Introducción al software estadístico R

Dae-Jin Lee < dlee@bcamath.org >

Marzo 2019

Índice general

Capítulo 1

Información y pre-requisitos

- **Requisitos:**
  - Última versión del software `R` (https://www.r-project.org)
  - `Rstudio` (https://www.rstudio.com).

- **Horario:**
  - Día 11 de marzo 10:30 a 14:30 horas
  - Días 12 y 13 de marzo de 09.00 a 13.00 horas
  - Día 14 de marzo 09.00 a 12.00 horas

- **Objetivos del curso**

El curso proporcionará conocimientos básicos y habilidades para utilizar el software estadístico `R` como entorno de trabajo y análisis de datos. Para ello, el curso ofrecerá una visión general de las herramientas ofrecidas por `R`, por ejemplo, desde el manejo de matrices y conjuntos de datos, a representaciones gráficas, pruebas y análisis estadísticos, programación en `R` y técnicas de modelización estadística y aprendizaje automático.

El objetivo es que al final del curso todos se hayan familiarizado con el software estadístico `R`.

- **Material complementario (en inglés)**
  - Table of useful `R` commands
  - Base `R` cheatsheet
  - Not so short list of `R` commands
  - Additional material: `R` for Data Science or here
  - Some interesting links:
    - Karl Broman's talks
    - Plots to avoid
    - RSeek - a Google search engine for R documentation and help. A MUST see!

Capítulo 2

Introducción al software estadístico `R`

`R` es un entorno y lenguaje de programación orientado al análisis estadístico, al cálculo, manipulación de datos, y representaciones gráficas. Es multiplataforma y parte del sistema GNU y se distribuye con licencia GNU-GPL.

Más información aquí.

2.1. Algunas cuestiones a tener en cuenta sobre `R`

- `R` distingue mayúsculas y minúsculas.
- Para asignar contenido a un objeto usamos `<-`. Por ejemplo, `x <- 10` asigna a `x` el valor `10`. También podemos usar `=`.
- Para ver el contenido de un objeto simplemente escribimos su nombre.
- Para usar los comandos escribimos el nombre del comando seguido de sus argumentos entre paréntesis. Por ejemplo, `ls()`da una lista de los objetos en el área de trabajo. Como no usamos argumentos (diferentes a los que el comando tenga por defecto) no escribimos nada en el paréntesis.
- Para obtener ayuda usamos el comando `help`. Por ejemplo, `help(mean)` para obtener ayuda sobre el comando `mean` que calcula la media.

2.2. Rstudio

RStudio es un IDE (*Integrated Development Environment*, o Entorno de Desarrollo Integrado) de código abierto para `R`, que permite interactuar con `R` de manera muy simple.

Entre otras ventajas, Rstudio utiliza diferentes colores para las distintas clases de objetos de `R`, permite autocompletar código, incluye un sistema de menús de ayuda muy completo, cuenta con un potente sistema para la gestión, descarga y construcción de librerías, dispone de un depurador de código que detecta posibles errores de sintaxis, es multiplataforma (existen versiones para Windows, Linux y Mac).

A la izquierda, la consola donde se ejecutan los comandos de `R`.

A la derecha, en la parte superior, tenemos una ventana que muestra nuestro entorno (environment) de trabajo, en el que iremos viendo las variables y funciones que vayamos cargando, creando, etc. Obsérvese que esta ventana tiene algunos iconos que permiten guardar el contenido de la memoria, cargar el contenido de la memoria de una sesión de trabajo anterior, importar archivos de datos que se hayan guardado como texto, y limpiar el contenido de la memoria.

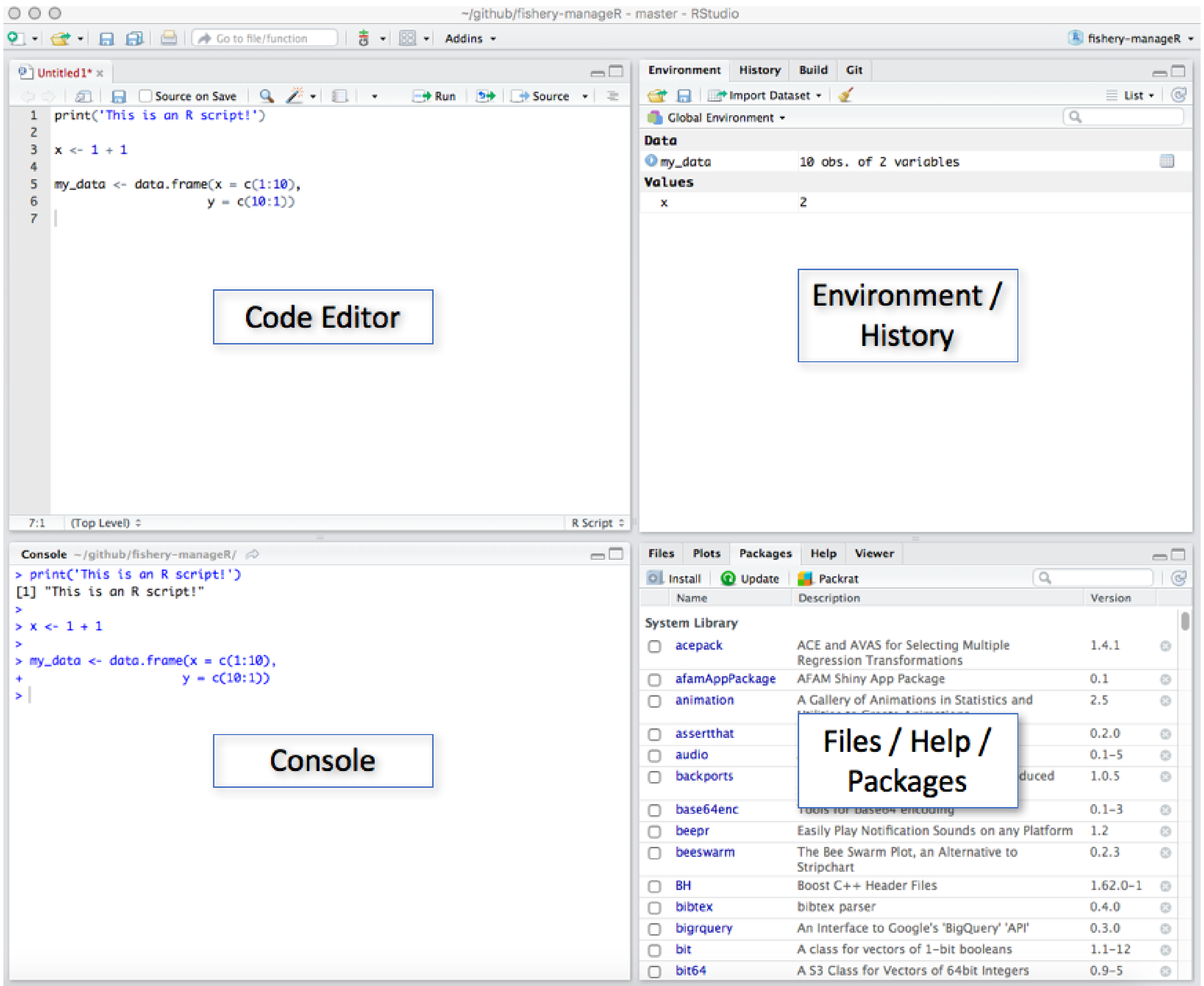


Figura 2.1: Interfaz RStudio

A la derecha, en la parte inferior, se muestra el contenido de nuestro directorio home donde R arranca por defecto. Observemos que esta ventana tiene varias pestañas:

- **Files:** archivos en el directorio actual.
- **Plots:** en esta ventana se irán mostrando los gráficos que generemos con el programa.
- **Packages:** permite ver qué librerías (colecciones de funciones que extienen la funcionalidad de R) tenemos instaladas; asimismo nos permite descargar e instalar nuevas librerías.
- **Help:** permite acceder a ayuda sobre ‘R.
- **Viewer:** Permite acceder a contenido web local.

2.3. Instalar un paquete de R

- Desde la consola

```
install.packages("< Nombre del paquete >")
```

- Ejemplo:

```
install.packages("DAAG")
# Varios paquetes con el comando c("< Paquete 1 >","< Paquete 2 >")
install.packages(c("DAAG","psych","gdata","foreign","Hmisc","xlsx",
                   "MASS","calibrate","corrplot","fields","RColorBrewer",
                   "ggplot2","lattice","visreg",
                   "car","InformationValue","ROCR"))
```

- Desde Rstudio: `Packages > Install`

2.4. Empezando con R

- Obtener el directorio de trabajo o *working directory*

```
getwd()
```

- listar los objetos en el espacio de trabajo o *workspace*

```
ls()
```

- Definir el *working directory*

```
setwd("/Users/dlee")
```

- Desde Rstudio: `Files > Click en ... > More > Set as workind directory`

- Ver los últimos comandos utilizados en la consola

```
history() # mostrar los últimos 25 comandos
history(max.show=Inf) # mostrar todos los comandos anteriores
```

- Guardar el historial de comandos

```
savehistory(file="myfile") # el valor por defecto es ".Rhistory"
```

- Cargar los commandos guardados en una sesión anterior

```
loadhistory(file="myfile") # el valor por defecto es ".Rhistory"
```

- Salvar todo el **workspace** en un fichero `.RData`

```
save.image()
```

- Guardar objectos especificos a un fichero (si no se especifica la ruta en el ordenador, se guardará en el directorio actual de trabajo).

```
save(<object list>,file="myfile.RData")
```

- Cargar un *workspace* en la sesión

```
load("myfile.RData")
```

- Salir de R. Por defecto R pregunta si deseas guardar la sesión.

```
q()
```

2.5. Cargar librerías en R

- Una vez instalada la librería, tenemos que cargarla con el comando `library` o `require`

```
library(DAAG) # o require(DAAG)
```

2.6. Lectura de datos

Consola de R

```
x <- c(7.82,8.00,7.95) # c de "combinar"
x
```

```
## [1] 7.82 8.00 7.95
```

Otra forma es mediante la función `scan()`

```
x <- scan()  # introducir números seguidos de ENTER y terminar con un ENTER
1: 7.82
2: 8.00
3: 7.95
4:
Read 3 items
```

Para crear un vector de caracteres

```
id <- c("John","Paul","George","Ringo")
```

To read a character vector

```
id <- scan(,"")
1: John
2: Paul
3: George
4: Ringo
5:
Read 4 items
```

```
id
```

```
## [1] "John"   "Paul"   "George" "Ringo"
```

2.7. Importar datos

En ocasiones, necesitaremos leer datos de un fichero independiente. Existen varias formas de hacerlo:

- `scan()` (`?scan` ver la ayuda)

```
# creamos el fichero ex.txt
cat("Example:", "2 3 5 7", "11 13 17", file = "ex.txt", sep = "\n")
scan("ex.txt", skip = 1)
```

```
## [1]  2  3  5  7 11 13 17
```

```
scan("ex.txt", skip = 1, nlines = 1) # only 1 line after the skipped one
```

```
## [1] 2 3 5 7
```

```
unlink("ex.data") # tidy up
```

- Existen diferentes formatos (`.txt`, `.csv`, `.xls`, `.xlsx`, `SAS`, `Stata`, etc…)
- Alguna librer?as de `R` para importar datos:

```
library(gdata)
library(foreign)
```

- Generalmente leeros datos en formato `.txt` o `.csv`

Vamos a crear una carpeta que llamaremos `data` y descargaremos los datos `cardata` en el siguiente enlace.

```
mydata1 <- read.table("data/cardata.txt")
mydata2 <- read.csv("data/cardata.csv")
```

- Otros formatos `.xls` and `.xlsx`

```
library(gdata)
mydata3 <- read.xls ("cardata/cardata.xls", sheet <- 1, header = TRUE)
```

- Minitab, SPSS, SAS or Stata

```
library(foreign)
mydata = read.mtp("mydata.mtp")  # Minitab
mydata = read.spss("myfile", to.data.frame=TRUE) # SPSS
mydata = read.dta("mydata.dta") # Stata
```

- O también

```
library(Hmisc)
mydata = spss.get("mydata.por", use.value.labels=TRUE)  # SPSS
```

2.8. Exportar datos

- Existen diferentes maneras de exportar datos desde `R` en diferentes formatos. Para SPSS, SAS y Stata. Por ejemplo, mediante la librería `foreign`. En Excel, la librería `xlsx`.
- Texto delimitado por tabulaciones:

```
mtcars
?mtcars
write.table(mtcars, "cardata.txt", sep="\t")
```

- Hoja de cálculo de Excel:

```
library(xlsx)
write.xlsx(mydata, "mydata.xlsx")
```

2.9. Vectores

- Crear dos vectores

```
weight<-c(60,72,57,90,95,72)
class(weight)
```

```
## [1] "numeric"
```

```
height<-c(1.75,1.80,1.65,1.90,1.74,1.91)
```

- calcular el Body Mass Index (*índice de masa corporal*)

```
bmi<- weight/height^2
bmi
```

```
## [1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

2.10. Estadística básica

- mean, median, st dev, variance

```
mean(weight)
median(weight)
sd(weight)
var(weight)
```

- Resumen de un vector

```
summary(weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   57.00   63.00   72.00   74.33   85.50   95.00
```

- o también

```
min(weight)
max(weight)
range(weight)
sum(weight)
length(weight)
```

- Cuantiles y percentiles

```
quantile(weight) # por defecto cuantil 25%, 50% y 75%
```

```
##   0%  25%  50%  75% 100%
## 57.0 63.0 72.0 85.5 95.0
```

```
quantile(weight,c(0.32,0.57,0.98))
```

```
##  32%  57%  98%
## 67.2 72.0 94.5
```

- Covarianza y correlación

La covarianza (σ_{xy}) indica el grado de variacion conjunta de dos variables aleatorias respecto a sus medias

- Si $\sigma_{xy} > 0$, hay dependencia directa (positiva), es decir, a grandes valores de x corresponden grandes valores de y.
- Si $\sigma_{xy} = 0$, hay una covarianza 0 se interpreta como la no existencia de una relación lineal entre las dos variables estudiadas.

- Si $\sigma_{xy} < 0$m hay dependencia inversa o negativa, es decir, a grandes valores de x corresponden pequeños valores de y.

$$\mathrm{Cov(x, y)} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{x_i} - \bar{\mathrm{x}})(\mathrm{y_i} - \bar{\mathrm{y}})$$

```
cov(weight,height)
```

```
## [1] 0.6773333
```

El *coeficiente de correlación* mide la relacion lineal (positiva o negativa) entre dos variables. Formalmente es el cociente entre la covarianza y el producto de las desviaciones típicas de ambas variables. Siendo σ_x y σ_y las desviaciones estandar y $\sigma_x y$ la covarianza entre x e y.

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \ \sigma_y}$$

```
cor(weight,height)
```

```
## [1] 0.437934
```

2.11. Vectores caracteres y variables factor

```
subject <- c("John","Peter","Chris","Tony","Mary","Jane")
sex <- c("MALE","MALE","MALE","MALE","FEMALE","FEMALE")
class(subject)
```

```
## [1] "character"
```

```
table(sex)
```

```
## sex
## FEMALE   MALE
##      2      4
```

2.12. Data frames

```
Dat <- data.frame(subject,sex,weight,height)
# añadir el bmi a Dat
Dat$bmi <- bmi  # o Dat$bmi <- weight/height^2
class(Dat)
```

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

```
str(Dat) # Ver la estructura del data.frame
```

```
## 'data.frame':    6 obs. of  5 variables:
##  $ subject: Factor w/ 6 levels "Chris","Jane",..: 3 5 1 6 4 2
##  $ sex    : Factor w/ 2 levels "FEMALE","MALE": 2 2 2 2 1 1
##  $ weight : num  60 72 57 90 95 72
##  $ height : num  1.75 1.8 1.65 1.9 1.74 1.91
##  $ bmi    : num  19.6 22.2 20.9 24.9 31.4 ...
```

```
# cambiar el nombre de las filas
rownames(Dat)<-c("A","B","C","D","E","F")

# Acceder a los elementos del data.frame
Dat[,1]      # columna 1
```

```
## [1] John  Peter Chris Tony  Mary  Jane
## Levels: Chris Jane John Mary Peter Tony
```

```
Dat[,1:3]    # columnas 1 a 3
```

```
##   subject    sex weight
## A    John   MALE     60
## B   Peter   MALE     72
## C   Chris   MALE     57
## D    Tony   MALE     90
## E    Mary FEMALE     95
## F    Jane FEMALE     72
```

```
Dat[1:2,]    # filas 1 a 2
```

```
##   subject  sex weight height      bmi
## A    John MALE     60   1.75 19.59184
## B   Peter MALE     72   1.80 22.22222
```

2.12.1. Trabajando con data frames

Ejemplo: analizar datos por grupos

- Obtener el peso (`weight`), altura (`height`) y `bmi` por `FEMALES` y `MALES`:

1. Seleccionado cada grupo y calculando la media por grupos

```
Dat[sex=="MALE",]
Dat[sex=="FEMALE",]

mean(Dat[sex=="MALE",3])  # weight average of MALEs
mean(Dat[sex=="MALE","weight"])
```

2. Mediante la función `apply` por columnas

```
apply(Dat[sex=="FEMALE",3:5],2,mean)
apply(Dat[sex=="MALE",3:5],2,mean)

# podemos utilizar la función apply con cualquier función
apply(Dat[sex=="FEMALE",3:5],2,function(x){x+2})
```

3. función `by` o `colMeans`

```
# 'by' divide los datos en factores y realiza
#   los cálculos para cada grupo
by(Dat[,3:5],sex, colMeans)
```

4. función `aggregate`

```
# otra opción
aggregate(Dat[,3:5], by=list(sex),mean)
```

2.13. Vectores lógicos

- Elegir los individuos con BMI>22

```
bmi
bmi>22
as.numeric(bmi>22) # convierte a numerico 0/1
which(bmi>22)  # nos devuelve la posicion del valor donde bmi>22
```

- ¿Qué valores están entre 20 y 25?

```
bmi > 20 & bmi < 25
which(bmi > 20 & bmi < 25)
```

2.14. Trabajando con vectores

- Concatenar

```
x <- c(2, 3, 5, 2, 7, 1)
y <- c(10, 15, 12)
z <- c(x,y)  # concatena x e y
```

- Lista de 2 vectores

```
zz <- list(x,y) # crea una lista
unlist(zz) # deshace la lista convirtiéndola en un vector concatenado
```

```
## [1]  2  3  5  2  7  1 10 15 12
```

- Subconjunto de vectores

```
x[c(1,3,4)]
```

```
## [1] 2 5 2
```

```
x[-c(2,6)] # simbolo - omite los elementos
```

```
## [1] 2 5 2 7
```

- Secuencias

```
seq(1,9) # ó 1:9
```

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

```
seq(1,9,by=1)
```

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

```
seq(1,9,by=0.5)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0
```

```
seq(1,9,length=20)
```

```
##  [1] 1.000000 1.421053 1.842105 2.263158 2.684211 3.105263 3.526316
##  [8] 3.947368 4.368421 4.789474 5.210526 5.631579 6.052632 6.473684
## [15] 6.894737 7.315789 7.736842 8.157895 8.578947 9.000000
```

- Réplicas

```
oops <- c(7,9,13)
rep(oops,3)   # repite el vector "oops" 3 veces
```

```
rep(oops,1:3) # repite cada elemento del vector las veces indicadas

rep(c(2,3,5), 4)
rep(1:2,c(10,15))

rep(c("MALE","FEMALE"),c(4,2)) # también funciona con caracteres
c(rep("MALE",3), rep("FEMALE",2))
```

2.15. Matrices y arrays

```
x<- 1:12
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
dim(x)<-c(3,4)  # 3 filas y 4 columnas

X <- matrix(1:12,nrow=3,byrow=TRUE)
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

```
X <- matrix(1:12,nrow=3,byrow=FALSE)
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# rownames, colnames

rownames(X) <- c("A","B","C")
X
```

```
##   [,1] [,2] [,3] [,4]
## A    1    4    7   10
## B    2    5    8   11
## C    3    6    9   12
```

```
colnames(X) <- LETTERS[4:7]
X
```

```
##   D E F  G
## A 1 4 7 10
## B 2 5 8 11
## C 3 6 9 12
```

```
colnames(X) <- month.abb[4:7]
X
```

```
##   Apr May Jun Jul
## A   1   4   7  10
## B   2   5   8  11
```

```
## C   3   6   9  12
```

- Concatenar filas y columnas `rbind()`, `cbind()`

```r
Y <- matrix(0.1*(1:12),3,4)

cbind(X,Y)  # bind column-wise
```

```
##   Apr May Jun Jul
## A   1   4   7  10 0.1 0.4 0.7 1.0
## B   2   5   8  11 0.2 0.5 0.8 1.1
## C   3   6   9  12 0.3 0.6 0.9 1.2
```

```r
rbind(X,Y)  # bind row-wise
```

```
##   Apr May Jun  Jul
## A 1.0 4.0 7.0 10.0
## B 2.0 5.0 8.0 11.0
## C 3.0 6.0 9.0 12.0
##   0.1 0.4 0.7  1.0
##   0.2 0.5 0.8  1.1
##   0.3 0.6 0.9  1.2
```

2.16. Factores

```r
gender<-c(rep("female",691),rep("male",692))
class(gender)
```

```
## [1] "character"
```

```r
# cambiar vector a factor (por ejemplo a una categoria)
gender<- factor(gender)
levels(gender)
```

```
## [1] "female" "male"
```

```r
summary(gender)
```

```
## female   male
##    691    692
```

```r
table(gender)
```

```
## gender
## female   male
##    691    692
```

```r
status<- c(0,3,2,1,4,5)     # Crear vector numerico,
                            #    transformarlo a niveles.
fstatus <- factor(status, levels=0:5)
levels(fstatus) <- c("student","engineer",
                     "unemployed","lawyer","economist","dentist")

Dat$status <- fstatus
Dat
```

```
##   subject    sex weight height      bmi     status
## A    John   MALE     60   1.75 19.59184    student
```

```
## B   Peter   MALE     72   1.80 22.22222     lawyer
## C   Chris   MALE     57   1.65 20.93664 unemployed
## D    Tony   MALE     90   1.90 24.93075   engineer
## E    Mary FEMALE     95   1.74 31.37799  economist
## F    Jane FEMALE     72   1.91 19.73630    dentist
```

2.17. Indexando vectores con condiciones lógicas

```
a <- c(1,2,3,4,5)
b <- c(TRUE,FALSE,FALSE,TRUE,FALSE)

max(a[b])
```

```
## [1] 4
```

```
sum(a[b])
```

```
## [1] 5
```

2.18. Valores faltantes

En R, los valores faltante (o *missing values*) se representan como `NA` (*not available*). Los valores imposibles (e.g., valores dividos por cero) se representan con el simbolo `NaN` (*not a number*).

```
a <- c(1,2,3,4,NA)
sum(a)
```

```
## [1] NA
```

El argumento `na.rm=TRUE` excluye los valores `NA` en el cálculo de algunos valores

```
sum(a,na.rm=TRUE)
```

```
## [1] 10
```

```
a <- c(1,2,3,4,NA)
is.na(a) # YES or NO
```

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

La función `complete.cases()` devuelve un vector lógico que indica los casos completos.

```
complete.cases(a)
```

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

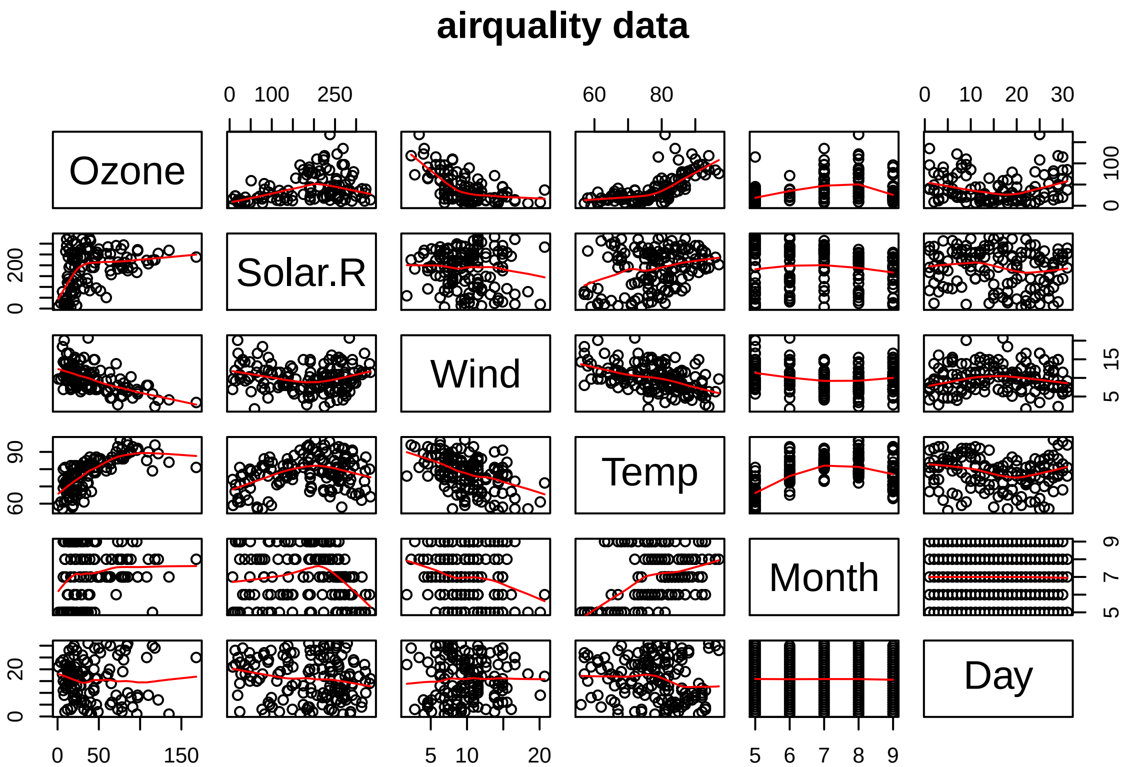
La función `na.omit()` devuelve un objeto sin los elementos `NA`.

```
na.omit(a)
```

```
## [1] 1 2 3 4
## attr(,"na.action")
## [1] 5
## attr(,"class")
## [1] "omit"
```

`NA` en data frames:

```r
require(graphics)
?airquality
pairs(airquality, panel = panel.smooth, main = "airquality data")
```




```r
ok <- complete.cases(airquality)
airquality[ok,]
```

2.19. Trabajando con data frames

- Los data frame se utilizand para guardas tablas de datos. Contiene elementos de la misma longitud.

```r
mtcars
?mtcars        # help(mtcars)
```

- Observemos las primeras filas

```r
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

- Estructura de un data frame

```r
str(mtcars) # visualiza la estructura del marco de datos
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
```

```
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

- Select a car model:

```
mtcars["Mazda RX4",] # usando nombres de las filas y las columnas
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
```

```
mtcars[c("Datsun 710", "Camaro Z28"),]
```

```
##             mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Datsun 710 22.8   4  108  93 3.85 2.32 18.61  1  1    4    1
## Camaro Z28 13.3   8  350 245 3.73 3.84 15.41  0  0    3    4
```

- O variables concretas

```
mtcars[,c("mpg","am")]
```

```
##                      mpg am
## Mazda RX4           21.0  1
## Mazda RX4 Wag       21.0  1
## Datsun 710          22.8  1
## Hornet 4 Drive      21.4  0
## Hornet Sportabout   18.7  0
## Valiant             18.1  0
## Duster 360          14.3  0
## Merc 240D           24.4  0
## Merc 230            22.8  0
## Merc 280            19.2  0
## Merc 280C           17.8  0
## Merc 450SE          16.4  0
## Merc 450SL          17.3  0
## Merc 450SLC         15.2  0
## Cadillac Fleetwood  10.4  0
## Lincoln Continental 10.4  0
## Chrysler Imperial   14.7  0
## Fiat 128            32.4  1
## Honda Civic         30.4  1
## Toyota Corolla      33.9  1
## Toyota Corona       21.5  0
## Dodge Challenger    15.5  0
## AMC Javelin         15.2  0
## Camaro Z28          13.3  0
## Pontiac Firebird    19.2  0
## Fiat X1-9           27.3  1
## Porsche 914-2       26.0  1
## Lotus Europa        30.4  1
```

```
## Ford Pantera L      15.8  1
## Ferrari Dino        19.7  1
## Maserati Bora       15.0  1
## Volvo 142E          21.4  1
```

```
library(psych)
describe(mtcars)
```

```
##      vars  n   mean     sd median trimmed    mad   min    max  range  skew
## mpg     1 32  20.09   6.03  19.20   19.70   5.41 10.40  33.90  23.50  0.61
## cyl     2 32   6.19   1.79   6.00    6.23   2.97  4.00   8.00   4.00 -0.17
## disp    3 32 230.72 123.94 196.30  222.52 140.48 71.10 472.00 400.90  0.38
## hp      4 32 146.69  68.56 123.00  141.19  77.10 52.00 335.00 283.00  0.73
## drat    5 32   3.60   0.53   3.70    3.58   0.70  2.76   4.93   2.17  0.27
## wt      6 32   3.22   0.98   3.33    3.15   0.77  1.51   5.42   3.91  0.42
## qsec    7 32  17.85   1.79  17.71   17.83   1.42 14.50  22.90   8.40  0.37
## vs      8 32   0.44   0.50   0.00    0.42   0.00  0.00   1.00   1.00  0.24
## am      9 32   0.41   0.50   0.00    0.38   0.00  0.00   1.00   1.00  0.36
## gear   10 32   3.69   0.74   4.00    3.62   1.48  3.00   5.00   2.00  0.53
## carb   11 32   2.81   1.62   2.00    2.65   1.48  1.00   8.00   7.00  1.05
##      kurtosis    se
## mpg     -0.37  1.07
## cyl     -1.76  0.32
## disp    -1.21 21.91
## hp      -0.14 12.12
## drat    -0.71  0.09
## wt      -0.02  0.17
## qsec     0.34  0.32
## vs      -2.00  0.09
## am      -1.92  0.09
## gear    -1.07  0.13
## carb     1.26  0.29
```

Capítulo 3

Análisis de datos básico en R

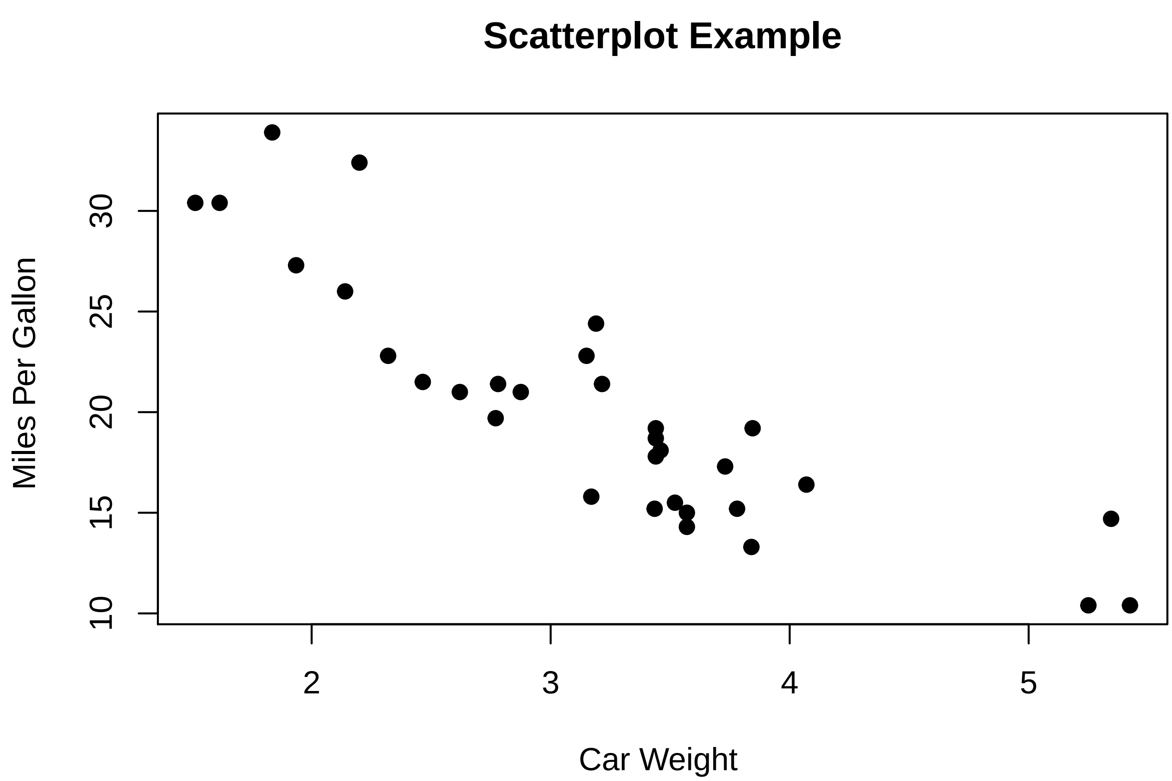
3.1. Gráficos sencillos

- Scatterplot

```
attach(mtcars)
```

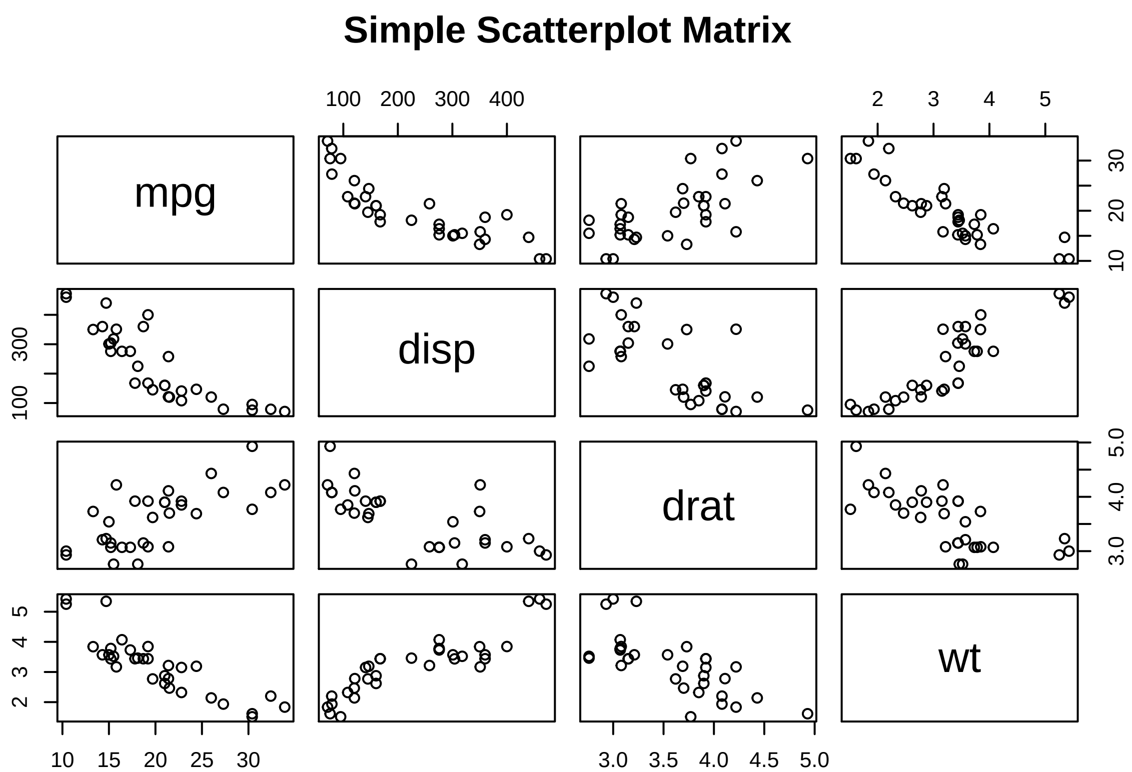
```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```
plot(wt, mpg, main="Scatterplot Example",
   xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
```



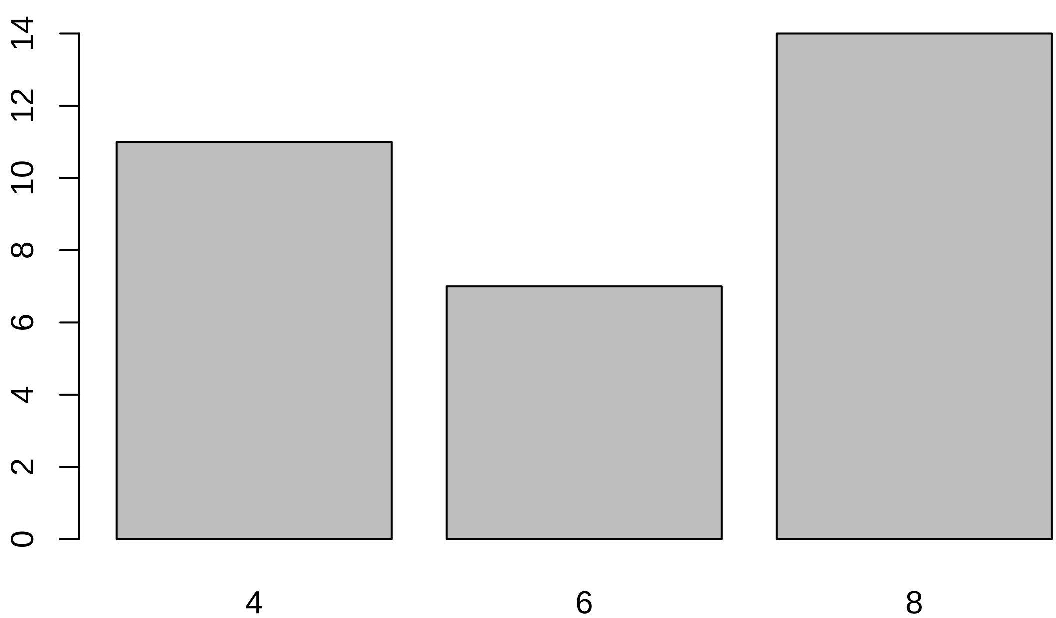
- Matriz scatterplot

```
pairs(~mpg+disp+drat+wt,data=mtcars,
   main="Simple Scatterplot Matrix")
```



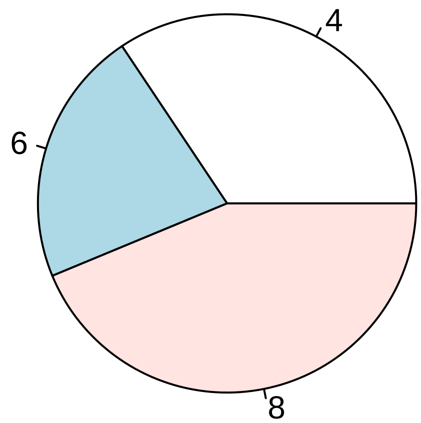
- Barplot o diagrama de barras

```
tab <- table(mtcars[,c("cyl")])
barplot(tab)
```



- Piechart o diagrama de tarta

```r
pie(tab)
```



Ejercicio:

1. El `data.frame` `VADeaths` contiene las tasas de mortalidad por cada 1000 habitantes en Virginia (EEUU) en 1940

- Las tasas de mortalidad se miden cada 1000 habitantes por año. Se encuentran clasificadas por grupo de edad (filas) y grupo de población (columnas). Los grupos de edad son: 50-54, 55-59, 60-64, 65-69, 70-74 y los grupos de población: `Rural/Male`, `Rural/Female`, `Urban/Male` and `Urban/Female`.

```r
data(VADeaths)
VADeaths
```

```
##       Rural Male Rural Female Urban Male Urban Female
## 50-54       11.7          8.7       15.4          8.4
## 55-59       18.1         11.7       24.3         13.6
## 60-64       26.9         20.3       37.0         19.3
## 65-69       41.0         30.9       54.6         35.1
## 70-74       66.0         54.3       71.1         50.0
```

- Calcula la media para cada grupo de edad.

  - **Result:**

```
##  50-54  55-59  60-64  65-69  70-74
## 11.050 16.925 25.875 40.400 60.350
```

- Calcula la media para cada grupo de población.

