Entrega de ejercicios Tema 3

Blanca Cano Camarero

10 de noviembre de 2022

Indice de contenidos

Ejercicio 5

```
library(ggplot2)
library(purrr)
library(nortest) # Para ejemplo del Lillie test

set.seed(100)
```

La función (ksnoest) calcula el estadístico de Kolmogorov-Smirnov cuando estamos interesados en contrastar si nuestros datos siguen una distribución normal estándar:

```
ksnoest <- function(datos){
    y <- ks.test(datos,pnorm)$statistic
    return(y)
}
```

Supongamos que queremos contrastar la hipótesis nula de que los datos son normales (con valores arbitrarios de la media y la desviación típica). Una posibilidad es estimar los parámetros de la normal y comparar la función de distribución empírica F_n con la función de distribución de una variable $N(\hat{\mu}, \hat{\sigma}^2)$. La siguiente función calcula el correspondiente estadístico de Kolmogorov-Smirnov:

```
ksest <- function(datos){
    mu <- mean(datos)
    stdev <- sd(datos)
    y <- ks.test(datos, pnorm, mean=mu, sd=stdev)$statistic
    return(y)
}
```

Apartado 1. Genera 1000 muestras de tamaño 20 y calcula ambos estadísticos (ksnoest y ksest) para cada una de ellas.

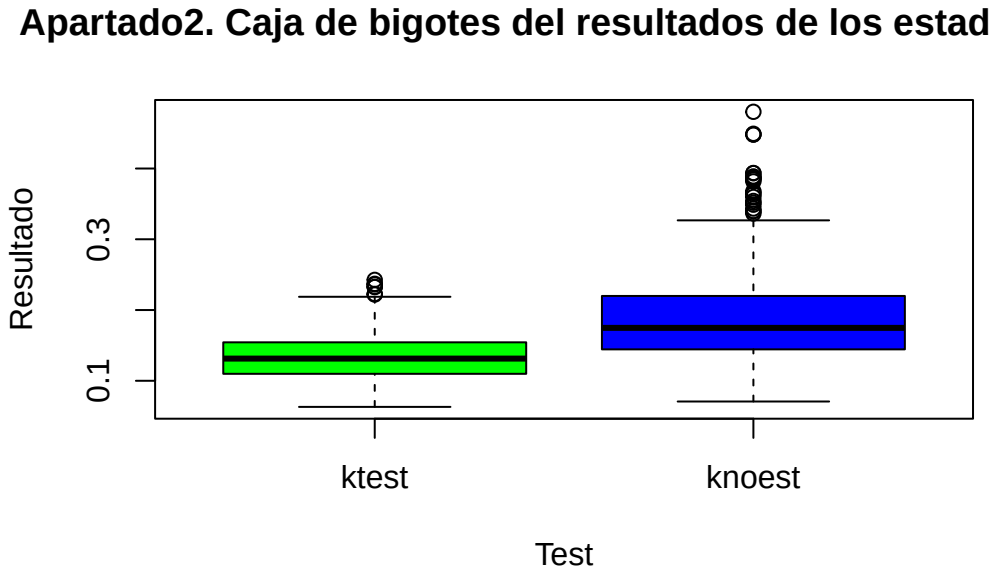
```
mean <- 0
sd <- 1
number_of_samples <- 1000
sample_size <- 20

samples <-  matrix(
   rnorm( number_of_samples * sample_size),#, mean=mean, sd= sd*),
   nrow = number_of_samples
)
```

```r
# Cálculo de ksnoest
knoest_results <- apply(X = samples,
                        MARGIN = 1, # rows 1, columns = 1
                       FUN = ksnoest
                    )
# Cálculo de ksest
kest_results <- apply(samples,1, ksest )
```

Apartado 2. Mediante diagramas de cajas, compara las distribuciones de ambos estadísticos. ¿En cuál de los dos casos se obtienen en media valores menores? ¿Podrías dar una razón intuitiva?

Solución

```r
boxplot(kest_results,knoest_results,
        main = "Apartado2. Caja de bigotes del resultados de los estadísticos",
        xlab = "Test",
        ylab = "Resultado",
        names = c("ktest", "knoest"),
        col = c("green","blue ")
)
```



En *boxplot* la media viene indicada como la línea negra central, a la vista del gráfico en ktest (el verde) se obtiene una media menor.

La motivación de esto es la siguiente: de acorde a la documentación de R (ejecute `help(ks.test)`) el estadístico de test es

$$D^+ = \max_u (F_x(u) - F_y(u))$$

Que en nuestro caso se corresponde con la diferencia entre la función de distribución empírica y la de una normal.

