9no Coloquio Uruguayo de Matemática 2025
Regresion Lineal y Regresión Logística

Mathias Bourel, FCEA

Table of contents

1- Regresión lineal 1
1.1 Primer ejemplo: data set cars 1
1.2 Segundo Ejemplo con datos simulados 11
1.3 Ejemplo de regresión lineal simple con datos reales 16
1.4 Ejemplo donde el modelo es globalmente significativo pero cada coeficiente por separado no lo es: 22
1.5 Regresión lineal si la variable es categórica 23
1.6 Ejemplo con una variable categórica 23
1.6 Comparación de modelos 25
2. Regresión logística 29
2.2 Reduccion en la desvianza del modelo 35
2.3 Otro ejemplo de regresion logística. 37

En este material abordaremos conceptos vinculados al análisis de regresión lineal y regresión logistica, es decir, al estudio de la relación entre dos variables continuas (regresión lineal) o una variable continua y una variable de respuesta categorica (regresión logística). El objetivo no es realizar una revisión exhaustiva del tema, sino ofrecer algunas primeras nociones esenciales para construir y trabajar con estos modelos con el programa R.

1- Regresión lineal

1.1 Primer ejemplo: data set cars

Recordemos que el modelo lineal simple que relaciona dos variables continuas Y y X queda definido por $Y = \beta_0 + \beta_1 X + \epsilon$ donde el intercepto β_0 y la pendiente β_1 se obtienen por ejemplo por el método de los minimos cuadrados y ϵ es una variable aleatoria que se identifica con el error considerado

A partir de un conjunto de datos, el objetivo es estimar los valores de β_1 y β_0 que mejor representen esta relación lineal.

Vamos a trabajar en este primer ejemplo con el dataset cars. Son **50 observaciones** sobre vehículos, con **dos variables numéricas**: **speed**, la velocidad del auto en *millas por hora* (mph) y **dist**, la distancia de frenado en *pies* (ft), es decir, la distancia que recorre el auto desde que el conductor pisa el freno hasta que el vehículo se detiene por completo.

```
rm(list=ls())
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

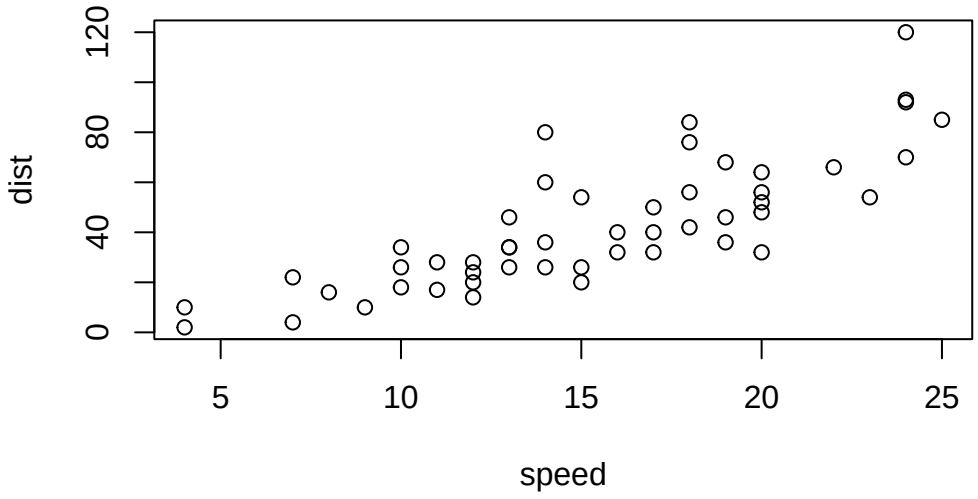
```
str(cars)
```

```
'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

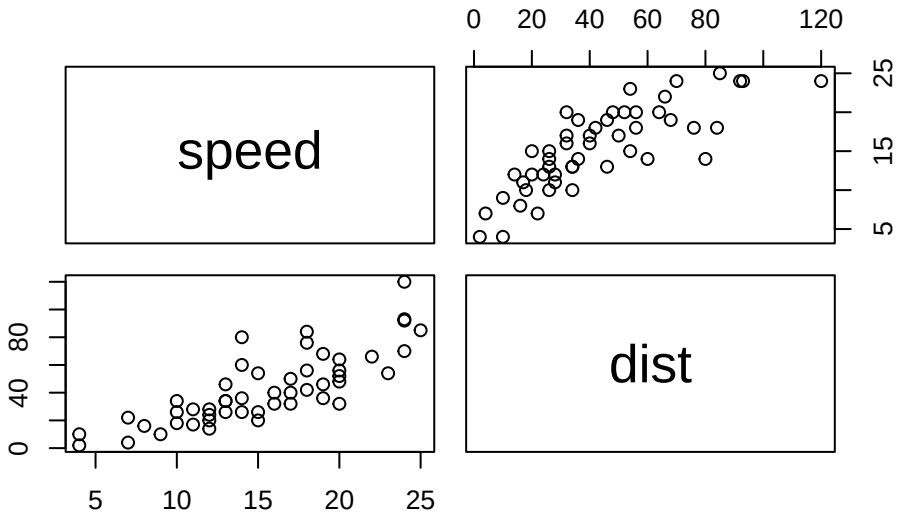
```
?cars
```

Vemos como varían las variables y obtengamos la recta de regresión lineal con el cálculo de β_0 y de β_1 como vimos en las transparencias.

```
plot(dist ~ speed, data = cars)
```




```
pairs(cars)
```



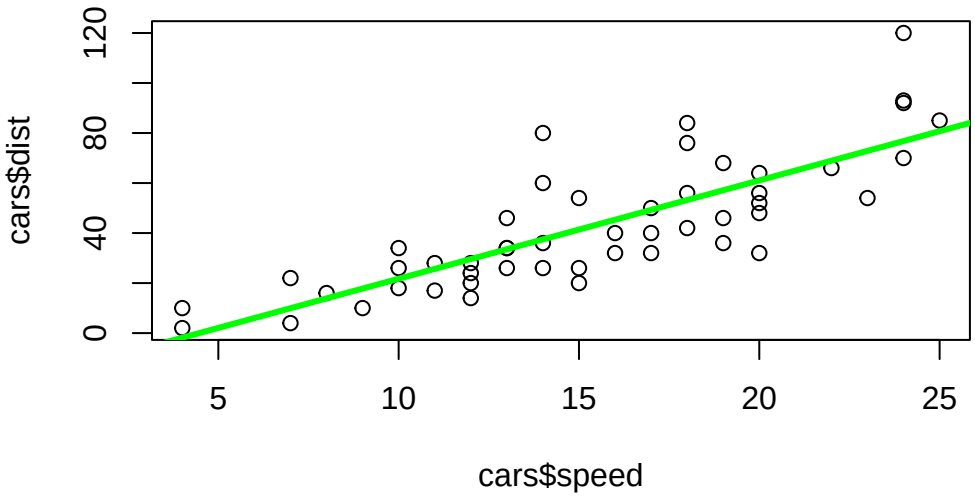

```
# Estimaciones de los coeficientes b_0 y b_1 de la recta de regresion
# y = b_0 + b_1*x usando las formulas
x <- cars$speed
y <- cars$dist
(b1 <- cov(x, y) / var(x))
```

```
[1] 3.932409
```

```r
(b0 <- mean(y) - b1 * mean(x))
```

```
[1] -17.57909
```

```r
plot(cars$dist ~ cars$speed)
abline(a=b0, b=b1, lwd = 3, col = 'green')
```



