

# Tarea 2

## Instrucciones

Resuelve los ejercicios de la clase (ver handout) y auxiliate del código de soporte.

### Ejercicio 0

El código siguiente está tomado del código de soporte (Garbuno 2022a), ajustando solamente el parámetro  $M$  como se pide en la pregunta.

Se crea la distribución objetivo, la aproximación normal y la función para muestrear por aceptación/rechazo.

```
set.seed(178906)
```

```
## Setup -----
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr    0.3.4
## v tibble   3.1.6     v dplyr    1.0.8
## v tidyr    1.2.0     v stringr  1.4.0
## v readr    2.1.2     vforcats 0.5.1

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()

library(patchwork)
library(scales)

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

## Cambia el default del tamaño de fuente
theme_set(theme_linedraw(base_size = 8))

## Cambia el número de decimales para mostrar
options(digits = 2)

sin_lineas <- theme(panel.grid.major = element_blank(),
                     panel.grid.minor = element_blank())
color.item <- c("#00362b", "#004a3b", "#00503f", "#006953", "#008367", "#009c7b", "#00b68f", NA)
```

```

sin_lineas <- theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
sin_leyenda <- theme(legend.position = "none")
sin_ejes <- theme(axis.ticks = element_blank(), axis.text = element_blank())

crea_mezcla <- function(weights){
  function(x){
    weights$w1 * dnorm(x, mean = -1.5, sd = .5) +
    weights$w2 * dnorm(x, mean = 1.5, sd = .7)
  }
}
objetivo <- crea_mezcla(list(w1 = .6, w2 = .4))

library(R6)
ModeloNormal <-
  R6Class("ProbabilityModel",
    list(
      mean = NA,
      sd = NA,
      ## Inicializador
      initialize = function(mean = 0, sd = 1){
        self$mean = mean
        self$sd = sd
      },
      ## Muestreador
      sample = function(n = 1){
        rnorm(n, self$mean, sd = self$sd)
      },
      ## Evaluacion de densidad
      density = function(x, log = TRUE){
        dnorm(x, self$mean, sd = self$sd, log = log)
      }
    ))
}

crea_rejection_sampling <- function(objetivo, aprox, M){
  function(niter){
    muestras <- matrix(nrow = niter, ncol = 3)
    for (ii in seq(1, niter)){
      propuesta <- aprox$sample()
      p <- objetivo(propuesta)
      g <- aprox$density(propuesta, log = FALSE)
      u <- runif(1)
      if (u < p/(M * g)) { ## Aceptamos
        muestras[ii, 1] <- 1
      } else { ## Rechazamos
        muestras[ii, 1] <- 0
      }
      muestras[ii, 2] <- propuesta
      muestras[ii, 3] <- u
    }
    colnames(muestras) <- c("accept", "value", "auxiliar")
    muestras
  }
}

```

```

modelo.muestreo <- ModeloNormal$new(mean = 0, sd = 2)

Primero vamos a observar el comportamiento con  $M$  grande. Tomamos  $M = 20$ .
M <- 20

muestreo_rechazo <- crea_rejection_sampling(objetivo, modelo.muestreo, M)

muestras <- muestreo_rechazo(50000) |>
  as.tibble() |>
  mutate(density = modelo.muestreo$density(value, log = FALSE))

## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.

g1 <- muestras |>
  ggplot(aes(value, auxiliar * modelo.muestreo$density(value, log = FALSE))) +
  geom_point(aes(color = factor(accept))) + sin_lineas + sin_ejes + sin_leyenda +
  xlab("") + ylab("") +
  ggtitle(paste("Muestras en el espacio (x,u), aceptación: ", mean(muestras$accept)))

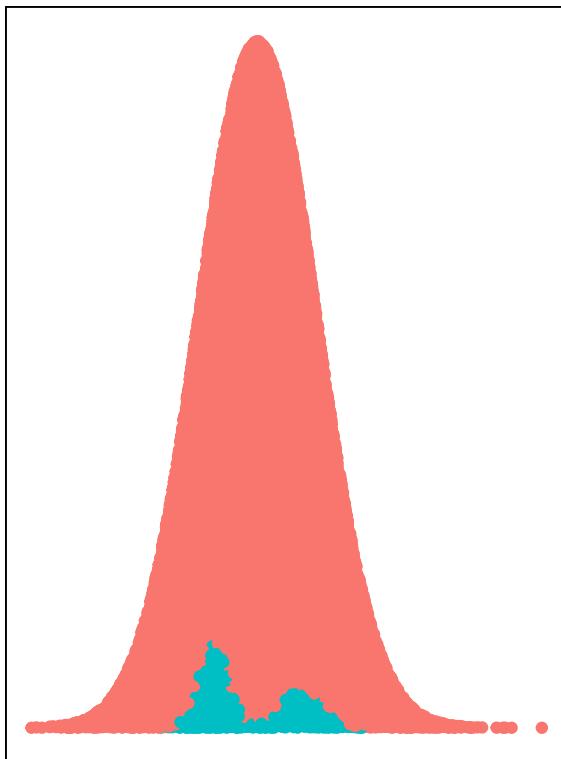
g2 <- muestras |>
  filter(accept == 1) |>
  ggplot(aes(value)) +
  geom_histogram(aes(y=..density..)) +
  geom_line(aes(value, objetivo(value)), color = "blue") +
  sin_lineas + sin_ejes + sin_leyenda +
  xlab("") + ylab("") +
  ggtitle("Histograma de las muestras generadas")

g1 + g2

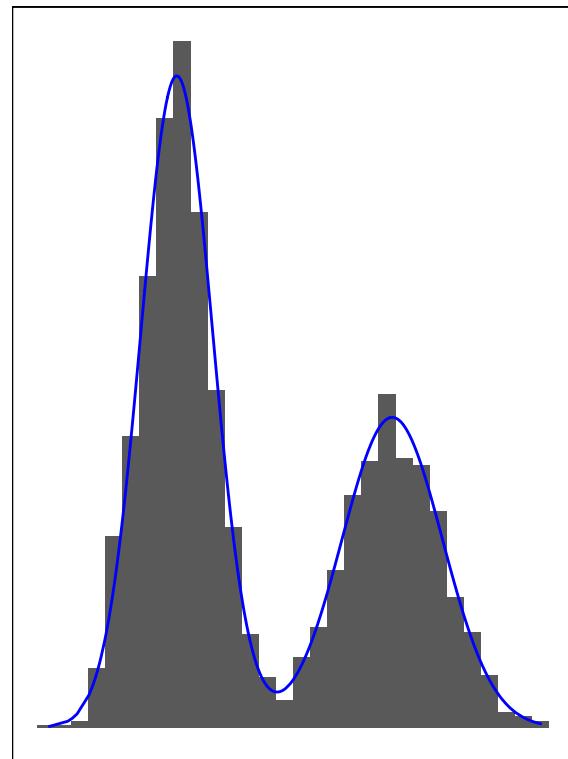
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

Muestras en el espacio (x,u), aceptación: 0.050-



Histograma de las muestras generadas



Lo que observamos es que aunque el muestreo se da de forma correcta de la distribución objetivo (el histograma parece coincidir con la densidad objetivo), el número de muestras rechazadas es mucho más grande. Por lo tanto el método, aunque correcto, es ineficiente, ya que se requieren muchas más simulaciones para obtener una muestra del mismo tamaño. (Notar que, de hecho, generamos 10 veces más muestras para que el histograma se viera bien).

Ahora veamos qué pasa con  $M$  chica. Tomemos  $M = 1$

