Regresión logística simple y múltiple

Joaquín Amat Rodrigo j.amatrodrigo@gmail.com

Agosto, 2016

Tabla de contenidos

Versión PDF: Github

Regresión logística simple

Introducción

La Regresión Logística Simple, desarrollada por David Cox en 1958, es un método de regresión que permite estimar la probabilidad de una variable cualitativa binaria en función de una variable cuantitativa. Una de las principales aplicaciones de la regresión logística es la de clasificación binaria, en el que las observaciones se clasifican en un grupo u otro dependiendo del valor que tome la variable empleada como predictor. Por ejemplo, clasificar a un individuo desconocido como hombre o mujer en función del tamaño de la mandíbula.

Es importante tener en cuenta que, aunque la regresión logística permite clasificar, se trata de un modelo de regresión que modela el logaritmo de la probabilidad de pertenecer a cada grupo. La asignación final se hace en función de las probabilidades predichas.

La existencia de una relación significativa entre una variable cualitativa con dos niveles y una variable continua se puede estudiar mediante otros test estadísticos tales como t-test o ANOVA (un ANOVA de dos grupos es equivalente al *t-test*). Sin embargo, la regresión logística permite además calcular la probabilidad de que la variable dependiente pertenezca a cada una de las dos categorías en función del valor que adquiera la variable independiente. Supóngase que se quiere estudiar la relación entre los niveles de colesterol y los ataques de corazón. Para ello, se mide el colesterol de un grupo de personas y durante los siguientes 20 años se monitoriza que individuos han sufrido un ataque. Un *t-test* entre los niveles de colesterol de las personas que han sufrido ataque *vs* las que no lo han sufrido permitiría contrastar la hipótesis de que el colesterol y los ataques al corazón están asociados. Si además se desea conocer la probabilidad de que una persona con un determinado nivel de colesterol sufra un infarto en los próximos 20 años, o poder conocer cuánto tiene que reducir el colesterol un paciente para no superar un 50% de probabilidad de padecer un infarto en los próximos 20 años, se tiene que recurrir a la regresión logística.

¿Por qué regresión logística y no lineal?

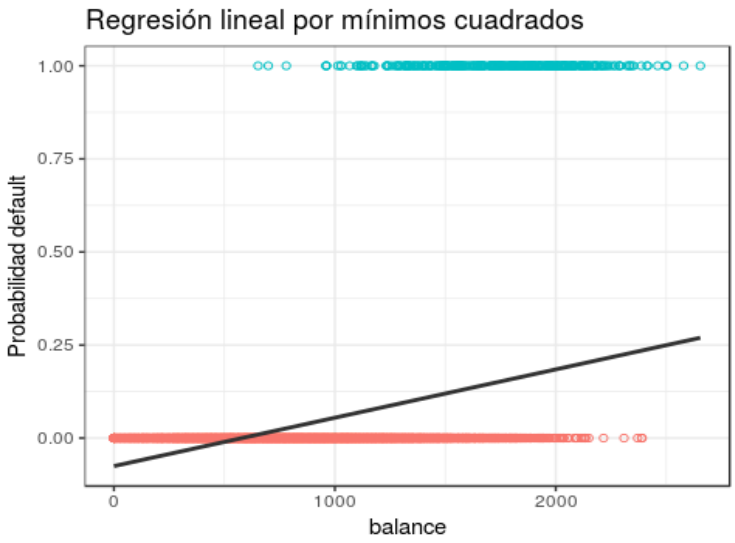
Si una variable cualitativa con dos niveles se codifica como *1* y *0*, matemáticamente es posible ajustar un modelo de regresión lineal por mínimos cuadrados $\beta_0 + \beta_1 x$. El problema de esta aproximación es que, al tratarse de una recta, para valores extremos del predictor, se obtienen valores de Y menores que 0 o mayores que 1, lo que entra en contradicción con el hecho de que las probabilidades siempre están dentro del rango [0,1].

En el siguiente ejemplo se modela la probabilidad de fraude por impago (*default*) en función del balance de la cuenta bancaria (*balance*).

```
library(tidyverse)
library(ISLR)
datos <- Default

# Se recodifican los niveles No, Yes a 1 y 0
datos <- datos %>%
         select(default, balance) %>%
         mutate(default = recode(default,
                                 "No"  = 0,
                                 "Yes" = 1))
head(datos)
```

```
##   default   balance
## 1       0  729.5265
## 2       0  817.1804
## 3       0 1073.5492
## 4       0  529.2506
## 5       0  785.6559
## 6       0  919.5885
```

```
# Ajuste de un modelo lineal por mínimos cuadrados.
modelo_lineal <- lm(default ~ balance, data = datos)

# Representación gráfica del modelo.
ggplot(data = datos, aes(x = balance, y = default)) +
  geom_point(aes(color = as.factor(default)), shape = 1) +
  geom_smooth(method = "lm", color = "gray20", se = FALSE) +
  theme_bw()  +
  labs(title = "Regresión lineal por mínimos cuadrados",
       y = "Probabilidad default") +
  theme(legend.position = "none")
```



Al tratarse de una recta, si por ejemplo, se predice la probabilidad de *default* para alguien que tiene un balance de *10000*, el valor obtenido es mayor que 1.

```
predict(object = modelo_lineal, newdata = data.frame(balance = 10000))
```

```
##       1
## 1.22353
```

Para evitar estos problemas, la regresión logística transforma el valor devuelto por la regresión lineal ($\beta_0 + \beta_1 X$) empleando una función cuyo resultado está siempre comprendido entre *0* y *1*. Existen varias funciones que cumplen esta descripción, una de las más utilizadas es la función logística (también conocida como función sigmoide):

$$Función\ sigmoide = \sigma(x) = \frac{1}{1+e^{-x}} \qquad (1)$$

Para valores de x muy grandes positivos, el valor de e^{-x} es aproximadamente 0 por lo que el valor de la función sigmoide es 1. Para valores de x muy grandes negativos, el valor e^{-x} tiende a infinito por lo que el valor de la función sigmoide es 0.

Sustituyendo la x de la ecuación 1 por la función lineal ($\beta_0 + \beta_1 X$) se obtiene que:

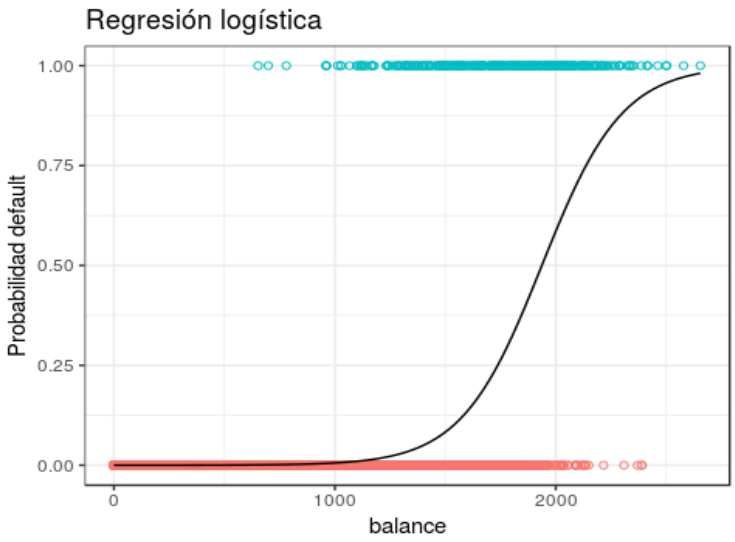
$$P(Y = k|X = x) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} = \frac{1}{\frac{e^{\beta_0+\beta_1 X}}{e^{\beta_0+\beta_1 X}}+\frac{1}{e^{\beta_0+\beta_1 X}}} = \frac{1}{\frac{1+e^{\beta_0+\beta_1 X}}{e^{\beta_0+\beta_1 X}}} = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

donde $Pr(Y = k|X = x)$ puede interpretarse como: la probabilidad de que la variable cualitativa Y adquiera el valor k (el nivel de referencia, codificado como 1), dado que el predictor X tiene el valor x.

Esta función, puede ajustarse de forma sencilla con métodos de regresión lineal si se emplea su versión logarítmica, obteniendo lo que se conoce como *LOG of ODDs*

$$ln(\frac{p(Y = k|X = x)}{1 - p(Y = k|X = x)}) = \beta_0 + \beta_1 X$$

```
# Ajuste de un modelo logístico.
modelo_logistico <- glm(default ~ balance, data = datos, family = "binomial")

# Representación gráfica del modelo.
ggplot(data = datos, aes(x = balance, y = default)) +
  geom_point(aes(color = as.factor(default)), shape = 1) +
  stat_function(fun = function(x){predict(modelo_logistico,
                                          newdata = data.frame(balance = x),
                                          type = "response")}) +
  theme_bw() +
  labs(title = "Regresión logística",
       y = "Probabilidad default") +
  theme(legend.position = "none")
```




```
# Con geom_smooth se puede obtener el gráfico directamente.
ggplot(data = datos, aes(x = balance, y = default)) +
  geom_point(aes(color = as.factor(default)), shape = 1) +
  geom_smooth(method = "glm",
              method.args = list(family = "binomial"),
              color = "gray20",
              se = FALSE) +
  theme_bw() +
  theme(legend.position = "none")
```

Concepto de ODDS o razón de probabilidad, ratio de ODDS y logaritmo de ODDS

En la regresión lineal simple, se modela el valor de la variable dependiente *Y* en función del valor de la variable independiente *X*. Sin embargo, en la regresión logística, tal como se ha descrito en la sección anterior, se modela la probabilidad de que la variable respuesta *Y* pertenezca al nivel de referencia 1 en función del valor que adquieran los predictores, mediante el uso de *LOG of ODDs*.

