## Estimación de densidades por núcleos

Hemos visto que la altura de un histograma en un punto cualquiera x puede escribirse como

$$\hat{f}_h(x) = (nh)^{-1}\sum_{i=1}^{n}\sum_{j} I(X_i \in B_j)I(x \in B_j)$$

donde Bj indica cada intervalo de clase: [x0 + (j-1)h, x0 +jh) , con j $\in$ Z. Si x0=0 resulta Bj = [(j-1)h, jh).

La sumatoria en j se reduce al único sumando en para el cual $x \in B_j$ , esto implica que j = j (x):

$$\hat{f}_h(x) = (nh)^{-1}\sum_{i=1}^{n} I(X_i \in B_x)$$

Para un conjunto de datos $x_1,\ldots,x_n$ tendremos

$$\hat{f}_h(x) = (nh)^{-1}\sum_{i=1}^{n} I(x_i \in B_x)$$

$$= (nh)^{-1}\sum_{i=1}^{n} I(h/2 \leq | x - \tilde{x}_i | < h/2)$$

siendo $\tilde{x}_i$ el centro del intervalo al cual pertenece $x_i$
Luego



$$\hat{f}_h(x) = (nh)^{-1}\sum_{i=1}^{n} I(x - \tilde{x}_i, h)$$

donde $I(z,h)$ es la función indicadora del intervalo [-h/2,h/2)

Como estimador de una densidad el histograma ha sido criticado por:

- Se desperdicia información al reemplazar cada dato por el centro del intervalo de clase en el cual cae.
- En la mayoría de las situaciones la función de densidad es suave mientras que el histograma no lo es
- El comportamiento del estimador es dependiente de la longitud utilizada para intervalos (o equivalentemente cajas)

Rosemblat (1956), Whittle (1958) y Parzen (1962) desarrollaron un enfoque que eliminan las primeras dos dificultades. Utilizan funciones de núcleos suaves en vez de una caja y estas funciones están centradas directamente sobre cada observación. La estimación por núcleos (**kernel estimation**) tiene la forma:

$$\hat{f}_h(x) = (nh)^{-1}\sum_{i=1}^{n} w\left(\frac{x - x_i}{h}\right)$$

siendo w(x), el núcleo, una función no negativa, simétrica y centrada en cero, cuya integral es 1.

Por ejemplo w(x) puede ser la densidad normal estándar.

El parámetro h (ancho de la ventana) controla su suavidad y corresponde a la longitud del intervalo de clase en un histograma. Si h es demasiado pequeña la estimación resulta demasiado rugosa. Si h es demasiado grande la estimación resulta demasiado desparramada.

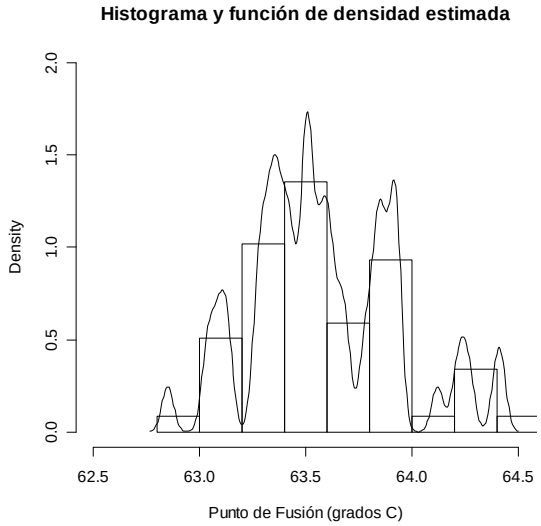



**Figura 3.** Histograma junto con una función de densidad estimada

La figura 3 muestra el histograma junto con una función de densidad estimada que, aunque es más suave que el histograma, es demasiado “rugosa”. Fue obtenida mediante las siguientes instrucciones:

```
hist(Cera $CERA,probability=T, ylim=c(0,2),
  xlim=c(62.5,64.5),main="",
  xlab="Punto de Fusión (grados C)")
  lines(density(Cera $CERA,n=300,width=0.11))
   title ( "Histograma y función de densidad
estimada")
```



