# Guia 24

Agustin Muñoz Gonzalez

15/7/2020

## Preparamos el entorno.

```
rm(list=ls())
library(ggplot2)
library(tidyr)
library(gganimate)
datos=read.csv('lidar.csv',header=TRUE)[,1:2]
names(datos)=c('range','logratio')
# armo los training set y test set tomando 80-20
indices=sample(length(datos$range),size=length(datos$range)/20)
test_set=datos[indices,]
training_set=datos[-indices,]
```

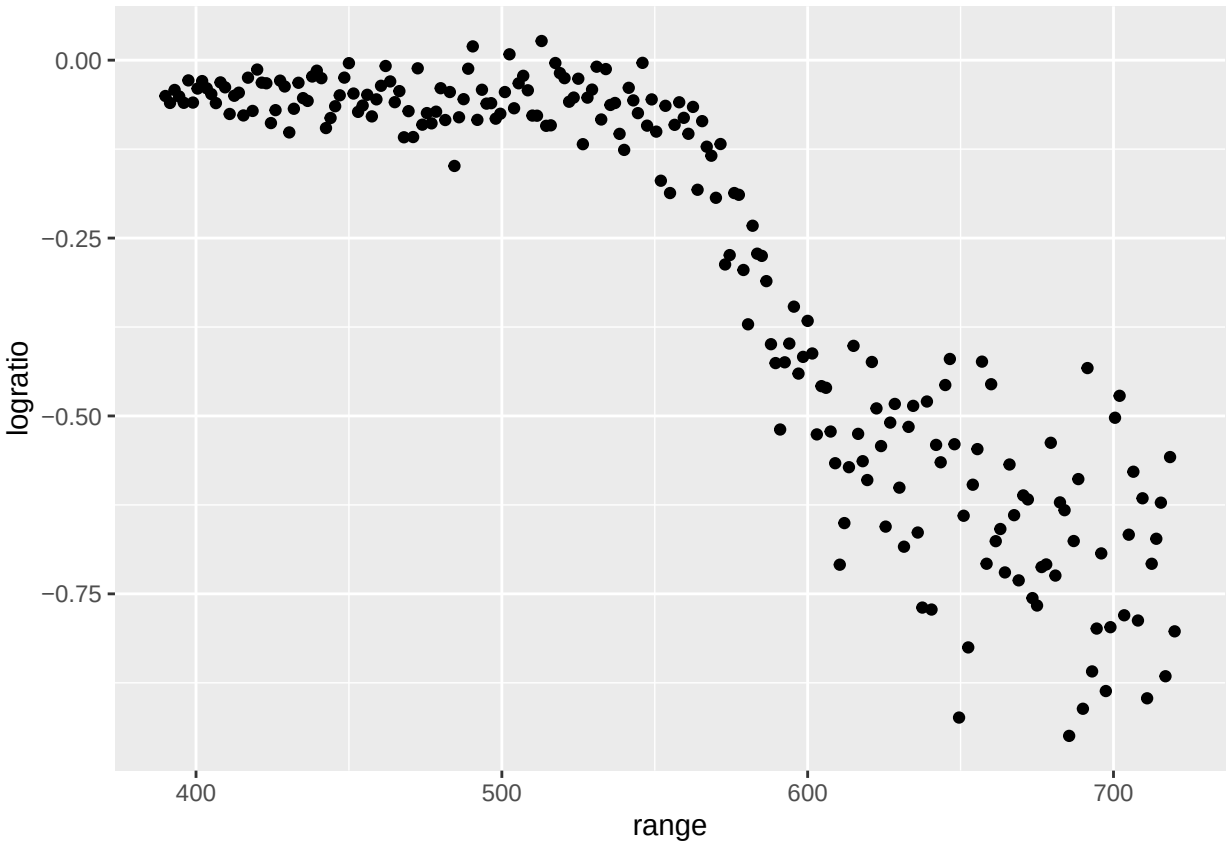
**Ejercicio I.** La técnica conocida como LIDAR (light detection and ranging) usa la reflexión de luz de láser emitida para detectar compuestos químicos en la atmósfera. Esta técnica ha probado ser una herramienta muy eficiente para el monitoreo de la distribución de diversos elementos polulantes en la atmósfera (Sigrist, 1994). En el archivo lidar.txt se encuentran datos medidos con a la técnica LIDAR. La variable **range** es la distancia recorrida antes de que la luz sea reflejada de regreso hacia su fuente. La variable **logratio** es el logaritmo del cociente de la luz recibida de dos fuentes de luz láser de distinta frecuencia.

1. A partir de los datos de lidar.txt realice un diagrama de dispersión o scatter plot de **range** (eje x) vs. **logratio** (eje y). Describa la relación entre ambas variables.

