

Informe_Met-Has

La estadística bayesiana se usa para poder dar respuestas a incertidumbres sobre parámetros que afectan a distintas situaciones. Se usan dos fuentes de información para ello. La primera es lo que uno cree conocer sobre los parámetros previamente al realizar un experimento, es decir, la información *a priori*, y la segunda, los resultados que se obtienen al realizarlo, que se lo denomina como la *verosimilitud*. Juntando estas informaciones se obtiene lo que se conoce como la distribución *a posteriori* del o los parámetros. Pero para poder encontrar respuesta al desconocimiento sobre el tema se debe poder sacar muestras de la distribución *a posteriori*. Para realizar esto, una de las técnicas que se pueden utilizar es Metropolis-Hastings.

Metropolis-Hastings

Esta tecnica es un método de Montecarlo basado en cadenas de Markov que consiste, a grandes rasgos, en recorrer los distintos valores de una distribución de probabilidad generando un secuencia de valores posibles x de dicha distribución $x(1), x(2), \dots, x(S)$ donde para obtener $x(i+1)$ usamos $x(i)$. De esta manera se quiere que el conjunto de los valores simulados se aproximen lo máximo posible a una muestra real de la distribución.

Esta técnica funciona de la siguiente manera:

1. En la iteración i estamos en el valor $x(i)$
2. En función del valor actual $x(i) = x$, proponemos un nuevo valor x' en función de $q(x' | x)$ siendo q la distribución de salto que uno debe proponer, que sera la que presente los puntos de salto posibles.
3. Decidimos si vamos a la nueva ubicación $x(i+1) = x'$ o si nos quedamos en $x(i+1) = x$:
 - i. Calcular la probabilidad de salto:
 - $\alpha_{x \rightarrow x'} = \min(1, f(x')f(x))$
 - ii. Saltar a x' con probabilidad $\alpha_{x \rightarrow x'}$:

$$x^{(i+1)} = \begin{cases} x' & \text{con probabilidad } \alpha_{x \rightarrow x'} \\ x & \text{con probabilidad } \alpha_{x \rightarrow x'} \end{cases}$$

La función para utilizar el método es la siguiente:

```
# n: Tamaño de la cadena que se quiere generar

# p_inicial: Primer punto con el que se inicial la cadena.
# Es igual a 0 por defecto.

# v_random: Utilizar si se quiere empezar la cadena con un punto aleatorio.
# Introducir un vector de dos números para obtener un número aleatorio
# entre ellos dos.

# d_objetivo: Función que calcula la densidad de la función que se
# quiere muestrear.

# r_propuesta: Función que genera un punto de la distribución de salto con la
# media sin especificar. Por defecto, se utiliza una normal estándar.

# d_propuesta: Función que devuelve la densidad en un punto de la distribución
# de salto con la media sin especificar. Por defecto, se utiliza una normal
# estándar.

sample_mh_1d = function(n, p_inicial = 0, d_objetivo, r_propuesta = rnorm,
                        d_propuesta = dnorm, v_random = NA){
  stopifnot(n > 0)
  muestras = numeric(n)
  #Selección del primer valor de la cadena
  if (length(v_random) == 2){
    muestras[1] = runif(1,min(v_random),max(v_random))
  } else {
    muestras[1] = p_inicial
  }
  #Generación de todos los n-1 puntos siguientes de la cadena
  for (i in 2:n) {
    #Proposición del nuevo valor
    puntos = valor_salto_1d(muestras, r_propuesta, i)
    p_actual = puntos[1]
    p_nuevo = puntos[2]
    #Cálculo de la probabilidad de salto
    prob = salto_1d(p_actual, p_nuevo, d_objetivo, d_propuesta)
    #Saltamos?
    salto = test_salto_1d(prob)
    #Salto
    muestras[i] = p_gen_1d(salto, p_actual, p_nuevo)
  }
}
```

```

    }
    return(muestras)
}

#Donde las funciones utilizadas son:

#Proposición del nuevo valor
valor_salto_1d = function(muestras, r_propuesta, i){
  p_actual = muestras[i-1]
  p_nuevo = r_propuesta(p_actual)
  puntos = c(p_actual, p_nuevo)
  return(puntos)
}

#Cálculo de la probabilidad de salto
salto_1d = function(p_actual, p_nuevo, d_objetivo, d_propuesta) {
  f_nuevo = d_objetivo(p_nuevo)
  f_actual = d_objetivo(p_actual)
  q_actual = d_propuesta(p_actual, p_nuevo)
  q_nuevo = d_propuesta(p_nuevo, p_actual)
  prob = (f_nuevo*q_actual) / (q_nuevo*f_actual)
  alfa = min(1, prob)
  return(alfa)
}

#Saltamos?
test_salto_1d = function(alfa){
  result = rbinom(1,1,alfa)
  return(result)
}

#Salto
p_gen_1d = function(result, p_actual, p_nuevo) {
  if (result){
    return(p_nuevo)
  } else {
    return(p_actual)
  }
}

