Teoría de la Información

Estimación de la entropía en series de datos

Vanessa Martínez Romero, Moisés Emiliano Arellano Ávila y Fernando Gómez Perera

25/11/2020

Profr. Julio César Ramírez Pacheco

El concepto de entropía

La entropía se puede interpretar como un indicativo de la complejidad en una serie de datos aleatorios $\{X_1, X_2, \ldots, X_N\}$. La entropía usualmente se calcula utilizando una función de masa de probabilidad p_j que cumple con las siguientes propiedades:

- $p_j \geq 0$, $j \in 0, 1, \ldots, N$
- $\sum_{j=1}^{N} p_j = 1$

`R` tiene múltiples paquetes y funcionalidades que permiten estimar la `pmf` de un conjunto de datos como el descrito anteriormente. El histograma es una herramienta que permite estimar la `pmf` de un conjunto de datos. En el ejemplo que sigue se muestra la forma de estimar la `pmf` en `R`:

```
set.seed(1234)                          # Para hacer el análisis reproducible
datos     <- rnorm(512,0,1)             # Se generan 512 valores normales
histogram <- hist(datos, plot=FALSE) # Se calcula el histograma
pmf       <- histogram$counts/sum(histogram$counts)  # Se calcula la pmf
sum(pmf)                                # Se verifica que cumpla con las propiedades
```

```
## [1] 1
```

Ejercicios:

- Estimar la `pmf` utilizando el utilizando los paquetes `ASH` y `KernSmoooth`.

R=

Usando el paquete `ASH`, la `pmf` se calcula de la siguiente forma.

```
pacman::p_load(ash)

# Se calcula el histograma desplazado promediado univariado (en inglés, univariate averaged shifted his
invisible(capture.output(histogram_ash <- ash1(bin1(datos))))

pmf <- histogram_ash$y / sum(histogram_ash$y)
sum(pmf)
```

```
## [1] 1
```

Usando el paquete `KernSmooth`, la `pmf` se calcula de la siguiente forma.

```
pacman::p_load(KernSmooth)

# Estimación de densidad de kernel agrupada (Binned Kernel Density Estimate) sobre los datos
bkde_kernsmooth <- bkde(datos)

pmf <- bkde_kernsmooth$y / sum(bkde_kernsmooth$y)
sum(pmf)
```

```
## [1] 1
```

- ¿Cuál es la ventaja de utilizar los métodos anteriores sobre el histograma?

R=

Los métodos que ofrecen estos paquetes son mucho más robustos que el histograma para calcular la pmf. Además, permiten aproximar la función de distintas formas, ya sea usando distintos kernels o llevando a cabo la estimación para una función en 2D.

- Utilizando el comando `hist` y los paquetes `ASH` y `KernSmooth` verifique el tiempo requerido para estimar la densidad de una serie de datos Gaussianos con $\mu = 1$ y varianza $\sigma^2 = 1$ y longitudes $N = 2^i$, $i = 8, 9, 10, 11, \ldots 16$. (es necesario incluir un gráfico en `highcharter`)

R=

Para calcular el tiempo de ejecución, se usa la función `sys.time` en cada ejecución de las 3 funciones con los distintos valores de i.

Los tiempos de ejecución se grafican con el paquete `highcharter`.

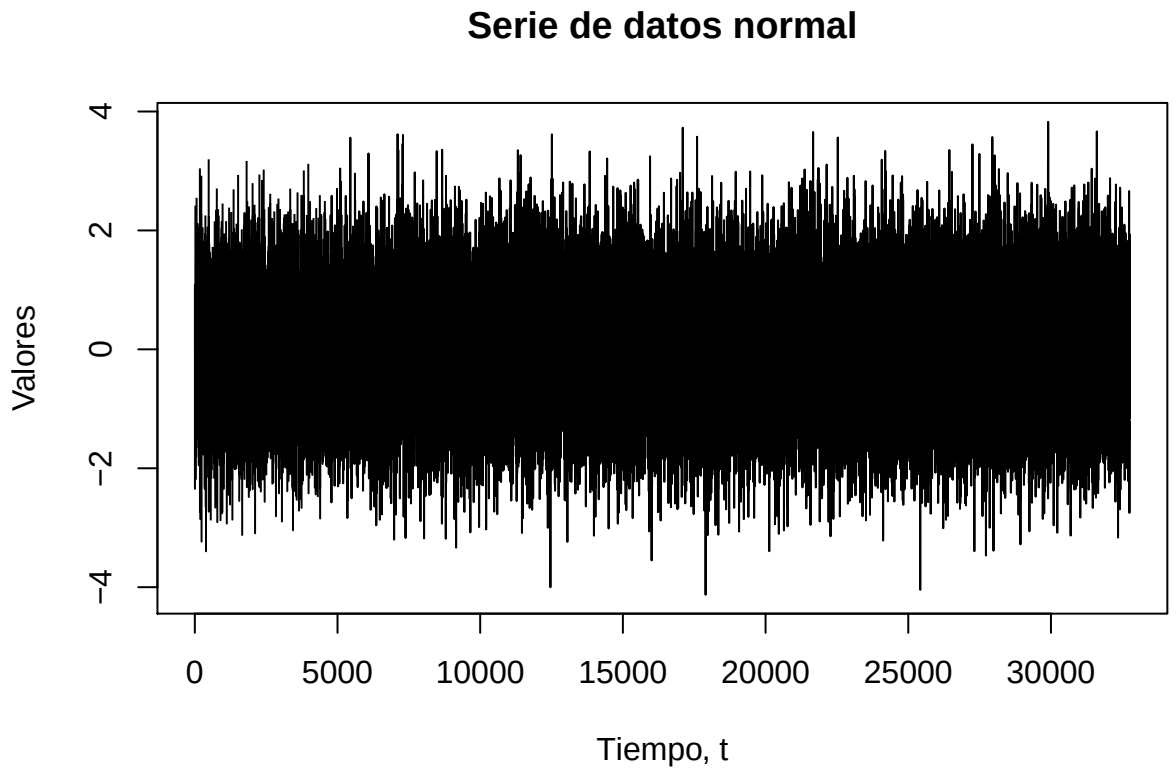
```
pacman::p_load(highcharter)

# Graficar los resultados usango highcharter
x <- 8:16
highchart() %>%
  hc_add_series(cbind(x, res_hist), name = "Función hist") %>%
  hc_add_series(cbind(x, res_ash1), name = "Función ash1") %>%
  hc_add_series(cbind(x, res_bkde), name = "Función bkde") %>%
  hc_add_theme(hc_theme_smpl()) %>%
  hc_title(text = "Tiempo de ejecución") %>%
  hc_subtitle(text = "IT0322 - Teoría de la información") %>%
  hc_xAxis(title = list(text = "i")) %>%
  hc_yAxis(title = list(text = "Tiempo"))
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

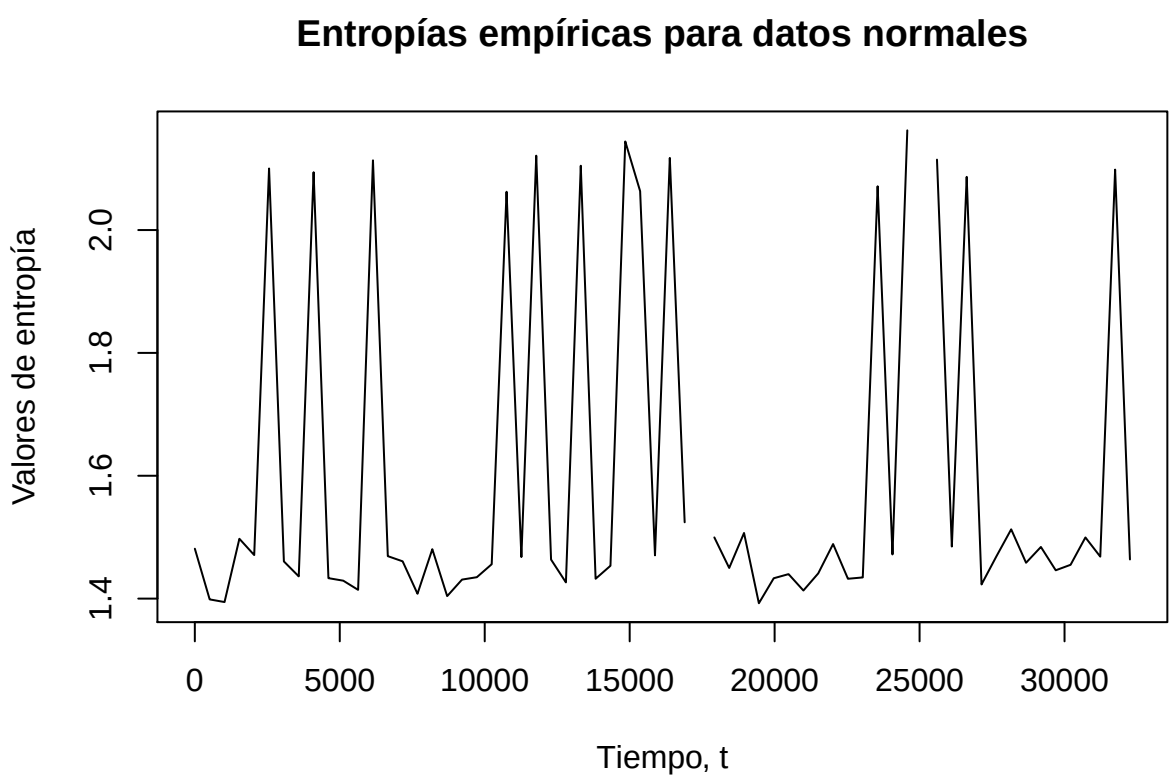
La entropía utilizando el histograma

Volviendo de nuevo al ejemplo anterior, podemos estimar la entropía de Shannon, utilizando la `pmf` obtenida mediante el histograma y así obtener un estimador empírico de la entropía de Shannon. A continuación mostramos la forma de obtener la entropía de una serie de datos obtenida en ventanas independientes o contiguas de longitud 512:

```
set.seed(1234)
datos        <- rnorm(32768)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  entropies[i] <- -1*sum(pmf*log(pmf))
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```



Preguntas

1. ¿Porqué existen valores discontinuos en la entropía?

R=

Porque existen puntos cuyos valores de pmf son 0, por lo que al calcular el logaritmo de la entropía sobre estos valores provoca que el valor resultante esté indefinido (en R esto sería obtener un `NaN`), provocando que estos valores se vean como discontinuidades en la gráfica.

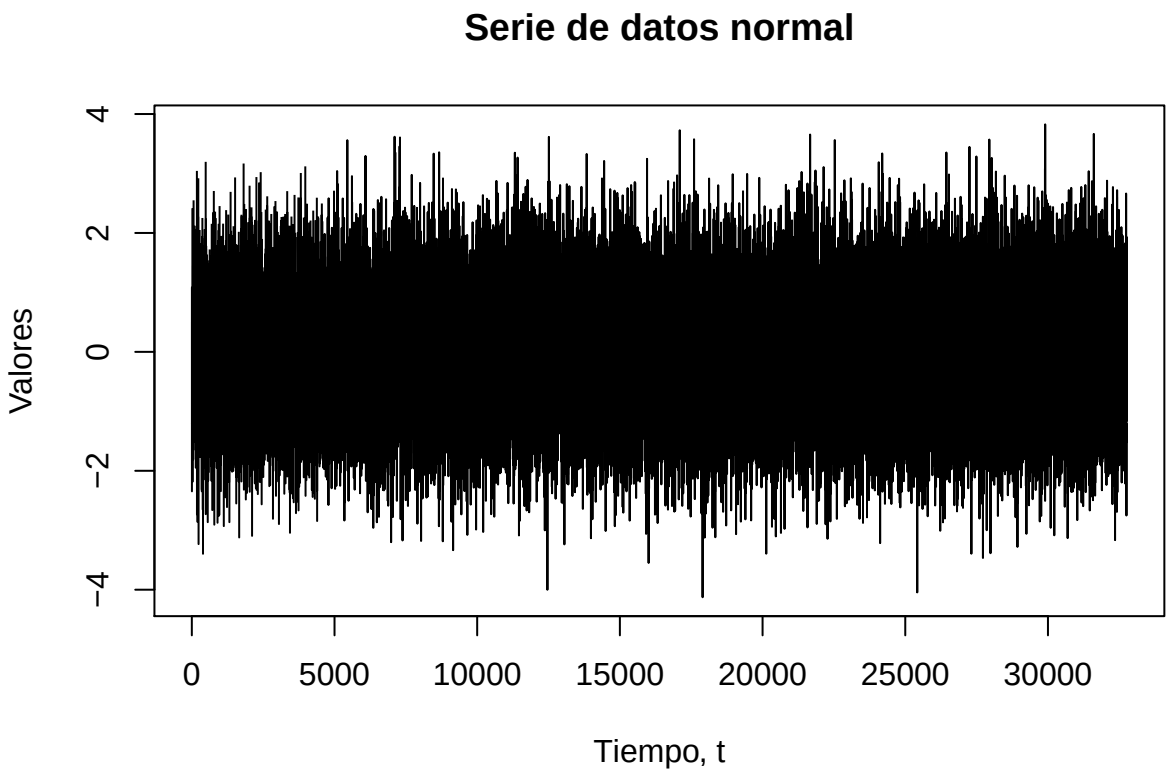
2. ¿Con qué código soluciona el problema de las discontinuidades?

R=

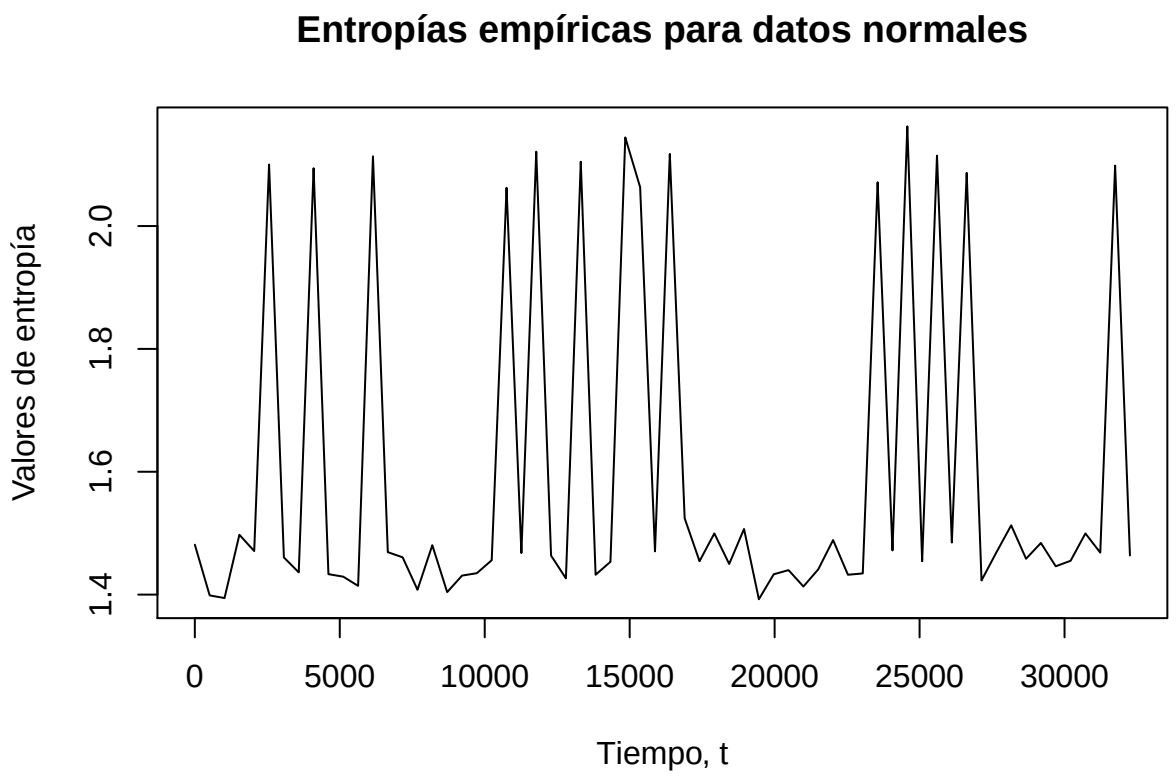
Si al momento de calcular la suma de la entropía usando la función `sum` excluyéramos estos valores con el parámetro `na.rm = TRUE` se eliminarían todos los valores `NaN` y por lo tanto ya no se mostrarían discontinuidades en la gráfica anterior.

