

Simulaciones de Monte Carlo

Alan Ledesma Arista.¹

¹alan.ledesma@bcrp.gob.pe

- 1 Integración
 - Idea básica
 - Calcular momentos
 - Calidad de integración
- 2 Simulación
 - Cuando F_X es analítico e invertible
 - Gibbs-sampling
- 3 Metropolis-Hastings
 - Aceptación y rechazo
 - Metropolis-Hastings

Idea básica

- Texto principal: [Huynh et al. 2011 \(cap. 5\)](#)
- Monte Carlo (MC): algoritmo utilizado para cálculo numérico de momentos sobre transformaciones de variables aleatorias
- Método: ejecutar una secuencia de experimentos y tomar el valor promedio
- Para calcular $y = \int_{\Omega} f(\mu) d\mu$ la metodología requiere de una secuencia $\{x_i\}_{i=0}^n$ –con $x_i \in \Omega$ para todo i – tal que

$$\hat{y} = \frac{1}{n} \sum_{i=1}^n f(x_i)$$

- Intuición: Integral de Riemann
- Si x_i es variable aleatoria, la ley de los grandes números justifica la metodología
 - x_i debe recorrer ‘la mayor parte de’ Ω
 - n debe ser ‘lo suficientemente grande’
- Ver ejemplo: [01_MCMC01_intro.ipynb](#)

Calcular momentos

- En general, si la variable aleatoria $x : (\Omega, \mathcal{F}, \mathbb{P}) \rightarrow \mathbb{R}^k$ tiene una Función de Distribución Acumulada (CDF) $F(x)$, entonces

$$h = \mathbb{E}[g(x)] = \int_{\mathbb{R}^k} g(\mu) dF(\mu) \rightarrow \hat{h} = \frac{1}{n} \sum_{i=1}^n g(x_i)$$

para realizaciones x_i de x

- Por la [ley de los grandes números](#)

$$\text{Prob} \left(|\hat{h} - h| < \epsilon \sigma_h \right) = \text{Prob} (|z| < \epsilon \sqrt{n}) \geq 1 - \frac{1}{n\epsilon^2} > 1 - \alpha$$

con $z = \sqrt{n}(\hat{h} - h)/\sigma_h$.

- Note que $\text{Prob} (|z| < \epsilon \sqrt{n}) = \text{erf} \left(\epsilon \sqrt{\frac{n}{2}} \right)$ entonces $n > 2 \left(\text{erf}^{-1}[1 - \alpha]/\epsilon \right)^2$
 - Si $\alpha = 0,05$ y $\epsilon = 0,01$ cuantas simulaciones como mínimo se requieren?
 - Que ocurre con n si se requiere de una mayor precisión (i.e., $\epsilon \rightarrow 0$)?
- Ver ejemplo: [uso de `erfinv`\(·\) en 01_MCMC02_ImportanceSampling](#)

Mejorar la calidad de integración

- La calidad de la simulación se puede mejorar en 2 aspectos: aumentar la precisión o reducir el tiempo de cómputo
- Revisaremos las dos metodologías más utilizadas:
 - Antithetic variables (reducción de varianza)
 - Importance sampling

Antithetic variables

- Reduce el número de simulaciones requeridas al encontrar una transformación de la variable aleatoria simulada que preserve la distribución y propiedades
- Sea $X \sim F_X$, entonces $X^a = T(X)$ es una variable perfectamente antitética de X si $X^a \sim F_X$ y $\text{corr}(X, X^a) = -1$
- Ejemplos:
 - Si $X \sim U(a, b)$: $X^a = b + a - X$ es Perf. antitética de X pues $X^a \sim U(a, b)$
 - Si $X \sim \mathcal{N}(\mu, \sigma^2)$: $X^a = 2\mu - X$ es Perf. antitética de X pues $X^a \sim \mathcal{N}(\mu, \sigma^2)$
 - $X \sim \exp(\lambda)$ no tiene paralelo Perf. antitético
- Si el perfecto antitético no existe, se puede diseñar un X^a tal que $\text{corr}(X, X^a) \rightarrow -1$
- Con un perfecto antitético, el número de simulaciones se reduce a la mitad, pues si se obtiene X_i de F_X entonces $X_i^a = T(X_i)$ también se puede considerar como simulación válida
- Ver ejemplo: [ver sección 4.1. de 01_MCMC02_ImportanceSampling.ipynb](#)

