Regresión logística. Tratamiento computacional con R.

José Luis Cañadas Reche

Junio 2013

Máster en Estadística Aplicada.

Departamento de Estadística e Investigación Operativa.

Facultad de Ciencias. Universidad de Granada

Línea de Investigación: Modelos de respuesta discreta en R y aplicación con datos reales

Tutor: Prof. D. Manuel Escabias Machuca

A mi madre y a mi hermano

Índice general

Capítulo 1

Introducción

En este trabajo se describe detalladamente el proceso de ajuste, inferencia y validación de los modelos de regresión logística mediante el uso del lenguaje R. R es software libre, con una gran comunidad de usuarios, muchos de los cuales desarrollan *paquetes* orientados a técnicas concretas, y que ponen a disposición del resto de usuarios. El uso de R ha aumentado considerablemente en los últimos años, pasando del ámbito académico e investigador a entornos de producción empresariales, tal es el caso de grandes empresas como Google, Facebook u Oracle. Para algunos, R se ha convertido en la *lingua franca* de la estadística, ya que es en este lenguaje dónde se implementan en primer lugar las nuevas técnicas de análisis de datos.

Por otra parte, la regresión logística es una de las técnicas más conocidas y utilizadas para modelar una variable de respuesta categórica en función de variables predictoras continuas o categóricas. Forma parte de los modelos lineales generalizados, introducidos por (McCullagh and Nelder, 1989) y se aplica en campos tan distintos como la epidemiología, ecología, sociología o en los sectores bancario y asegurador.

Este trabajo está enfocado de forma que, para cada aspecto teórico del análisis, se presenta un ejemplo de cómo llevarlo a cabo con R . Con ese fin, se han utilizado los datos de un estudio del INE sobre el uso de las tecnologías de la información y de la comunicación en los hogares españoles (TIC-H 2011) y más concretamente la muestra para Andalucía compuesta por 3.485 encuestas. La variable dependiente es el uso o no de internet, y como variables independientes se consideraron la edad, el sexo, el hábitat y el nivel de estudios de la persona seleccionada en el hogar.

Con objeto de recoger todas las fases del análisis estadístico se ha divido el trabajo en 5 capítulos.

- Capítulo 1 (Introducción). Breve introducción del trabajo, objetivos del mismo y descripción de su estructura.
- Capítulo 2 (Modelos de respuesta discreta). En este capítulo se realiza una introducción teórica a los modelos de regresión logística tanto simples como múltiples, su relación con los modelos lineales generalizados, la formulación del modelo e interpretación de los parámetros estimados.
- Capítulo 3 (Ajuste del modelo). Se describe el conjunto de datos utilizado y cómo realizar el ajuste utilizando la función glm en R. También se tratan las dos formas de considerar los datos, agrupados y sin agrupar, y por último se analiza la introducción en el modelo de variables explicativas categóricas y ordinales y la inclusión en el modelo de la interacción entre variables explicativas.

- Capítulo 4 (Inferencia). Con el fin de extrapolar los resultados del modelo a la población es necesario realizar inferencia. En este capítulo se describe el uso de los contrastes de hipótesis e intervalos de confianza sobre los parámetros, así como la evaluación de si el modelo se ajusta globalmente bien a los datos. En este último caso, se presentan tanto los contrastes clásicos basados en los estadísticos X^2, G^2 o el de Hosmer-Lemeshow, como medidas tipo R^2 y medidas basadas la tabla de clasificación. Por último se explican brevemente los algoritmos de selección automática de variables y la comparación de múltiples modelos utilizando criterios de información.
- Capítulo 5 (Diagnóstico y validación). Se explica en detalle el análisis de los residuos y el cálculo de los valores influyentes, tanto analítica como gráficamente. Se comenta brevemente como detectar si existe colinealidad entre las variables predictoras y por último se explica como realizar validación cruzada con R.

Al final del trabajo se añaden dos anexos. En el primero se comentan algunos algoritmos para el ajuste de modelos de regresión logística y cómo se puede utilizar R para su resolución, y en el segundo se incluye la definición de la *devianza ,* su relación con la verosimilitud de un modelo de regresión logística y por qué es distinta si los datos están agrupados o no.

Capítulo 2

Modelos de respuesta discreta.

En el análisis de datos es frecuente encontrarse con variables dicotómicas (sí/no, presencia /ausencia), o variables medidas en escala ordinal (satisfacción de usuario, intervalos de edad, grado de acuerdo con una afirmación). Una práctica usual, es tratar este tipo de variables como si fueran continuas, asignándoles una puntuación arbitraria basada en la codificación de las distintas categorías de respuesta, esta práctica, si bien pudiera considerarse correcta en el caso de variables ordinales, no lo es si las variables son simplemente nominales.

Existen técnicas estadísticas que permiten modelar una variable dependiente discreta con respecto a una o varias variables explicativas. Algunas de estas técnicas son no paramétricas como el algoritmo de los k-vecinos más cercanos (Cover and Hart, 1967), mientras que otras son paramétricas como la regresión logística, regresión multinomial (Agresti, 2002), el análisis discriminante lineal (Fisher, 1936) o más recientemente las máquinas de vectores soporte (Cortes and Vapnik, 1995).

En este capítulo describiremos teóricamente los modelos de regresión logística simple y múltiple, su formulación, interpretación de parámetros e inclusión de variables explicativas categóricas.

2.1. Modelos de respuesta binaria

En los modelos de respuesta binaria o dicotómica, se tiene que la variable de respuesta Y puede tomar dos valores, codificándolos usualmente como 1 para la categoría de interés y 0 para la otra.

La distribución de Y es una Bernoulli cuya esperanza es:

$$E\left[Y\right] = P\left[Y = 1\right] = p \quad (0 < p < 1)$$

Si tenemos una variable X, posible predictora de la variable Y, entonces la distribución condicional de Y sobre un valor de $X = x$, también sigue una distribución de Bernoulli de forma que la esperanza condicionada de Y sobre $X = x$ es :

$$E\left[Y|X = x\right] = P\left[Y = 1 \,|\, X = x\right] = p(x)$$

y la varianza condicionada.

$$Var[Y \,|\, X = x] = p(x) \cdot (1 - p(x))$$

Un modelo para la variable Y en función de X sería de la forma

$$Y = f\,(\text{parámetros}, x, \text{error})$$

Una primera aproximación al problema sería aplicar un modelo de regresión lineal clásico para estimar Y en función de X . Si X es continua el modelo sería:

$$Y = \alpha + \beta x + \epsilon(x)$$

dónde los errores son variables aleatorias independientes con esperanza 0, y cuya distribución es una Bernoulli. El modelo de regresión lineal sería:

$$E\left[Y \mid X = x\right] = p(x) = \alpha + \beta x$$

Es decir, un modelo lineal para estimar la probabilidad condicionada. Este modelo adolece de varios problemas, tales como:

- Falta de normalidad de la variable Y y por tanto de los errores. Tanto Y como $\epsilon(x)$ se distribuyen según una Bernoulli
- Heterocedasticidad (la varianza de la variable respuesta no es constante sobre los valores de x), sino que depende de la esperanza condicionada $E\left[Y \mid X = x\right] = p(x)$, y se tiene $Var[Y \mid X = x] = p(x) \cdot (1 - p(x))$
- No acota los valores de $p(x)$. La probabilidad está acotada entre 0 y 1, pero este modelo puede predecir probabilidades fuera de ese intervalo.
- El modelo asume una relación lineal entre X y $p(x)$ lo que llevaría a que variaciones iguales en X producen variaciones iguales en $p(x)$. Claramente, si para un valor de X, $p(x)$ está cercano a 1, una variación grande de X implica una menor variación en $p(x)$ que la misma variación en X cuando $p(x)$ está próxima a 0.5. Es decir, la probabilidad en los extremos varía más lentamente.

En la figura (2.1), se muestra el ajuste del modelo de probabilidad lineal. Este modelo estima probabilidades por encima y por debajo del intervalo $(0, 1)$

Debido a los problemas del modelo de probabilidad lineal, se han buscado modelos alternativos de la forma

$$Y = F\left(\alpha + \beta x\right) + \epsilon(x)$$

con $\epsilon(x)$ vv.aa independientes con esperanza 0, con lo que el modelo sobre la probabilidad condicionada se puede escribir como

$$p(x) = F(\alpha + \beta x)$$

con F función monótona creciente .También se puede expresar como sigue

$$F^{-1}\left(p(x)\right) = \alpha + \beta x$$

es decir, se busca una función F cuya inversa transforme las probabilidades condicionadas $p(x)$ y posteriormente, modelar linealmente esta transformación.

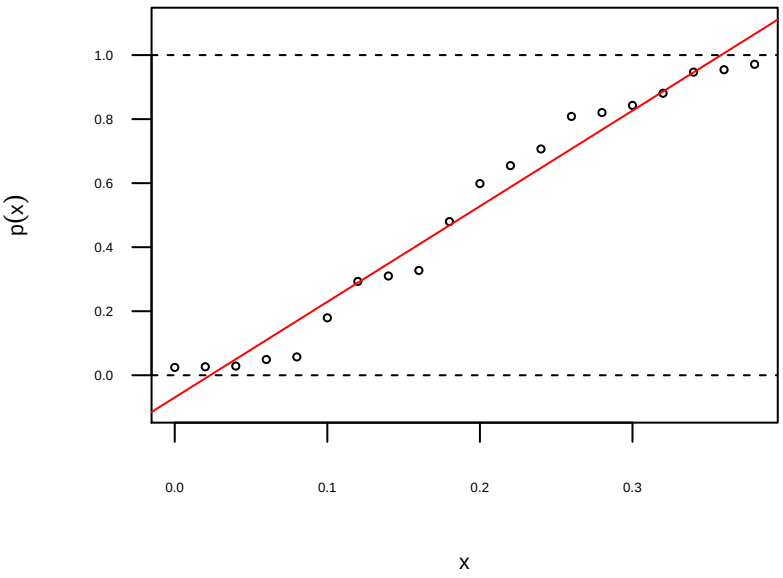


Figura 2.1: Modelo de probabilidad lineal

Según se elija una determinada función F se tienen distintas formulaciones.

- Transformación *logit*. La transformación logit es

$$logit(p(x)) = \ln \frac{p(x)}{1 - p(x)}$$

  Con lo que un modelo para la transformación logit sobre $p(x)$ sería

$$logit(p(x)) = \ln \frac{p(x)}{1 - p(x)} = \alpha + \beta x$$

  o en términos de $p(x)$

$$p(x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)} = \frac{1}{1 + \exp[-(\alpha + \beta x)]}$$

  Una de las ventajas de este modelo frente al modelo de probabilidad lineal es que la expresión para $p(x)$ está acotada entre 0 y 1, tal y como es deseable tratándose de una probabilidad. Otra ventaja es la sencillez de la interpretación, puesto que $\frac{p(x)}{1 - p(x)}$ se corresponde con la ventaja de la respuesta $Y = 1$ para el valor x.

- Transformación *probit* . La transformación *probit* consiste en considerar como función de transformación la inversa de la función de distribución de una normal estándar $\mathcal{N}(0, 1)$.

$$\mathcal{F}(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}t^2} \, \mathrm{d}t$$

y la expresión del modelo sería

$$\mathcal{F}^{-1}\left(p(x)\right) = \alpha + \beta x$$

Esta transformación tambien acota $p(x)$ entre 0 y 1
La función *probit* se acerca más rápidamente a probabilidades de 0 y 1 que la función *logit.*

- Transformación *cloglog* . Esta transformación es de la forma

$$p(x) = 1 - \exp\left[-\exp\left(\alpha + \beta x\right)\right]$$

que en forma lineal sería

$$\log\left[-\log\left(1 - p(x)\right)\right] = \alpha + \beta x$$

Esta transformación no tiene un comportamiento simétrico, sino que se aleja del valor de probabilidad 1 de forma más rápida de lo que se acerca al valor 0.

- Transformación *loglog.* Esta transformación es similar a la anterior

$$p(x) = \exp\left[-\exp\left(\alpha + \beta x\right)\right]$$

o

$$\log\left[-\log\left(p(x)\right)\right] = \alpha + \beta x$$

En este caso se obtiene el comportamiento inverso que la transformación *cloglog,* de forma que si la transformación *loglog* es adecuada para modelar la probalididad condicionada a $X = x$ de un suceso, la *cloglog* es adecuada para modelar el suceso complementario.

A las diversas transformaciones que se pueden utilizar se las conoce también como funciones *link* o vínculo, ya que relacionan una transformación sobre los valores de $p(x)$ con la recta $\alpha + \beta x$. Así, dependiendo de cómo sean los datos, será más útil un tipo de transformación u otra. En la figura (2.2) vemos las 4 transformaciones comentadas.

2.1.1. Relación con los modelos lineales generalizados

El tipo de modelos que se obtienen mediante las transformaciones descritas, pueden considerarse un caso particular de los modelos lineales generalizados (GLM) (McCullagh and Nelder, 1989), los cuales engloban a una gran cantidad de modelos, incluyendo a la mayoría de modelos de regresión usuales.

Hasta ahora hemos considerado que sólo tenemos una variable explicativa. Para ilustrar los modelos lineales generales vamos a considerar k variables explicativas. Estamos interesados en modelar la esperanza condicionada de Y en las diferentes observaciones de las variables X_K, es decir, $E\left[Y \mid X_1 = x_1, \ldots, X_k = x_k\right]$. Un modelo lineal generalizado para la esperanza condicionada es de la forma

$$g\left[\mu(x)\right] = \beta_o + \beta_1 x_1 + \cdots + \beta_k x_k$$

dónde $\mu(x) = E\left[Y \mid X_1 = x_1, \ldots, X_k = x_k\right]$ y g es la función de vínculo

Los GLM's tienen tres componentes.

1. Un componente aleatorio. Nos referimos a la distribución de la variable de respuesta y (y por ende la distribución de los errores), dados los predictores. Se considera la familia exponencial, de la que forman parte tanto la distribución de Poisson como la binomial, y por lo tanto los GLM's

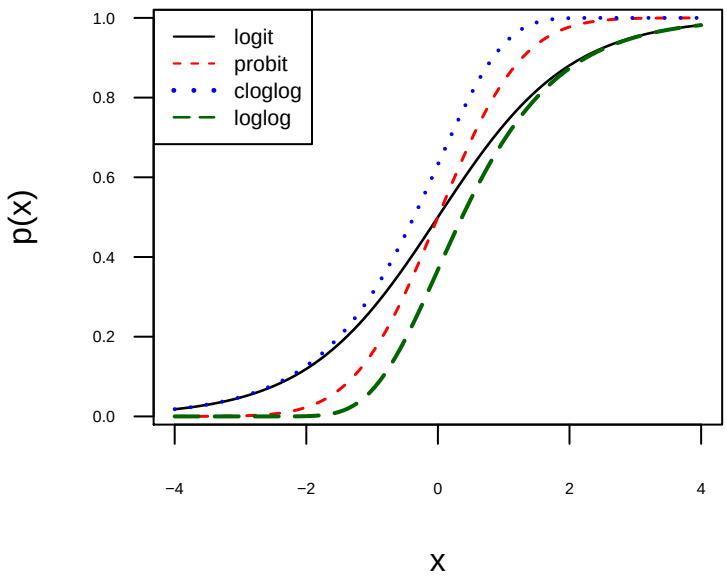


Figura 2.2: Funciones de transformación de $p(x)$. $\alpha = 0$, $\beta = 1$ para todas las transformaciones excepto para la transformación *loglog* dónde $\beta = -1$

son aptos para tratar con variables con dichas distribuciones, típicamente variables categóricas. En los modelos lineales clásicos no se especifica la distribución condicionada de la variable de respuesta y, aunque se asume que sigue una distribución normal, por lo que, los modelos lineales son un caso particular de los GLM's.

2. El predictor lineal. No es más que la función lineal de las variables explicativas, suponiendo que $X_1 = x_1, \ldots, X_k = x_k$ entonces el predictor lineal asociado a esa combinación de valores de las variables explicativas es:

$$\eta(x) = \beta_o + \beta_1 x_1 + \cdots + \beta_k x_k$$

3. La función de vínculo o link. Función que especifica la relación entre la esperanza condicionada $E[Y \mid X_1 = x_1, \ldots, X_k = x_k]$ y el predictor lineal.
En los *modelos lineales,* esta relación es directa siendo

$$E[Y \mid X_1 = x_1, \ldots, X_k = x_k] = \eta(x) = \beta_o + \beta_1 x_1 + \cdots + \beta_k x_k$$

En esta relación, la media puede tomar valores entre $(-\infty, +\infty)$ lo que no es válido para todos los GLM's, como por ejemplo cuando queremos estimar la media de una variable binaria (0,1). Debido a esto, se considera la función de vínculo g que relaciona la esperanza condicionada que queremos modelar con el predictor lineal.

$$g[\mu(x)] = \eta(x)$$

Con g una función estrictamente creciente.

En los cuadros (2.1) y (2.2) extraídos de (Fox and Weisberg, 2011) se muestran las funciones de vínculo

más utilizadas y con qué tipo de familias de distribución se utilizan.

Cuadro 2.1: *Funciones links más usadas y sus inversas. μ es el valor esperado de la respuesta y condicionado a los diferentes valores observados en las variables $X_1 \ldots, X_k$ y η es el predictor lineal.*

Función de vínculo	$\eta = g(\mu)$	$\mu = g^{-1}(\eta)$	Inversa de la función de vínculo
Identidad	μ	η	Identidad
Logarítmica	$\log_e \mu$	e^{η}	Exponencial
Inversa	μ^{-1}	η^{-1}	Inversa
Raíz cuadrada	$\sqrt{\mu}$	η^2	Cuadrado
Logit	$log_e \frac{\mu}{1-\mu}$	$\frac{1}{1+e^{-\eta}}$	Logística
probit	$\Phi(\mu)$	$\Phi^{-1}(\eta)$	Cuantiles de la normal
log-log complementario	$\log_e [-\log_e (1-\mu)]$	$1 - \exp [-\exp(\eta)]$	

Según el tipo de distribución, se tienen funciones link por defecto para cada tipo de familia, aunque se pueden usar otras funciones de vínculo.

Cuadro 2.2: *Función link por defecto, rango de la respuesta, función de la varianza condicionada para varias familias de modelos lineales generalizados. ϕ es el parámetro de dispersión o escala. Si no se muestra vale 1. $\mu = \mu(x)$ es la media condicionada de y dados los valores de las variables predictoras. En la familia binomial, N es el número de ensayos*

Familia	Función link por defecto	Rango de y	$Var(Y \mid X_1 = x_1, \ldots, X_k = x_k)$	Otras funciones de vínculo posibles
Gaussiana	Identidad	$(-\infty, +\infty)$	ϕ	Logarítmica
Binomial	logit	$\frac{0,1,\ldots,N}{N}$	$\frac{\mu(1-\mu)}{N}$	Logarítmica, probit, cloglog, loglog
Poisson	log	$0, 1, 2, \ldots$	μ	Identidad, raíz cuadrada
Gamma	Inversa	$(0, \infty)$	$\phi\mu^2$	Identidad, logarítmica

Los modelos con transformación logit y probit vistos anteriormente, modelan la esperanza condicionada de Y a un valor x de la variable explicativa X y no son más que un modelo lineal generalizado con una sola variable explicativa, dónde la distribución de la variable Y es binomial y la función de enlace es la función logit o probit, respectivamente.

En los GLM's, la varianza $\text{Var}(Y \mid X_1 = x_1, \ldots, X_k = x_k)$ viene dada por un parámetro de escala positivo, ϕ y por una función de la media condicionada de y dados los valores de las variables predictoras.

$$Var(Y \mid X_1 = x_1, \ldots, X_k = x_k) = \phi \times f\,[\mu(x)]$$

Para las distribuciones binomial o poisson, $\phi = 1$, y la varianza depende sólo de μ . Para la distribución normal (gaussiana), la varianza depende sólo del parámetro de dispersión ϕ, que como sabemos, en ese caso es σ^2.

2.2. Modelo de regresión logística simple

Se habla de regresión logística simple cuando se tiene una variable de respuesta binaria y una variable explicativa.

Formulación

Sea Y una variable de respuesta binaria, dónde se ha codificado como 1 a la categoría de interés y 0 para la otra, y X un variable explicativa continua, entonces la $E[Y \mid X = x] = P[Y = 1|X = x] = p(x)$ se puede modelizar mediante un modelo de regresión logística simple como sigue:

$$p(x) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)} = \frac{1}{1 + \exp[-(\beta_0 + \beta_1 x)]}$$

Equivalentemente, en función de la transformación logit:

$$logit[p(x)] = \ln\left[\frac{p(x)}{1 - p(x)}\right] = \beta_0 + \beta_1 x \tag{2.1}$$

Se obtiene un modelo lineal para el logaritmo de la ventaja de respuesta $Y = 1$. Así definido, el modelo entra en la clase de los modelos lineales generalizados, dónde la función *logit* hace el papel de la función *link*.

Interpretación de parámetros

- El signo de β_1 indica el sentido del cambio en la probabilidad a los cambios en X
- Si $\beta_1 = 0$ entonces $p(x)$ no depende de x y se interpreta como que la variable Y es independiente de X. En este caso la estimación por el modelo coincide con la proporción de unos que hay en la variable Y.
- β_0 es el valor del logaritmo de la ventaja de respuesta $Y = 1$ frente a $Y = 0$ cuando $\beta_1 = 0$ (independencia entre Y y X) o también es el valor del logaritmo de la ventaja para un caso cuyo valor en X sea 0.
- En términos de ventajas se tiene que

  $$\frac{p(x)}{1 - p(x)} = \exp[\beta_0 + \beta_1 x] = e^{\beta_0} \cdot e^{\beta_1 x}$$

  que significa que la ventaja de la respuesta 1 aumenta en e^{β_1} veces por cada unidad de aumento de X, como se comprueba al calcular el cociente de ventajas entre x y $x + 1$

$$\theta(x + 1, x) = \frac{\dfrac{p(x+1)}{1 - p(x+1)}}{\dfrac{p(x)}{1 - p(x)}} = \frac{\exp(\beta_0 + \beta_1(x + 1))}{\exp(\beta_0 + \beta_1 x)} = \frac{e^{\beta_0} \cdot e^{\beta_1 x} \cdot e^{\beta_1}}{e^{\beta_0} \cdot e^{\beta_1 x}} = e^{\beta_1}$$

- El cociente de ventajas para dos valores distintos de X es

$$\theta(x_1,x_2) = \frac{\dfrac{p(x_1)}{1-p(x_1)}}{\dfrac{p(x_2)}{1-p(x_2)}} = \frac{\exp(\beta_0+\beta_1 x_1)}{\exp(\beta_0+\beta_1 x_2)} = e^{\beta_1(x_1-x_2)}$$

2.3. Modelo de regresión logística múltiple

Formulación

Considerando ahora R variables cuantitativas $X_1 \ldots, X_R$, entonces para cada combinación de dichas variables, se tiene que la variable de respuesta Y sigue una distribución de Bernoulli

$$Y \mid (X_1 = x_1, \ldots X_R = x_R) \rightsquigarrow B(1, p(x_1, \ldots x_R))$$

al igual que en el caso del modelo simple, nos interesa modelar la esperanza condicionada

$$E[Y \mid X_1 = x_1, \ldots X_R = x_R] = P[Y = 1 \mid X_1 = x_1, \ldots X_R = x_R] = p(x_1, \ldots x_R)$$

El modelo de regresión logística múltiple para Y en términos de los valores de las variables X, se puede modelizar como:

$$p(x_1, \ldots x_R) = \frac{\exp\left(\alpha + \sum_{r=1}^{R} \beta_r x_r\right)}{1 + \exp\left(\alpha + \sum_{r=1}^{R} \beta_r x_r\right)}$$

si notamos $\alpha = \beta_0$ y $x_0 = 1$ la expresión quedaría cómo

$$p(x_1, \ldots x_R) = \frac{\exp\left(\sum_{r=0}^{R} \beta_r x_r\right)}{1 + \exp\left(\sum_{r=0}^{R} \beta_r x_r\right)} \tag{2.2}$$

que en términos matriciales sería

$$p(\boldsymbol{x}) = \frac{\exp \beta^t \boldsymbol{x}}{1 + \exp \beta^t \boldsymbol{x}} \tag{2.3}$$

con $\boldsymbol{x}$ el vector $1, x_1 \ldots x_R$ y $\boldsymbol{\beta} = \beta_0, \ldots \beta_R$

Al igual que en el caso de una sola variable explicativa, podemos considerar un modelo lineal para la transformación logit de $p(x)$ como sigue

$$\ln\left[\frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})}\right] = \sum_{r=0}^{R} \beta_r x_r.$$

con lo que tenemos un modelo lineal generalizado cuya función *link* es la transformación *logit*. En la figura (2.3) vemos la curva logística con dos variables explicativas en el intervalo $(-10, 10)$ y con todos los $\beta_r = 1$

Interpretación

- Si todos los β_r son iguales a 0 salvo β_o entonces $p(x) = \dfrac{e^{\beta_0}}{1+e^{\beta_0}}$, con lo que en este caso la variable Y es independiente de las explicativas.

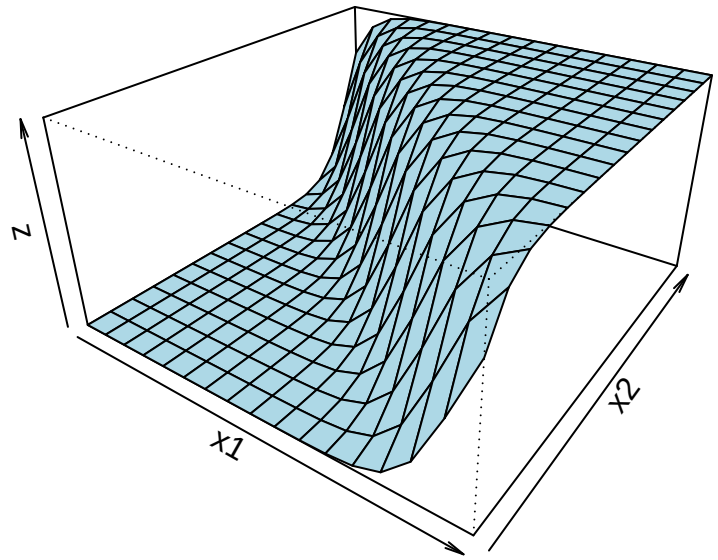


Figura 2.3: Función logit con dos variables explicativas continuas y con parámetros $\beta_0 = 1, \beta_1 = 1, \beta_2 = 1$. x_1y x_2 son las variables explicativas y z es la probabilidad estimada

- β_0 es el valor del logaritmo de la ventaja de respuesta $Y = 1$ frente a $Y = 0$ cuando $\beta_r = 0$ $\forall r = 1 \dots R$ o también es el valor del logaritmo de la ventaja para un caso donde $X_1 = X_2 = \dots = X_R = 0$

- El cociente de ventajas entre dos configuraciones de los valores de las variables explicativas, $x_1 = (1, x_{11}, \dots, x_{1R})$ y $x_2 = (1, x_{21}, \dots, x_{2R})$ sería.

$$\theta(x_1, x_2) = \frac{\dfrac{p(x_1)}{1 - p(x_1)}}{\dfrac{p(x_2)}{1 - p(x_2)}} = \frac{\exp\left(\sum_{r=0}^{R} \beta_r x_{1r}\right)}{\exp\left(\sum_{r=0}^{R} \beta_r x_{2r}\right)} = \exp\left(\sum_{r=1}^{R} \beta_r \left(x_{1r} - x_{2r}\right)\right)$$

  Si la diferencia entre x_1 y x_2 en cada valor de $X_1 \dots, X_R$ es de 1. Entonces

$$\theta(x_1, x_2) = \exp\left(\sum_{r=1}^{R} \beta_r\right) = \prod_{r=1}^{R} e^{\beta_r}$$

  Si la diferencia entre x_1 y x_2 es de 1, pero sólo en una de las variables explicativas, digamos en X_l mientras que sus valores son los mismos en el resto de variables, entonces

$$\theta(x_1, x_2) = e^{\beta_l}$$

Es decir, el exponencial del parámetro asociado a la variable X_l es la cantidad por la que queda multiplicada la ventaja de respuesta $Y = 1$ cuando el valor en X_l aumenta en una unidad, sin que cambien los valores en el resto de variables explicativas.

2.4. Variables explicativas nominales y ordinales

Cuando la variable explicativa es categórica, el modelo se construye considerando variables numéricas asociadas a la categórica, son las llamadas variables de diseño o auxiliares.

Cuando se tienen variables categóricas con más de dos categorías, digamos con I categorías, se construyen $I - 1$ variables de diseño. Existen diferentes formas de codificar esas variables de diseño, destacando los métodos parcial y marginal o, para variables ordinales, utilizar una codificación que considere distancias equidistantes entre las categorías de respuesta.

Codificación parcial

En la codificación parcial se elige una categoría de referencia, de modo que todas las variables de diseño toman el valor 0 para dicha categoría. Para cada una de las categorías restantes, su variable de diseño toma el valor 1 para la categoría asociada y 0 para el resto. Esta forma de codificación suele venir implementada en los diversos programas estadísticos, aunque según el que se use, se toma como referencia la primera categoría o la última. Suponiendo que se tienen I categorías en una variable explicativa categórica A y que se ha utilizado el método de codificación parcial asignando el valor 0 para la categoría l, el valor para la variables de diseño m-ésima asociada a la categoría A_m sería

$$X_{im}^A = X_m^A \mid (A = A_i) = \begin{cases} 1 & i = m \\ 0 & i \neq m \end{cases} \quad \forall m = 2 \ldots, I; i = 1 \ldots I$$

Codificación marginal

En este método, las variables de diseño toman el valor 1 para su categoría asociada y el valor 0 para las restantes, excepto para la categoría de referencia que toma el valor -1. La codificación sería, suponiendo la primera categoría como la de referencia.

$$X_{im}^A = X_m^A \mid (A = A_i) = \begin{cases} 1 & i = m \\ -1 & i = 1 \\ 0 & i \neq m, 1 \end{cases} \quad \forall m = 2 \ldots, I \quad i = 1 \ldots I$$

En la regresión logística se utiliza mayoritariamente el método de codificación parcial, debido a que facilita la interpretación en términos de cocientes de ventajas. Otro motivo por el que usar este tipo de codificación, se debe al uso de la regresión logística en epidemiología y en diseño de experimentos, dónde es usual tener un grupo de control no expuesto al tratamiento y con el cuál se quieren comparar los otros grupos.

Una vez que se han codificado las variables categóricas, el modelo se reduce al caso de regresión logística simple si lo que se tiene es una sóla variable explicativa que tenga sólo dos categorías, o al modelo de regresión logística si se tienen más variables explicativas o que se tenga sólo una pero con tres o más categorías.

Codificación de variables ordinales

Cuando se tienen variables explicativas ordinales, se pueden tratar como si fueran nominales y codificarlas por alguno de los métodos anteriores.

Otra forma de codificarlas es asignar puntuaciones monótonas a cada categoría, de forma que conserven el orden. Normalmente se consideran puntuaciones equidistantes entre categorías. Si se codifican de esta forma, las variables se incluyen en el modelo como variables cuantitativas cuyos valores serán los códigos asignados.

En el capítulo 4 de (Fox and Weisberg, 2011) sección 4.6, se realiza un análisis exhaustivo de otras formas de codificación, con un apartado específico sobre las diferentes formas de codificar variables ordinales, incluyendo el uso de polinomios ortogonales, utilizado sobre todo en análisis de la varianza.

Capítulo 3

Ajuste del modelo

En este capítulo se ilustra el ajuste de un modelo de regresión logística utilizando R . Para el ajuste se ha utilizado una encuesta del INE referente a la utilización del uso de tecnologías de la información y la comunicación en los hogares españoles en 2011. Se describe la función `glm` de R y cómo se utiliza para ajustar el modelo de regresión logística. Posteriormente se comenta cómo se realiza el ajuste si los datos están agrupados o no, y qué diferencias hay entre ambas formas de ajustar el modelo. En el apartado (3.3) se ajusta el modelo cuando se tienen variables explicativas nominales u ordinales. Por último se realiza el ajuste cuando se tienen variables explicativas nominales y cuantitativas, y el ajuste de modelos con interacción.

3.1. Archivo de datos

El fichero que se va a usar corresponde a la Encuesta sobre Equipamiento y Uso de Tecnologías de la Información y Comunicación en los hogares realizada en 2011 (TIC-H 2011) (http://www.ine.es/prodyser/micro_tich.htm), y más concretamente a la muestra para Andalucía de 3.485 personas. La variable dependiente será el uso de internet (*¿Ha usado internet alguna vez?*) de la persona seleccionada en el hogar. Como posibles variables explicativas consideraremos la *edad,* el *sexo,* el *nivel de estudios alcanzado* y el *hábitat.*

Los datos están en un fichero *sav* de SPSS[1], que se puede leer en R utilizando el paquete *foreign* (R Core Team, 2013a).

```
library(foreign)
# mostrar directorio actual
getwd()

## [1] "/home/jose/master_estadística/Trabajo_Fin_Master/Documentos_lyx"

# el fichero de datos está en el directorio Datos del nivel superior
# lectura del fichero con read.spss indicando el path, los '..' indican el
# directorio superior al actual
datos <- read.spss(file = "../Datos/t11And_18_mas.sav", use.value.labels = TRUE,
    to.data.frame = TRUE)
```

[1] El fichero original está en formato ascii, pero se convirtió a formato de spss. Para leerlo directamente se puede utiilzar la función `read.fwf`

Como sólo nos interesan algunas variables del total, vamos a construir un nuevo *data.frame* que contenga sólo a las variables de interés. También recodificamos la variable *uso_int* (uso de internet), para que valga 1[2] si la persona ha utilizado alguna vez internet y 0 en otro caso, utilizando la función `ifelse`[3].

```
uso_int <- datos$USO_INT
uso_int <- ifelse(uso_int == 1, 1, 0)
sexo <- datos$SEXO
edad <- datos$EDAD
nivelest <- datos$NIVELEST
habitat <- datos$HABITAT
datos.bin <- data.frame(uso_int, sexo, edad, nivelest, habitat)
# borramos todos los objetos creados salvo el data.frame datos.bin, para
# evitar confusiones
rm(uso_int, sexo, edad, nivelest, habitat)
```

Veamos los 6 primeros valores de datos.bin, utilizando la función `head`

```
head(datos.bin)

##   uso_int   sexo edad nivelest habitat
## 1       0 Hombre   52        2       1
## 2       0  Mujer   76        2       1
## 3       0  Mujer   73        2       1
## 4       0  Mujer   66        2       1
## 5       0 Hombre   53        3       1
## 6       1  Mujer   63        4       1
```

En algunas variables ha tomado el valor del código en vez de la etiqueta. En R es importante saber qué tipo de objeto es cada variable, puesto que la mayoría de funciones lo tienen en cuenta. Para ver el tipo de un objeto, se puede utilizar la función `class`[4]. Utilizamos la función `sapply` que toma como argumento el data.frame, y a cada elemento del data.frame (variables) le aplica la función que especifiquemos.

```
sapply(datos.bin, class)

##   uso_int      sexo      edad  nivelest   habitat
## "numeric"  "factor" "numeric" "numeric" "numeric"
```

El nivel de estudios y el habitat deberían ser variables categóricas, así que vamos a convertirlas a la clase factor.