  - **Resultado:**

```
##   Rural Male Rural Female   Urban Male Urban Female
##        32.74        25.18        40.48        25.28
```

2. El `data.frame` `rainforest` contiene diferentes variables de `species`

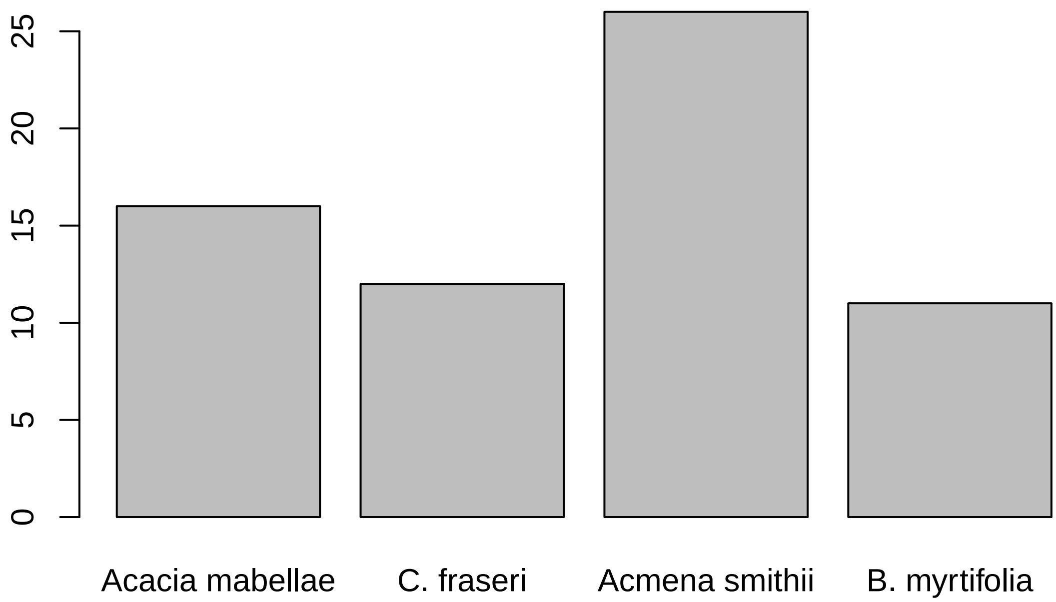
```r
library(DAAG)
rainforest
?rainforest
names(rainforest)
```

- Crear una tabla de conteos para cada `species` y realiza un gráfico descriptivo.

  - **Resultado:**

```
##
```

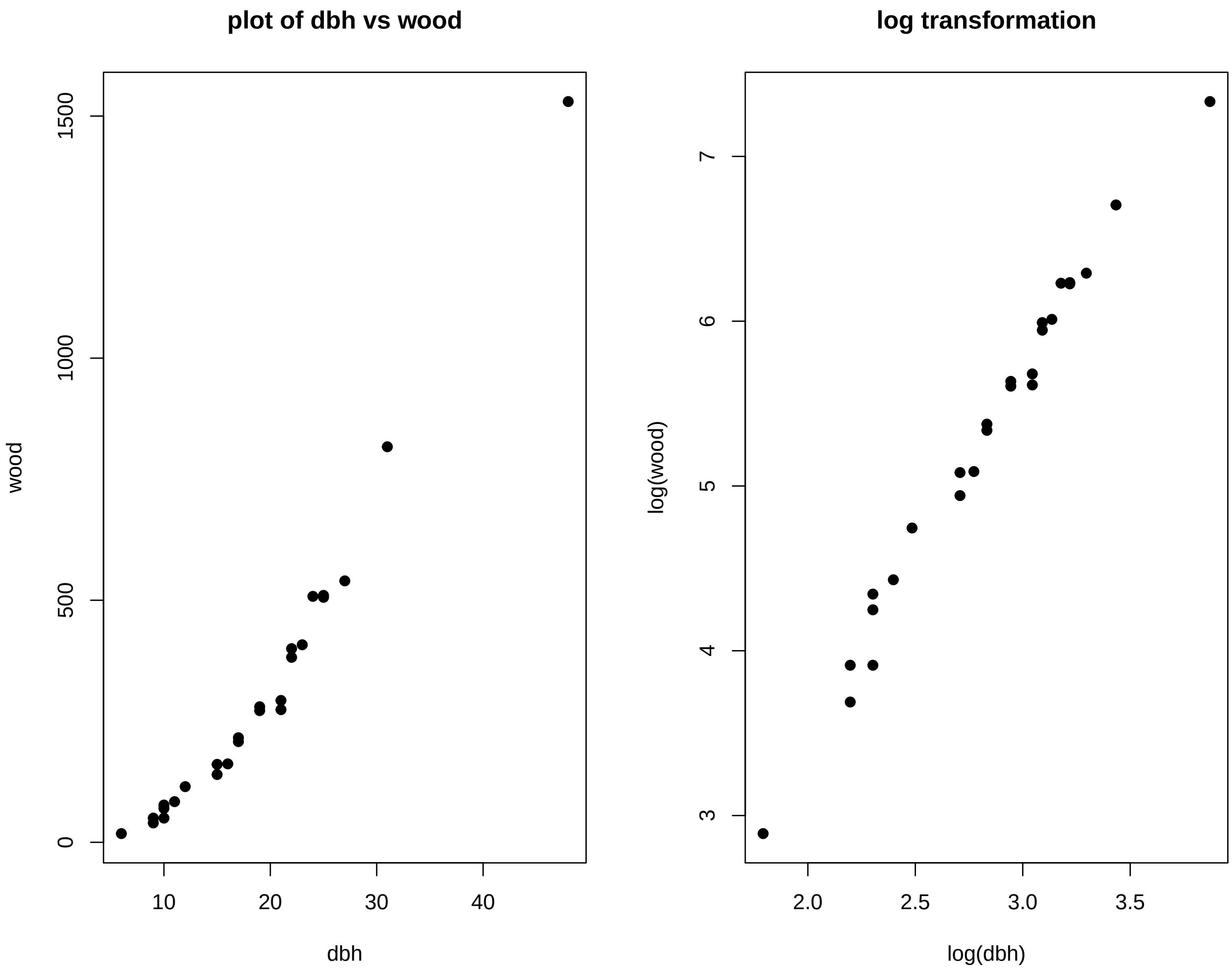
```
## Acacia mabellae       C. fraseri   Acmena smithii    B. myrtifolia
##               16               12               26               11
```



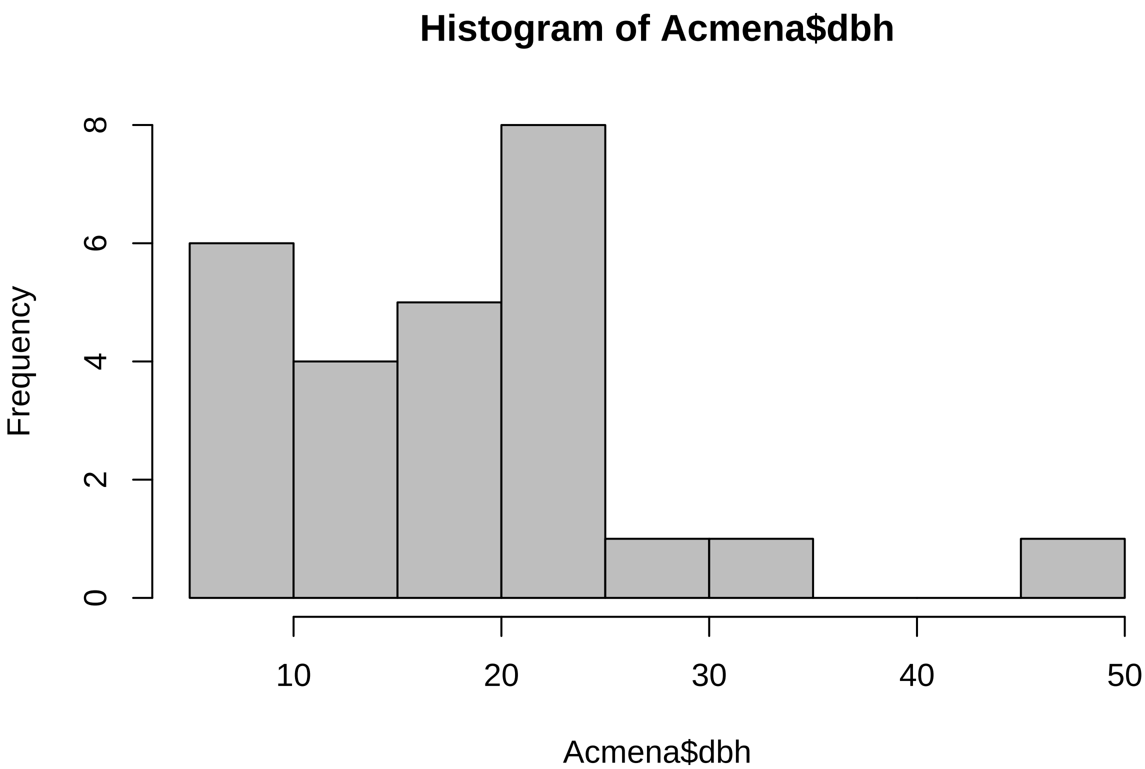
3. El `data.frame` `Acmena` est? creado a partir de `rainforest` mediante la función `subset`.

- Realiza un gráfico que relacione la biomasa de la madera (`wood`) y el di?metro a la altura del pecho (`dbh`). Utiliza tambi?n la escala logarítmica.

```
Acmena <- subset(rainforest, species == "Acmena smithii")
```



- Calcula un histograma de la variable `dbh` mediante la función `hist`



4. Crea un vector de n?meros enteros positivos impares the longitud 100 y calcula los valores entre 60 y 80.

    - **Result:**

```
##  [1] 61 63 65 67 69 71 73 75 77 79
```

- Soluciones aquí

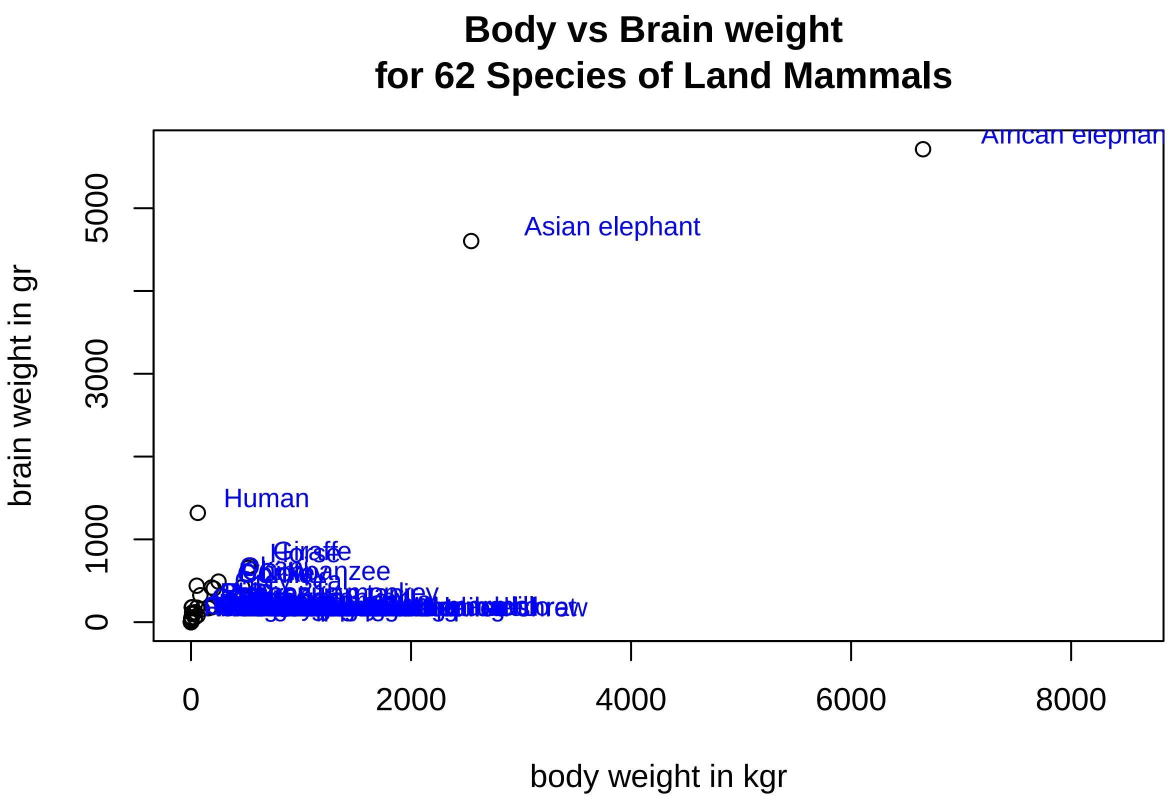
3.2. Scatterplots

```
library(MASS)
data("mammals")
?mammals
head(mammals)
```

```
##                     body brain
## Arctic fox         3.385  44.5
## Owl monkey         0.480  15.5
## Mountain beaver    1.350   8.1
## Cow              465.000 423.0
## Grey wolf         36.330 119.5
## Goat              27.660 115.0
```

```
attach(mammals)
species <- row.names(mammals)
x <- body
y <- brain
```

```
library(calibrate)
# scatterplot
plot(x,y, xlab = "body weight in kgr", ylab = "brain weight in gr",
     main="Body vs Brain weight \n for 62 Species of Land Mammals",xlim=c(0,8500))
textxy(x,y,labs=species,col = "blue",cex=0.85)
```

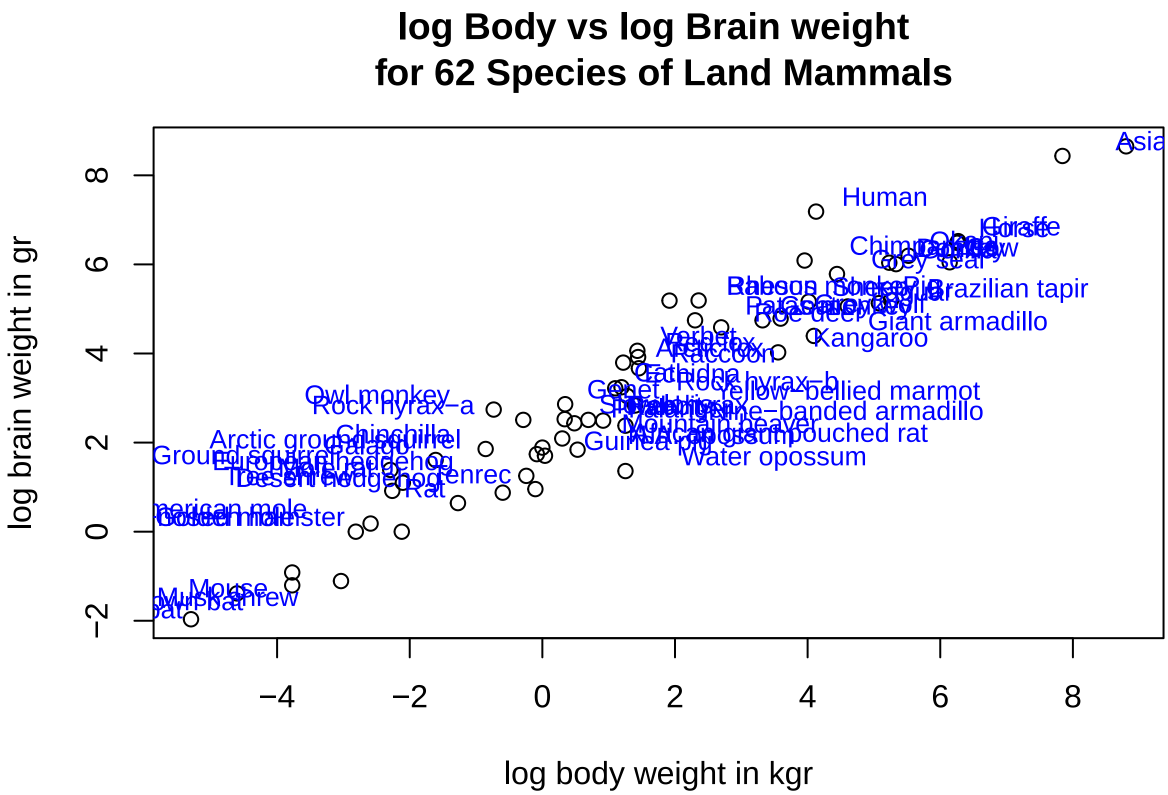


Identificar un punto en el scatterplot

```
identify(x,y,species)
```

En escala logarítmica

```
plot(log(x),log(y), xlab = "log body weight in kgr", ylab = "log brain weight in gr",
     main="log Body vs log Brain weight \n for 62 Species of Land Mammals")
textxy(log(x),log(y),labs=species,col = "blue",cex=0.85)
```



Identificar un punto en la escala logarítmica

```
identify(log(x),log(y),species)
```

3.3. Más opciones gráficas

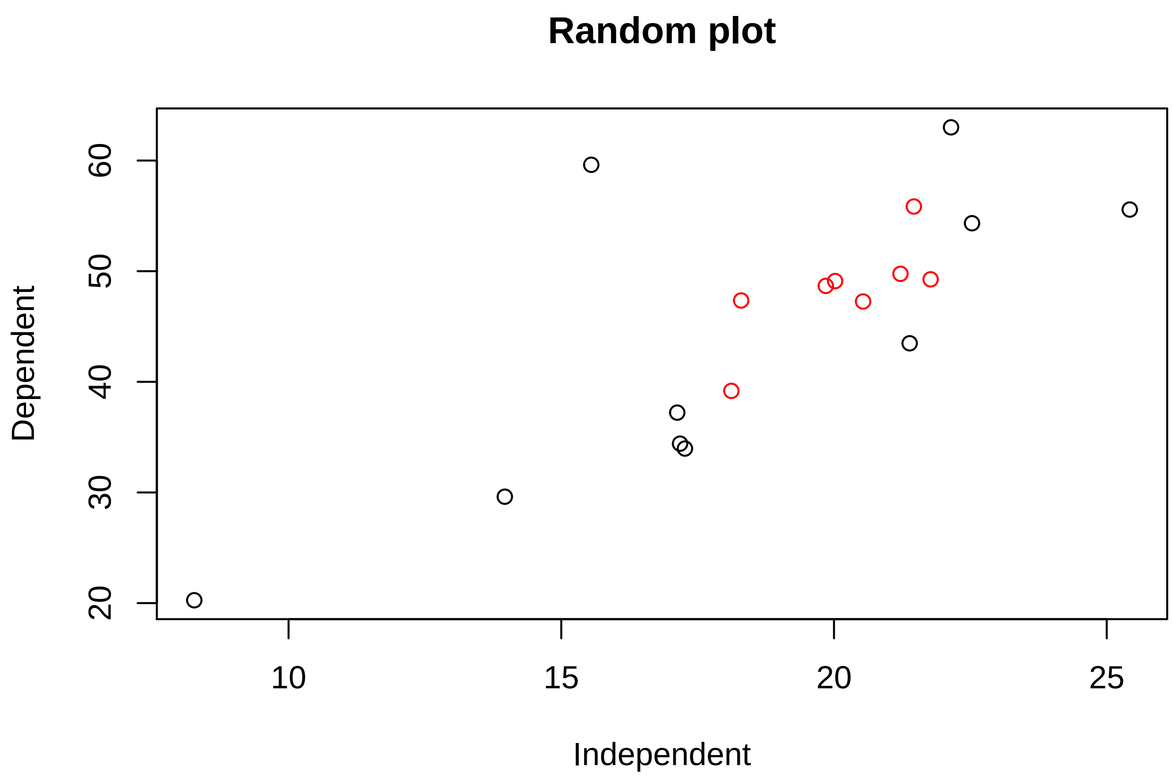
Varios conjuntos de datos en un sólo gráfico

Una vez realizado un `plot`, el comando `points` permite aadir nuevas observaciones.

```
set.seed(1234)
 x <- rnorm(10,sd=5,mean=20)
 y <- 2.5*x - 1.0 + rnorm(10,sd=9,mean=0)
 cor(x,y)
```

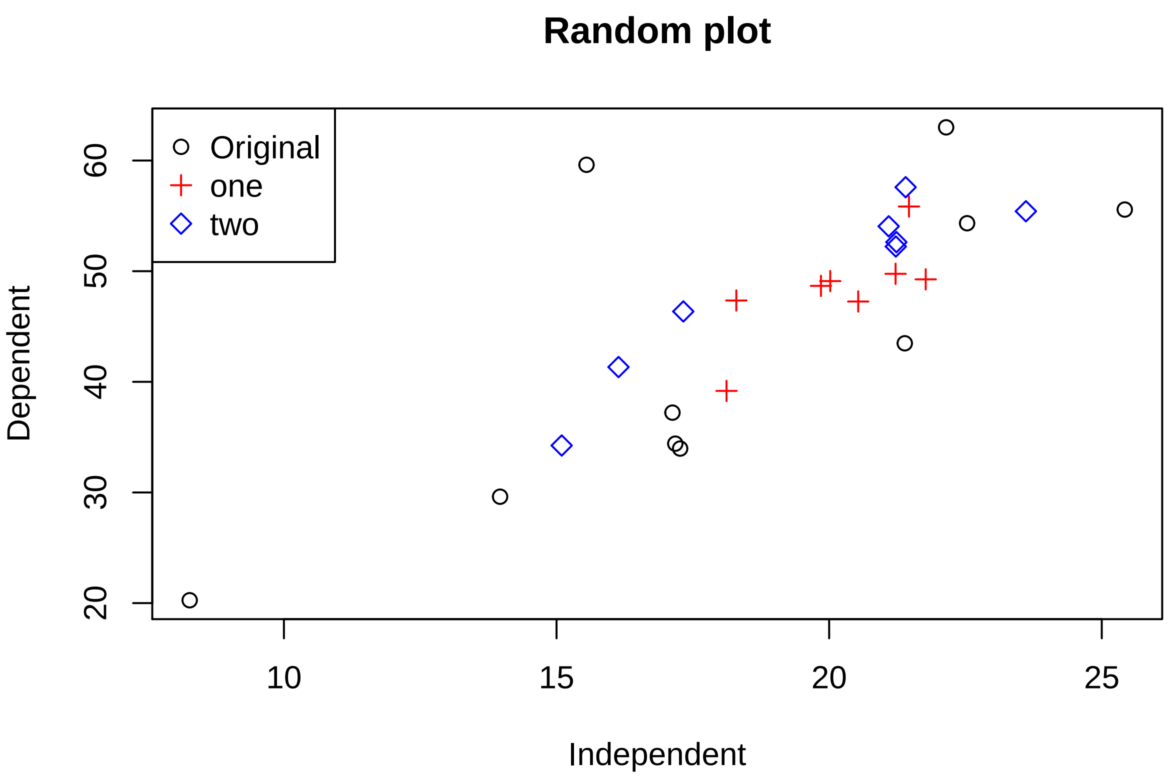
```
## [1] 0.7512194
```

```
 plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot")
 x1 <- runif(8,15,25)
 y1 <- 2.5*x1 - 1.0 + runif(8,-6,6)
 points(x1,y1,col=2)
```



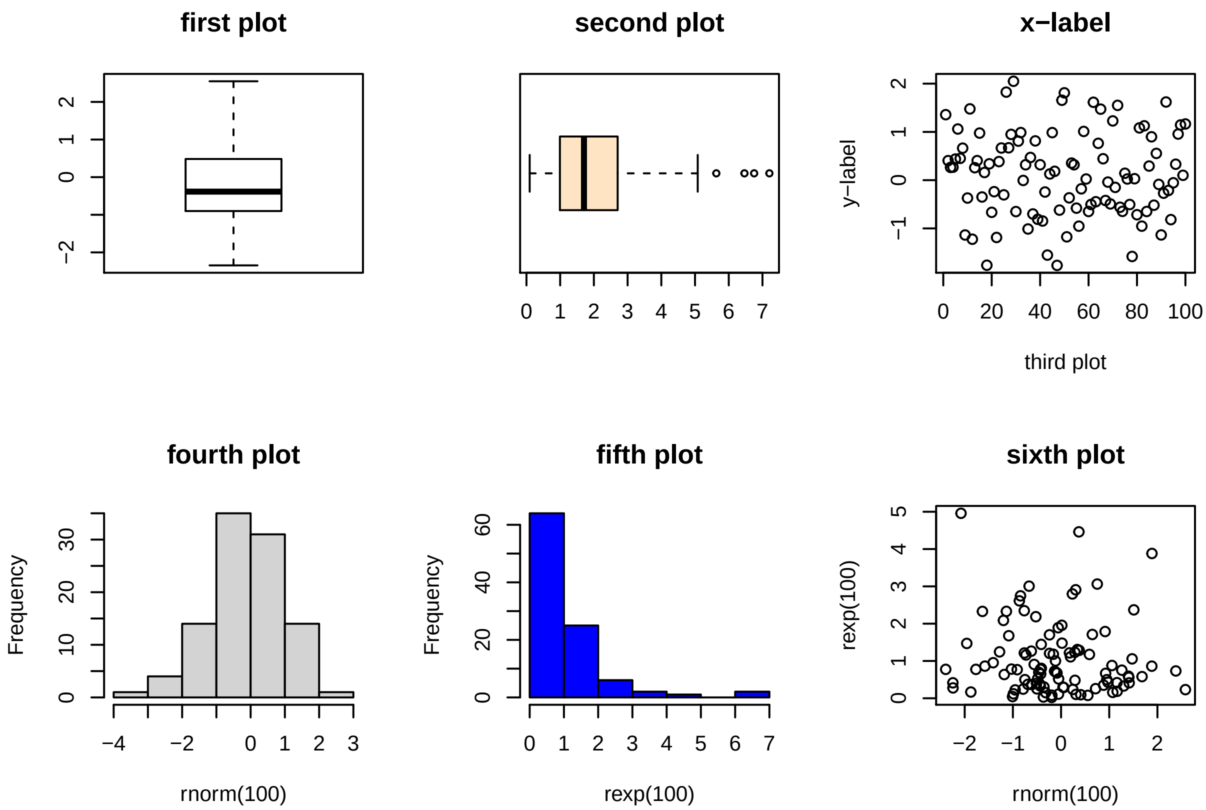
con la leyenda

```
set.seed(1234)
x2 <- runif(8,15,25)
y2 <- 2.5*x2 - 1.0 + runif(8,-6,6)
 plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot")
 points(x1,y1,col=2,pch=3)
 points(x2,y2,col=4,pch=5)
 legend("topleft",c("Original","one","two"),col=c(1,2,4),pch=c(1,3,5))
```



Varios gráficos en un sola imagen

```
set.seed(1234)
 par(mfrow=c(2,3))
 boxplot(rnorm(100),main="first plot")
 boxplot(rgamma(100,2),main="second plot", horizontal=TRUE,col="bisque")
 plot(rnorm(100),xlab="third plot",
      ylab="y-label",main="x-label")
 hist(rnorm(100),main="fourth plot",col="lightgrey")
 hist(rexp(100),main="fifth plot",col="blue")
 plot(rnorm(100),rexp(100),main="sixth plot")
```



Relaciones entre variables

```
uData <- rnorm(20)
vData <- rnorm(20,mean=5)
wData <- uData + 2*vData + rnorm(20,sd=0.5)
xData <- -2*uData+rnorm(20,sd=0.1)
yData <-  3*vData+rnorm(20,sd=2.5)
d <- data.frame(u=uData,v=vData,w=wData,x=xData,y=yData)
pairs(d)
```

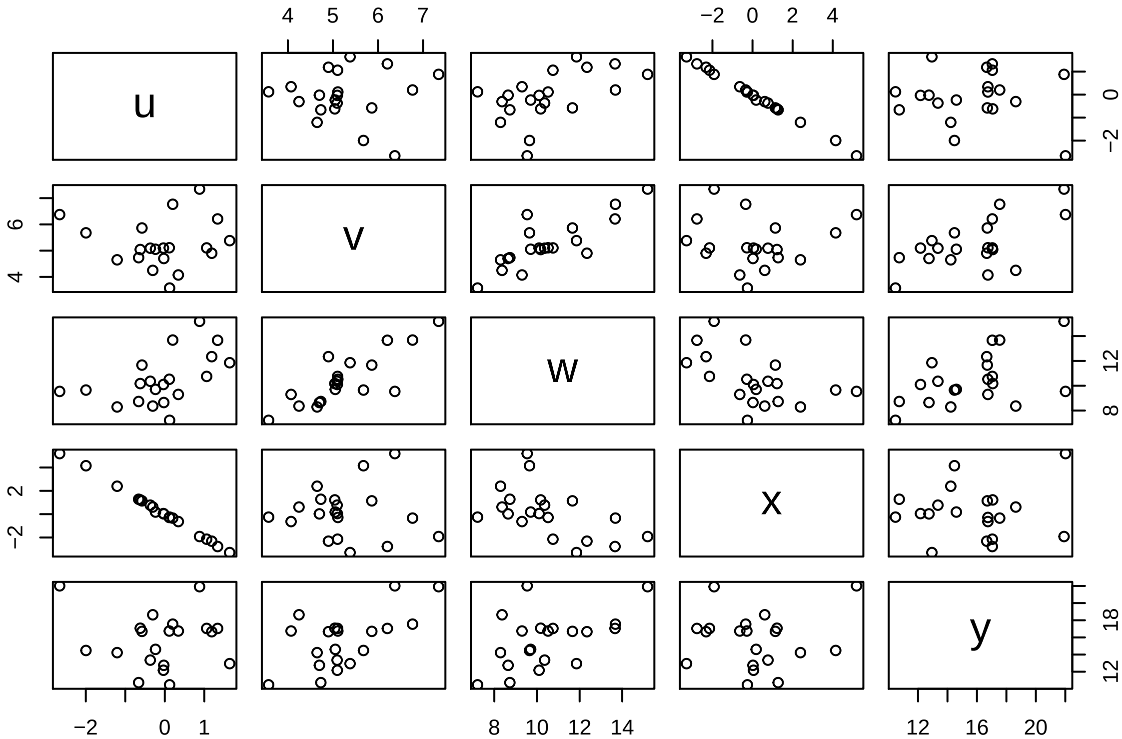


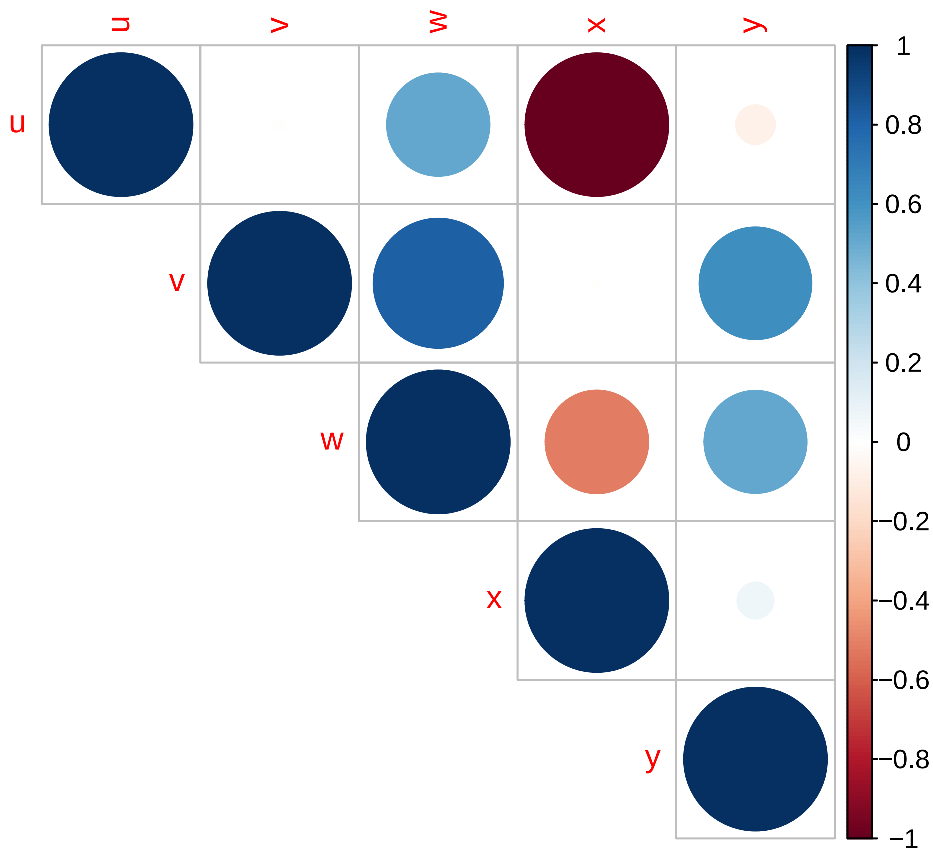
Gráfico de correlaciones

La función `corrplot` de la librer?a `corrplot` permite visualizar una matriz de correlaciones calculada mediante la función `cor`

```
library(corrplot)
```

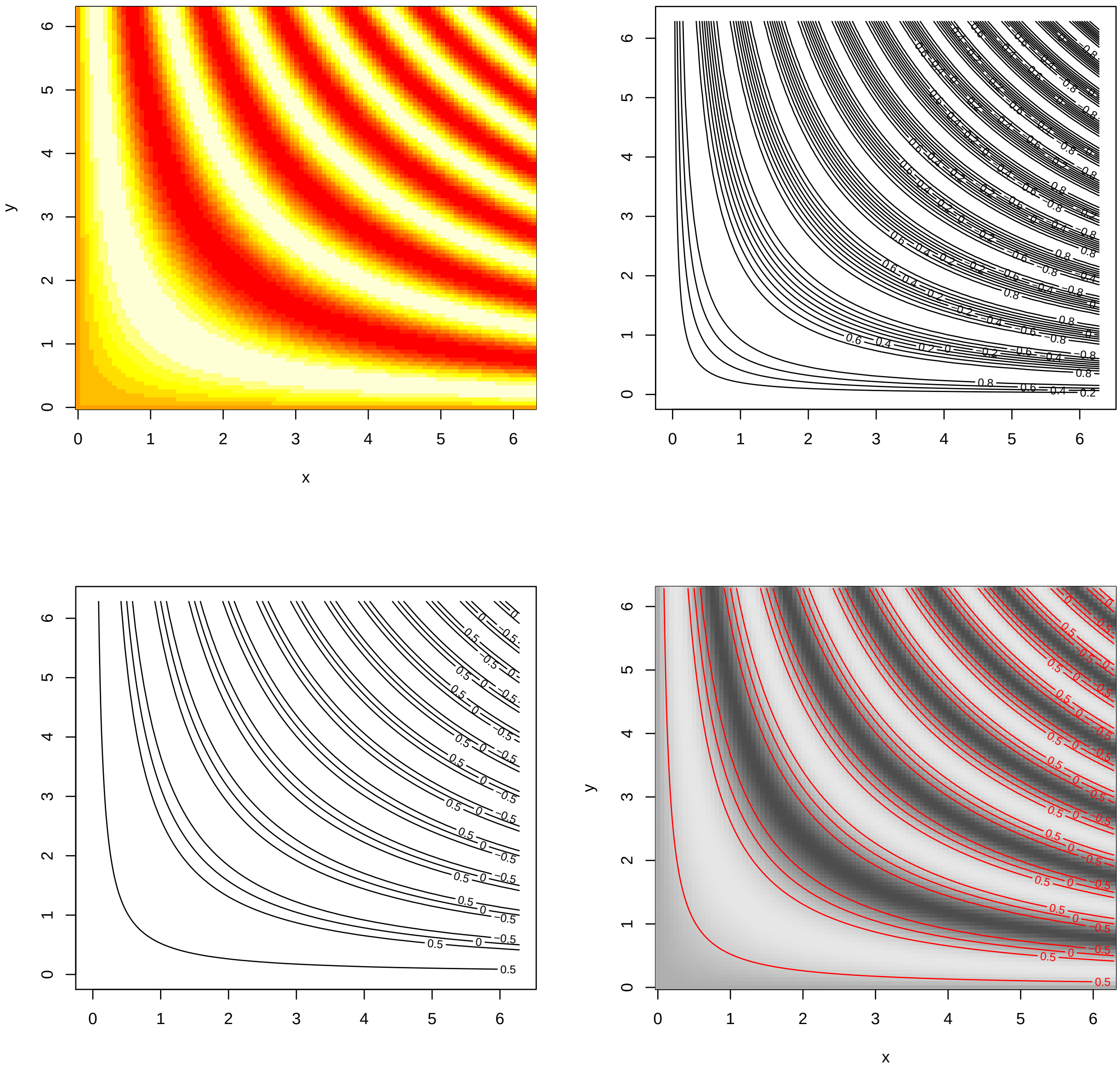
```
## corrplot 0.84 loaded
```

```
M <- cor(d)
corrplot(M, method="circle",type="upper")
```



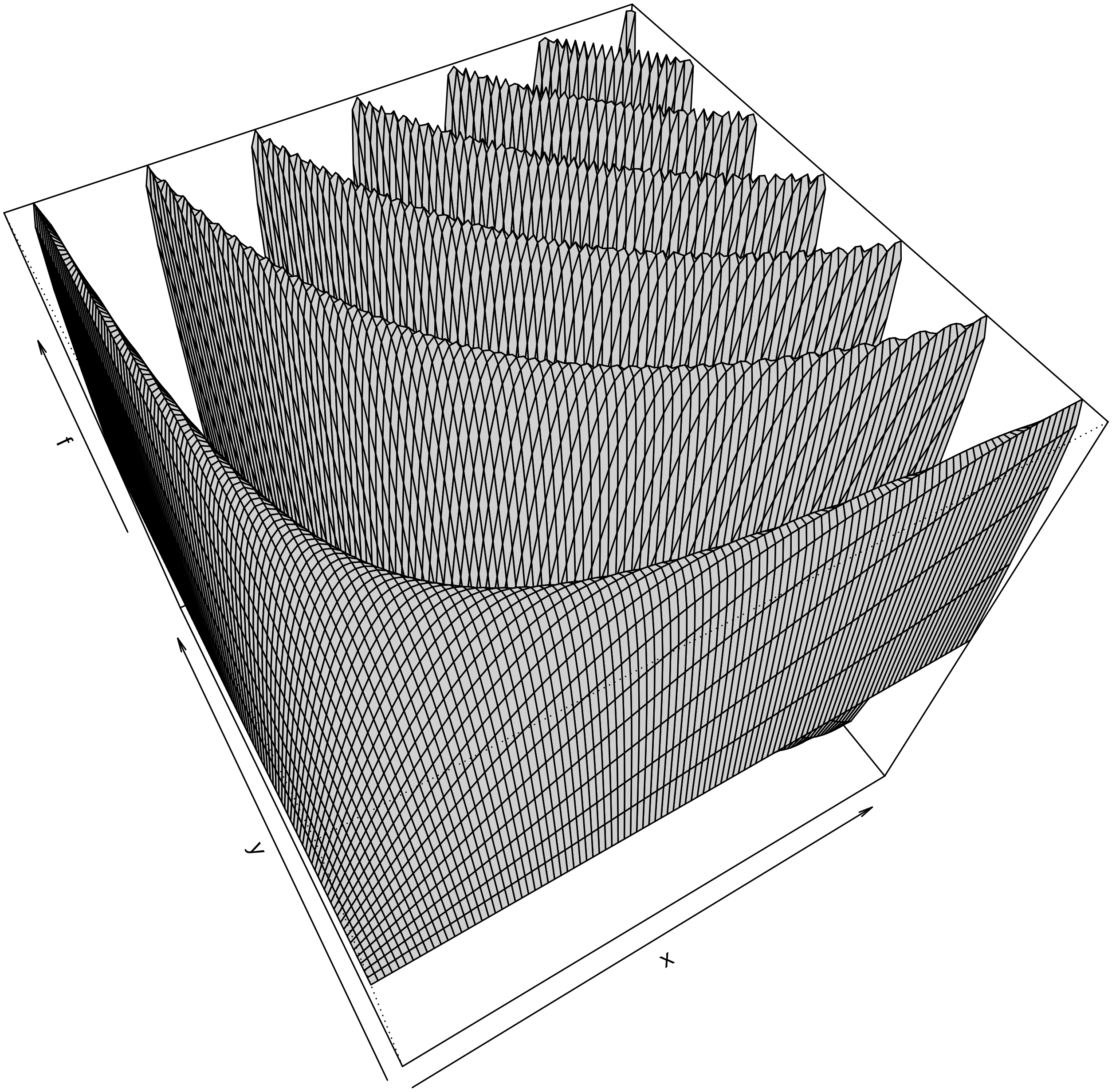
Gráficos de superficies: `image`, `contour` y `persp`

```
x <- seq(0,2*pi,by=pi/50)
y <- x
xg <- (x*0+1) %*% t(y)
yg <- (x) %*% t(y*0+1)
f <- sin(xg*yg)

par(mfrow=c(2,2))
image(x,y,f)
contour(x,y,f)
contour(x,y,f,nlevels=4)
image(x,y,f,col=grey.colors(100))
contour(x,y,f,nlevels=4,add=TRUE,col="red")
```