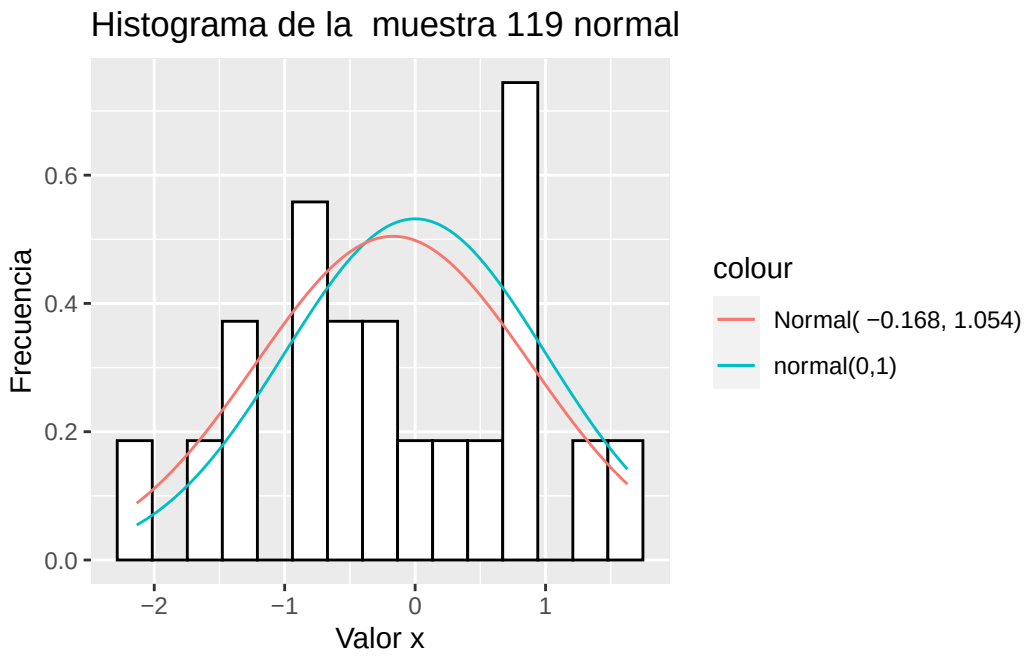
Por defecto, en *knoest* se está comparando con una normal de media 0 y desviación típica 1. Mientras que en *ktest*, la normal que se utiliza tiene como media la media y como desviación típica la calculada de la muestra. Al ser el tamaño de muestra relativamente pequeño, aunque los datos idealmente pertenecen a a una distribución de media cero y varianza 1, el error de la distribución empírica es menor con el de la distribución estimada.

Para ejemplificar lo dicho vamos a mostrar un histograma de los resultados, junto con dos funciones de probaibilidad, el de una normal de media cero y varianza 1 (la verde) y el de una normal de parámetros estimados con la muestra (curva naranja).

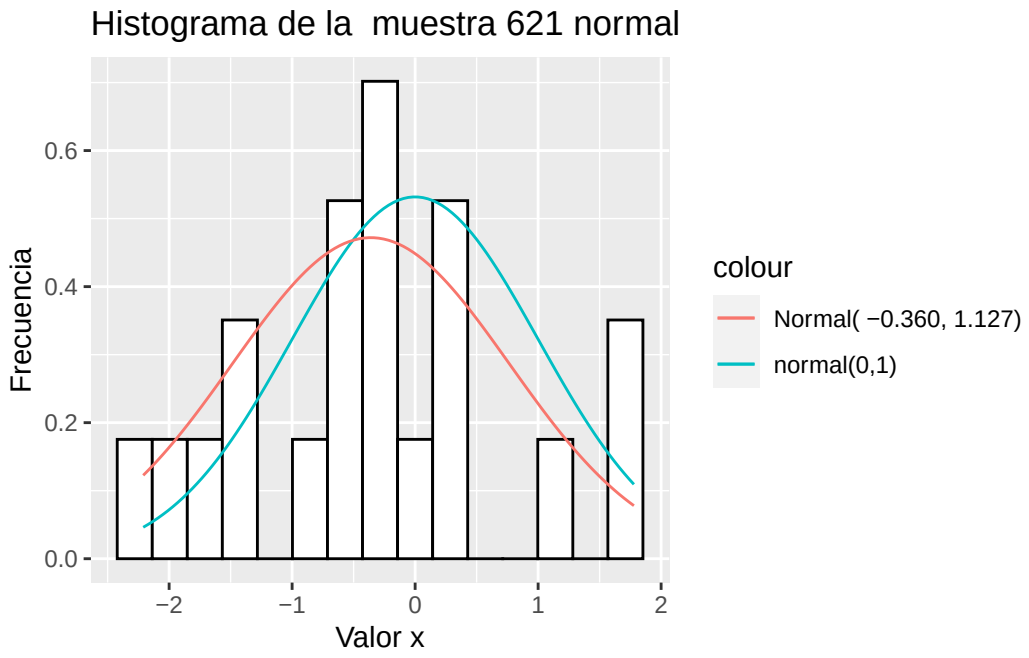
```
bins = 15

plot_hist_and_density <- function (index){

df <- data.frame(PF = samples[index,])
  scale <- sample_size/bins
  # get adapted norm
  scaled_norm <- function (x) scale*dnorm(x)
  scaled_norm_adapted <- function (x) scale*dnorm(x,mean = mean(df$PF), sd = sd(df$PF) )

  title = sprintf("Normal( %.3f, %.3f)", mean(df$PF), sd(df$PF))
  big_title =  sprintf("Histograma de la  muestra %i normal ", index)

  ggplot(df, aes(x = PF)) +
    ggtitle(big_title ) +
    xlab("Valor x") + ylab("Frecuencia") +
      geom_histogram(aes(y =..density..),
                     colour = "black",
                     fill = "white",
                     bins = bins) +
  geom_function(aes(colour = "normal(0,1)"), fun = scaled_norm ) +
  geom_function(aes(colour = title), fun = scaled_norm_adapted)

}
```

```
number_of_plots = 2
map(sample(1:number_of_samples, number_of_plots), plot_hist_and_density)
```

```
[[1]]
```




```
[[2]]
```



Apartado 3. Imagina que estimamos los parámetros y usamos las tablas de la distribución del estadístico de Kolmogorov-Smirnov para hacer el contraste a nivel α. El verdadero nivel de significación, ¿es mayor o menor que α?

Solución

El nivel de significación α es la probabilidad de rechazar la hipótesis nula siendo verdadera, será por lo general menor, porque nosotros estamos suponiendo que es igual a una normal de media y varianza estimadas; que son diferentes a los parámetros ideales *desconocidos*.