Para comprobarlo, se ejemplifica esto tomando como base la gráfica anterior.

```
set.seed(1234)
datos        <- rnorm(32768)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Se excluye de la suma los valores que son NA
  entropies[i] <- -1*sum(pmf*log(pmf), na.rm = TRUE)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```



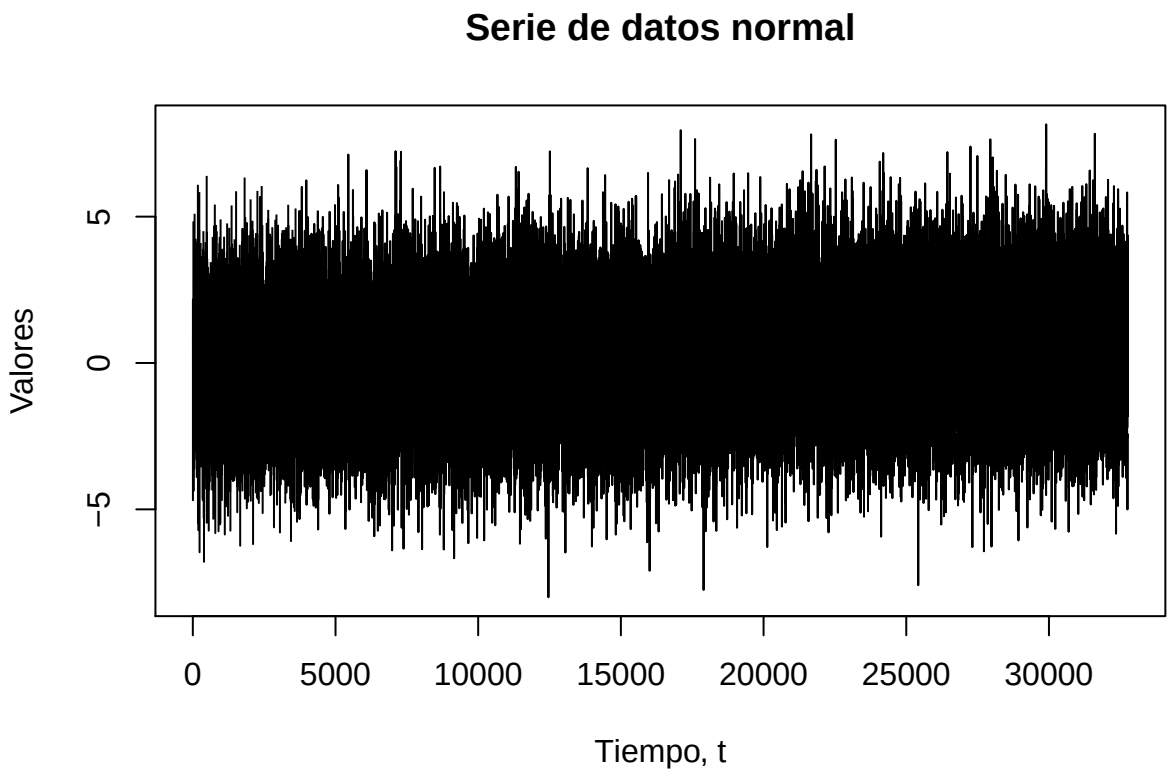
De igual forma, podría sumarse a todos los valores un número muy pequeño para que no se tengan probabilidades de 0, pero tampoco se modifiquen radicalmente las probabilidades obtenidas.

3. Ahora calcule la entropía de una serie de datos normales, denotados por X_t (con $\mu = 0$ y $\sigma = 2$) pero ahora añádanle (súmenle) una segunda función r_t, es decir, hallen la entropía de la serie $Y_t = X_t + r_t$ con r_t definida por:

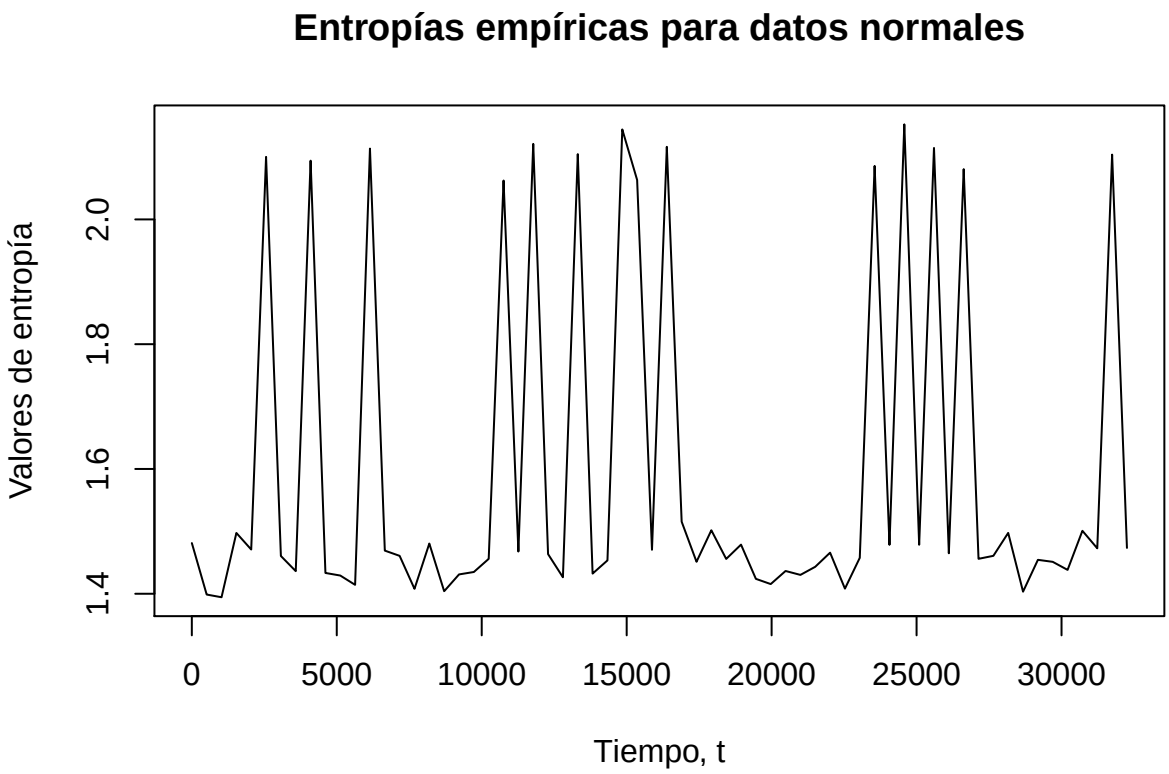
$$r_t = \begin{cases} \sigma/4 & t \geq 16384 \\ 0 & \text{otro caso} \end{cases}$$

R=

```r
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma / 4, 0)

set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, sigma = 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  entropies[i] <- -1*sum(pmf*log(pmf))
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

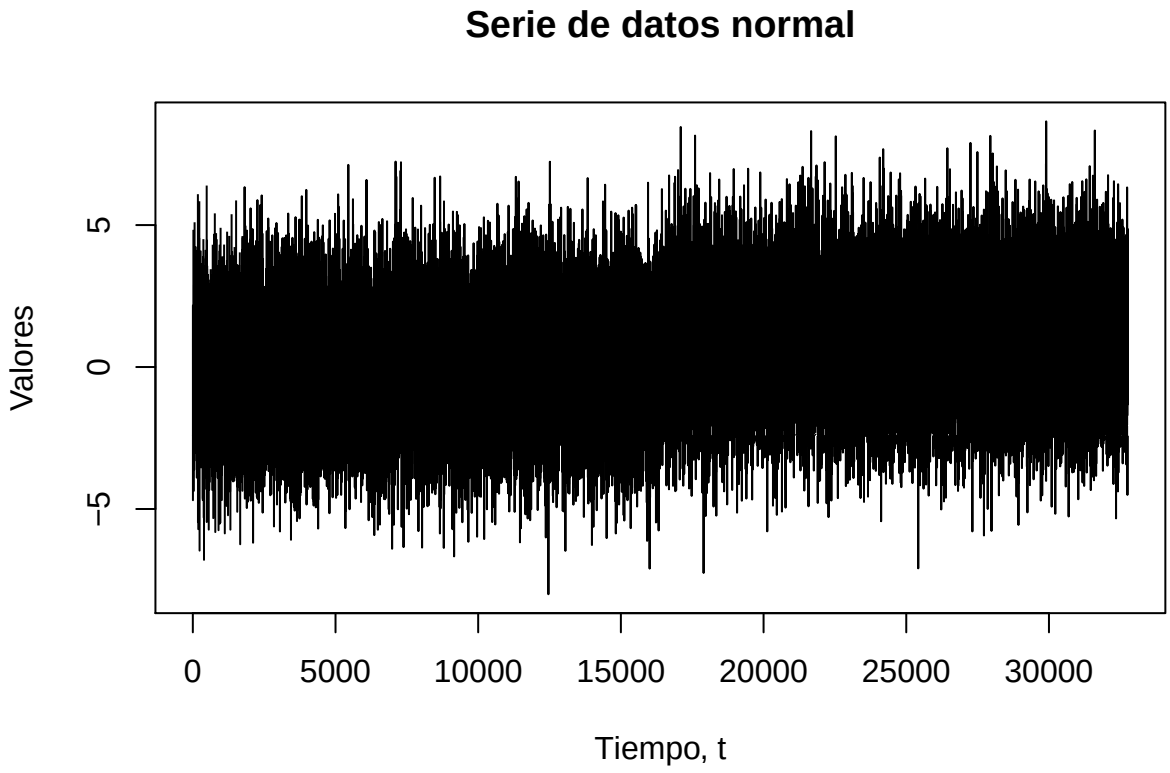


4. Repitan el paso 3 para r_t dada por:

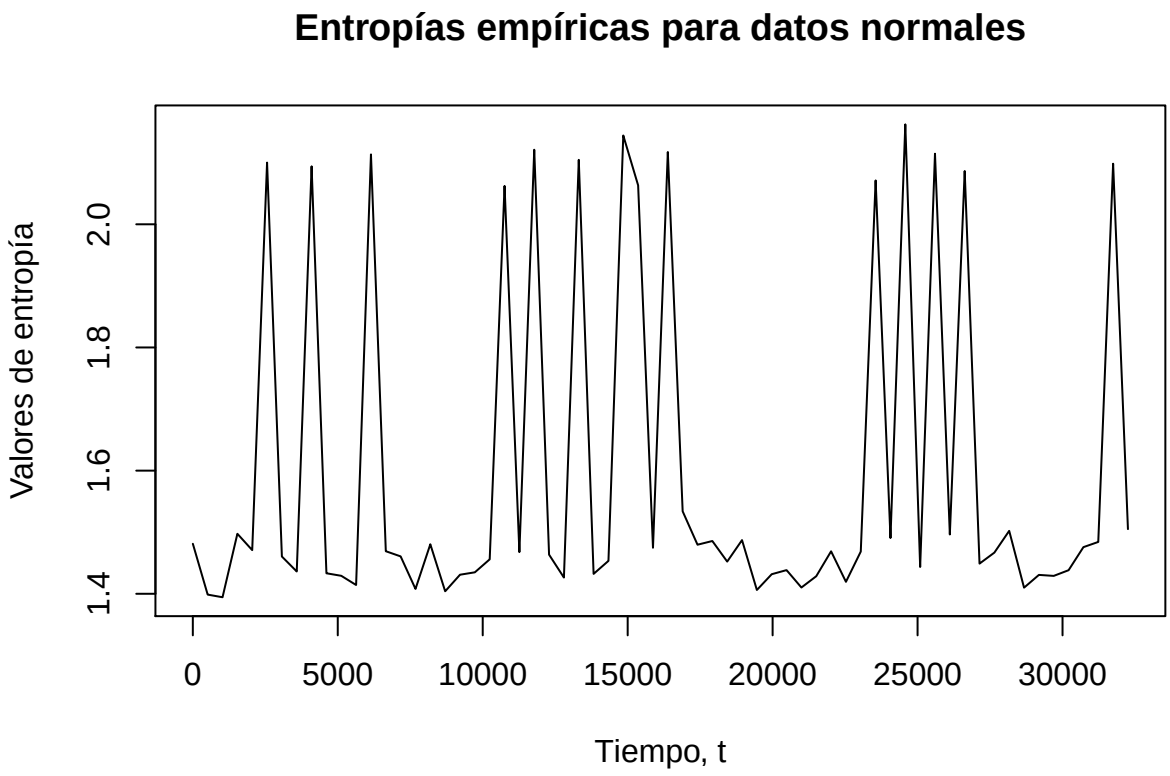
$$r_t = \begin{cases} \sigma/2 & t \geq 16384 \\ 0 & \text{otro caso} \end{cases}$$

