Notas del curso CA-403

1.2.6. Laboratorio

Comenzaremos con una librería bastante básica llamada `KernSmooth`.

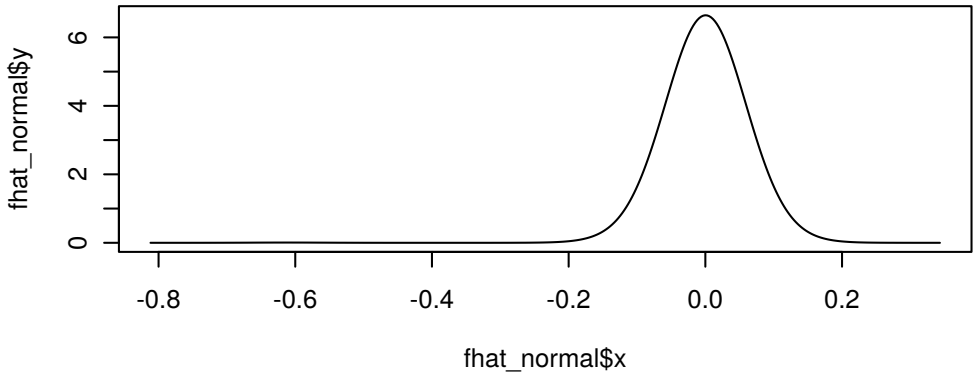
Efecto de distintos Kernels en la estimación

```
x <- read.csv("data/stockres.txt")
x <- x$STOCKRETURN
```

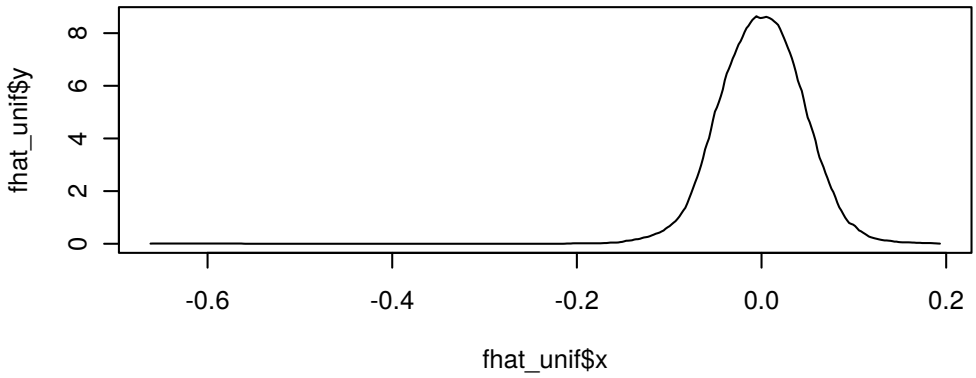
```
summary(x)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.6118200 -0.0204085 -0.0010632 -0.0004988  0.0215999  0.1432286
```

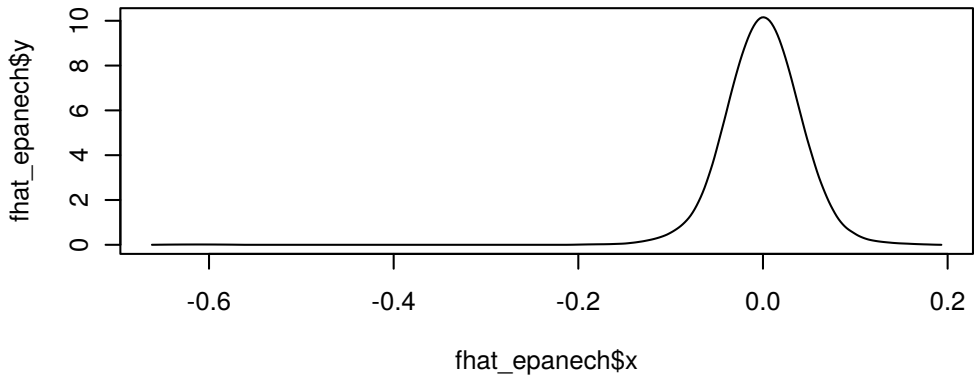
```
library(KernSmooth)

fhat_normal <- bkde(x, kernel = "normal", bandwidth = 0.05)
plot(fhat_normal, type = "l")
```



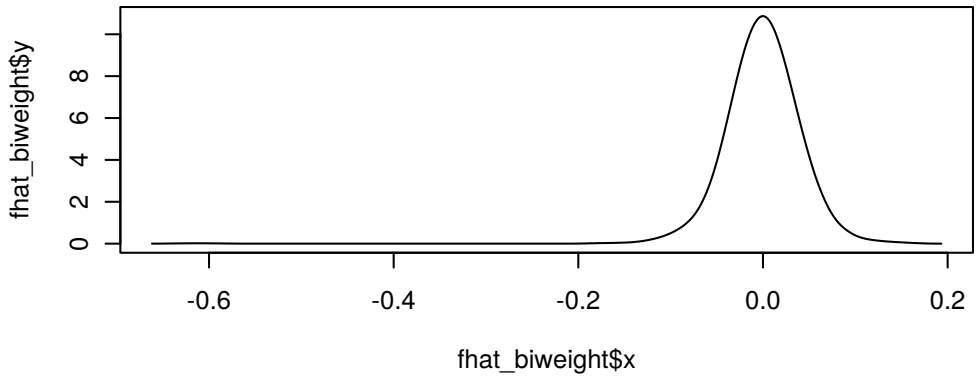

```
fhat_unif <- bkde(x, kernel = "box", bandwidth = 0.05)
plot(fhat_unif, type = "l")
```



