MAE: ENTREGA II

Jon Zorrilla Gamboa

1. Ejercicio 1

Una vez definidas las siguientes funciones:

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


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

Tenemos que estimar los parámetros de una normal y comparar la función de distribución empírica F_n con la función de distribución de una variable $N(\mu, \sigma^2)$.

1.1.

Para generar 1000 muestras de tamaño 20, hacemos lo siguiente:

```
ksnoest_list <- c()
ksest_list <- c()
for(i in 1:1000){
  x <- rnorm(20)
  ksnoest_list <- append(ksnoest_list, ksnoest(x), after = length(ksnoest_list))
```

```
  ksest_list <- append(ksest_list, ksest(x), after = length(ksest_list))
}
```

A partir de donde obtenemos dos listas de 1000 valores donde cada elemento de la lista está asociado al estadístico KS para cada muestra creada a partir de una distribución normal.

1.2.

Ahora, representaremos los diagramas de cajas para ambos estadísticos con las siguientes funciones:

```
boxplot(ksest_list, horizontal = TRUE, xlab = "ksest")
```

Para *ksest*:

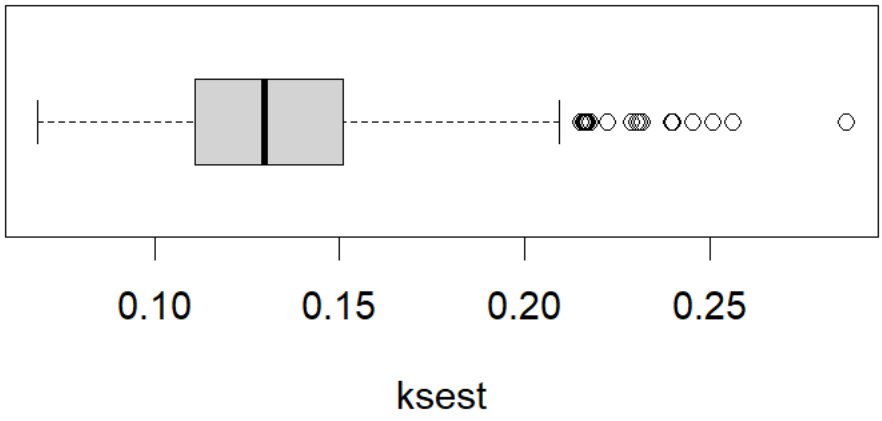

Figura 1: Diagrama de cajas de los estadísticos obtenidos para ksest.

y, para *ksnoest*:

```
boxplot(ksnoest_list, horizontal = TRUE, xlab = "ksnoest")
```

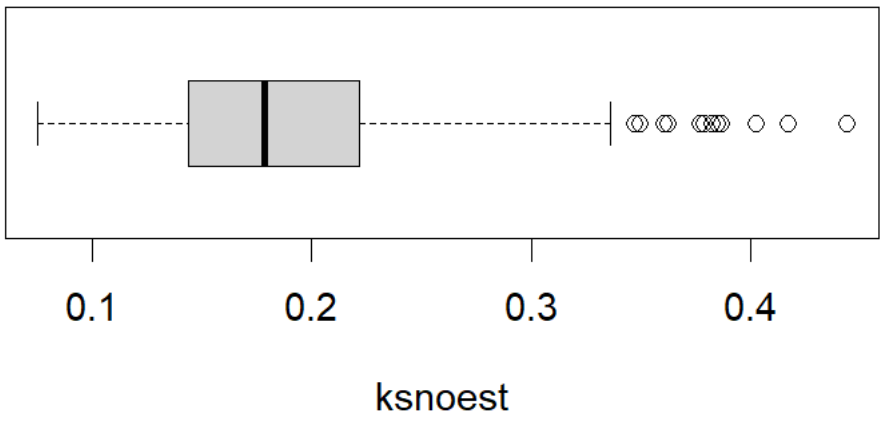

Figura 2: Diagrama de cajas de los estadísticos obtenidos para knoest.

Como se puede observar, se obtiene una media de 0,1329397 para *ksest* y 0,1875089 para *ksnoest*. Es decir, se obtiene una media inferior para ksest. Esto podría darse porque la función *ksest* compara la distribución generada por nosotros con la distribución normal con misma media y varianza que la que nosotros hemos generado. En este caso, debido a que no hemos simulado un número suficientemente grande de muestras la media de la distribución no será cero. Entonces, pese a que esta distribución (muestral) esté originalmente centrada en el cero, la media no será nula, por lo que el estadístico KS dará un valor más grande para *ksnoest*, y *ksest* hará una estimación mejor.