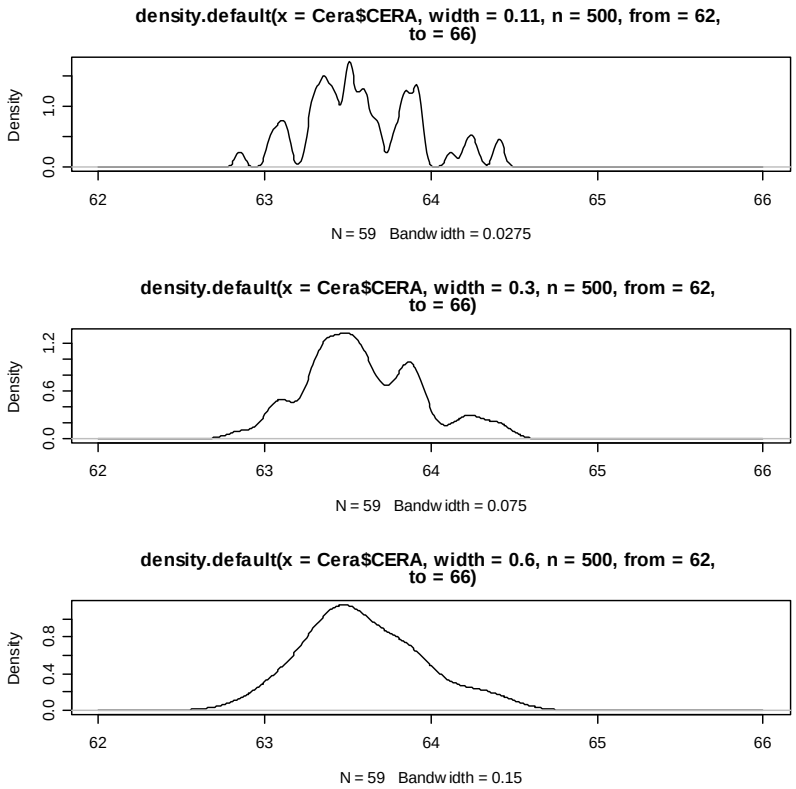



**Figura 4**. Estimaciones de densidad de probabilidad para los datos del punto de fusión de ceras naturales, para distintos tamaños del ancho de la ventana.

Los gráficos de la figura 4 fueron obtenidos mediante las siguientes instrucciones:

```
 par(mfrow=c(3,1))
 plot(density(Cera
$CERA,n=500,width=0.11,from=62,to=66))
```

```
plot(density(Cera$CERA,n=500,width=0.30,from=
62, to=66))

plot(density(Cera$CERA,n=500,width=0.60,from=
62, to=66))
```

**Sesgo y Varianza de los estimadores de densidad por núcleos**

$$Sesgo(\hat{f}_h(x)) = \frac{h^2}{2}\sigma_w^2 \, f''(x) + o(h^2) \qquad h \to 0$$

donde $\sigma_w^2 = \int x^2 w(x)dx$.

- El sesgo es cuadrático en *h*, debe elegirse pequeño para reducir el sesgo.
- El sesgo es proporcional a *f"*, la estimación por núcleos subestima la densidad (el sesgo es negativo) en los picos de la densidad verdadera *f* y la subestima en los valles, como vemos en el ejemplo siguiente.

Ejemplo:

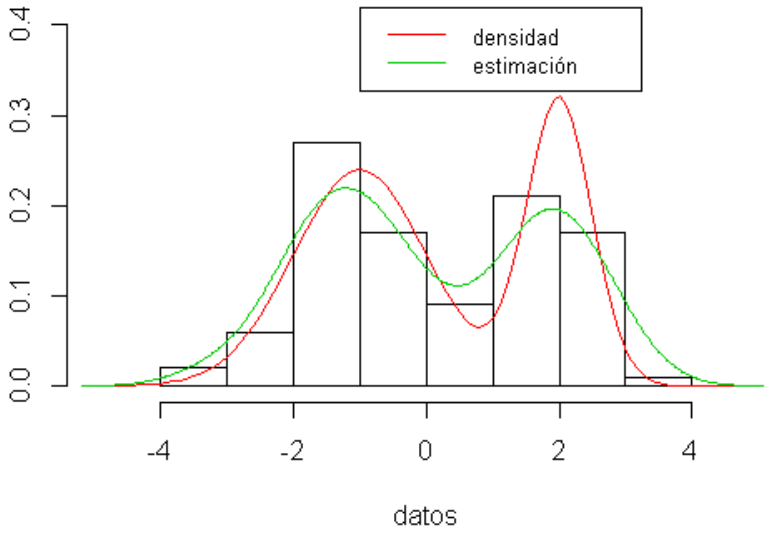
```
#el estimador por núcleos subestima en los picos y
sobreestima en los valles
fn1 <- rnorm(60,-1,1)
fn2 <- rnorm(40,2,0.5)

datos <- c(fn1,fn2)
densidad <- 0.6*dnorm(seq(-5,5,0.1),-1,1)+
0.4*dnorm(seq(-5,5,0.1),2,0.5)
```



```
par(mfrow=c(1,1))

hist(datos, probability=T, xlim=c(-5,5),
ylim=c(0,0.4), main="", ylab="")
lines(seq(-5,5,0.1),densidad,col=2)#func.de dens.
lines(density(datos,n=500),col=3)#estimación
legend(-1,.42, legend = c("densidad","estimación"),
col = 2:3,lty = 1,  cex = .8)
```




$$Var(\hat{f}_h(x)) = \frac{1}{nh} f(x)\alpha(w) + o((nh)^{-1}) \qquad nh \to \infty$$

donde $\alpha(w) = \int w^2(x)dx$.

- La varianza es inversamente proporcional al tamaño de la muestra como ocurre siempre. El término *nh* puede pensarse como controlando el tamaño muestral local.

- En forma análoga a lo que ocurre con el histograma, la varianza es proporcional a la altura de la densidad.

Igual que con los histogramas, tenemos que $h \to 0$ para reducir el sesgo, $\frac{1}{nh} \to 0$ para reducir la varianza y ambos dependen de la función de densidad que queremos estimar.

**Tipos de núcleos incorporados en R**

con `> density.default` se obtienen detalladamente las expresiones de los núcleos incorporados

```
> density
function (x, ...)
UseMethod("density")
<bytecode: 0x1042dd68>
<environment: namespace:stats>
> methods(density)
[1] density.default
> density.default
function (x, bw = "nrd0", adjust = 1, kernel
= c("gaussian",
    "epanechnikov", "rectangular",
"triangular", "biweight",
    "cosine", "optcosine"), weights = NULL,
window = kernel,
```



```
    width, give.Rkern = FALSE, n = 512, from,
to, cut = 3, na.rm = FALSE,
    ...)
{
    if (length(list(...)))
        warning("non-matched further
arguments are disregarded")
    if (!missing(window) && missing(kernel))
        kernel <- window
    kernel <- match.arg(kernel)
    if (give.Rkern)
        return(switch(kernel, gaussian = 1/(2
* sqrt(pi)), rectangular = sqrt(3)/6,
            triangular = sqrt(6)/9,
epanechnikov = 3/(5 * sqrt(5)),
            biweight = 5 * sqrt(7)/49, cosine
= 3/4 * sqrt(1/3 -
                2/pi^2), optcosine = sqrt(1 -
8/pi^2) * pi^2/16))
    if (!is.numeric(x))
        stop("argument 'x' must be numeric")
    name <- deparse(substitute(x))
    x <- as.vector(x)
    x.na <- is.na(x)
    if (any(x.na)) {
        if (na.rm)
            x <- x[!x.na]
        else stop("'x' contains missing
values")
    }
    N <- nx <- length(x)
    x.finite <- is.finite(x)
    if (any(!x.finite)) {
        x <- x[x.finite]
        nx <- length(x)
```

```
    }
    if (is.null(weights)) {
        weights <- rep.int(1/nx, nx)
        totMass <- nx/N
    }
    else {
        if (length(weights) != N)
            stop("'x' and 'weights' have
unequal length")
        if (!all(is.finite(weights)))
            stop("'weights' must all be
finite")
        if (any(weights < 0))
            stop("'weights' must not be
negative")
        wsum <- sum(weights)
        if (any(!x.finite)) {
            weights <- weights[x.finite]
            totMass <- sum(weights)/wsum
        }
        else totMass <- 1
        if (!isTRUE(all.equal(1, wsum)))
            warning("sum(weights) != 1  --
will not get true density")
    }
    n.user <- n
    n <- max(n, 512)
    if (n > 512)
        n <- 2^ceiling(log2(n))
    if (missing(bw) && !missing(width)) {
        if (is.numeric(width)) {
            fac <- switch(kernel, gaussian =
4, rectangular = 2 *
                sqrt(3), triangular = 2 *
sqrt(6), epanechnikov = 2 *
```

