La_Dibunetta

Jalil_Pañale_Yudcovsky

2024-07-06

1)-a

Consideremos el estimador de Nadaraya–Watson dado por:

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

donde

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

Para los valores predichos $\hat{Y}_i = \hat{m}_h(X_i)$, tenemos

$$\hat{Y}_i = \sum_{j=1}^n Y_j w_{j,h}(X_i),$$

donde

$$w_{j,h}(X_i) = \frac{K\left(\frac{X_j - X_i}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)}.$$

Podemos escribir esto en forma matricial. Sea $Y = (Y_1, Y_2, \ldots, Y_n)^T$ el vector de respuestas y $\hat{Y} = (\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_n)^T$ el vector de valores predichos. Entonces,

$$\hat{Y}_i = \sum_{j=1}^n S_{ij} Y_j,$$

donde $S_{ij} = w_{j,h}(X_i)$.

Para escribirlo en forma matricial, definimos la matriz S como una matriz $n \times n$ con elementos S_{ij} dados por

$$S_{ij} = w_{j,h}(X_i) = \frac{K\left(\frac{X_j - X_i}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)}.$$

Entonces, tenemos

$$\hat{Y} = SY,$$

donde S es la matriz de pesos $S_{ij} = w_{j,h}(X_i)$.

Esta matriz S captura cómo los valores de Y se transforman linealmente para obtener los valores predichos $\hat{Y}$ usando el estimador de Nadaraya–Watson.

1)-b

Consideremos el estimador de Nadaraya-Watson dado por:

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

donde

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

Para el estimador $\hat{m}_h^{-i}(X_i)$, debemos calcular el estimador sin la observación (X_i, Y_i):

$$\hat{m}_h^{-i}(X_i) = \frac{\sum_{\substack{j=1 \\ j\neq i}}^n Y_j K\left(\frac{X_j - X_i}{h}\right)}{\sum_{\substack{\ell=1 \\ \ell\neq i}}^n K\left(\frac{X_\ell - X_i}{h}\right)}.$$

Podemos escribir esto como:

$$\hat{m}_h^{-i}(X_i) = \frac{\sum_{j=1}^n Y_j K\left(\frac{X_j - X_i}{h}\right) - Y_i K\left(\frac{X_i - X_i}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - K\left(\frac{X_i - X_i}{h}\right)}.$$

simplificando,

$$\hat{m}_h^{-i}(X_i) = \frac{\sum_{j=1}^n Y_j K\left(\frac{X_j - X_i}{h}\right) - Y_i K(0)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - K(0)}.$$

Ahora, utilizando la definición de $\hat{m}_h(X_i)$, tenemos:

$$\hat{m}_h(X_i) = \frac{\sum_{j=1}^n Y_j K\left(\frac{X_j - X_i}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)}.$$

De aquí, se puede observar que:

$$\sum_{j=1}^n Y_j K\left(\frac{X_j - X_i}{h}\right) = \hat{m}_h(X_i) \sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right).$$

Sustituyendo esta relación en la expresión de $\hat{m}_h^{-i}(X_i)$, obtenemos:

$$\hat{m}_h^{-i}(X_i) = \frac{\hat{m}_h(X_i)\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - Y_i K(0)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - K(0)}.$$

Para simplificar, factorizamos el denominador y el numerador por $\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)$:

Denominador:

$$\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - K(0) = \sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)\left(1 - \frac{K(0)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)}\right).$$

Numerador:

$$\hat{m}_h(X_i)\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - Y_i K(0) = \hat{m}_h(X_i)\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right) - Y_i \frac{K(0)\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)}{\sum_{\ell=1}^n K\left(\frac{X_\ell - X_i}{h}\right)}.$$

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

Entonces, podemos simplificar la fracción:

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

$$= \frac{\hat{m}_h(X_i) - Y_i w_{i,h}(X_i)}{1 - w_{i,h}(X_i)}.$$

Esto prueba la igualdad (2):

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

Prueba de la igualdad (3)**:**

Usando la igualdad (2), tenemos:

$$Y_i - \hat{m}_h^{-i}(X_i) = Y_i - \left(\frac{\hat{m}_h(X_i) - Y_i w_{i,h}(X_i)}{1 - w_{i,h}(X_i)}\right).$$

Simplificando, obtenemos:

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

Multiplicando ambos lados por $1 - w_{i,h}(X_i)$, tenemos:

$$(Y_i - \hat{m}_h^{-i}(X_i))(1 - w_{i,h}(X_i)) = Y_i(1 - w_{i,h}(X_i)) - (\hat{m}_h(X_i) - Y_i w_{i,h}(X_i)).$$

$$= Y_i - Y_i w_{i,h}(X_i) - \hat{m}_h(X_i) + Y_i w_{i,h}(X_i).$$

$$= Y_i - \hat{m}_h(X_i).$$

Por lo tanto,

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

Elevando al cuadrado ambos lados,

$$(Y_i - \hat{m}_h^{-i}(X_i))^2 = \left(\frac{Y_i - \hat{m}_h(X_i)}{1 - w_{i,h}(X_i)}\right)^2 = \frac{(Y_i - \hat{m}_h(X_i))^2}{(1 - w_{i,h}(X_i))^2}.$$

Finalmente, la función objetivo de validación cruzada se puede escribir como:

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

Esto prueba la igualdad (3):