Resolución:

Realizamos el plot.

```
datos %>%
  ggplot()+
  geom_point(aes(x=range,y=logratio))
```




Se puede observar que en el rango 400~500 el logratio se concentra en el intervalo [-0.25,0], mientras que para valores del rango mayores a 500 se empieza a ver más dispersión en el logratio.

Dijo Mariela que: la dispersion en la respuesta va cambiando con los valores de la predictora. es un ejemplo dondo no hay homoscedasticidad= varianza constante .

2. La función de R **ksmooth** computa el estimador de Nadaraya-Watson a partir de un conjunto de datos (x 1 , y 1 ), . . . , (x n , y n ) y lo evalúa en un conjunto de puntos intermedios. Mediante la función de R **ksmooth** estime la función de regresión r que relaciona a las variables **range** y **logratio**, tomando como variable explicativa a **range**, a partir de los datos dados usando el núcleo normal con ventana h = 5. Con la función de regresión estimada, obtenga estimaciones de la función de regresión en los valores observados de **range**. Grafique la función de regresión estimada. Repita para valores de la ventana h = 10, 30, 50 y superponga en el mismo plot los puntos correspondientes a las observaciones y el valor estimado de la función de regresión obtenida para h = 5, 10, 30, 50. Compare los resultados obtenidos con las 4 ventanas.

Resolución:

## Un poco de contexto

Queremos estimar $P(Y = y|X = x)$. En la práctica 2 vimos dos métodos, el de k vecinos mas cercanos (k-nn) y el de promedio movil de ventana h. Este segundo método lo que hace es estima a $P(Y = y|X = x)$ por la fracción de puntos en $(x - h, x + h)$ cuya etiqueta es igual a $y$:

$$\begin{aligned}\widehat{P}(Y = y|X = x) &= \frac{\sum_{i=1}^{n} Y_i \mathcal{I}_{[x-h,x+h]}(X_i)}{\sum_{i=1}^{n} \mathcal{I}_{[x-h,x+h]}(X_i)} \\ &= \frac{\sum_{i=1}^{n} Y_i \mathcal{I}_{[-1,1]}(\frac{x-X_i}{h})}{\sum_{i=1}^{n} \mathcal{I}_{[-1,1]}(\frac{x-X_i}{h})}.\end{aligned}$$

Si escribimos $K(t) = \mathcal{I}_{[-1,1]}(t)$ es

$$\begin{aligned}\widehat{P}(Y = y|X = x) &= \frac{\sum_{i=1}^n Y_i K(\frac{x-X_i}{h})}{\sum_{i=1}^n K(\frac{x-X_i}{h})} \\ &= \sum_{i=1}^n Y_i \frac{K(\frac{x-X_i}{h})}{\sum_{i=1}^n K(\frac{x-X_i}{h})} \\ &= \sum_{i=1}^n Y_i W_i(x),\end{aligned}$$

donde $W_i(x)$ es un peso que pondera de acuerdo a la cercanía a $x$ (notar que, para un K genérico, a puntos mas cercanos le corresponden valores más grandes de $W_i$).

Supongamos ahora que Y es continua y está relacionada con X a través de una función r de la siguiente forma

$$Y = r(X) + \epsilon,$$

donde $\mathbb{E}(\epsilon) = 0$ y $r$ se llama **función de regresión**. De hecho, la **función de regresión** se define como $r(x) = E(Y|X = x)$ y es por eso que queremos calcular $P(Y = y|X = x)$.

Supongamos que tenemos entonces una muestra aleatoria $(X_i, Y_i)$ con $Y_i = r(X_i) + \epsilon_i$ y queremos estimar la función de regresión $r$. Podemos hacer

$$\begin{aligned}\hat{r}_h(x) &= \sum_{i=1}^n Y_i \frac{K(\frac{x-X_i}{h})}{\sum_{i=1}^n K(\frac{x-X_i}{h})} \\ &= \sum_{i=1}^n Y_i W_i(x),\end{aligned}$$

y se llama **Estimador No Paramétrico de la Regresión de Nadaraya–Watson**; $\hat{r}_h(x)$ media ponderada.

Podemos tomar en general otros núcleos K tales como

- Núcleo Rectangular: $K(t) = \frac{1}{2}\mathcal{I}_{[-1,1]}(t)$;
- Núcleo Triangular: $K(t) = (1 - |t|)\mathcal{I}_{[-1,1]}(t)$;
- Núcleo Gaussiano: $K(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}$;
- Núcleo Epanechnikov: $K(t) = \frac{3}{4}(1 - t^2)\mathcal{I}_{[-1,1]}(t)$.