Veamos que se obtiene lo mismo si hacemos las operaciones matriciales que lleven a las ecuaciones normales:

$$y = X\beta + \epsilon, \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \Rightarrow (X'X)\beta = X'y$$

```r
# Matriz de datos del modelo
X <- cbind(1, x)
head(X)
```

```
     x
[1,] 1 4
[2,] 1 4
[3,] 1 7
[4,] 1 7
[5,] 1 8
[6,] 1 9
```

```r
t(X) %*% X # X'X
```

```
          x
   50   770
x 770 13228
```

```r
length(x); sum(x); sum(x^2); # verificacion
```

```
[1] 50
```

```
[1] 770
```

```
[1] 13228
```

```r
t(X) %*% y # X'y
```

```
   [,1]
   2149
x 38482
```

```r
sum(y); sum(x * y) # verificacion
```

```
[1] 2149
```

```
[1] 38482
```

```r
# Sistema de ecuaciones normales: (X'X)beta = X'y
solve(t(X) %*% X, t(X) %*% y)
```

```
        [,1]
  -17.579095
x   3.932409
```

```r
b0; b1
```

```
[1] -17.57909
```

```
[1] 3.932409
```

Verificamos con la función lm de R

```
# Utilizando la funcion 'lm()'
?lm # para ver la ayuda
m1 <- lm(y ~ x)
m1
```

```

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    -17.579        3.932
```

Ahora con los mismos datos hagamos una regresion polinomial ajustando un polinomio de grado 2:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

```
# Matriz de diseño del modelo
X.2 <- cbind(1, x, 'x^2' = x^2)
head(X.2)
```

```
     x x^2
[1,] 1 4  16
[2,] 1 4  16
[3,] 1 7  49
[4,] 1 7  49
[5,] 1 8  64
[6,] 1 9  81
```

```
t(X.2) %*% X.2 # X'X
```

```
              x     x^2
      50    770   13228
x     770  13228  245090
x^2 13228 245090 4802308
```

```r
length(x); sum(x); sum(x^2); sum(x^3); sum(x^4)
```

```
[1] 50
```

```
[1] 770
```

```
[1] 13228
```

```
[1] 245090
```

```
[1] 4802308
```

```r
t(X.2) %*% y # X'y
```

```
      [,1]
      2149
x    38482
x^2 736548
```

```r
sum(y); sum(x * y); sum(x^2 * y)
```

```
[1] 2149
```

```
[1] 38482
```

```
[1] 736548
```

```r
# Resolviendo el sistema de ecuaciones normales:
solve(t(X.2) %*% X.2, t(X.2) %*% y)
```

```
         [,1]
    2.4701378
x   0.9132876
x^2 0.0999593
```

```r
# Usando la funcion 'lm()'
rpoli2 <- lm(y ~ x + I(x^2))
# Si se hace lm(y ~ x + x^2), el termino x + x^2 lo entiende solo como x.
# La solucion es escribir I(x^2), con lo cual evita confundirlos.
# La otra es crear una columna con los valores x^2 y pedir regresion de y
#contra las demas, esto es, y ~ . (el . quiere decir todas las que no son y, ver rpoli2b mas
rpoli2
```

```
Call:
lm(formula = y ~ x + I(x^2))

Coefficients:
(Intercept)            x       I(x^2)
    2.47014      0.91329      0.09996
```

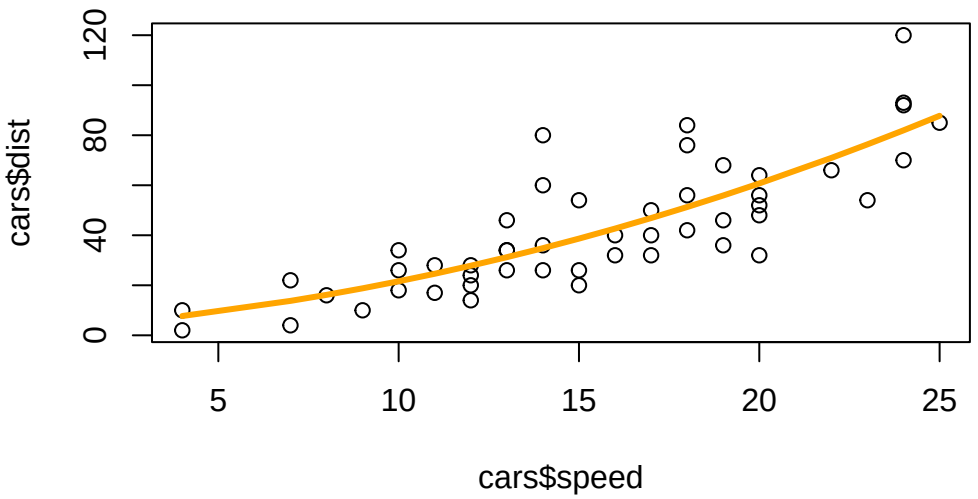
```r
summary(rpoli2)
```

```
Call:
lm(formula = y ~ x + I(x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-28.720  -9.184  -3.188   4.628  45.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.47014   14.81716   0.167    0.868
x            0.91329    2.03422   0.449    0.656
I(x^2)       0.09996    0.06597   1.515    0.136

Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-squared:  0.6673,    Adjusted R-squared:  0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

```r
coef(rpoli2)
```

```
(Intercept)           x      I(x^2)
  2.4701378   0.9132876   0.0999593
```

```r
plot(cars$dist ~ cars$speed)
lines(sort(x), rpoli2$fitted.values[order(x)], col = 'orange', lwd = 3)
```




```r
# Sin I()
autos <- cars
autos <- data.frame(autos, 'speed2'=cars$speed^2)
rpoli2b <- lm(dist ~ ., data = autos)
rpoli2b
```

```
Call:
lm(formula = dist ~ ., data = autos)

Coefficients:
(Intercept)        speed       speed2
    2.47014      0.91329      0.09996
```

```r
rpoli2
```

```
Call:
lm(formula = y ~ x + I(x^2))

Coefficients:
(Intercept)            x       I(x^2)
    2.47014      0.91329      0.09996
```

```r
summary(rpoli2b)
```

```
Call:
lm(formula = dist ~ ., data = autos)

Residuals:
    Min      1Q  Median      3Q     Max
-28.720  -9.184  -3.188   4.628  45.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.47014   14.81716   0.167    0.868
speed        0.91329    2.03422   0.449    0.656
speed2       0.09996    0.06597   1.515    0.136

Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-squared:  0.6673,    Adjusted R-squared:  0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

```r
summary(rpoli2)
```

```
Call:
lm(formula = y ~ x + I(x^2))

Residuals:
    Min      1Q  Median      3Q     Max
-28.720  -9.184  -3.188   4.628  45.152

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.47014   14.81716   0.167    0.868
x            0.91329    2.03422   0.449    0.656
I(x^2)       0.09996    0.06597   1.515    0.136

Residual standard error: 15.18 on 47 degrees of freedom
Multiple R-squared:  0.6673,    Adjusted R-squared:  0.6532
F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

1.2 Segundo Ejemplo con datos simulados

Generemos ahora 50 datos sintéticos donde conocemos la función $f(x) = 2 + 3x$:

$$Y = 2 + 3x + \epsilon,\ \epsilon \sim \mathcal{N}(0, 4)$$

```
set.seed(5)
library('ggplot2')
# Generación de datos

n <- 50
x <- sort(5 * runif(n))
y <- 2 + 3 * x + rnorm(n, sd = 2)
df <- data.frame(x = x, y = y)

fit <- lm(y ~ x)

summary(fit)
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-4.4790 -1.3639 -0.0485  1.5521  3.7921

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.8833     0.5451   5.289 2.99e-06 ***
x             2.6554     0.1909  13.912  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.938 on 48 degrees of freedom
Multiple R-squared:  0.8013,    Adjusted R-squared:  0.7971
F-statistic: 193.5 on 1 and 48 DF,  p-value: < 2.2e-16
```

