EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2023 — MCMC.
Objetivo. Estudiar el método general de integración Monte Carlo vía cadenas de Markov (MCMC). La estrategia será construir poco a poco utilizando los principios básicos que lo componen.
Lectura recomendada: Capítulo 7 de [1]. Sección 6 de [4] y Capítulo 3 de [3] (avanzado). Si te interesa saber mas sobre programación orientada a objetos dentro del contexto de `R` puedes consultar [5].

1. INTRODUCCIÓN

El interés es poder resolver

$$\mathbb{E}[f] = \int_{\Theta} f(\theta)\, \pi(\theta|y)\, \mathrm{d}\theta\,. \tag{1}$$

Sin embargo, `no podemos generar` $\theta^{(i)} \overset{\text{iid}}{\sim} \pi(\theta|y)$.

2. MUESTREO POR ACEPTACIÓN RECHAZO

Podemos utilizar una versión estocástica de muestreo por importancia.

Para muestrear de π necesitamos utilizar una distribución sustituto (lo mismo hicimos con muestreo por importancia). Sólo que ahora permitimos rechazar muestras que no correspondan con las regiones de alta densidad de nuestra distribución objetivo. El rechazo se realiza lanzando una moneda. La tasa de éxito depende del qué tanto cubre nuestra distribución sustituto.

2.1. Implementación

Necesitamos algunas cosas. Ser capaces de `evaluar` nuestra distribución objetivo. Ser capaces de `evaluar` **y** `muestrear` de nuestra distribución de muestreo.

```
crea_mezcla ← function(weights){
  function(x){
    weights$w1 * dnorm(x, mean = -1.5, sd = .5) +
      weights$w2 * dnorm(x, mean = 1.5, sd = .7)
  }
}
objetivo ← crea_mezcla(list(w1 = .6, w2 = .4))
M        ← 3.3
```

Listing 1. *Distribución objetivo.*

2.1.1. Disclaimer*:* El objetivo del curso **no** es enseñar programación orientada a objetos. Sin embargo, permitirá abstraer los puntos importantes y concentrarnos en las ideas generales y no preocuparnos por lo detalles.

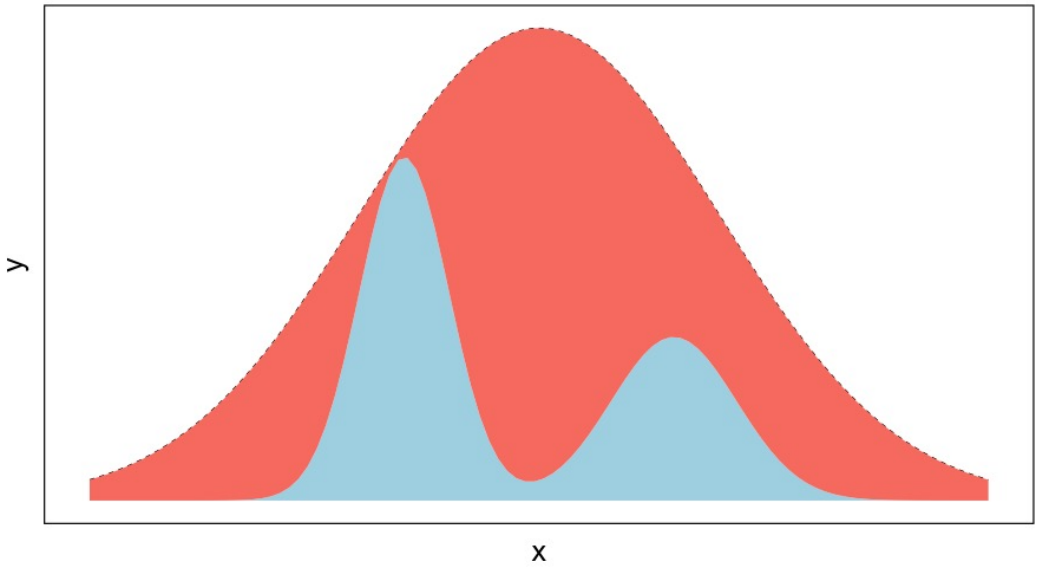


FIGURA 1. *Esquema de muestreo.*

2.1.2. Implementación de una distribución de muestreo. Recordemos que lo que queremos son dos cosas: generar números aleatorios y evaluar la función de densidad.

```
library(R6)
ModeloNormal ←
  R6Class("ProbabilityModel",
          list(
            mean = NA,
            sd = NA,
            ## Inicializador
            initialize = function(mean = 0, sd = 1){
              self$mean = mean
              self$sd   = sd
            },
            ## Muestreador
            sample = function(n = 1){
              rnorm(n, self$mean, sd = self$sd)
            },
            ## Evaluacion de densidad
            density = function(x, log = TRUE){
              dnorm(x, self$mean, sd = self$sd, log = log)
            }
          ))
```

