Métodos Markov Chain Monte Carlo

Víctor Morales Oñate
Sitio personal
ResearchGate
GitHub
LinkedIn

09 de mayo de 2019

Contents

Paquetes de esta sección

```
if(!require(ISLR)){install.packages("LearnBayes")}
```

- Vamos a ilustrar el uso de algoritmos MCMC para resumir distribuciones posteriores.
- También veremos dos variantes de MCMC: Metropolis-Hastings y Gibbs *sampling*, donde la cadena de Markov se configura a través de la distribución condicional de la posterior.

Cadena de markov discreta

Supongamos que una persona se mueve en la línea recta de valores 1,2,3,4,5 y 6. Si la persona está en un punto interior (2,3,4,5), en el siguiente segundo puede quedarse en ese punto o moverse a *algún* punto vecino. Si decide moverse, hay igual probabilidad de que se mueva a la izquierda o a la derecha. Si la persona está en uno de los extremos (1 o 6), al siguiente segundo puede, con igual probabilidad, quedarse o moverse *al* punto vecino.

Lo anterior es un ejemplo de una cadena de Markov discreta. **Una cadena de Markov describe el movimiento probabilístico entre un número de estados**.

En el ejemplo hay 6 posibles estados que describen la posición del caminante. Dado que están en una posición, se mueve a otro punto con una determinada probabilidad. **La probabilidad de que se mueva a otra posición depende solamente de su posición actual y no de las anteriores.**

Describimos los movimientos entre estados en términos de probabilidades de transición. Resumimos las probabilidades de transición con una **matriz de transición**:

$$P = \begin{bmatrix} 0.50 & 0.50 & 0 & 0 & 0 \\ 0.25 & 0.50 & 0.25 & 0 & 0 \\ 0 & 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

Esta matriz de transición tiene algunas propiedades interesantes, es:

- Irreducible: es posible ir desde cualquier estado hasta cualquier otro en uno o más pasos,
- Periódica: Si una persona puede únicamente regresar a este estado en intervalos regulares dado que una persona está en un estado particular. Nuestra matriz es *aperiódica*.

Podemos representar la posición actual como un vector de la forma:

$$p = (p_1, p_2, p_3, p_4, p_5, p_6),$$

donde p_i representa la probabilidad de que una persona está actualmente en el estado i.

Si p_i^j representa la posición del viajero en el paso j, entonces la posición del viajero en el paso $j+1$ se da por el producto matricial

$$p^{j+1} = p^j P.$$

Supongamos que podemos encontrar un vector w tal que $wP = w$. Entonces se dice que w es la **distribución estacionaria**.

Si una cadena de markov es estacionaria y aperiódica, entonces tiene una única distribución estacionaria.

En R

Hacemos el siguiente experimento. Iniciamos nuestra caminata aleatoria en un estado particular, digamos 3, y luego simulamos muchos pasos de la cadena de Markov usando la matriz de transición P.

La frecuencia relativa de nuestro viajero en las seis posiciones después de muchos pasos eventualmente se acerca a la distribución w.

```
P <- matrix(c(.5,.5,0,0,0,0,.25,.5,.25,0,0,0,0,.25,.5,.25,0,0,
 0,0,.25,.5,.25,0,0,0,0,.25,.5,.25,0,0,0,0,.5,.5),
 nrow=6,ncol=6,byrow=TRUE)
P
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 0.50 0.50 0.00 0.00 0.00 0.00
## [2,] 0.25 0.50 0.25 0.00 0.00 0.00
## [3,] 0.00 0.25 0.50 0.25 0.00 0.00
## [4,] 0.00 0.00 0.25 0.50 0.25 0.00
## [5,] 0.00 0.00 0.00 0.25 0.50 0.25
## [6,] 0.00 0.00 0.00 0.00 0.50 0.50
```

Indicamos que la ubicación de inicio para nuestro viajero es el estado 3 y realizamos un bucle para simular 50000 realizaciones de la cadena de Markov

```
s <- array(0,c(50000,1))
```

Simulamos:

```
set.seed(1)
s[1]=3
for (j in 2:50000)
  s[j]=sample(1:6,size=1,prob=P[s[j-1],])
```

Resumimos las frecuencias de visitas en diferentes puntos de corte y vemos la precuencia relativa:

```
m=c(500,2000,8000,50000)
for (i in 1:4)
  print(table(s[1:m[i]])/m[i])
```

```
##
##     1     2     3     4     5     6
## 0.094 0.170 0.188 0.198 0.262 0.088
##
##      1      2      3      4      5      6
## 0.0965 0.1900 0.2065 0.2110 0.2055 0.0905
##
##       1       2       3       4       5       6
## 0.09050 0.19275 0.21100 0.21350 0.20000 0.09225
##
##       1       2       3       4       5       6
## 0.10212 0.20108 0.19846 0.20088 0.19884 0.09862
```

