Guia 22 Densidad NP

Agustin Muñoz Gonzalez

8/7/2020

Preparamos el entorno.

```
rm(list=ls())
library(ggplot2)
library(tidyr)
library(gganimate)
```

Preliminares

Definamos los distintos núcleos que vimos en clase.

```
K_rect=function(t){
  1/2*ifelse(-1<= t & t<= 1,1,0)
}
K_triang=function(t){
  (1-abs(t))*ifelse(-1<= t & t<= 1,1,0)
}
K_Gauss=function(t){
  1/sqrt(2*pi)*exp(-1/2*t^2)
}
K_Epa=function(t){
  3/4*(1-t^2)*ifelse(-1<= t & t<= 1,1,0)
}
```

OJO QUE SI DEFINIS EL NUCLEO POR TU CUENTA PUEDE SER QUE HAYA QUE MULTIPLICARLO POR ALGUNA CTE PARA QUE COINCIDA CON DENSITY(...,KERNEL=...,BW=...). NOTAR ADEMAS QUE TAMBIEN EXISTE DENSITY(...,KERNEL=...,WINDOW=...). HABRIA QUE COMPARAR LOS 3 RESULTADOS! BUENO DENSITY LO QUE HACE ES USAR LA GRILLA SEQ(MIN(DATOS)-CUTxBW,MAX(DATOS)+CUTxBW,LENGTH.OUT=512) DONDE X DEFECTO DENSITY TOMA CUT=3 Y N=512.

Definimos ahora una función estimadora de f(z) que tome como parámetros un núcleo K, los datos, una ventana h y el punto z.

```
f_K_h=function(K,datos,h,z){
  mean(K((z-datos)/h))/h
}
```

Recordemos que los núcleos **deben ser densidades**, con lo cual podemos usar las funciones dnorm, dunif, dMONGO. O sea podemos usar una distribución ya conocida.

El comando para aproximar la densidad con R es density(datos, kernel='MONGO',...) donde kernel = c("gaussian", "epanechnikov", "rectangular","triangular", "biweight", "cosine", "optcosine"). Entre los datos

en ‘. . . ’ esta bw (binwidth) que es la ventana h que queremos usar.

Los datos que se muestran a continuación se hallan en el archivo buffalo.txt y corresponden a la mediciones de cantidad de nieve caída (en pulgadas) en Buffalo en los inviernos de 1910/1911 a 1972/1973.

Acomodamos los datos.

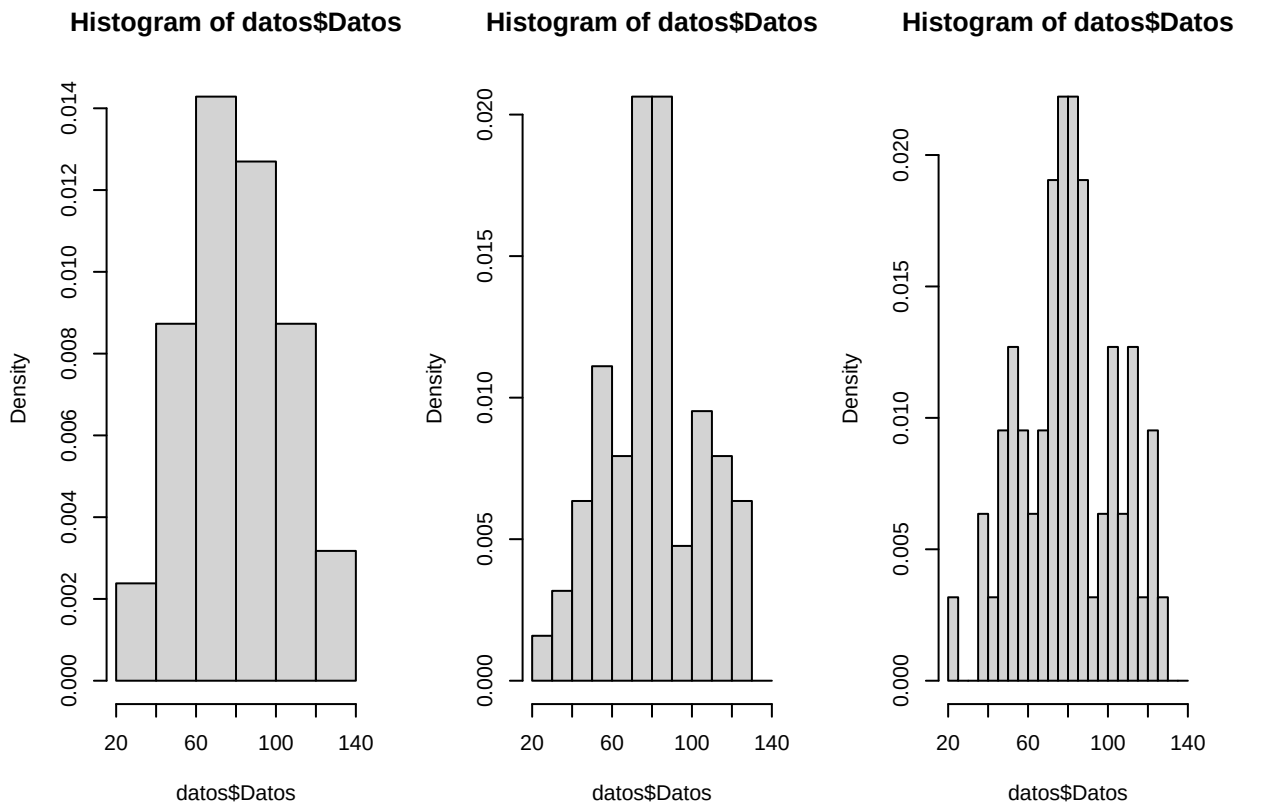
```
datos=read.csv('buffalo.csv',sep="",header=F)
# Transponemos los datos y nos quedamos con la primera col
datos=data.frame(stack(datos)[,1])
names(datos)='Datos'
# otra forma
buffalo=scan('buffalo.txt',sep = " ")
```

Resolvamos

1. Realice un histograma para estos datos utilizando los parámetros por default. Repetir eligiendo como puntos de corte las siguientes secuencias: i) de 20 a 140 con paso 10 y ii) de 20 a 140 con paso 5. Comparar los tres histogramas obtenidos. ¿Tiene algún efecto el refinamiento de los bins?

Resolución:

```
par(mfrow=c(1,3))
hist(datos$Datos,freq=F)
hist(datos$Datos,freq=F,
     breaks=seq(20,140,10))
hist(datos$Datos,freq=F,
     breaks=seq(20,140,5))
```




```
par(mfrow=c(1,1))
##############################################
p1=datos %>%
```

