Métodos de regresión no lineal: Regresión Polinómica, Regression Splines, Smooth Splines y GAMs

Joaquín Amat Rodrigo j.amatrodrigo@gmail.com

Febrero, 2017

Introducción

Para entender mejor el contenido de este capítulo es recomendable leer primero: Introducción a la Regresión Lineal Múltiple, Ejemplo práctico de regresión lineal simple, múltiple, polinomial e interacción entre predictores y Regresión logística simple y múltiple.

Los modelos lineales tienen la ventaja de ser fácilmente interpretables, sin embargo, pueden tener limitaciones importantes en capacidad predictiva. Esto se debe a que, la asunción de linealidad, es con frecuencia una aproximación demasiado simple para describir las relaciones reales entre variables. A continuación, se describen métodos que permiten relajar la condición de linealidad intentando mantener al mismo tiempo una interpretabilidad alta.

Regresión polinómica: Consigue añadir curvatura al modelo introduciendo nuevos predictores que se obtienen al elevar todos o algunos de los predictores originales a distintas potencias.

Step functions: Se divide el rango del predictor en K subintervalos de forma que, en cada uno, se emplean únicamente las observaciones que pertenecen a la región para ajustar el modelo.

Regression splines: Se trata de una extensión de la regresión polinómica y de las *step functions* que consigue una mayor flexibilidad. Consiste en dividir el rango del predictor X en K subintervalos. Para cada una de las nuevas regiones se ajusta una función polinómica, introduciendo una serie de restricciones que hacen que los extremos de cada función se aproximen a los de las funciones de las regiones colindantes.

Smoothing splines: El concepto es similar a *regression splines* pero consigue la aproximación de los extremos de las funciones colindantes de forma distinta.

Local regression: Se asemeja a *regression splines* y *smoothing splines* en cuanto a que también se realizan ajustes por regiones, pero en este método las regiones solapan las unas con las otras.

Generalized additive models: Es el resultado de extender los métodos anteriores para emplear múltiples predictores.

Regresión no lineal con un único predictor

Regresión polinómica

La forma más sencilla de incorporar flexibilidad a un modelo lineal es introduciendo nuevos predictores obtenidos al elevar a distintas potencias el predictor original.

Partiendo del modelo lineal

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Se obtiene un modelo polinómico de grado *d* a partir de la ecuación

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$

Los modelos polinómicos se pueden ajustar mediante regresión lineal por mínimos cuadrados ya que, aunque generan modelos no lineales, su ecuación no deja de ser una ecuación lineal con predictores $x, x^2, x^3, \ldots, x^d$. Por esta misma razón, las funciones polinómicas pueden emplearse en regresión logística para predecir respuestas binarias. Solo es necesario realizar una transformación *logit*.

$$P(y_i > Y | x_i = X) = \frac{exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d)}{1 + exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d)}$$

En el libro *Introduction to Statistical Learning* desaconsejan el uso de modelos polinómicos con grado mayor de 3 o 4 debido a un exceso de flexibilidad (*overfitting*), principalmente en los extremos del predictor *X*. La selección del grado de polinomio óptimo puede hacerse mediante *cross validation*.

Ejemplo 1

El set de datos Wage del paquete ISRL contiene información sobre 3000 trabajadores. Entre las 12 variables registradas se encuentra el salario (wage) y la edad (age). Dada la relación no lineal existente entre estas dos variables, se recurre a un modelo polinómico de grado 4 que permita predecir el salario en función de la edad.

La función `lm()` permite ajustar modelos lineales por mínimos cuadrados. En el argumento formula se especifica la variable dependiente y los predictores, que en este caso son `wage ~ age + age^2 + age^3 + age^4`. `lm()` tiene flexibilidad a la hora de interpretar el argumento fórmula, por lo que existen diferentes formas de obtener el mismo ajuste. Todas las que se muestran a continuación son equivalentes:

```
• lm(wage ~ age + I(age^2) + I(age^3) + I(age^4), data = Wage)
• lm(wage ~ cbind(age + age^2 + age^3 + age^4), data = Wage)
• lm(wage ~ poly(age, 4, raw = TRUE), data = Wage)
• lm(wage ~ poly(age, 4), data = Wage)
```

Con la función `poly()` se puede generar directamente un polinomio, evitando tener que escribir toda la fórmula. Cuando se especifica que su argumento `raw = TRUE`, devuelve una matriz formada por el valor de la variable original elevada a cada una de las potencias del polinomio. Si el argumento `raw = FALSE` se devuelve una matriz con polinomios ortogonales, en la que cada columna es una combinación lineal de las otras. Es importante tenerlo en cuenta al emplear la función `poly()` en el ajuste de modelos, ya que, aunque no influye en el valor de las predicciones que se obtienen (las curvas obtenidas son iguales) sí que cambia el valor estimado de los coeficientes.

```
library(ISLR)
data("Wage")
Wage$age[1:5]
```

```
## [1] 18 24 45 43 50
```

```
poly(Wage$age, degree = 4, raw = TRUE, simple = TRUE)[1:5, ]
```

```
##       1    2      3       4
## [1,] 18  324   5832  104976
## [2,] 24  576  13824  331776
## [3,] 45 2025  91125 4100625
## [4,] 43 1849  79507 3418801
## [5,] 50 2500 125000 6250000
```

```
poly(Wage$age, degree = 4, raw = FALSE, simple = TRUE)[1:5, ]
```

```
##                   1           2             3           4
## [1,] -0.0386247992  0.055908727 -0.0717405794  0.08672985
## [2,] -0.0291326034  0.026298066 -0.0145499511 -0.00259928
## [3,]  0.0040900817 -0.014506548 -0.0001331835  0.01448009
## [4,]  0.0009260164 -0.014831404  0.0045136682  0.01265751
## [5,]  0.0120002448 -0.009815846 -0.0111366263  0.01021146
```

```
# CÁLCULO DEL MODELO POLINÓMICO DE GRADO 4
# ----------------------------------------
modelo_poli4 <- lm(wage ~ poly(age, 4), data = Wage)
summary(modelo_poli4)
```

```
##
## Call:
## lm(formula = wage ~ poly(age, 4), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.707 -24.626  -4.993  15.217 203.693
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    111.7036     0.7287 153.283  < 2e-16 ***
## poly(age, 4)1  447.0679    39.9148  11.201  < 2e-16 ***
## poly(age, 4)2 -478.3158    39.9148 -11.983  < 2e-16 ***
## poly(age, 4)3  125.5217    39.9148   3.145  0.00168 **
## poly(age, 4)4  -77.9112    39.9148  -1.952  0.05104 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.91 on 2995 degrees of freedom
## Multiple R-squared:  0.08626,    Adjusted R-squared:  0.08504
## F-statistic: 70.69 on 4 and 2995 DF,  p-value: < 2.2e-16
```

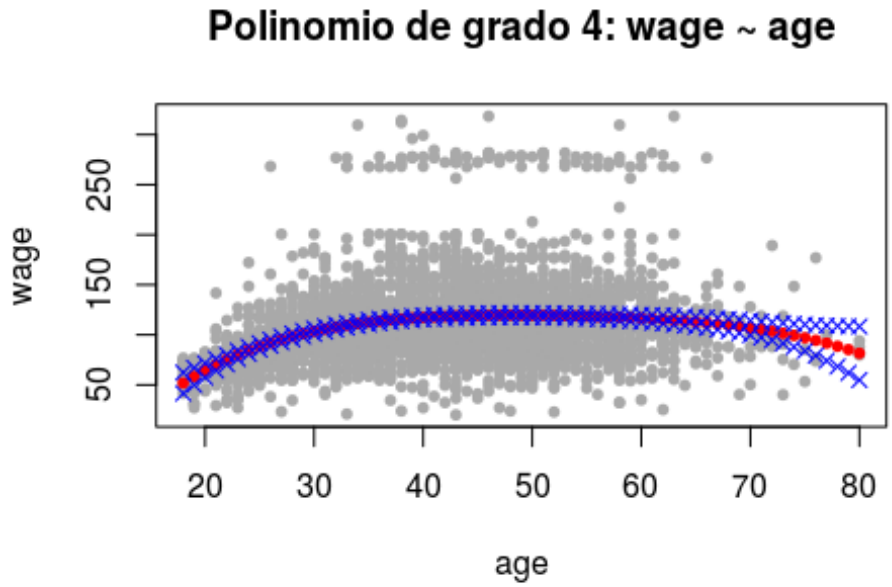
El *p-value* obtenido para el estadístico F es muy bajo (< 2.2e-16), lo que indica que al menos uno de los predictores introducidos en el modelo está relacionado con la variable respuesta *wage*. Los *p-values* individuales de cada predictor son todos muy bajos a excepción de *age^4*, lo que apunta a que un polinomio de grado 3 es suficiente para modelar el salario en función de la edad. Acorde al R^2, el modelo es capaz de explicar el 8.6% de la variabilidad observada en *wage*. (Es un % muy bajo).

Si se quiere generar una representación gráfica del modelo ajustado

$$wage = 111.70 + 447.07age - 478.32age^2 + 125.52age^3 - 77.91age^4)$$

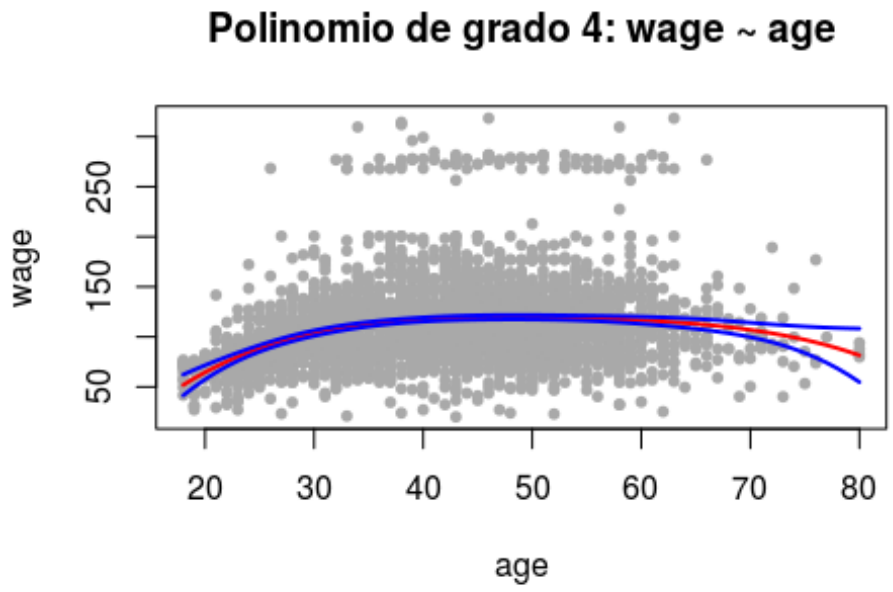
se tiene que representar la curva que describe su ecuación. Para conseguirlo, es necesario predecir el valor de la variable respuesta en puntos interpolados dentro del rango del predictor. Cuantos más puntos se interpolen, mejor será la representación. R permite obtener predicciones (junto con el intervalo de confianza) mediante la función `predict()`.

```
# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
predicciones <- predict(modelo_poli4, newdata = nuevos_puntos, se.fit = TRUE,
                        level = 0.95)

# CÁLCULO DEL INTERVALO DE CONFIANZA SUPERIOR E INFERIOR 95%
# ------------------------------------------------------------------------------
intervalo_conf <- data.frame(inferior = predicciones$fit -
                                        1.96*predicciones$se.fit,
                             superior = predicciones$fit +
                                        1.96*predicciones$se.fit)

attach(Wage)
plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("Polinomio de grado 4: wage ~ age")
points(x = nuevos_puntos$age, predicciones$fit, col = "red", pch = 20)
points(x = nuevos_puntos$age, intervalo_conf$inferior, col = "blue", pch = 4)
points(x = nuevos_puntos$age, intervalo_conf$superior, col = "blue", pch = 4)
```



