# Estadística - TP Final

Cesaroni, Finkelstein, Zanela

2024-07-10

## 1. Teórico

$$(X_i, Y_i) \sim (X, Y), \quad \text{donde } X, Y \in \mathbb{R} \text{ y } E(Y \mid X = x) = m(x),$$

con $m : \mathbb{R} \to \mathbb{R}$ suave, $K$ núcleo que satisface las propiedades a)-e) $h \in \mathbb{R}$ ventana, sea el estimador de Nadaraya–Watson de $m$ dado por

$$\hat{m}_h(x) = \sum_{i=1}^{n} Y_i \frac{K\left(\frac{X_i - x}{h}\right)}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - x}{h}\right)} = \sum_{i=1}^{n} Y_i w_{i,h}(x). \tag{1}$$

Llamemos $\mathbf{Y}$ al vector de respuestas con $i$-ésima componente dada por $Y_i$.

**(a)** $\hat{Y}_i = \hat{m}_h(X_i)$ y sea $\hat{Y}$ el vector de predichos donde la $i$-ésima componente es $\hat{Y}_i$.

Queremos ver que $\hat{Y} = SY$:

Como

$$\hat{m}_h(x) = \sum_{i=1}^{n} Y_i w_{i,h}(x) = \begin{pmatrix} w_{1,h}(x) & w_{2,h}(x) & \dots & w_{n,h}(x) \end{pmatrix} \cdot \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}$$

$\implies$

$$\hat{\mathbf{Y}} = \begin{pmatrix} \hat{m}_h(X_1) \\ \hat{m}_h(X_2) \\ \vdots \\ \hat{m}_h(X_n) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} Y_i w_{i,h}(X_1) \\ \sum_{i=1}^{n} Y_i w_{i,h}(X_2) \\ \vdots \\ \sum_{i=1}^{n} Y_i w_{i,h}(X_n) \end{pmatrix} = \underbrace{\begin{pmatrix} w_{1,h}(X_1) & w_{2,h}(X_1) & \cdots & w_{n,h}(X_1) \\ w_{1,h}(X_2) & w_{2,h}(X_2) & \cdots & w_{n,h}(X_2) \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,h}(X_n) & w_{2,h}(X_n) & \cdots & w_{n,h}(X_n) \end{pmatrix}}_{S} \cdot \underbrace{\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}}_{Y}$$

$\implies$

$$\hat{\mathbf{Y}} = S \cdot Y$$

**(b)**

**(i)** Queremos ver que $\hat{m}_h^{(-i)}(X_i) = \frac{\hat{m}_h(X_i) - Y_i w_{i,h}(X_i)}{1 - w_{i,h}(X_i)}$

Observamos en primer lugar que

$$
\begin{aligned}
\hat{m}_h(X_i) - Y_i w_{i,h}(X_i) &= \sum_{j=1}^{n} Y_j w_{j,h}(X_i) - Y_i w_{i,h}(X_i) \\
&= \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j w_{j,h}(X_j) \\
&= \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j \frac{K\left(\frac{X_j - X_i}{h}\right)}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \\
&= \frac{1}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j\, K\left(\frac{X_j - X_i}{h}\right)
\end{aligned}
$$

Por otro lado,

$$
\begin{aligned}
\frac{1}{1 - w_{i,h}(X_i)} \cdot \frac{1}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} &= \frac{1}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right) - \frac{K\left(\frac{X_i - X_i}{h}\right)}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \cdot \sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \\
&= \frac{1}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right) - K\left(\frac{X_i - X_i}{h}\right)} \\
&= \frac{1}{\sum_{\substack{\ell=1 \\ \ell\neq i}}^{n} K\left(\frac{X_\ell - X_i}{h}\right)}
\end{aligned}
$$