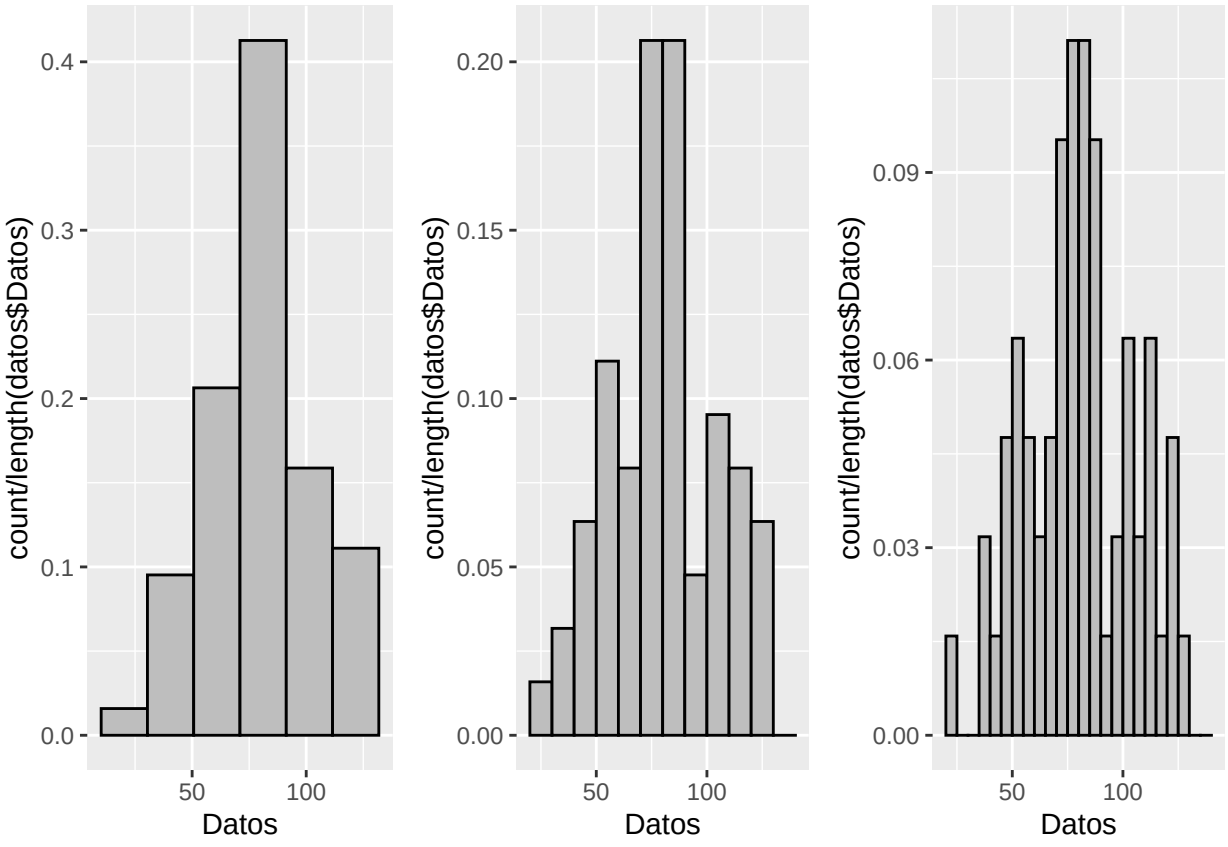
```
  ggplot()+
  geom_histogram(aes(x=Datos,y=stat(count)/length(datos$Datos)),
                 color='black', fill='grey', bins=nclass.Sturges(datos$Datos)-1)
# La cantidad de bins que usa hist por defecto es nclass.Sturges(datos$Datos)
p2=datos %>%
  ggplot()+
  geom_histogram(aes(x=Datos,y=stat(count)/length(datos$Datos)),
                 color='black', fill='grey',
                 breaks=seq(20,140,10))

p3=datos %>%
  ggplot()+
  geom_histogram(aes(x=Datos,y=stat(count)/length(datos$Datos)),
                 color='black', fill='grey',
                 breaks=seq(20,140,5))
library(Rmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: plyr
```

```
multiplot(p1,p2,p3,cols = 3)
```



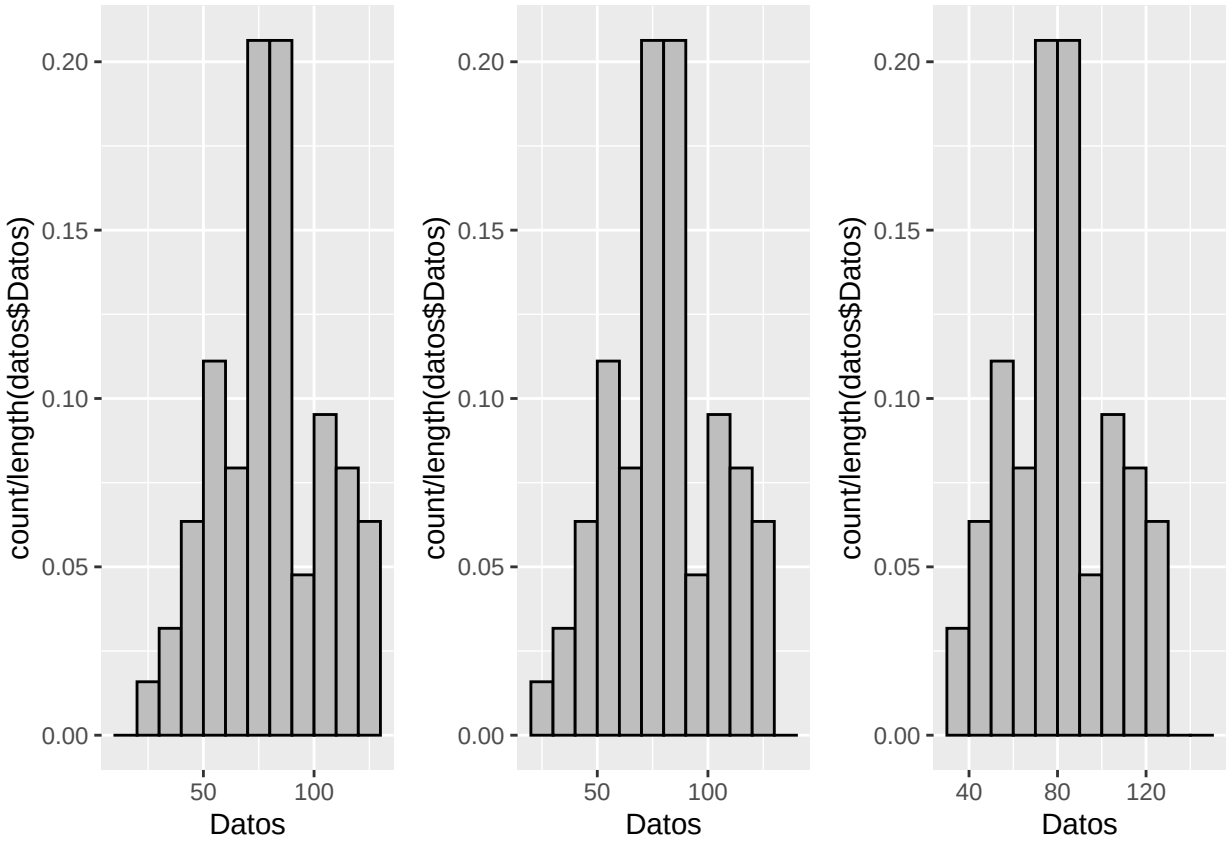
2. Realice un histograma para estas observaciones utilizando puntos de corte (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130). Repita corriendo el punto de inicio de los bins en 2 unidades 2 veces consecutivas. Compare los tres histogramas obtenidos. ¿Tiene algún efecto la elección del punto inicial en este ejemplo?

Resolución:

Los muevo 10 unidades 2 veces consecutivas porque sino no se nota.

```
p1=datos %>%
  ggplot()+
  geom_histogram(aes(x=Datos,y=stat(count)/length(datos$Datos)),
                 color='black', fill='grey',
                 breaks=c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130))
p2=datos %>%
  ggplot()+
  geom_histogram(aes(x=Datos,y=stat(count)/length(datos$Datos)),
                 color='black', fill='grey',
                breaks=c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130)+10)

p3=datos %>%
  ggplot()+
  geom_histogram(aes(x=Datos,y=stat(count)/length(datos$Datos)),
                 color='black', fill='grey',
                 breaks=c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130)+20)
library(Rmisc)
multiplot(p1,p2,p3,cols = 3)
```



3. Implemente una función densidad.est.parzen que tenga por argumento un conjunto de datos x = (x 1 , . . . , x n), una ventana h y un punto x y devuelva $\hat{f}_h(x)$, el valor de la estimación de la densidad f en el punto x, utilizando el núcleo uniforme (también llamado rectangular).