Parece que las frecuencias relativas están convergiendo a la distribución estacionaria $w = (0.1, 0.2, 0.2, 0.2, 0.2, 0.1)$. Podemos confirmar que efectivamente converge con $wP = w$:

```
w <- matrix(c(.1,.2,.2,.2,.2,.1),nrow=1,ncol=6)
w%*%P
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  0.1  0.2  0.2  0.2  0.2  0.1
```

El algoritmo Metropolis-Hastings.

- Una forma popular de simular a partir de una distribución posterior general es mediante el uso de los métodos de la cadena de Markov Monte Carlo (MCMC).
- Esto es esencialmente una generalización de valores continuos de la configuración de la cadena de Markov discreta descrita en la sección anterior.
- La estrategia de muestreo MCMC establece una cadena de Markov aperiódica e irreducible para la cual la distribución estacionaria es igual a la distribución posterior de interés.
- Una forma general de construir una cadena de Markov es mediante el uso de un algoritmo Metropolis-Hastings. En esta sección, nos centramos en dos variantes particulares de los algoritmos de Metropolis-Hastings, **la cadena de independencia** y la cadena de **caminata aleatoria**, que son aplicables a una amplia variedad de problemas de inferencia bayesianos.

Supongamos que queremos muestrear desde una densidad posterior $g(\theta|y)$ (que también notaremos como $g(\theta)$.

El almoritmo Metropolis-Hastings empieza con un valor inicial θ^0 y especifica una regla para simular el valor tth en la secuencia θ^t dado el valor $(t-1)$ en la secuencia θ^{t-1}.

Esta regla se llama **densidad propuesta** la cual simula un valor candidato θ^*, y el cálculo de una **probabilidad de aceptación** P. La cual indica la probabilidad de que un valor candidato sea aceptado como siguiente valor en la secuencia.

Específicamente, este algoritmo se describe:

1. Simula el valor candidato θ^* desde una densidad propuesta $p(\theta^*|\theta^{t-1})$.
2. Calcula el ratio

$$R = \frac{g(\theta^*)p(\theta^{t-1}|\theta^*)}{g(\theta^{t-1})p(\theta^*|\theta^{t-1})}$$

3. Calcula la probabilidad de aceptación $P = min\{R, 1\}$ 4. Muestrea un valor θ^t tal que $\theta^t = \theta^*$ con probabilidad P; en otro caso $\theta^t = \theta^{t-1}$.

La secuencia simulada $\theta^1, \theta^2, \ldots$ va a converger a una variable aleatoria que se distribuye como la distribución posterior $g(\theta)$.

Se configuran variantes del Metropolitan-Hastings dependiento de la función de densidad propuesta. Si la función de densidad propuesta es independiente del valor actual en la secuencia,

$$p(\theta^*|\theta^{t-1}) = p(\theta)$$

entonces el algoritmo resultante se llama *cadena indenpendiente*.

Otra opción es permitiendo que la densidad tenga la forma

$$p(\theta^*|\theta^{t-1}) = h(\theta^* - \theta^{t-1}),$$

donde h es una densidad simétrica respecto al origen. En este tipo de cadena de *caminata aleatoria*, el ratio R tiene la forma simple:

$$R = \frac{g(\theta^*)}{g(\theta^{t-1})}$$

Las funciones `rwmetrop` y `indepmetrop` del paquete `LearnBayes` las implementan.