[2] La codificación original de las variables se pueden ver en `ftp://www.ine.es/temas/tich/tich_disreg_11.xls`

[3] La función `ifelse` es una versión vectorial de la función `if.` Aplicada a un vector evalúa la condición sobre todos los elementos devolviendo un vector de la misma longitud que el original.

[4] Algunos tipos de objetos son *numeric, factor* (variables categóricas) o *character,* para texto

```
datos.bin$habitat <- factor(datos.bin$habitat, levels = 0:6, labels = paste("estrato",
    0:6, sep = ""))
datos.bin$nivelest <- factor(datos.bin$nivelest)
```

Hemos especificado que la convierta a factor y que tome las categorías de 0 a 6, con etiquetas iguales a estrato0, estrato1, etc.

Con la función `levels` vemos cuales son los niveles de los factores.

```
levels(datos.bin$habitat)

## [1] "estrato0" "estrato1" "estrato2" "estrato3" "estrato4" "estrato5"
## [7] "estrato6"

levels(datos.bin$nivelest)

## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

Cuando se pasan variables numéricas a factores, R toma como primer nivel, el código más pequeño. Si lo que se convierte a factores son variables que contienen caracteres, los ordena alfabéticamente.

Los códigos de la variable *hábitat* se corresponden con.

Estrato	Descripción
0	Capitales provincia de más de 500 mil habitantes
1	Resto de capitales de provincia
2	Municipios(no capitales) con más de 100 mil habitantes
3	Municipios(no capitales) con más de 50 mil y menos de 100 mil habitantes
4	Municipios con más de 20 mil y menos de 50 mil habitantes
5	Municipios con más de 10 mil y menos de 20 mil habitantes
6	Municipios con menos de 10 mil habitantes

Las frecuencias en cada estrato se pueden obtener utilizando `table`, y con `barplot` obtenemos su representación gráfica.

```
barplot(table(datos.bin$habitat), cex.names = 0.7, cex.axis = 0.7)
```

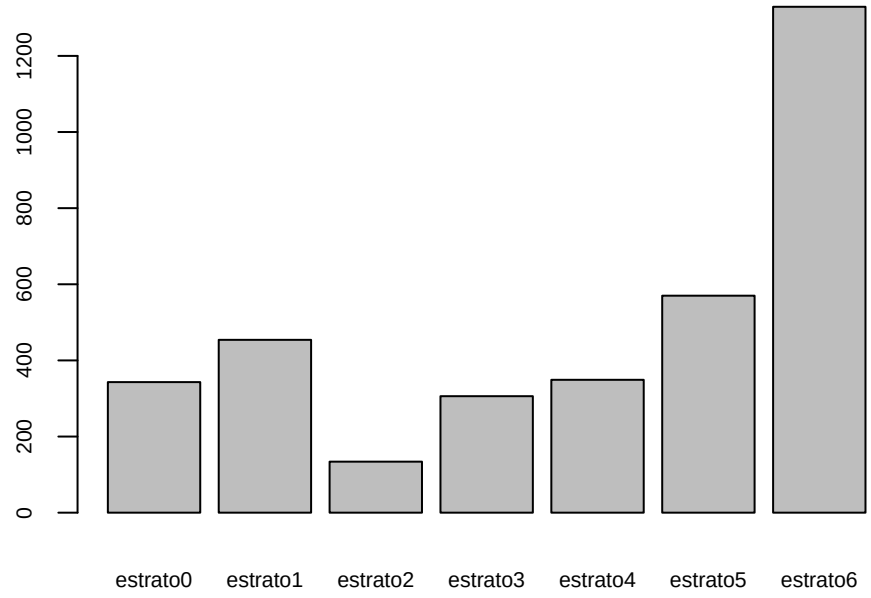


Figura 3.1: Frecuencias de encuestados en cada estrato de hábitat.

Se observa que el estrato con más casos es el que corresponde a los municipios de menos de 10.000 habitantes.

Para el nivel de estudios los códigos se corresponden con las siguientes categorías.

Código	Descripción
1	Analfabetos
2	Educación primaria
3	Primera etapa de la educación secundaria
4	Segunda etapa de la educación secundaria
5	Enseñanza postsecundaria no superior
6	Formación profesional de grado superior
7	Educación superior universitaria (excepto Doctores)
8	Título de Doctorado
9	No se puede codificar

Recodificamos las categorías del nivel de estudios utilizando por ejemplo la función `recode` del paquete `car`.

```
library(car)
datos.bin$nivelest <- with( datos.bin, recode( nivelest,
" c(1,9) = 'Analfabetos';
```

```
2:3 = 'Primaria';
4:6 = 'Secundaria y F.P';
7:8= 'Universitaria o superior'")
)
levels(datos.bin$nivelest)
```

```
## [1] "Analfabetos"              "Primaria"
## [3] "Secundaria y F.P"         "Universitaria o superior"
```

La función `with` permite no tener que utilizar la indexación o el símbolo $ para utilizar una variable que esté dentro de un data.frame. Se utilizará indistintamente el uso de `with` con el de los corchetes o el símbolo $.

Utilizando de nuevo `barplot`, obtenemos el gráfico de las frecuencias.

```
barplot(table(datos.bin$nivelest), cex.names = 0.6, cex.axis = 0.7)
```

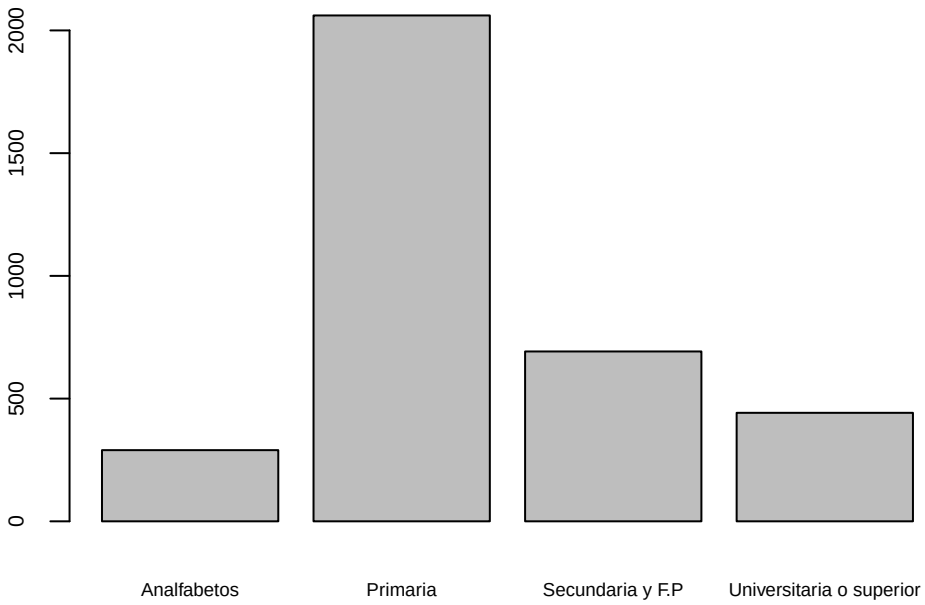


Figura 3.2: Frecuencias de encuestados en cada nivel de estudios.

Vemos un resumen de los datos con `summary`.

```
summary(datos.bin)

##     uso_int          sexo           edad
##  Min.   :0.000   Hombre:1517   Min.   : 18
```

```
##  1st Qu.:0.000   Mujer :1968   1st Qu.: 38
##  Median :0.000                 Median : 52
##  Mean   :0.468                 Mean   : 53
##  3rd Qu.:1.000                 3rd Qu.: 69
##  Max.   :1.000                 Max.   :101
##
##                      nivelest        habitat
##  Analfabetos             : 290   estrato0: 343
##  Primaria                :2061   estrato1: 454
##  Secundaria y F.P        : 692   estrato2: 134
##  Universitaria o superior: 442   estrato3: 306
##                                  estrato4: 349
##                                  estrato5: 570
##                                  estrato6:1329
```

La media de *uso_int* es de 0.468, es decir, un 46.8 % de los encuestados responden que han utilizado internet alguna vez.

La distribución de la variable uso_int según las distintas variables explicativas, la podemos obtener utilizando la función `prop.table`. La función `prop.table` aplicada a una tabla devuelve la proporción de cada celda, también se puede calcular la proporción de las filas o de las columnas especificando 1 o 2 en el argumento `margin`[5].

Distribución *uso_int* según sexo.

```
with(datos.bin, prop.table(table(sexo, uso_int), margin = 1))
```

```
##         uso_int
## sexo          0      1
##   Hombre 0.4891 0.5109
##   Mujer  0.5650 0.4350
```

Distribución *uso_int* según edad . Utilizamos la función `cut` para dividir la variable *edad* en 5 intervalos de igual amplitud.

```
with(datos.bin, prop.table(table(cut(edad, 5), uso_int), 1))
```

```
##              uso_int
##                     0       1
##   (17.9,34.6] 0.08396 0.91604
##   (34.6,51.2] 0.30900 0.69100
##   (51.2,67.8] 0.70644 0.29356
##   (67.8,84.4] 0.94514 0.05486
##   (84.4,101]  0.99187 0.00813
```

Distribución *uso_int* según *hábitat.*

[5] El nombre del argumento se puede obviar si se ha introducido en el orden correcto. R interpreta que si no se especifica el nombre del argumento, lo asigna en función de la posición en que aparece en la función. Utilizando args(funcion) se puede ver en qué orden aparecen los argumentos.

```
with(datos.bin, prop.table(table(habitat, uso_int), 1))
```

```
##           uso_int
## habitat         0      1
##   estrato0 0.4752 0.5248
##   estrato1 0.4692 0.5308
##   estrato2 0.3955 0.6045
##   estrato3 0.4837 0.5163
##   estrato4 0.5100 0.4900
##   estrato5 0.4947 0.5053
##   estrato6 0.6147 0.3853
```

Distribución *uso_int* según nivel de estudios.

```
with(datos.bin, prop.table(table(nivelest, uso_int), 1))
```

```
##                            uso_int
## nivelest                         0       1
##   Analfabetos              0.98966 0.01034
##   Primaria                 0.69529 0.30471
##   Secundaria y F.P         0.14451 0.85549
##   Universitaria o superior 0.07692 0.92308
```

Las tablas anteriores nos dan indicios de que las variables edad, nivel de estudios y hábitat están relacionadas con el uso de internet.

3.2. Estimación del modelo. Función glm

La función de verosimilitud y su logartimo son fundamentales para la estimación del modelo y la evaluación de la bondad del ajuste. Veamos cual es la función de verosimilitud en el caso de un modelo de regresión logística con una variable explicativa.

Sea y_i el número de éxitos en m_i ensayos de un proceso binomial dónde $i = 1, \ldots, n$ Entonces se tiene que

$$y_i|x_i \rightsquigarrow \text{Bin}(m_i, \theta(x_i))$$

con $\theta(x_i)$ la estimación del modelo logístico en cada x_i

$$\theta(x_i) = \frac{1}{1 + \exp\left(-\beta_o - \beta_1 x_i\right)}$$

y por lo tanto

$$P(Y = y_i|x_i) = \binom{m_i}{y_i} \theta(x_i)^{y_i} \left(1 - \theta(x_i)\right)^{m_i - y_i}$$

Asumiendo que las n observaciones son independientes, la función de verosimilitud viene dada por

$$L = \prod_{i=1}^{n} P\left(Y_i = y_i | x_i\right) = \prod_{i=1}^{n} \binom{m_i}{y_i} \theta(x_i)^{y_i} \left(1 - \theta(x_i)\right)^{m_i - y_i}$$

Y la log-verosimilitud

$$\begin{aligned} \log(L) &= \sum_{i=1}^{n} \left[\log \binom{m_i}{y_i} + \log\left(\theta(x_i)^{y_i}\right) + \log\left(\left(1 - \theta(x_i)\right)^{m_i - y_i}\right) \right] \\ &= \sum_{i=1}^{n} \left[y_i \log\left(\theta(x_i)\right) + (m_i - y_i) \log\left(1 - \theta(x_i)\right) + \log \binom{m_i}{y_i} \right] \\ &= \sum_{i=1}^{n} \left[y_i \log\left(\frac{\theta(x_i)}{1 - \theta(x_i)}\right) + m_i \log\left(1 - \theta(x_i)\right) + \log \binom{m_i}{y_i} \right] \end{aligned} \tag{3.1}$$

La estimación de los parámetros se obtienen maximizando la función de verosimilitud de los datos respecto de los parámetros del modelo. Para los modelos logit, la log-verosimilitud es una función cóncava y por tanto los estimadores máximo verosímiles existen y son únicos. El cálculo de los estimadores es más complejo que para los modelos lineales y requieren métodos de aproximación iterativa como el de Newton -Raphson, gradiente descendente o estimación por mínimos cuadrados iterativamente reponderados. En el apéndice A, se puede encontrar una descripción de las distintas formas de ajuste y cómo se pueden programar en R.

3.2.1. Función `glm`

Para ajustar un modelo lineal generalizado, la función genérica que se usa en R es `glm`. Cuyos argumentos son

```
args(glm)

## function (formula, family = gaussian, data, weights, subset,
##     na.action, start = NULL, etastart, mustart, offset, control = list(...),
##     model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL,
##     ...)
## NULL
```

Los argumentos más importantes de `glm` son `formula`, `family`, `data` y `subset`.

El argumento `formula` es ampliamente usado en la modelización con R y permite una sintaxis comprensible para expresar relaciones entre variables. La sintaxis de `formula` describe la relación entre la variable respuesta y las variables predictoras.

`formula` tiene tres partes: el lado izquierdo, el símbolo ~ y el lado derecho. En el lado izquierdo se especifica la variable respuesta, normalmente el nombre de la variable, aunque también se pueden poner expresiones matemáticas dentro de la función `glm` como por ejemplo `log(variable)`, o `sqrt(variable)`, sin necesidad de crear una nueva variable. El símbolo ~ se usa como separador. El lado derecho de una fórmula es una expresión especial que incluye los nombres de las variables predictoras. Por ejemplo, si utilizamos la función `lm` para ajustar regresión lineal y escribimos `lm(y~x1)`, se ajusta el modelo $y = \beta_0 + \beta_1 x_1 + \varepsilon$.

El argumento `family` sirve para indicar el componente aleatorio del modelo así como la función de enlace (link) que se utilizará. Si especificamos `family=binomial,` o simplemente `binomial,` la

función `glm` utilizará la función logit como función de enlace. La elección de otra función de enlace se especifica mediante el argumento `link`, por ejemplo, para ajustar un modelo probit se escribiría `family=binomial(link=probit)`

Los argumentos `data` y `subset` son para especificar respectivamente, el data frame dónde están los datos, y si se va a realizar el análisis sobre un subconjunto de los mismos. Si las variables del modelo no están en un data frame, el argumento `data` no es necesario.

En R, cuando la variable respuesta es binaria, ésta debe venir expresada bien en 0 y 1 (fracaso, éxito), o ser una variable lógica (con TRUE siendo el éxito y FALSE el fracaso) o como un factor, en cuyo caso la primera categoría[6] representa los fracasos y la otra los éxitos. También se puede considerar el caso general de una variable de respuesta binomial, dónde la variable de respuesta es el número de éxitos en uno o más ensayos. En este caso la sintaxis de `glm` varía levemente, como se verá cuando se trate el ajuste del modelo con datos agrupados.

Un ejemplo típico de la sintaxis de la función glm es.

```
glm(y ~ x, family = binomial, data = mis.datos)
```

Dónde y es una variable discreta con valores 0, 1 y x una variable continua (aunque se verá más adelante que también podría ser una categórica, en cuyo caso `glm` crea internamente las variables de diseño asociadas), que están en el data frame "mis.datos". Se ha especificado la familia binomial, la cual toma por defecto la función logit como función de enlace.

La función `glm` es la más usual para ajustar modelos lineales generalizados, si bien también existen alternativas en algunos paquetes desarrollados por la comunidad de R, tales como la función `lrm` en el paquete `rms` o `vglm` del paquete `VGAM`, o se pueden crear funciones propias implementando algún algoritmo iterativo de ajuste, como el de Newton-Raphson. En el apéndice A se verán algunas funciones sencillas para ajustar el modelo mediante otros procedimientos.

3.2.2. Datos sin agrupar

En el ajuste de un modelo de regresión logística, nos referimos a datos sin agrupar cuando tenemos los datos de forma que, para cada observación tenemos el valor de la variable repuesta. Los datos de la encuesta TIC son datos sin agrupar, dónde para cada individuo se tiene el valor 1 si ha utilizado internet alguna vez y 0 en caso contrario.

```
# 3 primeras filas
head(datos.bin)

##   uso_int   sexo edad         nivelest  habitat
## 1       0 Hombre   52         Primaria estrato1
## 2       0  Mujer   76         Primaria estrato1
## 3       0  Mujer   73         Primaria estrato1
## 4       0  Mujer   66         Primaria estrato1
## 5       0 Hombre   53         Primaria estrato1
## 6       1  Mujer   63 Secundaria y F.P estrato1
```

[6] La primera categoría es la que tenga el menor número de código, o la primera en orden alfabético si se ha creado el factor a partir de una variable de tipo carácter. La categoría de referencia se puede cambiar, bien recodificando la variable de respuesta o, utilizando funciones de R como `relevel`

En la figura (3.3) vemos el gráfico de *edad* frente a *uso_int.*

```
with(datos.bin, plot(jitter(edad), jitter(uso_int, 0.2), xlab = "Edad", ylab = "Uso de internet",
    cex = 0.4, cex.axis = 0.6, cex.lab = 0.6))
```

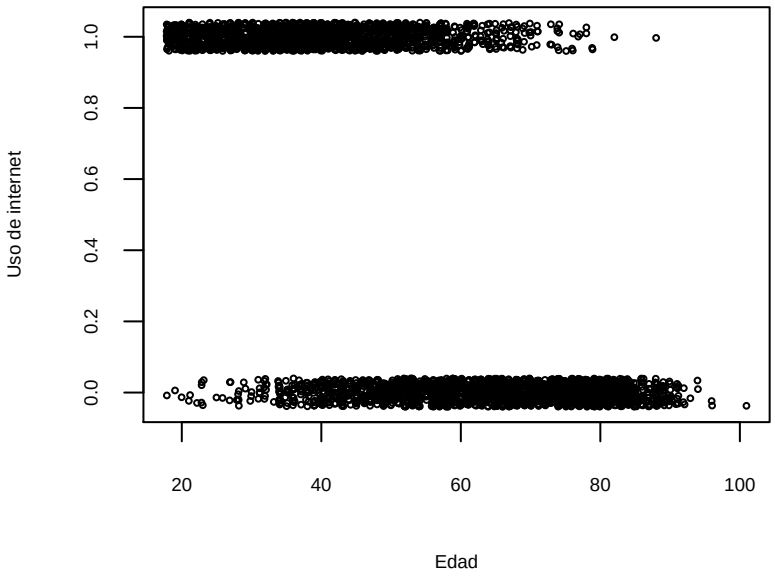


Figura 3.3: Uso de internet y Edad. Datos sin agrupar

Para evitar el solapamiento de los puntos en la figura (3.3), se ha utilizado la función `jitter` que añade un pequeño error aleatorio.

Al modelo de regresión logística con datos de este tipo se le denomina *regresión logística binaria*, ya que la variable respuesta toma sólo dos valores.

El ajuste de un modelo de regresión logística para la variable *uso_int,* tomando como variable explicativa la edad, utilizando la función `glm` es.

```
modelo.1 <- glm(uso_int ~ edad, data = datos.bin, family = binomial)
```

Al asignar a `modelo.1` el resultado de la función `glm`, hemos creado un objeto de tipo `glm`.

```
class(modelo.1)

## [1] "glm" "lm"
```

En este objeto se han guardado varias características y valores del ajuste. Para ver qué se ha guardado en `modelo.1` podemos utilizar varias funciones, como `str` (structure) o `names`.

```
names(modelo.1)
```

```
##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "call"              "formula"           "terms"
## [25] "data"              "offset"            "control"
## [28] "method"            "contrasts"         "xlevels"
```

En la ayuda de `glm` se explica con mayor detalle el uso de la función y qué resultados devuelve. Por ejemplo, en `fitted.values` se han guardado los valores predichos para $p(x)$. En `linear.predictors` se tienen los valores ajustados en la escala de la función link, es decir, los valores ajustados para $\log(\frac{p(x)}{1-(px)})$.

`residuals` no se corresponde con los residuos de pearson o los de la devianza, sino que son los últimos obtenidos en el algoritmo de reponderación por mínimos cuadrados utilizado en el ajuste. Para obtener los residuos correctamente se utilizará la función `residuals,` que aplicada a un objeto de tipo `glm,` permite obtener varios tipos de residuos. Otras funciones que extraen o calculan valores son `fitted` (valores ajustados), `coef` (coeficientes del modelo) o `predict`, que aplicada a un data.frame permite calcular los valores predichos por el modelo.

Para acceder a los valores guardados en `modelo.1`, se puede utilizar indistintamente la indexación mediante corchetes o el símbolo `$`.

```
# coeficientes del modelo
modelo.1[1]
```

```
## $coefficients
## (Intercept)        edad
##      5.6689     -0.1123
```

```
# devianza
modelo.1$deviance
```

```
## [1] 2917
```

```
# primeros 6 valores predichos para p(x)
head(modelo.1$fitted.values)
```

```
##       1       2       3       4       5       6
## 0.45687 0.05370 0.07364 0.14859 0.42916 0.19644
```

```
# primeros 6 valores de los residuos del método de ajuste
head(modelo.1$residuals)
```

```
##      1      2      3      4      5      6
## -1.841 -1.057 -1.079 -1.175 -1.752  5.091

# primeros 6 valores de los residuos de pearson
head(residuals(modelo.1, type = "pearson"))

##       1       2       3       4       5       6
## -0.9172 -0.2382 -0.2819 -0.4178 -0.8671  2.0225
```

Otra forma de acceder a la información del modelo es simplemente escribiendo el nombre del objeto

```
modelo.1

##
## Call:  glm(formula = uso_int ~ edad, family = binomial, data = datos.bin)
##
## Coefficients:
## (Intercept)         edad
##      5.6689      -0.1123
##
## Degrees of Freedom: 3484 Total (i.e. Null);  3483 Residual
## Null Deviance:     4817
## Residual Deviance: 2917  AIC: 2921
```

que muestra parte de la información más importante del modelo, tal como los coeficientes ajustados, la devianza del modelo ajustado y la del modelo nulo. R denomina modelo nulo al modelo sin variables explicativas, este modelo es el más simple que se puede considerar, y estima la misma respuesta para todas las observaciones, asignando como estimación común la proporción muestral de éxitos.

Un resumen del modelo se puede obtener utilizando la función genérica `summary`, la cual extrae diferente información según la clase del objeto a la que se le aplique.

```
summary(modelo.1)

##
## Call:
## glm(formula = uso_int ~ edad, family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.710  -0.597  -0.202   0.664   2.909
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.66889    0.18771    30.2   <2e-16 ***
## edad        -0.11234    0.00361   -31.2   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 2916.8  on 3483  degrees of freedom
## AIC: 2921
##
## Number of Fisher Scoring iterations: 5
```

`summary` ofrece un resumen del modelo de forma más ordenada, en primer lugar muestra la expresión utilizada para ajustar el modelo, junto con algunos valores descriptivos de los residuos de la devianza, como el mínimo, máximo, mediana y primer y tercer cuartil.

Los coeficientes del modelo los muestra en formato tabular, añadiendo el error estándar, y el valor z[7] que es el coeficiente dividido por el error. Este valor se utiliza en el test de Wald para contrastar si el coeficiente es significativo. En la tercera columna muestra el p-valor de ese contraste y a qué nivel de confianza es significativo. Por último se muestra la devianza del modelo nulo (null deviance) y del modelo ajustado (Residual deviance), con sus respectivos grados de libertad, así como el valor del AIC (Criterio de información de Akaike) que es una modificación de la devianza en la que se tiene en cuenta el número de parámetros ajustados.[8]

El modelo ajustado mediante `glm` tiene la siguiente expresión.

$$logit\left[p(x)\right] = \ln\left[\frac{p(x)}{1-p(x)}\right] = 5{,}67 - 0{,}11 \cdot x$$

o equivalentemente

$$p(x) = \frac{1}{1+\exp\left(-\left[5{,}67 - 0{,}11 \cdot x\right]\right)}$$

que nos permite calcular la probabilidad para cada valor de x.

[7] Se utiliza z porque la distribución asintótica del parámetro es normal, a diferencia de en los modelos lineales en los que la distribución es la t de Student.

[8] La devianza se utiliza en la evaluación del ajuste global del modelo, mientras que el valor de AIC es útil en la comparación de modelos y en la selección automática de variables (apartado 4.5)

```
with(datos.bin, plot(jitter(edad), jitter(uso_int, 0.2), xlab = "Edad", ylab = quote(p(x)),
    main = "Curva ajustada", cex = 0.4, cex.axis = 0.6, cex.lab = 0.6, cex.main = 0.7))
# Añadimos la función ajustada
curve(1/(1 + exp(-modelo.1$coefficients[1] - modelo.1$coefficients[2] * x)),
    add = TRUE)
```

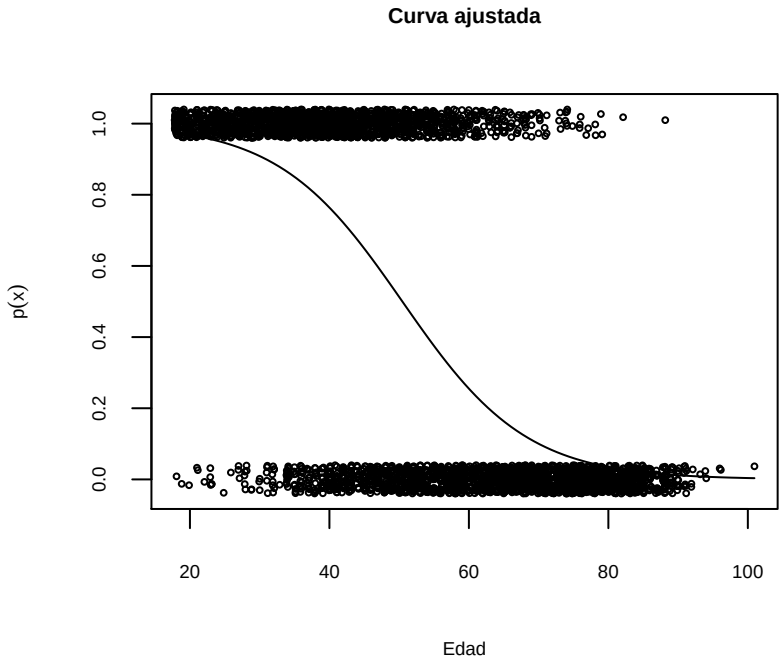


Figura 3.4: Curva logit ajustada, con datos no agrupados

3.2.3. Datos agrupados

Considerando el mismo ejemplo que en el apartado anterior, pero agrupando por edad se tiene la siguiente tabla.

	edad	no.internet	si.internet
1	18	1	31
2	19	1	28
3	20	1	35
4	21	2	26
5	22	1	30
6	23	5	24
7	24	0	31

Cuadro 3.1: Datos uso de internet agrupados por edad (Se muestran sólo los datos de las edades entre 18 y 24 años)

Estos datos los hemos guardado en un fichero separado por comas (csv), que podemos leer con R mediante la función `read.csv`

```
iagrupado <- read.csv("../Datos/internet_agrupAN.csv")
class(iagrupado)

## [1] "data.frame"

names(iagrupado)

## [1] "edad"        "no.internet" "si.internet"

nrow(iagrupado)  # número de filas del data.frame

## [1] 79

head(iagrupado)  # primeras 6 filas del data.frame

##   edad no.internet si.internet
## 1   18           1          31
## 2   19           1          28
## 3   20           1          35
## 4   21           2          26
## 5   22           1          30
## 6   23           5          24
```

La variable respuesta es el número de éxitos en un número fijo de N ensayos independientes. El ajuste de un modelo de regresión logística a este tipo de datos se conoce como *regresión logística binomial*. La *regresión logística binaria* es un caso particular dónde $N = 1$.

En el ajuste de este tipo de datos es necesario especificar tanto el número de éxitos como el número de casos en cada una de las combinaciones. En la función `glm` se especifica el número de éxitos y fracasos mediante el argumento `formula`, creando con la función `cbind` una matriz con dos columnas, la primera con el número de éxitos y la segunda con el número de fracasos. Una alternativa es poner en el argumento `formula` como variable respuesta la proporción de éxitos, y en el argumento `weights`, el número total de casos.

El código para el ajuste es.

```
modelo.2 <- glm(cbind(si.internet, no.internet) ~ edad, data = iagrupado, family = binomial)
```

Viendo el resultado con `summary`.

```
summary(modelo.2)

##
## Call:
## glm(formula = cbind(si.internet, no.internet) ~ edad, family = binomial,
##     data = iagrupado)
##
```

```
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.604  -0.633  -0.177   0.388   2.779
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.66889    0.18771    30.2   <2e-16 ***
## edad        -0.11234    0.00361   -31.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1974.599  on 78  degrees of freedom
## Residual deviance:   74.485  on 77  degrees of freedom
## AIC: 298
##
## Number of Fisher Scoring iterations: 4
```

Este modelo estima los **mismos parámetros que el `modelo.1`**, pero vemos que el valor de la **devianza es distinto**. Esto es debido a cómo se han considerado los datos, en el primer caso se ajustan 3.485 individuos, mientras en el segundo el ajuste es sobre 79 combinaciones de edad, nótese que los grados de libertad de la *Residual deviance,* son en el primer caso de 3.483 (3.485 individuos menos 2 parámetros ajustados) y en otro de 77 (79 valores de edad en la encuesta menos 2 parámetros ajustados).

La devianza, tal como se verá en el capítulo 4 provee una medida de la falta de ajuste de un modelo, pero sólo en el caso de que se trate con datos agrupados[9]. Una demostración de la diferencia del cálculo de la devianza, y de por qué no es una medida de la falta de ajuste en el caso de datos no agrupados, se puede encontrar en el libro "A Modern Approach to Regression with R" (Sheather, 2009) y que reproducimos en el apéndice B

Para representar gráficamente el ajuste, calculamos en primer lugar la proporción muestral del uso de internet en cada combinación de edad

```
iagrupado$prop <- with(iagrupado, si.internet/(si.internet + no.internet))
head(iagrupado)
```

```
##   edad no.internet si.internet    prop
## 1   18           1          31 0.96875
## 2   19           1          28 0.96552
## 3   20           1          35 0.97222
## 4   21           2          26 0.92857
## 5   22           1          30 0.96774
## 6   23           5          24 0.82759
```

y utilizando las funciones `plot` y `curve` representamos las proporciones muestrales y la curva logit ajustada por el modelo.

[9] El paquete estadístico SPSS en sus últimas versiones, considera los datos agrupados para el cálculo de la devianza.

```
plot(iagrupado$edad, iagrupado$prop, cex = 0.4, cex.axis = 0.6, cex.lab = 0.6,
    cex.main = 0.7, xlab = "Edad", ylab = quote(p(x)), main = "Curva ajustada para los datos agrupado
curve(1/(1 + exp(-modelo.2$coefficients[1] - modelo.2$coefficients[2] * x)),
    add = TRUE)
```

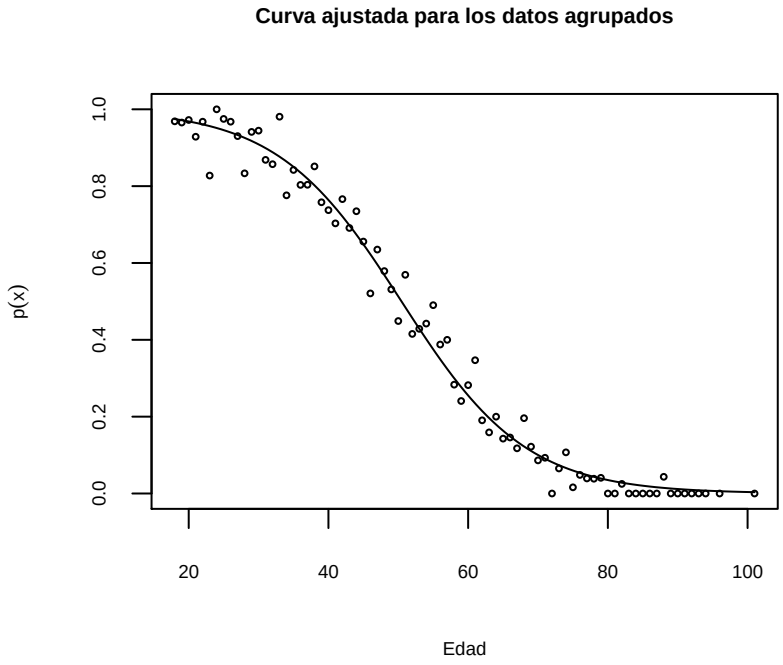


Figura 3.5: Curva logit ajustada para las diferentes combinaciones de edad

3.3. Variables explicativas nominales y ordinales.

Tal y como se comenta en la sección (2.4), cuando se tienen variables explicativas categóricas, el problema se reduce a codificarlas y tratar a las variables auxiliares obtenidas como numéricas. La codificación más utilizada es la codificación parcial y es la que R toma por defecto utilizando como categoría de referencia la primera. Cuando se tienen factores es conveniente utilizar la orden `levels(variable)` que nos devuelve las categorías ordenadas de la variable.

3.3.1. Una variable explicativa categórica

Retomando los datos del uso de internet entre los andaluces, teníamos como variables categóricas el sexo, el nivel de estudios y el hábitat. Para ver cómo las va a codificar R para los distintos procedimientos, utilizamos la función `contrasts`[10]. Para la variable hábitat, por ejemplo, se tiene.

```
contrasts(datos.bin$habitat)

##          estrato1 estrato2 estrato3 estrato4 estrato5 estrato6
## estrato0        0        0        0        0        0        0
```

[10] En la ayuda de la función, se explica como especificar otro tipo de codificaciones.

```
## estrato1        1        0        0        0        0        0
## estrato2        0        1        0        0        0        0
## estrato3        0        0        1        0        0        0
## estrato4        0        0        0        1        0        0
## estrato5        0        0        0        0        1        0
## estrato6        0        0        0        0        0        1
```

En las filas tenemos las categorías originales y en las columnas las variables auxiliares, tantas como categorías existentes menos una. El *estrato0* lo ha codificado con el valor 0 en todas las variables auxiliares y al resto de categorías les pone el valor 1 en una variable auxiliar y 0 en el resto.

En la tabla sobre el uso de internet en los diferentes estratos, vimos que el *estrato6* (municipios de menos de 10 mil habitantes), es dónde hay una menor proporción del uso de internet (0.3853), así que vamos a tomar este estrato como referencia. Utilizando la función `relevel` podemos cambiar la categoría de referencia.

```
# para que sea permanente el cambio, guardamos el resultado de relevel en
# la misma variable habitat
datos.bin$habitat <- relevel(datos.bin$habitat, ref = "estrato6")
```

Con la función `levels`, vemos que ahora la primera categoría es *estrato6*

```
levels(datos.bin$habitat)

## [1] "estrato6" "estrato0" "estrato1" "estrato2" "estrato3" "estrato4"
## [7] "estrato5"
```

Y con `contrasts`, como sería la nueva recodificación en variables auxiliares.