Además, si hubiésemos realizado el muestro de los datos provenientes de una normal no centrada en el origen, la función knoest habría dado un resultado mucho peor, pues la habría comparado con la función normal centrada en el origen.

1.3.

Si hacemos uso de las tablas KS para hacer el contraste a nivel de α, debemos de tener en cuenta que realmente, mediante el método KS, lo que se hace es comparar la distribución original con una distribución normal cuyos valores de media y varianza son los valores de media y varianza de la muestra original. Es decir, realmente no estamos comparando nuestra función con una función normal usual. Es por ello, que el verdadero nivel de significación no será el que valor al que estamos acostumbrados, si no que será un valor mucho menor, pues estamos descartando valores de manera mucho más precisa, ya que estamos comparando con una función que realmente se asemeja más.

1.4.

Hacemos lo siguiente:

```
ksest_list_n <- which(ksest_list>0.19)
length(ksest_list_n)/length(ksest_list)
```

De esta manera, hallamos la proporción de valores (obtenidos tras usar la función *ksest*) mayores a 0.19. Podemos ver que el porcentaje de valores obtenidos mayores a 0.19 es del 4,5 %.

1.5.

Realizamos la siguiente simulación:

```
ksnoest_list <- c()
ksest_list <- c()
```

```
for(i in 1:1000){
  x <- rnorm(40)
  ksnoest_list <- append(ksnoest_list, ksnoest(x), after = length(ksnoest_list))
  ksest_list <- append(ksest_list, ksest(x), after = length(ksest_list))
}

ksest_list_n <- which(ksest_list>0.12)
length(ksest_list_n)/length(ksest_list)
```

Volvemos a hacer el proceso anterior pero haciendo uso de 40 muestras aleatorias provenientes de la normal. El valor que se obtiene es $\alpha = 0{,}159$. Es decir, el 84,1 % de las veces el valor será menor a 0,12.

2. Ejercicio 2

Definimos la siguiente muestra:

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

Donde la varianza que se obtiene es de 30.84233.

2.1.

Dada la muestra original, hacemos uso del método bootstrap para determinar el error típico de este estimador de σ^2.

```
library(ggplot2)
set.seed(100)

# Parametros
R <- 100
n <- 10

# Generamos los datos
muestra_original <- c(1, 2, 3.5, 4, 7, 7.3, 8.6, 12.4, 13.8, 18.1)
var_original <- var(muestra_original)
```

```
# Generamos las remuestras
muestras_bootstrap <- sample(muestra_original, n*R, rep=TRUE)
muestras_bootstrap <- matrix(muestras_bootstrap, nrow = n)

# Varianza de las remuestras
var_bootstrap <- apply(muestras_bootstrap, 2, var)

# Histogramas de las medianas bootstrap
df <- data.frame(var_bootstrap = var_bootstrap)
ggplot(df, aes(x = var_bootstrap)) +
  geom_histogram(aes(y = ..density..),
                 bins = 20, fill = 'red', col = 'black') +
  geom_vline(xintercept = var_original, size = 1.1) +
  geom_density(aes(y=..density..))

# Estimador bootstrap de la desviacion típica de la varianza
sd_var <- sd(var_bootstrap)
sd_var
```

El valor obtenido para la desviación típica de la mediana es 10,46382. También podemos estudiar un histograma asociado a los valores obtenidos para la varianza.