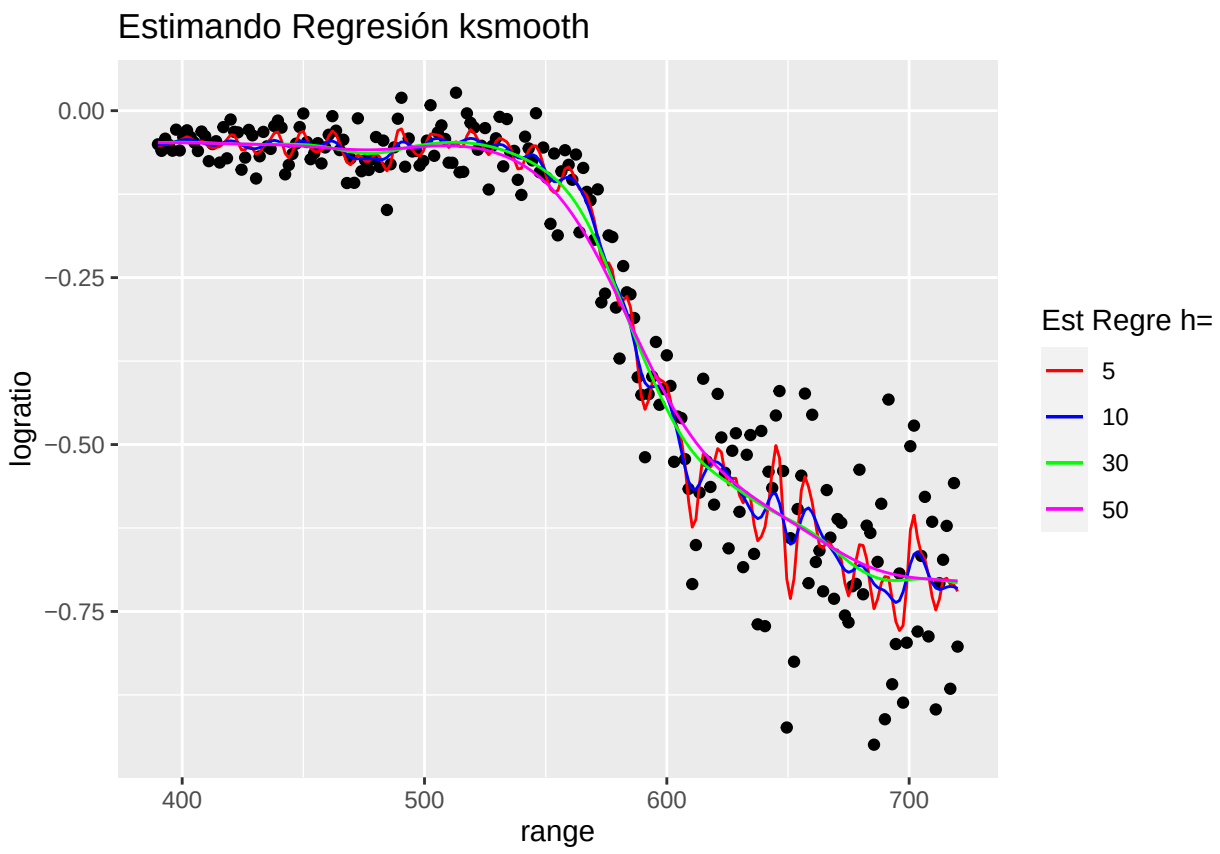
La gracia del estimador es que mediante este para cada $Y_i$ obtenemos un **valor predicho** (explicitamos la dependencia de la ventana h)

$$\hat{Y}_{i,h} = \hat{r}_h(X_i) = \sum_{j=1}^n Y_j \frac{K(\frac{X_i-X_j}{h})}{\sum_{j=1}^n K(\frac{X_i-X_j}{h})} = \sum_{j=1}^n Y_j W_{j,h}(X_i).$$

## Resolvamos el ejercicio

```
regre_5=ksmooth(datos$range,datos$logratio,kernel = 'normal', bandwidth = 5)
# si ponemos ademas el parametro x.points=datos$range la funcion la evalua en esos ptos.
regre_10=ksmooth(datos$range,datos$logratio,kernel = 'normal', bandwidth = 10)
regre_30=ksmooth(datos$range,datos$logratio,kernel = 'normal', bandwidth = 30)
regre_50=ksmooth(datos$range,datos$logratio,kernel = 'normal', bandwidth = 50)
```

```r
datos_plot=data.frame(cbind('Explicativa'=regre_5$x,
                            'Estimacion_5'=regre_5$y,
                            'Estimacion_10'=regre_10$y,
                            'Estimacion_30'=regre_30$y,
                            'Estimacion_50'=regre_50$y))
datos_plot %>%
  ggplot()+
  geom_point(data=datos,aes(x=range,y=logratio))+
  geom_line(aes(x=Explicativa,y=Estimacion_5,col='5'))+
  geom_line(aes(x=Explicativa,y=Estimacion_10,col='10'))+
  geom_line(aes(x=Explicativa,y=Estimacion_30,col='30'))+
  geom_line(aes(x=Explicativa,y=Estimacion_50,col='50'))+
  scale_color_manual("Est Regre h=",
                     breaks=c('5','10','30','50'),
                     values = c('red','blue','green','magenta'))+
  labs(title="Estimando Regresión ksmooth")
```