Resolución:

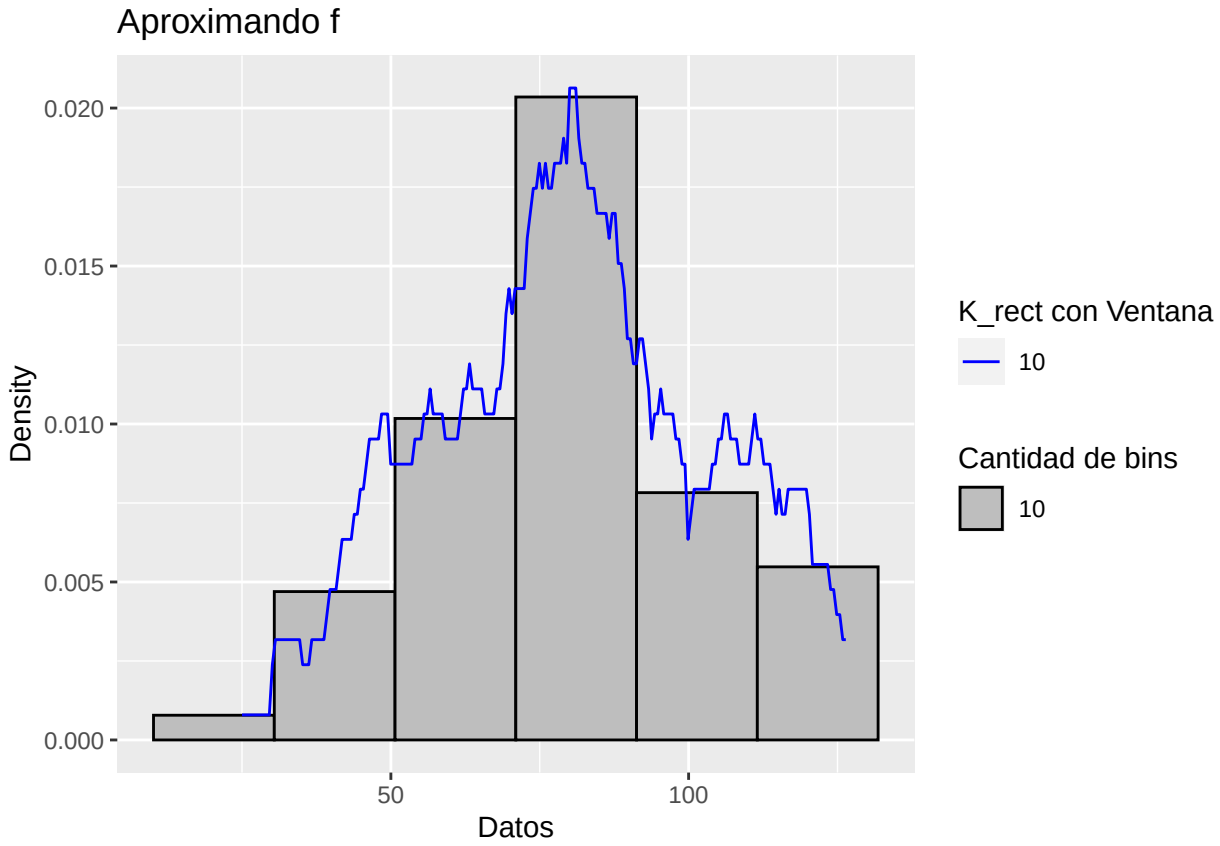
Definimos la función.

```
densidad.est.parzen=function(datos,h,z){
  f_K_h(K_rect,datos,h,z)
}
```

4. Con la función densidad.est.parzen implementada, estime la densidad f en el intervalo (25,126.4) (mínimo y máximo de las observaciones) sobre una grilla de 200 puntos equiespaciados para h = 10. Grafique el estimador $\hat{f}_h(x)$ obtenido.

Resolución:

```
grilla=seq(min(datos$Datos),max(datos$Datos),length.out = 200)
############################3
h=10
aproximaciones_h=c()
for(i in h){
  aproximaciones_h=cbind(aproximaciones_h,
                         sapply(grilla,densidad.est.parzen,datos=datos$Datos,h=i))
}
datos_plot=data.frame(cbind('Grilla'=grilla,
                            aproximaciones_h))
names(datos_plot)=c('Grilla',paste('Ventana_',h,sep = ""))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..,fill='Datos'),
                 bins=nclass.Sturges(datos$Datos)-1,col='black')+
  geom_line(aes(x=Grilla,y=Ventana_10,col='10'))+
  scale_color_manual("K_rect con Ventana",
                     breaks=c('10'),
                     values=c('blue'))+
  scale_fill_manual("Cantidad de bins",
                     breaks=c('Datos'),
                     values=c('grey'),
                    labels='10')+
ylab("Density") +
labs(title="Aproximando f")
```



Veamos si podemos hacer una función que tome como parámetros el núcleo K, los datos, la grilla y la ventana h, y devuelva la densidad aproximada.

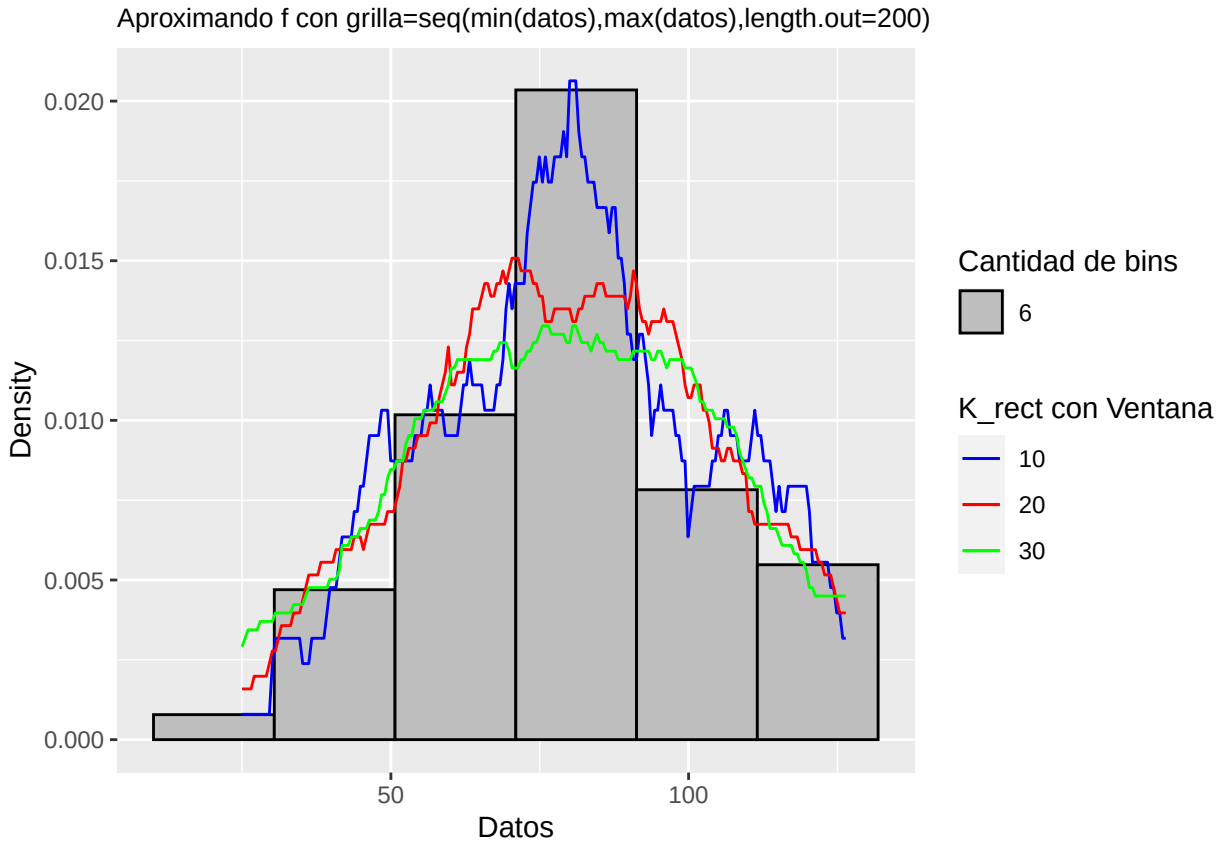
```
plot_density=function(K,datos,grilla,h){
  aproximaciones_h=c()
for(i in h){
  aproximaciones_h=cbind(aproximaciones_h,
                         sapply(grilla,f_K_h,K=K,datos=datos,h=i))
}
datos_plot=data.frame(cbind('Grilla'=grilla,
                            aproximaciones_h))
datos_plot %>%
  ggplot()+
  geom_line(aes(x=Grilla,y=V2,col='V2'))+
  scale_color_manual("K_rect con Ventana",
                     breaks=c('V2'),
                     values=c('blue'))+
ylab("Density") +
labs(title="Aproximando f")
}
```

VER COMO ITERO LOS GEOM_LINE

5. Estime la función de densidad f de la variable *pulgadas de nieve caída en invierno* a partir de los datos de Buffalo a través de la función densidad.est.parzen implementada en el item anterior usando h = 10. Realice un histograma para los datos de Buffalo y superponga la densidad estimada en los datos mediante la función densidad.est.parzen utilizando h = 10, 20 y 30. Observe como varía la rugosidad de los estimadores de f obtenidos.

