# 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

(Agregar como funciona metropolis-hasting)

Algoritmo Metrópolis-Hasting univariado

```r
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

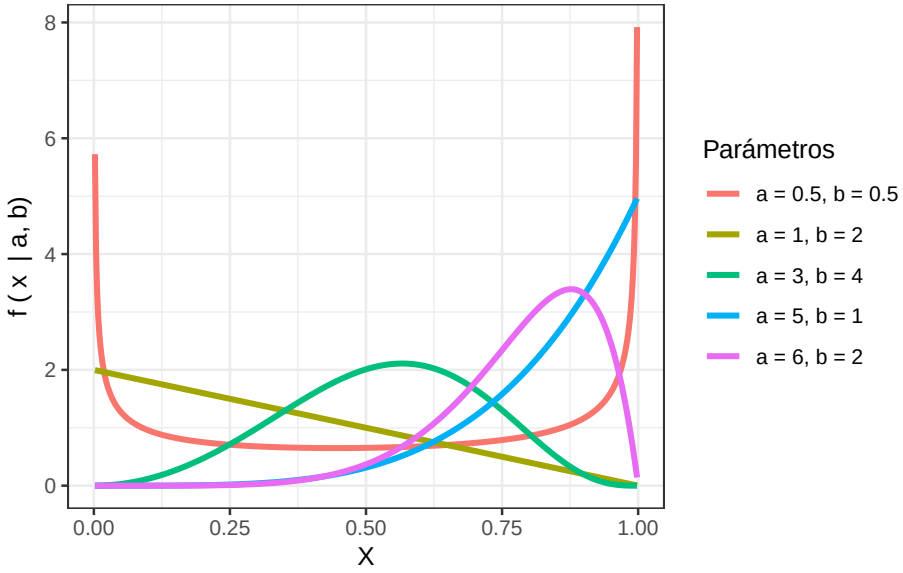



Figura 1: Función de densidad de la distribución de Kumaraswamy

## Metropolis-Hasting en una dimensión

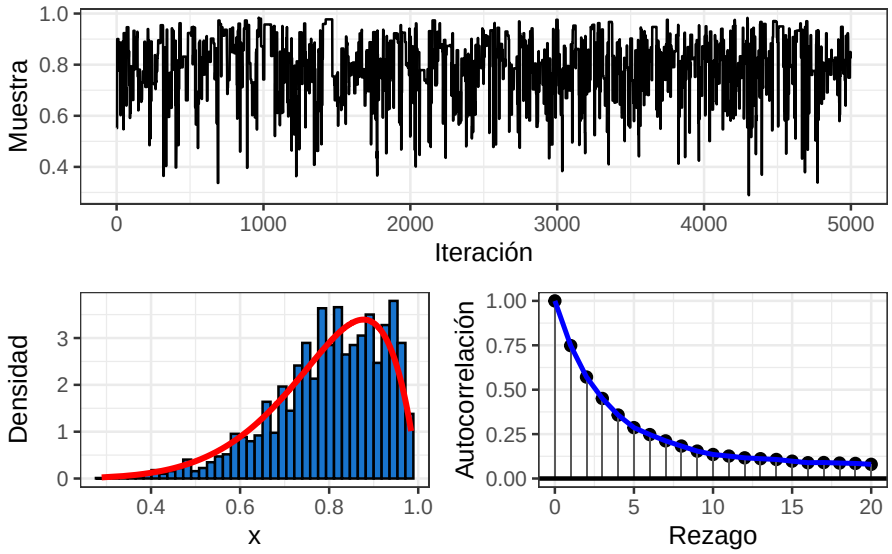