Luego,

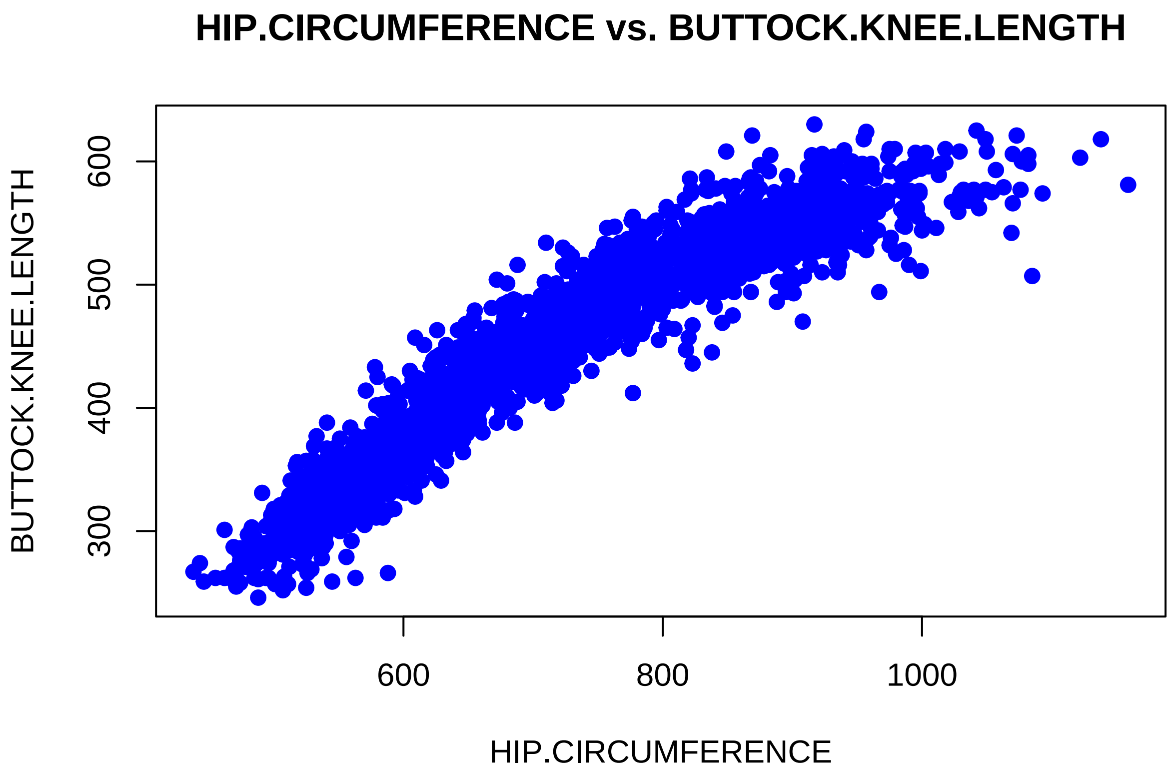
$$
\begin{aligned}
\frac{\hat{m}_h(X_i) - Y_i w_{i,h}(X_i)}{1 - w_{i,h}(X_i)} &= \frac{1}{1 - w_{i,h}(X_i)} \cdot \left( \frac{1}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j\, K\left(\frac{X_j - X_i}{h}\right) \right) \\
&= \left( \frac{1}{1 - w_{i,h}(X_i)} \cdot \frac{1}{\sum_{\ell=1}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \right) \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j\, K\left(\frac{X_j - X_i}{h}\right) \\
&= \frac{1}{\sum_{\substack{\ell=1 \\ \ell\neq i}}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j\, K\left(\frac{X_j - X_i}{h}\right) \\
&= \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j \frac{K\left(\frac{X_j - X_i}{h}\right)}{\sum_{\substack{\ell=1 \\ \ell\neq i}}^{n} K\left(\frac{X_\ell - X_i}{h}\right)} \\
&= \sum_{\substack{j=1 \\ j\neq i}}^{n} Y_j\, w_{j,h}^{-i}(X_i) \\
&= \hat{m}_h^{(-i)}(X_i)
\end{aligned}
$$

**(ii)** Vemos que $CV(h) = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{m}_h(X_i))^2}{1-w_{i,h}(X_i)}$

$$
\begin{aligned}
CV(h) &= \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{m}_h^{-i}(X_i))^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(Y_i - \frac{\hat{m}_h(X_i) - Y_i w_{i,h}(X_i)}{1-w_{i,h}(X_i)})^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - Y_i w_{i,h}(X_i) - \hat{m}_h(X_i) + Y_i w_{i,h}(X_i))^2}{(1-w_{i,h}(X_i))^2} \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{m}_h(X_i))^2}{(1-w_{i,h}(X_i))^2}
\end{aligned}
$$

## Ejercicio 2

```r
individuals <- read.table("individuals.csv", header = TRUE, sep = ";")
datos_femenino <- subset(individuals, SEX == 2 & HIP.CIRCUMFERENCE != 0
                         & BUTTOCK.KNEE.LENGTH!=0)
attach(datos_femenino)

plot(HIP.CIRCUMFERENCE, BUTTOCK.KNEE.LENGTH,
     xlab = "HIP.CIRCUMFERENCE",
     ylab = "BUTTOCK.KNEE.LENGTH",
     main = "HIP.CIRCUMFERENCE vs. BUTTOCK.KNEE.LENGTH",
     pch = 19, col = "blue")
```

