Metropolis-Hasting

Introducción

Uno de los grandes problemas que existen en el mundo de la estadística es que hay distribuciones cuyas funciones de densidad son tan complejas que resultan difíciles de trabajar. Es por esto que muchas veces resulta de utilidad trabajar con muestras aleatorias, y a partir de estas responder determinadas preguntas. En la práctica, es común encontrarse con variables aleatorias con funciones de densidad cuyo muestreo directo no es simple.

Para estos casos existen distintas técnicas que nos permiten obtener muestras que, si bien no son tomadas de manera realmente independiente, se comportan de manera muy similar a como lo haría una muestra aleatoria independiente tomada de la función de densidad.

Particularmente en el campo de la *Estadística Bayesiana*, esto resulta útil ya que permite obtener muestras de la distribución a posteriori, la cual se puede utilizar por ejemplo para obtener un intervalo de credibilidad del parámetro bajo estudio.

Un método sencillo de emplear y que reporta buenos resultados es el algoritmo de *Metropolis-Hastings*.

Metodología

En esta sección se presentan los algoritmo de metrópolis hasting para variables aleatorias unidimensionales y bidimensionales

:::callout-note ## Algoritmo Metrópolis-Hasting univariado

```
sample_mh <- function(n, d_objetivo, r_propuesta = NULL, d_propuesta = NULL, p_inicial = NULL){

  if (is.null(r_propuesta) | is.null(d_propuesta)) {
    r_propuesta <- function(media) rnorm(n = 1, media, sd = 1)
    d_propuesta <- function(x, media) dnorm(x = x,media, sd = 1)
  }

  stopifnot(n > 0)
  contador <- 0
  muestras <- numeric(n)

  muestras[1] <- p_inicial

  for(i in 2:n) {

    p_actual <- muestras[i-1]
    p_propuesta <- r_propuesta(p_actual)

    q_actual <- d_propuesta(p_actual, p_propuesta)
    q_nuevo <- d_propuesta(p_propuesta, p_actual)

    f_actual <- d_objetivo(p_actual)
    f_nuevo <- d_objetivo(p_propuesta)

    if (f_actual == 0 || q_nuevo == 0) {
      alfa <- 1
    } else {
      alfa <- min(1, (f_nuevo/f_actual)*(q_actual/q_nuevo))
    }

    muestras[i] <- sample(c(p_propuesta, p_actual),
    size = 1, prob = c(alfa, 1-alfa))

    if(muestras[i] != muestras[i-1]) {
      contador <- contador + 1
    }
  }
  return(list(cadena = data.frame(iteracion = 1:n, x = muestras),
       tasa_aceptacion = contador / n))
}
```

Algoritmo Metrópolis-Hasting bivariado

```r
sample_mh_mv <- function(n, d_objetivo, cov_propuesta = diag(2), p_inicial = numeric(2)) {

  if (length(p_inicial) != 2) {
    stop("El valor p_inicial debe ser bidimensional")
  }
  if ( n <= 0 || n %% 1 != 0) {
    stop("El tamaño de muestra n debe ser entero y mayor que 0")
  }
  if (any((dim(cov_propuesta) != c(2,2)))) {
    stop("La matriz de covariancia debe ser de 2x2")
  }

  contador <- 0

  r_propuesta <-  function(media) rmvnorm(n = 1,mean = media,sigma = cov_propuesta)
  d_propuesta <- function(x, media) dmvnorm(x = x,mean = media,sigma = cov_propuesta)

  muestras <- matrix(0,nrow = n,ncol = length(p_inicial))
  muestras[1, ] <- p_inicial

  for(i in 2:n) {
    p_actual <- muestras[i-1,]
    p_propuesta <- r_propuesta(p_actual)

    q_actual <- d_propuesta(p_actual, p_propuesta)
    q_nuevo <- d_propuesta(p_propuesta, p_actual)

    f_actual <- d_objetivo(p_actual)
    f_nuevo <- d_objetivo(p_propuesta)


    if (f_actual == 0 || q_nuevo == 0) {
      alfa <- 1
    } else {

      alfa <- min(1, (f_nuevo/f_actual)*(q_actual/q_nuevo))
    }

    aceptar <- rbinom(1,1,alfa)

    if (aceptar) {
      muestras[i,] <- p_propuesta
    }else{
      muestras[i,] <- p_actual
    }

    if(!any(muestras[i,] != muestras[i-1,])) {
      contador <- contador + 1
    }
  }
  salida <- data.frame(iteracion = 1:n, x = muestras)
  colnames(salida) <- c("iteracion", paste0("dim_",1:length(p_inicial)))
```

Discusiones

Distribución de Kumaraswamy

Ponemos algo de la distribucion??