```

Aplicación de Metropolis-Hastings en una dimensión

Distribución de Kumaraswamy

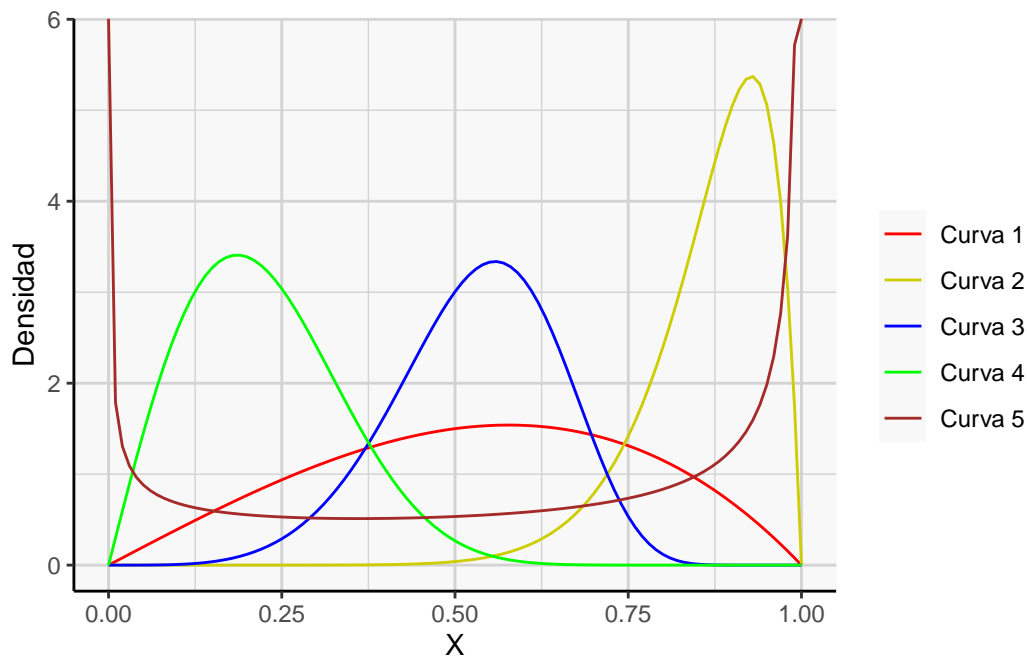
La distribución de Kumaraswamy es una distribución de probabilidad continua que se utiliza para modelar variables aleatorias con soporte en el intervalo $(0, 1)$. Su función de densidad se parece a la beta pero su expresión matemática resulta ser de cómputo más sencillo. Dicha función es:

$$p(x \mid a, b) = abx^{a-1}(1-x)^{b-1} \quad \text{con } a, b > 0$$

Para observar que forma tiene, se grafica su función de densidad para 5 valores de a y b .

Valores					
	Curva 1	Curva 2	Curva 3	Curva 4	Curva 5
a	2	10	5	2	0.5
b	2	2	15	15	0.33

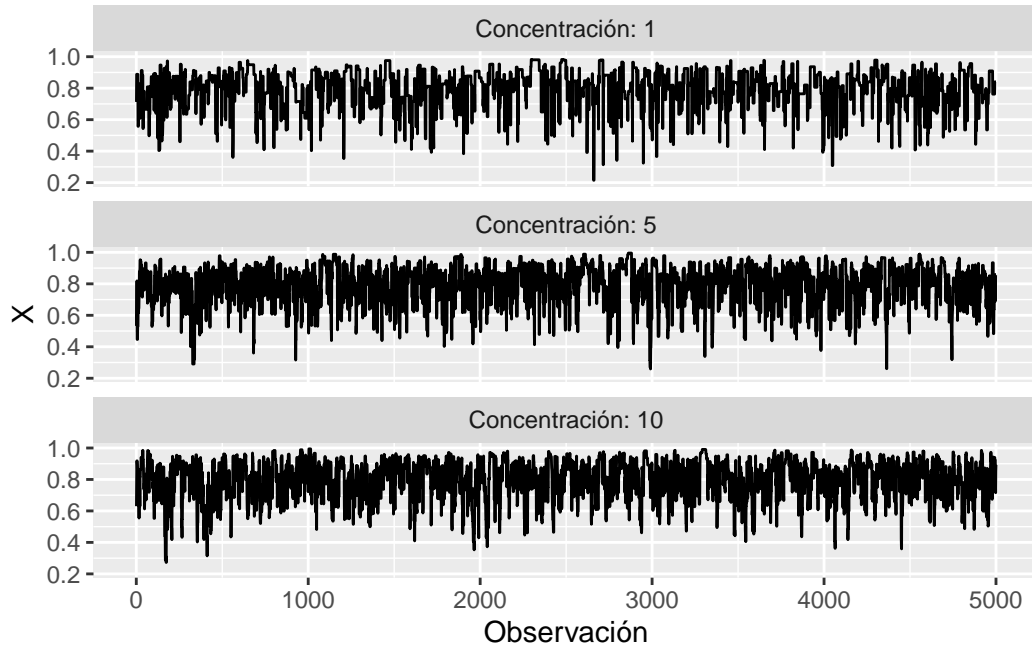
Tabla 1: Parametros de las distribuciones