```
M <- 1

muestreo_rechazo <- crea_rejection_sampling(objetivo, modelo.muestreo, M)

muestras <- muestreo_rechazo(50000) |>
  as.tibble() |>
  mutate(density = modelo.muestreo$density(value, log = FALSE))

g1 <- muestras |>
  ggplot(aes(value, auxiliar * modelo.muestreo$density(value, log = FALSE))) +
  geom_point(aes(color = factor(accept))) + sin_lineas + sin_ejes + sin_leyenda +
  xlab("") + ylab("") +
  ggtitle(paste("Muestras en el espacio (x,u), aceptación: ", mean(muestras$accept)))

g2 <- muestras |>
  filter(accept == 1) |>
  ggplot(aes(value)) +
  geom_histogram(aes(y=..density..)) +
  geom_line(aes(value, objetivo(value)), color = "blue") +
```

```

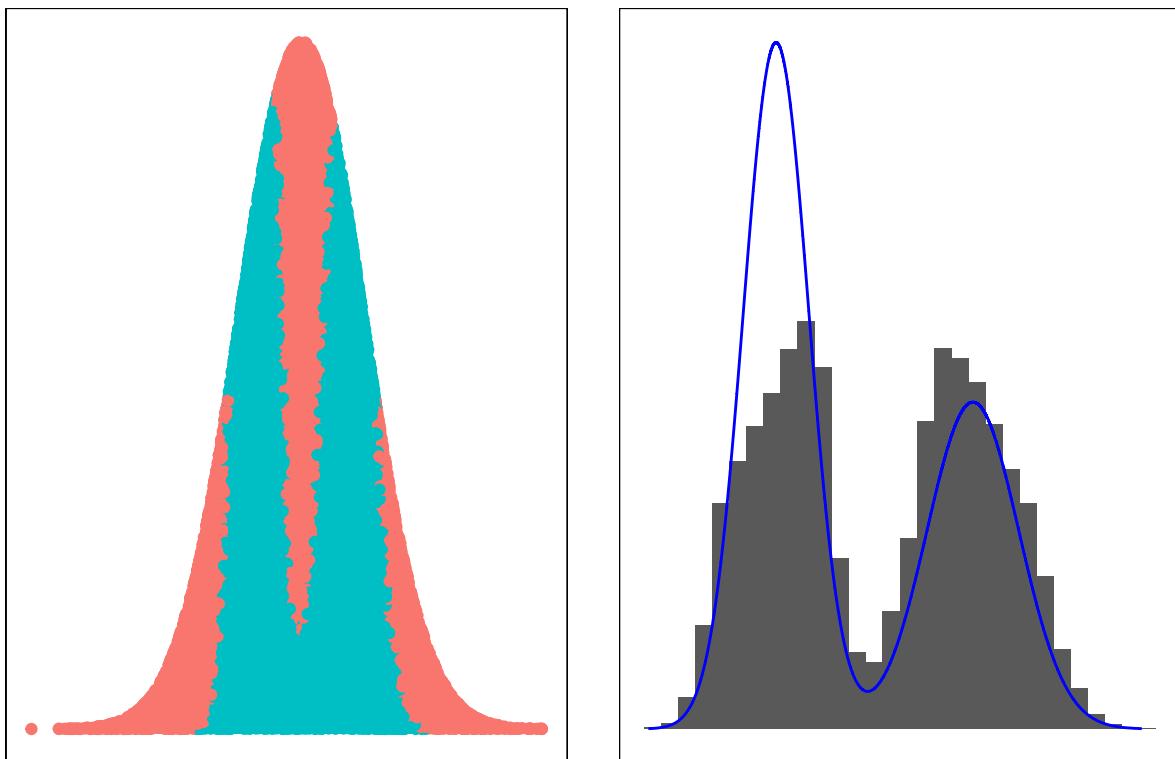
sin_lineas + sin_ejes + sin_leyenda +
xlab("") + ylab("") +
ggtitle("Histograma de las muestras generadas")

g1 + g2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

Muestras en el espacio (x,u), aceptación: 0.629: Histograma de las muestras generadas



Se observa que ahora el muestreo no se da de la forma correcta. La densidad está siendo truncada porque la aproximación normal ya no la cubre. Aunque estemos aceptando todos los puntos en el área truncada, no estamos generando suficientes puntos ahí para que la distribución coincida con la objetivo.