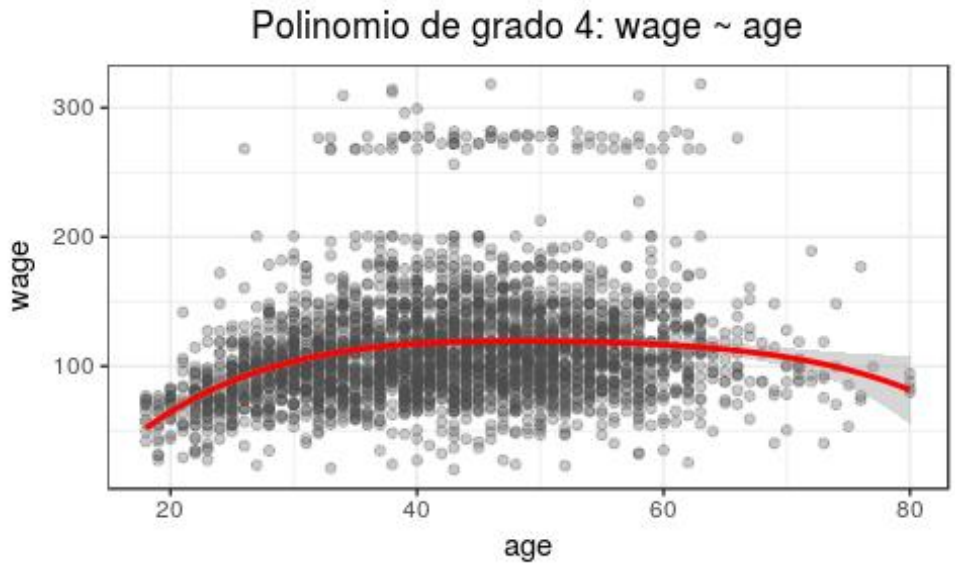
El ejemplo anterior muestra que, para conseguir el efecto visual de curva continua, es necesario interpolar muchos datos. La función `lines()` facilita el proceso uniendo los puntos con segmentos, lo que mejora la representación gráfica de curvas.

```
attach(Wage)
plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("Polinomio de grado 4: wage ~ age")
lines(x = nuevos_puntos$age, predicciones$fit, col = "red", lwd = 2)
lines(x = nuevos_puntos$age, intervalo_conf$inferior, col = "blue", lwd = 2)
lines(x = nuevos_puntos$age, intervalo_conf$superior, col = "blue", lwd = 2)
```



El paquete gráfico ggplot2 permite obtener representaciones de modelos de forma rápida. Solo con especificar la fórmula del modelo, ejecuta automáticamente todos los cálculos necesarios (ajuste, predicción...).

```
library(ggplot2)
ggplot(data = Wage, aes(x = age, y = wage)) +
  geom_point(color = "grey30", alpha = 0.3) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 4), color = "red") +
  labs(title = "Polinomio de grado 4: wage ~ age") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```



Cuando se realiza regresión polinómica, se debe decidir qué grado de polinomio emplear. Cuanto mayor sea el polinomio más flexibilidad tendrá el modelo pero, a su vez, más riesgo de *overfitting*. Acorde al principio de parsimonia, el grado óptimo es el grado más bajo que permita explicar la relación entre ambas variables. Para identificarlo se puede recurrir a dos estrategias distintas: contraste de hipótesis o *cross-validation*.

Comparación de modelos por contraste de hipótesis ANOVA

Identificar el modelo polinómico más simple que permite explicar la relación entre variables equivale a identificar el grado de polinomio a partir del cual ya no hay una mejora significativa del ajuste. Cuando se comparan dos modelos anidados (el modelo de menor tamaño está formado por un subset de predictores del modelo mayor), se puede saber si el modelo mayor aporta una mejora sustancial estudiando si los coeficientes de regresión de los predictores adicionales son distintos a cero. El test estadístico empleado para hacerlo es el ANOVA.

$$Modelo_{menor}: \ y = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$$

$$Modelo_{mayor}: \ y = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k + \beta_{k+1} x_{k+1} + \ldots + \beta_p x_p$$

La hipótesis a contrastar es que todos los coeficientes de regresión de los predictores adicionales son igual a cero, frente a la hipótesis alternativa de que al menos uno es distinto.

$$H_0: \beta_{k+1} = \ldots = \beta_p$$

El estadístico empleado es:

$$F = \frac{(SEE_{Modelo_{menor}} - SEE_{Modelo_{mayor}})/(p-k)}{SEE_{Modelo_{mayor}}/(n-p-1)}$$

Dado que un polinomio de orden *n* siempre va a estar anidado a uno de orden *n+1*, se pueden comparar modelos polinómicos dentro un rango de grados haciendo comparaciones secuenciales.

```
# Se ajustan modelos polinómicos de grado 1 a 5
# ------------------------------------------------------------------------------
modelo_1 <- lm(wage ~ age, data = Wage)
modelo_2 <- lm(wage ~ poly(age, 2), data = Wage)
modelo_3 <- lm(wage ~ poly(age, 3), data = Wage)
modelo_4 <- lm(wage ~ poly(age, 4), data = Wage)
modelo_5 <- lm(wage ~ poly(age, 5), data = Wage)

anova(modelo_1, modelo_2, modelo_3, modelo_4, modelo_5)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8888  0.001679 **
## 4   2995 4771604  1      6070   3.8098  0.051046 .
## 5   2994 4770322  1      1283   0.8050  0.369682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

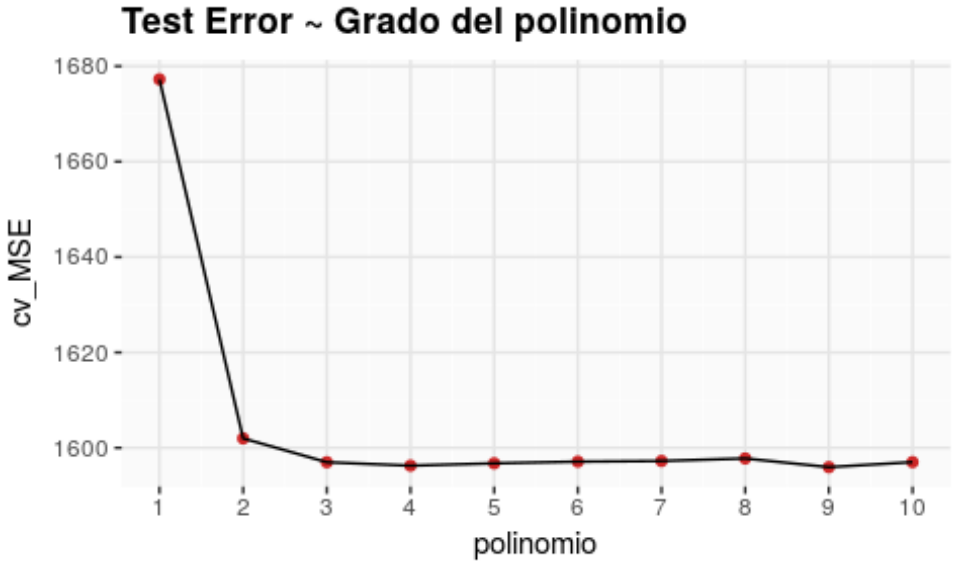
El *p-value* de la comparación entre el modelo lineal (modelo_1) y el cuadrático (modelo_2) es prácticamente cero (< 2.2e-16), indicando que el modelo lineal no es suficiente. La comparación entre el modelo cuadrático y el cúbico también ha resultado en un *p-value* muy pequeño (0.001679), evidencia suficiente de que el modelo cúbico es superior. El *p-value* obtenido al comparar el modelo cúbico con el de grado 4 está ligeramente por encima del 0.05 y el de comparar grado 4 con grado 5 es muy alto (0.37). En base a los resultados del ANOVA, el modelo cúbico es el mejor. Polinomios superiores no aportan una mejora significativa y polinomios inferiores pierden mucha capacidad de ajuste.

Es importante recordar que las comparaciones por ANOVA pueden hacerse entre cualquier par de modelos, siempre y cuando estos sean anidados.

Comparación de modelos por Cross-Validation

Otra forma de identificar con que polinomio se consigue el mejor modelo es mediante *cross-validation*. El proceso consiste en ajustar un modelo para cada grado de polinomio y estimar su *test error* (*Mean Square Error*). El mejor modelo es aquel a partir del cual ya no hay una reducción sustancial del *test error*. Para una descripción detallada de *cross-validation* y del código empleado a continuación ver capítulo Validación de modelos de regresión: Cross-validation, OneLeaveOut, Bootstrap.

```
library(boot)
cv_MSE_k10 <- rep(NA,10)

for (i in 1:10) {
  modelo <- glm(wage ~ poly(age, i), data = Wage)
  set.seed(17)
  cv_MSE_k10[i] <- cv.glm(data = Wage, glmfit = modelo, K = 10)$delta[1]
}
p4 <- ggplot(data = data.frame(polinomio = 1:10, cv_MSE = cv_MSE_k10),
             aes(x = polinomio, y = cv_MSE)) +
      geom_point(colour = c("firebrick3")) +
      geom_path()
p4 <- p4 + theme(panel.grid.major = element_line(colour  =  'gray90'))
p4 <- p4 + theme(plot.title = element_text(face  =  'bold'))
p4 <- p4 + theme(panel.background = element_rect(fill  =  'gray98'))
p4 <- p4 + labs(title  =  'Test Error ~ Grado del polinomio')
p4 <- p4 + scale_x_continuous(breaks = 1:10)
p4
```



El método de *cross-validation k=10* también indica que el mejor modelo es el que emplea un polinomio de grado 3.

Ejemplo 2. Regresión polinómica logística

Empleando el mismo set de datos Wage que en el ejercicio anterior, se pretende crear un modelo logístico que permita predecir la probabilidad de que un trabajador cobre más de 250.000 dólares en función de la edad que tiene.

Tal como se describe en el capítulo Regresión logística simple y múltiple, para modelar la probabilidad de que una observación se clasifique en un grupo u otro dependiendo del valor que tome un predictor continuo, se necesita:

Que la variable respuesta sea cualitativa con dos o más niveles (la regresión logística trabaja mejor con solo dos niveles). En el ejemplo que aquí se estudia, la variable respuesta de interés es si *wage* > *250*, cuyo resultado puede ser *verdadero* o *falso*. Dado que la variable disponible *wage* es continua, se tiene que realizar una transformación. La función `I(condición)` devuelve un *TRUE* si se cumple la condición y *FALSE* de lo contrario.

```
set.seed(365)
muestra <- sample(Wage$wage,6)
muestra
```

```
## [1] 277.79948  52.15617  99.68946 200.54326  99.68946 114.47571
```

```
I(muestra > 250)
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE FALSE
```

Ajuste por regresión empleando una transformación logística. Esto se consigue con la función `glm()` especificando el argumento `family = binomial`.

```
# REGRESIÓN LOGÍSTICA CON POLINOMIO DE GRADO 4
# ------------------------------------------------------

# Creación de una variable binaria si wage>250
Wage$wage_superior250 <- I(Wage$wage > 250)

# Ajuste del modelo logístico
modelo_logit <- glm(wage_superior250  ~ poly(age, 4), family = "binomial",
                    data = Wage)
```

