

# Extensiones del modelo lineal: Regression splines

## Contents

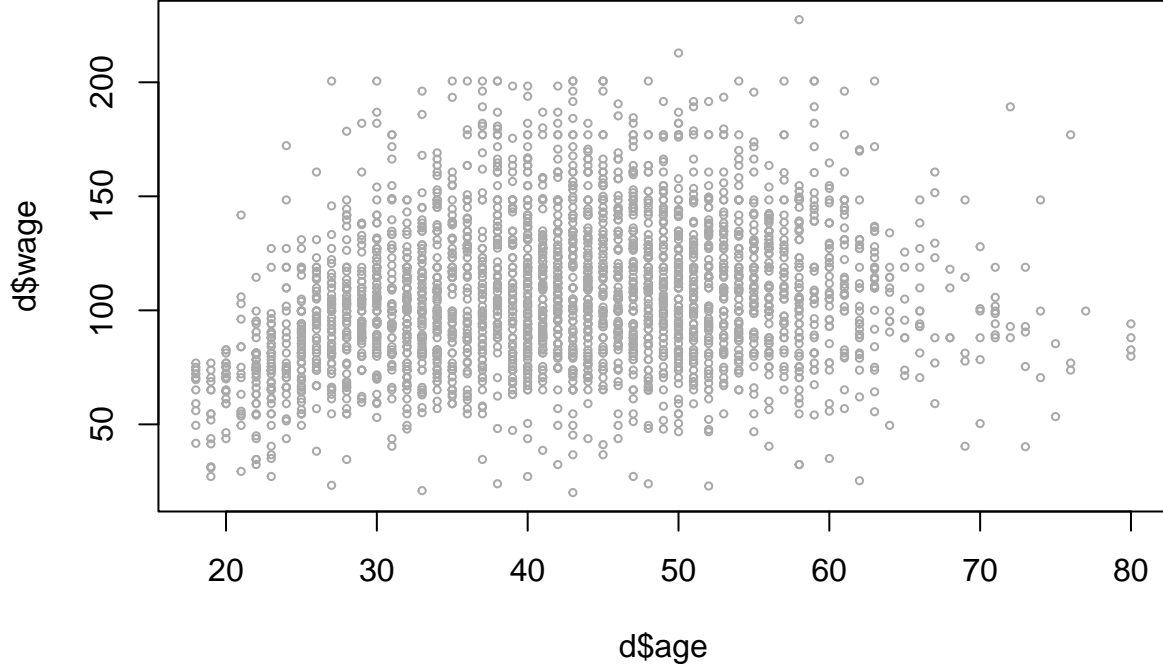
<b>1</b>	<b>Datos</b>	<b>1</b>
<b>2</b>	<b>B-splines o Basis-splines cúbicos</b>	<b>2</b>
2.1	Polinomios cúbicos a trozos . . . . .	2
2.2	Funciones base . . . . .	2
2.3	Funciones base en R . . . . .	3
2.4	Estimacion del modelo . . . . .	4
2.5	Prediccion . . . . .	4
<b>3</b>	<b>Splines cubicas naturales</b>	<b>5</b>
3.1	Definición . . . . .	5
3.2	Splines naturales en R . . . . .	5
3.3	Predicción . . . . .	6
<b>4</b>	<b>Selecccion del numero de nodos para splines cubicas</b>	<b>7</b>

## 1 Datos

Datos: Wage

Wage and other data for a group of 3000 male workers in the Mid-Atlantic region.

```
d = read.csv("datos/Wage.csv")
d = d[d$wage<250,]
plot(d$age,d$wage, cex = 0.5, col = "darkgrey")
```



## 2 B-splines o Basis-splines cúbicos

### 2.1 Polinomios cúbicos a trozos

- Se divide el rango de  $X$  por medio de  $k$  puntos (o nodos),  $c_1 < c_2 < \dots < c_k$ .
- En cada intervalo se estima un polinomio de orden 3.

$$y_i = \begin{cases} \beta_{00} + \beta_{10}x_i + \beta_{20}x_i^2 + \beta_{30}x_i^3 + u_i & \text{si } x < c_1, \\ \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + u_i & \text{si } c_1 \leq x \leq c_2, \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + u_i & \text{si } c_2 \leq x \leq c_3, \\ \dots & \dots, \\ \beta_{0k} + \beta_{1k}x_i + \beta_{2k}x_i^2 + \beta_{3k}x_i^3 + u_i & \text{si } x \geq c_k, \end{cases}$$