Importance sampling

- Transformación sobre la variable de interés tal que se reduzca la varianza de la variable a simular
- Suponga que $X \sim F_X$ con densidad f_X y considere la transformación $Y = h(X)$ entonces

$$E[Y] = \int h(\mu) dF_X(\mu) = \int h(\mu) f_X(\mu) d\mu \rightarrow \hat{m}_Y = \frac{1}{n} \sum_{i=1}^n h(X_i)$$

- Si consideramos $Y \sim F_Y$ con densidad f_Y . Entonces la integral de arriba se puede escribir como

$$E[Y] = \int h(\mu) \frac{f_X(\mu)}{f_Y(\mu)} f_Y(\mu) d\mu = \int Z(\mu) dF_Y(\mu)$$

con $Z(Y) = h(Y) \frac{f_X(Y)}{f_Y(Y)}$.

- **Note que $E[Z] = E[Y]$:** la aproximación MC sobre $E[Z]$ sirve para calcular $E[Y]$

$$\hat{m}_Z = \hat{m}_Y = \frac{1}{n} \sum_{i=1}^n Y_i \frac{f_X(Y_i)}{f_Y(Y_i)}$$

- El 'ejercicio' consiste en encontrar $f_Y(\cdot)$ tal que la $var(Z)$ es reducida
- Ver ejemplo: [ver sección 4.2. de 01_MCMC02_ImportanceSampling.ipynb](#)

Simulación

- Para la integración de MC, se requiere simular realizaciones X_i de F_X
- Nos concentraremos en simulación de variables aleatorias continuas
- En algunos casos, F_X es analítico e invertible
- En la mayoría de aplicaciones F_X o no es invertible o no tiene expresión analítica

Cuando F_X es analítico e invertible

- Suponga que $X \sim F_X$, por definición $\mathbb{P}(X \leq x) = F_X(x)$. Como la probabilidad $\mathbb{P}(X \leq x) \in [0, 1]$, ésta puede ser simulada por $U(0, 1)$
- Si $\mathbb{P}(X \leq x) \equiv U \sim U(0, 1)$ entonces de $U = F_X(X)$ se tiene que $X = F_X^{-1}(U)$
- Por lo tanto, si U_i es una simulación de $U(0, 1)$, entonces $X_i = F_X^{-1}(U_i)$ es una simulación de F_X
- Ver ejemplo: [sección 1.1. de 01_MCMC03_GS_MH.ipynb](#)

Normal multivariada

- Suponga que $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$
- La simulación requiere de obtener una matriz triangular inferior C tal que $CC^T = \Sigma$
- Cómo Σ es definida positiva y simétrica, entonces C existe y es regular
- La propuesta más común es $C = \text{Cholesky}(\Sigma)$
- La secuencia de simulación es
 - Simular k normal estándar realizaciones y apilarlos en el vector $\mathbf{Z}_i = [Z_1 \ Z_2 \ \dots \ Z_k]^T$ donde k es el rango de μ
 - La simulación \mathbf{X}_i es $\mathbf{X}_i = C\mathbf{Z}_i + \mu$
- Ver ejemplo: [sección 1.3. de 01_MCMC03_GS_MH.ipynb](#)