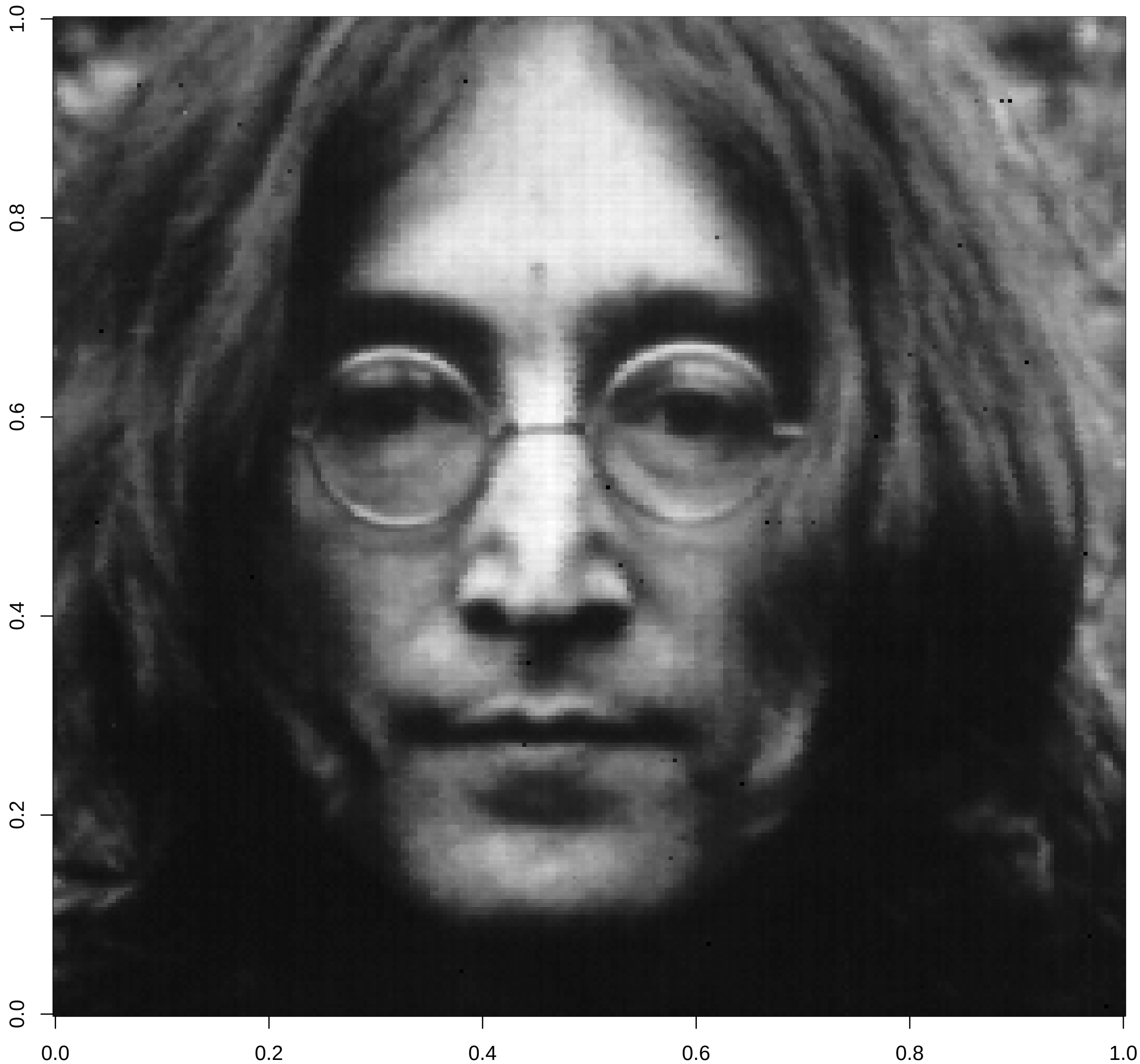
Podemos utilizar la función `persp`

```
persp(x,y,f,theta=-30,phi=55,col="lightgrey",shade=.01)
```



o representar im?genes

```
library(fields)
data(lennon)
image(lennon,col=grey(seq(0,1,l=256)))
```



3.4. Tablas de clasificación cruzada o de contigencia

```
library(MASS)
data(quine)
head(quine)
```

```
##   Eth Sex Age Lrn Days
## 1   A   M  F0  SL    2
## 2   A   M  F0  SL   11
## 3   A   M  F0  SL   14
## 4   A   M  F0  AL    5
## 5   A   M  F0  AL    5
## 6   A   M  F0  AL   13
```

```
attach(quine)
table(Sex)
```

```
## Sex
##  F  M
## 80 66
```

```r
table(Sex,Age)
```

```
##    Age
## Sex F0 F1 F2 F3
##   F 10 32 19 19
##   M 17 14 21 14
```

```r
# or xtabs
xtabs(~Sex+Age,data=quine)
```

```
##    Age
## Sex F0 F1 F2 F3
##   F 10 32 19 19
##   M 17 14 21 14
```

```r
xtabs(~Sex+Age+Eth,data=quine)
```

```
## , , Eth = A
##
##    Age
## Sex F0 F1 F2 F3
##   F  5 15  9  9
##   M  8  5 11  7
##
## , , Eth = N
##
##    Age
## Sex F0 F1 F2 F3
##   F  5 17 10 10
##   M  9  9 10  7
```

Cálculos sobre tablas de contigencia

```r
tapply(Days,Age,mean)
```

```
##       F0       F1       F2       F3
## 14.85185 11.15217 21.05000 19.60606
```

```r
tapply(Days,list(Sex,Age),mean)
```

```
##         F0       F1       F2       F3
## F 18.70000 12.96875 18.42105 14.00000
## M 12.58824  7.00000 23.42857 27.21429
```

```r
tapply(Days,list(Sex,Age),function(x) sqrt(var(x)/length(x)))
```

```
##         F0       F1       F2       F3
## F 4.208589 2.329892 5.299959 2.940939
## M 3.768151 1.418093 3.766122 4.569582
```

3.5. Datos cualitativos

Supongamos unos datos cualquiera de las variables `treatment` y `improvement` de pacientes a una enfermedad determinada.

```
treatment <- factor(rep(c(1, 2), c(43, 41)), levels = c(1, 2),
                    labels = c("placebo", "treated"))
improved <- factor(rep(c(1, 2, 3, 1, 2, 3), c(29, 7, 7, 13, 7, 21)),
                   levels = c(1, 2, 3),
                   labels = c("none", "some", "marked"))
```

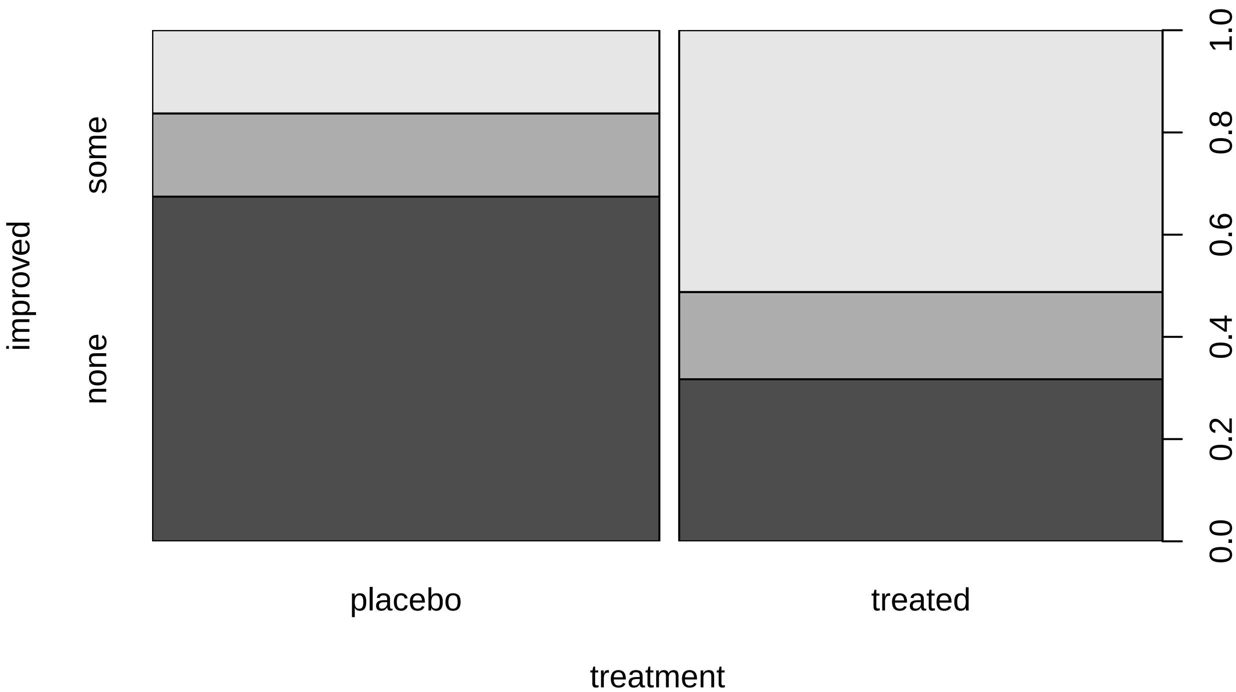
Tabla de contigencia

```
xtabs(~treatment+improved)
```

```
##          improved
## treatment none some marked
##   placebo   29    7      7
##   treated   13    7     21
```

De manera gráfica,

```
spineplot(improved ~ treatment)
```



El conjunto de datos de `R`, `UCBAdmissions`contiene los datos agregadps de los solicitantes a universidad de Berkeley a los seis departamentos más grandes en 1973 clasificados por sexo y admisión.

```
data("UCBAdmissions")
?UCBAdmissions
apply(UCBAdmissions, c(2,1), sum)
```

```
##         Admit
## Gender   Admitted Rejected
##   Male       1198     1493
##   Female      557     1278
```

```
prop.table(apply(UCBAdmissions, c(2,1), sum))
```

```
##         Admit
## Gender    Admitted  Rejected
```

```
##   Male   0.2646929 0.3298719
##   Female 0.1230667 0.2823685
```

```
ftable(UCBAdmissions)
```

```
##                 Dept   A   B   C   D   E   F
## Admit    Gender
## Admitted Male        512 353 120 138  53  22
##          Female       89  17 202 131  94  24
## Rejected Male        313 207 205 279 138 351
##          Female       19   8 391 244 299 317
```

Con `ftable` podemos presentar la información con mayor claridad

```
ftable(round(prop.table(UCBAdmissions), 3),
       row.vars="Dept", col.vars = c("Gender", "Admit"))
```

```
##      Gender     Male              Female
##      Admit   Admitted Rejected Admitted Rejected
## Dept
## A               0.113    0.069    0.020    0.004
## B               0.078    0.046    0.004    0.002
## C               0.027    0.045    0.045    0.086
## D               0.030    0.062    0.029    0.054
## E               0.012    0.030    0.021    0.066
## F               0.005    0.078    0.005    0.070
```

Resulta más intereseante mostrar la información por género `Gender` y `Dept` combinados (dimensiones 2 y 3 del array). Nótese que las tasas de admisión por `male` y `female` son más o menos similares en todos los departamentos, excepto en "A", donde las tasas de las mujeres es mayor.

```
# prop.table(UCBAdmissions, c(2,3))
ftable(round(prop.table(UCBAdmissions, c(2,3)), 2),
       row.vars="Dept", col.vars = c("Gender", "Admit"))
```

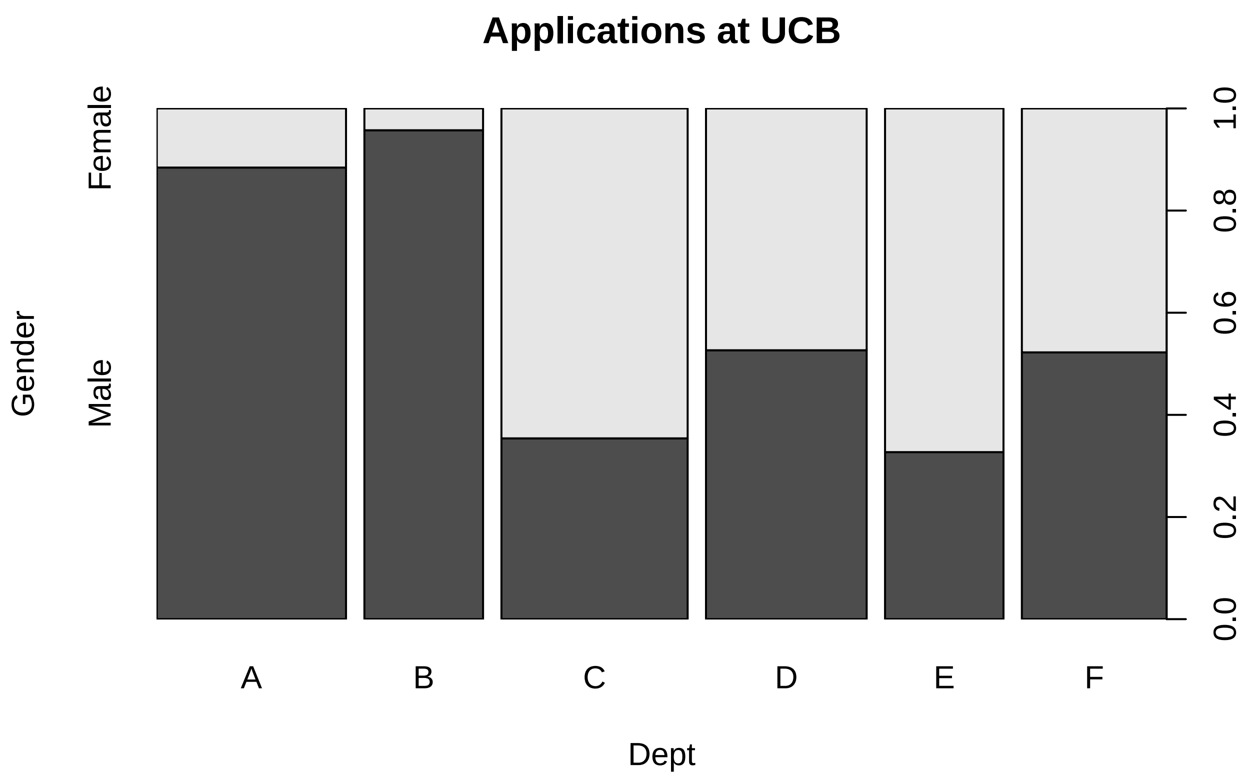
```
##      Gender     Male              Female
##      Admit   Admitted Rejected Admitted Rejected
## Dept
## A                0.62     0.38     0.82     0.18
## B                0.63     0.37     0.68     0.32
## C                0.37     0.63     0.34     0.66
## D                0.33     0.67     0.35     0.65
## E                0.28     0.72     0.24     0.76
## F                0.06     0.94     0.07     0.93
```

```
## Data aggregated over departments
apply(UCBAdmissions, c(1, 2), sum)
```

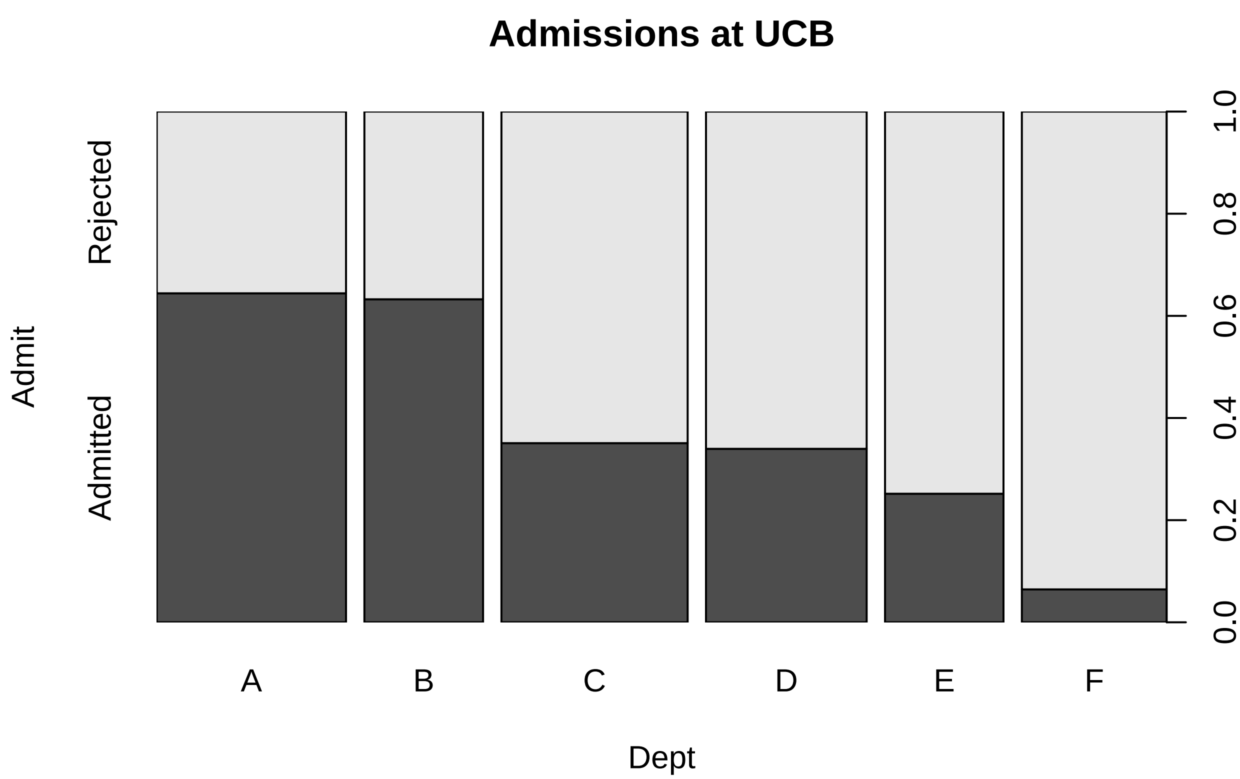
```
##           Gender
## Admit       Male Female
##   Admitted 1198    557
##   Rejected 1493   1278
```

Gráficamente

```
spineplot(margin.table(UCBAdmissions, c(3, 2)),
          main = "Applications at UCB")
```

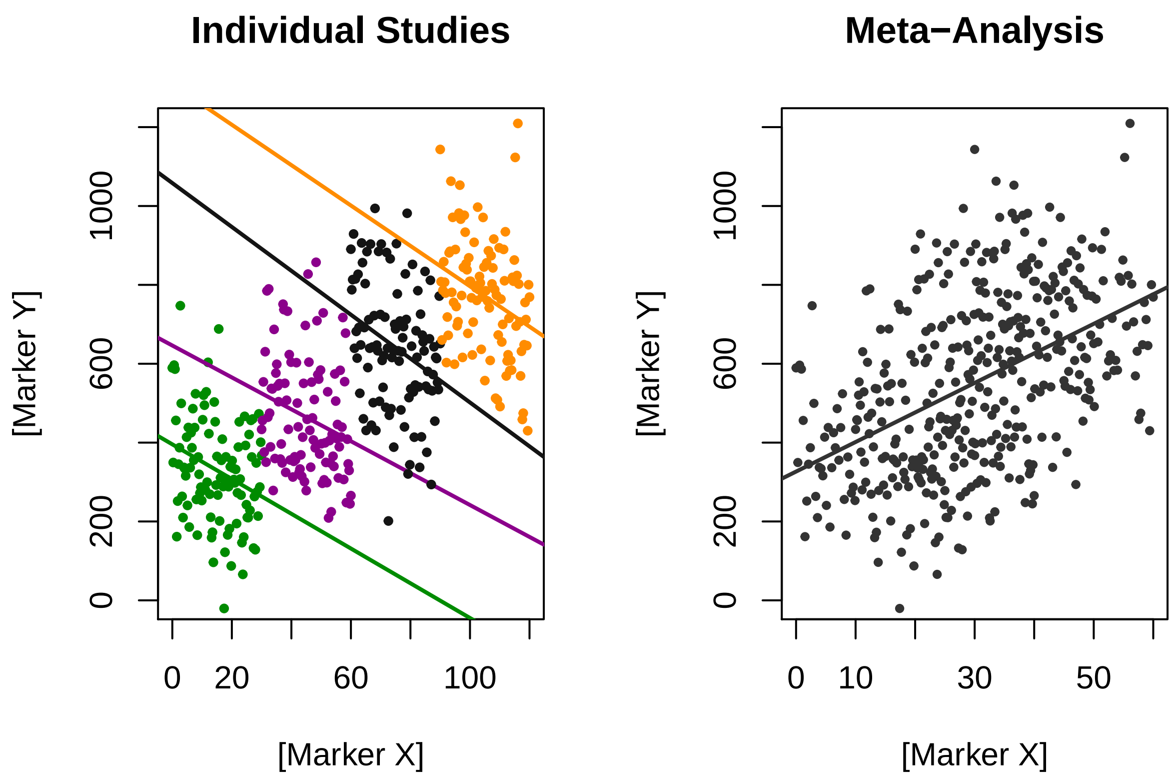



```
spineplot(margin.table(UCBAdmissions, c(3, 1)),
          main = "Admissions at UCB")
```



Estos datos ilustran la denominada *paradoja de Simpson*. Este hecho ha sido analizado como un posible caso de discriminación por sexo en las tasas de admisión en Berkeley. De los 2691 hombres que solicitaron se admitidos, 1198 (44.5 %) fueron admitidos, comparado con las 1835 mujeres de las cuales tan s?lo 557 (30.4 %) fueron admitidas. Se podr?a por tanto concluir que los hombres tienes tasas de admisión mayores que las mujeres. Wikipedia: Gender Bias UC Berkeley.

Otro ejemplo de la Paradoja de Simpson:



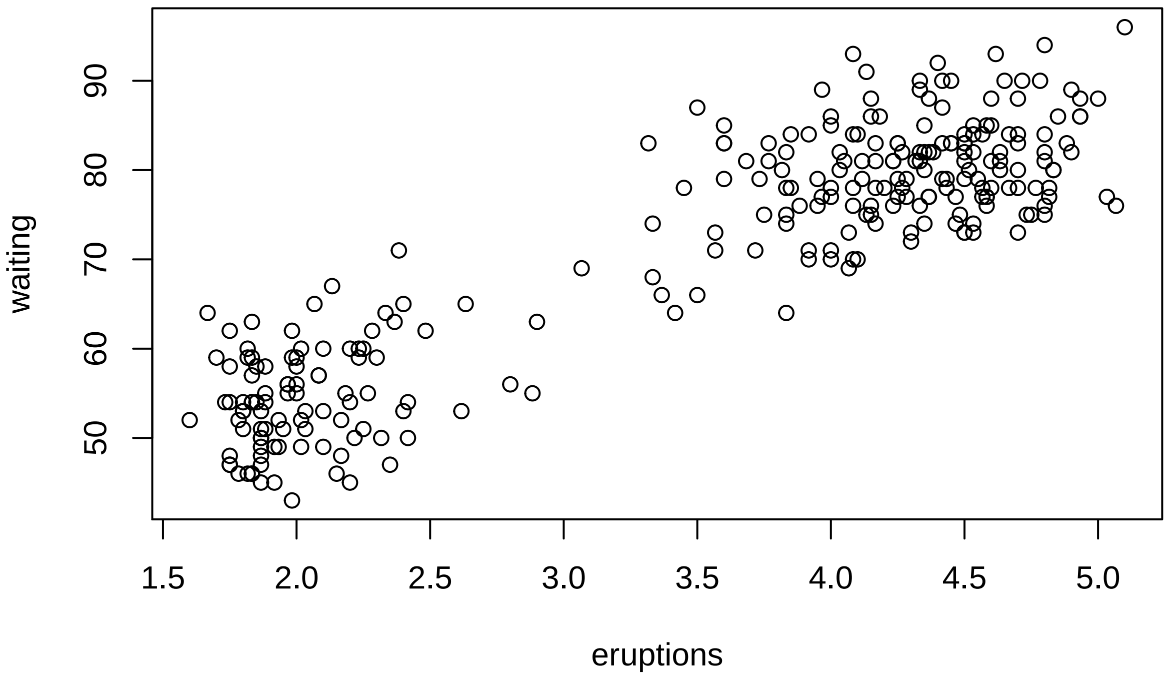
3.6. Datos cuantitativos

```r
head(faithful)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

Consideremos los datos del geyser Old Faithful en el parque nacional de Yellowstone, EEUU.

```r
plot(faithful)
```



3.6.1. Distribuciones de frecuencias

Vamos a utilizar el conjunto de datos `faithful`, para ilustrar el concepto de distribuci?n de frecuencias que consistir? en crear una series de categor?as o intervalos, en los que contaremos el n?mero de observaciones en cada categor?a.

```
duration <- faithful$eruptions
range(duration)
```

```
## [1] 1.6 5.1
```

Crearemos los sub-intervalos entre `[1.6, 5.1]` y la secuencia `{ 1.5, 2.0, 2.5, ... }`.

```
breaks <- seq(1.5,5.5,by=0.5)
breaks
```

```
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

La función `cut` nos permite divider el rango en los intervalos que especifiquemos, con el argumento `right=FALSE`, consideramos el intervalo cerrado por la derecha.

```
duration.cut = cut(duration, breaks, right=FALSE)
```

Con `table` generamos las frecuencias

```
duration.freq = table(duration.cut)
duration.freq
```

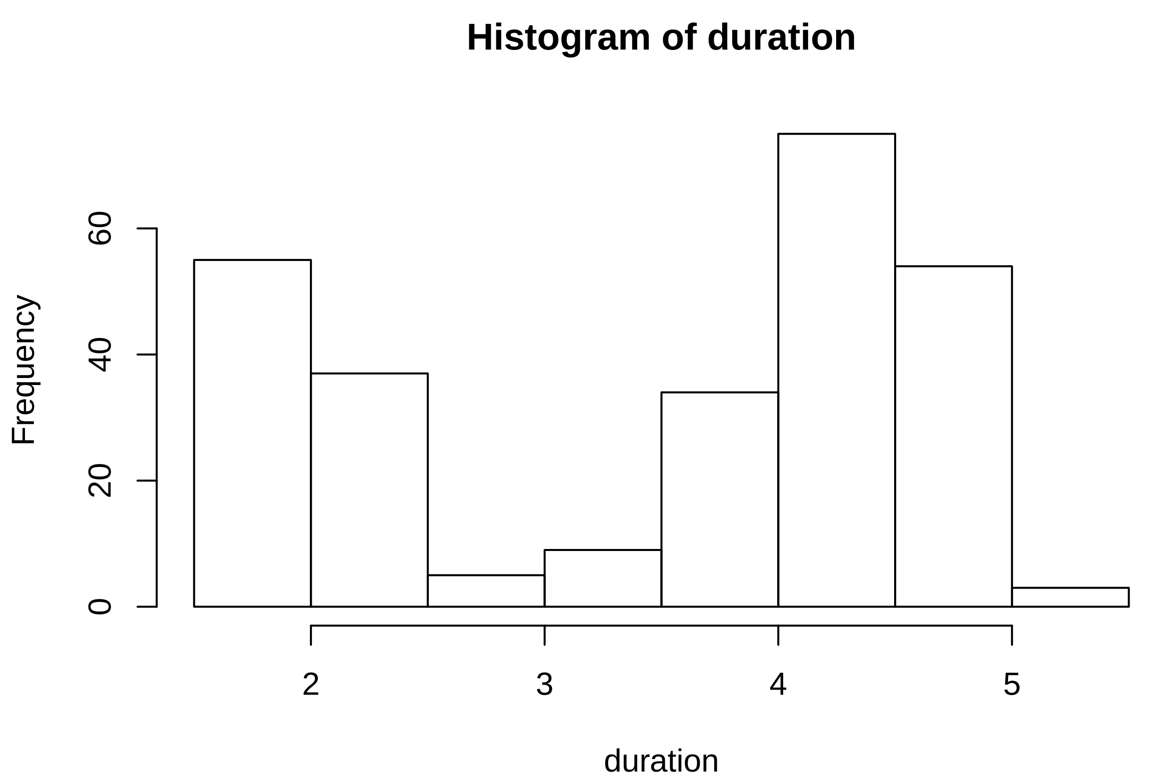
```
## duration.cut
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      41       5       7      30      73      61       4
```

Con `hist` podemos realizarlo de manera autom?tica:

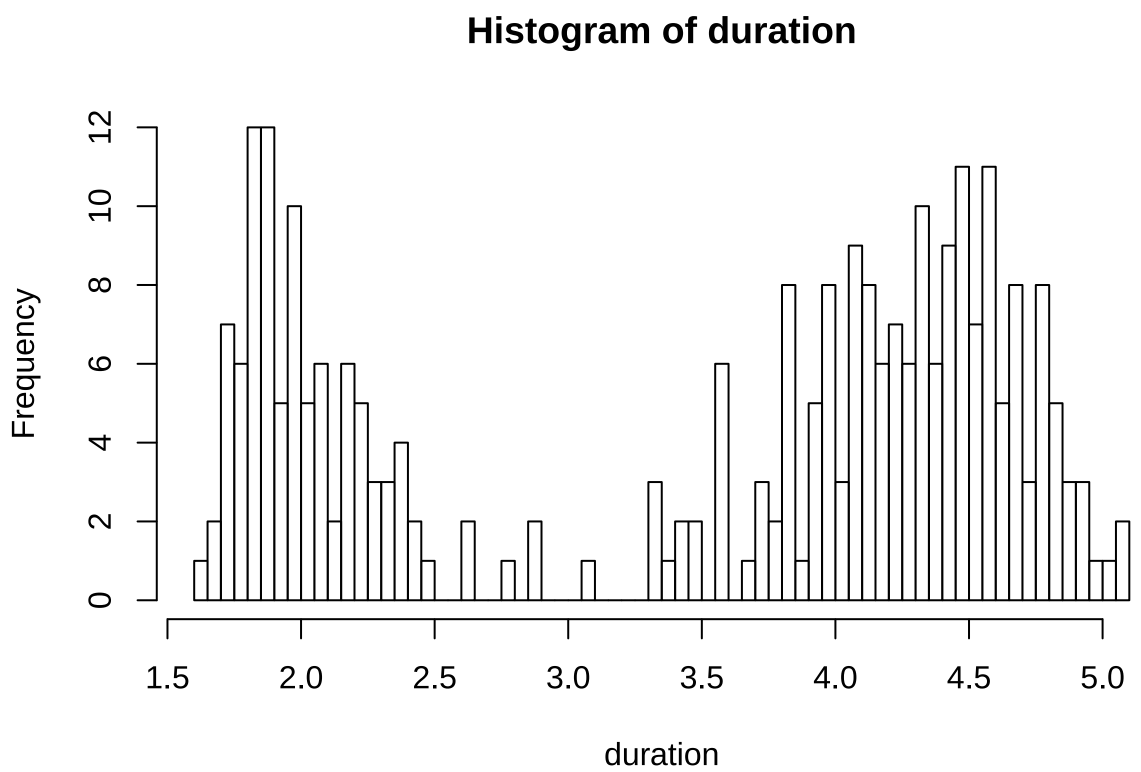
```
freq <- hist(duration)
freq
```

```
## $breaks
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
##
## $counts
## [1] 55 37  5  9 34 75 54  3
##
## $density
## [1] 0.40441176 0.27205882 0.03676471 0.06617647 0.25000000 0.55147059
## [7] 0.39705882 0.02205882
##
## $mids
## [1] 1.75 2.25 2.75 3.25 3.75 4.25 4.75 5.25
##
## $xname
## [1] "duration"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
freq <- hist(duration,breaks = breaks)
```




```
hist(duration,50)
```

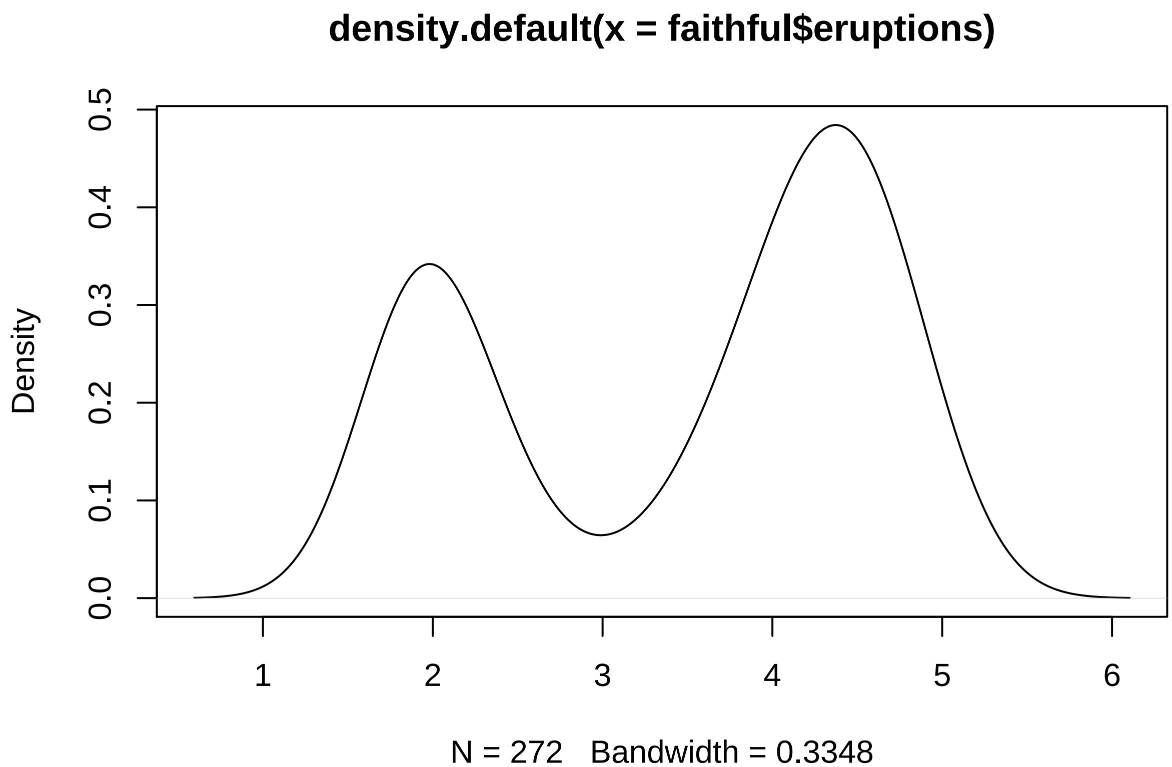


Estimación de densidad construye una estimación dada una distribucion de probabilidad para una muestra dada.

```
require(graphics)
d <- density(faithful$eruptions)
d
```

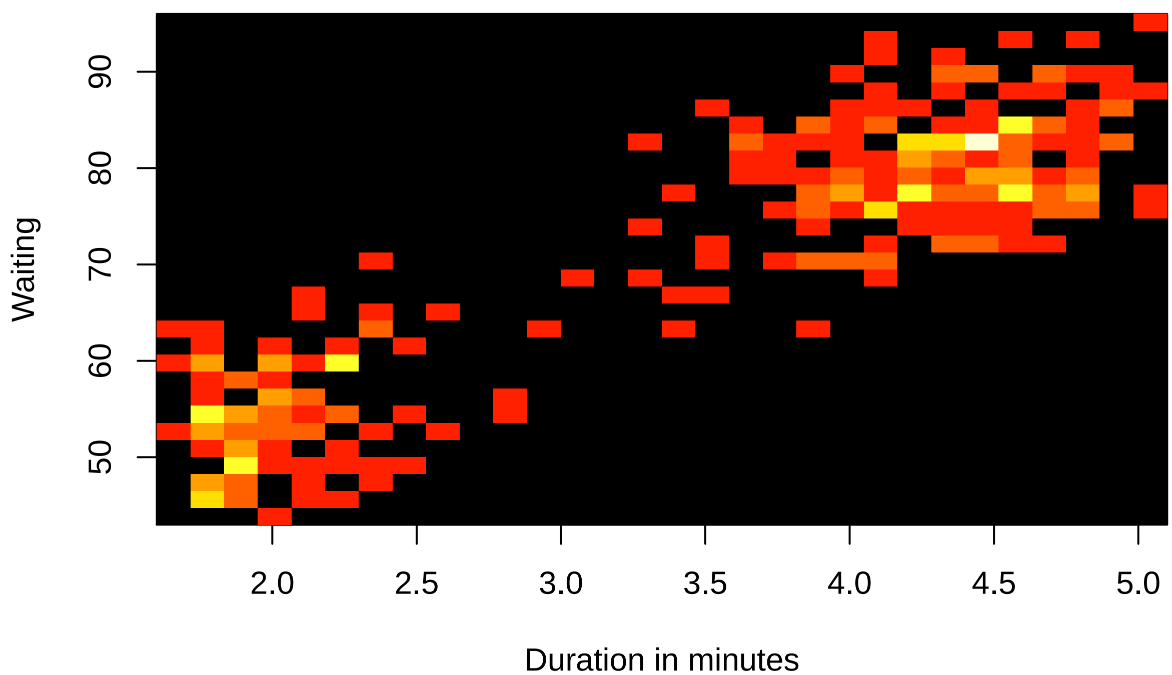
```
##
## Call:
##  density.default(x = faithful$eruptions)
##
## Data: faithful$eruptions (272 obs.); Bandwidth 'bw' = 0.3348
##
##        x                 y
##  Min.   :0.5957   Min.   :0.0002262
##  1st Qu.:1.9728   1st Qu.:0.0514171
##  Median :3.3500   Median :0.1447010
##  Mean   :3.3500   Mean   :0.1813462
##  3rd Qu.:4.7272   3rd Qu.:0.3086071
##  Max.   :6.1043   Max.   :0.4842095
```

```
plot(d)
```



En dos dimensiones:

```
library(gplots)
h2 <- hist2d(faithful, nbins=30,xlab="Duration in minutes",ylab="Waiting")
```




```
h2
```

```
##
## ----------------------------
```

```
## 2-D Histogram Object
## ----------------------------
##
## Call: hist2d(x = faithful, nbins = 30, xlab = "Duration in minutes",
##     ylab = "Waiting")
##
## Number of data points:  272
## Number of grid bins:  30 x 30
## X range: ( 1.6 , 5.1 )
## Y range: ( 43 , 96 )
```

```
names(h2)
```

```
## [1] "counts"   "x.breaks" "y.breaks" "x"        "y"        "nobs"
## [7] "call"
```

Frecuencias relativas

```
duration.relfreq <- duration.freq / nrow(faithful)
tab <- cbind(duration.freq, duration.relfreq)
apply(tab,2,sum)
```

```
##    duration.freq duration.relfreq
##              272                1
```