```
contrasts(datos.bin$habitat)

##          estrato0 estrato1 estrato2 estrato3 estrato4 estrato5
## estrato6        0        0        0        0        0        0
## estrato0        1        0        0        0        0        0
## estrato1        0        1        0        0        0        0
## estrato2        0        0        1        0        0        0
## estrato3        0        0        0        1        0        0
## estrato4        0        0        0        0        1        0
## estrato5        0        0        0        0        0        1
```

El modelo ajustado si sólo tenemos como variable explicativa el hábitat sería:

$$logit\left[p(x)\right] = \ln\left[\frac{p(x)}{1-p(x)}\right] = \alpha + \beta_1 habitat$$

que habría que expresarlo en función de las 5 variables auxiliares (estrato6 es la categoría de referencia)

$$logit\left[p(x)\right] = \ln\left[\frac{p(x)}{1-p(x)}\right] = \alpha + \tau_{est0} \cdot estrato0 + \ldots + \tau_{est5} \cdot estrato5$$

Al haber especificado que hábitat es de tipo factor, R construye las variables auxiliares automáticamente. La sintaxis sería la siguiente.

```
modelo.3 <- glm(uso_int ~ habitat, data = datos.bin, family = binomial)
```

Es decir, en la sintaxis de la función glm sólo se indica el nombre de la variable categórica y la función identifica que se trata de una variable tipo factor e incorpora las variables auxiliares en la "matriz del modelo". Dicha matriz contiene una columna de unos para el intercept y las 5 columnas con las variables auxiliares. Podemos ver la matriz del modelo con la función `model.matrix`.

```
# matriz del modelo para los 3 primeros individuos
head(model.matrix(modelo.3), 3)

##   (Intercept) habitatestrato0 habitatestrato1 habitatestrato2
## 1           1               0               1               0
## 2           1               0               1               0
## 3           1               0               1               0
##   habitatestrato3 habitatestrato4 habitatestrato5
## 1               0               0               0
## 2               0               0               0
## 3               0               0               0
```

Así, el modelo de regresión logística con una variable categórica explicativa, se reduce al ajuste de un modelo de regresión logística con tantas variables explicativas continuas como categorías de la variable categórica menos una.

El resumen del modelo, extraído con `summary` es.

```
summary(modelo.3)

##
## Call:
## glm(formula = uso_int ~ habitat, family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.362  -1.160  -0.987   1.169   1.381
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.4673     0.0564   -8.29  < 2e-16 ***
## habitatestrato0   0.5665     0.1219    4.65  3.4e-06 ***
## habitatestrato1   0.5908     0.1096    5.39  7.1e-08 ***
## habitatestrato2   0.8915     0.1854    4.81  1.5e-06 ***
## habitatestrato3   0.5327     0.1275    4.18  3.0e-05 ***
## habitatestrato4   0.4272     0.1210    3.53  0.00042 ***
## habitatestrato5   0.4884     0.1010    4.84  1.3e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 4751.6  on 3478  degrees of freedom
## AIC: 4766
##
## Number of Fisher Scoring iterations: 4
```

La expresión del modelo es.

$$\begin{aligned} logit\left[p(x)\right] &= \ln\left[\frac{p(x)}{1-p(x)}\right] = -0{,}47 + 0{,}57 \cdot estrato0 \\ &+ 0{,}59 \cdot estrato1 + 0{,}89 \cdot estrato2 \\ &+ 0{,}53 \cdot estrato3 + 0{,}43 \cdot estrato4 \\ &+ 0{,}49 \cdot estrato5 \end{aligned}$$

Los coeficientes del modelo se interpretan en base a la categoría de referencia elegida.

Las variables de estrato toman el valor 1 para los individuos de ese estrato, y 0 para los que no pertenecen a ese estrato. Para un individuo del estrato 5 (municipios entre 10 mil y 20 mil habitantes) el logit de la ventaja de respuesta 1 estimado por el modelo sería.

$$logit\left[p(x)\right] = \ln\left[\frac{p(x)}{1-p(x)}\right] = -0{,}47 + 0{,}49 = 0{,}02$$

El coeficiente del intercept (-0.47), es el logit de la ventaja de la respuesta uso_int =1 en el estrato 6. Para comprobarlo, podemos calcular la probabilidad de *uso_int*=1, en el estrato6 como el inverso de la transformación logit del `intercept` del modelo.

```
# función invlogit para pasar de logit a probabilidades
invlogit <- function(x) {
    1/(1 + exp(-x))
}
# aplicamos la función invlogit al primer coeficiente del modelo.3
invlogit(coef(modelo.3)[1])

## (Intercept)
##     0.38525
```

Que coincide la proporción muestral del uso de internet en el estrato6, de igual forma se puede obtener las proporciones muestrales del resto de los estratos, sin más que calcular el inverso del logit de la suma entre el intercept y los distintos coeficientes de las variables auxiliares.

```
# estrato0
invlogit(coef(modelo.3)[1] + coef(modelo.3)[2])

## (Intercept)
##     0.52478

# estrato1
invlogit(coef(modelo.3)[1] + coef(modelo.3)[3])

## (Intercept)
##     0.53084

# estrato5
invlogit(coef(modelo.3)[1] + coef(modelo.3)[7])

## (Intercept)
##     0.50526
```

De nuevo, se pueden considerar los datos en formato agrupado, que como ya hemos visto anteriormente, dan las mismas estimaciones del modelo pero con valores distintos en la devianza. La tabla con los datos agrupados, dónde tenemos los 7 estratos posibles y el número de personas que responden que han usado internet o no, en cada estrato es.

	habitat	no.internet	si.internet
1	estrato0	163	180
2	estrato1	213	241
3	estrato2	53	81
4	estrato3	148	158
5	estrato4	178	171
6	estrato5	282	288
7	estrato6	817	512

Podemos repetir el ajuste del modelo partiendo de esta tabla.

```
# lectura del archivo csv dónde tenemos los datos agrupados
habt_agrup <- read.csv("../Datos/habitat_agrupado.csv")
habt_agrup

##    habitat no.internet si.internet
## 1 estrato0         163         180
## 2 estrato1         213         241
## 3 estrato2          53          81
## 4 estrato3         148         158
## 5 estrato4         178         171
## 6 estrato5         282         288
## 7 estrato6         817         512
```

```r
levels(habt_agrup$habitat)

## [1] "estrato0" "estrato1" "estrato2" "estrato3" "estrato4" "estrato5"
## [7] "estrato6"

# ponemos como categoría de referencia el estrato6.
habt_agrup$habitat <- relevel(habt_agrup$habitat, ref = "estrato6")
# ajuste del modelo
modelo.3.agrp <- glm(cbind(si.internet, no.internet) ~ habitat, data = habt_agrup,
    family = binomial)
```

El resumen del modelo es.

```r
summary(modelo.3.agrp)

##
## Call:
## glm(formula = cbind(si.internet, no.internet) ~ habitat, family = binomial,
##     data = habt_agrup)
##
## Deviance Residuals:
## [1]  0  0  0  0  0  0  0
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.4673     0.0564   -8.29  < 2e-16 ***
## habitatestrato0   0.5665     0.1219    4.65  3.4e-06 ***
## habitatestrato1   0.5908     0.1096    5.39  7.1e-08 ***
## habitatestrato2   0.8915     0.1854    4.81  1.5e-06 ***
## habitatestrato3   0.5327     0.1275    4.18  3.0e-05 ***
## habitatestrato4   0.4272     0.1210    3.53  0.00042 ***
## habitatestrato5   0.4884     0.1010    4.84  1.3e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.5340e+01  on 6  degrees of freedom
## Residual deviance: 3.6859e-14  on 0  degrees of freedom
## AIC: 59.04
##
## Number of Fisher Scoring iterations: 2
```

Comparando con el modelo anterior, vemos que las estimaciones son las mismas, pero que el valor de la devianza es distinto. En este caso, dicho valor es prácticamente 0, al igual que los grados de libertad. Esto se debe a que hemos ajustado 7 perfiles, en vez de 3.485 individuos, y hemos obtenido

un modelo con 7 parámetros. Es decir, para esta tabla, en la que sólo se ha considerado la variable hábitat, el modelo ajustado se corresponde con el modelo saturado, ya que tenemos tantos datos como parámetros estimados.

Considerar los datos en formato agrupado, cuando las variables explicativas son categóricas, tiene la ventaja de que podemos dar una medida del ajuste global del modelo, mediante los contrastes basados en el estadístico X^2 y G^2 sección (4.4.1). También es útil cuando no se disponen de los datos originales, si no solamente de las tablas de contingencia entre las variables.

3.3.2. Variables ordinales

Al tratar con variables ordinales, podemos hacer dos cosas, tratarlas como variables nominales en cuyo caso utilizaremos alguna forma de codificación como el método parcial, o bien asignarle puntuaciones numéricas que sean equidistantes.

Para el primer caso, basta con decirle a R que la variable es de tipo factor, mientras que en el segundo se puede recodificar utilizando las diversas funciones disponibles. Veamos por ejemplo, la variable *nivel de estudios,* la cual va desde "Analfabetos" hasta "Universitaria o superior". Esta variable la hemos definido como un factor.

```
class(datos.bin$nivelest)

## [1] "factor"
```

Las variables auxiliares asociadas serían.

```
contrasts(datos.bin$nivelest)

##                          Primaria Secundaria y F.P Universitaria o superior
## Analfabetos                     0                0                        0
## Primaria                        1                0                        0
## Secundaria y F.P                0                1                        0
## Universitaria o superior        0                0                        1
```

Si quisiéramos asignarle puntuaciones equidistantes, como por ejemplo los códigos del 1 al 4, podemos hacerlo pasando la variable factor a numérica o creando una nueva variable.

```
datos.bin$nivelest.num <- as.numeric(datos.bin$nivelest)
class(datos.bin$nivelest.num)

## [1] "numeric"

table(datos.bin$nivelest.num)

##
##    1    2    3    4
##  290 2061  692  442
```

Y al introducir `nivelest.num` en el modelo, la función `glm` la trataría como una variable cuantitativa.

```
modelo.4 <- glm(uso_int ~ nivelest.num, data = datos.bin, family = binomial)
summary(modelo.4)

##
## Call:
## glm(formula = uso_int ~ nivelest.num, family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.710  -0.855  -0.304   0.660   2.489
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.2841     0.1908   -27.7   <2e-16 ***
## nivelest.num   2.2325     0.0843    26.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 3434.1  on 3483  degrees of freedom
## AIC: 3438
##
## Number of Fisher Scoring iterations: 5
```

En R existe otra opción, que es considerar un tipo especial de factor, llamado "factor ordenado". Si definimos una variable categórica como un factor ordenado, R creará variables auxiliares con diferentes codificaciones equidistantes. Para crear un factor ordenado a partir del nivel de estudios, utilizamos la función `ordered`.

```
datos.bin$nivelest.ord <- ordered(datos.bin$nivelest)
class(datos.bin$nivelest.ord)

## [1] "ordered" "factor"

table(datos.bin$nivelest.ord)

##
##              Analfabetos                 Primaria         Secundaria y F.P
##                      290                     2061                      692
## Universitaria o superior
##                      442
```

Aparentemente nada ha cambiado, pero si utilizamos contrasts, vemos que considera 3 codificaciones distintas.

```
contrasts(datos.bin$nivelest.ord)

##            .L   .Q       .C
## [1,] -0.67082  0.5 -0.22361
## [2,] -0.22361 -0.5  0.67082
## [3,]  0.22361 -0.5 -0.67082
## [4,]  0.67082  0.5  0.22361
```

La primera codificación (.L) se corresponde con un efecto lineal, (.Q) a un efecto cuadrático y (.C) al cúbico. Al introducir la variable *nivelest.ord* en el modelo se introducen las 3 codificaciones .L, .Q y .C, de forma que podemos ver si el efecto de la variable ordinal es lineal o de un orden mayor.

```
modelo.4.ord <- glm(uso_int ~ nivelest.ord, data = datos.bin, family = binomial)
summary(modelo.4.ord)

##
## Call:
## glm(formula = uso_int ~ nivelest.ord, family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.265  -0.853  -0.144   0.559   3.024
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.281      0.155   -1.81     0.07 .
## nivelest.ord.L    5.309      0.408   13.01  < 2e-16 ***
## nivelest.ord.Q   -1.515      0.309   -4.90  9.7e-07 ***
## nivelest.ord.C   -0.171      0.157   -1.09     0.28
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817  on 3484  degrees of freedom
## Residual deviance: 3379  on 3481  degrees of freedom
## AIC: 3387
##
## Number of Fisher Scoring iterations: 7
```

Si quisiéramos quedarnos sólo con los efectos lineales y cuadráticos, construimos las variables con dicha codificación.

```
datos.bin$nivelest.lin[datos.bin$nivelest == "Analfabetos"] <- -0.67082
datos.bin$nivelest.lin[datos.bin$nivelest == "Primaria"] <- -0.22361
datos.bin$nivelest.lin[datos.bin$nivelest == "Secundaria y F.P"] <- 0.22361
datos.bin$nivelest.lin[datos.bin$nivelest == "Universitaria o superior"] <- 0.67082
```

```
datos.bin$nivelest.cuad[datos.bin$nivelest == "Analfabetos"] <- 0.5
datos.bin$nivelest.cuad[datos.bin$nivelest == "Primaria"] <- -0.5
datos.bin$nivelest.cuad[datos.bin$nivelest == "Secundaria y F.P"] <- -0.5
datos.bin$nivelest.cuad[datos.bin$nivelest == "Universitaria o superior"] <- 0.5

# el tipo de las variables es numérica
class(datos.bin$nivelest.lin)

## [1] "numeric"

class(datos.bin$nivelest.cuad)

## [1] "numeric"
```

Las introducimos en el modelo.

```
modelo.4.cuad <- glm(uso_int ~ nivelest.lin + nivelest.cuad, data = datos.bin,
    family = binomial)
summary(modelo.4.cuad)

##
## Call:
## glm(formula = uso_int ~ nivelest.lin + nivelest.cuad, family = binomial,
##     data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.282  -0.856  -0.109   0.570   3.203
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.420      0.106   -3.97  7.1e-05 ***
## nivelest.lin     5.702      0.238   23.92  < 2e-16 ***
## nivelest.cuad   -1.758      0.255   -6.90  5.3e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 3380.1  on 3482  degrees of freedom
## AIC: 3386
##
## Number of Fisher Scoring iterations: 6
```

3.3.3. Dos o más variables explicativas categóricas

La introducción de dos o más variables explicativas categóricas, se reduce a considerar las respectivas variables auxiliares. Lo único que cambia es la interpretación de los coeficientes del modelo, ya que por ejemplo, el intercept del modelo es el logit de la ventaja de respuesta 1 para un individuo que esté en las categorías de referencia de las variables categóricas.

En nuestro ejemplo, tomando como variables explicativas el hábitat y el nivel de estudios.

```
# vemos los niveles de ambas variables, el primer nivel será la categoría
# de referencia
levels(datos.bin$habitat)

## [1] "estrato6" "estrato0" "estrato1" "estrato2" "estrato3" "estrato4"
## [7] "estrato5"

levels(datos.bin$nivelest)

## [1] "Analfabetos"              "Primaria"
## [3] "Secundaria y F.P"         "Universitaria o superior"


modelo.5 <- glm(uso_int ~ habitat + nivelest, data = datos.bin, family = binomial)
summary(modelo.5)

##
## Call:
## glm(formula = uso_int ~ habitat + nivelest, family = binomial,
##     data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.539  -0.819  -0.136   0.579   2.990
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -4.673      0.582   -8.02  1.0e-15 ***
## habitatestrato0                   -0.090      0.158   -0.57   0.5683
## habitatestrato1                   -0.201      0.142   -1.42   0.1566
## habitatestrato2                    0.728      0.223    3.26   0.0011 **
## habitatestrato3                    0.315      0.160    1.97   0.0488 *
## habitatestrato4                    0.215      0.148    1.45   0.1462
## habitatestrato5                    0.269      0.124    2.16   0.0305 *
## nivelestPrimaria                   3.751      0.583    6.44  1.2e-10 ***
## nivelestSecundaria y F.P           6.373      0.591   10.78  < 2e-16 ***
## nivelestUniversitaria o superior   7.127      0.609   11.71  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 3354.7  on 3475  degrees of freedom
## AIC: 3375
##
## Number of Fisher Scoring iterations: 7
```

La expresión del modelo es

$$\begin{aligned} logit\,[p(x)] &= \ln\left[\frac{p(x)}{1-p(x)}\right] = -4{,}67 - 0{,}09 \cdot estrato0 \\ & \quad -0{,}2 \cdot estrato1 + 0{,}73 \cdot estrato2 \\ &+ 0{,}32 \cdot estrato3 + 0{,}22 \cdot estrato4 \\ &+ 0{,}27 \cdot estrato5 + 3{,}75 \cdot nivelestPrimaria \\ &+ 6{,}37 \cdot nivelestSecundaria\ y\ FP + 7{,}13 \cdot nivelestUniversitaria\ o\ superior \end{aligned}$$

dónde cada variable auxiliar vale 1 si el individuo está en esa categoría y 0 si no lo está.

El intercept del modelo es el logit estimado de la ventaja de la respuesta uso_int=1 para los individuos del estrato6 y con nivelest=Analfabetos.

La probabilidad estimada de haber utilizado internet para una persona que esté en ambas categorías de referencia, la podemos calcular utilizando la función `invlogit` definida anteriormente.

```
invlogit(coef(modelo.5)[1])
```

```
## (Intercept)
##   0.0092607
```

La interpretación de los otros coeficientes se hace en el mismo sentido. Por ejemplo, el logit de la ventaja de respuesta uso_int=1 para un individuo con estudios primarios del estrato6, sería el intercept más el coeficiente asociado a dicho nivel de estudios.

Si notamos el perfil "estrato6 y nivelest=Analfabetos" como x_1, y el perfil "estrato6 y nivelest="Primaria" como x_2, el cociente de ventajas entre x_2 y x_1

$$\theta(x_2, x_1) = \frac{\dfrac{p(x_2)}{1-p(x_2)}}{\dfrac{p(x_1)}{1-p(x_1)}} = \frac{e^{-4{,}67} \cdot e^{3{,}75}}{e^{-4{,}67}} = e^{3{,}75} \approx 42{,}5$$

La ventaja de haber usado alguna vez internet frente a no usarlo es 42.5 veces mayor para un individuo con estudios primarios del estrato6, que para un individuo sin estudios del mismo estrato.

La probabilidad estimada por el modelo para los encuestados del estrato6 y con estudios universitarios es.

```
invlogit(coef(modelo.5)[1] + coef(modelo.5)[8])

## (Intercept)
##     0.28464
```

3.3.4. Variables explicativas categóricas y cuantitativas

En la mayoría de los casos tendremos variables explicativas tanto categóricas como cuantitativas. Como ya hemos visto, la función `glm` trata con las variables auxiliares (o de diseño) para las variables categóricas, con lo que la función toma como matriz de variables explicativas las variables de diseño más las variables cuantitativas.

En los datos sobre el uso de internet, podríamos considerar un modelo dónde tengamos como variables explicativas, la edad, el hábitat y el nivel de estudios.

```
modelo.6 <- glm(uso_int ~ edad + habitat + nivelest, data = datos.bin, family = binomial)
summary(modelo.6)

##
## Call:
## glm(formula = uso_int ~ edad + habitat + nivelest, family = binomial,
##     data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.772  -0.418  -0.058   0.414   3.838
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      2.02398    0.65540    3.09   0.0020 **
## edad                            -0.10928    0.00429  -25.48  < 2e-16 ***
## habitatestrato0                  0.58261    0.20644    2.82   0.0048 **
## habitatestrato1                  0.48333    0.18203    2.66   0.0079 **
## habitatestrato2                  0.78570    0.28211    2.79   0.0054 **
## habitatestrato3                  0.45203    0.20131    2.25   0.0247 *
## habitatestrato4                  0.21821    0.18582    1.17   0.2403
## habitatestrato5                  0.22866    0.15879    1.44   0.1499
## nivelestPrimaria                 2.46526    0.62190    3.96  7.4e-05 ***
## nivelestSecundaria y F.P         4.72038    0.63245    7.46  8.4e-14 ***
## nivelestUniversitaria o superior 5.82548    0.65456    8.90  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
```

```
## Residual deviance: 2203.4  on 3474  degrees of freedom
## AIC: 2225
##
## Number of Fisher Scoring iterations: 7
```

La interpretación de los coeficientes de las variables de diseño, se realiza de igual forma que en el modelo anterior, es decir, en relación a las categorías de referencia. El coeficiente asociado a edad (-0.109) es muy parecido al que obtuvimos en el modelo que sólo tenía la *edad* como variable explicativa (-0.112)

A partir de los coeficientes, podemos representar gráficamente el modelo ajustado. En la figura (3.6) vemos el ajuste para 4 de los 7 estratos. El código utilizado para obtener la figura es el siguiente.

```
# guardamos los coeficientes del modelo en un objeto
coef.m <- coef(modelo.6)

# en cond selecciono los encuestados del estrato6
cond <- datos.bin$habitat == "estrato6"

# edad vs uso_int añadiendo una pequeña perturbación aleatorio para evitar
# solapamiento de datos.  se han utilizado parámetros gráficos para
# ajustar color, tipo de líneas, y anchura ver help(par)
with(datos.bin[cond, ], plot(jitter(edad), jitter(uso_int, 0.2), xlab = "Edad",
    ylab = quote(p(x)), main = "Estrato 6", cex = 0.4, cex.axis = 0.6, cex.lab = 0.6,
    cex.main = 0.7))

# Curva del modelo cuando nivelest='Analfabetos' y habitat=estrato6
curve(1/(1 + exp(-(coef.m[1] + coef.m[2] * x))), col = "darkgreen", lwd = 2,
    add = TRUE)
# Curva del modelo cuando nivelest='Primaria'
curve(1/(1 + exp(-(coef.m[1] + coef.m[2] * x + coef.m[9]))), lty = 2, lwd = 2,
    add = TRUE)

# Curva del modelo cuando nivelest='Secundaria y F.P'
curve(1/(1 + exp(-(coef.m[1] + coef.m[2] * x + coef.m[10]))), lty = 3, lwd = 2,
    add = TRUE)
# Curva del modelo cuando nivelest='Universitaria o superior'
curve(1/(1 + exp(-(coef.m[1] + coef.m[2] * x + coef.m[11]))), lty = 4, lwd = 2,
    add = TRUE)

# guardamos los nombres de los niveles en un objeto que usamos para poner
# la leyenda
nombres <- levels(datos.bin$nivelest)
legend(x = 70, y = 0.95, nombres, col = c("darkgreen", rep("black", 3)), cex = 0.7,
    lty = 1:4, lwd = rep(2, 4))
```

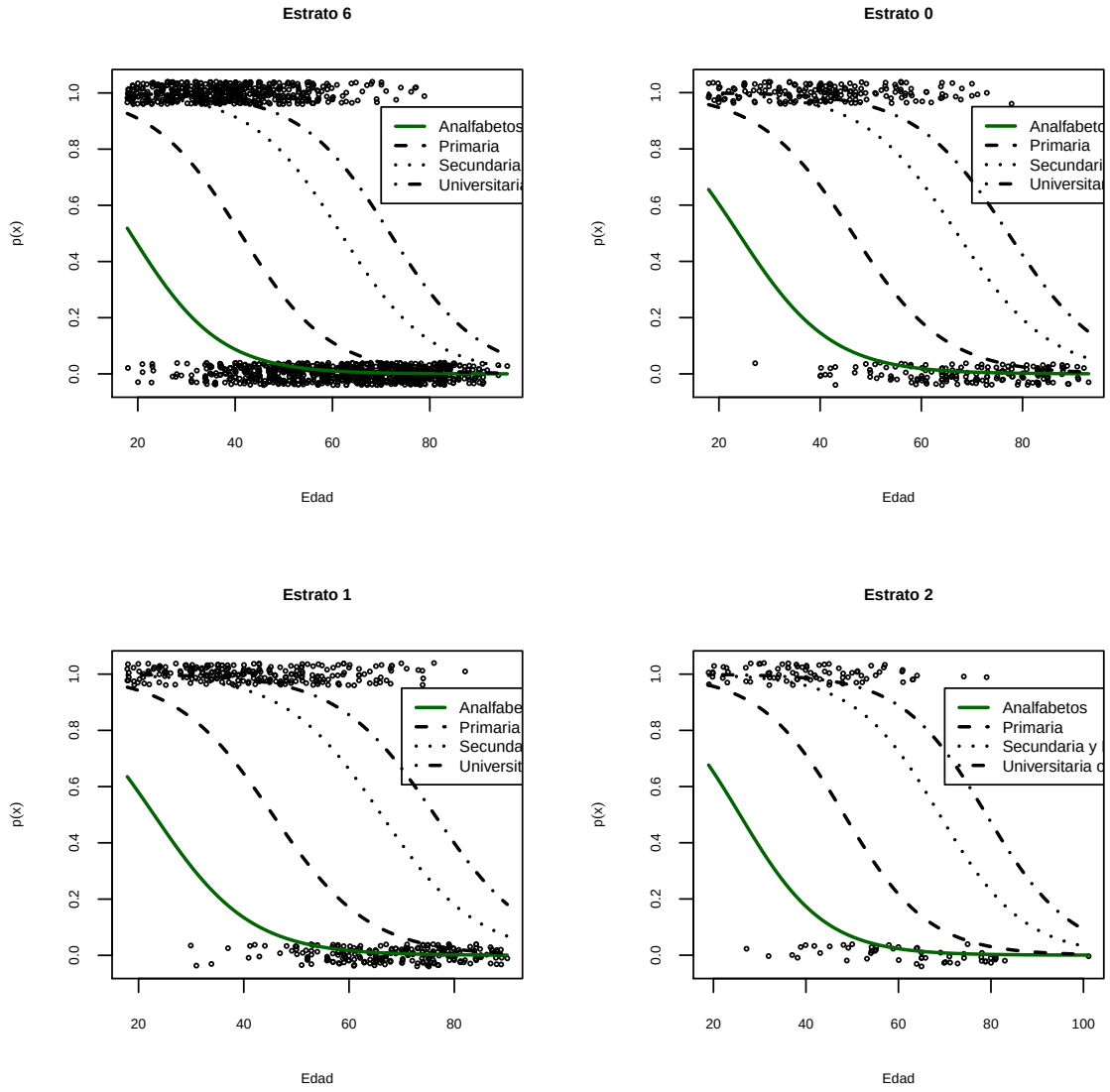


Figura 3.6: Curvas logit para el modelo con una variable cuantitativa (*edad*), y dos variables categóricas (*habitat* y *nivel de estudios*). Solo estratos 6,0, 1 y 2

3.4. Interacción

En el caso de variables predictoras continuas, la interacción entre dos o más de ellas se introduce en el modelo mediante el producto de sus valores. Cuando tenemos dos variables categóricas, la interacción se introduce como todos los posibles productos cruzados, es decir, se consideran todas las posibles combinaciones entre las categorías de ambas variables. La interacción entre una variable cuantitativa X y una cualitativa A, se introduce en el modelo como los productos de la variable cuantitativa por todas las variables de diseño asociadas a la variable A, lo que equivale a considerar la variable cuantitativa en cada uno de los niveles de la variable cualitativa.

En R se puede especificar la interacción mediante el operador "**:**" que especifica la interacción o mediante el operador "*****" que indica los efectos principales y la interacción.

Si, por ejemplo, queremos ajustar un modelo con los efectos principales de edad y nivel de estudios más

la interacción entre ambas variables, podemos utilizar cualquiera de las siguientes formas de expresar el modelo.

```
glm(uso_int ~ edad + nivelest + edad:nivelest, data = datos.bin, family = binomial)
glm(uso_int ~ edad * nivelest, data = datos.bin, family = binomial)
```

Ajustando el modelo obtenemos los coeficientes asociados a *edad, nivelest* y también un coeficiente para cada combinación de *edad* y *nivelest.*

```
modelo.7 <- glm(uso_int ~ edad * nivelest, data = datos.bin, family = binomial)
summary(modelo.7)
```

```
##
## Call:
## glm(formula = uso_int ~ edad * nivelest, family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.939  -0.427  -0.126   0.415   3.184
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           -2.4613     2.8080   -0.88   0.3807
## edad                                  -0.0296     0.0401   -0.74   0.4616
## nivelestPrimaria                       7.0517     2.8179    2.50   0.0123 *
## nivelestSecundaria y F.P               9.2508     2.8604    3.23   0.0012 **
## nivelestUniversitaria o superior      12.4570     3.0372    4.10  4.1e-05 ***
## edad:nivelestPrimaria                 -0.0772     0.0404   -1.91   0.0562 .
## edad:nivelestSecundaria y F.P         -0.0747     0.0414   -1.80   0.0711 .
## edad:nivelestUniversitaria o superior -0.1093     0.0442   -2.47   0.0135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 2213.5  on 3477  degrees of freedom
## AIC: 2229
##
## Number of Fisher Scoring iterations: 7
```

Los coeficientes asociados a cada combinación de *edad* y *nivelest,* nos indica cómo varía el efecto de la edad sobre el logit de la ventaja de respueste uso_int=1, en cada categoría del nivel de estudios.

Sin la interacción, el modelo equivale a considerar que el efecto de la variable edad es el mismo en

cada categoría del nivel de estudios. La expresión del modelo que incluye la interacción es

$$\begin{aligned}\text{logit}\left(\frac{p(x)}{1-p(x)}\right) &= -2{,}46 - 0{,}030 \cdot edad + 7{,}05 \cdot Primaria + 9{,}25 \cdot Secundaria + 12{,}46 \cdot Universitaria \\ &\quad -0{,}08 \cdot edad : primaria - 0{,}08 \cdot edad : secundaria - 0{,}11 \cdot edad : Universitaria\end{aligned}$$

dónde las variables *Primaria, Secundaria y Universitaria* valen 1 si el encuestado está en esa categoría y 0 si no lo está. Aplicando esta expresión a los encuestados cuyo nivel de estudios sea *Primaria,* nos queda.

$$\begin{aligned}\text{logit}\left(\frac{p(x)}{1-p(x)}\right) &= -2{,}46 - 0{,}030 \cdot edad + 7{,}05 - 0{,}08 \cdot edad : primaria \\ &= (-2{,}46 + 7{,}05) - (0{,}030 + 0{,}08) \cdot edad\end{aligned}$$

Comparando con la obtenida para *Universitaria o superior.*

$$\begin{aligned}\text{logit}\left(\frac{p(x)}{1-p(x)}\right) &= -2{,}46 - 0{,}030 \cdot edad + 12{,}46 - 0{,}11 \cdot edad : universitaria \\ &= (-2{,}46 + 12{,}46) - (0{,}030 + 0{,}11) \cdot edad\end{aligned}$$

Pertenecer a una categoría de nivel de estudios u otra modifica, tanto el término constante como la pendiente de las rectas de los logit de las ventajas. En un modelo que sólo tuviera los efectos principales, la pendiente asociada a la edad sería la misma en cada nivel de estudios y sólo variaría el término constante. En la figura (3.7) se han representado tanto en escala de los logits de las ventajas como en términos de probabilidad ajustada, el modelo sin interacción y con interacción. En el modelo sin interacción se tienen rectas paralelas para los logits de las ventajas, mientras que en el caso del modelo con interacción no lo son.

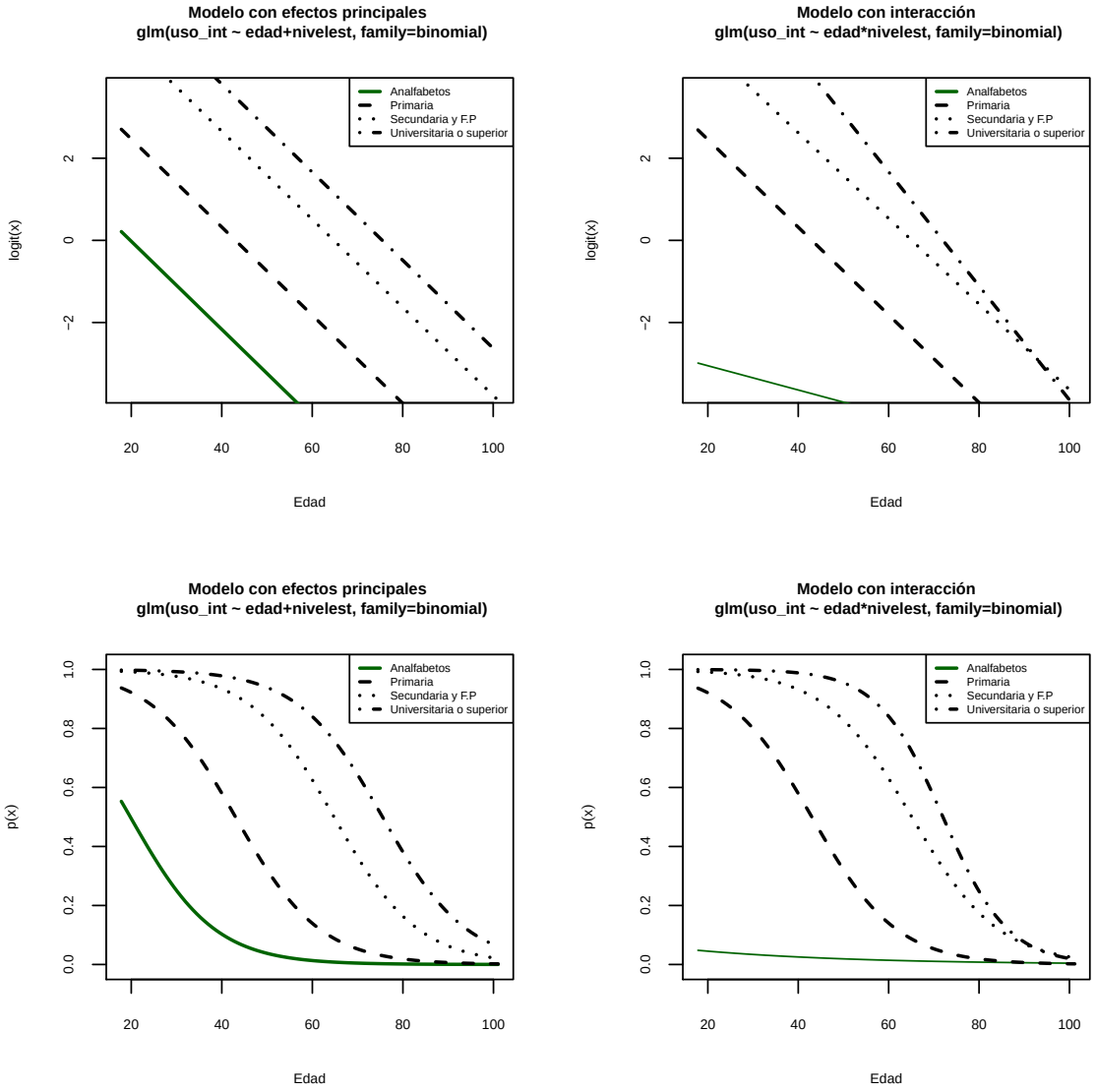


Figura 3.7: Curvas logit para el modelo con efectos principales de *edad* y *nivelest* (izquierda), y el modelo con los efectos principales de *edad* y *nivelest* más la interacción entre ambas (derecha).

Capítulo 4

Inferencia en modelos de regresión logística

Una vez estimado el modelo, realizamos inferencia con el fin de extrapolar los resultados al total de la población. En este capítulo explicamos como realizar dicha inferencia con R, incluyendo el cálculo de los contrastes de hipótesis e intervalos de confianza, obtención de valores predichos y residuos y los contrastes de bondad del ajuste global del modelo. Por último tratamos brevemente la selección automática de variables utilizando criterios de información.

4.1. Contraste sobre los parámetros

Una vez estimado el modelo, partiendo de que se ha realizado un muestreo probabilístico, nos interesa contrastar si los coeficientes estimados son significativamente distintos de 0. Es decir, si una determinada variable explicativa tiene un efecto significativo sobre la respuesta o no. Para dar respuesta a esta pregunta, utilizamos los contrastes de hipótesis sobre los parámetros, explicando dos métodos para realizarlos, los contrastes basados en el test de Wald y los basados en el test condicional de razón de verosimilitudes entre modelos anidados.

4.1.1. Contraste de Wald

Este contraste está basado en la normalidad asintótica de los estimadores. Se quiere contrastar si un parámetro $\beta_r = 0$, con $r = \{1, \ldots, R\}$, frente a que no lo sea.

$$\begin{aligned} H_0 : \beta_r &= 0 \\ H_1 : \beta_r &\neq 0 \end{aligned}$$

Wald, demostró que bajo la hipótesis nula

$$\frac{\widehat{\beta_r}}{\widehat{SE(\beta_r)}} \rightsquigarrow \mathcal{N}(0,1)$$

$\widehat{\beta_r}$ y $\widehat{SE(\beta_r)}$ son las estimaciones del modelo para β_r y el error estándar de β_r.

El estadístico de contraste es:

$$W_r = \frac{\hat{\beta}_r}{\widehat{SE\,(\beta_r)}}$$

Para ilustrar este contraste, consideremos el modelo de regresión logística para el *uso de internet,* con el *sexo* y la *edad* como variables explicativas.