Supóngase que la probabilidad de que un evento sea verdadero es de *0.8*, por lo que la probabilidad de evento falso es de *1 - 0.8 = 0.2*. Los *ODDs* o *razón de probabilidad* de verdadero se definen como el ratio entre la probabilidad de evento verdadero y la probabilidad de evento falso $\frac{p}{q}$. En este caso los *ODDs* de verdadero son *0.8 / 0.2 = 4*, lo que equivale a decir que se esperan 4 eventos verdaderos por cada evento falso.

La trasformación de probabilidades a *ODDs* es monotónica, si la probabilidad aumenta también lo hacen los *ODDs*, y viceversa. El rango de valores que pueden tomar los *ODDs* es de [0,∞]. Dado que el valor de una probabilidad está acotado entre [0,1] se recurre a una trasformación *logit* (existen otras) que consiste en el logaritmo natural de los *ODDs*. Esto permite convertir el rango de probabilidad previamente limitado a [0,1] a [−∞,+∞].

p	odds	Log(odds)
0.001	0.001001	-6.906755
0.01	0.010101	-4.59512
0.2	0.25	-1.386294
0.3	0.4285714	-0.8472978
0.4	0.6666667	-0.4054651
0.5	1	0
0.6	1.5	0.4054651
0.7	2.333333	0.8472978
0.8	4	1.386294
0.9	9	2.197225
0.999	999	6.906755
0.9999	9999	9.21024

Los *ODDs* y el *logaritmo de ODDs* cumplen que:

- Si p(verdadero) = p(falso), entonces odds(verdadero) = 1
- Si p(verdadero) < p(falso), entonces odds(verdadero) < 1
- Si p(verdadero) > p(falso), entonces odds(verdadero) > 1
- A diferencia de la probabilidad que no puede exceder el 1, los *ODDs* no tienen límite superior.
- Si odds(verdadero) = 1, entonces logit(p) = 0
- Si odds(verdadero) < 1, entonces logit(p) < 0
- Si odds(verdadero) > 1, entonces logit(p) > 0
- La transformación *logit* no existe para p = 0

Ajuste del modelo

Una vez obtenida la relación lineal entre el logaritmo de los *ODDs* y la variable predictora *X*, se tienen que estimar los parámetros β_0 y β_1. La combinación óptima de valores será aquella que tenga la máxima verosimilitud (*maximum likelihood ML)*, es decir el valor de los parámetros β_0 y β_1 con los que se maximiza la probabilidad de obtener los datos observados.

El método de *maximum likelihood* está ampliamente extendido en la estadística aunque su implementación no siempre es trivial. En el este enlace se puede obtener una descripción detallada de cómo encontrar los valores β_0 y β_1 de máxima verosimilitud empleando el método de Newton.

Otra forma para ajustar un modelo de regresión logística es empleando descenso de gradiente. Si bien este no es el método de optimización más adecuado para resolver la regresión logística, está muy extendido en el ámbito del *machine learning* para ajustar otros modelos. En los Anexos puede encontrarse una implementación completa.

Evaluación del modelo

Existen diferentes técnicas estadísticas para calcular la significancia de un modelo logístico en su conjunto (*p-value* del modelo). Todos ellos consideran que el modelo es útil si es capaz de mostrar una mejora respecto a lo que se conoce como modelo nulo, el modelo sin predictores, solo con β_0. Dos de los más empleados son:

- **Wald chi-square**: está muy expandido pero pierde precisión con tamaños muestrales pequeños.
- **Likelihood ratio**: usa la diferencia entre la probabilidad de obtener los valores observados con el modelo logístico creado y las probabilidades de hacerlo con un modelo sin relación entre las variables. Para ello, calcula la significancia de la diferencia de residuos entre el modelo con predictores y el modelo nulo (modelo sin predictores). El estadístico tiene una distribución *chi-cuadrado* con grados de libertad equivalentes a la diferencia de grados de libertad de los dos modelos comparados. Si se compara respecto al modelo nulo, los grados de libertad equivalen al número de predictores del modelo generado. *En el libro Handbook for biological statistics se recomienda usar este*.

Para determinar la significancia individual de cada uno de los predictores introducidos en un modelo de regresión logística se emplea el estadístico *Z* y el test *Wald chi-test*. En **R**, este es el método utilizado para calcular los *p-values* que se muestran al hacer `summary()` del modelo.

Interpretación del modelo

A diferencia de la regresión lineal, en la que β_1 se corresponde con el cambio promedio en la variable dependiente *Y* debido al incremento en una unidad del predictor *X*, en regresión logística, β_1 indica el cambio en el logaritmo de *ODDs* debido al incremento de una unidad de *X*, o lo que es lo mismo, multiplica los *ODDs* por e^{β_1}. Dado que la relación entre *p(Y)* y *X* no es lineal, β_1 no se corresponde con el cambio en la probabilidad de *Y* asociada con el incremento de una unidad de *X*. Cuánto se incremente la probabilidad de *Y* por unidad de *X* depende del valor de *X*, es decir, de la posición en la curva logística en la que se encuentre.

Condiciones

- **Independencia**: las observaciones tienen que ser independientes unas de otras.
- **Relación lineal entre el logaritmo natural de *odds* y la variable continua**: patrones en forma de U son una clara violación de esta condición.
- La regresión logística no precisa de una distribución normal de la variable continua independiente.
- **Número de observaciones**: no existe una norma establecida al respecto, pero se recomienda entre 50 a 100 observaciones.

Predicciones

Una vez estimados los coeficientes del modelo logístico, es posible conocer la probabilidad de que la variable dependiente pertenezca al nivel de referencia, dado un determinado valor del predictor. Para ello se emplea la ecuación del modelo:

$$\hat{p}(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \quad (2)$$

Más adelante en este documento, se emplea la función `glm()` con `family="binomial"` para ajustar modelos de regresión logística. Esta función predice por defecto el *log(ODDs)* de la variable respuesta. Para obtener las probabilidades $P(y = 1)$ hay que aplicar la ecuación (2), donde el valor $e^{\hat{\beta}_0+\hat{\beta}_1 X}$ es el *log(ODDs)* devuelto por el modelo. Otra opción es indicar el argumento `type="response"` en la función `predict()`.

Convertir probabilidad en clasificación

Una de las principales aplicaciones de un modelo de regresión logística es clasificar la variable cualitativa en función de valor que tome el predictor. Para conseguir esta clasificación, es necesario establecer un *threshold* de probabilidad a partir de la cual se considera que la variable pertenece a uno de los niveles. Por ejemplo, se puede asignar una observación al grupo 1 si $\hat{p}\,(Y = 1|X) > 0.5$ y al grupo 0 si de lo contrario.

Ejemplo

Un estudio quiere establecer un modelo que permita calcular la probabilidad de obtener una matrícula de honor al final del bachillerato en función de la nota que se ha obtenido en matemáticas. La variable matrícula está codificada como 0 si no se tiene matrícula y 1 si se tiene.

```
matricula <- as.factor(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
                         0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
                         0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
                         0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                         1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
                         1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
                         1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
                         0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                         0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
                         0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
                         0, 0, 0, 0, 1, 0, 0, 0, 1, 1))
matematicas <- c(41, 53, 54, 47, 57, 51, 42, 45, 54, 52, 51, 51, 71, 57, 50, 43,
                 51, 60, 62, 57, 35, 75, 45, 57, 45, 46, 66, 57, 49, 49, 57, 64,
                 63, 57, 50, 58, 75, 68, 44, 40, 41, 62, 57, 43, 48, 63, 39, 70,
                 63, 59, 61, 38, 61, 49, 73, 44, 42, 39, 55, 52, 45, 61, 39, 41,
                 50, 40, 60, 47, 59, 49, 46, 58, 71, 58, 46, 43, 54, 56, 46, 54,
                 57, 54, 71, 48, 40, 64, 51, 39, 40, 61, 66, 49, 65, 52, 46, 61,
                 72, 71, 40, 69, 64, 56, 49, 54, 53, 66, 67, 40, 46, 69, 40, 41,
                 57, 58, 57, 37, 55, 62, 64, 40, 50, 46, 53, 52, 45, 56, 45, 54,
                 56, 41, 54, 72, 56, 47, 49, 60, 54, 55, 33, 49, 43, 50, 52, 48,
                 58, 43, 41, 43, 46, 44, 43, 61, 40, 49, 56, 61, 50, 51, 42, 67,
                 53, 50, 51, 72, 48, 40, 53, 39, 63, 51, 45, 39, 42, 62, 44, 65,
                 63, 54, 45, 60, 49, 48, 57, 55, 66, 64, 55, 42, 56, 53, 41, 42,
                 53, 42, 60, 52, 38, 57, 58, 65)
datos <- data.frame(matricula, matematicas)
head(datos, 4)
```

```
##   matricula matematicas
## 1         0          41
## 2         0          53
## 3         0          54
## 4         0          47
```

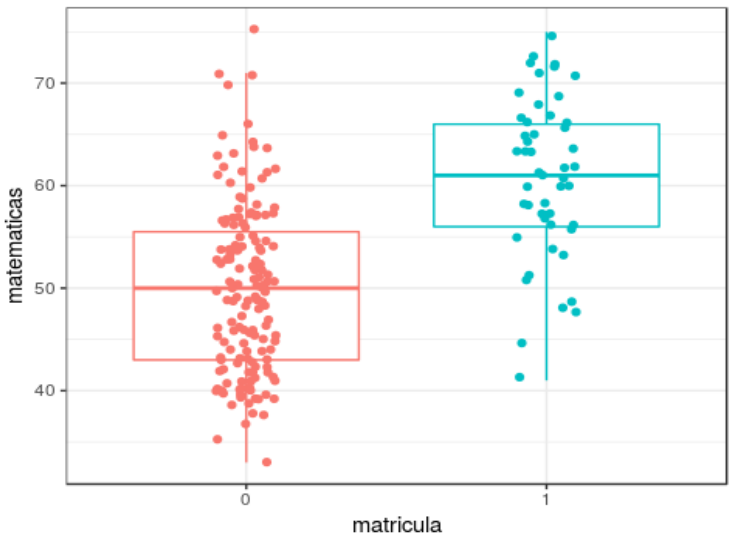
Representación de las observaciones

Representar las observaciones es útil para intuir si la variable independiente escogida está relacionada con la variable respuesta y, por lo tanto, puede ser un buen predictor.

```
library(ggplot2)
table(datos$matricula)
```

```
##
##   0   1
## 151  49
```

```
ggplot(data = datos, aes(x = matricula, y = matematicas, color = matricula)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "null")
```



Parece existir una diferencia entre la nota de las personas con matrícula y sin matrícula.

