

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 $x(1), x(2), \dots, x(S)$ 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 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 vector de tamaño n

# 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 esperar

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

sample_mh_1d = function(n, p_inicial = 0, d_objetivo, r_propuesta = rnorm, d_propuesta = dnorm) {
  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]

    #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

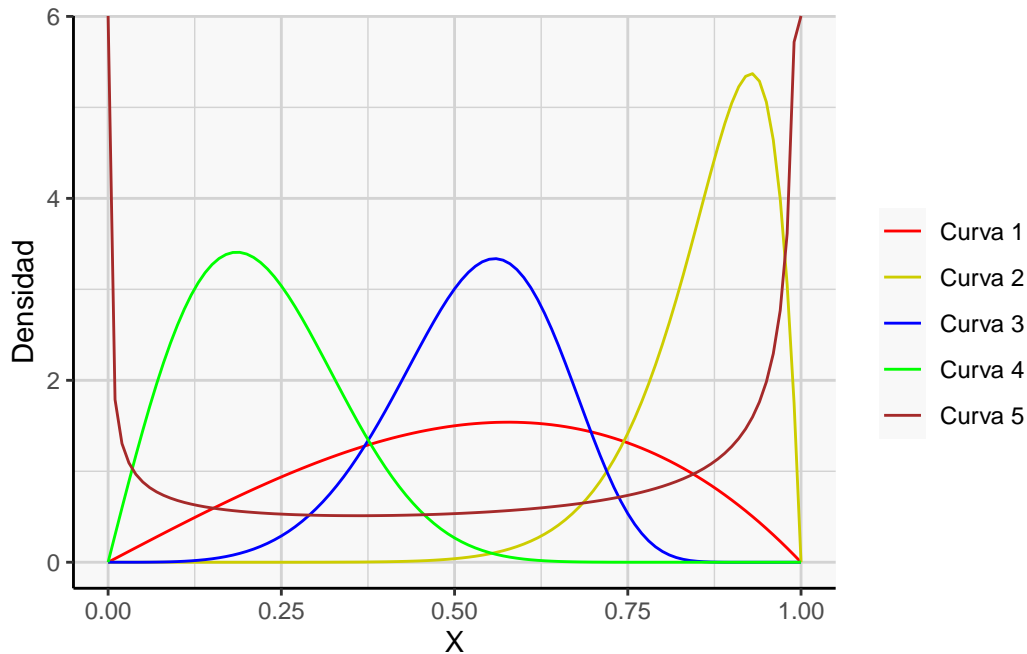
La distribución de Kumaraswamy es una distribución de probabilidad continua que se utiliza para modelar variables aleatorias con soporte en el intervalo $(0,1)$. Su función de densidad se parece a la beta pero su expresión matemática resulta ser de cómputo más sencillo. Dicha función es:

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

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

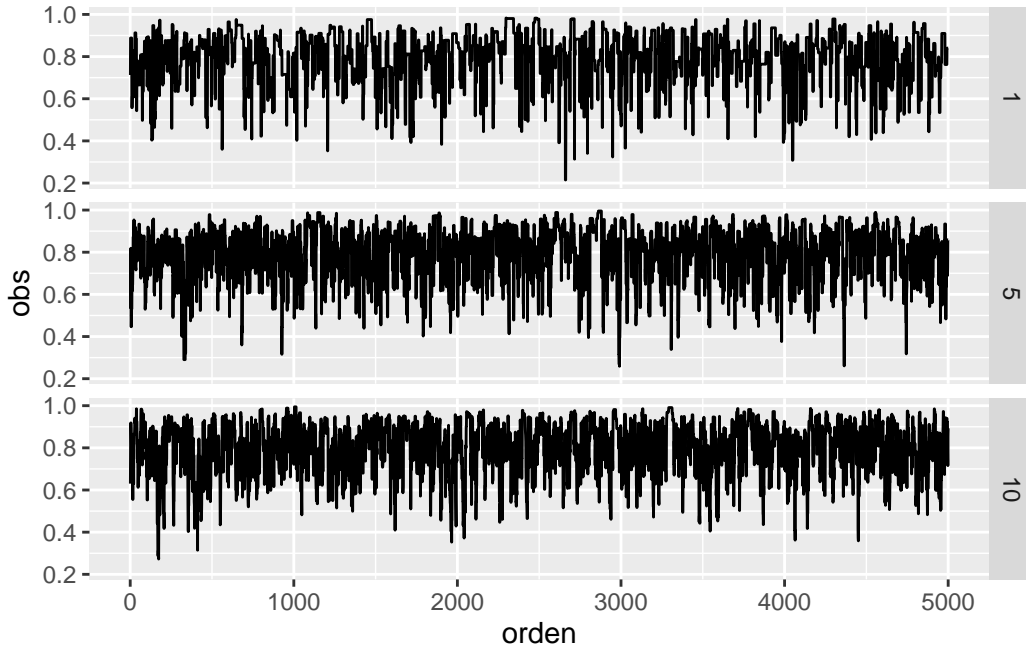
Table 1: Table to test captions and labels.



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.

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



	Concentración		
	1	5	10
Tasa de aceptación	0.22	0.22	0.618

Table 2: Table to test captions and labels.

Concentración	1	5	10
Tasa de aceptación	0.618	0.22	0.509

Table 3: 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.

