8. Modelos de Regresión - Suavizados

El objetivo del ajuste de un modelo de regresión es encontrar una relación entre variables (X_i,Y_i), donde se considera que X_i explica el valor de Y_i, utilizando los pares observados (x_i,y_i). La relación de regresión es modelada, en general, como

$$Y_i = \mu(x_i) + \varepsilon_i, \qquad (1)$$

con $i = 1,\ldots,n$ y ε_i una variable aleatoria para indicar la variación de Y alrededor de la curva de regresión $\mu(.)$ dada por $E(Y_i / X = x) = \mu(x)$

Dado un conjunto de pares de datos, (x_i,y_i), correspondientes a observaciones de las variables (X,Y), el objetivo de un suavizado en un diagrama de dispersión (Scatter-plot smoothing) es obtener la relación funcional entre las variables dada por $\mu(.)$ cuando se utiliza el modelo (1) .

En regresión lineal simple se supone un modelo parmétrico lineal para la media de Y dado que X=x:

$$\mu(x) = a + bx$$

Ahora el problema de estimar la función $\mu(x)$ se reduce a estimar los coficientes a y b de la recta de regresión:

$$y = a + bx$$

y todas las observaciones participan en dicha estimación.

8.1 Regresión local

La *regresión local* es un enfoque de ajuste de curvas y superficies a datos mediante suavizados en los que el ajuste en x se realiza utilizando únicamente observaciones en un entorno de x. Al realizar una regresión local se utiliza una familia paramétrica al igual que en un ajuste de regresión global pero solamente se realiza el ajuste localmente.

En la práctica se realizan ciertas suposiciones

- sobre la función de regresión $\mu(.)$ tales como continuidad y derivabilidad de manera que pueda estar bien aproximada *localmente* por polinomios de un cierto grado.
- sobre la variabilidad de Y alrededor de la curva $\mu(.)$, por ejemplo variabilidad constante

Los métodos de estimación que resultan de este tipo de modelos es relativamente simple:

- Para cada punto x, se define un entorno (en alguna métrica en el espacio de diseño d-dimensional de las variables independientes).
- Dentro de ese entorno suponemos que la función regresora es aproximada por algun miembro de la familia paramétrica que podría ser de polinomios cuadráticos: $g(u) = a_0 + a_1(u - x) + a_2(u - x)^2$.
- Luego se estiman los parámetros con las observaciones en el entorno.
- El ajuste local es el la función ajustada evaluada en x.

Generalmente se incorpora una función de peso, w(u), para dar mayor peso a los valores x_i que se encuentran cerca de x. Los criterios de estimación dependen de los supuestos que se realicen sobre la distribución de las Y's. Si, por ejemplo, suponemos que tienen distribución Gaussiana con varianza constante tiene sentido utilizar el método de Cuadrados Mínimos.

De acuerdo con to Cleveland y Loader (1995) los modelos de regresión local se remontan al siglo 19. Estos autores proveen una revisión histórica del trabajo realizado a partir de ese momento. El trabajo moderno se inicia por los años 1950 en el contexto de estimación de densidades (Rosenblatt,1956; Parzen,1962) y dentro del contexto de regresión (Nadaraya (1964); Watson (1964)).

8.1.1 Suavizados por núcleos - Kernel Smoothers

El método más simple de suavizado es el suavizado por núcleos. Se fija un punto x en el dominio de la función regresora $\mu(.)$ y se define una *ventana* de suavizado alrededor de ese punto. Si x está en la recta Real, generalmente esa ventana es simplemente un intervalo de la forma $(x-h, x+h)$ donde h es un parámetro fijo llamado ancho de banda - *bandwidth*. Para la estimación consideramos puntos x_i dentro de esa ventana:

$$x - h < x_i < x + h,$$

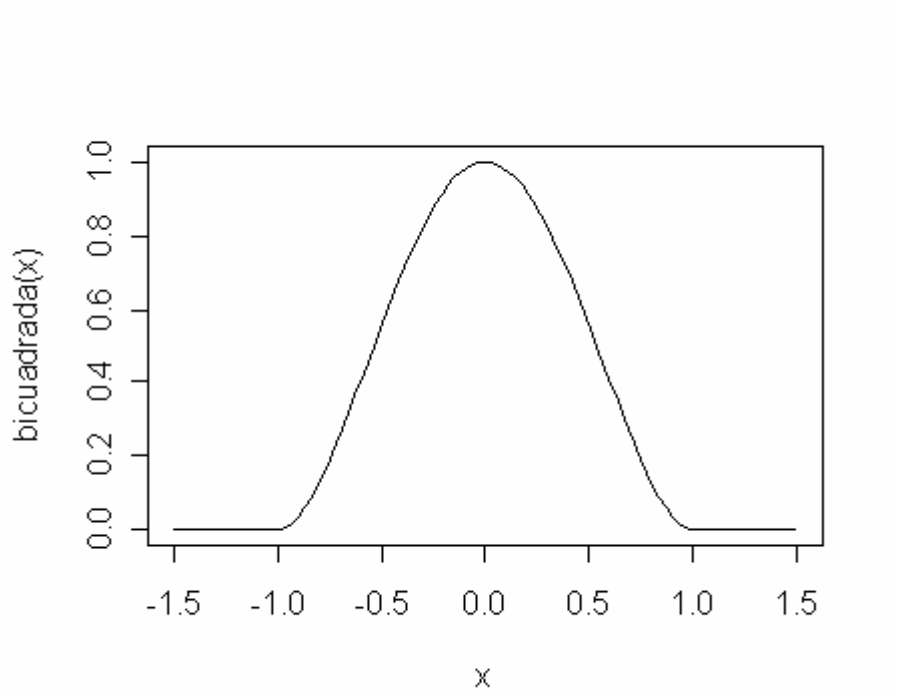
es decir que $| x - x_i | < h$

El estimador por núcleos (Nadaraya-Watson) es un promedio pesado de las observaciones y_i dentro de la ventana del suavizado:

$$\hat{\mu}(x) = \frac{\sum_{i=1}^{n} W(\frac{x_i - x}{h}) y_i}{\sum_{j=1}^{n} W(\frac{x_j - x}{h})} \qquad (2)$$

donde $W(.)$ es la función de pesos. La función de pesos es elegida de manera que la mayor parte del peso está sobre las observaciones cercanas al punto x sobre el que se está realizando el ajuste. Una posible elección es la función bicuadrada:

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



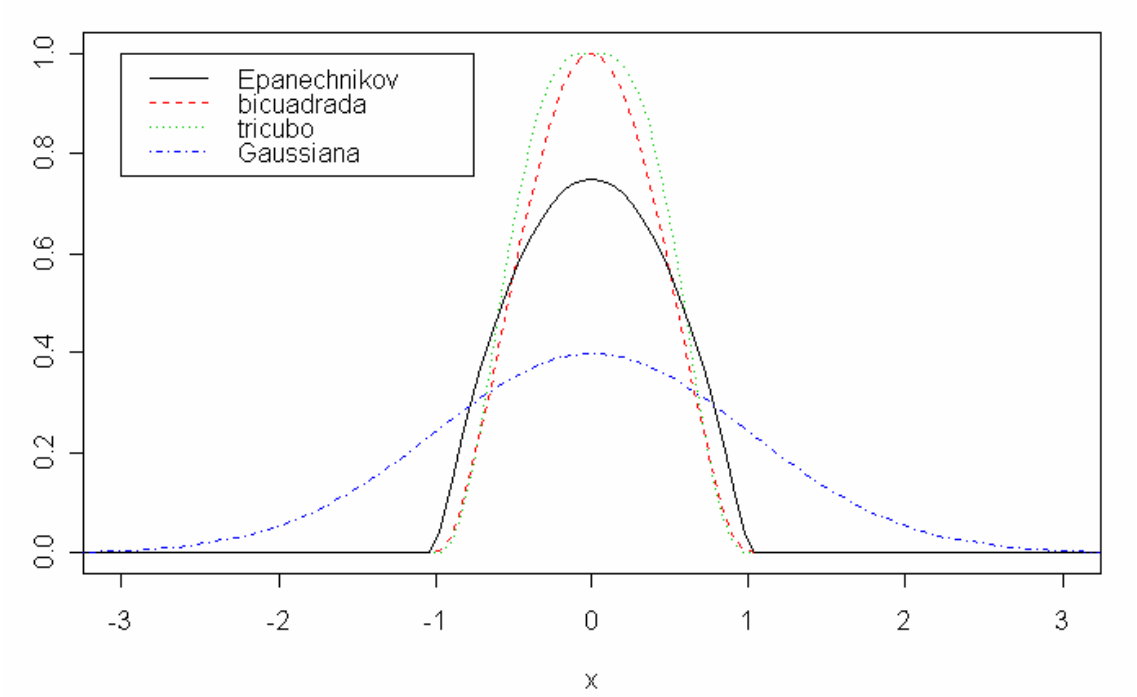
Otros núcleos utilizados son el núcleo cuadrático de *Epanechnikov*