## Ejercicio 1

De nuevo, tomamos el código de (Garbuno 2022a), modificando solo los parámetros. Creamos la uniforme, la función que va a crear nuestra cadena de Markov, y la distribución objetivo.

```

ModeloUniforme <-
R6Class("ProbabilityModel",
  list(
    a = NA,
    b = NA,
    initialize = function(a = 0, b = 1){
      self$a = a
      self$b = b
    },
    sample = function(n = 1){

```

```

        runif(n, self$a, self$b)
    },
density = function(x, log = TRUE){
    dunif(x, self$a, self$b, log = log)
}
))

crea_cadena_markov <- function(objetivo, muestreo){
    function(niter){
        muestras <- matrix(nrow = niter, ncol = 2)
        ## Empezamos en algun lugar
        estado <- muestreo$sample()
        muestras[1,2] <- estado
        muestras[1,1] <- 1
        for (ii in 2:niter){
            ## Generamos un candidato
            propuesta <- estado + muestreo$sample()
            p_propuesta <- objetivo$density(propuesta, log = FALSE)
            p_estado <- objetivo$density(estado, log = FALSE)
            ## Evaluamos probabilidad de aceptar
            if (runif(1) < p_propuesta/p_estado) {
                muestras[ii, 1] <- 1 ## Aceptamos
                muestras[ii, 2] <- propuesta
            } else {
                muestras[ii, 1] <- 0 ## Rechazamos
                muestras[ii, 2] <- estado
            }
            estado <- muestras[ii, 2]
        }
        colnames(muestras) <- c("accept", "value")
        muestras
    }
}
}

objetivo <- ModeloNormal$new(mean = 4, sd = .6)

```

Intentamos con  $\omega = 0.01$ .

```

omega = 0.01
muestreo <- ModeloUniforme$new(a = -omega, b = omega)

mcmc <- crea_cadena_markov(objetivo, muestreo)
muestras <- mcmc(5000)

g1 <- muestras |>
    as.tibble() |>
    mutate(iter = 1:n()) |>
    ggplot(aes(iter, value)) +
    geom_line() + sin_lineas +
    ggtitle(paste("Trayectoria, eficiencia: ", mean(muestras[,2])))
    
g2 <- muestras |>
    as.tibble() |>
    ggplot(aes(value)) +

```

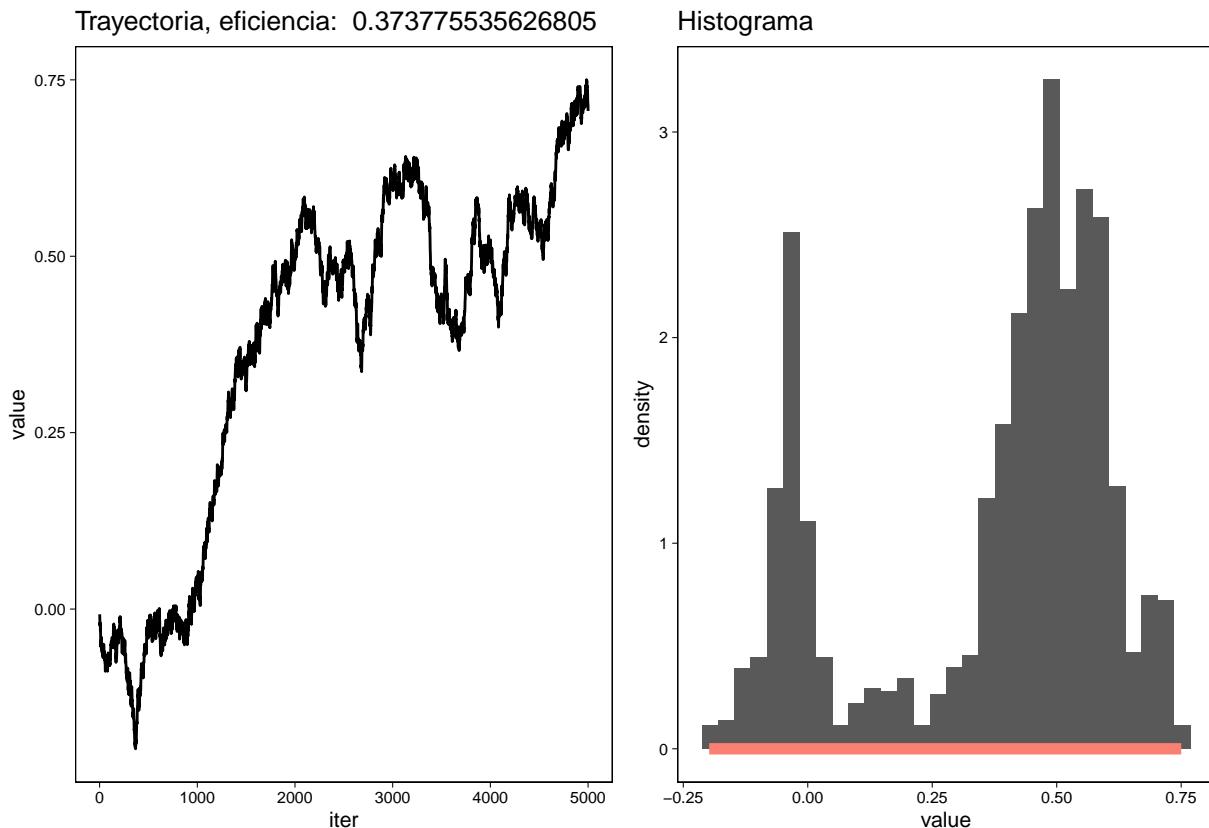
```

geom_histogram(aes(y = ..density..)) +
  stat_function(fun = objetivo$density,
    args = list(log = FALSE),
    color = "salmon",
    size = 2) + sin_lineas +
  ggtitle("Histograma")

g1 + g2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



