# Capítulo 7

# Estimación del modelo de regresión lineal con R.

En los capítulos anteriores han aparecido fragmentos de código ilustrando el modo de llevar a cabo diversos cálculos en R. Se presenta aquí la función `lm` y algunas otras, para ilustrar tanto los conceptos teóricos adquiridos como la potencia del entorno de modelización proporcionado por R.

Este capítulo es eminentemente práctico y puede ser omitido sin pérdida de continuidad por lectores que no estén interesados en utilizar R como herramienta de cálculo.

## 7.1. Tipología de variables explicativas.

Interesará distinguir dos tipos de variables: *cualitativas* (también llamadas categóricas) y *numéricas.* Las variables cualitativas se desglosan a su vez en *nominales* y *ordinales.*

Una variable cualitativa nominal especifica una característica o atributo que puede tomar un número entero (y habitualmente pequeño) de *niveles* o estados. Por ejemplo, una variable Zona podría tomar los niveles o estados: "Europa", "Africa", "Asia", "America" y "Oceanía". Requeriremos que las categorías sean exhaustivas, de forma que todo caso muestral pueda recibir un valor. Si es preciso, podemos crear una categoría especial como "Otros" o "Resto".

Una variable cualitativa ordinal se diferencia únicamente de una nominal en que hay una ordenación natural entre las categorías. Por ejemplo, en una variable como NIVEL DE ESTUDIOS podríamos tener categorías como: "Sin estudios", "Primarios", "Secundarios", "Superiores". La diferencia esencial con las variables nominales es que hay una ordenación entre los distintos niveles: cada una de las categorías en el orden en que se hay escrito implica "más" estudios que la categoría precedente. No había, en cambio, en el ejemplo anterior una ordenación natural entre las zonas geográficas.

Las variables que hemos denominado *numéricas* pueden en principio ponerse en correspondencia con un intervalo de números reales. Sería el caso de variables como PESO ó TEMPERATURA (aunque en la práctica el número de estados que pueden tomar es finito a causa de la precisión también finita de los instrumentos de medida que empleamos).

En cierto sentido, los tres tipos de variables, en el orden en que se han descrito, reflejan una mayor finura o contenido informativo: una variable numérica puede convertirse en ordinal fijando intervalos: por ejemplo, TEMPERATURA podría convertirse en una variable ordinal con niveles "Frío", "Templado" y "Caliente", al precio de un cierto sacrificio de información: dos temperaturas de, por ejemplo, 80C y 93C podrían ambas convertirse en "Caliente", perdiéndose la información de que la segunda es superior a la primera.

Análogamente, una variable ordinal puede tratarse como nominal, haciendo abstracción de su orden, también al precio de sacrificar cierta información.

**Observación 7.1** En general, no interesará "degradar" una variable tratándola como un tipo inferior, aunque en algunos casos, puede convenirnos hacerlo. Por ejemplo, si examinamos la influencia de la renta sobre el consumo de un cierto bien en una muestra de familias, medir la renta en euros da al coeficiente $\beta$ asociado la interpretación de "Incremento de consumo asociado a un incremento de renta de un euro". Típicamente, tendrá un valor muy pequeño. Además, el suponer una dependencia lineal del consumo sobre la renta será en la mayoría de los casos poco realista. En tal caso, podría convenirnos redefinir la variable renta en categorías. Los coeficientes estimados serán más fácilmente interpretables, y tendremos un modelo más flexible, que no fuerza una relación lineal entre renta y consumo. (Adicionalmente, si la variable se obtiene por encuestación, los sujetos podrían ser más veraces al encuadrarse en intervalos amplios de renta que al responder directamente sobre su valor.)

## 7.2. Factores y *dataframes.*

R ofrece excelentes facilidades para tratar variables de diferentes tipos como regresores. En la jerga de R, una variable cualitativa se denomina *factor*.

Hay factores ordinarios, que permiten manejar variables cualitativas nominales, y factores ordenados *(ordered factors)*, para variables cualitativas ordinales. El Ejemplo 8.1 a continuación ilustra la manera de operar con ellos.