Figura 2: Muestreo por Metropolis-Hastings con $\kappa = 1$

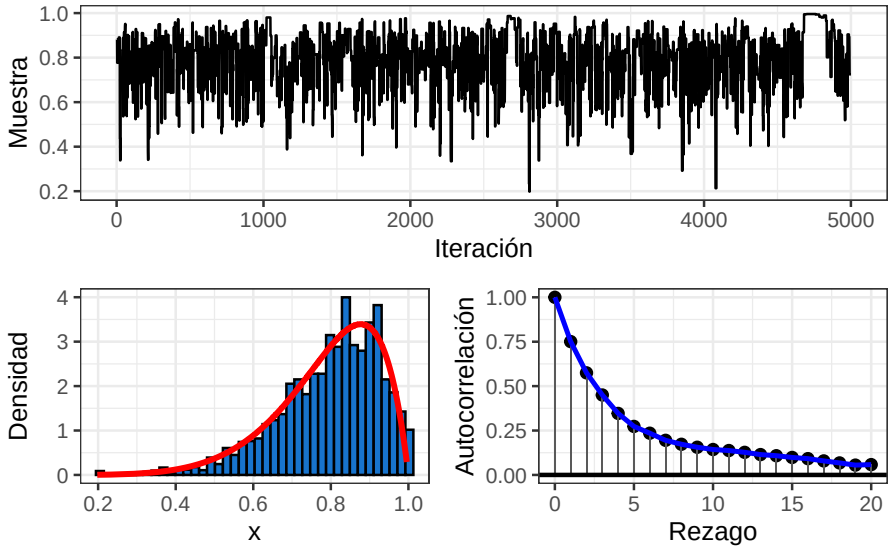



Figura 3: Muestreo por Metropolis-Hastings con $\kappa = 2$

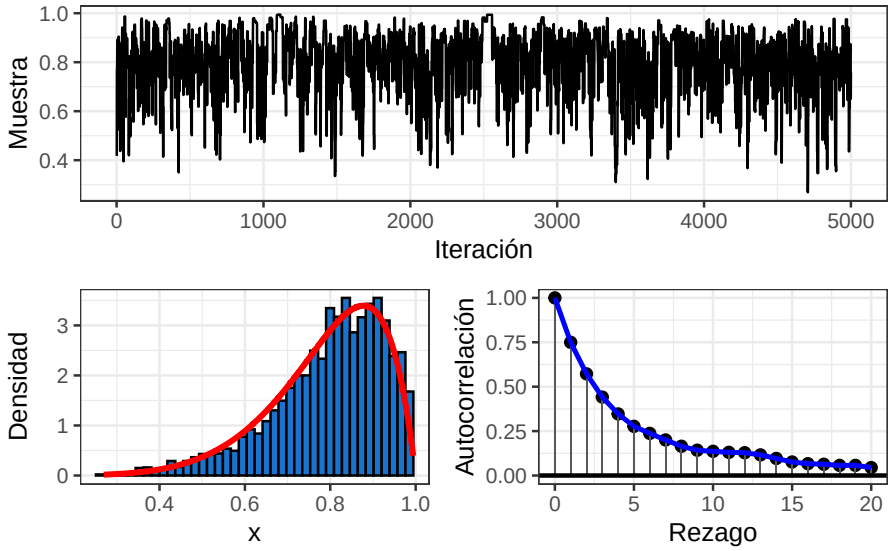



Figura 4: Muestreo por Metropolis-Hastings con $\kappa = 5$

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

| $f(x)$ | $\kappa$ | $\widehat{E(x)}$ | $\widehat{q_{0.05}}$ | $\widehat{q_{0.95}}$ |
|---|---|---|---|---|