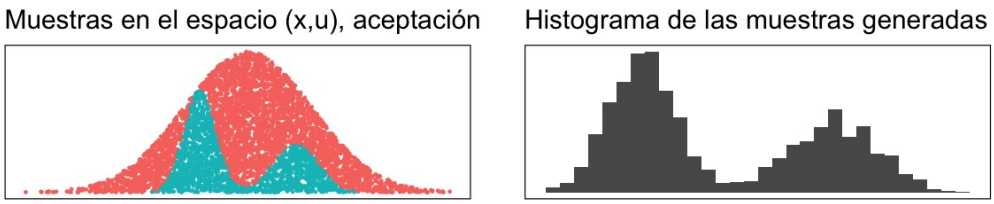
LISTING 2. *Distribución de muestreo.*

En muestreo con rechazo necesitamos definir una distribución de la cual **si podamos** generar números aleatorios. El inconveniente es, además, **conocer** qué tanto podemos inflar la densidad de nuestra propuesta para *cubrir* la distribución objetivo.

```
crea_rejection_sampling ← function(objetivo, aprox, M){
  function(niter){
    muestras ← matrix(nrow = niter, ncol = 3)
```

```
    for (ii in seq(1, niter)){
      propuesta ← aprox$sample()
      p ← objetivo(propuesta)
      g ← aprox$density(propuesta, log = FALSE)
      u ← runif(1)
      if (u < p/(M * g)) {  ## Aceptamos
        muestras[ii, 1] ← 1
      } else {              ## Rechazamos
        muestras[ii, 1] ← 0
      }
      muestras[ii, 2] ← propuesta
      muestras[ii, 3] ← u
    }
    colnames(muestras) ← c("accept", "value", "auxiliar")
    muestras
  }
}
```

LISTING 3. *Algoritmo de muestreo con rechazo.*



2.1.3. Ejercicio:

- ¿Qué pasa si M es demasiado grande? Juega con el código e interpreta los resultados.
- ¿Qué pasa si M no es suficiente para cubrir la distribución objetivo? Juega con el código e interpreta los resultados.

2.2. Propiedades

Lema (`Consistencia de muestreo por rechazo`). El método de muestreo por aceptación-rechazo genera muestras $x^{(i)}$ con $i = 1, \ldots, N$ que son independientes y distribuidas acorde a la distribución objetivo π.

Prueba. Usemos probabilidad condicional para medir

$$\pi(x|\mathsf{aceptar}) = \frac{\pi(\mathsf{aceptar}|x) \times \pi(x)}{\pi(\mathsf{aceptar})}. \tag{2}$$

3. ¿QUÉ HEMOS VISTO?

- El método Monte Carlo se puede utilizar para aproximar integrales.
- Se puede utilizar una distribución sustituto para generar números aleatorios que nos interesan.
- Podemos lanzar monedas para *filtrar* sólo los aleatorios que tengan altas probabilidades.
- Hemos utilizado el supuesto de independencia.

4. MUESTREO POR CADENAS DE MARKOV

Vamos a `relajar` el supuesto de `independencia`. Es decir, vamos a generar una secuencia de números aleatorios con cierta correlación.

4.0.1. `Definición` *[Cadena de Markov]:* Un **proceso estócastico** en tiempo discreto – una colección de variables aleatorias $X_1, X_2, \ldots$ – que satisface la propiedad de dependencia condicional

$$\mathbb{P}\left(X_{n+1} = x | X_1 = x_1, \ldots, X_n = x_n\right) = \mathbb{P}\left(X_{n+1} = x | X_n = x_n\right) , \tag{3}$$

se llama una **cadena de Markov** en tiempo discreto.