$R=$

```r
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma / 2, 0)

set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, sigma = 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  entropies[i] <- -1*sum(pmf*log(pmf))
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

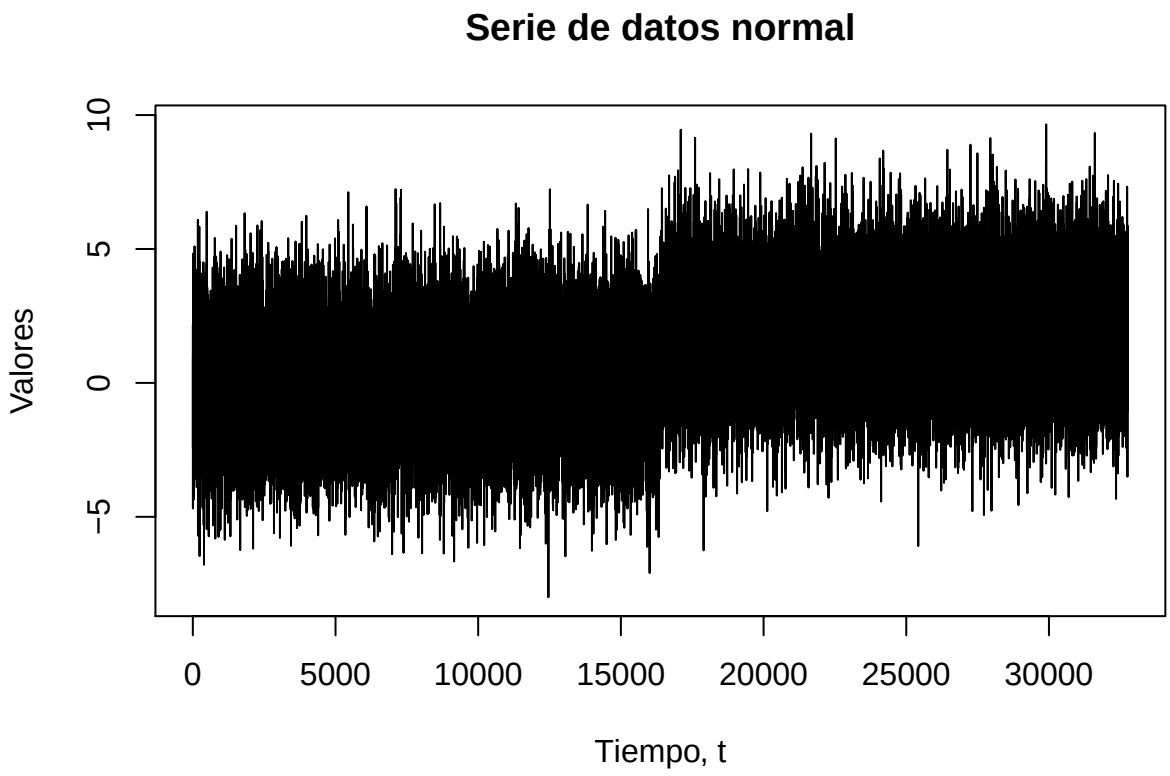


5. Repitan el paso 3 para r_t dada por:

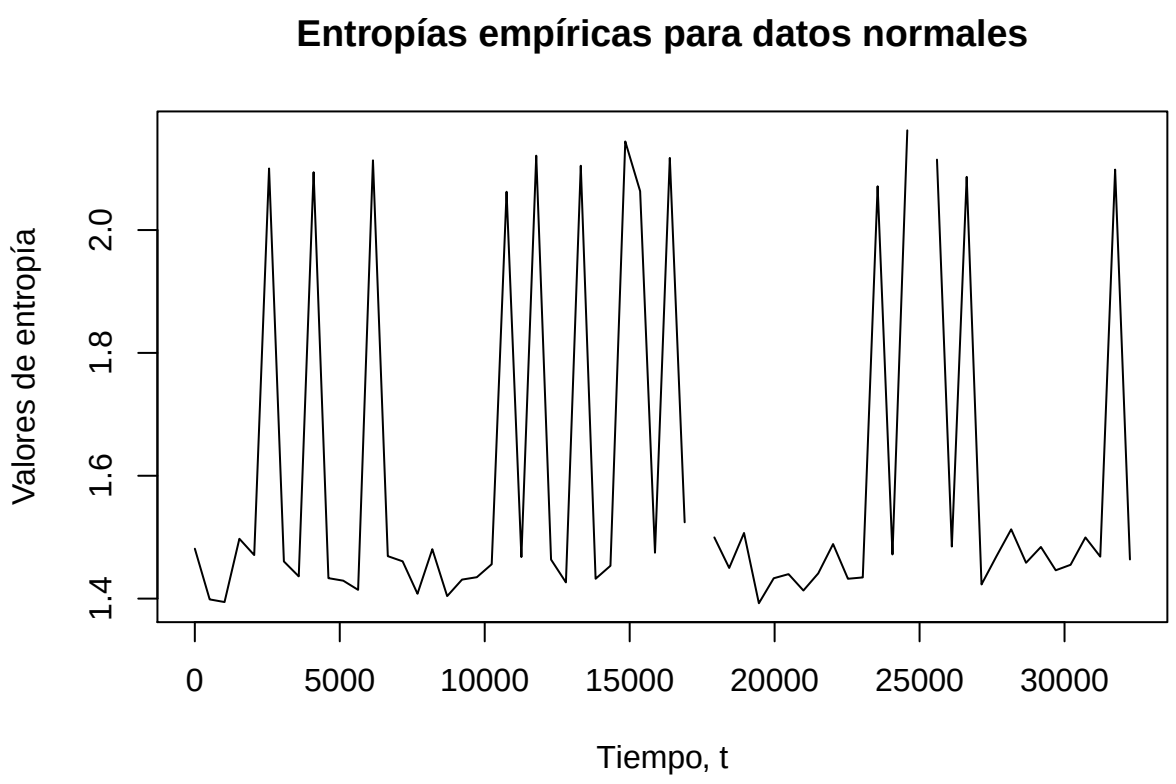
$$r_t = \begin{cases} \sigma & t \geq 16384 \\ 0 & \text{otro caso} \end{cases}$$

$R=$

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma, 0)

set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, sigma = 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  entropies[i] <- -1*sum(pmf*log(pmf))
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```



6. ¿Tiene algún efecto la longitud del salto en r_t en la forma de la entropía? Explique.

R=

En las primeras dos gráficas, donde r_t vale $\sigma/4$ y $\sigma/2$ respectivamente, la diferencia de longitud de salto en la forma de la entropía no es muy notoria ya que ambas gráficas son muy similares.

n el caso de la última gráfica donde r_t vale σ, la diferencia en la longitud de salto con respecto a las gráficas anteriores es más notoria y que además presenta discontinuidades.

En todos los casos las gráficas presentan diferencias en las crestas bajas, haciendo que los picos en estas zonas cambien en mayor o menor medida su forma. Y la mayoría de las variaciones no se encuentran en la entropía, sino en la forma de la serie de datos.

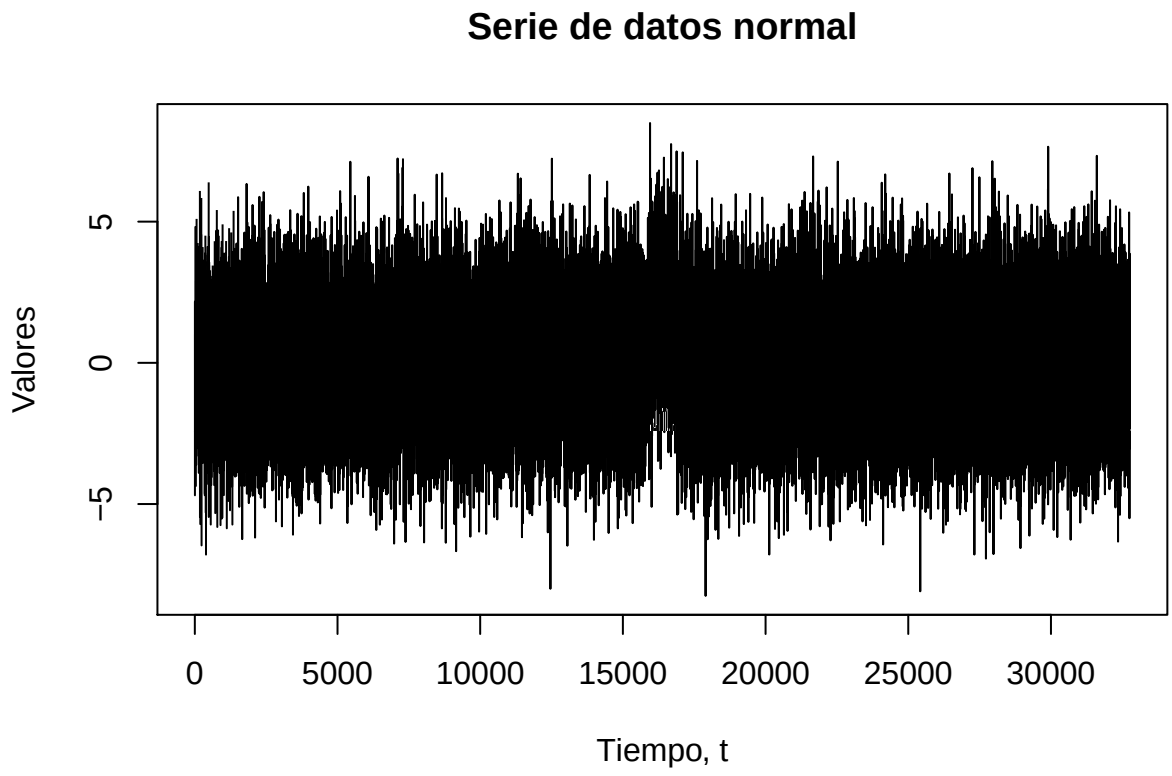
7. ¿Qué sucede ahora si r_t es de la forma:

$$r_t = \begin{cases} \sigma & 15872 \leq t \leq 16896 \\ 0 & \text{otro caso} \end{cases}$$

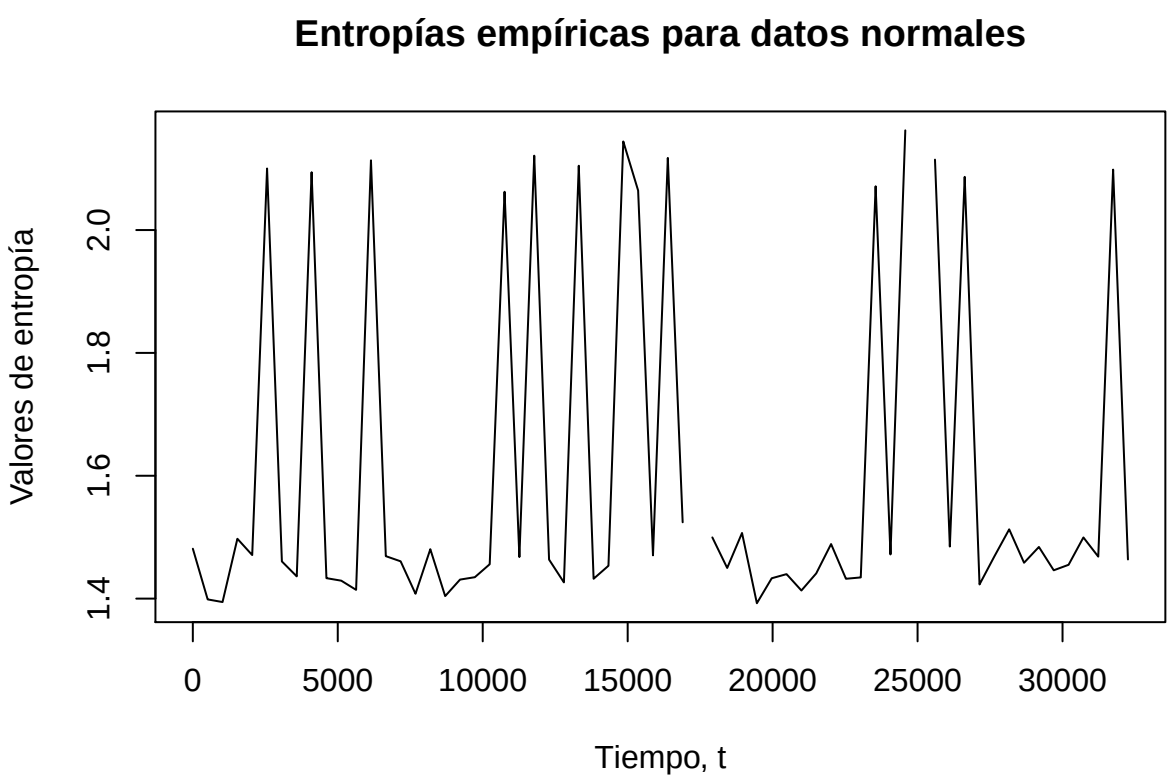
?

$R=$

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(15872 <= t & t<= 16896, sigma, 0)

set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  entropies[i] <- -1*sum(pmf*log(pmf))
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```



Respecto al gráfico anterior, la entropía tiene casi la misma forma. De hecho, las variaciones son muy pequeñas. Donde sí varía en mayor medida es en la forma de la serie de datos, que en general es muy diferente a todas las series de datos anteriores.

8. Repita los pasos 3-7 pero ahora usando la entropía de Havrda con parámetro $\alpha = 3$ y $\alpha = 9$. ¿Qué efecto tiene α?

R=

Primero definimos una función para la entropía de Havrda

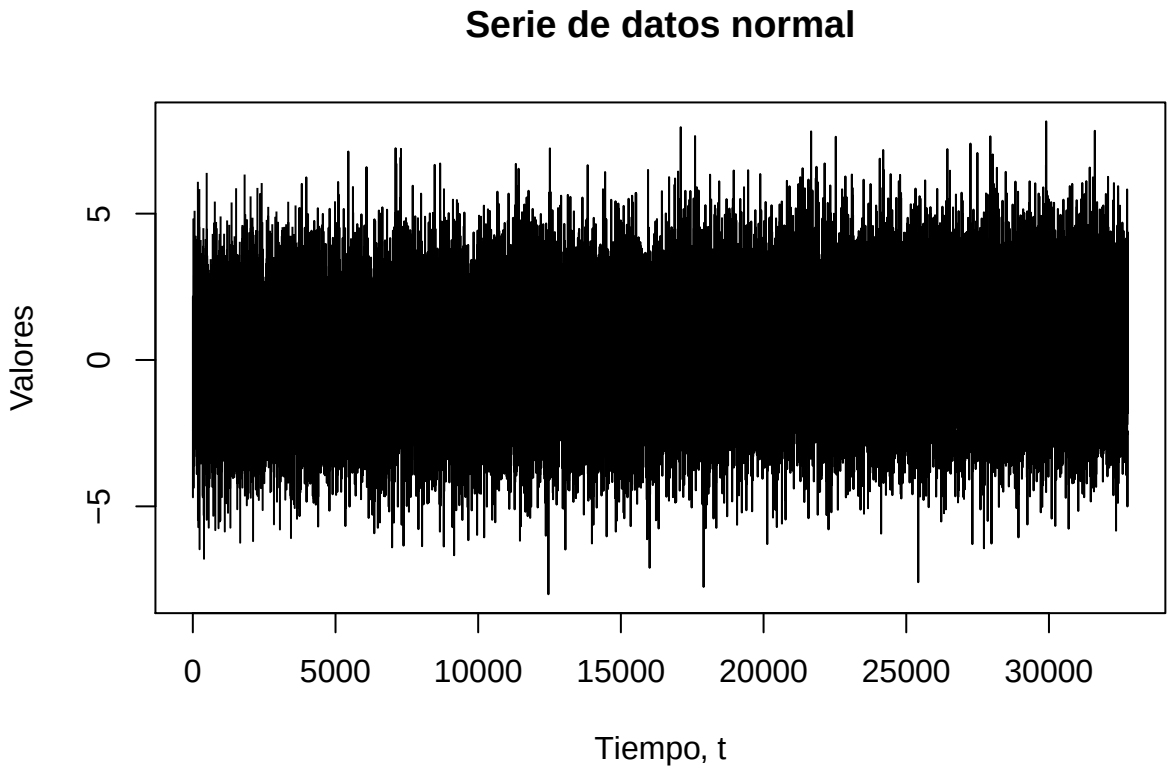
```
havrda <- function(pmf, alpha) sum(pmf^alpha - 1) / (2^(1 - alpha) - 1)
```

Repitiendo el paso 3, con ambos valores de alpha para la entropía de Havrda se obtiene lo siguiente:

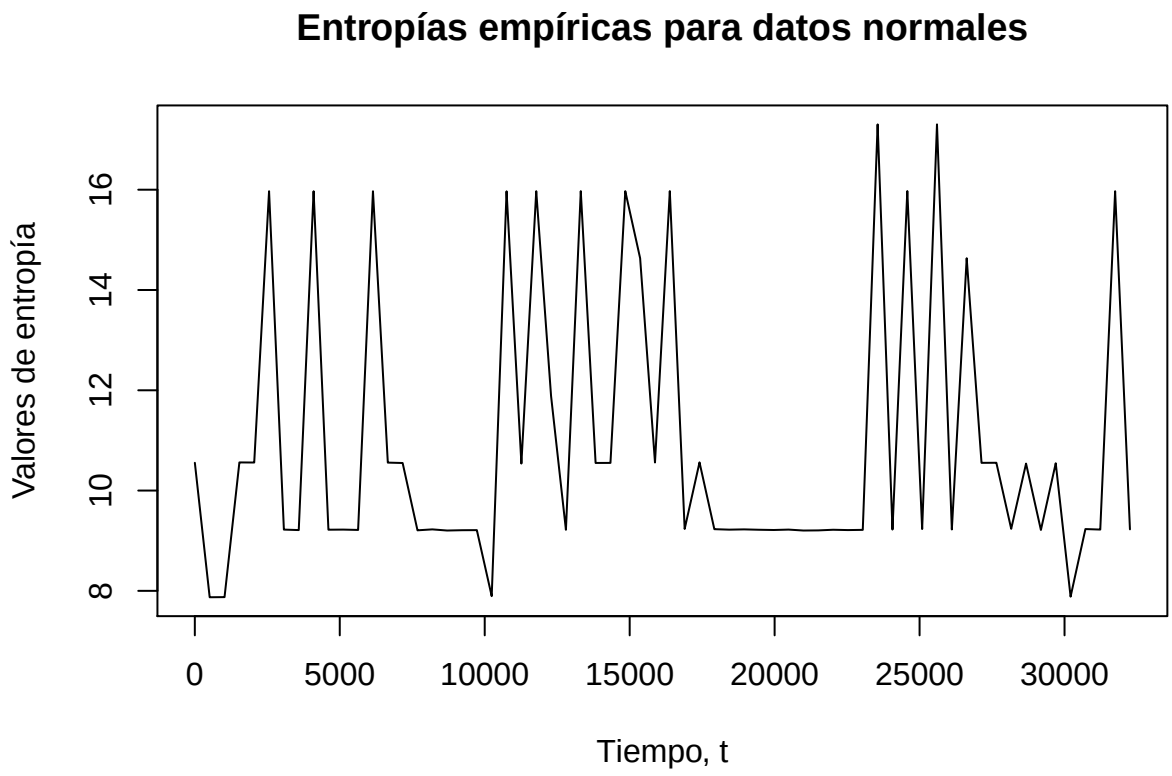
```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma / 4, 0)
```

Usando $\alpha = 3$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```

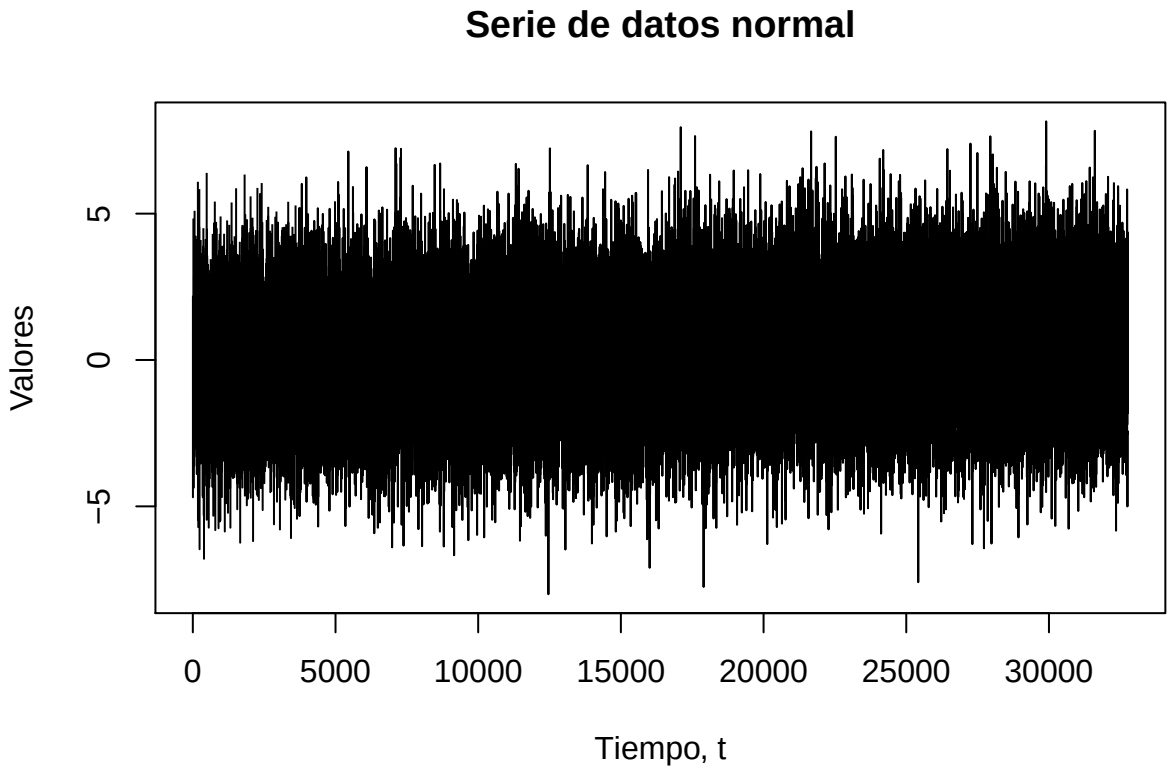



```r
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 3
  entropies[i] <- havrda(pmf, alpha = 3)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

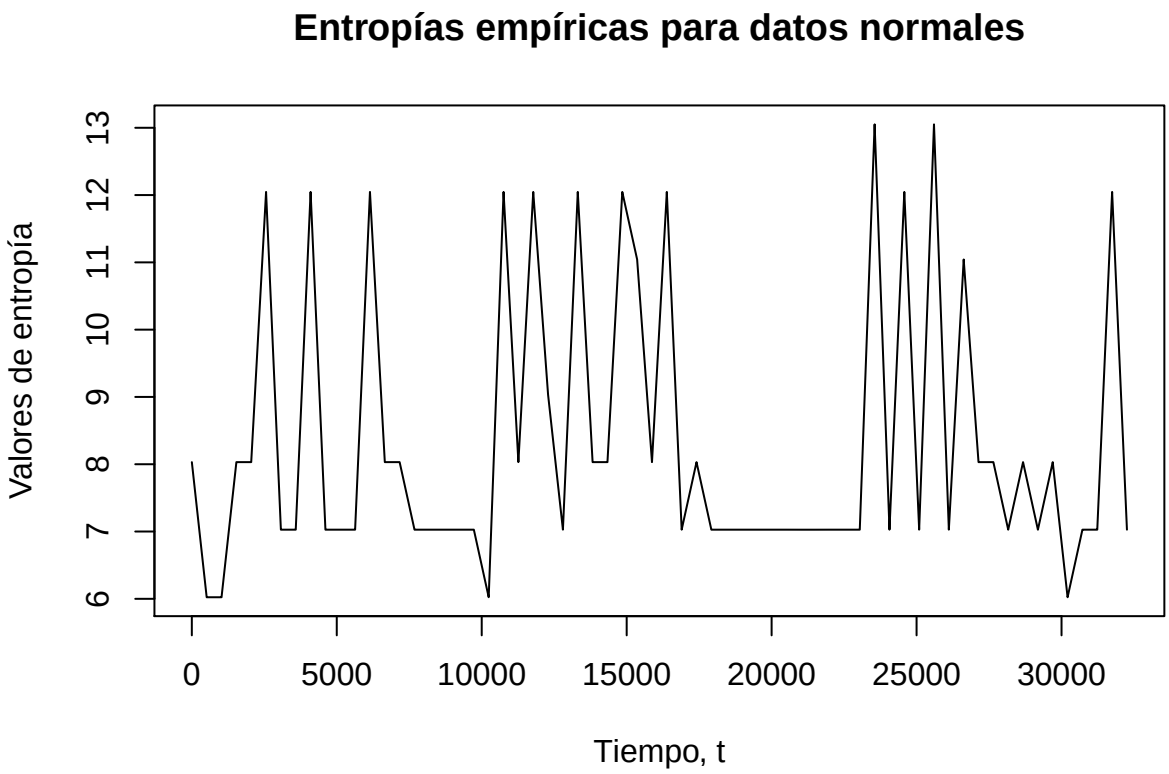


Usando $\alpha = 9$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 9
  entropies[i] <- havrda(pmf, alpha = 9)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

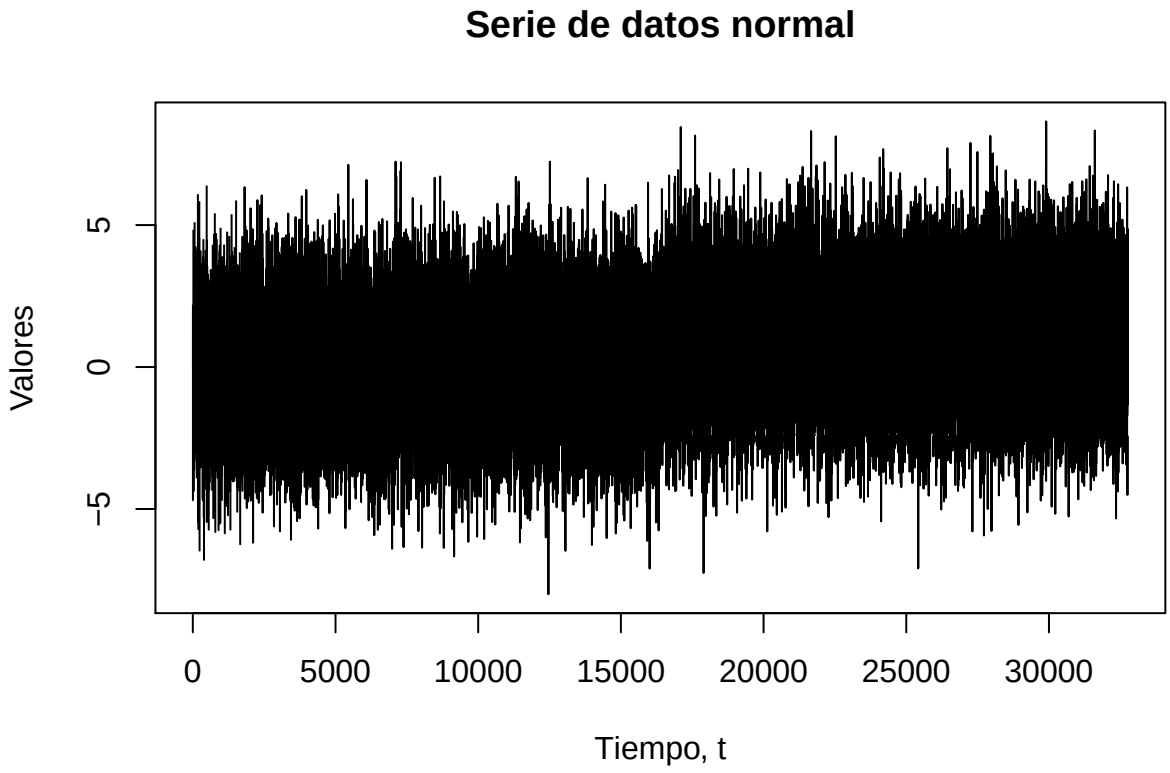


Repitiendo el paso 4, con ambos valores de alpha para la entropía de Havrda se obtiene lo siguiente:

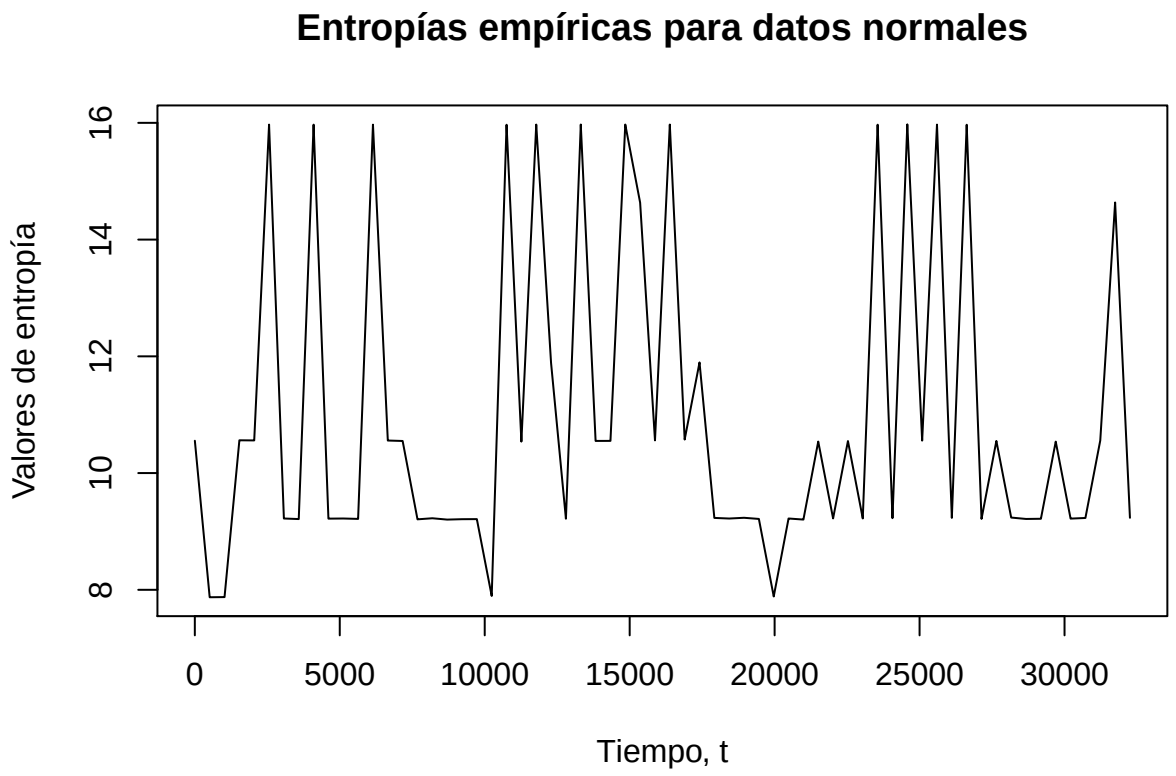
```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma / 2, 0)
```

Usando $\alpha = 3$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```

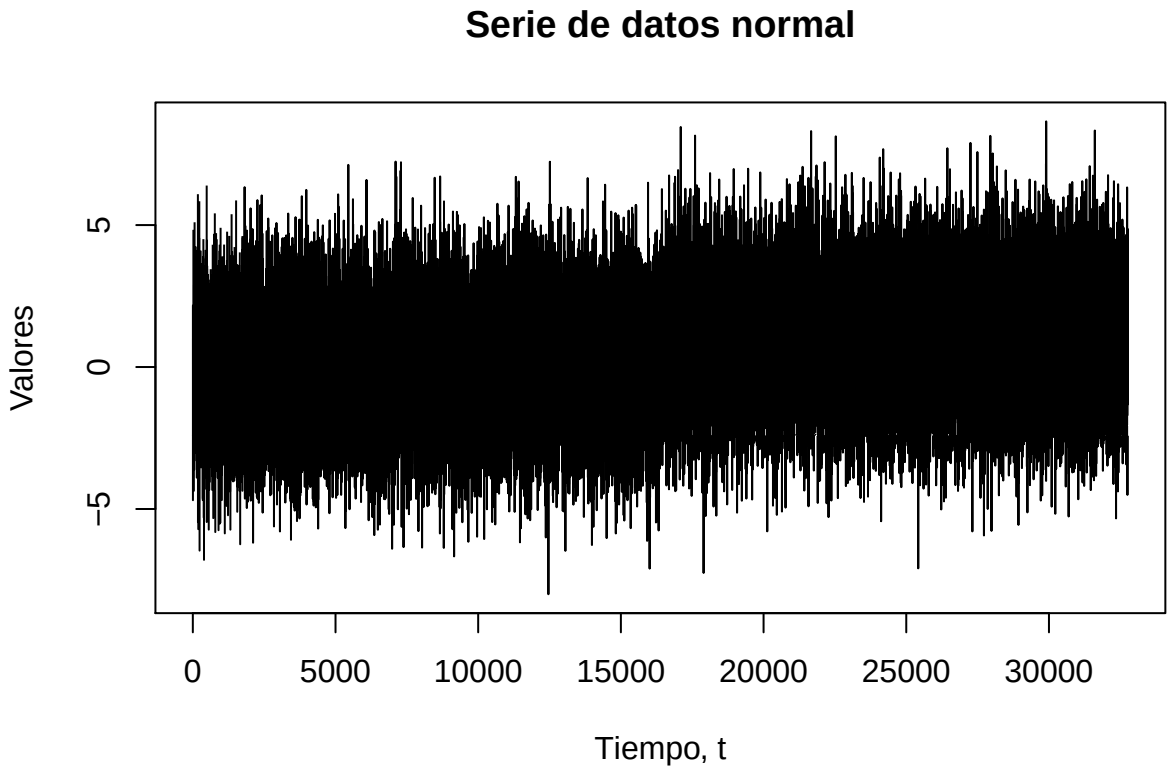



```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 3
  entropies[i] <- havrda(pmf, alpha = 3)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

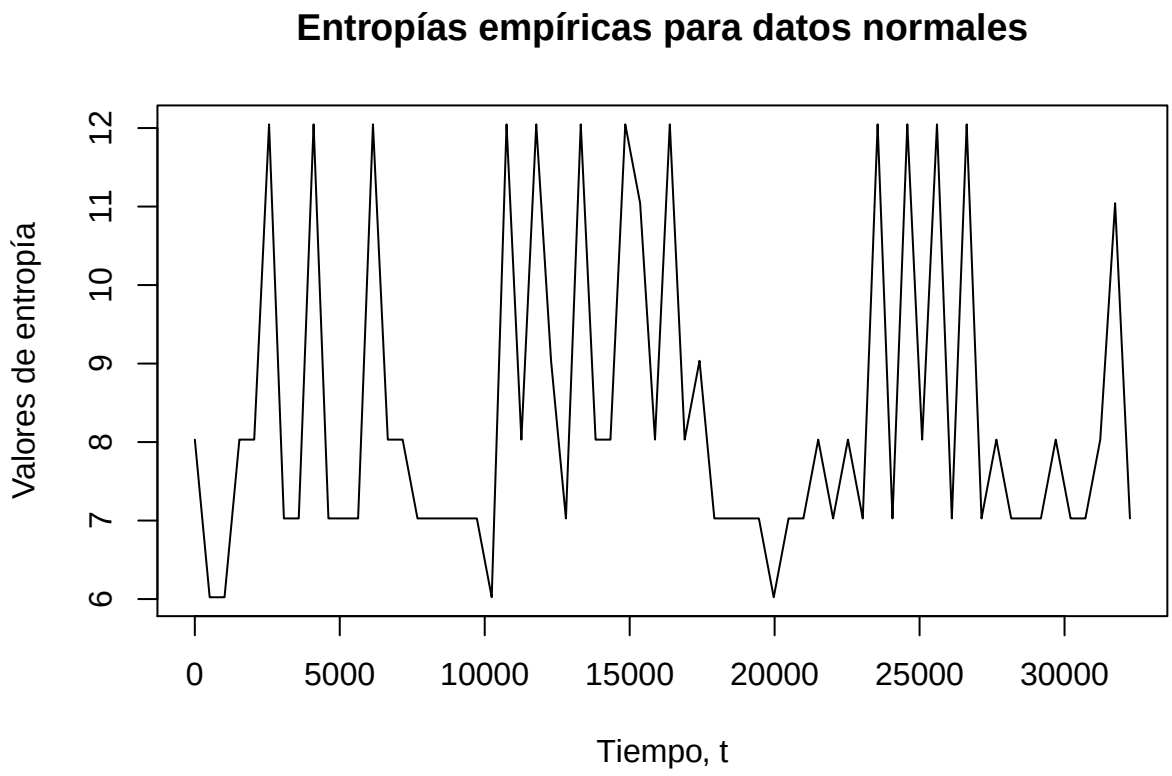


Usando $\alpha = 9$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 9
  entropies[i] <- havrda(pmf, alpha = 9)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

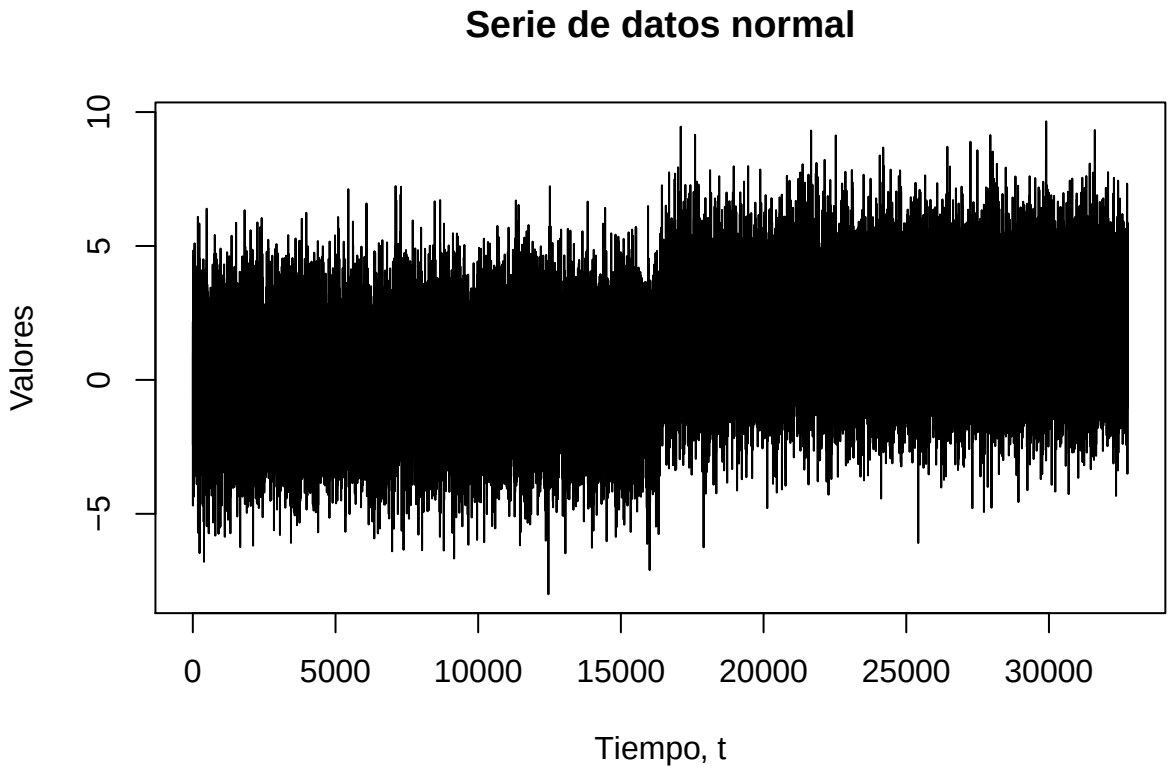


Repitiendo el paso 5, con ambos valores de alpha para la entropía de Havrda se obtiene lo siguiente:

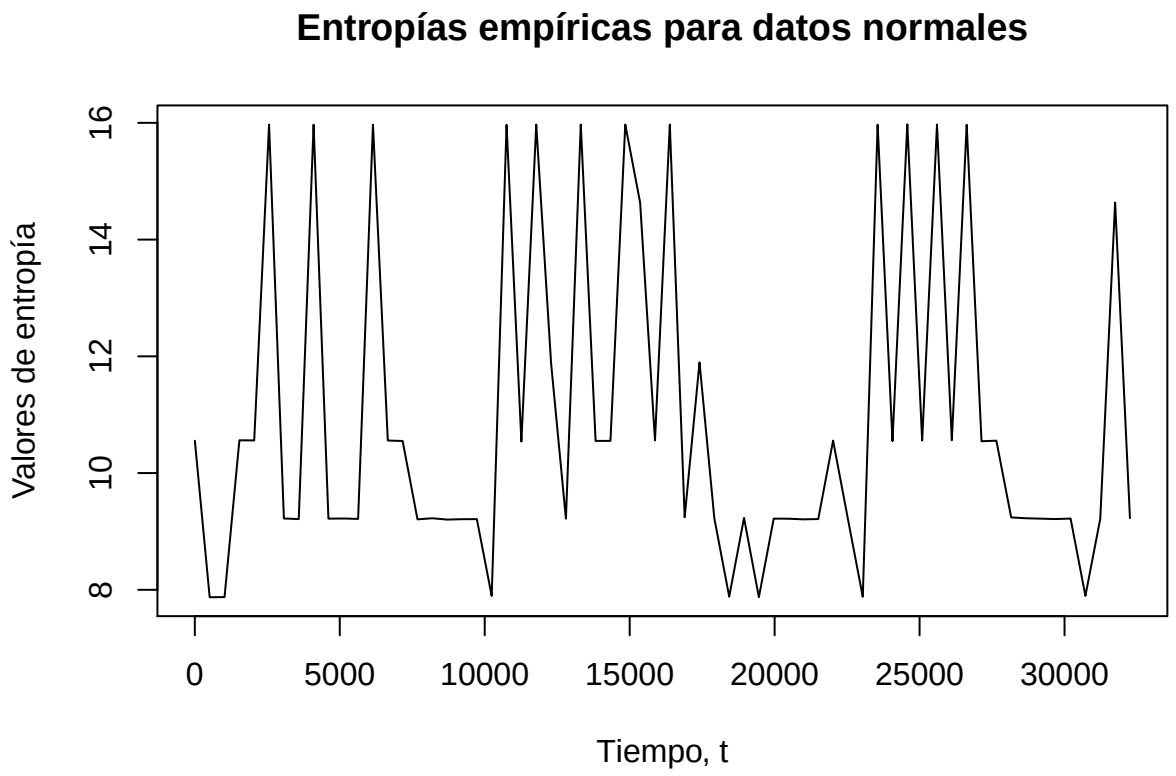
```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma, 0)
```

Usando $\alpha = 3$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```

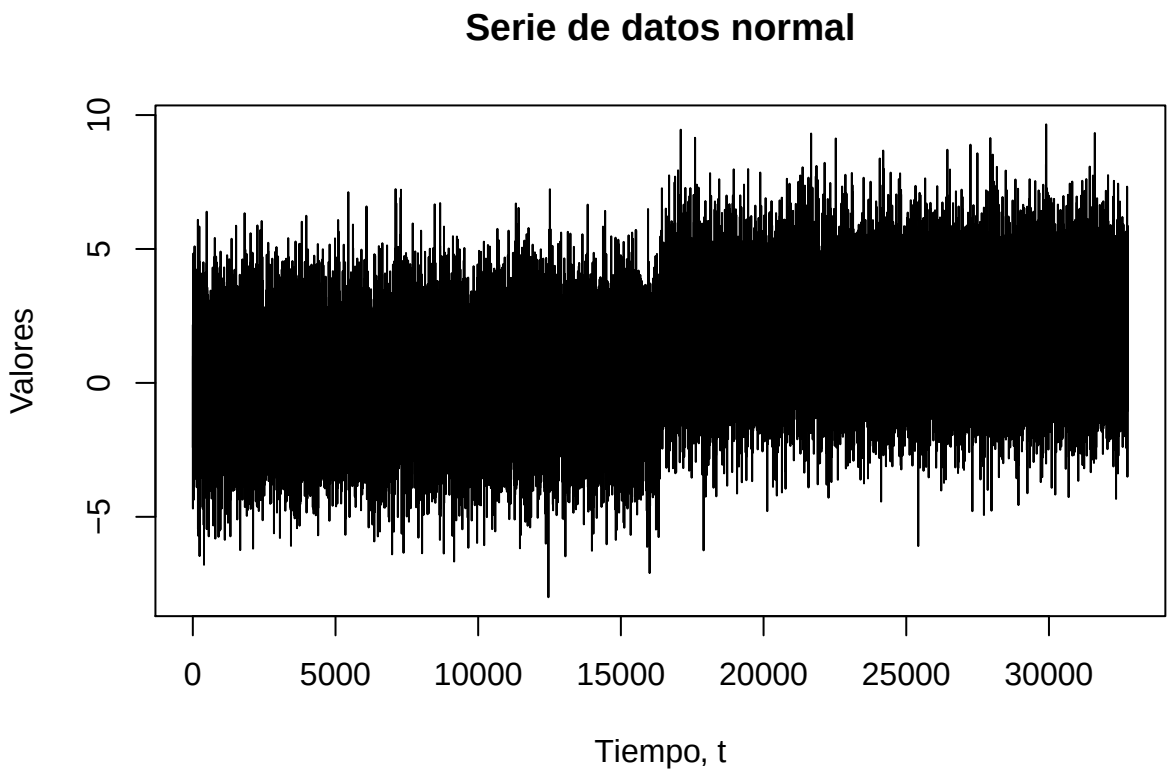



```r
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 3
  entropies[i] <- havrda(pmf, alpha = 3)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

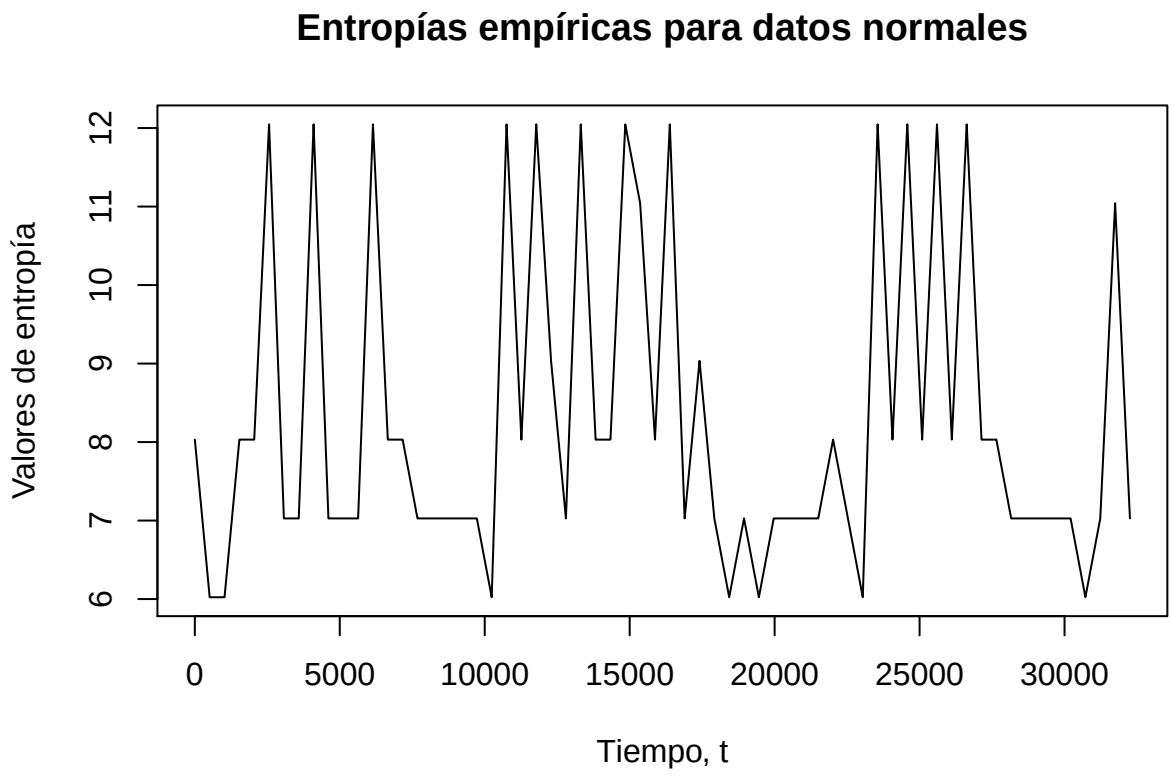


Usando $\alpha = 9$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 9
  entropies[i] <- havrda(pmf, alpha = 9)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```



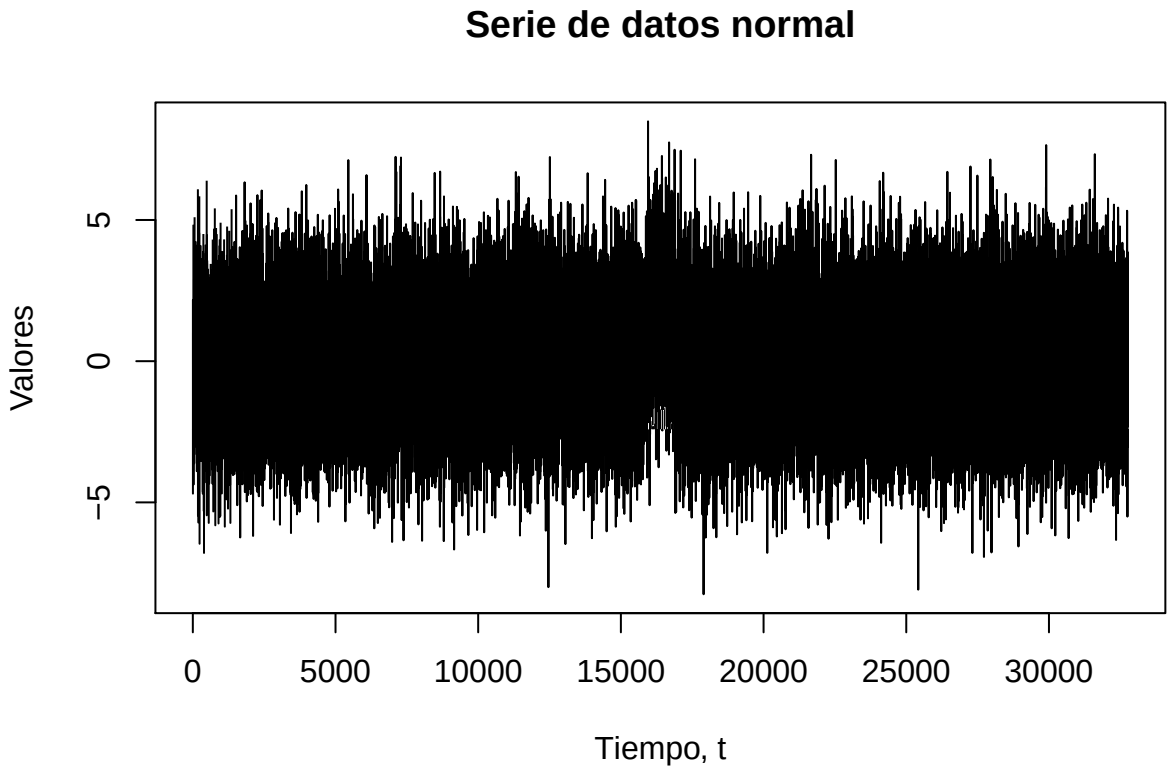
Repitiendo el paso 6, el salto de longitud que tiene r_t provoca que la entropía varíe un poco en sus picos, haciendo que aparezcan más picos conforme aumenta el valor de *sigma* o que se modifique su altura.