```
summary(fit)$r.squared #el R^2
```

```
[1] 0.8012769
```

```r
summary(fit)$adj.r.squared #el R^2 ajustado
```

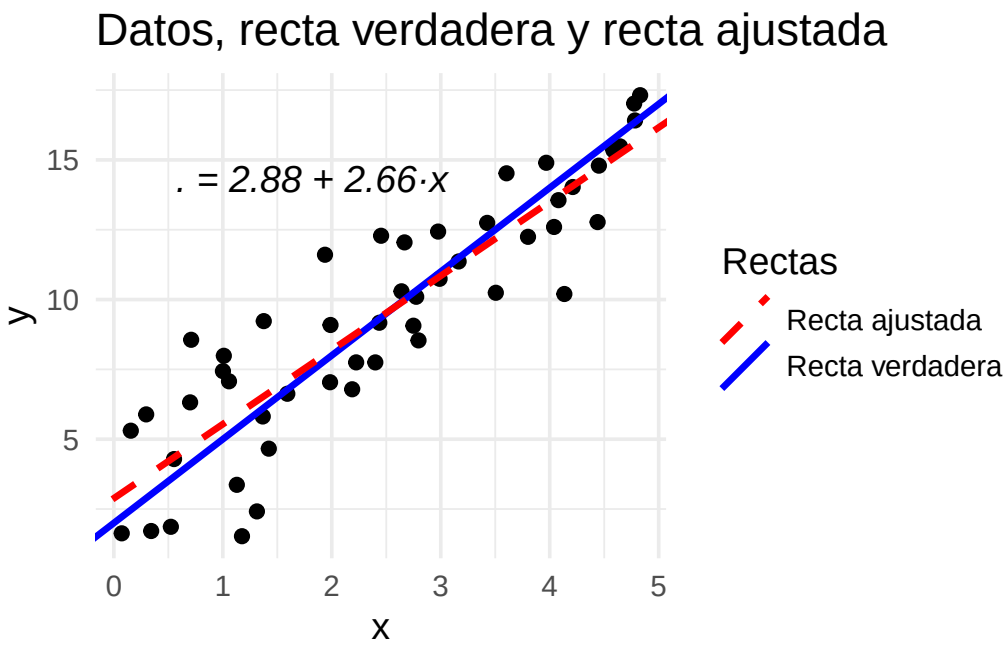
```
[1] 0.7971368
```

```r
summary(fit)$sigma # =raiz{SCR/n-2} el estimador de sigma
```

```
[1] 1.937774
```

```r
beta_0 <- coef(fit)[1]
beta_1 <- coef(fit)[2]

df_true <- data.frame(intercept = 2, slope = 3) # Recta verdadera
df_fit <- data.frame(intercept =beta_0, slope = beta_1) #Recta estimada

# Ecuación para mostrar en el gráfico
eq_label <- paste0("ŷ = ", round(beta_0, 2), " + ", round(beta_1, 2), "·x")

ggplot(df, aes(x, y)) +
  geom_point(size = 2) +
  # Recta verdadera
  geom_abline(data = df_true,
              aes(intercept = intercept, slope = slope, color = "Recta verdadera"),
              linewidth = 1.2) +
  # Recta ajustada
  geom_abline(data = df_fit,
              aes(intercept = intercept, slope = slope, color = "Recta ajustada"),
              linewidth = 1.2, linetype = "dashed") +

  scale_color_manual(values = c("Recta verdadera" = "blue",
                                "Recta ajustada" = "red")) +

  # Ecuación en el gráfico
  annotate("text",
           x = min(df$x) + 0.5,
           y = max(df$y) - 3,
           label = eq_label,
           hjust = 0,
           size = 5,
           fontface = "italic") +
```

```
  labs(title = "Datos, recta verdadera y recta ajustada",
       x = "x",
       y = "y",
       color = "Rectas") +
  theme_minimal(base_size = 14)
```



Los intervalos de confianza para los parámetros se obtienen con el comando confint. El parámetro level permite elegir el nivel de confianza del intervalo (por defecto es 0.95):

```
confint(fit)
```

```
                2.5 %   97.5 %
(Intercept) 1.787296 3.979299
x           2.271664 3.039225
```

```
confint(fit, level = 0.9)
```

```
                 5 %     95 %
(Intercept) 1.969039 3.797557
x           2.335304 2.975585
```

Hagamos predicciones ahora. Dando por válido el modelo ajustado, podemos utilizar la función genérica `predict()` para predecir el valor de nuevas observaciones.

```
# Grilla de predicción (opcional)
new <- data.frame(x = seq(0, 10, length.out = 50)) #50 puntos nuevos
predict(fit, newdat=new, interval = "confidence") #la predicción para
```

```
         fit       lwr       upr
1   2.883298  1.787296  3.979299
2   3.425225  2.396175  4.454275
3   3.967153  3.003338  4.930967
4   4.509080  3.608414  5.409746
5   5.051007  4.210929  5.891086
6   5.592935  4.810291  6.375579
7   6.134862  5.405753  6.863971
8   6.676790  5.996395  7.357185
9   7.218717  6.581109  7.856325
10  7.760645  7.158632  8.362657
11  8.302572  7.727626  8.877518
12  8.844500  8.286847  9.402152
13  9.386427  8.835375  9.937479
14  9.928354  9.372829 10.483880
15 10.470282  9.899469 11.041095
16 11.012209 10.416125 11.608293
17 11.554137 10.923999 12.184274
18 12.096064 11.424425 12.767703
19 12.637992 11.918691 13.357293
20 13.179919 12.407936 13.951902
21 13.721847 12.893119 14.550574
22 14.263774 13.375016 15.152532
23 14.805701 13.854249 15.757154
24 15.347629 14.331312 16.363946
25 15.889556 14.806594 16.972519
26 16.431484 15.280404 17.582563
27 16.973411 15.752989 18.193833
28 17.515339 16.224547 18.806131
29 18.057266 16.695236 19.419297
30 18.599194 17.165186 20.033202
31 19.141121 17.634502 20.647740
32 19.683048 18.103273 21.262823
```

```
33 20.224976 18.571571 21.878381
34 20.766903 19.039456 22.494351
35 21.308831 19.506979 23.110683
36 21.850758 19.974183 23.727334
37 22.392686 20.441104 24.344267
38 22.934613 20.907775 24.961451
39 23.476541 21.374222 25.578859
40 24.018468 21.840468 26.196468
41 24.560395 22.306534 26.814257
42 25.102323 22.772437 27.432209
43 25.644250 23.238192 28.050309
44 26.186178 23.703813 28.668542
45 26.728105 24.169313 29.286897
46 27.270033 24.634702 29.905364
47 27.811960 25.099988 30.523932
48 28.353888 25.565181 31.142594
49 28.895815 26.030288 31.761342
50 29.437742 26.495316 32.380169
```

```r
#cada nuevo x con el intervalo de confianza sobre la media
predict(fit, newdata = new, interval='prediction') #la predicción de
```

```
         fit         lwr       upr
1   2.883298 -1.16407842  6.930674
2   3.425225 -0.60453639  7.454987
3   3.967153 -0.04644573  7.980751
4   4.509080  0.51017597  8.507984
5   5.051007  1.06531246  9.036702
6   5.592935  1.61894893  9.566921
7   6.134862  2.17107208 10.098653
8   6.676790  2.72167022 10.631909
9   7.218717  3.27073329 11.166701
10  7.760645  3.81825296 11.703036
11  8.302572  4.36422265 12.240922
12  8.844500  4.90863759 12.780362
13  9.386427  5.45149484 13.321359
14  9.928354  5.99279328 13.863916
15 10.470282  6.53253366 14.408030
16 11.012209  7.07071858 14.953700
17 11.554137  7.60735247 15.500921
18 12.096064  8.14244154 16.049687
19 12.637992  8.67599381 16.599990
```

```
20 13.179919  9.20801899 17.151819
21 13.721847  9.73852847 17.705165
22 14.263774 10.26753524 18.260013
23 14.805701 10.79505383 18.816349
24 15.347629 11.32110021 19.374158
25 15.889556 11.84569173 19.933421
26 16.431484 12.36884701 20.494121
27 16.973411 12.89058587 21.056237
28 17.515339 13.41092920 21.619748
29 18.057266 13.92989889 22.184633
30 18.599194 14.44751774 22.750869
31 19.141121 14.96380933 23.318433
32 19.683048 15.47879794 23.887299
33 20.224976 15.99250844 24.457443
34 20.766903 16.50496622 25.028841
35 21.308831 17.01619709 25.601465
36 21.850758 17.52622716 26.175289
37 22.392686 18.03508282 26.750289
38 22.934613 18.54279058 27.326436
39 23.476541 19.04937707 27.903704
40 24.018468 19.55486892 28.482067
41 24.560395 20.05929272 29.061498
42 25.102323 20.56267492 29.641971
43 25.644250 21.06504185 30.223459
44 26.186178 21.56641957 30.805936
45 26.728105 22.06683391 31.389377
46 27.270033 22.56631037 31.973755
47 27.811960 23.06487411 32.559046
48 28.353888 23.56254991 33.145225
49 28.895815 24.05936216 33.732268
50 29.437742 24.55533478 34.320150
```

