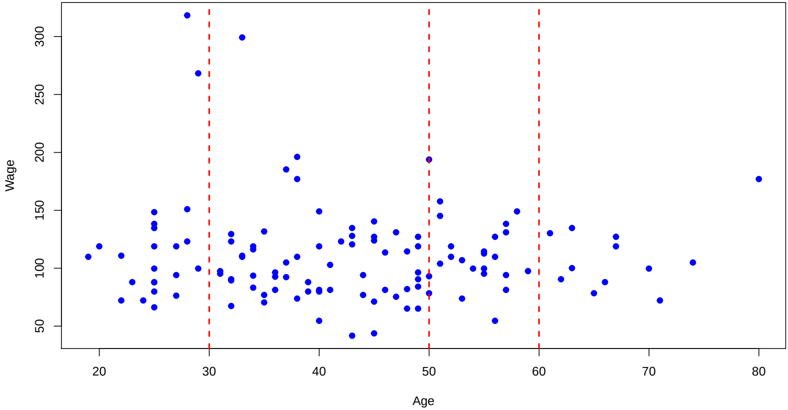
Clase 8 - Módulo 2: Introducción a la analítica

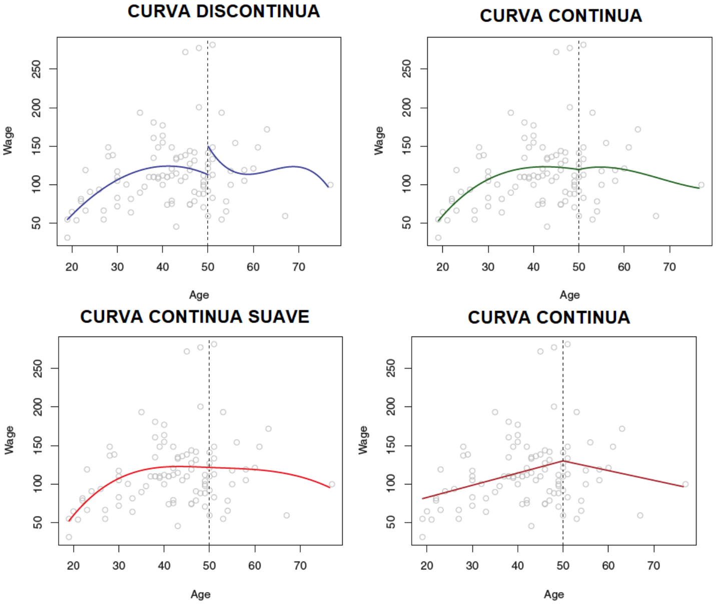
Mauricio Alejandro Mazo Lopera

Universidad Nacional de Colombia
Facultad de Ciencias
Escuela de Estadística
Medellín

El método de regresión por splines busca ajustar distintos polinomios sobre el rango de X en regiones separadas por puntos llamados **nodos**.



Ejemplos con un solo nodo en $Age = 50$



Splines cúbicos:

Un spline cúbico con k nodos se puede representar como:

$$Y_i = \beta_0 + \beta_1 b_1(X_i) + \beta_2 b_2(X_i) + \cdots + \beta_k b_k(X_i) + \epsilon_i$$

Existen varias formas de representar una base para un spline cúbico y entre estas la más directa comienza planteando un polinomio cúbico X, X^2, X^3 que va truncándose adicionando en cada nodo una función base dada por:

$$h(x,\xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3, & \text{si } x > \xi \\ 0, & \text{e.o.c.} \end{cases}$$

donde ξ es el nodo.

Splines cúbicos:

Así, si queremos ajustar un spline cúbico con k nodos, dados por $\xi_1, \xi_2, \ldots, \xi_k$, se debe plantear el modelo como:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \beta_{1h} h(X_i, \xi_1) + \cdots + \beta_{kh} h(X_i, \xi_k) + \epsilon_i$$

En el cual se deben estimar $k+4$ parámetros. Algunos paquetes estadísticos suelen designar estos con el nombre de grados de libertad y plantean que para un spline cúbico con k nodos, el número de grados de libertad es igual a $k+4$.

Splines cúbicos naturales:

Un spline cúbico natural se define como aquel que impone las condiciones en los límites inferior y superior del rango de la variable X.

Dichas condiciones consisten en asumir que antes del primer nodo y luego del último nodo, la función es lineal, es decir, de grado uno (una línea recta).

Esta condición busca producir mayor estabilidad en las fronteras del rango de X.