```
modelo.edadsex <- glm(uso_int ~ edad + sexo, data = datos.bin, family = binomial)
summary(modelo.edadsex)

##
## Call:
## glm(formula = uso_int ~ edad + sexo, family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.691  -0.585  -0.198   0.648   2.923
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.71475    0.19260   29.67   <2e-16 ***
## edad        -0.11214    0.00361  -31.08   <2e-16 ***
## sexoMujer   -0.10400    0.09307   -1.12     0.26
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 2915.6  on 3482  degrees of freedom
## AIC: 2922
##
## Number of Fisher Scoring iterations: 5
```

En el resultado de la función `summary` se muestran los símbolos "*" y ".", que indican la significación de los parámetros a diferentes niveles. A un nivel de significación $\alpha = 0{,}05$ los parámetros asociados al intercept y a la edad son significativamente distintos de 0, ya que el valor del estadístico de Wald es de 29.67 y -31.08 respectivamente, que en valor absoluto son mayores que el punto crítico $z_{\alpha/2} = 1{,}96$. El parámetro asociado a la variable *sexoMujer* no es significativamente distinto de 0 según este contraste.

Si tenemos interés en trabajar directamente con la tabla de coeficientes, la podemos extraer utilizando el símbolo `$`.

```
summary(modelo.edadsex)$coefficients

##             Estimate Std. Error z value   Pr(>|z|)
```

```
## (Intercept)   5.7147   0.192599  29.672 1.781e-193
## edad         -0.1121   0.003608 -31.084 3.956e-212
## sexoMujer    -0.1040   0.093073  -1.117  2.638e-01
```

Los valores del estadístico de Wald están en la tercera columna de la tabla anterior.

```
summary(modelo.edadsex)$coefficients[, 3]

## (Intercept)        edad   sexoMujer
##      29.672     -31.084      -1.117
```

4.1.2. Contraste condicional de razón de verosimilitud

Las hipótesis de este contraste son las mismas que en el anterior

$$\begin{aligned} H_0 : \beta_r &= 0 \\ H_1 : \beta_r &\neq 0 \end{aligned}$$

pero el enfoque es distinto. En vez de considerar la distribución de los parámetros se comparan dos modelos, uno dónde se haya estimado el parámetro β_r, frente a otro modelo que se diferencie del primero en que no esté dicho parámetro, pero si el resto de los parámetros. Es decir, comparamos modelos anidados. El test que se utiliza es el test condicional de razón de verosimilitudes.

Por ejemplo, si queremos contrastar si el parámetro asociado a la *edad* es 0, ajustamos un modelo con esa variable, (*modelo.edadsex*), y otro modelo en el que no esté esa variable (*modelo.sex*).

El estadístico asociado al contraste es

$$\begin{aligned} G^2_{\text{modelo.sex}} \mid G^2_{\text{modelo.edadsex}} &= -2 \log \frac{V_{\text{modelo.sex}}}{V_{\text{modelo.edadsex}}} \\ &= -2 \left(L_{\text{modelo.sex}} - L_{\text{modelo.edadsex}} \right) \\ &= -2 \left(L_{\text{modelo.sex}} - L_{Saturado} \right) - \left(-2 \left(L_{\text{modelo.edadsex}} - L_{Saturado} \right) \right) \\ &= G^2_{\text{modelo.sex}} - G^2_{\text{modelo.edadsex}} \end{aligned}$$

dónde V es la máxima verosimilitud y L la máxima log-verosimilitud en cada modelo. El estadístico del test es la disminución en la devianza que se produce al pasar de un modelo con la variable *sexo* a otro al que se le añade la variable *edad.*

Este estadístico sigue, bajo el modelo sin la variable *edad*, una distribución chi-cuadrado con grados de libertad igual a la diferencia de grados de libertad entre las distribuciones chi-cuadrado asintóticas de $G^2_{modelo.sex}$ y $G^2_{modelo.edadsex}$. Es decir, si la diferencia entre los dos modelos es en un parámetro, la distribución del estadístico, bajo la hipótesis nula de que ese parámetro es 0, sigue una distribución chi-cuadrado con un grado de libertad.

Se rechaza la hipótesis nula cuando

$\left(G^2_{\text{modelo.sex}} \mid G^2_{\text{modelo.edadsex}} \right) \geq \chi^2_{1;\alpha}$

con α el nivel de significación.

En R se calcularía de la siguiente forma

```
modelo.sex <- glm(uso_int ~ sexo, data = datos.bin, family = binomial)
G2 <- modelo.sex$deviance - modelo.edadsex$deviance
G2

## [1] 1882
```

Y el p-valor asociado al contraste

```
1 - pchisq(G2, df = 1)

## [1] 0
```

Con lo que se rechaza la hipótesis nula de que el parámetro asociado a la edad es 0.

Podemos calcular este contraste de forma más compacta utilizando la función `anova` del paquete `stats` (que se carga por defecto al inicio de la sesión), o bien utilizando la función `Anova`[1] del paquete `car`, la cual realiza los contrastes condicionales de razón de verosimilitud sobre los parámetros asociados a cada una de las variables del modelo, sin necesidad de especificar los distintos modelos.

```
# anova normal
anova(modelo.sex, modelo.edadsex, test = "Chisq")

## Analysis of Deviance Table
##
## Model 1: uso_int ~ sexo
## Model 2: uso_int ~ edad + sexo
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3483       4797
## 2      3482       2916  1     1882   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Anova del paquete car
library(car)
Anova(modelo.edadsex)

## Analysis of Deviance Table (Type II tests)
##
## Response: uso_int
##      LR Chisq Df Pr(>Chisq)
## edad     1882  1     <2e-16 ***
## sexo        1  1       0.26
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[1] R es sensible a mayúsculas y minúsculas, y entiende que la función `Anova` es distinta de `anova`

La función Anova ha realizado dos contrastes condicionales de razón de verosimilitud. En el primero se contrasta

$$\begin{aligned} H_0 : \beta_{edad} &= 0 \\ H_1 : \beta_{edad} &\neq 0 \end{aligned}$$

comparando el modelo con las variables *edad* y *sexo* con el modelo que sólo tiene la variable *sexo*. Es el mismo contraste que cuando escribimos `anova(modelo.sex,modelo.edadsex,test="Chisq")`.

El segundo contraste que realiza es

$$\begin{aligned} H_0 : \beta_{sexo} &= 0 \\ H_1 : \beta_{sexo} &\neq 0 \end{aligned}$$

comparando el modelo con *edad* y *sexo* con el modelo que sólo tiene la variable *edad*.

El contraste condicional de razón de verosimilitudes nos sirve también si queremos contrastar si un subconjunto de parámetros son iguales a 0, frente a que alguno de ellos no lo sea. Si queremos contrastar que los parámetros asociados a la *edad* y al *sexo* son iguales a 0, escribimos lo siguiente.

```
modelo.cte <- glm(uso_int ~ 1, data = datos.bin, family = binomial)
anova(modelo.cte, modelo.edadsex, test = "Chisq")

## Analysis of Deviance Table
##
## Model 1: uso_int ~ 1
## Model 2: uso_int ~ edad + sexo
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3484       4817
## 2      3482       2916  2     1901   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Y se rechazaría la hipótesis nula de que ambos parámetros son 0.

(Hauck and Donner, 1977) examinaron el desempeño del test de Wald y encontraron que a menudo falla para rechazar la hipótesis nula. En el artículo recomiendan el contraste condicional de razón de verosimilud . Tanto el test de Wald como el LRT[2] requieren el cálculo de la estimación máximo verosímil de β. Un test que simplifica estos cálculos es el test de Score, basado en las propiedades de las derivadas parciales de la función de verosimilitud. En R podemos utilizar el argumento `test="Rao"` dentro de la función `anova` para obtener este contraste. La potencia de cálculo ya no es un problema y el test Score no suele utilizarse.

[2]Contraste condicional de razón de verosimiludes por sus siglas en inglés (Likelihood Ratio test)

```
anova(modelo.sex, modelo.edadsex, test = "Rao")

## Analysis of Deviance Table
##
## Model 1: uso_int ~ sexo
## Model 2: uso_int ~ edad + sexo
##   Resid. Df Resid. Dev Df Deviance  Rao Pr(>Chi)
## 1      3483       4797
## 2      3482       2916  1     1882 1541   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2. Intervalos de confianza para los parámetros

Existe una relación estrecha entre los contrastes de hipótesis y los intervalos de confianza. Un intervalo de confianza a nivel $1-\alpha$, está directamente relacionado con un contraste cuyo nivel de significación sea α. Podemos construir por tanto, intervalos de confianza basados en el test de Wald y basados en el test condicional de razón de verosimilitudes. Otro tipo de intervalos de confianza, no asociados a los contrastes de hipótesis vistos, son los derivados de remuestrear los datos y estimar el modelo en cada muestra, a esta técnica se la conoce como bootstrap (Efron, 1979). Los intervalos de confianza más que dar una respuesta positiva o negativa sobre si un determinado parámetro es significativo, calculan un intervalo de valores plausibles.

4.2.1. Intervalos de confianza para los parámetros basados en el test de Wald

Intervalos de confianza para β_r Se basan en que los parámetros β_r siguen asintóticamente una distribución $\mathcal{N}\left(\beta_r, \hat{\sigma}^2(\hat{\beta}_r)\right)$ con lo que

$$P\left[-z_{\frac{\alpha}{2}} \leq \frac{\hat{\beta}_r - \beta_r}{\hat{\sigma}(\hat{\beta}_r)} \leq z_{\frac{\alpha}{2}}\right] = 1-\alpha$$

Y un intervalo aproximado para β_r a nivel $(1-\alpha)$ sería:

$$\hat{\beta}_r \pm z_{\frac{\alpha}{2}} \hat{\sigma}(\hat{\beta}_r)$$

Los intervalos de confianza a un nivel $(1-\alpha)$, se suelen expresar en términos de porcentajes, y se habla entonces de intervalos de confianza al 90 %, 95 % o 99 %, según sea $\alpha = 0{,}10$; 0,05 o 0,01. La interpretación usual de un intervalo de confianza al $100 \cdot (1-\alpha)$ % es, que si repitiéramos la muestra aleatoria un elevado número de veces y en cada una de ellas construyéramos un intervalo de confianza para el parámetro β_r, aproximadamente el $100 \cdot (1-\alpha)$ % de los intervalos contendrían al verdadero valor del parámetro.

Se calculan con la función `confint.default`, que toma por defecto un nivel de confianza $1-\alpha = 0{,}95$

```
# intervalos al 95%
confint.default(modelo.edadsex)

##                2.5 %   97.5 %
## (Intercept)  5.3373  6.09223
## edad        -0.1192 -0.10507
## sexoMujer   -0.2864  0.07842

# intervalos al 90%
confint.default(modelo.edadsex, level = 0.9)

##                  5 %     95 %
## (Intercept)  5.3979  6.03154
## edad        -0.1181 -0.10620
## sexoMujer   -0.2571  0.04909
```

Si el intervalo de confianza incluye el 0, significa que al nivel α elegido no se podría rechazar la hipótesis nula de que $\beta_r = 0$

En este caso, el intervalo de confianza asociado al coeficiente *sexoMujer,* incluye al 0.

Intervalos de confianza para los e^{β_r}

Debido a la interpretación de los parámetros en los modelos de regresión logística, se suelen calcular los intervalos de confianza para los exponenciales de los parámetros, que se corresponden con los cocientes de ventajas. En estos casos, el contraste asociado se define como

$$\begin{aligned} H_0 : e^{\beta_r} &= 1 \\ H_1 : e^{\beta_r} &\neq 1 \end{aligned}$$

Y los intervalos de confianza se obtienen a partir de los intervalos anteriores, sin más que aplicarles la función e^x, el cálculo para los intervalos de confianza de los e^{β_r} sería

```
exp(confint.default(modelo.edadsex))

##                 2.5 %   97.5 %
## (Intercept) 207.9420 442.4088
## edad          0.8876   0.9003
## sexoMujer     0.7509   1.0816
```

y llegamos a la misma conclusión sobre el I.C asociado a *sexoMujer,* ya que en este caso el 1 cae dentro del intervalo.

4.2.2. Intervalos de confianza para los parámetros basados en el test condicional de razón de verosimilitudes.

La idea consiste en invertir el contraste condicional de razón de verosimilitudes para obtener los I.C, para eso se considera una aproximación a través de la denominada *profile-likelihood* (Venzon and

Moolgavkar, 1988) .

Sea $\hat{\theta}$ la estimación máximo verosímil del vector de parámetros $\theta \in \Theta \subseteq \mathcal{R}^p$ y $L(\bullet)$ la función de log-verosimilitud, es decir.

$$L\left(\hat{\theta}\right) = \max_{\theta \in \Theta} L(\theta)$$

Suponiendo que θ_j es el parámetro de interés, se considera la restricción sobre el espacio de parámetros Θ, de forma que θ_j es fijo y su valor es β_0. El espacio de parámetros restringido se define como $\Theta_j(\beta_0) = \{\theta \in \Theta \mid \theta_j = \beta_0\}$, entonces a la función

$$\bar{L}_j(\beta) = \max_{\theta \in \Theta_j(\beta_0)} L(\theta) \tag{4.1}$$

se la denomina profile-likelihood para β_0. La evaluación de la ecuación (4.1) implica la maximización de la función de log-verosimilitud con la restricción de que $\theta_j = \beta_0$. Un intervalo de confianza a nivel $100\,(1-2\alpha)\,\%$ para θ_j basado en esta función, viene dado por

$$\left\{\beta \mid -2\left[\bar{L}_j(\beta) - L\left(\hat{\theta}\right)\right] \leq q_1(1-\alpha)\right\} \tag{4.2}$$

Dónde $q_1(1-\alpha)$ es el cuantil $(1-\alpha)$ de una distribución chi-cuadrado con un grado de libertad. Es decir, en el intervalo estarán todos los β's, posibles estimadores del parámetro θ_j, que cumplan que, menos dos veces la diferencia entre el máximo de la log-verosimilitud en el espacio $\Theta_j(\beta_0)$, y la log-verosimilitud en el espacio Θ sea menor o igual que $q_1(1-\alpha)$

Estos intervalos suelen exigir un mayor coste computacional que los basados en el estadístico de Wald, ya que implica calcular no sólo la estimación máximo verosímil para cada parámetro, sino el cálculo del rango de las estimaciones que, desvíandose de la estimación máximo verosímil, están dentro del intervalo definido en (4.2). Estos intervalos tienen las características de ser más precisos (Venables and Ripley, 2002) y de que no tienen por qué ser simétricos.

El paquete `MASS` de R incorpora la función `confint` para calcularlos.

```
library(MASS)
```

Para el modelo con sexo y edad, se tienen los siguientes intervalos de confianza al 95 % para los parámetros.

```
# I.C para los parametros
confint(modelo.edadsex)

## Waiting for profiling to be done...

##                 2.5 %   97.5 %
## (Intercept)  5.3439  6.09912
## edad        -0.1193 -0.10520
## sexoMujer   -0.2864  0.07852
```

Y los intervalos de confianza para el exponencial de los parámetros, se calculan utilizando la función `exp`

```
# I.C para los exp(theta)
exp(confint(modelo.edadsex))

## Waiting for profiling to be done...

##                 2.5 %   97.5 %
## (Intercept) 209.3289 445.4675
## edad          0.8875   0.9001
## sexoMujer     0.7509   1.0817
```

Al igual que con la función `confint.default` podemos especificar el nivel de confianza con el argumento `level`. Los intervalos de confianza para los parámetros al 90 % son.

```
confint(modelo.edadsex, level = 0.9)

## Waiting for profiling to be done...

##                  5 %     95 %
## (Intercept)  5.4026  6.03638
## edad        -0.1182 -0.10630
## sexoMujer   -0.2571  0.04915
```

Y para los e^{β_r}

```
exp(confint(modelo.edadsex, level = 0.9))

## Waiting for profiling to be done...

##                   5 %     95 %
## (Intercept) 221.9930 418.3768
## edad          0.8885   0.8992
## sexoMujer     0.7733   1.0504
```

4.2.3. Intervalos de confianza para los parámetros basados en técnicas de remuestreo (bootstrap)

La capacidad computacional actual permite derivar la distribución de los parámetros a través de técnicas de remuestreo como el bootstrap (Efron, 1979). El procedimiento consiste en tratar la muestra original como si fuera la población. A partir de esta población ficticia, se extraen, mediante un muestreo aleatorio con reemplazamiento, un elevado número de muestras. Si el número de muestras es lo suficientemente grande, se obtiene la distribución en el muestreo de los parámetros. Los intervalos de confianza se calculan quedándonos con el $100 \cdot (1 - \alpha)\,\%$ de los valores centrales en la distribuciones empíricas obtenidas.

Se pueden programar funciones que realicen bootstrap y obtener los intervalos de confianza, aunque existen librerías específicas para realizar bootstrap, tales como la librería `boot` y también funciones

específicas para determinadas técnicas. Vamos a ver, por un lado la función `bootCase` del paquete `car` (Fox and Weisberg, 2011) y por otro la utilización de la librería `boot` (Canty and Ripley, 2012).

La función `bootCase` es muy sencilla de utilizar y se puede utilizar con objetos de la clase `lm`, `glm` y `nls` (nonlinear least squares). Toma como argumentos el modelo ajustado, una función f que aplicada a cada muestra calcula el estadístico de interés (por defecto los coeficientes del modelo), y el número de muestras bootstrap a calcular (por defecto 999).

Calculamos los coeficientes para el modelo con variables explicativas *edad* y *sexo*. Consideramos 1.000 muestras.

```
# Primero guardamos los coeficientes del modelo ajustado con glm
betahat <- coef(modelo.edadsex)
# Ahora le decimos a bootCase que tome 1000 muestras con reemplazamiento y
# calcule los coeficientes.  Los guardamos en el objeto betahat.boot
betahat.boot <- bootCase(modelo.edadsex, B = 1000)
```

Los resultados para las 6 primeras muestras son

```
head(betahat.boot)
```

```
##      (Intercept)    edad sexoMujer
## [1,]       5.580 -0.1105 -0.054714
## [2,]       5.455 -0.1074 -0.032192
## [3,]       5.722 -0.1114 -0.151557
## [4,]       5.602 -0.1111  0.014625
## [5,]       5.741 -0.1138 -0.009477
## [6,]       5.697 -0.1089 -0.057202
```

Un resumen de los parámetros estimados

```
summary(betahat.boot)
```

```
##   (Intercept)        edad          sexoMujer
##  Min.   :5.14   Min.   :-0.123   Min.   :-0.3852
##  1st Qu.:5.61   1st Qu.:-0.115   1st Qu.:-0.1656
##  Median :5.73   Median :-0.112   Median :-0.1031
##  Mean   :5.73   Mean   :-0.112   Mean   :-0.1046
##  3rd Qu.:5.85   3rd Qu.:-0.110   3rd Qu.:-0.0388
##  Max.   :6.34   Max.   :-0.102   Max.   : 0.2083
```

En la figura (4.1) se muestran los histogramas de los coeficientes estimados mediante bootstrap.

```
par(mfrow = c(2, 2))
for (i in 1:3) hist(betahat.boot[, i], main = names(coef(modelo.edadsex))[i],
    xlab = "")
```

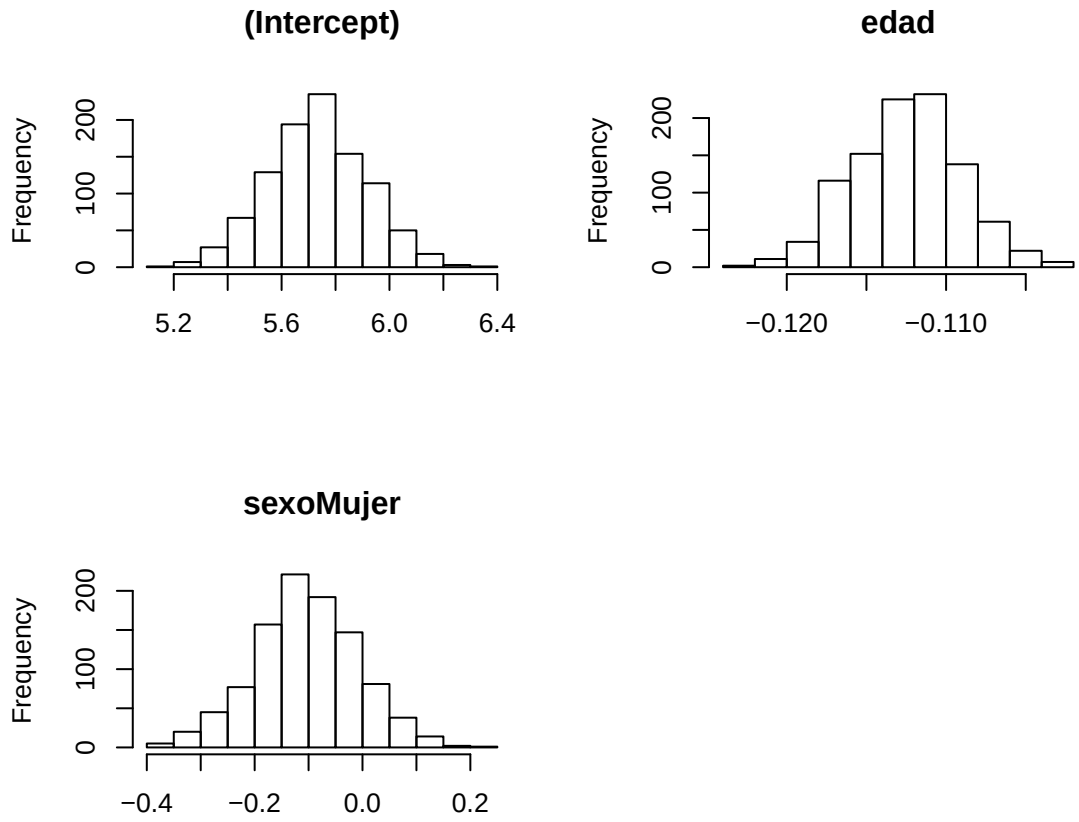


Figura 4.1: Histogramas de los coeficientes estimados, mediante `bootCase`

Para cacular los intervalos de confianza, podemos utilizar la función `quantile`.

```
# calculamos la desviación típica y los I.C uniéndolos en un solo objeto
boot.interval <- cbind(`Bootstrap SD` = apply(betahat.boot, 2, sd), t(apply(betahat.boot,
    2, function(x) quantile(x, c(0.025, 0.975)))))
boot.interval
```

```
##             Bootstrap SD    2.5%    97.5%
## (Intercept)     0.180250  5.3861  6.09098
## edad            0.003357 -0.1192 -0.10586
## sexoMujer       0.095439 -0.2964  0.08447
```

Al calcular los intervalos hemos combinado varias funciones en un solo comando, `cbind` para unir por columnas, `apply`, que permite aplicar una función ya existente o definida por nosotros, a las filas o a las columnas de una matriz y también la función para transponer `t`.

También podemos usar la librería `boot` para realizar el bootstrap. En este caso tenemos que construir antes una función que tome como argumentos el data.frame que se va a utilizar y un vector de índices.

La función debe calcular los valores de interés, que en nuestro caso son los coeficientes estimados en la regresión logística.

```
library(boot)
# Creamos una función que calcule los coeficientes de unos datos es
# importante que uno de los argumentos sean los indices
coeficientes <- function(formula, data, indices) {
    d <- data[indices, ]
    fit <- glm(formula, data = d, family = binomial)
    return(coef(fit))
}
# La función boot remuestrea los datos y calcula los coeficientes.
# establecemos una semilla para que siempre nos salga el mismo resultado
set.seed(234)
res.boot <- boot(data = datos.bin, statistic = coeficientes, R = 1000, formula = uso_int ~
    sexo + edad)
```

Vemos los primeros valores

```
head(res.boot$t)

##       [,1]     [,2]    [,3]
## [1,] 5.626 -0.08948 -0.1101
## [2,] 5.807  0.05827 -0.1169
## [3,] 5.523 -0.19157 -0.1081
## [4,] 5.572 -0.19575 -0.1104
## [5,] 5.620 -0.16599 -0.1097
## [6,] 5.600  0.01594 -0.1121
```

La librería `boot` tiene un método `plot` específico, que muestra el histograma de los valores obtenidos mediante bootstrap y también un gráfico comparándolos con los cuantiles de una normal estándard, figura (4.2)

```
plot(res.boot, index = 3)
```

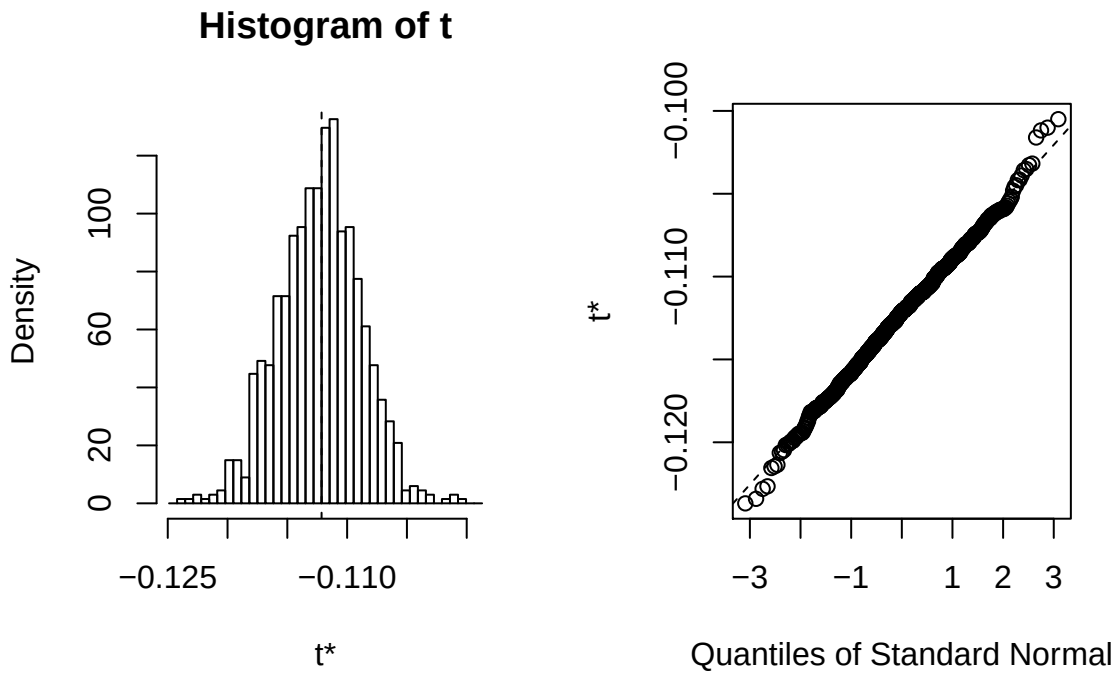


Figura 4.2: Histograma y cuantiles empíricos del coeficiente de la variable *edad* estimado mediante `boot`

Con la función `boot.ci` se pueden generar los intervalos de confianza. Esta función permite calcular los intervalos por varios métodos[3].

```
# Intercept
boot.ci(res.boot, index = 1, type = c("norm", "basic", "perc"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "perc"),
##     index = 1)
##
## Intervals :
## Level      Normal              Basic              Percentile
## 95%   ( 5.339,  6.076 )   ( 5.351,  6.071 )   ( 5.358,  6.079 )
## Calculations and Intervals on Original Scale

# sexoMujer
boot.ci(res.boot, index = 2, type = c("norm", "basic", "perc"))
```

[3]En la ayuda de la función, `help(boot.ci)`, se encuentran referencias sobre los diferentes métodos

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "perc"),
##     index = 2)
##
## Intervals :
## Level      Normal              Basic              Percentile
## 95%   (-0.2903,  0.0739 )   (-0.2848,  0.0785 )   (-0.2865,  0.0768 )
## Calculations and Intervals on Original Scale

# edad
boot.ci(res.boot, index = 3, type = c("norm", "basic", "perc"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res.boot, type = c("norm", "basic", "perc"),
##     index = 3)
##
## Intervals :
## Level      Normal              Basic              Percentile
## 95%   (-0.1187, -0.1053 )   (-0.1183, -0.1049 )   (-0.1194, -0.1060 )
## Calculations and Intervals on Original Scale
```

4.3. Valores ajustados, predicciones del modelo y residuos

4.3.1. Valores ajustados y predicciones

Partiendo de la expresión del modelo

$$p(x_1, \dots x_R) = \frac{\exp\left(\sum_{r=0}^{R} \beta_r x_r\right)}{1 + \exp\left(\sum_{r=0}^{R} \beta_r x_r\right)}$$

con $\alpha = \beta_0$ y $x_0 = 1$, podemos calcular $\hat{p}$ partiendo de las estimaciones de los β

$$\hat{p}(x_1, \dots x_R) = \frac{\exp\left(\sum_{r=0}^{R} \hat{\beta}_r x_r\right)}{1 + \exp\left(\sum_{r=0}^{R} \hat{\beta}_r x_r\right)}$$

En R hay funciones que nos ahorran el cálculo de estos valores. Los valores estimados por el modelo están guardados dentro del objeto `glm`, y se puede acceder a ellos mediante el operador `$`. A estos mismo valores se puede acceder con la función `fitted.values`.

```
head(modelo.edadsex$fitted.values)

##       1       2       3       4       5       6
## 0.47094 0.05158 0.07075 0.14303 0.44312 0.18940

head(fitted.values(modelo.edadsex))

##       1       2       3       4       5       6
## 0.47094 0.05158 0.07075 0.14303 0.44312 0.18940
```

Otra función que se puede utilizar es la función `predict`, que por defecto devuelve la estimación del predictor lineal para cada observación. Se trata de una función genérica que se utiliza para obtener predicciones de varios modelos. Para un objeto de tipo `glm`, la función llama a la función `predict.glm`, cuyos argumentos son

```
args(predict.glm)

## function (object, newdata = NULL, type = c("link", "response",
##     "terms"), se.fit = FALSE, dispersion = NULL, terms = NULL,
##     na.action = na.pass, ...)
## NULL
```

Los más importantes son `newdata`, dónde se indica un nuevo data.frame con las observaciones a predecir con sus respectivos valores en las variables explicativas y `type` dónde se le indica que calcule las predicciones en la escala de la función vínculo (`type="link`"), que es la opción por defecto, en la escala de la variable respuesta (`type="response`"),o la opción `type="terms`" en dónde calcula la predicción para cada término del modelo en la escala de la función vínculo.

Si no se especifica el argumento `newdata` la función calcula las predicciones para los datos utilizados en el ajuste.

```
head(predict(modelo.edadsex))

##       1       2       3       4       5       6
## -0.1164 -2.9117 -2.5753 -1.7903 -0.2285 -1.4539
```

Utilizando `type="response"` obtenemos las probabilidades estimadas.

```
head(predict(modelo.edadsex, type = "response"))

##       1       2       3       4       5       6
## 0.47094 0.05158 0.07075 0.14303 0.44312 0.18940
```

Con el argumento `se.fit=TRUE` se calculan también los valores del error estándar para cada estimación. En este caso, la función `predict` devuelve una lista cuyos elementos son las predicciones y los errores estándar.

```
prob.ajustadas <- predict(modelo.edadsex, type = "response", se.fit = TRUE)
# valores ajustados
head(prob.ajustadas[[1]])

##       1       2       3       4       5       6
## 0.47094 0.05158 0.07075 0.14303 0.44312 0.18940

# errores
head(prob.ajustadas[[2]])

##        1        2        3        4        5        6
## 0.017147 0.005400 0.006685 0.010217 0.017103 0.011776
```

Partiendo de las estimaciones y del error estándar, podemos construir intervalos de confianza para cada uno de los encuestados. Un posible intervalo de confianza sería

$$\hat{p_i} \pm 2 \cdot \widehat{se\,(p_i)}$$

suponiendo que $\hat{p_i}$ se distribuya según una distribución normal, este intervalo tendría un nivel de confianza aproximado del 95 %.

```
# calculamos los intervalos
valores.inf <- prob.ajustadas[[1]] - 2 * prob.ajustadas[[2]]
valores.sup <- prob.ajustadas[[1]] + 2 * prob.ajustadas[[2]]
# vemos los valores inferior y superior para los 6 primeros encuestados
head(valores.inf)

##       1       2       3       4       5       6
## 0.43664 0.04078 0.05738 0.12260 0.40891 0.16585

head(valores.sup)

##       1       2       3       4       5       6
## 0.50523 0.06238 0.08412 0.16347 0.47732 0.21295
```

Para obtener las predicciones del modelo para nuevos datos, indicamos el nuevo data.frame en el argumento `newdata` . El nombre de las variables de este data.frame ha de coincidir con el data.frame original. Construimos un data.frame para predecir la probabilidad de haber usado internet para hombres y mujeres de 45 a 50 años.