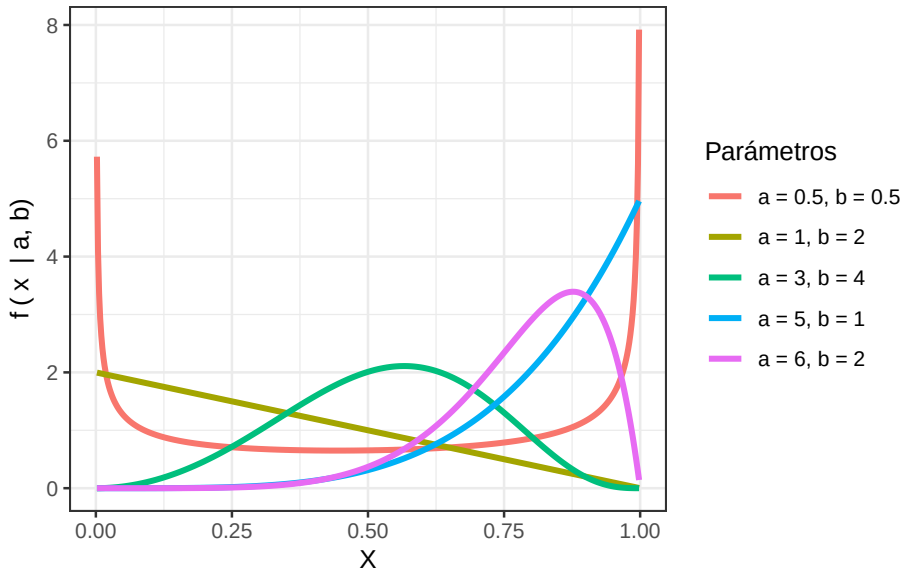


Figura 1: Función de densidad de la distribución de Kumaraswamy

Esta distribución puede ser utilizada en el ámbito de la estadística bayesiana a la hora de definir un prior para un parámetro con campo de variación en el intervalo (0, 1).

Por lo general la distribución elegida para estas situaciones suele ser la beta ya que presenta ventajas como ser una distribución conjugada de la binomial, lo cual puede facilitar mucho algunos cálculos. El problema es que la densidad de esta depende de la función gamma, la cual es una integral, y en algunas situaciones se puede complicar su cálculo.

La distribución de Kumaraswamy se comporta de manera muy similar a la beta, sin tener el problema de la dificultad del cálculo de la integral.

Metropolis-Hasting en una dimensión

A continuación utilizaremos el algoritmo de Metropolis-Hasting para generar 5000 muestras de la distribución de *Kumaraswamy* con parámetros $a = 6$ y $b = 2$, utilizando como distribución propuesta una $Beta(\mu, \kappa)$, donde μ representa la media de la distribución y κ el grado de la concentración de la distribución. Esto lo haremos para 3 valores distintos de κ.