Lo primero que se debe tener en cuenta es que entre mayor sea el número de nodos, mayor será la flexibilidad del modelo y entre menos nodos, más estabilidad. Ambas características en exceso pueden ser perjudiciales (¿Por qué?) y por tanto se debe buscar un equilibrio entre ambas.

La manera más utilizada de ubicar los nodos es considerando, de manera uniforme, los cuantiles de los datos en X. Así, por ejemplo, si se seleccionan 3 nodos y no se especifica donde ubicarlos, la manera más directa de ubicarlos es considerar:

$$\begin{aligned} \text{Primer nodo} &\longrightarrow \text{Percentil } 25\,\% \text{ de X} \\ \text{Segundo nodo} &\longrightarrow \text{Percentil } 50\,\% \text{ de X} \\ \text{Tercer nodo} &\longrightarrow \text{Percentil } 75\,\% \text{ de X} \end{aligned}$$

Una forma **subjetiva** de encontrar el número de nodos y su ubicación podría ser a través de un análisis visual, en donde se consideran distintas opciones y se selecciona la curva que “mejor se ajuste”.

Seleccionando el número de nodos y su ubicación:

Una forma **subjetiva** de encontrar el número de nodos y su ubicación podría ser a través de un análisis visual, en donde se consideran distintas opciones y se selecciona la curva que "mejor se ajuste".

Una forma **objetiva** consiste en utilizar validación cruzada.

En el software R existe un paquete llamado **splines** que contiene las funciones:

- **bs():** Permite generar un spline definiendo el número de nodos o los grados de libertad, además del grado de los polinomios. Por defecto, ajusta splines cúbicos.
- **ns():** Permite generar un spline cúbico natural y se pueden especificar el número de nodos o los grados de libertad.

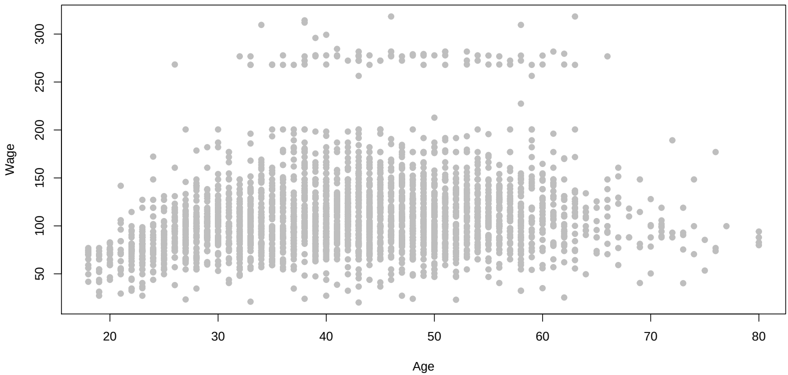
Relación entre **df** y el número de nodos en **bs()**

df	N° de nodos	Modelo resultante
3	0	$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$ (modelo usual sin splines)
4	1	$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - t_1)_+^3 + \epsilon$
5	2	$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - t_1)_+^3 + \beta_5 (x - t_2)_+^3 + \epsilon$
6	3	$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - t_1)_+^3 + \beta_5 (x - t_2)_+^3 + \beta_6 (x - t_3)_+^3 + \epsilon$
7	4	$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - t_1)_+^3 + \beta_5 (x - t_2)_+^3 + \beta_6 (x - t_3)_+^3 + \beta_7 (x - t_4)_+^3 + \epsilon$
Este patrón continua así		

Por ejemplo, al usar `bs(x, df=6)`

- Se creará un spline básico con tres nodos t_1, t_2 y t_3,
- Los nodos estarán equiespaciados en los percentiles 25%, 50% y 75%.
- No es necesario decirle los nodos porque éstos se elijen automáticamente.
- Si el usuario quiere sus propios nodos se usa entonces el parámetro `knots`.

```r
require(ISLR)
plot(Wage$age, Wage$wage, pch=19,
     col="gray",xlab="Age", ylab="Wage")
```



```r
require(splines)
# Splines cúbico con 3 nodos:
mod1<-lm(wage~bs(age,df=6,degree=3), data=Wage)
summary(mod1)
```