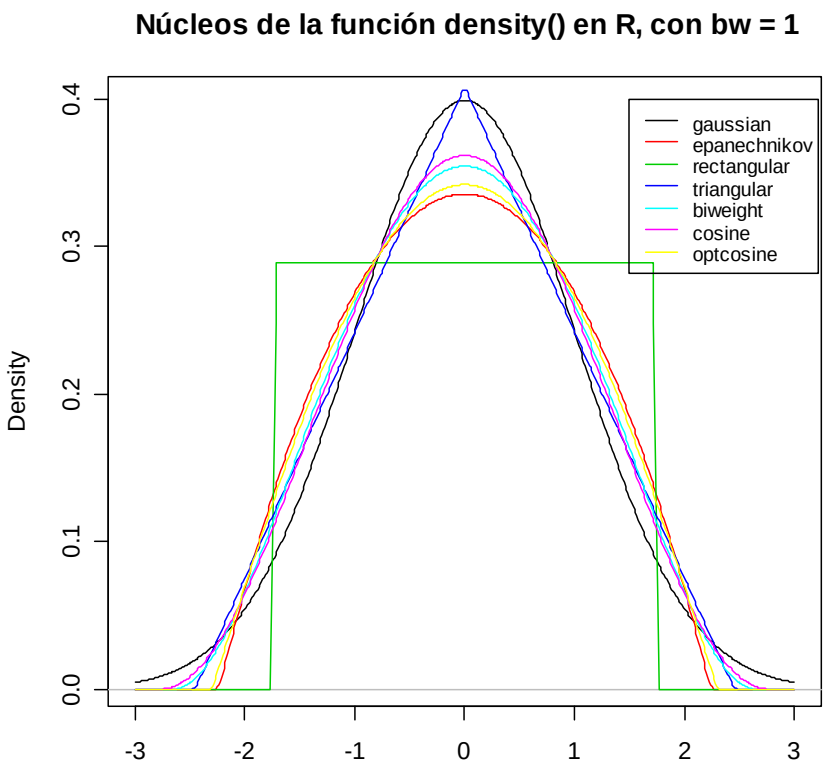


```
                sqrt(5), biweight = 2 *
sqrt(7), cosine = 2/sqrt(1/3 -
                2/pi^2), optcosine = 2/sqrt(1
- 8/pi^2))
            bw <- width/fac
        }
        if (is.character(width))
            bw <- width
    }
    if (is.character(bw)) {
        if (nx < 2)
            stop("need at least 2 points to
select a bandwidth automatically")
        bw <- switch(tolower(bw), nrd0 =
bw.nrd0(x), nrd = bw.nrd(x),
            ucv = bw.ucv(x), bcv = bw.bcv(x),
sj = , `sj-ste` = bw.SJ(x,
                method = "ste"), `sj-dpi` =
bw.SJ(x, method = "dpi"),
            stop("unknown bandwidth rule"))
    }
    if (!is.finite(bw))
        stop("non-finite 'bw'")
    bw <- adjust * bw
    if (bw <= 0)
        stop("'bw' is not positive.")
    if (missing(from))
        from <- min(x) - cut * bw
    if (missing(to))
        to <- max(x) + cut * bw
    if (!is.finite(from))
        stop("non-finite 'from'")
    if (!is.finite(to))
        stop("non-finite 'to'")
    lo <- from - 4 * bw
```

```
    up <- to + 4 * bw
    y <- .Call(C_BinDist, x, weights, lo, up,
n) * totMass
    kords <- seq.int(0, 2 * (up - lo),
length.out = 2L * n)
    kords[(n + 2):(2 * n)] <- -kords[n:2]
    kords <- switch(kernel, gaussian =
dnorm(kords, sd = bw),
        rectangular = {
            a <- bw * sqrt(3)
            ifelse(abs(kords) < a, 0.5/a, 0)
        }, triangular = {
            a <- bw * sqrt(6)
            ax <- abs(kords)
            ifelse(ax < a, (1 - ax/a)/a, 0)
        }, epanechnikov = {
            a <- bw * sqrt(5)
            ax <- abs(kords)
            ifelse(ax < a, 3/4 * (1 -
(ax/a)^2)/a, 0)
        }, biweight = {
            a <- bw * sqrt(7)
            ax <- abs(kords)
            ifelse(ax < a, 15/16 * (1 -
(ax/a)^2)^2/a, 0)
        }, cosine = {
            a <- bw/sqrt(1/3 - 2/pi^2)
            ifelse(abs(kords) < a, (1 +
cos(pi * kords/a))/(2 *
                a), 0)
        }, optcosine = {
            a <- bw/sqrt(1 - 8/pi^2)
            ifelse(abs(kords) < a, pi/4 *
cos(pi * kords/(2 *
                a))/a, 0)
```



```
        })
    kords <- fft(fft(y) * Conj(fft(kords)),
inverse = TRUE)
    kords <- pmax.int(0,
Re(kords)[1L:n]/length(y))
    xords <- seq.int(lo, up, length.out = n)
    x <- seq.int(from, to, length.out =
n.user)
    structure(list(x = x, y = approx(xords,
kords, x)$y, bw = bw,
        n = N, call = match.call(), data.name
= name, has.na = FALSE),
        class = "density")
}
<bytecode: 0x10aacc10>
<environment: namespace:stats>
>
```