Resolución:

```
grilla=seq(min(datos$Datos),max(datos$Datos),length.out = 200)
############################3
h=c(10,20,30)
aproximaciones_h=c()
for(i in h){
  aproximaciones_h=cbind(aproximaciones_h,
                         sapply(grilla,densidad.est.parzen,datos=datos$Datos,h=i))
}
datos_plot=data.frame(cbind('Grilla'=grilla,
                            aproximaciones_h))
names(datos_plot)=c('Grilla',paste('Ventana_',h,sep = ""))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..,fill='Datos'),
                 bins=nclass.Sturges(datos$Datos)-1,col='black')+
  geom_line(aes(x=Grilla,y=Ventana_10,col='10'))+
  geom_line(aes(x=Grilla,y=Ventana_20,col='20'))+
  geom_line(aes(x=Grilla,y=Ventana_30,col='30'))+
  scale_color_manual("K_rect con Ventana",
                     breaks=c('10','20','30'),
                     values=c('blue','red','green'))+
  scale_fill_manual("Cantidad de bins",
                     breaks=c('Datos'),
                     values=c('grey'),
                    labels='6')+
ylab("Density") +
labs(title="Aproximando f con grilla=seq(min(datos),max(datos),length.out=200)")+
  theme(plot.title = element_text(size=10))
```



CON ESA GRILLA LAS CURVAS QUEDAN MEDIO CORTADAS, PENSAR QUE GRILLA TOMAR!

6. La función de R **density** computa un estimador de la densidad a partir de un conjunto de datos x = (x 1 , . . . , x n) y la evalúa en un conjunto de puntos intermedios. Mediante la función de R **density** estime la función de densidad f de la variable pulgadas de nieve caída en invierno a partir de los datos de Buffalo y utilizando el núcleo normal, el rectangular y el de Epanechnikov con ventana h = 5. Realice un gráfico en el que superpone las tres estimaciones de f y compare los resultados.

Resolución:

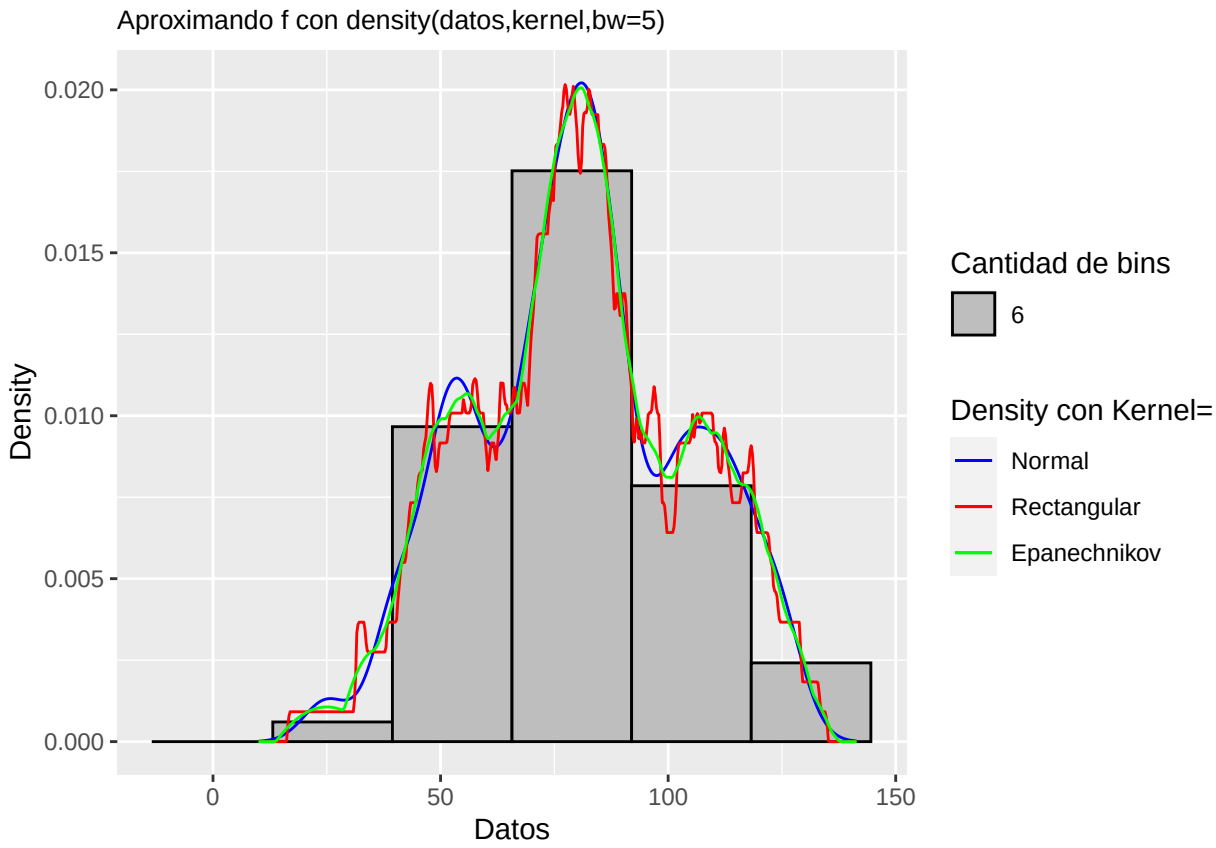
Definimos las densidades aproximadas por los 3 núcleos. density devuelve varias cosas pero las que nos importan son density(...)$x que son los n puntos de la grilla y density(...)$y que son n los valores de las densidades estimadas, donde por default n=512 (es uno de los inputs de density(...)).

Además voy a usar la función f_K_h que definí en los preliminares para compararlas (en teoría deberían dar lo mismo).