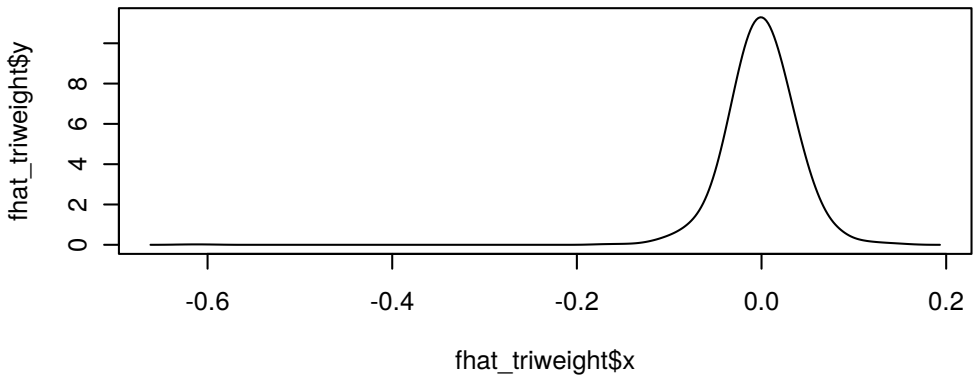

```
fhat_epanech <- bkde(x, kernel = "epanech", bandwidth = 0.05)
plot(fhat_epanech, type = "l")
```




```
fhat_biweight <- bkde(x, kernel = "biweight", bandwidth = 0.05)
plot(fhat_biweight, type = "l")
```




```
fhat_triweight <- bkde(x, kernel = "triweight", bandwidth = 0.05)
plot(fhat_triweight, type = "l")
```

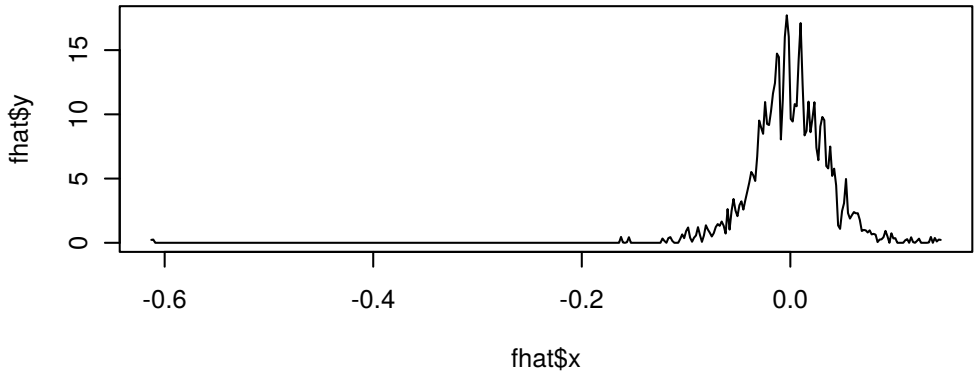


Efecto del ancho de banda en la estimación

```
fhat <- bkde(x, kernel = "box", bandwidth = 0.001)
```

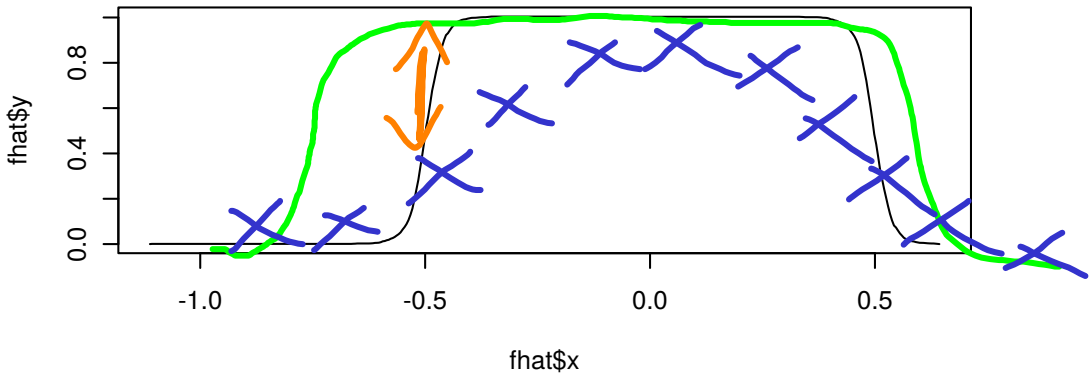
```
## Warning in bkde(x, kernel = "box", bandwidth = 0.001): Binning
grid too coarse for current (small) bandwidth: consider increasing
'gridsize'
```

```
plot(fhat, type = "l")
```