```
# Creamos un nuevo data.frame con el mismo nombre para las variables que
# el data.frame original
nuevos.datos <- data.frame(edad = 45:50, sexo = c(rep("Hombre", 6), rep("Mujer",
    6)))
nuevos.datos

##    edad   sexo
```

```
## 1    45 Hombre
## 2    46 Hombre
## 3    47 Hombre
## 4    48 Hombre
## 5    49 Hombre
## 6    50 Hombre
## 7    45  Mujer
## 8    46  Mujer
## 9    47  Mujer
## 10   48  Mujer
## 11   49  Mujer
## 12   50  Mujer
```

```
prob.estim <- predict(modelo.edadsex, newdata = nuevos.datos, type = "response")
cbind(nuevos.datos, prob.estim)
```

```
##    edad   sexo prob.estim
## 1    45 Hombre     0.6612
## 2    46 Hombre     0.6356
## 3    47 Hombre     0.6093
## 4    48 Hombre     0.5823
## 5    49 Hombre     0.5548
## 6    50 Hombre     0.5269
## 7    45  Mujer     0.6375
## 8    46  Mujer     0.6112
## 9    47  Mujer     0.5843
## 10   48  Mujer     0.5568
## 11   49  Mujer     0.5290
## 12   50  Mujer     0.5010
```

4.3.2. Residuos

Enumeramos a continuación los distintos tipos de residuos que podemos obtener para un modelo de regresión logística y cómo los podemos calcular utilizando R. El análisis de los residuos es fundamental para evaluar la adecuación del modelo y detectar los valores anómalos e influyentes. En el capítulo 5 sobre diagnóstico y validación se analiza un ejemplo con mayor profundidad. En este apartados nos limitamos a describir los diferentes tipos de residuos y qué funciones se utilizan para calcularlos.

- *Residuos de la respuesta* . Son simplemente la diferencia entre el valor observado y el valor estimado por el modelo: $y_q - \widehat{\mu}_q$. Estas diferencias se corresponden con los residuos usuales en el modelo lineal. En los modelos GLM, salvo en el caso de tomar la familia gaussiana, no se utilizan ya que ignoran la estructura de la varianza de los errores.
- *Residuos de pearson.* Serían los q-ésimos componentes del estadísticos X^2 de Pearson de bondad

del ajuste global, que veremos en la siguiente sección. Su expresión es:

$$e_{Pq} = \frac{y_q - n_q\widehat{p}_q}{\sqrt{n_q\widehat{p}_q(1-\widehat{p}_q)}}$$

dónde los valores estimados por el modelo en cada perfil (combinación de variables predictoras) son $n_q\widehat{p}_q$.
Estos residuos se pueden calcular fácilmente con R para un modelo `m1` con el comando `residuals(m1, type="pearson")`. Una vez obtenidos los residuos, se realiza el siguiente contraste.

$$\begin{aligned} H_0: \quad e_{Pq} &= 0 \\ H_0: \quad e_{Pq} &\neq 0 \end{aligned}$$

Bajo la hipótesis nula, e_{Pq}, tiene distribución asintótica normal con media 0 y varianza estimada menor que 1. No obstante, se suele considerar la distribución normal estándar, y se consideran significativos aquellos residuos cuyo valor absoluto sea mayor que 2.

```
res.p <- residuals(modelo.edadsex, type = "pearson")
# Residuos de pearson de los 6 primeros individuos de la encuesta
head(res.p)

##       1       2       3       4       5       6
## -0.9435 -0.2332 -0.2759 -0.4085 -0.8920  2.0688
```

- *Residuos de pearson estandarizados*. Se trata de una modificación de los anteriores, de forma que dichos residuos tengan distribuciones asintóticas normales estándar

$$e_{PSq} = \frac{e_{Pq}}{(1-h_q)^{1/2}}$$

dónde h_q es el elemento q-ésimo de la diagonal de la matriz

$$H = W^{\frac{1}{2}}X\left(X^tWX\right)^{-1}X^tW^{\frac{1}{2}}$$

con $W = Diag\left[n_q\widehat{p}_q(1-\widehat{p}_q)\right]$. La matriz H es la equivalente a la matriz *hat* del modelo de regresión lineal múltiple. En R se pueden calcular estos residuos utilizando la función `rstandard` sobre el modelo ajustado y especificando la opción `type="pearson"` .Serán significativos aquellos cuyo valor absoluto sea mayor de 2.

```
res.p.std <- rstandard(modelo.edadsex, type = "pearson")
head(res.p.std)

##       1       2       3       4       5       6
## -0.9440 -0.2333 -0.2760 -0.4087 -0.8926  2.0697
```

- *Residuos de la devianza*. Considerando que la devianza del modelo es.

$$G^2 = \sum_{q=1}^{Q} d_q^2 = \sum_{q=1}^{Q} 2\left[y_q \log\frac{y_q}{\hat{\mu}_q} + (n_q - y_q)\log\frac{n_q - y_q}{n_q - \hat{\mu}_q}\right]$$

los residuos de la devianza son los elementos d_q, tomando como signo el signo de $y_q - \mu_q$ con lo que serían:

$$e_{Dq} = d_q = \text{signo}(y_q - \widehat{\mu}_q)\left(2\left[y_q \log\frac{y_q}{\hat{\mu}_q} + (n_q - y_q)\log\frac{n_q - y_q}{n_q - \hat{\mu}_q}\right]\right)^{1/2}$$

Estos residuos, al igual que los de Pearson, bajo la hipótesis nula que son igual a 0, tienen distribución asintótica de media 0 y varianza estimada menor que 1. Se pueden obtener con la función `residuals` con la opción `type="deviance"`

```
res.d <- residuals(modelo.edadsex, type = "deviance")
head(res.d)

##       1       2       3       4       5       6
## -1.1284 -0.3254 -0.3831 -0.5556 -1.0820  1.8242
```

- *Residuos de la devianza estandarizados* . Al igual que con los residuos de Pearson, podemos estandarizarlos de la siguiente forma

$$e_{DS_q} = \frac{e_{Dq}}{(1 - h_q)^{1/2}}$$

de nuevo, utilizamos la función `rstandard` con la opción `type="deviance"`

```
res.dev.std <- rstandard(modelo.edadsex, type = "deviance")
head(res.dev.std)

##       1       2       3       4       5       6
## -1.1291 -0.3255 -0.3832 -0.5559 -1.0827  1.8250
```

- *Residuos studentizados.* En regresión lineal, el *q-ésimo* residuo studentizado, se calcula como la diferencia entre la respuesta y el valor ajustado, calculado quitando la observación q-ésima del modelo, esta diferencia se estandariza dividiendo por la varianza residual de cada modelo multiplicada por la raíz cuadrada de uno menos los valores hat, así, para la regresión lineal, los residuos studentizados serían

$$e_{Ti}(reg\_lineal) = \frac{e_i}{\hat{S}_{R(i)}\sqrt{1 - h_i}}$$

con e_i la diferencia entre la respuesta y el ajuste del modelo quitando la observación i-ésima. Estos residuos, bajo la hipótesis de normalidad de los residuos estandarizados (en regresión lineal), siguen una distribución t de Student con $n - (k + 1)$ grados de libertad.
En el caso de los modelos lineales generalizados, el cálculo de este tipo de residuos aunque es posible, implicaría un elevado coste computacional, sobre todo para un tamaño muestral elevado. Así se han propuesto aproximaciones como la siguiente (Williams, 1987)

$$e_{Tq} = signo(y_q - \widehat{\mu}_q)\sqrt{(1 - h_q)e^2_{DSq} + h_q e^2_{PSq}}$$

que implica a los residuos estandarizados de Pearson y de la devianza. En R, se utiliza la función `rstudent`

```
res.student <- rstudent(modelo.edadsex)
head(res.student)

##       1       2       3       4       5       6
## -1.1289 -0.3255 -0.3831 -0.5557 -1.0825  1.8253
```

4.4. Medidas de bondad del ajuste

En la práctica, no hay garantía de que un modelo de regresión logística se ajuste bien a los datos. Cuando los datos están en forma binaria, una manera de detectar la falta de ajuste es realizando contrastes condicionales de razón de verosimilitudes entre modelos anidados. Otra aproximación, cuando los datos se pueden agrupar, es considerar las frecuencias esperadas en cada combinación de las variables explicativas y comparar con las frecuencias observadas, mediante el contraste usual con el estadístico X^2 de Pearson o con el G^2 basado en la devianza. Cuando el número de combinaciones o perfiles posibles de las variables explicativas no es elevado y se tienen suficiente número de frecuencias observadas y esperadas en cada combinación; los estadísticos X^2 y G^2 se distribuyen asintóticamente según una chi-cuadrado con grados de libertad igual al número de perfiles distintos menos el número de parámetros en el modelo.

Este tipo de contrastes es adecuado cuando las variables explicativas son categóricas y se puedan considerar los datos agrupados (Agresti, 2002). Además, en el caso de datos no agrupados, el estadístico G^2 sólo depende del modelo ajustado y no provee una medida de la mejora del ajuste respecto del modelo saturado, ver apéndice B

Para solventar este tipo de problemas, se suele recurrir a la comparación entre modelos anidados, dónde se compara nuestro modelo con el que incluya todos los efectos principales e interacciones posibles. Otra opción es utilizar el estadístico de Hosmer-Lemeshow, que realiza una partición de los datos en base a las probabilidades predichas, y analiza posteriormente la tabla de contingencia resultante a través de su estadístico X^2 asociado.

Otras aproximaciones calculan medidas tipo R^2, o analizan el poder predictivo del modelo mediante la tasa de clasificaciones correctas o el análisis de curvas ROC (Franco-Nicolás and Vivo-Molina, 2007).

4.4.1. Contrastes clásicos. Estadísticos G^2 y X^2

La hipótesis que se contrasta, en el caso de un modelo de regresión logística múltiple es:

$$H_0 : p_q = \frac{\exp\left(\sum_{r=0}^{R} \beta_r x_{qr}\right)}{1 + \exp\left(\sum_{r=0}^{R} \beta_r x_{qr}\right)} \; \forall q = 1, \dots Q$$

frente a la hipótesis alternativa

$$H_1 : p_q \neq \frac{\exp\left(\sum_{r=0}^{R} \beta_r x_{qr}\right)}{1 + \exp\left(\sum_{r=0}^{R} \beta_r x_{qr}\right)} \text{ para algún } q$$

Dónde Q es el número de perfiles o combinaciones posibles de las R variables predictoras y p_q las probabilidades a estimar.

Al contrario de lo que es usual en los contrastes de hipótesis, el resultado deseado es que no se pueda rechazar la hipótesis nula de que el modelo se ajusta globalmente bien a los datos.

Para ilustrar estos contrastes vamos a considerar el ajuste del uso de internet utilizando el *sexo* y la *edad* como variables predictoras, pero con los datos agrupados. El data.frame con los datos agrupados por *sexo* y *edad* está guardado en un fichero RData, propio de R y que podemos recuperar con la función `load`.

```
load("../Datos/sexoedad.RData")
# los datos agrupados están en el data.frame res3 primeras 6 filas del
# data.frame
head(res3)

##   edad   sexo Freq uso_int proporcion
## 1   18 Hombre   18      18     1.0000
## 2   18  Mujer   14      13     0.9286
## 3   19 Hombre   16      15     0.9375
## 4   19  Mujer   13      13     1.0000
## 5   20 Hombre   16      15     0.9375
## 6   20  Mujer   20      20     1.0000

# número de filas, que serán los diferentes perfiles
nrow(res3)

## [1] 158
```

Observando el data.frame vemos que hay perfiles que están vacíos, en concreto 4 por lo que el número de perfiles distintos serían 154

```
tail(res3, 10)

##     edad   sexo Freq uso_int proporcion
## 149   92 Hombre    1       0          0
## 150   92  Mujer    2       0          0
## 151   93 Hombre    0      NA          0
## 152   93  Mujer    1       0          0
## 153   94 Hombre    0      NA          0
## 154   94  Mujer    2       0          0
## 155   96 Hombre    0      NA          0
## 156   96  Mujer    2       0          0
## 157  101 Hombre    0      NA          0
## 158  101  Mujer    1       0          0
```

Estadístico G^2 de Wilks de razón de verosimilitudes.
El estadístico tiene la siguiente expresión.

$$G^2(M) = 2\left[L_S - L_M\right]$$

dónde L_M es la log-verosimilitud del modelo ajustado y L_S la log-verosimilitud del modelo saturado. Este estadístico tiene, bajo la hipótesis nula de que el modelo M es adecuado, una distribución asintótica χ^2 con $Q-(R+1)$ grados de libertad, dónde Q es el número de combinaciones de valores de las R variables explicativas .

Ajustamos el modelo con R.

```
modelo.agrup <- glm(proporcion ~ edad + sexo, data = res3, weights = Freq, family = binomial)
summary(modelo.agrup)

##
## Call:
## glm(formula = proporcion ~ edad + sexo, family = binomial, data = res3,
##     weights = Freq)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.668  -0.705  -0.144   0.432   2.765
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.71475    0.19260   29.67   <2e-16 ***
## edad        -0.11214    0.00361  -31.08   <2e-16 ***
## sexoMujer   -0.10400    0.09307   -1.12     0.26
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2043.37  on 153  degrees of freedom
## Residual deviance:  142.01  on 151  degrees of freedom
## AIC: 490.2
##
## Number of Fisher Scoring iterations: 5
```

La salida de la función `summary` devuelve el valor del estadístico G^2 en el apartado Residual Deviance dando un valor de 142.01. No obstante, podemos comprobarlo sumando los cuadrados de los residuos de la devianza.

```
(residuos.deviance <- sum(residuals(modelo.agrup, type = "deviance")^2))

## [1] 142
```

Los grados de libertad asociados serán 154 perfiles distintos menos 3 parámetros ajustados. Nótese que los grados de libertad también se muestran en el resumen del modelo o con.

```
modelo.agrup$df.residual
```

```
## [1] 151
```

No existe una función que calcule directamente el p-valor, pero podemos utilizar la función `pchisq` que devuelve la probabilidad que queda a la izquierda de un valor proveniente de una distribución chi-cuadrado

$$P\left[X \leq x_{observado}\right] = p$$

dónde X es una variable aleatoria con distribución χ^2_{df} y $df = 151$ grados de libertad

El p-valor asociado al estadístico tenemos que calcular

$$\begin{aligned} p - valor &= P\left[X \geq G^2_{observado}\right] \\ &= 1 - P\left[X \leq G^2_{observado}\right] \end{aligned}$$

```
1 - pchisq(residuos.deviance, 151)
```

```
## [1] 0.6877
```

Que al ser mayor de 0.05 (para un nivel de confianza del 95 %), aceptaríamos la hipótesis nula de que el modelo se ajusta globalmente bien a los datos.

Estadístico X^2 de Pearson.

El estadístico tiene la siguiente expresión

$$X^2(M) = \sum_{q=1}^{Q} \frac{(y_q - n_q \widehat{p}_q)^2}{n_q \widehat{p}_q (1 - \widehat{p}_q)}$$

dónde $\widehat{p}_q$ son las probabilidades estimadas en cada combinación de las variables predictoras y n_q el número de casos totales en cada combinación.

Este estadístico tiene, bajo la hipótesis nula de que el modelo es adecuado, la misma distribución asintótica que el estadístico de Wilks.

El estadístico X^2 de Pearson se puede calcular fácilmente partiendo de los residuos de pearson. Y el p-valor lo obtenemos, de nuevo, con la función `pchisq`

```
(residuos.pearson <- sum(residuals(modelo.agrup, type = "pearson")^2))
```

```
## [1] 136
```

```
# p-valor
1 - pchisq(residuos.pearson, 151)
```

```
## [1] 0.8035
```

4.4.2. Contraste basado en el estadístico de Hosmer-Lemeshow

Como ya se ha comentado, el uso de los contrastes basados en los estadísticos X^2 o G^2 no resulta adecuado en el caso de datos no agrupados. Un contraste alternativo es el propuesto por (Hosmer and Lemeshow, 2000), dónde proponen crear grupos de la variable respuesta en base a las probabilidades estimadas por el modelo, y comparar las frecuencias de éxito observadas con las estimadas, mediante el estadístico usual X^2 de Pearson.

Hosmer y Lemeshow proponen crear 10 grupos, en cuyo caso el estadístico asociado sigue asintóticamente una distribución chi-cuadrado con 8 grados de libertad. Para la creación de los grupos hay que elegir los puntos de corte de las probabilidades estimadas. Veamos dos formas de elegir los puntos de corte.

La primera consiste en dividir las probabilidades estimadas en intervalos de igual amplitud. Por ejemplo, para las probabilidades estimadas por el modelo con variables explicativas *edad* y *sexo*, sería

```
yhat <- fitted.values(modelo.edadsex)
yhat.corte <- cut(yhat, breaks = 10, include.lowest = TRUE)
# los intervalos tienen igual amplitud pero el número de individuos en
# cada uno varía
table(yhat.corte)

## yhat.corte
## [0.00231,0.0998]   (0.0998,0.197]    (0.197,0.295]    (0.295,0.392]
##              812              369              234              213
##     (0.392,0.49]     (0.49,0.587]    (0.587,0.684]    (0.684,0.782]
##              170              227              171              341
##    (0.782,0.879]    (0.879,0.977]
##              374              574
```

La función `cut` divide los valores de una variable numérica en intervalos, y con el argumento `breaks` podemos indicar las marcas de clase, mediante un vector o, si sólo ponemos un valor, divide la variable en intervalos de igual amplitud. El resultado de aplicar `cut` a un vector numérico es una variable categórica (factor en R).

Otra alternativa es dividir los datos en base a los cuantiles de la distribución, con lo que se obtienen grupos más homogéneos

```
yhat.corte2 = cut(yhat, breaks = quantile(yhat, probs = seq(0, 1, 1/10)), include.lowest = TRUE)
table(yhat.corte2)

## yhat.corte2
## [0.00328,0.0374]  (0.0374,0.0779]   (0.0779,0.157]     (0.157,0.29]
##              350              351              365              349
##     (0.29,0.471]    (0.471,0.636]    (0.636,0.755]    (0.755,0.844]
##              348              346              356              326
##    (0.844,0.922]    (0.922,0.976]
##              346              348
```

A partir de estos puntos de corte, se construyen las tablas de valores observados y esperados y se calcula el estadístico chi-cuadrado asociado. Al estadístico obtenido mediante el primer método se le denomina *Hosmer-Lemeshow H statistic* y al obtenido por el segundo *Hosmer-Lemeshow C statistic.* En el artículo original se prefiere el segundo método frente al primero.

Para obtener el segundo estadístico se construyen las tablas de las frecuencias esperadas y observadas de la variable de respuesta.

La siguiente tabla es la de los valores observados, dónde las columnas V1 e y son respectivamente el número de individuos en cada grupo que tienen valor *uso_int=0* y *uso_int=1*

```
y <- datos.bin$uso_int
(obs <- xtabs(cbind(1 - y, y) ~ yhat.corte2))
```

```
##
## yhat.corte2          V1   y
##   [0.00328,0.0374] 348   2
##   (0.0374,0.0779]  332  19
##   (0.0779,0.157]   323  42
##   (0.157,0.29]     267  82
##   (0.29,0.471]     202 146
##   (0.471,0.636]    155 191
##   (0.636,0.755]    102 254
##   (0.755,0.844]     65 261
##   (0.844,0.922]     39 307
##   (0.922,0.976]     21 327
```

La tabla de valores esperados se construye de forma similar

```
(expect <- xtabs(cbind(1 - yhat, yhat) ~ yhat.corte2))
```

```
##
## yhat.corte2             V1    yhat
##   [0.00328,0.0374] 341.497   8.503
##   (0.0374,0.0779]  331.257  19.743
##   (0.0779,0.157]   321.443  43.557
##   (0.157,0.29]     269.282  79.718
##   (0.29,0.471]     213.785 134.215
##   (0.471,0.636]    154.295 191.705
##   (0.636,0.755]    104.818 251.182
##   (0.755,0.844]     62.746 263.254
##   (0.844,0.922]     38.824 307.176
##   (0.922,0.976]     16.053 331.947
```

El estadístico de contraste y p-valor asociado sería

```
(chisq <- sum((obs - expect)^2/expect))
```

```
## [1] 8.772
```

```
(P <- 1 - pchisq(chisq, 8))

## [1] 0.3619
```

Podemos escribir una función[4] que calcule el estadístico de Hosmer-Lemeshow para ambos casos

```
hosmerlem <- function(y, yhat, g = 10) {
    cutyhat1 = cut(yhat, breaks = quantile(yhat, probs = seq(0, 1, 1/g)), include.lowest = TRUE)
    obs = xtabs(cbind(1 - y, y) ~ cutyhat1)
    expect = xtabs(cbind(1 - yhat, yhat) ~ cutyhat1)
    chisq.C = sum((obs - expect)^2/expect)
    P.C = 1 - pchisq(chisq.C, g - 2)
    cutyhat2 = cut(yhat, breaks = g, include.lowest = TRUE)
    obs = xtabs(cbind(1 - y, y) ~ cutyhat2)
    expect = xtabs(cbind(1 - yhat, yhat) ~ cutyhat2)
    chisq.H = sum((obs - expect)^2/expect)
    P.H = 1 - pchisq(chisq.H, g - 2)
    res <- data.frame(c(chisq.C, P.C), c(chisq.H, P.H))
    colnames(res) <- c("Hosmer-Lemeshow C statistic", "Hosmer-Lemeshow H statistic")
    rownames(res) <- c("X-squared", "p.value")
    return(res)
}
```

Si consideramos el modelo que ajustaba el modelo con edad y sexo sobre los 3.485 individuos de la encuesta se tiene que

```
hosmerlem(datos.bin$uso_int, fitted.values(modelo.edadsex))

##           Hosmer-Lemeshow C statistic Hosmer-Lemeshow H statistic
## X-squared                      8.7720                      6.8225
## p.value                        0.3619                      0.5559
```

Y se aceptaría la hipótesis nula de que el modelo se ajusta globalmente a los datos.

Concluyendo, si tenemos los datos de forma que estén agrupados o que se puedan agrupar fácilmente y que además haya un número suficiente de casos en cada combinación de las variables predictoras, se pueden utilizar los contrastes basados en el estadístico de Wilks G^2 o el de Pearson X^2. Si los datos no están en ese formato, o no se pueden agrupar, hay que utilizar el contraste de Hosmer-Lemeshow o ver medidas alternativas como las medidas tipo R^2 o comparando modelos anidados mediante el test condicional de razón de verosimilitudes.

[4] Función extraída de la lista de correo R-help, aunque se ha modificado para que calcule las dos alternativas que proponen Hosmer y Lemeshow

4.4.3. Medidas tipo R^2

En la regresión lineal por mínimos cuadrados tenemos una medida R^2, que nos ofrece una medida de la bondad del ajuste comparando sumas de cuadrados.

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y}_i)^2}$$

Podríamos considerar esta medida en el caso de la regresión logística sustituyendo $\hat{y}_i$ por $n_i\hat{p}_i$, pero esta medida presenta varios inconvenientes, como que los estimadores máximo verosímiles no son los estimadores que maximizan esta medida o que no tiene en cuenta la dependencia de la varianza de Y respecto de p. Para solventar estos problemas se han propuesto medidas alternativas, denominadas pseudo R^2, las principales son los pseudo R^2de McFadden, Cox-Snell y Nagelkerke.

Pseudo R^2 de McFadden (McFadden, 1974) Esta medida compara la log-verosimilud del modelo ajustado con la log-verosimilud del modelo que sólo tiene el término constante.

$$R^2 = 1 - \frac{\hat{L}_{Modelo}}{\hat{L}_{intercept}}$$

asemejándose a la expresión del R^2 en la estimación por mínimos cuadrados, dónde $\hat{L}_{intercept}$ sería la suma de cuadrados totales y $\hat{L}_{Modelo}$ haría el papel de la suma de cuadrado de los errores. Este R^2 da una idea de en cuánto se reduce la devianza de los datos al ajustar el modelo.

En R se puede calcular fácilmente a partir de los valores guardados en el objeto glm

```
# var.explicativas, sexo y edad, modelo con 3485 datos
(RsqrMcFadden <- 1 - modelo.edadsex$deviance/modelo.edadsex$null.deviance)

## [1] 0.3947
```

Para el modelo equivalente pero ajustado sobre los datos agrupados se tiene

```
# var.explicativas, sexo y edad, modelo datos agrupados
(RsqrMcFadden.agrup <- 1 - modelo.agrup$deviance/modelo.agrup$null.deviance)

## [1] 0.9305
```

Que indica que, una vez más, es importante tener en cuenta cómo se han ajustado los datos y que, aunque el modelo sea el mismo, el ajuste es mucho mejor para los 154 *perfiles* de sexo y edad que para los 3.485 individuos.

Pseudo R^2 de Cox y Snell (Cox and Snell, 1989) R^2_{CN} tiene la siguiente expresión

$$R^2_{CN} = 1 - \left(\frac{V_{intercept}}{V_{Modelo}}\right)^{\frac{2}{N}}$$

dónde V es la máxima verosimilitud y N el número de perfiles ajustados. En términos de la log-verosimilitud se tiene que

$$R^2_{CN} = 1 - \exp\left(-\frac{\widehat{LR}}{N}\right)$$

dónde $\widehat{LR} = \hat{L}_{intercept} - \hat{L}_{Modelo}$

En R sería

```
LR <- modelo.edadsex$null.deviance - modelo.edadsex$deviance
N <- sum(weights(modelo.edadsex))  # 3485 en este caso
(RsqrCN <- 1 - exp(-LR/N))

## [1] 0.4205
```

Para el modelo con datos agrupados sale el mismo valor, debido a que la diferencia de devianzas es igual en los dos casos.

```
LR <- modelo.agrup$null.deviance - modelo.agrup$deviance
N <- sum(weights(modelo.agrup))
(RsqrCN.agrup <- 1 - exp(-LR/N))

## [1] 0.4205
```

Pseudo R^2de Nagelkerke(Nagelkerke, 1991) Se trata de una modificación del R^2_{CN} que alcanza su máximo en

$$\text{máx } R^2_{CN} = 1 - V_{intercept}^{\frac{2}{N}}$$

El R^2_N se define como

$$R^2_N = \frac{R^2_{CN}}{\text{máx } R^2_{CN}}$$

el valor máx R^2_{CN} se puede expresar en términos de la devianza como

$$\text{máx } R^2_{CN} = 1 - \exp\left(-\frac{\hat{L}_{intercept}}{N}\right)$$

El cálculo en R sería

```
L0.adj <- exp(-modelo.edadsex$null.deviance/N)
(RsqrNal <- RsqrCN/(1 - L0.adj))

## [1] 0.5614
```

4.4.4. Medidas basadas en la tabla de clasificación. Curvas ROC

Otra forma de evaluar el desempeño de un modelo de clasificación es a través de medidas relacionadas con la tabla de clasificación. Consideremos la siguiente tabla, en la que se muestra la estructura de

una tabla de clasificación, dónde en las filas se pone el estado real en la muestra y en las columnas la categoría predicha por el modelo.

	Clasificación: éxito	Clasificación: fracaso
Éxito	Verdaderos Positivos (VP)	Falsos Negativos (FN)
Fracaso	Falsos Positivos (FP)	Verdaderos negativos (VN)

Para determinar cómo clasifica el modelo a cada observación se elige un punto de corte, si la probabilidad predicha por el modelo es mayor que el punto de corte se clasifica como éxito y si es menor como fracaso. Evidentemente, los valores VP, FN, FP y VN variarán según el punto de corte elegido.

Para los datos del uso de internet se tiene que.

```
table(datos.bin$uso_int)

##
##    0    1
## 1854 1631
```

Considerando el modelo con *sexo* y *edad* y eligiendo 0.5 como punto de corte, obtenemos la categoría estimada para cada individuo. Utilizamos la función `fitted.values` para obtener las probabilidades e `ifelse` para asignar 1 a los individuos cuya probabilidad estimada sea mayor o igual que 0.5

```
prediccion <- ifelse(fitted.values(modelo.edadsex) >= 0.5, 1, 0)
table(prediccion)

## prediccion
##    0    1
## 1828 1657
```

Y la tabla de clasificación sería

```
table(datos.bin$uso_int, prediccion)

##    prediccion
##        0    1
##   0 1500  354
##   1  328 1303
```

Asociado a este punto de corte podemos calcular la tasa de clasificaciones correctas como los individuos correctamente clasificados entre el número total de individuos.

```
tabla.clasif <- table(datos.bin$uso_int, prediccion)
tcc <- 100 * sum(diag(tabla.clasif))/sum(tabla.clasif)
tcc

## [1] 80.43
```

Si elegimos como punto de corte $p = 0{,}5$ el modelo clasifica correctamente al 80.43 % de los individuos.

La librería ROCR (Sing, Sander, Beerenwinkel, and Lengauer, 2005) nos permite analizar varias medidas relacionadas con la tabla de clasificación, y representarlas gráficamente. Nos puede ayudar a elegir el punto de corte que maximiza la tasa de clasificaciones correctas, y ver como varía esta tasa según los diferentes puntos de corte.

```
library(ROCR)
```

La función `prediction` calcula los valores de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos para diferentes puntos de corte. Requiere como argumentos las probabilidades estimadas y las etiquetas de la variable respuesta en los datos (en este caso ceros y unos)

```
pred <- prediction(fitted.values(modelo.edadsex), datos.bin$uso_int)
```

Con la función `performance`, calculamos varias medidas asociadas a la tabla de clasificación. Para obtener la tasa de clasificaciones correctas especificamos el argumento `measure="acc".` En la ayuda de la función llaman a la tasa de clasificaciones correctas *Accuracy* y que el argumento a utilizar para calcularla es `measure="acc"`

```
perf1 <- performance(pred, measure = "acc")
# el punto de corte que maximiza 'acc' es
(posicion.max <- sapply(perf1@y.values, which.max))

## [1] 69

(punto.corte <- sapply(perf1@x.values, "[", posicion.max))

##  3443
## 0.473
```

Los valores de la *tcc* los obtenemos tecleando `perf@y.values`[5]

Con la función plot se obtiene una representación gráfica de la tasa de clasificaciones correctas para los distintos puntos de corte, figura (4.3)

[5] El uso de `@` en vez de `$` se debe a que la librería ROCR trabaja con las clases S4 en vez de las S3. Una descripción detallada de ambas estructuras se puede encontrar en el manual R Internals en http://www.r-project.org/

```
plot(perf1, col = "darkred")
# Añadimos una línea horizontal al valor de 0.8
abline(h = 0.8, lty = 2)
# Añadimos recta con el punto de corte que maximiza la tasa de
# clasificaciones correctas
abline(v = punto.corte, lty = 2)
```

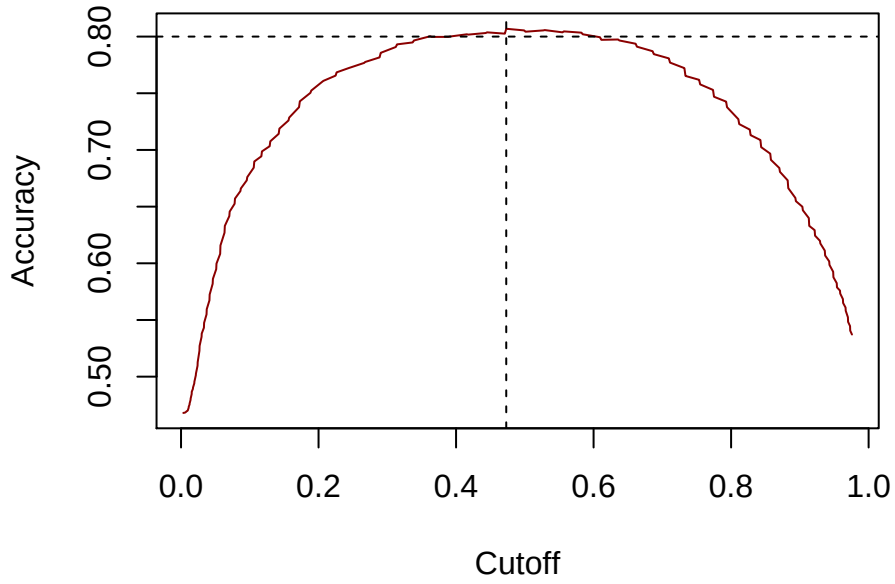


Figura 4.3: Tasa de clasificaciones correctas, para distintos puntos de corte

Otra forma de evaluar un modelo es a través de la curva ROC. La curva ROC representa en un gráfico la fracción de falsos positivos definida como FP/(FP + VN), frente a la fracción de verdaderos positivos definida como VP/ (VP +FN). Así, se puede evaluar un modelo según si el interés sea maximizar la fracción de verdaderos positivos, minimizar la fracción de falsos positivos u obtener la mejor tasa de clasificación correcta.

Si consideramos en el `modelo.edadsex` como positivos los individuos con *uso_int=1,* y un punto de corte igual a 0.5 tenemos los siguientes valores.

Fracción de falsos positivos= $\frac{354}{1500+354} = 0{,}19094$

Fracción de verdaderos positivos= $\frac{1303}{1303+328} = 0{,}79890$

Con la curva ROC se representan estos valores para varios puntos de corte, figura (4.4)

Se considera que un modelo es mejor que otro si la curva ROC se acerca al borde superior izquierdo, o lo que es lo mismo, que el área bajo la curva sea mayor (Franco-Nicolás and Vivo-Molina, 2007). Podemos calcular el área bajo la curva utilizando de nuevo la función `performance` sobre el objeto pred.