| $X$ | 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**

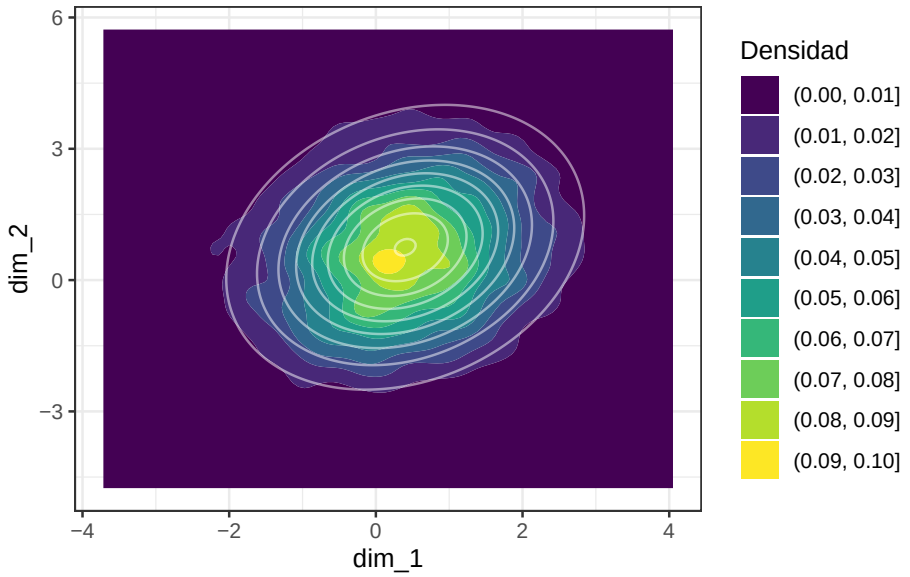
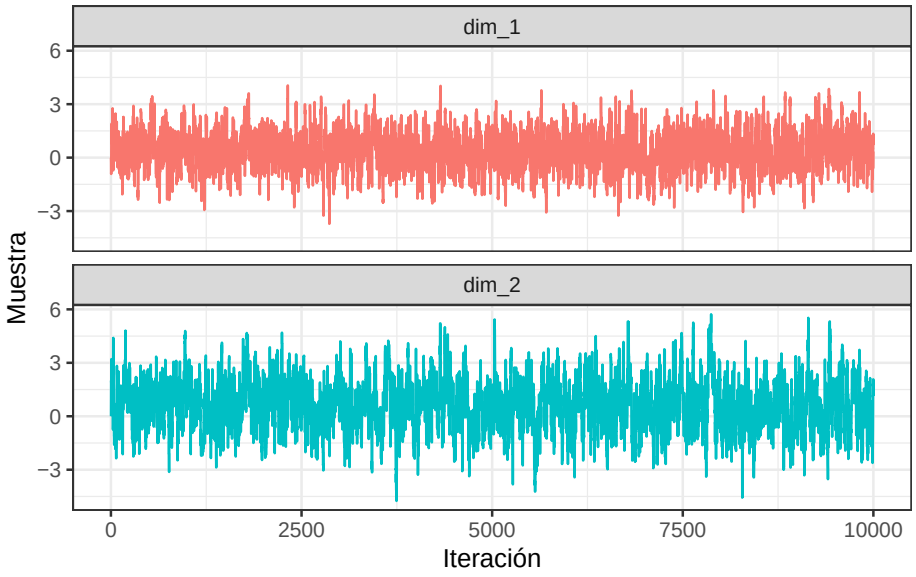
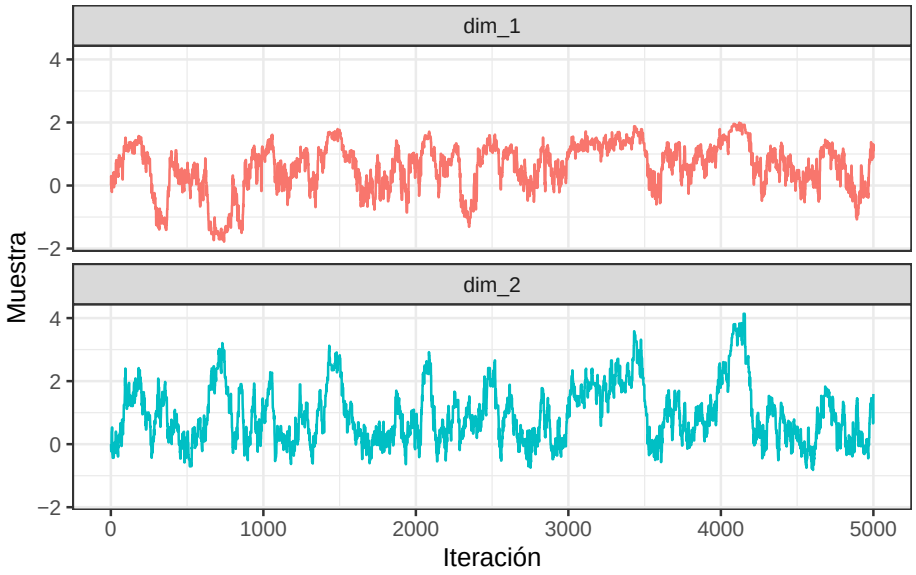
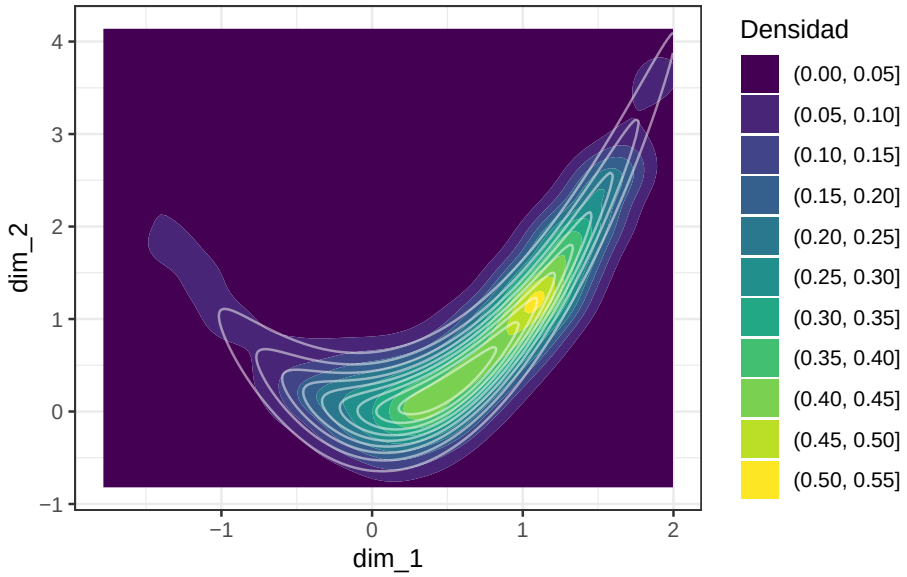
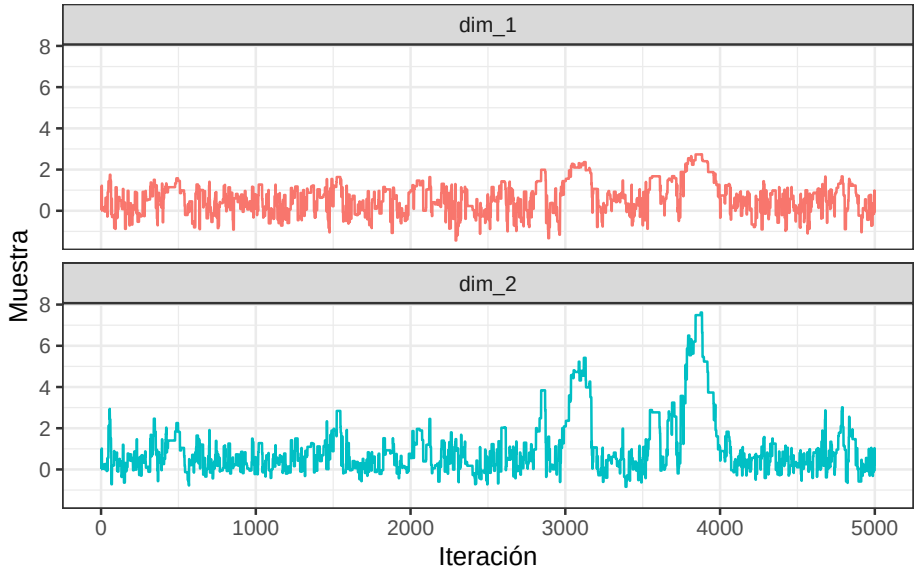


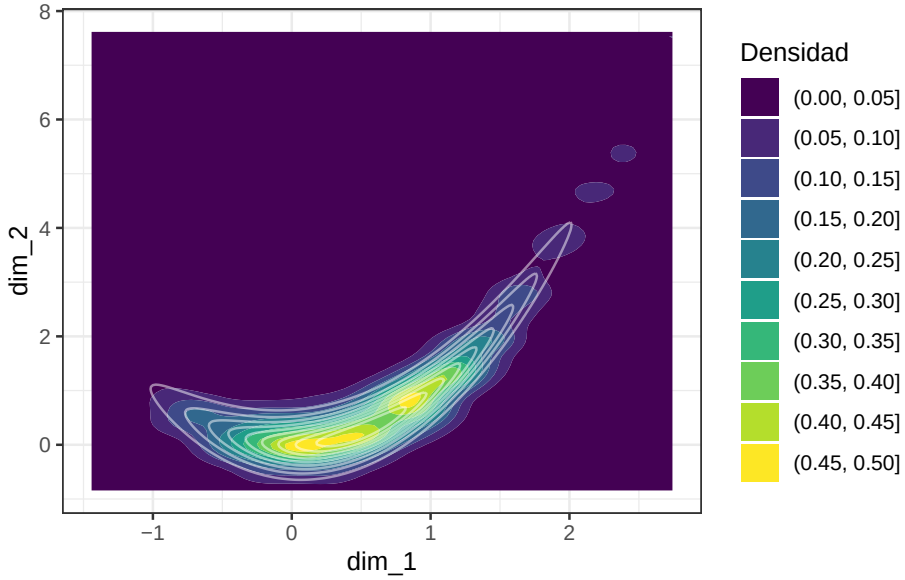

Figura 5: lallala

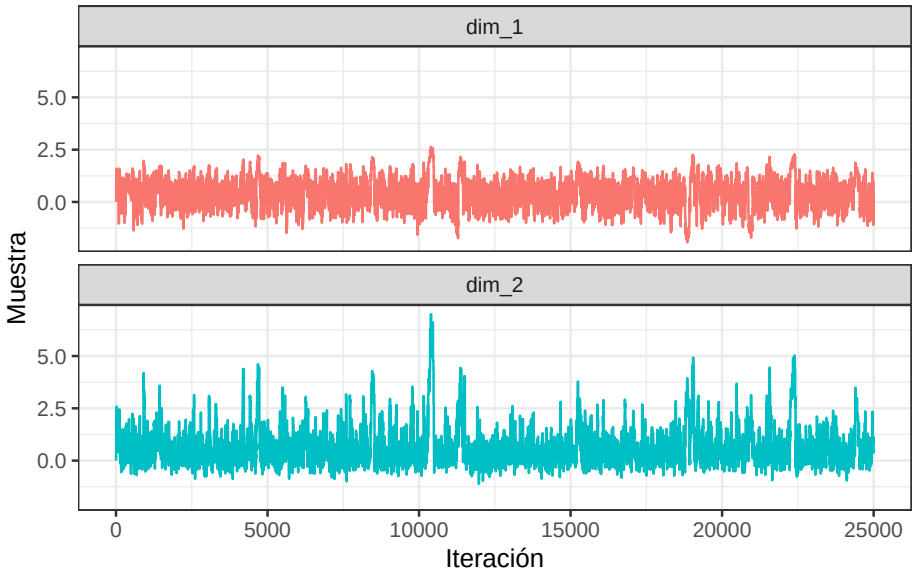

dim_1
dim_2
6
3
0
−3
Muestra
0
2500
5000
7500
10000