Distribución de frecuencias acumuladas:

```
cumsum(duration.freq)
```

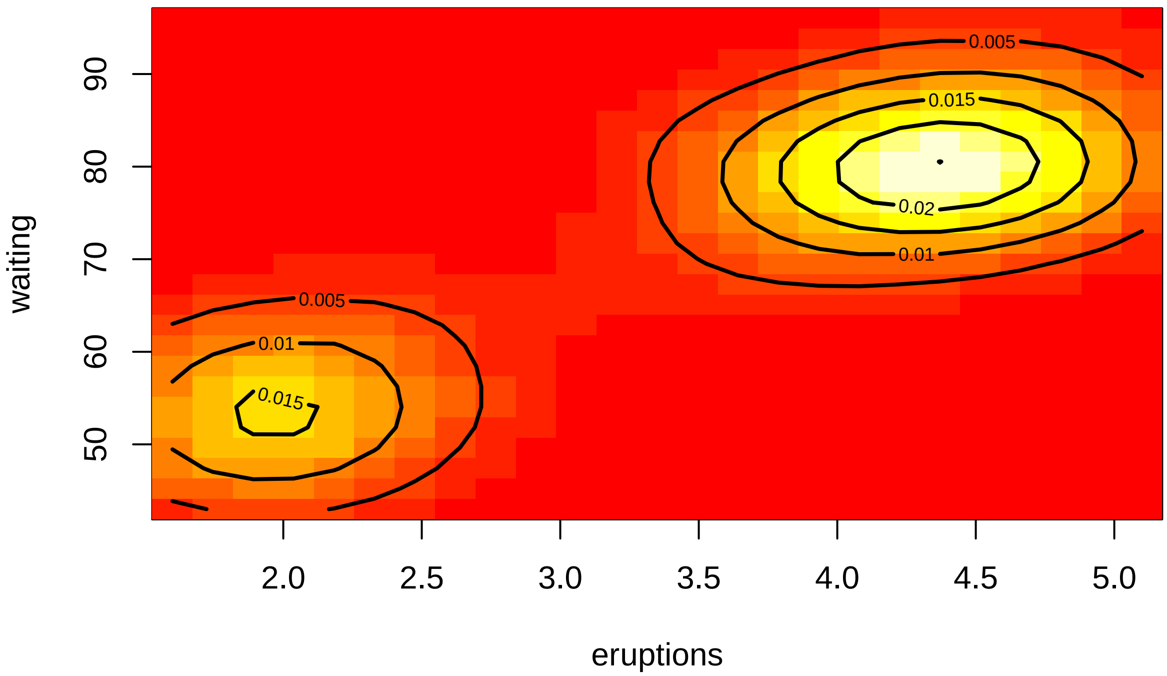
```
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      92      97     104     134     207     268     272
```

```
cumsum(duration.relfreq)
```

```
##   [1.5,2)   [2,2.5)   [2.5,3)   [3,3.5)   [3.5,4)   [4,4.5)   [4.5,5)
## 0.1875000 0.3382353 0.3566176 0.3823529 0.4926471 0.7610294 0.9852941
##   [5,5.5)
## 1.0000000
```

Estimación bivariante tipo kernel

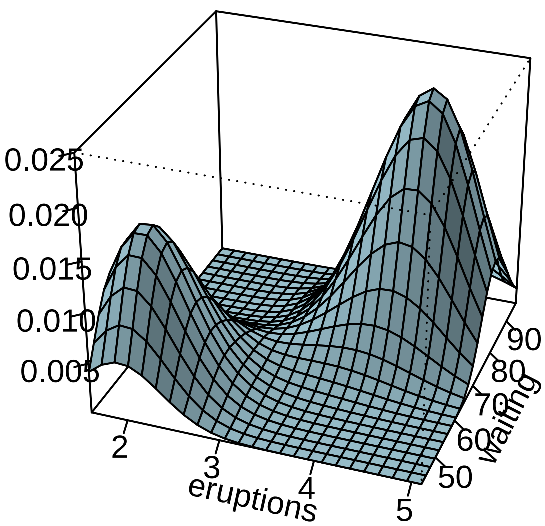
```
data("faithful")
attach(faithful)
Dens2d<-kde2d(eruptions,waiting)
image(Dens2d,xlab="eruptions",ylab="waiting")
contour(Dens2d,add=TRUE,col="black",lwd=2,nlevels=5)
```




```r
detach("faithful")
```

Gráficos persp

```r
persp(Dens2d,phi=30,theta=20,d=5,xlab="eruptions",ylab="waiting",zlab="",shade=.2,col="lightblue",expan
```



Capítulo 4

Introducción a la programación básica con R

4.1. Condicionales

Comparaciones

- equal: `==`

```
"hola" == "hola"
```

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

```
"hola" == "Hola"
```

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

```
1 == 2-1
```

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

- not equal: `!=`

```
a <- c(1,2,4,5)
b <- c(1,2,3,5)
a == b
```

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

```
a != b
```

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

- mayor/menor que: `>` `<`

```
set.seed(1)
a <- rnorm(10)
b <- rnorm(10)
a<b
```

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

- mayor/menor que o igual: `>=` `<=`

```
set.seed(2)
a <- rnorm(10)
```

```
b <- rnorm(10)
a >= b
```

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

- which

```
set.seed(3)
which(a>b)
```

```
## [1] 3 6 9
```

```
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
## [18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

```
which(LETTERS=="R")
```

```
## [1] 18
```

- which.min o which.max

```
set.seed(4)
a <- rnorm(10)
a
```

```
##  [1]  0.2167549 -0.5424926  0.8911446  0.5959806  1.6356180  0.6892754
##  [7] -1.2812466 -0.2131445  1.8965399  1.7768632
```

```
which.min(a)
```

```
## [1] 7
```

```
which.max(a)
```

```
## [1] 9
```

- is.na

```
 a[2] <- NA
is.na(a)
```

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

```
which(is.na(a))
```

```
## [1] 2
```

4.2. Operadores Lógicos

- and: &

```
z = 1:6
which(2 < z & z > 3)
```

```
## [1] 4 5 6
```

- or: |

```
z = 1:6
(z > 2) & (z < 5)
```

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

```
which((z > 2) & (z < 5))
```

```
## [1] 3 4
```

- not: `!`

```
x <- c(TRUE,FALSE,0,6)
y <- c(FALSE,TRUE,FALSE,TRUE)

!x
```

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

Ejemplo:

- `&&` vs `&`

```
x&y
```

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

```
x&&y
```

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

- `||` vs `|`

```
x||y
```

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

```
x|y
```

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

4.3. if

`if(cond1=true) { cmd1 } else { cmd2 }`

```
if(1==0) {
    print(1)
} else {
    print(2)
}
```

```
## [1] 2
```

4.4. ifelse

`ifelse(test, true_value, false_value)`

```
x <- 1:10 # Creates sample data
ifelse(x<5 | x>8, x, 0)
```

```
##  [1]  1  2  3  4  0  0  0  0  9 10
```

4.5. Loops o Bucles

Los más empleados en `R` son `for`, `while` y `apply`. Los menos habituales `repeat`. La función `break` sirve para salir de un bucle loop.

4.5.1. `for`

Sintaxis

```
for(variable in sequence) {
    statements
}
```

```r
for (j in 1:5)
{
  print(j^2)
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

Repetir el bucle guardando los resultados en un vector `x`.

```r
n = 5
x = NULL  # creates a NULL object
for (j in 1:n)
{
  x[j] = j^2
}
x
```

```
## [1]  1  4  9 16 25
```

Generamos el lanzamiento de un dado

```r
nsides = 6
ntrials = 1000
trials = NULL
for (j in 1:ntrials)
{
  trials[j] = sample(1:nsides,1)  # We get one sample at a time
}
mean(trials^2)
```

```
## [1] 14.563
```

Ejemplo:

```r
x <- 1:10
z <- NULL
for(i in seq(along=x)) {
    if (x[i]<5) {
        z <- c(z,x[i]-1)
    } else {
        stop("values need to be <5")
    }
}
## Error: values need to be <5
z
## [1] 0 1 2 3
```

4.5.2. `while`

Similar al bucle `for`, pero las iteraciones están controladas por una condición.

```
z <- 0
while(z < 5) {
    z <- z + 2
    print(z)
}
```

```
## [1] 2
## [1] 4
## [1] 6
```

Capítulo 5

Distribuciones de probabilidad en R

El paquete `stats` de `R` (que se instala por defecto al instalar `R`, y se carga en memoria siempre que iniciamos sesión) implementa numerosas funciones para la realización de cálculos asociados a distintas distribuciones de probabilidad.

Entre las utilizadas más comúnmente podemos citar:

Distribuciones discretas:

Distribución	Nombre en `R`
Binomial	`binom`
Poisson	`pois`
Geométrica	`geom`
Hipergeométrica	`hyper`

Distribuciones contínuas:

Distribución	Nombre en `R`
Uniforme	`unif`
Normal	`norm`
t-Student	`t`
F-Fisher	`F`
Chi-cuadrado	`chisq`

- Examinamos algunas de las operaciones básicas asociadas con las distribuciones de probabilidad.
- Hay un gran número de distribuciones de probabilidad disponibles, pero sólo observamos unas pocas.
- Para obtener una lista completa de las distribuciones disponibles en `R` puede utilizar el siguiente comando:

```
help("Distributions")
```

- Para cada distribución hay cuatro comandos. Los comandos para cada distribución están precedidos de una letra para indicar la funcionalidad:

- `d`: devuelve la función de densidad de probabilidad
- `p`: devuelve la función de densidad acumulada
- `q`: returns the inverse cumulative density function (quantiles)
- `r`: devuelve los números generados aleatoriamente

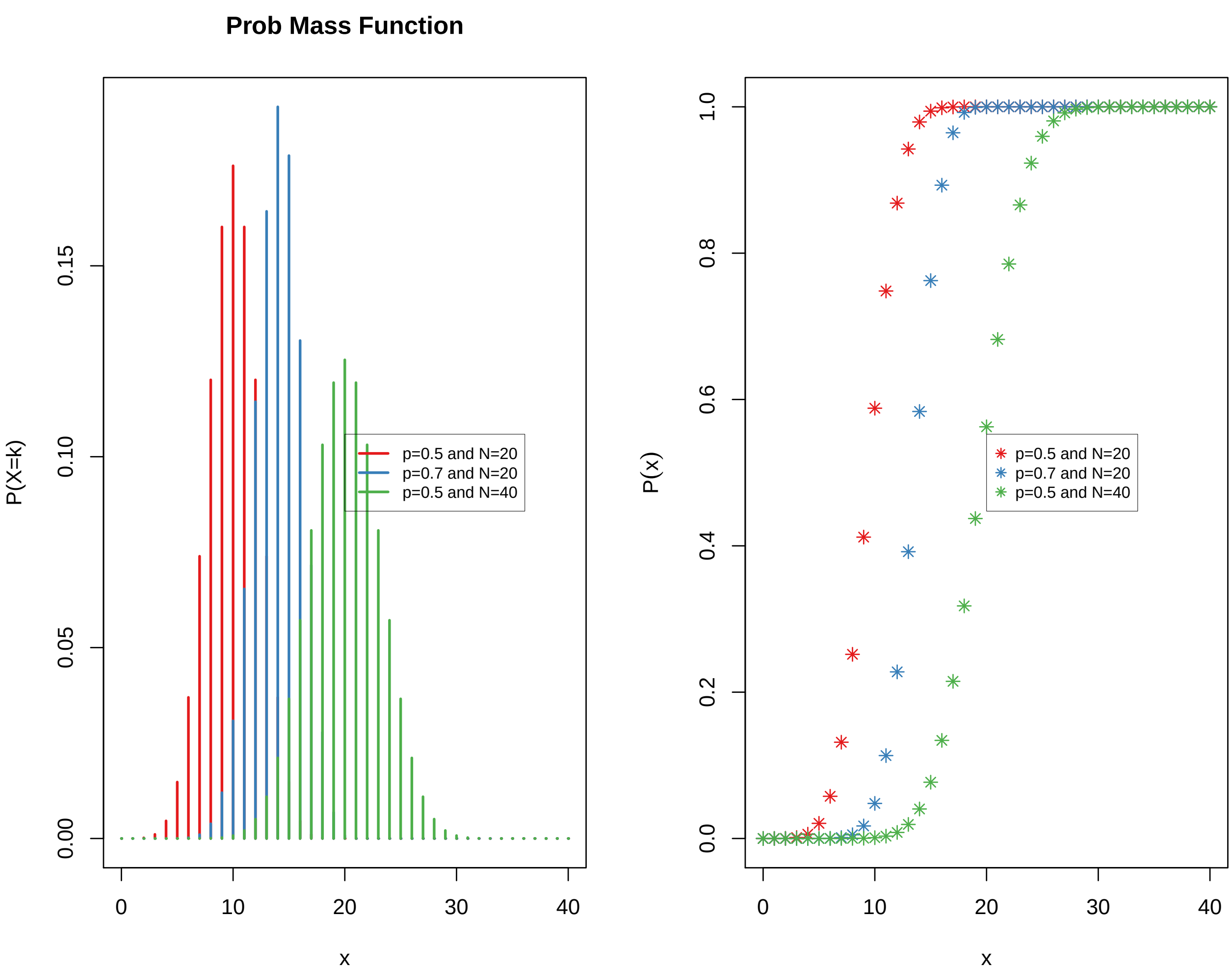
5.1. Distribución binomial $Bin(n, p)$

- La distribución binomial es una distribución de probabilidad discreta. Describe el resultado de ensayos independientes de n en un experimento. Se supone que cada ensayo tiene sólo dos resultados, ya sea éxito o fracaso. Si la probabilidad de un ensayo exitoso es de p, entonces la probabilidad de tener resultados exitosos de k en un experimento de ensayos independientes de n es dada por la **probabilidad de la función de masa**:

$$f(k, n, p) = \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, 2, ..., n$$

La **función de distribución acumulativa** puede expresarse como:

$$F(k; n, p) = \Pr(X \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$



with *mean* np and *variance* $np(1-p)$.

Pregunta:

Suponga que hay doce preguntas de opción múltiple en un examen de matemáticas. Cada pregunta tiene cinco posibles respuestas, y sólo una de ellas es correcta. Encuentre la probabilidad de tener cuatro o menos respuestas correctas si un estudiante intenta responder a cada pregunta al azar.

Solución:

Dado que sólo una de cada cinco respuestas posibles es correcta, la probabilidad de responder correctamente una pregunta al azar es de $1/5 = 0,2$. Podemos encontrar la probabilidad de tener exactamente 4 respuestas correctas por intentos aleatorios de la siguiente manera.

```
p = 1/5
n = 12
k = 4
dbinom(k,size=n,prob=0.2)
```

```
## [1] 0.1328756
```

Para encontrar la probabilidad de tener cuatro o menos respuestas correctas mediante intentos aleatorios, aplicamos la función `dbinom' conk`=0,1,2,4‘.

```
prob <- NULL
for(k in 0:4){
prob <- c(prob,dbinom(k,n,p))
prob
}
prob
```

```
## [1] 0.06871948 0.20615843 0.28346784 0.23622320 0.13287555
```

```
sum(prob)
```

```
## [1] 0.9274445
```

```
# or simply
sum(dbinom(0:4,n,p))
```

```
## [1] 0.9274445
```

Alternativamente, podemos usar la función de probabilidad acumulada para la distribución binomial ‘pbinom’.

```
pbinom(4,size=n,prob=0.2)
```

```
## [1] 0.9274445
```

Solución: La probabilidad de que cuatro o menos preguntas sean contestadas correctamente al azar en un cuestionario de opción múltiple de doce preguntas es del 92,7 %.

¿Cuál es la probabilidad de que 2 ó 3 preguntas sean respondidas correctamente?

```
sum(dbinom(2:3,n,p))
```

```
## [1] 0.519691
```

Pregunta:

Supongamos que la empresa **A** fabricó un producto **B** con una probabilidad de 0,005 de ser defectuoso. Suponga que el producto **B** se envía en una caja que contiene 25 **B** artículos. ¿Cuál es la probabilidad de que una caja elegida al azar contenga exactamente un producto defectuoso?

Solución:

Pregunta reformulada: ¿Cuál es $P(X=1)$ cuando X tiene la distribución $Bin(25, 0{,}005)$?

$$P(X=1) = \binom{25}{1} 0{,}005^1 (1-0{,}005)^{(25-1)}$$

```r
p=0.005
choose(25,1)*p^1*(1-p)^(25-1)
```

```
## [1] 0.1108317
```

```r
# or
dbinom(1,25,0.005)
```

```
## [1] 0.1108317
```

¿Cuál es la probabilidad de que una caja elegida al azar no contenga más de un artículo defectuoso?

Solución:

$$P(X \leq 1) = P(X=0) + P(X=1)$$

```r
sum(dbinom(0:1,25,p))
```

```
## [1] 0.9930519
```

```r
# or
pbinom(1,25,0.005)
```

```
## [1] 0.9930519
```

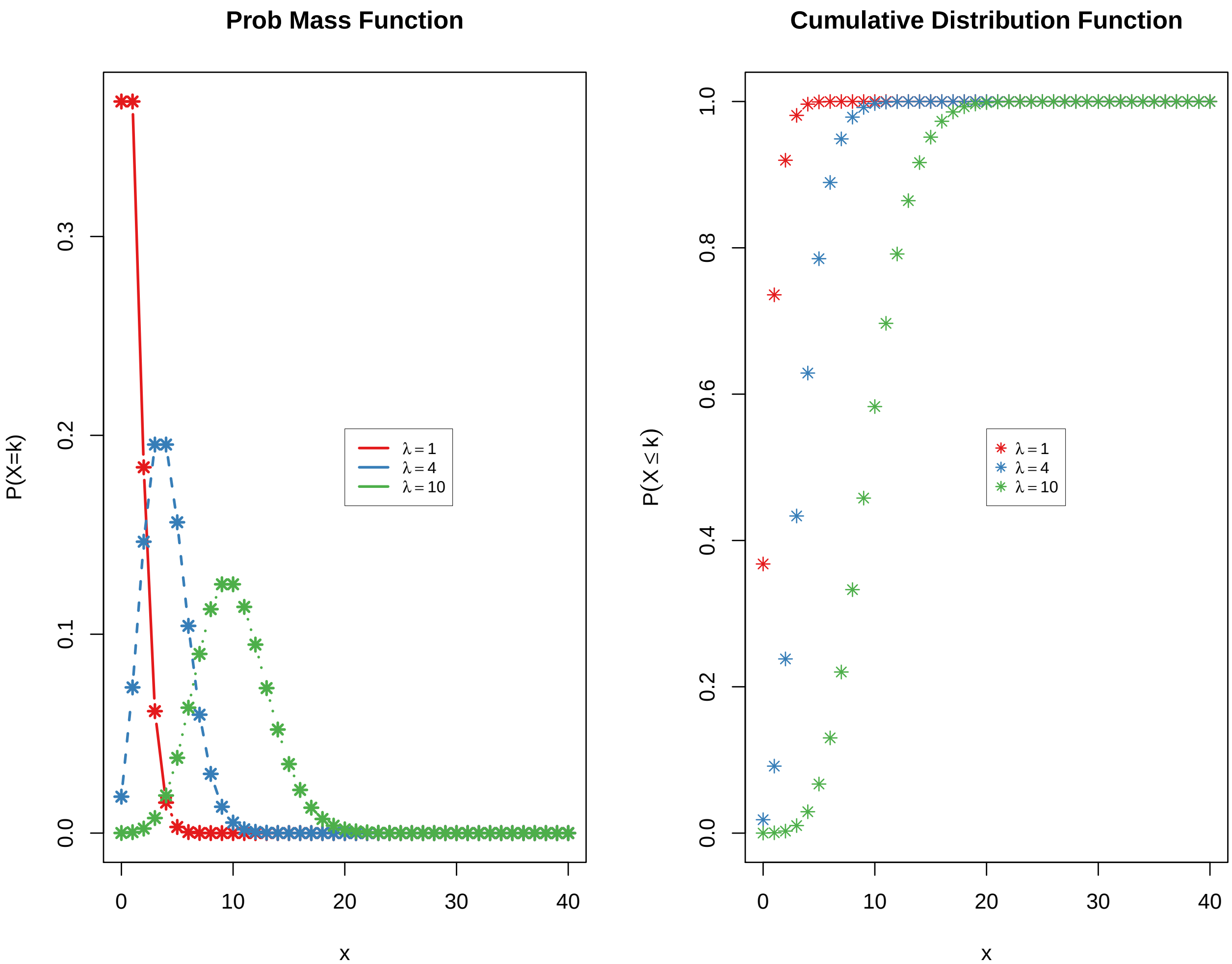
5.2. Distribución de Poisson $Pois(\lambda)$

La distribución de Poisson es la distribución de probabilidad de ocurrencias de eventos independientes en un intervalo. Si λ es la ocurrencia media por intervalo, entonces la probabilidad de tener ocurrencias k dentro de un intervalo dado es la función de masa de probabilidad dada por:

$$\Pr(k \text{ eventos en el intervalo}) = \frac{\lambda^k e^{-\lambda}}{k!}$$

La **función de densidad acumulada** para la función de probabilidad acumulativa de Poisson es

$$P(X \leq x \mid \lambda) = \frac{e^{-\lambda}\lambda^x}{x!} \quad \text{for } x = 0, 1, 2, \ldots$$



Pregunta:

Supongamos que el número de plantas individuales de una especie dada que esperamos encontrar en un cuadrado de un metro cuadrado sigue la distribución de Poisson con una media de $\lambda = 10$. Encuentra la probabilidad de encontrar exactamente 12 dólares por persona.

Pregunta:

Si hay doce coches cruzando un puente por minuto en promedio, encuentre la probabilidad de tener diecisiete o más coches cruzando el puente en un minuto en particular.

5.2.1. Aproximación de Binomial como Poisson

Example

El cinco por ciento (5 %) de las bombillas del árbol de Navidad fabricadas por una compañía son defectuosas. El Gerente de Control de Calidad de la compañía está muy preocupado y por lo tanto toma muestras aleatorias de 100 bulbos que salen de la línea de montaje. Sea X el número en la muestra que está defectuoso. ¿Cuál es la probabilidad de que la muestra contenga como máximo tres bulbos defectuosos?

```
p = 0.05
k = 3
n = 100
pbinom(k,size=n,prob=p)
```

```
## [1] 0.2578387
```

Se puede demostrar que la distribución Binomial puede ser aproximada con la función de masa de probabilidad de Poisson cuando n es grande. Usando la distribución de Poisson, la media $\lambda = np$

```
lambda <- n*p
sum(dpois(0:3,lambda))
```

```
## [1] 0.2650259
```

Es importante tener en cuenta que la aproximación de Poisson a la distribución binomial funciona bien sólo cuando n es grande y p es pequeño. En general, la aproximación funciona bien si $n \geq 20$ y $p \leq 0{,}05$, o si $n \geq 100$ y $p \leq 0{,}10$.

5.3. Distribution Exponencial $Exp(\lambda)$

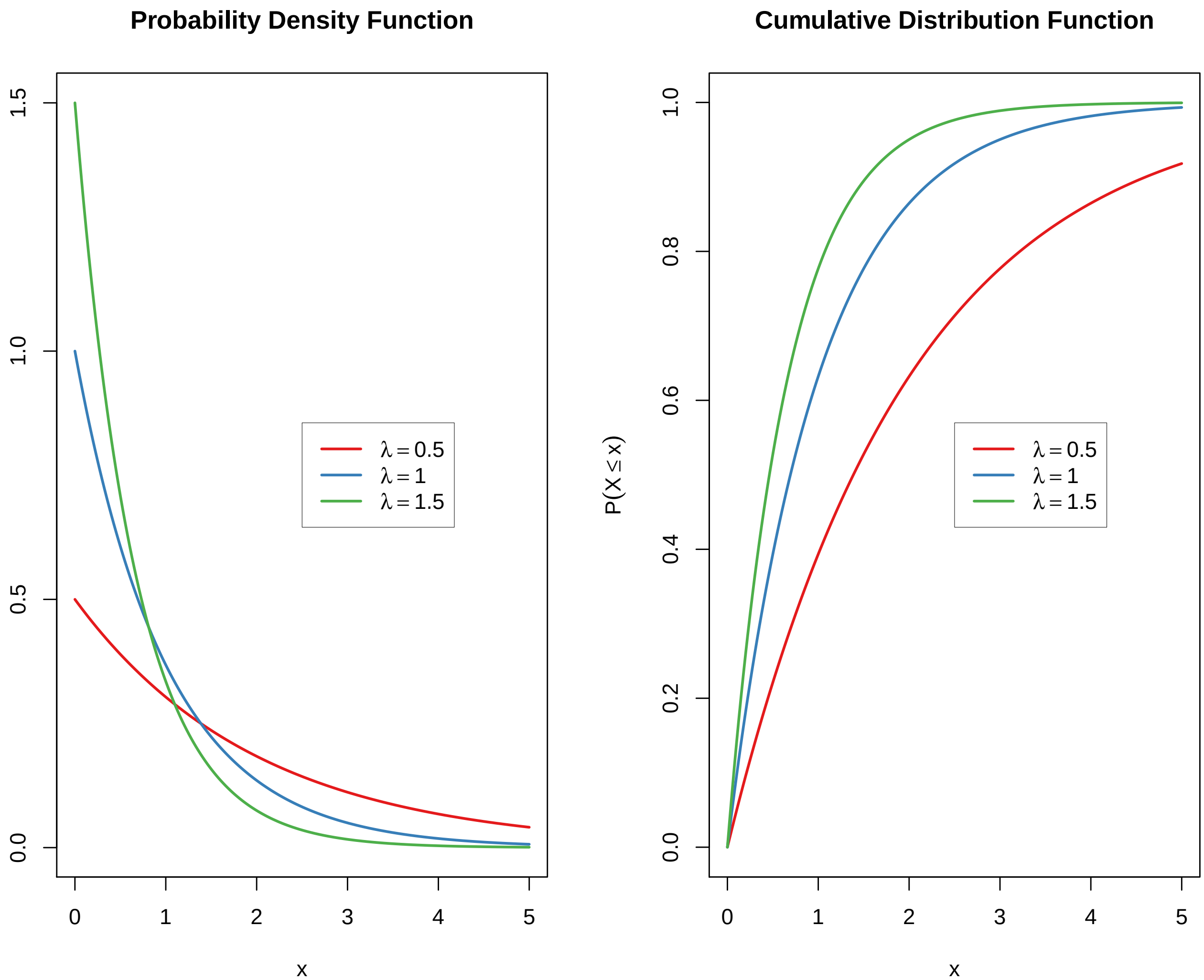
- La distribución exponencial es la distribución de probabilidad que describe el tiempo entre eventos en un proceso de Poisson, es decir, un proceso en el que los eventos ocurren continua e independientemente a una tasa promedio constante.

- Es un caso particular de la distribución gamma. Es el continuo análogo de la distribución geométrica, y tiene la propiedad clave de no tener memoria. Además de ser utilizado para el análisis de los procesos de Poisson, se encuentra en varios otros contextos.

- La función de densidad de probabilidad (pdf) de una distribución exponencial como

$$f(x; \lambda) = \lambda \exp(-\lambda x)$$

donde $\lambda > 0$ es la tasa del evento (también conocida como parámetro de tasa, tasa de llegada, tasa de mortalidad, tasa de fracaso, tasa de transición). La variable exponencial $x \in [0,$
$infty]$



- La función de distribución acumulada de la distribución exponencial es

$$F(x) = \Pr(X \leq x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Con media $\mathbb{E}(X) = 1/\lambda$, and $\mathbb{V}ar(X) = 1/\lambda^2$.

Pregunta:

Supongamos que la cantidad de tiempo que uno pasa en un banco se distribuye exponencialmente con una media de 10 minutos, $\lambda = 1/10$.

a. ¿Cuál es la probabilidad de que un cliente pase más de 15 minutos en el banco?
b. ¿Cuál es la probabilidad de que un cliente pase más de 15 minutos en el banco si aún está en el banco? después de 10 minutos?

Soluciones:

a.

$$P(X > 15) = e^{-15\lambda} = e^{-3/2} = 0{,}2231$$

```r
pexp(15,rate=1/10,lower.tail = FALSE) # or 1-pexp(15,rate=1/10)
```

```
## [1] 0.2231302
```

b.

$$P(X > 15|X > 10) = P(X > 5) = e^{-1/2} = 0{,}606$$

```r
pexp(5,rate=1/10,lower.tail = FALSE)
```

```
## [1] 0.6065307
```

5.4. Distribution Normal $\mathcal{N}(\mu, \sigma^2)$

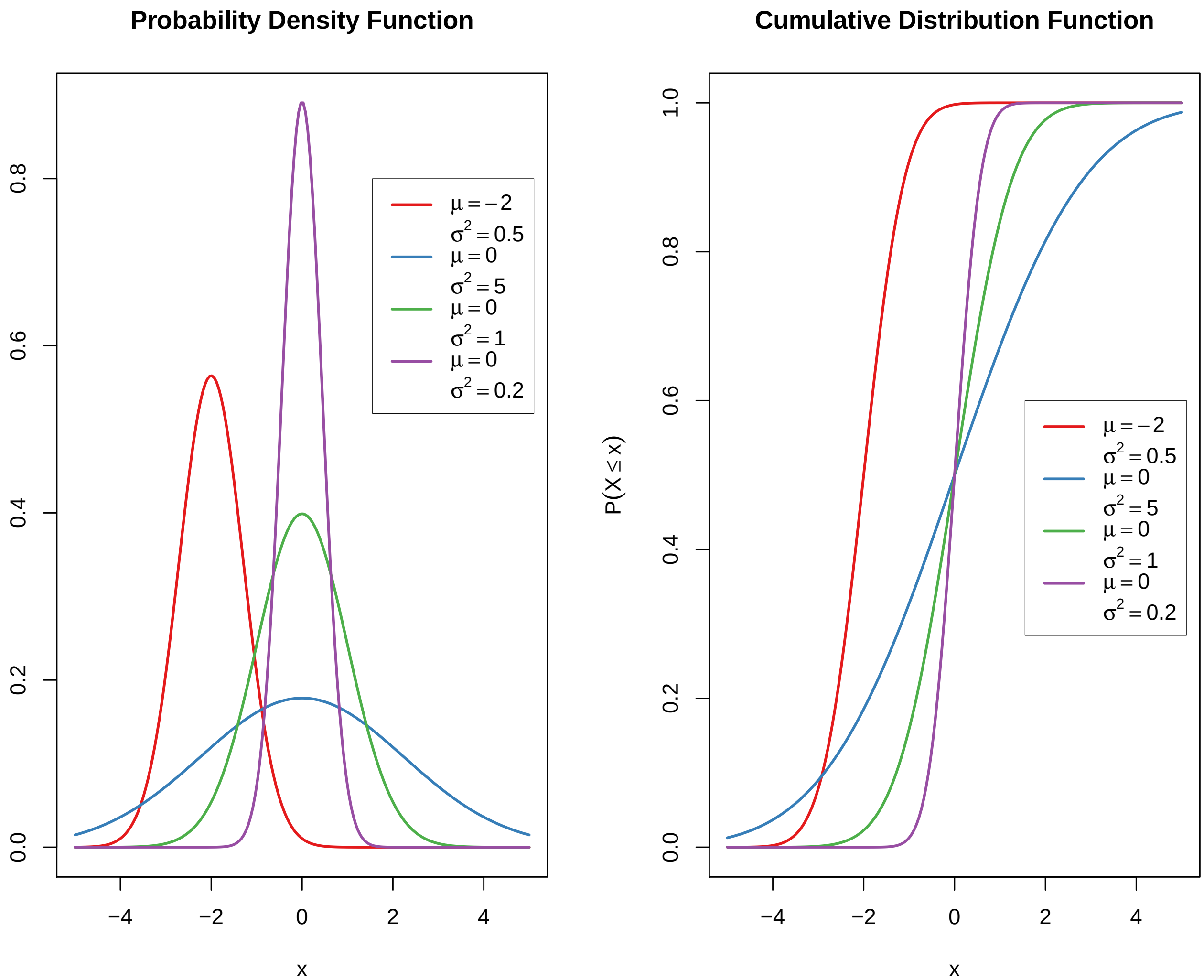
- La función de densidad de probabilidad de la distribución Normal es:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

dónde

- μ es la media de la distribución (también la mediana y el modo).
- σ es la desviación estándar ($\sigma > 0$).
- σ^2 es la variación.
- El proceso para estandarizar la distribución Normal consiste en transformar la variable Normal $N(\mu, \sigma)$ en $N(0, 1)$, es decir

$$Z = \frac{X - \mu}{\sigma} \sim N(0, 1)$$



Pregunta:

X es una variable normalmente distribuida con una media de $\mu = 30$ y una desviación estándar de $\sigma = 4$. Encontrar

a) $P(x < 40)$

b) $P(x > 21)$

c) $P(30 < x < 35)$

Solucion:

a) Para $x = 40$, la z estandarizada es $(40 - 30)/4 = 2{,}5$ y por tanto:

$$P(X < 40) = P(Z < 2{,}5) = 0{,}9938$$

```r
pnorm(2.5) # or
```

```
## [1] 0.9937903
```

```r
pnorm(40,mean=30,sd=4,lower.tail=TRUE)
```

```
## [1] 0.9937903
```

b) $P(x > 21)$

```
pnorm(21,mean=30,sd=4,lower.tail = FALSE)
```

```
## [1] 0.9877755
```

c) $P(30 < x < 35)$

```
pnorm(35,mean=30,sd=4,lower.tail = TRUE)-pnorm(30,mean=30,sd=4,lower.tail = TRUE)
```

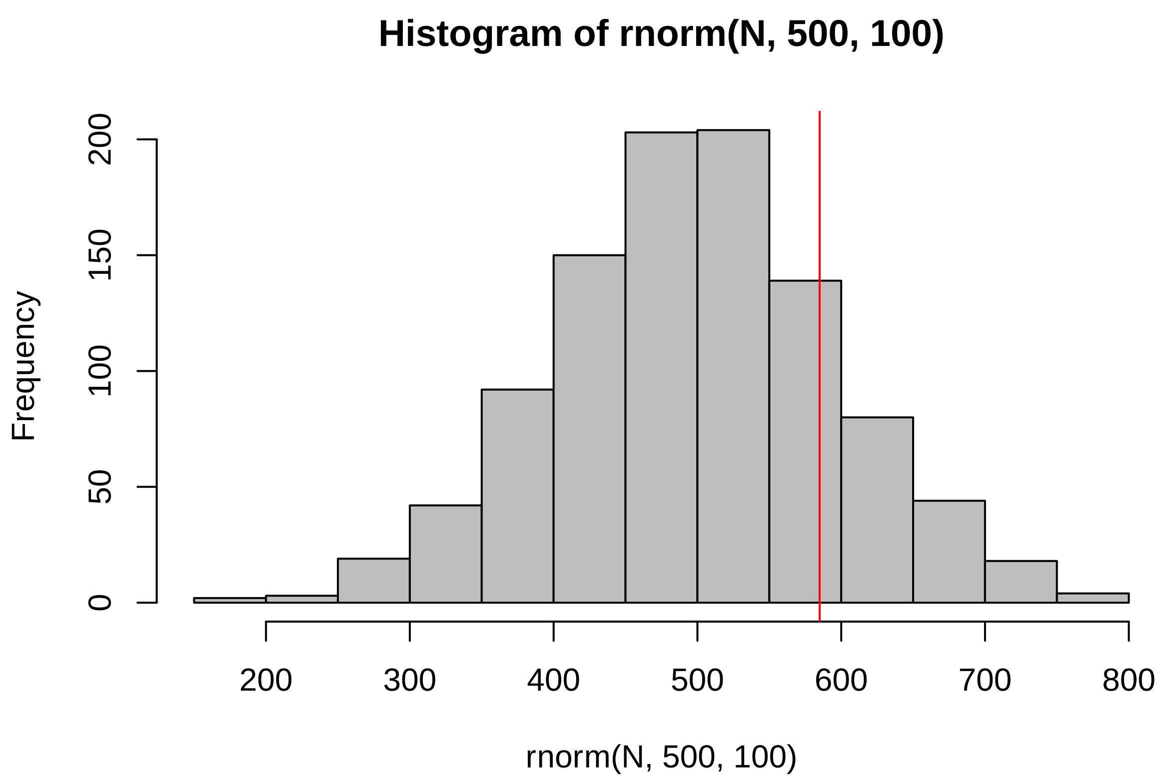
```
## [1] 0.3943502
```

Pregunta:

El ingreso a una determinada universidad se determina mediante un examen nacional. Los resultados de esta prueba se distribuyen normalmente con una media de 500 y una desviación estándar de 100. Tom quiere ser admitido en esta universidad y sabe que debe obtener mejores resultados que al menos el 70 % de los estudiantes que tomaron el examen. Tom toma el examen y saca 585 puntos. ¿Será admitido en esta universidad?

Solución:

```
N = 1000
hist(rnorm(N,500,100),20,col="grey")
abline(v=585,col=2)
```



Es $P(X < 585) >$70 %?

```
pnorm(585,mean=500,sd=100)
```

```
## [1] 0.8023375
```

Tom obtuvo una puntuación mejor que el 80.23 % de los estudiantes que tomaron el examen y será admitido en esta universidad.

5.5. Distribución Uniforme $U(a,b)$

- La distribución uniforme continua es la distribución de probabilidad de la selección de números aleatorios del intervalo continuo entre a y b. Su función de densidad está definida por lo siguiente.

$$f(x) = \begin{cases} \frac{1}{a-b} & a \leq x \leq b \\ 0 & \text{elsewhere} \end{cases}$$

- La función `runif()` puede ser usada para simular variables aleatorias uniformes independientes de n. Por ejemplo, podemos generar 5 números aleatorios uniformes en $[0,1]$ como sigue:

```
set.seed(1234)
runif(5)
```

```
## [1] 0.1137034 0.6222994 0.6092747 0.6233794 0.8609154
```

- Para generar números uniformes en un intervalo del formulario $[a,b]$, usamos los argumentos `min=a`. y `max=b`. Por ejemplo:

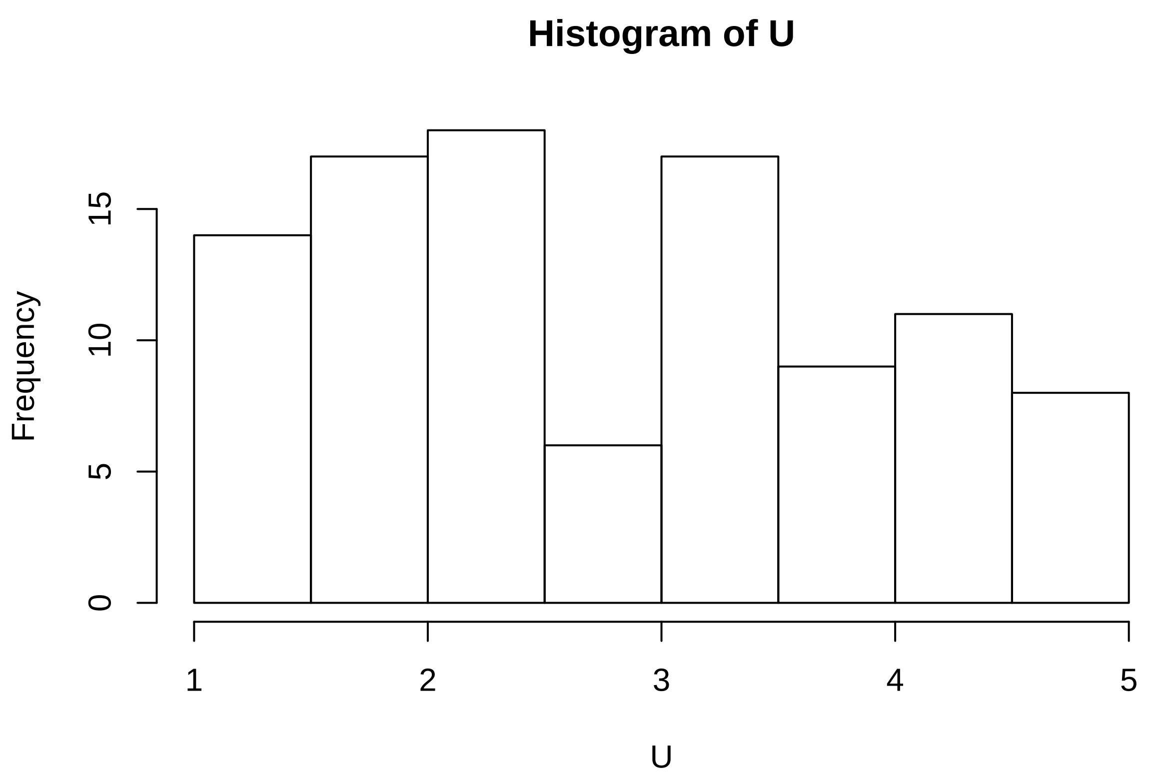
```
set.seed(1234)
runif(3, 1.2, 5.8)
```

```
## [1] 1.723036 4.062577 4.002664
```

da 3 números uniformes en [1,2, 5,8].