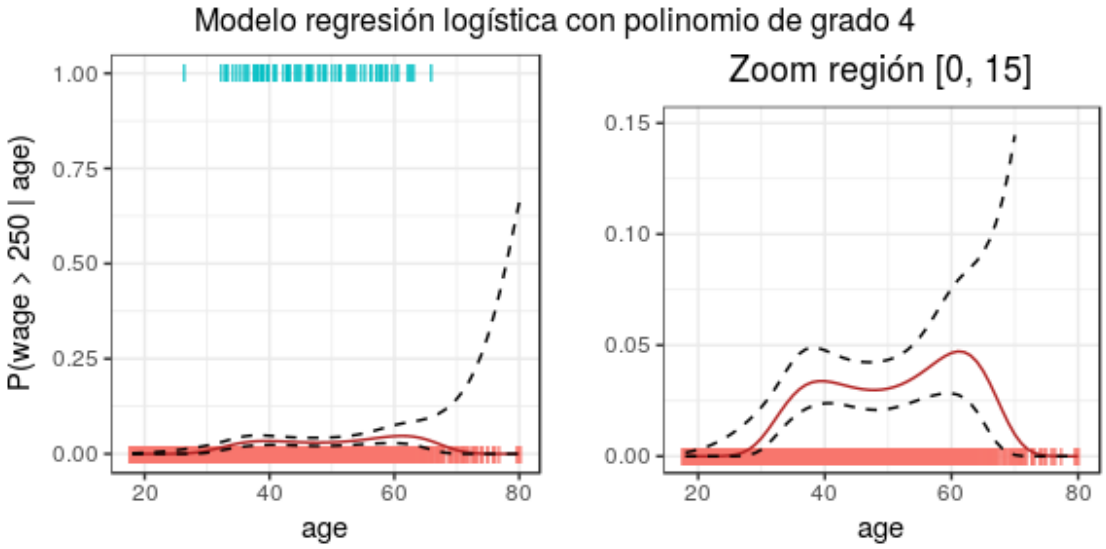
Al emplear la función `predict()` con un modelo logístico, es importante tener en cuenta que por defecto se devuelve el logaritmo de ODDs (*Log_ODDs*). Para transformarlos en probabilidad se invierte la función logística:

$$P(Y = 1|X = \frac{e^{Log\ ODDs}}{1 + e^{Log\ ODDs}})$$

Es posible obtener directamente la probabilidad de las predicciones seleccionando el argumento `type = "response"`. A pesar de que esta forma es más directa, si junto al valor predicho se quiere obtener su intervalo de confianza y que este caiga dentro de [0, 1], se tienen que realizar los cálculos con los *Log ODDs* para finalmente transformarlos a probabilidad.

```
# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
# Devuelve las predicciones en forma de log_ODDs
# Si se indica se.fit = TRUE se devuelve el error estándar de cada predicción
predicciones <- predict(modelo_logit, newdata = nuevos_puntos, se.fit = TRUE,
                        level = 0.95)

# CÁLCULO DEL INTERVALO DE CONFIANZA DEL 95%
# ------------------------------------------------------------------------------
limite_inferior <- predicciones$fit - 1.96 * predicciones$se.fit
limite_superior <- predicciones$fit + 1.96 * predicciones$se.fit

# TRANSFORMACIÓN DE LOG ODDs A PROBABILIDADES
# ------------------------------------------------------------------------------
fit_logit <- exp(predicciones$fit) / (1 + exp(predicciones$fit))
limite_inferior_logit <- exp(limite_inferior) / (1 + exp(limite_inferior))
limite_superior_logit <- exp(limite_superior) / (1 + exp(limite_superior))

# RUG PLOT: REPRESENATACIÓN GRÁFICA CON GGPLOT2
# ------------------------------------------------------------------------------
library(ggplot2)
library(gridExtra)
datos_curva <- data.frame(age = nuevos_puntos,
                          probabilidad = fit_logit,
                          limite_inferior_logit = limite_inferior_logit,
                          limite_superior_logit = limite_superior_logit)
```

```
p1 <- ggplot(Wage, aes(x = age, y = as.numeric(wage_superior250))) +
      #La variable respuesta y se convierte de vector lógico a 1/0
      geom_jitter(aes(color = as.factor(wage_superior250)), shape = "I",
                  size = 3, height = 0) +
      geom_line(data = datos_curva, aes(y = probabilidad), color = "firebrick") +
geom_line(data = datos_curva, aes(y = limite_inferior_logit),
                linetype = "dashed") +
      geom_line(data = datos_curva, aes(y = limite_superior_logit),
                linetype = "dashed") +
      theme_bw() +
      labs(y = "P(wage > 250 | age)") +
      theme(legend.position = "null") +
      theme(plot.title = element_text(hjust = 0.5))

# Zoom en la zona baja
p1_zoom <- p1 + lims(y = c(0,0.15)) +
  labs(title = "Zoom región [0, 15]", y = "")

grid.arrange(p1, p1_zoom, ncol = 2,
             top = "Modelo regresión logística con polinomio de grado 4")
```



Step Functions

La regresión polinómica explicada anteriormente persigue generar una única función global que describa el comportamiento de la variable dependiente *Y* en todo el rango del predictor *X*. La estrategia del método *step functions* consiste en dividir el rango del predictor *X* en varios subintervalos y ajustar una constante distinta para cada uno.

Supóngase que se crean *K* puntos de corte $c_1,\ c_2, \ldots,\ c_k$ en el rango del predictor *X* generando *K+1* intervalos. Para cada uno de estos intervalos se crea una variable *dummy* $C_0(X),\ C_1(X), \ldots,\ C_K(X)$. El valor de estas variables será 1 si *X* está dentro del intervalo asociado con la variable y 0 de lo contrario. Dado que cualquier valor de *X* va a estar comprendido en uno de los *K+1* intervalos y solo en uno, únicamente una de las variables *dummy* tendrá el valor de 1 y las demás serán cero.

VARIABLE	INTERVALO
$C_0(X)$	$I(X < c_1)$
$C_1(X)$	$I(c_1 \leq X < c_2)$
...	.....
$C_{k-1}(X)$	$I(c_{k-1} \leq X < c_k)$
$C_{k-1}(X)$	$I(X \leq c_K)$

El término $I()$ es un indicador de función que devuelve 1 si la condición se cumple y 0 si no se cumple.

Una vez generados los intervalos, mediante regresión por mínimos cuadrados se ajusta un modelo lineal que contenga como predictores las variables $C_0(X), C_1(X), \ldots, C_K(X)$.

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \ldots + \beta_K C_K(x_i) + \epsilon_i$$

Al igual que ocurre con las funciones polinómicas, las *step functions* pueden emplearse en modelos de regresión logística.

$$P(y_i > Y | x_i = X) = \frac{exp(\beta_0 + \beta_1 C_1(x_i) + \ldots + \beta_K C_K(x_i)}{1 + exp(\beta_0 + \beta_1 C_1(x_i) + \ldots + \beta_K C_K(x_i)}$$

La desventaja de este método deriva de que la mayoría de predictores no tienen puntos de corte establecidos, por lo que al imponerlos en determinadas posiciones se puede estar perdiendo la naturaleza de la relación.

Ejemplo

Empleando el mismo set de datos Wage que en el ejercicio anterior, se pretende crear un modelo formado por 4 step functions* que permita predecir el salario de un trabajador en función de la edad que tiene.*

En `R`, los modelos *step functions* se obtienen mediante una combinación de las funciones `lm()` y `cut()`. Dado un vector numérico como argumento, la función `cut()` establece *n* puntos de corte y devuelve una variable cualitativa que indica el subintervalo al que pertenece cada observación.

```
# Ejemplo de la función cut()
set.seed(357)
cut(runif(n = 20, min = 0, max = 100), breaks = 3)
```

```
##  [1] (5.41,36.7] (5.41,36.7] (5.41,36.7] (5.41,36.7] (36.7,67.9]
##  [6] (36.7,67.9] (67.9,99.2] (67.9,99.2] (67.9,99.2] (67.9,99.2]
## [11] (36.7,67.9] (36.7,67.9] (36.7,67.9] (5.41,36.7] (36.7,67.9]
## [16] (5.41,36.7] (5.41,36.7] (67.9,99.2] (67.9,99.2] (36.7,67.9]
## Levels: (5.41,36.7] (36.7,67.9] (67.9,99.2]
```

```
# Es posible indicarle a la función cut() dónde se quieren los puntos de corte
```

Cuando la función `lm()` reconoce que se está empleando `cut()` genera automáticamente las variables *dummy* necesarias para proceder con la regresión por mínimos cuadrados.

```
table(cut(Wage$age, 4))
```

```
##
## (17.9,33.5]   (33.5,49]   (49,64.5] (64.5,80.1]
##         750        1399         779          72
```

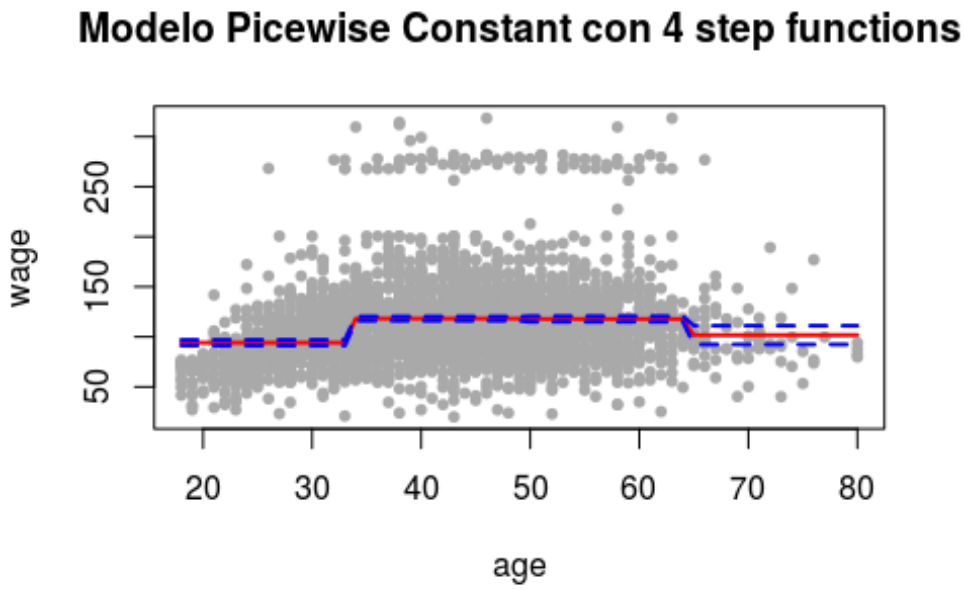
```
modelo_step_fun <- lm(wage ~ cut(age, 4), data = Wage)
summary(modelo_step_fun)
```

```
##
## Call:
## lm(formula = wage ~ cut(age, 4), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.126 -24.803  -6.177  16.493 200.519
##
```

```
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               94.158      1.476  63.790   <2e-16 ***
## cut(age, 4)(33.5,49]      24.053      1.829  13.148   <2e-16 ***
## cut(age, 4)(49,64.5]      23.665      2.068  11.443   <2e-16 ***
## cut(age, 4)(64.5,80.1]     7.641      4.987   1.532    0.126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.42 on 2996 degrees of freedom
## Multiple R-squared:  0.0625, Adjusted R-squared:  0.06156
## F-statistic: 66.58 on 3 and 2996 DF,  p-value: < 2.2e-16
```

La función `cut()` ha establecido los puntos de corte en los valores *age = 33.5, 49, 64.5*. En el `summary` del modelo obtenido no aparece el grupo *age < 33.5*, lo que significa que este es el grupo de referencia. El *intercept* (94.158) se interpreta como el salario medio de los trabajadores con *age < 33.5* y el coeficiente de regresión estimado de cada grupo como el incremento promedio de salario respecto al grupo de referencia.

```r
# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
predicciones <- predict(modelo_step_fun, newdata = nuevos_puntos, se.fit = TRUE,
                        level = 0.95)

# CÁLCULO DEL INTERVALO DE CONFIANZA SUPERIOR E INFERIOR
# ------------------------------------------------------------------------------
intervalo_conf <- data.frame(
                  inferior = predicciones$fit - 1.96*predicciones$se.fit,
                  superior = predicciones$fit + 1.96*predicciones$se.fit)
attach(Wage)
plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("Modelo Picewise Constant con 4 step functions")
lines(x = nuevos_puntos$age, predicciones$fit, col = "red", lwd = 2)
lines(x = nuevos_puntos$age, intervalo_conf$inferior, col = "blue",
      lwd = 2, lty = 2)
lines(x = nuevos_puntos$age, intervalo_conf$superior, col = "blue",
      lwd = 2, lty = 2)
```


Modelo Picewise Constant con 4 step functions
wage
50
150
250
20
30
40
50
60
70
80
age

Regression Splines

Dentro de la familia de *regression splines* se diferencian varias estrategias que combinan y extienden los conceptos vistos en la *regresión polinómica* y en las *step functions*.

Piecewise Polinomial

Se trata de una combinación directa de la regresión polinómica y de *step functions*. En lugar de ajustar un polinomio de grado alto a todo el rango del predictor X, este se divide en subintervalos y en cada uno de ellos se ajusta un polinomio de menor grado. Un *piecewise polinomial* de grado 3 emplea la función:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$$

donde los coeficientes $\beta_0,\ \beta_1,\ \beta_2,\ \beta_3$ toman diferente valor en cada una de las regiones en las que se divide el rango de X. A los puntos de corte que delimitan cada región o subintervalo se les denomina *knots* $(c_1, \ldots,\ c_k)$.

Si el modelo solo tiene un único punto de corte (*knot*) en el valor c, las funciones que describen el modelo son:

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{si } x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{si } x_i \geq c \end{cases}$$

Se ajustan dos funciones polinómicas por mínimos cuadrados. Para la primera se emplean las observaciones en las que $x_i < c$ obteniendo los coeficientes $\beta_{01},\ \beta_{11},\ \beta_{21},\ \beta_{31}$ y para la segunda aquellas en las que $x_i \geq c$, obteniendo los coeficientes $\beta_{02},\ \beta_{12},\ \beta_{22},\ \beta_{32}$.