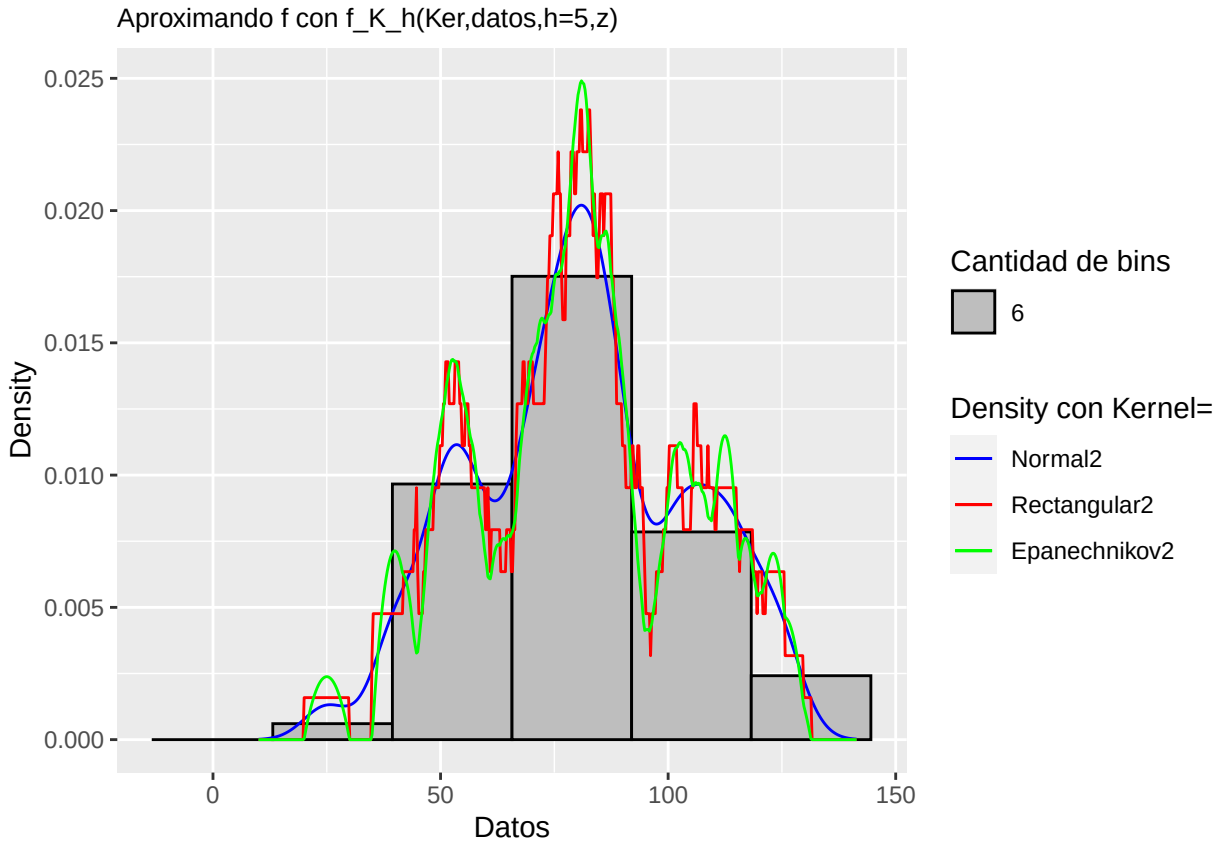
```
f_gauss=density(datos$Datos,kernel='gaussian',bw=5)
f_rect=density(datos$Datos,kernel='rectangular',bw=5)
f_epa=density(datos$Datos,kernel='epanechnikov',bw=5)
f_gauss2=function(z){
  f_K_h(K_Gauss,datos$Datos,h=5,z)
}
f_rect2=function(z){
  f_K_h(K_rect,datos$Datos,h=5,z)
}
f_epa2=function(z){
  f_K_h(K_Epa,datos$Datos,h=5,z)
}
```

Ploteamos las density por un lado y las otras por el otro porque sino es un choclo de colores.

```
# Defino la grilla como la categoría x de cualquiera de las anteriores
# (las 3 tienen la misma grilla
# seq(min(datos)-bw*cut,max(datos)+bw*cut,length.out=n),
# donde por defecto cut=3, n=512 y elegimos bw=5).
grilla=f_gauss$x
datos_plot=data.frame(cbind('Grilla'=grilla,
                            'Normal'=f_gauss$y,
                            'Rectangular'=f_rect$y,
                            'Epanechnikov'=f_epa$y))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..,fill='Datos'),
                 bins=nclass.Sturges(datos$Datos)-1,col='black')+
  geom_line(aes(x=Grilla,y=Normal,col='Normal'))+
  geom_line(aes(x=Grilla,y=Rectangular,col='Rectangular'))+
  geom_line(aes(x=Grilla,y=Epanechnikov,col='Epanechnikov'))+
  scale_color_manual("Density con Kernel=",
                     breaks=c('Normal','Rectangular','Epanechnikov'),
                     values=c('blue','red','green'))+
  scale_fill_manual("Cantidad de bins",
                     breaks=c('Datos'),
                     values=c('grey'),
                    labels='6')+
ylab("Density") +
labs(title="Aproximando f con density(datos,kernel,bw=5)")+
  theme(plot.title = element_text(size=10))
```




```
#####################################################################
datos_plot2=data.frame(cbind('Grilla'=grilla,
                             'Normal2'=sapply(grilla,f_gauss2),
                            'Rectangular2'=sapply(grilla,f_rect2),
                            'Epanechnikov2'=sapply(grilla,f_epa2)))
datos_plot2 %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..,fill='Datos'),
                 bins=nclass.Sturges(datos$Datos)-1,col='black')+
  geom_line(aes(x=Grilla,y=Normal2,col='Normal2'))+
  geom_line(aes(x=Grilla,y=Rectangular2,col='Rectangular2'))+
  geom_line(aes(x=Grilla,y=Epanechnikov2,col='Epanechnikov2'))+
  scale_color_manual("Density con Kernel=",
                     breaks=c('Normal2','Rectangular2','Epanechnikov2'),
                     values=c('blue','red','green'))+
  scale_fill_manual("Cantidad de bins",
                     breaks=c('Datos'),
                     values=c('grey'),
                    labels='6')+
ylab("Density") +
labs(title="Aproximando f con f_K_h(Ker,datos,h=5,z)")+
  theme(plot.title = element_text(size=10))
```




```
################################################
```

Se ve que Normal y Normal2 son iguales, pero Rectangular2 y Epa2 alcanzan valores mas altos. Debe ser que density hace alguna cosa mas.

7. Para los datos de Buffalo halle la ventana óptima h_S por la regla de Silverman para el núcleo normal, calcule la estimación de f basada en el núcleo normal con dicha ventana y grafique la estimación de la densidad obtenida.

Resolución:

Notemos que hasta el momento la ventana h la elegimos 'a ojo' jugando con distintos valores. Recordemos que el sesgo $(E(\hat{\theta}) - \theta)$ y la precisión (varianza) de nuestro estimador

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right),$$

son los siguientes:

$$Sesgo[\hat{f}(x)] \approx \frac{h^2}{2} C_1(K) f''(x), \quad \mathbb{V}(\hat{f}(x)) \approx \frac{1}{nh} C_2(K) f(x),$$

donde $C_1(K), C_2(K)$ son constantes que dependen del núcleo.

Entonces el error cuadrático medio EMC es

$$EMC[\hat{f}(x)] = Sesgo[\hat{f}(x)]^2 + \mathbb{V}(\hat{f}(x)) \approx \frac{h^4}{4} C_1^2(K)(f''(x))^2 + \frac{1}{nh} C_2(K) f(x).$$

Ahora bien, como estamos aproximando f en cada x nos interesa saber el 'error total' que obtenemos con nuestro estimador. Se define entonces el **error cuadrático medio integrado** (de $\hat{f}_h$), EMCI, como

$$\begin{aligned} EMCI[\hat{f}_h] &= \int ECM[\hat{f}_h(x)]dx \\ &\approx \frac{h^4}{4} C_1^2(K) \int (f''(x))^2 dx + \frac{1}{nh} C_2(K) \\ &= \frac{h^4}{4} C_1^2(K) ||f''||_2^2 \acute{1} + \frac{1}{nh} ||K||_2^2 \end{aligned}$$

Derivando respecto de h e igualando a 0 obtenemos el mínimo de ECMI en

$$h = \left\{ \frac{||K||_2^2}{C_1^2(K)||f''||_2^2} \right\}^{\frac{1}{5}} \approx C_K n^{-\frac{1}{5}}.$$

Pero Silverman propone reemplazar $||f''||_2^2$ por su valor cuando f es normal

$$||f''||_2^2 = \sigma^{-5} \frac{3}{8\sqrt{\pi}} \approx 0.212\sigma^{-5},$$

y a σ por su estimador $\widehat{\sigma} = S$.