Es una función sencilla donde su forma de densidad resulta muy flexible al variar a y b por lo que se puede usar fácilmente para la representación de la probabilidad *a priori* de parámetros que estén entre 0 y 1.

Usando la función de Metropolis-Hastings se obtiene una muestra de 5000 observaciones de la distribución de Kumaraswamy con $a = 6$ y $b = 2$. Se utiliza una distribución beta reparametrizada en su media y su concentración para proponer los nuevos valores y para obtener el primer valor de la cadena se genera aleatoriamente con igual probabilidad un punto entre 0 y 1.

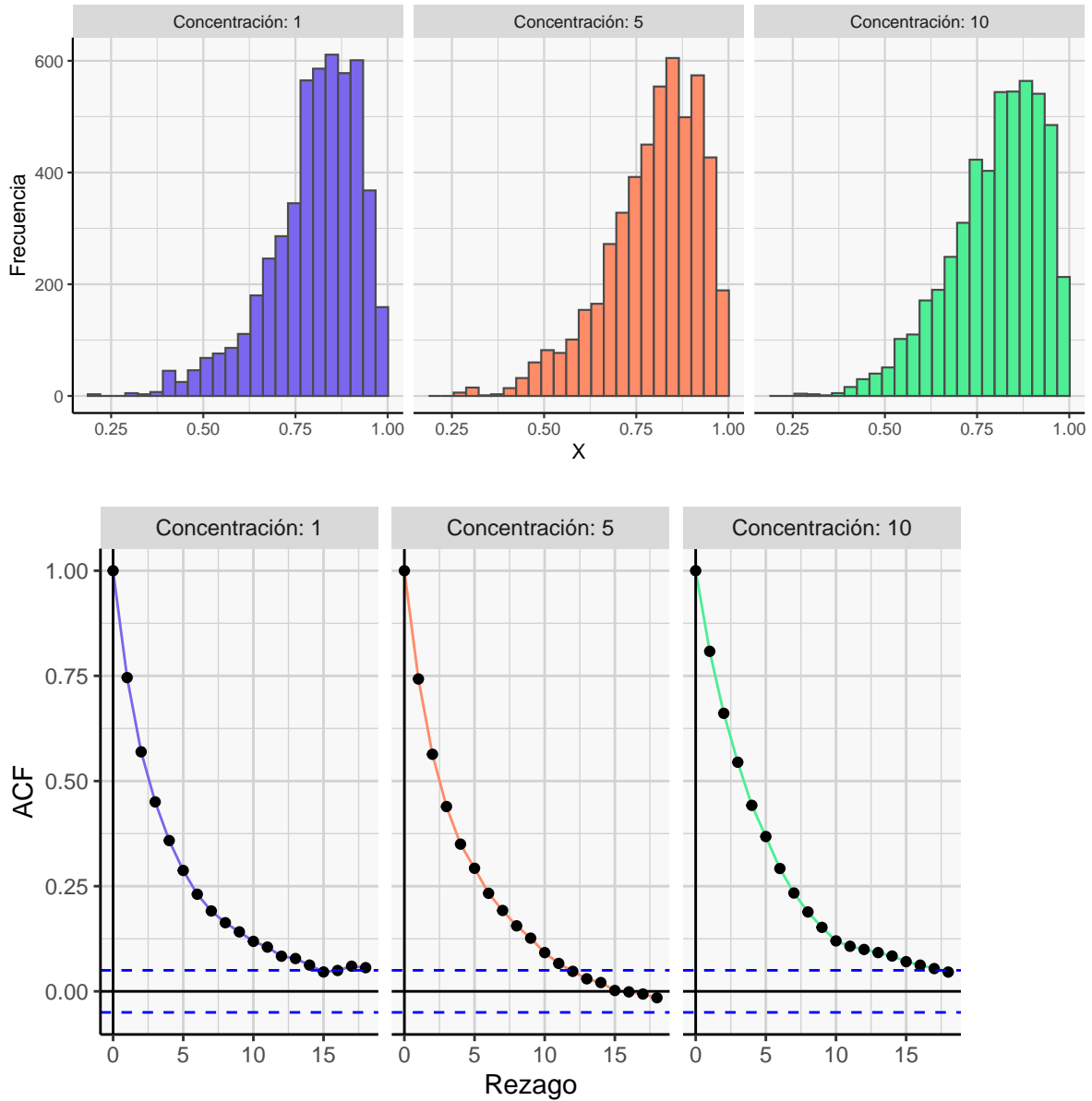
Se usan tres valores de concentración diferentes para evaluar la eficiencia del método.