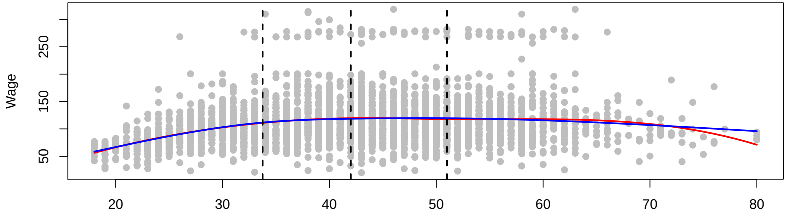
```
##
## Call:
## lm(formula = wage ~ bs(age, df = 6, degree = 3), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.681 -24.403  -5.202  15.441 201.413
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    56.314      7.258   7.759 1.17e-14 ***
## bs(age, df = 6, degree = 3)1   27.824     12.435   2.238   0.0253 *
## bs(age, df = 6, degree = 3)2   54.063      7.127   7.585 4.41e-14 ***
## bs(age, df = 6, degree = 3)3   65.828      8.323   7.909 3.62e-15 ***
## bs(age, df = 6, degree = 3)4   55.813      8.724   6.398 1.83e-10 ***
## bs(age, df = 6, degree = 3)5   72.131     13.745   5.248 1.65e-07 ***
## bs(age, df = 6, degree = 3)6   14.751     16.209   0.910   0.3629
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.91 on 2993 degrees of freedom
## Multiple R-squared:  0.08729,    Adjusted R-squared:  0.08546
## F-statistic: 47.71 on 6 and 2993 DF,  p-value: < 2.2e-16
```

Trabajando con R:

```r
# Splines cúbico con 3 nodos (df SE TOMA DIFERENTE A bs):
mod2<-lm(wage~ns(age,df=4), data=Wage)
summary(mod2)
```

```
##
## Call:
## lm(formula = wage ~ ns(age, df = 4), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.737 -24.477  -5.083  15.371 204.874
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        58.556      4.235  13.827   <2e-16 ***
## ns(age, df = 4)1   60.462      4.190  14.430   <2e-16 ***
## ns(age, df = 4)2   41.963      4.372   9.597   <2e-16 ***
## ns(age, df = 4)3   97.020     10.386   9.341   <2e-16 ***
## ns(age, df = 4)4    9.773      8.657   1.129    0.259
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.92 on 2995 degrees of freedom
## Multiple R-squared:  0.08598,    Adjusted R-squared:  0.08476
## F-statistic: 70.43 on 4 and 2995 DF,  p-value: < 2.2e-16
```

```
age.sel<-seq(from=min(Wage$age),to=max(Wage$age))
s.cubic<-predict(mod1,newdata=list(age=age.sel))
n.cubic<-predict(mod2,newdata=list(age=age.sel))
plot(Wage$age, Wage$wage, pch=19,
     col="gray",xlab="Age", ylab="Wage")
lines(age.sel,s.cubic, col="red", lwd=2)
lines(age.sel,n.cubic, col="blue", lwd=2)
abline(v=c(33.75,42.00,51.00),lwd=2,lty=2)
```



Los nodos se obtienen con la siguiente función:

- Para **splines cúbicos:**

```
attr(bs(Wage$age,df=6,degree=3),"knots")
```

```
##   25%   50%   75%
## 33.75 42.00 51.00
```

- Para **splines cúbicos naturales:**

```
attr(ns(Wage$age, df=4),"knots")
```

```
##   25%   50%   75%
## 33.75 42.00 51.00
```

Seleccionando el número de nodos del **spline cúbico** con validación cruzada:

```
require(boot)
MSE.bs<-vector()
set.seed(123)
for(i in 1:10){
glm.fit.bs<-glm(wage~bs(age,df=(3+i)),data=Wage)
cv.err.bs<-cv.glm(Wage,glm.fit.bs,K=10)
MSE.bs[i]<-cv.err.bs$delta[1]
}
plot(MSE.bs,type="b", main="Spline cúbico")
```

Seleccionando el número de nodos del **spline cúbico natural** con validación cruzada:

```
MSE.ns<-vector()
set.seed(123)
for(i in 1:10){
glm.fit.ns<-glm(wage~ns(age,df=(1+i)),data=Wage)
cv.err.ns<-cv.glm(Wage,glm.fit.ns,K=10)
MSE.ns[i]<-cv.err.ns$delta[1]
}
plot(MSE.ns,type="b", main="Spline cúbico natural")
```

Splines versus polinomios:

¿Cómo se comporta un polinomio de grado alto versus un spline con un alto número de nodos?

