

Simulación

Monte Carlo a través de Cadenas de Markov (MCMC) Metropolis-Hastings

Jorge de la Vega Góngora

Departamento de Estadística,
Instituto Tecnológico Autónomo de México

Clase 12: Semana del 23/25 de octubre de 2018

Introducción al MCMC

- MCMC fué inventado en Los Álamos en 1953 por Nicholas Metropolis, simulando el comportamiento de un líquido y su fase gaseosa. El problema era resolver expresiones de la forma:

$$\frac{\int F(\theta) \exp \left\{ -\frac{E(\theta)}{kT} \right\} d\theta}{\int \exp \left\{ -\frac{E(\theta)}{kT} \right\} d\theta}$$

en \mathbb{R}^{2N} , donde θ es un conjunto de N partículas en \mathbb{R}^2 y E denota una función compleja de θ que representa energía.

- Principalmente fue usado por químicos y físicos y hasta los 90's comenzó a usarse de manera sistemática en la estadística, particularmente en la estadística Bayesiana.



Figura: Nicholas Metropolis (derecha, con James Richardson) en Los Alamos National Laboratory.



Figura: Wilfred K. Hastings (1930 - 2016)

Wilfred K. Hastings generalizó el algoritmo en [su artículo de 1970](#). En relación a este artículo, Hastings explica:

"When I returned to the University of Toronto, after my time at Bell Labs, I focused on Monte Carlo methods and at first on methods of sampling from probability distributions with no particular area of application in mind. [University of Toronto Chemistry professor] John Valleau and his associates consulted me concerning their work. They were using Metropolis's method to estimate the mean energy of a system of particles in a defined potential field. With 6 coordinates per particle, a system of just 100 particles involved a dimension of 600. [When I learned how easy it was to generate samples from high dimensional distributions using Markov chains, I realised how important this was for Statistics, and I devoted all my time to this method and its variants which resulted in the 1970 paper.](#)"

- Un caso especial del algoritmo de Metropolis-Hastings fue introducido por **Stuart Geman y Donald Geman** en 1984, y se conoce como el *Gibbs Sampler*, nombrado así en honor Josiah Williard Gibbs, un físico americano del siglo XIX que junto con Maxwell y Boltzmann, crearon la mecánica estadística.
- **Alan E. Gelfand y Adrian F. M. Smith** popularizaron en los 90's el uso de MCMC para realizar inferencia Bayesiana:
 - dada una distribución inicial $\pi(\theta)$ para un vector de parámetros θ , y después de observar datos $y = (y_1, \dots, y_n)$, hay varias cantidades de interés que se pueden utilizar para realizar inferencias:

- La **distribución posterior**:

$$\pi(\theta|y) \propto \pi(\theta)f(y|\theta).$$

Usualmente de esta distribución se estiman funciones de θ , de la forma $E[h(\theta|y)]$ (media posterior).

- **distribuciones marginales** del vector θ :

$$\pi(\theta_j) \propto \int \pi(\theta_{-j}, \theta_j) d\theta_{-j}.$$

- **densidades predictivas**:

$$f(\tilde{y}) = \int f(\tilde{y}|\theta)\pi(\theta)d\theta.$$

- Muchas de estas expresiones son integrales que no son fáciles de resolver, por su dimensionalidad.

Los 10 algoritmos más importantes del siglo XX I

- 1946: The Metropolis Algorithm for Monte Carlo. Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly.
- 1947: Simplex Method for Linear Programming. An elegant solution to a common problem in planning and decision-making (Leonid Kantorovich, Nobel 1975, de manera independiente a Dantzig).
- 1950: Krylov Subspace Iteration Method. A technique for rapidly solving the linear equations that abound in scientific computation.
- 1951: The Decompositional Approach to Matrix Computations. A suite of techniques for numerical linear algebra.
- 1957: The Fortran Optimizing Compiler. Turns high-level code into efficient computer-readable code.
- 1959: QR Algorithm for Computing Eigenvalues. Another crucial matrix operation made swift and practical (John G. F. Francis y Vera N. Kublanovskaya)
- 1962: Quicksort Algorithms for Sorting. For the efficient handling of large databases.

Los 10 algoritmos más importantes del siglo XX II





- 1965: Fast Fourier Transform. Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.
- 1977: Integer Relation Detection. A fast method for spotting simple equations satisfied by collections of seemingly unrelated numbers.
- 1987: Fast Multipole Method. A breakthrough in dealing with the complexity of n-body calculations, applied in problems ranging from celestial mechanics to protein folding.

Fuente: IEEE

Recomendación de lectura histórica

Sharon Bertsch McGrayne: *The Theory that would not die: how Bayes' rule cracked the enigma code, hunted down russian submarines & emerged triumphant from two centuries of controversy*. Yale University Press, 2011.

- "Stuart Geman was attending Grenaders's seminar in pattern theory, and he and his brother Donald Geman tried restoring a blurry photograph of a roadside sign. The Geman brothers were interested in noise reduction and in finding ways to capture and exploit regularities to sharpen the lines and edges of unfocused images. Stuart had majored in physics as an undergraduate and knew about Monte Carlo sampling techniques. So the Geman brothers invented a variant of Monte Carlo that was particularly suited to imaging problems with lots of pixels and lattices." "Sitting at a table in Paris, Donald Geman thought about naming their system. A popular Mother's Day gift at the time was a Whitman's Sampler assortment of chocolate bonbons; a diagram inside the box top identified the filling hidden inside each candy. To Geman, the diagram was a matrix of unknown but enticing variables. 'Let's call it Gibbs sampler,' he said..."
- "The Gelfand-Smith paper was an 'epiphany in the world of statistics,' as Bayesians Christian P. Robert and George Casella reported. And just in case anyone missed their point, they added: 'Definition: epiphany n. A spiritual event... a sudden flash of recognition.' Years later, they still described its impact in terms of 'sparks,' 'flash,' 'shock,' 'impact,' and 'explosion.' Shedding their diffidence, Gelfand and Smith wrote a second paper six months later with Susan E. Hills and Amy Racine-Poon. This time they punctuated their mathematics exuberantly with words like 'surprising,' 'universality,' 'versatility,' and 'trivially implemented.' They concluded grandly, 'The potential of the methodology is enormous, rendering straightforward the analysis of a number of problems hitherto regarded as intractable from a Bayesian perspective.' Luke Tierney at Carnegie Mellon tied the technique to Metropolis's method, and the entire process—which physicists had called Monte Carlo—was baptized anew as Markov chain Monte Carlo, or MCMC for short. [The combination of Bayes and MCMC has been called 'arguably the most powerful mechanism ever created for processing data and knowledge.'](#)

the theory  that would
not die 
how bayes' rule cracked
the enigma code, 
hunted down russian
submarines & emerged
triumphant from two 
centuries of controversy
sharon bertsch mcgrayne

- La idea básica de MCMC es muy simple: Queremos simular observaciones $X_1, X_2, \dots, X_n \sim f$; construimos una cadena de Markov $\{X_i\}$, fácil de simular y cuya distribución estacionaria π corresponda a la distribución objetivo que nos interesa, $\pi = f$.
- Sin embargo, a diferencia de los métodos que hemos usado con anterioridad, *las muestras que se obtienen de la cadena de Markov $\{X_i\}$ son dependientes*.
- Para que una sucesión dependiente de variables aleatorias converja a una distribución estacionaria necesitamos que cumpla con propiedades *ergódicas*.
- A continuación repasamos algunos conceptos básicos asociados con las cadenas de Markov, que nos den elementos suficientes para definir nuestro modelo. En particular, los conceptos de *recurrencia*, *irreducibilidad* y *aperiodicidad* son las relevantes para definir una cadena de Markov ergódica.