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

la función tricubo

$$W(x) = \begin{cases} (1-|x|^3)^3 & \text{si } -1 \le x \le 1 \\ 0 & \text{fuera} \end{cases}$$

y la densidad Gaussiana. La figura siguiente compara los núcleos anteriores



La figura anterior fue obtenida en R mediante las siguientes funciones

```
bicuadrada <- function(x) {
y <- (1-x*x)*(1-x*x)*((x< 1)+0)*((x > -1)+0)
return(y)
                          }

Epanechnikov <- function(x) {
y <- 0.75*(1-x*x)*((x< 1)+0)*((x > -1)+0)
return(y)
                           }


tricubo <- function(x) {
y <- (1-abs(x*x*x))*(1-abs(x*x*x))*(1-abs(x*x*x))*((x< 1)+0)*
      ((x > -1)+0)
return(y)
                   }
```

y las siguientes instrucciones

```
> curve(tricubo(x),from=-3, to= 3,col=3,lty=3)
> curve(bicuadrada(x),add=TRUE, col= 2,lty=2)
> curve(Epanechnikov(x), add=TRUE,col = 1,lty=1)
> curve(dnorm(x), add=TRUE,col = 4,lty=4)
> legend(-3,1 , c( "Epanechnikov" , "bicuadrada" ,  "tricubo" ,
"Gaussiana" ), col=1:4 ,  lty=1:4)
```

8.1.2 Tipos de ventanas

Ancho de ventana fijo, ventana métrica

El parámetro h de la expresión (2) es el mismo para todos los puntos x.

Ancho de ventana variable. k- vecinos más cercanos

Necesitamos utilizar una notación más general: $h_k(x) = | x - x_{[k]} |$ donde $x_{[k]}$ es el k-ésimo x_i más cercano a x. En este caso cuanto más densamente estén distribuidos los puntos x_i alrededor de x tanto más pequeño será el ancho de la ventana. La longitud de la ventana es aleatoria.

Observaciones

- Las longitudes de ventana fijas tienden a mantener constante el sesgo del estimador, pero la varianza es inversamente proporcional a la densidad local.
- Para ventanas de k-vecinos más cercanos la varianza se mantiene constante pero el sesgo varía con la densidad local. El tratamiento de los empates ofrece una dificultad adicional.

8.2 Regresión polinomial local

La regresión polinomial local es una generalización del trabajo previo en estimación por núcleos. Más aún la estimación por núcleos corresponde al ajuste de un polinomio de grado cero, es decir una constante en un entorno.

El objetivo general de la regresión local es ajustar un polinomio de grado *p* al rededor de un punto utilizando los datos de un entorno. Esto incluye estimación por núcleos (*p*=0), regresión lineal local (*p*=1), etc .

El principio subyacente es que una función continua puede aproximarse bien por un polinomio de grado bajo. Por ejemplo una aproximación lineal está dada por:

$$\mu(x_i) \approx a_0 + a_1(x_i - x)$$

donde x_i se encontrará en dentro de un intervalo que contiene al punto para el cual se está realizando el ajuste, $x - h \le x_i \le x + h$. Una aproximación cuadrática es

$$\mu(x_i) \approx a_0 + a_1(x_i - x) + \frac{a_2}{2}(x_i - x)^2$$

Los polinomios locales pueden ajustarse utilizando mínimos cuadrados pesados localmente.