**R: Ejemplo 7.1** Para que una variable sea un factor, hay que especificarlo. Observemos el siguiente fragmento de código:

```
> Zona.chr <- c("Europa", "Europa", "Asia",
+     "Africa", "America", "Oceanía", "Asia")
> Zona <- as.factor(Zona.chr)
> Zona.chr
[1] "Europa"  "Europa"  "Asia"    "Africa"
[5] "America" "Oceanía" "Asia"
> Zona
[1] Europa  Europa  Asia    Africa  America
[6] Oceanía Asia
Levels: Africa America Asia Europa Oceanía
```

Obsérvese que `Zona.chr` y `Zona` se imprimen de manera similar, aunque uno es una cadena de caracteres y otro un factor. La diferencia estriba en las comillas en el primer caso y la línea adicional especificando los niveles en el segundo. Podemos preguntar la clase de objeto con la función `class` o ver la structura con la función `str` para ver la diferencia:

```
> class(Zona.chr)
[1] "character"
> class(Zona)
[1] "factor"
> str(Zona.chr)
 chr [1:7] "Europa" "Europa" "Asia" ...
> str(Zona)
```

```
 Factor w/ 5 levels "Africa","America",..: 4 4 3 1 2 5 3
```

Un factor tiene definidos *niveles,* en tanto una cadena de caracteres no:

```
> levels(Zona.chr)
```

```
NULL
```

```
> levels(Zona)
```

```
[1] "Africa"  "America" "Asia"    "Europa"
[5] "Oceanía"
```

Veamos ahora como definir un factor ordenado:

```
> Estudios <- ordered(c("Superiores", "Medios",
+     "Medios", "Primarios", "Ningunos"))
```

Si no se especifica lo contrario, el orden de los niveles se determina por el orden alfabético de sus denominaciones. Esto haría que en `Estudios` el nivel "Medios" precediera a "Ningunos", y éste a "Primarios", lo que es indeseable:

```
> Estudios
```

```
[1] Superiores Medios     Medios     Primarios
[5] Ningunos
4 Levels: Medios < Ningunos < ... < Superiores
```

Para especificar un orden, podemos crear el objeto `Estudios` así:

```
> Estudios <- ordered(c("Superiores", "Medios",
+     "Medios", "Primarios", "Ningunos",
+     "Medios", "Primarios"), levels = c("Ningunos",
+     "Primarios", "Medios", "Superiores"))
> Estudios
```

```
[1] Superiores Medios     Medios     Primarios
[5] Ningunos   Medios     Primarios
4 Levels: Ningunos < Primarios < ... < Superiores
```

Podemos de modo análogo reordenar los niveles. Si, por ejemplo, queremos revertir el orden, podemos hacerlo así:

```
> Estudios.1 <- ordered(Estudios, levels = c("Superiores",
+     "Medios", "Primarios", "Ningunos"))
```

o, mas simplemente podemos revertir el orden de los niveles mediante la funcion `rev`, sin necesidad de enumerarlos. Comprobemos a continuación que obtenemos en ambos casos el mismo objeto con el orden de los niveles deseado:

```
> Estudios.2 <- ordered(Estudios, levels = rev(levels(Estudios)))
> Estudios.1
[1] Superiores Medios     Medios     Primarios
[5] Ningunos   Medios     Primarios
4 Levels: Superiores < Medios < ... < Ningunos
> Estudios.2
[1] Superiores Medios     Medios     Primarios
[5] Ningunos   Medios     Primarios
4 Levels: Superiores < Medios < ... < Ningunos
```

Una manipulación que deseamos hacer de ordinario con factores no ordenados es la de poner en primer lugar uno de los niveles, el *nivel de referencia*. Podemos lograrlo cómodamente con la función `relevel`

```
> Zona
[1] Europa  Europa  Asia    Africa  America
[6] Oceanía Asia
Levels: Africa America Asia Europa Oceanía
> Zona <- relevel(Zona, ref = "Asia")
> Zona
[1] Europa  Europa  Asia    Africa  America
[6] Oceanía Asia
Levels: Asia Africa America Europa Oceanía
```

Veremos en el Ejemplo 8.5 la utilidad de esto. Definamos ahora dos variables numéricas:

```
> Ingresos <- c(13456, 12345, 3456, 1234,
+     6789, 4567, 2300)
> Mortalidad <- c(0.003, 0.004, 0.01, 0.02,
+     0.006, 0.005, 0.015)
```

Podemos reunir variables de diferentes tipos en una *dataframe*. A todos los efectos, es como una matriz, pero presenta la peculiaridad de que sus columnas pueden ser de diferentes tipos:

```
> Datos <- data.frame(Zona, Estudios, Ingresos,
+     Mortalidad)
> Datos
```

```
     Zona   Estudios Ingresos Mortalidad
1  Europa Superiores    13456      0.003
2  Europa     Medios    12345      0.004
3    Asia     Medios     3456      0.010
4  Africa  Primarios     1234      0.020
5 America   Ningunos     6789      0.006
6 Oceanía     Medios     4567      0.005
7    Asia  Primarios     2300      0.015
```

```
> str(Datos)
```

```
'data.frame':        7 obs. of  4 variables:
 $ Zona      : Factor w/ 5 levels "Asia","Africa",..: 4 4 1 2 3 5 1
 $ Estudios  : Ord.factor w/ 4 levels "Ningunos"<"Primarios"<..: 4 3 3 2 1 3 2
 $ Ingresos  : num  13456 12345 3456 1234 6789 ...
 $ Mortalidad: num  0.003 0.004 0.01 0.02 0.006 0.005 0.015
```