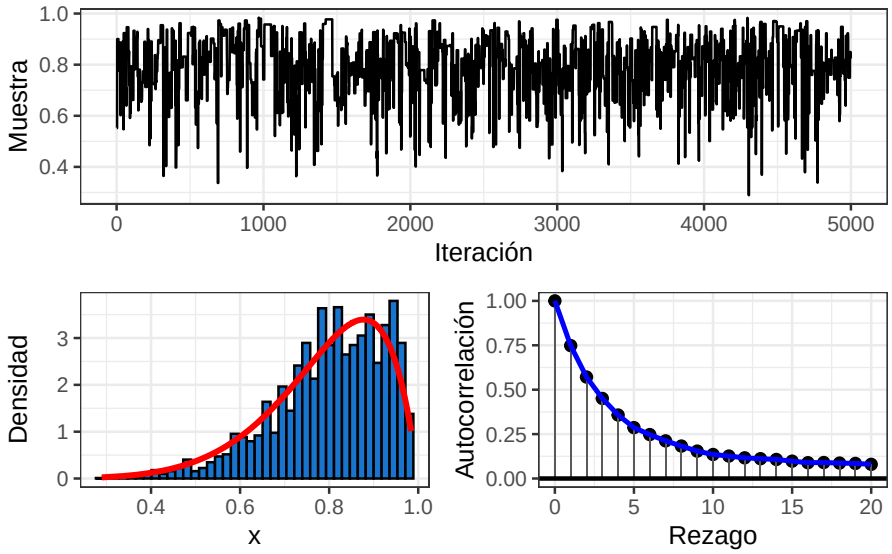


Figura 2: Muestreo por Metropolis-Hastings con $\kappa = 1$

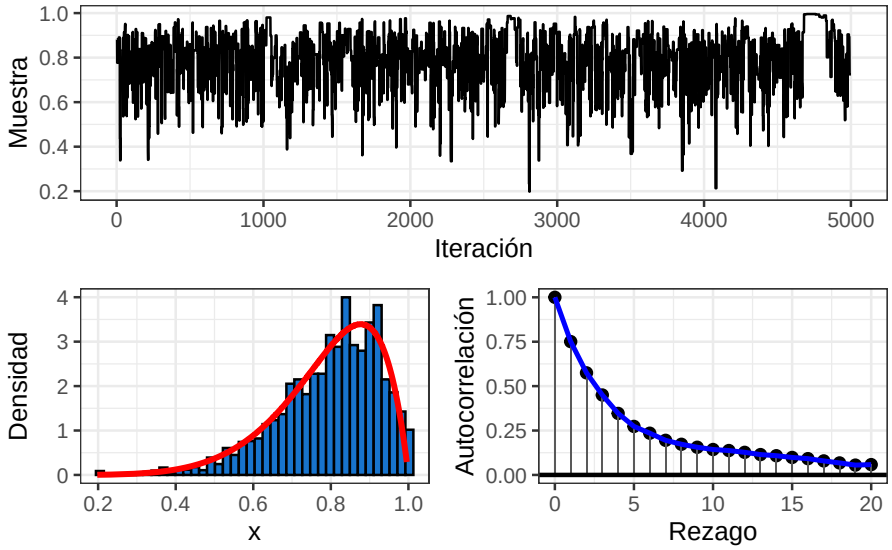


Figura 3: Muestreo por Metropolis-Hastings con $\kappa = 2$

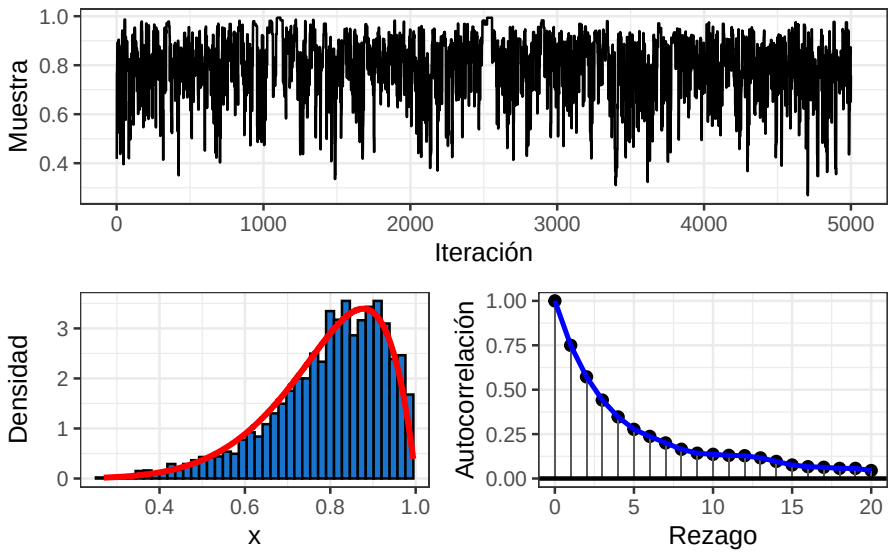


Figura 4: Muestreo por Metropolis-Hastings con $\kappa = 5$

(Agregar el n_{eff} para mejorar la comparación de las cadenas)

Tabla 1: Media y percentiles estimadas para las funciones X y $Logit(X)$

$f(x)$	κ	$\widehat{E(x)}$	$\widehat{q_{0.05}}$	$\widehat{q_{0.95}}$
X	1	0.802	0.571	0.959
	2	0.797	0.545	0.971
	5	0.799	0.534	0.969
$Logit(X)$	1	1.615	0.287	3.165
	2	1.636	0.182	3.528
	5	1.632	0.137	3.432