Tenemos entonces que la h óptima de Silverman es

$$h_S = \left(\frac{4\widehat{\sigma}^5}{3n} \right)^{\frac{1}{5}} \approx 1.06 \widehat{\sigma} n^{-\frac{1}{5}}.$$

Como $\widehat{\sigma}$ lo podemos aproximar por sd(datos) o IQR(datos) (distancia intercuantil), elegimos el menor de ellos

$$h_S = 1.06 \min\left(S, \frac{IQR}{1.349}\right) n^{-\frac{1}{5}}.$$

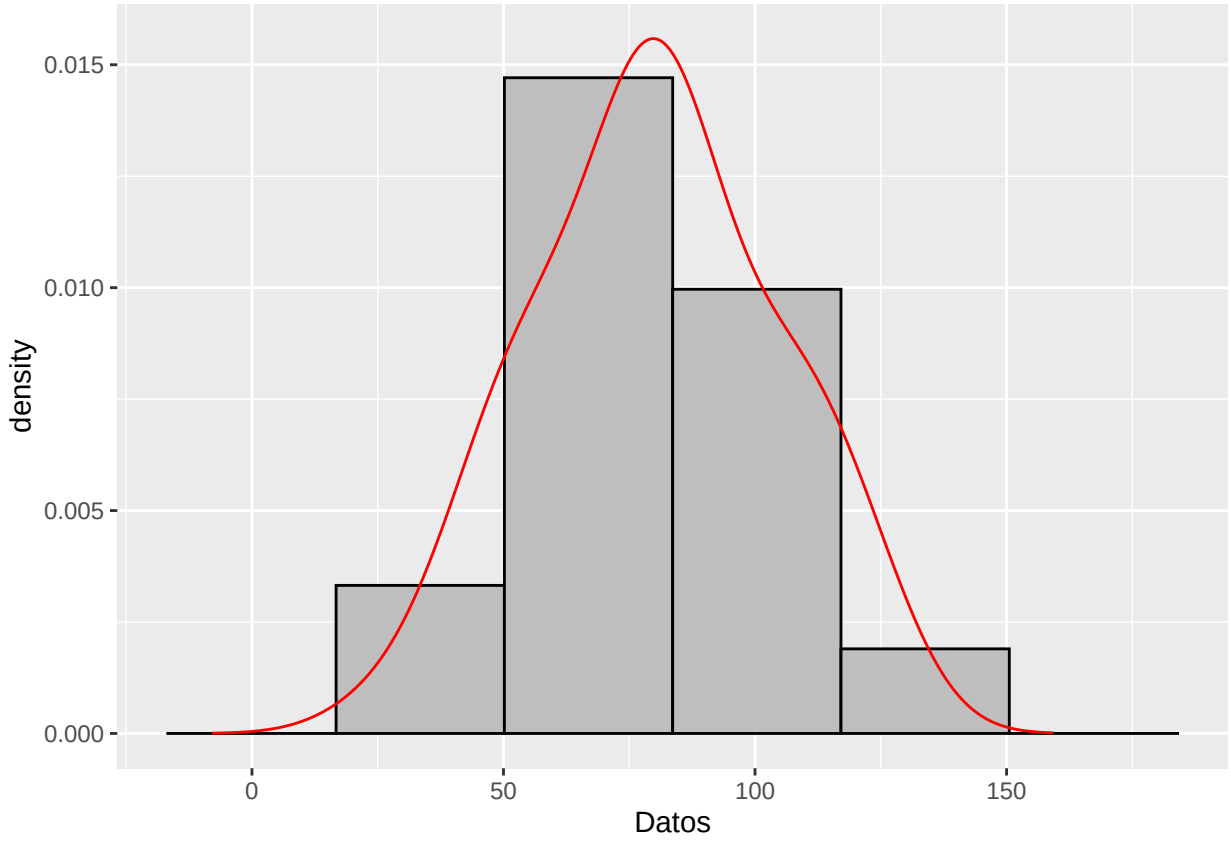
Calculemos entonces dicha ventana.

```r
h_S=1.06*min(sd(datos$Datos),IQR(datos$Datos)/1.349)/(length(datos$Datos)^(1/5))
h_S
```

```
## [1] 10.97865
```

Calculemos la estimación de f para ese ventana y para el núcleo normal, y plotiemos.

```r
f_gauss_h_S=density(datos$Datos,kernel='gaussian',bw=h_S)
#########################################################
grilla=f_gauss_h_S$x
datos_plot=data.frame(cbind('Grilla'=grilla,
                            'Normal_h_S'=f_gauss_h_S$y))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..),
                 bins=nclass.Sturges(datos$Datos)-1,
                 fill='grey',col='black')+
  geom_line(aes(x=Grilla,y=Normal_h_S),col='red')
```



8. Se puede realizar la estimación de la densidad f a partir de un conjunto de datos a través de la función **density** en un punto deseado x (f.hat.x) de la siguiente forma:

f.hat.x<- density(datos, from=x, to=x, n=1)

También mediante la función **approxfun** de R se puede aproximar la densidad estimada por la función de R **density** en el punto deseado x a partir de un conjunto de datos de la siguiente manera:

df <- approxfun(density(datos))

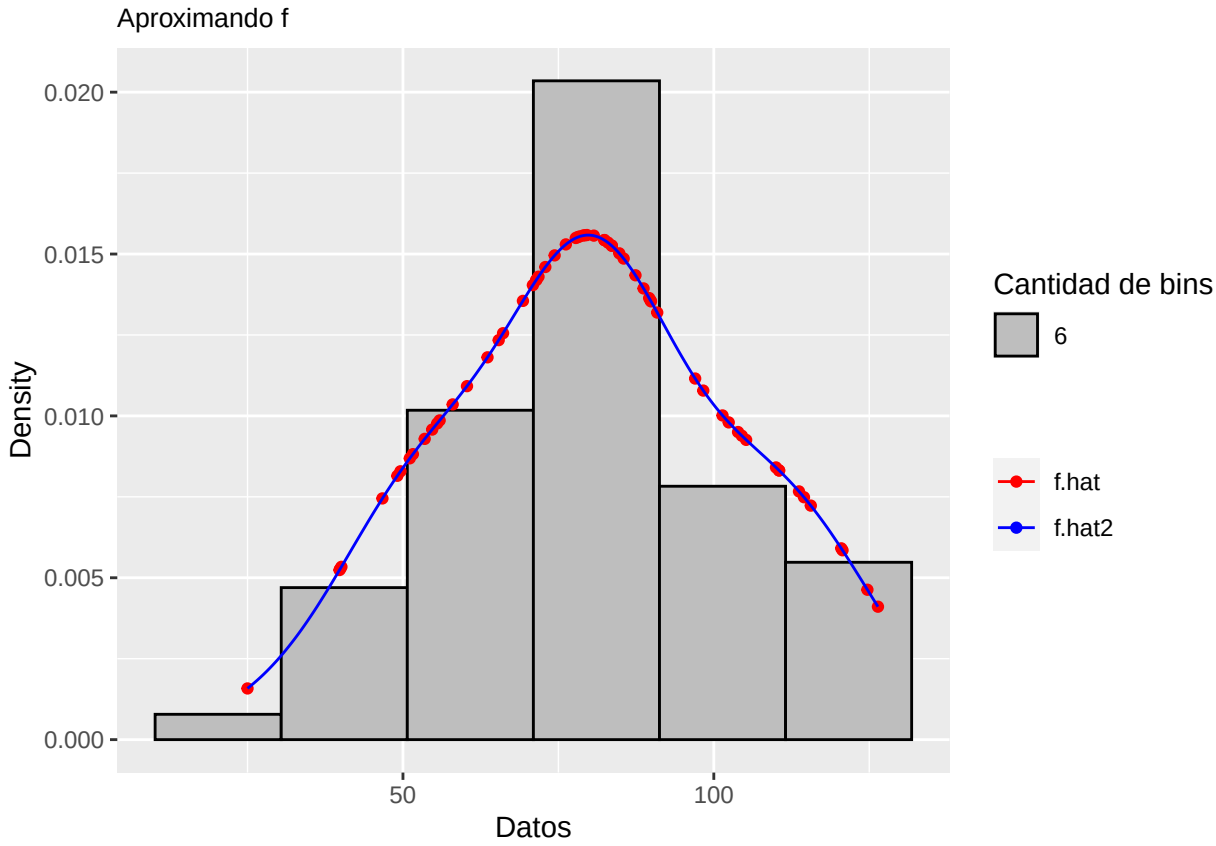
f.hat.x<-df(x)

Calcule la estimación de la densidad usando **density** a partir de los datos de Buffalo usando la ventana de Silverman y con esta estimación obtenga estimaciones de la densidad en los valores observados de nieve caída en Buffalo. Grafique la densidad estimada y superponga en el mismo plot los puntos correspondientes a las observaciones y el valor estimado de la densidad correspondiente en rojo.