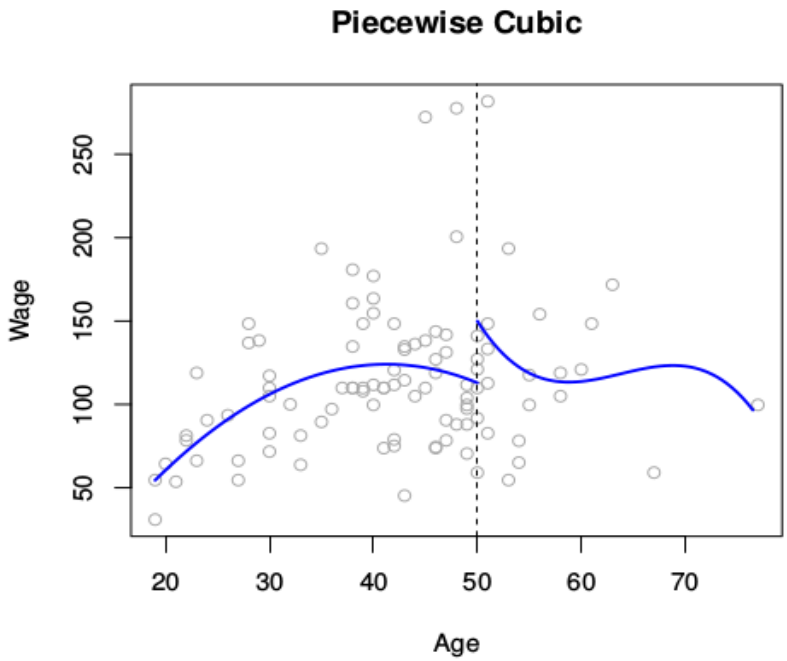


Imagen de un Piecewise Polinomial de grado 3 obtenida del libro ISLR

La flexibilidad de los modelos *piecewise polinomial* se puede controlar de dos formas:

- Con el número de puntos de corte (*knots*) introducidos. A mayor número mayor flexibilidad.
- Con el grado del polinomio empleado. El método *step functions* es un caso particular de *piecewise polinomial* en el que se emplea un polinomio de grado cero y por lo tanto el ajuste resultante es constante.

La desventaja de este método es que la función es discontinua, por lo que hay regiones ambiguas o de poca confianza.

Constrains and Splines

Para evitar la discontinuidad y exceso de flexibilidad de los *piecewise polinomial* se pueden imponer restricciones a los polinomios de cada región de forma que el modelo final sea una curva continua. La restricción más básica consiste en forzar a cada polinomio a pasar por los puntos de corte que lo flanquean, de esta forma, el polinomio de la región i termina en el mismo punto donde empieza el polinomio de la región $i + 1$.

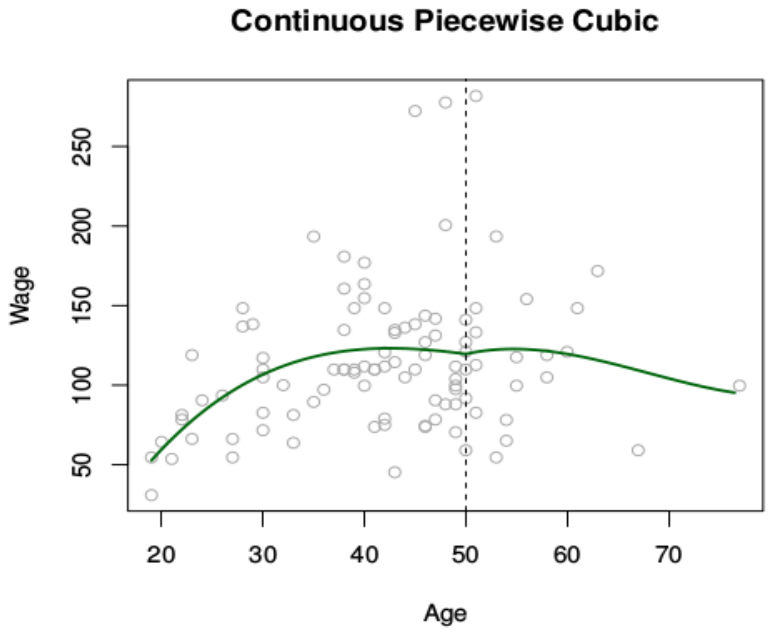


Imagen de un Piecewise Polinomial de grado 3 continuo obtenida del libro ISLR

A pesar de que esta restricción permite obtener una curva continua, el cambio de una región a otra es excesivamente abrupto y poco natural. Para conseguir que la transición de una región a otra, además de continua, sea suave, se imponen restricciones adicionales: que las *d-derivadas* de los polinomios sean continuas en los puntos de corte (que pasen por ellos), siendo *d* el grado del polinomio menos 1. En el caso de un *piecewise polinomial* de grado 3, la máxima continuidad de obtiene aplicando las siguientes restricciones:

- Los polinomios deben ser continuos. Cada polinomio debe pasar por los puntos de corte que lo delimitan.
- La primera derivada de los polinomios debe de ser continua. La primera derivada de cada polinomio debe pasar por los puntos de corte que lo delimitan.
- La segunda derivada de los polinomios debe de ser continua. La segunda derivada de cada polinomio debe pasar por los puntos de corte que lo delimitan.

A la curva final obtenida al imponer las restricciones de continuidad sobre un *piecewise polinomial de grado 3* se le conoce como *cubic spline* o *spline de grado 3*. Los *cubic spline* se emplean con frecuencia ya que a partir de la segunda derivada el ojo humano no aprecia la discontinuidad en los *knots*. A modo general, un *spline de grado-d* se define como un *piecewise polinomial de grado-d* con la condición de continuidad en las *d-1* derivadas en cada punto de corte (*knots*).

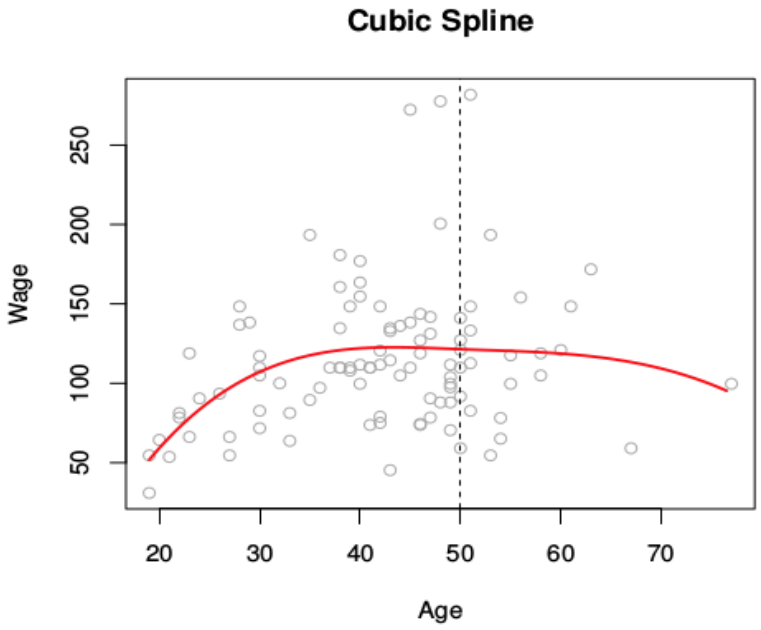


Imagen de un Cubic Spline de grado 3 continuo obtenida del libro ISLR

Para poder obtener el ajuste de un *cubic spline* con *K knots*, se emplea regresión por mínimos cuadrados sobre una ecuación formada por la intersección y $3 + K$ predictores:

$$\beta_0,\ X,\ X^2,\ X^3,\ h(X,\xi_1),\ h(X,\xi_2),\ldots,h(X,\xi_K)$$

donde $\xi_1),\ldots,\xi_K)$ son los *knots*.

El modelo resultante implica estimar un total de $4 + K$ coeficientes de regresión, por lo que el ajuste de un *cubic spline* con *K knots* tiene $4 + K$ grados de libertad.

Las *regression splines* pueden tener mucha varianza en los extremos superior e inferior del predictor, lo que genera intervalos de confianza muy amplios. Esto es así debido a que la primera y última región carecen de restricción de continuidad en uno de sus extremos, por lo que tienen un exceso de flexibilidad. Los *natural splines* evitan este problema incorporando una restricción extra a las *regression splines*: la función tiene que ser lineal en los extremos. Esto significa que las regiones para las que el predictor *X* es menor que el menor de los *knots* o mayor que el mayor de los *knots* siguen un ajuste lineal.

La siguiente imagen muestra un *natural cubic spline* y un *cubic spline*. Ambos son aproximadamente idénticos, con la excepción de que el *natural cubic spline* permite reducir la variabilidad en los extremos y, por lo tanto, tiene intervalos de confianza más estrechos.

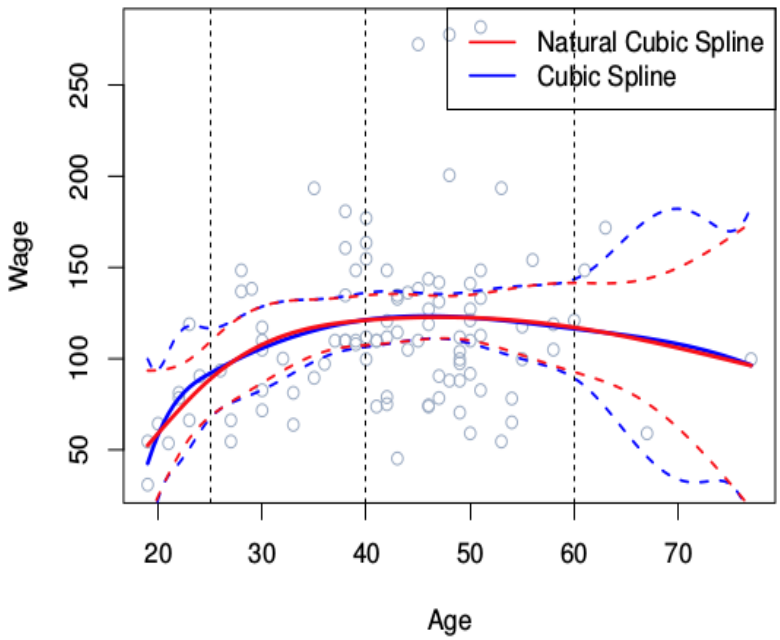


Imagen obtenida del libro ISLR

Posición y número de *knots*

El resultado de los métodos basados en *regression splines* depende en gran medida de la cantidad de *knots* que se introduzcan, así como de sus posiciones. Dado que cuantos más *knots* mayor la flexibilidad, una opción es concentrar un mayor número en regiones con alta varianza y menos en regiones estables. A pesar de que esta es una buena opción, en la práctica se suele preferir distribuir los *knots* de forma uniforme. Para conseguirlo, se indica al software empleado el número de grados de libertad que se desean y este distribuye el número de *knots* correspondientes en cuantiles uniformes.

A la hora de elegir el número de *knots* óptimo, o lo que es lo mismo, el número de grados de libertad (teniendo en cuenta que cada restricción de continuidad introducida reduce un grado de libertad) se recurre a *Cross Validation*. Tras calcular el *RSS* para un rango de *K knots*, se selecciona aquella cantidad para la que el *RSS* estimado es menor. Ver capítulo Validación de modelos de regresión: Cross-validation, OneLeaveOut, Bootstrap para más información sobre *Cross-Validation*.

Comparación de *regression splines* y regresión polinómica

El método de *regression splines* supera con frecuencia a la regresión polinómica por sufrir menos varianza. Los polinomios suelen necesitar grados altos para conseguir ajustes flexibles, sin embargo, los *regression splines* consiguen flexibilidad introduciendo mayor número de *knots* a la vez que mantienen los grados de libertad bajos. La segunda ventaja de los *regression splines* es que permiten controlar como se distribuye la flexibilidad del modelo mediante la localización y distribución de los *knots*.