Metropolis-Hasting en dos dimensiónes

> **Nota**

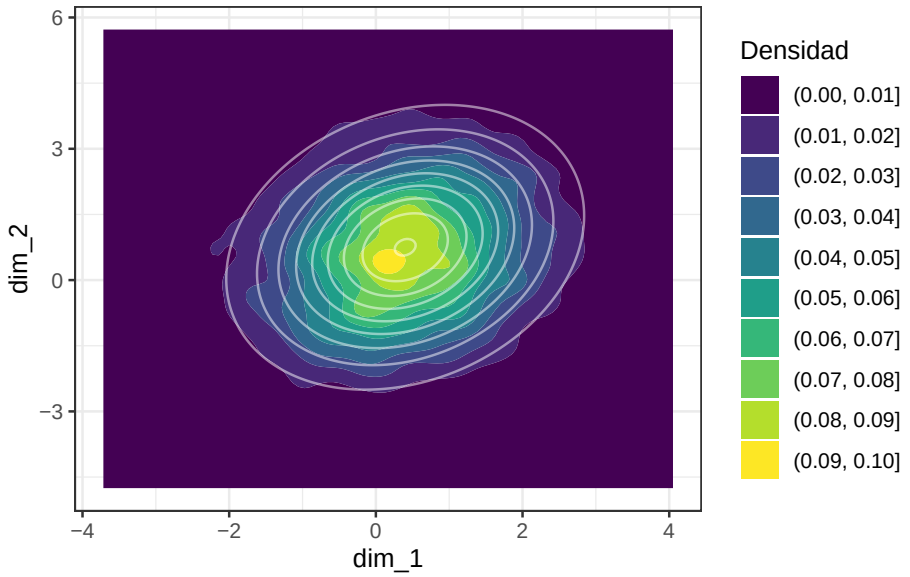
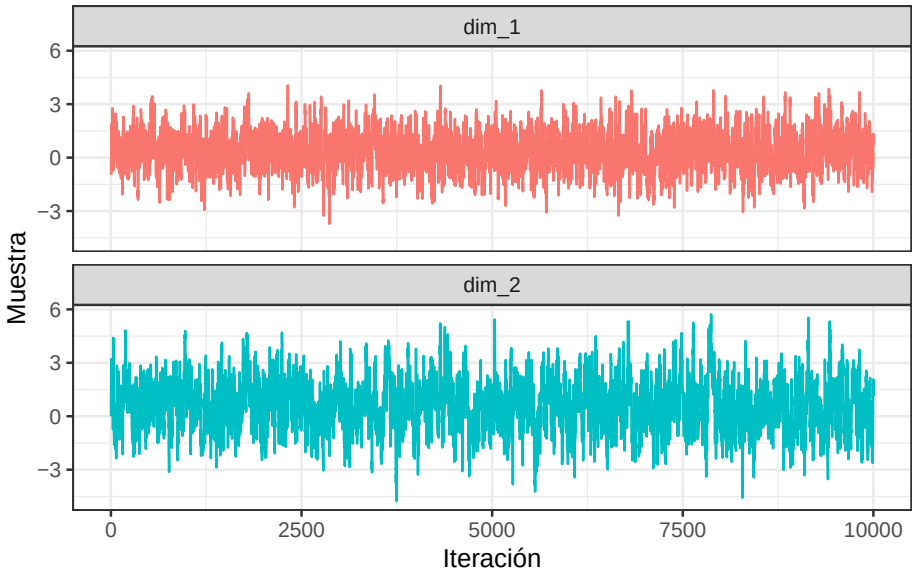


Figura 5: lallala

Titulos



Probabilidades

```
[1] 0.0796
```

```
[1] 0.0694
```

```
[1] 0.2501
```

```
[1] 0.06825141
```

```
[1] 0.08700867
```