4.1. Ejemplo:

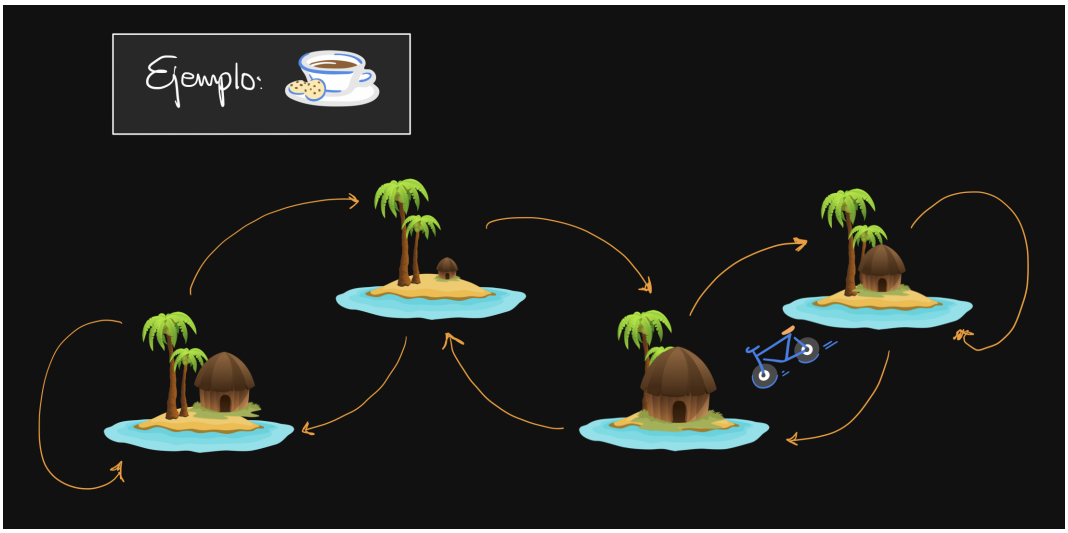


FIGURA 2. *Problema del café.*

El vendedor de galletas quiere satisfacer la demanda para acompañar un café. El vendedor:

- Viaja entre las islas.
- Decide si se queda o no se queda en la isla donde está.
- Se puede mover entre islas contiguas (a través de puentes).
- Tiene mala memoria y pregunta el número de casas en las islas aledañas (todos los días).
- Quiere visitar todas las islas y vender galletas.
- Viaja en bicicleta.

También es astuto. Sabe que en *donde haya /mucha gente venderá mas*, pero también sabe que una isla siempre lo *podría llevar a una mas grande*. Asi que a veces le convendrá viajar a una isla pequeña. Asi que utilizará el `principio de aceptación rechazo` para decidir si se moverá a la siguiente isla.

1. Lanza una moneda para decidir si se mueve a la izquierda o derecha.
2. Decide si se mueve de acuerdo al cociente de poblaciones.

4.2. Pregunta

En el contexto de nuestro problema ¿qué cambiaría si tuviera conocimiento censal del archipiélago y pudiera viajar en avión?

4.3. Modelación del tour de ventas

El vendedor se encuentra en el t -ésimo día. Supongamos que va a evaluar si se cambia a la isla de la derecha. Sea $\pi_\star$ la población de la isla propuesta y π_t la población de la isla actual. Entonces el vendedor acepta cambiar de isla con probabilidad

$$\alpha_{\text{mover}} = \frac{\pi_\star}{\pi_t}\,.$$

Nota que nunca dudará moverse a una isla mas grande. Por otro lado, entre mas parecidas sean las poblaciones de las islas mas **indeciso** será de moverse. Por definición $\alpha_{\text{mover}} \in (0, 1)$. De hecho, podemos definir la probabilidad de aceptar un viaje a otra isla por medio de

$$\alpha(t, \star) = \text{mín}\left\{1, \frac{\pi_\star}{\pi_t}\right\},$$

pues incluye los dos casos.

```
islas ← tibble(islas = 1:7, pob = c(1,2,3,4,5,4,3))
camina_isla ← function(i){ # i: isla actual
  u_izq ← runif(1) # Lanzamos volado para ver si nos vamos izq o der.
  v ← ifelse(u_izq < 0.5, i - 1, i + 1)  # Pedimos índice isla vecina.
  if (v < 1 | v > 7) { # si estas en los extremos y el volado indica salir
    return(i)
  }
  u_cambio ← runif(1) # Moneda de aceptacion de cambio
  p_cambio = min(islas$pob[v]/islas$pob[i], 1)
  if (u_cambio < p_cambio) {
    return(v) # isla destino
  }
  else {
    return(i) # me quedo en la misma isla
  }
}
```