Cadena de Markov

Una cadena de Markov es un proceso estocástico $\{X_n | n \in \mathbb{N}\}$ con valores en un *espacio muestral* S (finito o numerable) y que cumple con la propiedad de Markov:

$$\begin{aligned} P(X_{n+1} = j | X_0 = i_0, \dots, X_{n-1} = i_{n-1}, X_n = i) &= P(X_{n+1} = j | X_n = i) \\ &= p_{ij}^{n, n+1} \end{aligned}$$

para todos los puntos del tiempo n y todos los estados $i_0, \dots, i_{n-1}, i, j \in S$.

- La cadena de Markov es una sucesión de variables aleatorias $\{X_t\}$ tal que el estado actual sólo depende del estado anterior.
- Las probabilidades de transición $p_{ij}^{n, n+1}$ pueden depender del tiempo n de la transición. Cuando no dependen de n se dice que la cadena tiene *probabilidades de transición estacionarias*.
- Con probabilidades estacionarias, éstas se pueden ordenar en una matriz $\mathbf{P} = (p_{ij})$.

- Las *probabilidades de transición de n pasos* p_{ij}^n se definen como la probabilidad de que un proceso que está en el estado i llegue al estado j después de n transiciones.
- Estas probabilidades se pueden obtener a través de potencias de la matriz P , lo que da origen a las *ecuaciones de Chapman-Kolmogorov*:

$$p_{ij}^{n+m} = \sum_{k \in S} p_{ik}^n p_{kj}^m \quad (P^{n+m} = P^n P^m)$$

para toda $n, m \geq 0$ y todo $i, j \in S$.

Irreducibilidad de una Cadena de Markov

- Un estado $j \in S$ se dice que es *accesible* desde el estado i si para alguna n $p_{ij}^n > 0$.
- Dos estados que son accesibles entre sí se dice que están *comunicados*, y se escribe $i \leftrightarrow j$.
- Una cadena de Markov es *irreducible* si todos los pares de estados se comunican entre sí.

- La relación \leftrightarrow define una *relación de equivalencia* que particiona el espacio de estados S . Una vez que la cadena abandona la clase de 'comunicados', no puede volver a ella.

Recurrencia

- Definimos la *probabilidad de primer retorno al estado i en n pasos*, como

$$f_{ii}^n = P(X_0 = i, x_k \neq i, k \in \{1, 2, \dots, n-1\}, X_n = i)$$

- Entonces $f_i^* = \sum_{n=1}^{\infty} f_{ii}^n$ es la probabilidad de un eventual regreso al estado i habiendo partido de él.
- Un estado $i \in S$ es *recurrente* si $f_i^* = 1$ y *transitorio* si $f_i^* < 1$. Todos los estados que pertenecen a la clase de equivalencia de un estado recurrente son recurrentes.
- Si i es un estado recurrente, $\{f_{ii}^*, n \in \mathbb{N}\}$ es una distribución de probabilidad sobre \mathbb{N} y su media $\mu_i = \sum_{n=1}^{\infty} n f_{ii}^n$ es el número esperado de pasos para regresar a i .
- Un estado recurrente i es *recurrente positivo* si $\mu_i < \infty$ y *recurrente nulo* si $\mu_i = \infty$.

Si un estado i se visita con regularidad específica, entonces el proceso no puede converger a una distribución estacionaria. Para comprender si existe esta regularidad, se define el concepto de *periodicidad*.

Periodicidad

- Un estado i tiene periodo d si $p_{ii}^n = 0$ cuando n no es divisible por d y d es el entero más grande con esa propiedad. (d es el máximo común divisor de las $n \geq 1$ tales que $p_{ii}^n > 0$).
- Si $p_{ii}^n = 0$ para toda $n > 0$, el periodo de i es ∞ .
- Un estado con periodo 1 se dice que es *aperiodico*.

Por ejemplo en la matriz $\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{pmatrix}$ del proceso con estados $S = \{0, 1, 2, 3\}$, se tiene que, para el estado 0:

Cadenas de Markov II

Periodicidad de una Cadena de Markov

```
p <- matrix(c(0,1,0,0, 0,0,1,0, 0,0,0,1, 0.5,0,0.5,0),ncol=4,byrow=T)
p2 <- p%*%p
p3 <- p %*% p2
p4 <- p %*% p3
p5 <- p %*% p4
p6 <- p %*% p5
p7 <- p %*% p6
p8 <- p %*% p7
c(p[1,1],p2[1,1],p3[1,1],p4[1,1],p5[1,1],p6[1,1],p7[1,1],p8[1,1]) #transiciones de orden 1 a 8 en 0

[1] 0.000 0.000 0.000 0.500 0.000 0.250 0.000 0.375
```

En este ejemplo, las n tales que $p_{00}^n > 0$ son $\{4, 6, 8, \dots\}$ y por lo tanto $d(0) = 2$. Un resultado importante es que el periodo es una propiedad de la clase de equivalencia: si $i \leftrightarrow j$, entonces $d(i) = d(j)$.

- La **ergodicidad** tiene que ver con el comportamiento límite de promedios en el tiempo.

Ergodicidad

Una cadena de Markov que es irreducible, con estados recurrentes positivos y aperiódicos, es una cadena *ergódica*.

- La ley fuerte de los grandes números establece que si se tiene una sucesión de variables aleatorias Y_1, Y_2, \dots *independientes e idénticamente distribuidas* con media $\mu < \infty$, y $h : \mathbb{R} \rightarrow \mathbb{R}$ es una función acotada, entonces con probabilidad 1:

$$\lim_{n \rightarrow \infty} \frac{h(Y_1) + \dots + h(Y_n)}{n} = E(h(Y))$$

- Para cadenas de Markov, se puede considerar el equivalente Teorema ergódico, que establece esencialmente lo mismo que la ley fuerte de los grandes números, pero sin el supuesto de independencia y variables idénticamente distribuidas:

- El siguiente resultado es la base de aplicación del algoritmo de MCMC:

Teorema ergódico

Sea $\{X_t\}$ una cadena de Markov recurrente, irreducible y aperiódica, con espacio de estados Θ y distribución estacionaria π . Entonces, si $h : \mathbb{R} \rightarrow \mathbb{R}$ es una función acotada y si $n \rightarrow \infty$

i $X_n \rightarrow X$ donde $X \sim \pi$.

ii $\frac{\sum_{i=1}^n h(X_i)}{n} \rightarrow E[g(X)]$.

- Una prueba de una versión más general del Teorema ergódico se puede encontrar en el libro de James R. Norris: *Markov Chains* Cambridge University Press, 1997, (p. 53-55)
- El teorema ergódico asegura la convergencia del promedio aritmético a la integral, pero no nos dice cómo es la aproximación: ¿cuánto debe valer n para que la aproximación sea válida? No tenemos respuesta.