Generar el modelo de regresión logística

```
modelo <- glm(matricula ~ matematicas, data = datos, family = "binomial")
summary(modelo)
```

```
##
## Call:
## glm(formula = matricula ~ matematicas, family = "binomial", data = datos)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0332  -0.6785  -0.3506  -0.1565   2.6143
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.79394    1.48174  -6.610 3.85e-11 ***
## matematicas  0.15634    0.02561   6.105 1.03e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 222.71  on 199  degrees of freedom
## Residual deviance: 167.07  on 198  degrees of freedom
## AIC: 171.07
##
## Number of Fisher Scoring iterations: 5
```

El coeficiente estimado para la intersección es el valor esperado del *logaritmo de odds* de que un estudiante obtenga matrícula teniendo un 0 en en matemáticas. Como es de esperar, los *odds* son muy bajos $e^{-9.793942} = 5.579e^{-5}$, lo que se corresponde con una probabilidad de obtener matrícula de $p = \frac{e^{-9.793942}}{1+e^{-9.793942}} = 5.579e^{-5}$.

Acorde al modelo, el logaritmo de los *odds* de que un estudiante tenga matrícula está positivamente relacionado con la puntuación obtenida en matemáticas (coeficiente de regresión = 0.1563404). Esto significa que, por cada unidad que se incrementa la variable matemáticas, se espera que el logaritmo de *odds* de la variable matrícula se incremente en promedio 0.1563404 unidades. Aplicando la inversa del logaritmo natural ($e^{0.1563404} = 1.169$) se obtiene que, por cada unidad que se incrementa la variable matemáticas, los *odds* de obtener matrícula se incremente en promedio 1.169 unidades. No hay que confundir esto último con que la probabilidad de matrícula se incremente un 1.169 %.

A diferencia de la regresión lineal en la que β_1 se corresponde con el cambio promedio en la variable dependiente *Y* debido al incremento en una unidad del predictor *X*, en regresión logística, β_1 indica el cambio en el logaritmo de *odds* debido al incremento de una unidad de *X*, o lo que es lo mismo, multiplica los *odds* por e^{β_1}. Dado que la relación entre *p(Y)* y *X* no es lineal, β_1 no se corresponde con el cambio en la probabilidad de *Y* asociado con el incremento de una unidad de *X*. Cuánto se incremente la probabilidad de *Y* por unidad de *X* depende del valor de *X*, es decir, de la posición en la curva logística en la que se encuentre.

Además del valor de las estimaciones de los coeficientes parciales de correlación del modelo, es conveniente calcular sus correspondientes intervalos de confianza. En el caso de regresión logística, estos intervalos suelen calcularse empleando el método de *profile likelihood* (en **R** es el método por defecto si se tiene instalado el paquete MASS). Para una descripción más detallada ver: http://www.math.umt.edu/patterson/ProfileLikelihoodCI.pdf

```
confint(object = modelo, level = 0.95 )
```

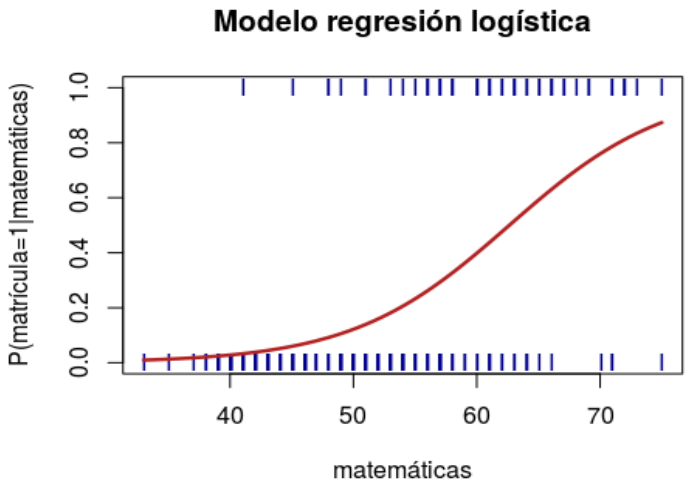
```
##                   2.5 %     97.5 %
## (Intercept) -12.9375208 -7.0938806
## matematicas   0.1093783  0.2103937
```

Gráfico del modelo

Dado que un modelo logístico modela el logaritmo de *ODDs*, estas son las unidades en las que se devuelven las predicciones. Es necesario convertirlas de nuevo en probabilidad mediante la función *logit*. En **R**, la función `predict()` puede devolver directamente las probabilidades en lugar de los *logODDs* si se indica el argumento `type="response"`. Sin embargo, si se quieren calcular intervalos de confianza y que estos no se salgan del rango [0, 1] es necesario emplear los *logODDs* y una vez que se les ha sustraído o sumado el margen de error (*Z x SE*) se transforman en probabilidades.

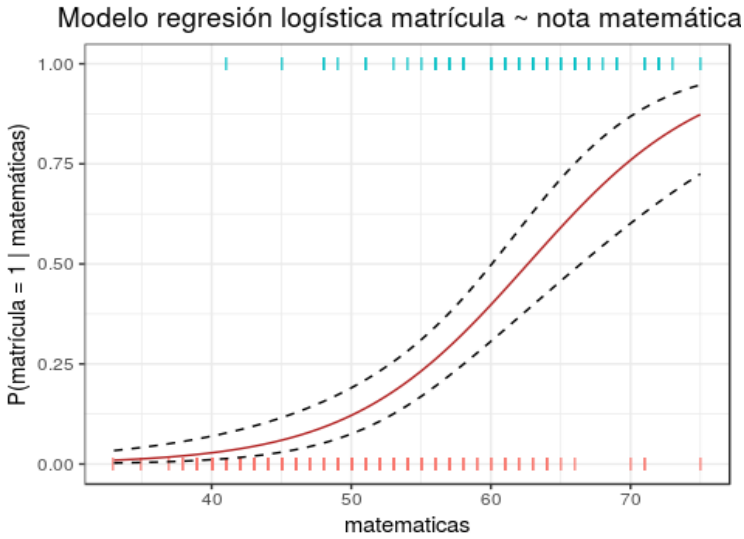
```
# MEDIANTE BASE GRAPHICS SIN INTERVALOS DE CONFIANZA

# Codificación 0,1 de la variable respuesta
datos$matricula <- as.character(datos$matricula)
datos$matricula <- as.numeric(datos$matricula)
plot(matricula ~ matematicas, datos, col = "darkblue",
     main = "Modelo regresión logística",
     ylab = "P(matrícula=1|matemáticas)",
     xlab = "matemáticas", pch = "I")
# type="response" devuelve las predicciones en forma de probabilidad en lugar de en
#log_ODDs
```

```r
curve(predict(modelo, data.frame(matematicas = x), type = "response"),
      col = "firebrick", lwd = 2.5, add = TRUE)
```




```r
# MEDIANTE GGPLOT2 INCLUYENDO INTERVALOS DE CONFIANZA
datos$matricula <- as.character(datos$matricula)
datos$matricula <- as.numeric(datos$matricula)

# Se crea un vector con nuevos valores interpolados en el rango de observaciones.
nuevos_puntos <- seq(from = min(datos$matematicas), to = max(datos$matematicas),
                     by = 0.5)
# Predicciones de los nuevos puntos según el modelo.
# Si se indica se.fit = TRUE se devuelve el error estándar de cada predicción
# junto con el valor de la predicción (fit).
predicciones <- predict(modelo, data.frame(matematicas = nuevos_puntos),
                        se.fit = TRUE)

# Mediante la función logit se transforman los log_ODDs a probabilidades.
predicciones_logit <- exp(predicciones$fit) / (1 + exp(predicciones$fit))

# Se calcula el límite inferior y superior del IC del 95% sustrayendo e
# incrementando el logODDs de cada predicción 1.95*SE. Una vez calculados los
# logODDs del intervalo se transforman en probabilidades con la función logit.
limite_inferior       <- predicciones$fit - 1.96 * predicciones$se.fit
limite_inferior_logit <- exp(limite_inferior) / (1 + exp(limite_inferior))
limite_superior       <- predicciones$fit + 1.96 * predicciones$se.fit
limite_superior_logit <- exp(limite_superior) / (1 + exp(limite_superior))

# Se crea un data frame con los nuevos puntos y sus predicciones
datos_curva <- data.frame(matematicas = nuevos_puntos,
                          probabilidad_matricula = predicciones_logit,
                          limite_inferior_logit = limite_inferior_logit,
                          limite_superior_logit = limite_superior_logit)
```

```
ggplot(datos, aes(x = matematicas, y = matricula)) +
      geom_point(aes(color = as.factor(matricula)), shape = "I", size = 3) +
      geom_line(data = datos_curva, aes(y = probabilidad_matricula),
                color = "firebrick") +
      geom_line(data = datos_curva, aes(y = limite_inferior_logit),
                linetype = "dashed") +
      geom_line(data = datos_curva, aes(y = limite_superior_logit),
                linetype = "dashed") +
      theme_bw() +
      labs(title = "Modelo regresión logística matrícula ~ nota matemáticas",
           y = "P(matrícula = 1 | matemáticas)", y = "matemáticas") +
      theme(legend.position = "null") +
      theme(plot.title = element_text(hjust = 0.5))
```



Evaluación del modelo

A la hora de evaluar la validez y calidad de un modelo de regresión logística, se analiza tanto el modelo en su conjunto como los predictores que lo forman.