Como ejemplo de cómo efecta la estimación de parámetros mostraré cómo varía el p-valor; la probabilidad de aceptar la hipótesis nula cuando esta es verdadera. Por la misma explicación anterior, ahora para los parámetros estimados el p-valor debería de ser mayor que para los fijos *ideales*. Este valor es uno de los atributos de `ks.test`.

```
ksnoest_alpha <- function(datos){
    p_value <- ks.test(datos,pnorm)$p
    return(p_value)
}

ksest_alpha <- function(datos){
    mu <- mean(datos)
    stdev <- sd(datos)
    p_value <- ks.test(datos, pnorm, mean=mu, sd=stdev)$p
```

```
    return(p_value)
}

significacion_sin_estimar <- mean(apply(samples, 2, ksnoest_alpha))
significacion_estimada <- mean(apply(samples, 2,ksest_alpha))
sprintf('La media del (para ksnoest) sin estimar los parámetros es %.4f.', significacion_s
```

```
[1] "La media del (para ksnoest) sin estimar los parámetros es 0.5682."
```

```
sprintf('La media de del p para ksest es: %.4f.', significacion_estimada)
```

```
[1] "La media de del p para ksest es: 0.7906."
```

Podemos observar como para parámetro estimado es menor en nuestro caso.

Apartado 4. Para resolver el problema se ha estudiado la distribución en el caso de muestras normales con parámetros estimados. Es lo que se conoce como contraste de normalidad de Kolmogorov-Smirnov-Lilliefors (KSL). Según la tabla del estadístico KSL, el nivel crítico para $\alpha = 0.05$ y $n = 20$ es 0.190. Esto significa que el porcentaje de valores de ksest mayores que 0.19 en nuestra simulación debe ser aproximadamente del 5%. Compruébalo a partir de los resultados de los apartados anteriores.

Solución

```
nivel_critico <- 0.19
# Sacamos en una lista los mayores
numero_de_mayores <-  length(kest_results[kest_results > nivel_critico])
porcentaje <- 100 * numero_de_mayores / number_of_samples

cat('El porcentaje de mayores es un ', porcentaje, '%' )
```

```
El porcentaje de mayores es un  5.2 %
```

```
# Podemos comprobar el p valor además haciendo
lillie.test(samples)
```

```
    Lilliefors (Kolmogorov-Smirnov) normality test

data:  samples
D = 0.0041207, p-value = 0.5647
```

Apartado 5. Haz una pequeña simulación para aproximar el nivel de significación del contraste KSL cuando se utiliza un valor crítico 0.12 para muestras de tamaño 40.

Solución

Planteamiento de la simulación. Para aproximar el nivel de significación lo que haremos será generar `number_of_samples <- 1000` muestras de tamaño `sample_size <- 40`. Calcularemos el porcentaje esas muestras se quedan por encima del valor crítico `nivel_critico <- 0.12`

```
# Parámetros
nivel_critico <- 0.120
number_of_samples <- 1000
sample_size <- 40
times_repeating_experiment = 50

# Variables auxiliares
porcentaje <- NULL

for(i in 1:times_repeating_experiment){
  samples <-  matrix(
     rnorm( number_of_samples * sample_size),
     nrow = number_of_samples
  )
  kest_results <- apply(samples,1, ksest)

  numero_de_mayores <-  length(kest_results[kest_results > nivel_critico])
  porcentaje <-  rbind(porcentaje,numero_de_mayores / number_of_samples)
}

cat('El porcentaje de mayores es de media', mean(porcentaje)*100, '%')
```

```
El porcentaje de mayores es de media 15.084 %
```

```
cat('\nEs decir alpha es approximadamente ',mean(porcentaje), ' +-', sd(porcentaje))
```

```
Es decir alpha es approximadamente  0.15084  +- 0.01225305
```

Si observamos la tablas de KOLMOGOROV-SMIRNOV, para $n = 40$ y nivel crítico 0.1204 se tiene que $\alpha = 0.15$.

```r
library(purrr)
library(ggplot2)
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v tibble  3.1.8      v dplyr   1.0.10
v tidyr   1.2.0      v stringr 1.4.1
v readr   2.1.2      v forcats 0.5.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```r
set.seed(5)
```

Ejercicio 7

```r
muestra <- c(1, 2, 3.5, 4, 7, 7.3, 8.6, 12.4, 13.8, 18.1)
varianza <- var(muestra)
```

Apartado 1. Usa bootstrap para determinar el error típico de este estimador de σ^2.

Solución Generaremos nuevas muestras a partir de las que ya tenemos, calcularemos sus varianzas y finalemente el error típico de éstas.

```r
size <- length(muestra)
number_of_samples <- 1000

# Paso 1: Remuestro de los datos
remuestreo <- matrix(
  sample(muestra, size*number_of_samples, replace=TRUE),
  nrow = number_of_samples
)
# Paso 2: Cálculo de la varianza de cada remuestreo
varianzas_remuestreo <- apply(remuestreo, 1, var)

# Paso 3: Cálculo del error típico
et <- sd(varianzas_remuestreo)

cat('El error típico de remuestreo es ', et)
```

```
El error típico de remuestreo es  10.34001
```

Apartado 2 Compara el resultado con el error típico que darías si, por ejemplo, supieras que los datos proceden de una distribución normal.

Solución

Bajo hipótesis de normalidad podría aplicarse el lema de Fisher, que dice así:

Sean $X_1, X_2, \ldots, X_n$ variables aleatorias independientes e identicamente distribuidas de una normal $\mathcal{N}(\mu, \sigma^2)$. Entonces:

1. $\bar{X}$ y S^2 son independientes.
2. $$\frac{(n-1)}{\sigma^2} S^2 \sim \chi^2_{n-1}$$
3. $\bar{X} \sim \mathcal{N}(\mu, \frac{\sigma^2}{n})$.