- Por tanto, hay  $(k + 1)(3 + 1)$  parámetros a estimar o grados de libertad (son las incógnitas).
- En los puntos  $c_1, c_2, \dots, c_k$  los polinomios han de ser continuos y suaves (segunda derivada continua). Por tanto, se añaden 3 ecuaciones en cada nodo (función continua, primera derivada continua, segunda derivada continua).
- En total se tienen  $(k + 1)(3 + 1) - 3k = k + 4$  grados de libertad.
- Por tanto, un modelo que utiliza splines de orden 3 y cuatro nodos ( $k = 4$ ) tiene 8 grados de libertad.
- Si se utilizan polinomios de orden  $d$  en cada trozo, el numero de grados de libertad sería  $(k+1)(d+1) - k*d = k + d + 1$ .

### 2.2 Funciones base

Estimar las ecuaciones anteriores con las restricciones correspondientes no es fácil. Una alternativa es usar *funciones base*.

Es fácil comprobar que el siguiente modelo cumple las condiciones de las splines cúbicas:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 h_1(x_i) + \cdots + \beta_{3+k} h_k(x_i) + u_i$$

donde:

$$h_i(x) = \begin{cases} (x - c_i)^3 & \text{si } x > c_i, \\ 0 & \text{si } x \leq c_i. \end{cases}$$

y  $c_i$  es el punto donde esta el nodo  $i$ . Por ejemplo, imaginemos un solo nodo en  $x_i = a$ . A la izquierda de a la spline es:

$$y_{1,i} = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3$$

y a la derecha de a sería:

$$y_{2,i} = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + (x_i - a)^3$$

Por tanto en  $x_i = a$  se verifica  $y_{1,i} = y_{2,i}$ ,  $y'_{1,i} = y'_{2,i}$ ,  $y''_{1,i} = y''_{2,i}$ .

Por otro lado hemos visto que se necesitan  $k+4$  parámetros en total para una spline cúbica con  $k$  nodos. Por eso tenemos que llegar hasta  $\beta_{k+3}$  (el otro parámetro sería  $\beta_0$ ).

Este polinomio también se puede expresar utilizando una base de tamaño  $k+3$ :

$$p(x) = a_0 + a_1 b_1(x) + a_2 b_2(x) + \cdots + a_{k+3} b_{k+3}(x)$$

Las  $b_k$  son las funciones base.

## 2.3 Funciones base en R

```
library(splines)
```

La función `bs()` define automáticamente una matriz con las funciones de base necesarias a partir de los nodos. Se puede hacer de dos maneras:

- Especificando los nodos:

```
dim(bs(d$age, knots = c(25, 40, 60)))
```

```
## [1] 2921    6
```

Tres nodos y splines cúbicos originan una base de tamaño 6 ( $k$  nodos originan  $k+4$  parámetros y  $k+3$  funciones base!):

- Especificando los grados de libertad:

```
# k = df - 3, se utilizan quantiles para definir los nodos
attr(bs(d$age, df = 6), "knots")
```

```
## [1] 33 42 50
```