Una *dataframe* tiene la misma representación interna que una lista. Podemos referirnos a sus términos como a los elementos de una lista, o proporcionando índices de fila y columna:

```
> Datos$Ingresos
```

```
[1] 13456 12345  3456  1234  6789  4567  2300
```

```
> Datos[[3]]
```

```
[1] 13456 12345  3456  1234  6789  4567  2300
```

```
> Datos[, "Ingresos"]
```

```
[1] 13456 12345  3456  1234  6789  4567  2300
```

```
> Datos[3, 2:3]
```

```
  Estudios Ingresos
3   Medios     3456
```

FIN DEL EJEMPLO ■

Una *dataframe* provee un entorno de evaluación. Muchas funciones en R admiten un argumento `data` que permite especificar la *dataframe* en la que es preciso buscar las variables que se nombran. Adicionalmente, la instrucción `attach` hace que las columnas en una *dataframe* sean accesibles como variables definidas en el espacio de trabajo. El Ejemplo 8.2, continuación del Ejemplo 8.1, lo ilustra.

**R: Ejemplo 7.2** Comencemos por eliminar del espacio de trabajo algunas variables:

```
> rm(Zona, Estudios, Ingresos, Mortalidad)
```

Si ahora tecleáramos el nombre de alguna de ellas obtendríamos un error. No obstante, tras invocar la función `attach` sus columnas son visibles como si variables en el espacio de trabajo se tratase:

```
> attach(Datos)
> Zona
```

```
[1] Europa  Europa  Asia    Africa  America
[6] Oceanía Asia
Levels: Asia Africa America Europa Oceanía
```

La función `detach` revierte el efecto de `attach`:

```
> detach(Datos)
```

Si un objeto existe en el espacio de trabajo, su valor oculta el de la columna del mismo nombre en una *dataframe* "attacheada":

```
> Zona <- c("a", "b", "c")
> attach(Datos)
```

```
        The following object(s) are masked _by_ .GlobalEnv :

         Zona
```

```
> Zona
[1] "a" "b" "c"
```

FIN DEL EJEMPLO ■

## 7.3. Fórmulas

Bastantes funciones en R hacen uso de *fórmulas*. Permiten, entre otras cosas, especificar de modo simple modelos de regresión, simplemente nombrando a la izquierda del símbolo ~ la variable respuesta, y a la derecha las variables regresores.

Una fórmula puede proporcionarse como argumento directamente para estimar un modelo de regresión lineal ordinaria (mediante la función `lm`; un ejemplo en la Sección 8.4), regresión lineal generalizada (mediante la función `glm`) o regresión no lineal (mediante la función `nlme` en el paquete del mismo nombre). Por razones didácticas, sin embargo, exploraremos primero el modo en que los diferentes tipos de variables son tratados en una fórmula por la función `model.matrix`.

La función `model.matrix` recibe como argumentos una fórmula y, opcionalmente, una *dataframe* en la que los términos de la fórmula son evaluados. Proporciona la matriz de diseño asociada al modelo que especificamos en la fórmula.

**R: Ejemplo 7.3** Supongamos que deseamos investigar la relación entre la variable `Mortalidad` y la variable `Ingresos`. Podemos construir la matriz de diseño así:

```
> X <- model.matrix(Mortalidad ~ Ingresos,
+     data = Datos)
> X
  (Intercept) Ingresos
1           1    13456
2           1    12345
3           1     3456
4           1     1234
```

```
5           1     6789
6           1     4567
7           1     2300
attr(,"assign")
[1] 0 1
```

Como podemos ver, se ha añadido automáticamente una columna de "unos". Si esto fuera indeseable por algún motivo, podríamos evitarlo incluyendo como regresor "-1".

```
> X <- model.matrix(Mortalidad ~ -1 + Ingresos,
+     data = Datos)
> X
```

```
  Ingresos
1    13456
2    12345
3     3456
4     1234
5     6789
6     4567
7     2300
attr(,"assign")
[1] 1
```

Obsérvese que la variable `Mortalidad` no juega ningún papel en la conformación de la matriz de diseño. Podríamos omitirla y dar sólo el lado derecho de la fórmula, así:

```
> X <- model.matrix(~Ingresos, data = Datos)
> X
```

```
  (Intercept) Ingresos
1           1    13456
2           1    12345
3           1     3456
4           1     1234
5           1     6789
6           1     4567
7           1     2300
attr(,"assign")
[1] 0 1
```

FIN DEL EJEMPLO ■

La comodidad que proporciona la utilización de fórmulas se hace más evidente, sin embargo, cuando tenemos regresores cualitativos. El Ejemplo 8.4 lo ilustra.