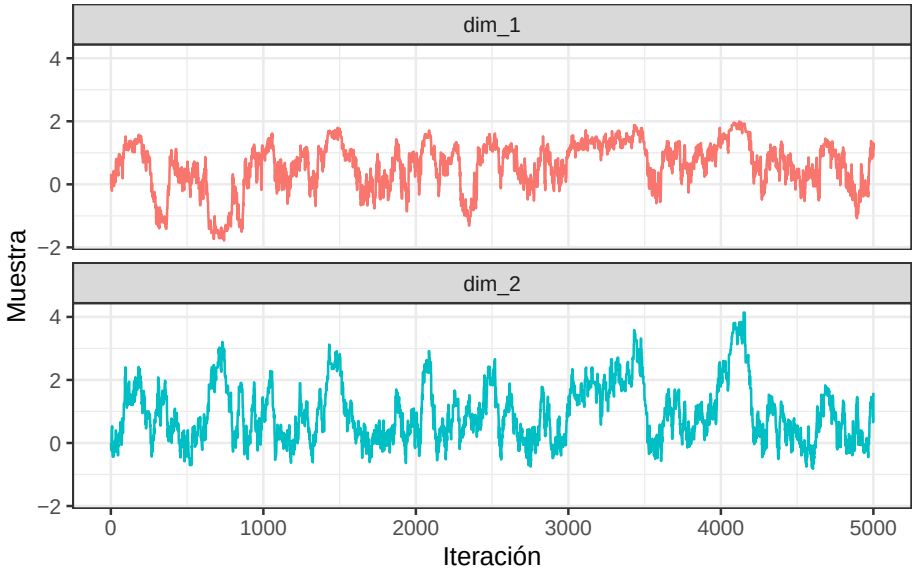
```
[1] 0.2856655
```

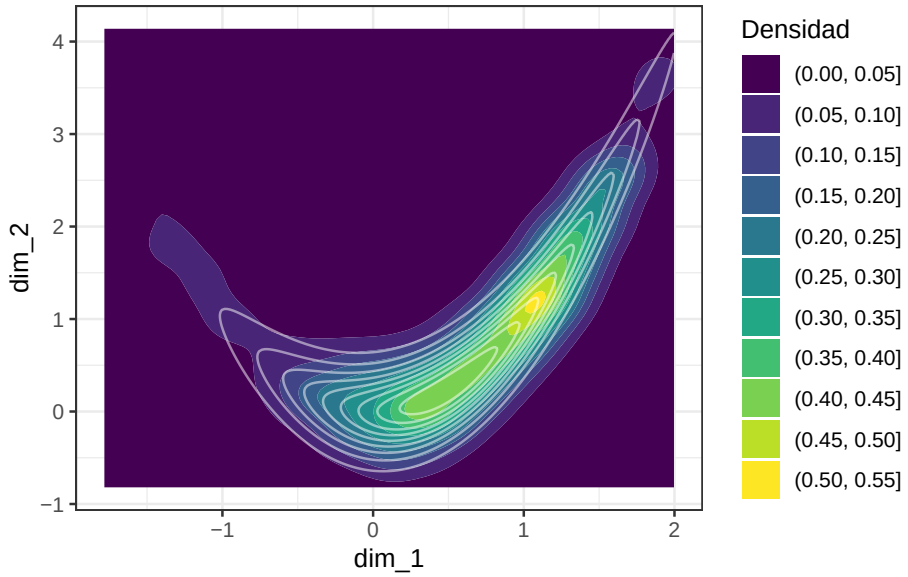
```
[1] 0.4398
```

```
Autocorrelations of series 'cadena_p$muestra_mh[-1]', by lag

, , dim_1

 dim_1       dim_2
 1.000 (  0) 0.472 (  0)
 0.980 (  1) 0.469 ( -1)
 0.962 (  2) 0.466 ( -2)
 0.947 (  3) 0.463 ( -3)
 0.932 (  4) 0.459 ( -4)
 0.919 (  5) 0.454 ( -5)
```

```
 0.907 (  6) 0.449 ( -6)
 0.895 (  7) 0.444 ( -7)
 0.884 (  8) 0.439 ( -8)
 0.872 (  9) 0.433 ( -9)
 0.861 ( 10) 0.427 (-10)
 0.852 ( 11) 0.421 (-11)
 0.842 ( 12) 0.414 (-12)
 0.831 ( 13) 0.407 (-13)
 0.821 ( 14) 0.401 (-14)
 0.811 ( 15) 0.395 (-15)
 0.802 ( 16) 0.389 (-16)
 0.793 ( 17) 0.384 (-17)
 0.783 ( 18) 0.378 (-18)
 0.775 ( 19) 0.373 (-19)
 0.768 ( 20) 0.368 (-20)
 0.760 ( 21) 0.363 (-21)
 0.753 ( 22) 0.359 (-22)
 0.745 ( 23) 0.355 (-23)
 0.738 ( 24) 0.352 (-24)
 0.730 ( 25) 0.348 (-25)
 0.722 ( 26) 0.344 (-26)
 0.714 ( 27) 0.339 (-27)
 0.707 ( 28) 0.335 (-28)
 0.699 ( 29) 0.331 (-29)
 0.693 ( 30) 0.328 (-30)
 0.685 ( 31) 0.326 (-31)
 0.678 ( 32) 0.324 (-32)
 0.671 ( 33) 0.321 (-33)

, , dim_2

 dim_1        dim_2
 0.472 (  0) 1.000 (  0)
 0.468 (  1) 0.982 (  1)
 0.464 (  2) 0.966 (  2)
 0.460 (  3) 0.950 (  3)
 0.456 (  4) 0.936 (  4)
 0.452 (  5) 0.922 (  5)
 0.447 (  6) 0.909 (  6)
 0.442 (  7) 0.896 (  7)
 0.437 (  8) 0.882 (  8)
 0.431 (  9) 0.869 (  9)
 0.426 ( 10) 0.855 ( 10)
 0.420 ( 11) 0.841 ( 11)
 0.415 ( 12) 0.827 ( 12)
 0.409 ( 13) 0.814 ( 13)
 0.404 ( 14) 0.801 ( 14)
 0.399 ( 15) 0.789 ( 15)
```

```
0.394 ( 16) 0.777 ( 16)
0.389 ( 17) 0.766 ( 17)
0.384 ( 18) 0.755 ( 18)
0.380 ( 19) 0.745 ( 19)
0.376 ( 20) 0.734 ( 20)
0.371 ( 21) 0.724 ( 21)
0.367 ( 22) 0.714 ( 22)
0.364 ( 23) 0.704 ( 23)
0.360 ( 24) 0.695 ( 24)
0.357 ( 25) 0.685 ( 25)
0.354 ( 26) 0.675 ( 26)
0.350 ( 27) 0.664 ( 27)
0.347 ( 28) 0.654 ( 28)
0.343 ( 29) 0.643 ( 29)
0.340 ( 30) 0.634 ( 30)
0.337 ( 31) 0.624 ( 31)
0.333 ( 32) 0.613 ( 32)
0.330 ( 33) 0.604 ( 33)
```