Se considera que el modelo es útil si es capaz de mostrar una mejora explicando las observaciones respecto al modelo nulo (sin predictores). El test *Likelihood ratio* calcula la significancia de la diferencia de residuos entre el modelo de interés y el modelo nulo. El estadístico sigue una distribución *chi-cuadrado* con grados de libertad equivalentes a la diferencia de grados de libertad de los dos modelos.

```
# Diferencia de residuos
# En R, un objeto glm almacena la "deviance" del modelo, así como la "deviance"
# del modelo nulo.
dif_residuos <- modelo$null.deviance - modelo$deviance

# Grados libertad
df <- modelo$df.null - modelo$df.residual

# p-value
p_value <- pchisq(q = dif_residuos,df = df, lower.tail = FALSE)

paste("Diferencia de residuos:", round(dif_residuos, 4))
```

```
## [1] "Diferencia de residuos: 55.6368"
```

```
paste("Grados de libertad:", df)
```

```
## [1] "Grados de libertad: 1"
```

```
paste("p-value:", p_value)
```

```
## [1] "p-value: 8.71759108087093e-14"
```

```
# El mismo cálculo se puede obtener directamente con:
anova(modelo, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: matricula
##
## Terms added sequentially (first to last)
##
##
##             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                          199     222.71
## matematicas  1   55.637       198     167.07 8.718e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En este caso, el modelo obtenido sí es significativo.

Para determinar si los predictores introducidos en un modelo de regresión logística contribuyen de forma significativa se emplea el estadístico *Z* y el test *Wald chi-test*. Este es el método utilizado para calcular los *p-values* que se muestran al hacer summary() del modelo. El predictor *matemáticas* sí contribuye de forma significativa (*p-value* = 1.03e-09).

A diferencia de los modelos de regresión lineal, en los modelos logísticos no existe un equivalente a R^2 que determine exactamente la varianza explicada por el modelo. Se han desarrollado diferentes métodos conocidos como $pseudoR^2$ que intentan aproximarse al concepto de R^2 pero que, aunque su rango oscila entre 0 y 1, no se pueden considerar equivalentes.

- **McFadden's**: $R^2_{McF} = 1 - \frac{ln\hat{L}(modelo)}{ln\hat{L}(modelo\ nulo)}$, siendo $\hat{L}$ el valor de *likelihood* de cada modelo. La idea de esta fórmula es que, $ln(\hat{L})$, tiene un significado análogo a la suma de cuadrados de la regresión lineal. De ahí que se le denomine $pseudoR^2$.
- Otra opción bastante extendida es el test de **Hosmer-Lemeshow**. Este test examina mediante un *Pearson chi-square test* si las proporciones de eventos observados son similares a las probabilidades predichas por el modelo, haciendo subgrupos.

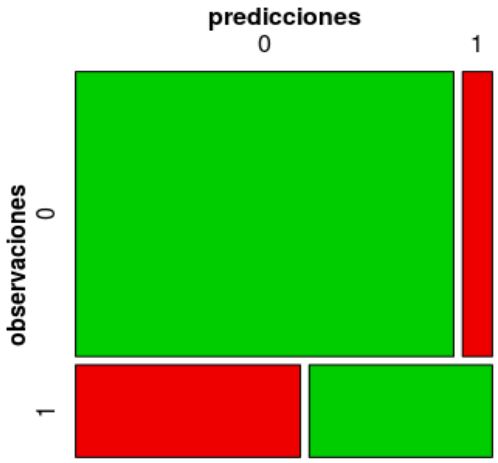
Comparación de clasificación predicha y observaciones

Para este estudio se va a emplear un *threshold* de 0.5. Si la probabilidad de que la variable adquiera el valor 1 (matrícula) es superior a 0.5, se asigna a este nivel, si es menor se asigna al 0 (no matrícula).

```
library(vcd)
predicciones <- ifelse(test = modelo$fitted.values > 0.5, yes = 1, no = 0)
matriz_confusion <- table(modelo$model$matricula, predicciones,
                          dnn = c("observaciones", "predicciones"))
matriz_confusion
```

```
##              predicciones
## observaciones   0   1
##             0 140  11
##             1  27  22
```

```
mosaic(matriz_confusion, shade = T, colorize = T,
       gp = gpar(fill = matrix(c("green3", "red2", "red2", "green3"), 2, 2)))
```



El modelo es capaz de clasificar correctamente $\frac{140+22}{140+22+27+11} = 0.81(81\%)$ de las observaciones cuando se emplean los datos de entrenamiento. No hay que olvidar que este es el error de entrenamiento, por lo que no es generalizable a nuevas observaciones. Para obtener una estimación más realista hay que calcular el error de test.

Conclusión

El modelo logístico creado para predecir la probabilidad de que un alumno obtenga matrícula de honor a partir de la nota de matemáticas es en conjunto significativo acorde al *Likelihood ratio* (*p-value* = 8.717591e-14). El *p-value* del predictor *matemáticas* es significativo (*p-value* = 1.03e-09).

$$\text{logit}(\text{matricula}) = -9.793942 + 0.1563404 * \text{nota matematicas}$$

$$\text{P}(\text{matricula}) = \frac{e^{-9.793942 + 0.1563404 * \text{nota matematicas}}}{1 + e^{-9.793942 + 0.1563404 * \text{nota matematicas}}}$$

Regresión logística múltiple

Idea intuitiva

La regresión logística múltiple es una extensión de la regresión logística simple. Se basa en los mismos principios que la regresión logística simple (explicados anteriormente) pero ampliando el número de predictores. Los predictores pueden ser tanto continuos como categóricos.

$$\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

$$logit(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

El valor de la probabilidad de *Y* se puede obtener con la inversa del logaritmo natural:

$$p(Y) = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i}}$$

A la hora de evaluar la validez y calidad de un modelo de regresión logística múltiple, se analiza tanto el modelo en su conjunto como los predictores que lo forman. Se considera que el modelo es útil si es capaz de mostrar una mejora respecto al modelo nulo, el modelo sin predictores. Existen 3 test estadísticos que cuantifican esta mejora mediante la comparación de los residuos: *likelihood ratio*, *score* y *Wald test*. No hay garantías de que los 3 lleguen a la misma conclusión, cuando esto ocurre parece ser recomendable basarse en el *likelihood ratio*. http://www.ats.ucla.edu/stat/mult_pkg/faq/general/nested_tests.htm.

Ejemplo1

Un estudio considera que existe relación entre el hecho de que un estudiante asista a clases de repaso de lectura (sí = 1, no = 0), la nota que obtiene en un examen de lectura estándar (realizado antes de iniciar las clases de repaso) y el sexo (hombre = 1, mujer = 0). Se quiere generar un modelo en el que a partir de las variables puntuación del examen y sexo, prediga la probabilidad de que el estudiante tenga que asistir a clases de repaso.

```
datos <- data.frame(sexo = c("hombre", "hombre", "mujer", "mujer", "mujer", "hombre",
                             "mujer", "hombre", "mujer", "mujer", "hombre", "hombre",
                             "hombre", "hombre", "mujer", "mujer", "hombre", "mujer",
                             "hombre", "mujer", "hombre", "mujer", "mujer", "hombre",
                             "hombre", "mujer", "mujer", "mujer", "hombre", "hombre",
                             "hombre", "mujer", "hombre", "mujer", "hombre", "mujer",
                             "mujer", "mujer", "mujer", "mujer", "hombre", "mujer",
                             "hombre", "mujer", "mujer", "mujer", "mujer", "hombre",
                             "mujer", "hombre", "mujer", "hombre", "mujer", "mujer",
                             "hombre", "hombre", "hombre", "hombre", "hombre", "hombre",
                             "hombre", "hombre", "hombre", "mujer", "hombre", "hombre",
                             "hombre", "hombre", "mujer", "hombre", "mujer", "hombre",
                             "hombre", "hombre", "mujer", "hombre", "mujer", "mujer",
                             "hombre", "mujer", "mujer", "mujer", "hombre", "hombre",
                             "hombre", "hombre", "hombre", "mujer", "mujer", "mujer",
                             "mujer", "hombre", "mujer", "mujer", "mujer", "mujer",
                             "mujer", "mujer", "mujer", "mujer","mujer", "mujer",
                             "hombre", "mujer", "hombre", "hombre", "mujer", "mujer",
                             "mujer", "hombre","mujer", "hombre", "mujer", "mujer",
                             "mujer", "hombre", "mujer", "hombre", "mujer", "hombre",
                             "mujer", "hombre", "mujer", "mujer", "mujer", "mujer",
                             "mujer", "mujer", "mujer", "mujer", "hombre", "mujer",
                             "hombre", "hombre", "hombre", "hombre", "hombre", "hombre",
                             "hombre", "mujer","mujer", "mujer", "hombre", "hombre",
                             "mujer", "mujer", "hombre", "mujer", "hombre", "hombre",
                             "hombre", "mujer", "mujer", "mujer", "mujer", "hombre",
                             "hombre", "mujer", "hombre", "hombre", "mujer", "hombre",
                             "hombre", "hombre", "hombre", "mujer", "hombre", "hombre",
                             "mujer", "mujer", "hombre", "hombre", "hombre", "hombre",
                             "hombre", "mujer", "mujer", "mujer", "mujer", "hombre",
                             "hombre", "hombre", "mujer", "hombre", "mujer", "hombre",
                             "hombre", "hombre", "mujer"),
                  examen_lectura = c(91, 77.5, 52.5, 54, 53.5, 62, 59, 51.5,
                                     61.5, 56.5, 47.5, 75, 47.5, 53.5, 50, 50,
                                     49, 59, 60, 60, 60.5, 50, 101, 60, 60,
                                     83.5, 61, 75, 84, 56.5, 56.5, 45, 60.5,
                                     77.5, 62.5, 70, 69, 62, 107.5, 54.5, 92.5,
                                     94.5, 65, 80, 45, 45, 66, 66, 57.5, 42.5,
                                     60, 64, 65, 47.5, 57.5, 55, 55, 76.5,
                                     51.5, 59.5, 59.5, 59.5, 55, 70, 66.5,
                                     84.5, 57.5, 125, 70.5, 79, 56, 75, 57.5,
                                     56, 67.5, 114.5, 70, 67, 60.5, 95, 65.5,
```