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

Conclusión

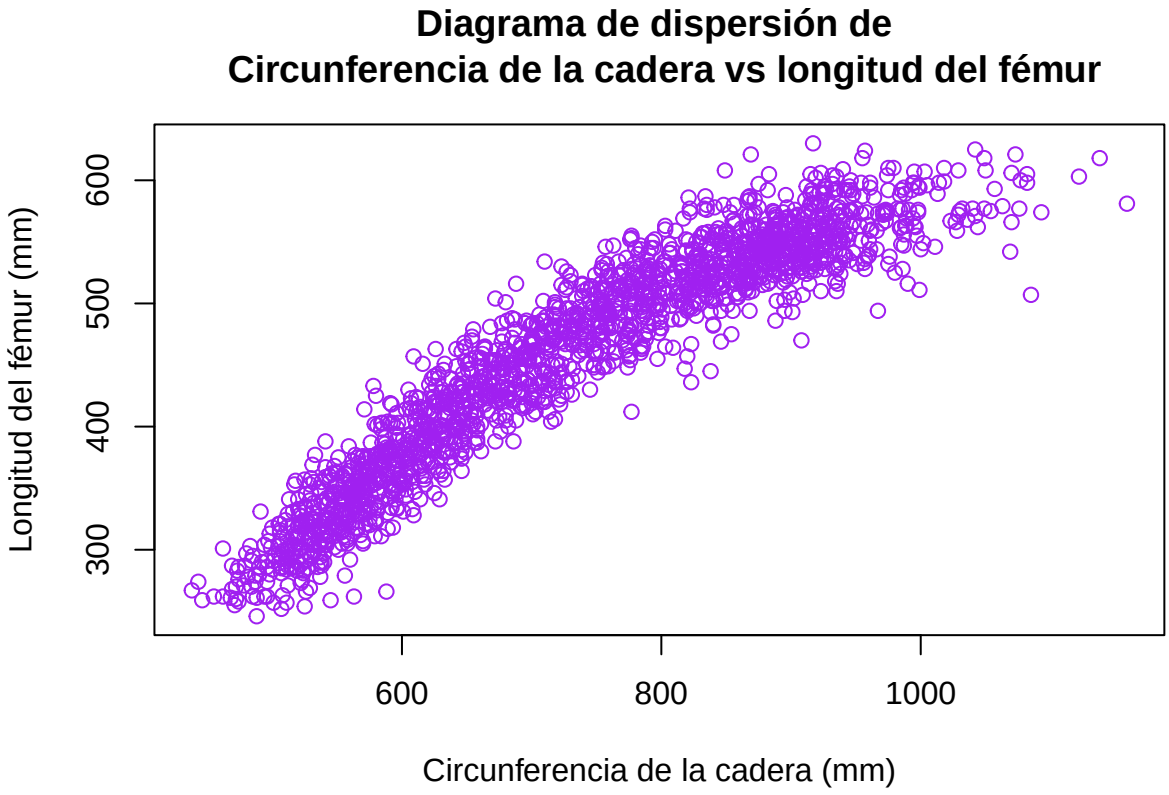
Demostramos que:

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

Esto muestra cómo simplificar el cálculo de la función objetivo de validación cruzada sin necesidad de recalcular el estimador para cada observación.

2)-a

```
library(ggplot2)
datos <- read.csv("individuals.csv", sep = ";")
mujeres <- subset(datos, SEX == 2)
mujeres <- subset(mujeres, HIP.CIRCUMFERENCE != 0 & BUTTOCK.KNEE.LENGTH != 0)
plot(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH, main = "Diagrama de dispersión de \n Circu
     xlab = "Circunferencia de la cadera (mm)", ylab = "Longitud del fémur (mm)", col = "purple")
```



Podemos observar una clara relación creciente entre las variables. Por la curvatura observada, en principio, parecería que no es simplemente una relación lineal. Además, no parece haber datos atípicos.

2)-b

```
mujeres_ordenado <- mujeres[order(mujeres$AGE.IN.MONTHS), ]
grupo_etario_1  <- mujeres_ordenado[1:466, ]
grupo_etario_2 <-  mujeres_ordenado[467:932, ]
grupo_etario_3 <-  mujeres_ordenado[933:1399, ]
grupo_etario_4<-  mujeres_ordenado[1400:1866, ]
grupos <- list(grupo_etario_1, grupo_etario_2, grupo_etario_3, grupo_etario_4)
medianas <- numeric(length(grupos))
for (i in 1:length(grupos)) {
  medianas[i] <- median(grupos[[i]]$HIP.CIRCUMFERENCE)
}
for (i in 1:4) {
  cat("Mediana grupo", i, ":" ,medianas[i], "\n")
}
```

```
## Mediana grupo 1 : 556
## Mediana grupo 2 : 672
## Mediana grupo 3 : 792
## Mediana grupo 4 : 903
```

