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-Hasting.

Metropolis-Hasting

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 $x(1), x(2), ..., x(S)$ 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 $q(x' \mid x)$ 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:
      - $\alpha_{x \to x'} = min(1, f(x')f(x))$
   ii. Saltar a x' con probabilidad $\alpha_{x \to x'}$:

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

La función para utilizar el metodo 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 ve

# d_objetivo: Función que calcule 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 espe

# d_propuesta: Función que devuelve la densidad en un punto de la distribución de salto con

sample_mh_1d = function(n, p_inicial = 0, d_objetivo, r_propuesta = rnorm, d_propuesta = dno
  stopifnot(n > 0)
  muestras = numeric(n)
  #Seleccion 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]

    #Calculo 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)
}

#Calculo 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-Hasting en una dimensión

Distribución de Kumaraswamy

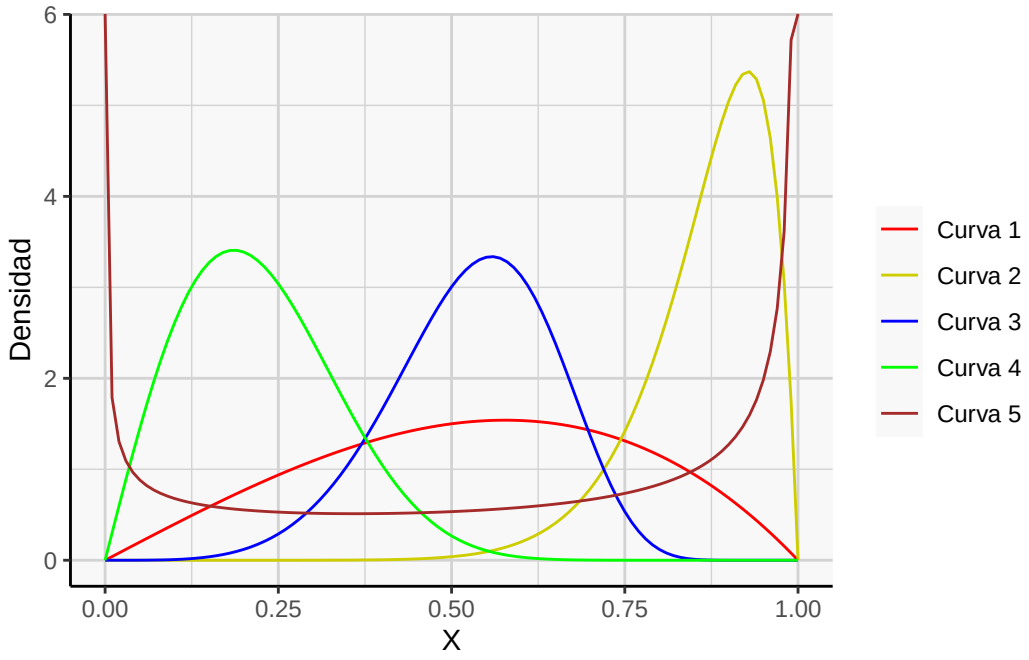
La distribución de Kumaraswamy es una distribución de probabilidad continua que se utiliza para modelar variables aleatorias con soporte en el intervalo $(0, 1)$. Su función de densidad se parece a la beta pero su expresión matematica resulta ser de computo mas sencillo. Dicha función es:

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

Para observar que forma tiene, se grafica su funcion 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

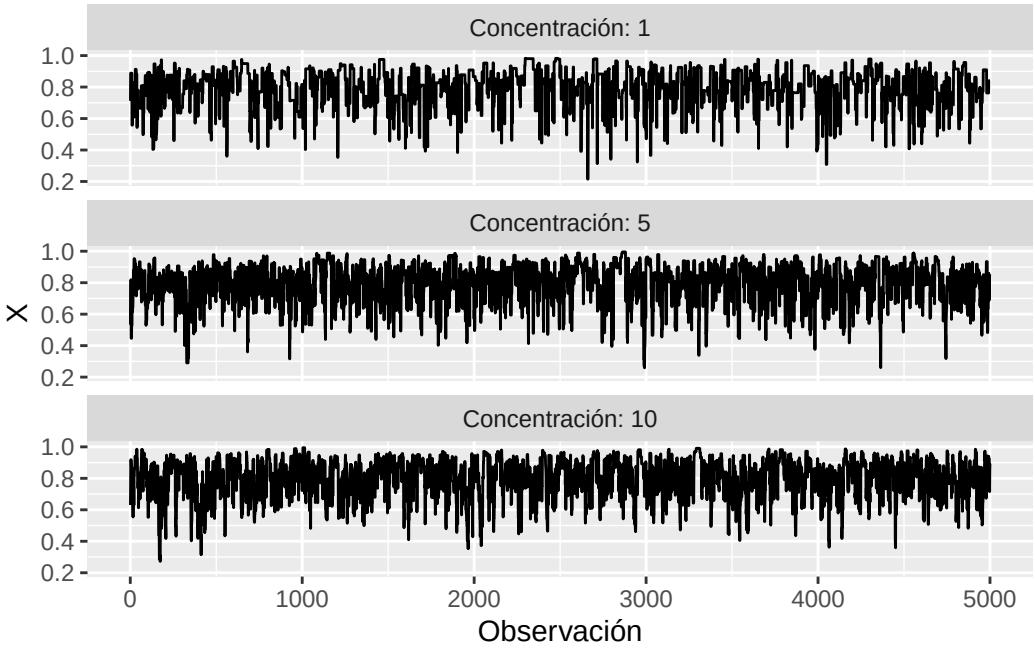
Table 1: Table to test captions and labels.