**Obtenemos la figura anterior con las siguientes instrucciones**:

```
kernels <-
eval(formals(density.default)$kernel)
plot (density(0, bw = 1), xlab = "",
  main=" Núcleos de la función density() en
R, con bw = 1")
for(i in 2:length(kernels))
```



```
   lines(density(0, bw = 1, kernel =
kernels[i]), col = i)
legend(1.5,.4, legend = kernels, col =
seq(kernels), lty = 1, cex = .8, y.intersp =
1)
```

Núcleo cuadrático de Epanechnikov:

$$w(x) = \begin{cases} \frac{3}{4}(1-x^2) & \text{si } -1 \le x \le 1 \\ 0 & \text{fuera} \end{cases},$$

minimiza el error cuadrático medio integrado asintótico entre todos los núcleos no negativos con soporte compacto.

Función bicuadrada, introducida por Tukey en el contexto de estimación robusta:

$$w(x) = \begin{cases} (1-x^2)^2 & \text{si } -1 \le x \le 1 \\ 0 & \text{fuera} \end{cases}$$

**Diagramas de Tallo-Hoja (Stem-and-Leaf)**

Un diagrama de tallo-hoja (Tukey, 1977) es un histograma que conserva información numérica.

De manera similar al histograma permite ver el lote como un todo y advertir aspectos como:

- Cuán aproximadamente simétricos son los datos.

- Cuán dispersos están los valores.
- La aparición de valores inesperadamente más frecuentes.
- Si algunos valores están alejados del resto.
- Si hay concentraciones de valores.
- Si hay grupos separados.