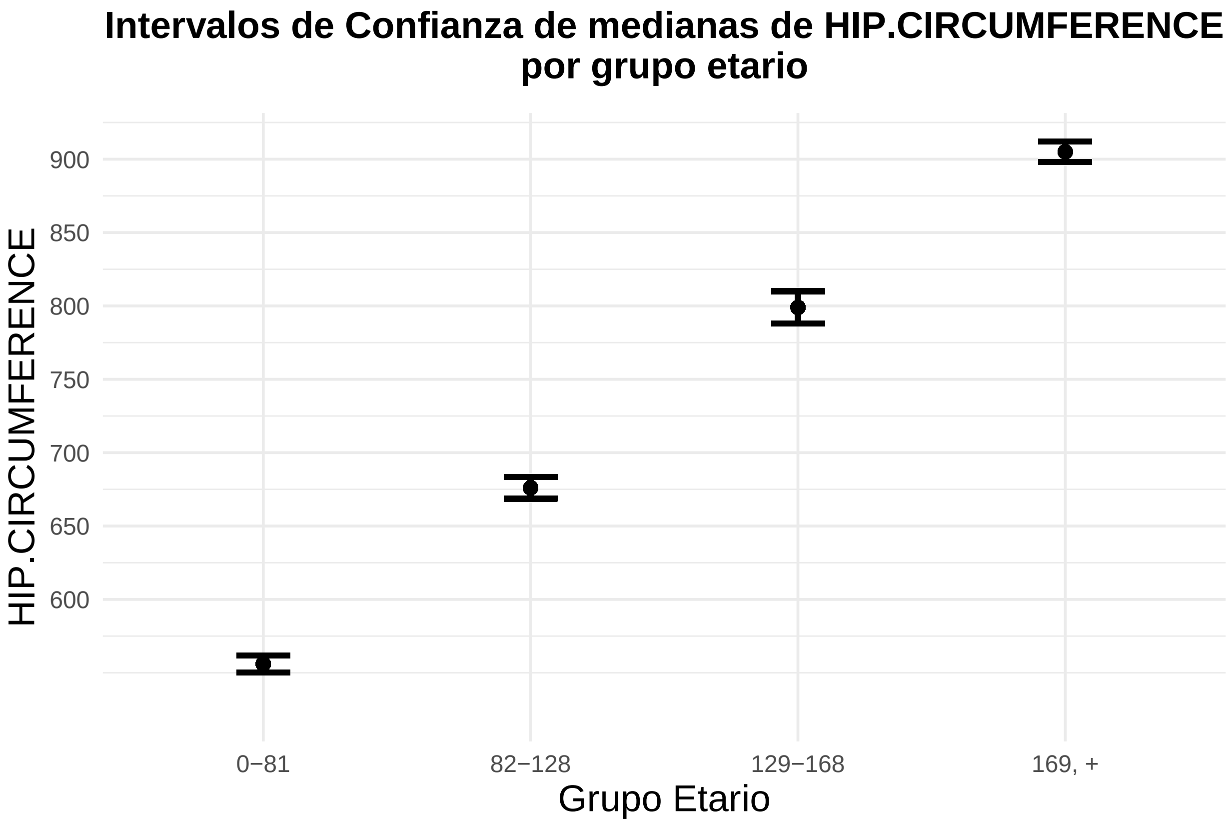



**(a)**

Este gráfico sugiere que la longitud del fémur y el perímetro de la cadera están relacionados de forma monotónicamente creciente. Cuanto mayor es la la circunferencia de la cadera, mayor es la longitud del fémur.

```
# Division de grupos etarios por cuartiles
cuartiles <- quantile(datos_femenino$AGE.IN.MONTHS,probs = c(0,0.25, 0.5, 0.75,1))
datos_femenino$AGE.GROUP <- cut(datos_femenino$AGE.IN.MONTHS,cuartiles,
                                include.lowest = TRUE,
                                labels = c("0-81","82-128","129-168","169, +"))

#Estimacion de medianas
medianas_estimadas <- tapply(datos_femenino$HIP.CIRCUMFERENCE,
                             datos_femenino$AGE.GROUP,median)


# Función para estimación del se de las medianas

estimar_se_mediana <- function (x, B = 1000){
  titahatboot <- rep(0,B)
  for(i in 1:B){
    Xboot <- sample(x,length(x),replace = TRUE)
    titahatboot[i] <- median(Xboot)
  }
  return(sqrt(mean((titahatboot - mean(titahatboot))^2)))
```

```r
}

# Cálculo de intervalo bootstrap nivel 0.95

intervalos_confianza <- tapply(datos_femenino$HIP.CIRCUMFERENCE,
                               datos_femenino$AGE.GROUP,
                               function(x){
  se_boot <- estimar_se_mediana(x)
  intervalo_boot <- c(median(x) - 1.96*se_boot,median(x) + 1.96*se_boot)
  return(intervalo_boot)
})


# Gráfico

# Los siguientes data frames son unicamente creados para generar el plot

df_medianas <- data.frame(
  AGE.GROUP = names(medianas_estimadas),
  MEDIAN = medianas_estimadas
)
df_intervalos <- data.frame(
  AGE.GROUP = names(intervalos_confianza),
  INTERVAL = intervalos_confianza
)
df_intervalos$LOWER <- sapply(df_intervalos$INTERVAL, function(x) x[1])
df_intervalos$UPPER <- sapply(df_intervalos$INTERVAL, function(x) x[2])

df_plot <- merge(datos_femenino,merge(df_medianas,df_intervalos,by="AGE.GROUP"),
                 by="AGE.GROUP")


ggplot(df_plot, aes(x = AGE.GROUP, y = MEDIAN)) +
  geom_point(size = 2) +
  geom_errorbar(aes(ymin = LOWER, ymax = UPPER), width = 0.2, linewidth = 1) +
  scale_y_continuous(limits = c(min(df_plot$LOWER) * 0.95, max(df_plot$UPPER)),
                     breaks = seq(from = 600, to = 1000, by = 50)) +
  labs(
    title = "Intervalos de Confianza de medianas de HIP.CIRCUMFERENCE\npor grupo etario",
       x = "Grupo Etario", y = "HIP.CIRCUMFERENCE") +
  theme_minimal() + theme(
        plot.title = element_text(hjust = 0.5, face = "bold", size = 14,
                                  margin = margin(10, 0, 10, 0)),
        axis.title = element_text(size = 14))
```




**(b)**

Para calcular los intervalos mencionados, utilizamos el método de intervalo de bootstrap normal. Necesitamos el error estándar de la mediana, el cual obtenemos mediante una estimación bootstrap con la función `estimar_se_mediana`. Esta función realiza un muestreo `B` veces con reposición de HIP.CIRCUMFERENCE y calcula la mediana de estos datos, almacenándolos en un vector. Una vez que tenemos el vector con las `B` medianas calculadas, calculamos el error estándar de la siguiente manera:

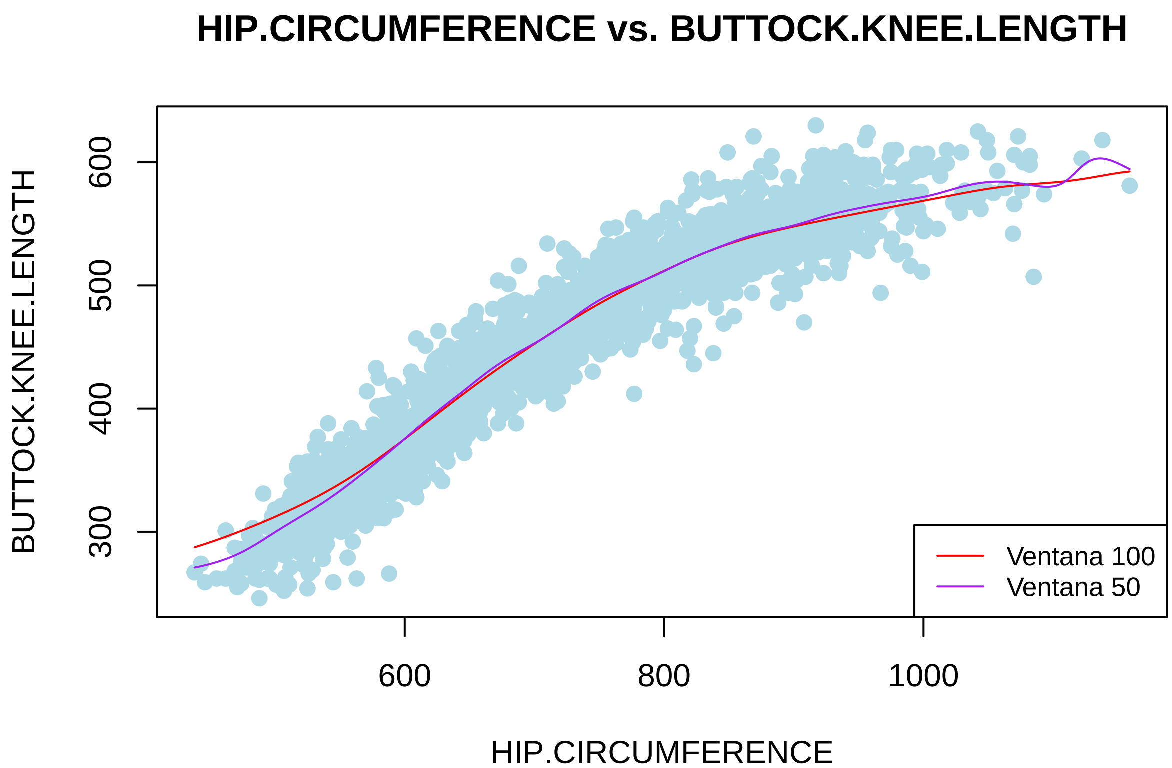
$$\sqrt{\frac{1}{n}\sum\left((\hat{se}_{\text{boot}} - \overline{\hat{se}_{\text{boot}}})^2\right)}$$

Y este procedimiento lo hacemos por cada grupo etario.

**(c)**

```r
regresion_100 <- ksmooth(HIP.CIRCUMFERENCE, BUTTOCK.KNEE.LENGTH, "normal",
                         bandwidth = 100)
regresion_50 <- ksmooth(HIP.CIRCUMFERENCE, BUTTOCK.KNEE.LENGTH, "normal", bandwidth = 50)


plot(datos_femenino$HIP.CIRCUMFERENCE, datos_femenino$BUTTOCK.KNEE.LENGTH,
     xlab = "HIP.CIRCUMFERENCE",
     ylab = "BUTTOCK.KNEE.LENGTH",
     main = "HIP.CIRCUMFERENCE vs. BUTTOCK.KNEE.LENGTH",
     pch = 19, col = "light blue")
```

```r
lines(regresion_100, col = "red")
lines(regresion_50, col = "purple")

legend("bottomright", legend = c("Ventana 100", "Ventana 50"),
       col = c("red", "purple"), lty = 1, cex = 0.8)
```