Es una funcion sencilla donde su forma de densidad resulta muy flexible al variar a y b por lo que se puede usar facilmente para la representacion de la probabilidad a *priori* de parametros que esten entre 0 y 1.

Usando la función de Metropolis-Hasting se obtiene un 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 .

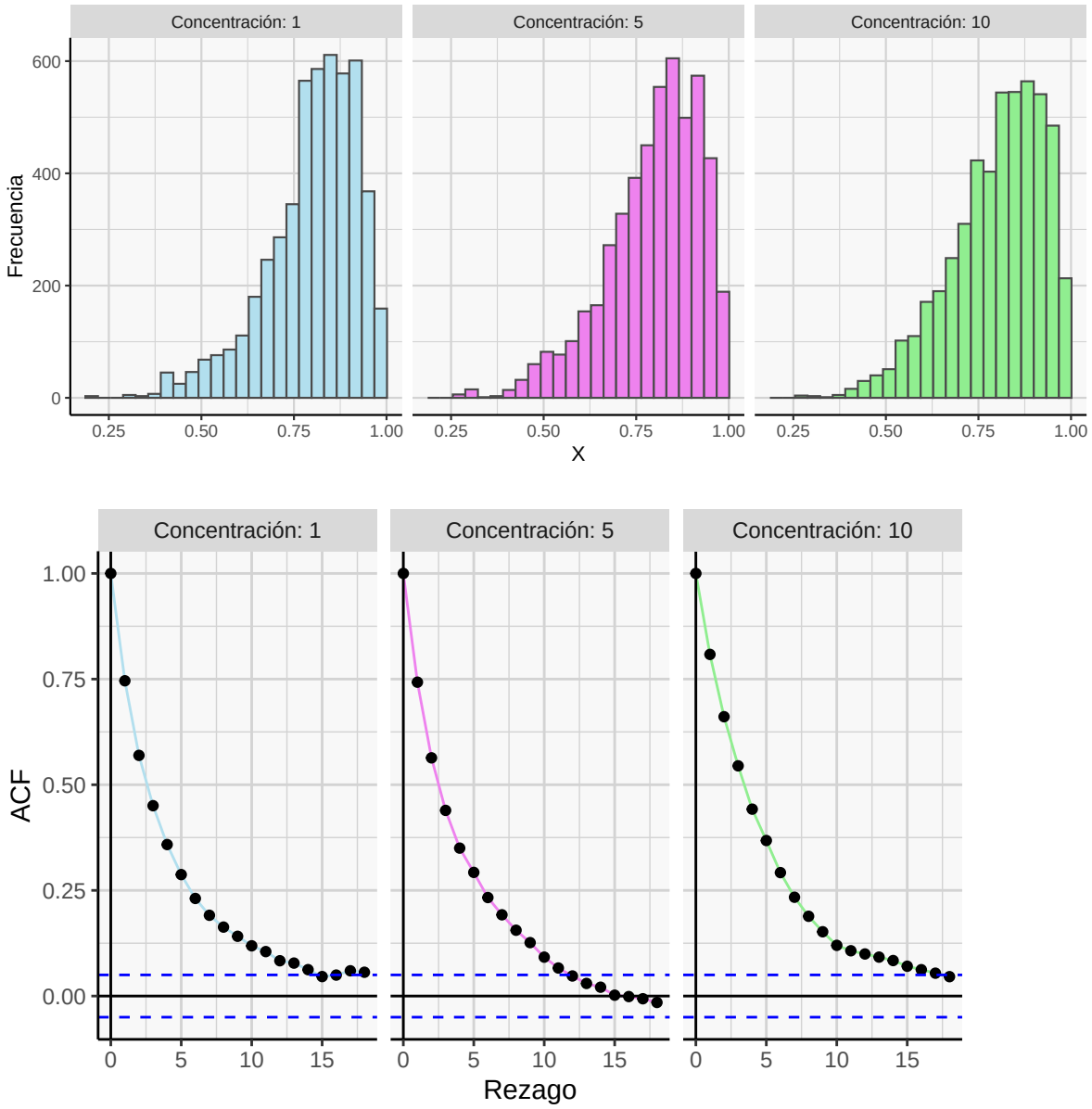
Se usan tres valores de concentración diferentes para evaluar la eficiencia del método.



Concentración	1	5	10
Tasa de aceptación	0.219	0.509	0.618

Table 2: Table to test captions and labels.

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



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 metodo 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
$logit(X_1)$	1.5742044	0.2039405	3.0000926
$logit(X_5)$	1.5952465	0.1565376	3.2647543
$logit(X_{10})$	1.6198775	0.2343745	3.2378466

Table 3: Table to test captions and labels.

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 minimas 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 multivariadias. Igual que la situación anterior, no es necesario que las funciones que se quieren muestrear esten 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 metodo funciona bien y la segunda, una función dificil 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 dstribución 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 variancias y covariancias 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)
```

```r
#Seleccion 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]

    #Calculo 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)
}

#Calculo de la probabilidad de salto
```

```r
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 = dmvnorm(p_actual, p_nuevo, sigma = matrix_var)
  q_nuevo = dmvnorm(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.

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

Se controlará que tan buenas son las muestras obtenidas a través de evaluación de la cadena generada por el método y qué tanto se asimilan las probabilidades estimadas por la distribución muestral obtenida a las reales. La normal bivariada de interés tiene la siguiente distribución.

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

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, la densidad del posterior suele tomar formas semejantes a dicha función, más específicamente y con las que se trabajará es la forma de la función p^*:

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

Más específicamente, se utilizará la función con $a = 0.5$ y $b = 0.5$.

Para tratar de generar muestras más eficientes de la distribución se probará la distribución de salto propuesta con distintas matrices de varianzas y covarianzas evaluando el rendimiento del Metropolis-Hastings con cada una.