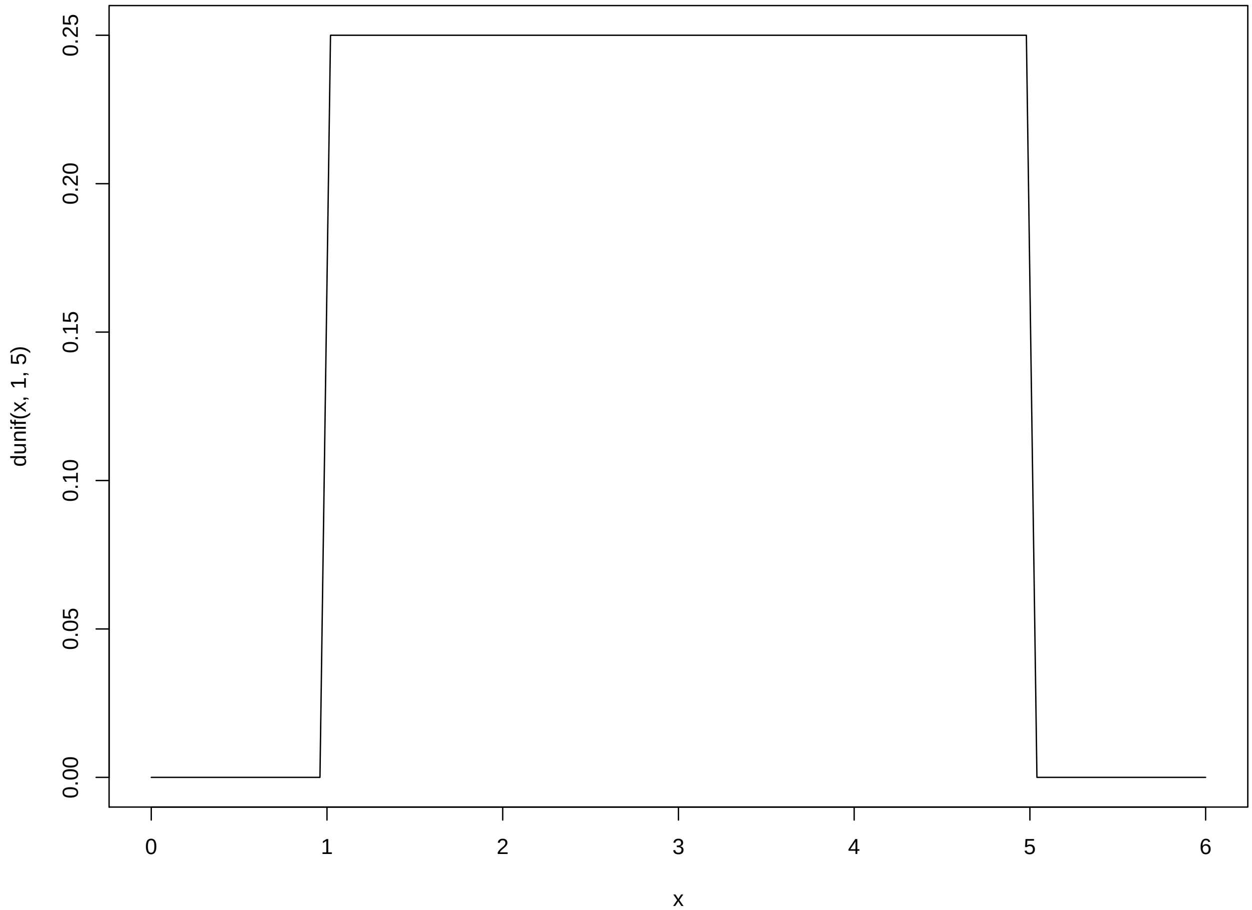
- En el siguiente ejemplo, asignaremos 100 números uniformes independientes en el intervalo $[1,5]$ a un objeto vectorial llamado `U`:

```
set.seed(1234)
U <- runif(100, 1, 5)
hist(U)
```



- La función de densidad puede ser calculada usando `dunif()`

```r
curve(dunif(x, 1, 5), from=0, to=6)
```



La media de U es $\mathbb{E}[U] = frac12(a+b)$, la mediana es $\mathbb{M}[U] = \frac{1}{2}(a+b)$ y la Varianza es $\text{Var}[U] = \frac{1}{12}(b-a)^2$.

Capítulo 6

Modelos lineales y análisis de la varianza

6.1. Principios de la modelización estadística

- Dado un conjunto de variables, cada una de las cuales es un vector de lecturas de un rasgo específico de las muestras en un experimento.
- **Problema:** ¿De qué manera una variable Y depende de otras variables $X_1, ..., X_n$ en el estudio?
- Un **modelo estadístico** define una relación matemática entre los X_i y Y. El modelo es una representación del real Y que pretende reemplazarlo en la medida de lo posible. Al menos el modelo debería capturar la dependencia de Y de los X_i.

6.1.1. Identificar y Caracterizar Variables

Este es el primer paso en el modelado:

- Qué variable es la variable de respuesta;
- Qué variables son las variables explicativas;
- ¿Son las variables explicativas continuas, categóricas o una mezcla de ambas?
- ¿Cuál es la naturaleza de la variable de respuesta? ¿Es una medición contínua, un conteo, una proporción, una categoría o un tiempo hasta un evento?

6.1.2. Tipos de variables y tipo de modelo

- En función de las variables explicativas:

Las variables explicativas	**Modelo**
Todas las variables explicativas continuas	Regresión
Todas las variables explicativas categóricas	Análisis de varianza (ANOVA)
Variables explicativas tanto continuas como categóricas	Regresión, Análisis de Covarianza (ANCOVA)

- En función de la variable respuesta:

La variable de respuesta	*¿Qué tipo de datos es?*
Continuo	Regresión normal, Anova, Ancova
Proporción	Regresión logística
Conteos	Modelos log-lineales (también conocidos como regresión de Poisson)
Binario	Regresión logística binaria
Tiempo hasta evento	Análisis de supervivencia

6.2. El modelo lineal general

Los modelos lineales son una de las herramientas más importantes del análisis cuantitativo. Los utilizamos cuando queremos predecir –o explicar– una variable dependiente a partir de una o más variables independientes.

Se trata de un modelo para el **análisis de regresión**, que tiene como objetivo determinar una función matemática que describa el comportamiento de una variable dados los valores de otra u otras variables.

En el **Análisis de regresión simple**, se pretende estudiar y explicar el comportamiento de una variable que notamos y, y que llamaremos **variable respuesta**, **variable dependiente** o **variable de interés**, a partir de otra variable, que notamos x, y que llamamos **variable explicativa**, **variable independiente**, **covariable** o **regresor**. El principal objetivo de la regresión es encontrar la función que mejor explique la relación entre la variable dependiente y las independientes.

- Una forma muy general para el modelo sería

$$y = f(x_1, x_2, ..., x_p) + \epsilon,$$

donde f es una función desconocida y ϵ es el error en esta representación. Puesto que normalmente no tenemos suficientes datos para intentar estimar f directamente (*problema inverso*), normalmente tenemos que asumir que tiene alguna forma restringida.

- La forma más simple y común es el **modelo lineal (LM)**.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_z + \epsilon,$$

donde β_i $i = 0, 1, 2$ son parámetros *desconocidos*. β_0 se llama el término de intercepción. Por lo tanto, el problema se reduce a la estimación de cuatro valores en lugar de los complicados e infinitos f dimensionales.

- Un modelo lineal simple con una sola variable exploratoria se define como:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

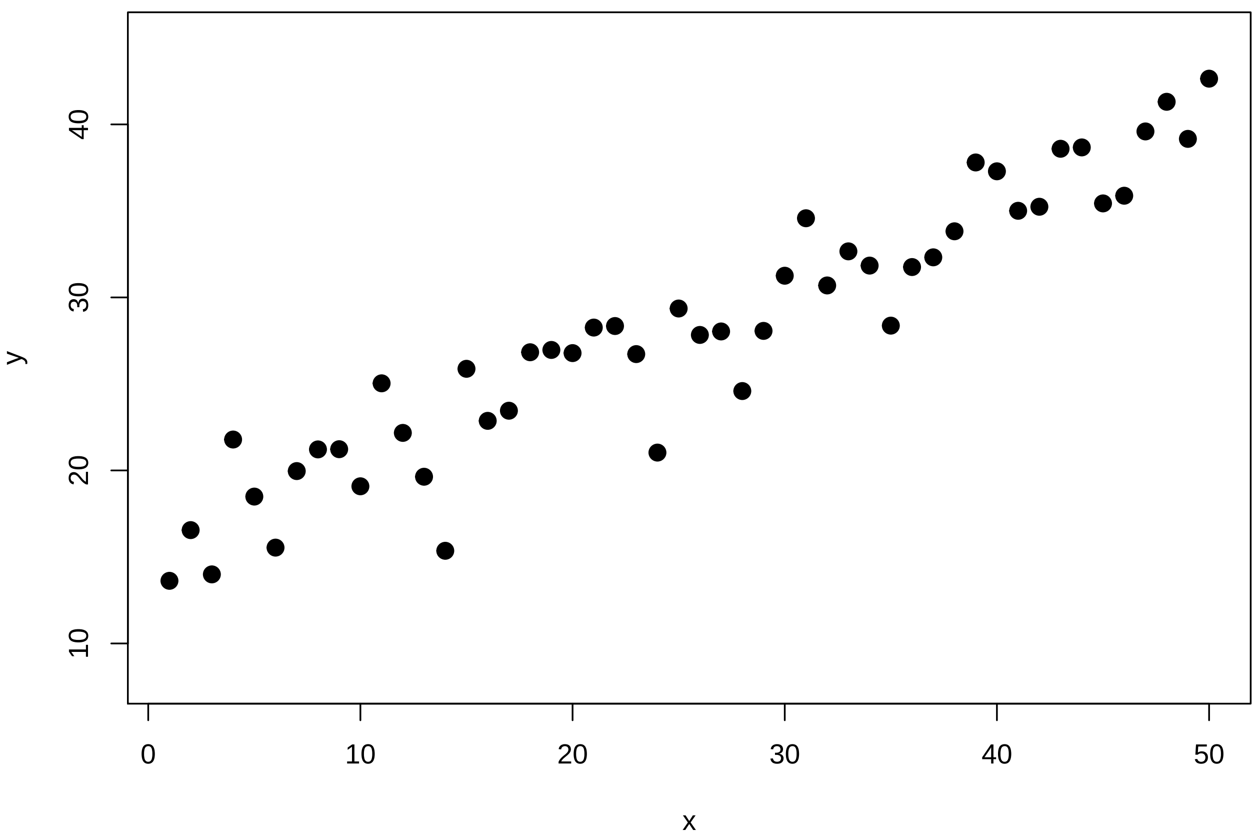
donde $\hat{y}$ son los valores ajustados para $\hat{beta}_0$ (intercepto) y $\hat{\beta}_1$ (pendiente). Luego por un x_i dado obtenemos un $\hat{y}_i$ que se aproxima a y_i

Vamos a crear un ejemplo (con $p = 1$):

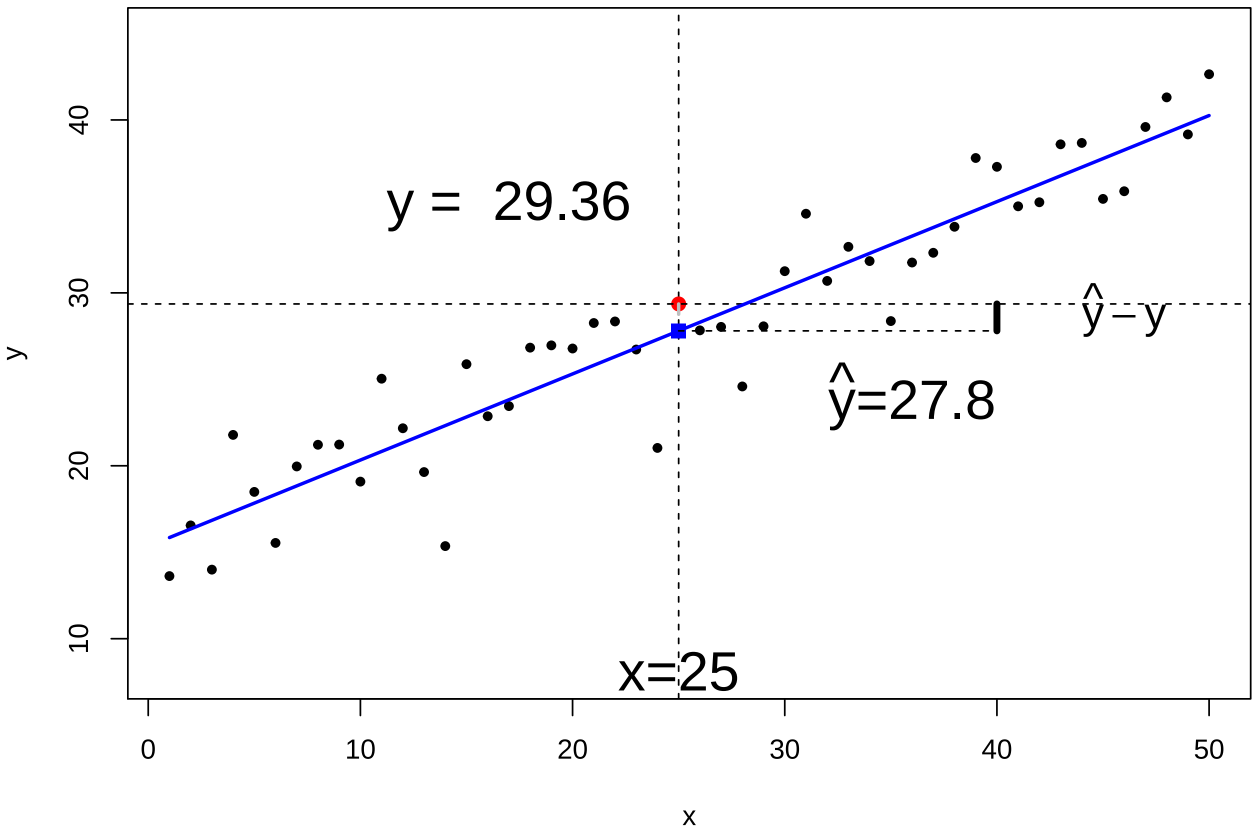
```
set.seed(1)
n <- 50

x <- seq(1,n)
 beta0 <- 15
 beta1 <- 0.5

sigma <- 3 # desviación típica de los errores
eps <- rnorm(n,mean=0,sd=3) # generar errores aleatorios gaussianos

# Generar datos aleatorios
 y <- beta0 + beta1*x  +  eps
```

Representamos los datos

```
plot(x,y,ylim = c(8,45), cex=1.3, xlab = "x", ylab="y",pch=19)
```



Un procedimiento matemático para encontrar la curva que mejor se ajusta a un conjunto dado de puntos minimizando la suma de los cuadrados de los residuos de los puntos de la línea ajustada. Ilustración de los mínimos cuadrados de ajuste



Podemos calcular directamente las cantidades de interés, es decir, la solución ordinaria de mínimos cuadrados:

$$\min_{\beta_0,\beta_1} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Donde

$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$ y $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$

En forma matricial, con $X = [1 : x_1 : ... : x_p]$

$$\hat{\beta} = (X'X)^{-1}X'y$$

donde $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$

6.3. Definición de modelos en `R`

Para completar una regresión lineal usando `R` primero es necesario entender la sintaxis para definir los modelos.

- Un aspecto fundamental de los modelos es el uso de fórmulas modelo para especificar las variables involucradas en el modelo y las posibles interacciones entre las variables explicativas incluidas en el modelo.
- Una fórmula modelo se introduce en una función que realiza una regresión lineal o anova, por ejemplo.
- Mientras que una fórmula modelo se parece un poco a una fórmula matemática, los símbolos de la "ecuación" significan cosas diferentes a las del álgebra.

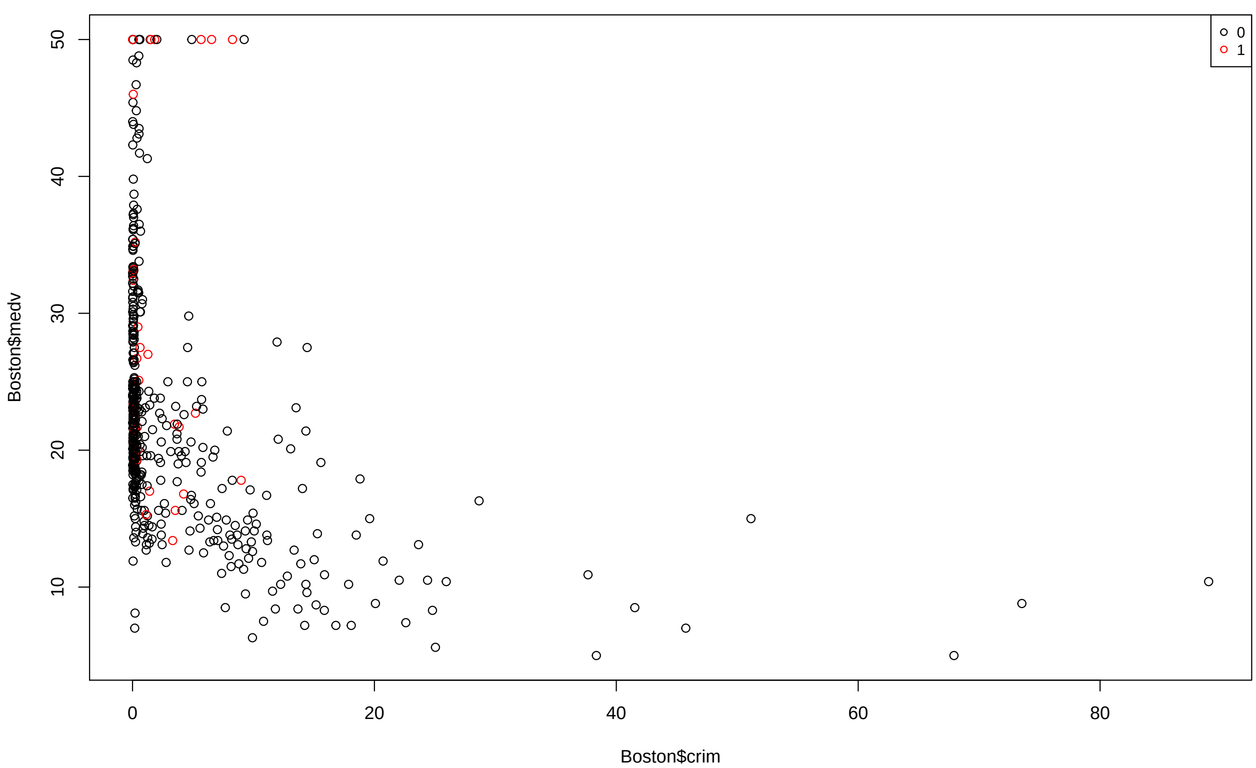
Sintaxis	Modelo	Observaciones
`y ~ x`	$y = \beta_0 + \beta_1 x$	Línea recta con intercepto implícita
`y ~ -1 + x`	$y = \beta_1 x$	Línea recta sin intercepción; es decir, un ajuste forzado (0,0)
`y ~ x + I(x^2)`	$y = \beta_0 + \beta_1 x + \beta_2 x^2$	Modelo Polinomial; `I()` permite símbolos matemáticos
`y ~ x + z`	$y = \beta_0 + \beta_1 x + \beta_2 z$	Regresión Lineal Múltiple
`y ~ x:z`	$y = \beta_0 + \beta_1 xz$	Modelo con interacción entre x y z
`y ~ x*z`	$y = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz$	Equivalente a `y~x+z+x:z`

6.3.1. Ejemplo: Datos de precios de viviendas en Boston

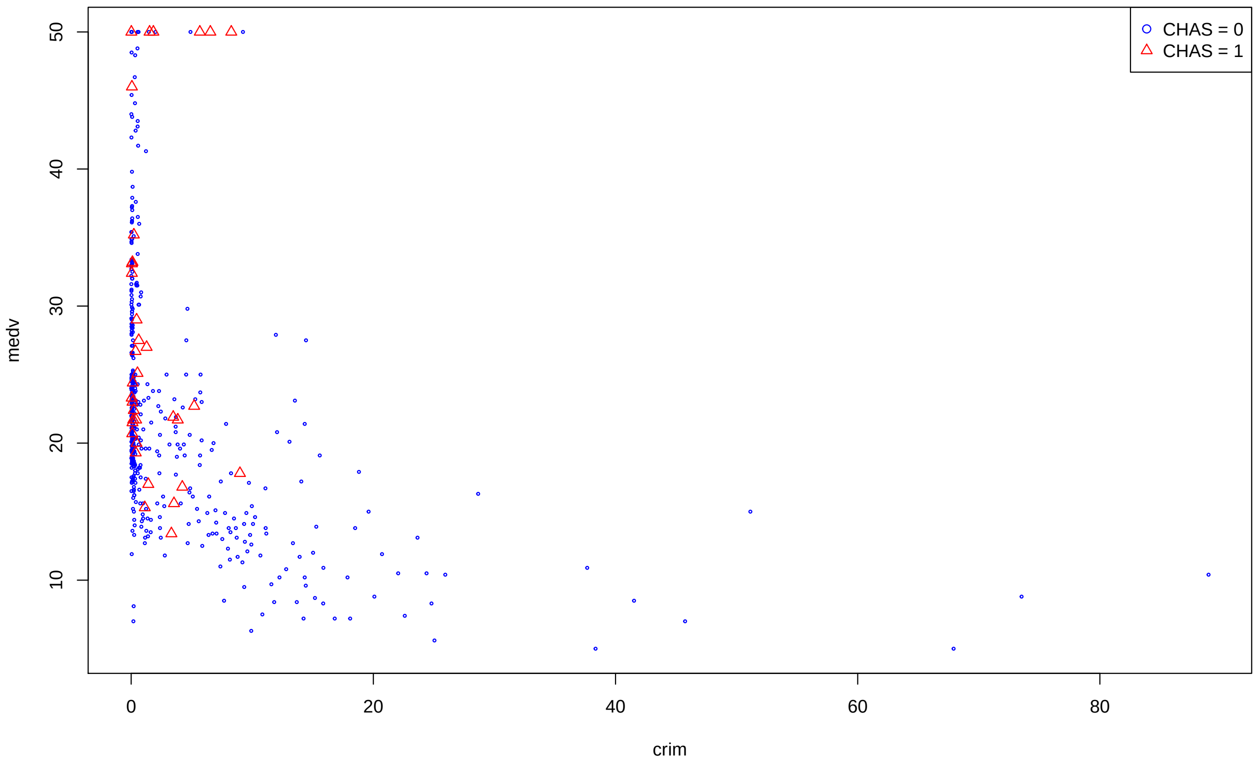
La librería `MASS` contiene el conjunto de datos de Boston, que registra `medv` (valor mediano de la casa) de 506 vecindarios alrededor de Boston. Trataremos de predecir el medv usando 13 predictores tales como `rm` (número promedio de habitaciones por casa), `age` (edad promedio de las casas), y `lstat` (porcentaje de hogares con bajo estatus socioeconómico).

```
library(MASS)
data("Boston")
?Boston
```

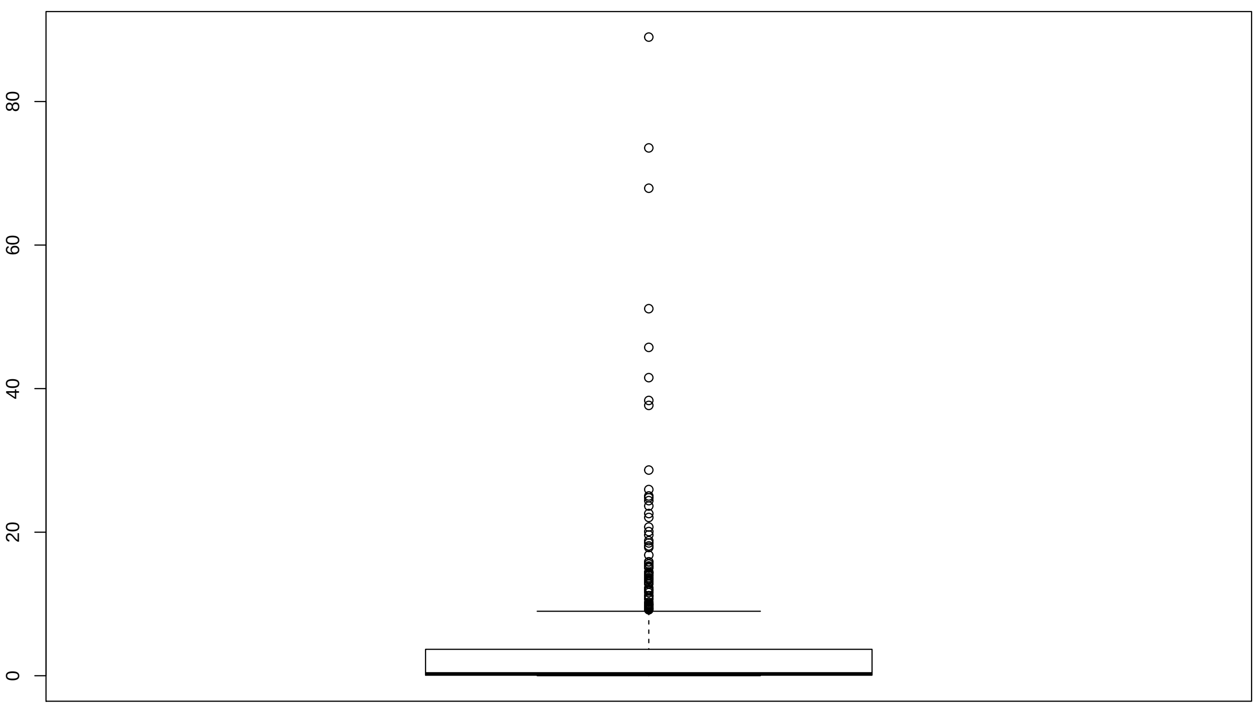
```
# Gráficos
plot(Boston$crim,Boston$medv,col=1+Boston$chas)
legend('topright', legend = levels(factor(Boston$chas)), col = 1:2, cex = 0.8, pch = 1)
```



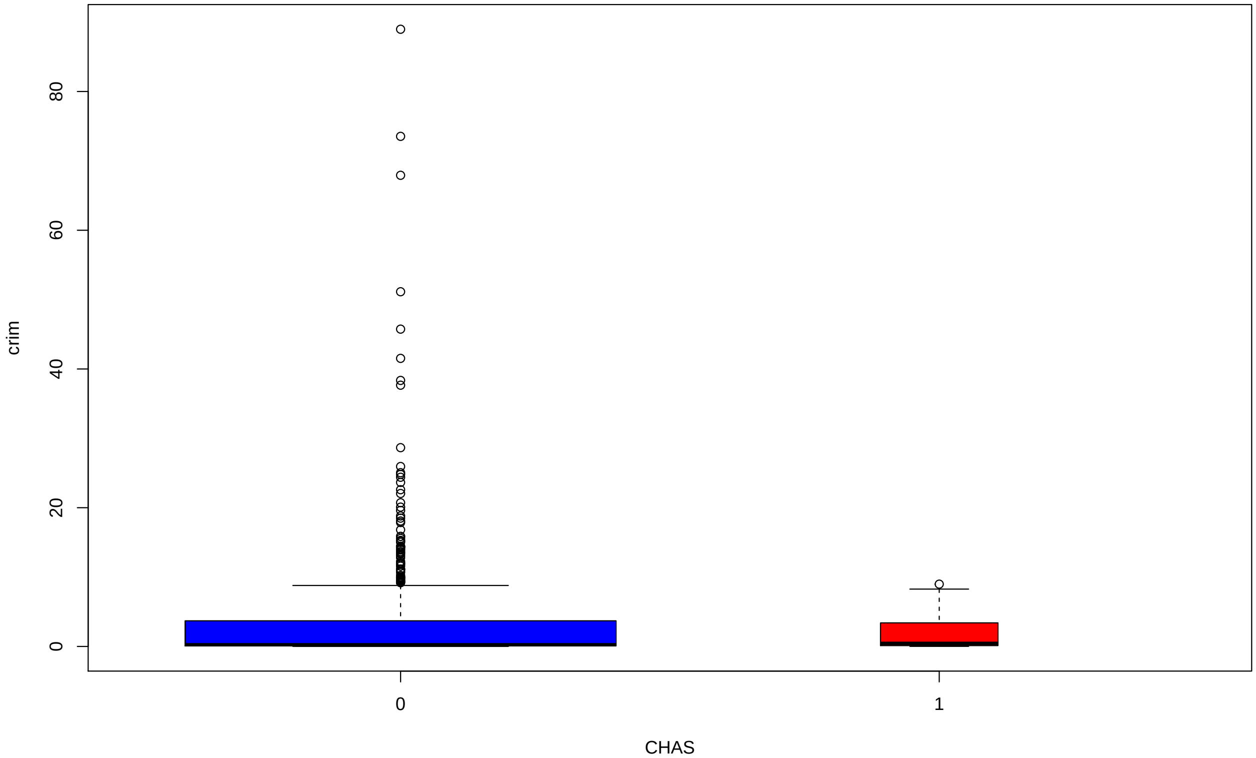

```
plot(Boston[Boston$chas==0,c("crim","medv")],xlim=range(Boston$crim),ylim=range(Boston$medv),col="blue"
points(Boston[Boston$chas==1,c("crim","medv")],col="red",pch=2)
legend("topright",c("CHAS = 0", "CHAS = 1"), col=c(4,2),pch=c(1,2))
```



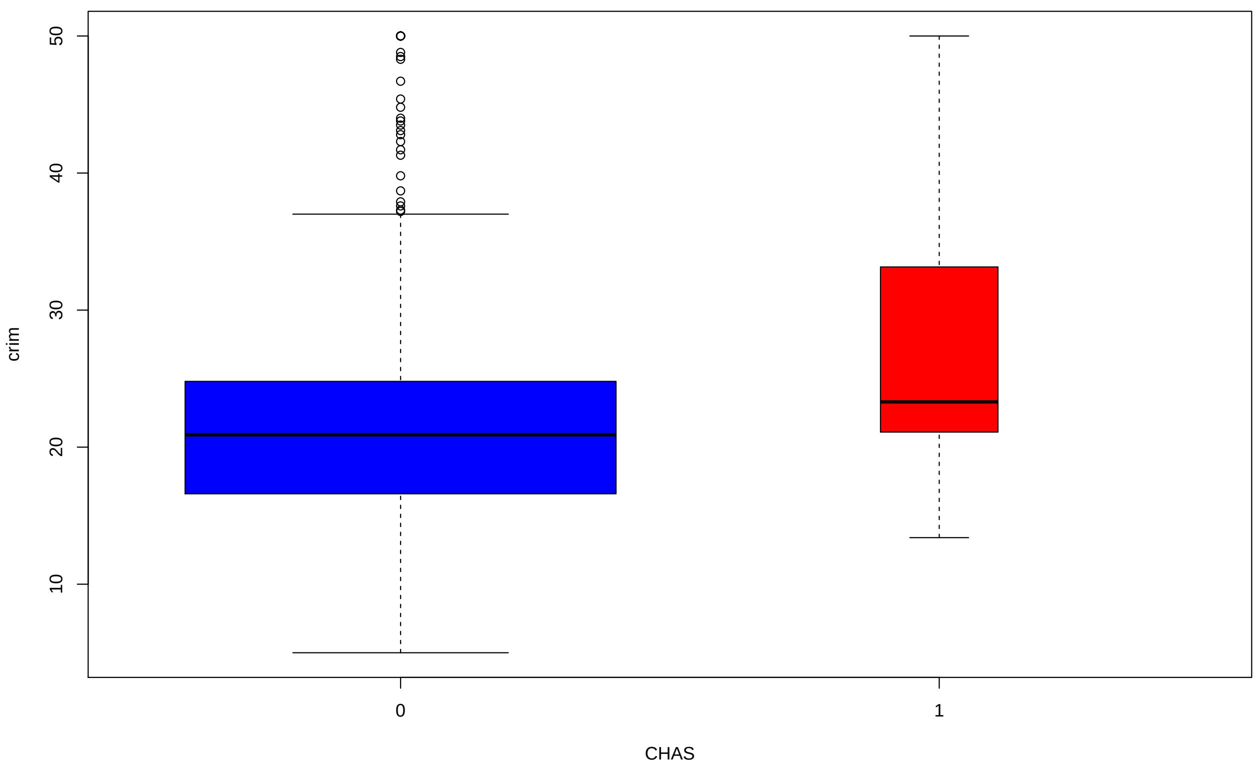

```r
attach(Boston)

boxplot(crim)
```



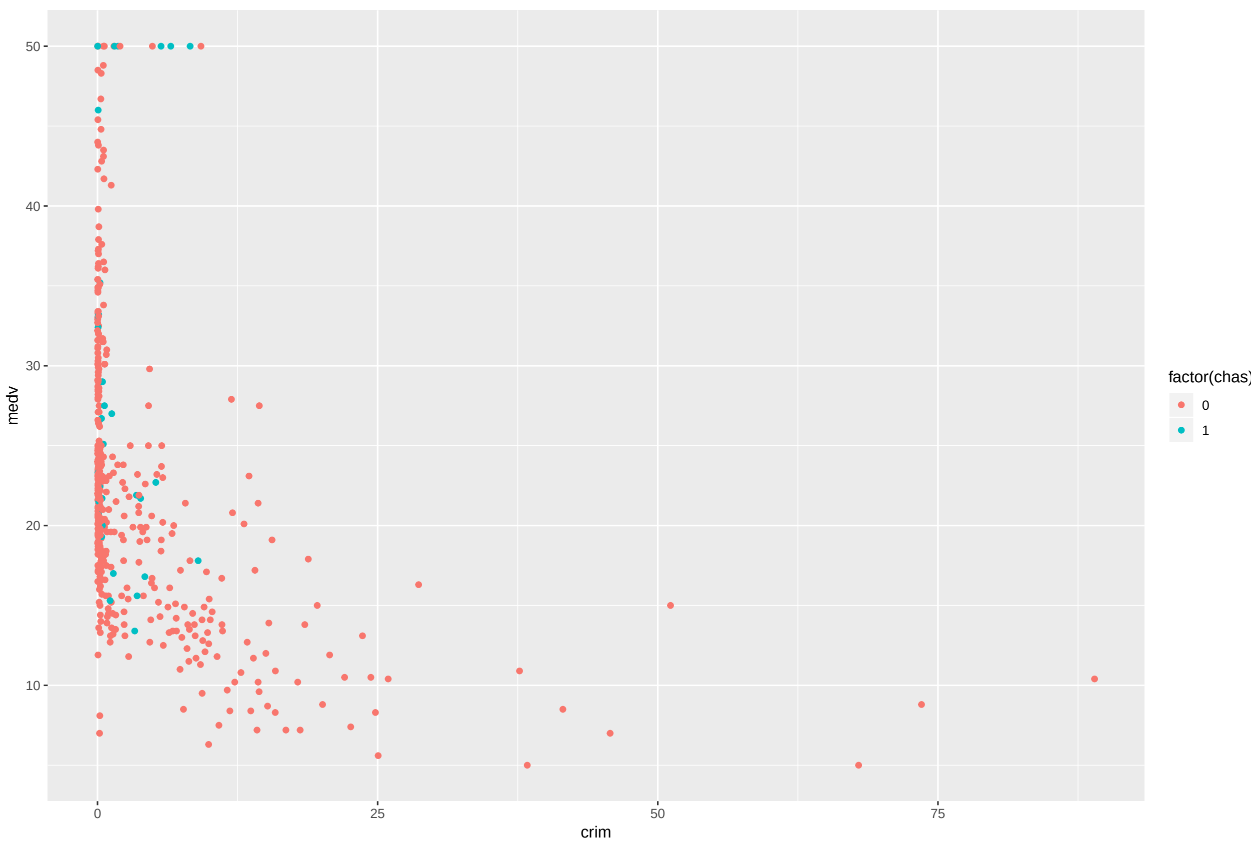

```r
boxplot(crim ~ factor(chas), data = Boston,xlab="CHAS",ylab="crim",col=c(4,2),varwidth=TRUE)
```



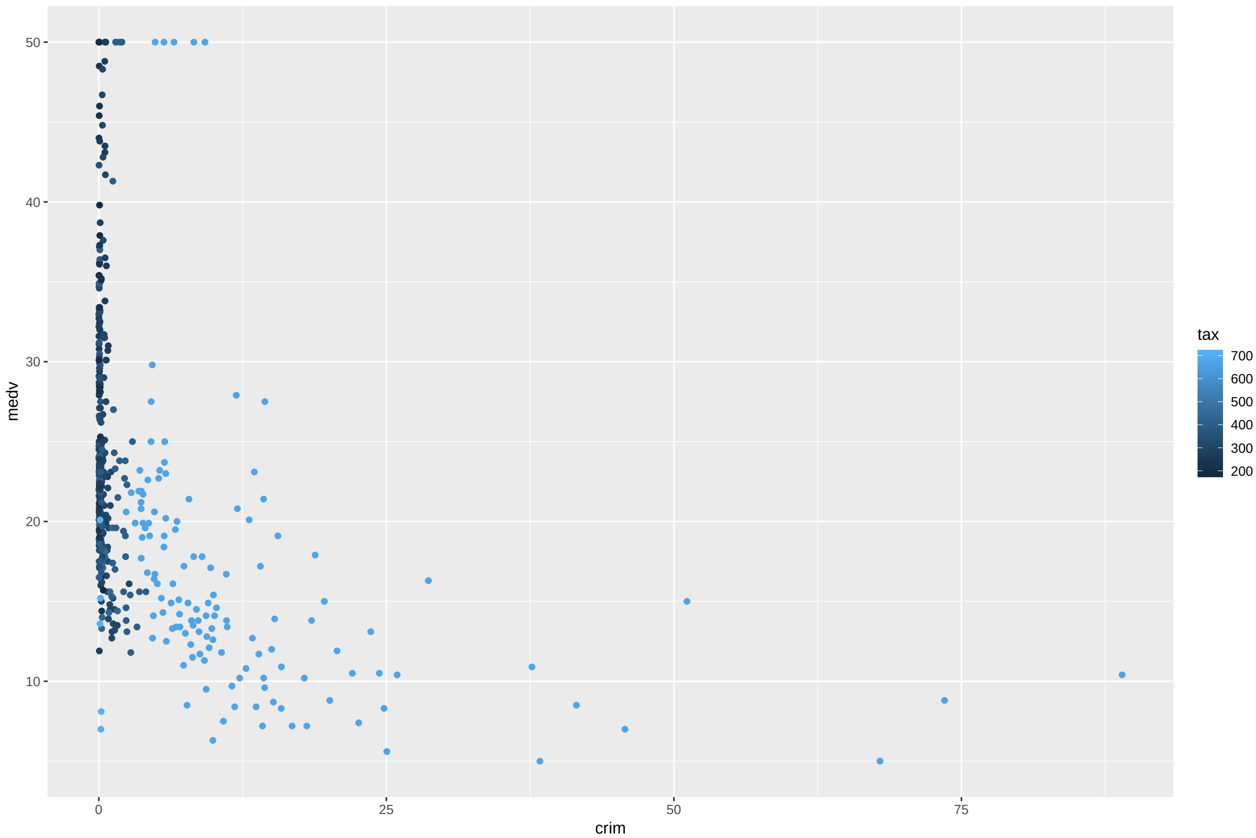

```
boxplot(medv ~ factor(chas), data = Boston,xlab="CHAS",ylab="crim",col=c(4,2),varwidth=TRUE)
```