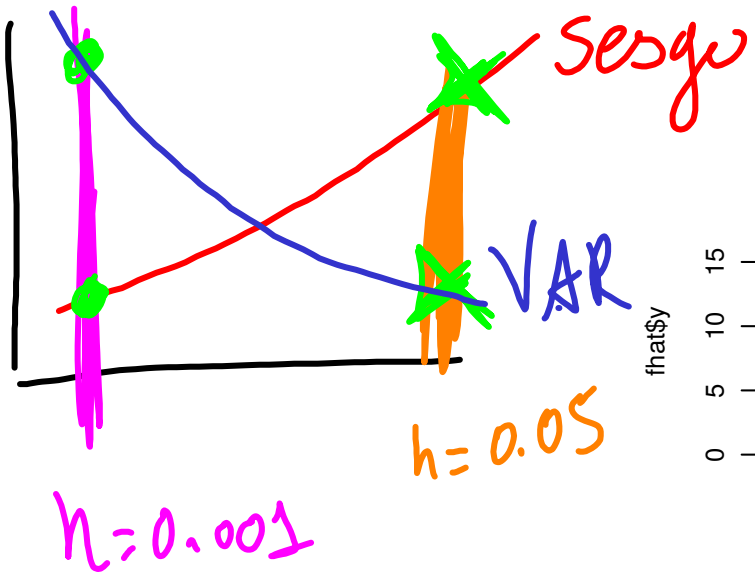


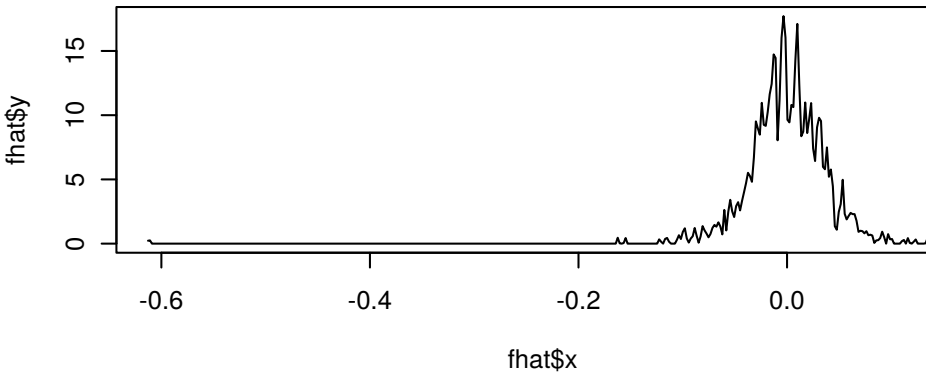
Kernel uniforme

```
fhat <- bkde(x, kernel = "box", bandwidth = 0.5)
plot(fhat, type = "l")
```



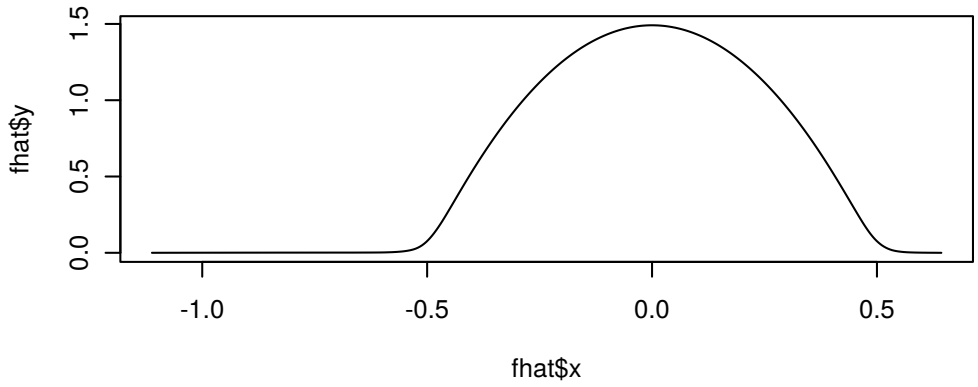
```
fhat <- bkde(x, kernel = "epa", bandwidth = 0.001)

## Warning in bkde(x, kernel = "epa", bandwidth = 0.001): Binning
grid too coarse for current (small) bandwidth: consider increasing
'gridsize'

plot(fhat, type = "l")
```





Kernel Epanechnikov

```
fhat <- bkde(x, kernel = "epa", bandwidth = 0.5)
plot(fhat, type = "l")
```




```
suppressMessages(library(tidyverse))
library(gganimate)

fani <- tibble()

for (b in seq(0.001, 0.02, length.out = 40)) {
 f <- bkde(x, kernel = "epa", bandwidth = b, gridsize = length(x))
 fani <- fani %>% bind_rows(tibble(xreal = sort(x), x = f$x, y = f$y, bw = b))
}

ggplot(data = fani) +
 geom_line(aes(x, y), color = "blue") +
 labs(title = paste0("Ancho de banda = {closest_state}")) +
 transition_states(bw) +
```

```
 view_follow() +
 theme_minimal(base_size = 20)


anim_save("manual_figure/bandwidth-animation.gif")
```

Pregunta 1.2.22

1. Construya una variable llamada ‘u‘ que sea una secuencia de -0.15 a 0.15 con un paso de 0.01
2. Asigne ‘x‘ a los datos ‘stockrel‘ y calcule su media y varianza.
3. Usando la función ‘dnorm‘ construya los valores de la distribución de los datos usando la media y varianza calculada anteriormente. Asigne a esta variable ‘f_param‘.
4. Defina un ancho de banda ‘h‘ en 0.02
5. Construya un histograma para estos datos con ancho de banda

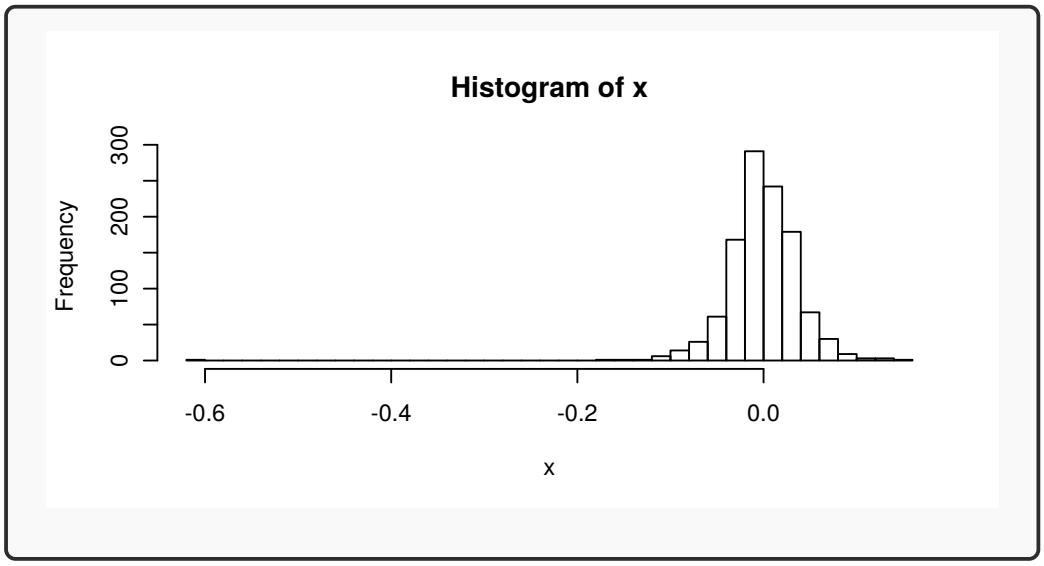
'h'. Llame a esta variable 'f_hist'

6. Usando el paquete 'KernSmooth' y la función 'bkde', construya una función que calcule el estimador no paramétrico con un núcleo Epanechivok para un ancho de banda h. Llame a esta variable 'f_epa'.

