EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022.
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 3 de [2]. Capítulo 7 de [1].

1. INTRODUCCION

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.*

El objetivo del curso **no** es programación orientada a objetos. Pero todxs lxs jóvenes *cool* lo utilizan.

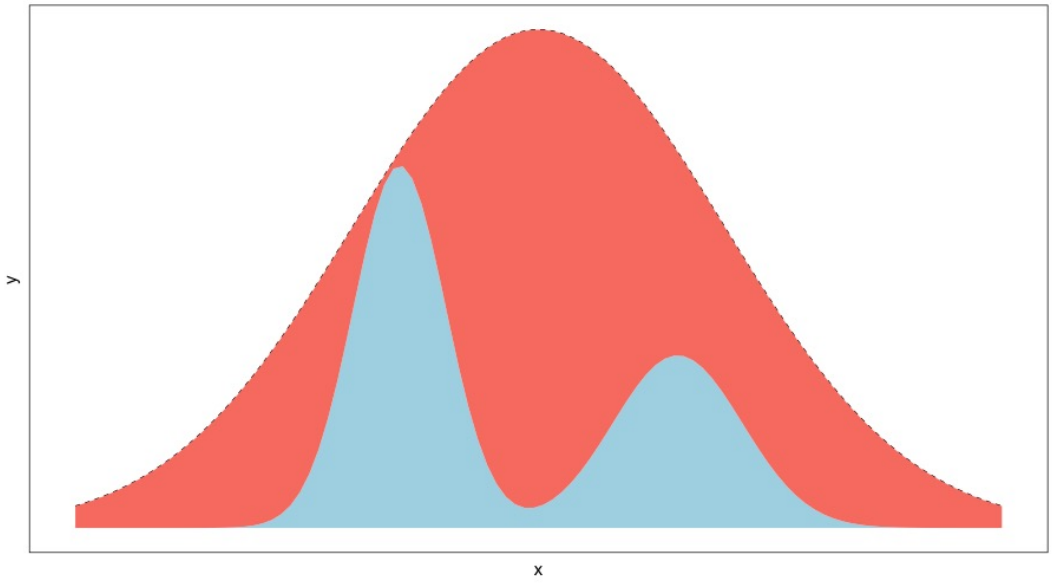


FIGURA 1. *Esquema de muestreo.*

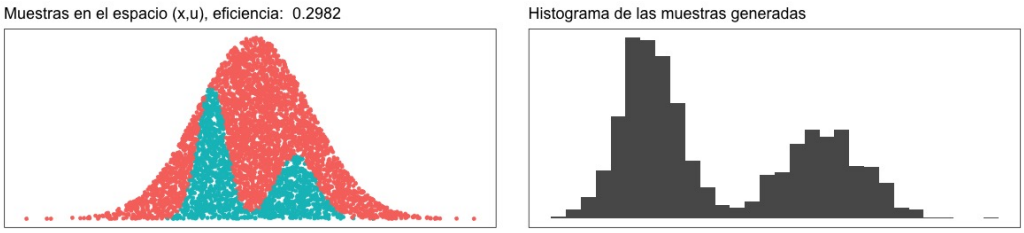
```r
library(R6)
ModeloNormal <-
  R6Class("ProbabilityModel",
          list(
            mean = NA,
            sd = NA,
            ## Inicializador
            initialize = function(mean = 0, sd = 1){
              self$mean = mean
              self$sd   = sd
            },
            ## Muestreador
            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 por aceptación 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.

```r
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, 2] <- 1
      } else {              ## Rechazamos
        muestras[ii, 2] <- 0
      }
      muestras[ii, 1] <- propuesta
      muestras[ii, 3] <- u
    }
    muestras
  }
}
```



2.2. Tarea

- ¿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.

2.3. Propiedades

Lemma (`Consistencia de muestreo por rechazo`). El método de muestreo de 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 que *esperamos* se comporte tal cómo nos interesa.

4.1. Ejemplo:

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

- Viaja entre las islas.

FIGURA 2. *Problema del café.*

- 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 cambia de isla con probabilidad

$$\alpha_{\mathsf{mover}} = \frac{\pi^\star}{\pi_t}.$$

Entre mas parecidas sean las poblaciones de las islas mas **indeciso** será de moverse. Por definición $\alpha_{\mathsf{mover}} \in (0,1)$. De hecho, podemos definir la probabilidad de viajar a otra isla por medio de

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

pues incluye los dos casos.

El tiempo que un vendedor pase en cada isla corresponderá al tamaño relativo de ellas.