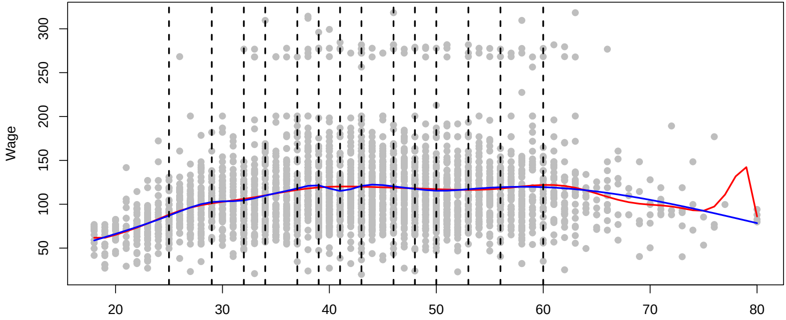
```
mod_pol<- lm(wage ~ poly(age, degree=15), data=Wage)
mod_spl<- lm(wage ~ ns(age, df=15),data=Wage)

age.sel<-seq(from=min(Wage$age),to=max(Wage$age))

fit.pol<- predict(mod_pol, newdata=list(age=age.sel))
fit.spl<- predict(mod_spl, newdata=list(age=age.sel))
```

Splines versus polinomios:

```
plot(Wage$age, Wage$wage, pch=19,
     col="gray",xlab="Age", ylab="Wage")
lines(age.sel,fit.pol, col="red", lwd=2)
lines(age.sel,fit.spl, col="blue", lwd=2)
nodos.ns<-attr(ns(Wage$age, df=15),"knots")
abline(v=nodos.ns,lwd=2,lty=2)
```



Smoothing Splines:

Originalmente, cuando se desea ajustar una curva a un conjunto de puntos, se busca una función, $g(\cdot)$, tal que la suma cuadrática de los residuales:

$$RSS = \sum_{i=1}^{n} [y_i - g(x_i)]^2$$

sea pequeña.

Smoothing Splines:

Originalmente, cuando se desea ajustar una curva a un conjunto de puntos, se busca una función, $g(\cdot)$, tal que la suma cuadrática de los residuales:

$$RSS = \sum_{i=1}^{n} [y_i - g(x_i)]^2$$

sea pequeña.

En general, se busca que $g(\cdot)$ sea lo más suave posible y pensando en esto se plantea una penalización con respecto a la segunda derivada de $g(\cdot)$ (asumiendo que dicha derivada existe). Este método se conoce como **smoothing spline** y busca minimizar:

$$RSS_\lambda = \sum_{i=1}^{n} [y_i - g(x_i)]^2 + \underbrace{\lambda \int g''(t)^2 dt}_{penalización}$$

En este método λ es un hiperparámetro, que se puede encontrar con validación cruzada, y para el cual se cumple que:

- Si $\lambda = 0$ entonces la penalización no tiene efecto.
- Si $\lambda \longrightarrow \infty$ entonces la penalización es tan grande que $g(\cdot)$ se "suaviza" hasta el punto de convertirse en una línea recta.

Smoothing Splines:

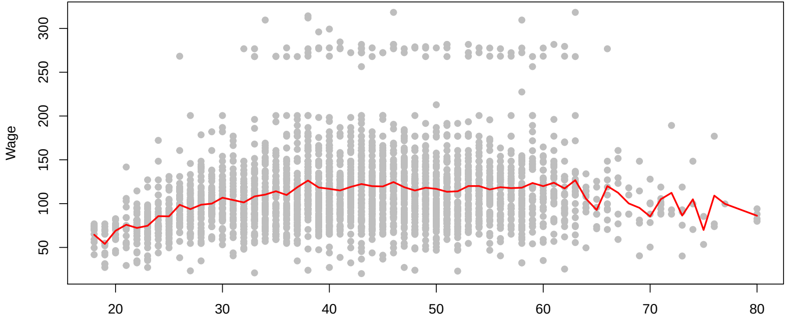
En este método λ es un hiperparámetro, que se puede encontrar con validación cruzada, y para el cual se cumple que:

- Si $\lambda = 0$ entonces la penalización no tiene efecto.
- Si $\lambda \longrightarrow \infty$ entonces la penalización es tan grande que $g(\cdot)$ se "suaviza" hasta el punto de convertirse en una línea recta.

En este método, la función $g(\cdot)$ es spline cúbico natural con nodos en $x_1, x_2, \ldots, x_n$. Esto indica que los grados de libertad de este spline es tan alto como el número de individuos, pero se controlan con el valor de λ (encontrado con LOOCV).