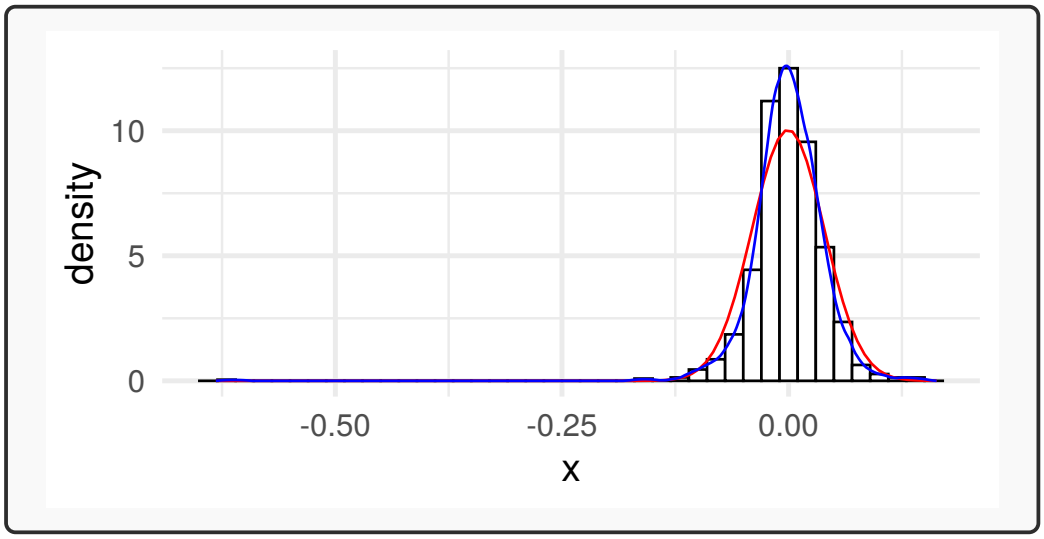
7. Dibuje en el mismo gráfico la estimación paramétrica y no paramétrica.

Solución 1.2.23

```
x <- read.csv("data/stockres.txt")

x <- unlist(x)

# Eliminar nombres de las columnas

names(x) <- NULL


u <- seq(-0.15, 0.15, by = 0.01)


mu <- mean(x)

sigma <- sd(x)


f_param <- dnorm(u, mean = mu, sd = sigma)


h <- 0.02


n_bins <- floor(diff(range(x)) / h)


f_hist <- hist(x, breaks = n_bins)
```


Histogram of x
Frequency
0
100
200
300
-0.6
-0.4
-0.2
0.0
x

```r
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

x_df <- data.frame(x)

library(ggplot2)

ggplot(x_df, aes(x)) +
 geom_histogram(
   aes(y = ..density..),
   binwidth = 0.02,
   col = "black",
   fill = "white"
 ) +
 stat_function(
   fun = dnorm,
   args = list(mean = mu, sd = sigma),
   color = "red"
 ) +
 geom_line(data = f_epa, aes(x, y), color = "blue") +
 theme_minimal(base_size = 20)
```



Ancho de banda óptimo

Usemos la regla de la normal o también conocida como Silverman. **Primero recuerde que en este caso se asume que $f(x)$ sigue una distribución normal**. En este caso, lo que se obtiene es que

$$\begin{aligned} \|f''\|_2^2 &= \sigma^{-5} \int \{\phi''\}^2 dx \\ &= \sigma^{-5} \frac{3}{8\sqrt{\pi}} \approx 0{,}212\sigma^{-5} \end{aligned}$$

donde ϕ es la densidad de una normal estándar.