Repitiendo el paso 7, con ambos valores de alpha para la entropía de Havrda se obtiene lo siguiente:

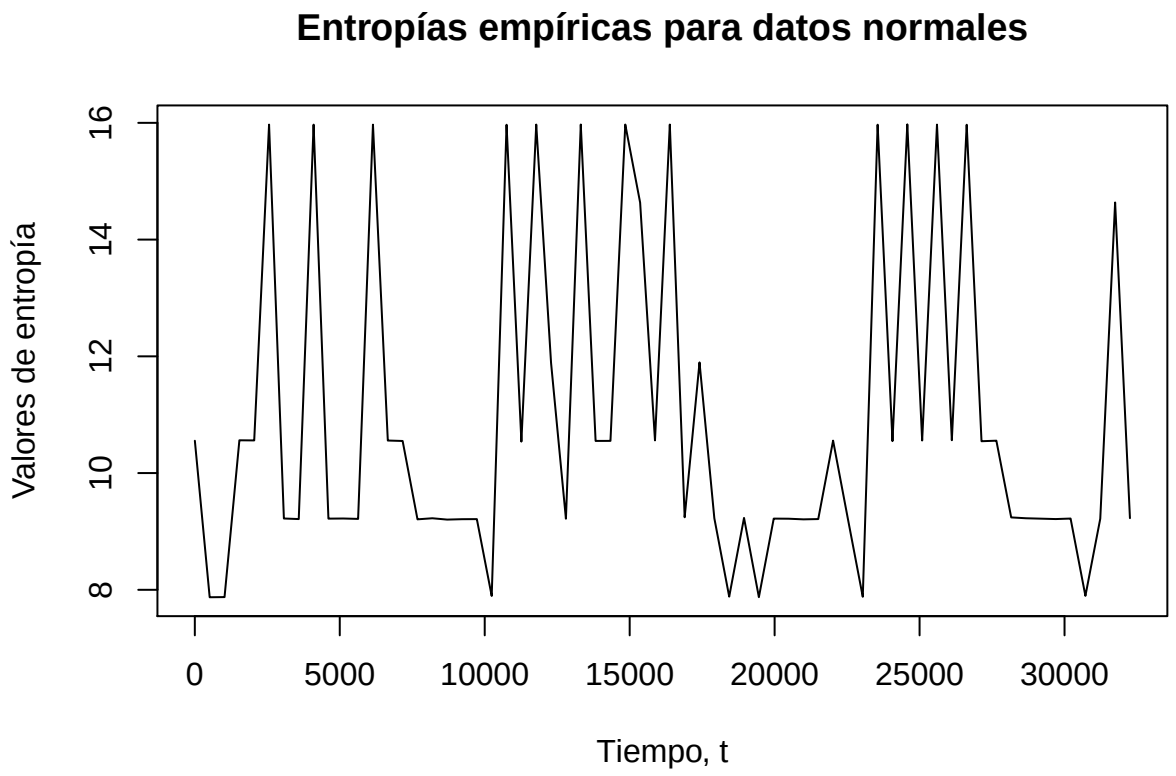
Usando $\alpha = 3$:

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(15872 <= t & t<= 16896, sigma, 0)
```

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```

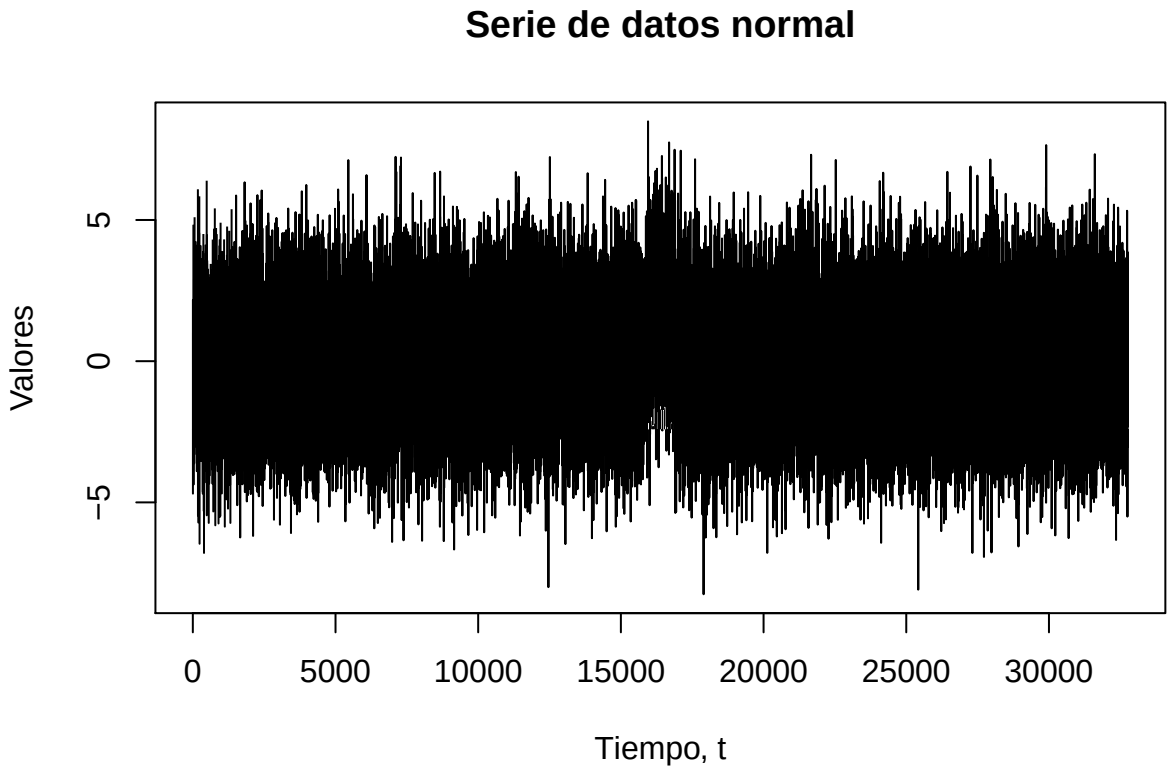



```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 3
  entropies[i] <- havrda(pmf, alpha = 3)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```

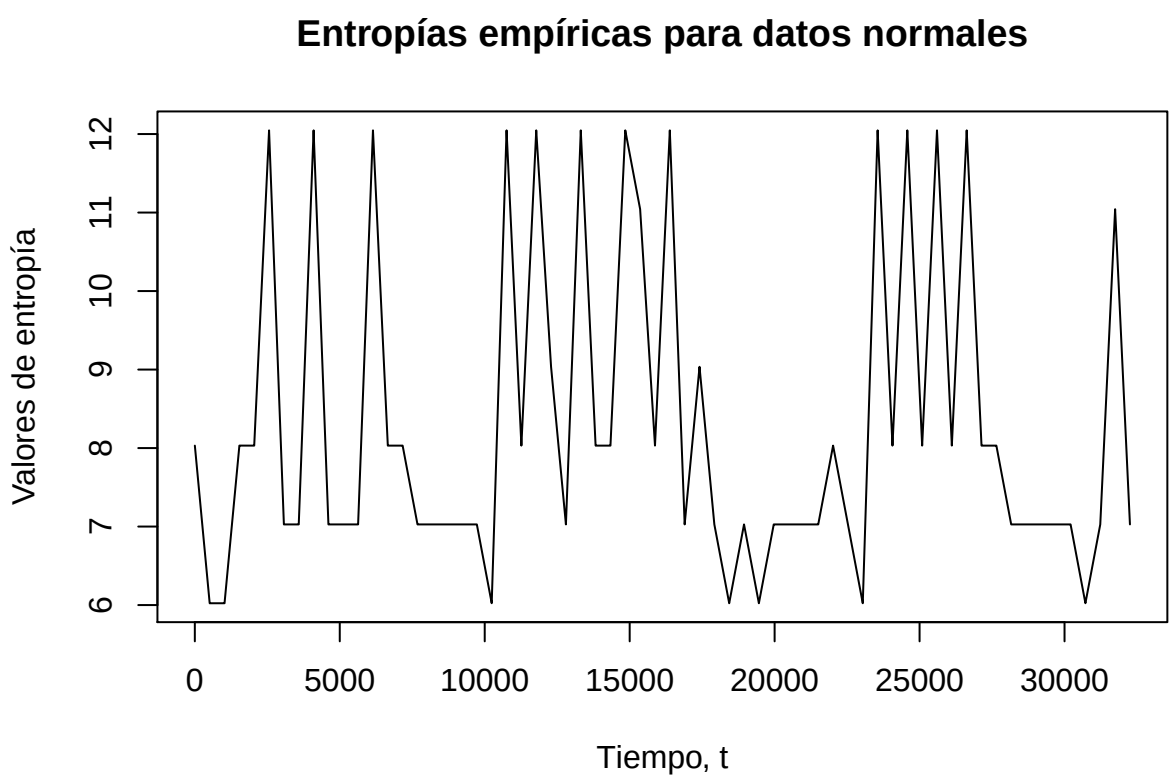


Usando $\alpha = 9$:

```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
wLength      <- 512
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```




```
noVentanas    <- length(datos)/wLength
entropies     <- numeric(noVentanas)
index         <- numeric(noVentanas)
for(i in 1:noVentanas)
{

  dataW        <- datos[(wLength*(i-1)+1):(wLength*i)]
  histo        <- hist(dataW, breaks=8,plot=FALSE)
  pmf          <- histo$counts/sum(histo$counts)
  # Entropia de Havrda con alpha = 9
  entropies[i] <- havrda(pmf, alpha = 9)
  index[i]     <- wLength*(i-1)+1
}
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales", xlab="Tiempo, t", yla
```



El efecto que tiene el parámetro α en la entropía de Havrda es aumentar los valores de la entropía como se observa en las gráficas anteriores, ya que, a mayor valor de α, mayores serán los valores en la entropía. Sin embargo, el patrón de la entropía es el mismo en cada caso, solamente variando la altura de acuerdo a su valor de α, como ya se explicó.

Entropía utilizando ventanas deslizantes

El cálculo de la entropía por ventanas independientes dada arriba resulta útil en casos en dónde la función no tiene dependencia en los valores futuros (descorrelacionadas). Para el caso de funciones correlacionadas, el cálculo de la entropía por ventanas deslizantes traslapadas resulta útil para descubrir ciertas fenomenologías en los datos. El cálculo por ventanas deslizantes de tamaño W se realiza sobre una secuencia de datos $X_1, X_2, \ldots, X_N$. La ventana ($W \leq N$) se va deslizando sobre los datos con factor Δ y de esta forma subconjuntos de los datos X_i toman la siguiente forma:

$$X(m; W, \Delta) = x_j \times \Pi(\frac{t - m\Delta}{W} - \frac{1}{2}),$$

donde $m\Delta \leq j \leq m\Delta + W$ y $m = 0, 1, 2, \ldots$. Finalmente se puede graficar $nW + \Delta, n = 1, 2, 3, \ldots$ contra las entropías y verificar algún patrón en los datos.

Ejercicios

- Implementar en `R` la metodología del cálculo de la entropía de Havrda normalizada por ventanas deslizantes. La función debe tener la forma `havrda_deslizante(datos, w.length=512, s.factor=10, a.parameter=0.8, ent.type=c("hist", "ash", "kern"))`, donde `w.length` es la longitud de la ventana, `s.factor` es el factor de deslizamiento y `a.parameter` es el parámetro α de la entropía de Havrda. Además la función puede calcular la entropia usando el histograma, por el método ash o por alguna metodología kernel (con el parámetro `ent.type`).
- Aplicar la entropía calculada con los datos generados anteriormente, es decir:
  - Los pasos $3, 4, 5$ y 7 en donde los datos se dan como $X_t + r_t$.
  - ¿Tiene algún efecto el parámetro a en la forma de la entropía? ¿Tiene alguna ventaja calcular la entropía de Havrda por otro método diferente al histograma?
  - Además del histograma y los estimadores tipo kernel, existen otros métodos para estimar la distribución de una serie de datos. Investigue: ¿en qué consiste la entropía de permutación?

$R=$

Se usa el paquete `zeallot` para regresar más de un valor

```
pacman::p_load(zeallot)
```

Primero se define la función de Havrda normalizada

```
# Definicion de la funcion de Havrda normalizada
havrda_normalizada <- function(pmf, N, a.parameter) sum(pmf^a.parameter - 1) / (N^(1 - a.parameter) - 1
```

Después, esta función se usa para implementar la metodología del cálculo de la entropía por ventanas deslizantes.

```
# Definicion del calculo de la entropia de Havrda por ventanas deslizantes
havrda_deslizante <- function(datos, w.length = 512, s.factor = 10, a.parameter = 0.8, ent.type = c("his
  # Se almacena el numero de observaciones
  N <- length(datos)
  # Se inicializa la secuencia para la creacion de ventanas deslizantes
  m <- 0
  # Se inicializa la secuencia n
  n <- 1
  # Se inicializan los limites del rango de datos
  inf <- 0
  sup <- 0
  # Se inicializa el vector de entropias
  entropies <- c()
  # Se inicializa el vector de indices
  index <- c()
  # Se inicializan los limites del rango
  inf <- m * s.factor + 1
  sup <- m * s.factor + w.length
  # Se crean las ventanas deslizantes mientras existan datos
  while (sup <= N) {
    # Se obtienen los datos dentro de la ventana deslizantes
    dataW <- datos[inf:sup]
    # Se calcula la pmf usando el metodo adecuado
    pmf <- switch (ent.type,
```

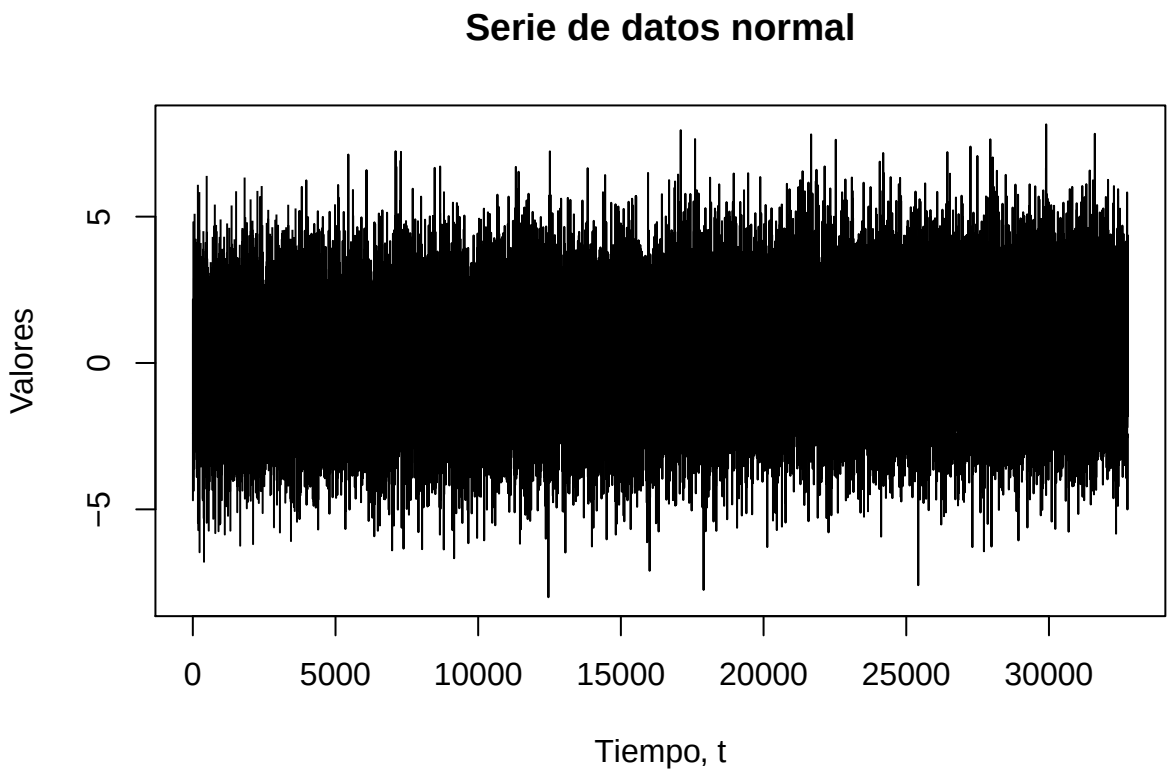
```
      "hist" = {
        hist <- hist(dataW, breaks = 1000, plot = FALSE)
        hist$counts / sum(hist$counts)
      },
      "ash" = {
        invisible(capture.output(hist <- ash1(bin1(dataW))))
        hist$y / sum(hist$y)
      },
      "kern" = {
        hist <- bkde(dataW)
        hist$y / sum(hist$y)
      }
    )
    # Se calcula la entropía de Havrda normalizada
    entropies <- append(entropies, havrda_normalizada(pmf, N, a.parameter))
    # Se agrega el indice
    index <- append(index, n*w.length + s.factor)
    # Se actualiza el valor de m y n
    m <- m + 1
    n <- n + 1
    # Se actualizan los limites del rango
    inf <- m * s.factor + 1
    sup <- m * s.factor + w.length
  }
  # Se regresa el vector X y el de entropias
  return(list(index, entropies))
}
```

Con esta formula se repiten los pasos 3, 4, 5 y 7.

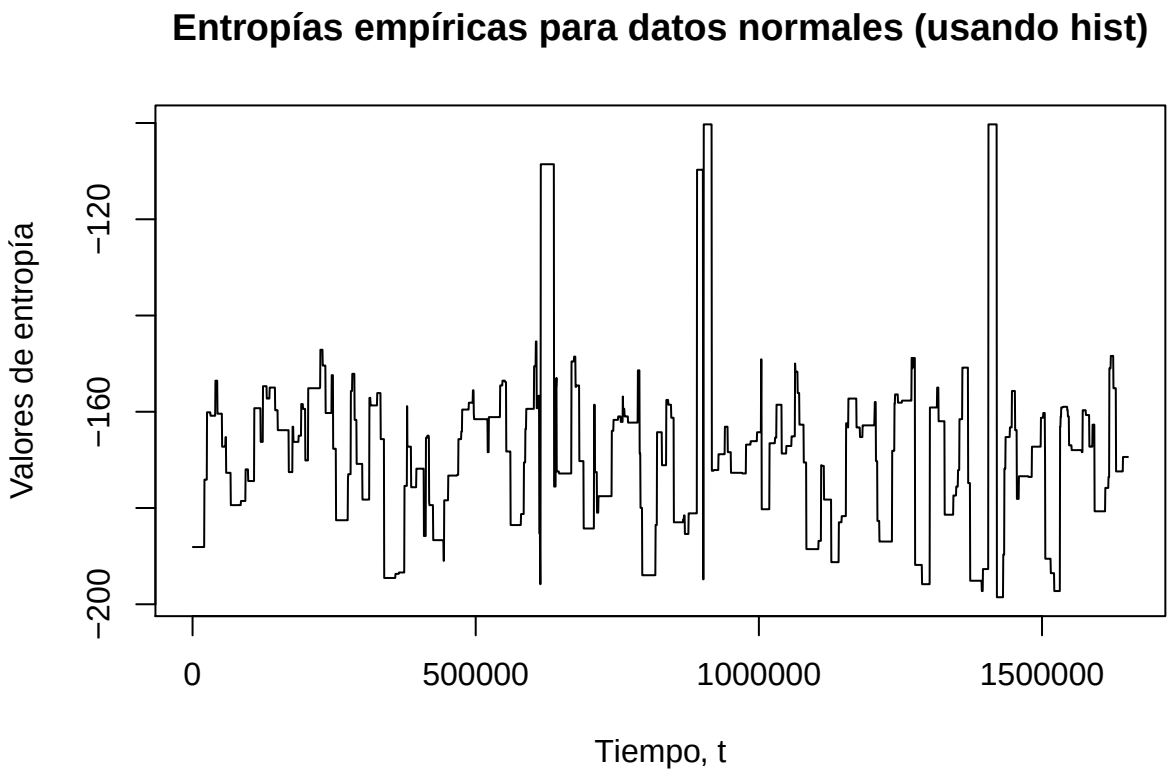
- Repitiendo el paso 3:

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma / 4, 0)
```