3. Para cada una de las 4 estimaciones obtenidas en el item anterior compute el Error Cuadrático de Predicción Promediado (ECPP (h)). ¿Cuál de las 4 ventanas consideradas da el menor ECPP (h)? ¿Cómo se puede justificar lo que está ocurriendo?

Resolución:

## Un poco de contexto

Dado el estimador de N-W y la estimación

$$\hat{Y}_{i,h} = \hat{r}_h(X_i) = \sum_{j=1}^{n} Y_j W_{j,h}(X_i),$$

podemos calcular

- El **i-ésimo Error de Predicción (ECP)**

$$Y_i - \hat{Y}_{i,h} = Y_i - \hat{r}_h(X_i);$$

- El **i-ésimo Error Cuadrático de Predicción (ECP)**

$$(Y_i - \hat{Y}_{i,h})^2 = (Y_i - \hat{r}_h(X_i))^2;$$

- El **Training Error - Error Cuadrático de Predicción Promediado (ECPP)**

$$ECPP(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_{i,h})^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{r}_h(X_i))^2.$$

Lo ideal es usar 3 conjuntos de datos:

- Training set-Muestra de entrenamiento: observaciones que se utilizan para construir $\hat{r}$;
- Validation set-Muestra de validación: observaciones que se utilizan para la elección del modelo;
- Testing set-Muestra de testeo: observaciones que se utilizan para calcular el error generalizado del modelo final elegido.

En general usaremos (nosotros) Training set=80% de los datos y Test set=20% de los datos.

Hay otras formas de hacer data-splitting para armarnos los diferentes sets: 80-20 // k folder //leave one our, etc.

## Resolvamos el ejercicio

Definamos primero una función estimadora usando ksmooth, que tome como parámetros el valor $x$ (de la variable explicativa a la cual le estamos estimando la variable respuesta $y$) y la ventana $h$.

```
estim_regre=function(x,h){
  f=approxfun(ksmooth(datos$range, datos$logratio, kernel='normal',bandwidth=h))
  f(x)
}
```

Definamos ahora el ECPP.

```
ECPP=function(h){
  estimacion=ksmooth(datos$range, datos$logratio, kernel='normal',bandwidth=h)$y
  mean((datos$logratio-estimacion)^2)
}
```

Calculemos el ECPP para cada una de las ventanas del item anterior.

```
hs=c(5,10,20,50)
errores=c()
for(i in hs){
  errores=c(errores,ECPP(i))
}
names(errores)=paste('Error h=',hs)
errores
```

```
##  Error h= 5 Error h= 10 Error h= 20 Error h= 50
## 0.003848877 0.005164205 0.005819510 0.006200361
```

4. Halle mediante el criterio de Convalidación Cruzada CV (h) la ventana óptima. Realice la búsqueda en una grilla para valores de h entre 3 y 165 con paso 1. Realice un plot de h vs. CV (h).

Resolución:

Se define la **Pérdida de Convalidación Cruzada** como

$$CV(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{r}_{h,-i}(X_i))^2,$$

donde

$$\hat{r}_{h,-i}(X_i) = \sum_{j\neq i}^{n} Y_j \frac{K(\frac{X_i - X_j}{h})}{\sum_{j\neq i}^{n} K(\frac{X_i - X_j}{h})}.$$

Y se define la **Ventana de Convalidación Cruzada** como

$$h_{CV} = argmin_h \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{r}_{h,-i}(X_i))^2.$$

## Resolvamos el ejercicio.

Defino la función $\hat{r}_{h,-i}$ que toma como parámetros el valor de $x$ al cual estamos estimando $r(x)$, la coordenada $i$ a dejar afuera y la ventana $h$.

```
r_i=function(x,i,h){
  f=approxfun(ksmooth(datos$range[-i], datos$logratio[-i], kernel='normal',bandwidth=h))
  f(x)
}
```

Defino ahora la función CV(h).

```
CV=function(h){
  estimaciones_sin_i=c()
  for(i in 1:length(datos$range)){
    x_i=datos$range[i]
    estimaciones_sin_i=c(estimaciones_sin_i,r_i(x_i,i,h))
  }
  # le saco la primera y ultima coord porque son NA
  estimaciones_sin_i=estimaciones_sin_i[-1][-length(datos$range)+1]
  # para calcular la media tambien le saco esas coordenadas a datos$logratio
  mean((datos$logratio[-1][-length(datos$range)+1]-estimaciones_sin_i)^2)
}
```

Por último defino la función h_CV que devuelve la ventana de Convalidación Cruzada.