Resolución:

Definimos las fcs y ploteamos lo que entendí que pide el ejcio.

```
f.hat=function(x){
  density(datos$Datos, from=x, to=x, n=1,bw=h_S)$y
}
########################################
df <- approxfun(density(datos$Datos, kernel='gaussian',bw=h_S))
f.hat2=function(x){
  df(x)
}
########################################
grilla=seq(min(datos$Datos),max(datos$Datos),length.out = 200)
datos_plot=data.frame(cbind('Grilla'=grilla,
                            'f.hat2'=sapply(grilla,f.hat2)))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..,fill='Datos'),
                 bins=nclass.Sturges(datos$Datos)-1,col='black')+
  # Ploteo la density puntual en los datos como puntos rojos
  geom_point(data=data.frame('x'=datos$Datos,
                             'y'=sapply(datos$Datos,f.hat)),
             aes(x=x,y=y,col='f.hat'))+
  geom_line(aes(x=Grilla,y=f.hat2,col='f.hat2'))+
  scale_color_manual("",
                     breaks=c('f.hat','f.hat2'),
                     values=c('red','blue'))+
  scale_fill_manual("Cantidad de bins",
                     breaks=c('Datos'),
                     values=c('grey'),
                    labels='6')+
ylab("Density") +
labs(title="Aproximando f")+
  theme(plot.title = element_text(size=10))
```


Aproximando f
0.020
0.015
0.010
0.005
0.000
Density
50
100
Datos
Cantidad de bins
6
f.hat
f.hat2

9. Llamemos $\hat{f}_h^{(-i)}$ a la densidad estimada sin utilizar al punto x i y usando la ventana h. Calcule las estimaciones $\hat{f}_h^{(-i)}$ para i = 17, 20, 51 de la densidad f usando la función **density** a partir de los datos de Buffalo usando el núcleo de Epanechnikov y la ventana h = 5. Grafique en un mismo plot la estimación de f basada en todos los datos usando el núcleo de Epanechnikov y la ventana h = 5 y las estimaciones $\hat{f}_h^{(-i)}$ para i = 17, 20, 51. Compare las estimaciones obtenidas.

Resolución:

Notemos que otra forma posible de elegir la ventana h sería por el método de máxima verosimilitud para nuestros estimadores $\hat{f}$, el problema es que el h es degenerado, $h_{MV} = 0$. La alternativa será entonces maximizar usando la pseudo-verosimilitud computada sacando de los datos una observación a la vez (leave-one-out-Cross-Validation).

Dados los datos $x_1, \ldots, x_n$ nos quedamos con

$$h^*_{MV} = argmax \left\{ \frac{1}{n} \sum_{i=1}^{n} \log(\hat{f}_h^{(-i)}(x_i)) \right\},$$

donde $\hat{f}_h^{(-i)}(x_i)$ es la densidad estimada en el punto x_i sin utilizar el punto x_i

$$\hat{f}_h^{(-i)}(x_i) = \frac{1}{(n-1)h} \sum_{j \neq i} K\left(\frac{x_i - x_j}{h}\right).$$

Surge entonces la pregunta, ¿Qué h usamos para la pseudo-verosimilitud? En general se busca el máximo sobre una grilla $h_1, \ldots, h_q$ (por ejemplo alrededor del h_{Sil}) y eventualmente se refina.

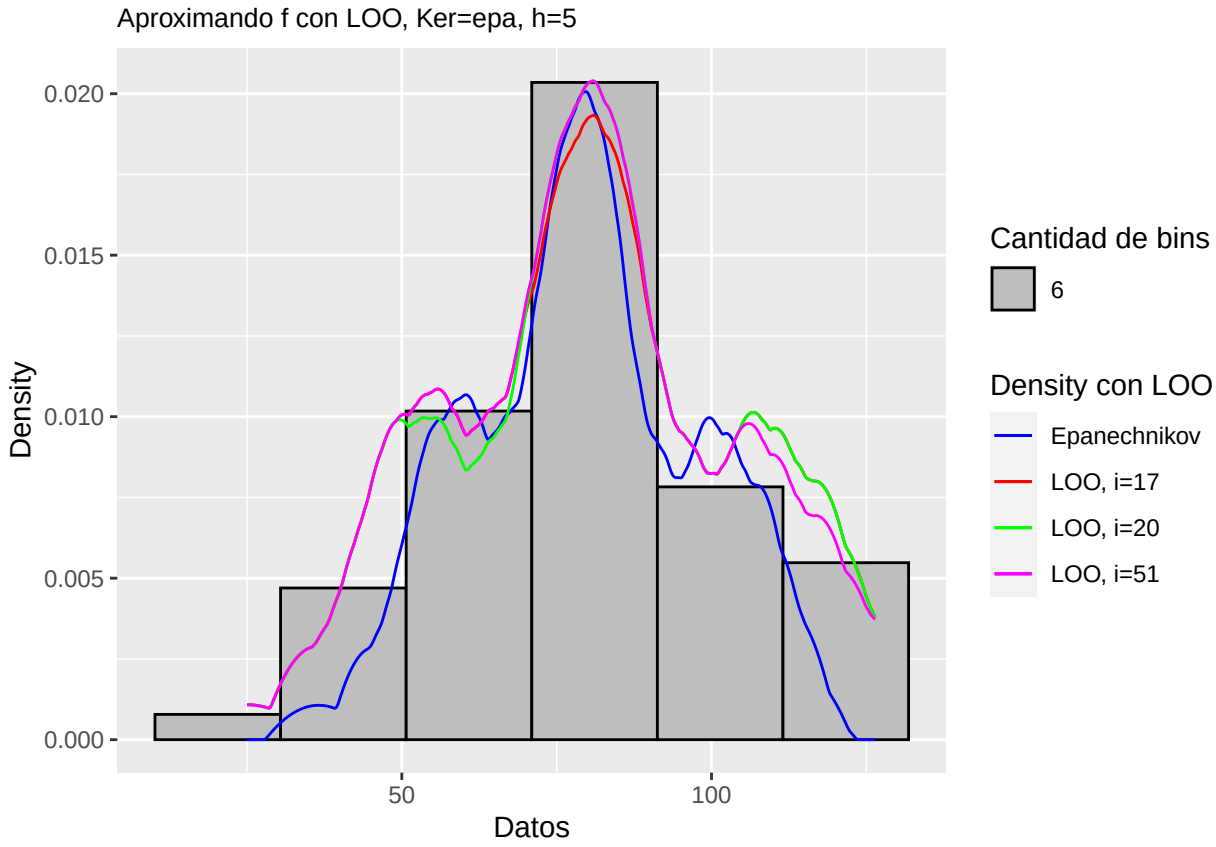
Resolvamos entonces lo que nos pide el ejercicio. Defino la función f_LOO (LOO=Leave-One-Out) que tome como parámetros la coordenada i a dejar afuera, la ventana h y el elemento x al cual le queremos estimar $f(x)$.

```
f_LOO_epa=function(x,i,h){
  density(datos$Datos[-i], kernel = 'epanechnikov',
          from=x, to=x, n=1,bw=h)$y
}
```

Ploteamos.

```
grilla=seq(min(datos$Datos),max(datos$Datos),length.out = 512)
datos_plot=data.frame(cbind('Grilla'=grilla,
                            'Epanechnikov'=f_epa$y,
                            'f_LOO_epa_17'=sapply(grilla,f_LOO_epa,i=17,h=5),
                            'f_LOO_epa_20'=sapply(grilla,f_LOO_epa,i=20,h=5),
                            'f_LOO_epa_51'=sapply(grilla,f_LOO_epa,i=51,h=5)))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..,fill='Datos'),
                 bins=nclass.Sturges(datos$Datos)-1,col='black')+
  geom_line(aes(x=Grilla,y=Epanechnikov,col='Epanechnikov'))+
  geom_line(aes(x=Grilla,y=f_LOO_epa_17,col='LOO, i=17'))+
  geom_line(aes(x=Grilla,y=f_LOO_epa_20,col='LOO, i=20'))+
  geom_line(aes(x=Grilla,y=f_LOO_epa_51,col='LOO, i=51'))+
  scale_color_manual("Density con LOO",
                     breaks=c('Epanechnikov','LOO, i=17','LOO, i=20','LOO, i=51'),
                     values=c('blue','red','green','magenta'))+
  scale_fill_manual("Cantidad de bins",
                     breaks=c('Datos'),
```

```
                     values=c('grey'),
                    labels='6')+
ylab("Density") +
labs(title="Aproximando f con LOO, Ker=epa, h=5")+
  theme(plot.title = element_text(size=10))
```



10. Halle la ventana de convalidación cruzada por máxima verosimilitud para la estimación de la densidad basada en el núcleo de Epanechnikov utilizando los datos de Buffalo mediante la función de R **density**. Utilice una grilla para la ventana h que varíe entre 5 y 25 con paso 0.25. Realice un gráfico de h versus la pseudo–log–verosimilitud calculada para cada h.

Resolución:

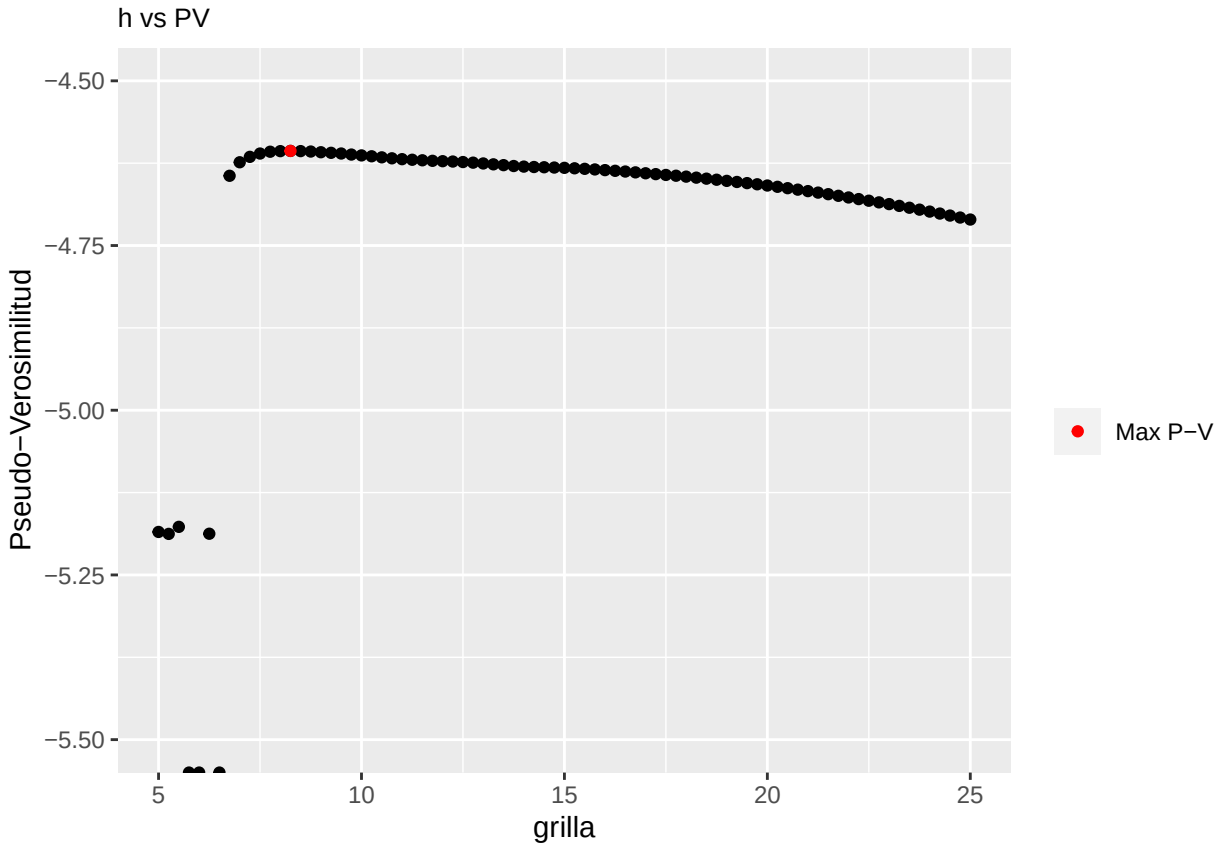
Dada una ventana alrededor de h_{Sil} nos quedaremos con el valor h de esa ventana que hace que $\frac{1}{n}\sum_{i=1}^{n}\log(\hat{f}_h^{(-i)}(x_i))$ sea máximo.

Defino una función que toma un vector de ventanas y devuelve la ventana h que maximiza la pseudo-verosimilitud. Además defino una función que toma una ventana h y devuelve el valor de la pseudo-verosimilitud (i.e. $\frac{1}{n}\sum_{i=1}^{n}\log(\hat{f}_h^{(-i)}(x_i))$).

```
PV_LOO_epa=function(h){
  vect_de_logs=c()
  for(i in 1:length(datos$Datos)){
      x_i=datos$Datos[i]
      vect_de_logs=c(vect_de_logs,log(f_LOO_epa(x_i,i,h=h)))
    }
    mean(vect_de_logs)
}
#########################################################
h_MPV_LOO_epa=function(valores_h){
  # Armamos el vector a maximizar
  vect_to_max=sapply(valores_h,PV_LOO_epa)
  valores_h[which.max(vect_to_max)]
}
```

Ploteamos.

```
grilla=seq(5,25,0.25)
datos_plot=data.frame('Grilla'=grilla,
                      'Pseudo_Verosimilitudes'=sapply(grilla,PV_LOO_epa))
datos_plot %>%
  ggplot()+
  geom_point(aes(x=grilla,y=Pseudo_Verosimilitudes))+
  geom_point(data=data.frame('x'=h_MPV_LOO_epa(grilla),
                             'y'=PV_LOO_epa(h_MPV_LOO_epa(grilla))),
             aes(x=x,y=y,col='Max P-V'))+
  scale_color_manual("",
                     breaks=c('Max P-V'),
                     values=c('red'))+
  ylab("Pseudo-Verosimilitud") +
  ylim(-5.5,-4.5)+
  labs(title="h vs PV")+
  theme(plot.title = element_text(size=10))
```



NOTAR QUE ALGUNOS VALORES VALEN $-\infty$ PORQUE NO TENES DATOS EN ESAS PARTES Y ESTAS HACEIDNO LOG(0) QUE DA $-\infty$. EN GENERAL ESTO PASA PORQUE VENTANAS CHICAS LLEVAN A COSAS DEGENERADAS. CREO QUE ES PORQUE ESTAS EVALUANDO EL NUCLEO EN UN LUGAR FUERA DE SU SOPORTE. O SEA NO SON VENTANAS ADECUADAS! ESTAMOS HACIEDNO ALGO NO PARAMETRICO EN UN CONTEXTO DE POCOS DATOS, METODOS NO PARAMETRICOS ES PARA CUANDO TENES MUCHOS DATOS!!

11. Grafique en un mismo plot las estimaciones de la densidad obtenidas con el núcleo de Epanechnikov usando la ventana óptima obtenida en el item anterior y usando la ventana h_S hallada para el núcleo normal. Compare los resultados.

Resolución:

```
h_MPV=h_MPV_LOO_epa(seq(5,25,0.25))
f_epa_h_MPV=density(datos$Datos,kernel='epanechnikov',bw=h_MPV)
grilla=f_gauss_h_S$x
datos_plot=data.frame(cbind('Grilla'=grilla,
                            'Normal_h_S'=f_gauss_h_S$y,
                            'Epanechnikov_h_MPV'=f_epa_h_MPV$y))
datos_plot %>%
  ggplot()+
  geom_histogram(data=datos,aes(x=Datos,y=..density..),
                 bins=nclass.Sturges(datos$Datos)-1,
                 col='black', fill='grey')+
  geom_line(aes(x=Grilla,y=Normal_h_S,col='Normal_h_S'))+
  geom_line(aes(x=Grilla,y=Epanechnikov_h_MPV,col='Epanechnikov_h_MPV'))+
  scale_color_manual("Density con LOO",
                     breaks=c('Normal_h_S','Epanechnikov_h_MPV'),
                     values=c('blue','red'))+
ylab("Density") +
labs(title="Aproximando f con Epanechnikov_h_MPV y Normal_h_S")+
  theme(plot.title = element_text(size=10))
```