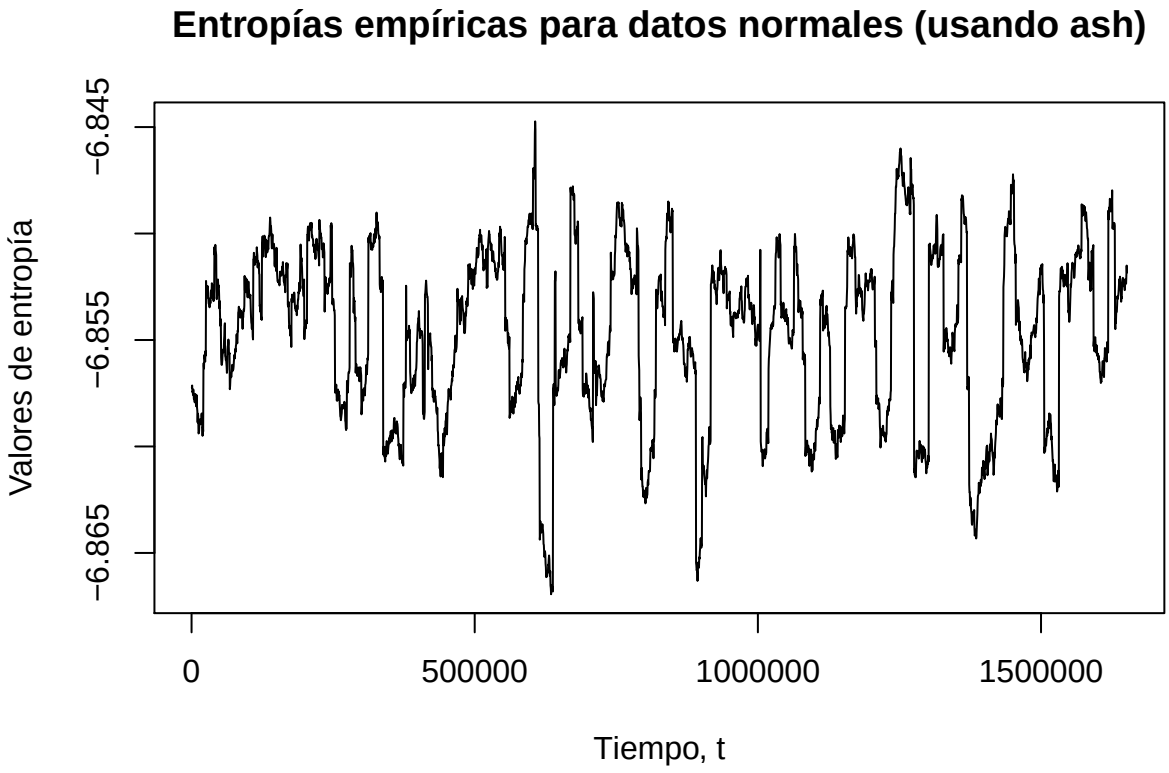
```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```



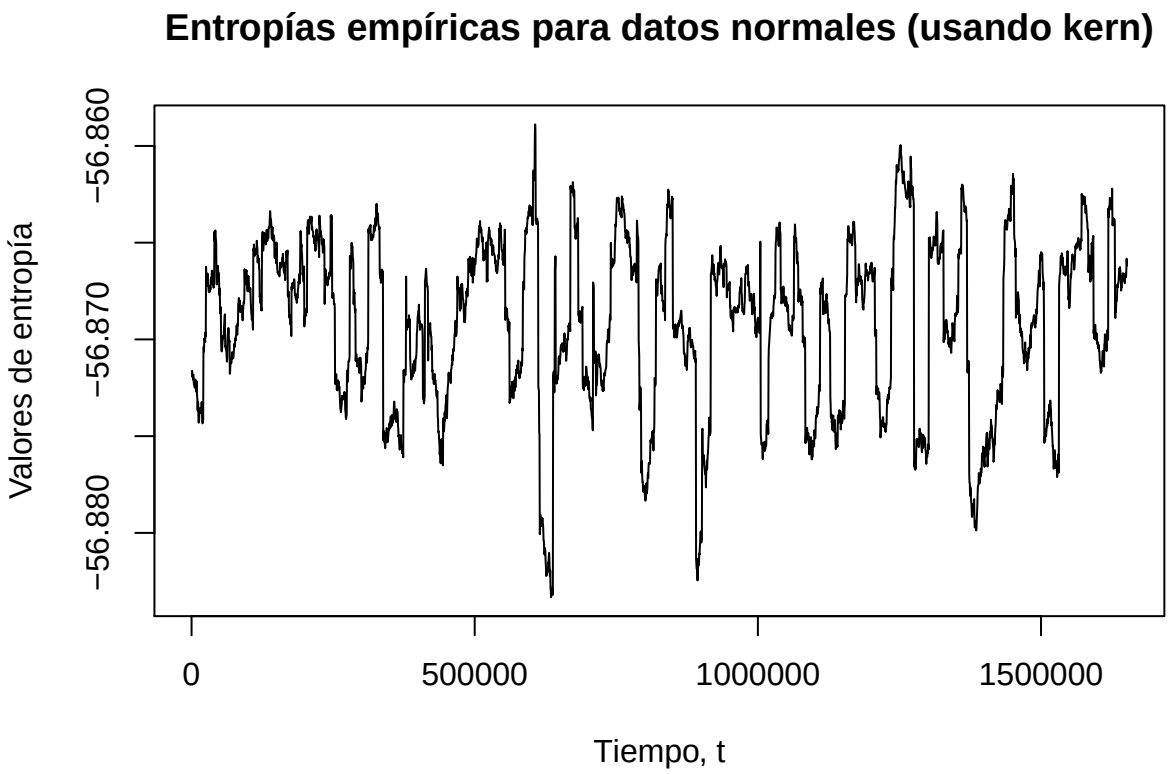

```r
# Usando hist
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "hist")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando hist)", xlab="
```




```
# Usando ash
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "ash")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando ash)", xlab="T
```



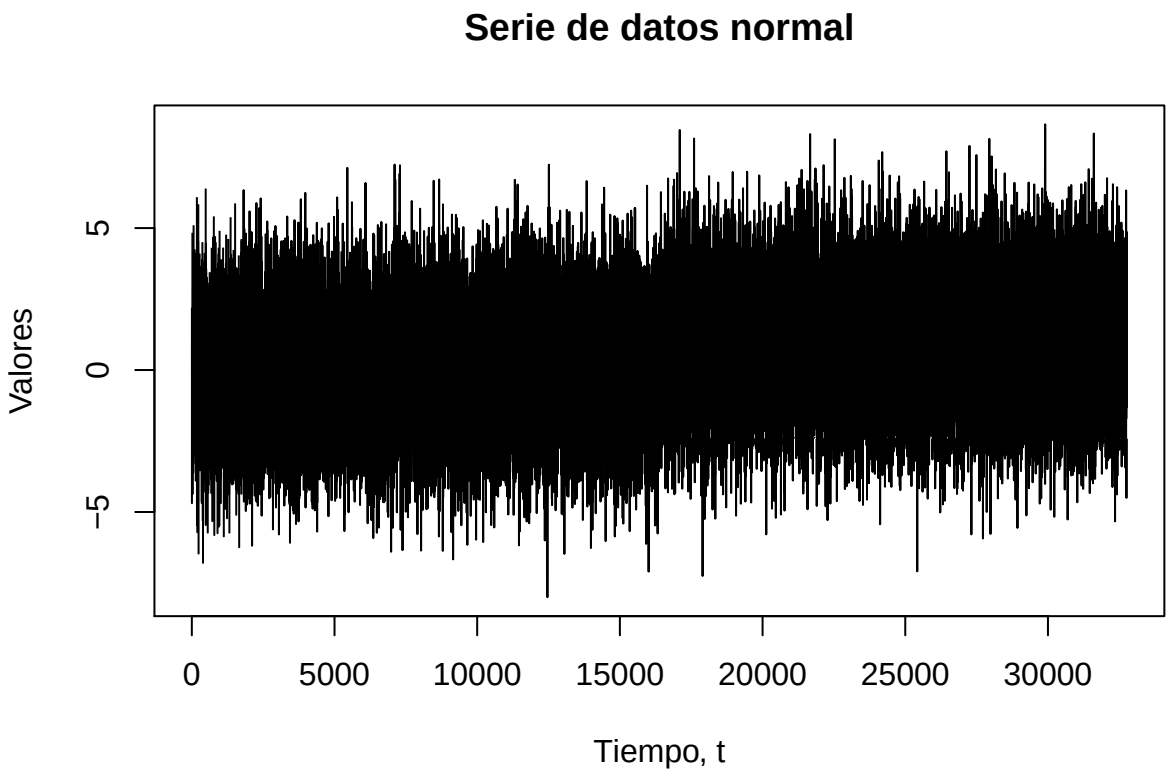

```
# Usando kern
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



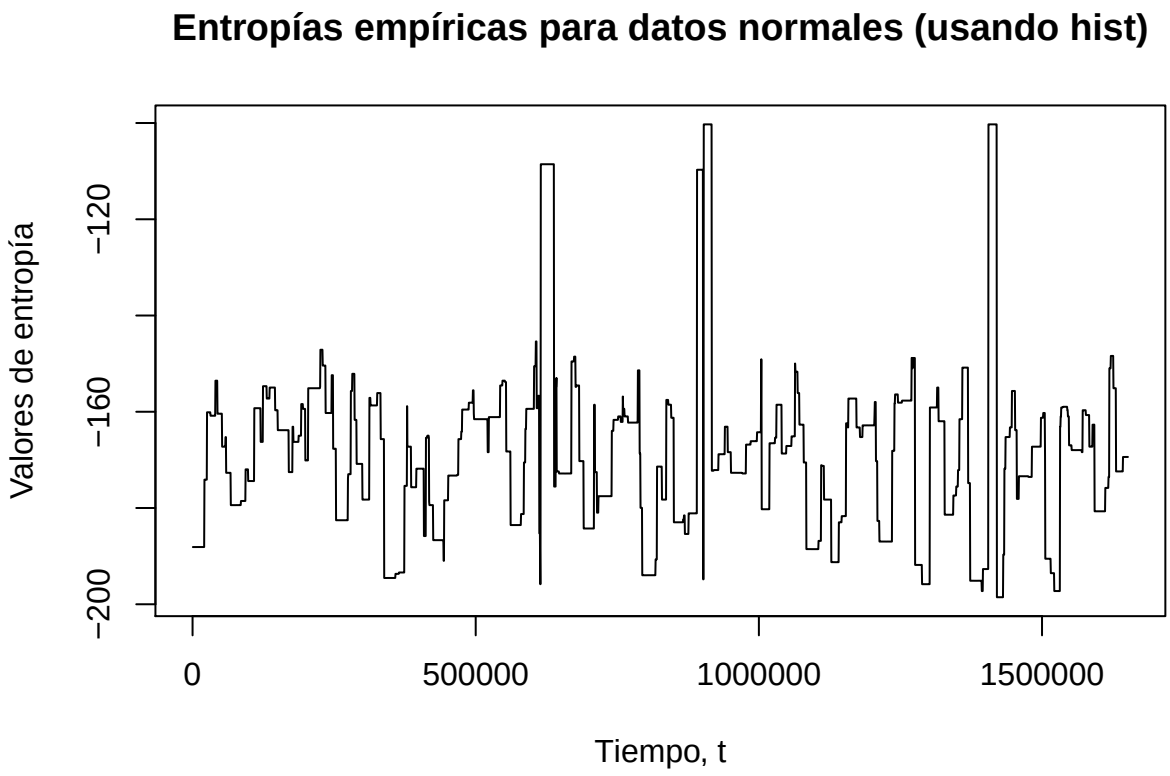
- Repitiendo el paso 4:

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma / 2, 0)
```

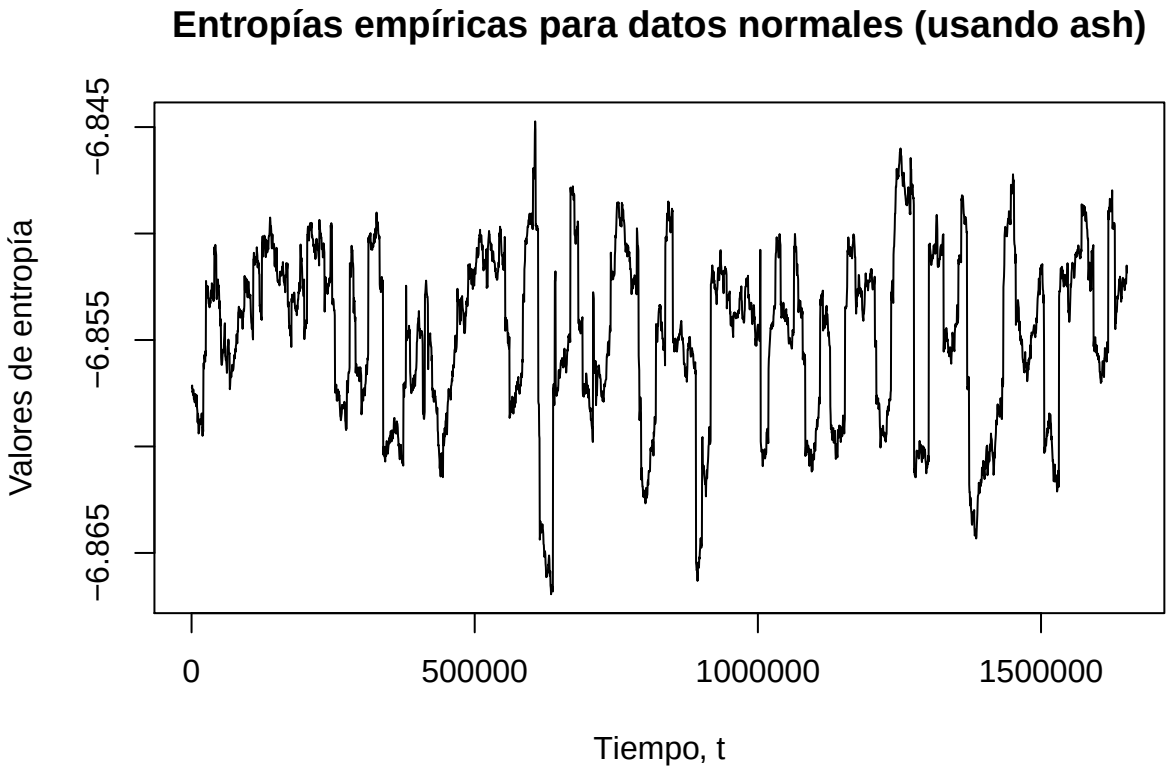
```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```