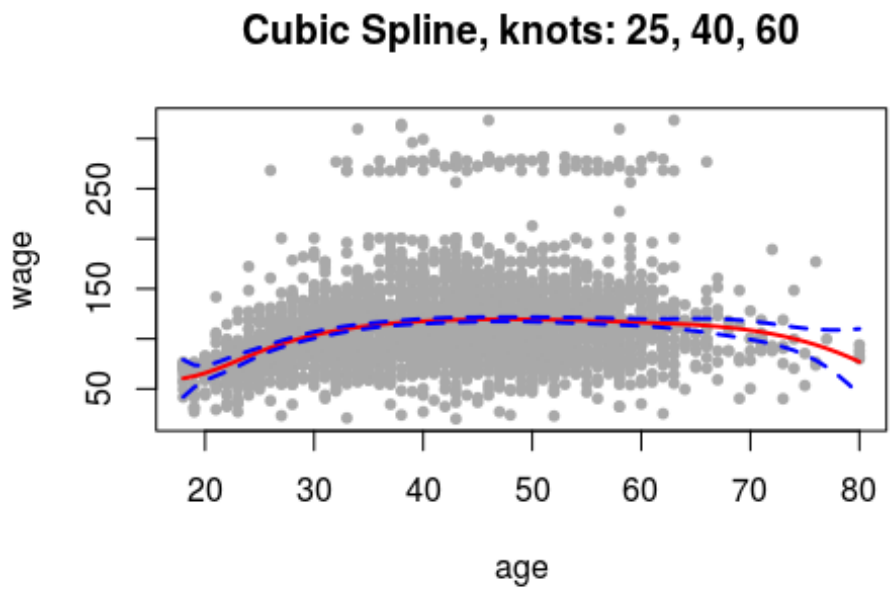
Ejemplo Regression Spline

Empleando el mismo set de datos Wage que en el ejercicio anterior, se pretende crear un modelo de regression splines que permita predecir el salario de un trabajador en función de la edad que tiene.

Para ajustar *regression splines* en `R` se emplea la librería `splines`, que viene instalada por defecto. La función `bs()`, cuyo nombre viene de *B-spline*, genera la matriz de ecuaciones necesaria para crear las *splines* acorde a los *knots* indicados. Por defecto, `bs()` se generan polinomios de grado 3 (*cubic splines*) pero esto puede cambiarse con el argumento `degree`.

```
# AUSTE DEL MODELO
# ------------------------------------------------------------------------------
library(splines)
modelo_splines <- lm(wage ~ bs(age, knots = c(25,40,60), degree = 3),
                     data = Wage)

# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
predicciones <- predict(modelo_splines, newdata = nuevos_puntos, se.fit = TRUE,
                        level = 0.95)
```

```
# CÁLCULO DEL INTERVALO DE CONFIANZA SUPERIOR E INFERIOR
# ------------------------------------------------------------------------------
intervalo_conf <- data.frame(
                  inferior = predicciones$fit - 1.96*predicciones$se.fit,
                  superior = predicciones$fit + 1.96*predicciones$se.fit)
attach(Wage)
plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("Cubic Spline, knots: 25, 40, 60")
lines(x = nuevos_puntos$age, predicciones$fit, col = "red", lwd = 2)
lines(x = nuevos_puntos$age, intervalo_conf$inferior, col = "blue",
      lwd = 2, lty = 2)
lines(x = nuevos_puntos$age, intervalo_conf$superior, col = "blue",
      lwd = 2, lty = 2)
```



Dado que se han especificado 3 *knots* en los valores 25, 40 y 60, la *spline* resultante tiene 6 funciones. Esto equivale a 7 grados de libertad, uno por el *intercept* más 6 por las funciones que lo forman.

Para que los *knots* estén equidistribuidos, en lugar de indicar las posiciones en el argumento `knots`, se indican los grados de libertad del *spline*.

```
# CUBIC SPLINE CON 3 KNOTS EQUIDISTRIBUIDOS
# ------------------------------------------------------------------------------
modelo_splines <- lm(wage ~ bs(age, df = 6, degree = 3), data = Wage)
attr(bs(age, df = 6, degree = 3), "knots")

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

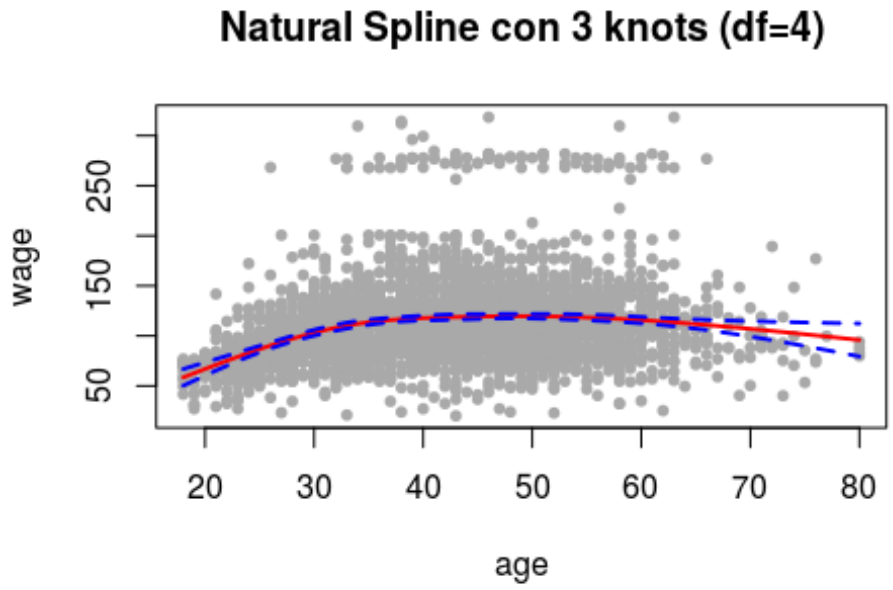
```
# CUBIC SPLINE CON 5 KNOTS EQUIDISTRIBUIDOS
# ------------------------------------------------------------------------------
modelo_splines <- lm(wage ~ bs(age, df = 8, degree = 3), data = Wage)
attr(bs(age, df = 8, degree = 3), "knots")

## 16.66667% 33.33333%       50% 66.66667% 83.33333%
##        30        37        42        48        54

# Si se aumenta el grado del polinomio y se mantienen los grados de
# libertad, el número de knots introducidos es menor.

attr(bs(age, df = 8, degree = 4), "knots")

## 20% 40% 60% 80%
##  32  39  46  53
```

Ejemplo Natural Spline

Para ajustar *natural splines* en `R`, el proceso es equivalente al seguido en el ajuste de *regression splines* pero empleando la función `ns()`. Dado que las *naturals spline* introducen dos restricciones adicionales (linealidad en los extremos), se necesitan dos grados de libertad menos en comparación con las *regression splines* para introducir el mismo número de *knots*.

```
# AUSTE DEL MODELO
# ------------------------------------------------------------------------------
library(splines)
attach(Wage)
modelo_nsplines <- lm(wage ~ ns(age, df = 4), data = Wage)
attr(ns(age, df = 4), "knots")

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

# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
predicciones <- predict(modelo_nsplines, newdata = nuevos_puntos, se.fit = TRUE,
                        level = 0.95)
```

```
# CÁLCULO DEL INTERVALO DE CONFIANZA SUPERIOR E INFERIOR
# ------------------------------------------------------------------------------
intervalo_conf <- data.frame(inferior = predicciones$fit -
                                        1.96*predicciones$se.fit,
                             superior = predicciones$fit +
                                        1.96*predicciones$se.fit)

plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("Natural Spline con 3 knots (df=4)")
lines(x = nuevos_puntos$age, predicciones$fit, col = "red", lwd = 2)
lines(x = nuevos_puntos$age, intervalo_conf$inferior, col = "blue",
      lwd = 2, lty = 2)
lines(x = nuevos_puntos$age, intervalo_conf$superior, col = "blue",
      lwd = 2, lty = 2)
```



Smoothing Splines

El concepto de *smoothing splines* es similar al de *regression splines* descrito anteriormente, la diferencia está en la estrategia seguida para obtener la curva final.

El objetivo cuando se ajusta una *smooth curve* es encontrar una función *g(x)* que se ajuste bien a las observaciones, es decir, que minimice la suma de los residuos al cuadrado $RSS = \sum_{i=1}^{n}(y_i - g(x_i))^2$. El problema de esta aproximación es que si no se añade alguna restricción sobre *g(x)* se obtiene una función cuyo $RSS = 0$, una curva que pasa por todos los puntos. Esto tiene como resultado un modelo excesivamente flexible con *overfitting*. Una forma de evitarlo es seguir el mismo concepto *Loss + Penalty* empleado en *lasso* y *ridge regression*, introduciendo en la ecuación un elemento de penalización de forma que *g(x)* pase a ser la función que minimiza:

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2\, dt$$

El término $\sum_{i=1}^{n}(y_i - g(x_i))^2$ se conoce como *loss function* y es la parte que hace que *g(x)* tienda ajustarse a los datos, el término $\lambda \int g''(t)^2\, dt$ se conoce como *penalty* y penaliza la variabilidad de *g(x)* haciendo que sea *smooth* (que no sea excesivamente flexible). A continuación, se describe en detalle este último término.

La primera derivada (*g'(t)*) mide la pendiente de la función *g* en el valor *t* y la segunda derivada (*g''(t)*) mide la variación en la pendiente. La segunda derivada puede interpretarse como una medida de irregularidad, (la segunda derivada de una recta es 0). Una integral $\int$ es el sumatorio a lo largo de un rango *t*, por lo que, $\int g''(t)^2\, dt$ es una medida de la variación total de *g'(t)* en todo su rango, es decir, de la variación total de la pendiente. Si la curva descrita por la función *g* es *smooth*, entonces *g'(t)* será prácticamente constante y *g''(t)* tomará valores muy pequeños. Por lo contrario, si *g* es irregular, *g'(t)* variará sustancialmente y *g''(t)* tomará valores altos.

El término λ es un *tunning parameter* positivo que determina el impacto de la penalización, por lo tanto, controla el balance entre bias y varianza del *smoothing spline*. Cuando $\lambda = 0$, el penalti no tiene efecto y la función *g* se ajusta perfectamente a todos los datos. Cuando $\lambda \to \infty$, *g* es totalmente *smooth* por lo que describe una recta que pasa lo más cerca posible a todos los puntos. En este caso, el resultado es equivalente a un ajuste lineal por mínimos cuadrados.

El resultado de la curva *g(x)* que minimiza la función $\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2\, dt$ resulta ser un *natural cubic spline* con *knots* en cada uno de los valores $x1, x2, \ldots, xn$, sin embargo, no es el mismo *natural cubic spline* que se obtendría mediante el proceso descrito en la sección *Regression Splines*.

En el apartado *posición y número de knots* se estudió cómo la introducción de *knots* aumenta los grados de libertad y por ende la flexibilidad del modelo. A priori, un *spline* con *knots* en cada observación x_i tendría una flexibilidad muy por encima de lo deseado debido a la gran cantidad de grados de libertad, sin embargo, la restricción que impone el *tunning parameter* λ hace que los *effective dregrees of fredom* sean mucho menores. El término *effective dregrees of fredom* (df_λ) puede entenderse como el impacto real de los grados de libertad sobre la flexibilidad de un modelo. En la mayoría de casos, los grados de libertad hacen referencia al número de parámetros libres, como por ejemplo el número de coeficientes de regresión en un modelo, de ahí que sean un indicativo de la flexibilidad del modelo. Cuando se imponen restricciones (*shrinkage*), la libertad de los parámetros se reduce, por lo que, aunque el número de grados de libertad se mantiene, la flexibilidad disminuye. Es por esta razón por la que en modelos con restricciones se habla de grados de libertad efectivos.

Elección del parámetro λ

A diferencia de las *regression splines*, en las *smooth splines* no se tiene que elegir el número y posición de los *knots*, ya que hay uno en cada observación. En su lugar, se tiene que escoger un valor de λ que determine como de estricta es la penalización. Una forma de encontrar el λ óptimo es mediante *cross-validation*, ya que, por las características matemáticas, el método *leave-one-out* puede implementarse de forma muy eficiente.

Ejemplo

De nuevo, se pretende crear un modelo que permita predecir el salario de un trabajador en función de la edad que tiene, esta vez empleando smooth splines.

La función `smooth.spline()` de `R` permite ajustar *smooth splines* de forma sencilla, con la ventaja añadida de que el valor óptimo de *smothness* (λ) puede identificarse por *cross-validation*.