```

omega = 100
muestreo <- ModeloUniforme$new(a = -omega, b = omega)

mcmc <- crea_cadena_markov(objetivo, muestreo)
muestras <- mcmc(5000)

g1 <- muestras |>
  as.tibble() |>
  mutate(iter = 1:n()) |>
  ggplot(aes(iter, value)) +
  geom_line() + sin_lineas +
  ggtitle(paste("Trayectoria, eficiencia: ", mean(muestras[,2])))

g2 <- muestras |>

```

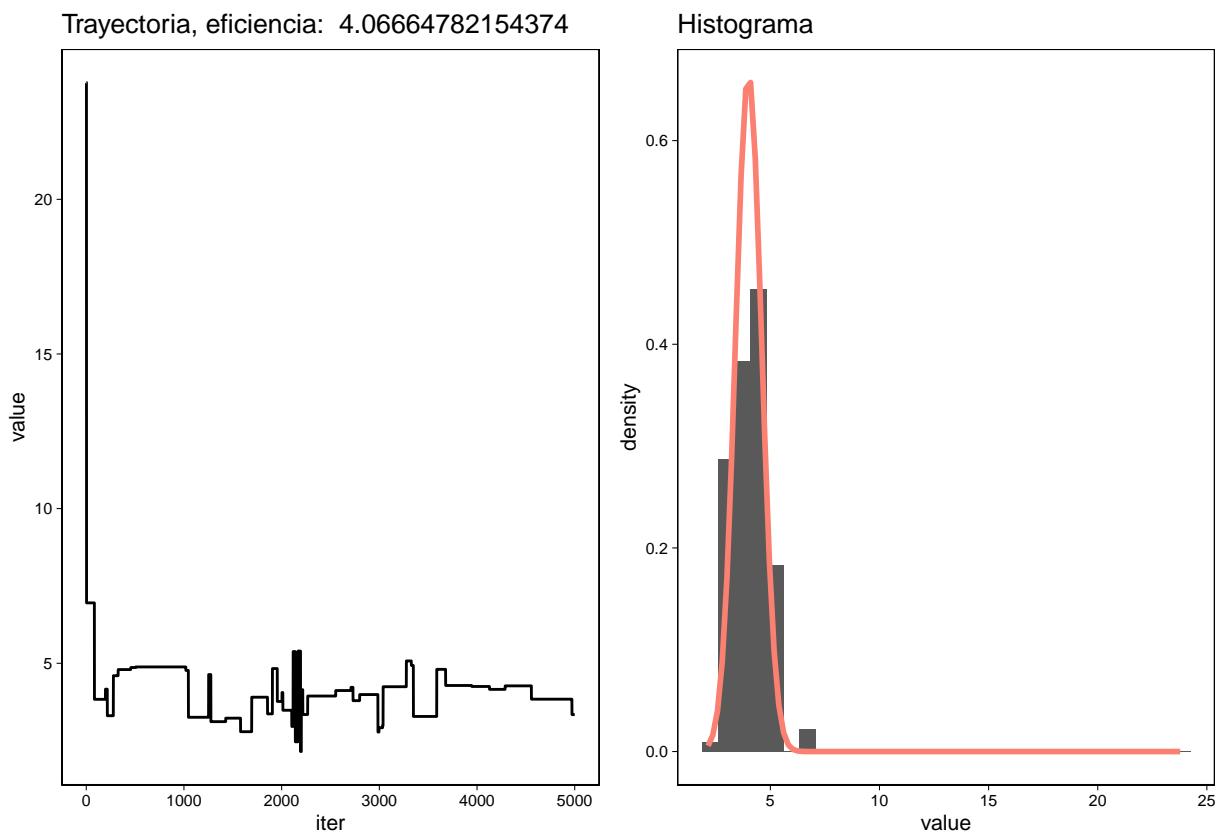
```

as.tibble() |>
ggplot(aes(value)) +
geom_histogram(aes(y = ..density..)) +
stat_function(fun = objetivo$density,
             args = list(log = FALSE),
             color = "salmon",
             size = 1) + sin_lineas +
ggtitle("Histograma")

g1 + g2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



Lo que se observa es que cada uno de los valores tiene sus ventajas y sus desventajas. Si  $\omega$  es chica, entonces la cadena va dando pasos pequeños hacia la distribución objetivo. La convergencia se da muy lentamente ya que la muestra se tarda mucho en llegar a los valores más probables bajo la distribución objetivo. De hecho, con  $\omega = 0.01$  podemos ver que con 5,000 iteraciones la cadena ni siquiera se acerca a los valores que debería tomar la distribución objetivo más frecuentemente. El histograma ni siquiera está localizado cerca de los valores que la distribución debería tomar.