Gibbs-sampling

- Texto principal: [Kim y Nelson 1999 \(cap. 7\)](#)
- Texto secundario: [Gelfand 2000](#)
- Gibbs-sampling es una simulación de Monte Carlo Cadena de Markov (MCMC) útil para aproximar distribuciones conjuntas a partir de las distribuciones condicionales
- Suponga que se requiere simular $\mathbf{z} = (z_1 \dots z_k)$ cuya distribución conjunta es $f(z_1, z_2, \dots, z_k)$. Sin embargo, no contamos con la expresión analítica de $f(\cdot)$ pero sí del set completo de distribuciones condicionales: $f_{i|-i}(z_i|\mathbf{z}_{-i})$ para todo i (donde $\mathbf{z}_{-i} \equiv \{z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_k\}$). Entonces, la simulación se implementa según
 1. Proponer un punto inicial $\mathbf{z}^{(0)} = (z_1^{(0)} \dots z_k^{(0)})$
 2. Generar la simulación j de en la secuencia de abajo desde $j = 1$ hasta $j = n + b$
 - 2.1. Simular $z_1^{(j)}$ de $f_{1|-1}(z_1|z_2^{(j-1)}, z_3^{(j-1)}, z_4^{(j-1)}, \dots, z_{k-1}^{(j-1)}, z_k^{(j-1)})$
 - 2.2. Simular $z_2^{(j)}$ de $f_{2|-2}(z_2|z_1^{(j)}, z_3^{(j-1)}, z_4^{(j-1)}, \dots, z_{k-1}^{(j-1)}, z_k^{(j-1)})$
 - 2.3. Simular $z_3^{(j)}$ de $f_{3|-3}(z_3|z_1^{(j)}, z_2^{(j)}, z_4^{(j-1)}, \dots, z_{k-1}^{(j-1)}, z_k^{(j-1)})$
 - \vdots
 - \vdots
 2. $k-1$. Simular $z_{k-1}^{(j)}$ de $f_{k|-k}(z_k|z_1^{(j)}, z_2^{(j)}, z_3^{(j)}, \dots, z_{k-2}^{(j)}, z_k^{(j-1)})$
 2. k . Simular $z_k^{(j)}$ de $f_{k|-k}(z_k|z_1^{(j)}, z_2^{(j)}, z_3^{(j)}, \dots, z_{k-2}^{(j)}, z_{k-1}^{(j)})$
 - 3 Eliminar las primeras b simulaciones (burning sample)
- [Geman y Geman 1984](#) demuestran que la secuencia anterior converge exponencialmente a simulaciones de la distribución conjunta

Gibbs-sampling: ejemplo

- La estimación Bayesiana en el modelo lineal general: $Y = \beta X + u$ con distribuciones prior $\beta|\sigma_u^2 \sim \mathcal{N}(\beta_*, \Sigma_*)$ y $\sigma_u^2|\beta \sim \Gamma^{-1}\left(\frac{\nu_*}{2}, \frac{\delta_*}{2}\right)$ conduce a las siguientes distribuciones posteriors

$$\beta|\sigma_u^2, Y \sim \mathcal{N}(\beta^*, \Sigma^*) \text{ y } \sigma_u^2|\beta \sim \Gamma^{-1}\left(\frac{\nu^*}{2}, \frac{\delta^*}{2}\right) \text{ con}$$

$$\beta^* = (\Sigma_*^{-1} + \sigma_u^{-2} X^T X)^{-1} (\Sigma_*^{-1} \beta_* + \sigma_u^{-2} X^T Y)$$

$$\Sigma^* = (\Sigma_*^{-1} + \sigma_u^{-2} X^T X)^{-1}$$

$$\nu^* = \nu_* + T \text{ and } \delta^* = \delta_* + (Y - X\beta)^T (Y - X\beta)$$

- Utilice el log-crecimiento del PBI para estimar un proceso AR(4)
- Genere 10^6 simulaciones de β y σ_u^2
- Ver ejemplo: [sección 1.4. de 01_MCMC03_GS_MH.ipynb](#)

Entendiendo el algoritmo de Metropolis-Hastings

- En este segmento seguimos a [Chib y Greenberg 1995](#)
- Algoritmo de Acceptance-Rejection
- Algoritmo de Metropolis-Hasting
- La metodología se hace pública en [Metropolis et al. 1953](#)

Acceptance-Rejection

- Queremos simular $g(x) = f(x)/k$
 - $f(x)$: Densidad no normalizada
 - k : constante normalizadora (potencialmente desconocida)
- Suponga: *i*) que se puede simular fácilmente $h(x)$ y *ii*) se conoce una constante c tal que $f(x) \leq ch(x)$
- El algoritmo es:
 1. Generar una simulación candidata z a partir de una simulación de $h(x)$
 2. Simular u a partir de $U(0, 1)$
 3. Si $u \leq f(z)/ch(z)$ aceptar z como una simulación de $g(x)$
 4. De lo contrario, rechazar z y regresar al paso 1
- El algoritmo es optimizado si $c = \sup_x \frac{f(x)}{h(x)}$
- Por lo general, el algoritmo resulta en un número grande e indeseable de rechazos
- Ver ejemplo: [sección 1.2. de 01_MCMC03_GS_MH.ipynb](#)