```
# AUSTE DEL MODELO
# ------------------------------------------------------------------------------
attach(Wage)
modelo_smooth_splines <- smooth.spline(wage ~ age, cv = TRUE)
modelo_smooth_splines$df
```

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

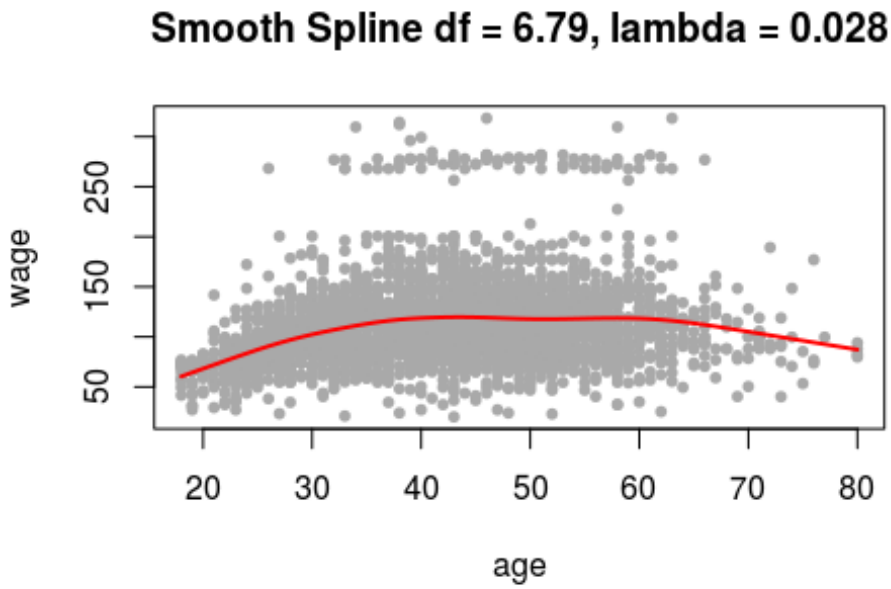
```
modelo_smooth_splines$lambda
```

```
## [1] 0.02792303
```

```
modelo_smooth_splines$spar
```

```
## [1] 0.6988943
```

```
# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
predicciones <- predict(modelo_smooth_splines, newdata = nuevos_puntos)
# predict() no devuelve el error de la predicción de un modelo smooth.spline

plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("Smooth Spline df = 6.79, lambda = 0.028")
lines(x = predicciones$x, predicciones$y, col = "red", lwd = 2)
```



Local Regression

La regresión local (*LOESS* o *LOWESS*) es otra aproximación para obtener modelos no lineales. Se caracteriza por ajustar modelos simples a distintas subregiones del rango del predictor empleando únicamente las observaciones cercanas, evitando así recurrir a una única función global compleja. El concepto puede recordar al descrito en *regression splines* y *smoothing splines*, de hecho, es muy similar, la diferencia es que en la regresión local las regiones son solapantes, cosa que no ocurre en los otros dos métodos.

A continuación, se describe el algoritmo con el que se obtiene un *LOESS*:

Para cada punto x_0 del set de datos, se ajusta un polinomio de grado bajo empleando únicamente las observaciones cercanas a esa posición. El ajuste se realiza mediante *weighted least squares regression*, dando más peso a los puntos cercanos a x_0 y menos cuanto más alejados están. El ajuste *LOESS* finaliza cuando el proceso se ha repetido para cada una de las n observaciones del set de datos.

Para cada posición x_0 ocupada por una observación:

1. Se identifican la fracción $s = k/n$ de observaciones más cercanas al punto x_0.
2. Se asigna un peso a cada observación vecina (dentro de las seleccionadas en el paso 1) de forma que la observación más alejada tiene peso 0 y la más cercana tiene el mayor peso. Todas las demás observaciones del set de datos tienen peso 0.
3. Ajuste *weighted least squares regression*.
4. El valor predicho de x_0 viene dado por el ajuste obtenido en el paso 3.

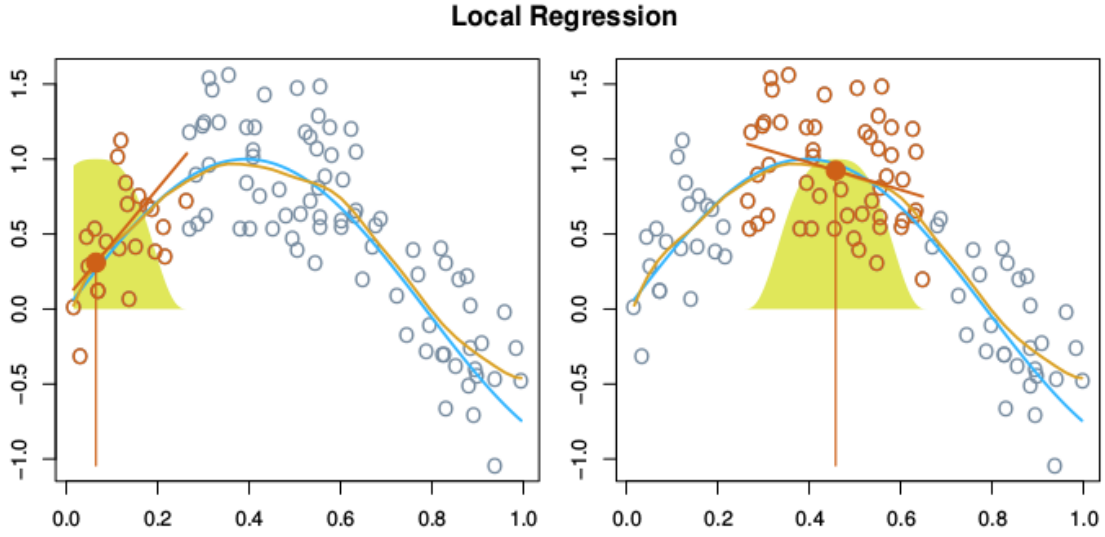


Imagen de regresión local obtenida del libro ISLR

En la imagen se muestra la idea detrás del método *LOESS* para dos puntos distintos. La curva azul representa la verdadera función que da lugar a las observaciones y la línea amarilla representa la función estimada por *LOESS*. Los puntos naranja representan las posiciones objetivo x_0 y la zona sombreada indica el peso asignado a cada observación vecina, disminuyendo a medida que se aleja de x_0.

A la hora de realizar una regresión local se tienen que especificar múltiples parámetros, a continuación se describe brevemente las implicaciones de cada uno.

Span

El *span* es el parámetro más importante y con mayor impacto en la regresión local. Su función es similar al parámetro λ en las *smoothing splines*, controla la flexibilidad del ajuste. Cuanto menor es el *span*, menor la fracción de observaciones vecinas empleadas en cada ajuste local y, por lo tanto, mayor flexibilidad del modelo. En contraposición, valores altos de *span* hacen que en cada ajuste local se empleen muchas observaciones vecinas generando un ajuste más robusto, pero menos flexible. La identificación del valor óptimo de *span* puede hacerse mediante *cross-validation*.

Grado del polinomio

El polinomio local empleado para ajustar cada subset de datos suele ser siempre de primer o segundo grado, es decir, o bien un ajuste lineal o bien uno cuadrático. Aunque desde el punto de vista teórico se pueden emplear polinomios de mayor grado, estos tienden a producir *overfit* y reducen la precisión del modelo.

Función para designar pesos

Como se ha mencionado previamente, la función de peso atribuye mayor importancia a las observaciones cercanas al punto que se está estimando y menor a las observaciones más alejadas. La función de peso que se emplea con más frecuencia es la *tri-cube weight function*.

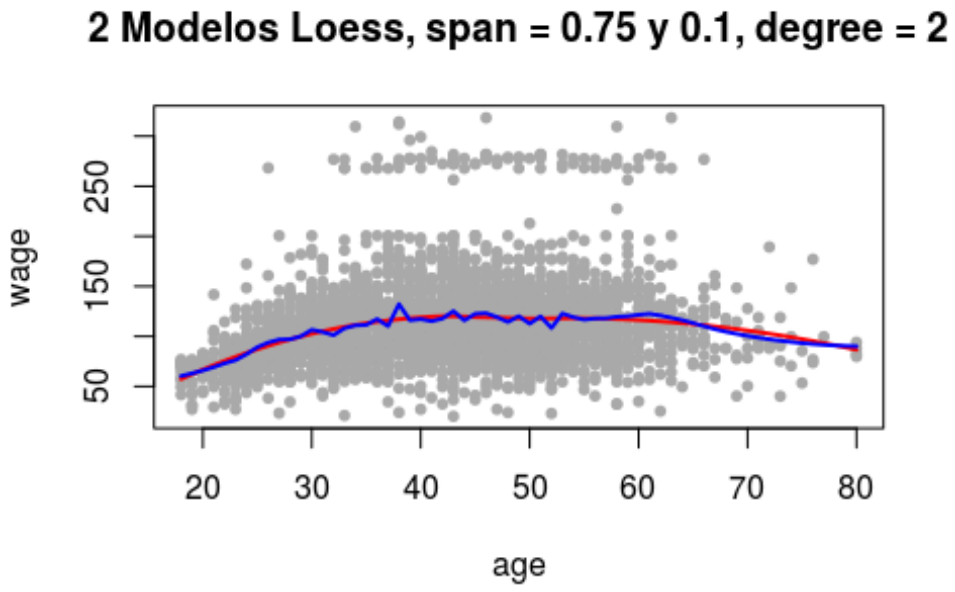
$$w(x) = (1 - |x|^3)^3 \; I[|x| < 1]$$

El método de regresión local se puede generalizar fácilmente a varios predictores, sin embargo, a partir de 3 o 4 la capacidad del modelo se reduce en gran medida ya que por lo general hay muy pocas observaciones cercanas a x_0. (Problema de dimensionalidad).

Ejemplo

La función `loess()` del paquete base de `R` permite obtener modelos ajustados por regresión local. Por defecto se emplea un polinomio de grado 2 y el *span* de 0.75, pero estos parámetros se pueden modificar para aumentar o disminuir la flexibilidad del modelo.

```
# AUSTE DEL MODELO
# ------------------------------------------------------------------------------
attach(Wage)
modelo_local_reg1 <- loess(wage ~ age, span = 0.75, degree = 2, data = Wage)
# Se ajusta un segundo modelo con un span menor
modelo_local_reg2 <- loess(wage ~ age, span = 0.1, degree = 2, data = Wage)

# INTERPOLACIÓN DE PUNTOS DENTRO DEL RANGO DEL PREDICTOR
# ------------------------------------------------------------------------------
limites <- range(Wage$age)
nuevos_puntos <- seq(from = limites[1], to = limites[2], by = 1)
nuevos_puntos <- data.frame(age = nuevos_puntos)

# PREDICCIÓN DE LA VARIABLE RESPUESTA Y DEL ERROR ESTÁNDAR
# ------------------------------------------------------------------------------
predicciones1 <- predict(modelo_local_reg1, newdata = nuevos_puntos)
predicciones2 <- predict(modelo_local_reg2, newdata = nuevos_puntos)
# No es posible calcular el error de la predicción de un modelo loess


plot(x = age, y = wage, pch = 20, col = "darkgrey")
title("2 Modelos Loess, span = 0.75 y 0.1, degree = 2")
lines(x = nuevos_puntos$age, predicciones1, col = "red", lwd = 2)
lines(x = nuevos_puntos$age, predicciones2, col = "blue", lwd = 2)
```



La representación de los modelos muestra que, para un mismo grado de polinomio, cuanto menor es el valor de *span* más se ajusta el modelo a las observaciones.

Con la función `geom_smooth()` del paquete gráfico `ggplot2` también se pueden obtener representaciones gráficas de un modelo ajustado por regresión local. En este caso, el grado del polinomio por defecto es 1.