Tenemos 1866 datos y lo dividimos en 4 partes iguales. Podmeos observar que los grupos están en orden

creciente de edad, y las medianas que obtuvimos también fueron valores crecientes (mientras más edad, más hip circumference.)

```
estimar_se_mediana <- function(datos, B = 1000) {
  theta_boot <- numeric(B)
  for (i in 1:B) {
    n <- length(datos)
    datos_boot <- sample(datos, n, replace = TRUE) #Generamos una muestra para bootstrap resampleando n
    theta_boot[i] <- median(datos_boot) # obtenemos su mediana y los guardamos
  }
  se_boot <- sqrt(mean((theta_boot - mean(theta_boot))^2))  # luego obtenemos el desvio de estas median
  return(se_boot)
}
intervalos_bootstrap <- list()
for (i in (1:4)){
  se_boot <- estimar_se_mediana(grupos[[i]]$HIP.CIRCUMFERENCE)
  intervalo_boot <- c(medianas[i] - 1.96 * se_boot , medianas[i] + 1.96 * se_boot)
  intervalos_bootstrap[[i]] <- intervalo_boot
}
for (i in 1:4) {
  cat("Intervalo de confianza bootstrap para grupo", i, ": [", round(intervalos_bootstrap[[i]][1], 2),
}
```

```
## Intervalo de confianza bootstrap para grupo 1 : [ 550.32 ,  561.68 ]
## Intervalo de confianza bootstrap para grupo 2 : [ 664.25 ,  679.75 ]
## Intervalo de confianza bootstrap para grupo 3 : [ 782.29 ,  801.71 ]
## Intervalo de confianza bootstrap para grupo 4 : [ 895.88 ,  910.12 ]
```

Para obtener los intervalos bootstrap el procedimiento es el siguiente:

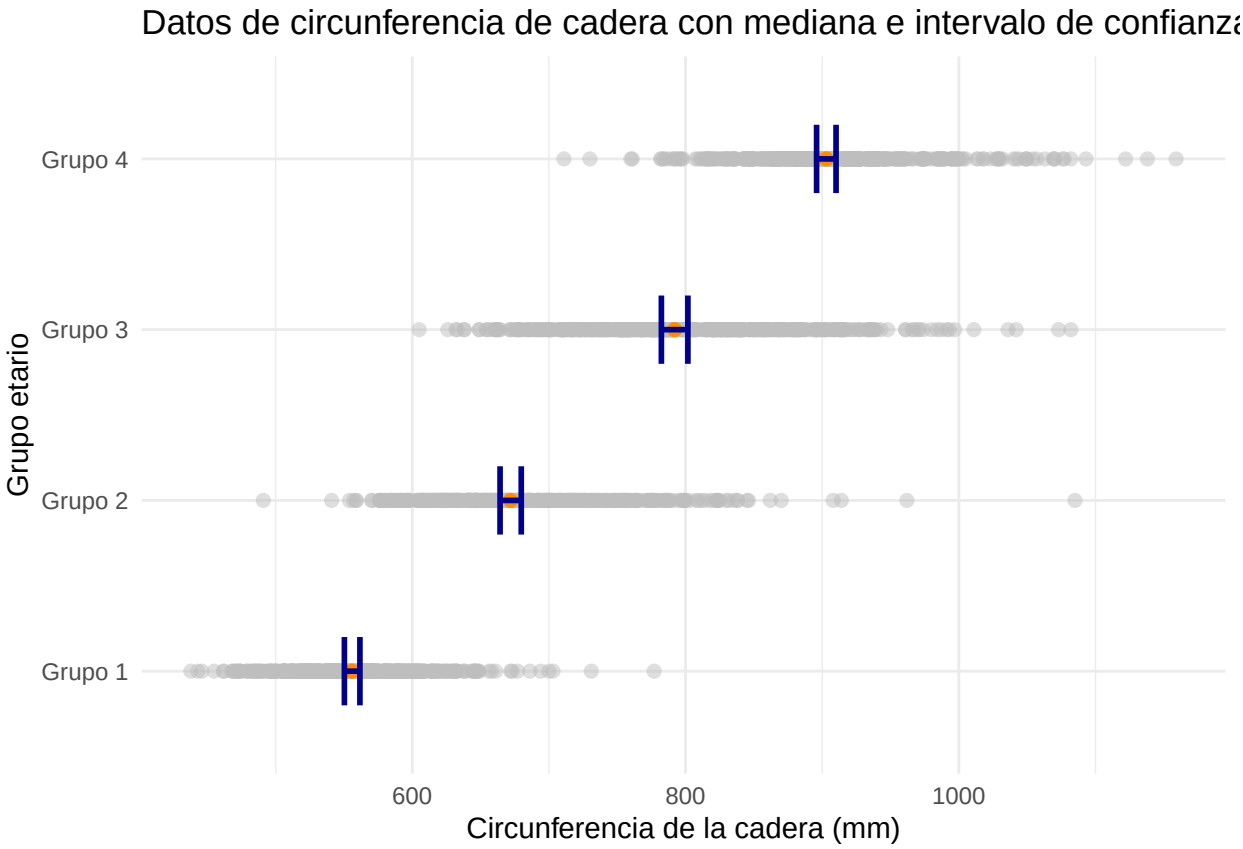
primero armamos una distribución que controlamos nosotros (en este caso resamplear los datos que ya tenemos con replacement) para simular la distribución subyacente.

usando esta distribución, hacemos varios experimentos y recopilamos el valor del estimador para cada uno.

luego, para estimar el desvio estandar del estimador, tomamos el desvio estandar del resultado de los experimentos (En vez de estar estimando el desvio del estimador sobre la distribución real, lo estamos haciendo sobre la distribución bootstrap. La calidad de esta estimación depende de la calidad de esta distribución simulada).