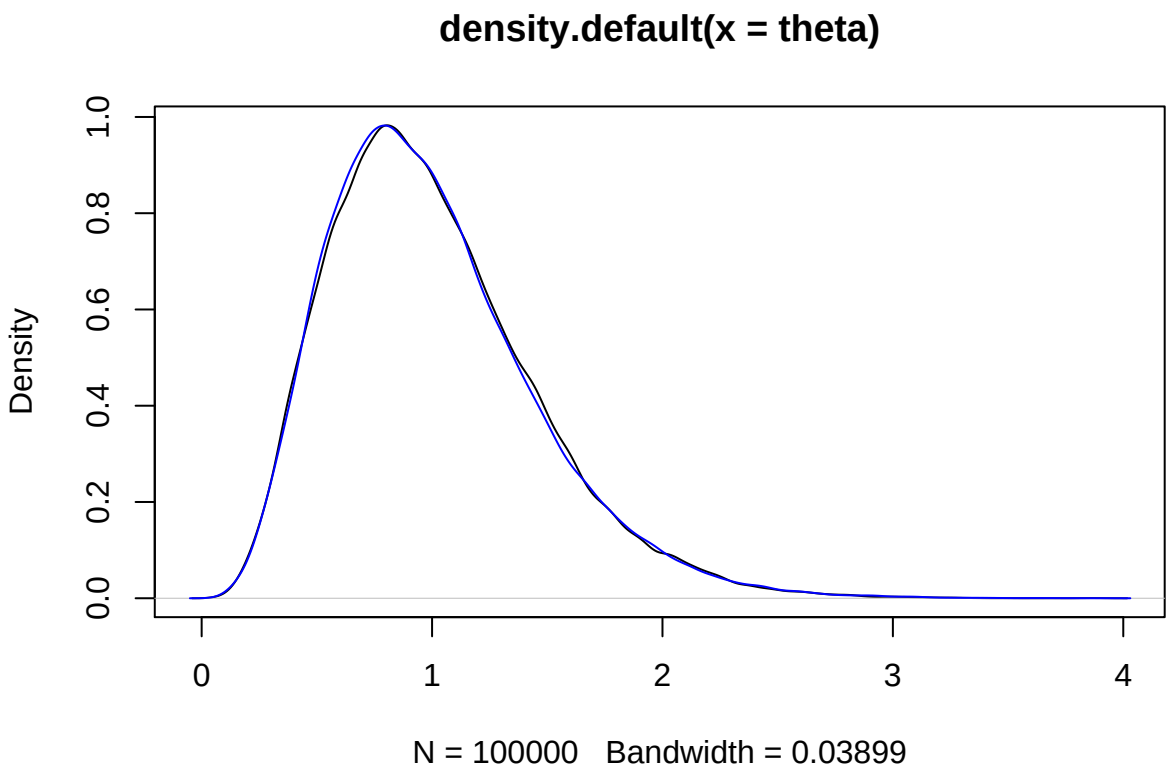
```
#Definimos los parametros:
nreps <- 100000
prop_sd <- 1
dens <- function(x){dgamma(x,shape = 5, rate = 5)}
start <- 2

theta <- numeric(nreps)
theta[1] <- start

for (i in 2:nreps)
{
  # i = 2
  theta_star <- rnorm(1, mean = theta[i - 1], sd = prop_sd)
  alpha <- dens(theta_star)/dens(theta[i - 1])


  if(runif(1) < alpha){
    theta[i] <- theta_star
  }else
  {
      theta[i] <- theta[i - 1]
  }
}

plot(density(theta))
lines(density(rgamma(nreps, 5,5)), col = "blue")
```



Muestreo de Gibbs

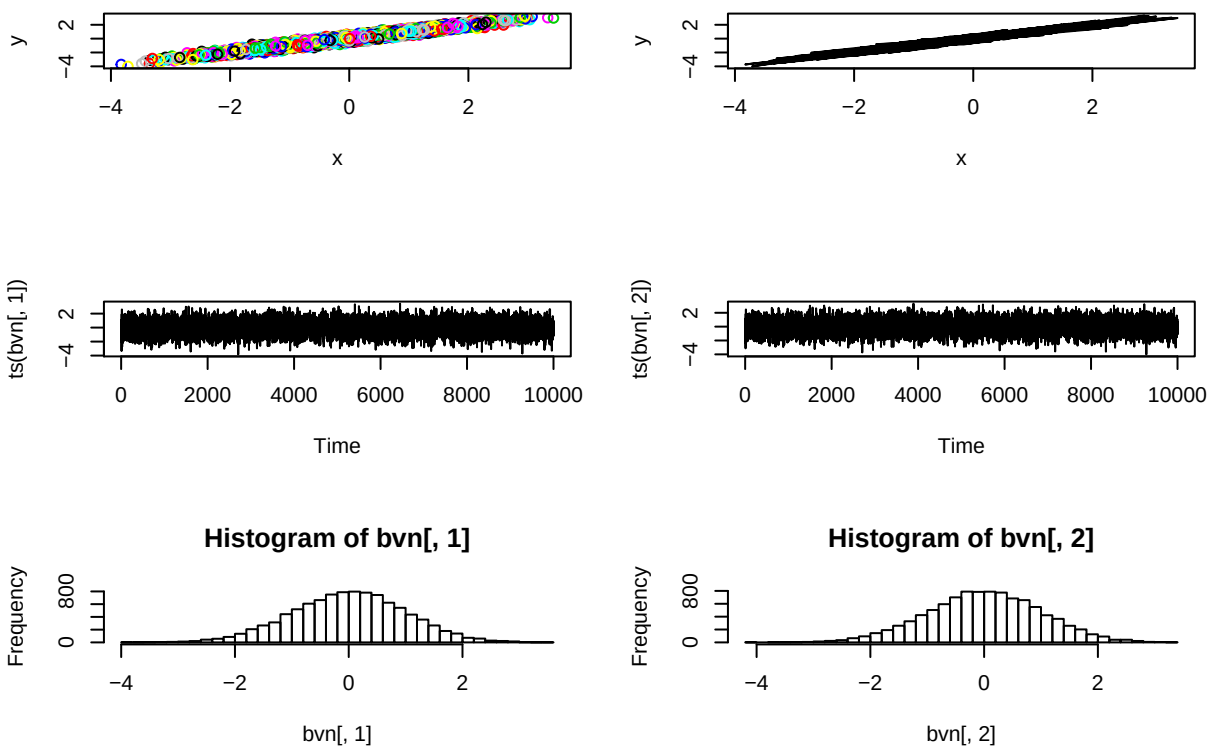
Veamos la simulación a partir de una normal bivariada con una media cero y una varianza 1 para las marginales, pero una correlación de *rho* entre los dos componentes.