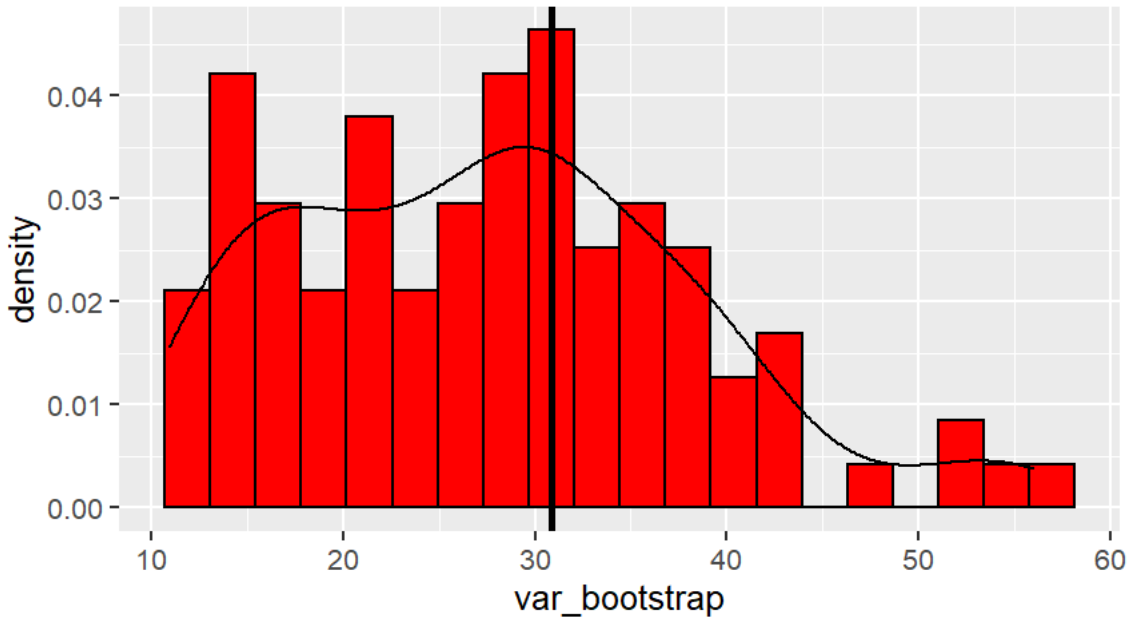

Figura 3: Histograma de las varianzas en método bootstrap.

2.2.

Para comparar este error típico obtenido con el que obtendríamos si los datos procediesen de una distribución normal, haremos uso de la siguiente ecuación:

$$\frac{(n-1)S^2}{\sigma^2} \simeq \chi^2_{n-1} \tag{1}$$

Si tomamos la varianza en ambos lados de la ecuación,

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

Entonces,

$$Var(S^2) = \frac{2\sigma^4}{n-1} \tag{3}$$

De esta manera, podemos calcular el estimador pues conocemos $n = 10$ y $\hat{\sigma^4} = S^4$, con $S^2 = 30{,}8423$. Entonces, para calcular $\sigma_{\hat{S^2}} = \sqrt{Var(S^2)}$.

$$\sigma_{\hat{S^2}} = \sqrt{\frac{2 \cdot 30{,}8423^2}{9}} \approx 14{,}5392 \tag{4}$$

2.3.

Para calcular un intervalo de confianza de σ^2 usando el método de bootstrap híbrido, definimos el siguiente estadístico:

$$T(X_1, X_2, ..., X_n; F) = \sqrt{n}(S_n^2 - \sigma^2) \tag{5}$$

que sabemos que si su distribución $H_n(x)$ fuese conocida, existiría un intervalo de confianza $1-\alpha$ tal que:

$$(S^2 - n^{-\frac{1}{2}} H_n^{-1}(1-\alpha/2), S^2 - n^{-\frac{1}{2}} H_n^{-1}(\alpha/2)) \tag{6}$$

Para conocer la distribución de H_n, haremos uso de nuevo del método bootstrap. Para ello, realizaremos R remuestreos bootstrap, para cada remuestra calculamor el valor del estadístico $T = \sqrt{n}((S^2)^{*i} - S^2)$, y, una vez ordenados, consideramos los que dejan una proporción de valores $\frac{\alpha}{2}$ a ambos lados.

```r
set.seed(100)

# Parametros
R <- 1000
n <- 10
alpha <- 0.05
```

```r
# Muestra original
muestra_original <- c(1, 2, 3.5, 4, 7, 7.3, 8.6, 12.4, 13.8, 18.1)
var_original <- var(muestra_original)

# Bootstrap
muestras_bootstrap <- sample(muestra_original, n*R, rep = TRUE)
muestras_bootstrap <- matrix(muestras_bootstrap, nrow = n)

# Varianza de las remuestras bootstrap
var_bootstrap <- apply(muestras_bootstrap, 2, var)

# Estimador T
T_bootstrap <- sqrt(n)*(var_bootstrap - var_original)

# Limites del intervalo de confianza
ci_min <- var_original - quantile(T_bootstrap, 1 - alpha/2)/(sqrt(n))
ci_max <- var_original - quantile(T_bootstrap, alpha/2)/sqrt(n)

ci_min
ci_max
```