```
islas <- tibble(islas = 1:5, pob = seq(60, 100, by = 10))
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  ndice  isla vecina.
  if (v < 1 | v > 5) { # 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
  }
}
```

```
pasos <- 100000; iteraciones <- numeric(pasos)
iteraciones[1] <- sample(1:5, 1) # isla inicial
for (j in 2:pasos) {
    iteraciones[j] <- camina_isla(iteraciones[j - 1])
}
caminata <- tibble(paso = 1:pasos, isla = iteraciones)
```

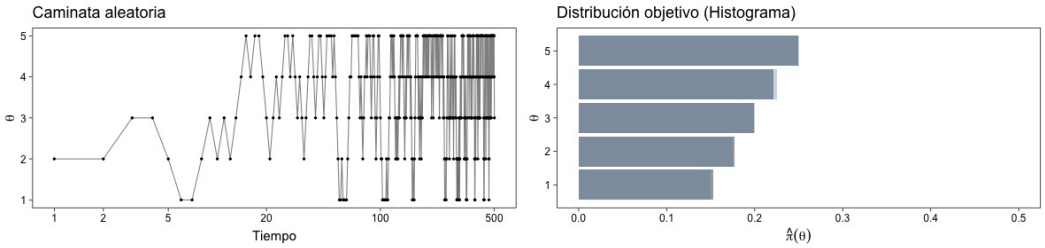


FIGURA 3. *Caminata aleatoria entre 5 islas.*

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

$$Y|\mu \sim \mathsf{N}(\mu, 0.75^2)\,, \tag{3}$$

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

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

$$\mu|y \sim \mathsf{N}(4, 0.6^2)\,. \tag{5}$$

`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

$$\mu_\star|\mu_t \sim \mathsf{Uniforme}(\omega - \mu_t, \omega + \mu_t)\,. \tag{6}$$

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

$$\alpha(\mu_t, \mu_\star) = \text{mín}\left\{1, \frac{\pi(\mu_\star|y)}{\pi(\mu_t|y)}\right\}\,. \tag{7}$$

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 3. *Modelo de muestreo uniforme.*

```
crea_cadena_markov <- function(objetivo, muestreo){
  function(niter){
    muestras <- matrix(nrow = niter, ncol = 2)
    ## Empezamos en algun lugar
    estado   <- muestreo$sample()
    muestras[1,1] <- estado
    muestras[1,2] <- 1
    for (ii in 2:niter){
      propuesta   <- estado + muestreo$sample()
      p_propuesta <- objetivo$density(propuesta, log = FALSE)
      p_estado    <- objetivo$density(estado, log = FALSE)
      if (runif(1) < p_propuesta/p_estado) {
        muestras[ii, 2] <- 1 ## Aceptamos
        muestras[ii, 1] <- propuesta
```

```
      } else {
        muestras[ii, 2] <- 0 ## Rechazamos
        muestras[ii, 1] <- estado
      }
      estado <- muestras[ii, 1]
    }
    colnames(muestras) <- c("value", "accept")
    muestras
  }
}
```

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

mcmc <- crea_cadena_markov(objetivo, muestreo)
muestras <- mcmc(5000)
```

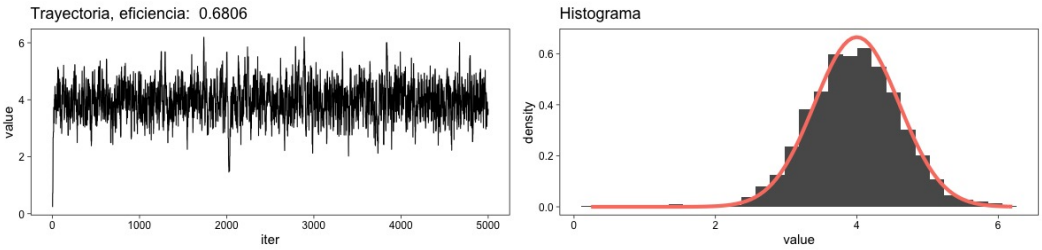


FIGURA 4. *Nuestra segunda cadena de Markov.*

5.4. Tarea (1)

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

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

5.5. Tarea (2)

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.

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 transición.

☐ Vamos a generar propuestas de la siguiente manera

$$\theta_\star|\theta_t \sim \text{Beta}(\alpha, \beta)\,. \tag{8}$$

☐ Vamos a calcular la probabilidad de transición a través de

$$\alpha(\theta_t, \theta_\star) = \text{mín}\left\{1, \frac{\pi(\theta_\star|y)}{\pi(\theta_t|y)} \cdot \frac{g(\theta_t)}{g(\theta_\star)}\right\}, \tag{9}$$

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

Modifica el código de clase para implementar este muestreador. Compara con muestras exactas del modelo posterior.

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

$$\theta_\star|\theta_t \sim q(\theta_\star|\theta_t)\,. \tag{10}$$

☐ Calculamos la probabilidad de transición como

$$\alpha(\theta_t, \theta_\star) = \text{mín}\left\{1, \frac{\pi(\theta_\star)}{\pi(\theta_t)} \cdot \frac{q(\theta_t|\theta_\star)}{q(\theta_\star|\theta_t)}\right\}\,, \tag{11}$$

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

☐ Repasemos `los métodos anteriores`.

REFERENCIAS

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

[2] 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