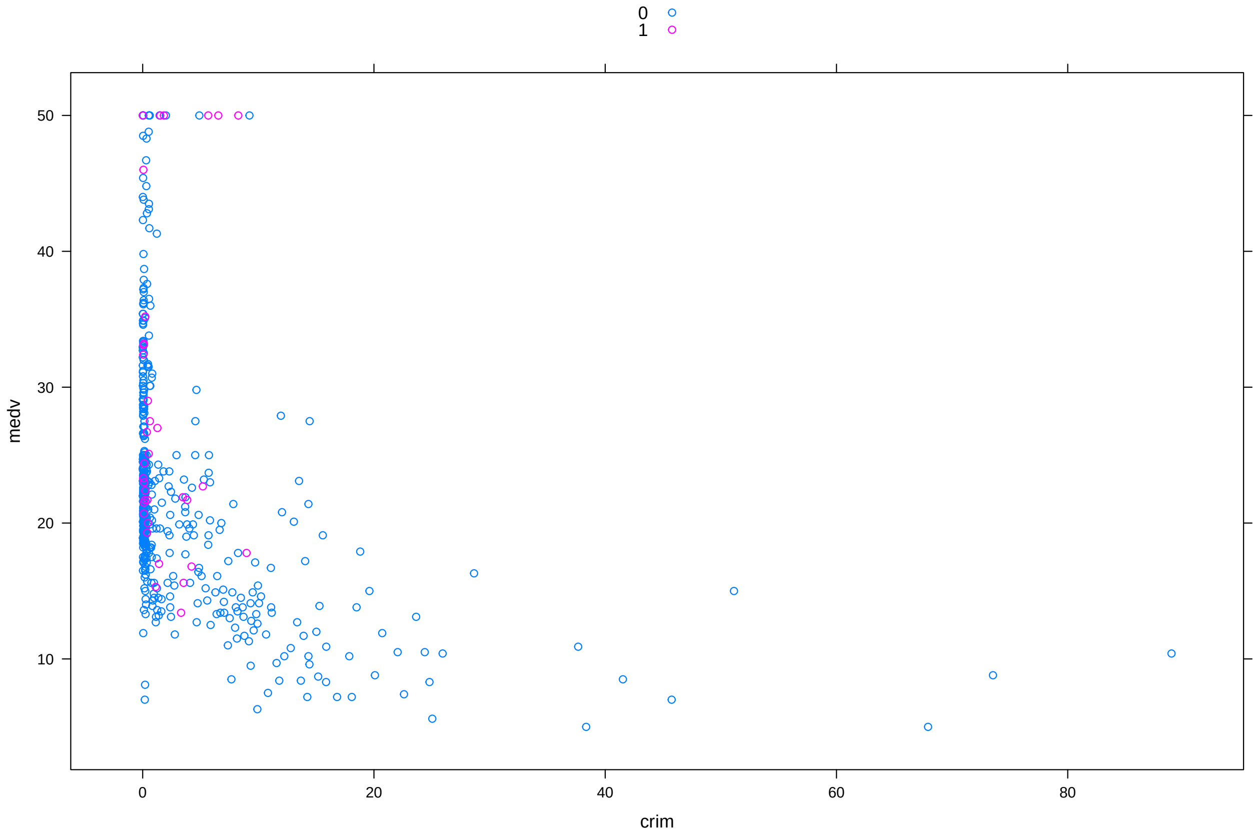

```
library(ggplot2)

qplot(crim,medv,data=Boston, colour=factor(chas))
```



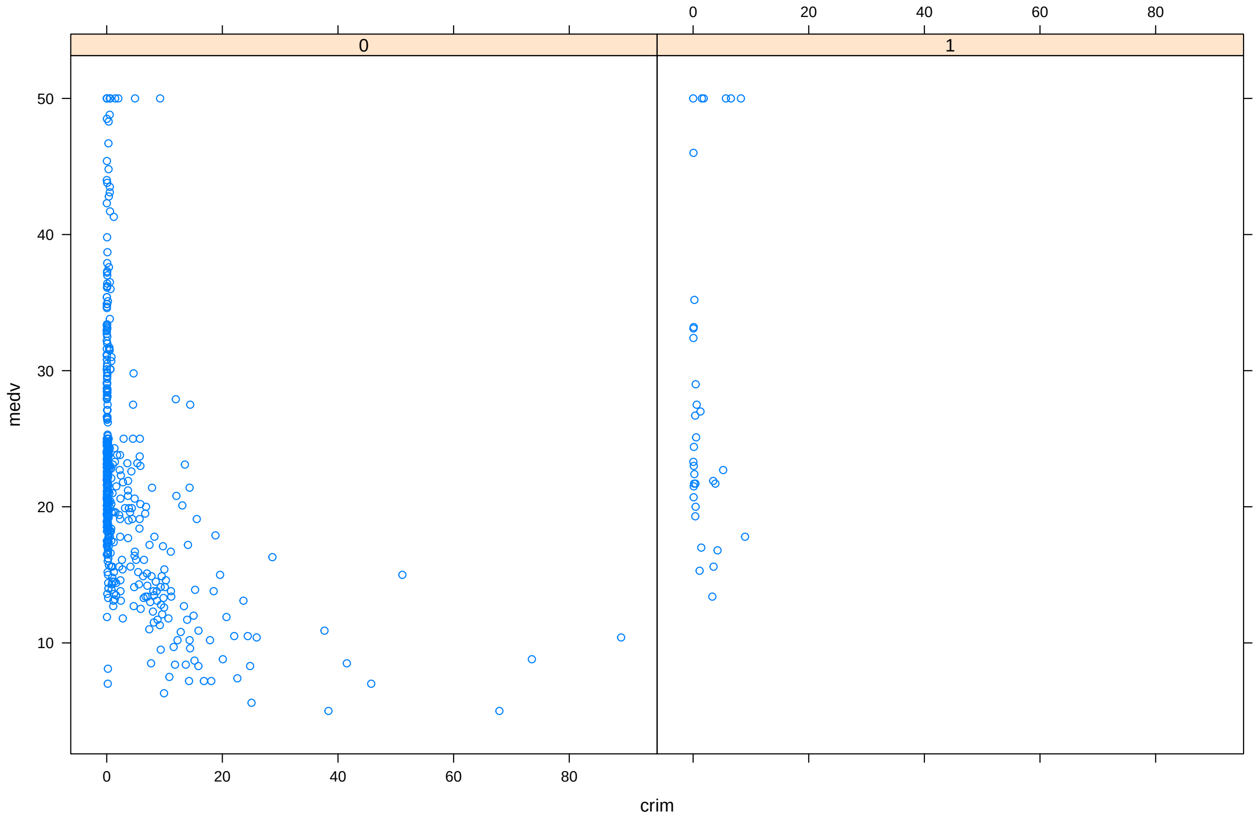

```
qplot(crim,medv,data=Boston, colour=tax)
```




```
library(lattice)
xyplot(medv~crim,groups=factor(chas),auto.key = TRUE)
```




```
xyplot(medv~crim|factor(chas),auto.key = TRUE)
```




```r
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```r
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Comenzaremos usando la función `lm()` para encajar un modelo de regresión lineal simple, con `medv` como respuesta y `lstat` como predictor. La sintaxis básica de `lm()` es `lm(y~x,data)`, donde `y` es la respuesta, `x` es el predictor, y los datos son el conjunto de datos en el que se guardan estas dos variables.

```
lm.fit <- lm(medv ~ lstat, data=Boston)
```

Si escribimos `lm.fit`, se obtiene información básica sobre el modelo. Para información más detallada, usamos `summary(lm.fit)`. Esto nos da valores de p y errores estándar para los coeficientes, así como la estadística R^2 y el estadístico para el modelo.

```
lm.fit
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

Podemos usar la función `names()` para averiguar qué otra información se almacena en `lm.fit`. Aunque podemos extraer estas cantidades por nombre - por ejemplo `lm.fit$coefficients` - es más seguro usar las funciones del extractor como `coef()` para acceder a ellas.

```
names(lm.fit)
```

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

```
lm.fit$coefficients
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

```
lm.fit[[1]]
```

```
## (Intercept)       lstat
```

```
## 34.5538409 -0.9500494
```

```r
coef(lm.fit)
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

Para obtener un intervalo de confianza para las estimaciones del coeficiente, podemos usar el comando `confint()`.

```r
confint(lm.fit, level = 0.95)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

Considere la posibilidad de construir un intervalo de confianza para β_1 utilizando la información proporcionada por el resumen de `lm.fit`:

```r
summary(lm.fit)$coefficients
```

```
##               Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 34.5538409 0.56262735  61.41515 3.743081e-236
## lstat       -0.9500494 0.03873342 -24.52790  5.081103e-88
```

La función `predict()` puede ser usada para producir intervalos de confianza e intervalos de predicción para la predicción de `medv` para un valor dado de `lstat`.

```r
CI <- predict(object = lm.fit, newdata = data.frame(lstat = c(5, 10, 15)),
              interval = "confidence")
CI
```

```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

```r
PI <- predict(object = lm.fit, newdata = data.frame(lstat = c(5, 10, 15)),
              interval = "predict")
PI
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

NOTA:

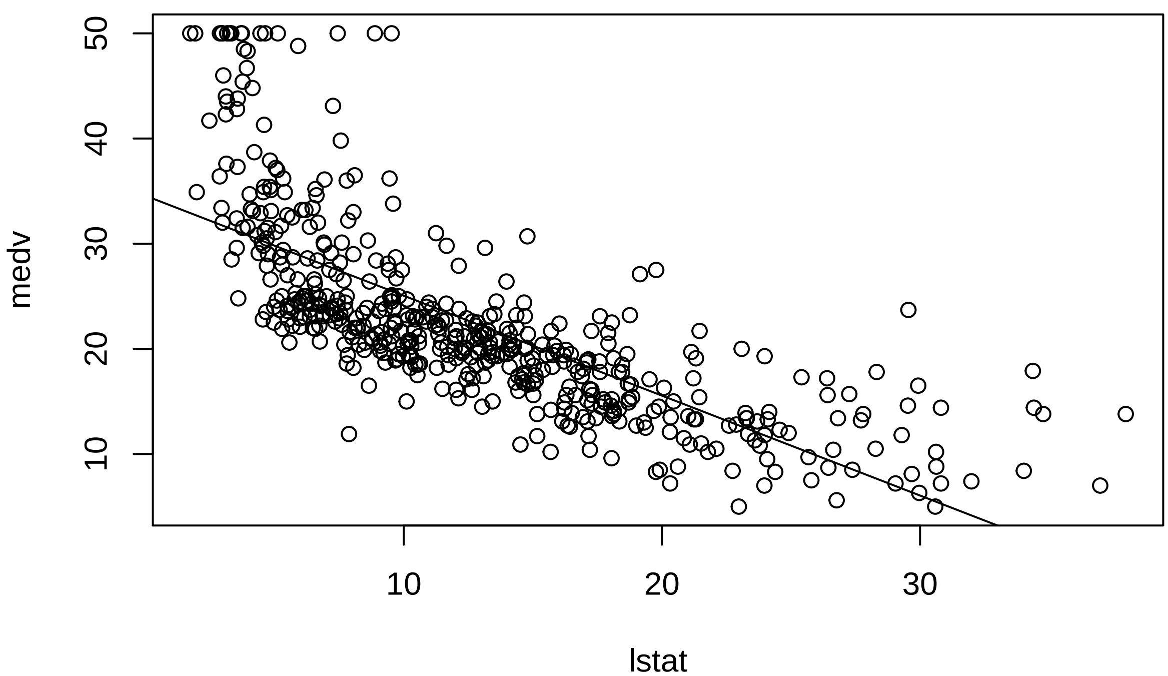
Un **intervalo de predicción** es un intervalo asociado con una variable aleatoria que aún no ha sido observada (predicción).

Un **intervalo de confianza** es un intervalo asociado a un parámetro y es un concepto de frecuentista.

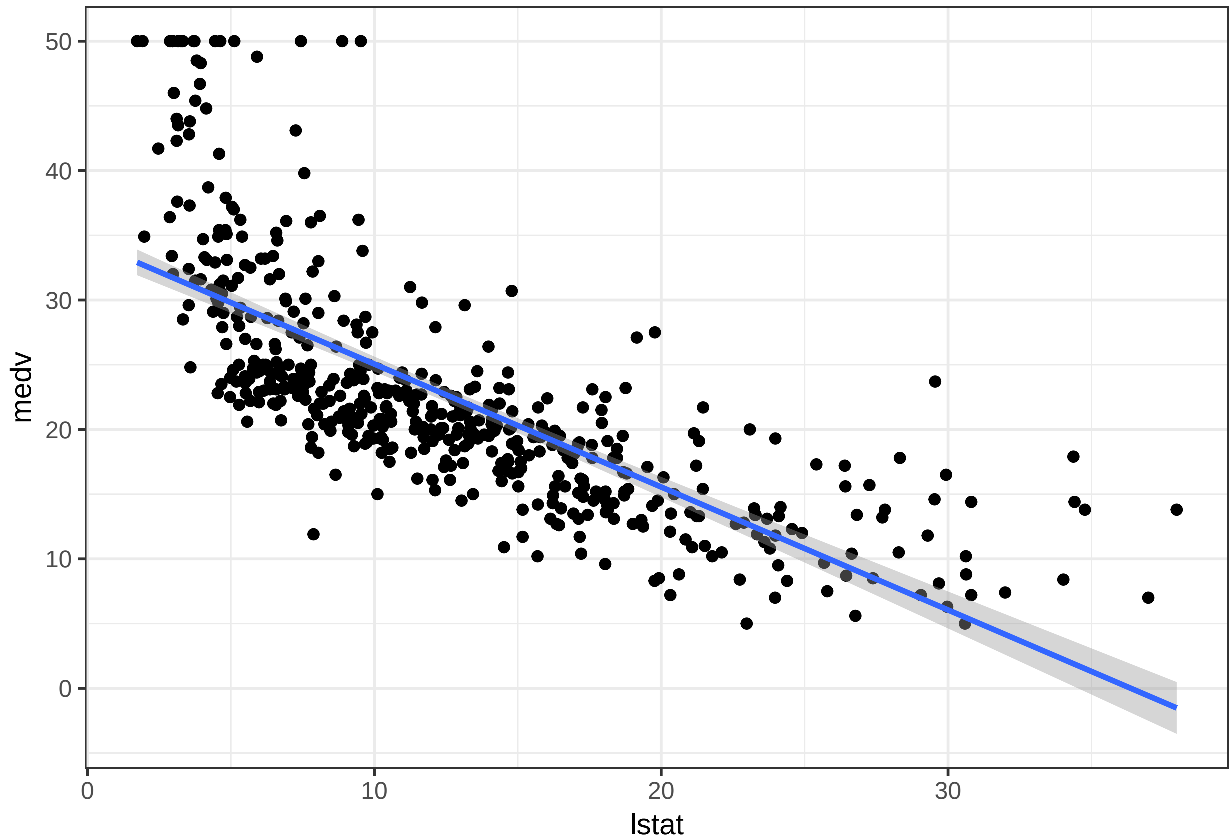
Por ejemplo, el intervalo de confianza del 95 % asociado con un valor "stat" de 10 es (24,474132, 25,6325627) y el intervalo de predicción del 95 % es (12,8276263, 37,2790683). Como se esperaba, los intervalos de confianza y predicción se centran en el mismo punto (un valor predicho de 25,0533473 para medv cuando `lstat` es igual a 10), pero estos últimos son sustancialmente más amplios.

Ahora trazaremos `medv` y `lstat` junto con la línea de regresión de los mínimos cuadrados usando las funciones `plot()` y `abline()`.

```
plot(medv ~ lstat, data = Boston)
abline(lm.fit)
```

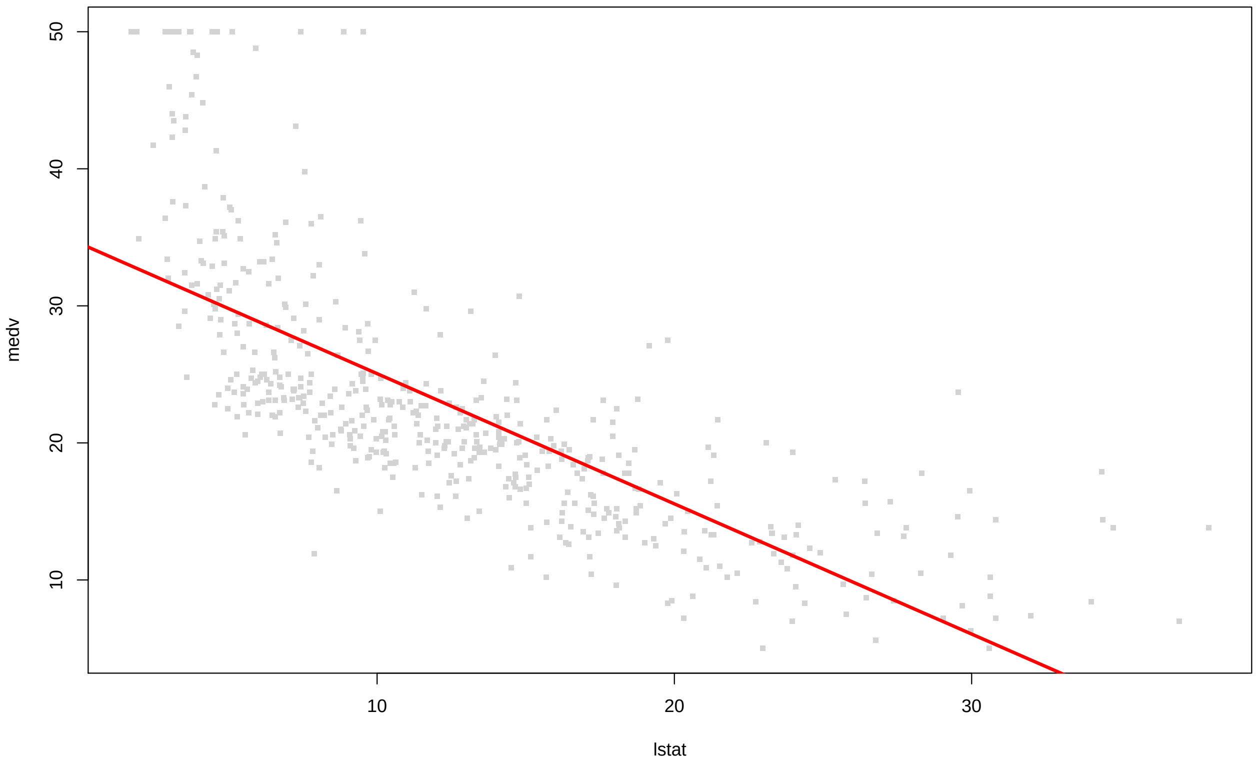



```
# Or using ggplot2
library(ggplot2)
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_bw()
```



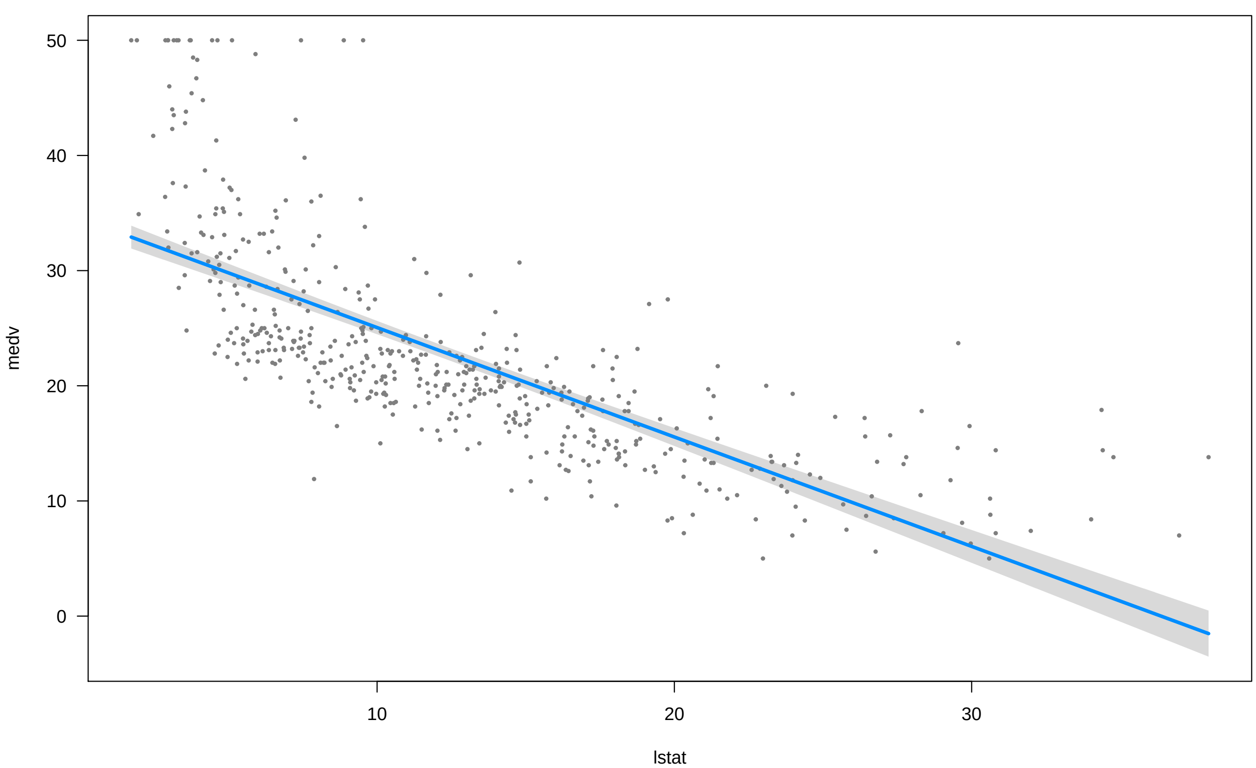
Hay alguna evidencia de no linealidad en la relación entre `lstat` y `medv`. Esta cuestión se discutirá más adelante.

```
plot(medv ~ lstat, data = Boston,pch=15,cex=.65,col="lightgrey")
abline(lm.fit, lwd = 3, col = "red")
```



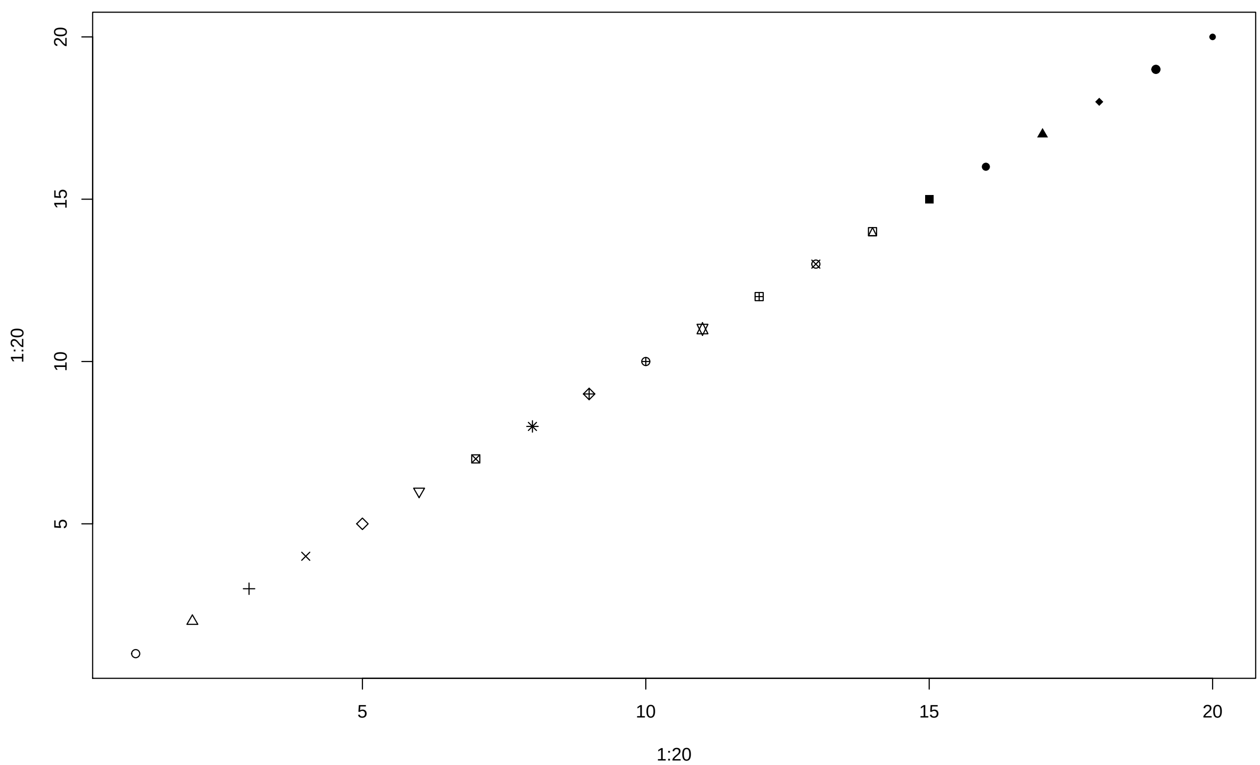
La `librería(visreg)` permite visualizar las funciones de regresión

```r
# install.packages("visreg")
library(visreg)
visreg(lm.fit)
```



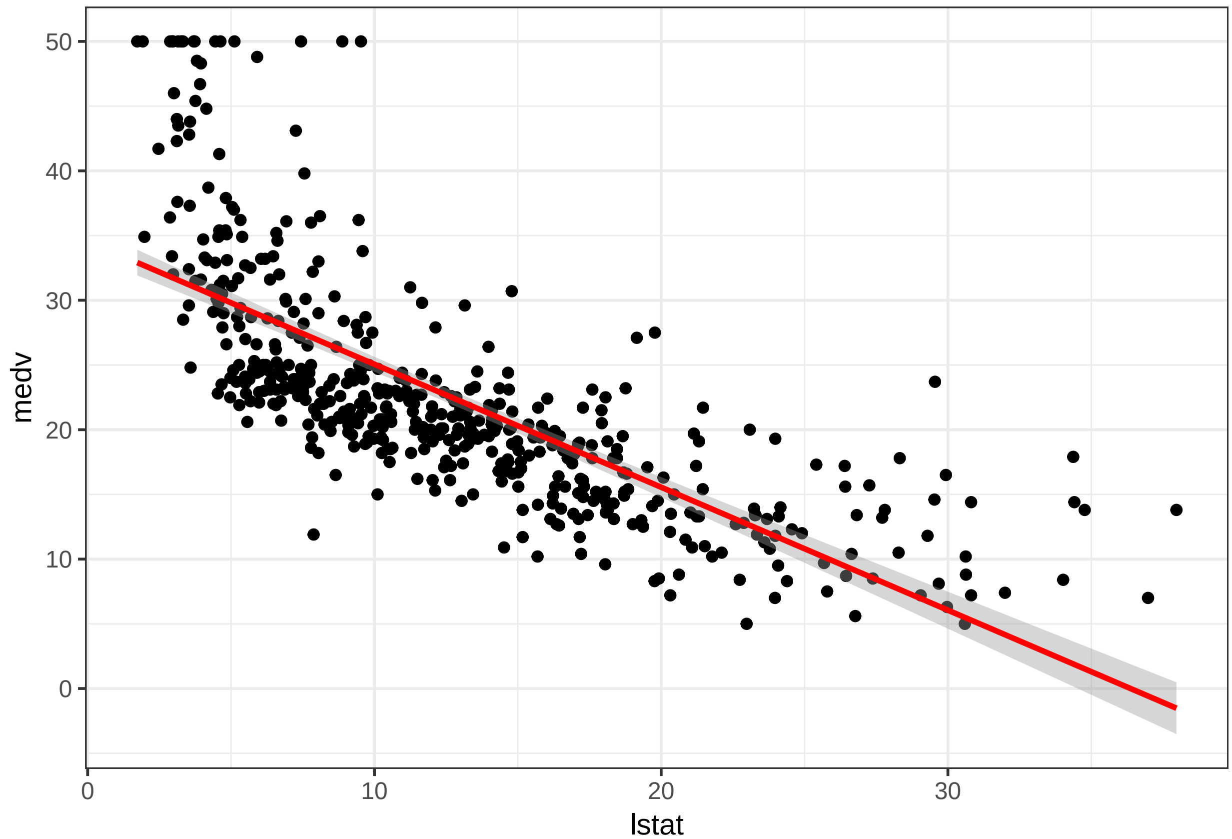
Opciones `pch`

```
plot(1:20, 1:20, pch = 1:20)
```

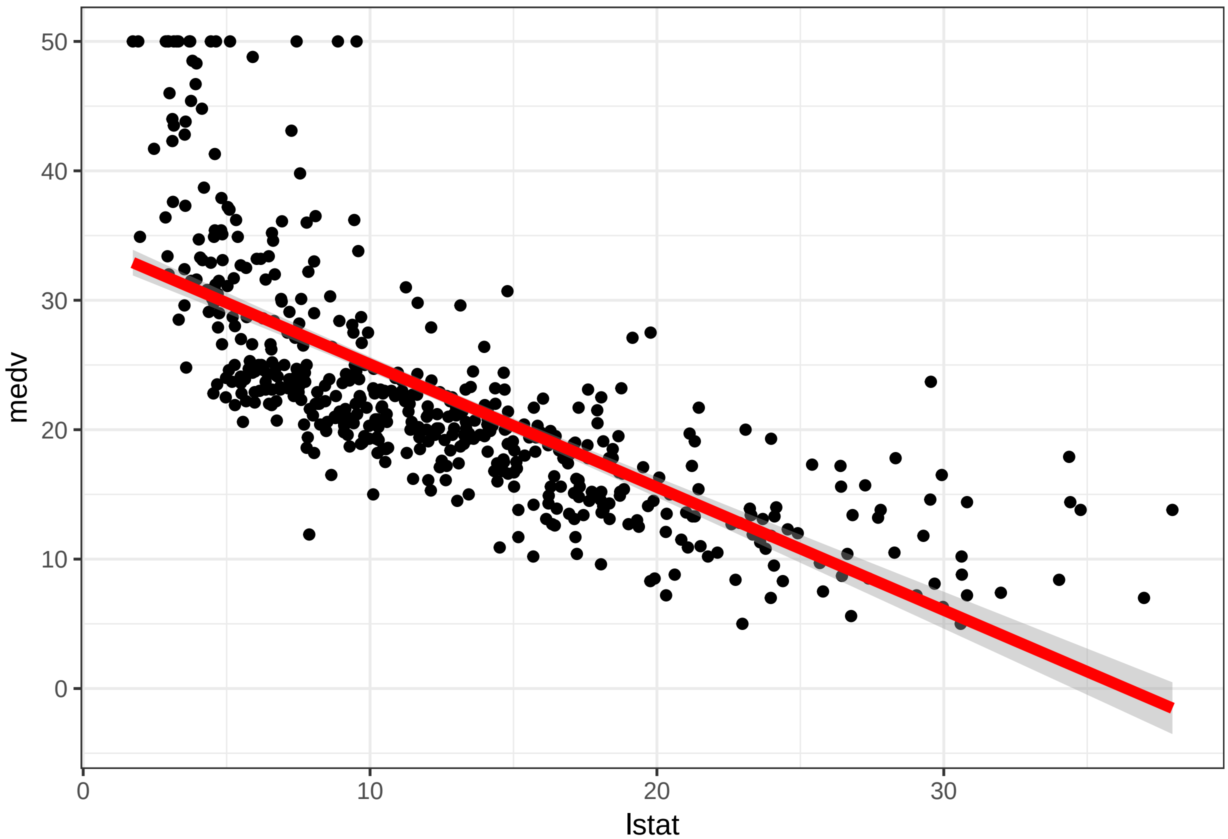


Librería ggplot2 con lm

```
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point() +
  geom_smooth(method = "lm", color = "red") +
  theme_bw()
```

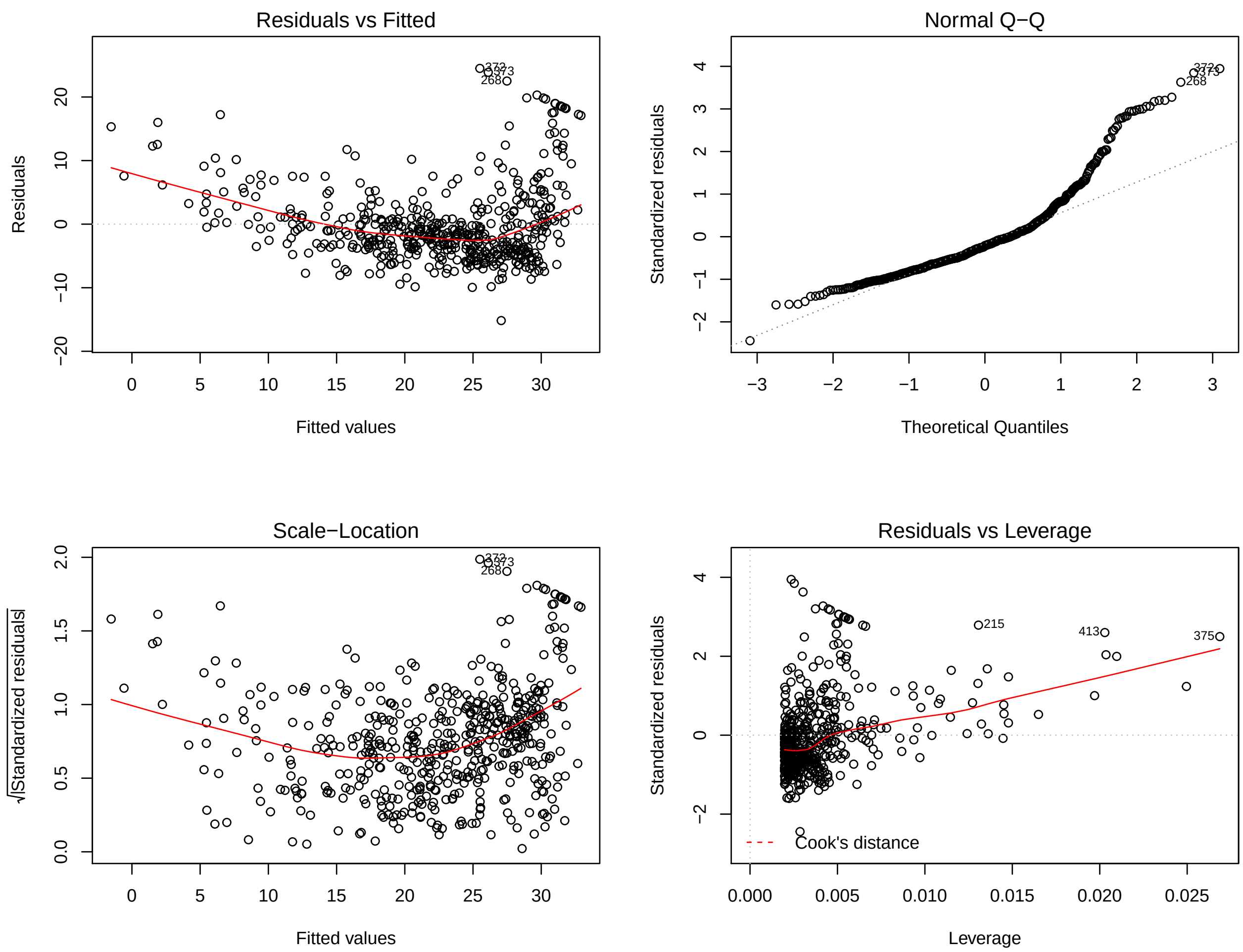



```
# thicker line
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point() +
  geom_smooth(method = "lm", color = "red", size = 2) +
  theme_bw()
```



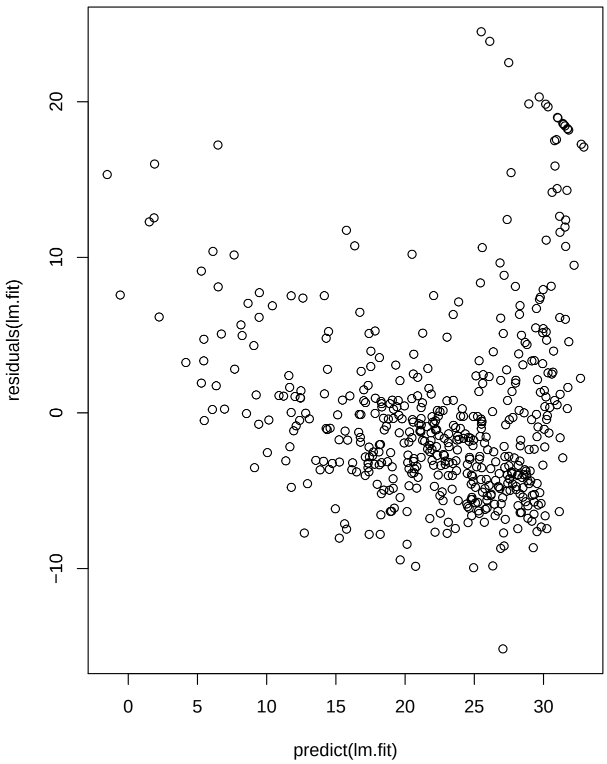
A continuación examinamos algunas gráficos de diagnóstico. Se producen automáticamente cuatro gráficas de diagnóstico aplicando la función `plot()` directamente a la salida de `lm()`. En general, este comando producirá un gráfico a la vez, y al presionar Enter se generará el siguiente gráfico. Sin embargo, a menudo es conveniente ver las cuatro parcelas juntas. Podemos lograr esto usando la función `par()`, que le dice a `R` que divida la pantalla en paneles separados para que se puedan ver múltiples gráficos simultáneamente. Por ejemplo, `par(mfrow = c(2, 2))` divide la región de trazado en una cuadrícula de paneles de 2$ por 2$.

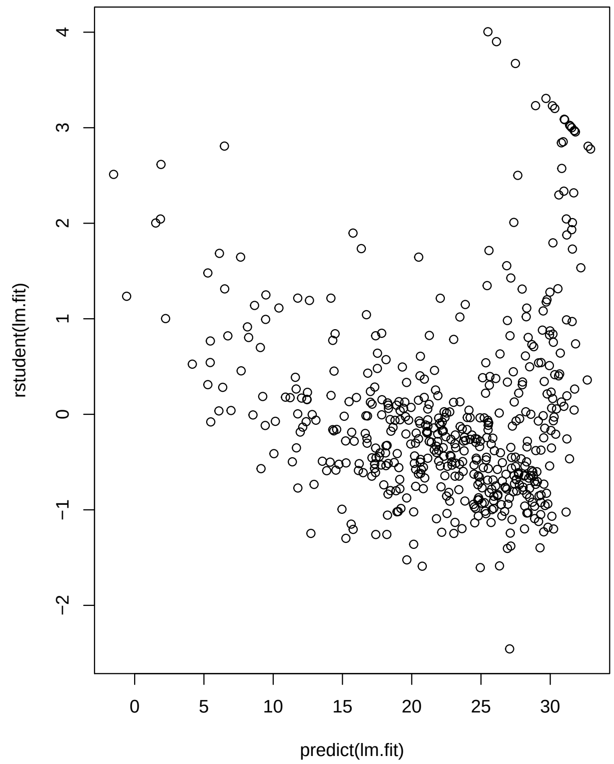
```
par(mfrow = c(2, 2))
plot(lm.fit)
```



Alternatively, we can compute the residuals from a linear regression fit using the `residuals()` function. The function `rstudent()` will return the studentized residuals, and we can use this function to plot the residuals against the fitted values.

```
par(mfrow = c(1, 2))
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
```





La biblioteca `car` tiene una función `residualPlots` para evaluar los residuos (calcula una prueba de curvatura para cada una de las parcelas añadiendo un término cuadrático y probando que la cuadrática sea cero). Ver `?residualPlots`.