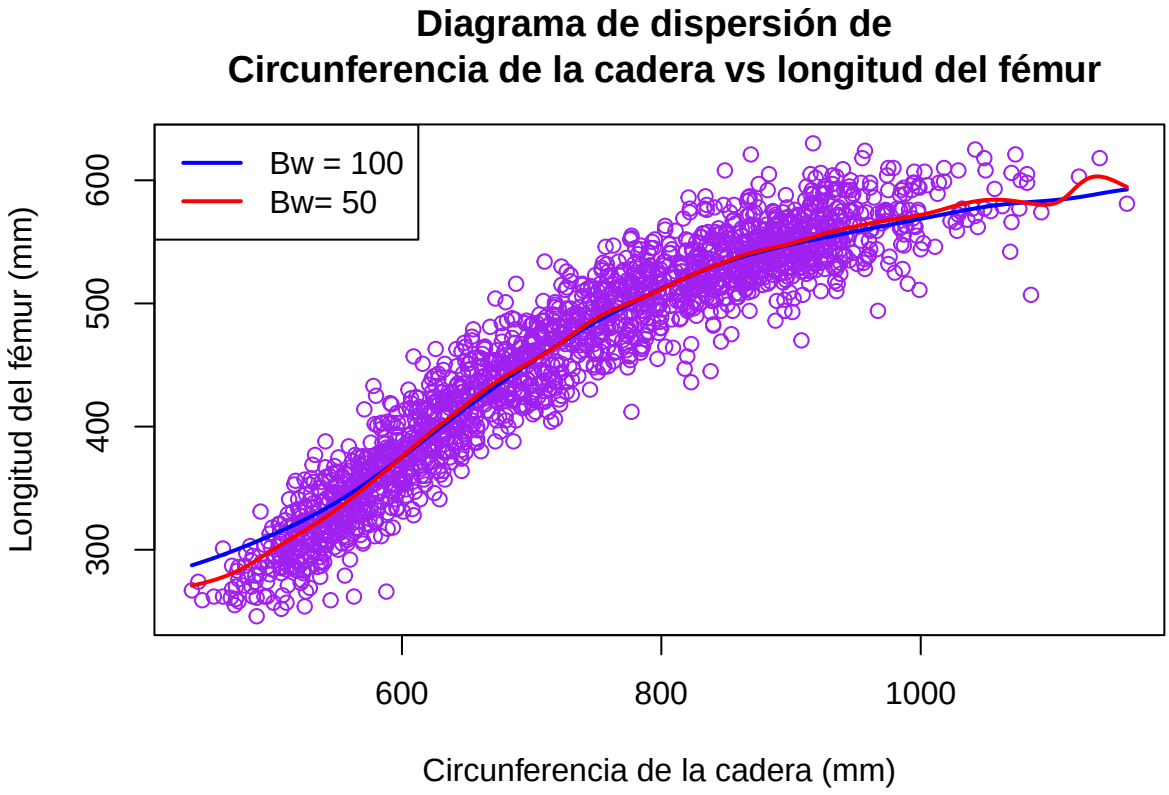
```
estadisticas <- data.frame(
  Grupo = c("Grupo 1", "Grupo 2", "Grupo 3", "Grupo 4"),
  Mediana = medianas,
  IC_lower = sapply(intervalos_bootstrap, `[`, 1),
  IC_upper = sapply(intervalos_bootstrap, `[`, 2)
)

estadisticas$label_pos <- estadisticas$IC_upper + 10  # Posición de la etiqueta para las barras de erro
ggplot() +
  geom_point(data = grupo_etario_1, aes(x = HIP.CIRCUMFERENCE, y = rep("Grupo 1", nrow(grupo_etario_1))
  geom_point(data = grupo_etario_2, aes(x = HIP.CIRCUMFERENCE, y = rep("Grupo 2", nrow(grupo_etario_2))
  geom_point(data = grupo_etario_3, aes(x = HIP.CIRCUMFERENCE, y = rep("Grupo 3", nrow(grupo_etario_3))
  geom_point(data = grupo_etario_4, aes(x = HIP.CIRCUMFERENCE, y = rep("Grupo 4", nrow(grupo_etario_4))
  geom_point(data = estadisticas, aes(x = Mediana, y = Grupo), color = "darkorange", size = 2) +
  geom_errorbarh(data = estadisticas, aes(xmin = IC_lower, xmax = IC_upper, y = Grupo), height = 0.4, co
```

```
  labs(title = "Datos de circunferencia de cadera con mediana e intervalo de confianza por grupo etario
       x = "Circunferencia de la cadera (mm)",
       y = "Grupo etario") +
  theme_minimal()
```



2)-c i

```
ajuste_bw_100 <- ksmooth(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH, kernel = "normal", ban
ajuste_bw_50 <- ksmooth(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH, kernel = "normal", bandw
plot(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH,
     main = "Diagrama de dispersión de \n Circunferencia de la cadera vs longitud del fémur",
     xlab = "Circunferencia de la cadera (mm)",
     ylab = "Longitud del fémur (mm)",
     col = "purple")
lines(ajuste_bw_100$x, ajuste_bw_100$y, col = "blue", lwd = 2)
lines(ajuste_bw_50$x, ajuste_bw_50$y, col = "red", lwd = 2)
legend("topleft",legend = c("Bw = 100","Bw= 50"), col = c("blue","red"),lwd = 2)
```



¿Que sugiere el grafico obtenido? ¿Cual de las dos ventanas usaría?