Por otro lado, si tenemos un valor grande de  $\omega$ , la propuesta en cada paso va a estar lejos de la anterior, pero esto va a provocar que la mayoría de los valores sean rechazados, ya que la propuesta es a valores que serían muy poco probables bajo la distribución objetivo. La gráfica está compuesta por muchos segmentos planos y unos pocos saltos.

Lo ideal sería tener un valor de  $\omega$  intermedio, para que la cadena converja rápidamente al objetivo pero que no tengamos tantos valores rechazados. Otra opción podría ser comenzar con un valor grande que vaya

disminuyendo, para acercarnos rápidamente a los valores más probables en un principio y luego movernos a valores más cercanos (similar a recorrido simulado).

## Ejercicio 2

De clase tenemos que:

$$\theta|Y = k \sim Beta(2 + k, 3 + 2 - k)$$

Vamos a utilizar el código de (Garbuno 2022a) que utilizamos la pregunta anterior, modificándolo para este nuevo modelo.

```
ModeloBeta <-
  R6Class("ProbabilityModel",
    list(
      alpha = NA,
      beta = NA,
      initialize = function(alpha = 0, beta = 1){
        self$alpha = alpha
        self$beta = beta
      },
      sample = function(n = 1){
        rbeta(n, self$alpha, self$beta)
      },
      density = function(x, log = TRUE){
        dbeta(x, self$alpha, self$beta, log = log)
      }
    )))
crea_cadena_markov_indep <- function(objetivo, muestreo){
  # A diferencia de la anterior, esta cadena no suma el valor
  # del estado actual (las propuestas no dependen del valor actual).
  function(niter){
    muestras <- matrix(nrow = niter, ncol = 2)
    ## Empezamos en algun lugar
    estado <- muestreo$sample()
    muestras[1,2] <- estado
    muestras[1,1] <- 1
    for (ii in 2:niter){
      ## Generamos un candidato
      propuesta <- muestreo$sample()
      p_propuesta <- objetivo$density(propuesta, log = FALSE)
      p_estado <- objetivo$density(estado, log = FALSE)
      q_propuesta <- muestreo$density(propuesta, log = FALSE)
      q_estado <- muestreo$density(estado, log = FALSE)
      ## Evaluamos probabilidad de aceptar
      if (runif(1) < (p_propuesta/p_estado) * (q_estado/q_propuesta)) {
        muestras[ii, 1] <- 1 ## Aceptamos
        muestras[ii, 2] <- propuesta
      } else {
        muestras[ii, 1] <- 0 ## Rechazamos
        muestras[ii, 2] <- estado
      }
    }
  }
}
```

```

        estado <- muestras[ii, 2]
    }
    colnames(muestras) <- c("accept", "value")
    muestras
}
}

```

Creamos la distribución objetivo, la posterior de  $\theta$  con  $Y = 1$ .

```

k <- 1
objetivo <- ModeloBeta$new(alpha = 2+k, beta = 5-k)

```

Ahora vamos a tratar de muestrear de esta distribución utilizando distintos valores de  $\alpha$  y  $\beta$ .