**R: Ejemplo 7.4** Consideremos un modelo que tiene como regresores `Zona`, `Ingresos` y `Estudios`. Podemos construir su matriz de diseño así:

```
> X <- model.matrix(~Zona + Estudios + Ingresos,
+     data = Datos)
```

Las variables `Zona` y `Estudios` son cualitativas. Requieren ser tratadas de manera especial, y la función `model.matrix` así lo hace. Veamos la matriz de diseño que proporciona:

```
> X
  (Intercept) ZonaAfrica ZonaAmerica ZonaEuropa
1           1          0           0          1
2           1          0           0          1
3           1          0           0          0
4           1          1           0          0
5           1          0           1          0
6           1          0           0          0
7           1          0           0          0
  ZonaOceanía Estudios.L Estudios.Q Estudios.C
1           0    0.67082        0.5    0.22361
2           0    0.22361       -0.5   -0.67082
3           0    0.22361       -0.5   -0.67082
4           0   -0.22361       -0.5    0.67082
5           0   -0.67082        0.5   -0.22361
6           1    0.22361       -0.5   -0.67082
7           0   -0.22361       -0.5    0.67082
  Ingresos
1    13456
2    12345
3     3456
4     1234
5     6789
6     4567
7     2300
```

```
attr(,"assign")
[1] 0 1 1 1 1 2 2 2 3
attr(,"contrasts")
attr(,"contrasts")$Zona
[1] "contr.treatment"

attr(,"contrasts")$Estudios
[1] "contr.poly"
```

La variable `Ingresos` (numérica) ha sido dejada tal cual. La variable `Zona` es cualitativa nominal, y requiere ser desglosada en tantas columnas como niveles tiene (así, el $\beta$ asociado a cada columna recoge el efecto del correspondiente nivel). Eso es lo que ha hecho `model.matrix`, salvo que se ha omitido uno de los niveles (el primero) para evitar la multicolinealidad exacta que se hubiera producido de otro modo. El nivel omitido (Asia) pasa así a formar parte del caso de referencia: la función `relevel` (ver Ejemplo 8.1) permitiría cambiar fácilmente el nivel que forma parte del caso de referencia.

El tratamiento de las variables ordinales como `Estudios` es algo más elaborado. En una variable ordinal hay una noción natural de proximidad entre niveles: el nivel de estudios `Medios` está más cerca del nivel `Superiores` que el nivel `Primarios`. Lo que hace `model.matrix` es conceptualmente equivalente a hacer lo siguiente (detalles en la Observación 8.2, pág. 100):

1. Asignar a cada nivel de `Estudios` un valor entero, respetando el orden de la variable: "Ningunos"=1, "Primarios"=2, "Medios"=3 y "Superiores"=4.
2. Con la variable `Estudios` así codificada, crear tantas columnas para la variable `Estudios` como niveles tenga, de la forma: $(\texttt{Estudios})^0$, $(\texttt{Estudios})^1$, $(\texttt{Estudios})^2$, $(\texttt{Estudios})^3$.

La primera columna, que es constante, es automáticamente desechada si en la matriz de diseño existe columna de "unos", para evitar la multicolinealidad. Las restantes son rotuladas con las letras "L" (**L**inear), "Q" (**Q**uadratic), "C" (**C**ubic), y así sucesivamente.

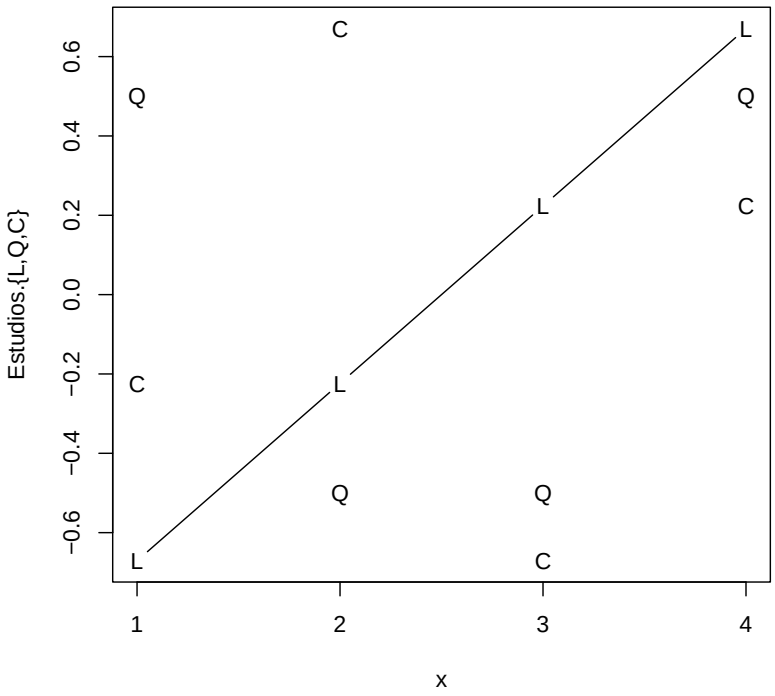
Si empleamos todas las columnas que `model.matrix` crea para una variable ordinal, obtenemos exactamente el mismo subespacio que habríamos obtenido con columnas de ceros y unos como las empleadas para una variable nominal: la ventaja de utilizar una base de dicho subespacio como la que `model.matrix` construye, es que permite en ocasiones realizar una modelización más simple: podemos, a voluntad, emplear en un modelo de regresión algunas, varias o todas las columnas