## 2.4 Estimacion del modelo

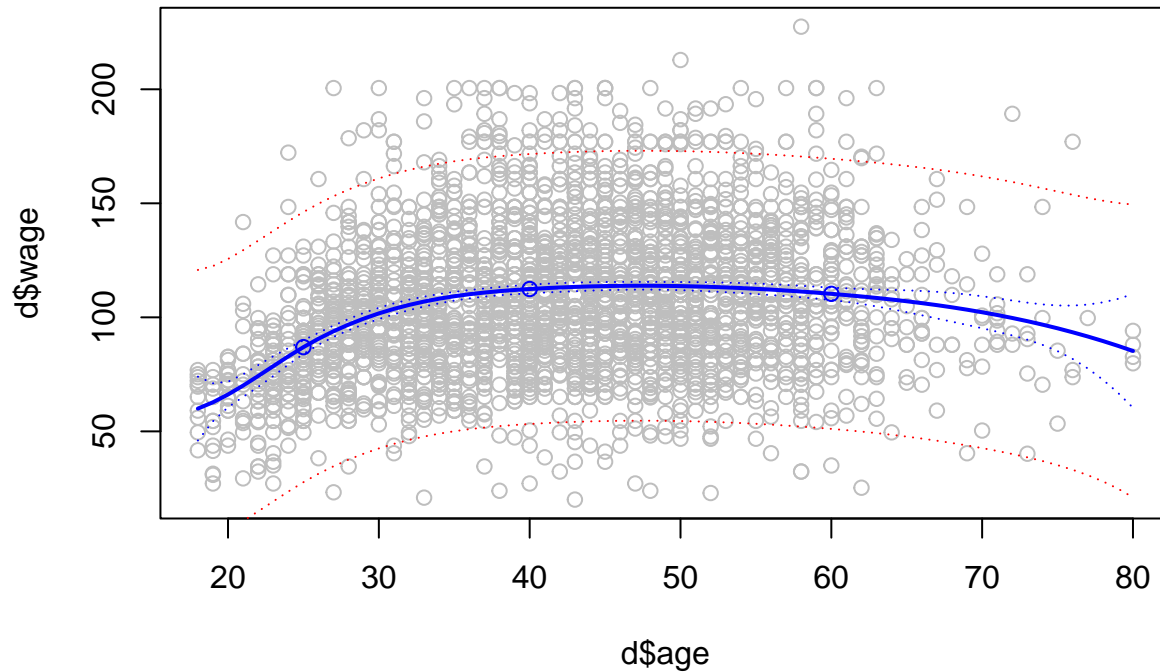
```
nodos = c(25,40,60)
m1 = lm(wage ~ bs(age, knots = nodos), data = d)
summary(m1)

##
## Call:
## lm(formula = wage ~ bs(age, knots = nodos), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -93.271 -20.467  -2.006   17.424  116.146
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      59.978      7.151   8.388 < 2e-16 ***
## bs(age, knots = nodos)1      5.534      9.484   0.583 0.559625
## bs(age, knots = nodos)2     44.434      7.280   6.104 1.17e-09 ***
## bs(age, knots = nodos)3     55.949      8.149   6.866 8.04e-12 ***
## bs(age, knots = nodos)4     52.762      8.119   6.498 9.53e-11 ***
## bs(age, knots = nodos)5     40.726     10.942   3.722 0.000201 ***
## bs(age, knots = nodos)6     25.289     14.470   1.748 0.080630 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 30.17 on 2914 degrees of freedom
## Multiple R-squared:  0.1096, Adjusted R-squared:  0.1077
## F-statistic: 59.76 on 6 and 2914 DF,  p-value: < 2.2e-16
```

## 2.5 Prediccion

```
age_grid = seq(from = min(d$age), to = max(d$age), by = 1)
yp = predict(m1, newdata = data.frame(age = age_grid), interval = "confidence", level = 0.95)
yp1 = predict(m1, newdata = data.frame(age = age_grid), interval = "prediction", level = 0.95)

plot(d$age, d$wage, col = "gray")
lines(age_grid, yp[, "fit"], col = "blue", lwd = 2)
lines(age_grid, yp[, "lwr"], col = "blue", lty = 3)
lines(age_grid, yp[, "upr"], col = "blue", lty = 3)
lines(age_grid, yp1[, "lwr"], col = "red", lty = 3)
lines(age_grid, yp1[, "upr"], col = "red", lty = 3)
# incluimos los nodos
nodos_pred = predict(m1, newdata = data.frame(age = nodos))
points(nodos, nodos_pred, col = "blue")
```



### 3 Splines cubicas naturales

#### 3.1 Definición

- Las splines tienen el problema de que en los extremos las predicciones tienen varianza elevada.
- Esto se debe a que fuera del rango de la variable (por debajo del mínimo y por encima del máximo), la spline sería cúbica.
- Una opción es obligar a que la spline sea lineal en estas zonas. Esto corrige la varianza alta de los bordes.

$$y_i = \begin{cases} \beta_{00} + \beta_{10}x_i + u_i & \text{si } x < \min(x_i), \\ \beta_{00} + \beta_{10}x_i + \beta_{20}x_i^2 + \dots + \beta_{d0}x_i^d + u_i & \text{si } \min(x_i) \leq x \leq c_1, \\ \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \dots + \beta_{d1}x_i^d + u_i & \text{si } c_1 \leq x \leq c_2, \\ \dots & \dots, \\ \beta_{0k} + \beta_{1k}x_i + \beta_{2k}x_i^2 + \dots + \beta_{dk}x_i^d + u_i & \text{si } c_k \leq x \leq \max(x_i), \\ \beta_{0k} + \beta_{1k}x_i + u_i & \text{si } x \geq \max(x_i), \end{cases}$$