Listing 4. *Mecanismo de cambio o permanencia desde la isla i.*

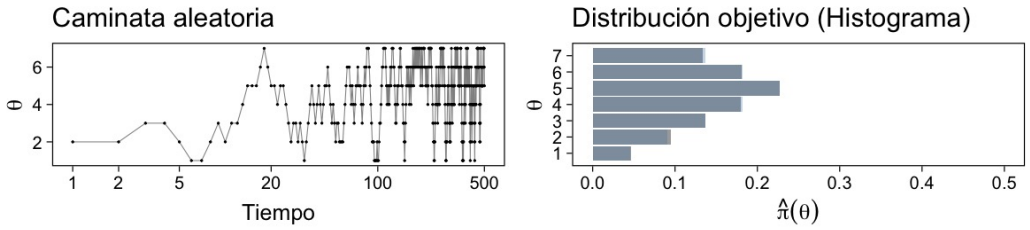


Figura 3. *Caminata aleatoria en un archipiélago de 7 islas.*

4.4. Conclusiones

- La estrategia del vendedor le permitirá, en el `largo plazo`, visitar todas las islas.
- El tiempo que pasa en cada isla[†] corresponde a la población relativa.
- Al principio, aún no representa dicha proporción.

5. GENERALIZANDO...

Supongamos que tenemos un modelo

$$Y|\mu \sim \mathsf{N}(\mu, 0.75^2)\,, \tag{4}$$
$$\mu \sim \mathsf{N}(0, 1^2)\,. \tag{5}$$

`Verifica` que bajo la observación $y = 6.25$ la distribución posterior que nos interesa es

$$\mu|y \sim \mathsf{N}(4, 0.6^2)\,. \tag{6}$$

`Vamos a suponer` que **no** sabemos muestrear de una Normal. Asi que usaremos una estrategia parecida que con el vendedor de galletas. La estrategia será:

1. Generar una propuesta $\mu_\star$ para cambiarnos de nuestro valor actual μ_t.
2. Decidir si nos movemos utilizando un cociente que tome en cuenta los pesos relativos.

5.1. Pseudo-código

- Vamos a proponer una "moneda" para lanzar la **dirección** de movimiento. Esto lo haremos con

$$\mu_\star|\mu_t \sim \mathsf{Uniforme}(\mu_t - \omega, \mu_t + \omega)\,. \tag{7}$$

- Vamos a decidir si nos movemos de acuerdo a los pesos relativos

$$\alpha(\mu_t, \mu_\star) = \text{mín}\left\{1, \frac{\pi(\mu_\star|y)}{\pi(\mu_t|y)}\right\}\,. \tag{8}$$

5.2. Desentrañando

Escribamos el cociente en términos de la densidad de la distribución posterior y simplifiquemos. ¿Qué observas?

5.3. Implementación

Veamos cómo implementarlo. Vamos a suponer una distribución de muestreo con un intervalo de longitud 2. Es decir, $\omega = 1$.

```
ModeloUniforme ←
  R6Class("ProbabilityModel",
          list(
            a = NA,
            b = NA,
            initialize = function(a = 0, b = 1){
              self$a = a
              self$b = b
            },
            sample = function(n = 1){
              runif(n, self$a, self$b)
            },
            density = function(x, log = TRUE){
              dunif(x, self$a, self$b, log = log)
            }
          ))
```

Listing 5. *Modelo de muestreo uniforme.*

```r
crea_cadena_markov ← function(objetivo, muestreo){
  function(niter){
    muestras ← matrix(nrow = niter, ncol = 2)
    ## Empezamos en algun lugar
    estado   ← muestreo$sample()
    muestras[1,1] ← 1
    muestras[1,2] ← estado
    for (ii in 2:niter){
      ## Generamos un candidato (caminata aleatoria)
      propuesta   ← estado + muestreo$sample()
      p_propuesta ← objetivo$density(propuesta, log = FALSE)
      p_estado    ← objetivo$density(estado, log = FALSE)
      ## Evaluamos probabilidad de aceptar
      if (runif(1) < p_propuesta/p_estado) {
        muestras[ii, 1] ← 1 ## Aceptamos
        muestras[ii, 2] ← propuesta
      } else {
        muestras[ii, 1] ← 0 ## Rechazamos
        muestras[ii, 2] ← estado
      }
      estado ← muestras[ii, 2]
    }
    colnames(muestras) ← c("accept", "value")
    muestras
  }
}
```

Listing 6. *Nuestra segunda cadena de Markov.*

```r
objetivo ← ModeloNormal$new(mean = 4, sd = .6)
muestreo ← ModeloUniforme$new(a = -1, b = 1)

mcmc ← crea_cadena_markov(objetivo, muestreo)
muestras ← mcmc(5000)
```