Iteración

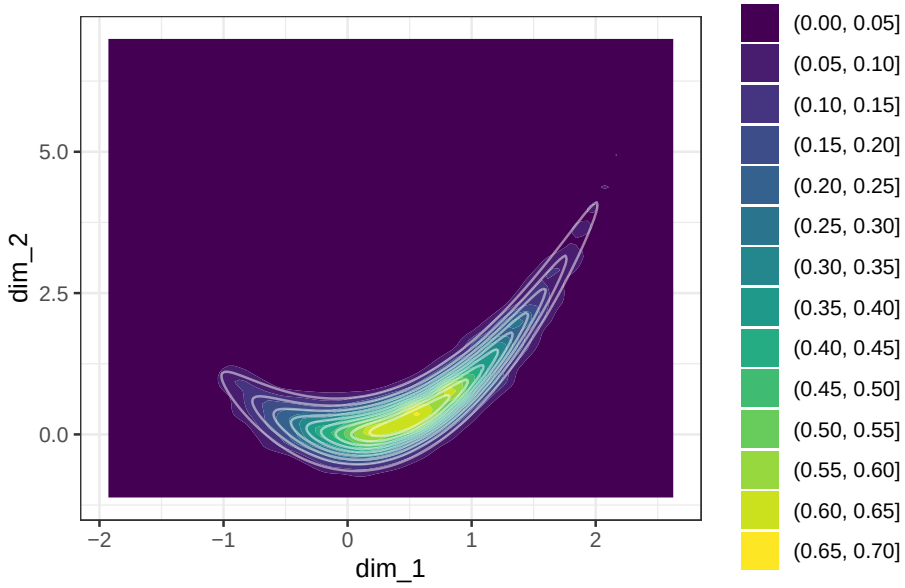

dim_1
dim_2
Muestra
Iteración
4
2
0
−2
0
1000
2000
3000
4000
5000


dim_2
dim_1
4
3
2
1
0
−1
−1
0
1
2
Densidad
(0.00, 0.05]
(0.05, 0.10]
(0.10, 0.15]
(0.15, 0.20]
(0.20, 0.25]
(0.25, 0.30]
(0.30, 0.35]
(0.35, 0.40]
(0.40, 0.45]
(0.45, 0.50]
(0.50, 0.55]


dim_1
dim_2
Muestra
Iteración
0
1000
2000
3000
4000
5000


dim_2
dim_1
Densidad
(0.00, 0.05]
(0.05, 0.10]
(0.10, 0.15]
(0.15, 0.20]
(0.20, 0.25]
(0.25, 0.30]
(0.30, 0.35]
(0.35, 0.40]
(0.40, 0.45]
(0.45, 0.50]


dim_1
dim_2
Muestra
5.0
2.5
0.0
0
5000
10000
15000
20000
25000
Iteración


dim_2
dim_1
5.0
2.5
0.0
−2
−1
0
1
2
(0.00, 0.05]
(0.05, 0.10]
(0.10, 0.15]
(0.15, 0.20]
(0.20, 0.25]
(0.25, 0.30]
(0.30, 0.35]
(0.35, 0.40]
(0.40, 0.45]
(0.45, 0.50]
(0.50, 0.55]
(0.55, 0.60]
(0.60, 0.65]
(0.65, 0.70]

## Preguntas y propuestas