Puesto que nuestro objetivo es encontrar un estimador de la $Var(S^2)$ tomando la varianza de ambos miembros (2) resulta:

$$Var\left(\frac{n-1}{\sigma^2} S^2\right) = Var(\chi^2_{n-1})$$

que por las propiedades de la varianza y que $var(\chi^2_k) = 2k$ se tiene

$$\frac{(n-1)^2}{\sigma^4} Var\left(S^2\right) = 2(n-1),$$

Por lo que concluímos que

$$Var\left(S^2\right) = \frac{2\sigma^4}{n-1}.$$

Por el apartado anterior, σ puede ser estimada con S, por lo que finalmente podemos concluir que

$$\widehat{Var(S^2)} = \frac{2S^4}{n-1} \tag{1}$$

```r
# El estimador de la varianza de una normal
et_2 <- 2*(varianza ^2)/(size - 1)
cat('Var(S^2) aprox ',et_2)
```

```
Var(S^2) aprox  211.3888
```

```
cat('\nEl erro típico es ', sqrt(et_2))
```

```
El erro típico es  14.53922
```

Notemos que esto solo se puede utilizar para una normal.

Además si comparamos los resultados con los del apartado primero, vemos que conocida la normal, la estimación del error típico es mayor; algo esperable ya que analizando la Ecuación 1, a mayor tamaño de muesta menor varianza y en nuestro caso tenemos una muestra relativamente pequeña.

Como moraleja, a pequeños tamaños de muestra no deberíamos de fiarnos en exceso del resultado boostrap (y ojo, tengamos presente que nuestro error típico procede de una estimación).

Apartado 3 Calcula un intervalo de confianza para σ^2 usando el método bootstrap híbrido. Fija $1-\alpha = 0.95$.

Solución

Para explicar la idea que subyace en el diseño del algoritmo de *boostrap híbrido*, comenzaremos con las siguientes

Se define la proporción como

$$\tilde{H}_n(x) = \frac{1}{B}\sum_{b}^{B} I_{T^{*(b)} \leq x}.$$

Sea

$$H_n(x) = P_F\left(\sqrt{n}\left(\bar{X} - \mu\right) \leq x\right)$$

que por no ser conocido aproximaremos como

$$\hat{H}_n(x) = P_F\left(\sqrt{n}\left(\bar{X}^* - \bar{X}\right) \leq x\right)$$

$$1-\alpha = P\left\{H_n^{-1}\left(\frac{\alpha}{2}\right) \leq \sqrt{n}\left(\hat{\theta} - \theta\right) \leq H_n^{-1}\left(1-\frac{\alpha}{2}\right)\right\}$$

dando lugar al intervalo de confianza

$$\left[\hat{\theta} - \sqrt{n}H_n^{-1}\left(1-\frac{\alpha}{2}\right), \hat{\theta} - \sqrt{n}H_n^{-1}\left(\frac{\alpha}{2}\right)\right]$$

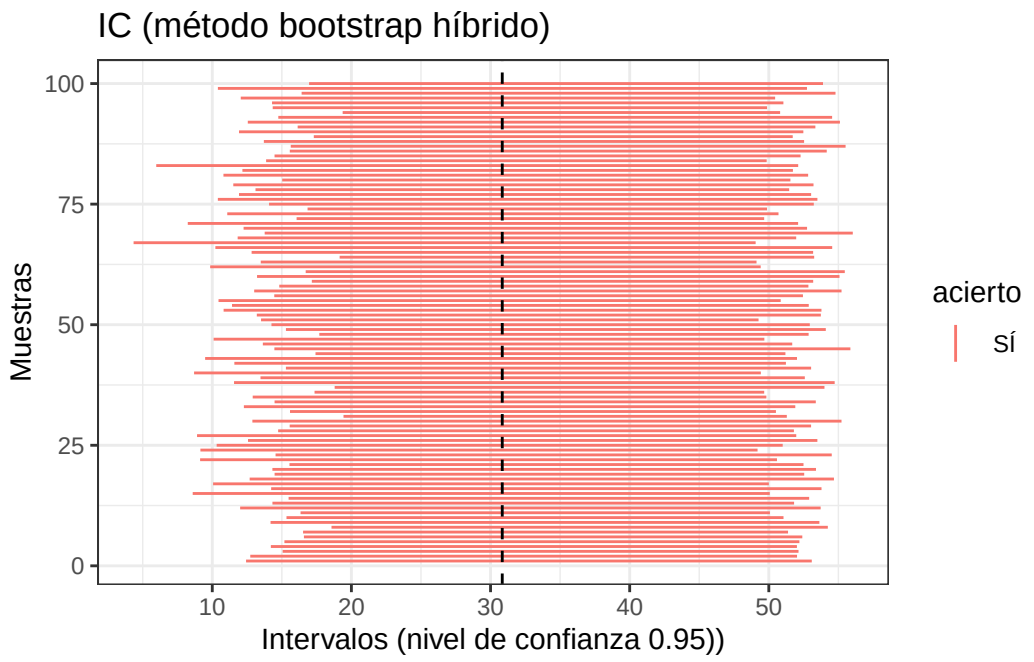
Puesto que H_n no es conocido los sustituiremos por el estimador de *bootstrap* $\hat{H}_n$ (que será la función `quantile` definida en R) y es el llamado *método híbrido*.