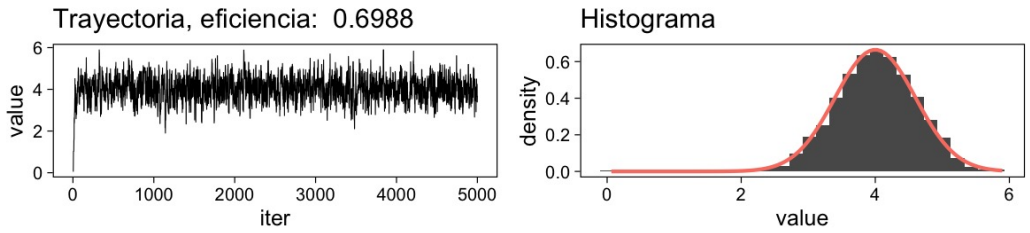


Figura 4. *Nuestra segunda cadena de Markov.*

5.3.1. Ejercicio: Sin modificar el número de iteraciones, considera cambiar la dispersión de la distribución de muestreo.

- ¿Qué observas si $\omega = 0.01$?
- ¿Qué observas si $\omega = 100$?

5.3.2. Ejercicio: Regresa a nuestro ejemplo conjugado Beta-Binomial. Considera una previa $\theta \sim \text{Beta}(2, 3)$ y una verosimilitud $Y|\theta \sim \text{Binomial}(2, \theta)$. Escribe la distribución posterior asumiendo $Y = k$.

Para este caso tenemos un ligero inconveniente. El soporte para θ es el intervalo cerrado $[0, 1]$ y utilizar una propuesta como en el caso anterior nos podría colocar (casi seguramente) fuera del intervalo. Así que lo que haremos será un pequeña modificación a cómo generamos nuestra propuesta y cómo evaluamos la probabilidad de aceptar dicha propuesta.

- Vamos a generar propuestas de la siguiente manera

$$\theta_\star|\theta_t \sim \mathsf{Beta}(\alpha, \beta)\,. \tag{9}$$

- Vamos a calcular la probabilidad de aceptar dicho movimiento a través de

$$\alpha(\theta_t, \theta_\star) = \text{mín}\left\{1, \frac{\pi(\theta_\star|y)}{\pi(\theta_t|y)} \cdot \frac{g(\theta_t)}{g(\theta_\star)}\right\}\,, \tag{10}$$

  donde g denota la densidad de la distribución de muestreo definida arriba.

5.3.3. Tarea: Modifica el código de clase para implementar este muestreador. Utiliza distintas configuraciones de a, b para la distribución de propuesta. Compara con muestras exactas del modelo posterior bajo la observación $Y = 1$.

6. EL MÉTODO METROPOLIS-HASTINGS

La forma más general que tenemos para generar una cadena de muestras es el método de Metropolis-Hastings.

- Generamos propuestas en cada iteración por medio de

$$\theta_\star|\theta_t \sim q(\theta_\star|\theta_t)\,. \tag{11}$$

- Calculamos la probabilidad de aceptar la propuesta como

$$\alpha(\theta_t, \theta_\star) = \text{mín}\left\{1, \frac{\pi(\theta_\star)}{\pi(\theta_t)} \cdot \frac{q(\theta_t|\theta_\star)}{q(\theta_\star|\theta_t)}\right\}\,, \tag{12}$$

  donde la notación hace énfasis en que este mecanismo puede generar muestras de la distribución π utilizando un generador q.

6.1. Ejercicio (3)

- Repasemos `los métodos anteriores`.
- ¿Qué pasa si desconocemos la constante de normalización de la distribución objetivo?

6.2. Distribución propuesta

El *arte* está en proponer una distribución de muestreo eficiente. Como ya hemos discutido, si no está bien calibrada podríamos tener un comportamiento no deseado. Supongamos que queremos muestrear de una $\mathsf{Gamma}(20, 100)$. Para esto veamos tres configuraciones de la distribución de muestreo que será $\mathsf{N}(\theta_t, \sigma^2)$.

```
# A tibble: 4 × 3
  configuracion  media tasa.aceptacion
  <chr>          <dbl>           <dbl>
1 Paso chico    0.0863          0.944
2 Paso grande   0.309           0.00667
3 Paso justo    0.197           0.463
4 Teorica       0.2            NA
```