```
# auc : Area under curve
AUC <- performance(pred, "auc")
AUC@y.name

## [1] "Area under the ROC curve"

AUC@y.values

## [[1]]
## [1] 0.8889
```

```
# con performance se selecciona tpr (true positive rate) y fpr (false
# positive rate)
perf2 <- performance(pred, "tpr", "fpr")
plot(perf2, colorize = TRUE)  # mostramos colores según el punto de corte
# Añadimos la recta y=x que sería la correspondiente al peor clasificador
abline(a = 0, b = 1)
# añadimos el valor del área bajo la curva
text(0.4, 0.6, paste(AUC@y.name, "\n", round(unlist(AUC@y.values), 3)), cex = 0.7)
```

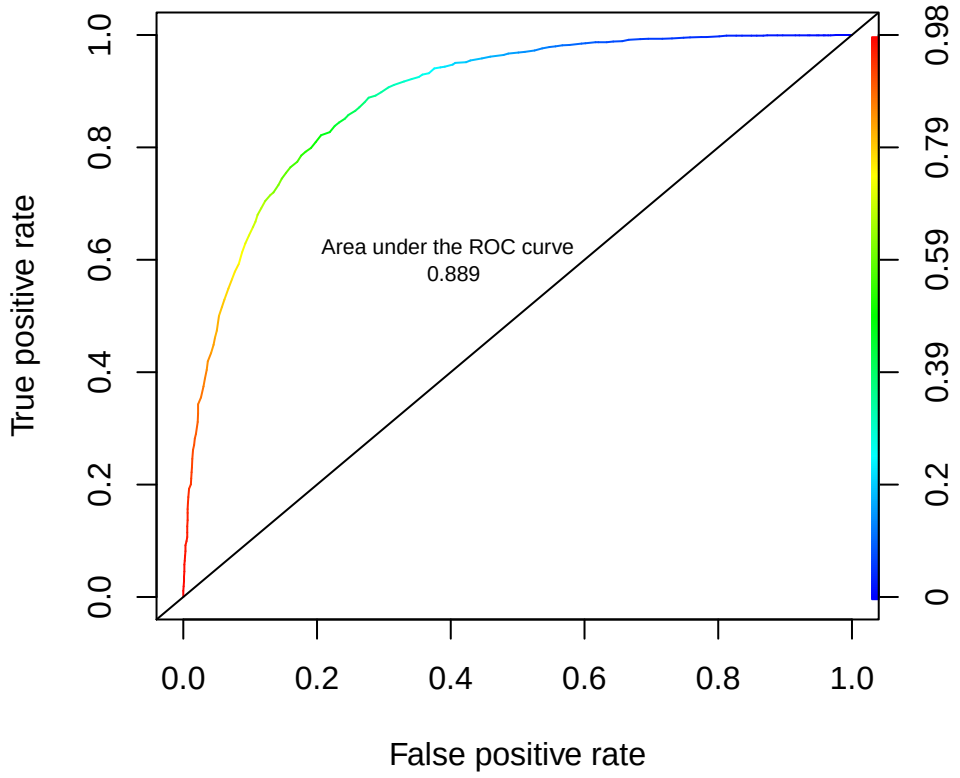


Figura 4.4: Curva ROC para el modelo con *sexo* y *edad*.

Podemos representar en un mismo gráfico las curvas ROC de diferentes modelos, lo que permite una

comparación rápida de la eficacia de cada modelo, figura (4.5)

```
pred.modelo.sex <- prediction(fitted.values(modelo.sex), datos.bin$uso_int)
perf.modelo.sex <- performance(pred.modelo.sex, "tpr", "fpr")
plot(perf2, col = "darkred")
# Añadimos la recta y=x que sería la correspondiente al peor clasificador
abline(a = 0, b = 1)
# añadimos la curva ROC para el modelo.sex utilizando add=TRUE
plot(perf.modelo.sex, col = "darkblue", lty = 2, add = TRUE)
legend("bottomright", c("Reg.logist uso_int ~ sexo+edad", "Reg.logist uso_int ~ sexo"),
    col = c("darkred", "darkblue"), lty = 1:2, cex = 0.7)
```

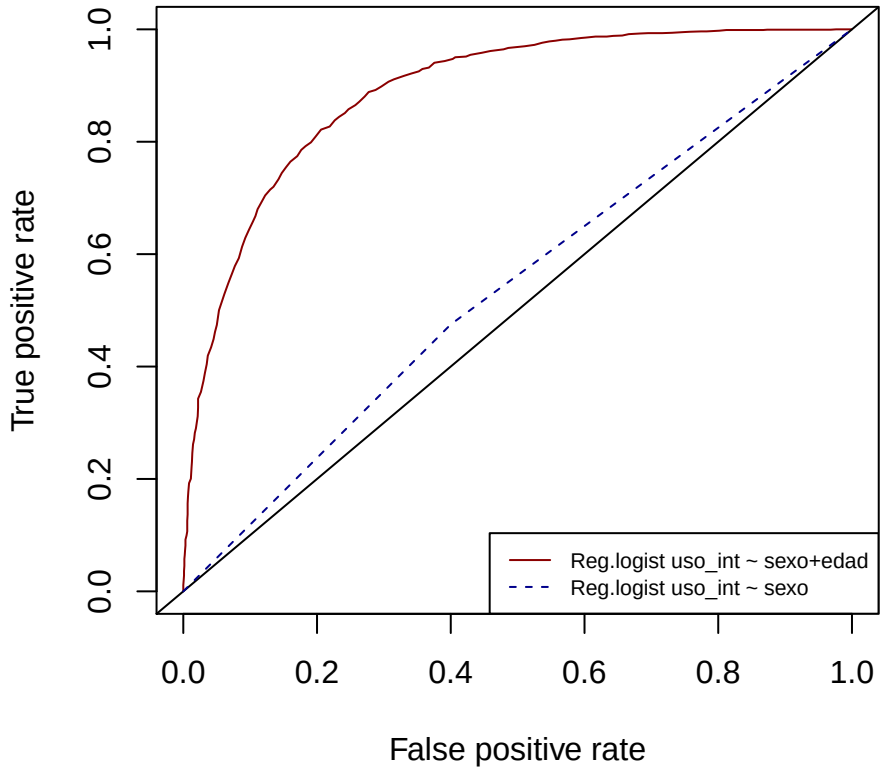


Figura 4.5: Comparación curvas ROC.

La comparación de las curvas ROC indica que el modelo que añade la edad como variable explicativa es mucho mejor que el modelo que sólo consideraba el sexo.

4.5. Métodos de selección de variables

La selección automática de variables se basa en la comparación de la devianza entre modelos. El procedimiento habitual es el llamado *stepwise* (paso a paso) en el que, mediante contrastes condicionales de razón de verosimilitudes se comparan modelos con diferentes variables. La selección *forward* (hacia adelante) empieza con el modelo más simple y va añadiendo en cada paso la variable más significativa

según el contraste condicional de razón de verosimilitudes. El proceso se para cuando se llega al modelo con el mayor número de variables e interacciones posibles, o cuando ningún modelo mejora al actual. Otra forma de proceder es partiendo del modelo con más variables e interacciones, e ir eliminando variables, tomando otra vez el criterio de los contrastes de razón de verosimilitudes. También se pueden mezclar ambos procedimientos, de forma que en cada paso, se contrasta si entra una variable nueva o si sale una que ya esté en el modelo. Para eso se fija un nivel de significación α_1 para el contraste con los modelos que añaden una variable y un nivel de significación α_2 para la eliminación de variables, con $\alpha_2 > \alpha_1$. En cada paso se realizan varios contrastes, tanto de inclusión de variables como de eliminación, y el proceso continua hasta que los contrastes dejen de ser significativos, es decir, que no se incluyan más variables, ni se elimine ninguna de las que entraron.

La selección automática de modelos no está exenta de polémica, ya que según sea el método utilizado , "forward", "backward" o "stepwise" puede dar lugar a modelos totalmente diferentes. El empleo de estos métodos nunca debe sustituir al criterio del investigador (Silva and Barroso, 2004), ni considerar sin más a las variables que se "quedan" en el modelo, como principales responsables del efecto en la variable dependiente.

La librería `MASS` de R incorpora la función `stepAIC` que implementa la selección automática de variables, pero en vez de utilizar los contrastes condicionales de razones de verosimilitudes, utiliza el criterio de información de Akaike.

Este criterio se define en base a la log-verosimilitud y al número de parámetros del modelo, el valor del AIC se calcula como

$$AIC = -2 \cdot L_{Modelo} + k \cdot \text{número parámetros estimados}$$

con $k = 2$ y L la log-verosimilitud.

Este criterio penaliza los modelos con muchos parámetros frente a los modelos más parsimoniosos. Dentro de la función `stepAIC`, se puede variar el criterio de penalización elgiendo un valor diferente para k . Con $k = \log(N)$ se obtiene un AIC modificado al que se le denomina BIC[6] (Venables and Ripley, 2002). La función `stepAIC` extrae el AIC asociado a los diferentes modelos que resultan de añadir o quitar del modelo actual cada una de las variables, y se elige el modelo con menor valor de AIC. El proceso continúa hasta que la inclusión o eliminación de alguna variable incremente el valor del AIC en vez de reducirlo.

Ejemplo de selección automática de variables En el ejemplo sobre el uso de internet en Andalucía, realizamos la selección automática del modelo mediante un procedimiento stepwise basado en el Criterio de Información de Akaike. Como modelo máximo tomamos el modelo que contiene las interacciones de orden 4 entre sexo, edad, habitat y nivelest, así como todas las interacciones de orden inferior y los efectos principales. Como modelo inicial tomamos el que sólo tiene el término constante.

```
modelo.full <- glm(uso_int ~ edad * sexo * nivelest * habitat, data = datos.bin,
    family = binomial)
modelo.inicial <- glm(uso_int ~ 1, data = datos.bin, family = binomial)
```

Utilizamos la función stepAIC, dónde en el argumento `scope` indicamos el modelo más complejo y con el argumento `direction="both"` indicamos que el procedimiento sea stepwise. La función stepAIC

[6] Criterio de información bayesiano, por sus siglas en inglés

comprueba en cada paso como varía el AIC con la inclusión o supresión de variables. El procedimiento termina cuando la inclusión o supresión de variables no disminuye el valor del AIC.

```
modelo.stp <- stepAIC(modelo.inicial, scope = list(upper = modelo.full), direction = "both")
```

```
## Start:  AIC=4819
## uso_int ~ 1
##
##            Df Deviance  AIC
## + edad      1     2917 2921
## + nivelest  3     3379 3387
## + habitat   6     4752 4766
## + sexo      1     4797 4801
## <none>              4817 4819
##
## Step:  AIC=2921
## uso_int ~ edad
##
##            Df Deviance  AIC
## + nivelest  3     2222 2232
## + habitat   6     2816 2832
## <none>              2917 2921
## + sexo      1     2916 2922
## - edad      1     4817 4819
##
## Step:  AIC=2232
## uso_int ~ edad + nivelest
##
##                 Df Deviance  AIC
## + habitat        6     2203 2225
## + edad:nivelest  3     2213 2229
## <none>                 2222 2232
## + sexo           1     2220 2232
## - nivelest       3     2917 2921
## - edad           1     3379 3387
##
## Step:  AIC=2225
## uso_int ~ edad + nivelest + habitat
##
##                    Df Deviance  AIC
## + edad:nivelest     3     2195 2223
## <none>                    2203 2225
## + sexo              1     2202 2226
## + edad:habitat      6     2193 2227
## - habitat           6     2222 2232
## + nivelest:habitat 18     2193 2251
## - nivelest          3     2816 2832
```

```
## - edad              1     3355 3375
##
## Step:  AIC=2223
## uso_int ~ edad + nivelest + habitat + edad:nivelest
##
##                    Df Deviance  AIC
## <none>                    2195 2223
## + sexo              1     2193 2223
## - edad:nivelest     3     2203 2225
## + edad:habitat      6     2186 2226
## - habitat           6     2213 2229
## + nivelest:habitat 18     2183 2247
```

Obteniendo un modelo que no incluye el sexo, pero si el habitat, el nivel de estudios y la interacción entre edad y el nivel de estudios.

Con `summary` obtenemos el resumen del modelo ajustado

```
summary(modelo.stp)
```

```
##
## Call:
## glm(formula = uso_int ~ edad + nivelest + habitat + edad:nivelest,
##     family = binomial, data = datos.bin)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.981  -0.420  -0.121   0.408   3.170
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           -2.7311     2.8417   -0.96   0.3365
## edad                                  -0.0286     0.0406   -0.70   0.4812
## nivelestPrimaria                       7.1657     2.8506    2.51   0.0119 *
## nivelestSecundaria y F.P               9.4472     2.8939    3.26   0.0011 **
## nivelestUniversitaria o superior      12.5347     3.0655    4.09  4.3e-05 ***
## habitatestrato0                        0.5936     0.2076    2.86   0.0042 **
## habitatestrato1                        0.4995     0.1834    2.72   0.0064 **
## habitatestrato2                        0.7975     0.2829    2.82   0.0048 **
## habitatestrato3                        0.4462     0.2014    2.22   0.0267 *
## habitatestrato4                        0.2185     0.1863    1.17   0.2410
## habitatestrato5                        0.2331     0.1592    1.46   0.1431
## edad:nivelestPrimaria                 -0.0796     0.0409   -1.94   0.0518 .
## edad:nivelestSecundaria y F.P         -0.0802     0.0419   -1.91   0.0556 .
## edad:nivelestUniversitaria o superior -0.1133     0.0447   -2.54   0.0112 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4817.0  on 3484  degrees of freedom
## Residual deviance: 2194.8  on 3471  degrees of freedom
## AIC: 2223
##
## Number of Fisher Scoring iterations: 7
```

Para evaluar el ajuste global, podemos calcular el contraste de Hosmer-Lemeshow, utilizando la función `hosmerlem` definida anteriormente

```
hosmerlem(y = datos.bin$uso_int, yhat = fitted.values(modelo.stp))
```

```
##           Hosmer-Lemeshow C statistic Hosmer-Lemeshow H statistic
## X-squared                      3.7602                      9.6383
## p.value                        0.8781                      0.2913
```

Que da un p-valor cercano a 0.9 para el estadístico C, con lo que se aceptaría la hipótesis nula de que el modelo se ajusta globalmente bien a los datos.

Otra forma de evaluar el modelo, es mediante el análisis de la curva ROC, comparándola con las curvas ROC de los modelos anteriores.

```
pred.modelo.stp <- prediction(fitted.values(modelo.stp), datos.bin$uso_int)
perf.modelo.stp <- performance(pred.modelo.stp, "tpr", "fpr")
plot(perf.modelo.stp, col = "darkred")  # modelo.stp (edad+habitat+nivelest+edad:nivelest)
plot(perf2, col = "darkblue", lty = 2, add = TRUE)  # modelo.edadsex (edad+sexo)
plot(perf.modelo.sex, lty = 3, add = TRUE)  # modelo.sex (sexo)
abline(a = 0, b = 1)
legend("bottomright", c("R.L uso_int ~ habitat + edad*nivelest", "R.L uso_int ~ sexo + edad",
    "R.L uso_int ~ sexo"), col = c("darkred", "darkblue", "black"), lty = 1:3, cex = 0.7)
```

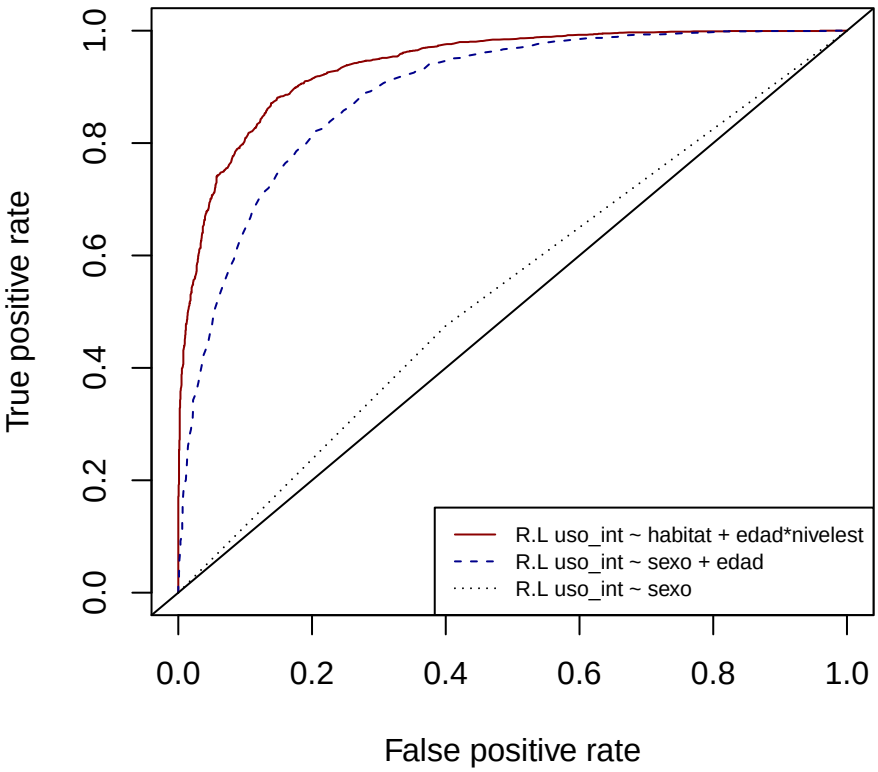


Figura 4.6: Comparación curvas ROC.

La comparación de las curvas ROC muestra que el modelo obtenido por el procedimiento stepwise tiene una mayor capacidad predictiva que el modelo que contenía sólo la variable edad. El área bajo la curva ROC sería

```
AUC.modelo.stp <- performance(pred.modelo.stp, "auc")
AUC.modelo.stp@y.values

## [[1]]
## [1] 0.9387
```

La tabla de clasificación asociada al modelo.stp, para un punto de corte de 0.5 sería

```
predicho.modelo.stp <- ifelse(fitted.values(modelo.stp) >= 0.5, 1, 0)
(tabla.clas.modelo.stp <- table(datos.bin$uso_int, predicho.modelo.stp))

##    predicho.modelo.stp
##        0    1
##   0 1620  234
##   1  249 1382
```

Y la tasa de clasificaciones correctas sería del 86 %, frente al 80 % de clasificaciones correctas del modelo con *edad* y *sexo.*

Para finalizar, podemos calcular los intervalos de confianza para los parámetros.

```
cbind(coef = coef(modelo.stp), confint(modelo.stp))

## Waiting for profiling to be done...

##                                        coef     2.5 %    97.5 %
## (Intercept)                         -2.73105 -10.00545  1.718633
## edad                                -0.02862  -0.09967  0.067359
## nivelestPrimaria                     7.16566   2.69045 14.450649
## nivelestSecundaria y F.P             9.44725   4.85107 16.786084
## nivelestUniversitaria o superior    12.53465   7.51491 20.101392
## habitatestrato0                      0.59364   0.18702  1.000994
## habitatestrato1                      0.49948   0.14023  0.859308
## habitatestrato2                      0.79750   0.24897  1.358648
## habitatestrato3                      0.44620   0.05174  0.841665
## habitatestrato4                      0.21847  -0.14696  0.583828
## habitatestrato5                      0.23310  -0.07885  0.545464
## edad:nivelestPrimaria               -0.07958  -0.17596 -0.007873
## edad:nivelestSecundaria y F.P       -0.08023  -0.17803 -0.006312
## edad:nivelestUniversitaria o superior -0.11330  -0.21532 -0.033710
```

La selección por pasos, ya sea mediante el contraste condicional de razón de verosimilitud entre modelo anidados, o mediante el uso de criterios de información como el de Akaike, son los métodos de selección de modelos más utilizados actualmente. No obstante existen otros métodos de selección de variables como la selección de los mejores subconjuntos de modelos posibles (Hosmer, Jovanovic, and Lemeshow, 1989), o basados en la penalización del tamaño de los parámetros estimados como el método *lasso* (Tibshirani, 1996). Recientemente se han publicado paquetes en R que realizan este tipo de selección de modelos, como los paquete `MuMIn` (Bartoń, 2013) o `bestglm` (McLeod and Changjiang, 2011), o para la selección vía *lasso,* el paquete `glmnet` (Friedman, Hastie, and Tibshirani, 2010). Tanto el paquete `bestglm` como el paquete `glmnet` están todavía en un fase prematura y no incorporan la capacidad de introducir el modelo mediante una fórmula. En el caso del paquete `bestglm` se añade el problema de no poder introducir las interacciones, problema que también se da en el paquete `glmnet` dónde además es complicado incorporar variables predictoras categóricas.

Vamos a utilizar el paquete `MuMIn` y dentro de este la función `dredge` que, partiendo de un modelo ajusta todos los posibles modelos con menos parámetros y calcula el criterio de información de Akaike

de segundo orden AICc (Burnham and Anderson, 2002). El AICc es una modificación del AIC y se define como.

$$AICc = AIC + \frac{2k\,(k+1)}{n-k-1}$$

dónde n es el número de observaciones y k el número de parámetros estimados. Este criterio penaliza aún más que el AIC los modelos con mayor número de parámetros. (Burnham and Anderson, 2002) recomiendan encarecidamente utilizar este critero cuando n es pequeño o k es elevado, concretamente cuando $n/k < 40$.

Como modelo de partida vamos a tomar el modelo con todas las posibles interacciones y los efectos principales. Es decir, el modelo que hemos llamado `modelo.full` en la selección mediante `stepAIC`. Este modelo tiene 112 parámetros y se tiene que 3485/112=31.1.

```
library(MuMIn)
allModels <- dredge(modelo.full)
```

El objeto `allModels` es un objeto que pertenece a dos clases , la clase `model.selection` propia de la librería `MuMIn` y la clase `data.frame`.

```
class(allModels)

## [1] "model.selection" "data.frame"
```

En el objeto se han guardado los coeficientes estimados, o presencia /ausencia (indicado por el símbolo +) para cada modelo. También nos devuelve los grados de libertad, la log-verosimilitud, el valor del criterio de información y la diferencia entre el valor del AICc para el modelo con menor AICc y el resto. El data.frame está ordenado de forma creciente por el valor del AICc.

Tenemos un total de 167 modelos ajustados, los 5 modelos con menor AICc son.

```
allModels[1:5, ]

## Global model call: glm(formula = uso_int ~ edad * sexo * nivelest * habitat, family = binomial,
##     data = datos.bin)
## ---
## Model selection table
##     (Int)  edd      hbt nvl sex edd:nvl edd:sex df logLik AICc delta weight
## 40  -2.731 -0.02862 +   +       +               14 -1097  2223 0.00  0.332
## 48  -2.671 -0.02803 +   +   +   +               15 -1096  2223 0.04  0.324
## 112 -2.890 -0.02344 +   +   +   +       +       16 -1096  2224 1.22  0.180
## 8    2.024 -0.10930 +   +                       11 -1102  2226 2.63  0.089
## 16   2.092 -0.10900 +   +   +                   12 -1101  2226 2.98  0.075
```

El mejor modelo es el que tiene los efectos principales de edad, hábitat y nivel de estudios más la interacción entre edad y nivel de estudios, que coincide con el modelo ajustado mediante `stepAIC`.

La última columna se corresponde con la importancia relativa (según el valor del AICc) de cada modelo con respecto al total. Si seleccionamos sólo los 3 primeros modelos la importancia relativa varía.

```
allModels[1:3, ]

## Global model call: glm(formula = uso_int ~ edad * sexo * nivelest * habitat, family = binomial,
##     data = datos.bin)
## ---
## Model selection table
##     (Int)  edd      hbt nvl sex edd:nvl edd:sex df logLik AICc delta weight
## 40  -2.731 -0.02862 +   +       +               14 -1097  2223 0.00  0.396
## 48  -2.671 -0.02803 +   +   +   +               15 -1096  2223 0.04  0.388
## 112 -2.890 -0.02344 +   +   +   +       +       16 -1096  2224 1.22  0.216
```

A partir de la importancia de cada modelo, la función `importance` calcula la importancia relativa de cada variable, de forma que las variables que están en los modelos con menor AICc tienen mayor importancia.

```
# round redondea los resultados
round(importance(allModels), 3)

##                         edad                    nivelest
##                        1.000                       1.000
##                      habitat                edad:nivelest
##                        0.976                       0.774
##                         sexo                    edad:sexo
##                        0.651                       0.224
##                 edad:habitat                 habitat:sexo
##                        0.185                       0.103
##                nivelest:sexo            edad:habitat:sexo
##                        0.062                       0.002
##           edad:nivelest:sexo             habitat:nivelest
##                        0.001                       0.000
##        edad:habitat:nivelest        habitat:nivelest:sexo
##                        0.000                       0.000
## edad:habitat:nivelest:sexo
##                        0.000
```

En la figura (4.7) representamos las variables según su importancia.

```
plot(importance(allModels), axes = FALSE, pch = 19, type = "b", xlab = "", ylab = "Importancia variab
axis(2, las = 2)
axis(1, at = 1:15, labels = names(importance(allModels)), las = 2, cex.axis = 0.7)
```

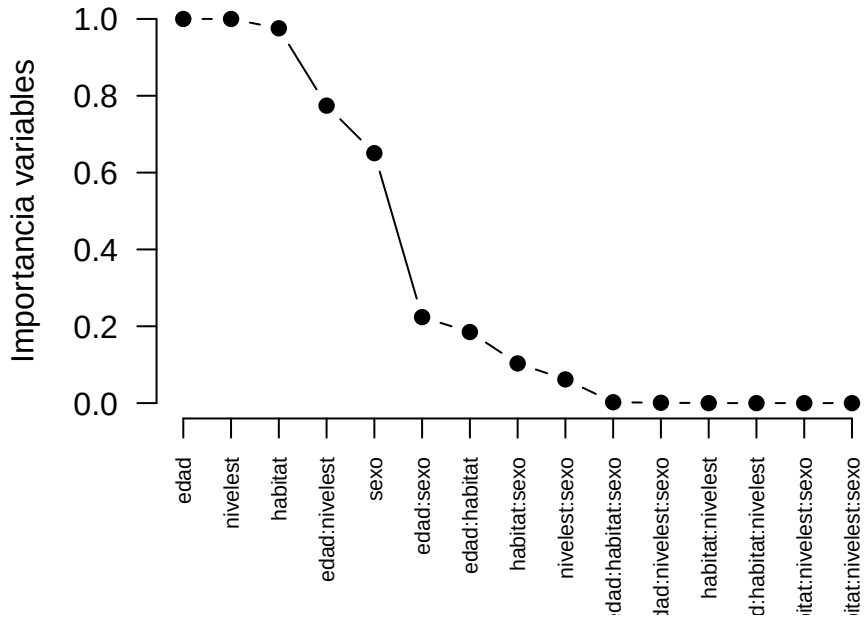


Figura 4.7: Importancia de las variables

Los efectos principales de edad, nivel de estudios y hábitat aparecen sistemáticamente en los modelos con menor valor de AICc. Los siguientes efectos en importancia son, la interacción entre edad y nivel de estudios, y el efecto principal del sexo.

Capítulo 5

Diagnóstico y validación

Una vez comprobado el ajuste global del modelo de regresión logística, vamos a estudiar la falta de ajuste a nivel de cada observación y cómo afectan las observaciones al modelo general. Para eso, recurrimos al análisis de los residuos y de los valores influyentes.

En los modelos lineales generalizados, el análisis de los residuos puede indicar también si la posible falta de ajuste se debe por ejemplo a una mala elección de la función *link* o si el efecto de la variables explicativas no es lineal.

Para ilustrar cómo se realizaría el diagnóstico y validación, consideramos el modelo que incorpora los efectos principales de edad, nivel de estudios y hábitat más la interacción entre edad y nivel de estudios. El modelo se ha ajustado utilizando el siguiente código.

```
modelo.stp <- glm(formula = uso_int ~ edad + nivelest + habitat + edad:nivelest,
    family = binomial, data = datos.bin)
```

5.1. Análisis de los residuos

En la subsección 4.3.2 del capítulo anterior se describieron los distintos tipos de residuos que se pueden calcular en un modelo lineal generalizado. La distribución de los residuos, bajo la hipótesis nula de que el modelo se ajusta bien a los datos, se puede asumir como una nomal de media 0 y varianza estimada menor que 1. A efectos de diagnóstico asumiremos que un residuo es significativamente distinto de 0 si su valor absoluto es mayor que 2.

5.1.1. Cálculo de residuos

Utilizando las funciones `residuals` , `rstandard` y `rstudent` se calculan todos los tipos de residuos vistos en el capítulo anterior.

Residuos de pearson

```
res.p <- residuals(modelo.stp, type = "pearson")
# Residuos de pearson de los 6 primeros individuos de la encuesta
head(res.p)
```

```
##       1       2       3       4       5       6
## -0.7074 -0.1931 -0.2271 -0.3317 -0.6702  0.8361
```

Podemos ver cuántos son significativos

```
res.p.sig <- abs(res.p) > 2
table(res.p.sig)

## res.p.sig
## FALSE  TRUE
##  3335   150
```

Que suponen menos del 5 % de los residuos

Podemos ordenar el conjunto de datos de mayor a menor valor de los residuos.

```
res.orde <- sort(abs(res.p[res.p.sig]), decreasing = TRUE)
# mostramos solo los más altos
head(res.orde)

##   1826    172    700   2434   1646   1457
## 12.285  9.166  9.163  8.562  7.819  7.017

# con names(res.orde) obtenemos la fila a la que corresponde, y ordenamos
# el data.frame utilizando names(res.orde) como índice
head(datos.bin[names(res.orde), ])

##      uso_int   sexo edad  habitat                 nivelest
## 1826       1  Mujer   88 estrato5              Analfabetos
## 172        0  Mujer   41 estrato3 Universitaria o superior
## 700        1 Hombre   67 estrato4              Analfabetos
## 2434       0  Mujer   43 estrato0 Universitaria o superior
## 1646       1 Hombre   79 estrato6                 Primaria
## 1457       1 Hombre   77 estrato6                 Primaria
```

Vemos que los residuos más altos corresponden a personas que, tanto por la edad, hábitat o nivel de estudios terminados, el modelo le asigna la pertenencia a la categoría opuesta en el uso de internet de la que tienen. El residuo de pearson más alto corresponde a una mujer sin estudios de 88 años que vive en un municipio de entre 10 mil y 20 mil habitantes. El modelo asigna a ese perfil una probabilidad del uso de internet de prácticamente 0.

```
fitted.values(modelo.stp)[1826]

##     1826
## 0.006583
```

Lo mismo ocurre con el resto de individuos con residuos altos. Su configuración de variables predictoras, hace que el modelo los clasifique con una alta probabilidad en la categoría contraria de la que respondieron.

```
cbind(observado = datos.bin$uso_int, predicho = fitted.values(modelo.stp))[c(1826,
    700, 172, 2434, 1646, 1457), ]

##      observado predicho
## 1826         1 0.006583
## 700          1 0.011771
## 172          0 0.988239
## 2434         0 0.986543
## 1646         1 0.016094
## 1457         1 0.019905
```

Residuos de pearson estandarizados

Función `rstandard`

```
res.p.std <- rstandard(modelo.stp, type = "pearson")
```

Serán significativos aquellos cuyo valor absoluto sea mayor de 2.

```
res.p.std.sig <- abs(res.p.std) > 2
table(res.p.std.sig)

## res.p.std.sig
## FALSE  TRUE
##  3335   150

# utilizamos res.p.std.sig como indice para ver el valor de los
# significativos
head(res.p.std[res.p.std.sig])

##     14     17     94     97     99    154
##  2.433 -3.966 -2.649  2.380 -3.263 -2.455
```

Al igual que antes, ordenamos de mayor a menor y vemos qué valor tienen en las variables predictoras.

```
res.orde <- sort(abs(res.p.std[res.p.std.sig]), decreasing = TRUE)
# mostramos solo los más altos
head(res.orde)

##   1826    700    172   2434   1646   1457
## 12.324  9.182  9.178  8.573  7.821  7.019

head(datos.bin[names(res.orde), ])

##      uso_int   sexo edad  habitat                nivelest
## 1826       1  Mujer   88 estrato5              Analfabetos
## 700        1 Hombre   67 estrato4              Analfabetos
```

```
## 172        0  Mujer   41 estrato3 Universitaria o superior
## 2434       0  Mujer   43 estrato0 Universitaria o superior
## 1646       1 Hombre   79 estrato6                 Primaria
## 1457       1 Hombre   77 estrato6                 Primaria
```

Residuos de la devianza

```
res.d <- residuals(modelo.stp, type = "deviance")
# significativos
res.d.sig <- abs(res.d) > 2
table(res.d.sig)

## res.d.sig
## FALSE  TRUE
##  3388    97
```

Residuos de la devianza estandarizados

```
res.dev.std <- rstandard(modelo.stp, type = "deviance")
# significativos
table(abs(res.dev.std) > 2)

##
## FALSE  TRUE
##  3387    98
```

Siendo menos de un 3 % los residuos significativos

Residuos studentizados

```
res.student <- rstudent(modelo.stp)
res.orde <- sort(res.student, decreasing = TRUE)
head(res.orde)

##  1826   700  1646  1457  2535  3267
## 3.321 3.040 2.879 2.804 2.779 2.766

head(datos.bin[names(res.orde), ])

##      uso_int   sexo edad  habitat    nivelest
## 1826       1  Mujer   88 estrato5 Analfabetos
## 700        1 Hombre   67 estrato4 Analfabetos
## 1646       1 Hombre   79 estrato6    Primaria
## 1457       1 Hombre   77 estrato6    Primaria
## 2535       1  Mujer   50 estrato2 Analfabetos
## 3267       1  Mujer   76 estrato6    Primaria

table(abs(res.student) > 2)
```

```
##
## FALSE  TRUE
##  3387    98
```

El número de residuos significativo, es en todos los casos, inferior al 5 %.

5.1.2. Gráficos de los residuos

Cuando tenemos un elevado número de observaciones, como en el caso de la encuesta del uso de las TIC, es útil la representación gráfica de los residuos. La forma más habitual es representar los residuos frente a los valores predichos por el modelo. Este gráfico, en el caso de tener observaciones repetidas, es decir, con los datos agrupados, debería dar lugar a una banda horizontal alrededor del cero.

Vamos a ir viendo los diferentes gráficos que podemos construir para analizar los residuos. Podemos empezar dibujando los valores de los residuos de la devianza para cada encuestado y añadir las rectas y=2 e y=-2.

```
plot(res.dev.std, cex = 0.6)
abline(h = c(-2, 2), col = "red")
```

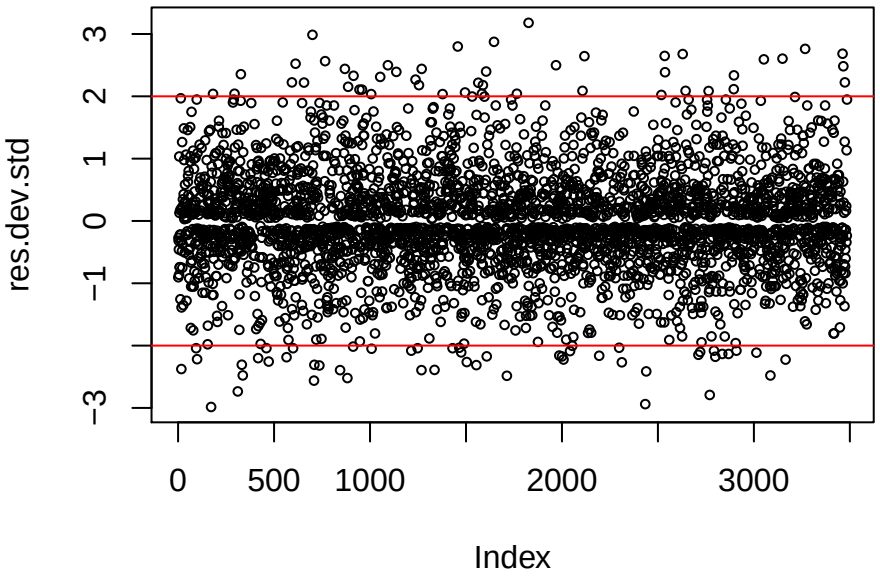


Figura 5.1: Residuos de la devianza estandarizados

Representamos sólo los que son mayores que 2 en valor absoluto y los etiquetamos.

```
signif <- which(abs(res.dev.std) > 2)
plot(res.dev.std[signif], type = "n")
text(1:length(signif), res.dev.std[signif], label = signif, cex = 0.4)
```

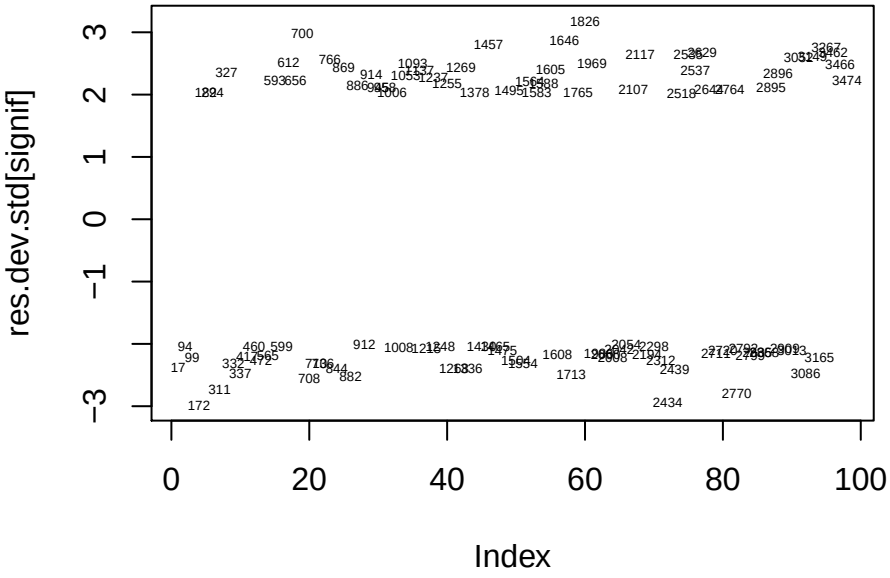


Figura 5.2: Residuos significativos

También se puede obtener el histograma y añadirle las líneas que representan los cuantiles 0.025 y 0.975, con lo que, entre esas dos líneas estarían el 95 % de los residuos.