```
#cada nuevo x con un intervalo de confianza para cada nueva observación.
```

1.3 Ejemplo de regresión lineal simple con datos reales

Esta parte está adaptada del muy buen material disponible libremente en https://rpubs.com/Cristina_Gil/Regresion_Lineal_Simple

El conjunto de datos `trees` del paquete dplyr contiene los datos sobre la circunferencia (en pulgadas), altura (en pies) y volumen (en pies cúbicos) del tronco de 31 árboles de cere-

zos.Vamos a ajustar un modelo de regresión lineal simple para predecir el volumen en función del diámetro.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

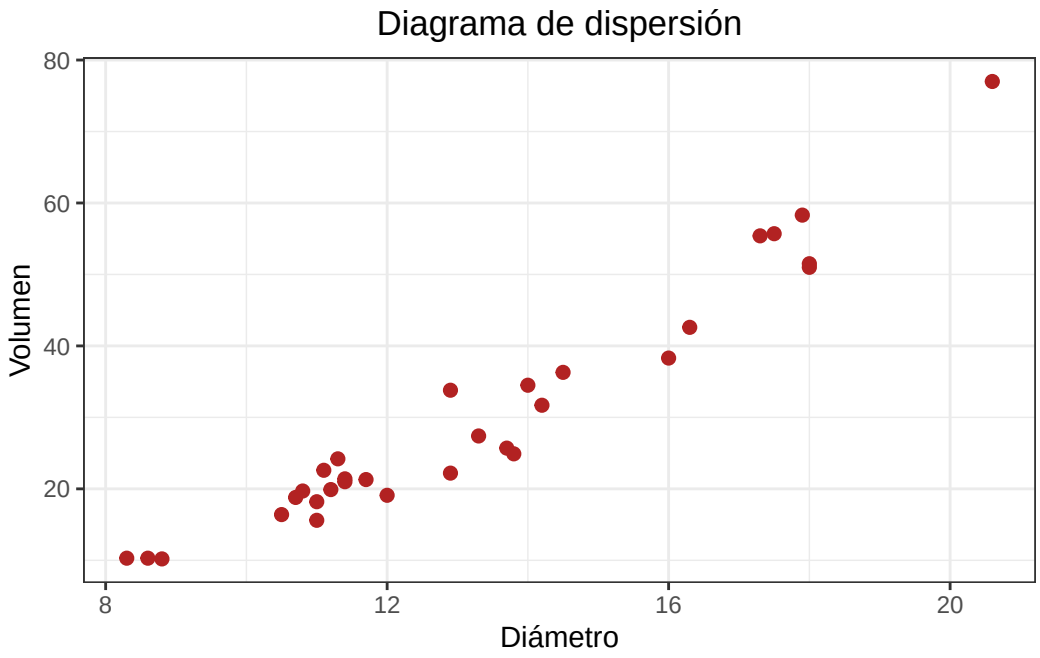
```
head(trees)
```

```
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
```

```
datos=trees
summary(datos)
```

```
     Girth           Height       Volume     
 Min.   : 8.30   Min.   :63   Min.   :10.20  
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40  
 Median :12.90   Median :76   Median :24.20  
 Mean   :13.25   Mean   :76   Mean   :30.17  
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30  
 Max.   :20.60   Max.   :87   Max.   :77.00  
```

```r
ggplot(data = trees, mapping = aes(x = Girth, y = Volume)) +
geom_point(color = "firebrick", size = 2) +
labs(title = "Diagrama de dispersión", x = "Diámetro", y = "Volumen") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))
```




```r
fit <- lm(Volume ~ Girth, data = trees)
summary(fit)
```

```
Call:
lm(formula = Volume ~ Girth, data = trees)

Residuals:
   Min     1Q Median     3Q    Max
-8.065 -3.107  0.152  3.495  9.587

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
Girth         5.0659     0.2474   20.48  < 2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.252 on 29 degrees of freedom
Multiple R-squared:  0.9353,    Adjusted R-squared:  0.9331
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

```r
names(fit)
```

```
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

El summary del modelo contiene los errores estándar, el valor del estadístico t y el correspondiente *p-valor* de ambos parámetros β_0 y β_1. El *p-valor* nos permite determinar si los estimadores de los parámetros son significativamente distintos de 0, es decir, que contribuyen al modelo. El parámetro que suele ser más útil en estos modelos es la pendiente.

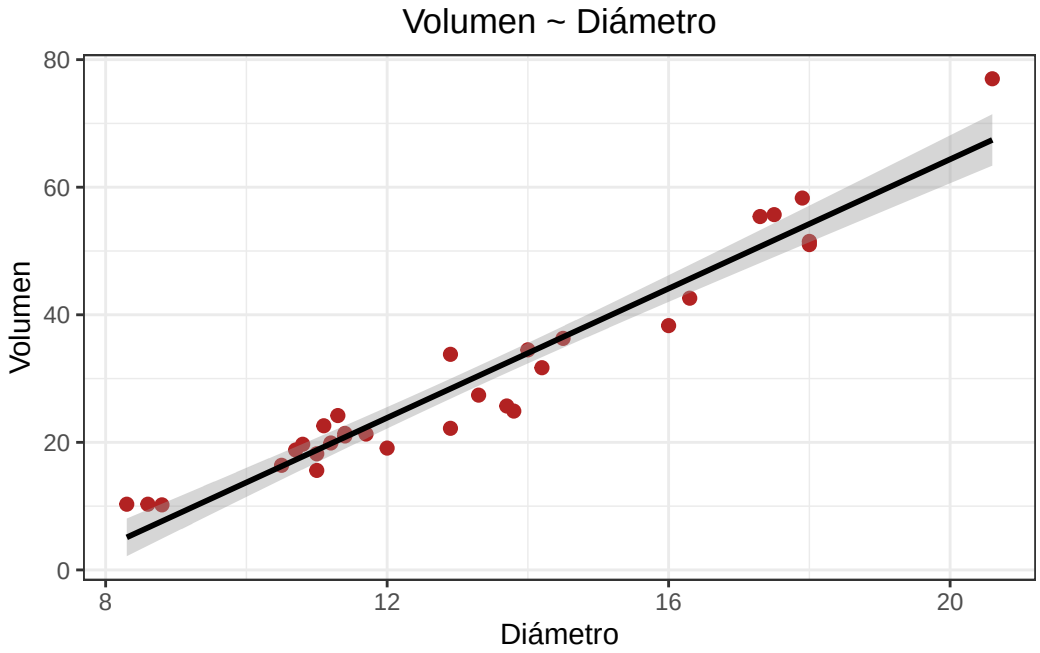
Conclusiones:

1. Tanto la ordenada en el origen como la pendiente son significativas (*p-valor* cercano a 0).
2. El coeficiente de determinación R^2 indica que el modelo es capaz de explicar el 93% de la variabilidad presente en la variable de respuesta (volumen) mediante la variable independiente (diámetro).
3. El *p-valor* del test F indica que es significativamente superior la varianza explicada por el modelo en comparación con la varianza total, por lo que podemos aceptar nuestro modelo como válido y útil.
4. La ecuación del modelo es $Volume = \text{-}\ 36{,}94 + 5{,}065 Girth$ y podemos interpretar que por cada unidad que se incrementa el diámetro, el volumen aumenta en promedio de 5,065 unidades.

Podemos dibujar la recta de mínimos cuadrados, representar el intervalo de confianza (límites superior e inferior) para cada predicción con la función `geom_smooth( )` de `ggplot2`. Esto permite identificar la región en la que se encuentra el promedio de la variable respuesta según el modelo generado y para un determinado nivel de confianza.

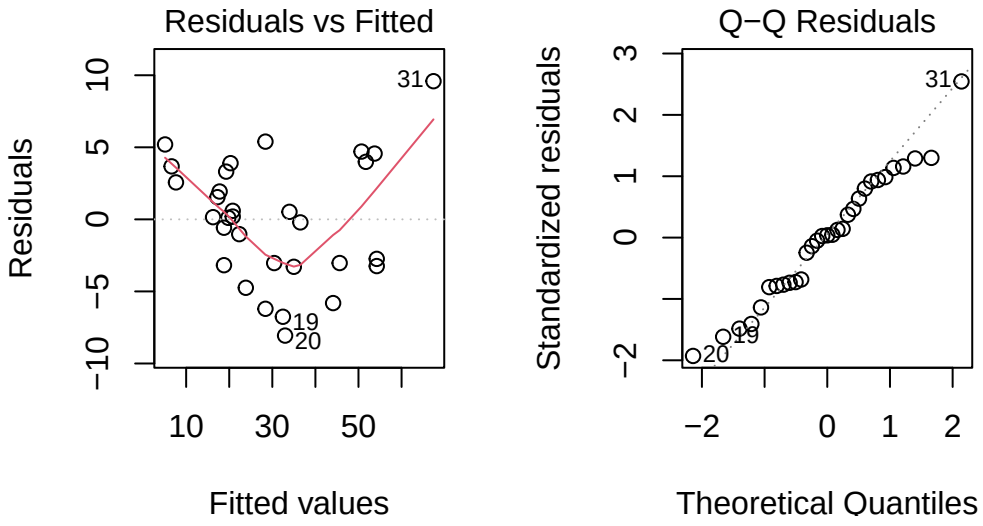
```r
ggplot(data = trees, mapping = aes(x = Girth, y = Volume)) +
geom_point(color = "firebrick", size = 2) +
geom_smooth(method = "lm", se = TRUE, color = "black") +
labs(title = "Volumen ~ Diámetro", x = "Diámetro", y = "Volumen") +
theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```