```
library(ggplot2)
ggplot(data = Wage, aes(x = age, y = wage)) +
  geom_point(color = "darkgray") +
  geom_smooth(formula = y ~ poly(x, 2), span = 0.75, se = TRUE, level = 0.95,
              color = "firebrick") +
  theme_bw() +
  labs(title = "Modelo Loess, span = 0.75, degree = 1")
```

Generalized Additive Models (GAM). Regresión no lineal con múltiples predictores.

Los modelos *GAM* permiten obtener ajustes no lineales empleando múltiples predictores. Son el resultado de extender un modelo lineal múltiple permitiendo que cada elemento del modelo sea una función no lineal de un predictor y manteniendo la aditividad. Son por lo tanto una combinación lineal de funciones no lineales. Al igual que ocurre con los modelos lineales múltiples, se pueden incorporar tanto predictores continuos como cualitativos.

Partiendo de un modelo de regresión lineal múltiple con p predictores

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon_i$$

se pueden incorporar relaciones no lineales entre los predictores y la variable respuesta reemplazando cada componente lineal $\beta_j x_{ij}$ por una función no lineal $f_j(x_{ij})$

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j\,(x_{ij}) + \epsilon_i$$

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_p(x_{ip}) + \epsilon_i$$

El resultado es un modelo aditivo en el que se calcula una función no lineal f_j separada para cada predictor X_j y luego se suman todas sus contribuciones.

La estructura de bloques característica de los *GAM* permite aplicar las diferentes técnicas de ajuste no linear descritas en los apartados anteriores a cada uno de los predictores, logrando una gran flexibilidad. La siguiente imagen muestra el resultado de ajustar el modelo $wage = \beta_0 + f_1(year) + f_2(age) + f_3(education) + \epsilon$ empleando *natural splines* para los dos primeros predictores y una constante distinta para cada nivel del tercer predictor (*dummy* variables).

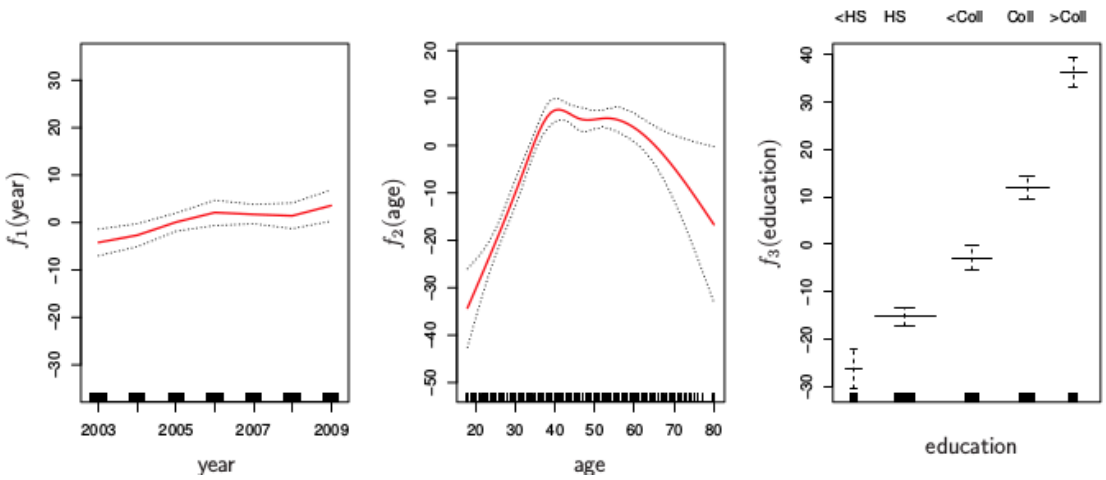


Imagen de un GAM obtenida del libro ISLR

Que los modelos *GAM* sean aditivos permite analizar la influencia de cada predictor sobre la variable respuesta de forma individual, algo muy útil a la hora de hacer inferencia. El panel izquierdo de la imagen anterior muestra que, mantenido constantes las variables *age* y *education*, la variable respuesta *wage* aumenta con *year*. El panel central muestra que, si se mantienen constantes *year* y *education*, *wage* tiende a ser mayor para valores intermedios de *age*. Por último, el panel derecho indica que, manteniendo constantes *year* y *age*, *wage* se incrementa a medida que lo hace la educación.

Si bien en este ejemplo se han utilizado *natural splines* para los predictores continuos y constantes para el predictor cualitativo, se podrían haber empleado otros métodos como regresión polinómica, *smooth splines*..., o cualquier combinación de los mismos.

GAM para problemas de clasificación. Logistic regression GAM

Los modelos *GAM* también pueden aplicarse a problemas de clasificación en los que la variable respuesta Y es cualitativa, permitiendo modular el logaritmo de la probabilidad. Para ello, se parte de un modelo de regresión logística múltiple

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

y se sustituye cada componente lineal $\beta_j x_{ij}$ por una función no lineal $f_j(x_{ij})$

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p)$$

Ventajas y desventajas de los GAM

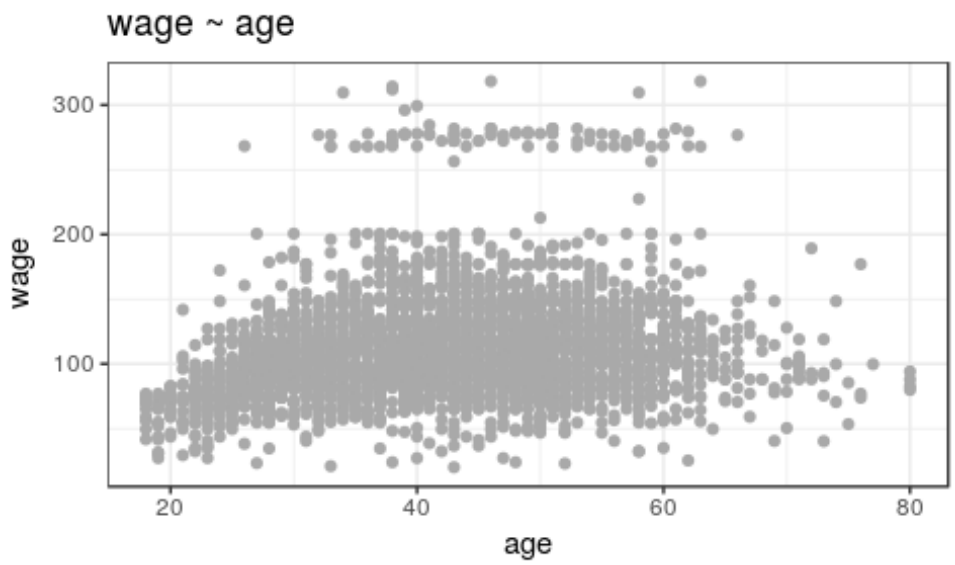
- Los *GAM* permiten ajustar una función no lineal f_j distinta a cada predictor X_j. Esto hace posible incorporar la relación de cada predictor con la variable respuesta de forma precisa.
- Dado que el modelo es aditivo, se puede estudiar el efecto de cada predictor X_j sobre la variable Y, manteniendo constantes el resto de predictores. Para esto, las representaciones gráficas son de mucha ayuda.
- La flexibilidad de cada función f_j puede, tanto controlarse como resumirse, mediante los grados de libertad.
- Los *GAM* son un buen punto intermedio entre los modelos lineales y los modelos no paramétricos (*random forest*, *boosting* ...).
- La mayor limitación de los modelos *GAM* es que, al ser aditivos, no contemplan interacciones entre predictores de forma automática. Al igual que en los modelos lineales múltiples, se puede introducir manualmente interacciones mediante la creación de nuevos predictores de tipo $X_j x X_k$.

Ejemplo

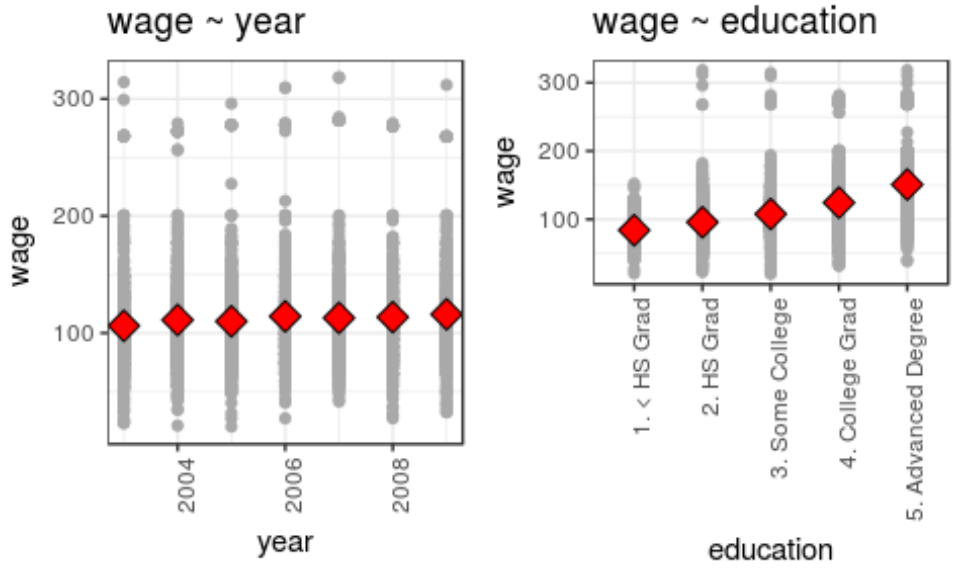
Empleando el mismo set de datos Wage que en el ejercicio anterior, se pretende crear un modelo GAM que permita predecir el salario (wage) de un trabajador en función de su edad (age), año en el que se registró la información (year) y nivel educativo (education).

```
library(ggplot2)
library(gridExtra)

p1 <- ggplot(data = Wage, aes(x = age, y = wage)) +
        geom_point(color = "darkgray") +
        theme_bw() +
        labs(title = "wage ~ age")

p2 <- ggplot(data = Wage, aes(x = year, y = wage)) +
        geom_point(color = "darkgray") +
        stat_summary(fun.y = "mean", fill = "red", size = 4,
                     geom = "point", shape = 23) +
        theme_bw() +
        theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
        labs(title = "wage ~ year")
```

```
p3 <- ggplot(data = Wage, aes(x = education, y = wage)) +
        geom_point(color = "darkgray") +
        stat_summary(fun.y = "mean", fill = "red", size = 4,
                     geom = "point", shape = 23) +
        theme_bw() +
        theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
        labs(title = "wage ~ education")

p1
```




```
grid.arrange(p2, p3, ncol = 2)
```



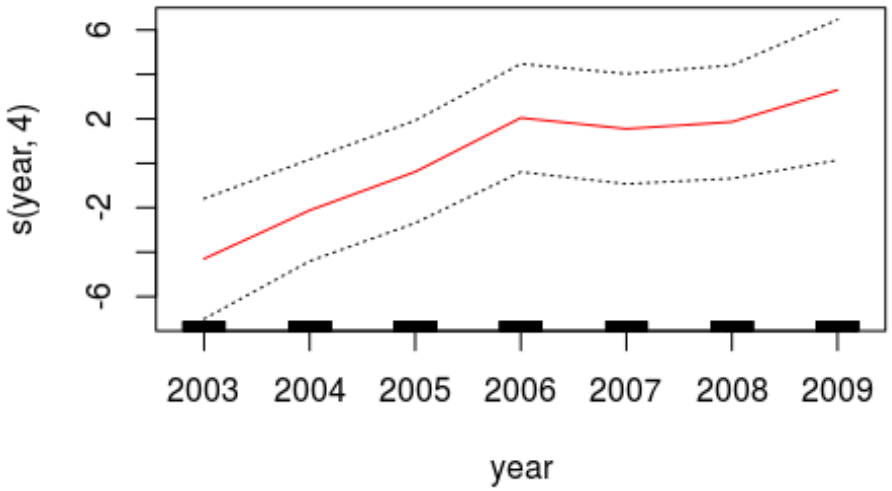
Los predictores *age* y *year* son variables continuas. La relación entre *age* y *wage* es claramente no lineal, mientras que en el caso de *year* no queda claro a simple vista si es o no lineal. El predictor *education* es de tipo cualitativo con 5 niveles.

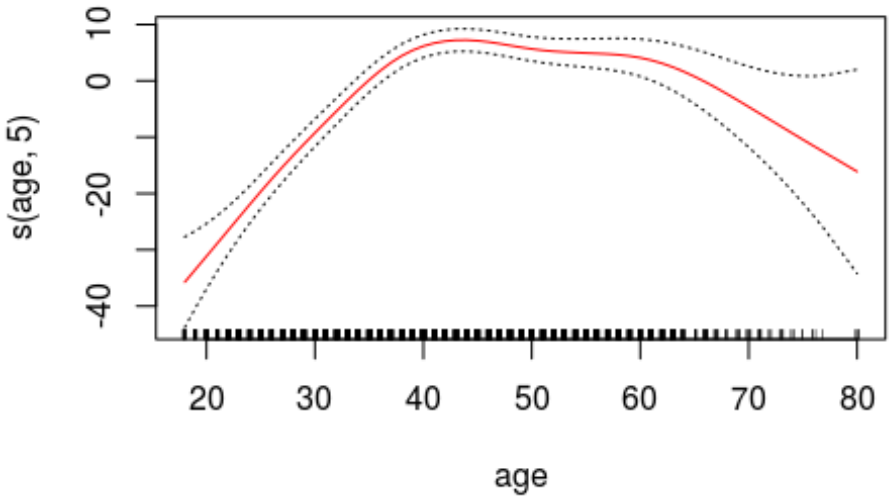
Dado que un modelo *GAM* es en esencia un modelo de regresión lineal múltiple formado por funciones independientes para cada predictor, se puede ajustar empleando la función `lm()`. Se procede a ajustar un GAM que contenga *natural splines* para las variables continuas y constantes *dummy* para la variable cualitativa.

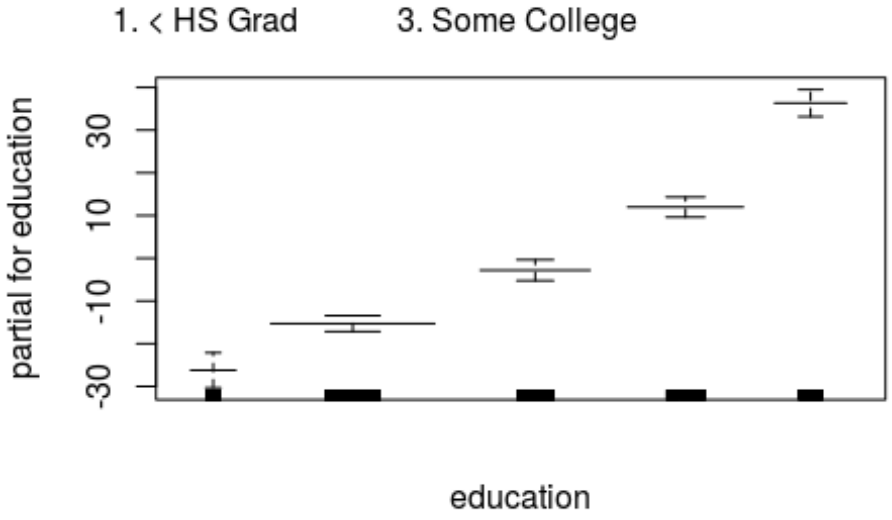
```
library(splines)
modelo_gam <- lm(wage ~ ns(year, 4) + ns(age, 5) + education, data = Wage)
# plot.gam(modelo_gam)
```

Para crear modelos *GAM* más complejos, que incluyan *smooth splines* o *local regression* se emplean las funciones `gam()`, `s()` y `lo()` del paquete `gam`. A continuación se crea un modelo similar al anterior pero esta vez empleando una *smooth spline* con 4 grados de libertad para *year* y otra con 5 grados de libertad para *age*. Una vez ajustado, se obtiene una representación gráfica de cada una de las funciones que componen el modelo mediante la función `plot.gam()`. Esto es aplicable tanto a modelos *GAM* obtenidos con la función `gam()` como con `lm()`.