```
                                  85, 55, 63.5, 61.5, 60, 52.5, 65, 87.5,
                                  62.5, 66.5, 67, 117.5, 47.5, 67.5, 67.5,
                                  77, 73.5, 73.5, 68.5, 55, 92, 55, 55, 60,
                                  120.5, 56, 84.5, 60, 85, 93, 60, 65, 58.5,
                                  85, 67, 67.5, 65, 60, 47.5, 79, 80, 57.5,
                                  64.5, 65, 60, 85, 60, 58, 61.5, 60, 65,
                                  93.5, 52.5, 42.5, 75, 48.5, 64, 66, 82.5,
                                  52.5, 45.5, 57.5, 65, 46, 75, 100, 77.5,
                                  51.5, 62.5, 44.5, 51, 56, 58.5, 69, 65,
                                  60, 65, 65, 40, 55, 52.5, 54.5, 74, 55,
                                  60.5, 50, 48, 51, 55, 93.5, 61, 52.5,
                                  57.5, 60, 71, 65, 60, 55, 60, 77, 52.5,
                                  95, 50, 47.5, 50, 47, 71, 65),
                clases_repaso = c("0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "0", "0", "0", "0", "0", "0",
                                  "0", "0", "1", "1", "1", "1", "1", "1",
                                  "1", "1", "1", "1", "1", "1", "1", "1",
                                  "1", "1", "1", "1", "1", "1", "1", "1",
                                  "1", "1", "1", "1", "1", "1", "1", "1",
                                  "1", "1", "1", "1", "1", "1", "1", "1",
                                  "1", "1", "1", "1", "1", "1", "1", "1",
                                  "1", "1", "1", "1", "1"))
```

Análisis de las observaciones

Las tablas de frecuencia así como representaciones gráficas de las observaciones son útiles para intuir si las variables independientes escogidas están relacionadas con la variable respuesta y por lo tanto ser buenos predictores.

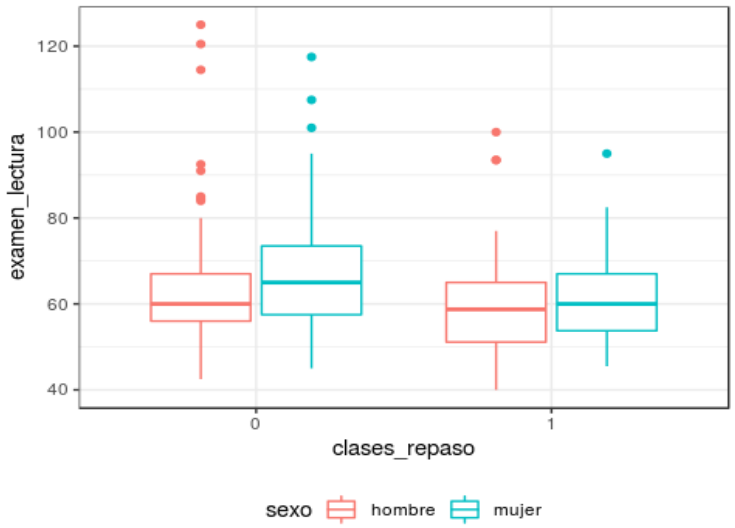
```
library(ggplot2)
tabla <- table(datos$clases_repaso, datos$sexo,
               dnn = c("clases de repaso","sexo"))
addmargins(tabla)
```

```
##                 sexo
## clases de repaso hombre mujer Sum
##              0       57    73 130
##              1       36    23  59
##              Sum     93    96 189
```

```
tabla_frecuencias <- prop.table(tabla)*100
addmargins(tabla_frecuencias)
```

```
##                 sexo
## clases de repaso    hombre     mujer       Sum
##              0    30.15873  38.62434  68.78307
##              1    19.04762  12.16931  31.21693
##              Sum  49.20635  50.79365 100.00000
```

```
ggplot(data = datos, aes(x = clases_repaso, y = examen_lectura, colour = sexo)) +
  geom_boxplot() +
  theme(legend.position = "bottom")
  theme_bw() +
```



El número de estudiantes en la muestra es semejante para ambos sexos (96, 93). Parece ser mayor el porcentaje de hombres que necesitan clases de repaso (19.04762, 12.16931). El promedio de las notas de lectura de los estudiantes que requieren de clases particulares es menor que el de los que no requieren clases. En vista de estos datos, tiene sentido considerar el *sexo* y la *nota* como posibles predictores.

Generar el modelo de regresión logística

```
modelo_glm <- glm(clases_repaso ~ examen_lectura + sexo, data = datos,
                  family = "binomial")
summary(modelo_glm)
```

```
##
## Call:
## glm(formula = clases_repaso ~ examen_lectura + sexo, family = "binomial",
##     data = datos)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2079  -0.8954  -0.7243   1.2592   2.0412
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.18365    0.78559   1.507   0.1319
## examen_lectura -0.02617    0.01223  -2.139   0.0324 *
## sexomujer      -0.64749    0.32484  -1.993   0.0462 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 234.67  on 188  degrees of freedom
## Residual deviance: 224.64  on 186  degrees of freedom
## AIC: 230.64
##
## Number of Fisher Scoring iterations: 4
```

Acorde al modelo, el *logaritmo de odds* de que un estudiante necesite clases de repaso esta negativamente relacionado con la puntuación obtenida en el examen de lectura (coeficiente parcial = -0.02617), siento significativa esta relación (*p-value* = 0.0324). También existe una relación significativa positiva entre el *logaritmo de odds* de necesitar clases de repaso y el género del estudiante (*p-value* = 0.0462), siendo, para un mismo resultado en el examen de

lectura, mayor si el estudiante es hombre. En concreto los *odds* de que un hombre requiera clases de repaso son $e^{0.64749} = 1.910739$ mayores que los de las mujeres. (Esto se puede ver gráficamente representando el modelo para hombres y mujeres).

Además del valor estimado de los coeficientes parciales de correlación calculados por el modelo, es conveniente generar sus correspondientes intervalos de confianza. En el caso de regresión logística, estos intervalos suelen calcularse basados en *profile likelihood* (en **R** es el método por defecto si se tiene instalado el paquete `MASS`).

```
confint(modelo_glm)
```

```
##                     2.5 %       97.5 %
## (Intercept)    -0.2973940  2.799267537
## examen_lectura -0.0518029 -0.003536647
## sexomujer      -1.2931216 -0.015658932
```

```
# En caso de querer los intervalos basados en el error estándar.
confint.default(modelo_glm)
```

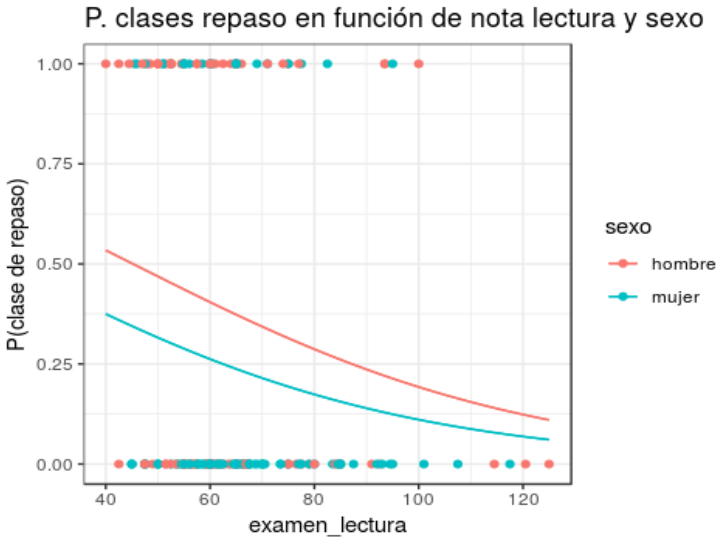
```
##                     2.5 %       97.5 %
## (Intercept)    -0.35608462  2.723378445
## examen_lectura -0.05015001 -0.002192983
## sexomujer      -1.28416605 -0.010807125
```

Representación gráfica del modelo

Al tratarse de un modelo con 2 predictores, no se puede obtener una representación en 2D en la que se incluyan ambos predictores a la vez. Sí es posible representar la curva del modelo logístico cuando se mantiene constante uno de los dos predictores. Por ejemplo, al representar las predicciones del modelo diferenciando entre hombres y mujeres (fijando el valor del predictor sexo) se aprecia que la curva de los hombres siempre está por encima. Esto se debe a que, como indica el coeficiente parcial de correlación del predictor sexo, para una misma nota en el examen de lectura, el *logaritmo de ODDs* de necesitar clases de repaso es 0.64749 veces mayor en hombres.