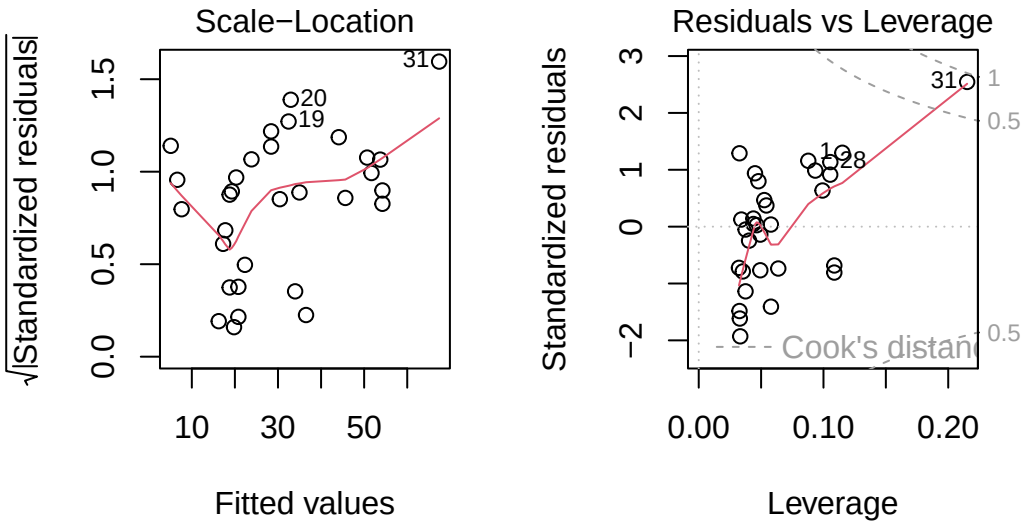
```
`geom_smooth()` using formula = 'y ~ x'
```



Para evaluar las condiciones que permiten dar como válido el modelo lineal, miramos principalmente del análisis de los residuos y si los residuos no rechazan la hipótesis de normalidad:

```
par(mfrow=c(1,2))
plot(fit)
```






```
# Contraste de hipótesis (normalidad de los residuos)
shapiro.test(fit$residuals)
```

```
    Shapiro-Wilk normality test

data:  fit$residuals
W = 0.97889, p-value = 0.7811
```

Veamos ahora las predicciones de nuevas observaciones a partir del modelo. En nuestro ejemplo, podemos predecir el volumen a partir de mediciones del diámetro de nuevos árboles:

```
# Volumen PROMEDIO que esperaríamos de árboles de 15 pulgadas
predict(fit, data.frame(Girth = 15), interval = "confidence")
```

```
# Volumen esperado de UN árbol de 15 pulgadas
predict(modelo.lineal, data.frame(Girth = 15), interval = "prediction")
```

Observar que el intervalo de predicción es mayor que el de confianza. Podemos interpretar la predicción de la siguiente manera: se espera que en promedio un árbol de diámetro 15 tenga un volumen de 39,04. Podemos afirmar con un 95% de confianza que el verdadero valor promedio se encuentra entre (37,24 – 40.84), mientras que el intervalo de predicción para un solo árbol de este diámetro será de (30,16 – 47, 92).

1.4 Ejemplo donde el modelo es globalmente significativo pero cada coeficiente por separado no lo es:

```r
set.seed(123)
n <- 100

# Aca simulo dos variables fuertemente correlacionadas
x1 <- rnorm(n)
x2 <- x1 + rnorm(n, sd = 0.01)   # casi igual a x1
cor(x1,x2)
```

```
[1] 0.999944
```

```r
# Variable respuesta como combinación de ambas
y <- 3*x1 + 3*x2 + rnorm(n)

# Ajustar modelo múltiple
modelo <- lm(y ~ x1 + x2)
summary(modelo)
```

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
    Min      1Q  Median      3Q     Max
-1.8730 -0.6607 -0.1245  0.6214  2.0798

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.13507    0.09614   1.405    0.163
x1           0.48570    9.89483   0.049    0.961
x2           5.38113    9.89947   0.544    0.588

Residual standard error: 0.9513 on 97 degrees of freedom
Multiple R-squared:   0.97, Adjusted R-squared:  0.9694
F-statistic:  1567 on 2 and 97 DF,  p-value: < 2.2e-16
```

El p-valor del test F es nulo mientras que los de los coeficientes por separado es mayor que 0.1. Esto pasa porque el modelo no puede separar el efecto de `x1` del de `x2` ya que son casi la misma variable.

1.5 Regresión lineal si la variable es categórica

En $y = \beta_0 + \beta_1 X$, si X es continua

- β_1 mide el **cambio promedio en** y cuando X aumenta una unidad.
- Se interpreta como una **pendiente**.

$$\beta_1 = \frac{\Delta y}{\Delta X}$$

Por ejemplo, si X es la edad (en años) y $\beta_1 = 2$,
entonces un año adicional se asocia, en promedio, con un aumento de **2 unidades** en y.

Cuando X es categórica (o dummy 0/1), β_1 representa la **diferencia promedio en (y)** entre dos grupos: la categoría $X = 1$ y la categoría de referencia $X = 0$: $\beta_1 = \mathbb{E}[Y \mid X = 1] - \mathbb{E}[Y \mid X = 0]$

Por ejemplo, si (X=1) = tratamiento y (X=0) = control,entonces β_1 es el **efecto promedio del tratamiento**.

1.6 Ejemplo con una variable categórica

Vamos a simular un dataset con:

- `ingresos` (variable respuesta)
- `sexo` (categórica: Hombre / Mujer)
- `edad` (numérica)
- `educacion` (numérica)

Y después ajustamos una regresión múltiple.