Concentración	1	5	10
Tasa de aceptación	0.219	0.509	0.618

Tabla 2: Evaluación de las cadenas de Kumaraswamy

En la figura x se observa que el método funciona mejor con una concentración alta porque oscila más veces recorriendo la densidad de una manera más eficiente. Esto se verifica observando las tasas de aceptación de los saltos en la tabla 2.



Las tres cadenas convergen a la misma distribución y la cantidad de rezagos que tardan hasta llegar a una autocorrelación despreciable (<0.05) es baja y similar en todas por lo que el método funciona correctamente aún variando la dispersión de la distribución del salto.

	Media	Percentil 5	Percentil 95
X_1	0.7987838	0.5508091	0.9525783
X_5	0.7972954	0.5390545	0.9631994
X_{10}	0.8007066	0.5583269	0.9622339
$\text{logit}(X_1)$	1.5742044	0.2039405	3.0000926
$\text{logit}(X_5)$	1.5952465	0.1565376	3.2647543
$\text{logit}(X_{10})$	1.6198775	0.2343745	3.2378466

Tabla 3: Medidas descriptivas estimadas de las distribuciones

Donde:

$$\text{logit}(X) = \log\left(\frac{X}{1-X}\right) = \log(X) - \log(1-X)$$

Todas las medidas descriptivas de X son parecidas para las distintas cadenas generadas, pero si se le aplica la transformación *logit* a X esas diferencias mínimas se agrandan un poco.

Aplicación de Metropolis-Hastings en dos dimensiones

La verdadera utilidad del método presentado anteriormente se encuentra en la extracción de muestras de distribuciones multivariadas. Igual que la situación anterior, no es necesario que las funciones que se quieren muestrear estén normalizadas. Se presentan dos situaciones en las que se puede aplicar la técnica. La primera, un ejemplo simple donde se obtienen muestras de la distribución normal bivariada donde este método funciona bien y la segunda, una función difícil de obtener muestras válidas en donde se pueden ver las fallas de Metropolis-Hastings.

Para realizar estas muestras, no sirve la función presentada en la primera parte del informe. Por eso se tiene la necesidad de modificar dicha función.

```
# Se utiliza como distribución de salto propuesto una normal bivariada

# n: Tamaño de la cadena que se quiere generar

# p_inicial: Primer punto con el que se inicial la cadena. Es igual a 0 por
# defecto.

# d_objetivo: Función que calcule la densidad de la función que se quiere
# muestrear.

# matrix_var: Matrix de varianzas y covarianzas de la función de salto normal
# bivariada.
```

```

sample_mh_2d = function(n, p_inicial , d_objetivo, matrix_var = diag(1,2)){
  stopifnot(n > 0)
  muestras = matrix(nrow = n, ncol = 2)
#Selección del primer valor de la cadena
  muestras[1,] = p_inicial

  #Generación de todos los n-1 puntos siguientes de la cadena
  for (i in 2:n) {
    #Proposición del nuevo valor
    puntos = valor_salto_2d(muestras, matrix_var, i)
    p_actual = puntos[1:2]
    p_nuevo = puntos[3:4]

    #Cálculo de la probabilidad de salto
    prob = salto_2d(p_actual, p_nuevo, d_objetivo, matrix_var)

    #Saltamos?
    salto = test_salto_2d(prob)

    #Salto
    muestras[i,] = p_gen_2d(salto, p_actual, p_nuevo)
  }
  if (length(p_inicial) == 1) {
    return(as.vector(muestras))
  }
  return(muestras)
}

#Proposición del nuevo valor

valor_salto_2d = function(muestras, matrix_var, i){
  p_actual = muestras[i-1,]
  p_nuevo = rmvnorm(1, mean = p_actual, sigma = matrix_var)
  puntos = c(p_actual, p_nuevo)
  return(puntos)
}

#Cálculo de la probabilidad de salto

salto_2d = function(p_actual, p_nuevo, d_objetivo, matrix_var) {
  f_nuevo = d_objetivo(p_nuevo)
  f_actual = d_objetivo(p_actual)

```



```

q_actual = dmnorm(p_actual, p_nuevo, sigma = matrix_var)
q_nuevo = dmnorm(p_nuevo, p_actual, sigma = matrix_var)
prob = (f_nuevo*q_actual) / (q_nuevo*f_actual)
alfa = min(1, prob)
return(alfa)
}

#Saltamos?

test_salto_2d = function(alfa){
  result = rbinom(1,1,alfa)
  return(result)
}

#Salto

p_gen_2d = function(result, p_actual, p_nuevo) {
  if (result){
    return(p_nuevo)
  } else {
    return(p_actual)
  }
}