Y, obtenemos $ci_{min} = 11{,}99636$ asociado al valor más pequeño (2,5 %) y $ci_{max} = 54{,}01567$ asociado al valor más grande (97,5 %). Esto implica que el valor de σ^2 estará acotado entre estos dos valores el 95 % de las veces;

$$P(11{,}99636 < \sigma^2 < 54{,}01567) = 0{,}95 \tag{7}$$

3. Ejercicio 3

Consideramos una variable aleatoria que sigue una distribución beta de parámetros $\alpha = 3$ y $\beta = 6$.

3.1.

En primer lugar, representaremos la función de densidad:

Hacemos uso del siguiente código:

```
densidad <- ggplot()+
  ggtitle("Funcion de densidad Beta(3, 6)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_function(fun = dbeta, args = list(3, 6),size = 1.1, col = 'black')+
  labs(x="x",y="Densidad")+
  xlim(0.0, 1.0)
```

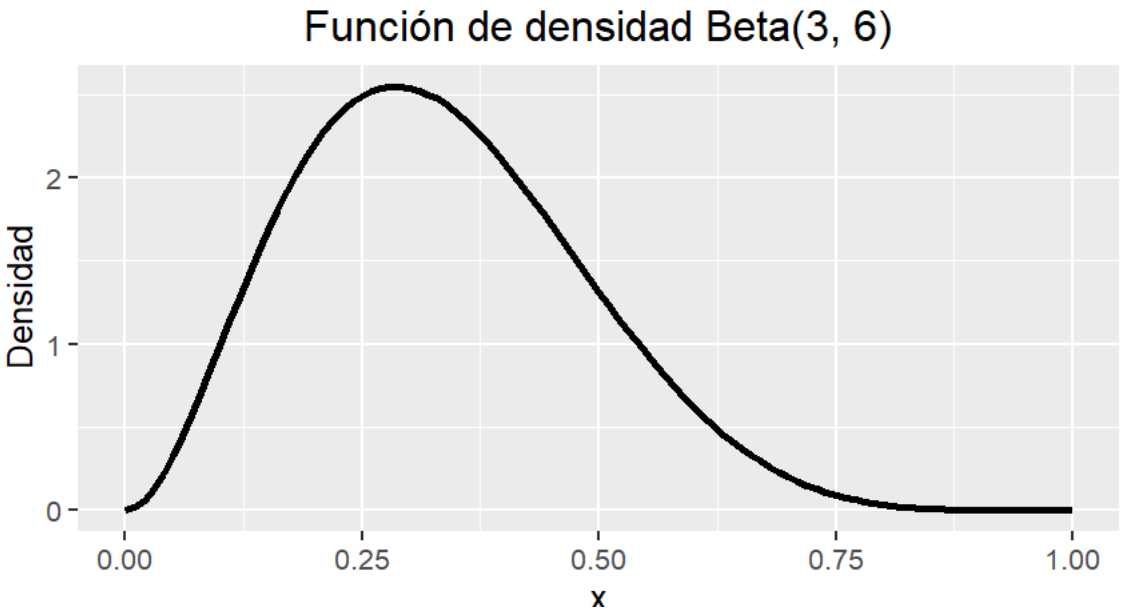

Figura 4: Función densidad de beta.

Para representar la función de probabilidad, hacemos lo siguiente:

```
graf2 <- ggplot()+
  ggtitle("Funcion de distribucion Beta(3, 6)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_function(fun = pbeta, args = list(3,6),size = 1.1, col = 'red') +
  labs(x="x",y="Distribucion")+
  xlim(0.0, 1.0)
```

representamos la función:

3.2.

A partir de la muestra proveniente de la distribución beta, simularemos el estimador del núcleo y la función de densidad.