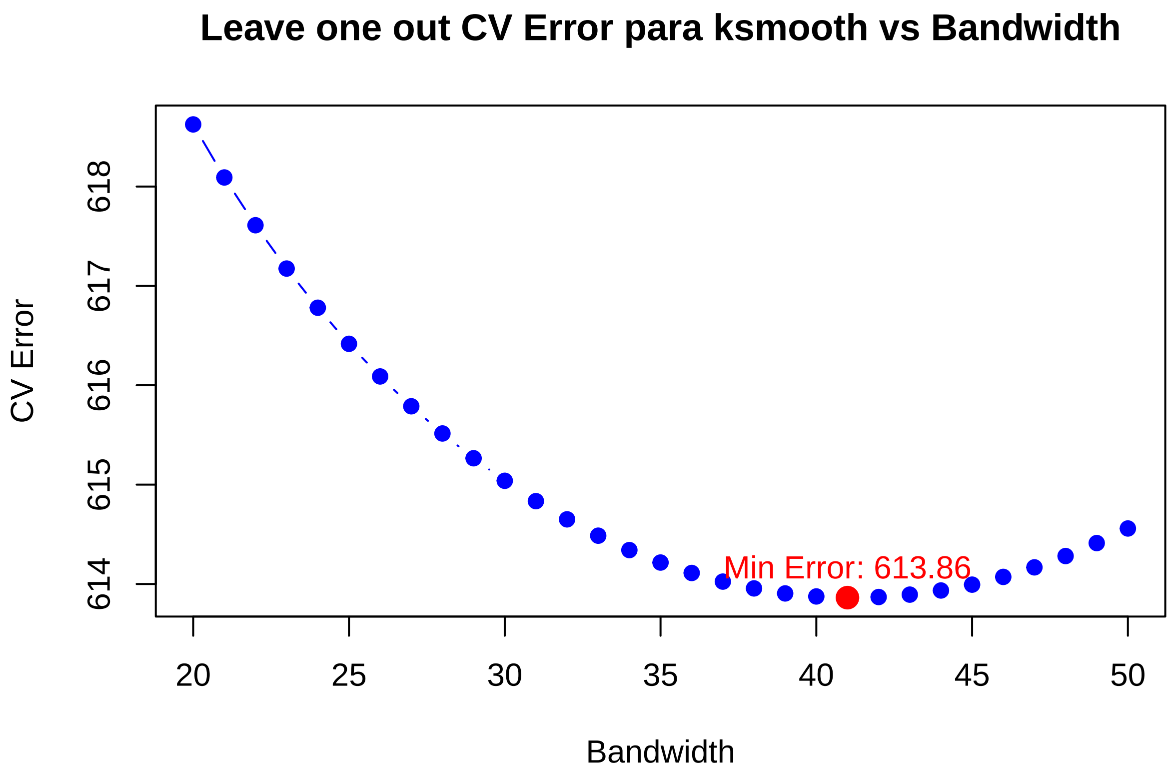



**i)**

El gráfico muestra que la ventana de 100 suaviza la regresión y es menos sensible a los outliers, mientras que la ventana de 50 ajusta mejor los datos en la parte inferior izquierda, evitando que los valores altos afecten las predicciones de los valores más bajos. Usaríamos la ventana de 50, puesto tiene un mejor ajuste a los datos con valores bajos de HIP.CIRCUMFERENCE y no pareciera estar suficientemente errada en los datos con valores altos de HIP.CIRCUMFERENCE como para que valga más la pena elegir la ventana de 100.

```r
# Defino la grilla de bandwidths
bandwidths <- 20:50

# Defino la función de Leave-One-Out-CV
cv_leave_one_out <- function(x, y, bandwidth) {

  n <- length(x)
  errors <- numeric(n)

  for (i in 1:n) {
```

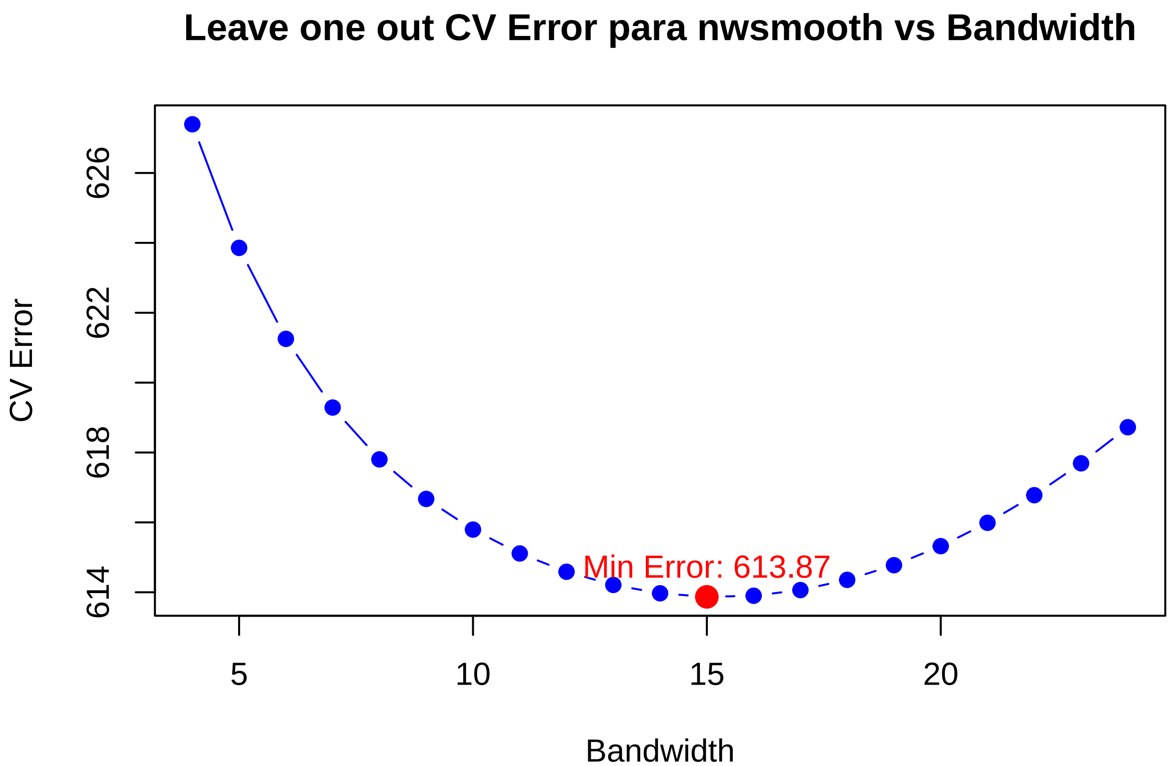
```r
    # Ajusto el modelo
    y_hat <- ksmooth(x[-i], y[-i], "normal", bandwidth = bandwidth, x.points = x[i])$y
    # Calculo el mse y lo agrego a errors
    mse <- (y[i] - y_hat)^2
    errors[i] <- mse
  }

  return(mean(errors))
}

# Calculo el error de cv para cada valor de h en la grilla
errors <- sapply(bandwidths, cv_leave_one_out, x = HIP.CIRCUMFERENCE,
                 y = BUTTOCK.KNEE.LENGTH)

# Encuentro el valor mínimo de error y su índice correspondiente
min_error <- min(errors)
min_error_index <- which.min(errors)
ksmooth_best_bandwidth <- bandwidths[min_error_index]

# Ploteo la función objectivo marcando el valor mínimo
plot(bandwidths, errors, type = "b",
     main = "Leave one out CV Error para ksmooth vs Bandwidth",
     xlab = "Bandwidth", ylab = "CV Error", col = "blue", pch = 19)

# Marca el valor mínimo de error en el plot
points(ksmooth_best_bandwidth, min_error, col = "red", pch = 19, cex = 1.5)
text(ksmooth_best_bandwidth, min_error, labels =
       paste("Min Error:", round(min_error, 2)), pos = 3, col = "red")
```