```
set.seed(123)
n <- 200
# Variables
sexo <- factor(sample(c("Hombre", "Mujer"), n, replace = TRUE))
edad <- rnorm(n, mean = 40, sd = 10)
educacion <- rnorm(n, mean = 14, sd = 2)
# Generamos ingresos con efecto verdadero:
# - Mujeres ganan 2000 menos (ajustado)
# - Edad aumenta ingresos
# - Educación también
ingresos <- 30000 +
```

```r
            1500*edad +
            2500*educacion +
            ifelse(sexo=="Mujer", -2000, 0) +
            rnorm(n, sd = 5000)

datos <- data.frame(ingresos, sexo, edad, educacion)

# Modelo múltiple
modelo <- lm(ingresos ~ sexo + edad + educacion, data = datos)
summary(modelo)
```

```
Call:
lm(formula = ingresos ~ sexo + edad + educacion, data = datos)

Residuals:
     Min       1Q   Median       3Q      Max
-13620.0  -3450.8    300.4   3413.2  12202.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30559.40    2917.35  10.475  < 2e-16 ***
sexoMujer   -2362.90     704.73  -3.353  0.00096 ***
edad         1525.85      36.48  41.831  < 2e-16 ***
educacion    2365.03     173.51  13.630  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4938 on 196 degrees of freedom
Multiple R-squared:  0.9114,    Adjusted R-squared:   0.91
F-statistic: 671.7 on 3 and 196 DF,  p-value: < 2.2e-16
```

La interpretación es que **una mujer gana, en promedio, 2362 unidades menos que un hombre, manteniendo fija la edad y la educación.** No significa "las mujeres ganan 2000 menos en promedio", sino "dada la misma edad y el mismo nivel educativo". El coeficiente de una categoría representa la diferencia en Y entre esa categoría y la referencia, ajustada por el resto de las variables.

1.6 Comparación de modelos

```r
set.seed(123)

n <- 300

metros <- rnorm(n, mean = 120, sd = 25)
habitaciones <- rpois(n, lambda = 3)
antiguedad <- runif(n, 0, 60)

# variable categórica: barrio
barrio <- factor(sample(c("Centro", "Norte", "Sur"), n, replace = TRUE))

# Precio "real"
precio <- 50000 +
  1100*metros +
  15000*habitaciones -
  800*antiguedad +
  ifelse(barrio=="Centro", 30000, ifelse(barrio=="Norte", 15000, 0)) +
  rnorm(n, sd = 30000)

mod_reducido <- lm(precio ~ metros + habitaciones + antiguedad)
mod_completo <- lm(precio ~ metros + habitaciones + antiguedad + barrio)
summary(mod_reducido)
```

```
Call:
lm(formula = precio ~ metros + habitaciones + antiguedad)

Residuals:
   Min     1Q Median     3Q    Max
-88683 -25829     63  24415  85217

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  61163.63   10961.66   5.580 5.44e-08 ***
metros        1102.24      81.31  13.556  < 2e-16 ***
habitaciones 15374.41    1124.71  13.670  < 2e-16 ***
antiguedad    -733.24     116.05  -6.318 9.69e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 33190 on 296 degrees of freedom
Multiple R-squared:  0.5626,    Adjusted R-squared:  0.5582
F-statistic: 126.9 on 3 and 296 DF,  p-value: < 2.2e-16
```

```
summary(mod_completo)
```

```

Call:
lm(formula = precio ~ metros + habitaciones + antiguedad + barrio)

Residuals:
   Min     1Q Median     3Q    Max
-79986 -20049   -236  21323  78610

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   82984.60   10857.33   7.643 3.02e-13 ***
metros         1051.53      75.76  13.880  < 2e-16 ***
habitaciones  15260.46    1046.07  14.588  < 2e-16 ***
antiguedad     -767.22     107.90  -7.110 8.82e-12 ***
barrioNorte  -11975.31    4399.07  -2.722  0.00687 **
barrioSur    -30785.72    4381.60  -7.026 1.48e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30770 on 294 degrees of freedom
Multiple R-squared:  0.6266,    Adjusted R-squared:  0.6202
F-statistic: 98.66 on 5 and 294 DF,  p-value: < 2.2e-16
```

```
anova(mod_reducido, mod_completo)
```

```
Analysis of Variance Table

Model 1: precio ~ metros + habitaciones + antiguedad
Model 2: precio ~ metros + habitaciones + antiguedad + barrio
  Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
1    296 3.2609e+11
2    294 2.7841e+11  2 4.7679e+10 25.175 8.095e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
AIC(mod_reducido, mod_completo)
```

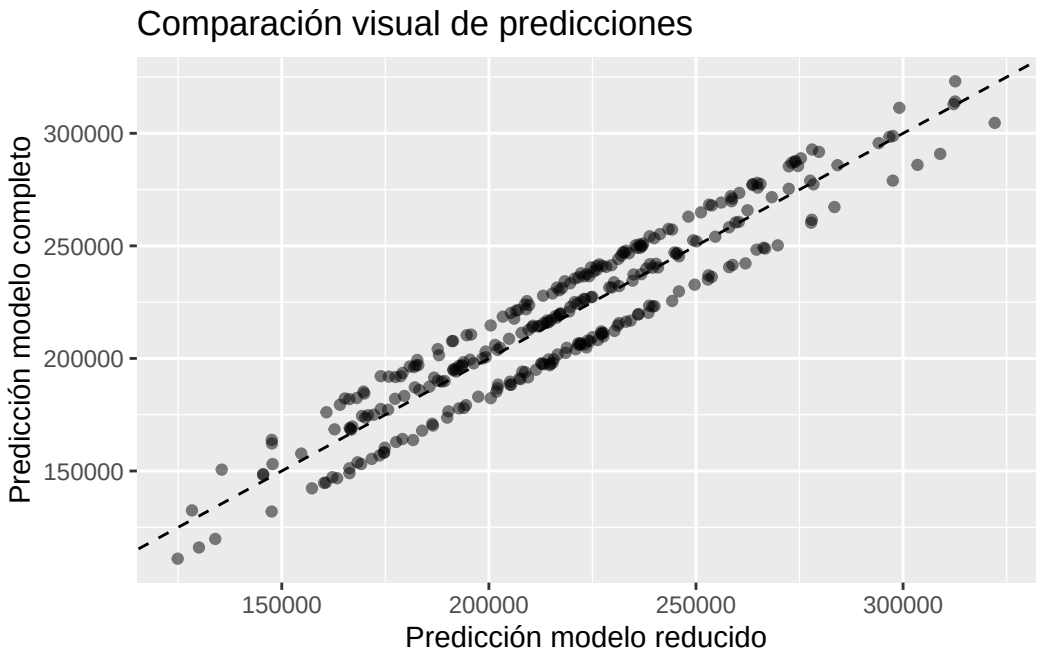
```
             df      AIC
mod_reducido  5 7103.358
mod_completo  7 7059.936
```

```r
BIC(mod_reducido, mod_completo)
```

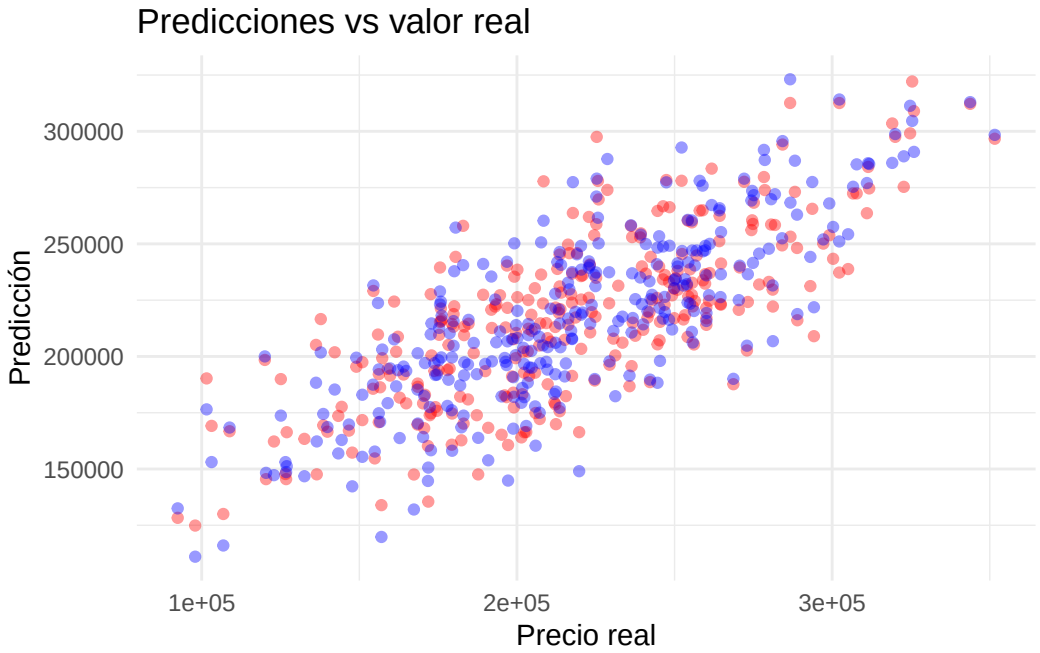
```
             df      BIC
mod_reducido  5 7121.877
mod_completo  7 7085.862
```

- El **Test F** indica si agregar la variable *barrio* mejora el ajuste.
- AIC y BIC verifican si el modelo más complejo vale la pena.