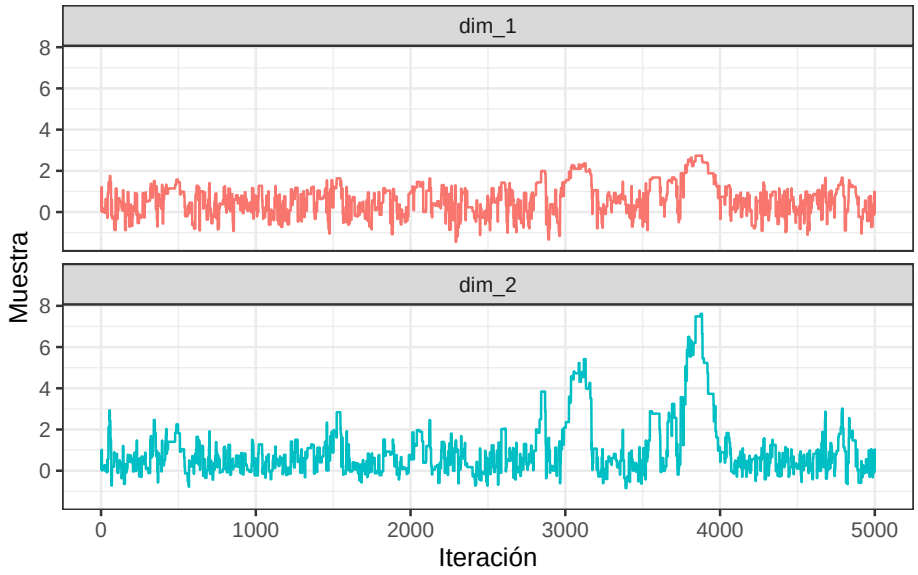


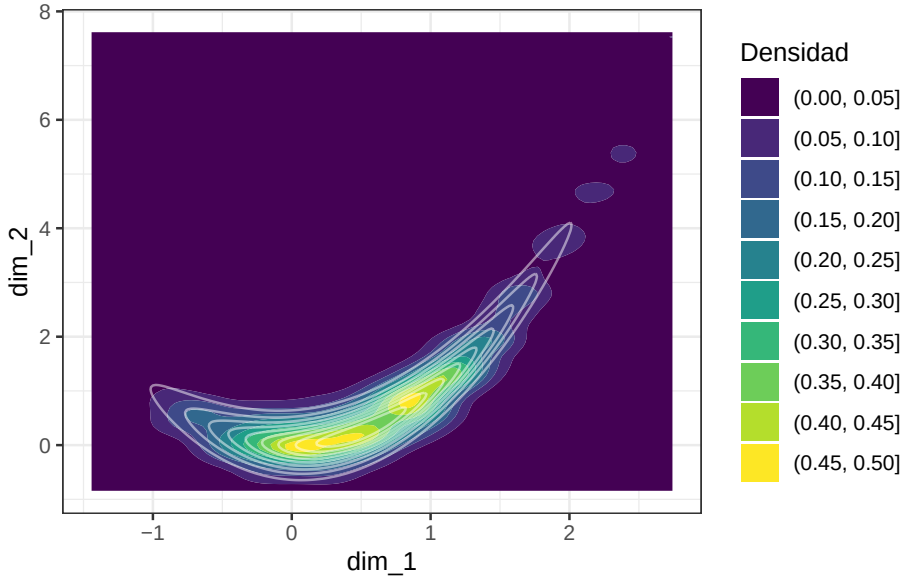

```
[1] 0.8012
```

```
Autocorrelations of series 'cadena_p$muestra_mh[-1]', by lag

, , dim_1

 dim_1       dim_2
 1.000 (  0) 0.787 (  0)
 0.934 (  1) 0.773 ( -1)
 0.878 (  2) 0.760 ( -2)
 0.828 (  3) 0.747 ( -3)
 0.783 (  4) 0.733 ( -4)
 0.744 (  5) 0.719 ( -5)
 0.708 (  6) 0.707 ( -6)
 0.675 (  7) 0.695 ( -7)
 0.645 (  8) 0.685 ( -8)
 0.623 (  9) 0.676 ( -9)
 0.602 ( 10) 0.667 (-10)
 0.582 ( 11) 0.659 (-11)
 0.562 ( 12) 0.651 (-12)
 0.546 ( 13) 0.644 (-13)
 0.531 ( 14) 0.634 (-14)
 0.516 ( 15) 0.625 (-15)
 0.501 ( 16) 0.616 (-16)
 0.486 ( 17) 0.608 (-17)
 0.473 ( 18) 0.602 (-18)
 0.464 ( 19) 0.596 (-19)
 0.458 ( 20) 0.591 (-20)
 0.451 ( 21) 0.585 (-21)
 0.445 ( 22) 0.581 (-22)
```

```
 0.441 ( 23) 0.575 (-23)
 0.436 ( 24) 0.570 (-24)
 0.432 ( 25) 0.565 (-25)
 0.429 ( 26) 0.560 (-26)
 0.427 ( 27) 0.556 (-27)
 0.426 ( 28) 0.552 (-28)
 0.428 ( 29) 0.547 (-29)
 0.427 ( 30) 0.542 (-30)
 0.425 ( 31) 0.537 (-31)
 0.422 ( 32) 0.532 (-32)
 0.419 ( 33) 0.527 (-33)

, , dim_2

 dim_1       dim_2
 0.787 (  0) 1.000 (  0)
 0.772 (  1) 0.977 (  1)
 0.758 (  2) 0.957 (  2)
 0.745 (  3) 0.940 (  3)
 0.731 (  4) 0.924 (  4)
 0.719 (  5) 0.910 (  5)
 0.705 (  6) 0.896 (  6)
 0.694 (  7) 0.884 (  7)
 0.683 (  8) 0.873 (  8)
 0.673 (  9) 0.863 (  9)
 0.663 ( 10) 0.855 ( 10)
 0.654 ( 11) 0.847 ( 11)
 0.644 ( 12) 0.838 ( 12)
 0.635 ( 13) 0.830 ( 13)
 0.624 ( 14) 0.821 ( 14)
 0.615 ( 15) 0.813 ( 15)
 0.607 ( 16) 0.804 ( 16)
 0.598 ( 17) 0.796 ( 17)
 0.590 ( 18) 0.788 ( 18)
 0.580 ( 19) 0.781 ( 19)
 0.571 ( 20) 0.774 ( 20)
 0.566 ( 21) 0.767 ( 21)
 0.559 ( 22) 0.760 ( 22)
 0.553 ( 23) 0.753 ( 23)
 0.548 ( 24) 0.746 ( 24)
 0.543 ( 25) 0.739 ( 25)
 0.539 ( 26) 0.733 ( 26)
 0.534 ( 27) 0.727 ( 27)
 0.529 ( 28) 0.721 ( 28)
 0.528 ( 29) 0.717 ( 29)
 0.526 ( 30) 0.712 ( 30)
 0.525 ( 31) 0.708 ( 31)
 0.524 ( 32) 0.704 ( 32)
```

```
0.521 ( 33) 0.699 ( 33)
```