De esta manera resulta:

```r
# --- Funciones auxiliares ---
# Construción de la inversa de H(H, muestra_ordenada, B^{-1})
H_inv <- function (alpha, muestra_ordenada, B_inv, acumulado = 0, index = 0) {
  if(acumulado < alpha){
    return (H_inv(alpha, muestra_ordenada, B_inv, acumulado + B_inv, index+1 ))
  }
  else{
    return(muestra_ordenada[index])
  }
}
# En lugar de emplear esta función utilizaremos la función `quantile`

# \hat \theta: Estimador de la varianza
```

```r
# Parámetros
a = 0.05 # alpha
B = length(muestra) # tamaño del reemuestro
numero_remuestreos = 100
repeticiones_experimento = 100

## variable auxiliares
B_inv = 1/B
acierto <- NULL
intervalo <- NULL

for(i in 1:repeticiones_experimento){

  muestras_boostrap <- matrix(
    sample(muestra, B*numero_remuestreos, rep=TRUE),
    nrow = numero_remuestreos
  )

  varianzas_bootstrap = apply(muestras_boostrap, 1, var)
  muestras_normalizadas <- sqrt(B)*(varianzas_bootstrap - varianza)
  ic_min <-varianza - quantile(muestras_normalizadas, 1-a/2)/sqrt(B)
  ic_max <-varianza - quantile(muestras_normalizadas, a/2)/sqrt(B)
  intervalo <- rbind(intervalo, c(ic_min, ic_max))
   acierto <- c(acierto, ic_min < varianza & ic_max > varianza)
}

df <- data.frame(
```

```r
  ic_min <-intervalo[, 1],
  ic_max <- intervalo[, 2],
  ind = 1:numero_remuestreos,
  acierto = acierto
)


ggplot(df) +
  geom_linerange(aes(xmin = ic_min, xmax = ic_max, y = ind, col = acierto)) +
  scale_color_hue(labels = c("SÍ", "NO")) +
  geom_vline(aes(xintercept = varianza), linetype = 2) +
  theme_bw() +
  labs( y= 'Muestras', x = 'Intervalos (nivel de confianza 0.95))',
        title = 'IC (método bootstrap híbrido)'
  )
```



Ejercicio 15

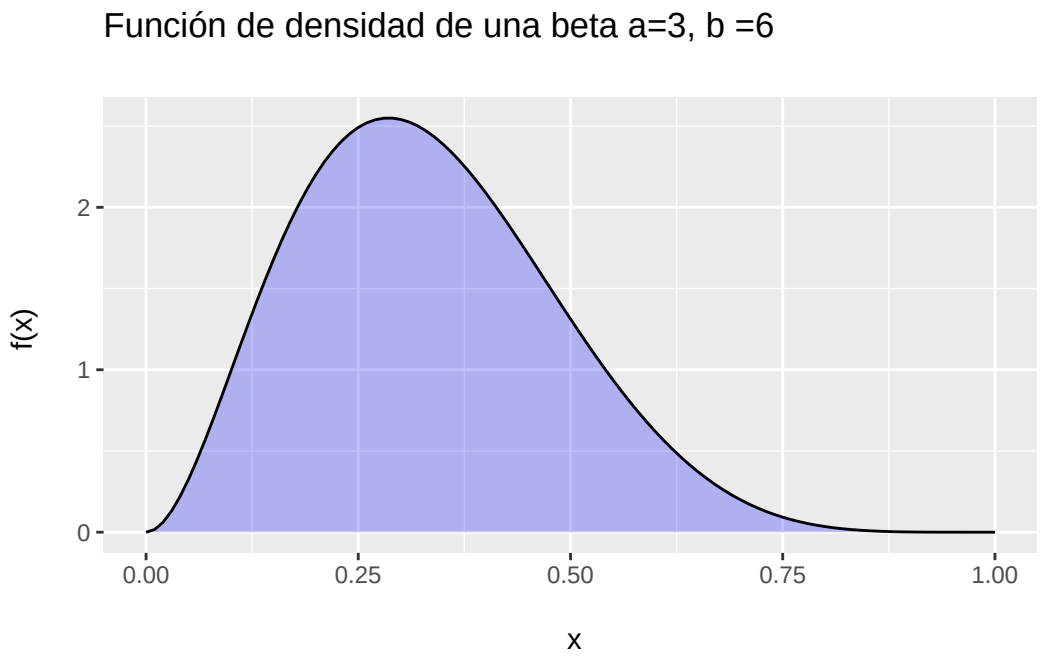
Considera una variable aleatoria con distribución beta de parámetros $\alpha = 3, \beta = 6$.

Apartado 1 Representa gráficamente la función de densidad y la función de distribución.