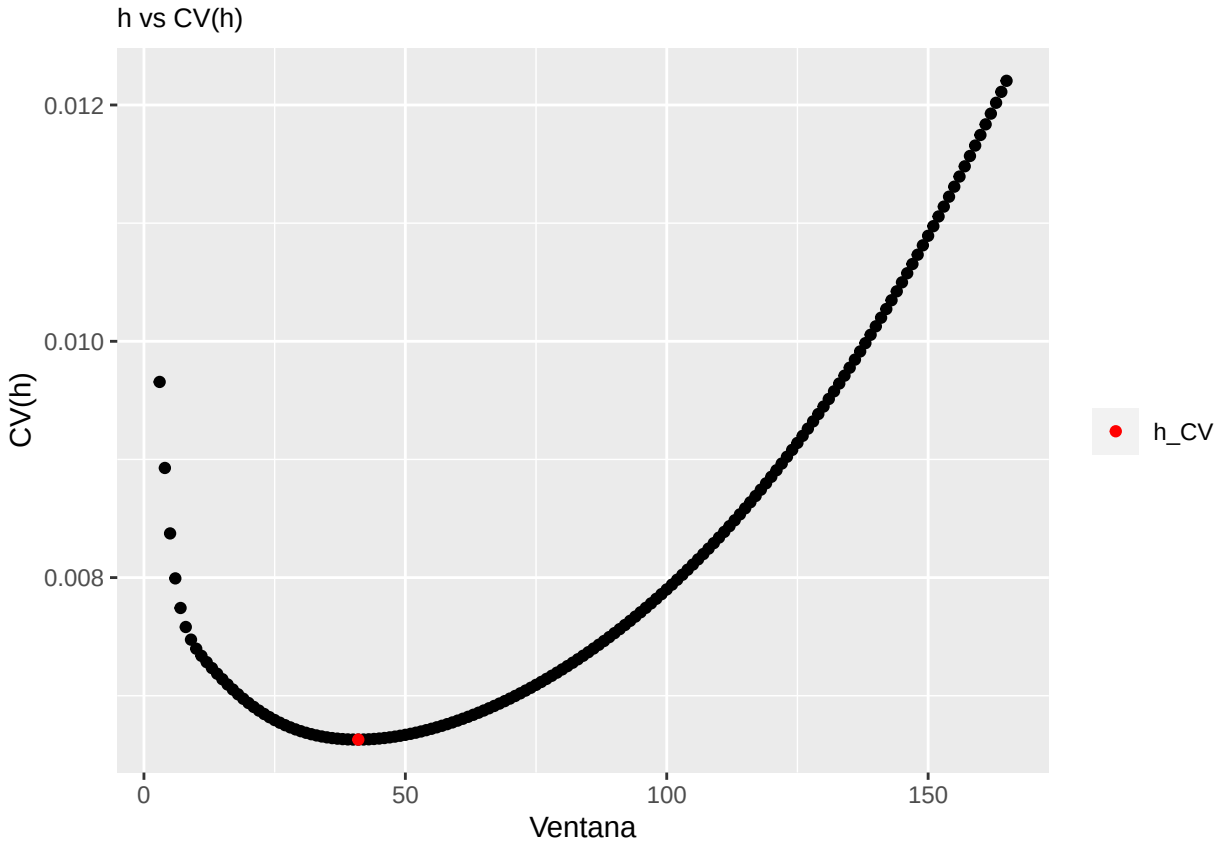
```
h_CV=function(valores_h){
  vect_to_min=sapply(valores_h,CV)
  valores_h[which.min(vect_to_min)]
}
```

Calculemos finalmente la ventana de Convalidación Cruzada, h_CV, para valores_h=seq(3,165,1) y ploteamos h vs CV(h), poniendo en rojo el h_CV.

```
ventanas=seq(3,165,1)
vect_CV=sapply(ventanas,CV)
h_CV_plot=h_CV(ventanas)
h_CV_plot
```

```
## [1] 41
```

```r
datos_plot=data.frame(cbind('Ventana'=ventanas,
                            'CV'=vect_CV))
datos_plot %>%
  ggplot()+
  geom_point(aes(x=Ventana,y=CV))+
  geom_point(data=data.frame('x'=h_CV_plot,
                             'y'=CV(h_CV_plot)),
             aes(x=x,y=y,col='h_CV'))+
  scale_color_manual("",
                     breaks=c('h_CV'),
                     values=c('red'))+
  ylab("CV(h)") +
  labs(title="h vs CV(h)")+
  theme(plot.title = element_text(size=10))
```