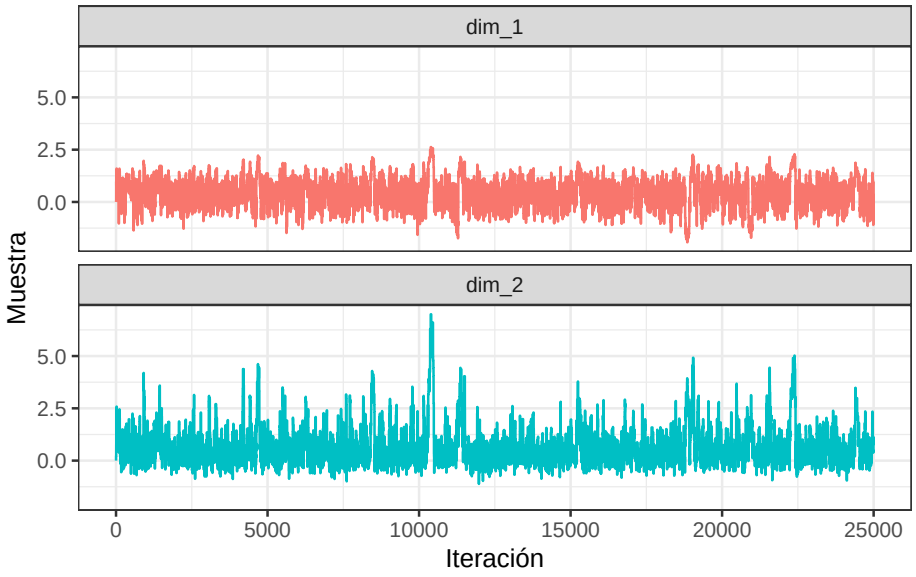


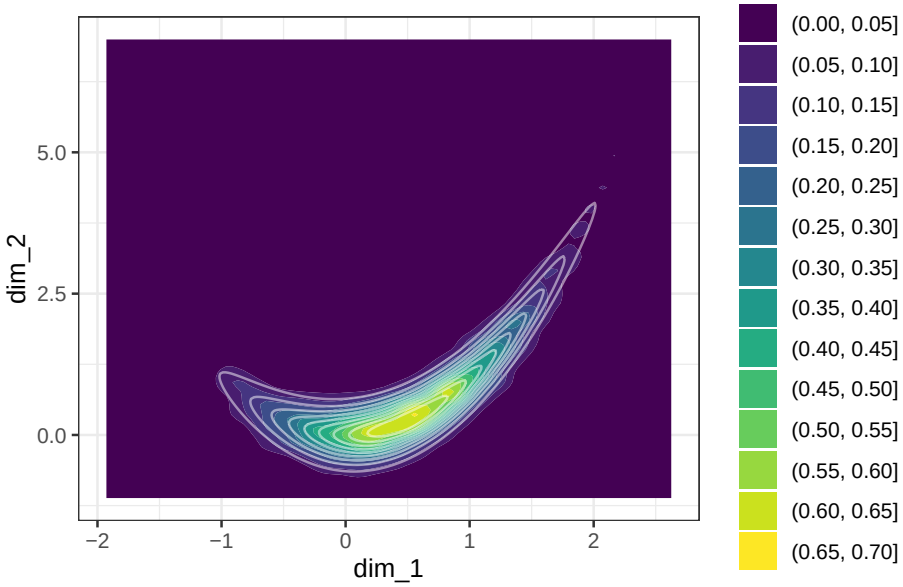

```
[1] 0.70944
```

```
Autocorrelations of series 'cadena_p$muestra_mh[-1]', by lag

, , dim_1

 dim_1       dim_2
 1.000 (  0) 0.607 (  0)
 0.929 (  1) 0.589 ( -1)
```

```
 0.867 (  2) 0.572 ( -2)
 0.814 (  3) 0.555 ( -3)
 0.769 (  4) 0.537 ( -4)
 0.729 (  5) 0.519 ( -5)
 0.690 (  6) 0.501 ( -6)
 0.656 (  7) 0.484 ( -7)
 0.625 (  8) 0.468 ( -8)
 0.596 (  9) 0.454 ( -9)
 0.569 ( 10) 0.440 (-10)
 0.544 ( 11) 0.426 (-11)
 0.522 ( 12) 0.414 (-12)
 0.502 ( 13) 0.401 (-13)
 0.481 ( 14) 0.388 (-14)
 0.463 ( 15) 0.375 (-15)
 0.445 ( 16) 0.362 (-16)
 0.430 ( 17) 0.352 (-17)
 0.417 ( 18) 0.341 (-18)
 0.404 ( 19) 0.331 (-19)
 0.392 ( 20) 0.322 (-20)
 0.380 ( 21) 0.312 (-21)
 0.369 ( 22) 0.303 (-22)
 0.358 ( 23) 0.293 (-23)
 0.348 ( 24) 0.284 (-24)
 0.337 ( 25) 0.274 (-25)
 0.326 ( 26) 0.266 (-26)
 0.314 ( 27) 0.257 (-27)
 0.304 ( 28) 0.249 (-28)
 0.294 ( 29) 0.241 (-29)
 0.285 ( 30) 0.234 (-30)
 0.276 ( 31) 0.227 (-31)
 0.269 ( 32) 0.221 (-32)
 0.263 ( 33) 0.215 (-33)
 0.258 ( 34) 0.209 (-34)
 0.252 ( 35) 0.203 (-35)
 0.246 ( 36) 0.198 (-36)
 0.239 ( 37) 0.192 (-37)
 0.232 ( 38) 0.186 (-38)
 0.225 ( 39) 0.181 (-39)
 0.218 ( 40) 0.176 (-40)

, , dim_2

 dim_1       dim_2
 0.607 (  0) 1.000 (  0)