- En general, para procesos Markovianos en general (no necesariamente cadenas) la distribución condicional es $X_t|X_{t-1} \sim K(X_t, X_{t-1})$ donde K es el *kernel de Markov* o *kernel de transición* y determina las reglas de salto entre estados del proceso, y $X_t \in \Theta$ es el espacio de estados $\forall t$.
- Un proceso de Markov tiene probabilidades de transición *estacionarias* si $K(X_t, X_{t+k})$ no depende de t .
- La distribución de un proceso de Markov queda completamente determinada por
 - la distribución inicial π_0 de X_0 y
 - su kernel de transición $K(\cdot, \cdot)$
- El kernel y la distribución estacionaria satisfacen la ecuación:

$$\pi(x) = \int_{y \in \Theta} \pi(y) K(y, x)$$

- En el caso particular de las cadena de Markov la ecuación anterior es:

$$\pi_0 = \pi_0 \mathbf{P}.$$

Una forma de determinar si una distribución π es estacionaria es verificar si se cumple la *condición de reversibilidad* que es suficiente para una serie ser estacionaria:

Condición de balance o reversibilidad

Una cadena de Markov con kernel de transición K cumple con la *condición de balance* si para alguna función π se cumple que

$$K(x, y)\pi(x) = K(y, x)\pi(y)$$

Si además π es una función de densidad, se tiene que

$$\begin{aligned}\int_{\Omega_y} K(y, B)\pi(y)dy &= \int_{\Omega_y} \int_B K(y, x)\pi(y)dx dy \\ &= \int_B \int_{\Omega_y} K(x, y)\pi(x)dy dx \\ &= \int_B \pi(x)dx \\ &= \pi(B)\end{aligned}$$

Es decir: $\pi(B) = \int_{\Omega_y} K(y, B)\pi(x)dx$, por lo que π es la distribución invariante de la cadena de Markov. En este caso se dice que la cadena es *reversible*.

- No hay guía teórica que indique de qué manera las muestras obtenidas convergen y proveen una aproximación razonable a la densidad posterior.
- Por lo anterior, los primeros valores simulados (conocidas como **iteraciones de “burn-in”**), se eliminan porque no corresponden al estado estacionario y pueden introducir un sesgo en la estimación.
- Sin embargo, ver la nota de [Charles Geyer](#).
- En la práctica, se monitorea el comportamiento de un algoritmo MCMC inspeccionando el valor de la tasa de aceptación, construyendo algunas gráficas y calculando estadísticas diagnósticas sobre los valores muestreados. Este análisis se conoce como *análisis de salida de MCMC* (*output analysis*). Con este análisis exploratorio, se decide si la cadena ha explorado de manera suficiente la distribución posterior y si la sucesión de valores ha convergido aproximadamente.

Este ejemplo muestra cómo un problema se puede simplificar considerando la simulación de una cadena de Markov.

- Consideremos sucesiones de lanzamientos de monedas de longitud m . Sol es 1 y cara es 0.
- Decimos que una sucesión es *buenas* si no tiene 1's adyacentes. Por ejemplo, si $m = 4$, las buenas sucesiones son:

1001, 0000, 1010, ...

Ejemplos de sucesiones no buenas:

1100, 1011, 1110, ...

- **Determinar el número esperado de 1's en secuencias buenas.** Por ejemplo, con $m = 4$, hay 2^4 posibles secuencias, de las cuales 8 son buenas:

0000, 1000, 0100, 0010, 0001, 1010, 1001, 0101

En este ejemplo, la esperanza buscada es

$$\frac{1}{8}(0 + 1 + 1 + 1 + 1 + 2 + 2 + 2) = 1.25$$

Ejemplo II

- Para m general, el número esperado de 1's es $\mu = \sum_k k\pi_k$ donde $\pi_k = P[h(x) = k]$ y $h(x)$ es el número de 1's en una sucesión buena x . Aunque se puede probar que $\pi_k = \frac{\binom{m-k+1}{k}}{2^m}$ para $k = 0, 1, \dots, \lceil m/2 \rceil$, no hay una fórmula cerrada para μ .
- Si se pueden simular sucesiones buenas de longitud m , Y_1, \dots, Y_n , el estimador de Monte Carlo de μ , por la Ley de los Grandes Números será:

$$\hat{\mu} = \frac{h(Y_1) + \dots + h(Y_n)}{n} \approx \mu$$

para n suficientemente grande. Así que el problema se puede reducir a simular buenas secuencias. Hay dos vías para hacerlo:

- De manera directa.
- Usando cadenas de Markov

De manera directa. Repetir n veces el siguiente algoritmo:

- Genera sucesión de longitud m de 1's y 0's sin restricciones. Hay 2^m posibles sucesiones.
- Si la sucesión es buena, quedarse con ella, si no, rechazarla.
- Calcular $h(Y_i)$

Este método no funciona en la práctica, porque se rechaza *demasiado*. Por ejemplo, si $m = 100$, hay $2^{100} \approx 10^{30}$ sucesiones, y de éstas 10^{21} son buenas, por lo que la probabilidad de una buena sucesión en un ensayo es 10^{-9} . El método es excesivamente lento.

Usando cadenas de Markov.

- Construir una cadena de Markov $X_0, X_1 \dots X_n$ que tenga espacio de estados las sucesiones buenas y cuya distribución límite sea la distribución uniforme.
- Calcular, utilizando el teorema ergódico, $\mu \approx \frac{h(X_1) + \dots + h(X_n)}{n}$ para n suficientemente grande.

Se puede construir una cadena de la siguiente manera. Dada una buena sucesión X_n ,

- Selecciona uno de sus m componentes al azar, digamos c .
- Aplica la transformación siguiente para generar una nueva buena sucesión Y :

$$T(c) = \begin{cases} 1 & c = 0 \\ 0 & c = 1 \end{cases} \text{ y el resultado es una buena secuencia}$$

Ejemplo IV

- Mueve a la nueva secuencia si es buena o quedarse en la actual si no es buena:

$$X_{n+1} = \begin{cases} X_n & \text{si la secuencia no es buena} \\ Y & \text{si la secuencia es buena} \end{cases}$$

Por ejemplo, si $m = 4$, hay 8 posibles buenas cadenas:

$$S = \{0000, 1000, 0100, 0010, 0001, 1010, 0101, 1001\}$$

Con estos estados y aplicando las reglas anteriores, se puede generar la matrix de transición siguiente:

$$P = \begin{pmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 & 0 & 1/4 & 0 & 1/4 \\ 1/4 & 0 & 2/4 & 0 & 0 & 0 & 1/4 & 0 \\ 1/4 & 0 & 0 & 2/4 & 0 & 1/4 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 1/4 & 0 & 1/4 & 0 & 2/4 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 1/4 & 0 & 2/4 & 0 \\ 0 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 1/4 \end{pmatrix}$$

La distribución límite estacionaria de esta matriz es la distribución uniforme. Para simular este proceso, utilizamos el siguiente código:

Ejemplo V

```
adyacentes <- function(init,n){  
  #init: es la secuencia inicial  
  #n: número de iteraciones a correr en la cadena  
  m <- length(init) #longitud de las secuencias  
  nunos <- 0 # número total de 1's  
  nueva <- c(2,init,2) #identia las secuencias que se generaron usando 2 como sep  
  for(i in 1:n) {  
    indice <- 1+ sample(1:m,1) #agrego el uno por el separador  
    flip <- !nueva[indice] #cambia el número  
    if (flip==0){  
      nueva[indice] <- 0  
      nunos <- nunos + sum(nueva)  
      next  
    } else {  
      if(nueva[indice-1] == 1 | nueva[indice+1] == 1){  
        nunos <- nunos + sum(nueva)  
        next  
      } else {  
        nueva[indice] <- 1  
        nunos <- nunos + sum(nueva)}  
      }  
    }  
  }  
  return(nunos/n - 4)  
}  
adyacentes(rep(0,100),100000)  
  
[1] 27.80658
```

El análisis teórico da para $m = 100$ es $\mu = 27.7921$.