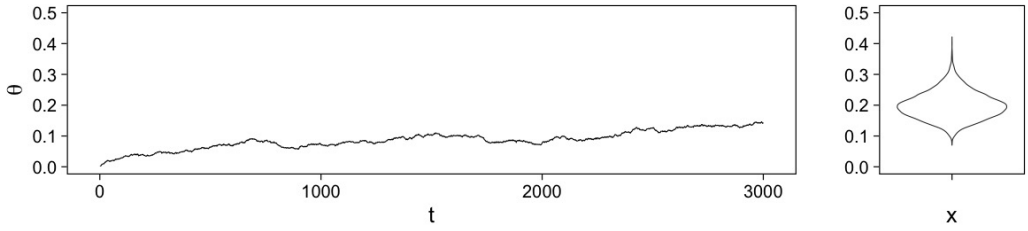


FIGURA 5. *Metropolis-Hastings en acción con un tamaño de paso muy pequeño.*

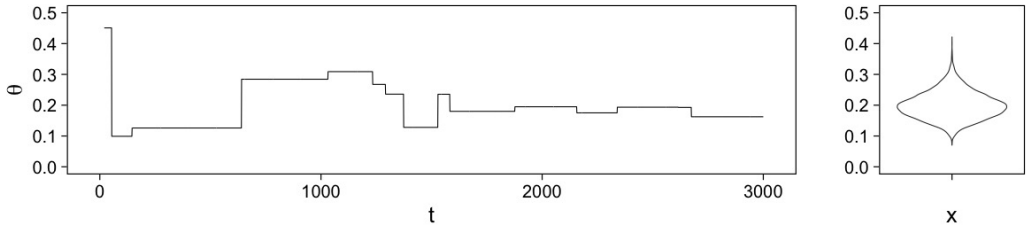


FIGURA 6. *Metropolis-Hastings en acción con un tamaño de paso muy grande.*

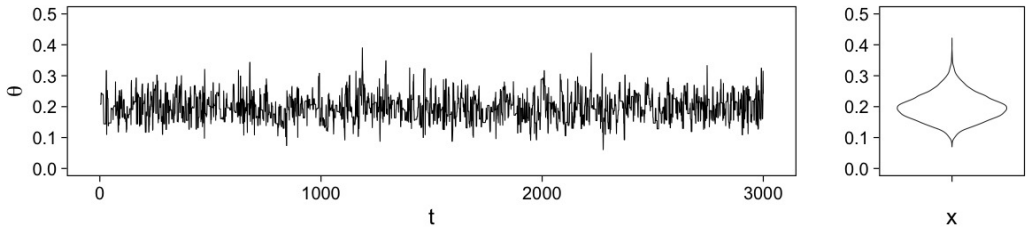


FIGURA 7. *Metropolis-Hastings en acción con un tamaño de paso* justo.

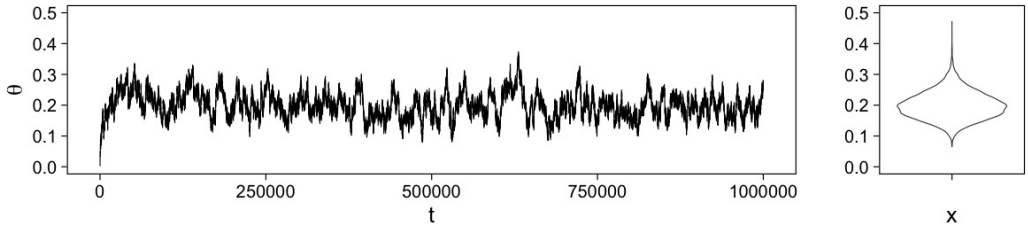


FIGURA 8. *Metropolis-Hastings en acción con un tamaño de paso* pequeño *y un periodo suficientemente amplio.*

7. EN MÁS DIMENSIONES

Consideremos la siguiente distribución objetivo

$$\theta \sim \mathsf{N}(\mathsf{m}, \mathsf{S}), \qquad \mathsf{m} = (1, 2)^\top, \qquad \mathsf{S} = \begin{pmatrix} 1 & .75 \\ .75 & 1 \end{pmatrix}, \tag{13}$$

y utilicemos el modelo de muestreo

$$\theta \sim \mathsf{N}(0, \Sigma), \qquad 0 \in \mathbb{R}^2, \qquad \Sigma = \sigma^2 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{14}$$

```
library(mvtnorm)
ModeloNormalMultivariado ←
  R6Class("ProbabilityModel",
          list(
            mean = NA,
            cov  = NA,
            initialize = function(mu = 0, sigma = 1){
              self$mean = mu
              self$cov  = sigma ▷ as.matrix()
            },
            sample = function(n = 1){
              rmvnorm(n, mean = self$mean, sigma = self$cov)
            },
            density = function(x, log = TRUE){
              dmvnorm(x, self$mean, self$cov, log = log)
            }
          ))
```