```
library(ggplot2)


# Para graficar los valores en ggplot junto con la curva, la variable respuesta
# tiene que ser numérica en lugar de factor.
datos$clases_repaso <- as.numeric(as.character(datos$clases_repaso))

# Se crea un dataframe que contenga la probabilidad de que se necesiten clases
# de repaso dada una determinada nota en el examen de lectura y siendo hombre.
```

```
# Vector con nuevos valores interpolados en el rango de observaciones.
nuevos_valores_examen <- seq(from = min(datos$examen_lectura),
                             to = max(datos$examen_lectura), by = 0.5)
sexo <- as.factor(rep(x = "hombre", length(nuevos_valores_examen)))
# Predicciones de los nuevos puntos según el modelo. type = "response" devuelve
# las predicciones en forma de probabilidad en lugar de en log_ODDs.
predicciones <- predict(object = modelo_glm,
                        newdata=data.frame(examen_lectura=nuevos_valores_examen,
                                           sexo = sexo),
                        type = "response")
# Se crea un data frame con los nuevos puntos y sus predicciones para graficar
# la curva.
datos_curva_hombre <- data.frame(examen_lectura = nuevos_valores_examen,
                                 sexo = sexo,
                                 clases_repaso = predicciones)

# Mismo proceso para mujeres (sexo = 0).
nuevos_valores_examen <- seq(from = min(datos$examen_lectura),
                             to = max(datos$examen_lectura), by = 0.5)
sexo <- as.factor(rep("mujer", length(nuevos_valores_examen)))
predicciones <- predict(object = modelo_glm,
                        newdata=data.frame(examen_lectura=nuevos_valores_examen,
                                           sexo = sexo),
                        type = "response")
datos_curva_mujer <- data.frame(examen_lectura = nuevos_valores_examen,
                                sexo = sexo, clases_repaso = predicciones)

# Se unifican los dos dataframe.
datos_curva <- rbind(datos_curva_hombre, datos_curva_mujer)

ggplot(data = datos, aes(x = examen_lectura, y = as.numeric(clases_repaso),
                         color = sexo)) +
    geom_point() +
    geom_line(data = datos_curva, aes(y = clases_repaso)) +
    geom_line(data = datos_curva, aes(y = clases_repaso)) +
    theme_bw() +
    labs(title = "P. clases repaso en función de nota lectura y sexo",
         y = "P(clase de repaso)")
```



Evaluación del modelo

Likelihood ratio:

```
# Diferencia de residuos
dif_residuos <- modelo_glm$null.deviance - modelo_glm$deviance

# Grados libertad
df <- modelo_glm$df.null - modelo_glm$df.residual
# p-value
p_value <- pchisq(q = dif_residuos,df = df, lower.tail = FALSE)

paste("Diferencia de residuos:", round(dif_residuos, 4))
```

```
## [1] "Diferencia de residuos: 10.0334"
```

```
paste("Grados de libertad:", df)
```

```
## [1] "Grados de libertad: 2"
```

```
paste("p-value:", round(p_value, 4))
```

```
## [1] "p-value: 0.0066"
```

El modelo en conjunto sí es significativo y, acorde a los *p-values* mostrados en el summary(), también es significativa la contribución al modelo de ambos predictores.

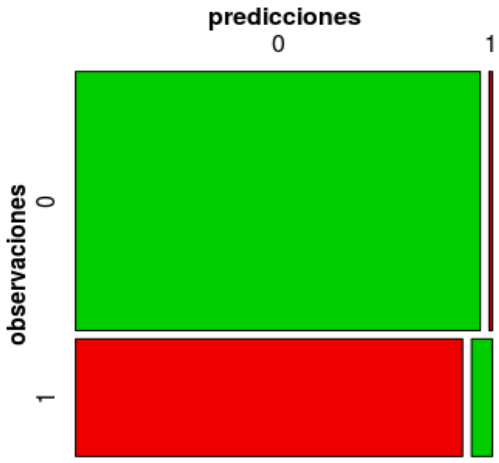
Comparación de las predicciones con las observaciones

Para este estudio se va a emplear un *threshold* de 0.5. Si la probabilidad predicha de asistir a clases de repaso es superior a 0.5 se asigna al nivel 1 (sí asiste), si es menor se asigna al nivel 0 (no clases de repaso).

```
predicciones <- ifelse(test = modelo_glm$fitted.values > 0.5, yes = 1, no = 0)
matriz_confusion <- table(modelo_glm$model$clases_repaso, predicciones,
                          dnn = c("observaciones", "predicciones"))
matriz_confusion
```

```
##              predicciones
## observaciones   0   1
##             0 129   1
##             1  56   3
```

```
mosaic(matriz_confusion, shade = T, colorize = T,
       gp = gpar(fill = matrix(c("green3", "red2", "red2", "green3"), 2, 2)))
```



El modelo es capaz de clasificar correctamente $\frac{129+3}{129+3+56+1} = 0.698(69.8\%)$ de las observaciones de entrenamiento. Si se analiza en detalle cómo se distribuye el error, se aprecia que el modelo solo ha sido capaz de identificar correctamente a 3 de los 59 alumnos que realmente asisten a clases de repaso. El porcentaje de falsos negativos es muy alto. Seleccionar otro *threshold* puede mejorar la exactitud del modelo.

```
library(vcd)
predicciones <- ifelse(test = modelo_glm$fitted.values > 0.45, yes = 1, no = 0)
matriz_confusion <- table(modelo_glm$model$clases_repaso, predicciones,
                          dnn = c("observaciones", "predicciones"))
matriz_confusion
```

```
##              predicciones
## observaciones   0   1
##             0 122   8
##             1  45  14
```

Conclusión

El modelo logístico creado para predecir la probabilidad de que un alumno tenga que asistir a clases de repaso a partir de la nota obtenida en un examen de lectura y el sexo del alumno es en conjunto significativo acorde al *Likelihood ratio* (*p-value* = 0.0066). El *p-value* de ambos predictores es significativo (examen_lectura = 0.0324, sexo1 = 0.0462). El ratio de error obtenido empleando las observaciones con las que se ha entrenado el modelo muestra un porcentaje de falsos negativos muy alto.

$$\text{logit}(\text{clases de repaso}) = 0.53616 - 0.02617 * \text{examen lectura} + 0.64749 * \text{sexo}$$

$$\text{P}(\text{clases de repaso}) = \frac{e^{0.53616 - 0.02617\ \text{examen lectura} + 0.64749\ \text{sexo}}}{1 + e^{0.53616 - 0.02617\ \text{examen lectura} + 0.64749\ \text{sexo}}}$$

No hay que olvidar que los errores calculados son de entrenamiento, por lo que no son generalizables a nuevas observaciones. Para obtener una estimación más realista hay que calcular el error de test.

Ejemplo2

Se dispone de un registro que contiene cientos de emails con información de cada uno de ellos. El objetivo de estudio es intentar crear un modelo que permita filtrar qué emails son "spam" y cuáles no, en función de determinadas características. Ejemplo extraído del libro OpenIntro Statistics.

```
library(openintro)
data(email)
str(email)
```

```
## 'data.frame':    3921 obs. of  21 variables:
##  $ spam        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ to_multiple : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ from        : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ cc          : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ sent_email  : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ time        : POSIXct, format: "2012-01-01 07:16:41" "2012-01-01 08:03:59"
...
##  $ image       : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ attach      : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ dollar      : num  0 0 4 0 0 0 0 0 0 0 ...
##  $ winner      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ inherit     : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ viagra      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ password    : num  0 0 0 0 2 2 0 0 0 0 ...
##  $ num_char    : num  11.37 10.5 7.77 13.26 1.23 ...
##  $ line_breaks : int  202 202 192 255 29 25 193 237 69 68 ...
##  $ format      : num  1 1 1 1 0 0 1 1 0 1 ...
##  $ re_subj     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ exclaim_subj: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ urgent_subj : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ exclaim_mess: num  0 1 6 48 1 1 1 18 1 0 ...
##  $ number      : Factor w/ 3 levels "none","small",..: 3 2 2 2 1 1 3 2 2 2 ...
```

En este caso se van a emplear únicamente como posibles predictores variables categóricas. Esto se debe a que los *outliers* complican bastante la creación de estos modelos y, en el data set que se emplea como ejemplo, las variables cuantitativas son muy asimétricas. En particular, las variables que se van a estudiar como posibles predictores son:

- spam: si el email es spam (1) si no lo es (0).
- to_multiple: si hay más de una persona en la lista de distribución.
- format: si está en formato HTLM.
- cc: si hay otras direcciones en copia.
- attach: si hay archivos adjuntos.

- dollar: si el email contiene la palabra dollar o el símbolo $.
- inherit: si contiene la palabra inherit.
- winner: si el email contiene la palabra winner.
- password: si el email contiene la palabra password.
- re_subj: si la palabra "Re:" está escrita en el asunto del email.
- exclaim_subj: si se incluye algún signo de exclamación en el email.

En primer lugar se genera el modelo completo introduciendo todas las variables como predictores.

```
modelo_completo <- glm(spam ~ to_multiple + format + cc + attach + dollar +
                         winner + inherit + password + re_subj + exclaim_subj,
                         data = email,
                         family = binomial)
summary(modelo_completo)
```

```
## Call:
## glm(formula = spam ~ to_multiple + format + cc + attach + dollar +
##     winner + inherit + password + re_subj + exclaim_subj, family = binomial,
##     data = email)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6348  -0.4325  -0.2566  -0.0945   3.8846
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.79976    0.08935  -8.950  < 2e-16 ***
## to_multiple  -2.84097    0.31158  -9.118  < 2e-16 ***
## format       -1.52284    0.12270 -12.411  < 2e-16 ***
## cc            0.03134    0.01895   1.654 0.098058 .
## attach        0.20351    0.05851   3.478 0.000505 ***
## dollar       -0.07304    0.02306  -3.168 0.001535 **
## winneryes     1.83103    0.33641   5.443 5.24e-08 ***
## inherit       0.32999    0.15223   2.168 0.030184 *
## password     -0.75953    0.29597  -2.566 0.010280 *
## re_subj      -3.11857    0.36522  -8.539  < 2e-16 ***
## exclaim_subj  0.24399    0.22502   1.084 0.278221
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2437.2  on 3920  degrees of freedom
## Residual deviance: 1936.2  on 3910  degrees of freedom
## AIC: 1958.2
##
## Number of Fisher Scoring iterations: 7
```

Se mejora el modelo excluyendo aquellos predictores cuyo *p-values* no es significativo. El resultado es el siguiente modelo.