El estimador para σ es

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}.$$

Y usando el cálculo realizado anteriormente, se obtiene que

$$h_{normal} = \left(\frac{4s^5}{3n}\right)^{1/5} \approx 1{,}06 s n^{-1/5}.$$

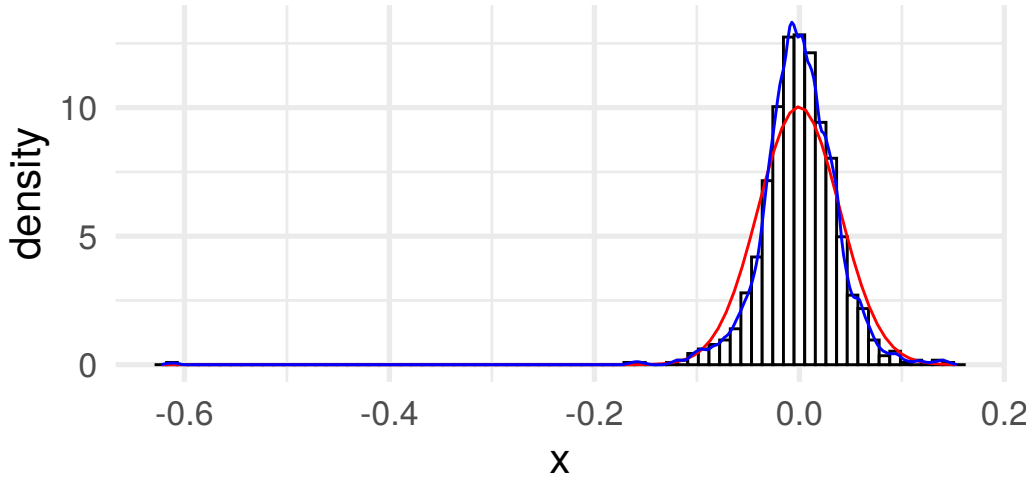
Un estimador más robusto es

$$h_{normal} = 1{,}06 \min\left\{s, \frac{IQR}{1{,}34}\right\} n^{-1/5}.$$

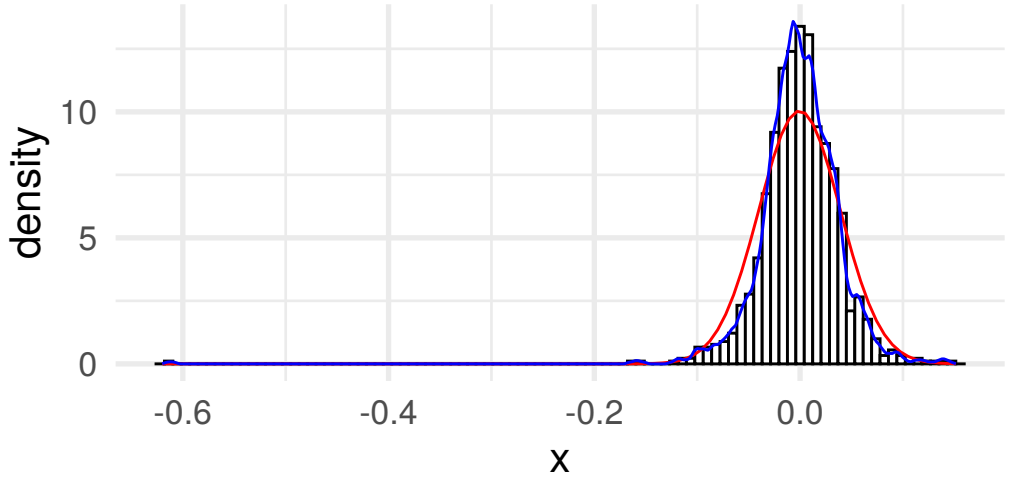
¿Por qué es $IQR/1{,}34$?

```
s <- sd(x)
n <- length(x)

h_normal <- 1.06 * s * n^(-1 / 5)

h <- h_normal

n_bins <- floor(diff(range(x)) / h)
f_hist <- hist(x, breaks = n_bins, plot = FALSE)
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

ggplot(x_df, aes(x)) +
```

```
geom_histogram(
  aes(y = ..density..),
  binwidth = h,
  col = "black",
  fill = "white"
) +
stat_function(
  fun = dnorm,
  args = list(mean = mu, sd = sigma),
  color = "red"
) +
geom_line(data = f_epa, aes(x, y), color = "blue") +
theme_minimal(base_size = 20)
```



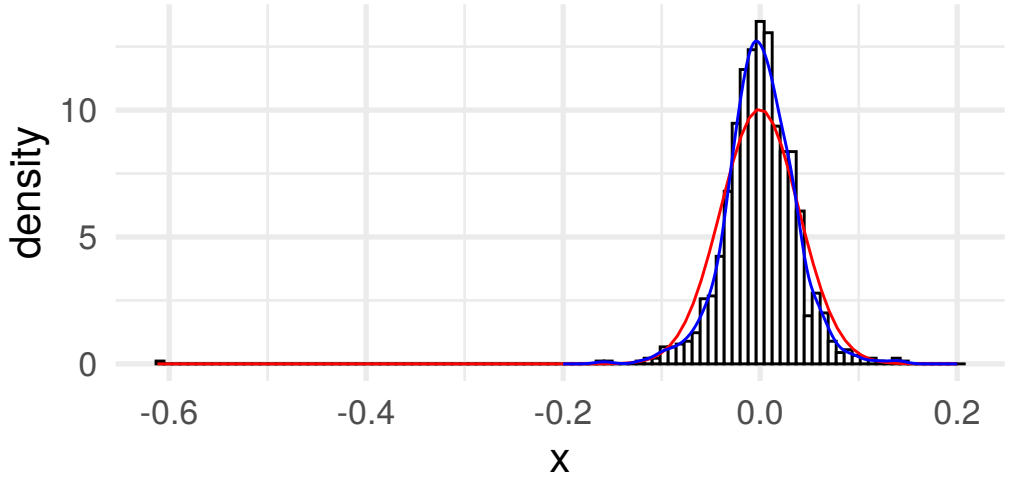
```r
h_iqr <- 1.06 * min(s, IQR(x) / 1.34) * n^(-1 / 5)

h <- h_iqr

n_bins <- floor(diff(range(x)) / h)
f_hist <- hist(x, breaks = n_bins, plot = FALSE)
f_epa <- as.data.frame(bkde(x, kernel = "epa", bandwidth = h))

ggplot(x_df, aes(x)) +
 geom_histogram(
   aes(y = ..density..),
   binwidth = h,
   col = "black",
   fill = "white"
 ) +
 stat_function(
   fun = dnorm,
   args = list(mean = mu, sd = sigma),
   color = "red"
 ) +
 geom_line(data = f_epa, aes(x, y), color = "blue") +
 theme_minimal(base_size = 20)
```