como regresores, para modelizar un efecto más o menos "suave" sobre la variable respuesta.

FIN DEL EJEMPLO ■

**Observación 7.2** Se indica en el Ejemplo 8.4 que el efecto de una variable ordinal se recoge de modo *conceptualmente equivalente* a construir potencias de orden creciente de la variable ordinal codificada por valores enteros que respetan el orden. Ayudará representar gráficamente las columnas correspondientes de la matriz `X` frente a los enteros codificando los niveles de la variable `Estudios`. Para ello, eliminamos primero niveles duplicados y representaremos los restantes:

```
> x <- as.numeric(Datos[, "Estudios"])
> i <- !duplicated(x)
> plot(x[i], X[i, "Estudios.L"], type = "b",
+     pch = "L", xaxp = c(1, 4, 3), xlab = "x",
+     ylab = "Estudios.{L,Q,C}")
> points(x[i], X[i, "Estudios.Q"], pch = "Q")
> points(x[i], X[i, "Estudios.C"], pch = "C")
```



Hemos dibujado una línea uniendo las "L" para destacar su crecimiento lineal. Las "Q" puede verse que se sitúan sobre una parábola y las "C" sobre una función cúbica.

Un vistazo al gráfico anterior muestra, sin embargo, que el término lineal, por ejemplo, no toma los valores 1, 2, 3 4, ni el cuadrático 1, 4, 9, 16. En efecto,

```
> X[i, 6:8]
```

```
  Estudios.L Estudios.Q Estudios.C
1    0.67082        0.5    0.22361
2    0.22361       -0.5   -0.67082
4   -0.22361       -0.5    0.67082
5   -0.67082        0.5   -0.22361
```

En realidad se han rescalado las columnas *y se han ortogonalizado:*

```
> round(crossprod(X[i, 6:8]))
```

```
           Estudios.L Estudios.Q Estudios.C
Estudios.L          1          0          0
Estudios.Q          0          1          0
Estudios.C          0          0          1
```

Ello se hace por razones de conveniencia numérica y de interpretación.

Aunque por razones didácticas hemos construido primero la matriz de diseño y extraido luego un subconjunto de filas y columnas para ver como se codificaba la variable `Estudios`, R proporciona un modo más simple de hacerlo:

```
> contrasts(Datos[, "Estudios"])
```

```
                .L   .Q       .C
Ningunos   -0.67082  0.5 -0.22361
Primarios  -0.22361 -0.5  0.67082
Medios      0.22361 -0.5 -0.67082
Superiores  0.67082  0.5  0.22361
```

**Observación 7.3** El anterior es el comportamiento "por omisión" de la función `model.matrix`. Podemos alterarlo especificando distintos modos de desdoblar los factores y factores ordenados. Ello se hace invocando la función `options` de modo similar al siguiente:

```
options(contrasts=c("contr.treatment","contr.poly"))
```

La primera opción en el argumento `contrasts` se aplica a los factores, la segunda a los factores ordenados. Por ejemplo, para los factores podemos especificar que se desdoblen en tantas columnas como niveles haya, sin incluir ningún nivel en el caso de referencia. Para ello, deberemos proporcionar `contr.sum` como primer valor de `contrasts`:

```
options(contrasts=c("contr.sum","contr.poly"))
```

Véase la documentación de `contrasts` para más detalles.