Random-Walk Metropolis-Hastings

- En el algoritmo anterior, los candidatos se simulaban con distribuciones i.i.d. Ahora, la simulación de los candidatos dependerá de los estados corrientes (Markov-Chain Monte Carlo)
- El candidato $x^{(i)}$ para $g(x)$ se obtienen de $x^{(i)} = x^{(i-1)} + z$ con $z \sim \mathcal{N}(0, \Omega)$ donde $x^{(i-1)}$ es una simulación previamente aceptada (de aquí el nombre Random-Walk)
- Según la cadena de Markov, la probabilidad de transición es $q(x^{(i-1)}, x^{(i)})$ que para el caso descrito (Random-Walk + Normal) es reversible (i.e., $q(x^{(i-1)}, x^{(i)}) = q(x^{(i)}, x^{(i-1)})$)
- La probabilidad de tomar el candidato $x^{(i)}$ es $g(x^{(i-1)})q(x^{(i-1)}, x^{(i)})$ mientras que la probabilidad de mantener la simulación previa es $g(x^{(i)})q(x^{(i)}, x^{(i-1)})$

Random-Walk Metropolis-Hastings

- En general $g(x^{(i-1)})q(x^{(i-1)}, x^{(i)}) \geq (x^{(i)})q(x^{(i)}, x^{(i-1)})$; sin embargo, para evitar el problema de 'muchos' rechazos (o aceptaciones) deberíamos buscar la igualdad. Para lo cual creamos la probabilidad $\alpha(x^{(i-1)}, x^{(i)})$ tal que $p_{MH}(x^{(i-1)}, x^{(i)}) = q(x^{(i-1)}, x^{(i)})\alpha(x^{(i-1)}, x^{(i)})$ y

$$g(x^{(i-1)})p_{MH}(x^{(i-1)}, x^{(i)}) = g(x^{(i)})p_{MH}(x^{(i)}, x^{(i-1)})$$

- De la igualdad anterior y fijando $\alpha(x^{(i)}, x^{(i-1)}) = 1$

$$g(x^{(i-1)})q(x^{(i-1)}, x^{(i)})\alpha(x^{(i-1)}, x^{(i)}) = g(x^{(i)})q(x^{(i)}, x^{(i-1)})$$

por lo tanto

$$\alpha(x^{(i-1)}, x^{(i)}) = \min \left[\frac{g(x^{(i)})q(x^{(i)}, x^{(i-1)})}{g(x^{(i-1)})q(x^{(i-1)}, x^{(i)})}, 1 \right]$$







- En el caso descrito acá (Random-Walk + Normal) ocurre que $q(x^{(i)}, x^{(i-1)}) = q(x^{(i-1)}, x^{(i)})$ y además $g(x) = f(x)/k$ por lo tanto

$$\alpha(x^{(i-1)}, x^{(i)}) = \min \left[\frac{f(x^{(i)})}{f(x^{(i-1)})}, 1 \right]$$

Random-Walk Metropolis-Hastings

- El algoritmo es
 1. Proponer un valor inicial $x^{(0)}$ de $g(\cdot)$
 2. Para $j > 0$, simular x^* de $q(x^{(j-1)}, x^*)$ y u de $U(0, 1)$
 3. Calcular la probabilidad $\alpha(x^{(j-1)}, x^*)$
 4. Si $u \leq \alpha(x^{(j-1)}, x^*)$ fijar $x^{(j)} = x^*$
 5. De lo contrario fijar $x^{(j)} = x^{(j-1)}$
 6. Repetir [1-5] hasta alcanzar $j = J$
- Ver ilustración: [MCMC-demo](#)
- Ver ejemplo: [sección 1.5. de 01_MCMC03_GS_MH.ipynb](#)

References I

-  Huynh, Huu Tue, Van Son Lai e Issouf Soumaré (2011). *Stochastic Simulation and Applications in Finance with MATLAB Programs*. John Wiley & Sons.
-  Kim, Chang-Jin y Charles R. Nelson (1999). *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*. The MIT press.
-  Gelfand, Alan E (2000). "Gibbs sampling". En: *Journal of the American statistical Association* 95.452, págs. 1300-1304.
-  Geman, Stuart y Donald Geman (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". En: *IEEE Transactions on pattern analysis and machine intelligence* 6, págs. 721-741.
-  Chib, Siddhartha y Edward Greenberg (1995). "Understanding the Metropolis-Hastings Algorithm". En: *The American Statistician* 49.4, págs. 327-335. DOI: [10.1080/00031305.1995.10476177](https://doi.org/10.1080/00031305.1995.10476177).
-  Metropolis, Nicholas et al. (1953). "Equation of state calculations by fast computing machines". En: *The journal of chemical physics* 21.6, págs. 1087-1092.