Consideremos primero que tomamos los mismos parámetros de la distribución de la que queremos muestrear:  $\alpha = 2 + k$  y  $\beta = 5 - k$  (aunque en la práctica no tendría sentido utilizar MCMC si ya podemos muestrear de la distribución objetivo).

```

alpha <- 2+k
beta <- 5-k
muestreo <- ModeloBeta$new(alpha = alpha, beta = beta)

mcmc <- crea_cadena_markov_indep(objetivo, muestreo)
muestras <- mcmc(5000)

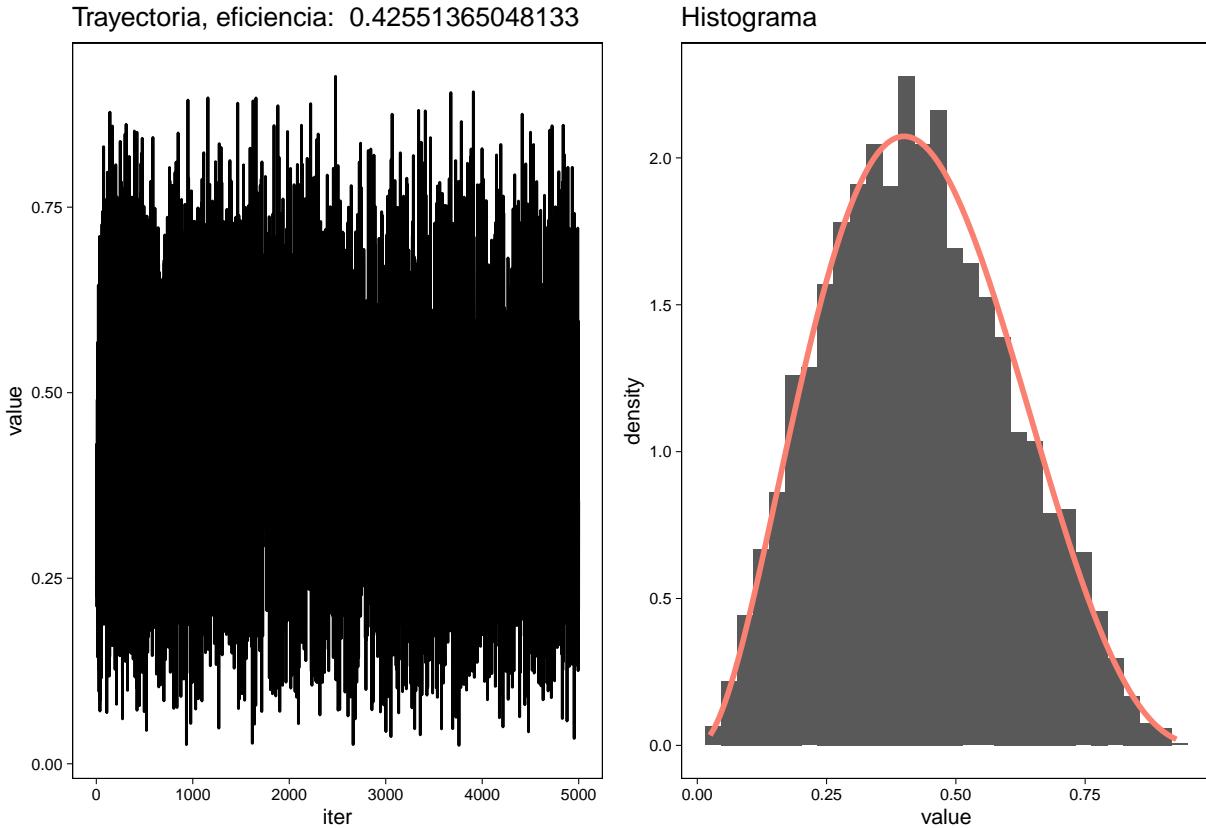
g1 <- muestras |>
  as.tibble() |>
  mutate(iter = 1:n()) |>
  ggplot(aes(iter, value)) +
  geom_line() + sin_lineas +
  ggtitle(paste("Trayectoria, eficiencia: ", mean(muestras[,2])))

g2 <- muestras |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = objetivo$density,
                args = list(log = FALSE),
                color = "salmon",
                size = 1) + sin_lineas +
  ggtitle("Histograma")

g1 + g2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



En efecto, vemos que el método funciona muy bien. El histograma coincide con la densidad objetivo y las observaciones no presentan correlación serial. La tasa de aceptación es de 100% (como la densidad objetivo es igual a la de muestreo,  $\pi = q$ , la probabilidad de aceptación es siempre 1).

Ahora consideraremos un caso en que los parámetros son distintos a los de la distribución objetivo

```
alpha <- 7
beta <- 1
muestreo <- ModeloBeta$new(alpha = alpha, beta = beta)

mcmc <- crea_cadena_markov_indep(objetivo, muestreo)
muestras <- mcmc(5000)

g1 <- muestras |>
  as.tibble() |>
  mutate(iter = 1:n()) |>
  ggplot(aes(iter, value)) +
  geom_line() + sin_lineas +
  ggtitle(paste("Trayectoria, eficiencia: ", mean(muestras[,2])))

g2 <- muestras |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = objetivo$density,
                args = list(log = FALSE),
                color = "salmon",
```

```

      size = 1) + sin_lineas + xlim(0, 1)
ggtitle("Histograma")

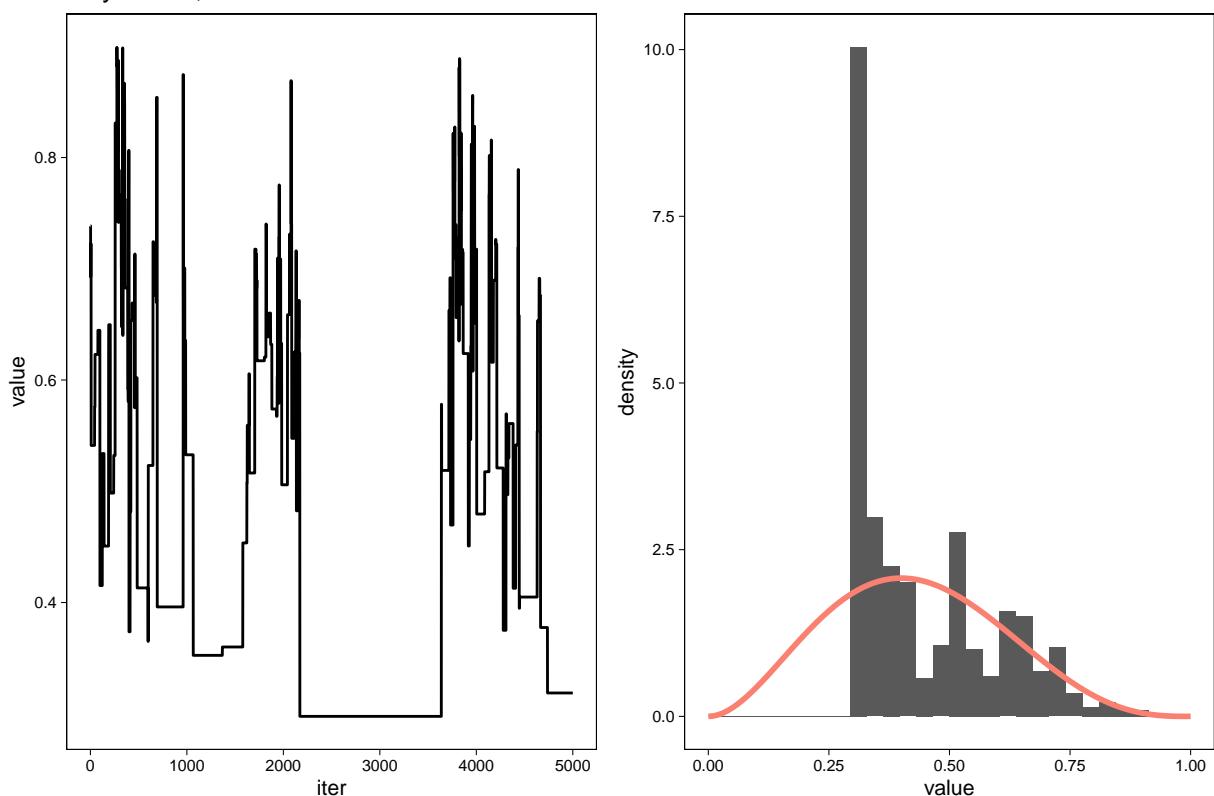
## $title
## [1] "Histograma"
##
## attr(,"class")
## [1] "labels"
g1 + g2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom_bar).

```

Trayectoria, eficiencia: 0.441953063508676



Como podemos observar, el histograma de la muestra ya no se asemeja mucho a la densidad objetivo. Está mucho más sesgada hacia la derecha, esto se debe a que la densidad de muestreo ( $Beta(7, 1)$ ) va a proponer muchos más valores más altos. Por la misma razón, cuando la cadena alcanza valores más bajos se queda ahí por un tiempo, rechazando los valores más altos. Se requerirían más simulaciones para alcanzar la convergencia.

El caso contrario es análogo con el sesgo hacia la izquierda.