Adicionalmente, podemos invocar directamente las funciones

```
contr.sum, contr.treatment, contr.poly, contr.helmert
```

para obtener información sobre el diferente modo en que quedaría codificado un factor. Por ejemplo,

```
> NivelEstudios <- levels(Datos[, "Estudios"])
> contr.sum(NivelEstudios)
           [,1] [,2] [,3]
Ningunos      1    0    0
Primarios     0    1    0
Medios        0    0    1
Superiores   -1   -1   -1
> contr.treatment(NivelEstudios)
           Primarios Medios Superiores
Ningunos           0      0          0
Primarios          1      0          0
Medios             0      1          0
Superiores         0      0          1
> contr.poly(NivelEstudios)
              .L   .Q       .C
[1,] -0.67082  0.5 -0.22361
[2,] -0.22361 -0.5  0.67082
[3,]  0.22361 -0.5 -0.67082
[4,]  0.67082  0.5  0.22361
```

Obsérvese que mientras `contrasts` se invoca tomando como argumento un factor, las funciones `contr.sum` y similares toman como argumento *el vector de niveles* de un factor.

## 7.4. La función `lm`.

La función `lm` es un instrumento potente y cómodo de utilizar para el análisis de regresión lineal. Puede utilizarse con tan solo dos argumentos: una fórmula y una *dataframe* que suministra los valores para evaluar las expresiones en dicha fórmula. Por ejemplo, así:

```
ajuste <-  lm(y ~ x1 + x2 + x4, data=datos)
```

La función `lm` construye entonces la matriz de diseño mediante la función `model.matrix` y estima el modelo deseado, suministrando un cúmulo de información sobre la estimación. El Ejemplo 8.5 a continuación proporciona detalles.

**R: Ejemplo 7.5** Veamos en primer lugar los datos que utilizaremos. Se trata de datos correspondientes a 47 estados en EE.UU. y referidos al años 1960. Forman parte del paquete MASS (soporte del libro Venables and Ripley (1999b)) que hemos de cargar (mediante una instrucción `library(MASS)`). Tras hacerlo, podemos obtener información detallada sobre los datos tecleando `help(UScrime)`.

```
> library(MASS)
> UScrime[1:3, 1:5]
```

```
    M So  Ed Po1 Po2
1 151  1  91  58  56
2 143  0 113 103  95
3 142  1  89  45  44
```

```
> str(UScrime)
```

```
'data.frame':        47 obs. of  16 variables:
 $ M   : int  151 143 142 136 141 121 127 131 157 140 ...
 $ So  : int  1 0 1 0 0 0 1 1 1 0 ...
 $ Ed  : int  91 113 89 121 121 110 111 109 90 118 ...
 $ Po1 : int  58 103 45 149 109 118 82 115 65 71 ...
 $ Po2 : int  56 95 44 141 101 115 79 109 62 68 ...
 $ LF  : int  510 583 533 577 591 547 519 542 553 632 ...
 $ M.F : int  950 1012 969 994 985 964 982 969 955 1029 ...
 $ Pop : int  33 13 18 157 18 25 4 50 39 7 ...
 $ NW  : int  301 102 219 80 30 44 139 179 286 15 ...
 $ U1  : int  108 96 94 102 91 84 97 79 81 100 ...
 $ U2  : int  41 36 33 39 20 29 38 35 28 24 ...
 $ GDP : int  394 557 318 673 578 689 620 472 421 526 ...
```

```
 $ Ineq: int  261 194 250 167 174 126 168 206 239 174 ...
 $ Prob: num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
 $ Time: num  26.2 25.3 24.3 29.9 21.3 ...
 $ y   : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

La función `str` permite ver la estructura de cualquier objeto en R. Lo que muestra en el fragmento anterior es que `UScrime` es una *dataframe*. En este caso, todas las variables son numéricas, algunas reales (`num`) y otras enteras (`int`). Vemos también que tiene 47 filas (=observaciones) y 16 columnas (=posibles regresores).

Probemos ahora a hacer una regresión[1]. La variable `y` (tasa de criminalidad) podemos relacionarla con la desigualdad(`Ineq`), probabilidad de ser encarcelado (`Prob`) y con un indicador de Estado sureño (`So`):

```
> fit <- lm(y ~ Ineq + Prob + So, data = UScrime)
> fit

Call:
lm(formula = y ~ Ineq + Prob + So, data = UScrime)

Coefficients:
(Intercept)         Ineq         Prob
    1538.36        -1.58     -8698.46
         So
     242.99
```

El objeto `fit`, al imprimirlo, proporciona una información muy sumaria: apenas la descripción del modelo ajustado y los coeficientes estimados. El empleo de la función `summary`, sin embargo, proporciona un estadillo con información mucho más completa.