```r
library(car)
```

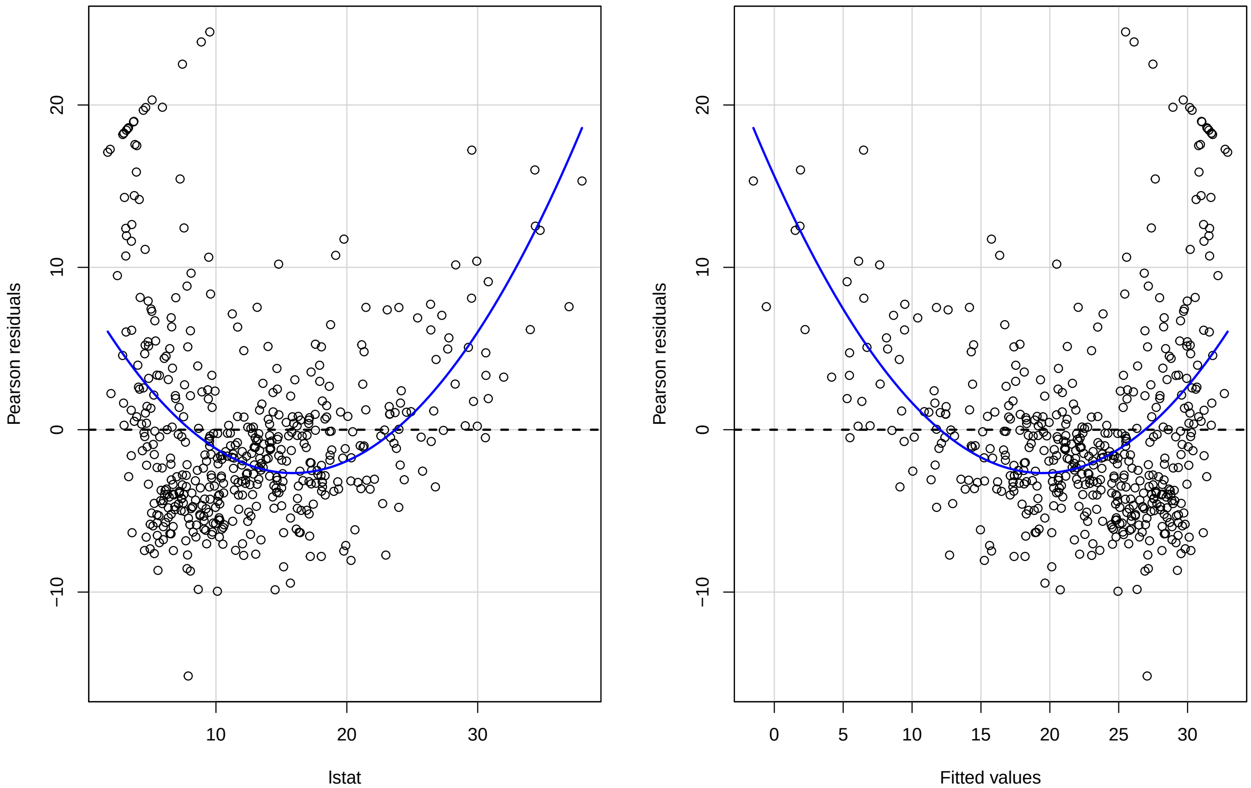
```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:DAAG':
##
##     vif
```

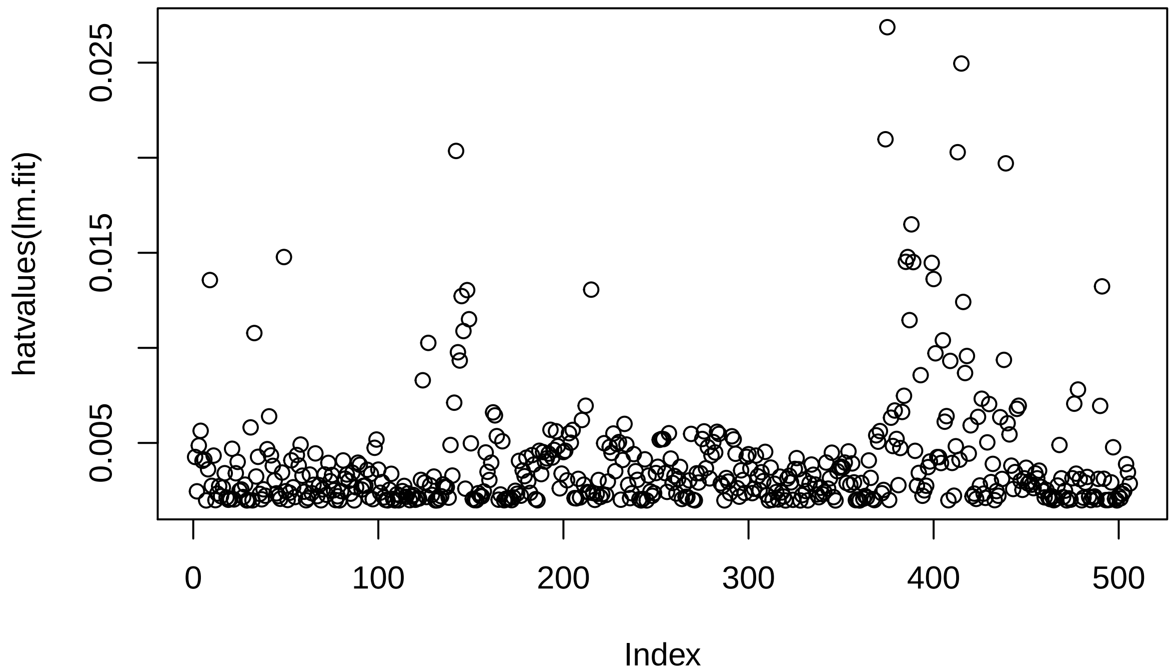
```r
residualPlots(lm.fit)
```




```
##            Test stat Pr(>|Test stat|)
## lstat         11.627        < 2.2e-16 ***
## Tukey test    11.627        < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sobre la base de las gráficos de los residuos, hay alguna evidencia de no linealidad. Las estadísticas de apalancamiento pueden ser calculadas para cualquier número de predictores usando la función `hatvalues`. La función `influenceIndexPlot` del paquete `car` crea cuatro gráficos de diagnóstico que incluyen un gráfico de los valores de sombrero.

```
plot(hatvalues(lm.fit))
```




```
which.max(hatvalues(lm.fit))
```

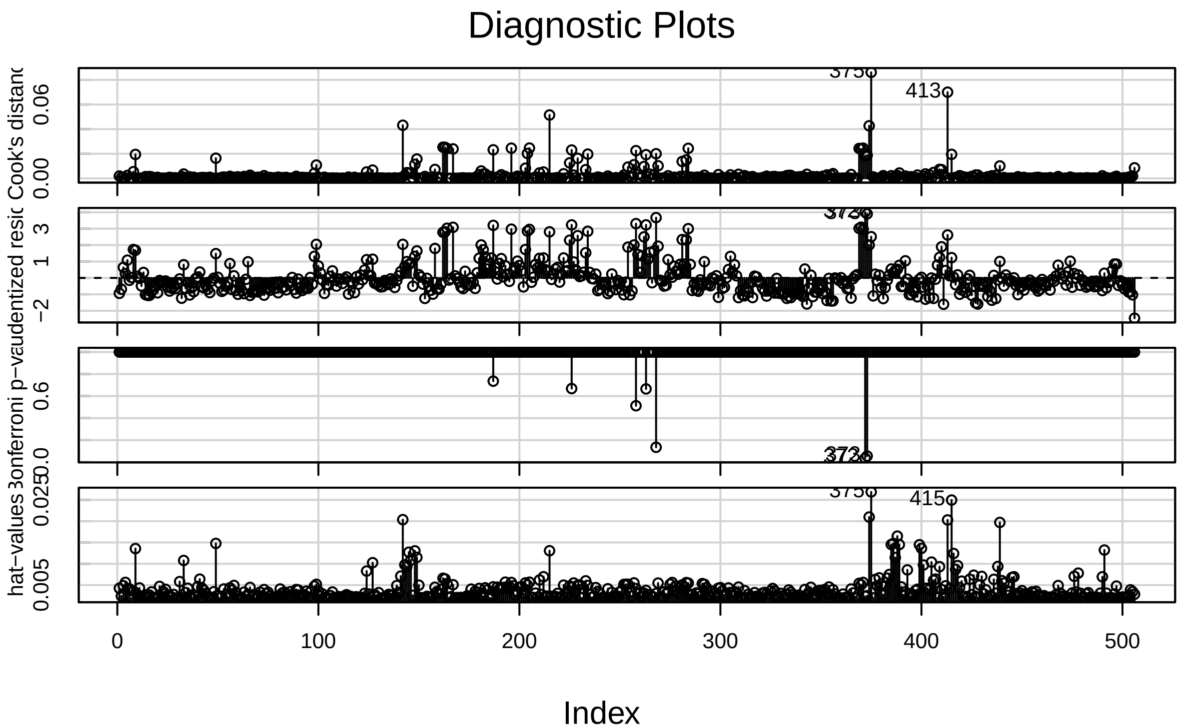
```
## 375
## 375
```

```
influenceIndexPlot(lm.fit, id.n = 5)
```

```
## Warning in plot.window(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.n" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in box(...): "id.n" is not a graphical parameter

## Warning in title(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.window(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.n" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in box(...): "id.n" is not a graphical parameter
```

```
## Warning in title(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.window(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.n" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in box(...): "id.n" is not a graphical parameter

## Warning in title(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.window(...): "id.n" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.n" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.n" is not
## a graphical parameter

## Warning in box(...): "id.n" is not a graphical parameter

## Warning in title(...): "id.n" is not a graphical parameter
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.n" is not a
## graphical parameter
```



6.3.2. Regresión lineal múltiple

Los modelos de correlación lineal múltiple requieren de las mismas condiciones que los modelos lineales simples más otras adicionales.

Para ajustar un modelo de regresión lineal múltiple usando mínimos cuadrados, utilizamos de nuevo la función `lm()`. La sintaxis `lm(y ~ x1 + x2 + x3)` se utiliza para ajustar un modelo con tres predictores, `x1`, `x2`, y `x3`. La función `summary()` produce ahora los coeficientes de regresión para todos los predictores.

```r
ls.fit <- lm(medv ~ lstat + age, data = Boston)
summary(ls.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

El conjunto de datos `Boston` contiene 13 variables, por lo que sería engorroso tener que escribirlas todas para poder realizar una regresión utilizando todos los predictores. En su lugar, podemos utilizar la siguiente abreviatura:

```r
ls.fit <- lm(medv ~ ., data = Boston)
summary(ls.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

Podemos acceder a los componentes individuales de un objeto de resumen por nombre (escriea `?summary.lm` para ver qué hay disponible). Por lo tanto `summary(lm.fit)$r.sq` nos da los R^2, y `summary(lm.fit)$sigma` nos da *hatsigma*.

Si queremos realizar una regresión usando todas las variables pero excepto una, podemos eliminarla usando `-`. Por ejemplo, en la salida de regresión anterior, `age` tiene un alto valor p. Así que tal vez queramos hacer una regresión excluyendo este predictor. La siguiente sintaxis resulta en una regresión usando todos los predictores excepto `age`.

```r
ls.fit1 <- lm(medv ~ . - age, data = Boston)
summary(ls.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

6.3.2.1. Interacciones

Es fácil incluir términos de interacción en un modelo lineal usando la función `lm()`. La sintaxis `lstat:black` indica a `R` que incluya un término de interacción entre `lstat` y `black`. La sintaxis `lstat*age` incluye simultáneamente `lstat`,`age`, y el término de interacción `lstat` $\times$`age` como predictores; es una abreviatura de `lstat + age + lstat:age`.

```r
summary(lm(medv ~ lstat*age, data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
```

```
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

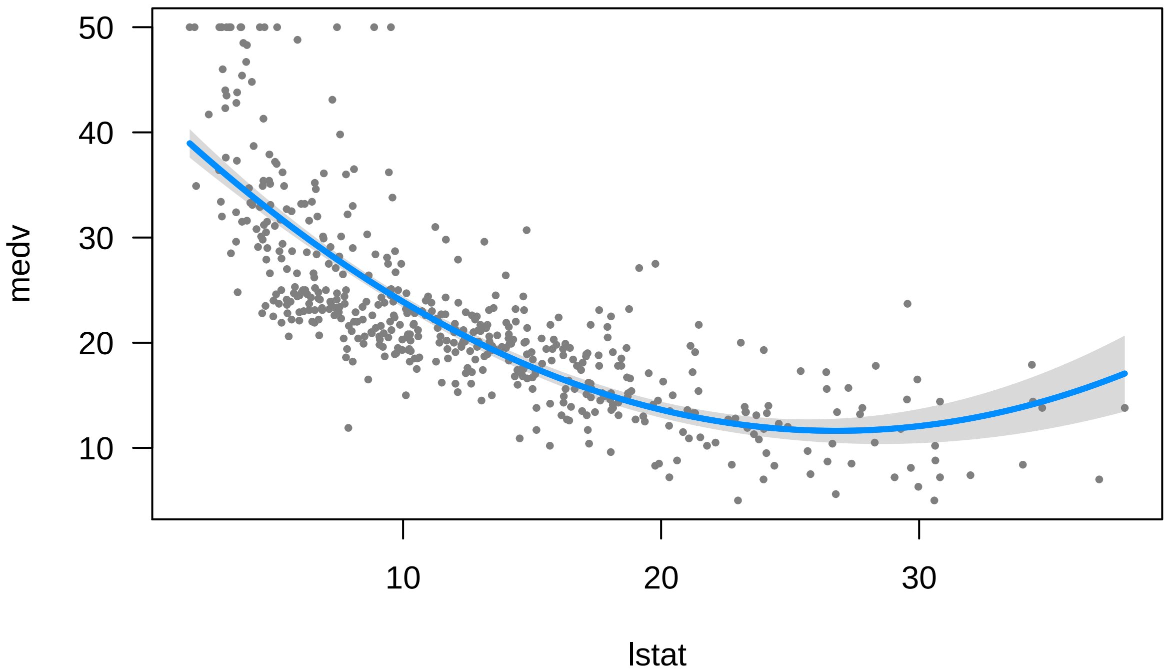
6.3.2.2. Transformaciones no lineales para los pronosticadores

La función `lm()` también puede acomodar transformaciones no lineales de los predictores. Por ejemplo, dado un predictor X podemos crear un predictor X^2 usando `I(X^2)`. La función `I()` es necesaria ya que `^` tiene un significado especial en una fórmula; envolviendo como lo hacemos permite el uso estándar en `R`, que es `I()` para elevar X a la potencia 2. Ahora realizamos una regresión de `medv` sobre `lstat` y `lstat`^{2}.

```
lm.fit2 <- lm(medv ~ lstat + I(lstat^2), data = Boston)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
# plot
visreg(lm.fit2)
```



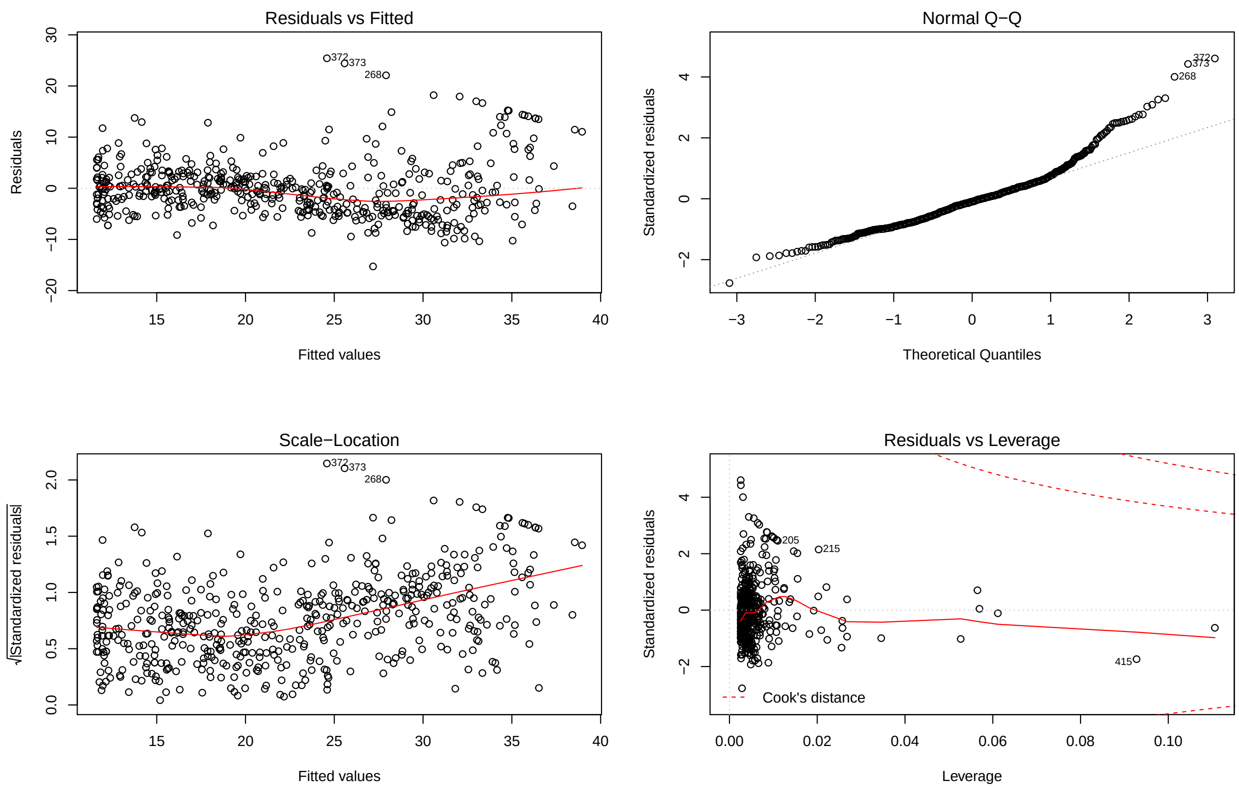
El valor p cercano a cero asociado con el término cuadrático sugiere que conduce a un modelo mejorado. Usamos la función `anova()` para cuantificar aún más hasta qué punto el ajuste cuadrático es superior al ajuste lineal.

```
anova(lm.fit, lm.fit2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + I(lstat^2)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 19472
## 2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aquí el Modelo 1 (`lm.fit`) representa el submodelo lineal que contiene sólo un predictor, `lstat`, mientras que el Modelo 2 (`lm.fit2`) corresponde al modelo cuadrático más grande que tiene dos predictores, `lstat` y $\texttt{lstat}^2$. La función `anova()` realiza una prueba de hipótesis comparando los dos modelos. La hipótesis nula es que los dos modelos se ajustan a los datos igualmente bien, y la hipótesis alternativa es que el modelo completo es superior. Aquí la estadística F es `135.1998221` y el valor p asociado es virtualmente cero. Esto proporciona una evidencia muy clara de que el modelo que contiene los predictores `lstat` y $\texttt{lstat}^2$ es muy superior al modelo que sólo contiene el predictor `lstat`. Esto no es sorprendente, ya que antes vimos evidencia de no linealidad en la relación entre `medv` y `lstat`. Si escribimos

```
par(mfrow = c(2,2))
plot(lm.fit2)
```




```r
par(mfrow = c(1, 1))
```

entonces vemos que cuando el término `lstat`^{2} se incluye en el modelo, hay poco patrón discernible en los residuos.

Para crear un ajuste cúbico, podemos incluir un predictor de la forma `I(X^3)`. Sin embargo, este enfoque puede empezar a ser engorroso para los polinomios de orden superior. Un mejor enfoque implica usar la función `poly()` para crear el polinomio dentro de `lm()`. Por ejemplo, el siguiente comando produce un ajuste polinómico de quinto orden:

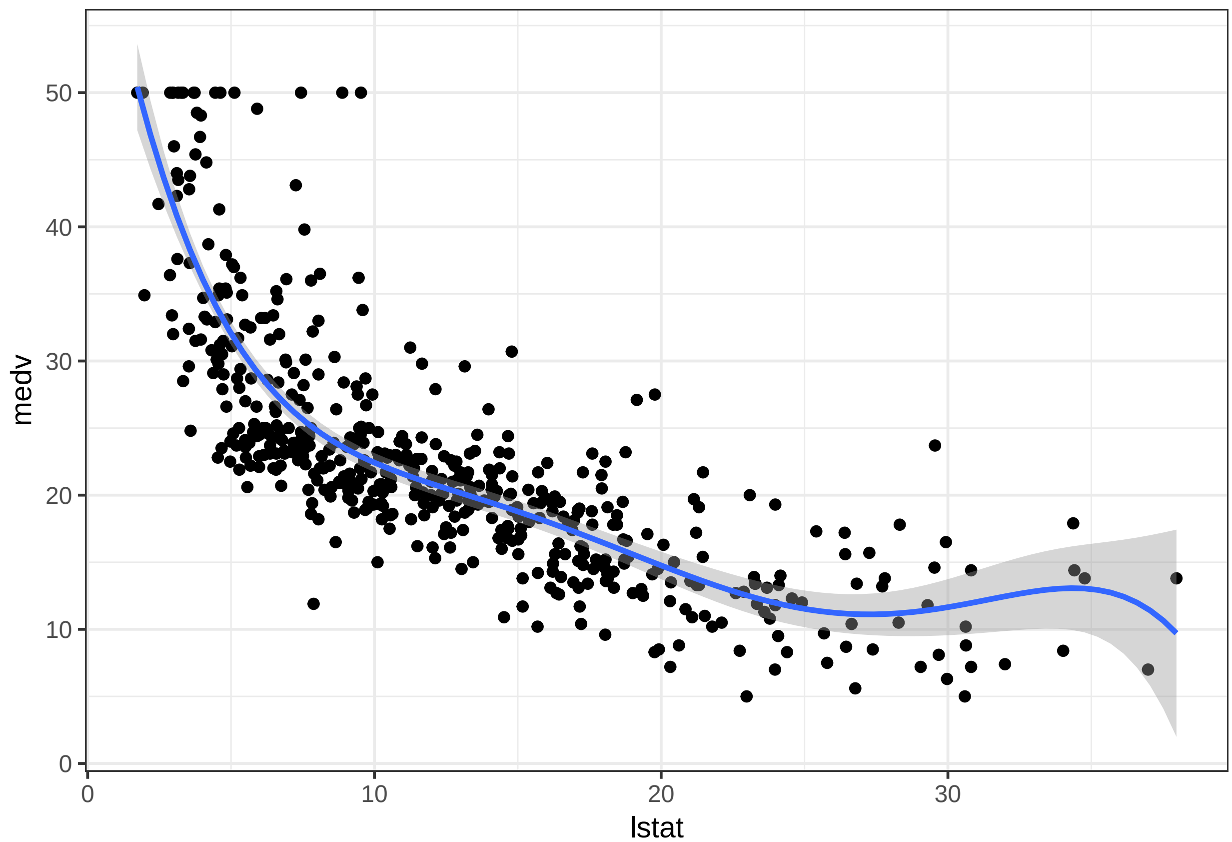
```r
lm.fit5 <- lm(medv ~ poly(lstat, 5), data = Boston)
summary(lm.fit5)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

Esto sugiere que incluir términos polinómicos adicionales, hasta el quinto orden, conduce a una mejora en el ajuste del modelo. Sin embargo, una investigación adicional de los datos revela que ningún término polinómico más allá del quinto orden tiene valores p significativos en un ajuste de regresión.

```r
library(ggplot2)
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point() +
  theme_bw() +
  stat_smooth(method = "lm", formula = y ~ poly(x, 5))
```



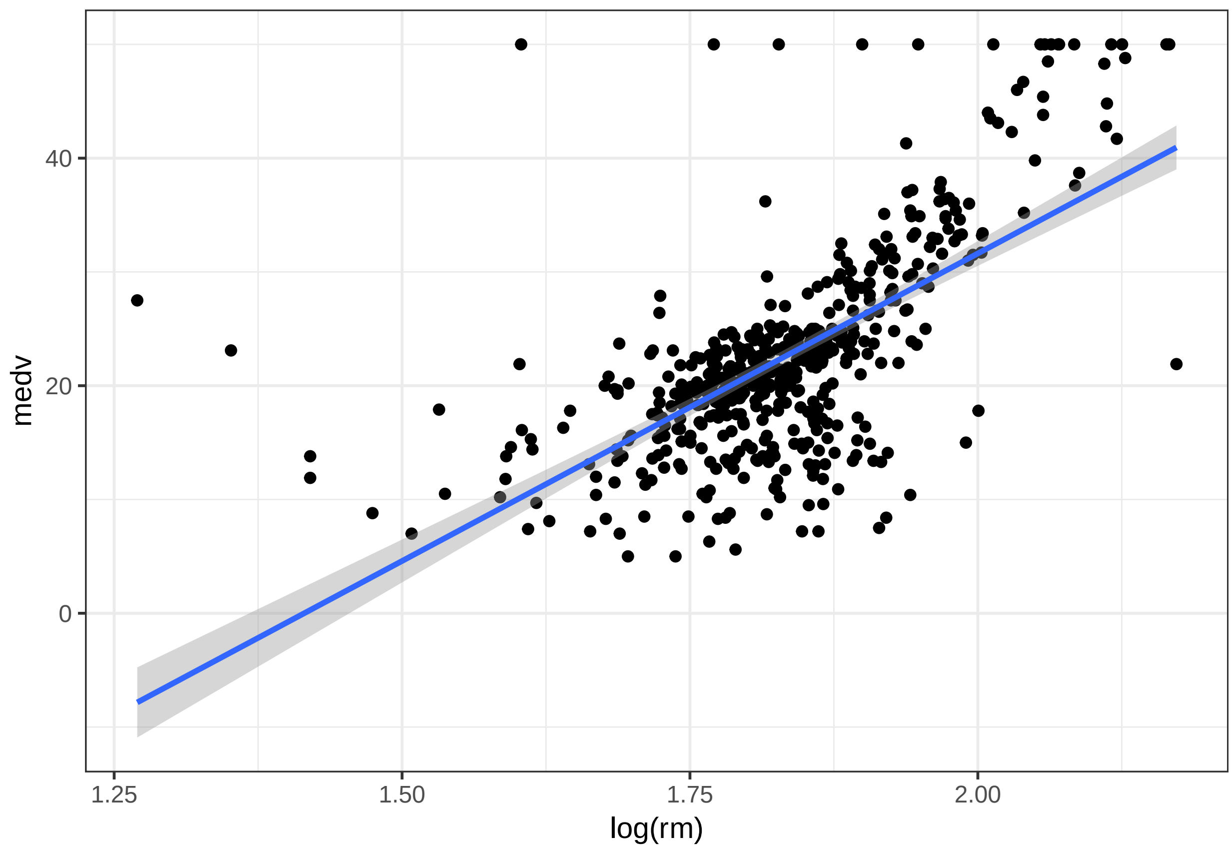
Por supuesto, no estamos restringidos a usar transformaciones polinómicas de los predictores. Aquí probamos una transformación logarítmica.

```r
summary(lm(medv ~ log(rm), data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ log(rm), data = Boston)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.487  -2.875  -0.104   2.837  39.816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -76.488      5.028  -15.21   <2e-16 ***
## log(rm)       54.055      2.739   19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.915 on 504 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4347
## F-statistic: 389.3 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
ggplot(data = Boston, aes(x = log(rm), y = medv)) +
  geom_point() +
  theme_bw() +
  stat_smooth(method = "lm")
```



6.3.3. Ejemplo:

Se desea conocer el gasto de alimentación mensual de una familia en función del ingreso mensual, el tamaño de la familia y el número de hijos en la universidad.

```r
gastos <- c(1000, 580, 520, 500, 600, 550, 400)
ingresos <- c(50000, 2500, 2000, 1900, 3000, 4000, 2000)
tamaño <- c(7, 4, 3, 3, 6, 5, 2)
```

```
hijosU <- c(3,1,1,0,1,2,0)
datos <- data.frame(gastos,ingresos,tamaño,hijosU)
datos
```

```
##   gastos ingresos tamaño hijosU
## 1   1000    50000      7      3
## 2    580     2500      4      1
## 3    520     2000      3      1
## 4    500     1900      3      0
## 5    600     3000      6      1
## 6    550     4000      5      2
## 7    400     2000      2      0
```

Vamos a ajustar un nuevo modelo de regresión lineal múltiple que explique el gasto de alimentación mensual en función de las variables descritas anteriormente.

```
reg_lin_mul <- lm(gastos ~ ingresos + tamaño + hijosU, data=datos)
summary(reg_lin_mul)
```

```
##
## Call:
## lm(formula = gastos ~ ingresos + tamaño + hijosU, data = datos)
##
## Residuals:
##       1       2       3       4       5       6       7
##   1.216  48.164  29.125  15.209 -10.134 -35.402 -48.178
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.590e+02  6.291e+01   5.706   0.0107 *
## ingresos    7.247e-03  1.802e-03   4.021   0.0276 *
## tamaño      3.734e+01  2.046e+01   1.825   0.1655
## hijosU      5.359e+00  4.061e+01   0.132   0.9034
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 48.57 on 3 degrees of freedom
## Multiple R-squared:  0.9677, Adjusted R-squared:  0.9353
## F-statistic: 29.93 on 3 and 3 DF,  p-value: 0.009772
```

En `Coefficients` se muestran los parámetros estimados de regresión $\beta_0 = 359.009$. $\beta_{\text{ingresos}} = 0.007$, $\beta_{\text{tamaño}} = 37.338$ y $\beta_{\text{hijosU}} = 5.359$.

```
round(coef(reg_lin_mul),3)
```

```
## (Intercept)    ingresos      tamaño      hijosU
##     359.009       0.007      37.338       5.359
```

- Los gastos estimados son iguales a 359.009 euros (constantes las demás variables)
- Por cada mil euros de ingresos, los gastos aumentan en 0.007, supuesto que permanecen constantes las otras variables.
- Por cada aumento del tamaño de la familia en un familiar, los gastos estimados aumentan en 37.338, suponiendo que se mantienen constantes las otras variables.
- Por cada aumento del número de hijos estudiando en la Universidad, los gastos estimados aumentan en 5.359, suponiendo que se mantienen constantes las otras variables.

Tanto la interpretación como la comprobación de la significación de los parámetros se realizan de forma similar al caso en que se cuenta con una única variable independiente. Igualmente, la validación se lleva a cabo del mismo modo que para la regresión lineal simple.

El p-valor asociado al contraste (0.009772) es menor que $\alpha = 0{,}05$, por lo que rechazamos la hipótesis nula. Esto implica que al menos una de las variables independientes contribuye de forma significativa a la explicación de la variable respuesta.

Para las variables tamaño familiar y número de hijos en la Universidad, los p-valores son 0.1655 y 0.9034, respectivamente. Ambos mayores que 0.05, por lo que no rechazamos la hipótesis nula de significación de ambas variables. Estas variables no son válidas para predecir los gastos alimentación mensual de una familia y por tanto se pueden eliminar del modelo.

Con respecto a las representaciones gráficas, se pueden representar gráficos de dispersión de la variable dependiente con respecto a cada una de las variables independientes mediante el comando plot, como se ha mostrado anteriormente.

6.3.3.1. Otros modelos de regresión: regresión cuadrática

Aunque los modelos de regresión lineal (tanto simple como múltiple) funcionan bien en una amplia mayoría de situaciones, en ocasiones es necesario considerar modelos más complejos para conseguir un mejor ajuste a los datos.

Un ejemplo de este tipo de modelos es la regresión cuadrática. El modelo más sencillo de regresión cuadrática es el siguiente:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i}^2 + \epsilon_i$$

Para ajustar un modelo de regresión cuadrático, basta con indicar en el argumento formula de la función `lm` que una de las variables independientes está elevada al cuadrado mediante el símbolo `^2`.

```
lm(formula = gastos ~ ingresos + I(tamaño^2),data=datos)
```

```
##
## Call:
## lm(formula = gastos ~ ingresos + I(tamaño^2), data = datos)
##
## Coefficients:
## (Intercept)     ingresos  I(tamaño^2)
##   4.341e+02    6.959e-03    4.435e+00
```

6.3.4. Modelos con interacciones

Supongamos que el departamento de ventas de una empresa quiere estudiar la influencia que tiene la publicidad a través de distintos canales sobre el número de ventas de un producto. Se dispone de un conjunto de datos que contiene los ingresos (en millones) conseguido por ventas en 200 regiones, así como la cantidad de presupuesto, también en millones, destinado a anuncios por radio, TV y periódicos en cada una de ellas.

```
tv <- c(230.1, 44.5, 17.2, 151.5, 180.8, 8.7, 57.5, 120.2, 8.6, 199.8, 66.1,
    214.7, 23.8, 97.5, 204.1, 195.4, 67.8, 281.4, 69.2, 147.3, 218.4, 237.4,
    13.2, 228.3, 62.3, 262.9, 142.9, 240.1, 248.8, 70.6, 292.9, 112.9, 97.2,
    265.6, 95.7, 290.7, 266.9, 74.7, 43.1, 228, 202.5, 177, 293.6, 206.9, 25.1,
    175.1, 89.7, 239.9, 227.2, 66.9, 199.8, 100.4, 216.4, 182.6, 262.7, 198.9,
    7.3, 136.2, 210.8, 210.7, 53.5, 261.3, 239.3, 102.7, 131.1, 69, 31.5, 139.3,
    237.4, 216.8, 199.1, 109.8, 26.8, 129.4, 213.4, 16.9, 27.5, 120.5, 5.4,
    116, 76.4, 239.8, 75.3, 68.4, 213.5, 193.2, 76.3, 110.7, 88.3, 109.8, 134.3,
    28.6, 217.7, 250.9, 107.4, 163.3, 197.6, 184.9, 289.7, 135.2, 222.4, 296.4,
    280.2, 187.9, 238.2, 137.9, 25, 90.4, 13.1, 255.4, 225.8, 241.7, 175.7,
```

```
    209.6, 78.2, 75.1, 139.2, 76.4, 125.7, 19.4, 141.3, 18.8, 224, 123.1, 229.5,
    87.2, 7.8, 80.2, 220.3, 59.6, 0.7, 265.2, 8.4, 219.8, 36.9, 48.3, 25.6,
    273.7, 43, 184.9, 73.4, 193.7, 220.5, 104.6, 96.2, 140.3, 240.1, 243.2,
    38, 44.7, 280.7, 121, 197.6, 171.3, 187.8, 4.1, 93.9, 149.8, 11.7, 131.7,
    172.5, 85.7, 188.4, 163.5, 117.2, 234.5, 17.9, 206.8, 215.4, 284.3, 50,
    164.5, 19.6, 168.4, 222.4, 276.9, 248.4, 170.2, 276.7, 165.6, 156.6, 218.5,
    56.2, 287.6, 253.8, 205, 139.5, 191.1, 286, 18.7, 39.5, 75.5, 17.2, 166.8,
    149.7, 38.2, 94.2, 177, 283.6, 232.1)
radio <- c(37.8, 39.3, 45.9, 41.3, 10.8, 48.9, 32.8, 19.6, 2.1, 2.6, 5.8, 24,
    35.1, 7.6, 32.9, 47.7, 36.6, 39.6, 20.5, 23.9, 27.7, 5.1, 15.9, 16.9, 12.6,
    3.5, 29.3, 16.7, 27.1, 16, 28.3, 17.4, 1.5, 20, 1.4, 4.1, 43.8, 49.4, 26.7,
    37.7, 22.3, 33.4, 27.7, 8.4, 25.7, 22.5, 9.9, 41.5, 15.8, 11.7, 3.1, 9.6,
    41.7, 46.2, 28.8, 49.4, 28.1, 19.2, 49.6, 29.5, 2, 42.7, 15.5, 29.6, 42.8,
    9.3, 24.6, 14.5, 27.5, 43.9, 30.6, 14.3, 33, 5.7, 24.6, 43.7, 1.6, 28.5,
    29.9, 7.7, 26.7, 4.1, 20.3, 44.5, 43, 18.4, 27.5, 40.6, 25.5, 47.8, 4.9,
    1.5, 33.5, 36.5, 14, 31.6, 3.5, 21, 42.3, 41.7, 4.3, 36.3, 10.1, 17.2, 34.3,
    46.4, 11, 0.3, 0.4, 26.9, 8.2, 38, 15.4, 20.6, 46.8, 35, 14.3, 0.8, 36.9,
    16, 26.8, 21.7, 2.4, 34.6, 32.3, 11.8, 38.9, 0, 49, 12, 39.6, 2.9, 27.2,
    33.5, 38.6, 47, 39, 28.9, 25.9, 43.9, 17, 35.4, 33.2, 5.7, 14.8, 1.9, 7.3,
    49, 40.3, 25.8, 13.9, 8.4, 23.3, 39.7, 21.1, 11.6, 43.5, 1.3, 36.9, 18.4,
    18.1, 35.8, 18.1, 36.8, 14.7, 3.4, 37.6, 5.2, 23.6, 10.6, 11.6, 20.9, 20.1,
    7.1, 3.4, 48.9, 30.2, 7.8, 2.3, 10, 2.6, 5.4, 5.7, 43, 21.3, 45.1, 2.1,
    28.7, 13.9, 12.1, 41.1, 10.8, 4.1, 42, 35.6, 3.7, 4.9, 9.3, 42, 8.6)
periodico <- c(69.2, 45.1, 69.3, 58.5, 58.4, 75, 23.5, 11.6, 1, 21.2, 24.2,
    4, 65.9, 7.2, 46, 52.9, 114, 55.8, 18.3, 19.1, 53.4, 23.5, 49.6, 26.2, 18.3,
    19.5, 12.6, 22.9, 22.9, 40.8, 43.2, 38.6, 30, 0.3, 7.4, 8.5, 5, 45.7, 35.1,
    32, 31.6, 38.7, 1.8, 26.4, 43.3, 31.5, 35.7, 18.5, 49.9, 36.8, 34.6, 3.6,
    39.6, 58.7, 15.9, 60, 41.4, 16.6, 37.7, 9.3, 21.4, 54.7, 27.3, 8.4, 28.9,
    0.9, 2.2, 10.2, 11, 27.2, 38.7, 31.7, 19.3, 31.3, 13.1, 89.4, 20.7, 14.2,
    9.4, 23.1, 22.3, 36.9, 32.5, 35.6, 33.8, 65.7, 16, 63.2, 73.4, 51.4, 9.3,
    33, 59, 72.3, 10.9, 52.9, 5.9, 22, 51.2, 45.9, 49.8, 100.9, 21.4, 17.9,
    5.3, 59, 29.7, 23.2, 25.6, 5.5, 56.5, 23.2, 2.4, 10.7, 34.5, 52.7, 25.6,
    14.8, 79.2, 22.3, 46.2, 50.4, 15.6, 12.4, 74.2, 25.9, 50.6, 9.2, 3.2, 43.1,
    8.7, 43, 2.1, 45.1, 65.6, 8.5, 9.3, 59.7, 20.5, 1.7, 12.9, 75.6, 37.9, 34.4,
    38.9, 9, 8.7, 44.3, 11.9, 20.6, 37, 48.7, 14.2, 37.7, 9.5, 5.7, 50.5, 24.3,
    45.2, 34.6, 30.7, 49.3, 25.6, 7.4, 5.4, 84.8, 21.6, 19.4, 57.6, 6.4, 18.4,
    47.4, 17, 12.8, 13.1, 41.8, 20.3, 35.2, 23.7, 17.6, 8.3, 27.4, 29.7, 71.8,
    30, 19.6, 26.6, 18.2, 3.7, 23.4, 5.8, 6, 31.6, 3.6, 6, 13.8, 8.1, 6.4, 66.2,
    8.7)
ventas <- c(22.1, 10.4, 9.3, 18.5, 12.9, 7.2, 11.8, 13.2, 4.8, 10.6, 8.6, 17.4,
    9.2, 9.7, 19, 22.4, 12.5, 24.4, 11.3, 14.6, 18, 12.5, 5.6, 15.5, 9.7, 12,
    15, 15.9, 18.9, 10.5, 21.4, 11.9, 9.6, 17.4, 9.5, 12.8, 25.4, 14.7, 10.1,
    21.5, 16.6, 17.1, 20.7, 12.9, 8.5, 14.9, 10.6, 23.2, 14.8, 9.7, 11.4, 10.7,
    22.6, 21.2, 20.2, 23.7, 5.5, 13.2, 23.8, 18.4, 8.1, 24.2, 15.7, 14, 18,
    9.3, 9.5, 13.4, 18.9, 22.3, 18.3, 12.4, 8.8, 11, 17, 8.7, 6.9, 14.2, 5.3,
    11, 11.8, 12.3, 11.3, 13.6, 21.7, 15.2, 12, 16, 12.9, 16.7, 11.2, 7.3, 19.4,
    22.2, 11.5, 16.9, 11.7, 15.5, 25.4, 17.2, 11.7, 23.8, 14.8, 14.7, 20.7,
    19.2, 7.2, 8.7, 5.3, 19.8, 13.4, 21.8, 14.1, 15.9, 14.6, 12.6, 12.2, 9.4,
    15.9, 6.6, 15.5, 7, 11.6, 15.2, 19.7, 10.6, 6.6, 8.8, 24.7, 9.7, 1.6, 12.7,
    5.7, 19.6, 10.8, 11.6, 9.5, 20.8, 9.6, 20.7, 10.9, 19.2, 20.1, 10.4, 11.4,
    10.3, 13.2, 25.4, 10.9, 10.1, 16.1, 11.6, 16.6, 19, 15.6, 3.2, 15.3, 10.1,
    7.3, 12.9, 14.4, 13.3, 14.9, 18, 11.9, 11.9, 8, 12.2, 17.1, 15, 8.4, 14.5,
    7.6, 11.7, 11.5, 27, 20.2, 11.7, 11.8, 12.6, 10.5, 12.2, 8.7, 26.2, 17.6,
```

```
    22.6, 10.3, 17.3, 15.9, 6.7, 10.8, 9.9, 5.9, 19.6, 17.3, 7.6, 9.7, 12.8,
    25.5, 13.4)

datos <- data.frame(tv, radio, periodico, ventas)
```

El modelo lineal múltiple que se obtiene empleando las variables `tv`, `radio` y `periodico` como predictores de ventas es el siguiente:

```
modelo <- lm(ventas ~ tv + radio + periodico, data = datos)
summary(modelo)
```

```
##
## Call:
## lm(formula = ventas ~ tv + radio + periodico, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## tv           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## periodico   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

De acuerdo al p-value obtenido para el coeficiente parcial de regresión de `periodico`, esta variable no contribuye de forma significativa al modelo. Como resultado de este análisis se concluye que las variables `tv` y `radio` están asociadas con la cantidad de ventas.

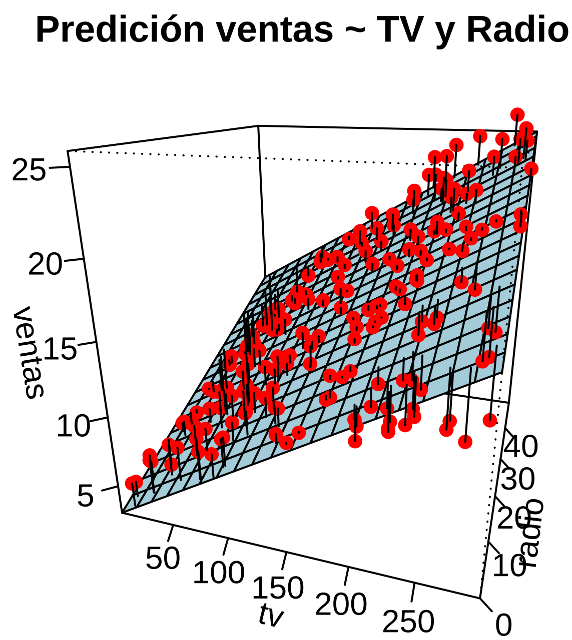
Con el comando `update()`, podemos actualizar el `modelo`. Por ejemplo:

```
modelo <- update(modelo, .~. -periodico)
summary(modelo)
```

```
##
## Call:
## lm(formula = ventas ~ tv + radio, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7977 -0.8752  0.2422  1.1708  2.8328
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.92110    0.29449   9.919   <2e-16 ***
## tv           0.04575    0.00139  32.909   <2e-16 ***
## radio        0.18799    0.00804  23.382   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.681 on 197 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8962
## F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

Al ser un modelo con dos predictores continuos se puede representar en 3D



El modelo lineal a partir del cual se han obtenido las conclusiones asume que el efecto sobre las ventas debido a un incremento en el presupuesto de uno de los medios de comunicación es independiente del presupuesto gastado en los otros. Por ejemplo, el modelo lineal considera que el efecto promedio sobre las ventas debido a aumentar en una unidad el presupuesto de anuncios en TV es siempre de 0.04575, independientemente de la cantidad invertida en anuncios por radio. Sin embargo, la representación gráfica muestra que el modelo tiende a sobrevalorar las ventas cuando el presupuesto es muy alto en uno de los medios pero muy bajo en el otro. Por contra, los valores de ventas predichos por el modelo están por debajo de las ventas reales cuando el presupuesto está repartido de forma equitativa entre ambos medios. Este comportamiento sugiere que existe interacción entre los predictores, por lo que el efecto de cada uno de ellos sobre la variable respuesta depende en cierta medida del valor que tome el otro predictor.