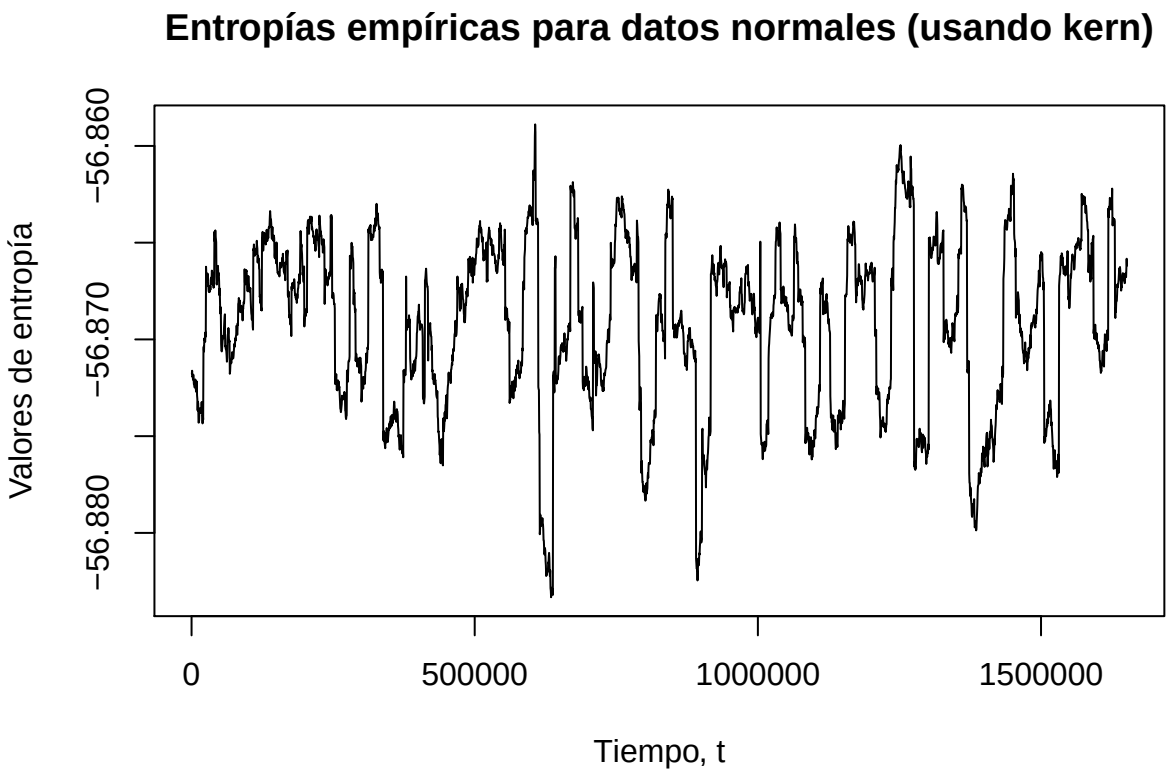

```r
# Usando hist
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "hist")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando hist)", xlab="
```




```r
# Usando ash
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "ash")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando ash)", xlab="T
```



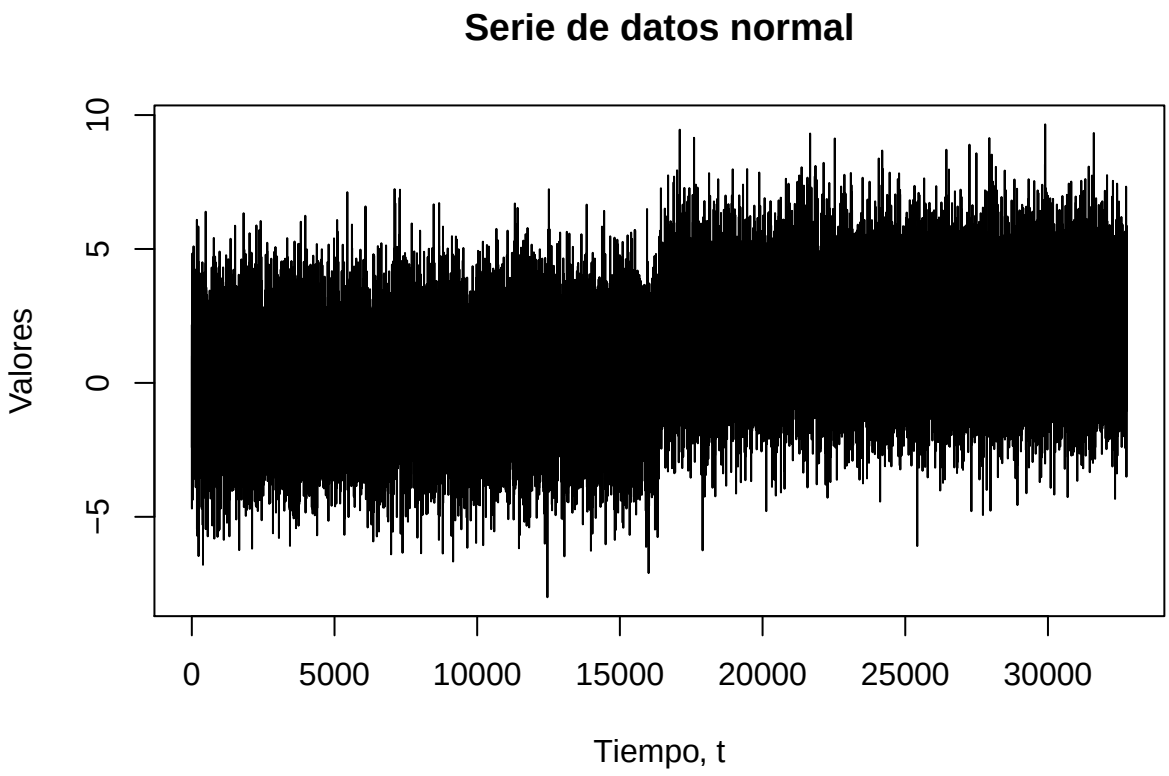

```
# Usando kern
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



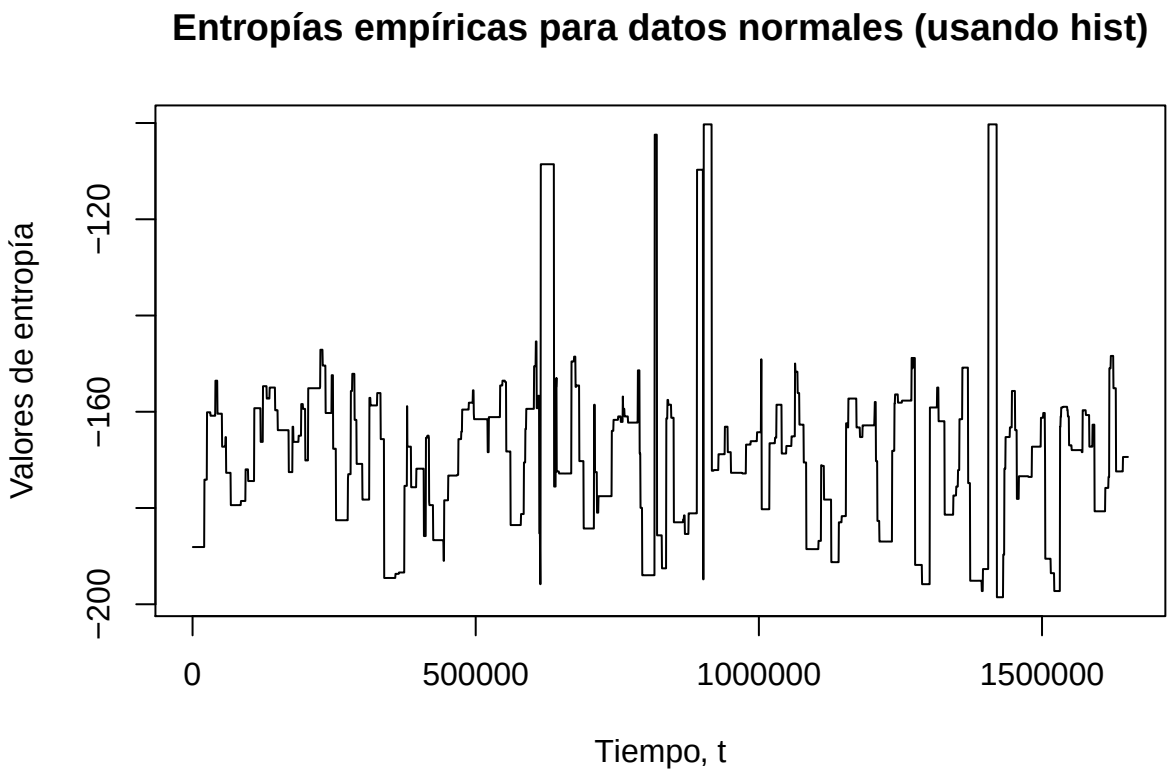
- Repitiendo el paso 5:

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(t >= 16384, sigma, 0)
```

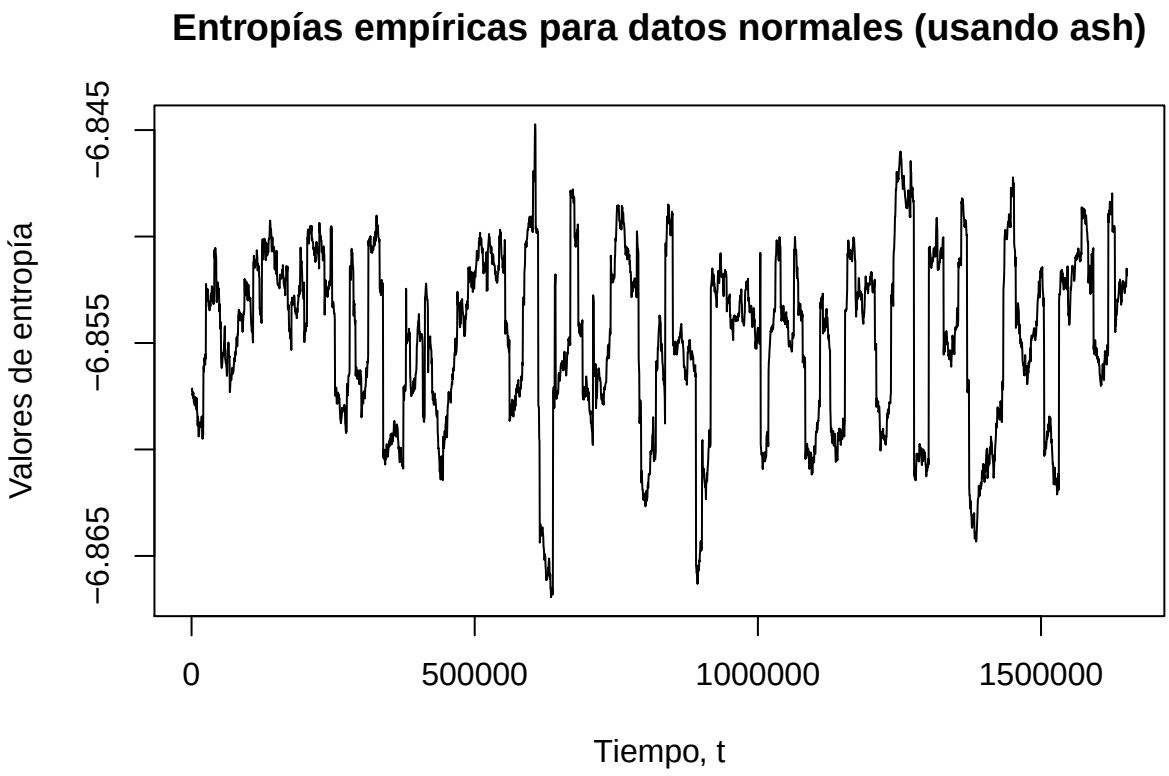
```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```



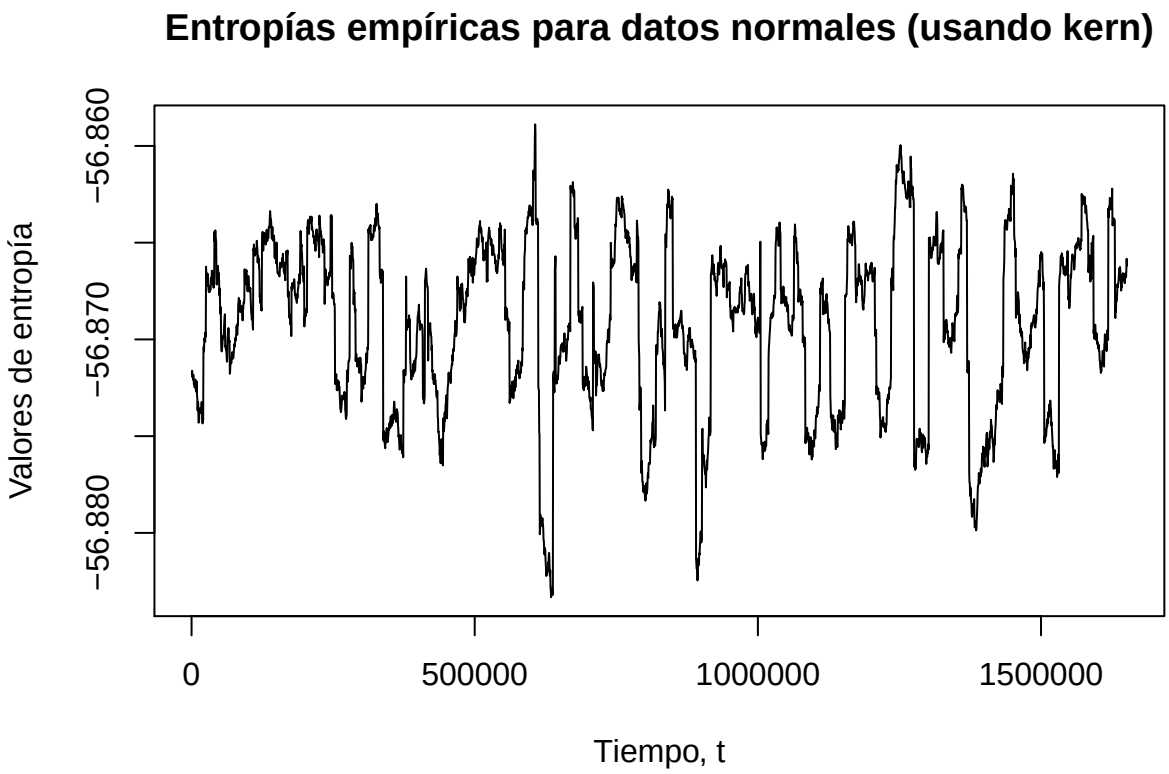

```
# Usando hist
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "hist")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando hist)", xlab="
```




```r
# Usando ash
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "ash")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando ash)", xlab="T
```



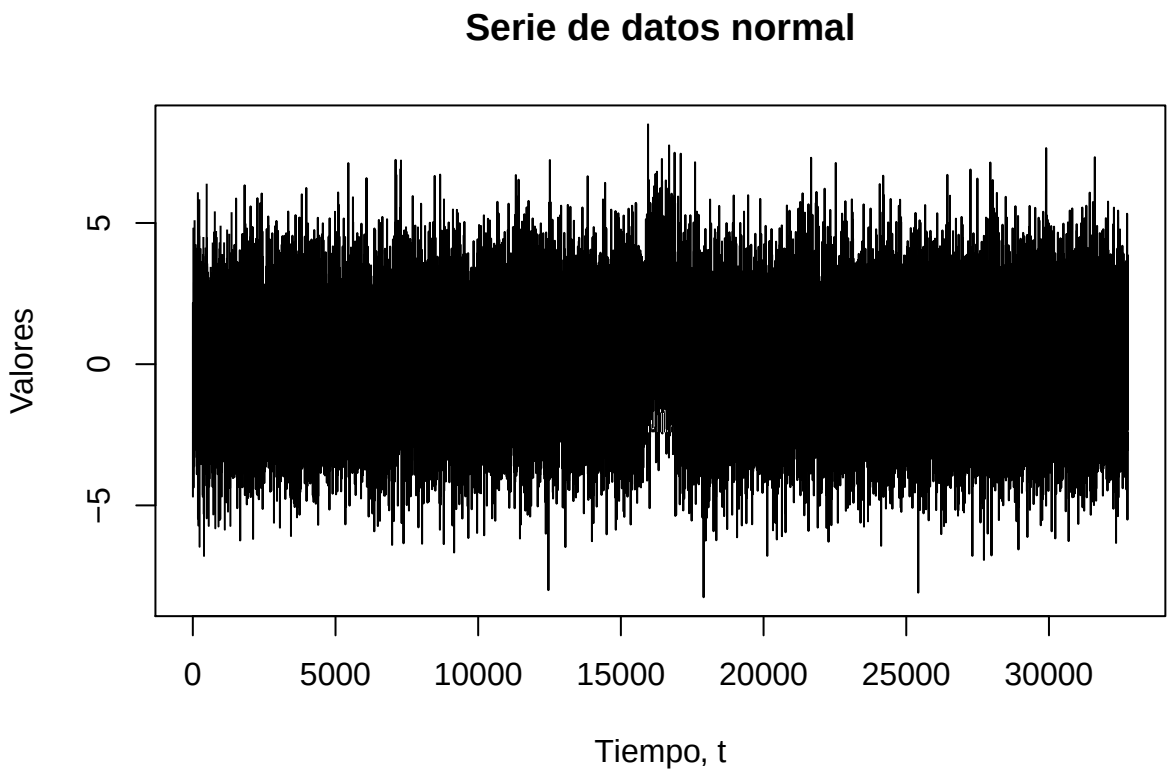

```
# Usando kern
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



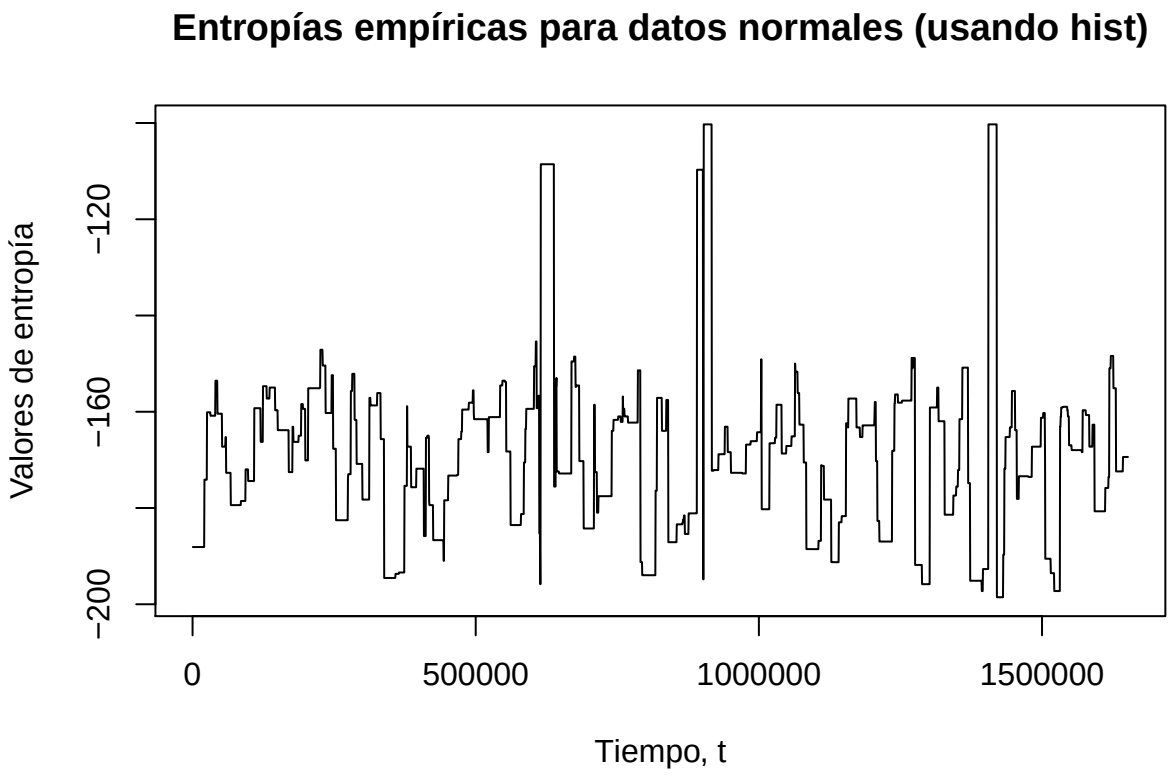
- Repitiendo el paso 7:

```
# Funcion r_t
r_t <- function(t, sigma) ifelse(15872 <= t & t<= 16896, sigma, 0)
```