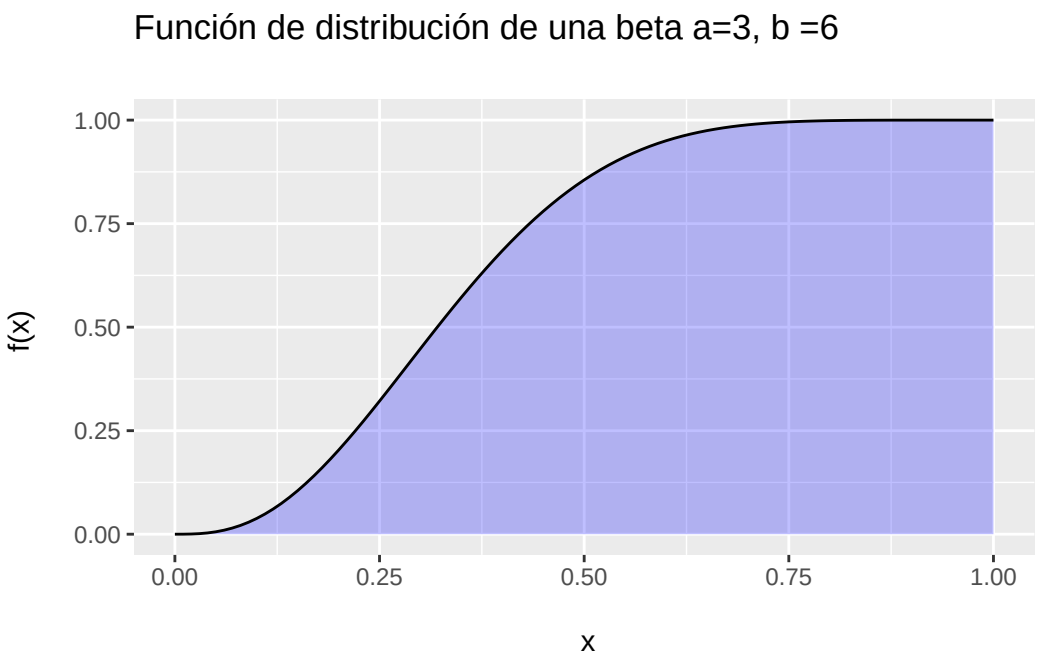
Solución

```
library(ggplot2)
library(purrr)
set.seed(123)
```

```
# Initialize some values.
# Parámetros
a <- 3
b <- 6


# Función de densidad
ggplot() +
  stat_function(fun = dbeta, args = list(shape1 = a, shape2= b),
                geom = "area",
                fill = "blue", alpha = 0.25) +
  stat_function(fun = dbeta, args = list(shape1 = a, shape2= b)) +
  labs(x = "\n x", y = "f(x) \n",
       title = "Función de densidad de una beta a=3, b =6 \n")
```



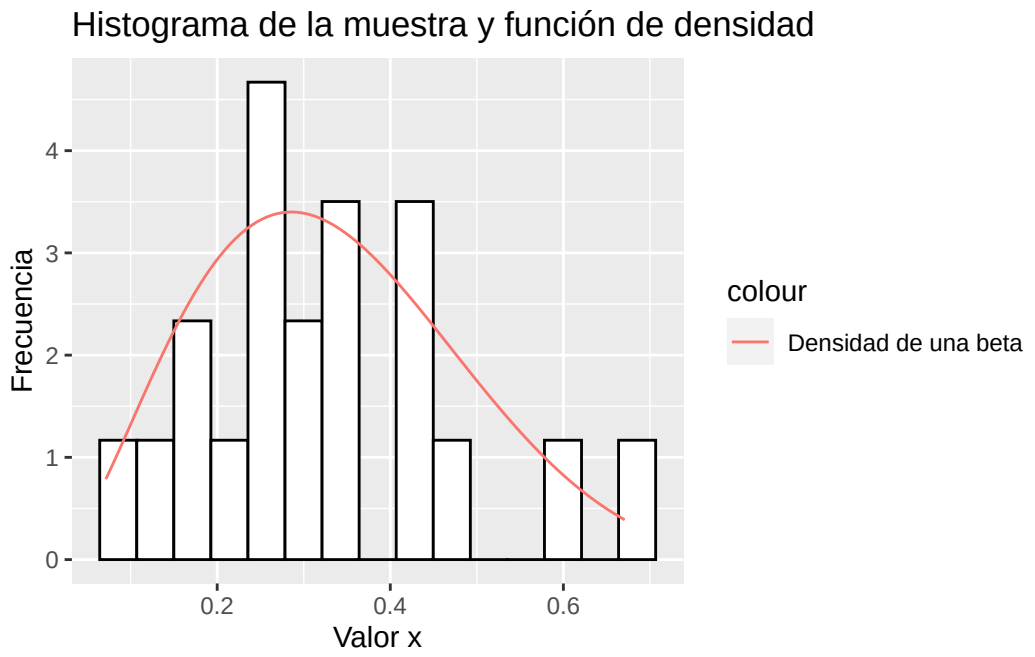

```
## Función de distribución
  ggplot() +
  stat_function(fun = pbeta, args = list(shape1 = a, shape2= b),
                geom = "area",
                fill = "blue", alpha = 0.25) +
  stat_function(fun = pbeta, args = list(shape1 = a, shape2= b)) +
  labs(x = "\n x", y = "f(x) \n",
       title = "Función de distribución de una beta a=3, b =6 \n")
```



Apartado 2 Simula una muestra de tamaño 20 de esta distribución. A continuación, representa en los mismos gráficos del apartado (a) las estimaciones de F y f obtenidas respectivamente mediante la función de distribución empírica F_N y un estimador del núcleo $\widehat{f}$ obtenidos a partir de la muestra simulada.

```
# Paso 1: Simulación de la muestra
sample_size <- 20
sample <- rbeta(sample_size, a, b)

# Paso 2: Representación de las estimaciones
## Función de densida y frecuencia
bins = 15
df <- data.frame(muestras = sample)
  scale <- sample_size/bins
  # get adapted norm
  scaled_norm <- function (x) scale*dbeta(x, a,b)

  ggplot(df, aes(x = sample)) +
    ggtitle("Histograma de la muestra y función de densidad" ) +
    xlab("Valor x") + ylab("Frecuencia") +
      geom_histogram(aes(y =..density..),
                     colour = "black",
                     fill = "white",
```