En el caso de una regresión lineal local los coeficientes estimdados $\hat{a}_0, \hat{a}_1$ se eligen de manera de minimizar

$$\sum_{i=1}^{n} W(\frac{x_i - x}{x})(Y_i - (a_0 + a_1(x_i - x)))^2$$

Se trata de un problema de estimación por cuadrados mínimos con pesos cuya solución tiene una expresión cerrada general y la solución numérica de las ecuaciones resultantes es inmediata.

El estimador lineal local es

$$\hat{\mu}(x) = \hat{a}_0$$

Para cada punto x se tiene una estimación diferente pues los pesos cambian y por lo tanto los estimadores $\hat{a}_0, \hat{a}_1$

Los coeficientes estimados se obtienen resolviendo las ecuacines normales:

$$\boldsymbol{X}^\top \boldsymbol{W} \left(Y - \boldsymbol{X} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} \right) = 0,$$

donde $\boldsymbol{X}$ es la *matriz de diseño*:

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1 - x \\ \vdots & \vdots \\ 1 & x_n - x \end{pmatrix}$$

tiene n filas y 2 columnas para la regresión lineal local.

$\boldsymbol{W}$ es una matriz diagonal con valores

$$W\left(\frac{x_i - x}{h}\right)$$

e

$$Y = \begin{pmatrix} Y_1 & \dots & Y_n \end{pmatrix}^\top$$

.

Cuando $\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}$ es inversible, se obtiene una expresión explícita de los estimadores:

$$\begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} = \left(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{Y}.$$

8.3 Regresión polinomial local utilizando R

Lowess y Loess

Los nombres "lowess" y "loess" provienen de "**l**ocally **we**ighted **s**catter plot **s**mooth," ya que ambos métodos provienen de regresiones pesadas localmente lineales para suavizar los datos. Podemos decir que se trata de suavizados para diagramas de dispersión, "scatter plot smoothers".

Por defecto las funciones **`lowess`** y **`loess`** realizan un ajuste localmente lineal y localmente cuadrático respectivamente.