```

Distribución normal bivariada

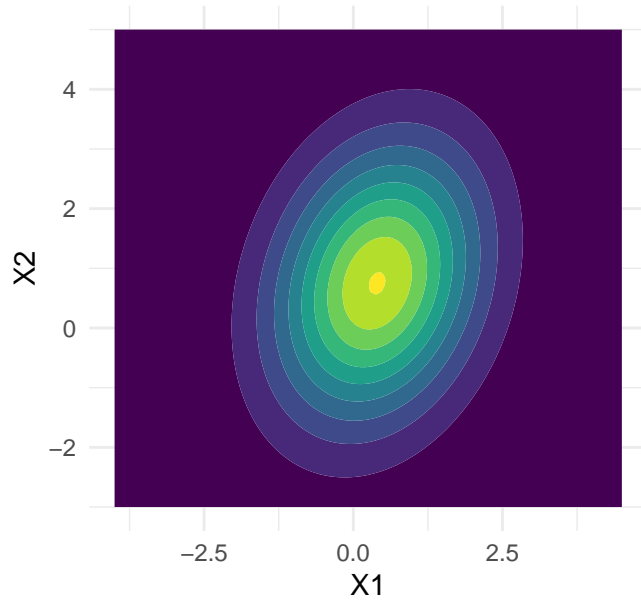
La distribución a muestrear en este caso será una normal bivariada.

$$p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

Donde μ es el vector de esperanzas y Σ la matrix de varianzas y covarianzas.

Se controla que tan buenas son las muestras obtenidas a través de la evaluación de la cadena generada por el método y de qué tanto se asimilan las probabilidades estimadas por la distribución muestral obtenida a las reales. La normal bivariada de interés viene dada por los siguientes parámetros.

$$\mu^* = \begin{bmatrix} 0.4 \\ 0.75 \end{bmatrix} \quad \Sigma^* = \begin{bmatrix} 1.35 & 0.4 \\ 0.4 & 2.4 \end{bmatrix}$$



Usando la función de Metropolis-Hastings se obtiene una muestra de 5000 observaciones. Para obtener las muestras, primero se debe seleccionar la matriz de covarianzas de la distribución de salto. Esto se logra evaluando el número de muestras efectivas de las dos dimensiones de las muestras generadas por separado, variando los valores en la matriz y quedándose con la que tenga el mayor número de muestras efectivas totales.

El cálculo realizado es:

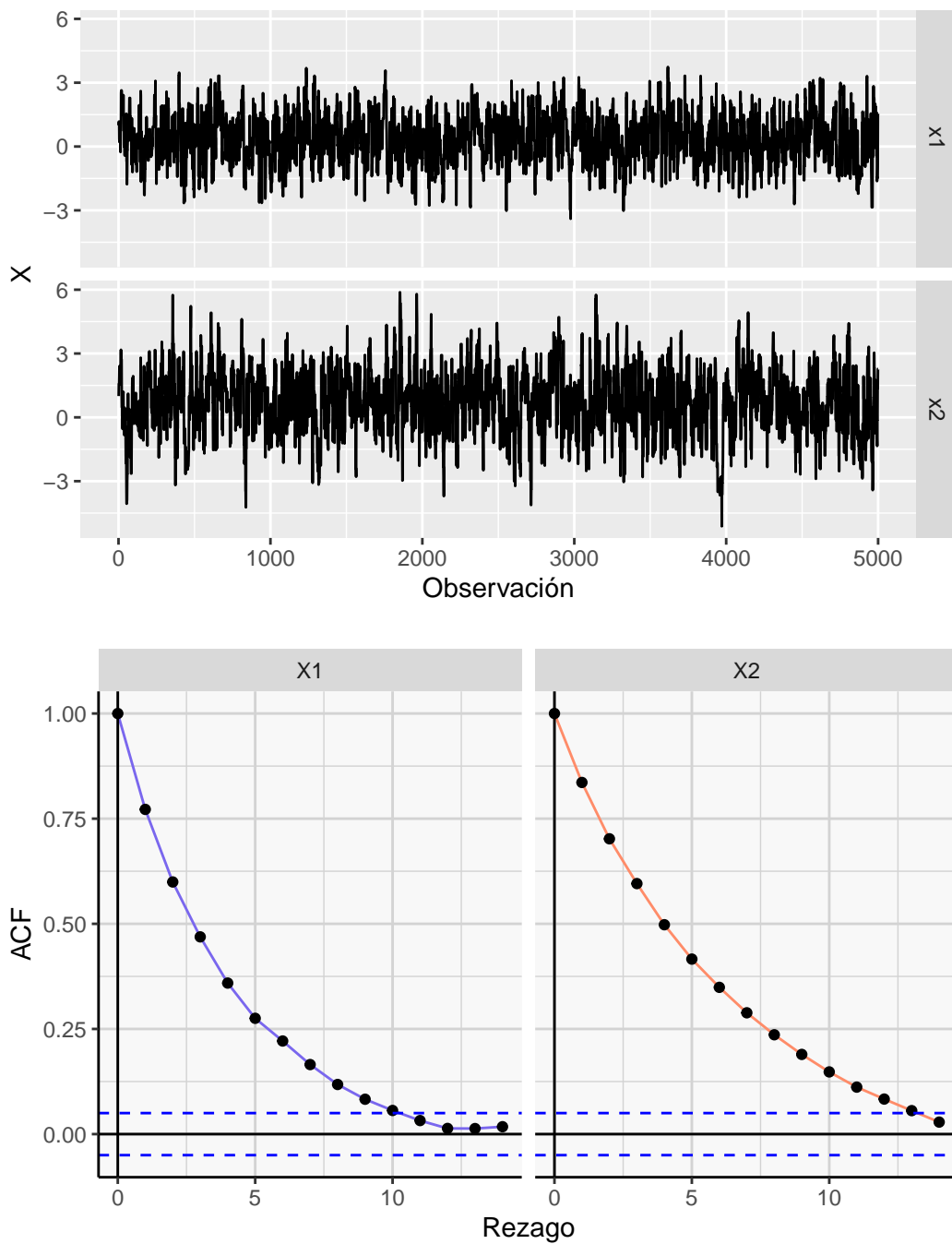
$$N_{eff} = \frac{S}{1 + 2 \sum_{k=1}^N ACF(k)}$$