- Hay varias posibles formas para implementar MCMC. Aquí nos concentraremos básicamente en dos métodos:
 - 1 El algoritmo de Metropolis-Hastings (es el método más general)
 - 2 El Gibbs sampler. (es un caso particular del método anterior)
- Otras posibles formas se mencionan:
 - Adaptive directional Metropolis-within-Gibbs
 - Adaptive Hamiltonian Monte Carlo
 - Delayed rejection Metropolis
 - Preconditioned Crank-Nicholson
 - Refractive sampling
 - Boltzmann algorithm
 - ...

Consultar: http://m-clark.github.io/docs/ld_mcmc/

Metropolis-Hastings

El algoritmo de Metropolis-Hastings (MH) simula una cadena de Markov $\{\theta_n\}$ cuya distribución estacionaria es $\pi(\theta|x)$.

- Se comienza con un valor inicial θ_0 para la cadena.
- En el paso n , dado un valor θ_n de la cadena, se simula un nuevo valor candidato $\tilde{\theta}$, de una ‘densidad propuesta’ $q(\tilde{\theta}|\theta_n)$ (esta función es el Kernel de un proceso Markoviano)
- Con el mismo principio utilizado en la técnica de aceptación-rechazo (A-R), el valor candidato se podrá usar o no, en función de una probabilidad α que hay que determinar.
- Se requiere que la razón $\frac{\pi(y)}{q(y|\theta_t)}$ sea conocida (excepto quizá por una constante), independientemente de θ_n y que $q(\cdot|\theta_n)$ tenga suficiente dispersión para cubrir el soporte de π .
- El algoritmo de Metropolis-Hastings determina, para una densidad q propuesta dada, un kernel que tiene a π como distribución estacionaria.

- El algoritmo es el siguiente:

Algoritmo de Metropolis-Hastings

- Cuando el estado actual es $X_n = x$, el algoritmo propone moverse a un estado y , con densidad condicional $q(\cdot|x)$ de la siguiente manera:
- Muestra $y \sim q(\cdot|x)$.
- Calcula la *razón de Hastings*:

$$r(x, y) = \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}.$$

- Acepta el movimiento propuesto haciendo $X_{n+1} = y$ con probabilidad $\alpha(x, y) = \min\{1, r(x, y)\}$, o definir $X_{n+1} = x$ con probabilidad $1 - \alpha(x, y)$
- La sucesión de valores $\{X_n\}$ forma una cadena de Markov, por lo tanto la sucesión no es independiente.
- La elección de la función q puede impactar de manera importante la eficiencia del algoritmo. Es necesario considerar métodos de selección de q .

Idea de la prueba del algoritmo de MH I

Consideraremos una versión discreta del algoritmo que es más fácil de probar.

- Sea $\pi = (\pi_1, \pi_2, \dots)$ es una distribución discreta sobre un espacio de estados S , y \mathbf{P} es la matriz de transición para cualquier cadena de Markov irreducible con el mismo espacio de estados S , usaremos la matriz \mathbf{P} como la cadena propuesta.
- El algoritmo de Metropolis-Hastings adaptado a esta versión dice:
Partiendo de $X_n = i \in S$,
 - 1 Elige $j \in S$ con probabilidad p_{ij} como el nuevo estado propuesto.
 - 2 Decide si aceptar j o no. Sea $\alpha(i, j) = \frac{\pi_j P_{ji}}{\pi_i P_{ij}}$. Obtener $u \sim \mathcal{U}(0, 1)$ y definir

$$X_{n+1} = \begin{cases} j & \text{si } u < \alpha(i, j) \\ i & \text{si } u > \alpha(i, j) \end{cases}$$

Entonces $\{X_0, X_1, \dots\}$ es una cadena de Markov reversible cuya distribución estacionaria es π .

La prueba se puede esbozar así:

- Sea T la matriz de transición de la cadena construida. El resultado se tendrá si la cadena es reversible, esto es si $\pi_i T_{ij} = \pi_j T_{ji}$. Para $i \neq j$, $T_{ij} = P(X_1 = j | X_0 = i)$. Dado $X_0 = i$, entonces $X_1 = j$ si y sólo si: (i) j es propuesto y (ii) j es aceptado.

Idea de la prueba del algoritmo de MH II

- La condición (i) ocurre con probabilidad P_{ij} . La condición (ii) ocurre si $u < \alpha(i, j)$ donde $u \sim \mathcal{U}(0, 1)$
- Entonces:

$$\begin{aligned} P(u \leq \alpha(i, j)) &= \begin{cases} \alpha(i, j) & \alpha(i, j) \leq 1 \\ 1 & \alpha(i, j) > 1 \end{cases} \\ &= \begin{cases} \alpha(i, j) & \pi_j P_{ji} \leq \pi_i P_{ij} \\ 1 & \pi_j P_{ji} > \pi_i P_{ij} \end{cases} \end{aligned}$$

- Por lo anterior,

$$T_{ij} = \begin{cases} P_{ij} \alpha(i, j) & \pi_j P_{ji} \leq \pi_i P_{ij} \\ P_{ij} & \alpha(i, j) > \pi_j P_{ji} > \pi_i P_{ij} \end{cases}$$

- Ahora bien, si $\pi_j P_{ji} \leq \pi_i P_{ij}$, entonces:

$$\pi_i T_{ij} = \pi_i P_{ij} \alpha(i, j) = \pi_i P_{ij} \frac{\pi_j P_{ji}}{\pi_i P_{ij}} = \pi_j P_{ji} = \pi_j T_{ji}$$

Por otro lado, si $\pi_j P_{ji} > \pi_i P_{ij}$,

$$\pi_i T_{ij} = \pi_i P_{ij} = \pi_i P_{ij} \frac{\pi_j P_{ji}}{\pi_j P_{ji}} = \pi_j P_{ji} \alpha(i, j) = \pi_j T_{ji}$$

- No es necesaria la forma exacta de π para implementar el algoritmo de Metropolis-Hastings. El algoritmo utiliza razones de la forma π_j/π_i , por lo que se puede determinar π hasta proporcionalidad.
- Si la matriz propuesta \mathbf{P} es simétrica, entonces $\alpha(i, j) = \pi_j/\pi_i$.
- El algoritmo trabaja para cualquier cadena irreducible propuesta. Así que el usuario tiene margen para encontrar una cadena propuesta que es eficiente en el contexto de su problema.