- Para conseguir estas rectas se obliga a que la segunda derivada en los bordes sea cero (si la primera derivada es distinta de cero y la segunda es cero, tenemos una recta).
- Luego se añade 1 restricción en cada extremo, en total hay  $k + 4 - 2 = k + 2$  grados de libertad,  $k + 1$  funciones base.

#### 3.2 Splines naturales en R

- En R, se definen splines cúbicas naturales por medio de la función `ns()`.

```
dim(ns(d$age, knots = c(25, 40, 60)))
```

```
## [1] 2921    4
```

- Se estima el modelo:

```

nodos = c(25, 40, 60)
m2 = lm(wage ~ ns(age, knots = nodos), data = d)
summary(m2)

##
## Call:
## lm(formula = wage ~ ns(age, knots = nodos), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -93.310 -20.630  -1.956   17.398  116.207
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      55.978      3.887   14.400  <2e-16 ***
## ns(age, knots = nodos)1  59.970      3.804   15.765  <2e-16 ***
## ns(age, knots = nodos)2  44.366      4.353   10.193  <2e-16 ***
## ns(age, knots = nodos)3  81.434      9.110    8.939  <2e-16 ***
## ns(age, knots = nodos)4   8.094      7.454    1.086    0.278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 30.16 on 2916 degrees of freedom
## Multiple R-squared:  0.1094, Adjusted R-squared:  0.1081
## F-statistic: 89.52 on 4 and 2916 DF,  p-value: < 2.2e-16

```

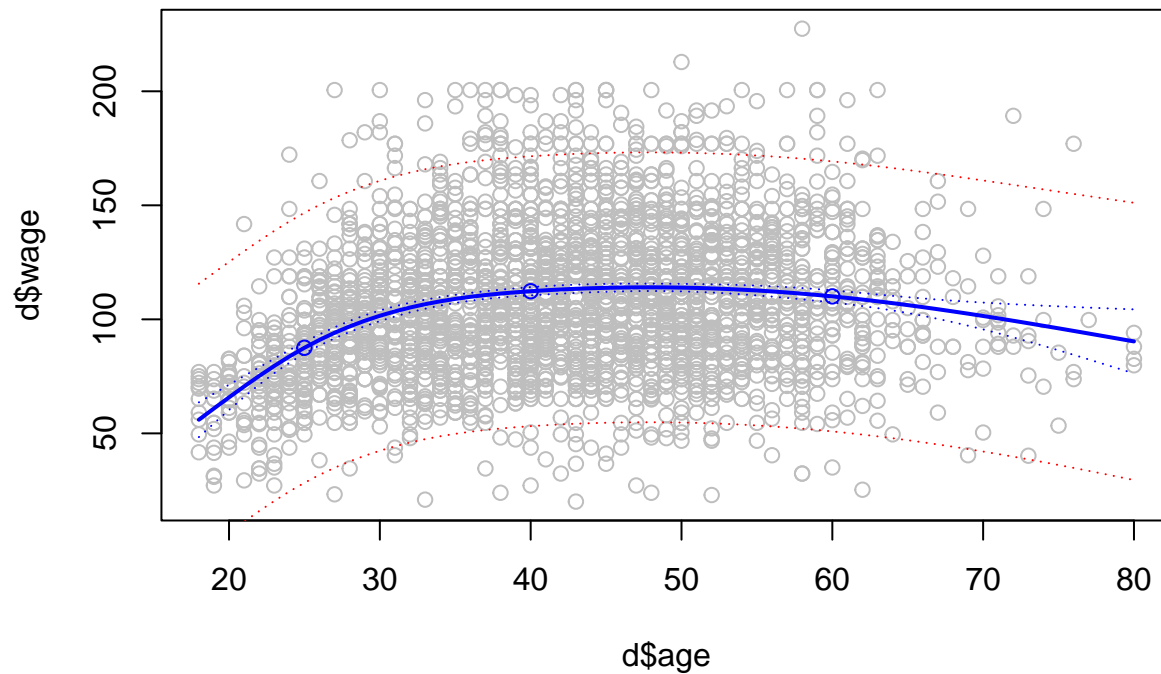
### 3.3 Predicción

```

ypn = predict(m2, newdata = data.frame(age = age_grid), interval = "confidence", level = 0.95)
ypn1 = predict(m2, newdata = data.frame(age = age_grid), interval = "prediction", level = 0.95)

plot(d$age, d$wage, col = "gray")
lines(age_grid, ypn[, "fit"], col = "blue", lwd = 2)
lines(age_grid, ypn[, "lwr"], col = "blue", lty = 3)
lines(age_grid, ypn[, "upr"], col = "blue", lty = 3)
lines(age_grid, ypn1[, "lwr"], col = "red", lty = 3)
lines(age_grid, ypn1[, "upr"], col = "red", lty = 3)
# incluimos los nodos
nodos_pred = predict(m2, newdata = data.frame(age = nodos))
points(nodos, nodos_pred, col = "blue")