```

alpha <- 1
beta <- 7
muestreo <- ModeloBeta$new(alpha = alpha, beta = beta)

mcmc <- crea_cadena_markov_indep(objetivo, muestreo)
muestras <- mcmc(5000)

```

```

g1 <- muestras |>
  as.tibble() |>
  mutate(iter = 1:n()) |>
  ggplot(aes(iter, value)) +
  geom_line() + sin_lineas +
  ggtitle(paste("Trayectoria, eficiencia: ", mean(muestras[,2])))

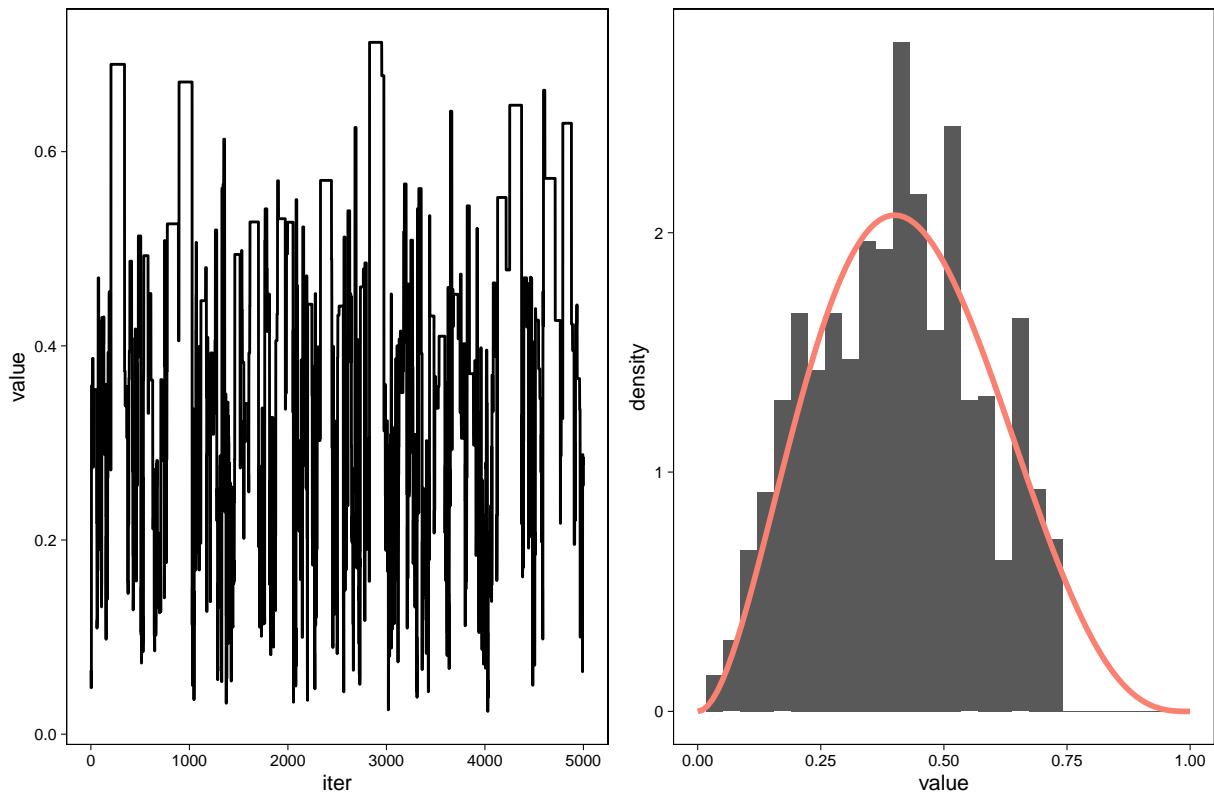
g2 <- muestras |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = objetivo$density,
                args = list(log = FALSE),
                color = "salmon",
                size = 1) + sin_lineas + xlim(0, 1)
  ggtitle("Histograma")

## $title
## [1] "Histograma"
##
## attr(,"class")
## [1] "labels"
g1 + g2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 2 rows containing missing values (geom_bar).

```

Trayectoria, eficiencia: 0.405292429286024



Por último, utilizaremos parámetros más cercanos a la distribución objetivo pero sin ser exactamente los mismos. Tomaremos los parámetros de la previa:  $\alpha = 2$  y  $\beta = 5$ .