El grafico obtenido sugiere que ambos ajustes parecen seguir bastante bien a los datos en las zonas con muchas observaciones.

En los extremos, al haber menos datos, sucede algo interesante.

Por un lado, en el extremo izquierdo el estimador con la ventana más grande parece estar sesgado hacia arriba. Esto tiene sentido, ya que los datos se cortan abruptamente, y en el borde todos los datos cercanos son más grandes porque sólo los hay por la derecha. el estimador con la ventana chica no parece tener este problema, al captar sólo datos muy locales no se sesga, y además, como justo estos datos parecen concentrarse por la izquierda, termina estimando muy bien por este extremo.

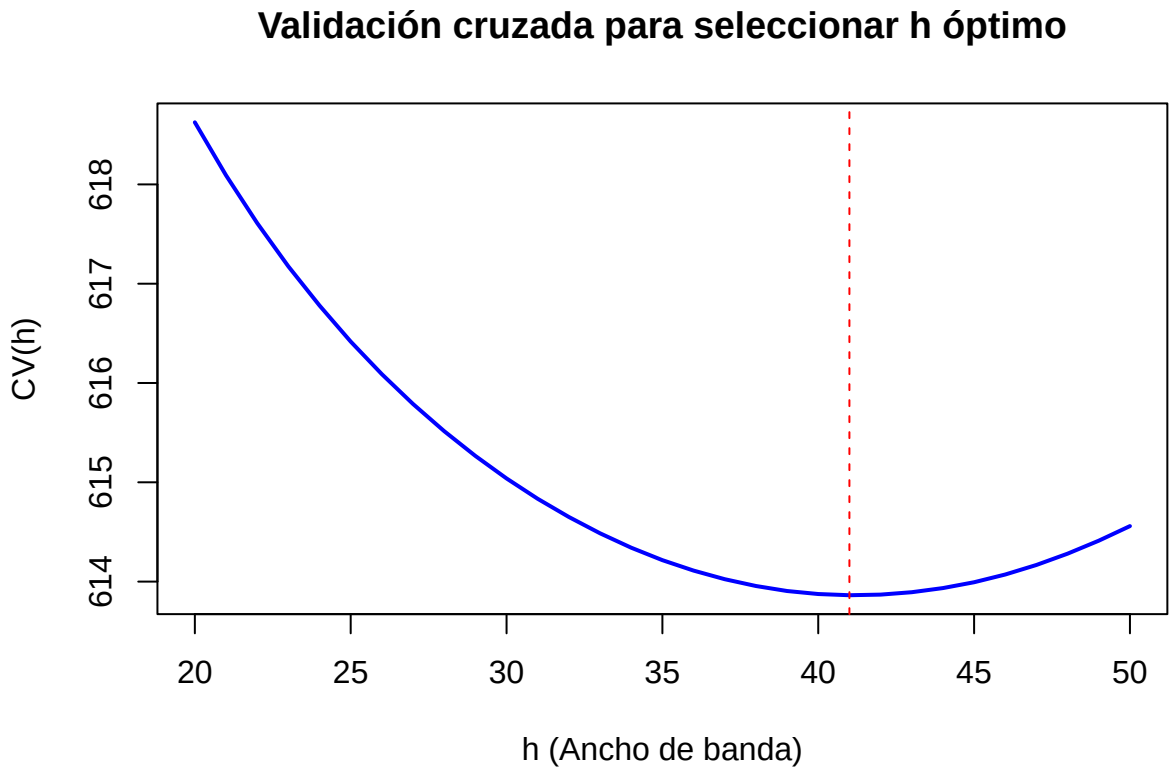
Por otro lado, en el extremo derecho sucede lo contrario, los datos se van desvaneciendo lentamente, y creciendo menos agresivamente, por lo que el estimador con la ventana grande no tiene problemas de sesgo, en cambio, el estimador con la ventana chica comienza a sufrir la escasez de datos, y a sobreajustarse a los pocos que tiene cerca de cada punto.

si tuvieramos que elegir un estimador ideal para este caso, sería uno que estime con ventana chica por izquierda, que hay menos variabilidad, y por derecha con el de ventana grande. Igualmente, por cuestiones de simplicidad y que hay que elegir uno, decidimos que el de ventana más chica va a ser confiable mientras no estimemos en valores muy extremos.

2)-c ii

```
f_obj<- function(h, x, y) {
  cv <- rep(0, length(y))
```

```
  for (i in 1:length(y)) {
    xi <- x[-i]
    yi <- y[-i]
    m_estimado <- ksmooth(xi, yi, kernel = "normal", bandwidth = h, x.points = x[i])
    cv[i] <- (y[i] - m_estimado$y)^2
  }
  return(mean(cv))
}
h_optimos <- seq(20, 50, by = 1)
valores_del_error_h_optimo <- sapply(h_optimos, f_obj, x = mujeres$HIP.CIRCUMFERENCE, y = mujeres$BUTTO
h_optimo <- h_optimos[which.min(valores_del_error_h_optimo)]
plot(h_optimos, valores_del_error_h_optimo, type = "l", col = "blue", lwd = 2,
     xlab = "h (Ancho de banda)",
     ylab = "CV(h)",
     main = "Validación cruzada para seleccionar h óptimo")
abline(v = h_optimo, col = "red", lty = 2)
```