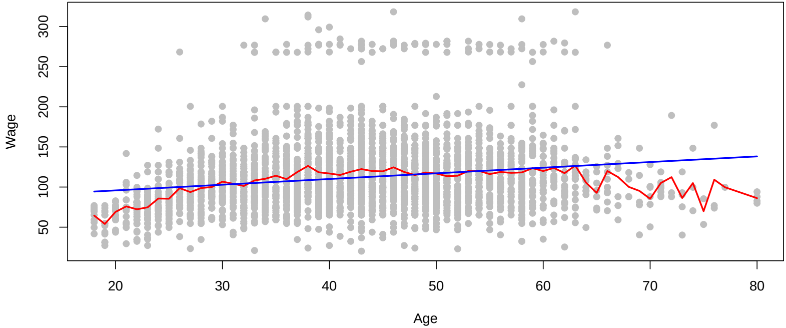
Smoothing Splines:

```
plot(Wage$age, Wage$wage, pch=19,
     col="gray",xlab="Age", ylab="Wage")
attach(Wage)
mod1<-smooth.spline(age,wage,lambda=0)
lines(mod1,col ="red",lwd =2)
```



Smoothing Splines:

```
plot(Wage$age, Wage$wage, pch=19,
     col="gray",xlab="Age", ylab="Wage")
mod2<-smooth.spline(age,wage,lambda=1000)
lines(mod1,col ="red",lwd =2) # Con lambda=0
lines(mod2,col ="blue",lwd =2) # lambda grande
```



Smoothing Splines:

```r
mod3<-smooth.spline(age,wage,df=16)
mod4<-smooth.spline(age,wage,cv=TRUE) # Con LOOCV

names(mod4)
```

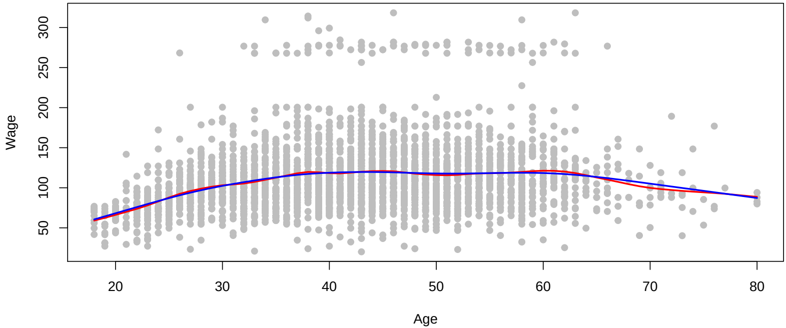
```
##  [1] "x"          "y"          "w"          "yin"
##  [6] "data"       "no.weights" "lev"        "cv.crit
## [11] "crit"       "df"         "spar"       "ratio"
## [16] "iparms"     "auxM"       "fit"        "call"
```

```r
mod4$df
```

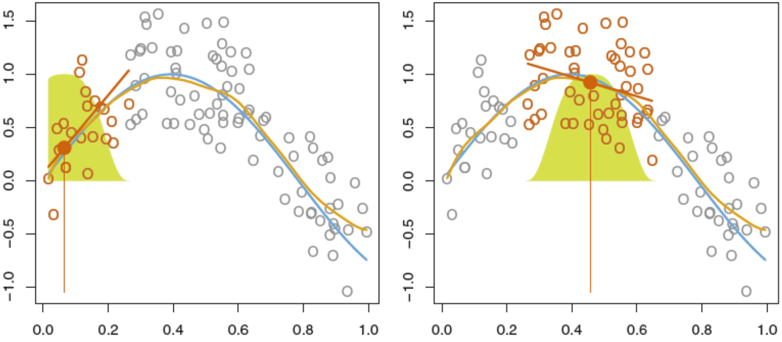
```
## [1] 6.794596
```

Smoothing Splines:

```r
plot(Wage$age, Wage$wage, pch=19,
     col="gray",xlab="Age", ylab="Wage")
lines(mod3,col ="red",lwd =2) # df grande
lines(mod4,col ="blue",lwd =2) # LOOCV
```



Regresión local



Regresión local

Se plantea realizar un suavizamiento local alrededor de un punto $X = x_0$. Un algoritmo simple consiste en:

1. Tome una fracción $s = k/n$ de puntos x_i, alrederor de x_0.
2. Dele un peso $K_{i0} = K(x_i, x_0)$ a cada uno de los puntos vecinos de x_0, de tal manera que los vecinos más cercanos tengan mayor peso que los más lejanos. En muchos casos se toma K_{i0} como la función tricubo.
3. Estime los parametros β_0 y β_1 minimizando la suma cuadrática ponderada:

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2$$

4. El valor ajustado en x_0 y que sirve para elaborar la curva es:

$$\widehat{y}_i = \widehat{\beta_0} + \widehat{\beta}_1 x_0$$

Usando la función **loess** (locally weighted smoothing) ajuste una curva a los datos de la base de datos **Wage** (wage en función de age).