Listing 7. *Modelo de muestreo multivariado.*

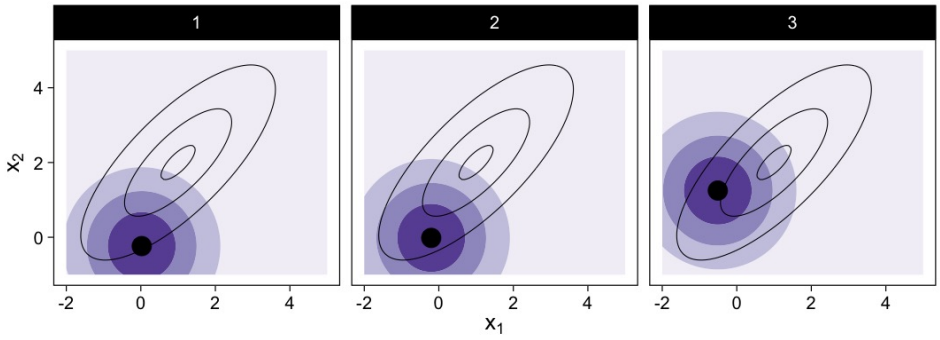


Figura 9. *Propuestas Gaussianas (morado) contra densidad objetivo (línea sólida). Tres primeras iteraciones.*

8. ¿POR QUÉ FUNCIONA?

Ya vimos cómo funciona y describimos una versión suficientemente robusta. Ahora estudiaremos el por qué esa manera de operar las transiciones nos lleva a tener un mecanismo que genera muestras de la distribución (en el largo plazo).

Para esto tenemos que preguntarnos sobre las probabilidades de transición entre dos estados. Es decir, la probabilidad de movernos al estado $\theta_\star$ condicional en estar en θ. Lo denotamos por

$$\mathbb{P}\left(\theta_{t+1} = \theta_\star | \theta_t = \theta\right). \tag{15}$$

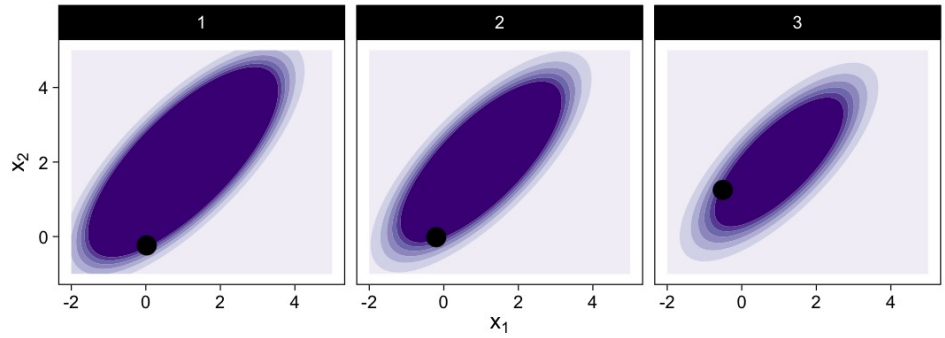


FIGURA 10. *Probabilidad de aceptación de la propuesta de transición.*

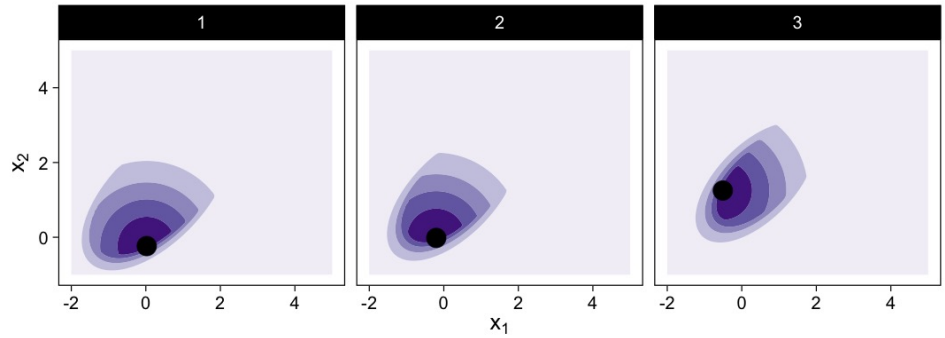


FIGURA 11. *Probabilidad de transición (morado) = probabilidad de proponer un nuevo estado multiplicada por la probabilidad de aceptar dicha transición.*