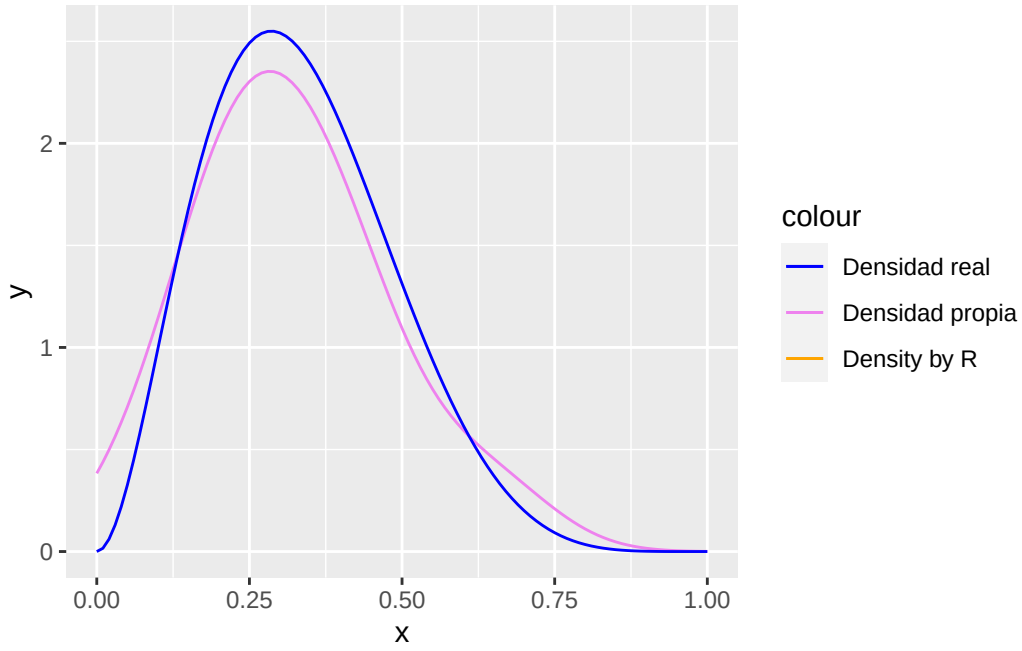
```
                    bins = bins) +
  geom_function(aes(colour = "Densidad de una beta"), fun = scaled_norm )
```



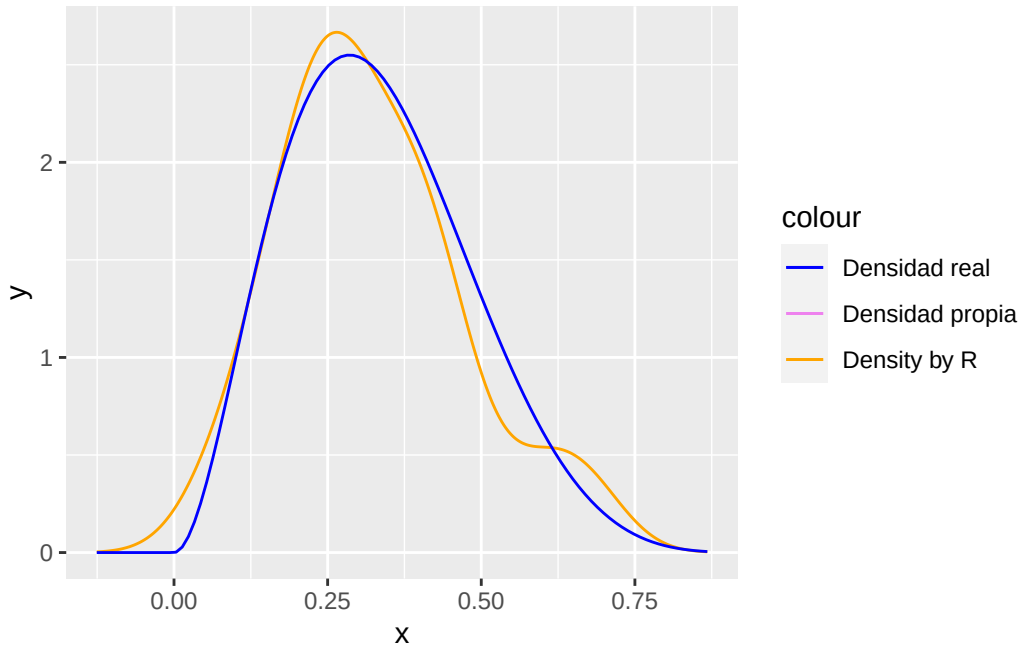
Esto nos da una idea intuitiva de la distribución, vamos a calcularla apriximadamente con estimadores de núcleo:

```
# Parámetros
h <- 0.1
K <- function(x) dnorm(x,0,1) # Utilizaremos un núcleo gaussiano

f <- function(x, our_sample = sample){
  sample_size = length(our_sample)
  return (sum(map_dbl(our_sample, function(xi)(K((x-xi)/h))))/(h*sample_size))
}
f_vectorial <- function(x, s) c(map(x, function(y)f(y,s)))

x <- seq(0,1,0.01)
y <- map_dbl(x,f)
estimador_nucleo<- density(sample) # versión de R

colors <- c("Densidad real" = "blue",
            "Densidad propia" = "violet",
```

```
            "Density by R" = "orange")

# Densidad propia
df <- data.frame(x=x, y=y)
ggplot(data = df, aes(x, y, color = "Densidad propia")) +
  geom_line() +
  stat_function(fun = dbeta, args = list(shape1 = a, shape2= b),
                aes(color = "Densidad real"))+
  scale_color_manual(values = colors)
```