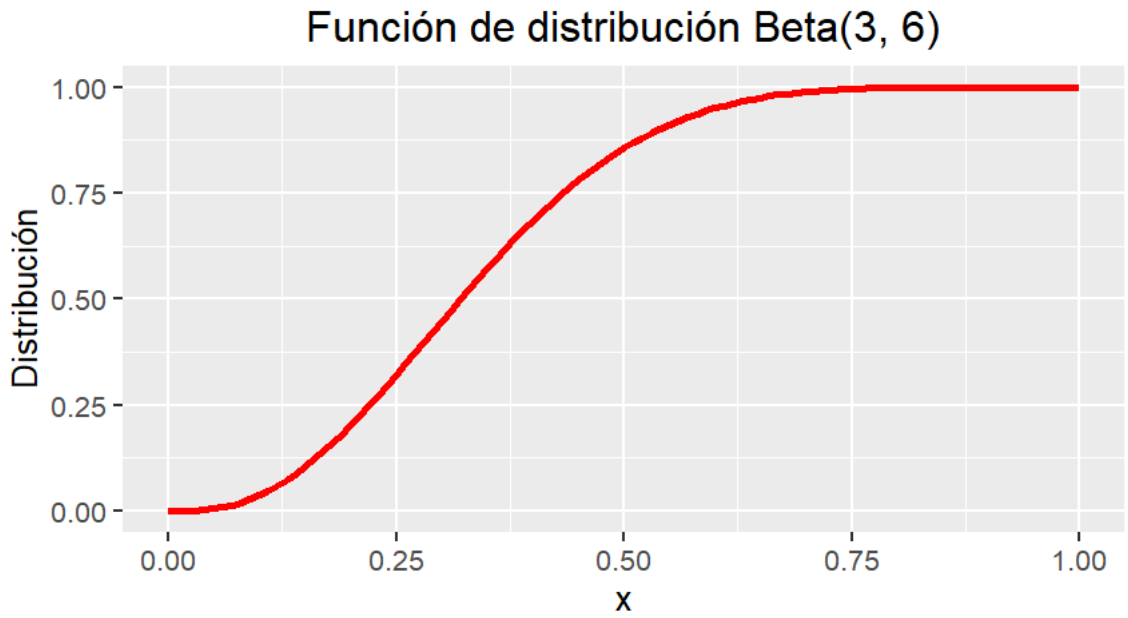

Figura 5: Función de distribución beta.

```
n <- 20
muestra <- rbeta(n,alpha,beta)
est_nucleo <- density(muestra)
plot(est_nucleo,
     main = "Funcion de densidad vs. estimador del nucleo",
     xlab = "Muestra", ylab = "Densidad")
curve(dbeta(x, alpha, beta), from = 0, to = 1, lty = 2, add = TRUE, col = 'black')
```