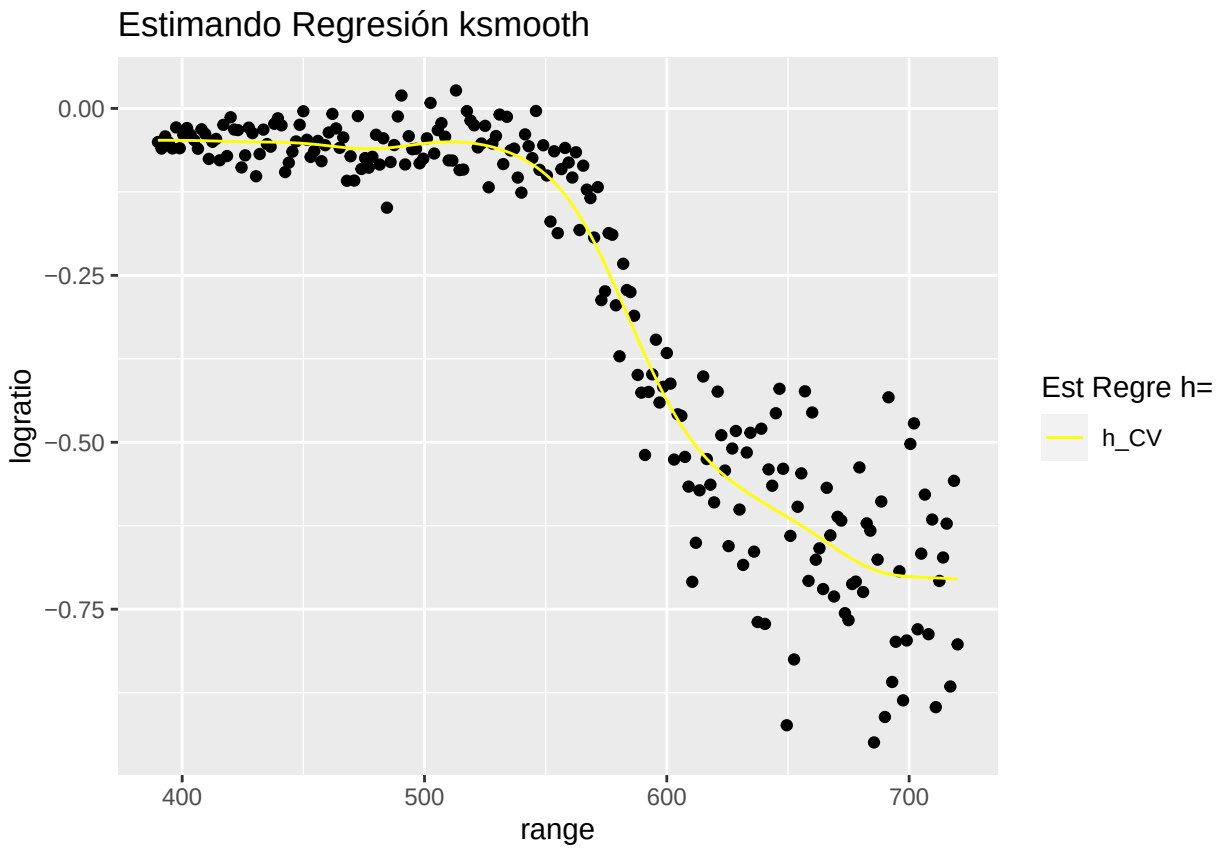



5. Grafique los puntos observados y la función de regresión estimada por el método de Nadaraya-Watson con la ventana óptima hallada.