Donde S es el número de unidades generadas en la cadena, $ACF(k)$ la autocorrelación de la cadena con k rezagos y N la cantidad de rezagos hasta que $ACF(k) < 0.05$ por primera vez.

Los valores de la matriz de varianza evaluados son:

$$\Sigma_{Salto} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \quad \sigma_1^2 \in [0.85, 1.85]; \quad \sigma_2^2 \in [1.90, 2.90]; \quad \sigma_{12} \in [0.20, 0.60]$$

La combinación de dichos valores con el mayor número de muestras efectivas totales es 1.85, 2.15 y 0.5 para σ_1^2 , σ_2^2 y σ_{12} respectivamente.



Las cadenas en la figura x se ven relativamente aleatorias y parece recorrer el soporte de la función objetivo, que es indicio de una autocorrelación baja entre las observación generadas. Esto se aprecia en la figura x+1, donde la autocorrelación de ambas cadenas disminuye rápidamente al aumentar el rezago. El número de muestras efectivas para x_1 y x_2 es 684.7769494

y 496.2193513 respectivamente. Estos son valores relativamente grandes, que dan una imagen positiva de la calidad de la muestra generada.

Se calculan tres probabilidades usando las muestras generadas y se comparan con las reales para ver que tan precisas son.

Probabilidades	$\hat{\mathbf{X}}$	\mathbf{X}	Error relativo
$\Pr(X_1 > 1, X_2 < 0)$	0.0624	0.0683	0.0857
$\Pr(X_1 > 1, X_2 > 2)$	0.0834	0.087	0.0415
$\Pr(X_1 > 0.40, X_2 > 0.75)$	0.2814	0.2857	0.0149

Tabla 4: Evaluación de las probabilidades estimadas con las muestras generadas por M-H

En la tabla 4 se observa que cuando se quiere estimar la probabilidad de un evento correspondiente a un área lejana de la masa central de probabilidades, la precisión de la aproximación disminuye, es decir, aumenta el error. Aún así, el método logra obtener una muestra aceptable al no ser tan grande el error relativo en dichos casos.

Función de Rosenbrock

El “valle de Rosenbrock”, “banana de Rosenbrock” o simplemente función de Rosenbrock es una función matemática utilizada comunmente en problemas de optimización y prueba para algoritmos de optimización numérica.