podemos ver que la ventana óptima da 41

2)-c iii

implementemos nwsmooth y una función para calcular el ECM para cada ventana usando cross validation

```
nwsmooth <- function(punto, datos_x, datos_y, ancho_banda) {
  # K es la función de núcleo normal
```

```
  K <- function(u) {
    return(dnorm(u))
  }

  # Calcular los pesos
  pesos <- K((datos_x - punto) / ancho_banda)

  # Calcular el numerador y denominador
  numerador <- sum(datos_y * pesos)
  denominador <- sum(pesos)

  # Retornar el estimador de Nadaraya-Watson
  if (denominador == 0) {
    return(NA) #sin esto se nos rompía
  } else {
    return(numerador / denominador)
  }
}
nwsmooth_MSE <- function(ancho_banda, datos_x, datos_y) {
  cv <- rep(0, length(datos_y))
  for (i in 1:length(datos_y)) {
    xi <- datos_x[-i]
    yi <- datos_y[-i]
    m_estimado <- nwsmooth(datos_x[i], xi, yi, ancho_banda)
    cv[i] <- (datos_y[i] - m_estimado)^2
  }
  return(mean(cv, na.rm = TRUE)) # Promedio del ECM, ignorando los NA
}
```

ahora calculemos la ventana óptima

```
y = mujeres$BUTTOCK.KNEE.LENGTH
x = mujeres$HIP.CIRCUMFERENCE
ventanas <- seq(20, 50, by = 1)
cv_errors <- rep(0, length(ventanas))
for (j in seq_along(ventanas)) {
  h <- ventanas[j]
  cv_error <- 0
  for (i in 1:length(y)) {
    xi <- x[-i]
    yi <- y[-i]
    m_estimado <- nwsmooth( x[i],xi, yi,h)
    cv_error <- cv_error + (y[i] - m_estimado)^2
    cv_errors[j] <- cv_error/length(y)
    }
  cv_errors[j] <- cv_error/length(y)
}
h_optimo_nwmooth <- ventanas[which.min(cv_errors)]
print(h_optimo_nwmooth)
```

```
## [1] 20
```

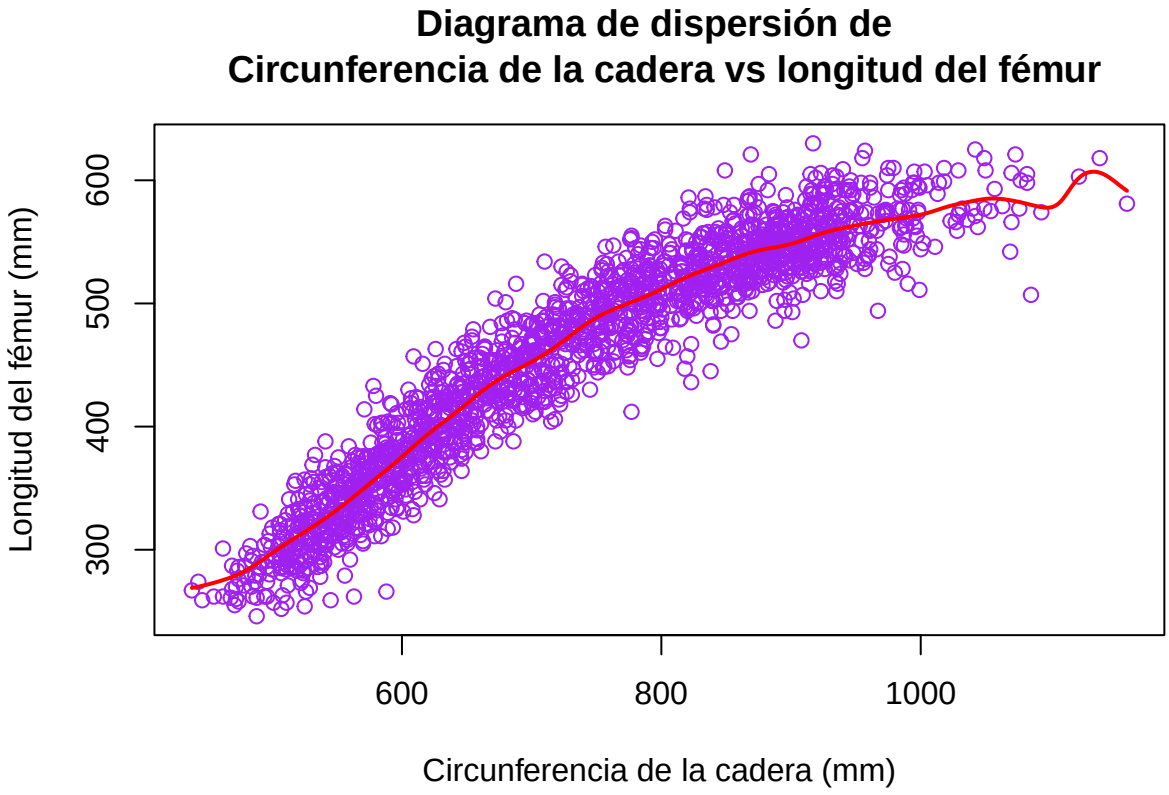
vemos que la ventana optima da 20, que es el extremo de la grilla. Propongamos entonces una grilla centrada en 20 para buscar el mejor valor:

```r
y = mujeres$BUTTOCK.KNEE.LENGTH
x = mujeres$HIP.CIRCUMFERENCE
ventanas <- seq(5, 30, by = 1)
cv_errors <- rep(0, length(ventanas))
for (j in seq_along(ventanas)) {
  h <- ventanas[j]
  cv_error <- 0
  for (i in 1:length(y)) {
    xi <- x[-i]
    yi <- y[-i]
    m_estimado <- nwsmooth( x[i],xi, yi,h)
    cv_error <- cv_error + (y[i] - m_estimado)^2
    cv_errors[j] <- cv_error/length(y)
    }
  cv_errors[j] <- cv_error/length(y)
}
h_optimo_nwmooth <- ventanas[which.min(cv_errors)]
print(h_optimo_nwmooth)
```

```
## [1] 15
```