```
alpha <- 2
beta <- 5
muestreo <- ModeloBeta$new(alpha = alpha, beta = beta)

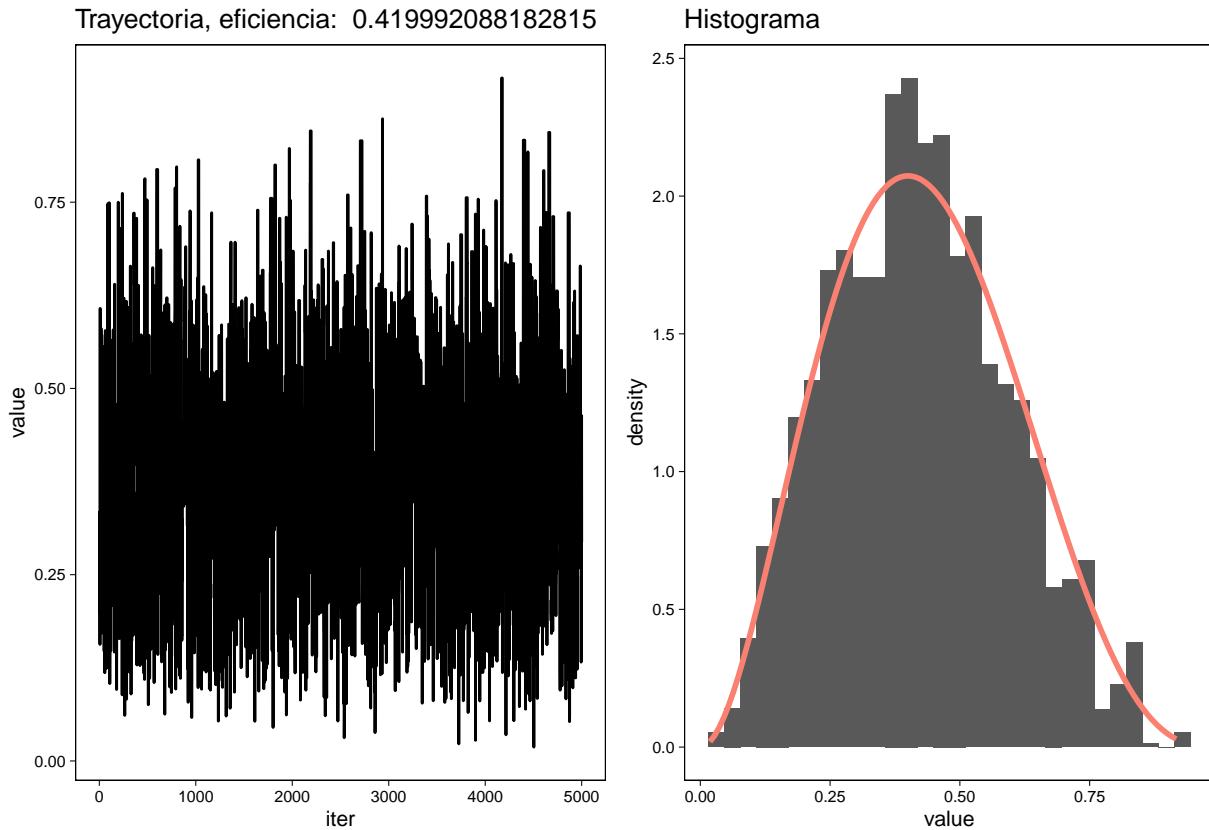
mcmc <- crea_cadena_markov_indep(objetivo, muestreo)
muestras <- mcmc(5000)

g1 <- muestras |>
  as.tibble() |>
  mutate(iter = 1:n()) |>
  ggplot(aes(iter, value)) +
  geom_line() + sin_lineas +
  ggtitle(paste("Trayectoria, eficiencia: ", mean(muestras[,2])))

g2 <- muestras |>
  as.tibble() |>
  ggplot(aes(value)) +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = objetivo$density,
                args = list(log = FALSE),
                color = "salmon",
                size = 1) + sin_lineas +
  ggtitle("Histograma")
```

```
g1 + g2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



El resultado es que la muestra es bastante cercana a la objetivo, aunque no tanto como la primera con los parámetros exactos.

**Conclusión:** El caso que mejor funcionó fue cuando los parámetros son los mismos que la distribución objetivo. En la práctica no tendría sentido utilizar MCMC si se puede muestrear de la distribución objetivo. Sin embargo, lo que sí tiene relevancia y que observamos aquí es que mientras más cercana sea la distribución propuesta a la objetivo, más eficiente va a ser el algoritmo, menos valores van a ser rechazados y la convergencia va a ser más rápida.

## Referencias

- Chang, Winston. 2013. *R Graphics Cookbook*. <http://www.cookbook-r.com/>.
- De la Vega, Jorge F. 2021. “Introducción a Markov Chain Monte Carlo (Notas Para La Clase de Simulación).” ITAM.
- Garbuno, Alfredo. 2022a. “Código de Soporte MCMC.” <https://github.com/agarbuno/modelacion-bayesiana/blob/spring-2022/rscripts/02-mcmc.R>.
- . 2022b. “Notas de Clase MCMC.” <https://github.com/agarbuno/modelacion-bayesiana/blob/spring-2022/docs/02-mcmc.pdf>.
- Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. <http://r4ds.had.co.nz/>.