**ii)**

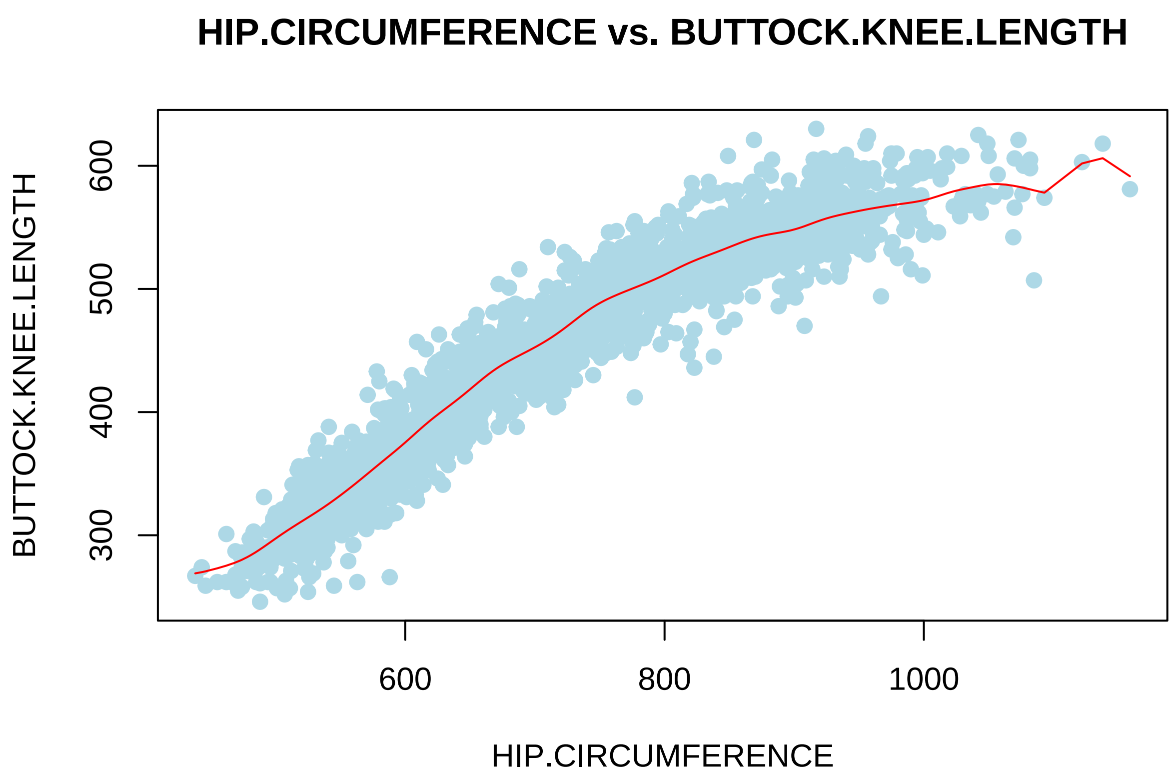
```
# Defino el estimador de NW para un x0
nwsmooth <- function(x0, x, y, bandwidth){

  # Armo el vector al que le aplico el núcleo normal
  vect <- (x - x0)/bandwidth

  # Calculo los pesos
  weights <- dnorm(vect)

  return(sum(weights*y)/sum(weights))
}


# Calculo el leave-one-out CV
cv_nwsmooth <- function(bandwidth, x, y){

  # Vector m, con nwsmooth aplicado a x_i en la posición i
  m <- sapply(x, nwsmooth, x, y, bandwidth)

  # Armo el vector w, que tiene el coeficiente w_i,h en la posición i
  # Matriz con (x[i]-x[j])/bandwidth en la posición ij
  arg_matrix <- outer(x, x, "-") * (1/bandwidth)
  # Núcleo normal aplicado a cada posición
```

```
  kernel_matrix <- apply(arg_matrix, c(1, 2), dnorm)
  # Sumo las columnas de kernel_matrix, me queda el denominador de w_i,h
  column_sums <- colSums(kernel_matrix)
  # El num de w_i,h para todo i
  gaussian_kernel_0 <- dnorm(0)
  w <- gaussian_kernel_0 / column_sums  # Consigo w

  # Vector con error de predicción de m_h^-i en la posición i
  errors <- ((y-m)/(1-w))^2

  return(mean(errors))
}


# Defino grilla de bandwidths y calculo el error de cv para cada valor de h en la grilla
bandwidths <- 4:24
errors <- sapply(bandwidths, cv_nwsmooth, x = HIP.CIRCUMFERENCE, y = BUTTOCK.KNEE.LENGTH)


# Encuentro el valor mínimo de error y su índice correspondiente
min_error <- min(errors)
min_error_index <- which.min(errors)
nwsmooth_best_bandwidth <- bandwidths[min_error_index]


# Ploteo la función objectivo marcando el valor mínimo
plot(bandwidths, errors, type = "b",
     main = "Leave one out CV Error para nwsmooth vs Bandwidth",
     xlab = "Bandwidth", ylab = "CV Error", col = "blue", pch = 19)

# Marca el valor mínimo de error en el plot
points(nwsmooth_best_bandwidth, min_error, col = "red", pch = 19, cex = 1.5)
text(nwsmooth_best_bandwidth, min_error,
     labels = paste("Min Error:", round(min_error, 2)), pos = 3, col = "red")
```