Resolución:

```r
regre_h_CV=ksmooth(datos$range,datos$logratio,kernel = 'normal', bandwidth = h_CV_plot)
datos_plot=data.frame(cbind('Explicativa'=regre_h_CV$x,
                            'Estimacion_h_CV'=regre_h_CV$y
                            # 'Estimacion_5'=regre_5$y,
                            # 'Estimacion_10'=regre_10$y,
                            # 'Estimacion_20'=regre_20$y,
                            # 'Estimacion_50'=regre_50$y
                            ))
```

```r
datos_plot %>%
  ggplot()+
  geom_point(data=datos,aes(x=range,y=logratio))+
  geom_line(aes(x=Explicativa,y=Estimacion_h_CV,col='h_CV'))+
  # geom_line(aes(x=Explicativa,y=Estimacion_5,col='5'))+
  # geom_line(aes(x=Explicativa,y=Estimacion_10,col='10'))+
  # geom_line(aes(x=Explicativa,y=Estimacion_20,col='20'))+
  # geom_line(aes(x=Explicativa,y=Estimacion_50,col='50'))+
  scale_color_manual("Est Regre h=",
                     breaks=c('h_CV'
                              #'5','10','20','50'
                              ),
                     values = c('yellow'
                                #'red','blue','green','magenta'
                                ))+
  labs(title="Estimando Regresión ksmooth")
```




6. Compute el Error Cuadrático de Predicción Promediado de la estimación provista por el método de Nadaraya-Watson con la ventana óptima hallada: ECPP (h opt )

Resolución:

```r
ECPP(h_CV_plot)
```

```
## [1] 0.006074147
```

7. Compute la Perdida de Convalidación Cruzada asociada al la estimación provista por el método de Nadaraya-Watson con la ventana óptima hallada. CV (h opt )

Resolución:

```
CV(h_CV_plot)
```

```
## [1] 0.006628977
```

Notemos que el error es muy parecido a ECPP(h_CV_plot), donde (si entiendo bien) ECPP vendría a ser el testing error (con nuestros datos chequeamos el error del estimador... que armamos con esos mismos datos, en algun sentido es como pisarse la cola) y CV(h_CV_plot) es como un error test (donde el test set lo obtenemos del training set a partir del metodo de convalidación cruzada).

LO HICE "MAL" PORQUE USE EL MISMO SET DE DATOS EN TODOS LADOS Y ACA TENDRIA QUE HABER USADO EL TEST SET.

8. ¿Qué se puede hacer con knn?

Resolución:

Es repetir lo que hicimos en el Ejercicio 4 pero para el estimador $\hat{r}_h$ resultante del método knn. Este método lo que hace es estimar a $P(Y = y|X = x)$ por la fracción de puntos en $N_x$ cuya etiqueta es igual a $y$, con $N_x$ el conjunto de los k puntos mas cercanos a x:

$$\widehat{P}(Y = y|X = x) = \frac{1}{k}\sum_{i=1}^{n} \mathcal{I}(y_i = y).$$

Defino el estimador asociado al metodo knn que tome como parámetros $x$ y $k$. No voy a usar exactamente estas fcs pero nos van a servir para entender como definir $r_{h-i}$ en breve.

```
N_x=function(x,k){
  distancias=abs(datos$range-x)
  posiciones=c()
  for(i in (1:k)){
    menor_distancia=which.min(distancias)
    posiciones=c(posiciones,menor_distancia)
    distancias[menor_distancia]=NA
    # Le pongo NA al lugar de la menor distancia, asi en la
    # proxima iteracion la menor distancia cambia.
  }
  posiciones
}
r_knn=function(x,k){
  mean(datos$logratio[N_x(x,k)])
}
```

Y ahora repitamos lo que hicimos en el 4. Llamemos $r^{knn}$ a este estimador.

Definimos la **Pérdida de Convalidación Cruzada knn** como

$$CV_{knn}(k) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - r^{knn}_{k,-i}(X_i))^2.$$