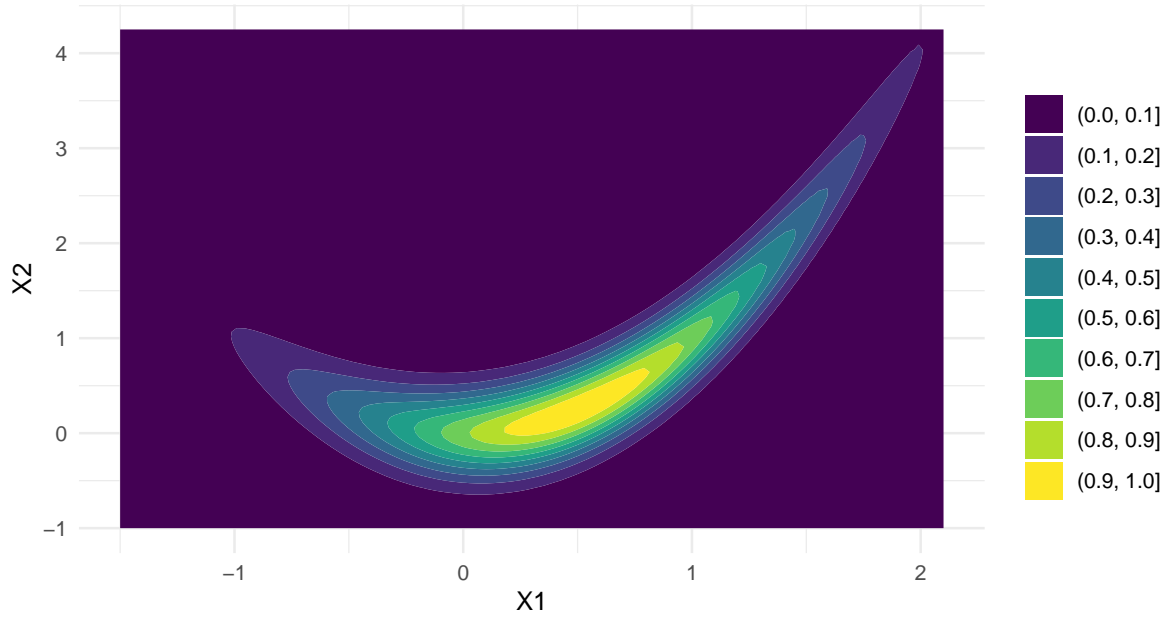
La función está definida por:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2$$

En estadística bayesiana, en ciertos casos, se obtiene una densidad de distribución *a posteriori* que toma una forma semejante a dicha función.

Un ejemplo de esto es la función p^* :

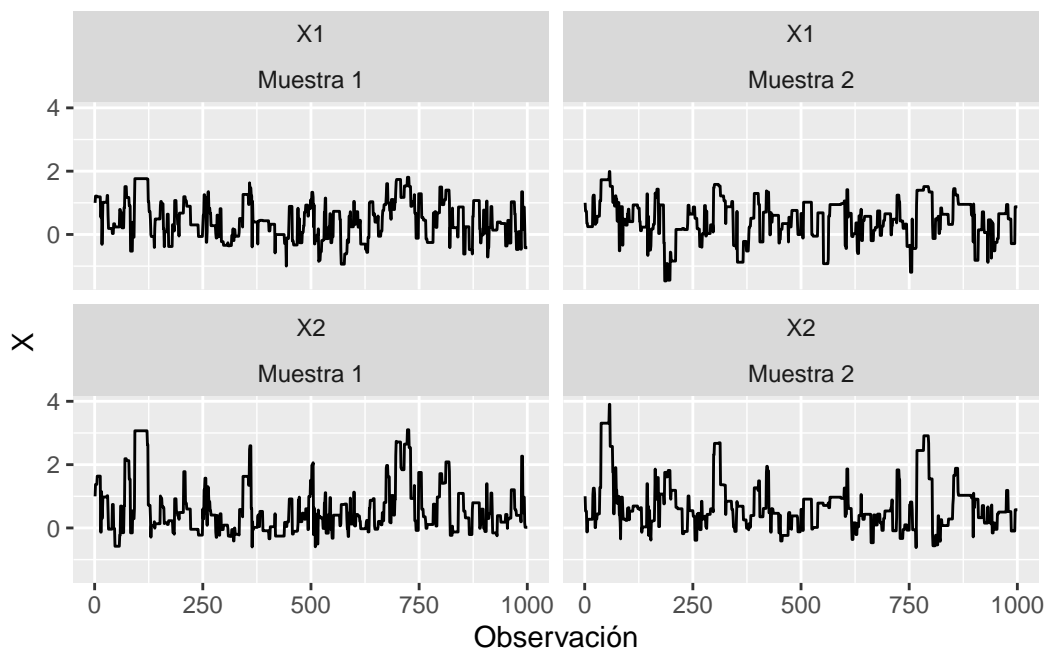
$$p^*(x_1, x_2 \mid a, b) = \exp \{ - [(a - x_1)^2 + b(x_2 - x_1^2)^2] \}$$