- Hay varias maneras de proponer el proceso candidato para tomar ventaja del algoritmo de Metropolis-Hastings.
- Aquí revisaremos cómo se simplifica el procedimiento si tomamos (i) q simétrica, (ii) considerando una q independiente, (iii) considerando una caminata aleatoria.

Simétrica

Se puede seleccionar q de tal manera que sea simétrica: $q(x|y) = q(y|x)$
Entonces la razón de Hastings se simplifica a:

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

Caminata aleatoria

construye el valor en $t + 1$ como el valor de t más un error estocástico:

$$y = x + \epsilon_t, \quad \epsilon \sim g \perp\!\!\!\perp x$$

Si g es la distribución Normal, entonces $y \sim \mathcal{N}(x, \sigma^2)$ y $\frac{q(y|x)}{q(x|y)} = h(|y - x|)$ para alguna función h . Por lo tanto es simétrica también.

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

muestreador independiente

Se elige una q tal que $q(y|x) = q(y)$ no depende de x Entonces

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(x)}{\pi(x)q(y)}\right\} = \min\left\{1, \frac{w(y)}{w(x)}\right\}$$

donde $w(x) = \pi(x)/q(x)$.

Ejemplos

Ejemplo 1 I

Queremos obtener una muestra de una normal truncada,

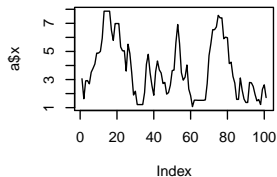
$\pi(x) = N(3, 16)I(1 \leq x \leq 8)$, i.e.

$$\pi(x) \propto e^{-\frac{(x-3)^2}{32}} I(1 \leq x \leq 8)$$

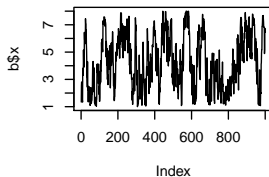
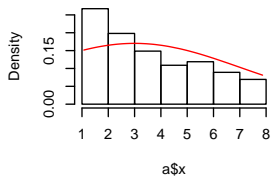
Consideremos una función propuesta $q(y|x) = N(x, 1)$ y consideremos como valor inicial $x^{(0)} = 3$.

```
simula <- function(n) {  
  f <- function(x) {exp(-(x-3)^2/32)*ifelse(x>1 & x<8,1,0)}  
  x <- NULL  
  x0 <- 3  
  c <- 1/(sqrt(2*pi*16)*(pnorm(8,mean=3,sd=4)-pnorm(1,mean=3,sd=4)))  
  for(i in 0:n){  
    w <- ifelse(i==0,x0,x[i])  
    y <- rnorm(1, mean = w, sd = 1)  
    alfa <- (f(y)*dnorm(w,mean=y,sd=1))/(f(w)*dnorm(y,mean=w,sd=1))  
    x <- append(x,ifelse(runif(1)<alfa,y,w))  
  }  
  return(list(x=x,f=c*f(sort(x))))  
}  
  
a <- simula(100)  
b <- simula(1000)  
c <- simula(10000)  
par(mfrow = c(2,3))  
plot(a$x,type="l"); plot(b$x,type="l"); plot(c$x,type="l")  
hist(a$x,probability=T); lines(sort(a$x),a$f,col="red")  
hist(b$x,probability=T); lines(sort(b$x),b$f,col="red")  
hist(c$x,probability=T); lines(sort(c$x),c$f,col="red")
```

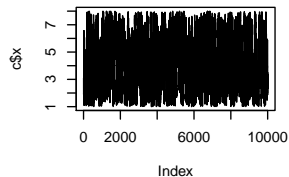
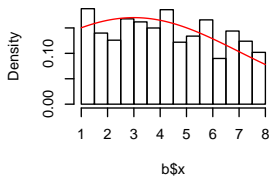
Ejemplo 1 II



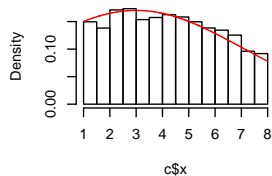
Histogram of $a\$x$



Histogram of $b\$x$



Histogram of $c\$x$



Ejemplo 2 (Robert & Casella, 6.1) I

Queremos simular $X_1, X_2, \dots, X_n \sim \mathcal{Be}(2.7, 6.3)$ usando el algoritmo de Metropolis-Hastings. En este caso la distribución objetivo es $\pi(x) = \mathcal{Be}(2.7, 6.3)$ y se requiere seleccionar una distribución candidata $q(y|x)$ que pueda recorrer el dominio de la distribución objetivo. Trivialmente se puede suponer que q es la distribución $U(0, 1)$. Esta candidata no depende del valor previo de x de la cadena.

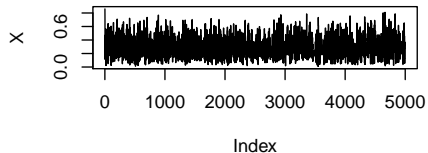
```
a <- 2.7 ; b <- 6.3 #valores iniciales
nsim <- 5000 #número de simulaciones
X <- rep(runif(1), nsim) #inicializa la cadena de Markov
for(i in 2:nsim){
  y <- runif(1) #genera y con la distribución q(y|x)
  alpha <- min(1, dbeta(y, a, b) / dbeta(X[i-1], a, b)) # Calcula la probabilidad de aceptación M-H
  X[i] <- ifelse(runif(1) < alpha, y, X[i-1])
}
```

Hacemos un "zoom" sobre una región para ver un comportamiento típico de las cadenas de Markov: por pequeños intervalos de tiempo la serie no cambia porque se rechazan los valores de y . Estas repeticiones deben mantenerse en el conjunto de datos, de otra manera la serie no tendrá la distribución que requerimos.

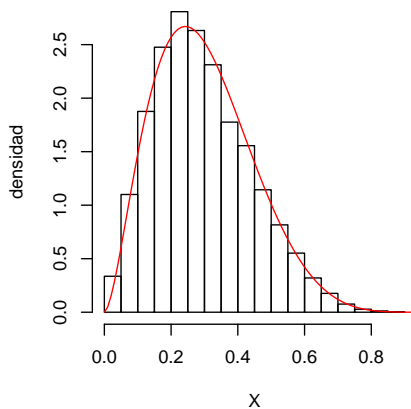
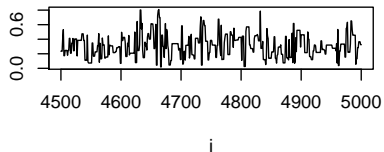
Ejemplo 2 (Robert & Casella, 6.1) II

```
layout(matrix(c(1,3,2,3),2,2,byrow=T))
plot(X, type = "l")
plot(max(1, nsim-500):nsim, X[max(1, nsim-500):nsim], type = "l", main = "Muestra de la Cadena X", xlab = "i", ylab = "X")
hist(X, probability = T, main = "", ylab="densidad")
lines(seq(0,1,0.01), dbeta(seq(0,1,0.01),a,b), col = "red")
```

Ejemplo 2 (Robert & Casella, 6.1) III

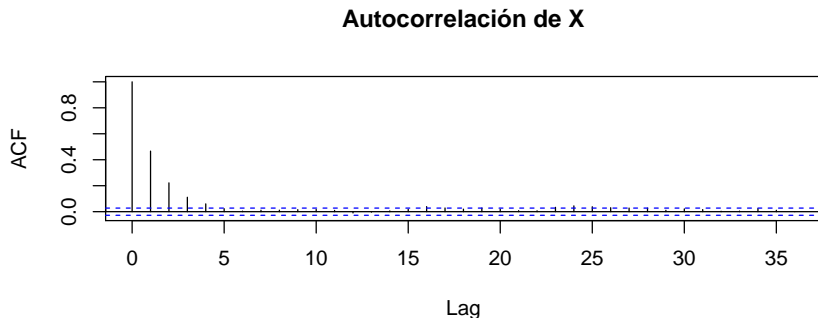


Muestra de la Cadena X



- Como la serie es una serie *dependiente*, y por lo tanto tiene autocorrelación, la muestra se degrada en relación a una muestra de variables aleatorias independientes, por lo que es necesario aumentar el tamaño de muestra para mantener un nivel de precisión dado.

```
acf(X, main = "Autocorrelación de X")
```



- El tiempo que toma la cadena en converger varía dependiendo del punto de inicio.

Con la finalidad de romper la dependencia entre observaciones de la cadena, se ha sugerido mantener en la muestra cada d observación: $X_d, X_{2d}, X_{3d}, \dots$. Este procedimiento se conoce como *adelgazamiento* (thinning).

Ventajas

- Posiblemente se obtengan observaciones más cercanas a una muestra aleatoria (iid).
- Se ahorra memoria pues se desechan algunas observaciones.

Desventajas

- No se requiere independencia por el teorema ergódico.
- Se obtiene un incremento en la varianza de las observaciones en los estimadores Monte Carlo.

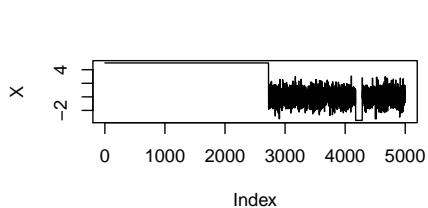
Ejemplo 3 (Robert & Casella, 6.2) I

Consideren generar muestras de una distribución $\pi = \text{Cauchy}(0, 1) = t_{(1)}$.
Como q utilizamos una distribución normal estándar.

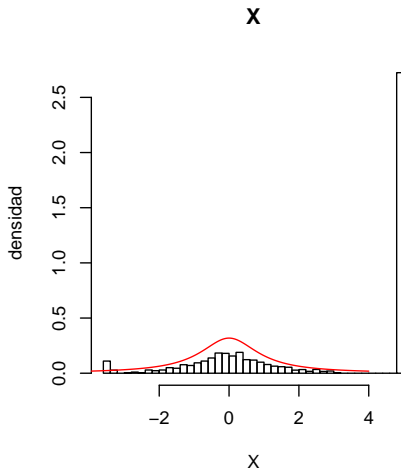
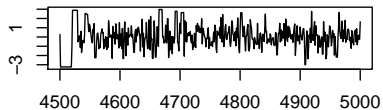
```
set.seed(10)
nsim <- 5000
#comienza con un valor de X muy grande, por ejemplo, 5
X <- NULL
X <- append(5, X)
for(i in 2:nsim){
  y <- rnorm(1) #genera y con la distribución q(y|x) = N(0,1)
  alfa <- dt(y, 1) * dnorm(X[i-1]) / (dt(X[i-1], 1) * dnorm(y)) # Calcula la probabilidad de aceptación M-H
  X <- append(X, ifelse(runif(1) < alfa, y, X[i-1]))
}

layout(matrix(c(1, 3, 2, 3), 2, 2, byrow=T))
plot(X, type="l")
plot(max(1, nsim-500):nsim, X[max(1, nsim-500):nsim], type="l", main="Muestra de X", xlab="", ylab="")
hist(X, probability = T, breaks = 50, main = "X", ylab = "densidad")
lines(seq(-4, 4, 0.01), dt(seq(-4, 4, 0.01), 1), col="red")
```

Ejemplo 3 (Robert & Casella, 6.2) II



Muestra de X



Ejercicios a considerar:

- 1 ¿Qué pasa cuando el valor inicial de la cadena es muy grande? Vean por ejemplo qué pasa cuando empiezan con un valor como 10.
- 2 ¿Cómo se comporta la cadena si en lugar de que q sea $N(0, 1)$, usamos por ejemplo $q = t_{(1/2)}$ o $q = t_{(3)}$
- 3 Calcula $P(X < 3)$ usando una cadena que utiliza q normal o q t con 0.5 grados de libertad. ¿Cuál converge más rápido?
- 4 Pueben con diferentes tamaños de simulaciones y construyan una tabla comparativa.

Ejemplo 4. I

En este ejercicio compararemos la generación de $\pi = N(0, 1)$ con tres casos de q

1. Con q general por ejemplo $q(y|x) \sim N(x, 10)$.
2. Con q considerando una caminata aleatoria.
3. Considerando q una caminata aleatoria con distribución uniforme entre $[-\delta, \delta]$, con $\delta = 0.1, 1, 10$ (este es el problema original resuelto por Metropolis en 1953). (para ustedes)

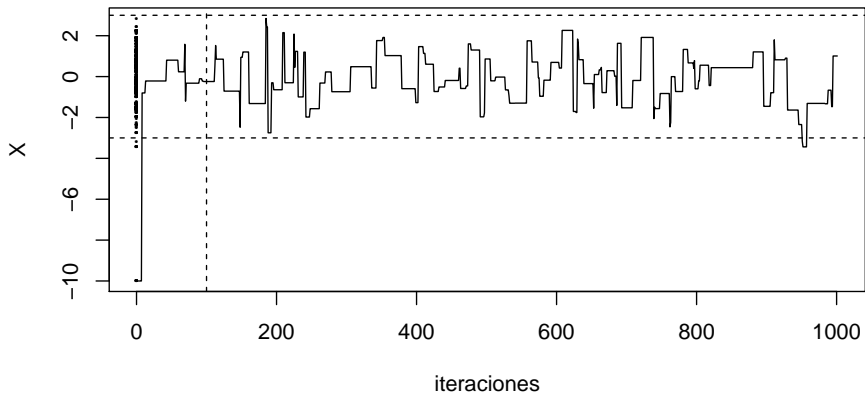
Se agregan líneas comparativas y se comienza la cadena en el mismo punto para tener una referencia y poder comparar la convergencia.

```
#Versión 1:  $q \sim N(x, 10)$ 

nsim <- 1000 #número de simulaciones
X <- NULL #inicializa la cadena
X[1] <- -10
qd <- function(x,y,sdev) {dnorm(x,mean=y,sd=sdev)}
alfa <- function(x,y,sdev) {min(1,dnorm(y)*qd(x,y,sdev)/(dnorm(x)*qd(y,x,sdev)))}
sdev <- 10

for(j in 1:nsim){
  y <- rnorm(1,mean=X[j],sd=sdev) #muestra de q
  u <- runif(1)
  X[j+1] <- ifelse(u<alfa(X[j],y,sdev),y,X[j])
}
plot(X,type="l",xlab="iteraciones")
points(rep(0,length(X)),X,pch=16,cex=0.3)
abline(h=c(-3,3),lty=2)
abline(v=100,lty=2)
```