```



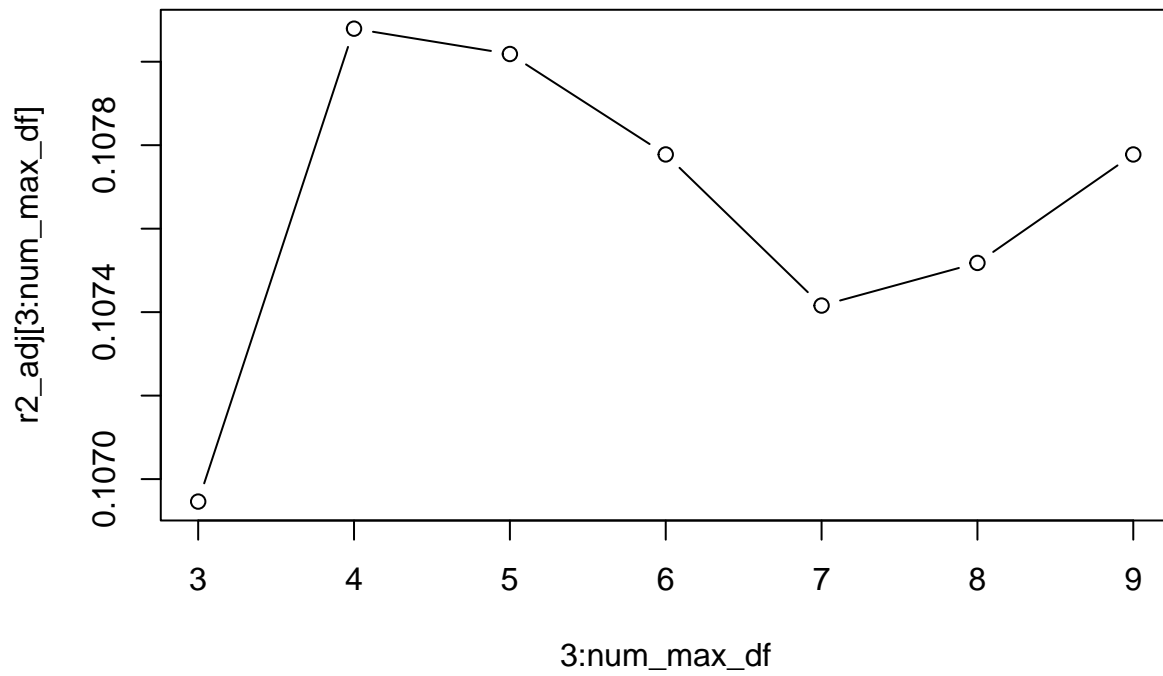
## 4 Selección del número de nodos para splines cubicas

Se van a utilizar B-splines cubicas.

```
# numero maximo de funciones base => numero maximo de nodos = nmu. max. fb - 3
num_max_df = 9

r2_adj = rep(0, num_max_df)
# empezamos en 3 porque cero nodos son 3 df, un polinomio cubico
for (i in 3:num_max_df){
  m = lm(wage ~ bs(age, df = i), data = d)
  m_summary = summary(m)
  r2_adj[i] = m_summary$adj.r.squared
}

plot(3:num_max_df, r2_adj[3:num_max_df], type = "b")
```



Luego colocar un nodo en el centro parece lo más adecuado.