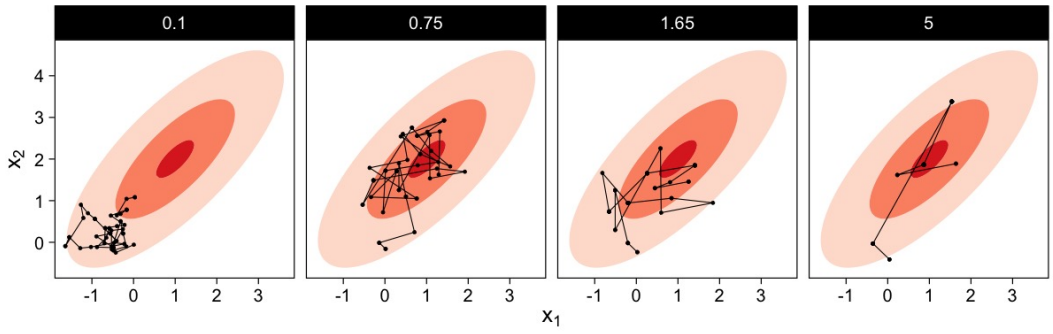


FIGURA 12. *Caminata aleatoria utilizando Metropolis-Hastings para* $\theta \in \mathbb{R}^2$.

Si el algoritmo es capaz de mantener un balance entre las probabilidades condicionales entre dos estados de acuerdo a su frecuencia relativa, entonces el algoritmo será capaz de preservar las frecuencias.

En palabras (bueno...), buscamos que

$$\frac{\mathbb{P}\left(\theta_{t+1}=\theta_\star|\theta_t=\theta\right)}{\mathbb{P}\left(\theta_{t+1}=\theta|\theta_t=\theta_\star\right)}=\frac{\pi(\theta_\star)}{\pi(\theta)}\,, \tag{16}$$

donde $\pi(\cdot)$ denota la probabilidad objetivo.

Sólo nos falta calcular la probabilidad de transición. Esto lo logramos con dos pasos: 1) generar la propuesta y 2) aceptar o rechazar la propuesta. Por lo tanto

$$\mathbb{P}\left(\theta_{t+1}=\theta_\star|\theta_t=\theta\right)=q(\theta_\star|\theta)\cdot\alpha(\theta,\theta_\star)=q(\theta_\star|\theta)\cdot\text{mín}\left\{1,\frac{\pi(\theta_\star)}{\pi(\theta)}\cdot\frac{q(\theta|\theta_\star)}{q(\theta_\star|\theta)}\right\}\,. \tag{17}$$

8.0.1. ***Definición*** *[Invarianza]:* Decimos que la distribución π es `invariante` ante un mecanismo de transición Markoviana ($p(u,v)$) si satisface que

$$\int\pi(u)\,p(u,v)\mathrm{d}u=\pi(v)\,. \tag{18}$$

> Lo que aprendemos de esto es que si tenemos un mecanismo de transición Markoviana que satisface las ecuaciones de balance entonces se mantendrá el comportamiento aleatorio de la distribución objetivo. Lo importante es que la transición preserva la distribución objetivo.

8.0.2. ***Lema*** *[Comportamiento asintótico de Metropolis-Hastings]:* El mecanismo de MH descrito anteriormente tiene como distribución límite $\pi(\cdot)$.

> Lo que aprendemos de esto es que en particular MH preserva las ecuaciones de balance. Por lo tanto, si la cadena empieza en la distribución que nos interesa, entonces se mantendrá en ese comportamiento. Estudiar formalmente las condiciones y la tasa de convergencia para llegar a esa distribución escapa a los intereses del curso y se puede encontrar un tratamiento mas cuidadoso de esto en [2]. Sin embargo, podemos entenderlo bajo el argumento que MH busca las zonas de alta densidad. Tal como el vendedor ambulante prefería de manera consistente las islas mas grandes.

REFERENCIAS

[1] A. Johnson, M. Ott, and M. Dogucu. *Bayes Rules! An Introduction to Applied Bayesian Modeling*. 2021. 1

[2] S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Springer London, London, 1993. ISBN 978-1-4471-3269-1 978-1-4471-3267-7. . 12

[3] S. Reich and C. Cotter. *Probabilistic Forecasting and Bayesian Data Assimilation*. Cambridge University Press, Cambridge, 2015. ISBN 978-1-107-06939-8 978-1-107-66391-6. 1

[4] D. Sanz-Alonso, A. M. Stuart, and A. Taeb. Inverse Problems and Data Assimilation. *arXiv:1810.06191 [stat]*, jul 2019. 1

[5] H. Wickham. *Advanced R, Second Edition*. CRC Press, may 2019. ISBN 978-1-351-20129-2. 1