 0.590 (  1) 0.958 (  1)
 0.572 (  2) 0.920 (  2)
 0.554 (  3) 0.887 (  3)
 0.536 (  4) 0.856 (  4)
```

```
0.518 (  5) 0.829 (  5)
0.501 (  6) 0.802 (  6)
0.485 (  7) 0.779 (  7)
0.469 (  8) 0.756 (  8)
0.456 (  9) 0.735 (  9)
0.443 ( 10) 0.715 ( 10)
0.432 ( 11) 0.697 ( 11)
0.420 ( 12) 0.680 ( 12)
0.409 ( 13) 0.663 ( 13)
0.398 ( 14) 0.646 ( 14)
0.386 ( 15) 0.630 ( 15)
0.374 ( 16) 0.615 ( 16)
0.362 ( 17) 0.600 ( 17)
0.352 ( 18) 0.585 ( 18)
0.341 ( 19) 0.570 ( 19)
0.331 ( 20) 0.555 ( 20)
0.322 ( 21) 0.541 ( 21)
0.313 ( 22) 0.527 ( 22)
0.303 ( 23) 0.514 ( 23)
0.294 ( 24) 0.501 ( 24)
0.285 ( 25) 0.489 ( 25)
0.275 ( 26) 0.477 ( 26)
0.267 ( 27) 0.465 ( 27)
0.258 ( 28) 0.453 ( 28)
0.251 ( 29) 0.443 ( 29)
0.244 ( 30) 0.433 ( 30)
0.238 ( 31) 0.423 ( 31)
0.232 ( 32) 0.414 ( 32)
0.225 ( 33) 0.405 ( 33)
0.219 ( 34) 0.395 ( 34)
0.213 ( 35) 0.385 ( 35)
0.207 ( 36) 0.376 ( 36)
0.202 ( 37) 0.366 ( 37)
0.197 ( 38) 0.358 ( 38)
0.193 ( 39) 0.351 ( 39)
0.189 ( 40) 0.343 ( 40)
```






```
[1] 0.38096
```

```
[1] 0.14172
```

```
[1] 0.05012
```

```
[1] 0.5137135
```

```
[1] 0.2021936
```

```
[1] 0.08382305
```

Preguntas y propuestas

Agregar n_eff en mh univarido Preguntar por la probabilidad de salto en mh_multivariado comparar las probabilidades tmb por monte carlo