26.69   27.89   33.12   35.02
```

```r
summary(1 - rejectionRate(fit_MCMC)) # Tasa de aceptación
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1907  0.1907  0.1907  0.1907  0.1907  0.1907
```

```r
# Trazas
plot(fit_MCMC[, 2:3])
```




```r
#knitr::purl("MHbinreg.Rmd")
```