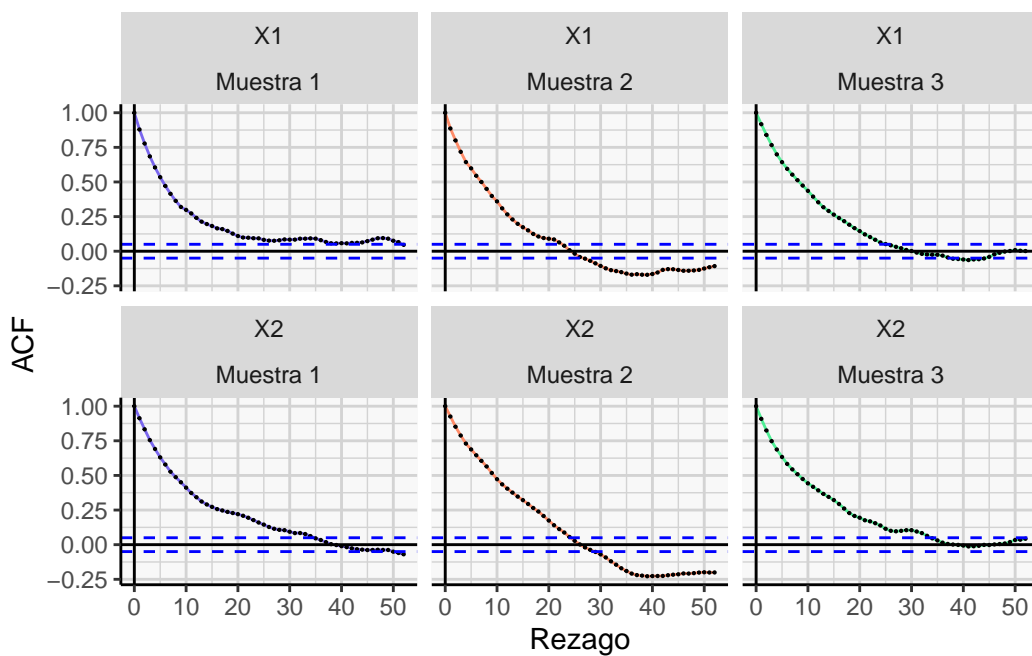
Se quiere probar que tan bien puede Metropolis-Hastings generar muestras de dicha función de probabilidad y para ello se proponen distintas matrices de varianzas y covarianzas de la distribución de salto propuesta evaluando el rendimiento obtenido con cada una de ellas. Se utilizan los valores 0.5 y 5 para los parámetros a y b respectivamente para la generación de las 1000 observaciones.

	Muestra 1	Muestra 2	Muestra 3
σ_1^2	1	1	1
σ_2^2	1	1	2
σ_{12}	0.7	0.3	0

Tabla 5: Valores de las matrices de varianzas de cada muestra generada.



En comparación con la figura x del caso Normal bivariado, se puede notar que las cadenas de la figura x tienen una tasa de aceptación menor, generando más estancamientos en un valor y menos picos. Además, el número efectivo de muestras es menor.



En la figura x, todas las cadenas bajan lentamente hasta tener una autocorrelación despreciable, pero la segunda lo hace de forma más rápida.

	Muestra 1	Muestra 2	Muestra 3
X1	49.2386648	60.0288002	50.1060959
X2	44.0839863	46.3958397	43.7480022

Tabla 6: Número efectivo de muestras

El número de muestras efectivas para cada cadena es bajo. Esto tiene sentido, al ver los resultados de la figura x y x.

Se calculan tres probabilidades usando las muestras generadas y se comparan con las reales para ver que tan precisas son.

Probabilidades	$\hat{\mathbf{X}}$	\mathbf{X}	Error relativo
$\Pr(0 < X_1 < 1, 0 < X_2 < 1)$	0.323	0.553	0.4158799
$\Pr(-1 < X_1 < 0, 0 < X_2 < 1)$	0.147	0.771	0.8092427
$\Pr(1 < X_1 < 2, 2 < X_2 < 3)$	0.051	0.09	0.4347672

Tabla 7: Evaluación de las probabilidades estimadas con las muestras generadas por M-H

Los errores relativos de la tabla 7 son muy grande, por lo que da un indicio de lo mala que es la estimación de la probabilidad. Por esto, y todo lo comentado anteriormente, es razonable pensar que la distribución de las muestras generadas por Metropolis-Hastings difieren en gran medida de la verdadera distribución de p^* .

Conclusiones

Metropolis-Hastings es un método útil en ciertas ocasiones. Es simple, fácil de programar y entender, y además se sabe que cuando el número de muestras tiende a infinito, la distribución de probabilidad de la secuencia tiende a la verdadera función objetivo, pero en casos donde la función de densidad de la distribución sea compleja es poco preciso. Para obtener una muestra con buena representatividad, es decir, que haya explorado el rango completo de la distribución, se requeriría una muestra de tamaño grande. Entonces, el método no sería de utilidad al requerir tantas iteraciones para llegar a un resultado aceptable. Además, al aumentar la dimensión se incrementa el tiempo que le toma a la computadora realizar los cálculos. Esto, combinado con el hecho de que hay métodos más eficientes hace a Metropolis-Hastings un método no recomendado. Para obtener muestras de una función de densidad, es preferible el uso de otros métodos como Hamiltonian-Montecarlo.