Una librería más especializada es **np** (non-parametric).

```
library(np)

x.eval <- seq(-0.2, 0.2, length.out = 200)

h_normal_np <- npudensbw(dat = x, bwmethod = "normal-reference")

dens.ksum <- npksum(
 txdat = x,
 exdat = x.eval,
 bws = h_normal_np$bw
)$ksum / (n * h_normal_np$bw[1])


dens.ksum.df <- data.frame(x = x.eval, y = dens.ksum)
```

```
ggplot(x_df, aes(x)) +
 geom_histogram(
   aes(y = ..density..),
   binwidth = h_normal_np$bw,
   col = "black",
   fill = "white"
 ) +
 stat_function(
   fun = dnorm,
   args = list(mean = mu, sd = sigma),
   color = "red"
 ) +
 geom_line(data = dens.ksum.df, aes(x, y), color = "blue") +
 theme_minimal(base_size = 20)
```



Validación cruzada

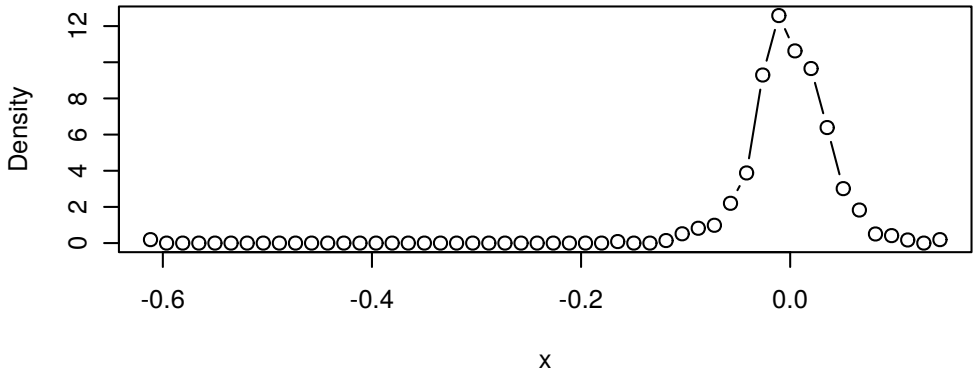
La forma que vimos en clase es la de validación cruzada por mínimos cuadrados"least-square cross validation" la cual se puede ejecutar con este comando.

```
h_cv_np_ls <- npudensbw(
 dat = x,
 bwmethod = "cv.ls",
 ckertype = "epa",
 ckerorder = 2
)

##
Multistart 1 of 1 |
```

```
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 /
Multistart 1 of 1 |
Multistart 1 of 1 |
```

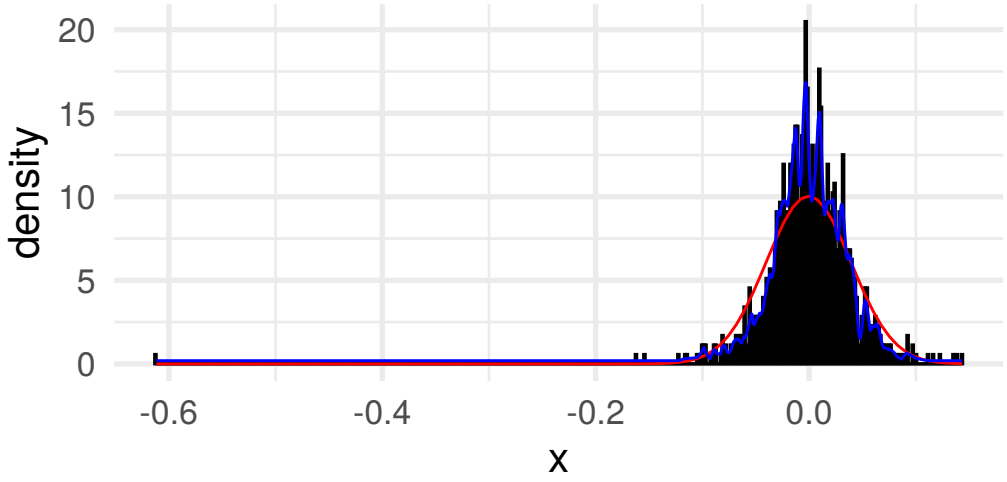
```
dens.np <- npudens(h_cv_np_ls)
```

```
plot(dens.np, type = "b")
```




```
dens.np.df <- data.frame(
 x = dens.np$eval[, 1],
 y = dens.np$dens
)
```

```
ggplot(x_df, aes(x)) +
 geom_histogram(
   aes(y = ..density..),
   binwidth = h_cv_np_ls$bw,
   col = "black",
   fill = "white"
 ) +
 stat_function(
   fun = dnorm,
   args = list(mean = mu, sd = sigma),
   color = "red"
 ) +
 geom_line(data = dens.np.df, aes(x, y), color = "blue") +
 theme_minimal(base_size = 20)
```



Temas adicionales

Reducción del sesgo Como lo mencionamos en el texto, una forma de mejorar el sesgo en la estimación es suponer que la función de densidad es más veces diferenciable.

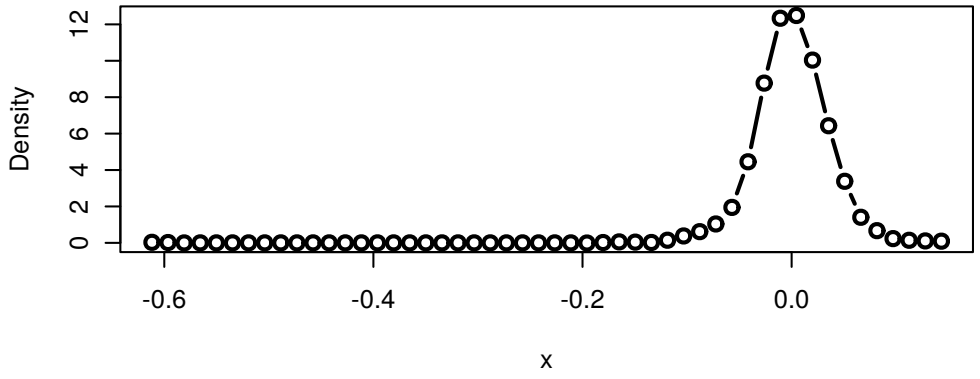
Esto se logra asumiendo que el Kernel es más veces diferenciable.

```
h_cv_np_ls <- npudensbw(
 dat = x,
 bwmethod = "cv.ls",
 ckertype = "epa",
 ckerorder = 4
)

##
```

```
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 /
Multistart 1 of 1 |
Multistart 1 of 1 |
```

```
dens.np <- npudens(h_cv_np_ls)
```