```
library(gam)
modelo_gam <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
plot.gam(modelo_gam, se = TRUE, col = "red")
```







El `summary()` de un modelo *gam* genera un tabla resumen del ajuste. En la sección *Anova for Nonparametric Effects* se muestran *p-values* para cada predictor. Estos *p-values* se corresponden con el contraste de hipótesis de que la relación entre predictor y variable respuesta es lineal, frente a la alternativa de que no lo es.

```
summary(modelo_gam)
```

```
##
## Call: gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -119.43  -19.70   -3.33   14.17  213.48
##
```

```
## (Dispersion Parameter for gaussian family taken to be 1235.69)
##
##     Null Deviance: 5222086 on 2999 degrees of freedom
## Residual Deviance: 3689770 on 2986 degrees of freedom
## AIC: 29887.75
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##              Df  Sum Sq Mean Sq F value    Pr(>F)
## s(year, 4)    1   27162   27162  21.981 2.877e-06 ***
## s(age, 5)     1  195338  195338 158.081 < 2.2e-16 ***
## education     4 1069726  267432 216.423 < 2.2e-16 ***
## Residuals  2986 3689770    1236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar F  Pr(F)
## (Intercept)
## s(year, 4)        3  1.086 0.3537
## s(age, 5)         4 32.380 <2e-16 ***
## education
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El *p-value* obtenido para la función del predictor *year* (0.35) no muestra evidencias de que la relación entre *wage* y *year* no sea lineal, lo que lleva a preguntarse si sería mejor emplear un ajuste lineal en lugar de una *smooth spline*, reduciendo así la complejidad del modelo. Un análisis ANOVA sirve para dar respuesta a esta pregunta. A continuación, se comparan 3 posibles modelos de menor a mayor complejidad: Un modelo (*m_1*) que no contiene el predictor *year*, un modelo (*m_2*) que emplea una función lineal para *year* y un tercer modelo (*m_3*) que emplea *smooth spline*.

```
library(gam)
m_1 <- gam(wage ~  s(age, 5) + education, data = Wage)
m_2 <- gam(wage ~ year + s(age, 5) + education, data = Wage)
m_3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
# método anova para objetos de tipo gam
anova(object = m_1, m_2, m_3, test = "F")
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
```

```
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance       F    Pr(>F)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 14.4771 0.0001447 ***
## 3      2986    3689770  3   4071.1  1.0982 0.3485661
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El ANOVA muestra una clara evidencia (*p-value = 0.00014*) de que el modelo *GAM* con una función lineal de *year* es superior a un *GAM* que no incluya este predictor. Sin embargo, no hay evidencia de que una función no lineal de *year* sea necesaria (*p-value = 0.35*). Acorde al principio de parsimonia, el modelo *m_2* es el mejor.

Al igual que con todos los tipos de modelos anteriormente explicados, la función `predict()` también es aplicable a modelos generados con `gam()`.

Ejemplo GAMs para regresión logística

Empleando el mismo set de datos Wage que en el ejercicio anterior, se pretende crear un modelo logístico que permita predecir la probabilidad de que un trabajador cobre más de 250.000 dólares en función de la edad que tiene.

Los modelos *GAM* pueden emplearse también para regresión logística. Solo es necesario convertir la variable respuesta en tipo dicotómico e indicar en el argumento de la función `gam()` que `family = binomial`.

Cuando se generan modelos logísticos que incluyen un predictor cualitativo, es importante comprobar si hay algún nivel del predictor para el cual ninguna observación es de tipo verdadero. Por ejemplo, no hay ningún trabajador con nivel educativo inferior a *HS Grad* y que tenga un salario mayor de 250.000 dólares.

```
table(Wage$education, I(Wage$wage > 250))
```

```
##
##                      FALSE TRUE
##   1. < HS Grad         268    0
##   2. HS Grad           966    5
##   3. Some College      643    7
##   4. College Grad      663   22
##   5. Advanced Degree   381   45
```

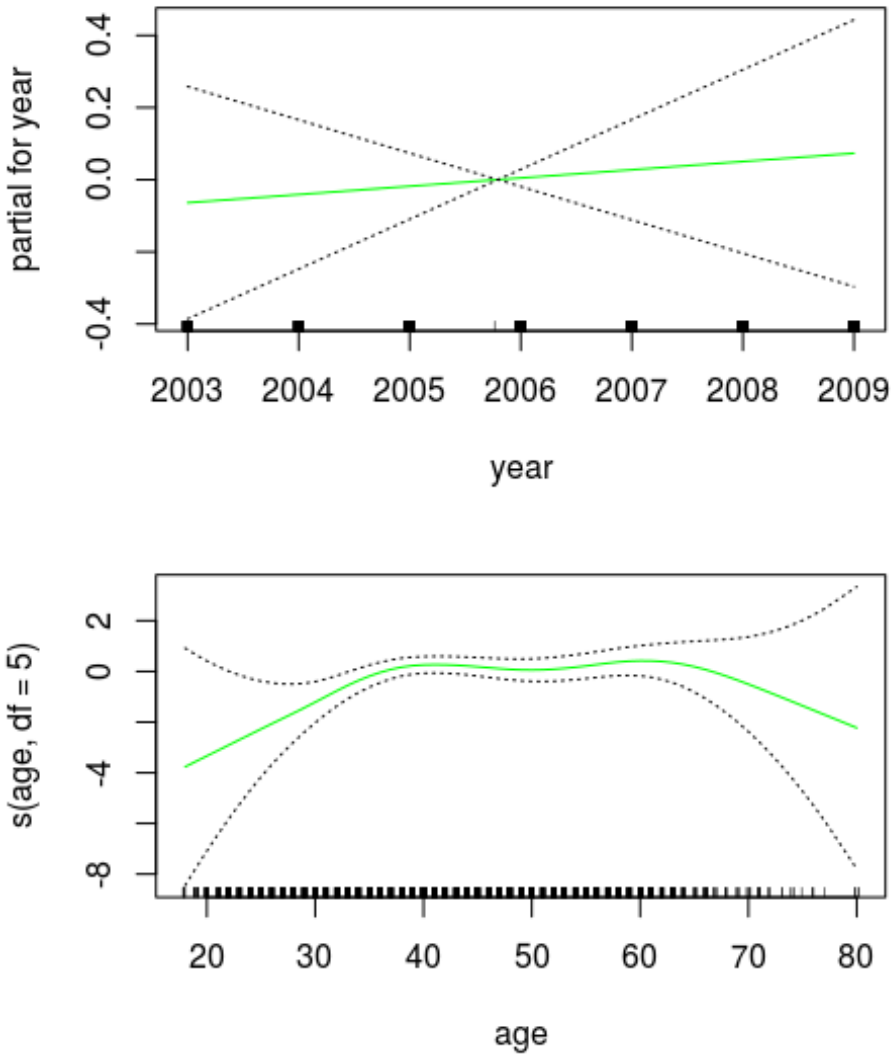
Por lo tanto, basándose en los datos disponibles, la probabilidad de que un trabajador con educación < *HS Grad* tenga un salario superior a 250.000 es cero. Para que estas observaciones, cuya predicción está clara, no influyan en el resto del modelo es conveniente excluirlas. De esta forma se suele mejorar la precisión.

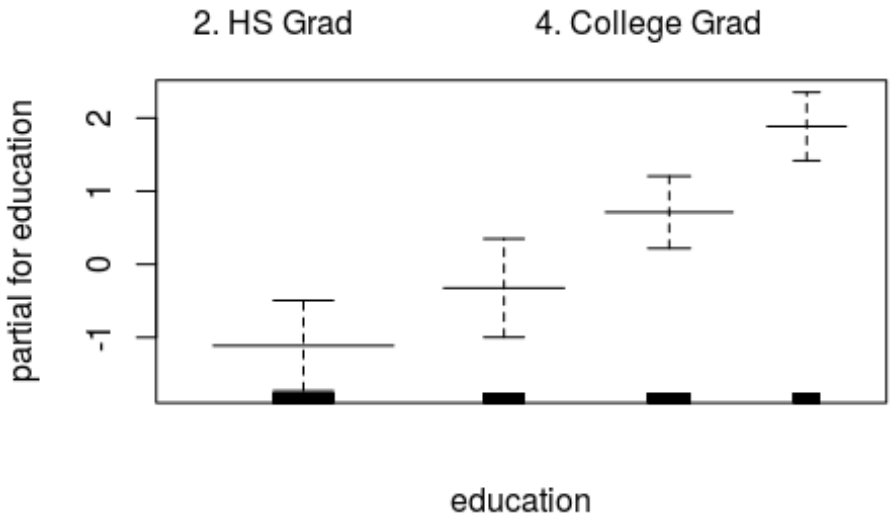
```
library(gam)
modelo_gam_logit <- gam(I(wage > 250) ~ year + s(age, df = 5) + education,
                        family = "binomial", data = Wage,
                        subset = (education != "1. < HS Grad"))
summary(modelo_gam_logit)
```

```
##
## Call: gam(formula = I(wage > 250) ~ year + s(age, df = 5) + education,
##     family = "binomial", data = Wage, subset = (education !=
##         "1. < HS Grad"))
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5821 -0.2760 -0.1415 -0.1072  3.3124
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 715.5412 on 2731 degrees of freedom
## Residual Deviance: 602.4588 on 2722 degrees of freedom
## AIC: 622.4586
##
## Number of Local Scoring Iterations: 11
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq F value    Pr(>F)
## year               1    0.48  0.4845  0.5459   0.46004
## s(age, df = 5)     1    3.83  3.8262  4.3116   0.03795 *
## education          3   65.80 21.9339 24.7166 8.933e-16 ***
## Residuals       2722 2415.55  0.8874
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                 Npar Df Npar Chisq  P(Chi)
## (Intercept)
## year
## s(age, df = 5)        4     10.364 0.03472 *
## education
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Al tratarse de un modelo con más de 2 predictores, no se puede hacer una representación gráfica simultánea del modelo logístico. En lugar de eso, se suele representar la influencia de cada predictor.

```
plot.gam(modelo_gam_logit, se = TRUE, col = "green")
```





Bibliografía

Introduction to Statistical Learning, Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

Wikipedia: https://en.wikipedia.org/wiki/Spline_(mathematics)

Wikipedia: https://en.wikipedia.org/wiki/Spline_interpolation