```
modelo_final <- glm(spam ~ to_multiple + format + attach + dollar + winner +
                    inherit + password + re_subj,
                    data = email,
                    family = binomial)
summary(modelo_final)
```

```
##
## Call:
## glm(formula = spam ~ to_multiple + format + attach + dollar +
##     winner + inherit + password + re_subj, family = binomial,
##     data = email)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6591  -0.4373  -0.2544  -0.0944   3.8707
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.78138    0.08860  -8.820  < 2e-16 ***
## to_multiple -2.77682    0.30752  -9.030  < 2e-16 ***
## format      -1.51770    0.12226 -12.414  < 2e-16 ***
## attach       0.20419    0.05789   3.527  0.00042 ***
## dollar      -0.06970    0.02239  -3.113  0.00185 **
## winneryes    1.86675    0.33652   5.547  2.9e-08 ***
## inherit      0.33614    0.15073   2.230  0.02575 *
## password    -0.76035    0.29680  -2.562  0.01041 *
## re_subj     -3.11329    0.36519  -8.525  < 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: 2437.2  on 3920  degrees of freedom
## Residual deviance: 1939.6  on 3912  degrees of freedom
## AIC: 1957.6
##
## Number of Fisher Scoring iterations: 7
```

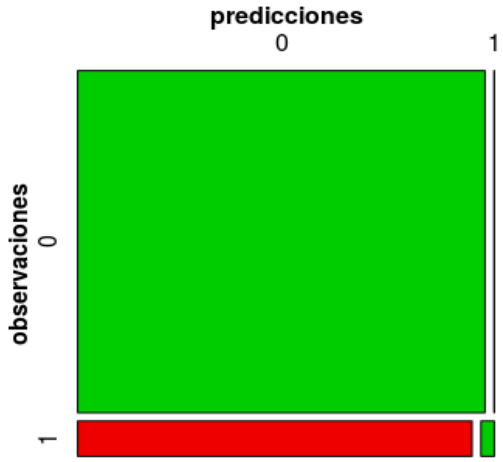
Con este modelo podemos saber la probabilidad, dadas unas determinadas características, de que el email sea *spam* (valor 1 de la variable).

Para evaluar el modelo, se puede comparar el valor real (si realmente es *spam*) con el predicho por el modelo.

```
library(vcd)
predicciones <- ifelse(test = modelo_final$fitted.values > 0.5, yes = 1, no = 0)
matriz_confusion <- table(modelo_final$model$spam, predicciones,
                          dnn = c("observaciones", "predicciones"))
matriz_confusion
```

```
##              predicciones
## observaciones    0    1
##             0 3551    3
##             1  355   12
```

```
mosaic(matriz_confusion, shade = T, colorize = T,
       gp = gpar(fill = matrix(c("green3", "red2", "red2", "green3"), 2, 2)))
```



El modelo generado no es capaz de diferenciar bien entre *spam* y no *spam*, solo 12 de los 367 correos *spam* se han identificado correctamente. Si fuese capaz, la probabilidad calculada por el modelo para aquellos *emails* del *dataset* que sí son *spam* deberían estar por encima del 0.5.

Anexos

Ajuste por descenso de gradiente

El método de optimización por descenso de gradiente sigue un proceso iterativo en el que, partiendo de una posición inicial dentro del dominio de la función, realiza desplazamientos en la dirección correcta hasta alcanzar el mínimo. Para poder llevar a cabo el proceso se necesita:

- La función objetivo a minimizar.
- La posición desde la que iniciar la búsqueda.
- La distancia (*step* o *learning rate*) que se va a desplazar el algoritmo en cada iteración de búsqueda. Es importante escoger un *learning rate* adecuado en cada escenario. Si es muy pequeño, se tardará demasiado en llegar al mínimo y, si es demasiado grande, el algoritmo saltará de una región a otra pasando por encima del mínimo sin alcanzarlo.
- La dirección de búsqueda en la que se produce el desplazamiento. El algoritmo de descenso de gradiente se caracteriza porque emplea como dirección de búsqueda aquella en la que la función desciende en mayor medida. Esta dirección se corresponde con el valor negativo del gradiente de la función. El gradiente de la función es el vector formado por las derivadas parciales de la función respecto a cada variable.
- La tolerancia: Al tratarse de un algoritmo iterativo, hay que indicar una regla de parada. Lo idóneo es que el algoritmo se detenga al encontrar el mínimo, pero como normalmente se desconoce cuál es, hay que conseguir que se pare lo más cerca posible. Una forma de cuantificar la proximidad al mínimo es controlar cuanto desciende el valor de la función entre iteraciones consecutivas. Si el descenso es muy pequeño, significa que se encuentra muy cerca del valor. La tolerancia se define como un valor tal que, si la diferencia en el descenso es menor o igual, se considera que se ha alcanzado el mínimo (el algoritmo converge). Una alternativa a calcular la distancia entre las dos iteraciones es medir la longitud del vector gradiente en cada iteración. Si su longitud es muy pequeña, también lo es la distancia al mínimo.
- Iteraciones máximas: Si la posición de inicio en la búsqueda está muy alejada del mínimo de la función y el *learning rate* es muy pequeño, se pueden necesitar muchas iteraciones para alcanzarlo. Para evitar un proceso iterativo excesivamente largo, se suele establecer un número máximo de iteraciones permitidas.

Funciones auxiliares

Importante: los cálculos se han implementado asumiendo que, en la matriz de datos X, cada fila es una observación y cada columna un predictor.

```
fun_sigmoide <- function(x){
  # Esta función calcula el valor de la función sigmoide.
  #
  # Argumentos:
  #   x: un escalar.
  #
  # Retorno:
  #  El valor de la función sigmoide en x.

  s <- 1 / (1 + exp(-x))
  return(s)
}
```

```
propagar <- function(W, b, X, Y){
  # Esta función ejecuta una iteración forward (predicción) y backward
  # (retropropagación) de un modelo de regresión logística.
  #
  # Argumentos:
  #   W: vector con los pesos del modelo.
  #   b: bias (un escalar).
  #   X: matriz con el valor de los predictores para cada observación. Cada fila
  #      es una observación y cada columna un predictor.
  #   Y: vector con el valor de la variable respuesta (1 o 0) para cada
  #      observación.
  #
  # Retorno:
  #  activacion: vector con el valor de activación para cada observación de X.
  #  coste: valor medio de la función de coste para el conjunto de observaciones.
  #  dW: gradiente de la función de coste respecto de W.
  #  db: gradiente de la función de coste respecto de b.

  # Comprobar que las dimensiones de cada elemento coinciden.
  if(!length(Y) == nrow(X)){
    stop("Y debe contener tantos valores como filas tiene X")
  }
  if(!length(W) == ncol(X)){
    stop("W debe contener tantos valores como columnas tiene X")
  }
  if(!length(b) == 1 | !is.numeric(b)){
    stop("b debe ser un unico valor numérico")
  }
```

```
  # Se convierte W e Y en vectores columna (n,1)
  W <- matrix(W, ncol = 1)
  Y <- matrix(Y, ncol = 1)

  # Se calcula el valor z para cada observación.
  # La multiplicación matricial es: fila de la izquierda x columna de la derecha.
  # El resultado es un vector columna con el valor z de cada observación en una
  # fila distinta. Este orden de multiplicación es correcto solo si, el vector
  # de pesos es un vector columna, y en la matriz X cada observación es una fila.
  Z <- X %*% W + b

  # Se obtiene el valor de la sigmoide (activación) de cada valor z.
  A <- fun_sigmoide(Z)

  # Se calcula el valor de la función de coste.
  c = (-1/nrow(X)) * sum((Y*log(A) + (1-Y)*log(1-A)))

  # Gradientes respecto a W y b.
  dW <- (1/nrow(X)) * t(A-Y) %*% X
  db <- (1/nrow(X)) * sum(A-Y)

  # Se transpone el gradiente dW para que tenga las mismas dimensiones que W.
  dW <- t(dW)

  # Se compueba que las dimensiones son las esperadas.
  stopifnot(dim(dW) == dim(W))
  stopifnot(is.numeric(db))

  return(list(activacion=A, coste=c, dW=dW, db=db))
}
```

```
test <- propagar(W = c(1,2),
                 b = 2,
                 X = rbind(c(1, 3), c(2, 4), c(-1, -3.2)),
                 Y = c(1, 0, 1))
print(test, digits = 16)
```

```
## $activacion
##                       [,1]
## [1,] 0.999876605424013687
## [2,] 0.999993855825397793
## [3,] 0.004496273160941178
##
## $coste
## [1] 5.801545319394553
##
## $dW
##                   [,1]
```

```
## [1,] 0.998456014637956
## [2,] 2.395072388486207
##
## $db
## [1] 0.001455578136784208
```

```r
optimizar <- function(W, b, X, Y, max_iter, learning_rate){

  # Esta función optimiza los valores de los pesos (W) y bias(b) de un modelo de
  # regresión logística empleando el método de descenso de gradiente.
  #
  # Argumentos:
  #   W: vector con los pesos del modelo.
  #   b: bias (un escalar).
  #   X: matriz con el valor de los predictores para cada observación. Cada fila
  #      es una observación y cada columna un predictor.
  #   Y: vector con el valor de la variable respuesta (1 o 0) para cada
  #      observación.
  #   max_iter: número máximo de iteraciones en la optimización.
  #   learning_rate: learning rate del algoritmo de descenso de gradiente.
  #
  # Retorno:
  #  W: valor de los pesos tras la optimización.
  #  b: valor del bias tras la optimización.
  #  dW: gradientes finales de W.
  #  db: gradiente finale de b.
  #  costes: valor del coste en cada iteración proceso de optimización.

  costes <- rep(NA, max_iter)
  for (i in 1:max_iter) {

    # Cálculo de los gradientes y coste en la iteración i.
    propagacion <- propagar(W = W, b = b, X = X, Y = Y)
    dW <- propagacion[["dW"]]
    db <- propagacion[["db"]]
    costes[i] <- propagacion[["coste"]]

    # Actualización de los gradientes
    W <- W - learning_rate*dW
    b <- b - learning_rate*db
  }

  return(list(W=W, b=b, dW=dW, db=db, costes = costes))
}
```

```
test <- optimizar(W = c(1,2),
                  b = 2,
                  X = rbind(c(1, 3), c(2, 4), c(-1, -3.2)),
                  Y = c(1, 0, 1),
                  max_iter = 100,
                  learning_rate = 0.009)
print(test, digits = 16)
```