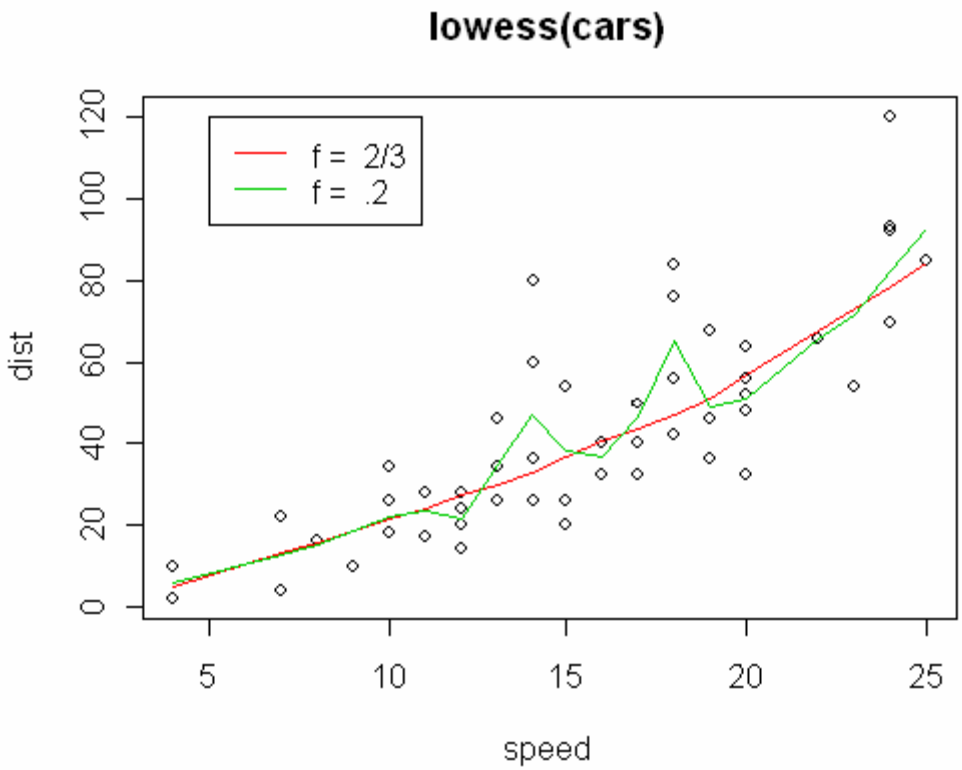
Del help del R tenemos

lowess: Scatter Plot Smoothing
Usage:
lowess(x, y = NULL, f = 2/3, iter = 3, delta = 0.01 * diff(range(xy$x[o])))
Arguments:
x, y: vectors giving the coordinates of the points in the scatter plot. Alternatively a single plotting structure can be specified.
f: the smoother span. This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness.
iter: the number of robustifying iterations which should be performed. Using smaller values of 'iter' will make 'lowess' run faster.
delta: values of 'x' which lie within 'delta' of each other are replaced by a single value in the output from 'lowess'. Defaults to 1/100th of the range of 'x'.
See Also:
'loess', a newer formula based version of 'lowess' (with different defaults!)

Veamos un ejemplo

Utilizamos un conjunto de datos incorporados al R **`cars`** que consiste en un data frame con la velocidad (**`speed`** en millas por hora) y la distancia que requieren para frenar (**`dist`** en pies). Los datos fueron registrados en 1920.

```
>   plot(cars, main = "lowess(cars)")
>   lines(lowess(cars), col = 2)
>   lines(lowess(cars, f=.2), col = 3)
>   legend(5, 120, c(paste("f = ", c("2/3", ".2"))), lty = 1, col = 2:3)
```



Veamos que nos da la nueva versión: loess

`loess`

Description:

Fit a polynomial surface determined by one or more numerical predictors, using local fitting.

Usage:

```
loess(formula, data, weights, subset, na.action, model = FALSE,
      span = 0.75, enp.target, degree = 2,
      parametric = FALSE, drop.square = FALSE, normalize = TRUE,
      family = c("gaussian", "symmetric"),
      method = c("loess", "model.frame"),
```

control = loess.control(...), ...)

Arguments: varios ...

.....

Detalles:

El ajuste se realiza localmente. Para span = alfa < 1 el entorno incluye una proporción alfa de puntos con una función de peso proporcional a

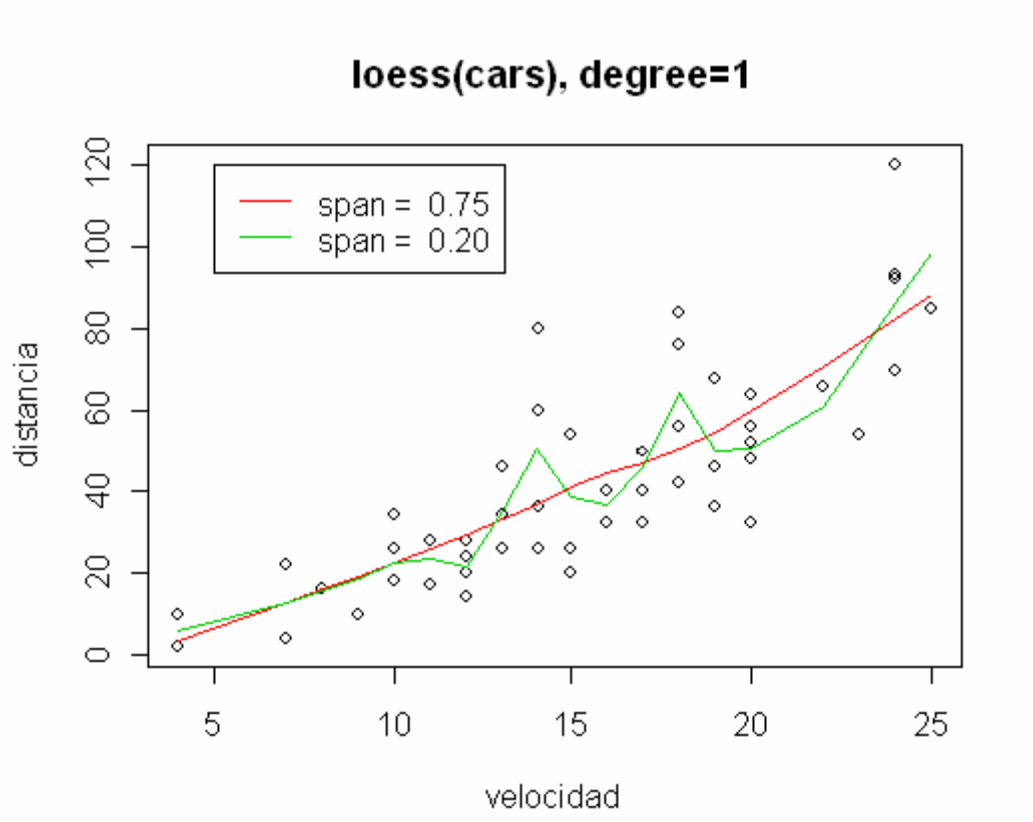
$$w_i = \left(1 - \left|\frac{x - x_i}{d(x)}\right|^3\right)^3$$