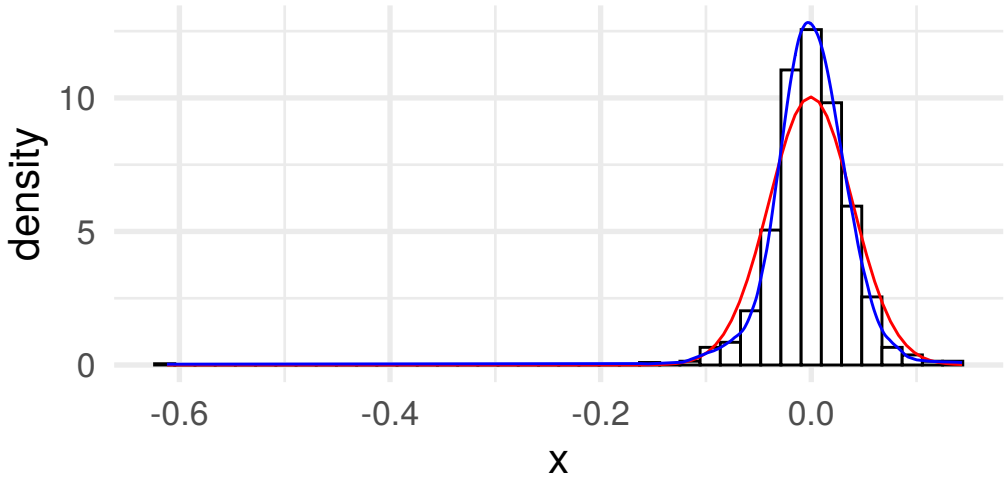
```
plot(dens.np, type = "b", lwd = 2)
```




```
dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)
```

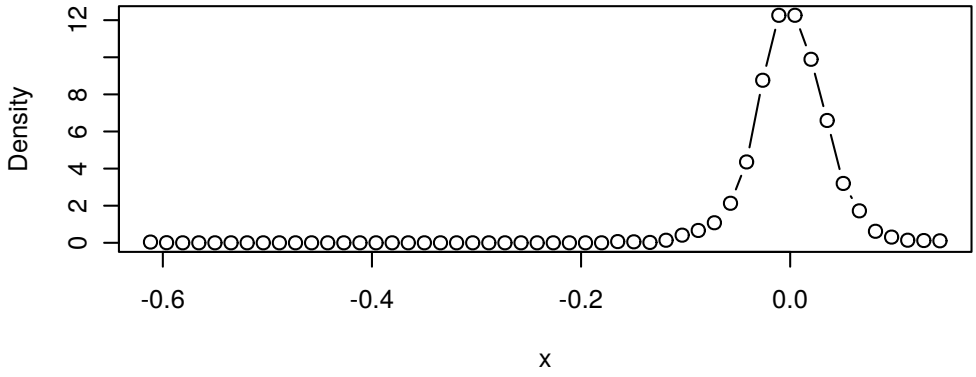
```
ggplot(x_df, aes(x)) +
```

```
geom_histogram(
  aes(y = ..density..),
  binwidth = h_cv_np_ls$bw,
  col = "black",
  fill = "white"
) +
stat_function(
  fun = dnorm,
  args = list(mean = mu, sd = sigma),
  color = "red"
) +
geom_line(data = dens.np.df, aes(x, y), color = "blue") +
theme_minimal(base_size = 20)
```



Otra forma de estimar el ancho de banda Otra forma de estimar ancho de bandas óptimos es usando máxima verosimilitud. Les dejo de tarea revisar la sección 1.1 del artículo de Hall [2] para entender su estructura.

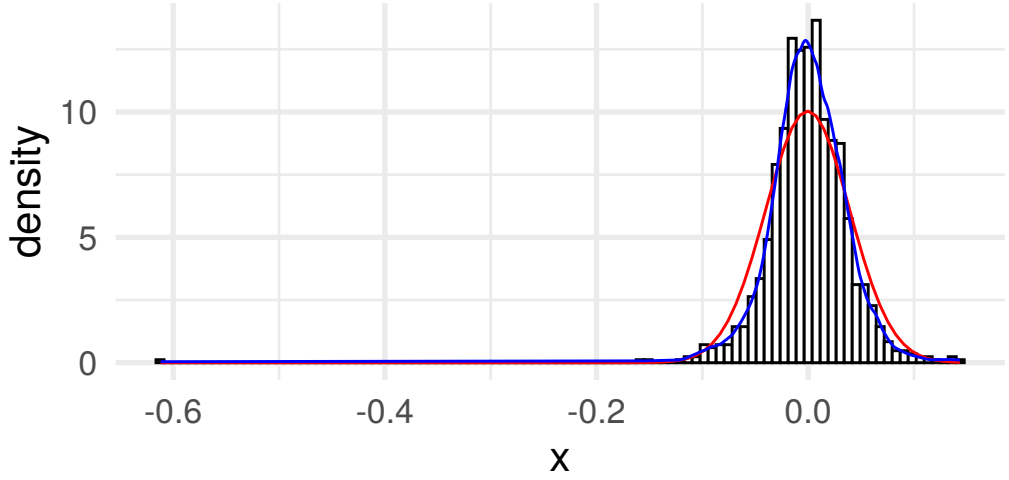
```
h_cv_np_ml <- npudensbw(
 dat = x,
 bwmethod = "cv.ml",
 ckertype = "epanechnikov"
)

##
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 /
Multistart 1 of 1 |
Multistart 1 of 1 |

dens.np <- npudens(h_cv_np_ml)

plot(dens.np, type = "b")
```




```
dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)
```

```
ggplot(x_df, aes(x)) +
 geom_histogram(
   aes(y = ..density..),
   binwidth = h_cv_np_ml$bw,
   col = "black",
   fill = "white"
 ) +
 stat_function(
   fun = dnorm,
   args = list(mean = mu, sd = sigma),
   color = "red"
 ) +
```

```
 geom_line(data = dens.np.df, aes(x, y), color = "blue") +
 theme_minimal(base_size = 20)
```