Al utilizar los dígitos de los valores de los mismos datos, en vez de simplemente encerrando áreas, ofrece **ventajas**:

- Es más fácil de construir a mano.
- Facilita el ordenamiento de los datos.
- Permite, por lo tanto, hallar la mediana y otras medidas resumen basadas en el lote ordenado.
- Permite ver la distribución de los datos dentro de cada intervalo como patrones dentro de los datos.

Por ejemplo podríamos descubrir que todos los valores son múltiplos de 3.

- Facilita la identificación de una observación y la información que la acompaña.

| > stem(Cera$CERA) | > stem(Cera$CERA,scale=2) |
|---|---|
| The decimal point is 1 digit(s) to the left of the \| | The decimal point is 1 digit(s) to the left of the \| |



| | |
|---|---|
| 628 \| 5<br>630 \| 358033<br>632 \| 77001446669<br>634 \| 013350000113668<br>636 \| 0013689 88<br>638 \| 33466822223<br>640 \| 2<br>642 \| 147<br>644 \| 02 | 628 \| 5<br>629 \|<br>630 \| 358<br>631 \| 033<br>632 \| 77<br>633 \| 001446669<br>634 \| 01335<br>635 \| 0000113668<br>636 \| 0013689<br>637 \| 88<br>638 \| 334668<br>639 \| 22223<br>640 \|<br>641 \| 2<br>642 \| 147<br>643 \|<br>644 \| 02 |

En su apariencia global el diagrama se asemeja a un histograma (con ancho de intervalo igual a 0.2 $^0$C o 0.1 $^0$C respectivamente)

El 95% (56/59) de los puntos de fusión de la cera natural de abeja se encuentra entre 62.9 y 64.3.

## Profundidad

A cada dato se le puede asignar un *rango*, contando desde cada extremo en el lote ordenado. La *profundidad* es el menor de los dos valores.

Por ejemplo, en la figura 3, el 63.03 tiene rango 2 contando desde 62.85 hacia valores crecientes y rango 58 contando desde 64.42 hacia valores decrecientes.

**Figura 3**

```
> stem.leaf(cera$CERA)
1 | 2: represents 0.12
 leaf unit: 0.01
            n: 59
    1     628 | 5
          629 |
    4     630 | 358
    7     631 | 033
    9     632 | 77
   18     633 | 14466691010
   23     634 | 01335
  (10)    635 | 0000113668
   26     636 | 0013689
   19     637 | 88
   17     638 | 334668
   11     639 | 22223
          640 |
    6     641 | 2
    5     642 | 147
          643 |
    2     644 | 02
```

La primera columna indica la profundidad que se alcanza en cada fila,

excepto en la línea central que contiene la mediana, la máxima profundidad de los datos de esa fila.

La profundidad facilita hallar estadísticos de orden.

**Para obtener la columna de profundidad utilizamos la función**



**stem.leaf de la biblioteca aplpack que utiliza el Rcmdr para desplegar los diagramas tallo-hoja.**

**Tambien podemos utilizar la función stem.leaf.backback para obtener diagramas de 2 conjuntos de datos compartiendo los tallos**

**> library (aplpack)**
**> help(stem.leaf)**

```
stem.leaf(data, unit, m, Min, Max, rule.line = c("Dixon", "Velleman", "Sturges"),
    style = c("Tukey", "bare"), trim.outliers = TRUE, depths = TRUE,
    reverse.negative.leaves = TRUE, na.rm = FALSE, printresult = TRUE)
stem.leaf.backback(x,y, unit, m, Min, Max, rule.line = c("Dixon", "Velleman",
    "Sturges"), style = c("Tukey", "bare"), trim.outliers = TRUE,
    depths = TRUE, reverse.negative.leaves = TRUE, na.rm = FALSE,
    printresult=TRUE, show.no.depths = FALSE, add.more.blanks = 0,
    back.to.back = TRUE)
```

**Observación:** Los diagramas de tallo-hoja no son adecuados para datos cuyo rango tiene varios órdenes de magnitud, en esos casos es conveniente construir un diagrama tallo-hoja para el logaritmo de los datos.