donde d(x) es la distancia máxima entre los puntos centrados en x y que corresponden a la proporción especificada.

Para la familia por defecto "gaussian" el ajuste se realiza por cuadrados mínimos pesados
Para 'family="symmetric"' se realizan unas pocas iteraciones de un M-estimador donde se utiliza la función Tukey biweight. Se debe saber que como el ajuste inicial es un ajuste de cuadrados mínimos, este puede no resultar en un ajuste muy resistente.

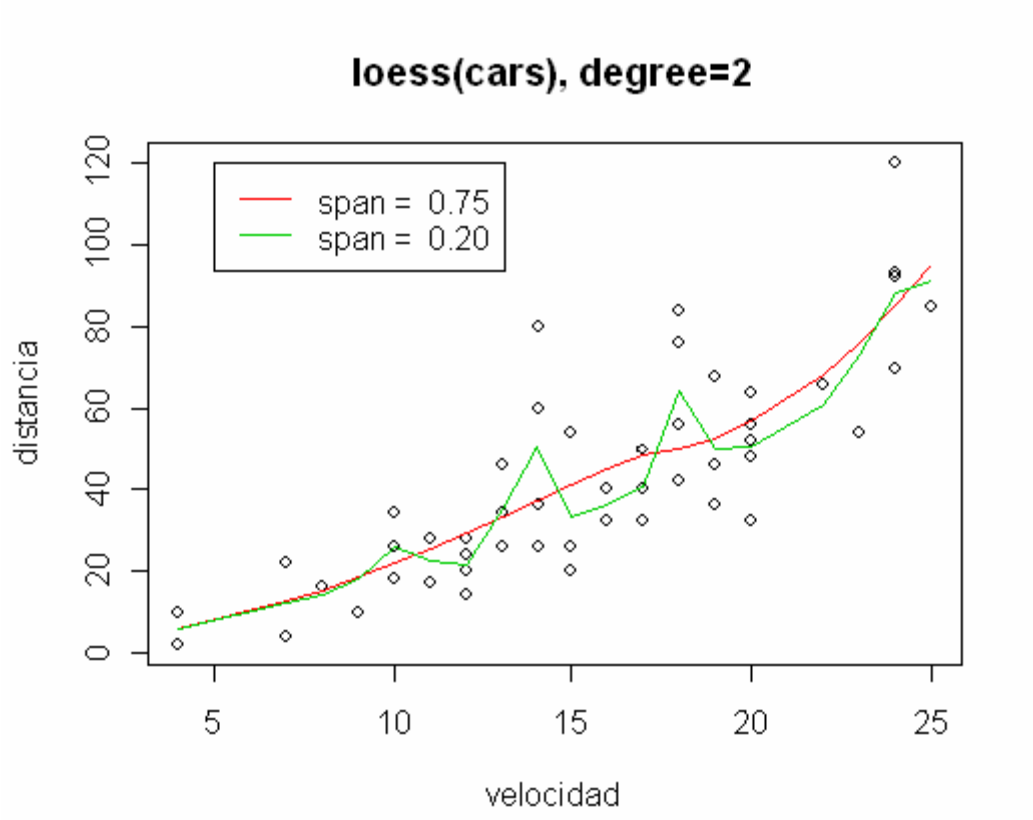
Veamos un ejemplo con `loess`

```
cars.lo <- loess(dist ~ speed, cars, span=0.75, degree=1)
cars.lo2 <- loess(dist ~ speed, cars, span=0.20, degree=1)

plot(cars$speed, cars$dist, xlab= "velocidad", ylab="distancia",
        main = "loess(cars), degree=1")
      lines(cars$speed, cars.lo$fit, col = 2)
      lines(cars$speed, cars.lo2$fit, col = 3)
legend(5, 120, c(paste("span = ", c("0.75", "0.20"))), lty = 1,
col = 2:3)
```