```
hist(res.dev.std, breaks = 40)
abline(v = quantile(res.dev.std, probs = c(0.05/2, 1 - 0.05/2)), lty = 2)
```

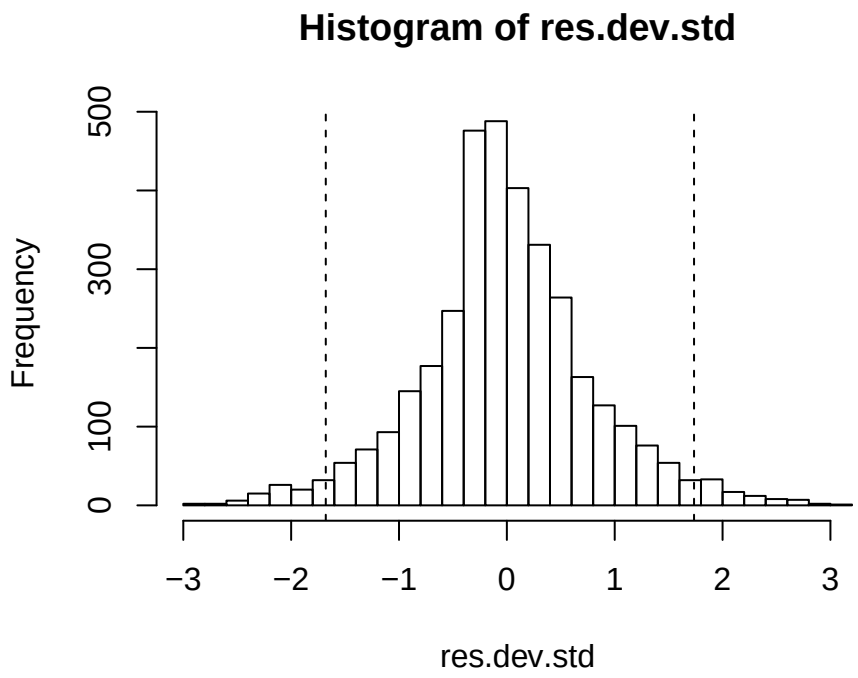


Figura 5.3: Histograma de los residuos

El gráfico de los residuos frente a los valores predichos (figura 5.4) no es fácil de interpretar cuando los datos no están agrupados y da lugar a gráficos que muestran un claro patrón. Esto es debido a que la variable respuesta sólo puede tomar el valor 0 o 1 para cada observación.

```
plot(fitted.values(modelo.stp), res.dev.std, xlab = "Prob.predichas", ylab = "Residuos")
```

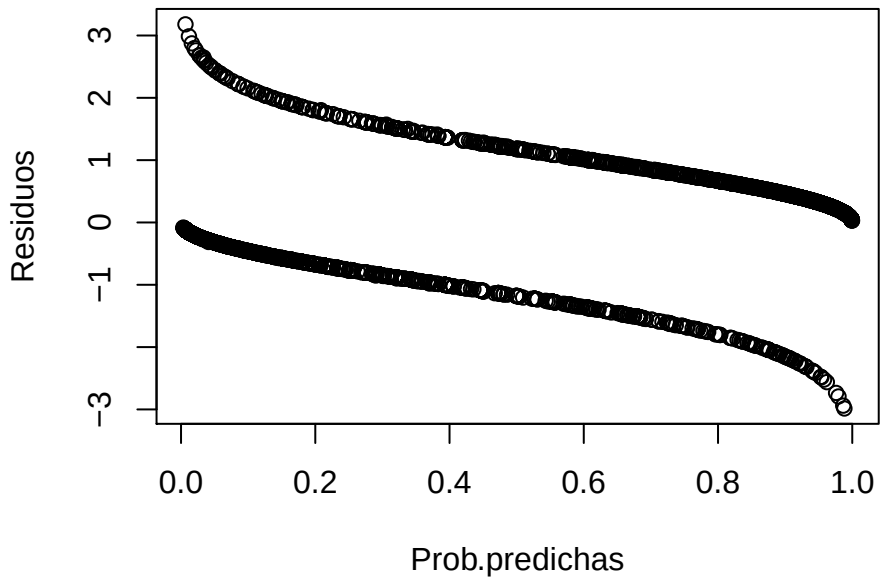


Figura 5.4: Residuos frente a probabilidades predichas

La representación de las observaciones frente a los valores predichos que, si el modelo ajusta bien, debería dar lugar a puntos alrededor de la recta de pendiente uno y constante cero si tenemos datos agrupados. Si los datos no están agrupados, el gráfico debería mostrar que los valores predichos cerca de cero tienen la mayoría de sus observaciones iguales a cero, que valores predichos cercanos a uno deberían tener mayoría de observaciones iguales a uno y que valores predichos de 0.5 deberían tener un número parecido de observaciones con valor cero y uno.

```
plot(datos.bin$uso_int, fitted.values(modelo.stp), xlab = "Valores observados",
    ylab = "Valores predichos")
```

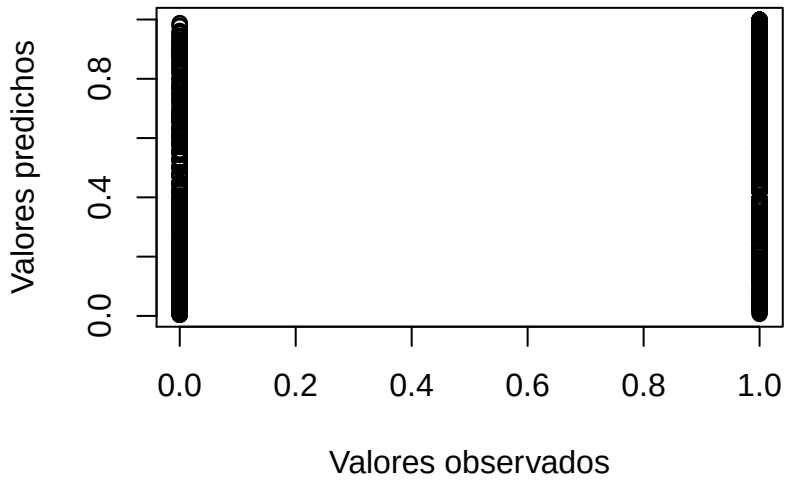


Figura 5.5: Valores observados frente a predichos

El solapamiento de los datos no permite analizar el gráfico. Se puede arreglar añadiendo una pequeña variación a los datos del eje x

```
plot(jitter(datos.bin$uso_int), fitted.values(modelo.stp), cex = 0.5, xlab = "Valores observados",
    ylab = "Valores predichos")
```

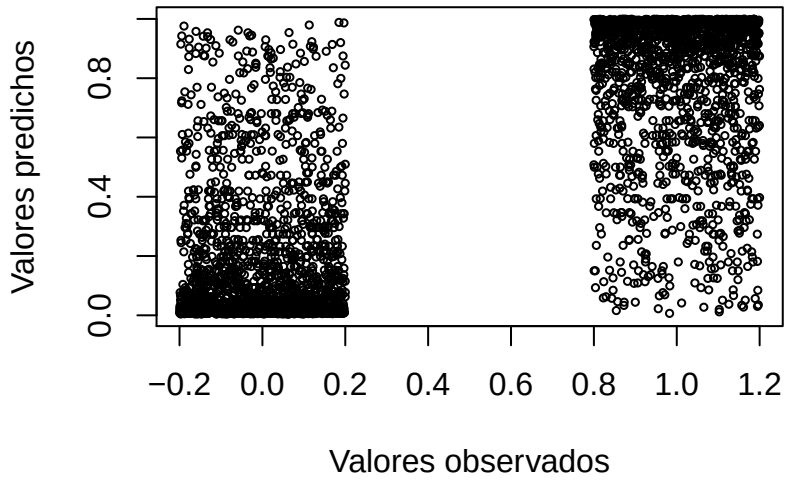


Figura 5.6: Valores observados frente a predichos. Añadiendo una pequeña variación a los datos observados.

En este último gráfico, si se observa que los valores predichos cercanos a cero se corresponden mayoritariamente con valores observados iguales a cero y viceversa.

Otro gráfico útil es el de los residuos frente a las variables explicativas, así se puede observar mejor el valor en las variables explicativas de los posibles residuos significativos

```
plot(datos.bin$edad, res.dev.std, cex = 0.5)
```

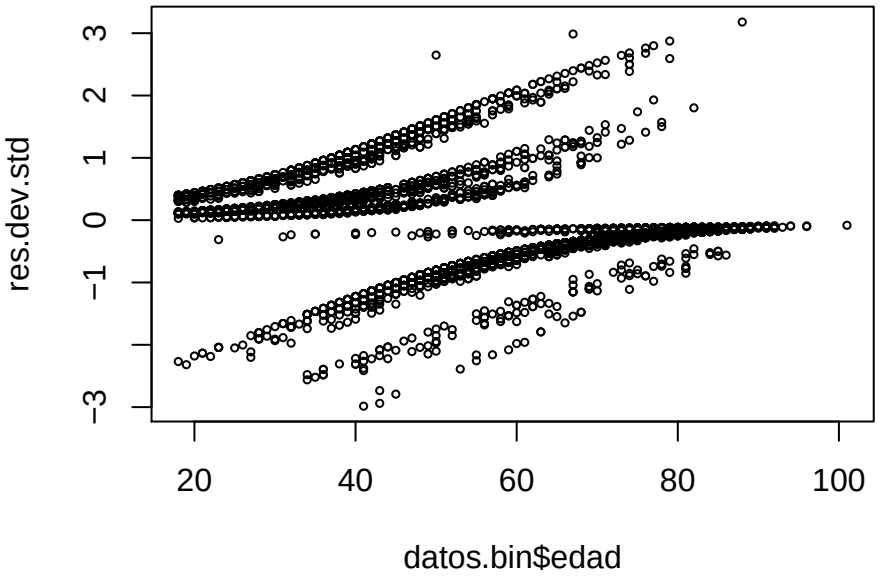


Figura 5.7: Residuos frente a edad.

Al igual que en el gráfico de los residuos frente a las probabilidades predichas, se observa un patrón de los residuos. Esto es debido a que para cada edad se tienen 56 posibles residuos (7 niveles de hábitat $\times$ 4 niveles de nivel de estudios $\times$ 2 posibles respuestas de uso internet). En general, para una edad concreta, es improbable que haya individuos en todas las combinaciones de habitat y nivel de estudios. Por ejemplo, la tabla de contingencia de hábitat y nivel de estudios para edad=52 es

```
with(datos.bin, xtabs(edad == 52 ~ habitat + nivelest))
```

```
##           nivelest
## habitat    Analfabetos Primaria Secundaria y F.P Universitaria o superior
##   estrato6           2       15                2                         2
##   estrato0           0        2                1                         0
##   estrato1           1        6                0                         0
##   estrato2           0        0                2                         0
##   estrato3           0        8                1                         0
##   estrato4           0        9                2                         2
##   estrato5           0        5                3                         2
```

Para observar la relación entre los residuos y una variable predictora categórica, dibujamos los boxplots de los residuos en cada categoría. Este gráfico no debería mostrar ningún patrón y los boxplots deberían estar centrados alrededor de la recta y=0.

```
plot(datos.bin$habitat, res.dev.std, cex = 0.5, cex.axis = 0.7)
```

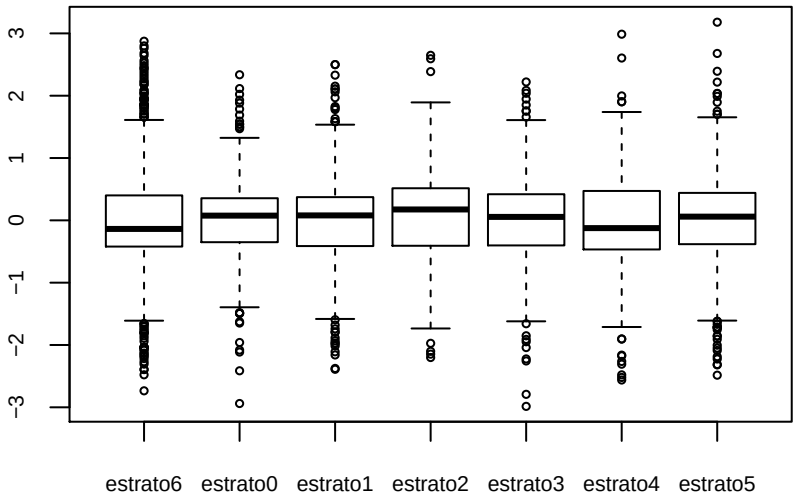
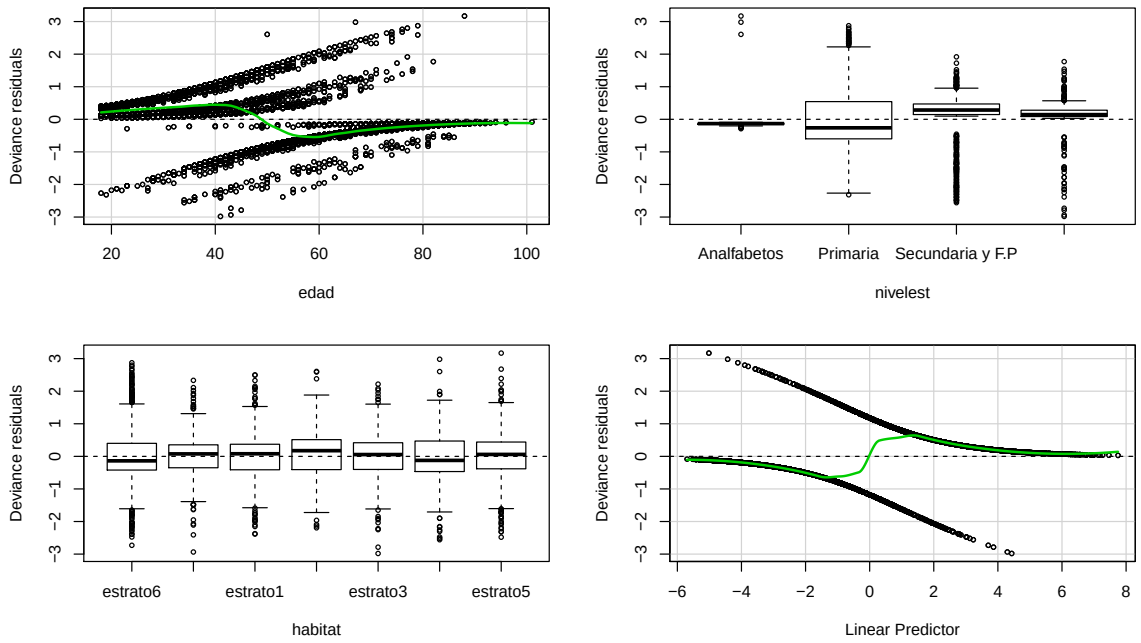


Figura 5.8: Residuos frente a habitat.

Una función que simplifica la creación de estos gráficos es `residualPlots`, del paquete `car`, que representa los residuos (por defecto los de Pearson, pero se pueden cambiar) frente a las variables predictoras, y también frente a las transformaciones logit (predictor lineal). Esta función incluye, para las variables continuas y el predictor lineal, un ajuste mediante regresión suavizada (función `loess` en R), además realiza un test de falta de ajuste para las variables numéricas. En el caso de la variable edad el test tiene un p-valor alto que indica que no hay falta de ajuste. Según la ayuda de la función, el test contrasta si añadir el cuadrado de la variable, en este caso edad, es significativo.

```
residualPlots(modelo.stp, type = "deviance", cex = 0.6)
```




```
##          Test stat Pr(>|t|)
## edad         0.186    0.666
## nivelest        NA       NA
## habitat         NA       NA
```

Figura 5.9: Gráficos de residuos para modelo.stp

En nuestros datos, el número de residuos significativos es muy pequeño comparado con el total de residuos. Al final del capítulo 4, vimos que los tests de bondad de ajuste globales y el análisis de la curva ROC para este modelo indicaban un ajuste excelente.

En el caso de tener un mayor número de observaciones con residuos significativos, se podría pensar en cómo influye cada una de esas observaciones sobre el modelo de regresión logística.

5.2. Medidas de influencia

Pueden existir observaciones que, situándose lejos del resto pueden influir en las estimaciones del modelo. Las medidas de influencia detectan los valores influyentes analizando el efecto que esos valores tienen en los parámetros del modelo. En el ámbito de la regresión lineal se puede encontrar una descripción detallada en (Belsley, Kuh, Welsch, et al., 2004). Para modelos lineales generalizados podemos encontrar una aproximación a las medidas de influencia en (Williams, 1987) y en (Cook and Weisberg, 1982).

Las medidas más usuales son los *hat values* y las *distancias de cook,* que se pueden aproximar para los modelos glm con la siguiente expresión.

$$D_q = \frac{e^2_{PSq}}{k+1} \times \frac{h_{qq}}{1-h_{qq}}$$

Tanto los valores h_{qq} como las distancias de cook se pueden calcular con las funciones `hatvalues` y `cooks.distance`.

```
distancias.cook <- cooks.distance(modelo.stp)
head(distancias.cook)

##         1         2         3         4         5         6
## 2.172e-04 3.893e-06 6.671e-06 2.290e-05 1.880e-04 5.269e-04

hat.valores <- hatvalues(modelo.stp)
head(hat.valores)

##        1        2        3        4        5        6
## 0.006003 0.001457 0.001804 0.002897 0.005793 0.010334
```

Se consideran valores influyentes aquellos cuya distancia de cook sea mayor que 1 (Cook and Weisberg, 1982) .

```
table(distancias.cook > 1)

##
## FALSE
##  3485
```

En nuestro caso, no hay ninguna distancia de cook significativa.

Otras medidas alternativas son los DFBETAS. Los DFBETAS proporcionan el cambio en los coeficientes estimados $\hat{\beta}_r$, cuando se elimina cada una de las observaciones, los DFBETAS son muy útiles para ver sobre qué variable del modelo es influyente cada observación.

Estas medidas junto con las distancias de cook y algunas más, se pueden obtener con la función `influence.measures` o directamente con las funciones `dfbetas, cooks.distances` o `dffits`[1]. Para nuestro modelo, las medidas de influencia para las 6 primeras observaciones son.

```
medidas.infl <- influence.measures(modelo.stp)
colnames(medidas.infl$infmat)

##  [1] "dfb.1_"   "dfb.edad" "dfb.nvlP" "dfb.nSyF" "dfb.nUos" "dfb.hbt0"
##  [7] "dfb.hbt1" "dfb.hbt2" "dfb.hbt3" "dfb.hbt4" "dfb.hbt5" "dfb.ed:P"
## [13] "dfb.e:yF" "dfb.e:os" "dffit"    "cov.r"    "cook.d"   "hat"

# mostramos las medidas para las primeras 6 observaciones
head(medidas.infl$infmat)

##      dfb.1_   dfb.edad   dfb.nvlP   dfb.nSyF   dfb.nUos   dfb.hbt0
## 1 1.913e-04  1.462e-04 -1.475e-04 -0.0011336 -4.617e-04 -0.0050206
```

[1]Las funciones `influence.measures`, `hatvalues` ,`cooks.distance` ,`dfbetas` y `dffits` forman parte del paquete `stats` , que se carga por defecto al iniciar R

```
## 2 1.386e-05  1.335e-05  5.826e-04 -0.0001133 -4.639e-05  0.0002040
## 3 2.033e-05  1.882e-05  6.906e-04 -0.0001576 -6.449e-05  0.0001421
## 4 4.748e-05  4.065e-05  9.028e-04 -0.0003307 -1.351e-04 -0.0003472
## 5 1.765e-04  1.358e-04  6.031e-05 -0.0010562 -4.303e-04 -0.0044440
## 6 1.214e-04 -2.030e-04  1.815e-04 -0.0080500  3.126e-04 -0.0081459
##    dfb.hbt1   dfb.hbt2   dfb.hbt3   dfb.hbt4   dfb.hbt5   dfb.ed:P
## 1 -0.075664 -1.300e-03 -0.0011971 -7.765e-04 -0.0022680 -0.0005037
## 2 -0.008062  7.545e-05  0.0001968  3.696e-06 -0.0002739 -0.0009421
## 3 -0.011116  6.372e-05  0.0002102 -1.119e-05 -0.0003716 -0.0011316
## 4 -0.022868 -5.408e-05  0.0001518 -9.781e-05 -0.0007366 -0.0015590
## 5 -0.070660 -1.143e-03 -0.0010106 -6.965e-04 -0.0021287 -0.0007644
## 6  0.069275 -3.519e-03 -0.0017531 -3.108e-03 -0.0066568 -0.0006408
##    dfb.e:yF   dfb.e:os    dffit cov.r    cook.d      hat
## 1 0.0024281  0.0018769 -0.08831 1.005 2.172e-04 0.006003
## 2 0.0002478  0.0001919 -0.01301 1.005 3.893e-06 0.001457
## 3 0.0003437  0.0002661 -0.01697 1.005 6.671e-06 0.001804
## 4 0.0007165  0.0005544 -0.03101 1.006 2.290e-05 0.002897
## 5 0.0022639  0.0017501 -0.08293 1.005 1.880e-04 0.005793
## 6 0.0144833 -0.0015514  0.13302 1.008 5.269e-04 0.010334
```

Además de los DFBETAS se obtienen los DFFITS (medida de cómo cambia el valor ajustado para la observación i cuando esta no se ha utilizado en el ajuste del modelo), el COVRATIO (que mide el cambio, al eliminar la observación i, en el determinante de la matriz de covarianzas de las variables predictoras), las distancias de Cook y los valores h_{qq}.

5.2.1. Gráficos para medidas de influencia

Cuando tenemos muchas observaciones, los métodos gráficos son de gran ayuda. En R, una vez que hemos calculado los valores influyentes, podemos guardarlos en objetos y representarlos de la forma más conveniente, con histogramas, boxplots etcétera. Aunque lo más cómodo es utilizar funciones que vienen implementadas en la instalación base o en algunos paquetes como `car`.

Por ejemplo, se puede utilizar la función genérica `plot` sobre un objeto glm, que dibuja algunos gráficos de diagnóstico.

```
par(mfrow = c(2, 2))
plot(modelo.stp, cex = 0.6)
```

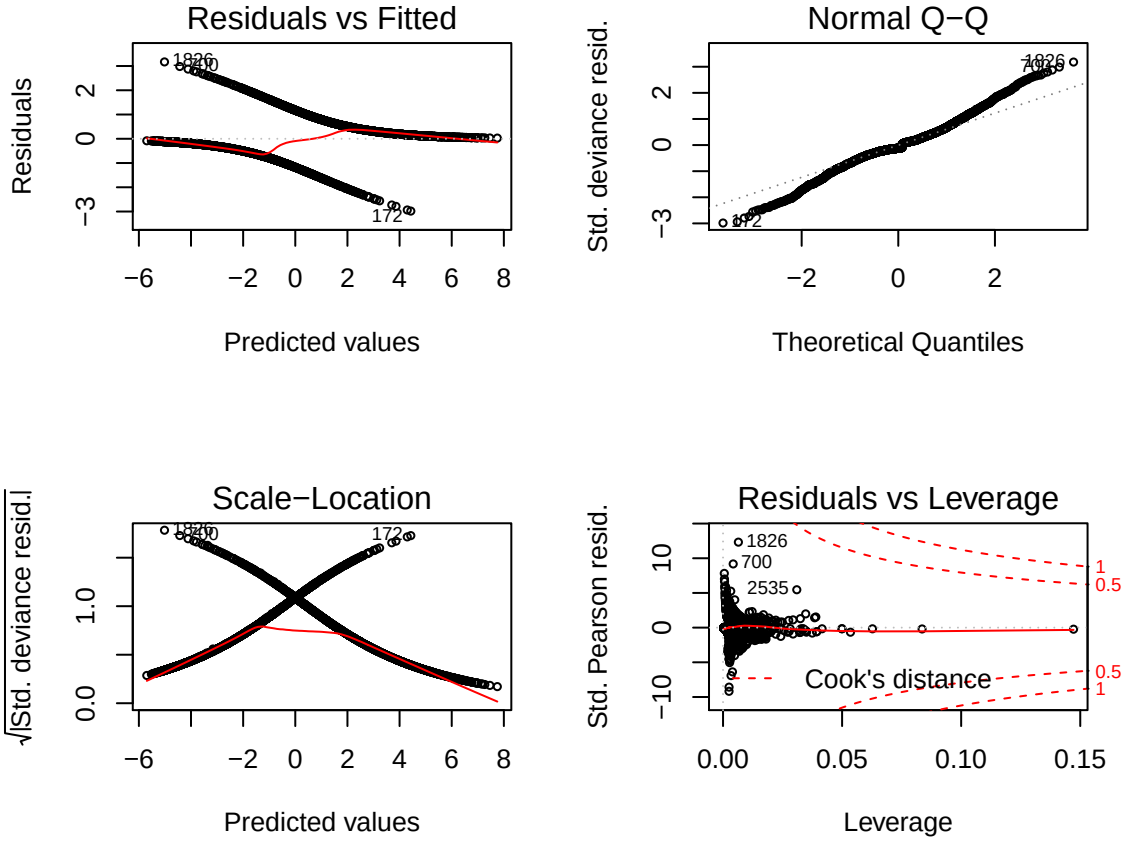


Figura 5.10: Función plot aplicada sobre un objeto glm.

El primer gráfico, empezando por arriba y de izquierda a derecha, se muestran los residuos de la devianza frente al predictor lineal (transformaciones logit). El segundo compara los residuos de la devianza estandarizados con los cuantiles de una normal estándar. El tercer gráfico es de nuevo un gráfico de los residuos frente al predictor lineal, pero tomando la raíz cuadrada del valor absoluto de los residuos de la devianza estandarizados. El cuarto gráfico es el más útil para detectar valores influyentes, ya que compara los residuos estandarizados de Pearson con los *hat values* y además muestra líneas de contorno para las distancias de cook. Como ya habíamos visto, no hay ninguna observación que tenga un valor elevado. En el cuarto gráfico se han etiquetado por defecto las 3 observaciones con una mayor distancia de cook.

Si sólo estamos interesados en ver las observaciones influyentes, podemos utilizar dos gráficos que no se muestran por defecto, como es el que muestra las distancias de cook y el que las compara con los h_{qq}. Para eso, especificamos la opción `which` dentro de la función plot.

```
par(mfrow = c(1, 2))
plot(modelo.stp, which = 4)
plot(modelo.stp, which = 6)
```

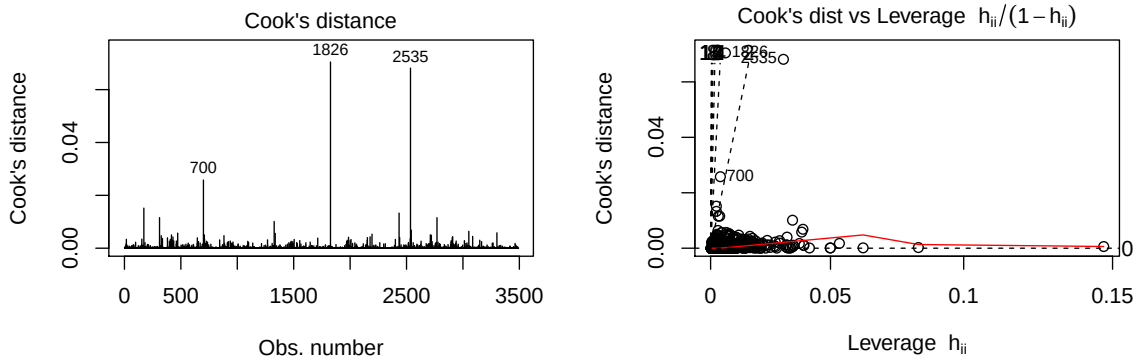


Figura 5.11: Distancias de cook y distancias de cook frente a los *hat values*

Vemos que ninguna de las 3 observaciones con mayor distancia de cook tiene además un valor alto de h_{qq}.

Otra función para analizar gráficamente los valores influyentes es la función `influenceIndexPlot` del paquete `car`, que muestra las distancias de cook, los residuos estudentizados, los p-valores (corregidos por el método de Bonferroni) de los contrastes de si los residuos son distintos de 0 y los *hat values.*

```
# con id.n=3 indicamos que muestre los tres mayores valores en cada
# gráfico
influenceIndexPlot(modelo.stp, id.cex = 0.7, id.n = 3)
```

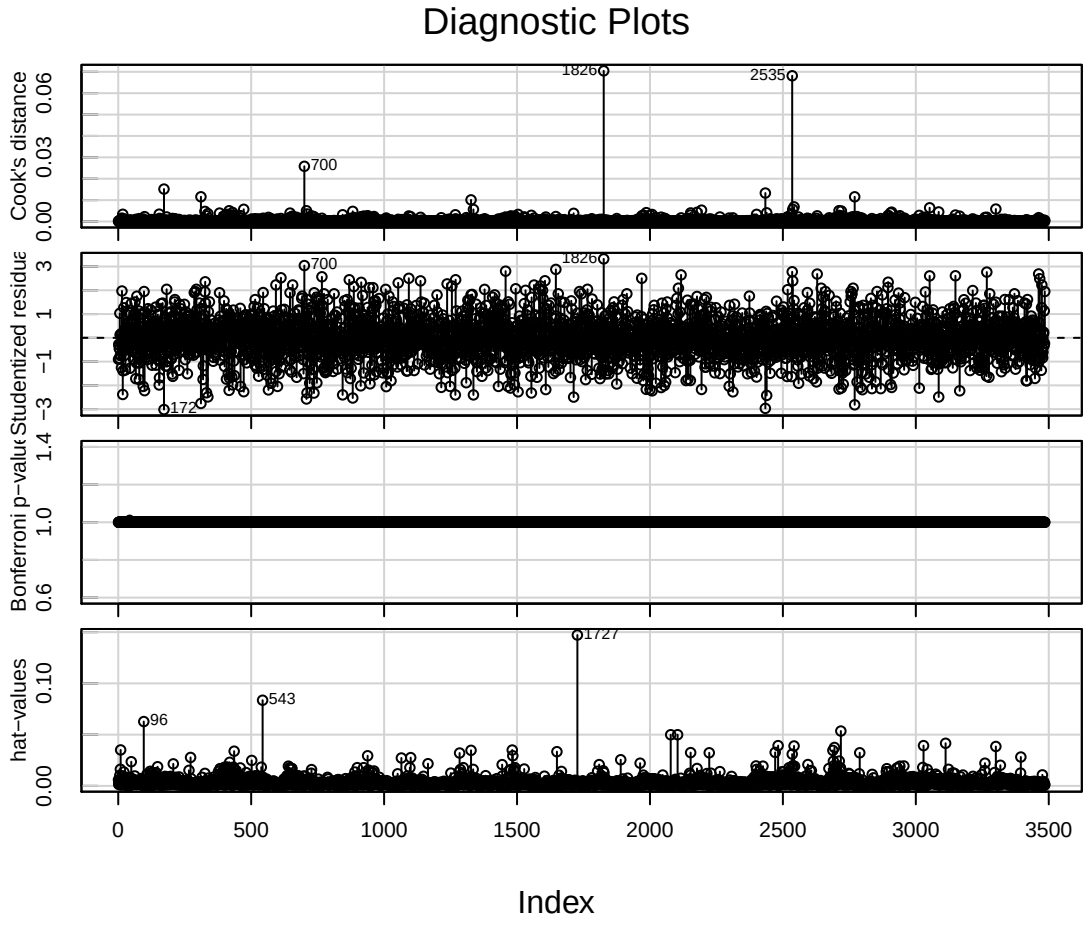


Figura 5.12: Gráficos de influencia con influenceIndexPlot

5.3. Colinealidad y Factores de inflación de la varianza (VIF)

Cuando existe una relación lineal fuerte entre los predictores de una regresión lineal se reduce la precisión de los coeficientes estimados, es decir, se incrementa su varianza. La varianza estimada para un coeficiente β_j en regresión lineal es:

$$\widehat{Var}(\beta_j) = \frac{\widehat{\sigma^2}}{(n-1)s_j^2} \times \frac{1}{1-R_j^2}$$

dónde $\widehat{\sigma^2}$ es la estimación de la varianza de los errores, s_j^2 es la varianza muestral del predictor x_j y $1/(1-R_j^2)$ es el factor de inflación de la varianza (VIF) de β_j. R_j^2 es el coeficiente de correlación múltiple de la regresión de x_j sobre los otros predictores del modelo. Así, si una variable x_j está muy poco correlacionada con el resto, el valor del VIF es próximo a 1.

La raíz cuadrada de VIF nos dice como se incrementa el intervalo de confianza para el parámetro β_j

en comparación con ese mismo intervalo en ausencia de multicolinealidad.

Tal y como se describe en (Fox and Weisberg, 2011) VIF no es adecuado para modelos en los que haya variables categóricas como predictoras. (Fox and Monnette, 1992) generalizan la noción de inflación de la varianza asociada al incremento de la región de confianza de un conjunto de regresores[2]. Obtuvieron una medida que llamaron *factor de inflación de la varianza generalizado* (GVIF)[3]. Si tenemos una variable predictora que tiene p regresores, el valor de $GVIF^{1/2p}$ es una medida de cómo disminuye la precisión de la estimación de los coeficientes debido a la existencia de colinealidad. Al igual que con el factor de inflación de la varianza, un valor próximo a 1 de $GVIF^{1/2p}$ indicaría la ausencia de colinealidad.

Vamos a examinar la posible multicolinealidad en el modelo ajustado. Al tener variables categóricas como el nivel de estudios y el hábitat, hay que utilizar el factor de inflación de la varianza generalizado.

```
# Utilizamos la función vif del paquete car
vif(modelo.stp)

##                    GVIF Df GVIF^(1/(2*Df))
## edad             101.56  1          10.078
## nivelest       10962.72  3           4.713
## habitat            1.12  6           1.010
## edad:nivelest 48337.67  3           6.036
```

Nos da valores de $GVIF^{1/2p}$ muy elevados, lo que indica colinealidad. Esto es debido a que en nuestro modelo, al considerar la interacción entre edad y nivel de estudios, se introducen en el modelo un conjunto de regresores muy correlacionados con edad y nivel de estudios. Podemos calcular el GVIF sobre el modelo que no tenga la interacción.