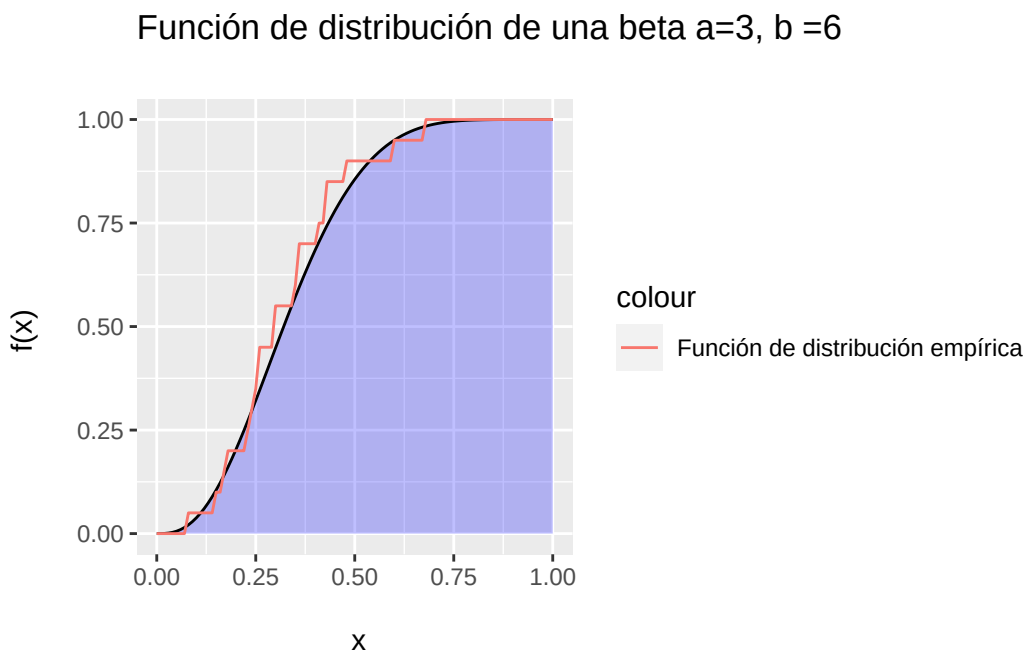



```
# Densidad de R
df <- data.frame(x = estimador_nucleo$x, y = estimador_nucleo$y)
ggplot(data = df, aes(x, y, color = "Density by R")) +
  geom_line() +
  stat_function(fun = dbeta, args = list(shape1 = a, shape2= b),
                aes(color = "Densidad real"))+
  scale_color_manual(values = colors)
```




```
  ## Función de distribución y la empírica
  # Función de distribución a partir de la muestra
F <- function (x, X=sample){
    n = length(X)
    lower = length(X[X <= x])
    return (lower / n)
}
F_vectorial <- function (vector_of_x,X = sample){
  lower_vector <- NULL
  n = length(X)
  for(x in vector_of_x){
    lower_vector <- rbind(lower_vector,length(X[X <= x]))
  }
  return(lower_vector / n)
}
ggplot() +
  stat_function(fun = pbeta, args = list(shape1 = a, shape2= b),
                geom = "area",
                fill = "blue", alpha = 0.25) +
  stat_function(fun = pbeta, args = list(shape1 = a, shape2= b)) +
  geom_function(fun = F_vectorial,
                aes(colour = "Función de distribución empírica")) +
```

```
  xlim(0,1) +
  labs(x = "\n x", y = "f(x) \n",
       title = "Función de distribución de una beta a=3, b =6 \n")
```



Apartado 3 Verifica empíricamente el grado de aproximación alcanzado en las estimaciones de F y f. Para ello, genera 200 muestras de tamaño 20 y para cada una de ellas evalúa el error (medido en la norma del supremo, es decir, el máximo de las diferencias entre las funciones) cometido al aproximar F por F_n y f por $\widehat{f}$.Por último, calcula el promedio de los 200 errores obtenidos.

```
# Generación de muestras
number_of_samples <- 200
sample_size = 20
samples_matrix <- matrix(
  rbeta(sample_size*number_of_samples, a, b),
  nrow = number_of_samples
)

# Estimamos
error_f_opt <- NULL
error_f <- NULL
error_F <- NULL
```

```r
error_fr<- NULL

dominious = seq(0,1,0.05)
for(i in 1:number_of_samples){
  # Error de la función de densidad
  diferencef <-function(x) (abs(f(x, samples_matrix[i, ]) - pbeta(x, a,b)))
  error_f <- rbind(error_f,max(unlist(map(dominious, diferencef))))
  # Densididad de R
  dr <- density(samples_matrix[i, ])
  error_fr <-rbind(error_fr, max(abs(dr$y - pbeta(dr$x, a,b))))
  # Función de distribución
  diferenceF <-function(x) (abs(F_vectorial(x, samples_matrix[i, ]) - dbeta(x, a,b)))
  error_F <- rbind(error_F,
                   max(unlist(diferenceF(dominious)))
                   )
}


# Mostramos los resultados
## ¡Error en la función de densidad
media_f <- mean(c(error_f))
std_f <- sd(c(error_f))
cat("El error de la función de densidad propia diferencia con la real es de",
    media_f , "+-", std_f)
```

```
El error de la función de densidad propia diferencia con la real es de 1.855399 +- 0.3109275
```

```r
## ¡Error en la función de densidad calculada con r
media_fr <- mean(c(error_fr))
std_fr <- sd(c(error_fr))
cat("\nEl error de la función de densidad de r  con la real es de",
    media_fr, '+-', std_fr)
```

```
El error de la función de densidad de r  con la real es de 2.322613 +- 0.7106429
```

```r
## Distribución empírica
media_F <- mean(c(error_F))
std_F <- sd(c(error_F))
cat("\nEl error de la función de distribución empírica con la real es de",
    media_F, '+-', std_F)
```

```
El error de la función de distribución empírica con la real es de 2.174541 +- 0.09127971
```

Podemos apreciar que el error de la implementación nuestra es en media mejor (aunque abría que realizar un test para poder afirmarlo), esto puede deberse a que hemos ejecutado la función `density` sin buscar hiperparámetros mejores.