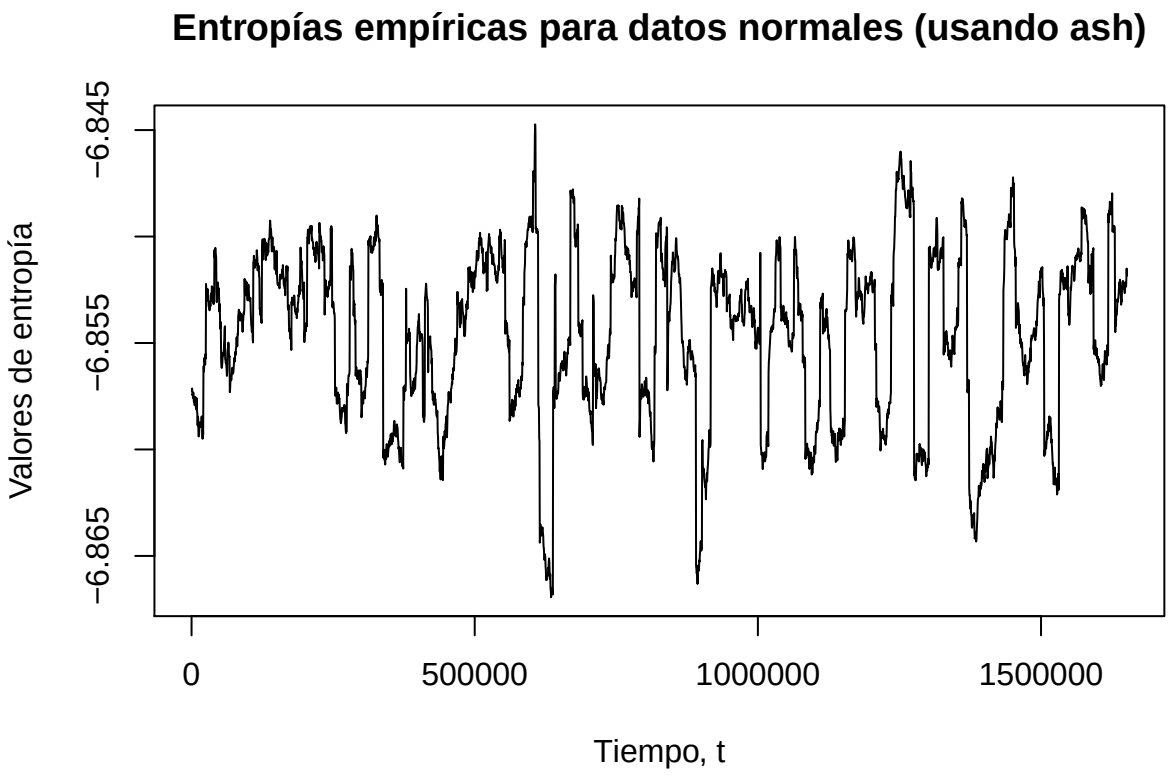
```
set.seed(1234)
datos        <- rnorm(32768, mean = 0, sd = 2) + r_t(1:32768, 2)
plot(datos, type = "l", main="Serie de datos normal", ylab="Valores", xlab="Tiempo, t")
```



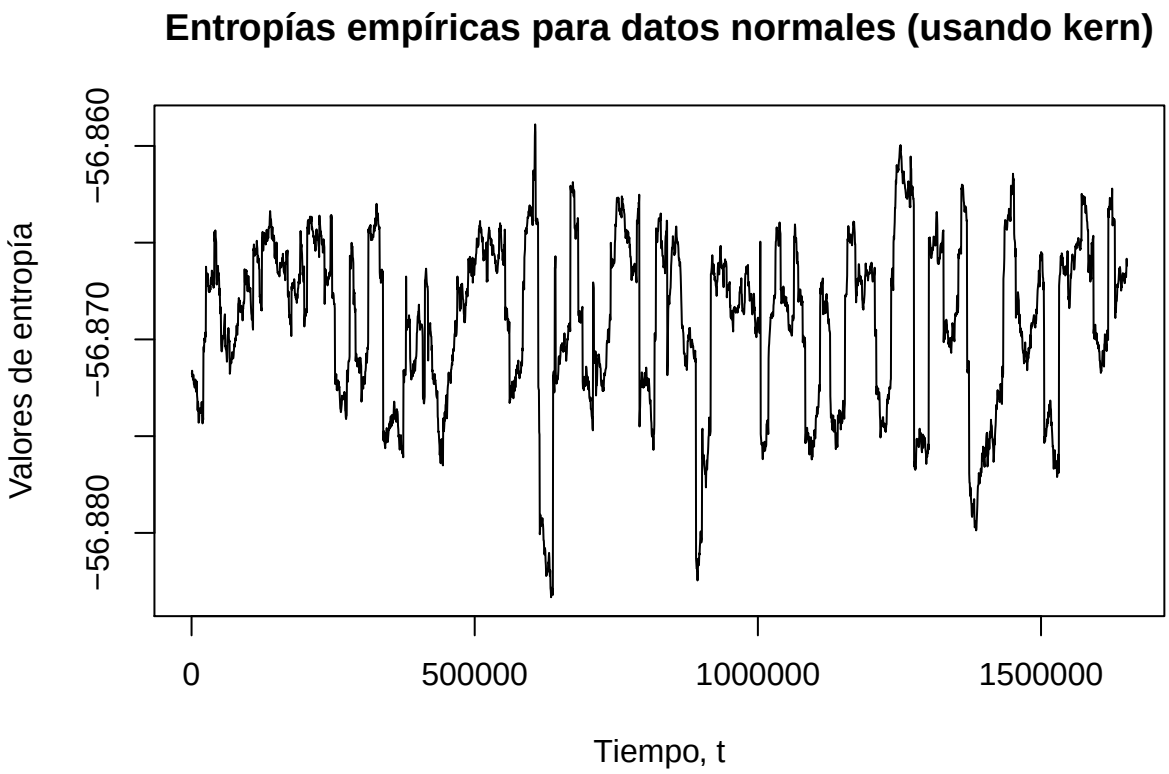

```r
# Usando hist
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "hist")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando hist)", xlab="
```




```
# Usando ash
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "ash")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando ash)", xlab="T
```



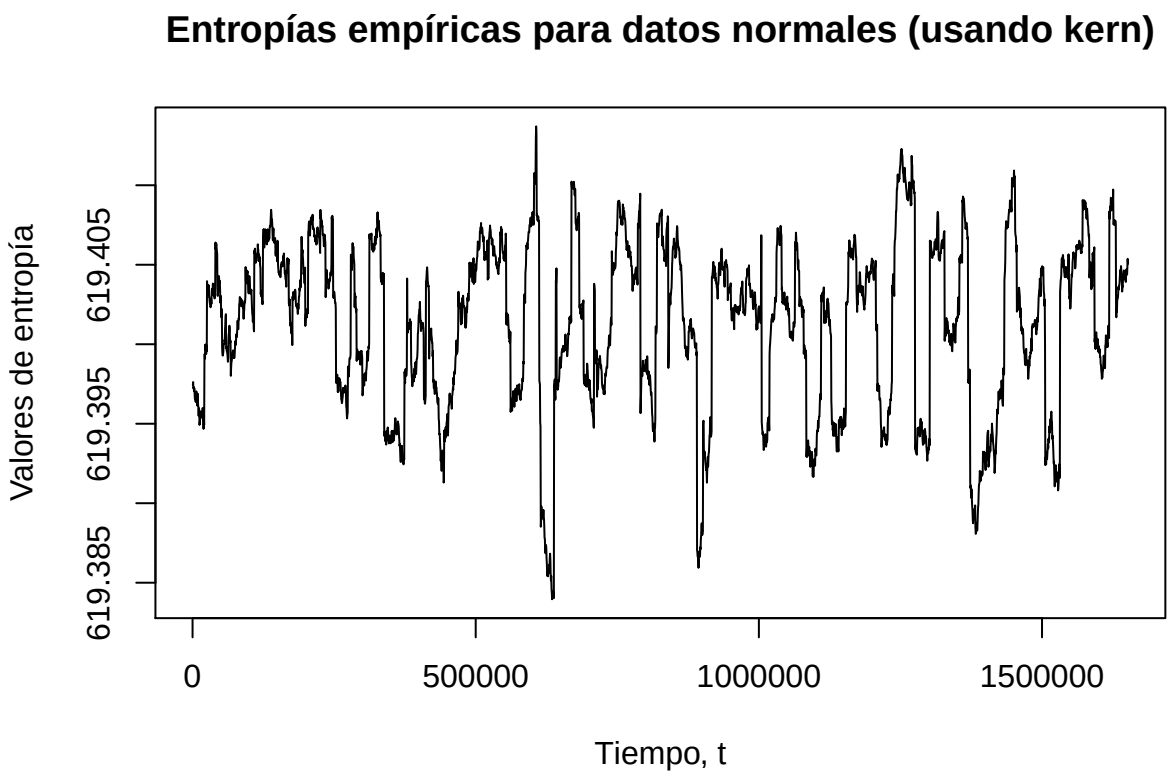

```
# Usando kern
c(index, entropies) %<-% havrda_deslizante(datos, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



- Para entender el efecto de α sobre la forma de la entropía, se repetirá el caso 7 con distintos valores de α y usando `kern`.

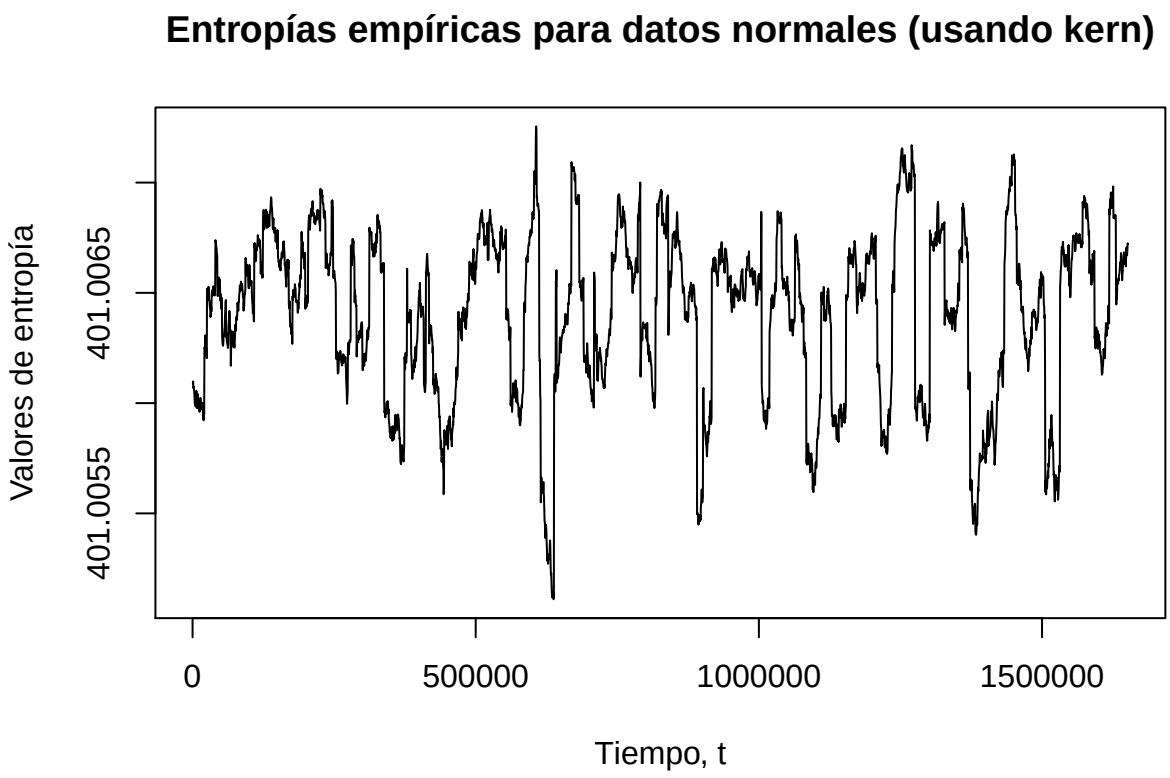
Con $\alpha = 1.1$

```
c(index, entropies) %<-% havrda_deslizante(datos, a.parameter = 1.1, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



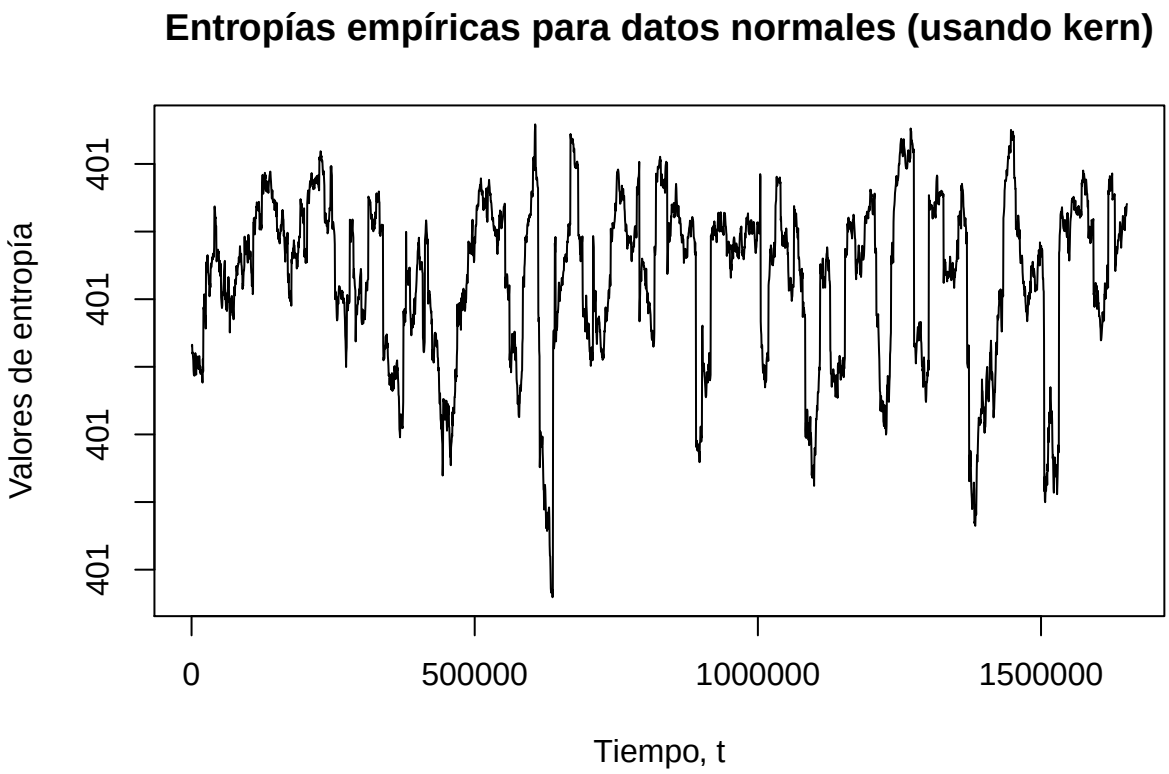
Con $\alpha = 2$:

```
c(index, entropies) %<-% havrda_deslizante(datos, a.parameter = 2, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



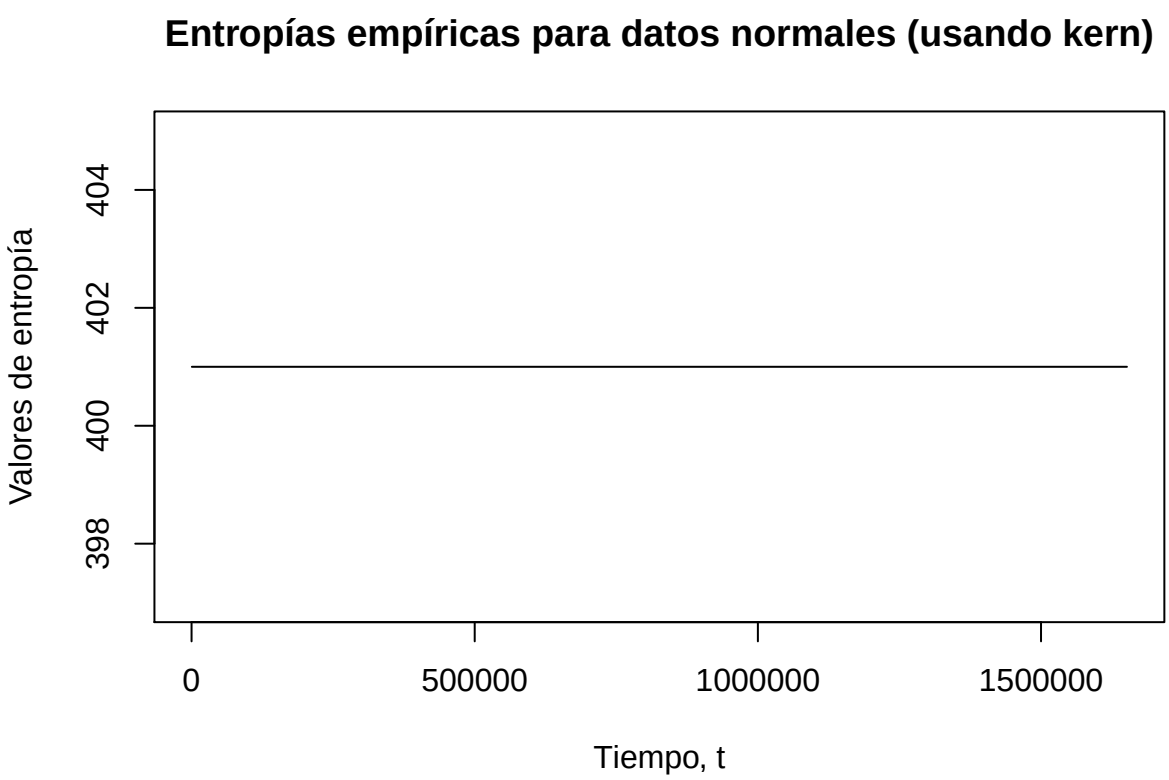
Con $\alpha = 4$:

```
c(index, entropies) %<-% havrda_deslizante(datos, a.parameter = 4, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



Con $\alpha = 7$

```
c(index, entropies) %<-% havrda_deslizante(datos, a.parameter = 7, ent.type = "kern")
plot(index, entropies, type = "l", main="Entropías empíricas para datos normales (usando kern)", xlab="
```



Aunque la forma de la entropía sigue siendo la misma, lo que hace α es trasladar la función sobre el eje de la Y, así como disminuir el tamaño de la misma, haciéndola cada vez más pequeña hasta llegar a un punto donde la entropía se vuelve una línea horizontal.

Finalmente, calcular la entropía de Havrda con un método diferente al histograma tiene la ventaja de que la gráfica de los valores de la entropía sale más definida, ya que los paquetes *ash* y *kernsmoothh* suavizan más la gráfica a diferencia del histograma, que hasta con 1,000 bins se sigue viendo muy cuadrada.

- La entropía de permutación fue introducida por Christoph Bandt y Bernd Pompe, y define una medida de entropía tomando en cuenta la causalidad del tiempo de la serie de tiempo, comparando los valores vecinos en la serie de tiempo.Por lo que esta entropía es equivalente a la Entropía de Shannon de una secuencia de patrones ordinales (símbolos discretos que codifican qué tanto las entradas consecutivas de la serie de tiempo se relacionan entre sí en términos de la posición y valor). Su función se define como:

$$PeEn = -\sum_{i}^{N!} p_i' log(p_i')$$

Donde p_i' representa las frecuencias relativas de los posibles patrones de secuencias de símbolos, denominadas permutaciones. Las permutación se relaciona a la secuencia de m (dimensión embebida) valores de la serie original.

Referencias:

Baldini, G. (2020). **On the Application of Entropy Measures with Sliding Window for Intrusion Detection in Automotive In-Vehicle Networks**. Entropy 2020, 22, 1044.