```r
library(ggplot2)

pred_df <- data.frame(
precio_real = precio,
pred_reducido = predict(mod_reducido),
pred_completo = predict(mod_completo)
)

ggplot(pred_df, aes(x = pred_reducido, y = pred_completo)) +
geom_point(alpha = 0.5) +
geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
labs(title = "Comparación visual de predicciones",
x = "Predicción modelo reducido",
y = "Predicción modelo completo")
```




```
library(ggplot2)

pred_df <- data.frame(
  precio_real = precio,
  reducido = predict(mod_reducido),
  completo = predict(mod_completo)
)

ggplot(pred_df, aes(x = precio_real)) +
  geom_point(aes(y = reducido), alpha = 0.4, color = "red") +
  geom_point(aes(y = completo), alpha = 0.4, color = "blue") +
  labs(title = "Predicciones vs valor real",
       x = "Precio real",
       y = "Predicción") +
  theme_minimal()
```



La nube azul más cerca de la diagonal indica mejor modelo.

2. Regresión logística

En el ejemplo siguiente vamos a trabajar sobre el conjunto de datos default del paquete ISLR que contiene información sobre diez mil clientes. El objetivo aquí es predecir qué clientes incumplirán con su deuda de tarjeta de crédito.

Las variables explicativas son si es o no estudiante (student), el saldo promedio (balance) que el cliente tiene pendiente en su tarjeta de crédito después de realizar su pago mensual y su ingreso (Income) y la variable a explicar (default) es un factor con niveles No y Sí que indica si el cliente incumplió su deuda.

```
library('ISLR2')
datos=Default
head(datos)
```

```
  default student   balance    income
1      No      No  729.5265 44361.625
2      No     Yes  817.1804 12106.135
3      No      No 1073.5492 31767.139
4      No      No  529.2506 35704.494
5      No      No  785.6559 38463.496
6      No     Yes  919.5885  7491.559
```

```r
summary(datos)
```

```
 default    student       balance           income
 No :9667   No :7056   Min.   :   0.0   Min.   :  772
 Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340
                       Median : 823.6   Median :34553
                       Mean   : 835.4   Mean   :33517
                       3rd Qu.:1166.3   3rd Qu.:43808
                       Max.   :2654.3   Max.   :73554
```

Vamos hacer un modelo de regresión logística usando la función `glm()` para modelos lineales generalizados, una clase de modelos que generalizan los modelos lineales en los que se incluye el modelo logístico. Para ello especificamos el argumento `family = binomial`.

```r
modelo <- glm(default~., data = datos, family = binomial)
summary(modelo)
```

```
Call:
glm(formula = default ~ ., family = binomial, data = datos)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

Los valores Null deviance y Residual deviance corresponden a los residuos del modelo nulo sin predictor y al modelo completo, respectivamente. Los grados de libertad son iguales a cantidad de observaciones menos cantidad de parametros.

Se puede obtener intervalos de confianza sobre los parametros:

```
confint(object=modelo,level=0.9)
```

```
Waiting for profiling to be done...
```

```
                     5 %           95 %
(Intercept) -1.169639e+01 -1.007628e+01
studentYes  -1.034772e+00 -2.571454e-01
balance      5.364197e-03  6.127463e-03
income      -1.046035e-05  1.653459e-05
```

Según el modelo que elaboramos, el ser estudiante o no y el balance son los únicos predictores estadísticamente significativos. El coeficiente negativo para *student* en la regresión logística múltiple indica que, para un valor fijo de *balance* e *income*, un estudiante es menos propenso a incumplir que un no estudiante.

En una regresión logística, el **modelo nulo** (*null model*) es el modelo más sencillo posible: solo contiene un **intercepto** y **no incluye ninguna variable explicativa**. Su forma es

$$\log\left(\frac{p}{1-p}\right) = \beta_0$$

donde p es la probabilidad global de que la variable respuesta tome el valor 1 (por ejemplo, la probabilidad de default en el dataset *Default*).

El modelo nulo asume que todas las observaciones tienen la misma probabilidad y no se usan covariables para distinguir entre individuos. Por lo tanto, el estimador de p es simplemente la proporción de casos positivos $p = \frac{n_1}{n}$ donde n_1 es el número de observaciones con respuesta 1 y n es el total de observaciones. El intercepto se obtiene de la manera siguiente: $n = 10000, n_1 = 333$ de donde $p = 0.0333$ por lo que $\hat{\beta}_0 = \log\left(\frac{\hat{p}}{1-\hat{p}}\right)$ ce que toda persona tiene una probabilidad 3.3% de incumplir, sin tener en cuenta su `balance`, `income`, ni si es `student`.

```
p_hat <- mean(datos$default == "Yes")
beta0_hat <- log(p_hat / (1 - p_hat))
beta0_hat
```

```
[1] -3.368331
```

La null deviance corresponde al modelo nulo, que solo incluye un intercepto. En el dataset `Default` del paquete la probabilidad estimada de default es $\hat{p} = \frac{n_1}{n} = 0.0333$

La null deviance se calcula como: Null deviance $= -2\left[n_1 \log(\hat{p}) + n_0 \log(1-\hat{p})\right]$

```r
n1 <- sum(datos$default == "Yes")
n0 <- sum(datos$default == "No")
n <- n1 + n0
p_hat <- n1 / n #Proba
null_dev_manual <- -2 * (n1 * log(p_hat) + n0 * log(1 - p_hat))
null_dev_manual
```

```
[1] 2920.65
```

que da lo mismo que:

```r
mod_nulo <- glm(default ~ 1, data = datos, family = binomial)
mod_nulo$null.deviance
```

```
[1] 2920.65
```

La bondad de ajuste se mide mediante la desvianza del modelo y su relación a la desvianza del modelo nulo:

```r
anova(modelo, test='Chisq')
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: default

Terms added sequentially (first to last)

        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                     9999     2920.7
student  1    11.97      9998     2908.7 0.0005416 ***
balance  1  1337.00      9997     1571.7 < 2.2e-16 ***
income   1     0.14      9996     1571.5 0.7115139
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El p-valor rechaza la hipótesis nula de que nuestro modelo se asemeje a un modelo tan simple como el modelo nulo.

Observemos que el coeficiente de la variable student es negativo.

Qué significa?: **si dos personas tienen el mismo balance e income, la persona estudiante tiene *menor* probabilidad de default.**

En el dataset:

- Los **estudiantes tienden a tener balances más altos** (cuentas más cargadas).
- Y el **balance es el predictor dominante** del default (fuertemente relacionado).

Entonces:

- Sin ajustar por balance: los estudiantes parecen tener más default (coeficiente positivo).
- Ajustando por balance: a igual balance, los estudiantes defaultan *menos (*coeficiente negativo).

Esto es exactamente la **paradoja de Simpson**: *una relación que es positiva en los promedios globales, se vuelve negativa cuando se controla por otras variables relevantes.* Este fenómeno ocurre cuando una variable (student) está correlacionada con otra variable fuerte (balance), generando conclusiones contradictorias si no se controlan las covariables.

- Los estudiantes defaultan más *porque tienden a tener balances más altos.*
- Pero **a igualdad de balance**, los estudiantes defaultan menos.
- Por eso el signo del coeficiente cambia entre el modelo simple (coef positivo) y el modelo múltiple (coef negativo).