Por supuesto, no necesitamos una muestra de Gibbs para simular esto, simplemente podríamos simular desde el marginal para X, y luego desde el condicional para $Y|X$. En R, podríamos hacer esto de la siguiente manera:

```
rbvn<-function (n, rho)
{
        x <- rnorm(n, 0, 1)
        y <- rnorm(n, rho * x, sqrt(1 - rho^2))
        cbind(x, y)
}
```

Esto crea un vector de valores X, luego los usa para construir vectores de valores Y condicionales en esos valores X. Estos se unen entonces en una matriz $n \times 2$. Podemos probarlo con:

```
bvn<-rbvn(10000,0.98)
par(mfrow=c(3,2))
plot(bvn,col=1:10000)
plot(bvn,type="l")
plot(ts(bvn[,1]))
plot(ts(bvn[,2]))
hist(bvn[,1],40)
hist(bvn[,2],40)
```




```
par(mfrow=c(1,1))
```

Esto proporciona un par de gráficos de dispersión de los puntos, gráficos de series de tiempo de las marginales para confirmar que estamos muestreando de forma independiente, y luego histogramas de las dos marginales.

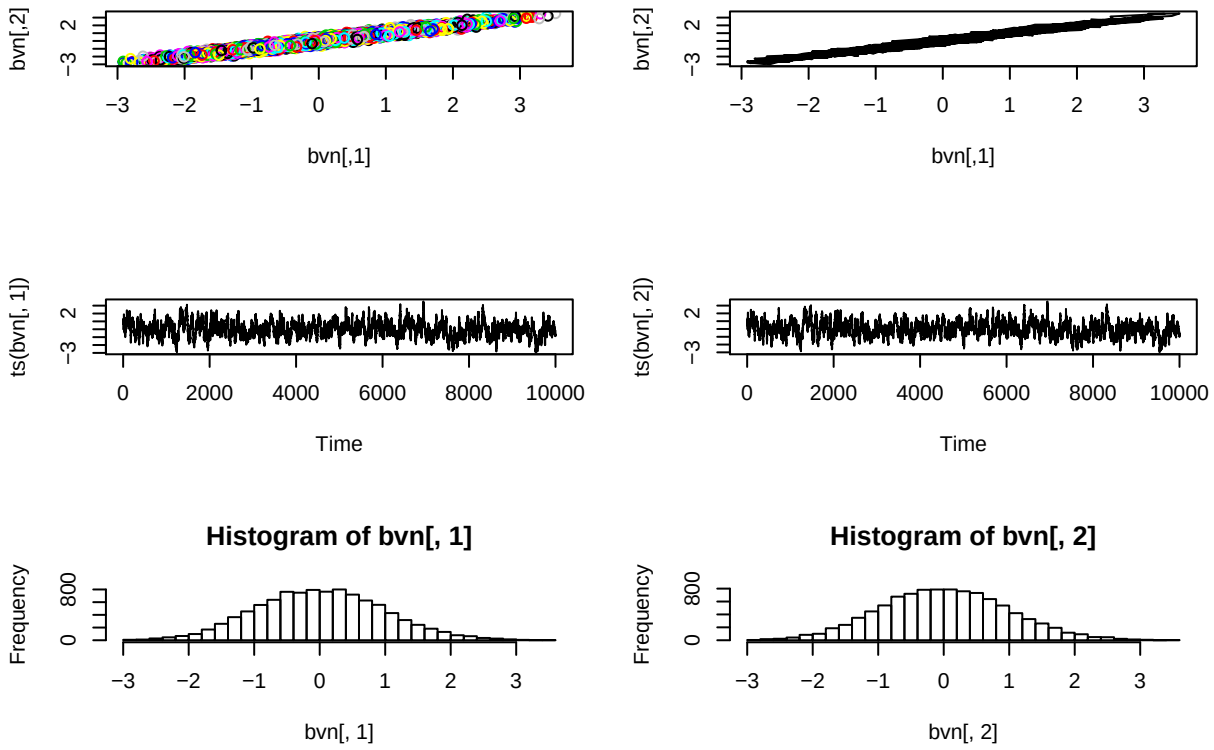
Sin embargo, también podemos hacerlo usando muestreo de Gibbs. Podemos construir una muestra de Gibbs para este problema muestreando sucesivamente las distribuciones condicionales.

```
gibbs<-function (n, rho)
{
        mat <- matrix(ncol = 2, nrow = n)
        x <- 0
        y <- 0
        mat[1, ] <- c(x, y)
        for (i in 2:n) {
                x <- rnorm(1, rho * y, sqrt(1 - rho^2))
                y <- rnorm(1, rho * x, sqrt(1 - rho^2))
                mat[i, ] <- c(x, y)
        }
        mat
}
```