```
# usamos la función update para quitar el término de interacción en el
# modelo
vif(update(modelo.stp, ~. - edad:nivelest))

##           GVIF Df GVIF^(1/(2*Df))
## edad     1.118  1           1.057
## nivelest 1.106  3           1.017
## habitat  1.111  6           1.009
```

Y vemos que el $GVIF^{1/2p}$ está próximo a 1, lo que indica ausencia de colinealidad.

[2] Entendiendo como regresores a las variables explicativas del modelo, y en su caso, a las variables auxiliares construidas a partir de variables categóricas. Una variable categórica con $p+1$ categorías, tiene asociadas p variables regresoras.

[3] Sea $\boldsymbol{R}_{11}$ la matriz de correlaciones entre el conjunto de regresores en cuestión (como los p regresores de una variable con p+1 categorías), $\boldsymbol{R}_{22}$ la matriz de correlaciones entre los otros regresores del modelo y $\boldsymbol{R}$ la matriz de correlaciones entre todos los regresores del modelo, el factor de inflación de la varianza generalizado se calcula como

$$GVIF = \frac{\det \boldsymbol{R}_{11} \det \boldsymbol{R}_{22}}{\det \boldsymbol{R}}$$

cuyo valor es igual a 1 si los regresores de interés (con matriz de correlaciones $\boldsymbol{R}_{11}$) están incorrelados con el resto de regresores del modelo.

Si se considera la región de confianza conjunta asociada a un conjunto de regresores, (Fox and Monnette, 1992) demostraron que el valor de GVIF realiza un papel análogo al del factor de inflación de la varianza en la regresión lineal.

5.4. Validación cruzada

Un problema recurrente en el ajuste de modelos es el sobreajuste, que implica que el modelo se ajusta muy bien a nuestro conjunto de datos pero puede no ser útil en el ajuste de otros datos, es decir, es menos generalizable. Una forma de validar el modelo es comprobar si el modelo predice correctamente un nuevo conjunto de datos. En nuestro caso, podríamos pensar en utilizar el modelo ajustado para predecir la variable *uso de internet* en los datos de encuestas anteriores o posteriores a 2011. No obstante, normalmente no se dispone de otro conjunto de datos con el que poder validar el modelo. Una alternativa, si se disponen de las suficientes observaciones es dividir la muestra en dos submuestras, de forma que una se utilice para ajustar el modelo y la otra para evaluación del mismo.

La validación cruzada parte de esta última alternativa, considerando dos variantes. En la primera, la llamada *K-Fold cross-validation,* se divide la muestra en K submuestras, de forma que se utilizan K-1 para estimar el modelo y la restante como submuestra de evaluación, este proceso se repite K veces, de forma que cada submuestra es utilizada una vez para evaluar el modelo y *K-1* veces para el ajuste. Como medida de validación se suele utilizar la media de las tasas de clasificaciones correctas, o su complementario que sería la tasa de clasificaciones incorrectas. A la segunda aproximación se la conoce como *leave-one-out cross-validation* (LOOCV) y consiste en tomar K igual al número de observaciones, este método exige un mayor coste computacional, debido a que hay que ajustar tantos modelos como observaciones.

En R podemos programar la validación cruzada o utilizar funciones que vienen incluidas en algunas librerías. Dentro de la librería boot (Canty and Ripley, 2012) existe la función `cv.glm` cuyos argumentos los podemos ver con `args(cv.glm)`.

```
library(boot)
args(cv.glm)

## function (data, glmfit, cost = function(y, yhat) mean((y - yhat)^2),
##     K = n)
## NULL
```

Vemos que toma 4 argumentos, el conjunto de datos, el modelo ajustado, una función de coste y el número de grupos en los que se va a dividir la muestra. La función de coste toma por defecto la media de los errores al cuadrado. En la ayuda de `cv.glm` se indica que la función de coste ha de tomar como primer argumento los valores observados y como segundo los valores predichos por el modelo. En el caso de la regresión logística, podemos construirnos una función de coste más apropiada.

```
cost <- function(r, pi) mean(abs(r - pi) > 0.5)
```

La función `cost` mide para cada observación si el modelo la clasifica incorrectamente mediante `abs(r-pi)>0.5,` que aplicado a todas las observaciones devuelve un vector con el valor TRUE si el valor absoluto de la diferencia entre los valores observados y predichos es mayor de 0.5, y FALSE en caso contrario. Recordemos que la codificación utilizada es 1 para las personas que si han utilizado internet, y 0 para las que no[4]. Por tanto, si el modelo clasifica incorrectamente a un encuestado el valor de `abs(r-pi)>0.5` es TRUE. Con `mean(abs(r-pi)>0.5)` lo que estamos haciendo es obtener la proporción de individuos mal clasificados.

[4]Si la variable que tenemos es de tipo factor con dos categorías habría que recodificarla adecuadamente.

Utilizando ahora la función `cv.glm` con la nueva función `cost` y con K=10, realizamos la validación cruzada.

```
res.10.fold <- cv.glm(datos.bin, modelo.stp, cost, K = 10)
```

Dentro del objeto `res.10.fold` se ha guardado el valor medio de la tasa de casos incorrectamente clasificados a la que se accede mediante el operador `$`.

```
res.10.fold$delta

## [1] 0.1400 0.1396
```

Obtenemos 2 valores, en la ayuda de la función se dice que el primer valor corresponde a la media de la tasa de clasificaciones incorrectas, y que el segundo es un valor corregido de dicha tasa para compensar el sesgo que se introduce al no utilizar el método *leave-one-out cross-validation (Efron, 1974).* El valor obtenido indica que, en media, el modelo clasifica correctamente a más del 80 % de los encuestados cuando estos no han sido usados en el ajuste del modelo.

```
1 - res.10.fold$delta

## [1] 0.8600 0.8604
```

Si lo que queremos es validación cruzada mediante el método *leave-one-out cross-validation,* elegimos K igual al número de observaciones en los datos.

```
# puede tardar en realizar el cálculo, 7-9 minutos
res.loocv <- cv.glm(datos.bin, modelo.stp, cost, K = nrow(datos.bin))
res.loocv$delta

## [1] 0.1423 0.1423
```

Y obtenemos valores similares que con el método de K-fold.

Otros paquetes también incorporan sus propias funciones para validación cruzada, tal es el caso del paquete DAAG (Maindonald and Braun, 2013) que incorpora la función `CVbinary` o del paquete rms (Harrel, 2013) con la función `validate`. Como ejemplo, vamos a utilizar la función `CVbinary`.

```
library(DAAG)
args(CVbinary)

## function (obj, rand = NULL, nfolds = 10, print.details = TRUE)
## NULL
```

La función toma como argumentos el modelo ajustado, un vector opcional con las asignaciones de cada observación a una submuestra (si no se indica, la asignación es aleatoria), el número de submuestras a considerar (por defecto 10), y un argumento lógico si queremos que nos muestre la salida al terminar.

La sintaxis sería

```
res.daag <- CVbinary(modelo.stp)

##
## Fold:  1 10 4 7 5 3 6 2 8 9
## Internal estimate of accuracy = 0.861
## Cross-validation estimate of accuracy = 0.859
```

Esta función devuelve la tasa de clasificaciones correctas, dando por un lado la mejor estimación en un conjunto de entrenamiento y por otro la estimación mediante validación cruzada. Estos valores también se han guardado en el objeto `res.daag`, así como los valores predichos para cada observación.

```
res.daag$acc.cv

## [1] 0.8588
```

Y los valores predichos mediante validación cruzada.

```
# mostramos sólo los 6 primeros
head(res.daag$cvhat)

## [1] 0.34411 0.03383 0.04733 0.09635 0.31998 0.59962
```

A la vista de la alta tasa de clasificaciones correctas calculada mediante validación cruzada, de la no existencia de valores influyentes y del escaso porcentaje de residuos significativos, concluimos que el modelo no presenta falta de ajuste ni tampoco problemas de sobreajuste.

Apéndice A

Ajuste de modelos logit

La estimación de un modelo logit se obtiene maximizando la función de verosimilitud de los datos respecto de los parámetros. Para los modelos logit la verosimilitud es una función cóncava y por tanto los estimadores máximo verosímiles existen y son únicos. En la práctica, se maximiza el logaritmo de la verosimilitud. El cálculo de los estimadores es más complejo que para los modelos lineales y requieren métodos de aproximación iterativa como el de Newton-Raphson. A continuación explicamos los diferentes métodos del cálculo de las estimaciones y cómo implementarlo en R. Los códigos que se reproducen para el método de Newton-Raphson y la estimación por métodos de optimización generales se han adaptado del libro "An R Companion to Applied Regression" (Fox and Weisberg, 2011).

A.1. Método de Newton-Rapshon

Suponemos una función $g(\beta)$ dónde β es un escalar, y queremos encontrar el valor de β que maximiza $g(\beta)$ dado un valor actual b_t . Entonces la serie de Taylor que aproxima $g(\beta)$ es

$$g(\beta) = g(b_t) + (\beta - b_t)\frac{dg(\beta)}{d\beta} + \frac{1}{2}(\beta - b_t)^2\frac{d^2g(\beta)}{d\beta^2} + residuo \tag{A.1}$$

Diferenciando respecto a β e igualando a 0 se obtiene

$$(\beta - b_t) = \left[-\frac{d^2g(\beta)}{d\beta^2}\right]^{-1}\left[\frac{dg(\beta)}{d\beta}\right]$$

y se actualiza b_{t+1} a

$$b_{t+1} = b_t + \left[-\frac{d^2g(\beta)}{d\beta^2}\right]^{-1}\left[\frac{dg(\beta)}{d\beta}\right]$$

El proceso se repite hasta que se estabiliza el valor de b_{t+1}

La generalización del algoritmo para el caso de más de un parámetro comienza con la versión vectorizada de la ecuación (A.1)

$$g(\boldsymbol{\beta}) = g(\mathbf{b_t}) + (\boldsymbol{\beta} - \mathbf{b_t})^t\frac{\partial g(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} + \frac{1}{2}(\boldsymbol{\beta} - \mathbf{b_t})^t\frac{\partial^2 g(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}(\partial \boldsymbol{\beta})^t}(\boldsymbol{\beta} - \mathbf{b_t}) + residuo$$

Diferenciando de nuevo, se tiene que la actualización del vector $\mathbf{b_{t+1}}$ es

$$\mathbf{b_{t+1}} = \mathbf{b_t} + \left[-\frac{\partial^2 g(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \left(\partial \boldsymbol{\beta}\right)^t} \right]^{-1} \left[\frac{\partial g(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \right] \tag{A.2}$$

El vector de las primeras derivadas se denomina *gradiente*. Al vector de las segundas derivadas multiplicado por -1 se le denomia *matriz Hessiana*

Con $g\left(\boldsymbol{\beta}\right)$ la log-verosimilitud de los datos para una regresión logística binaria, el gradiente y la matriz Hessiana son

$$\begin{aligned} \text{gradiente} &= \mathbf{X}'\left(\mathbf{y} - \mathbf{p_t}\right) \\ &= \sum \left(y_i - p_{it}\right) \mathbf{x}_i \\ \text{Hessiana} &= \left(\mathbf{X}'\mathbf{V}_t\mathbf{X}\right)^{-1} \end{aligned}$$

$\mathbf{X}$ es la matriz del modelo, con $\mathbf{x}_i$ la fila i-ésima, $\mathbf{y}$ es el vector de respuesta,que contiene 0's y 1's, con y_i es el elemento i-ésimo. $\mathbf{p}_t$ es el vector de las probabiliades ajustadas en la última iteración, con

$$p_{it} = \frac{1}{1 + \exp\left(-\mathbf{x}_i'\mathbf{b}_t\right)}$$

V_t es una matriz diagonal, con elementos $p_{it}\left(1 - p_{it}\right)$. Entonces la ecuación (A.2) sería

$$\mathbf{b}_{t+1} = \mathbf{b_t} + \left(\mathbf{X}'\mathbf{V_t}\mathbf{X}\right)^{-1} \mathbf{X}'(\mathbf{y} - \mathbf{p}_t)$$

y se repite el proceso hasta que la diferencia entre el vector $\mathbf{b_t}$ y $\mathbf{b_{t-1}}$ es lo suficientemente cercana a 0 o se alcanza el número máximo de iteraciones especificado.

Ahora ya podemos construir una pequeña función en R para ajustar una regresión logística binaria mediante el método de Newton-Rapshon.

```
lreg1 <- function(X, y, max.iter = 10, tol = 1e-06, verbose = FALSE) {
    # X es el model matrix, si hay variables categóricas hay que introducir la
    # matriz con las variables ya codificadas.  y es el vector de respuestas
    # de 0,1

    X <- cbind(1, X)  # se añade una columna de unos, para el intercept
    b <- b.last <- rep(0, ncol(X))
    it <- 1

    while (it <= max.iter) {
        if (verbose)
            cat("\niteration = ", it, ",", b)
        p <- as.vector(1/(1 + exp(-X %*% b)))  # prob ajustadas
        V <- diag(p * (1 - p))
        var.b <- solve(t(X) %*% V %*% X)  # inversa de la matriz hessiana, en la diagonal
        # se tienen los valores de la varianza de los parámetros
        b <- b + var.b %*% t(X) %*% (y - p)
        if (max(abs(b - b.last)/(abs(b.last) + 0.001 * tol)) < tol)
```

```
            break
        b.last <- b
        it <- it + 1
    }
    if (verbose)
        cat("\n")
    if (it > max.iter)
        warning("número máximo de iteraciones excedido")
    list(coefficients = as.vector(b), var = var.b, iterations = it)
}
```

Podemos crear unos datos simulados y comparar el resultado de nuestra función `lreg1` con la obtenida por `glm`

1. Simulamos los datos

```
set.seed(234)  # para que siempre salgan los mismos valores simulados
# simulamos 40 valores de ensayos de bernoulli con probabilidad de éxito
# 0.4
y <- rbinom(40, 1, 0.4)
head(y)

## [1] 1 1 0 1 0 1

# 40 valores de una normal con media 2 y desviación típica 5
X <- rnorm(40, 2, 5)
head(X)

## [1]  3.027  7.081  4.045 -1.526  3.178  3.726
```

2. Estimación mediante el método de Newton-Raphson `lreg1`

```
lreg1(X, y)

## $coefficients
## [1] -0.06321 -0.09794
##
## $var
##                       X
##    0.116851 -0.009182
## X -0.009182  0.006949
##
## $iterations
## [1] 4

lreg1(X, y, verbose = TRUE)  # para ver el ajuste en cada iteración
```

```
##
## iteration =  1 , 0 0
## iteration =  2 , -0.05999 -0.09435
## iteration =  3 , -0.0632 -0.09792
## iteration =  4 , -0.06321 -0.09794
## $coefficients
## [1] -0.06321 -0.09794
##
## $var
##                      X
##    0.116851 -0.009182
## X -0.009182  0.006949
##
## $iterations
## [1] 4
```

3. Estimación mediante la función `glm`

```
res <- glm(y ~ X, family = binomial)
res

##
## Call:  glm(formula = y ~ X, family = binomial)
##
## Coefficients:
## (Intercept)            X
##     -0.0632      -0.0979
##
## Degrees of Freedom: 39 Total (i.e. Null);  38 Residual
## Null Deviance:     55.1
## Residual Deviance: 53.6  AIC: 57.6
```

Obteniendo la misma estimación en ambos casos

A.2. Estimación por métodos de optimización generales.

Otra aproximación para ajustar un modelo de regresión logística es usar un "*optimizador*" de propósito general *(Nocedal and Wright, 2006),* para maximizar la función de log-verosimilitud

$$\log\, L = \sum y_i \log p_i + (1 - y_i) \log (1 - p_i)$$

dónde en cada iteración t, las probabilidades ajustadas para la observación i son: $p_i = 1/\left[1 + \exp\left(-\mathbf{x_i'b_t}\right)\right]$

Los optimizadores evalúan el vector de derivadas parciales de la log-verosimilitud (gradiente) y en cada iteración se busca conseguir que el gradiente esté lo suficientemente cercano a 0. Dependiendo

del "optimizador" se puede utilizar la expresión analítica del gradiente o una aproximación numérica. En R existen varias funciones para optimizar funciones, `optim` es la función por defecto. Con `optim` se **minimiza** una función objetivo y por tanto vamos a considerar la log-verosimilitud cambiada de signo.

La función `optim` permite varios métodos de optimización, como el `BFGS` (quasi-Newton método) o el `CG` (Gradiente conjugado). Una descripción más detallada de la función se puede encontrar en la ayuda de `optim` y en (Baquela and Redchuk, 2013)

- El primer argumento de `optim` es para indicar los valores iniciales de los parámetros, ceros en nuestro caso.
- El segundo argumento es la función a minimizar, que será la log-verosimilitud negativa. El tercero es el gradiente, si no se le da un valor, `optim` lo calcula numéricamente.
- Al especificar `hessian=TRUE`, `optim` calcula la matriz Hessiana, cuya inversa nos da una estimación de la matriz de covarianzas de los parámetros estimados
- Por último hay que especificar la la matriz del modelo X, matriz con la primera columnas de 1's para la constante y los valores de las variables predictoras y el vector de respuestas y.

Nos creamos la función `lreg2` dónde especificamos la log-verosimilitud negativa, el gradiente y ejecutamos `optim` para que nos de la estimación.

```
lreg2 <- function(X, y, method = "BFGS") {
    X <- cbind(1, X)  # si hay variables categóricas se deben introducir
    # las variables auxiliares correspondientes
    negLogL <- function(b, X, y) {
        p <- as.vector(1/(1 + exp(-X %*% b)))
        -sum(y * log(p) + (1 - y) * log(1 - p))
    }
    grad <- function(b, X, y) {
        p <- as.vector(1/(1 + exp(-X %*% b)))

        -colSums((y - p) * X)
    }
    result <- optim(rep(0, ncol(X)), negLogL, gr = grad, hessian = TRUE, method = method,
        X = X, y = y)
    list(coefficients = result$par, var = solve(result$hessian), deviance = 2 *
        result$value, converged = result$convergence == 0)
}
```

```
lreg2(X, y)

## $coefficients
## [1] -0.06324 -0.09793
##
## $var
##           [,1]      [,2]
```

```
## [1,]  0.116851 -0.009182
## [2,] -0.009182  0.006949
##
## $deviance
## [1] 53.62
##
## $converged
## [1] TRUE
```

A.3. Estimación por mínimos cuadrados iterativamente reponderados

La función `glm` en R utiliza la estimación por mínimos cuadrados iterativamente reponderados, IWLS o IRLS por sus siglas en inglés.

IWLS parte de una aproximación local cuadrática a la función de log-verosimilitud, la maximización de esta aproximación se realiza mediante un modelo lineal ponderado. Suponiendo $\boldsymbol{\beta}^{(t)}$ el vector que contiene la estimación de los parámetros del GLM en la iteración t . Entonces $\eta_i^{(t)} = \mathbf{x}'\boldsymbol{\beta}^{(t)}$ es el predictor lineal para la observación i-ésima y $\mu_i^{(t)} = g^{-1}(\eta_i^{(t)})$ los valores ajustados (probabilidades ajustadas en el caso de regresión logística y usando la función logit como función de enlace g), la función de la varianza sería $\nu_i^{(t)} = \text{Var}\left(\mu_i^{(t)}\right)/\phi$, dónde ϕ es el parámetro de dispersión o escala (2.1.1). Tenemos unos valores de respuesta intermedios o de trabajo *(working response)*. Se les llama de trabajo porque cambian en cada iteración.

$$z_i^{(t)} = \eta_i^{(t)} + (y_i - \mu_i^{(t)}) \left(\frac{\partial \eta_i}{\partial \mu_i}\right)^{(t)}$$

y unos ponderaciones de trabajo

$$w_i^{(t)} = \frac{1}{c_i \nu_i^{(t)} \left[\left(\frac{\partial \eta_i}{\partial \mu_i}\right)^{(t)}\right]^2}$$

Dónde c_i son constantes fijas (para la familia binomial $c_i = n_i^{-1}$)

El algoritmo ajusta una regresión por mínimos cuadrados ponderada de $z^{(t)}$ sobre los xs predictores lineales, minimizando la suma de cuadrados ponderada $\sum_{i=1}^{n} w_i \left(z_i - \mathbf{x}_i'\boldsymbol{\beta}\right)^2$, dónde $\mathbf{x}_i'$ es la fila i-ésima de la matriz de los $\mathbf{X}$ regresores (incluyendo columna de unos). De esta forma se obtienen unos nuevos parámetros $\boldsymbol{\beta}^{(t+1)}$. Este proceso continua hasta que se estabilizan los parámetros, obteniendo la estimación máximo verosímil $\hat{\boldsymbol{\beta}}$.

La estimación asíntotica de la matriz de covarianzas se obtiene en la última iteración del procedimiento IWLS de la forma

$$\widehat{\text{Var}}\left(\hat{\boldsymbol{\beta}}\right) = \hat{\phi}\left(\mathbf{X}'\mathbf{W}\mathbf{X}\right)^{-1}$$

con $\mathbf{W} = \text{diag}\{w_i\}$ y $\hat{\phi}$ [1]es el estadístico de Pearson dividido por los grados de libertad residuales del modelo.

[1] ϕ se estima cuando se tiene sobredispersión, lo cual puede ocurrir, por ejemplo, cuando se ajustan datos binomiales, pero no si los datos son binarios.

En el caso de regresión logística sería.

$$\begin{aligned}
\mu_i^{(t)} &= \left[1 + \exp\left(-\eta_i^{(t)}\right)\right]^{-1} \\
\nu &= \mu_i^{(t)}\left(1 - \mu_i^{(t)}\right) \\
\left(\frac{\partial \eta_i}{\partial \mu_i}\right)^{(t)} &= \frac{1}{\mu_i^{(t)}\left(1 - \mu_i^{(t)}\right)} \\
z_i^{(t)} &= \eta_i^{(t)}(y_i - \mu_i^{(t)})/\nu_i^{(t)} \\
w_i^{(t)} &= N_i \nu_i
\end{aligned}$$

con N_i el número de ensayos asociados a la observación i-ésima y $\phi = 1$ para la familia binomial.

A continuación vemos un ejemplo de cómo se ajustaría un modelo de regresión logística binaria con R utilizando este algoritmo. Básicamente es el mismo procedimiento que utiliza la función `glm` en R, aunque `glm` es más compleja ya que incorpora que se pueda utilizar la notación tipo `formula` de R, la codificación automática de variables categóricas a sus variables auxiliares correspondientes e incorpora también otro tipo de familias distintas de la binomial.

```
lreg3 <- function(X, y, max.iter = 10, tol = 1e-06, verbose = FALSE) {
    # X es el model matrix y es el vector de respuestas de 0,1

    X <- cbind(1, X)  # se añade columna de unos para estimar el intercept
    # si hay variables categóricas se deben introducir las variables
    # auxiliares correspondientes
    b <- b.last <- rep(0, ncol(X))  # se inicializa el vector de parámetros en 0's
    it <- 1
    while (it <= max.iter) {
        if (verbose)
            cat("\niteration = ", it, ",", b)
        p <- as.vector(1/(1 + exp(-X %*% b)))  # prob ajustadas
        v <- (p * (1 - p))
        z <- (X %*% b) + (y - p)/v
        # N ha de ser el número de pruebas binomiales en cada i
        N <- 1
        w <- N * v
        V <- diag(v)
        var.b <- solve(t(X) %*% V %*% X)  # matriz de varianzas/covarianzas de los parámetros
        result <- lm(z ~ X[, -1], weights = w)
        b <- result$coefficients
        if (max(abs(b - b.last)/(abs(b.last) + 0.001 * tol)) < tol)
            break
        b.last <- b
        it <- it + 1
    }
    if (verbose)
        cat("\n")
```

```
    # if (it > max.iter) warning('número máximo de iteraciones excedido')
    list(coefficients = as.vector(b), var = var.b, iterations = it)

}
```

La estimación mediante mínimos cuadrados iterativamente reponderados sería

```
lreg3(X, y)

## $coefficients
## [1] -0.06321 -0.09794
##
## $var
##                      X
##    0.116851 -0.009182
## X -0.009182  0.006949
##
## $iterations
## [1] 4
```

Apéndice B

Devianza para datos agrupados y no agrupados

La función de verosimilitud en un proceso binomial con y_i el número de éxitos en m_i $i = 1 \dots n$, y un sólo predictor X con valores x_i

$$y_i|x_i \rightsquigarrow \text{Bin}(m_i, \theta(x_i))$$

con $\theta(x_i)$ la estimación del modelo logístico en cada x_i

$$\theta(x_i) = \frac{1}{1 + \exp\left(-\beta_o - \beta_1 x_i\right)}$$

es de la forma

$$L = \prod_{i=1}^{n} P\left(Y_i = y_i|x_i\right) = \prod_{i=1}^{n} \binom{m_i}{y_i} \theta(x_i)^{y_i} \left(1 - \theta(x_i)\right)^{m_i - y_i}$$

Y la log-verosimilitud

$$\begin{aligned} \log(L) &= \sum_{i=1}^{n} \left[\log \binom{m_i}{y_i} + \log\left(\theta(x_i)^{y_i}\right) + \log\left(\left(1 - \theta(x_i)\right)^{m_i - y_i}\right) \right] \\ &= \sum_{i=1}^{n} \left[y_i \log\left(\theta(x_i)\right) + (m_i - y_i) \log\left(1 - \theta(x_i)\right) + \log \binom{m_i}{y_i} \right] \\ &= \sum_{i=1}^{n} \left[y_i \log\left(\frac{\theta(x_i)}{1 - \theta(x_i)}\right) + m_i \log\left(1 - \theta(x_i)\right) + \log \binom{m_i}{y_i} \right] \end{aligned} \tag{B.1}$$

En la regresión logística el concepto de suma de cuadrados se sustituye por el de *devianza* . La devianza asociada a un modelo (M) se basa en comparar la log-verosimilitud bajo dicho modelo con la log-verosimilitud bajo el modelo saturado (S). La devianza se define entonces como dos veces la diferencia entre estas dos log-verosimilitudes.

En el modelo saturado (S) se estima $\theta(x_i)$ por la proporción observada de éxitos, es decir por y_i/m_i. En el modelo de regresión logística (M) se estima $\theta(x_i)$ por $\widehat{\theta}_M(x_i)$ y los valores predichos de y_i se estiman por $\widehat{y}_i = m_i \widehat{\theta}_M(x_i)$ con lo que $\widehat{\theta}_M(x_i) = \dfrac{\widehat{y}_i}{m_i}$

Así, utilizando la ecuación B.1 y sustituyendo $\theta(x_i)$ en la log-verosimilitud, tanto del modelo saturado (S) como del modelo de regresión logística (M), la devianza del modelo M viene dada por.

$$\begin{aligned} G^2 &= 2\left[\log\left(L_S\right) - \log\left(L_M\right)\right] \\ &= 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{m_i}\right) + (m_i - y_i)\log\left(1 - \frac{y_i}{m_i}\right)\right] \\ &\quad -2\sum_{i=1}^{n}\left[y_i \log\left(\frac{\widehat{y}_i}{m_i}\right) + (m_i - y_i)\log\left(1 - \frac{\widehat{y}_i}{m_i}\right)\right] \\ &= 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\widehat{y}_i}\right) + (m_i - y_i)\,\log\left(\frac{m_i - y_i}{m_i - \widehat{y}_i}\right)\right] \end{aligned} \tag{B.2}$$

Bajo la hipótesis nula de que el modelo (M) es adecuado y los m_i son suficientemente grandes, G^2 se distribuye aproximadamente como una χ^2_{n-p-1}, dónde n = es el número de datos o muestras binomiales y p =el número de variables predictoras del modelo, ($p+1$ es el número de parámetros estimados). La diferencia entre la devianza de dos modelos anidados también sigue una distribución χ^2 y es la base de los procedimientos de selección de modelos.

En el caso de datos **no agrupados**, se tiene que todos los m_i son iguales a uno y por tanto $\theta_S(x_i) = y_i$. Bajo el modelo de regresión $\hat{\theta}_M(x_i) = \hat{y}_i$. Calculando de nuevo la devianza.

$$\begin{aligned} G^2 &= 2\left[\log\left(L_S\right) - \log\left(L_M\right)\right] \\ &= 2\sum_{i=1}^{n}\left[y_i \log\left(y_i\right) + (1 - y_i)\log\left(1 - y_i\right)\right] \\ &\quad -2\sum_{i=1}^{n}\left[y_i \log\left(\hat{y}_i\right) + (1 - y_i)\log\left(1 - \hat{y}_i\right)\right] \\ &= -2\sum_{i=1}^{n}\left[y_i \log\left(\hat{y}_i\right) + (1 - y_i)\log\left(1 - \hat{y}_i\right)\right] \end{aligned} \tag{B.3}$$

Si nos fijamos en los términos asociados a $\log\left(L_s\right)$ cuando $y_i = 1$ el primer término es 0 y el segundo es el límite de

$$\lim_{z\to 0}\; z\log(z)$$

con $z = 1 - y_i$. Este límite se resuelve utilizando la regla de L'Hopital con $f(z) = \log(z)$ y $g(z) = 1/z$

$$\lim_{z\to 0} f(z) = -\infty, \lim_{z\to 0} g(z) = \infty$$

se tiene que

$$\lim_{z\to 0}\left(z\;\log(z)\right) = \lim_{z\to 0}\frac{f(z)}{g(z)} = \lim_{z\to 0}\frac{f'(z)}{g'(z)} = \lim_{z\to 0}\frac{z^{-1}}{-z^{-2}} = \lim_{z\to 0} -z = 0$$

Cuando $y_i = 0$ el segundo término es 0 y el primero es el límite

$$\lim_{y\to 0}\; y\;\log(y)$$

que como ya hemos visto tiende a 0. Con lo que se tiene que en el caso de datos no agrupados, la devianza sólo depende de $\log(L_M)$, es decir, de la log-verosimilitud del modelo ajustado.

En este caso, la devianza no es una medida de la bondad del ajuste del modelo M, y tampoco sigue una distribución χ^2. No obstante, la diferencia entre las devianzas de dos modelos ajustados si se distribuye como una χ^2 y se puede utilizar para seleccionar el modelo con mejor ajuste.

Bibliografía

Agresti, A. (2002): *Categorical Data Analysis*. WILEY, second edn.

Baquela, E., and A. Redchuk (2013): *Optimización matemática con R. Volumen I. Introducción al modelado y resolución de problemas*. Bubok Publishing S.L.

Bartoń, K. (2013): *MuMIn: Multi-model inference* R package version 1.9.5.

Belsley, D. A., E. Kuh, R. E. Welsch, et al. (2004): *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity* . WILEY.

Burnham, K. P., and D. R. Anderson (2002): *Model Selection and Multimodel Inference: a practical information-theoretic approach*. Springer, New York, 2nd edn.

Canty, A., and B. Ripley (2012): *Bootstrap R (S-Plus) Functions.* R package version 1.3-7.

Cook, R. D., and S. Weisberg (1982): *Residuals and Influence in Regression*. New York: Chapman and Hall.

Cortes, C., and V. Vapnik (1995): "Support-vector networks," *Machine Learning*, 20(3), 273–297.

Cover, T. M., and P. E. Hart (1967): "Nearest neighbor pattern classification ," *Information Theory, IEEE Transactions on*, 13(1), 21–27.

Cox, D., and E. Snell (1989): *The Analysis of Binary Data*, Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman and Hall, second edn.

Crawley, M. J. (2007): *The R Book*. WILEY.

Dahl, D. B. (2012): *xtable: Export tables to LaTeX or HTML.* R package version 1.7-0.

Davison, A. (1997): *Bootstrap methods and their application*. Cambridge University Press, Cambridge New York, NY, USA.

Efron, B. (1974): " Cross-validation choice and assessment of statistical predictions (with Discussion)," *Journal of the Royal Statistical Society. Series B (Methodological)*, 36, 111–147.

——— (1979): "Bootstrap methods: Another look at the jackknife," *The Annals of Statistic*, 7(1), 1–26.

Elff, M. (2012): *memisc: Tools for Management of Survey Data, Graphics, Programming, Statistics, and Simulation* R package version 0.95-38.

Everitt, B. S., and T. Hothorn (2012): *HSAUR: A Handbook of Statistical Analyses Using R* R package version 1.3-0.

Fisher, R. (1936): "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, 7, 179–188.

Fox, J. (2003): "Effect Displays in R for Generalised Linear Models," *Journal of Statistical Software*, 8(15), 1–27.

Fox, J., and G. Monnette (1992): "Generalized collinearity diagnostics," *Journal of the American Statistical Association*, 87(417), 178–183.

Fox, J., and S. Weisberg (2011): *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, second edn.

Franco-Nicolás, M., and J. M. Vivo-Molina (2007): *Análisis de Curvas Roc : Principios básicos y aplicaciones*. Editorial La Muralla, Madrid.

Friedman, J., T. Hastie, and R. Tibshirani (2010): "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistcal Software*, 33(1), 1–22.

Harrel, F. E. J. (2013): *rms: Regression Modeling Strategies* R package version 3.6-3.

Harrell, F. (2001): *Regression modeling strategies : with applications to linear models, logistic regression and survival analysis*. Springer, New York etc.

Hauck, W., and A. Donner (1977): "Wald's Test as Applied to Hypotheses in Logit Analysis.," *Journal of the American Statistical Association*, 72.

Hosmer, D. W., T. Hosmer, S. Le Cessie, and S. Lemeshow (1997): "A comparison of goodness-of-fit tests for the logistic regression model," *Statistics in Medicine*, 16, 965–980.

Hosmer, D. W., B. Jovanovic, and S. Lemeshow (1989): "Best Subsets Logistic Regression," *Biometrics*, 45, 1265–1270.

Hosmer, D. W., and S. Lemeshow (2000): *Applied Logistic Regression*. WILEY.

Le Cessie, S., and J. Houwelingen (1991): "A goodness-of-fit test for binary data based on smoothing residuals," *Biometrics*, 47, 1267—1282.

Leisch, F. (2002): "Sweave, Part I: Mixing R and LaTeX," *R News*, 2(3), 28–31.

Maindonald, J., and W. J. Braun (2013): *DAAG: Data Analysis And Graphics data and functions* R package version 1.16.

McCullagh, P., and J. Nelder (1989): *Generalized linear models*. Chapman and Hall, London New York.

McFadden, D. (1974): "Conditional Logit Analysis of Qualitative Choice Behavior," *In: Frontiers in Economics, P. Zarembka*, pp. 105–142.

McLeod, A., and X. Changjiang (2011): *bestglm: Best Subset GLM* R package version 0.33.

Muenchen, R. A. (2009): *R for SAS and SPSS Users*. Springer.

Nagelkerke, N. J. D. (1991): "A note on a general definition of the coefficient of determination," *Biometrika*, 78(3), 691–692.

Nocedal, J., and S. Wright (2006): *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering. Springer, 2nd edn.

R Core Team (2013a): *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...*R package version 0.8-53.

——— (2013b): *R: A Language and Environment for Statistical Computing* R Foundation for Statistical Computing, Vienna, Austria.

Sheather, S. (2009): *A Modern Approach to Regression with R (Springer Texts in Statistics)*. Springer.

Silva, L. C., and I. M. Barroso (2004): *Regresión Logística*. La Muralla.

Sing, T., O. Sander, N. Beerenwinkel, and T. Lengauer (2005): "ROCR: visualizing classifier performance in R," *Bioinformatics*, 21(20), 7881.

Thompson, L. A. (2007): *S-PLUS (and R) Manual to Acccompany Agresti's Categorical Data Analysis*.

Tibshirani, R. (1996): "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1), 267–288.

Torgo, L. (2011): *Data mining with R : learning with case studies*. Chapman & Hall/CRC, Boca Raton.

Venables, W., and B. Ripley (2002): *Modern Applied Statistics with S*. Springer, fourth edition edn.

Venables, W. , Smith, D. and R Development Core Team (2012): *An Introduction to R*.

Venzon, D., and S. Moolgavkar (1988): "A Method for Computing Profile-Likelihood-Based Confidence Intervals," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 37(1), 87–94.

Williams, D. (1987): "Generalized linear model diagnostics using the deviance and single case deletions," *Applied Statistics*, 36, 181–191.

Xie, Y. (2012): *knitr: A general-purpose package for dynamic report generation in R*. R package version 0.5.