```
## $W
##                     [,1]
## [1,] 0.1903359088860433
## [2,] 0.1225915924656141
##
## $b
## [1] 1.925359830084575
##
## $dW
##                     [,1]
## [1,] 0.6775204222153582
## [2,] 1.4162549526380881
##
## $db
## [1] 0.2191945045406766
##
## $costes
##   [1] 5.801545319394553 5.740950502809333 5.680375426895574
##   [4] 5.619821667177627 5.559290922630777 5.498785024864459
##   [7] 5.438305948070814 5.377855819606451 5.317436931189180
##  [10] 5.257051750930272 5.196702936040730 5.136393346354229
##  [13] 5.076126058631377 5.015904381700863 4.955731872427952
##  [16] 4.895612352538271 4.835549926277942 4.775548998918993
##  [19] 4.715614296080809 4.655750883842328 4.595964189588867
##  [22] 4.536260023537635 4.476644600852061 4.417124564226807
##  [25] 4.357707006833278 4.298399495423375 4.239210093432046
##  [28] 4.180147383816733 4.121220491375143 4.062439104215671
##  [31] 4.003813494033473 3.945354534777397 3.887073719253970
##  [34] 3.828983173173290 3.771095666089990 3.713424618643993
##  [37] 3.655984105487431 3.598788853232609 3.541854232753109
##  [40] 3.485196245157411 3.428831500765508 3.372777190449042
##  [43] 3.317051048752004 3.261671308280287 3.206656644961654
##  [46] 3.152026113905351 3.097799075761740 3.043995113670948
##  [49] 2.990633941112374 2.937735301207008 2.885318858284809
##  [52] 2.833404082794996 2.782010130904063 2.731155720379143
##  [55] 2.680859004584204 2.631137446610226 2.582007695704986
##  [58] 2.533485468253436 2.485585435579075 2.438321120781088
##  [61] 2.391704806691144 2.345747456830605 2.300458650974883
##  [64] 2.255846536602117 2.211917797125800 2.168677637404739
##  [67] 2.126129786604603 2.084276518069984 2.043118685473982
##  [70] 2.002655774155893 1.962885966252069 1.923806217977676
##  [73] 1.885412347235535 1.847699129612681 1.810660400776116
```

```
##  [76] 1.774289163290400 1.738577695946010 1.703517663798483
##  [79] 1.669100227266227 1.635316148808326 1.602155895893939
##  [82] 1.569609739172299 1.537667844949709 1.506320361270197
##  [85] 1.475557497074795 1.445369594077312 1.415747191138824
##  [88] 1.386681081048879 1.358162359727219 1.330182467947435
##  [91] 1.302733225753853 1.275806859797353 1.249396023856577
##  [94] 1.223493812840241 1.198093770586379 1.173189891787273
##  [97] 1.148776618376643 1.124848830720323 1.101401833954285
## [100] 1.078431339816471
```

```r
predecir <- function(W, b, X, probabilidad=TRUE){
  # Se convierte W en vectores columna (n,1).
  W <- matrix(W, ncol = 1)

  # Valor z para cada observación.
  # La multiplicación matricial es: fila de la izquierda x columna de la derecha.
  # El resultado es un vector columna con el valor z de cada observación en una
  # fila distinta. Este orden de multiplicación es correcto solo si, el vector
  # de pesos es un vector columna y en la matriz X cada observación es una fila.
  Z <- X %*% W + b

  # Se obtiene el valor de la sigmoide.
  A <- fun_sigmoide(Z)

  if(probabilidad){
    predicciones <- A
  }else{
    predicciones <- ifelse(A>0.5, 1, 0)
  }

  return(predicciones)
}
```

```r
predecir(
  W = c(0.1124579, 0.23106775),
  b = -0.3,
  X = cbind(c(1.,-1.1,-3.2),c(1.2,2.,0.1))
        )
```

```
##           [,1]
## [1,] 0.5224198
## [2,] 0.5096068
## [3,] 0.3459797
```

Función principal

```
entrenar_modelo <- function(X_train, Y_train, max_iter=2000, learning_rate=0.005){

  # Inicialización de pesos y bias
  W <- rep(0, ncol(X_train))
  b <- 0

  # Optimización
  optimizacion <- optimizar(W=W,
                            b=b,
                            X=X_train,
                            Y=Y_train,
                            max_iter=max_iter,
                            learning_rate=learning_rate)

  return(list(W=optimizacion[["W"]],
              b = optimizacion[["b"]],
              costes=optimizacion[["costes"]])
         )
}
```

Ejemplo

```
x1 <- c(7.0, 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5.0, 5.9, 6.0, 6.1, 5.6,
        6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7, 6.0, 5.7,
        5.5, 5.5, 5.8, 6.0, 5.4, 6.0, 6.7, 6.3, 5.6, 5.5, 5.5, 6.1, 5.8, 5.0, 5.6,
        5.7, 5.7, 6.2, 5.1, 5.7, 6.3, 5.8, 7.1, 6.3, 6.5, 7.6, 4.9, 7.3, 6.7, 7.2,
        6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5, 7.7, 7.7, 6.0, 6.9, 5.6, 7.7, 6.3, 6.7,
        7.2, 6.2, 6.1, 6.4, 7.2, 7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6.0, 6.9,
        6.7, 6.9, 5.8, 6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9)

x2 <- c(3.2, 3.2, 3.1, 2.3, 2.8, 2.8, 3.3, 2.4, 2.9, 2.7, 2.0, 3.0, 2.2, 2.9, 2.9,
        3.1, 3.0, 2.7, 2.2, 2.5, 3.2, 2.8, 2.5, 2.8, 2.9, 3.0, 2.8, 3.0, 2.9, 2.6,
        2.4, 2.4, 2.7, 2.7, 3.0, 3.4, 3.1, 2.3, 3.0, 2.5, 2.6, 3.0, 2.6, 2.3, 2.7,
        3.0, 2.9, 2.9, 2.5, 2.8, 3.3, 2.7, 3.0, 2.9, 3.0, 3.0, 2.5, 2.9, 2.5, 3.6,
        3.2, 2.7, 3.0, 2.5, 2.8, 3.2, 3.0, 3.8, 2.6, 2.2, 3.2, 2.8, 2.8, 2.7, 3.3,
        3.2, 2.8, 3.0, 2.8, 3.0, 2.8, 3.8, 2.8, 2.8, 2.6, 3.0, 3.4, 3.1, 3.0, 3.1,
        3.1, 3.1, 2.7, 3.2, 3.3, 3.0, 2.5, 3.0, 3.4, 3.0)

x3 <- c(4.7, 4.5, 4.9, 4.0, 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5, 4.2, 4.0, 4.7, 3.6,
        4.4, 4.5, 4.1, 4.5, 3.9, 4.8, 4.0, 4.9, 4.7, 4.3, 4.4, 4.8, 5.0, 4.5, 3.5,
        3.8, 3.7, 3.9, 5.1, 4.5, 4.5, 4.7, 4.4, 4.1, 4.0, 4.4, 4.6, 4.0, 3.3, 4.2,
        4.2, 4.2, 4.3, 3.0, 4.1, 6.0, 5.1, 5.9, 5.6, 5.8, 6.6, 4.5, 6.3, 5.8, 6.1,
        5.1, 5.3, 5.5, 5.0, 5.1, 5.3, 5.5, 6.7, 6.9, 5.0, 5.7, 4.9, 6.7, 4.9, 5.7,
        6.0, 4.8, 4.9, 5.6, 5.8, 6.1, 6.4, 5.6, 5.1, 5.6, 6.1, 5.6, 5.5, 4.8, 5.4,
        5.6, 5.1, 5.1, 5.9, 5.7, 5.2, 5.0, 5.2, 5.4, 5.1)
```

```
x4 <- c(1.4, 1.5, 1.5, 1.3, 1.5, 1.3, 1.6, 1.0, 1.3, 1.4, 1.0, 1.5, 1.0, 1.4, 1.3,
        1.4, 1.5, 1.0, 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7, 1.5, 1.0,
        1.1, 1.0, 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2, 1.4, 1.2, 1.0, 1.3,
        1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8, 2.2, 2.1, 1.7, 1.8, 1.8, 2.5,
        2.0, 1.9, 2.1, 2.0, 2.4, 2.3, 1.8, 2.2, 2.3, 1.5, 2.3, 2.0, 2.0, 1.8, 2.1,
        1.8, 1.8, 1.8, 2.1, 1.6, 1.9, 2.0, 2.2, 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1,
        2.4, 2.3, 1.9, 2.3, 2.5, 2.3, 1.9, 2.0, 2.3, 1.8)

X <- cbind(x1, x2, x3 ,x4)
Y <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)


modelo_logit_custom <- entrenar_modelo(X_train = X,
                                       Y_train = Y,
                                       max_iter = 5000,
                                       learning_rate = 0.005)
print(modelo_logit_custom$W, digits = 19)
```

```
##                      [,1]
## x1  1.275074329076017143
## x2  0.982365538228177404
## x3 -1.797522101518687077
## x4 -1.632167147076912572
```

```
print(modelo_logit_custom$b, digits = 15)
```

```
## [1] 0.644761927696687
```

```
predicciones <- predecir(W = modelo_logit_custom$W,
                         b = modelo_logit_custom$b,
                         X = X,
                         probabilidad = TRUE)
head(predicciones)
```

```
##           [,1]
## [1,] 0.8787306
## [2,] 0.8040351
## [3,] 0.7741682
## [4,] 0.6468464
## [5,] 0.7244246
## [6,] 0.6112673
```

Bibliografía

Introduction to Statistical Learning

OpenIntro Statistics

An introduction to Logistic Regresion Analysis and Reporting. Chao-Ying Joanne Peng

http://www.ats.ucla.edu/stat/r/dae/logit.htm

http://ww2.coastal.edu/kingw/statistics/R-tutorials/index.html

Points of Significance: Logistic regression Jake Lever, Martin Krzywinski & Naomi Altman. Nature Methods