**iii)**

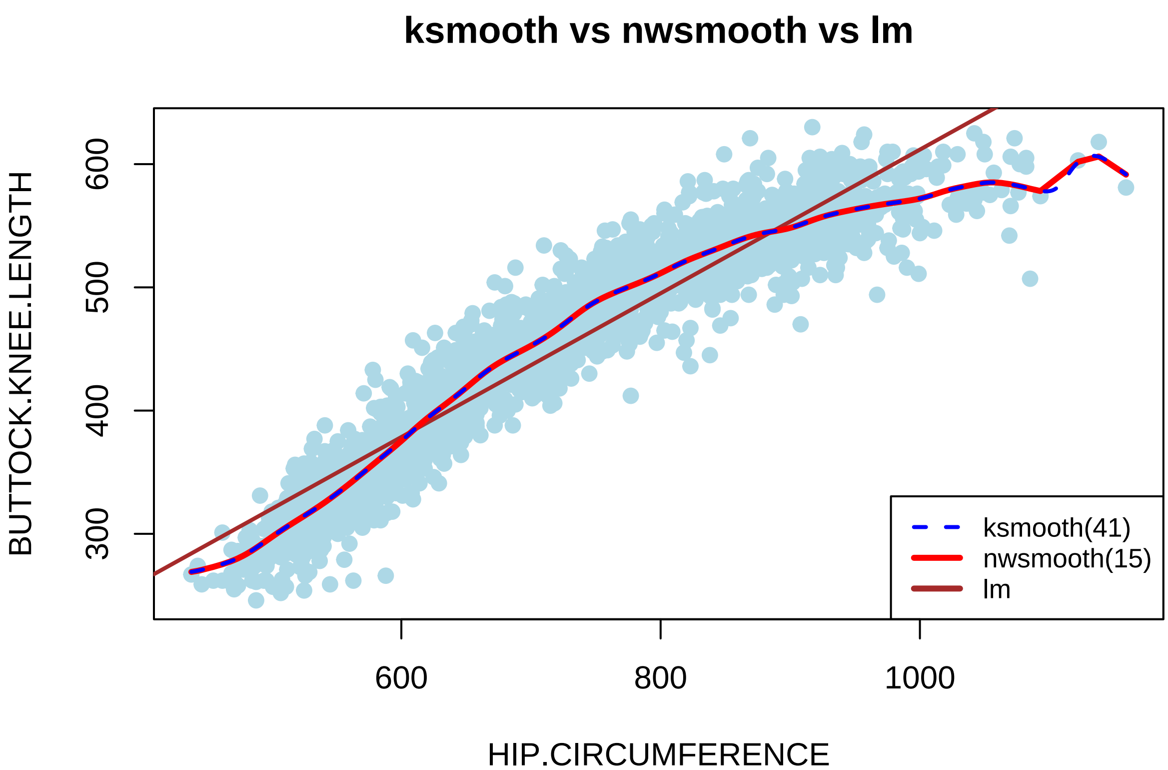
```
# Calculo la regresión
ordenado <- order(HIP.CIRCUMFERENCE)
nwfit <- sapply(HIP.CIRCUMFERENCE[ordenado], nwsmooth, HIP.CIRCUMFERENCE,
                BUTTOCK.KNEE.LENGTH, nwsmooth_best_bandwidth)


# Grafico
plot(datos_femenino$HIP.CIRCUMFERENCE, datos_femenino$BUTTOCK.KNEE.LENGTH,
     xlab = "HIP.CIRCUMFERENCE",
     ylab = "BUTTOCK.KNEE.LENGTH",
     main = "HIP.CIRCUMFERENCE vs. BUTTOCK.KNEE.LENGTH",
     pch = 19, col = "light blue")

lines(HIP.CIRCUMFERENCE[ordenado], nwfit, col = "red")
```




Al utilizar la misma grilla que en el ítem anterior, obtuvimos 20 como *bandwidth* óptimo, por lo que agrandamos la grilla para considerar ventanas más chicas. Además, teniendo en cuenta que las operaciones del *leave-one-out-cv*, basadas en el ítem teórico, están vectorizadas, como esto conlleva un menor costo computacional, podemos ampliar la grilla sin problemas.

```
# Ajusto los modelos
recta_cuad_min <- lm(BUTTOCK.KNEE.LENGTH ~ HIP.CIRCUMFERENCE)
ksmoothfit <- ksmooth(HIP.CIRCUMFERENCE, BUTTOCK.KNEE.LENGTH, "normal",
                      bandwidth = ksmooth_best_bandwidth)
ordenado <- order(HIP.CIRCUMFERENCE)
nwfit <- sapply(HIP.CIRCUMFERENCE[ordenado], nwsmooth, HIP.CIRCUMFERENCE,
                BUTTOCK.KNEE.LENGTH, nwsmooth_best_bandwidth)

# Grafico
plot(HIP.CIRCUMFERENCE, BUTTOCK.KNEE.LENGTH,
     xlab = "HIP.CIRCUMFERENCE",
     ylab = "BUTTOCK.KNEE.LENGTH",
     main = "ksmooth vs nwsmooth vs lm",
     pch = 19, col = "light blue")

abline(recta_cuad_min, col = "brown", lwd = 2)
lines(HIP.CIRCUMFERENCE[ordenado], nwfit, col = "red", lwd = 3)
lines(ksmoothfit, col = "blue", lwd = 2, lty = 2)
```

```r
legend("bottomright", legend = c("ksmooth(41)", "nwsmooth(15)", "lm"),
       col = c("blue", "red", "brown"), lty = c(2,1,1), cex = 0.8,
       lwd = c(2,3,3))
```