Se crea una matriz para los resultados, luego la cadena se inicializa en $(0, 0)$. El bucle principal luego toma muestras sucesivas de los condicionales completos, almacenando los resultados en la matriz. Podemos probar esto de la siguiente manera.

```
bvn<-gibbs(10000,0.98)
par(mfrow=c(3,2))
plot(bvn,col=1:10000)
plot(bvn,type="l")
plot(ts(bvn[,1]))
plot(ts(bvn[,2]))
```

```
hist(bvn[,1],40)
hist(bvn[,2],40)
```




```
par(mfrow=c(1,1))
```

Con un poco de suerte, esto dará resultados que se verán muy similares a los obtenidos anteriormente, además de las gráficas de series de tiempo de los marginales, que muestran una autocorrelación distinta entre los valores sucesivos.

En Bayesiano...

Supongamos que $Y \sim N(mean = \mu, Var = \frac{1}{\tau})$.

Basado en esta muestra, obtener las distrubuciones posteriores de μ y τ usando el muestreo de Gibbs.

Notación

μ = media poblacional τ = precision (1/varianza) n = tamaño muestral $\bar{y}$ = media muestral s^2 = varianza muestral

Algoritmo

En la iteración i ($i = 1, \ldots, , N$):

- muestrea $\mu^{(i)}$ de $f(\mu|\tau^{(i-1)}, datos)$
- muestrea $\tau^{(i)}$ de $f(\tau|\mu^{(i)}, datos)$

La teoría asegura que después de un gran número de iteraciones, T, el conjunto $\{(\mu^{(i)}, \tau^{(i)}) : i = T+1, \ldots, N\}$ puede ser visto como una muestra aleatoria de la distribución conjunta posterior.

Distribuciones apriori

$$f(\mu, \tau) = f(\mu) \times f(\tau)$$

con $f(\mu) \propto 1$ y $f(\tau) \propto \tau^{-1}$.

Condicional posterior para la media, dada la precisión

$$(\mu \,|\, \tau, \text{data}) \sim \text{N}\Big(\bar{y}, \frac{1}{n\tau}\Big)$$

Condicional posterior para la precisión, dada la media

$$(\tau \,|\, \mu, \text{data}) \sim \text{Gam}\Big(\frac{n}{2}, \frac{2}{(n-1)s^2 + n(\mu - \bar{y})^2}\Big)$$

En R

```r
# resumen estadistico de la meustra
n    <- 30
ybar <- 15
s2   <- 3

# muestra de la posterior conjunta (mu, tau | data)
mu     <- rep(NA, 11000)
tau    <- rep(NA, 11000)
T      <- 1000    # datos quemados
tau[1] <- 1  # valor de inicio
for(i in 2:11000) {
    mu[i]  <- rnorm(n = 1, mean = ybar, sd = sqrt(1 / (n * tau[i - 1])))
    tau[i] <- rgamma(n = 1, shape = n / 2, scale = 2 / ((n - 1) * s2 + n * (mu[i] - ybar)^2))
}
mu  <- mu[-(1:T)]   # remuevo los quemados
tau <- tau[-(1:T)] #  remuevo los quemados
```

Referencias

Albert, J. (2009) Bayesian Computation with R. Springer. Casella, G. & George, E. I. (1992). *Explaining the Gibbs Sampler*. The American Statistician, 46, 167–174.]