Veamos qué ocurre cuando realizamos un ajuste local con un polinomio de grado 2

```
cars.lo <- loess(dist ~ speed, cars, span=0.75, degree=2)
cars.lo2 <- loess(dist ~ speed, cars, span=0.20, degree=2)

plot(cars$speed, cars$dist, xlab= "velocidad", ylab="distancia",
       main = "loess(cars), degree=2")
      lines(cars$speed, cars.lo$fit, col = 2)
      lines(cars$speed, cars.lo2$fit, col = 3)
      legend(5, 120, c(paste("span = ", c("0.75", "0.20"))),
      lty = 1, col = 2:3)
```



Observaciones

- El suavizado es afectado fundamentalmente por el tamaño de la ventana.
- La función **`lowess`** realiza unicamente un suavizado en una dimensión, esto es que x e y son vectores.
- La función **`loess`** es más general permitiendo que el suavizado sea multidimensional.

8.4 M-estimadores

El método de mínimos cuadrados habitual minimiza la suma de cuadrados de los residuos r_i (diferencia entre el dato i observado y su valore ajustado) para estimar los parámetros de un modelo. Esto es hallar los valores que hacen

$$\text{mín} \sum_i r_i^2$$

Este método es muy sensible a la presencia de outliers. Los M-estimadores reducen este efecto reemplazando los cuadrados de los residuos por otra función. Esto lleva a determinar

$$\text{mín} \sum_i \rho(r_i)$$

donde ρ es una función simétrica, definida positiva y con un único mínimo. Es elegida de manera que sea menos creciente que el cuadrado. En vez de resolver directamente esta minimización puede implementarse un procedimiento de mínimos cuadrados iterados como veremos a continuación.

Sea $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ el vector de parámetros que deben ser estimados. El M- estimador de $\boldsymbol{\theta}$ basado en la función ρ es el vector $\boldsymbol{\theta}$ que es solución del siguiente sistema de *m* ecuaciones

$$\sum_i \psi(r_i) \frac{\partial r_i}{\partial \theta_j} = 0, \quad \text{para j} = 1, ..., \text{m} \tag{1}$$

donde $\psi(x) = \dfrac{d\rho(x)}{dx}$ es la llamada función de influencia

Si definimos la *función de peso*

$$w(x) = \frac{\psi(x)}{x},$$

el sistema de ecuaciones (1) puede escribirse como

$$\sum_i w(r_i) r_i \frac{\partial r_i}{\partial \theta_j} = 0, \quad \text{para j} = 1, ..., \text{m} \tag{2}$$

Este es exactamente el sistema de ecuaciones que se obtiene si resolvemos un problema de mínimos cuadrados pesado iterado (iterated reweighted least-squares problem)

$$\text{mín} \sum_i w(r_i^{(k-1)}) r_i^{\,2} = 0, \quad \text{para j} = 1, ..., \text{m}$$

donde el supraíndice k indica el número de iteración. El peso $w(r_i^{(k-1)})$ debe ser recalculado en cada caso.

Una una opción para la elección de la función ρ es la propuesta por Tukey donde la función de peso resulta ser la función bicuadrada:

	$\rho(x)$	$\psi(x)$	$w(x)$
Tukey $\begin{cases} \lvert x\rvert \le c \\ \lvert x\rvert > c \end{cases}$	$\begin{cases} \frac{c^2}{6}\left(1 - [1 - (x/c)^2]^3\right) \\ (c^2/6) \end{cases}$	$\begin{cases} x[1 - (x/c)^2]^2 \\ 0 \end{cases}$	$\begin{cases} [1 - (x/c)^2]^2 \\ 0 \end{cases}$