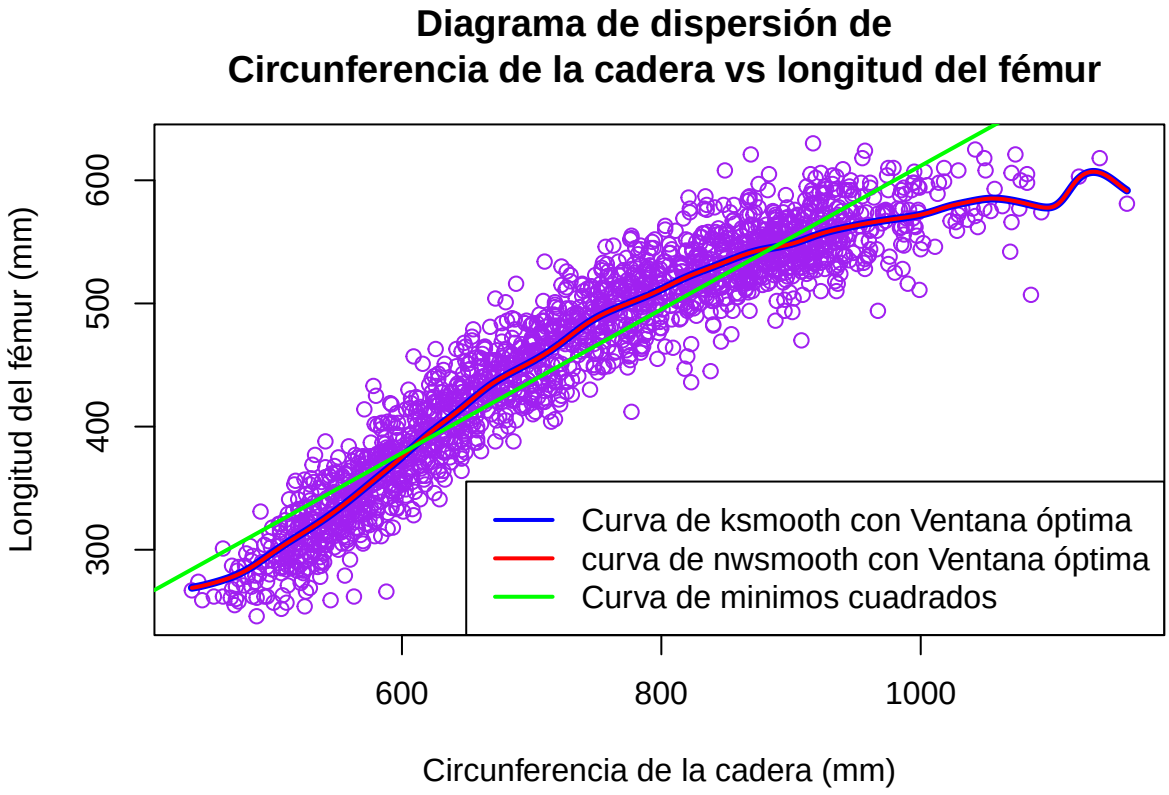
La ventana óptima parece ser 15, hagamos el gráfico del estimador con esta ventana

```r
plot(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH, main = "Diagrama de dispersión de \n Circu
     xlab = "Circunferencia de la cadera (mm)", ylab = "Longitud del fémur (mm)", col = "purple")


grilla <- seq(min(mujeres$HIP.CIRCUMFERENCE), max(mujeres$HIP.CIRCUMFERENCE), length.out=500)

# Calcular los valores suavizados para cada punto en x
y_smooth <- sapply(grilla, function(x0) nwsmooth(x0, mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LE


lines(grilla, y_smooth, col="red", lwd=2)
```



2)-c iv

```
ajuste_bw_optimo <- ksmooth(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH, kernel = "normal", l
ajuste_lm <- lm(BUTTOCK.KNEE.LENGTH ~ HIP.CIRCUMFERENCE, data = mujeres)
plot(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH,
     main = "Diagrama de dispersión de \n Circunferencia de la cadera vs longitud del fémur",
     xlab = "Circunferencia de la cadera (mm)",
     ylab = "Longitud del fémur (mm)",
     col = "purple")
lines(ajuste_bw_optimo$x, ajuste_bw_optimo$y, col = "blue", lwd = 4)
lines(grilla, y_smooth, col="red", lwd=2)
abline(ajuste_lm, col = "green", lwd = 2)

legend("bottomright",legend = c("Curva de ksmooth con Ventana óptima","curva de nwsmooth con Ventana óp
```



Podemos observar que las curvas de ksmooth y nwsmooth dan iguales, por lo que pareciera que implementamos nuestra propia versión de ksmooth

2)-d i/ii

Esta función calculará el estimador lineal local utilizando el método de mínimos cuadrados ponderados por un núcleo normal. Utilizaremos la fórmula matricial mencionada en el texto para encontrar el intercepto y la pendient, que minimizan la suma ponderada de los residuos cuadrados.