```
> summary(fit)

Call:
lm(formula = y ~ Ineq + Prob + So, data = UScrime)

Residuals:
   Min     1Q Median     3Q    Max
-662.8 -163.8  -56.1   82.5 1057.4
```

[1]No se afirma que el modelo que ensayamos sea el mejor en ningún sentido: es sólo una ilustración. El Capítulo 13 abordará la cuestión de cómo seleccionar modelos.

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1538.36     345.84    4.45    6e-05
Ineq           -1.58       1.95   -0.81   0.4220
Prob        -8698.46    2725.42   -3.19   0.0026
So            242.99     169.48    1.43   0.1589

(Intercept) ***
Ineq
Prob        **
So
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 353 on 43 degrees of freedom
Multiple R-squared: 0.22,        Adjusted R-squared: 0.166
F-statistic: 4.05 on 3 and 43 DF,  p-value: 0.0127
```

Desmenucemos la salida anterior. Se imprime, en primer lugar, el modelo ajustado y unos estadísticos sobre los residuos (mínimo, máximo y cuartiles, es decir, valores dejando a su izquierda el 25 %, 50 % y 75 % de los residuos; el segundo cuartil es la mediana). A continuación, tenemos un estadillo proporcionando para cada regresor mencionado al margen:

1. Su $\hat{\beta}_i$ (bajo `Estimate`).
2. Su $\hat{\sigma}_{\hat{\beta}_i}$ (bajo `Std. Error`).
3. Su estadístico $t$,
$$\frac{\hat{\beta}_i}{\hat{\sigma}_{\hat{\beta}_i}}$$
(bajo `t value`).
4. La probabilidad bajo la hipótesis nula $H_0 : \beta_i = 0$ de obtener un valor del estadístico $t$ tan o más alejado de cero que el obtenido (bajo `Pr(>|t|)`).

A continuación tenemos
$$\sqrt{\frac{SSE}{N-p}},$$
(`Residual standard error`), que estima $\sigma_\epsilon$, los grados de libertad $N - p$, (`43 degrees of freedom`), $R^2$ (que toma el valor 0.22) y $\overline{R}^2$ (`Adjusted R-squared`; este último estadístico será introducido en el Capítulo 13). Finalmente, tenemos el estadístico $Q_h$ para contrastar

significación conjunta de la regresión, como se indica en la Sección 7.2.2 (`F-statistic`). Aquí toma el valor 4.05. Dicho valor deja a su derecha en una distribución $\mathcal{F}_{3,43}$ una cola de probabilidad 0.0127, que es el nivel de significación conjunto de la regresión ajustada.

El objeto compuesto `fit` contiene la información que ha permitido imprimir todos los anteriores resultados y mucha otra, cuyos nombres son autoexplicativos:

```
> attributes(fit)
```

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

$class
[1] "lm"
```

Podemos referirnos a los componentes de `fit` y emplearlos en cálculos subsiguientes. Por ejemplo, para obtener la suma de cuadrados de los residuos, SSE, podríamos hacer:

```
> SSE <- sum(fit$residuals^2)
> SSE
```

```
[1] 5363970
```

El estadillo anterior sugería que el regresor `Prob` era muy significativo, en tanto los restantes no lo eran. Podemos contrastar la hipótesis $H_0$ : $\beta_{\texttt{Ineq}} = \beta_{\texttt{So}} = 0$ del modo sugerido al final del Ejemplo 7.2, pág. 82: ajustamos una segunda regresión eliminando los regresores `Ineq` y `So`,

```
> fit.h <- lm(y ~ Prob, data = UScrime)
```

calculamos la suma de cuadrados de sus residuos,

```
> SSE.h <- sum(fit.h$residuals^2)
```

y a continuación el estadístico $Q_h$ asociado a la hipótesis y los grados de libertad del mismo:

```
> N <- nrow(UScrime)
> q <- 2
> p <- 4
> Qh <- ((SSE.h - SSE)/q)/(SSE/(N - p))
> Qh
```

```
[1] 1.0417
```

La probabilidad que el valor 1.0417 del estadístico deja en la cola a su derecha es

```
> 1 - pf(Qh, q, N - p)
```

```
[1] 0.3616
```

lo que sugiere que podemos prescindir de dichos dos regresores.

La instrucción `anova` proporciona una descomposición de la suma de cuadrados de los residuos correpondiente a cada regresor *cuando se introducen en el orden dado*. Compárese por ejemplo,

```
> anova(fit)
```

```
Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value Pr(>F)
Ineq       1  220530  220530    1.77  0.191
Prob       1 1040010 1040010    8.34  0.006 **
So         1  256417  256417    2.06  0.159
Residuals 43 5363970  124743
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

con:

```
> fit2 <- lm(y ~ Prob + Ineq + So, data = UScrime)
> anova(fit2)
```

```
Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value Pr(>F)
Prob       1 1257075 1257075   10.08 0.0028 **
```

```
Ineq       1    3466    3466    0.03 0.8684
So         1  256417  256417    2.06 0.1589
Residuals 43 5363970  124743
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

FIN DEL EJEMPLO ■

No hay ninguna necesidad ni aparente ventaja en hacerlo así, pero a efectos puramente ilustrativos re-estimaremos la regresión anterior convirtiendo previamente la variable indicadora `So` (Estado del Sur) en una variable nominal y la variable `Ineq` en una variable ordinal (o factor ordenado). Para lo primero, basta que reemplacemos la columna `So` de la *dataframe* del siguiente modo:

```
> UScrime[, "So"] <- factor(UScrime[, "So"],
+     labels = c("Norte", "Sur"))
```

Para la segunda variable, dividiremos su recorrido en tres intervalos, y a continuación definimos un factor ordenado con tres categorías:

```
> Temp <- ordered(cut(UScrime[, "Ineq"],
+     breaks = 3), labels = c("Baja", "Media",
+     "Alta"))
> UScrime[, "Ineq"] <- Temp
```

Podemos ahora repetir la estimación anterior:

**R: Ejemplo 7.6** *(continuación del Ejemplo 8.5)*

```
> fit3 <- lm(y ~ Prob + Ineq + So, data = UScrime)
> summary(fit3)

Call:
lm(formula = y ~ Prob + Ineq + So, data = UScrime)

Residuals:
   Min     1Q Median     3Q    Max
-641.9 -195.5  -55.4  124.3 1059.5
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1212.4      134.8    8.99  2.4e-11
Prob         -9013.8     2717.7   -3.32   0.0019
Ineq.L        -143.2      132.7   -1.08   0.2866
Ineq.Q         -10.6      110.4   -0.10   0.9238
SoSur          284.8      184.3    1.55   0.1298

(Intercept) ***
Prob        **
Ineq.L
Ineq.Q
SoSur
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 355 on 42 degrees of freedom
Multiple R-squared: 0.232,        Adjusted R-squared: 0.159
F-statistic: 3.17 on 4 and 42 DF,  p-value: 0.0229
```

La variable ordinal `Ineq` da lugar a tres términos (constante, omitido por colineal con la columna de unos, lineal y cuadrático). La variable nominal `So` se desglosa también en dos: el nivel "Norte" se integra en el caso de referencia y el parámetro restante mide el efecto deferencial del nivel "Sur" respecto al nivel "Norte". A título ilustrativo, podemos ajustar la anterior regresión empleando un diferente desdoblamiento del regresor cualitativo `So`:

```
> options(contrasts = c("contr.sum", "contr.poly"))
> fit4 <- lm(y ~ Prob + Ineq + So, data = UScrime)
> summary(fit4)

Call:
lm(formula = y ~ Prob + Ineq + So, data = UScrime)

Residuals:
   Min     1Q Median     3Q    Max
-641.9 -195.5  -55.4  124.3 1059.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1354.7      151.0    8.97  2.6e-11
Prob         -9013.8     2717.7   -3.32   0.0019
```

```
Ineq.L        -143.2      132.7   -1.08   0.2866
Ineq.Q         -10.6      110.4   -0.10   0.9238
So1           -142.4       92.1   -1.55   0.1298

(Intercept) ***
Prob        **
Ineq.L
Ineq.Q
So1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 355 on 42 degrees of freedom
Multiple R-squared: 0.232,        Adjusted R-squared: 0.159
F-statistic: 3.17 on 4 and 42 DF,  p-value: 0.0229
```

(Véase la Observación 8.3.) Vemos un sólo regresor asociado a `So1`, el primer nivel de `So`; el asociado al segundo nivel es su opuesto, ya que `contr.sum` fuerza los coeficientes asociados a un regresor nominal a sumar cero.

Si observamos los dos ajustes, vemos que son idénticos. Lo único que se altera es la interpretación de los parámetros. En `fit3`, el tratarse de un Estado del Sur tenía como efecto incrementar la tasa de criminalidad en 284.8, respecto de la tasa prevalente en un Estado del Norte de análogas características. La parametrización en el model `fit4` expresa lo mismo de otro modo: en un Estado del Norte, la criminalidad desciende en -142.4 sobre el nivel promedio de Norte y Sur, mientras que en un Estado del Sur aumenta en 142.4. La diferencia entre ambos niveles continúa siendo 284.8.

Puede encontrarse una discusión exhaustiva de las diferentes opciones de parametrización disponibles en Venables and Ripley (1999a), Sec. 6.2.

FIN DEL EJEMPLO ■

## 7.5. Lectura recomendada.

**Sobre R.** Son ya bastantes las obras que es posible consultar sobre la utilización de R como herramienta para los cálculos que requiere la regresión lineal. Una excelente referencia es Venables and Ripley (1999a). Exclusivamente orientado a modelos lineales es Faraway (2005).