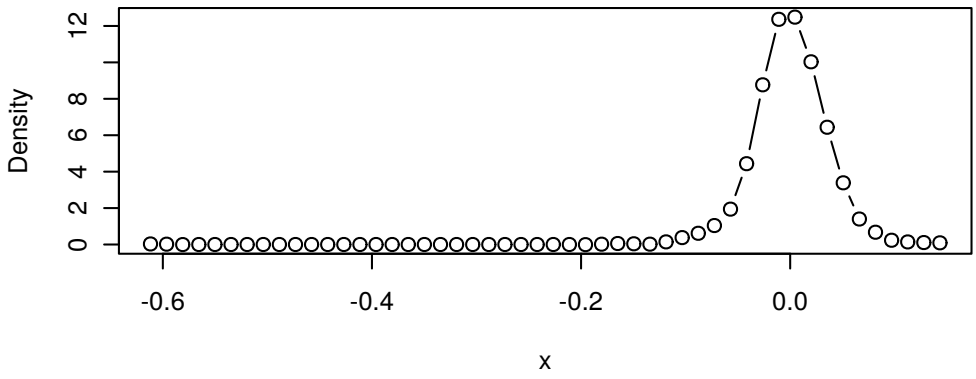



```
h_cv_np_ml <- npudensbw(
 dat = x,
 bwmethod = "cv.ml",
 ckertype = "epanechnikov",
 ckerorder = 4
)

##
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 |
Multistart 1 of 1 /
```

```
Multistart 1 of 1 |
Multistart 1 of 1 |
```

```
dens.np <- npudens(h_cv_np_ml)
```

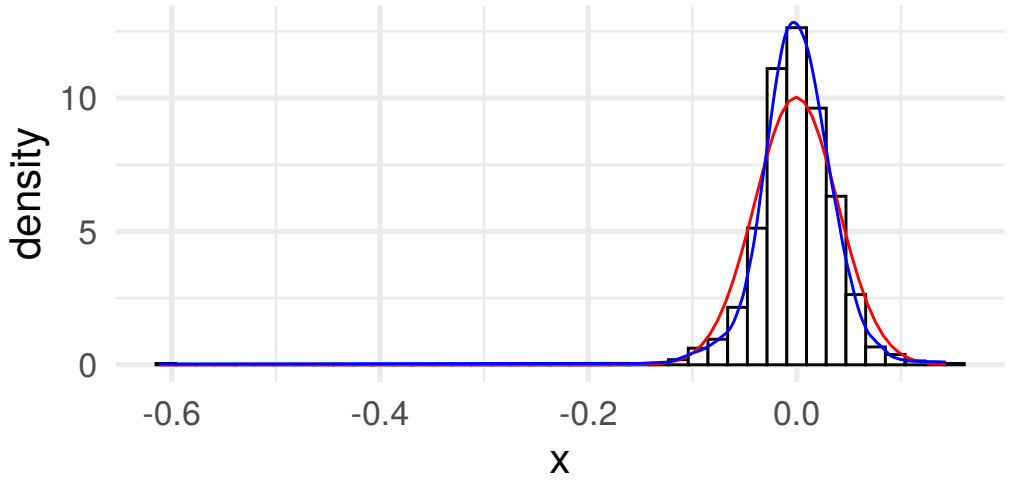
```
plot(dens.np, type = "b")
```




```
dens.np.df <- data.frame(x = dens.np$eval[, 1], y = dens.np$dens)
```

```
ggplot(x_df, aes(x)) +
 geom_histogram(
   aes(y = ..density..),
   binwidth = h_cv_np_ml$bw,
   col = "black",
```

```
    fill = "white"
  ) +
  stat_function(
    fun = dnorm,
    args = list(mean = mu, sd = sigma),
    color = "red"
  ) +
  geom_line(data = dens.np.df, aes(x, y), color = "blue") +
  theme_minimal(base_size = 20)
```




```
fani <- tibble()

for (b in seq(0.001, 0.05, length.out = 40)) {
  f <-
    npudens(
```

```
      tdat = x,
      ckertype = "epanechnikov",
      bandwidth.compute = FALSE,
      bws = b
    )
  fani <-
    fani %>% bind_rows(tibble(
      xreal = sort(x),
      x = f$eval$x,
      y = f$dens,
      bw = b
    ))
}

ggplot(data = fani) +
 geom_line(aes(x, y), color = "blue") +
 labs(title = paste0("Ancho de banda = {closest_state}")) +
 transition_states(bw) +
 view_follow() +
 theme_minimal(base_size = 20)

anim_save("manual_figure/bandwidth-animation-np.gif")
```

Tarea 1.2.24

Implementar el intervalo confianza visto en clase para estimadores de densidades por núcleos y visualizarlo de en ggplot.

Si se atreven: ¿Se podría hacer una versión animada de ese gráfico para visualizar el significado real de este el intervalo de confianza?