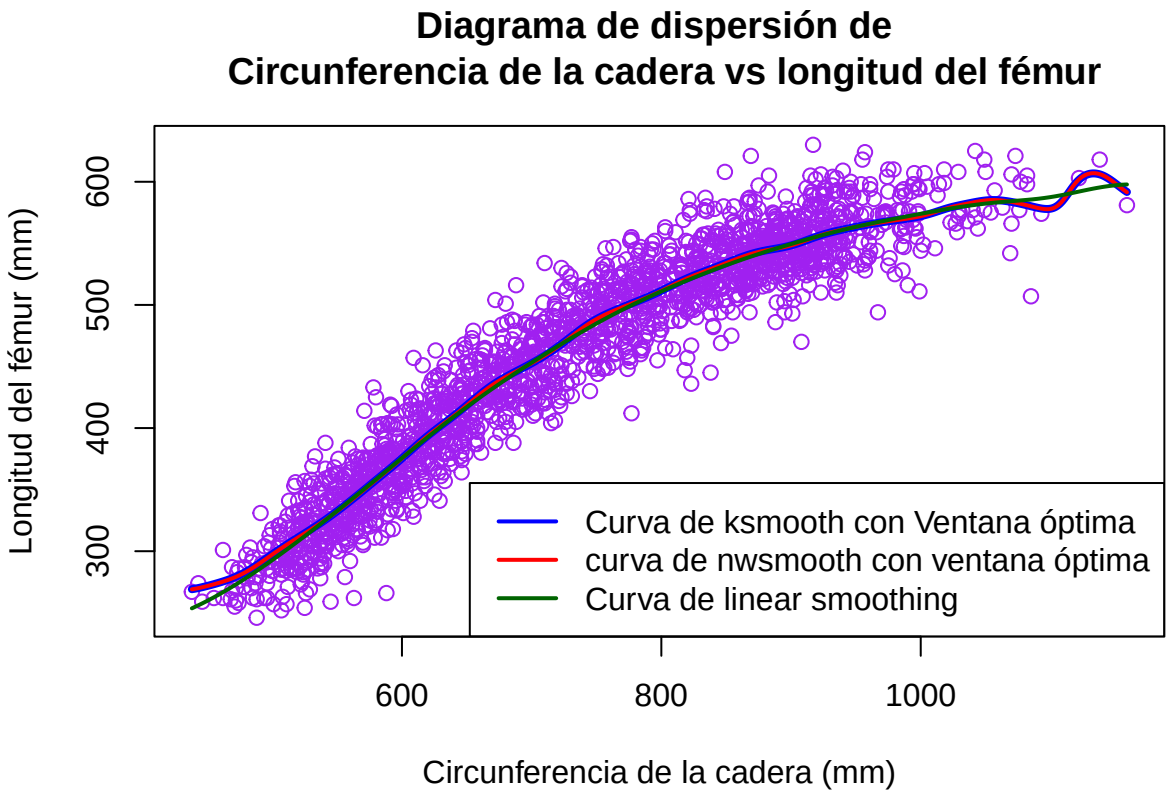
En el contexto de la regresión local (como Nadaraya-Watson o regresión local lineal), se utilizan pesos basados en un núcleo (kernel) que asigna mayor peso a las observaciones cercanas al punto donde se realiza la estimación y menor peso a las observaciones más lejanas. Esto ayuda a que las observaciones más cercanas tengan una influencia mayor en la estimación local.

```
# Definir la función linearsmooth
linearsmooth <- function(x0,x, y, h) {
  n <- length(x)
  K <- dnorm((x0 - x) / h)  # Kernel normal en el punto x0
  X <- cbind(1, x)
  weights <- diag(K)
  coef <- solve(t(X) %*% weights %*% X) %*% t(X) %*% weights %*% y
  ordenada <- coef[1]
  pendiente <- coef[2]
  return(ordenada + x0*pendiente)
}

grilla_lsmooth <- seq(min(mujeres$HIP.CIRCUMFERENCE), max(mujeres$HIP.CIRCUMFERENCE), length.out=500)
```

```r
# Calcular el estimador para cada x
valores_lsmooth <- sapply(grilla, function(x0) linearsmooth(x0, mujeres$HIP.CIRCUMFERENCE, mujeres$BUTT
```

2)-d iii

```r
plot(mujeres$HIP.CIRCUMFERENCE, mujeres$BUTTOCK.KNEE.LENGTH,
     main = "Diagrama de dispersión de \n Circunferencia de la cadera vs longitud del fémur",
     xlab = "Circunferencia de la cadera (mm)",
     ylab = "Longitud del fémur (mm)",
     col = "purple")
lines(ajuste_bw_optimo$x, ajuste_bw_optimo$y, col = "blue", lwd = 4)
lines(grilla, y_smooth, col="red", lwd=2)
lines(grilla_lsmooth, valores_lsmooth, col="darkgreen", lwd=2)


legend("bottomright",legend = c("Curva de ksmooth con Ventana óptima","curva de nwsmooth con ventana óp
```



vemos que el nuevo estimador resuelve los problemas de los que hablabamos antes, ahora estima bien en todos lados, aún en los extremos.