```r
mod1 <- glm(default ~ student, data = datos, family = binomial)
summary(mod1)
```

```
Call:
glm(formula = default ~ student, family = binomial, data = datos)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
studentYes   0.40489    0.11502    3.52 0.000431 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 2908.7  on 9998  degrees of freedom
```

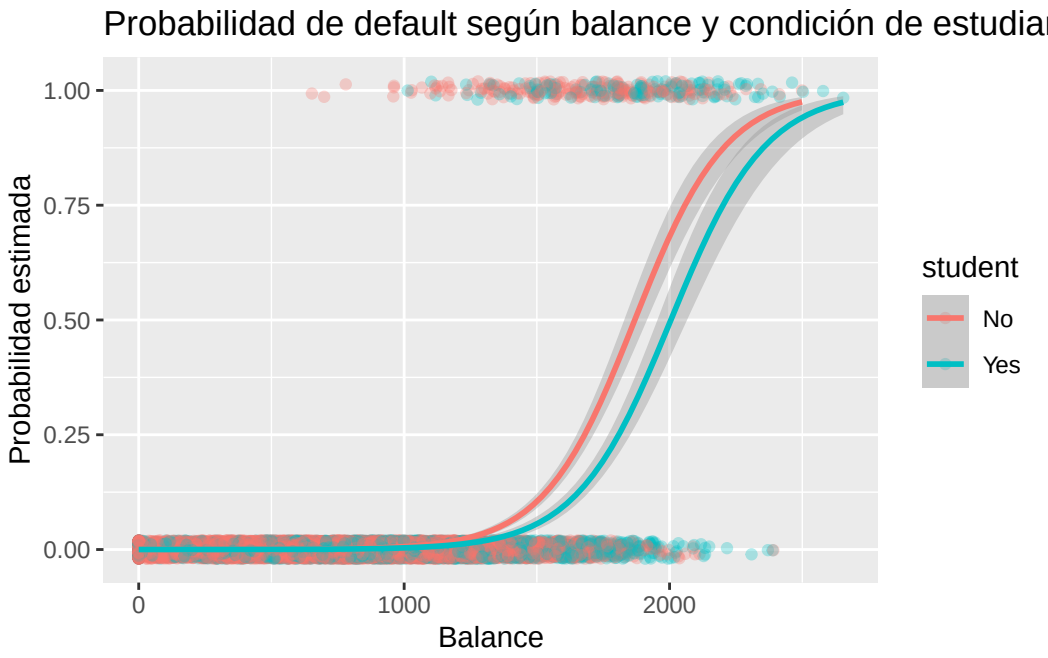
```
    AIC: 2912.7

    Number of Fisher Scoring iterations: 6
```

```
mod2 <- glm(default ~ balance + income + student, data = datos, family = binomial)
summary(mod2)
```

```

Call:
glm(formula = default ~ balance + income + student, family = binomial,
    data = datos)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

```
ggplot(datos, aes(x = balance, y = as.numeric(default == "Yes"), color = student)) +
geom_jitter(height = 0.02, alpha = 0.3) +
geom_smooth(method = "glm", method.args = list(family = "binomial")) +
labs(
title = "Probabilidad de default según balance y condición de estudiante",
y = "Probabilidad estimada",
x = "Balance"
)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



- A **igual balance**, los estudiantes tienen menor probabilidad.
- Pero el grupo de estudiantes tiene balances más altos en promedio, y esto **revierte** la comparación global.

2.2 Reduccion en la desvianza del modelo

```
datos <- Default

mod_completo <- glm(default ~ balance + income + student,
                    data = datos, family = binomial)

mod_completo$deviance          # devianza residual
```

```
[1] 1571.545
```

```
mod_completo$df.residual       # grados de libertad residuales
```

```
[1] 9996
```

```r
summary(mod_completo)
```

```
Call:
glm(formula = default ~ balance + income + student, family = binomial,
    data = datos)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

```r
anova(mod_nulo, mod_completo, test="Chisq")
```

```
Analysis of Deviance Table

Model 1: default ~ 1
Model 2: default ~ balance + income + student
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1      9999     2920.7
2      9996     1571.5  3   1349.1 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La devianza residual se define como: $D = -2\left[\ell(\hat{p}) - \ell_{\text{sat}}\right]$ donde $\ell(\hat{p})$ es la log-verosimilitud del modelo ajustado y ℓ_{sat} es la log-verosimilitud del modelo saturado (que vale 0). Toda desvianza es una medida relativa al modelo saturado.

Para datos binarios ($y_i \in \{0,1\}$), la fórmula explícita utilizada por glm es: $D = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{p}_i}\right) + (1-y_i)\log\left(\frac{1-y_i}{1-\hat{p}_i}\right)\right]$.

$D \geq 0$, cuanto mayor es D peor es el ajuste, si el modelo predice perfectamente ($\hat{p}_i = y_i$), entonces $D = 0$.

- La **null deviance** (modelo sin predictores) 2920.6;
- La **residual deviance** 1571.5 muestra cuánto error queda al usar `balance`, `income` y `student`.
- La gran reducción (1349) indica que los predictores, especialmente `balance`, mejoran mucho el ajuste. Esto se formaliza con `anova(mod_nulo, mod_completo, test="Chisq")`

2.3 Otro ejemplo de regresion logística.

Estos datos son de 100 pacientes a quienes le relevamos la edad (AGE) y si tiene o no una enfermedad coronaria (CHD): 1 si presenta la enfermedad, 0 si no.

```
datos=read.table('AGECHD.csv',dec=',',sep=';',header=T)
summary(datos)
```

```
      AGE             CHD        
 Min.   :20.00   Min.   :0.00  
 1st Qu.:34.75   1st Qu.:0.00  
 Median :44.00   Median :0.00  
 Mean   :44.38   Mean   :0.43  
 3rd Qu.:55.00   3rd Qu.:1.00  
 Max.   :69.00   Max.   :1.00  
```

```
datos$CHD=as.factor(datos$CHD)
summary(datos)
```

```
      AGE        CHD    
 Min.   :20.00   0:57  
 1st Qu.:34.75   1:43  
 Median :44.00         
 Mean   :44.38         
 3rd Qu.:55.00         
 Max.   :69.00         
```

```r
modelo=glm(CHD~AGE,family=binomial,datos)
summary(modelo)
```

```
Call:
glm(formula = CHD ~ AGE, family = binomial, data = datos)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.30945    1.13365  -4.683 2.82e-06 ***
AGE          0.11092    0.02406   4.610 4.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 136.66  on 99  degrees of freedom
Residual deviance: 107.35  on 98  degrees of freedom
AIC: 111.35

Number of Fisher Scoring iterations: 4
```

Los coeficientes β_0 y β_1 resultan ser significativos. El modelo es $ln(p/(1-p)) = -5.30945 + 0.11092X$ y la probabilidad de padecer una enfermedad coronaria en función de la edad X es

$$p = \frac{1}{(1 + e^{-5.30945+0.11092X})}$$

Interpretamos el coeficiente $\beta_1 = 0.11092 = ln(OR)$ de la siguiente manera : el $OR = e^{\beta_1} = 1.1173$ indica como aumenta el riesgo de padecer enfermedad coronaria al aumentar la edad de un año. Si la edad aumenta de 20 años este riesgo aumenta de $e^{20*\beta_1} = 9,2$

Una predicción:

```r
pp=predict(modelo,data.frame(AGE=36),type="response")
```

La bondad de ajuste se mide mediante la desvianza del modelo y su relación a la desvianza del modelo nulo:

```r
anova(modelo, test='Chisq')
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: CHD

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                    99     136.66
AGE   1    29.31        98     107.35 6.168e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El p-valor rechaza la hipótesis nula de que nuestro modelo se asemeje a un modelo tan simple como el modelo nulo.