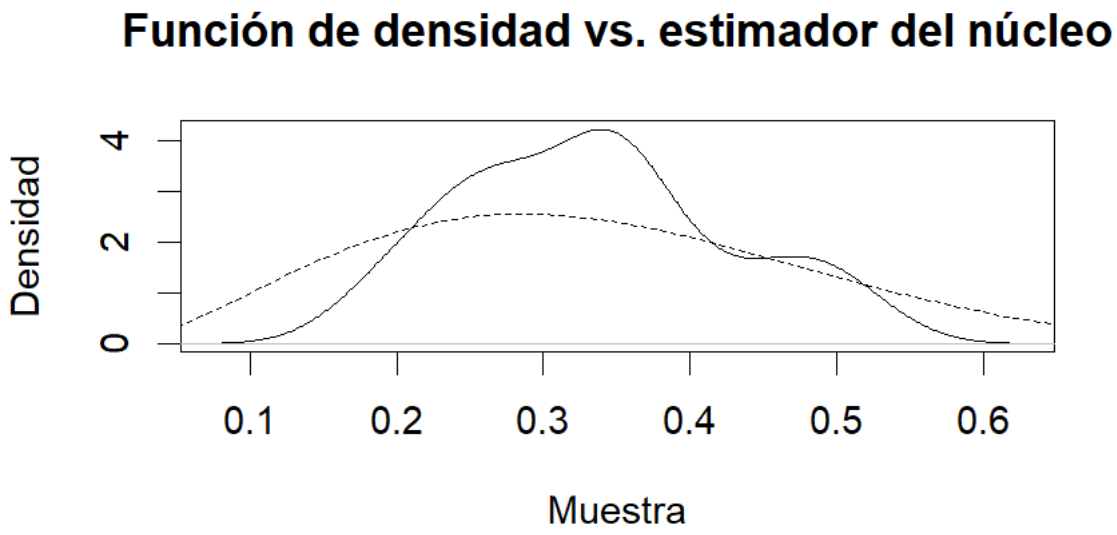

Figura 6: Comparación de función de densidad vs. función estimador del núcleo.

```
df <- data.frame(muestra)
ggplot(df, aes(muestra))+
    ggtitle("Funcion de distribucion vs. función empirica") +
    theme(plot.title = element_text(hjust = 0.5)) +
    stat_ecdf(aes(muestra, linetype='Funcion empirica')) +
    stat_ecdf(aes(muestra), geom = "point") +
    geom_function(fun = pbeta, args = list(alpha, beta),
                  col = "red", aes(linetype = "F(x)")) +
    labs(x = "Muestra", y = "Distribucion", linetype="") +
    xlim(0.0, 1.0)
```

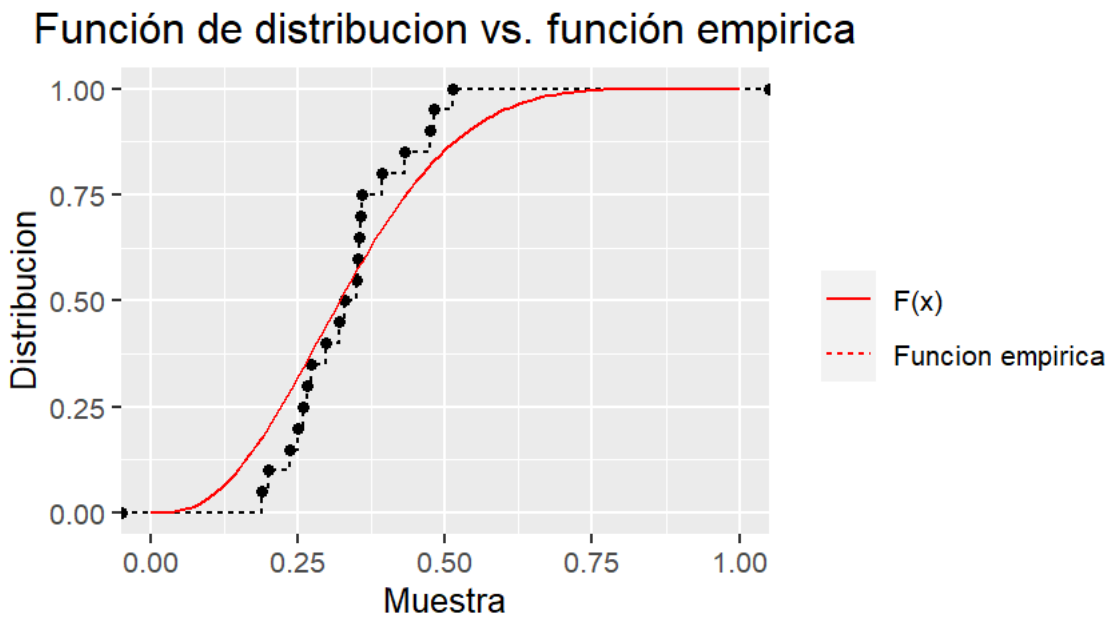

Figura 7: Comparación de función de distribución beta vs. función empírica.

3.3.

Para verificar el grado de aproximación en las estimaciones de F y f, con los parámetros previamente definidos, generamos 200 muestras de tamaño 20 y hallamos el error (el supremo) al aproximar F por F_n y f por $\hat{f}$:

```
set.seed(100)
n <- 20
alpha <- 3
beta <- 6
```

```
error1 <- NULL
error2 <- NULL

for (i in 1:200){
  muestra <- rbeta(n,alpha,beta)
  F_ks <- ks.test(muestra,"pbeta",alpha,beta)
  error1 <- c(error1,F_ks$statistic)
  estimador <- dbeta(muestra,alpha,beta)
  nucleo <- density(muestra,n=20)$y
  f_ks <- ks.test(nucleo,estimador)
  error2 <- c(error2,f_ks$statistic)
}

mean(error1)
mean(error2)
```

Obtenemos las siguientes medias de los errores: 0,1921112 al aproximar F por F_n y 0,506 al aproximar f por $\hat{f}$.