**iv)**

**(d)**

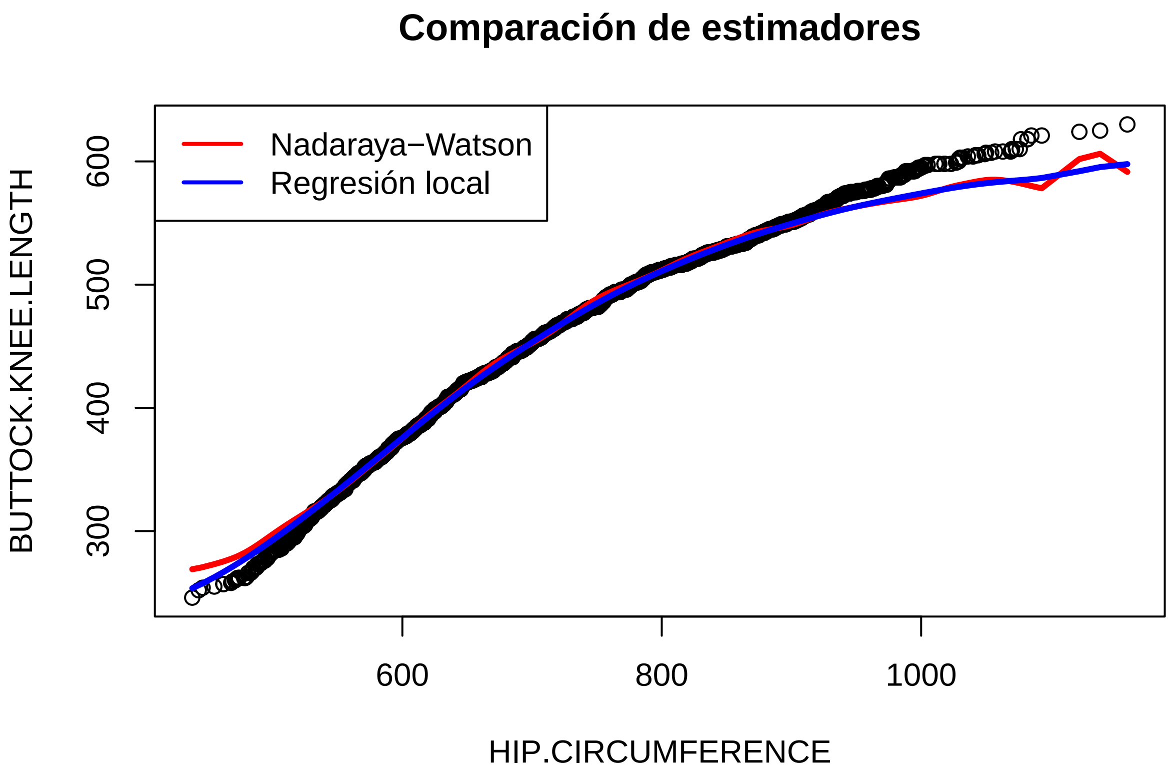
```r
linearsmooth <- function(x, X_data, Y_data, h){
  K <- function(u) dnorm(u)
  y_est <- rep(NA, length(X_data))
  for(i in 1:length(x)){
    X <- matrix(0, nrow=length(X_data), ncol=2)
    X[,1] <- 1
    X[,2] <- X_data - x[i]
    W <- diag(K((X_data-x[i])/h))
    Y <- Y_data
    inv <- solve(t(X)%*%W%*%X)
    mean_sq_est <- inv%*%t(X)%*%W%*%Y
    y_est[i] <- mean_sq_est[1]
  }

  return(y_est)
}
```

**i)**

```r
reg_est <- linearsmooth(datos_femenino$HIP.CIRCUMFERENCE,
                        datos_femenino$HIP.CIRCUMFERENCE,
                        datos_femenino$BUTTOCK.KNEE.LENGTH, 40)
reg_est <- sort(reg_est)
```

**ii)**

```r
y <- sort(datos_femenino$BUTTOCK.KNEE.LENGTH)
x <- sort(datos_femenino$HIP.CIRCUMFERENCE)
plot(x, y,
     main = "Comparación de estimadores",
     xlab = "HIP.CIRCUMFERENCE",
     ylab = "BUTTOCK.KNEE.LENGTH",
     )

# Agregar los estimadores al gráfico
lines(x, nwfit, col = "red", lwd = 3)
lines(x, reg_est, col = "blue", lwd = 3)

legend("topleft", legend = c("Nadaraya-Watson", "Regresión local"),
       col = c("red", "blue"), lty = 1, lwd = 2)
```




**iii)**

Podemos ver que el estimador de Nadaraya-Watson tiene más volatilidad, y esto se debe a que la ventana óptima de este es más pequeña que la del estimador de regresión local. Por otro lado, en los puntos de la izquierda vemos que el estimador de Nadaraya-Watson tiene un sesgo mayor, y esto se debe a que los valores de "y" con los que se ajusta el estimador son casi todos mayores a los valores verdaderos de ese sector. Este sesgo inherente al estimador de Nadaraya-Watson se intenta mejorar con el estimador de regresión local. Vemos esa mejoría en el sector izquierdo, aunque no se percibe en el derecho. Más allá de eso, los dos estimadores son bastente similares visualmente.