Y definimos la **Ventana de Convalidación Cruzada knn** como

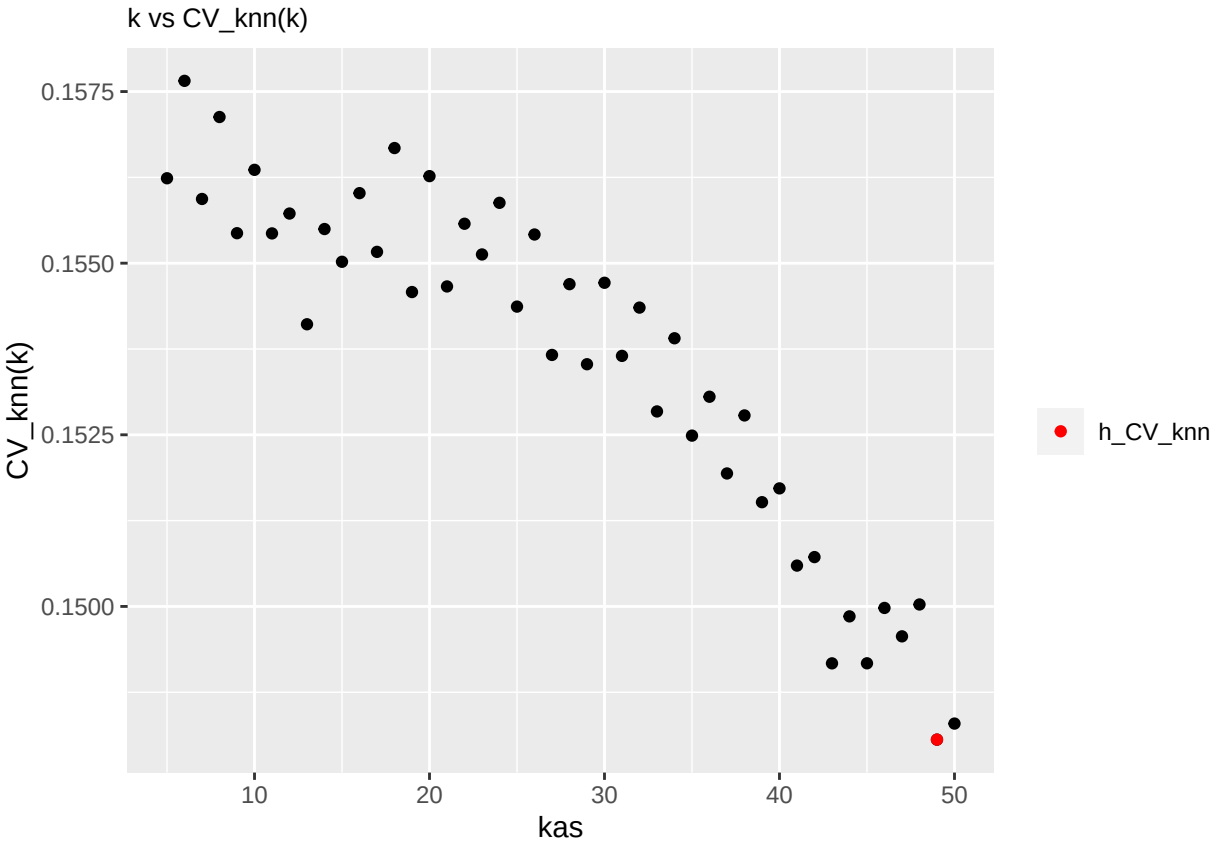
$$h_{CV} = argmin_k CV_{knn}(k).$$

Definimos las funciones N_x_i que devuelve el $N_x$ pero sacando el i-esimo dato, además de las funciones $r^{knn}_{k,-i}$, $CV_{knn}$ y $h_{CV_{knn}}$.

```
N_x_i=function(x,k,i){
  distancias=abs(datos$range[-i]-x)
  posiciones=c()
  for(i in (1:k)){
    menor_distancia=which.min(distancias)
    posiciones=c(posiciones,menor_distancia)
    distancias[menor_distancia]=NA
    # Le pongo NA al lugar de la menor distancia, asi en la
    # proxima iteracion la menor distancia cambia.
  }
  posiciones
}
r_knn_i=function(x,k,i){
  mean(datos$logratio[N_x_i(x,k,i)])
}
#####################################
CV_knn=function(k){
  estimaciones_sin_i=c()
  for(i in 1:length(datos$range)){
    x_i=datos$range[i]
    estimaciones_sin_i=c(estimaciones_sin_i,r_knn_i(x_i,k,i))
  }
  mean((-estimaciones_sin_i)^2)
}
#####################################
h_CV_knn=function(valores_k){
  vect_to_min=sapply(valores_k,CV_knn)
  valores_k[which.min(vect_to_min)]
}
```

Calculemos finalmente la ventana de Convalidación Cruzada, h_CV_knn, para valores_k=seq(5,50,1) y ploteamos k vs CV_knn(k), poniendo en rojo el h_CV_knn.

```
kas=seq(5,50,1)
vect_CV_knn=sapply(kas,CV_knn)
h_CV_knn_plot=h_CV_knn(kas)
datos_plot=data.frame(cbind('kas'=kas,
                      'CV_knn'=vect_CV_knn))
datos_plot %>%
  ggplot()+
  geom_point(aes(x=kas,y=CV_knn))+
  geom_point(data=data.frame('x'=h_CV_knn_plot,
                             'y'=CV_knn(h_CV_knn_plot)),
             aes(x=x,y=y,col='h_CV_knn'))+
  scale_color_manual("",
                     breaks=c('h_CV_knn'),
                     values=c('red'))+
  ylab("CV_knn(k)") +
  labs(title="k vs CV_knn(k)")+
  theme(plot.title = element_text(size=10))
```




FALTA HACER UN GRAFICO DE LA ESTIMACION!

**Ejercicio II.** Bonus Track Revisitemos la guía de alturas con este nuevo marco teórico; elija h y k con las diferentes propuestas consideradas en clase.

Resolución: