César Isaí García Cornejo
cesar.cornejo@cimat.mx
Computo Científico

Tarea 9

Semestre 3

CIMAT

Ejercicio 1
En ambos problemas hay que diseñar e implementar el MCMC, investigar sobre su convergencia y tener algún grado de certeza si sí se está simulando de la posterior correspondiente. Más aún, recuerde que se trata de un problema de inferencia. Hay que hablar del problema en si, comentar sobre las posteriores simuladas y posibles estimadores (a partir de la muestra de posterior) que se puede proporcionar de cada parámetro.

Ejercicio (Problema en ecología)
Sean $X_1, ..., X_m$ variables aleatorias donde X_i denota el número de individuos de una especie en cierta región. Suponga que $X_i|N, p \sim Binomial(N, p)$, entonces

$$f(\bar{X}|N, p) = \prod_{i=1}^{m} \frac{N!}{x_i!(N - x_i)!} p^{x_i}(1 - p)^{N-x_i}.$$

Asumiendo la distribución a priori $p \sim Beta(\alpha, \beta)$ y $N \sim h(\cdot)$, donde h es una dist. discreta en $\{0, 1, 2, ..., N_{max}\}$, se tiene definida la distribución posterior $f(N, P|\bar{x})$. A partir del algoritmo MH, simule valores de a distribución posterior usando un kernel híbirido. Para ello considere como sugerencia la siguiente distribución inicial para el MCMC

$$p \sim U(0, 1) \quad y \quad N \sim U_d \left\{ \max_{i \in \{1,...,m\}} (x_i), \max_{i \in \{1,...,m\}} (x_i) + 1, ..., N_{max} \right\}$$

y las propuestas

1. Propuesta 1: De la condicional total de p (kener Gibbs).
2. Propuesta 2: De la a priori
3. Propuesta 3: Propuesta hipergeométrica
4. Propuesta 4: Poisson: $N_p \sim \max_{i \in \{1,...,m\}}(x_i) + Poisson(\cdot)$.
5. Propuesta 5: Caminata aleatoria

$$N_p = N + \varepsilon, \quad \mathbb{P}(\varepsilon = 1) = \frac{1}{2} = \mathbb{P}(\varepsilon = -1)$$

Los datos son estos: 7,7,8,8,9,4,7,5,5,6,9,8,11,7,5,5,7,3,10,3.

A priori, esperamos que sea difícil observar a los individuos entonces $\alpha = 1, \beta = 20$. La especie no es muy abundadnte y entonces $N_{max} = 1000$ y $h(N) = 1/(N_{max} + 1); N \in \{0, 1, 2, ..., N_{max}\}$. Las propuestas y distribución inicial para el MCMC de arriba son solamente sugerencia, propongan otras propuestas, experimente y comenten.

Solución:

Tenemos un grupo de individuos que se modelan según una ley binomial para cierta región. Luego, desconocemos los parámetros de dicha ley. Con la metodología dada por la inferencia bayesiana, se propone

cada parámetro como una variable aleatoria y aplicamos la metodología usual para determinar la distribución a posterior, que es la distribución de los parámetros dadas las observaciones. Como establece el enunciado, las distribuciones a priori se propusieron como beta y uniforme discreta, respectivamente.

La función objetivo es en nuestro caso la distribución a posteriori, cuya expresión explícita es

$$\begin{aligned} f(p,N) &\propto \mathcal{L}(p,N|\bar{x})f(p,N) \\ &\propto \left(\prod_{i=1}^{m} \frac{N!}{x_i!(N-x_i!)} p^{x_i}(1-p_i)^{N-x_i}\right) \cdot p^{\alpha-1}(1-p)^{\beta-1} \cdot \frac{1}{N_{max}+1}\mathbb{1}_{\{0,1,\ldots,N_{max}\}}(N) \end{aligned}$$

Se pueden quitar ciertas constantes y reacomodar la expresión para obtener la expresión simplificada. Sin embargo, como es usual, y por estabilidad numérica buscamos el logaritmo de la objetivo obteniendo

$$\begin{aligned} \log f(p,N|\bar{x}) \propto m\log(N!) - \sum \log x_i! - \sum \log(N-x_i)! + \sum x_i \log p + \\ +(mN - \sum x_i)\log(1-p) + (\alpha-1)\log p + (\beta-1)\log(1-p) \end{aligned}$$

Se proponen los puntos iniciales tal como es sugerido con el siguiente comando

```
N_max = 1000
w = np.arange(max(x)+ 1, N_max)

p_0 = uniform.rvs(0,1)
N_0 = random.choice(w)
```

Notemos que para simular de una uniforme discreta se uso random.choice que selecciona al azar de un elemento en la lista previamente definida.

Se define cada propuesta, para tomar de forma equiprobable con kerneles híbridos. La primer propuesta solo salta de valores en p y mantiene N constante, está ayuda a bajar los valores altos de N en la siguiente iteración. La segunda propuesta por si sola es suficiente ya que con ella se obtendría gráficas parecidas a la obtenida con el kernel híbrido. La tercer y cuarta propuesta se parecen en el sentido de que se agrega una cantidad no nula, solamente hay que cuidar que no supere a N_{max}, ambas aumentan el valor de N. Por último la quinta propuesta es una caminata aleatoria sobre N que también aumenta su valor a la vez que lo disminuye.

Luego, con el método de Metropolis-Hastings tenemos una realización de la trayectoria que se muestra a continuación

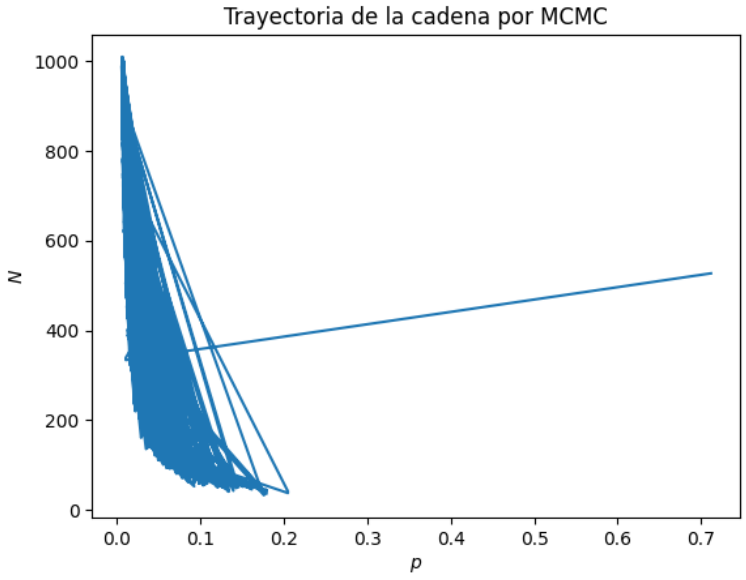

Figura 1: Trayectoria de la cadena por MCMC.

Vemos que tenemos un salto pronunciado, esto se debe al punto inicial que es tomado al azar en cualquier parte del recuadro. Veamos de que tamaño necesita ser el burn-in para que se consideren el resto de las simulaciones como muestras de la función objetivo.

Graficamos la evolución de la cadena para cada iteración obteniendo

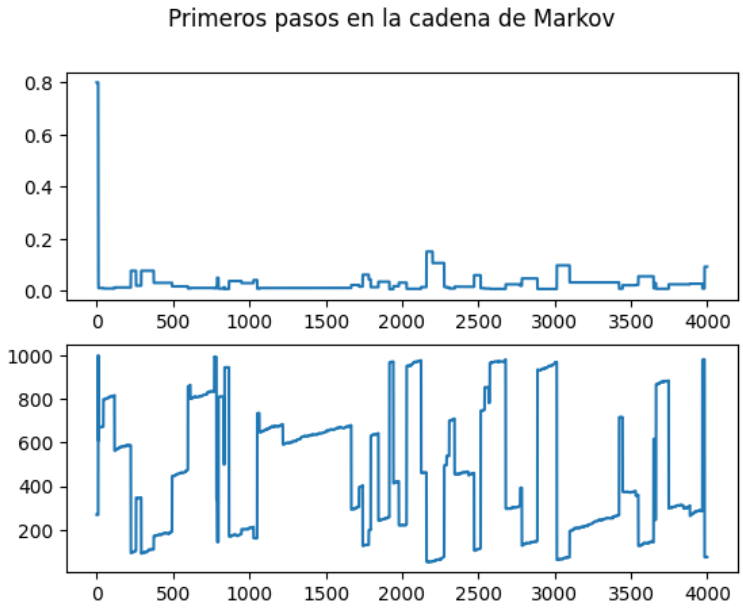

Figura 2: Evolución de la cadena para las primeras 4000 iteraciones.

Notamos que para la variable p tenemos una disparidad, considerable del valor inicial con los posteriores. En contraste, dicha disparidad no ocurre para N. Como la cadena empieza relativamente cerca a su distribución convergente, entonces podemos tomar un burn in no muy grande. Para nuestro propósito $n = 1000$ es suficiente.

La trayectoria de las simulaciones con burn in es ahora

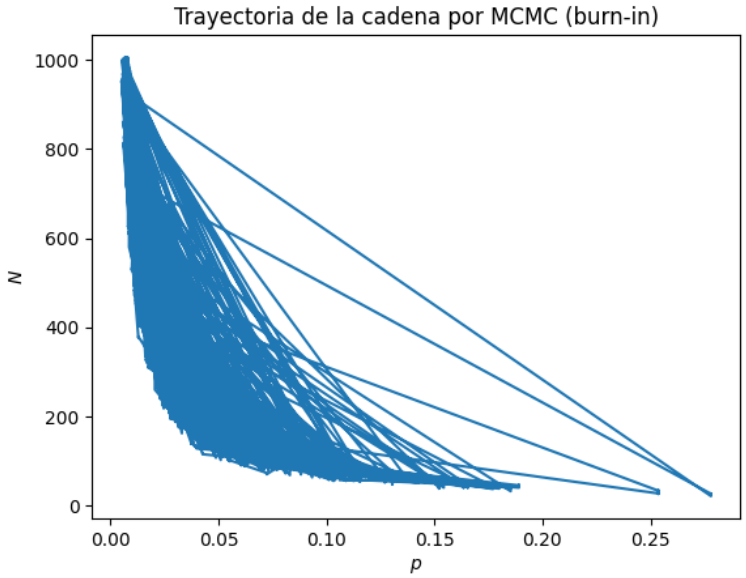

Figura 3: Trayectoria de la cadena por MCMC con burn-in.

Notemos que se limpia la trayectoria inicial con un cambio abrupto. Luego, obtenemos los histograma de las distribuciones marginales.

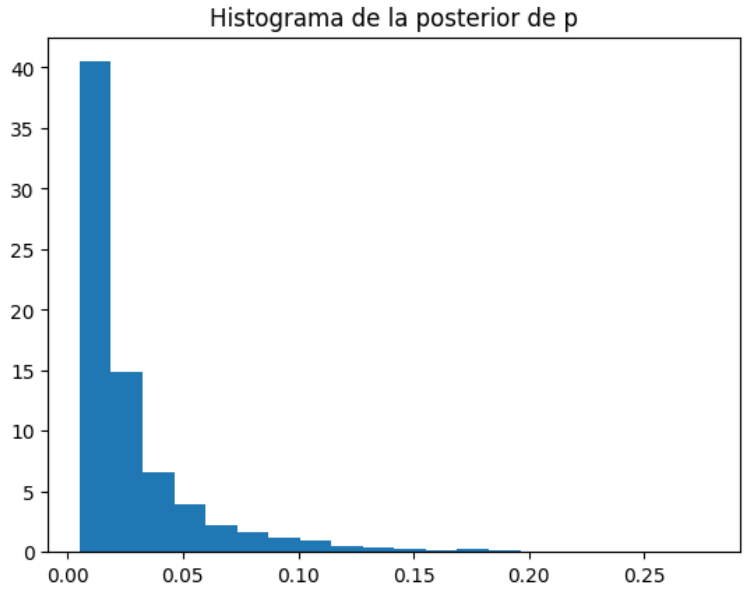

Figura 4: Histograma para la distribución posterior de p.

y el histograma para M es

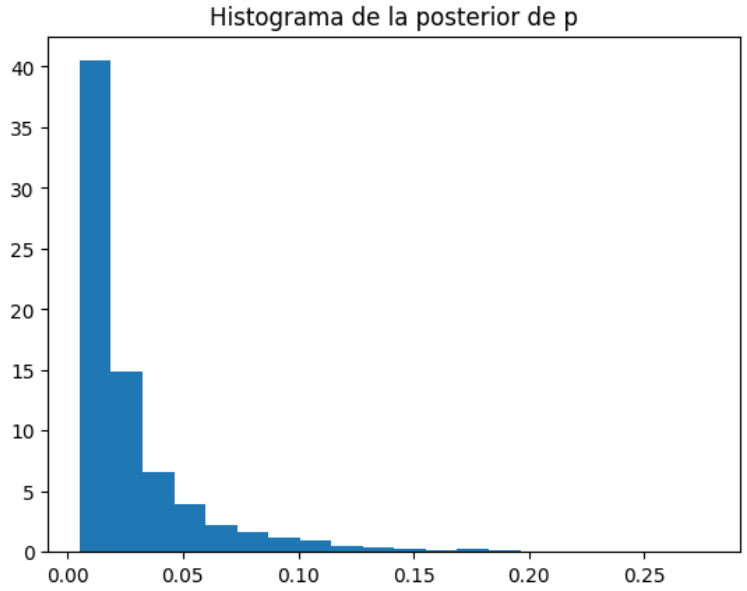

Figura 5: Histograma para la distribución posterior de p.

Para obtener una estimación puntual de los parámetros, con el estimador de bayes muestral obtenemos la media de las simulaciones dando:

```
Media muestral para p:  0.027603744890256174
Media muestral para N: 448.74062
```

que nos deja la estimación bayesiana del modelo y concluye el ejercicio.

Ejercicio (Estudio de mercado)
Se tiene un producto y se realiza una encuesta con el fin de estudiar cuánto se consume dependiendo de la edad. Sea Y_i el monto de compra y X_i la covariable la cuál representa la edad. Sea Y_i el monto de compra y X_i la covariable la cual representa la edad.

Suponga que $Y_i \sim Po(\lambda_i)$

$$\lambda_i = cg_b(x_i - a)$$

para g_b la siguiente función de liga

$$g_b(x) = \exp\left(-\frac{-x^2}{2b^2}\right)$$

O sea, se trata de regresión Poisson con una función de liga no usual. Si $\lambda_i = 0$ entonces $\mathbb{P}(Y_i = 0) =$ 1.a = años medios del segmento (años), c = gastos promedio (pesos), b = 'amplitud' del segmento (años).

Considere la distribución a priori

$$\alpha \sim N(35, 5), \quad c \sim Gama(3, 3/350) \quad b \sim Gama(2, 2/5).$$

El segundo parámetro de la normal es la desviación estándar y el segundo parámetro de las gamas es la taza.

Usando MH simule de la distribución posterior de a, c y b. Los datos son estos n= 100.

Solución:

Ejercicio 3
Investiga y describe brevemente los softwares OpenBugs, Nimble, JAGS, DRAM, Rtwalk, Mcee Hammer, PyMCMC.

Solución:
OpenBUGS

Es un software libre utilizado para análisis estadístico, específicamente para modelos bayesianos. Tiene funcionalidades avanzadas para el muestreo MCMC y puede ejecutarse de forma nativa en Linux. En este se pueden definir modelos estadísticos utilizando una notación similar a la de un modelo probabilístico y utiliza métodos de muestro de Monte Carlo y Gibbs para aproximar la distribución posterior. Otra ventaja es que se puede utilizar mediante lenguajes como R y Python, entre otros.

Por ejemplo, si se quiere usar en Python, se debe de importar "pybugs", y escribir el modelo en formato BUGS, después se definen los parámetros, se crea una instancia y finalmente se realiza el muestreo.

NIMBLE

La sigla NIMBLE proviene de "Numerical Inference for Statistical Models Using Bayesian and Likehood Estimation". Es una paquetería para construir y compartir modelos estadísticos. Facilita la programación de algoritmos estadísticos, es una extensión de BUGS y también se puede entender como una biblioteca de algoritmos como MCMC. Para ejecutar nimble en R, se debe de usar la paquetería "nimble", definir el modelo, crear un modelo con "nimbleModel", compilarlo y hacer el muestreo de Gibbs para inferencia bayesiana con "MCMC".

JAGS La sigla JAGS significa "Just Another Gibbs Sampler" es un programa para el análisis jerárquico bayesiano que utiliza la simulación MCMC no tan distinta a BUGS. Es un software que permite especificar modelos estadísticos en un formato similar al lenguaje BUGS, lo que hace que sea sencillo definir

los modelos. Algunas características son: especificación del modelo similar a BUGS, implementación de muestreo de Gibbs, puede utilizarse para modelos bayesianos simples o complejos, se puede utilizar en R, Python, matlab, entre otros.

Para utilizarlo en R, se debe de instalar "rjags" y después sigue un procedimiento similar a los ya mencionados.

DRAM Significa "Delayed Rejection Adaptive Metropolis", es un método de optimización utilizado para el MCMC. Se usa específicamente para mejorar la eficiencia y la capacidad de exploración del espacio de parámetros. Se basa en el método Metropolis-Hastings, y su idea principal es la introducción de mecanismos de rechazo diferido para mejorar la eficiencia de adaptación; esto permite realizar múltiples intentos para generar una muestra aceptable. Para usar este método se debe de instalar "MCMCpack", y para ejecutar DRAM se utiliza la función "MCMCmetrop1R".

Rtwalk Es un algoritmo de simulación utilizado para MCMC. Se usa para explorar y muestrear de forma eficiente el espacio de parámetros. Este método realiza pasos en el espacio de parámetros basándose en una caminata aleatoria para explorar diferentes regiones y capturar muestras representativas de la distribución. Es un algoritmo adaptable, y con ello mejora la eficiencia de la exploración.

No es un algoritmo estándar en R, pero se puede hacer a partir de generar muestras de una función objetivo, realizar pasos aleatorios y aceptando o rechazando los pasos.

The MCMC Hammer Es una herramienta introducida por la comunidad astronómica. Está diseñada para implementar el algoritmo MCMC, para realizar inferencia bayesiana y estimación de parámetros astronómicos complejos.

PyMCMC Ahora llamado "PyMC" es un módulo de Python que implementa estadística Bayesiana incluida MCMC. Incluye un gran conjunto de distribuciones estadísticas, utiliza Numpy para lo numérico, utiliza un modulo para modelar procesos Gausianos, puede hacer resúmenes, también hay muchos diagnósticos de convergencia, etcétera. Para usarlo se debe de importar "pymc" y utilizar funciones tipo "pymc.Normal, pymc.Binomial".

Referencias

[1] Robert, C. P., Casella, G., and Casella, G. (1999). Monte Carlo statistical methods (Vol. 2). New York: Springer.

[2] Wasserman, L. (2004). All of statistics: a concise course in statistical inference (p. 413). New York: Springer.