Ejemplo 4. II



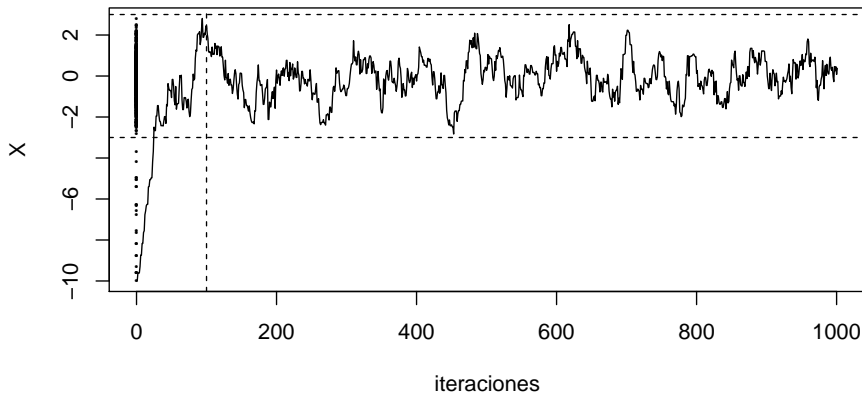
Ejemplo 4. III

```
#Versión 2: q Caminata Aleatoria

X <- NULL #inicializa
sdev <- 0.5
X[1] <- -10
pid <- dnorm
qd <- function(x,y,sdev){dnorm(x,mean=y,sd=sdev)}
alfa <- function(x,y,sdev){min(1,pid(y)/pid(x))}
for(j in 1:nsim){
  y <- X[j] + rnorm(1,sd=sdev) #muestrea de q la diferencia; Caminata aleatoria
  u <- runif(1)
  X[j+1] <- ifelse(u < alfa(X[j],y,sdev), y, X[j])
}

plot(X,type="l",xlab="iteraciones")
points(rep(0,length(X)),X,pch=16,cex=0.3)
abline(h=c(-3,3),lty=2)
abline(v=100,lty=2)
```


Ejemplo 4. IV



Este es un ejemplo de inferencia Bayesiana. Cuando se simula de una distribución posterior $\pi(\theta|x) \propto \pi(\theta)f(x|\theta)$ la q candidata independiente puede ser una normal o una t centrada en el estimador de máxima verosimilitud y con matriz de varianzas y covarianzas la inversa de la matriz de información de Fisher.

Consideren el conjunto de datos `cars` que relaciona distancia de frenado (y) con la velocidad (x) en una muestra de carros. A continuación se muestra el modelo de regresión lineal ajustado. El modelo es de la forma $dist = a + b \times speed + c \times speed^2 + \epsilon$

Ejemplo 5 II

```
data(cars)
head(cars,3)

  speed dist
1     4     2
2     4    10
3     7     4

with(cars,plot(speed,dist))
m <- lm(dist ~ speed + I(speed^2), data = cars)
summary(m)

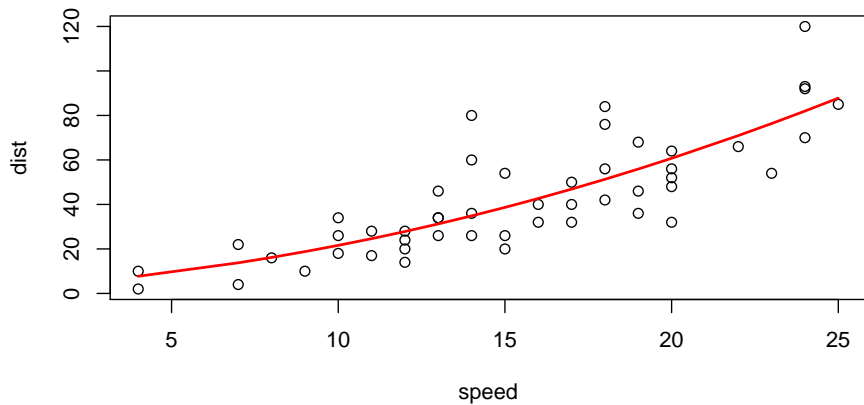
Call:
lm(formula = dist ~ speed + I(speed^2), data = cars)

Residuals:
    Min       1Q   Median       3Q      Max
-28.720  -9.184  -3.188   4.628  45.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.47014    14.81716   0.167   0.868
speed        0.91329     2.03422   0.449   0.656
I(speed^2)   0.09996     0.06597   1.515   0.136

Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-squared:  0.6673, Adjusted R-squared:  0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12

lines(cars$speed,m$fitted.values,col="red",lwd=2)
```



La función de verosimilitud de los parámetros $\theta = (a, b, c, \sigma^2)$ del modelo cuadrático es proporcional a la función:

$$\pi(\theta|x) \propto \left(\frac{1}{\sigma^2}\right)^{N/2} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{ij} (y_{ij} - a - bx_i - cx_i^2)^2 \right\}$$

Esta función de verosimilitud se puede ver como una posterior en θ , considerando una distribución inicial para θ no informativa (incluyendo para σ^2 , con $\pi(\sigma^2) \propto 1/\sigma^2$). Podemos intentar muestrear de esta distribución con el algoritmo de M-H, usando distribuciones normales provenientes de la verosimilitud, para los coeficientes del modelo.

El siguiente código hace esa estimación, simplificando un poco los cálculos.

Ejemplo 5 V

```
#Sección de inicialización
nsim <- 50000
n <- dim(cars)[1]           #número de datos
p <- 3                      #numero de coeficientes en la regresión
s2 <- sum(m$residuals^2)/(n-p) #estimador de sigma cuadrada
s2hat <- s2                 #valor constante que quedará fijo
Sig <- diag(coef(summary(m))[2]) #desviaciones estándar
bhat <- m$coefficient       #coeficientes de la regresión
y <- cars$dist; x <- cars$speed
x2 <- x^2

#Definimos la función de verosimilitud (en logaritmos para tener estabilidad numérica),
loglike <- function(a, b, c, s2){ -(n/2) * log(s2) - sum((y - a - b * x - c * x2)^2)/(2*s2) }

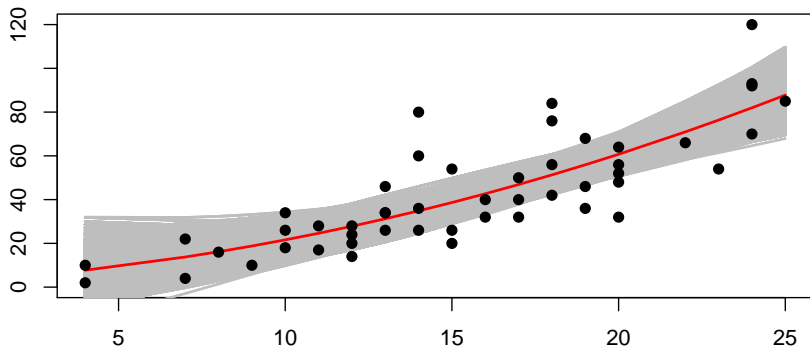
#Definimos la función candidata a partir de normales independientes centradas en el MLE
# también en logaritmos para evitar inestabilidad numérica
q1 <- function(a, b, c, s2) {
  dnorm(a, bhat[1], sd = diag(Sig)[1], log = TRUE) +
  dnorm(b, bhat[2], sd = diag(Sig)[2], log = TRUE) +
  dnorm(c, bhat[3], sd = diag(Sig)[3], log = TRUE) - (n/2) * log(s2) - s2hat*(n-p)/(2 * s2)
}
```

Ejemplo 5 VI

```
#Define la cadena
X <- matrix(rep(0,nsim*4),nrow=nsim)
X[1,] <- c(bhat,s2) #valor inicial
for (i in 2:nsim) {
  Y <- c(rnorm(1, bhat[1], diag(Sig)[1]),
        rnorm(1, bhat[2], diag(Sig)[2]),
        rnorm(1, bhat[3], diag(Sig)[3]),
        1/rgamma(1, n/2, rate = s2hat*(n-p)/2))

  alfa <- min(exp(loglike(Y[1],Y[2],Y[3],Y[4]) -
                    loglike(X[i-1,1],X[i-1,2],X[i-1,3],X[i-1,4]) +
                    q1(X[i-1,1],X[i-1,2],X[i-1,3],X[i-1,4]) -
                    q1(Y[1],Y[2],Y[3],Y[4])), 1)
  if(runif(1) < alfa) X[i,] <- Y else X[i,] <- X[i-1,]
}

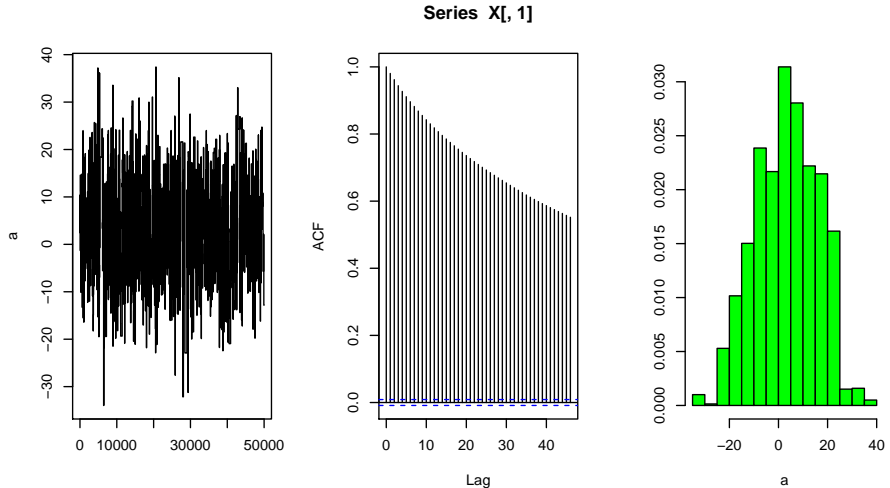
plot(x, X[nsim,1] + X[nsim,2]*x + X[nsim,3]*x2, type="l", col="grey", xlab="", ylab="", ylim=c(0, 120), lwd = 2)
for (i in 1:nsim) lines(x, X[i,1] + X[i,2]*x + X[i,3]*x2, col = "grey", lwd = 2)
lines(x, bhat[1] + bhat[2]*x + bhat[3]*x2, col = "red", lwd = 2)
points(x, y, pch = 19)
```



Análisis de Convergencia

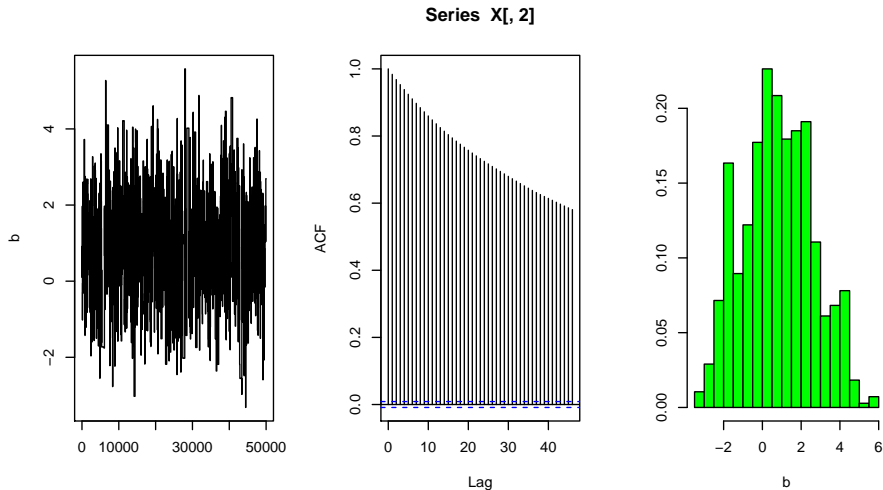
En las siguientes gráficas, podemos evaluar si las series son convergentes.

```
par(mfrow=c(1,3))  
plot(X[,1],type="l",xlab="",ylab="a")  
acf(X[,1])  
hist(X[,1],prob=T,main="",yla="",xla="a",col="green")
```



Ejemplo 5 X

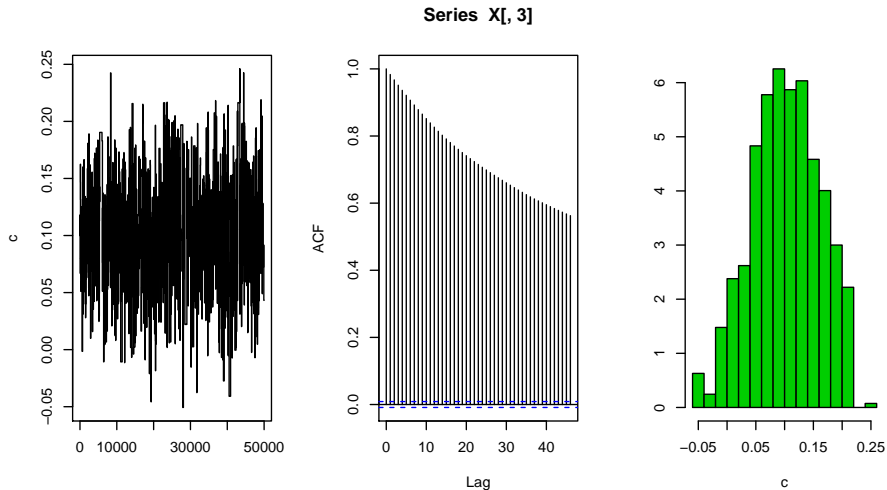
```
par(mfrow=c(1,3))  
plot(X[,2],type="l",xlab="",ylab="b")  
acf(X[,2])  
hist(X[,2],prob=T,main="",yla="",xla="b",col="green1")
```



Ejemplo 5 XII

```
par(mfrow=c(1,3))  
plot(X[,3],type="l",xlab="",ylab="c")  
acf(X[,3])  
hist(X[,3],prob=T,main="",ylab="",xlab="c",col="green3")
```

Ejemplo 5 XIII



Ejemplo 5 XIV

```
par(mfrow=c(1,3))  
plot(X[,4],type="l",xlab="",ylab="c")  
acf(X[,4])  
hist(X[,4],prob=T,main="",ylab="",xlab="c",col="green4")
```