Tal y como se ha definido previamente, un modelo lineal con dos predictores sigue la ecuación:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

De acuerdo a esta definición, el incremento de una unidad en el predictor X_1 produce un incremento promedio de la variable Y de β_1. Modificaciones en el predictor X_2 no alteran este hecho, y lo mismo ocurre con X_2 respecto a X_1. Para que el modelo pueda contemplar la interacción se introduce un tercer predictor, que se construye con el producto de los predictores X_1 y X_2.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$

La reorganización de los términos resulta en:

$$Y = \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon$$

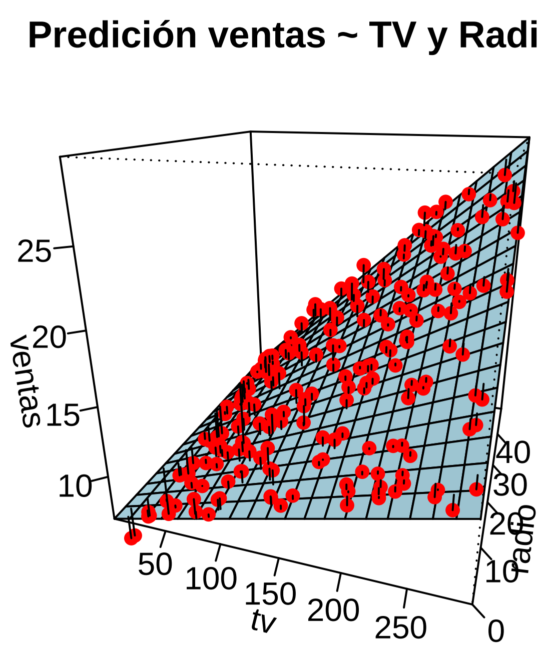
El efecto de X_1 sobre Y ya no es constante, sino que depende del valor que tome X_2.

En R se puede introducir interacción entre predictores de dos formas, indicando los predictores individuales y entre cuales se quiere evaluar la interacción, o bien de forma directa.

```
modelo_interaccion <- lm(formula = ventas ~ tv + radio + tv:radio, data = datos)
summary(modelo_interaccion)
```

```
##
## Call:
## lm(formula = ventas ~ tv + radio + tv:radio, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3366 -0.4028  0.1831  0.5948  1.5246
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.750e+00  2.479e-01  27.233   <2e-16 ***
## tv          1.910e-02  1.504e-03  12.699   <2e-16 ***
## radio       2.886e-02  8.905e-03   3.241   0.0014 **
## tv:radio    1.086e-03  5.242e-05  20.727   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9435 on 196 degrees of freedom
## Multiple R-squared:  0.9678, Adjusted R-squared:  0.9673
## F-statistic:  1963 on 3 and 196 DF,  p-value: < 2.2e-16
```

```
# O tambien
# lm(formula = ventas ~ tv * radio, data = datos) es equivalente.
```



Los resultados muestran una evidencia clara de que la interacción `tv:radio` es significativa y de que el modelo que incorpora la interacción (Adjusted R-squared = 0.9673) es superior al modelo que solo contemplaba el efecto de los predictores por separado (Adjusted R-squared = 0.8956).

Se puede emplear un ANOVA para realizar un test de hipótesis y obtener un p-value que evalúe la hipótesis nula de que ambos modelos se ajustan a los datos igual de bien.

```r
anova(modelo, modelo_interaccion)
```

```
## Analysis of Variance Table
##
## Model 1: ventas ~ tv + radio
## Model 2: ventas ~ tv + radio + tv:radio
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    197 556.91
## 2    196 174.48  1    382.43 429.59 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En los modelos de regresión lineal múltiple que incorporan interacciones entre predictores hay que tener en cuenta que si se incorpora al modelo una interacción entre predictores, se deben incluir siempre los predictores individuales que participan en la interacción, independientemente de que su p-value sega significativo o no.

La interacción entre predictores no está limitada a predictores cuantitativos, también puede crearse interacción entre predictor cuantitativo y cualitativo.

Capítulo 7

Regresión logística

Una regresión logística se utiliza típicamente cuando hay una variable de resultado dicotómica (como ganar o perder), y una variable predictiva continua que está relacionada con la probabilidad o "odds" de la variable de resultado. También se puede utilizar con predictores categóricos y con múltiples predictores.

Si usamos una regresión lineal para modelar una variable dicotómica (como Y), es posible que el modelo resultante no restrinja los Y pronosticados dentro de 0 y 1. Además, otros supuestos de regresión lineal como la normalidad del error pueden ser violados. Así que en vez de eso, modelamos las probabilidades del evento de $log (logit) o logit, donde, p es la probabilidad del evento.

$$z_i = \ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

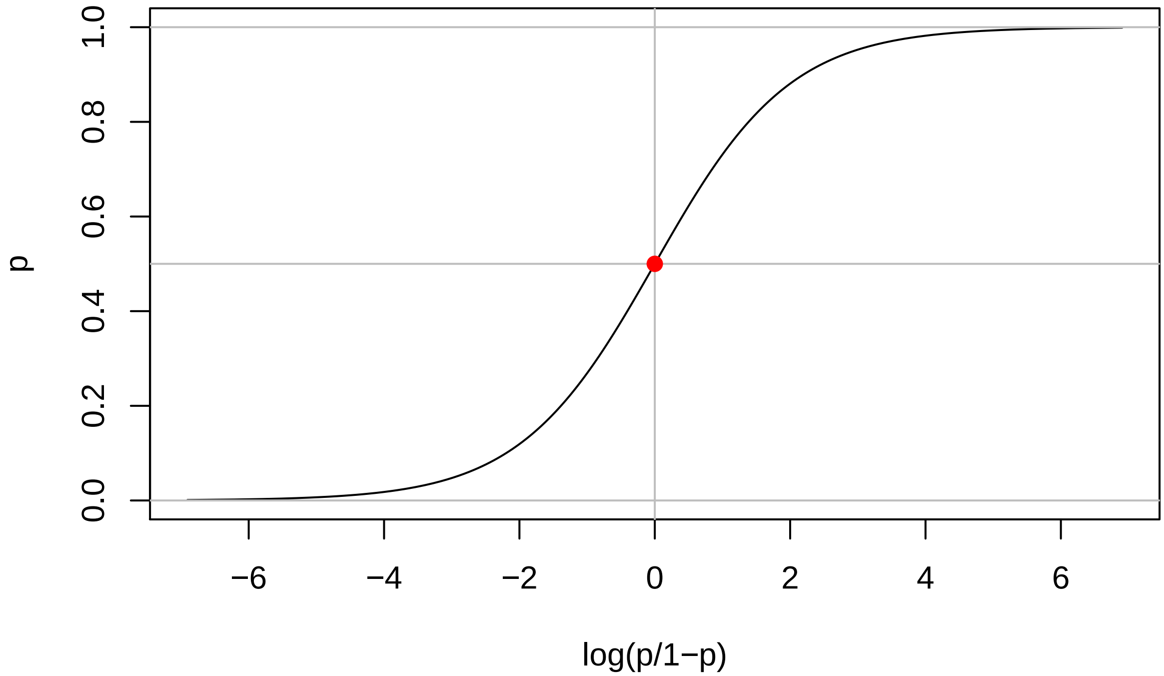
La ecuación anterior puede ser modelada usando `glm()` mediante el argumento `family="binomial`. Pero estamos más interesados en la probabilidad del evento que en las probabilidades logarítmicas del evento. Por lo tanto, los valores pronosticados del modelo anterior, es decir, las probabilidades logarítmicas del evento, pueden convertirse a probabilidad de evento de la siguiente manera:

$$p_i = 1 - \frac{1}{1+\exp(z_i)}$$

A esto se le llama la *logit-inversa*, `?plogis`.

El siguiente gráfico relaciona `p` y `logit(p)`.

```
p <- seq(0,1,l=1000)
logitp <- log(p/(1-p))
plot(logitp,p,t='l',xlab="log(p/1-p)")
abline(h=c(0,0.5,1),v=0,col="grey")
points(0,0.5,pch=19,col="red")
```



7.1. Ejemplo: Datos de crédito en Alemania (*Credit scoring*)

Cuando un banco recibe una solicitud de préstamo, basada en el perfil del solicitante, el banco tiene que tomar una decisión sobre si procede o no con la aprobación del préstamo. Hay dos tipos de riesgos asociados a la decisión del banco:

- Si el solicitante tiene un buen riesgo de crédito, es decir, es probable que pague el préstamo, entonces no aprobar el préstamo a la persona resulta en una pérdida de negocio para el banco.
- Si el solicitante tiene un riesgo de crédito malo, es decir, no es probable que pague el préstamo, entonces la aprobación del préstamo a la persona resulta en una pérdida financiera para el banco.

El objetivo de este tipo de análisis es **minimizar el riesgo y maximizar el beneficio del banco**.

Para minimizar las pérdidas desde la perspectiva del banco, el banco necesita una regla de decisión sobre a quién dar la aprobación del préstamo y a quién no. Los perfiles demográficos y socioeconómicos de un solicitante son considerados por los administradores de préstamos antes de que se tome una decisión sobre su solicitud de préstamo.

Los datos German Credit Data contiene datos sobre 20 variables y la clasificación de si un solicitante es considerado un *Bueno* o un *Malo* riesgo de crédito para 1000 solicitantes de préstamos.

Se espera que un *modelo predictivo* elaborado a partir de estos datos sirva de guía al gerente del banco para tomar una decisión sobre la aprobación de un préstamo a un posible solicitante en función de su perfil.

Ver descripción (en inglés)

```
# German Credit Data
gcreditdata <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/ger

colnames(gcreditdata)<-c("account.status","months",
```

```
                "credit.history","purpose","credit.amount",
                "savings","employment","installment.rate","personal.status",
                "guarantors","residence","property","age","other.installments",
                "housing","credit.cards","job","dependents","phone","foreign.worker","credit.rating")

head(gcreditdata)
```

```
##   account.status months credit.history purpose credit.amount savings
## 1            A11      6            A34     A43          1169     A65
## 2            A12     48            A32     A43          5951     A61
## 3            A14     12            A34     A46          2096     A61
## 4            A11     42            A32     A42          7882     A61
## 5            A11     24            A33     A40          4870     A61
## 6            A14     36            A32     A46          9055     A65
##   employment installment.rate personal.status guarantors residence
## 1        A75                4             A93       A101         4
## 2        A73                2             A92       A101         2
## 3        A74                2             A93       A101         3
## 4        A74                2             A93       A103         4
## 5        A73                3             A93       A101         4
## 6        A73                2             A93       A101         4
##   property age other.installments housing credit.cards  job dependents
## 1     A121  67               A143    A152            2 A173          1
## 2     A121  22               A143    A152            1 A173          1
## 3     A121  49               A143    A152            1 A172          2
## 4     A122  45               A143    A153            1 A173          2
## 5     A124  53               A143    A153            2 A173          2
## 6     A124  35               A143    A153            1 A172          2
##   phone foreign.worker credit.rating
## 1  A192           A201             1
## 2  A191           A201             2
## 3  A191           A201             1
## 4  A191           A201             1
## 5  A191           A201             2
## 6  A192           A201             1
```

Cambie el nombre de algunos niveles de factor:

```
table(gcreditdata$credit.rating) # Good = 1 / Bad = 2
```

```
##
##   1   2
## 700 300
```

```
gcreditdata$credit.rating <- ifelse(gcreditdata$credit.rating==1,1,0)
table(gcreditdata$credit.rating) # Good = 1 / Bad = 0
```

```
##
##   0   1
## 300 700
```

```
levels(gcreditdata$account.status) <- c("<0DM","<200DM",">200DM","NoStatus")
levels(gcreditdata$credit.history) <- c("No","Allpaid","Allpaidtillnow","Delayinpaying","Critical")
levels(gcreditdata$purpose) <- c("car(new)","car(used)","furniture/equipment","radio/television","domes
"repairs","education","vacation-doesnotexist?","retraining","business","others")
```

Dividamos los datos en dos grupos (entrenamiento y prueba), 70 %/30 %.

```
gcredit.matrix <- model.matrix(credit.rating ~ . , data = gcreditdata)

n<- dim(gcreditdata)[1]

set.seed(1234) # select a random sample with
 train <- sample(1:n , 0.7*n)
xtrain <- gcredit.matrix[train,]
xtest  <- gcredit.matrix[-train,]

ytrain <- gcreditdata$credit.rating[train]
ytrain <- as.factor(ytrain-1) # convert to 0/1 factor
 ytest <- gcreditdata$credit.rating[-train]
ytest  <- as.factor(ytest-1) #  convert to 0/1 factor
```

Un modelo logístico puede ser ajustado con la función `glm`.

```
m1 <- glm(credit.rating ~ . , family = binomial, data= data.frame(credit.rating= ytrain, xtrain))
summary(m1)
```

```
##
## Call:
## glm(formula = credit.rating ~ ., family = binomial, data = data.frame(credit.rating = ytrain,
##     xtrain))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8353  -0.6178   0.3415   0.6729   2.0514
##
## Coefficients: (2 not defined because of singularities)
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    5.259e-01  1.371e+00   0.384 0.701257
## X.Intercept.                          NA         NA      NA       NA
## account.status.200DM           3.532e-01  2.719e-01   1.299 0.193876
## account.status.200DM.1         8.198e-01  4.626e-01   1.772 0.076353 .
## account.statusNoStatus         1.487e+00  2.782e-01   5.344 9.08e-08 ***
## months                        -3.687e-02  1.156e-02  -3.189 0.001428 **
## credit.historyAllpaid         -1.347e-01  6.539e-01  -0.206 0.836779
## credit.historyAllpaidtillnow   5.296e-01  4.975e-01   1.065 0.287038
## credit.historyDelayinpaying    7.156e-01  5.575e-01   1.284 0.199266
## credit.historyCritical         1.439e+00  5.073e-01   2.837 0.004559 **
## purposecar.used.               1.838e+00  4.856e-01   3.786 0.000153 ***
## purposefurniture.equipment     1.095e+00  9.231e-01   1.186 0.235728
## purposeradio.television        8.879e-01  3.289e-01   2.700 0.006944 **
## purposedomesticappliances      9.928e-01  3.107e-01   3.196 0.001395 **
## purposerepairs                 5.399e-01  9.006e-01   0.599 0.548881
## purposeeducation              -4.889e-01  6.587e-01  -0.742 0.457933
## purposevacation.doesnotexist. -1.800e-01  4.648e-01  -0.387 0.698616
## purposeretraining              2.093e+00  1.266e+00   1.653 0.098417 .
## purposebusiness                5.772e-01  3.985e-01   1.448 0.147489
```

```
## purposeothers                         NA         NA      NA       NA
## credit.amount                 -1.107e-04  5.456e-05  -2.028 0.042526 *
## savingsA62                     5.219e-01  3.548e-01   1.471 0.141326
## savingsA63                     8.722e-01  5.710e-01   1.528 0.126623
## savingsA64                     1.746e+00  6.533e-01   2.673 0.007514 **
## savingsA65                     1.354e+00  3.350e-01   4.042 5.30e-05 ***
## employmentA72                  1.729e-01  5.327e-01   0.324 0.745566
## employmentA73                  2.462e-01  5.116e-01   0.481 0.630356
## employmentA74                  1.182e+00  5.660e-01   2.089 0.036736 *
## employmentA75                  4.924e-02  5.076e-01   0.097 0.922712
## installment.rate              -4.157e-01  1.098e-01  -3.787 0.000152 ***
## personal.statusA92             5.279e-02  4.718e-01   0.112 0.910902
## personal.statusA93             6.791e-01  4.577e-01   1.484 0.137906
## personal.statusA94             2.017e-01  5.550e-01   0.363 0.716253
## guarantorsA102                 1.880e-01  5.068e-01   0.371 0.710613
## guarantorsA103                 1.024e+00  5.814e-01   1.761 0.078259 .
## residence                      5.357e-02  1.047e-01   0.512 0.608926
## propertyA122                  -4.618e-01  3.142e-01  -1.470 0.141584
## propertyA123                  -4.679e-01  2.906e-01  -1.610 0.107366
## propertyA124                  -7.642e-01  5.658e-01  -1.351 0.176834
## age                            1.872e-02  1.172e-02   1.598 0.110121
## other.installmentsA142        -1.259e-01  5.197e-01  -0.242 0.808648
## other.installmentsA143         3.400e-01  2.892e-01   1.176 0.239669
## housingA152                    5.768e-01  2.935e-01   1.965 0.049384 *
## housingA153                    7.825e-01  6.261e-01   1.250 0.211387
## credit.cards                  -3.720e-01  2.456e-01  -1.515 0.129882
## jobA172                       -8.402e-01  8.307e-01  -1.011 0.311822
## jobA173                       -9.497e-01  7.970e-01  -1.192 0.233388
## jobA174                       -9.513e-01  8.262e-01  -1.152 0.249525
## dependents                    -2.108e-01  3.080e-01  -0.685 0.493620
## phoneA192                      4.876e-01  2.556e-01   1.908 0.056400 .
## foreign.workerA202             9.540e-01  7.550e-01   1.264 0.206386
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 839.40  on 699  degrees of freedom
## Residual deviance: 596.11  on 651  degrees of freedom
## AIC: 694.11
##
## Number of Fisher Scoring iterations: 5
```

Variables significativas:

```
sig.var<- summary(m1)$coeff[-1,4] <0.01
names(sig.var)[sig.var == TRUE]
```

```
## [1] "account.statusNoStatus"    "months"
## [3] "credit.historyCritical"    "purposecar.used."
## [5] "purposeradio.television"   "purposedomesticappliances"
## [7] "savingsA64"                "savingsA65"
## [9] "installment.rate"
```

Con `predict` podemos predecir con el modelo logístico el conjunto de test.

```
pred1<- predict.glm(m1,newdata = data.frame(ytest,xtest), type="response")
result1<- table(ytest, floor(pred1+1.5))
result1
```

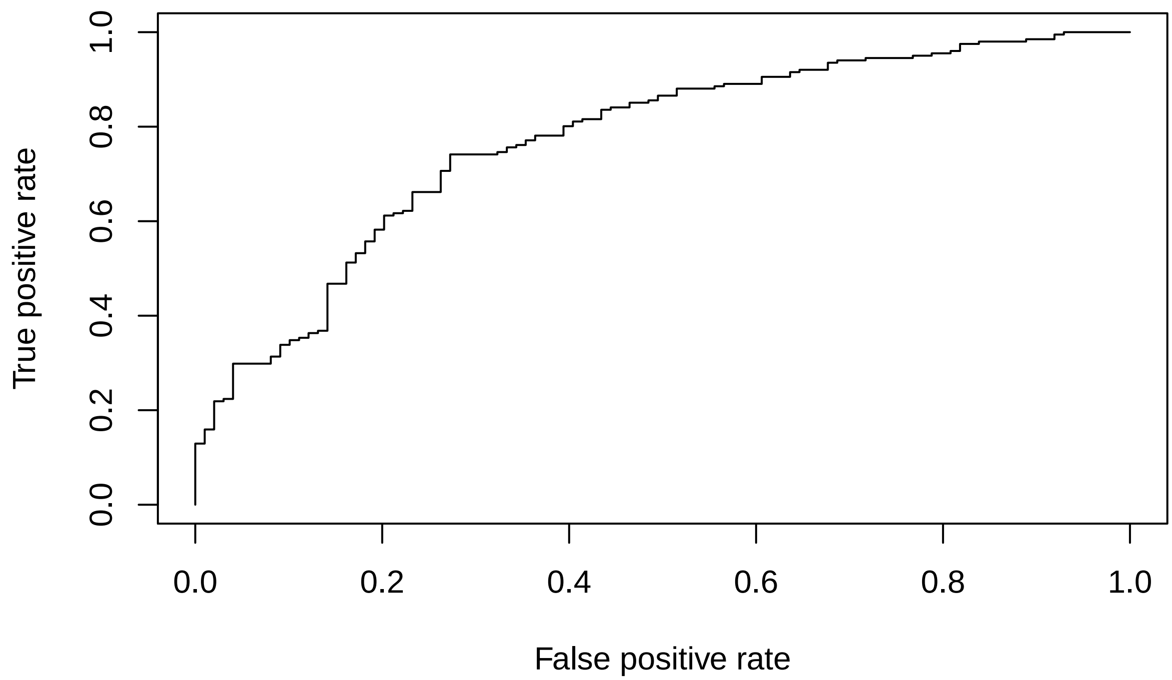
```
##
## ytest   1   2
##    -1  48  51
##    0   25 176
```

```
error1<- sum(result1[1,2], result1[2,1])/sum(result1)
error1
```

```
## [1] 0.2533333
```

Curva ROC con la librería `ROCR`:

```
library(ROCR)
pred = prediction(pred1,ytest)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
```




```
AUCLog1=performance(pred, measure = "auc")@y.values[[1]]
cat("AUC: ",AUCLog1,"n")
```

```
## AUC:  0.7699382 n
```

7.2. Ejemplo: Predecir el salario de los trabajadores

Consideremos los datos `adult.csv`. Intentaremos predecir la variable de respuesta `ABOVE50k` (Salario >50k) mediante una regresión logística basada en variables demográficas explicativas.

```
inputData <- read.csv("http://idaejin.github.io/courses/R/data/adult.csv")
head(inputData)
```

```
##   AGE         WORKCLASS FNLWGT  EDUCATION EDUCATIONNUM       MARITALSTATUS
## 1  39         State-gov  77516  Bachelors           13       Never-married
## 2  50  Self-emp-not-inc  83311  Bachelors           13  Married-civ-spouse
## 3  38           Private 215646    HS-grad            9            Divorced
## 4  53           Private 234721       11th            7  Married-civ-spouse
## 5  28           Private 338409  Bachelors           13  Married-civ-spouse
## 6  37           Private 284582    Masters           14  Married-civ-spouse
##          OCCUPATION   RELATIONSHIP   RACE     SEX CAPITALGAIN CAPITALLOSS
## 1      Adm-clerical  Not-in-family  White    Male        2174           0
## 2   Exec-managerial        Husband  White    Male           0           0
## 3 Handlers-cleaners  Not-in-family  White    Male           0           0
## 4 Handlers-cleaners        Husband  Black    Male           0           0
## 5    Prof-specialty           Wife  Black  Female           0           0
## 6   Exec-managerial           Wife  White  Female           0           0
##   HOURSPERWEEK  NATIVECOUNTRY ABOVE50K
## 1           40  United-States        0
## 2           13  United-States        0
## 3           40  United-States        0
## 4           40  United-States        0
## 5           40           Cuba        0
## 6           40  United-States        0
```

Verificar sesgo de clase

Idealmente, la proporción de eventos y no eventos en la variable Y debería ser aproximadamente la misma. Por lo tanto, primero verifiquemos la proporción de clases en la variable dependiente `ABOVE50K`.

```
table(inputData$ABOVE50K)
```

```
##
##     0     1
## 24720  7841
```

Claramente, existe un sesgo de clase, una condición observada cuando la proporción de eventos es mucho menor que la proporción de no eventos. Por lo tanto, debemos muestrear las observaciones en proporciones aproximadamente iguales para obtener mejores modelos.

Crear Muestras de Entrenamiento y Pruebas

Una manera de abordar el problema del sesgo de clase es muestrear los 0 y 1 para los `trainingData` (muestra de entrenamiento) en proporciones iguales. Al hacerlo, pondremos el resto de los `inputData` no incluidos para la formación en `testData` (muestra de validación). Como resultado, el tamaño de la muestra de entrenamiento será menor que el de la validación, lo que está bien, porque hay un gran número de observaciones (`>10K`).

```
# Create Training Data
 input_ones <- inputData[which(inputData$ABOVE50K == 1), ]  # all 1's
input_zeros <- inputData[which(inputData$ABOVE50K == 0), ]  # all 0's

set.seed(100)  # for repeatability of samples

input_ones_training_rows <- sample(1:nrow(input_ones), 0.7*nrow(input_ones))  # 1's for training
input_zeros_training_rows <- sample(1:nrow(input_zeros), 0.7*nrow(input_ones))  # 0's for training.

# Pick as many 0's as 1's
```

```
training_ones <- input_ones[input_ones_training_rows, ]
training_zeros <- input_zeros[input_zeros_training_rows, ]
trainingData <- rbind(training_ones, training_zeros)  # row bind the 1's and 0's

# Create Test Data
 test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]

testData <- rbind(test_ones, test_zeros)  # row bind the 1's and 0's
```

Construir modelos de Logit y predecir

```
logitMod <- glm(ABOVE50K ~ RELATIONSHIP + AGE + CAPITALGAIN + OCCUPATION + EDUCATIONNUM, data=trainingDa
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
predicted <- plogis(predict(logitMod, testData))  # predicted scores
# or
predicted <- predict(logitMod, testData, type="response")  # predicted scores
```

Cuando usamos la función de predicción en este modelo, predecirá las probabilidades de la variable Y. Para convertirlo en una probabilidad de predicción que esté entre 0 y 1, usamos el `plogis()`.

Decidir la probabilidad de corte de predicción óptima para el modelo.

El puntaje de probabilidad de la predicción de corte por defecto es de 0,5 o la proporción de 1's y 0's en los datos de entrenamiento. Pero a veces, afinar el corte de probabilidad puede mejorar la precisión tanto en las muestras de desarrollo como en las de validación. La función `InformationValue::optimalCutoff` proporciona formas de encontrar el punto de corte óptimo para mejorar la predicción de 1, 0, 1 y 0 y reducir el error de clasificación. Permite calcular la puntuación óptima que minimiza el error de clasificación para el modelo anterior.

```
library(InformationValue)
optCutOff <- optimalCutoff(testData$ABOVE50K, predicted)[1]
optCutOff
```

```
## [1] 0.89
```

Error de clasificación

El error de clasificación errónea es el desajuste porcentual de los valores predefinidos frente a los reales, independientemente de que sean 1 o 0. Cuanto menor sea el error de clasificación, mejor será su modelo.

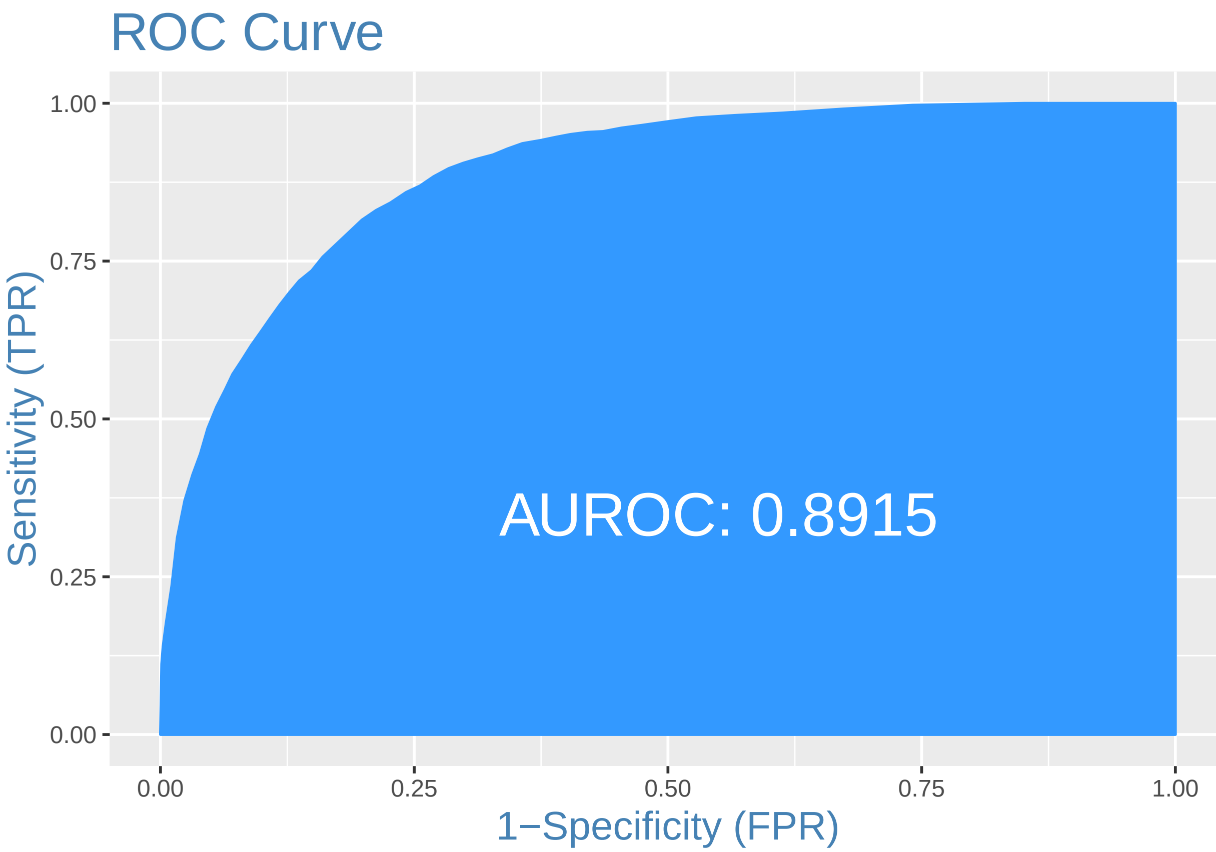
```
misClassError(testData$ABOVE50K, predicted, threshold = optCutOff)
```

```
## [1] 0.0892
```

ROC

Receiver Operating Characteristics a curva traza el porcentaje de verdaderos positivos pronosticados con precisión por un modelo logit dado a medida que la probabilidad de corte de la predicción se reduce de 1 a 0. Para un buen modelo, a medida que se reduce el corte, debería marcar más de 1 real como positivo y menos de 0 real como 1. Por lo tanto, para un buen modelo, la curva debería subir bruscamente, indicando que el TPR (eje Y) aumenta más rápido que el FPR (eje X) a medida que disminuye la puntuación de corte. Cuanto mayor sea el área bajo la curva ROC, mejor será la capacidad de predicción del modelo.

```
plotROC(testData$ABOVE50K, predicted)
```



Especificidad y sensibilidad

La **Sensibilidad** (o `Tasa Verdaderos Positivos`) es el porcentaje de 1's (reales) correctamente predichos por el modelo, mientras que, **especificidad** es el porcentaje de 0's (reales) correctamente predicho. **La especificidad** también puede calcularse como `1-Tasa Falsos Positivos`.

$$\text{Sensitivity} = \frac{\#\text{Actual 1's and Predicted as 1's}}{\#\text{of Actual 1's}}$$
$$\text{Specificity} = \frac{\#\text{Actual 0's and Predicted as 0's}}{\#\text{of Actual 0's}}$$

```
sensitivity(testData$ABOVE50K, predicted, threshold = optCutOff)
```

```
## [1] 0.3442414
```

```
specificity(testData$ABOVE50K, predicted, threshold = optCutOff)
```

```
## [1] 0.9800853
```

Los números anteriores se calculan a partir de la muestra de validación que no se utilizó para la formación del modelo. Así que, una tasa de detección de la verdad de 34.42 % en los datos de prueba es bueno.

Matriz de Confusión Las columnas son reales, mientras que las filas son predicciones.

```
confusionMatrix(testData$ABOVE50K, predicted, threshold = optCutOff)
```

```
##       0    1
## 0 18849 1543
```

```
## 1   383  810
```

7.3. Ejemplo: Datos de los supervivientes del Titanic

El conjunto de datos es una colección de datos sobre algunos de los pasajeros, y el objetivo es predecir la supervivencia (1 si el pasajero sobrevivió o 0 si no lo hizo) basándose en algunas características como la clase de servicio, el sexo, la edad, etc. Como puede ver, vamos a utilizar variables categóricas y continuas.

```
VARIABLE DESCRIPTIONS:
pclass          Passenger Class
                (1 = 1st; 2 = 2nd; 3 = 3rd)
survival        Survival
                (0 = No; 1 = Yes)
name            Name
sex             Sex
age             Age
sibsp           Number of Siblings/Spouses Aboard
parch           Number of Parents/Children Aboard
ticket          Ticket Number
fare            Passenger Fare
cabin           Cabin
embarked        Port of Embarkation
                (C = Cherbourg; Q = Queenstown; S = Southampton)
boat            Lifeboat
body            Body Identification Number
home_dest       Home/Destination
```

Descripción completa (Aquí)

Descarga los datos aquí

Leemos los datos de entreamiento `train` y `test`:

```
train <- read.csv('data/titanic_train.csv',header=TRUE,row.names=1)
 test <- read.csv('data/titanic_test.csv',header=TRUE,row.names=1)
```

Questions:

- Ajustar un modelo logístico con `pclass` como variable explicativa. ¿Cuál es la interpretación del modelo ajustado?
- Encontrar el mejor modelo de regresión logística posible basado en todas las variables disponibles.

```
model <- glm(Survived ~.,family=binomial(link='logit'),data=train)
summary(model)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6064  -0.5954  -0.4254   0.6220   2.4165
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.137627   0.594998   8.635  < 2e-16 ***
```

```
## Pclass      -1.087156   0.151168  -7.192 6.40e-13 ***
## Sexmale     -2.756819   0.212026 -13.002  < 2e-16 ***
## Age         -0.037267   0.008195  -4.547 5.43e-06 ***
## SibSp       -0.292920   0.114642  -2.555   0.0106 *
## Parch       -0.116576   0.128127  -0.910   0.3629
## Fare         0.001528   0.002353   0.649   0.5160
## EmbarkedQ   -0.002656   0.400882  -0.007   0.9947
## EmbarkedS   -0.318786   0.252960  -1.260   0.2076
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1065.39  on 799  degrees of freedom
## Residual deviance:  709.39  on 791  degrees of freedom
## AIC: 727.39
##
## Number of Fisher Scoring iterations: 5
```

```
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Survived
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                       799    1065.39
## Pclass    1   83.607       798     981.79 < 2.2e-16 ***
## Sex       1  240.014       797     741.77 < 2.2e-16 ***
## Age       1   17.495       796     724.28 2.881e-05 ***
## SibSp     1   10.842       795     713.43  0.000992 ***
## Parch     1    0.863       794     712.57  0.352873
## Fare      1    0.994       793     711.58  0.318717
## Embarked  2    2.187       791     709.39  0.334990
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod1 <- glm(Survived ~ as.factor(Pclass), family=binomial, data=train)
summary(mod1)
```

```
##
## Call:
## glm(formula = Survived ~ as.factor(Pclass), family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3787  -0.7515  -0.7515   0.9887   1.6747
##
## Coefficients:
```

```
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.4616     0.1474   3.131  0.00174 **
## as.factor(Pclass)2  -0.5455     0.2138  -2.551  0.01074 *
## as.factor(Pclass)3  -1.5816     0.1844  -8.575  < 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: 1065.39  on 799  degrees of freedom
## Residual deviance:  979.94  on 797  degrees of freedom
## AIC: 985.94
##
## Number of Fisher Scoring iterations: 4
```

```
anova(mod1,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Survived
##
## Terms added sequentially (first to last)
##
##
##                    Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                 799    1065.39
## as.factor(Pclass)  2   85.452       797     979.94 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

efecto de interacción entre la clase de pasajero y el sexo, ya que la clase de pasajero mostró una diferencia mucho mayor en la tasa de supervivencia entre las mujeres en comparación con los hombres (es decir, las mujeres de clase superior tenían muchas más probabilidades de sobrevivir que las mujeres de clase inferior, mientras que los hombres de primera clase tenían más probabilidades de sobrevivir que los hombres de segunda o tercera clase, pero no por el mismo margen que las mujeres).

```
mod2 <- glm(Survived ~ Pclass + Sex  + Age + SibSp, family = binomial(logit), data = train)
summary(mod2)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp, family = binomial(logit),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6595  -0.6125  -0.4247   0.6149   2.4302
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.05604    0.50130  10.086  < 2e-16 ***
## Pclass      -1.14391    0.12585  -9.089  < 2e-16 ***
## Sexmale     -2.75564    0.20471 -13.461  < 2e-16 ***
## Age         -0.03725    0.00812  -4.588 4.48e-06 ***
```

```
## SibSp       -0.33075    0.10892  -3.037  0.00239 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1065.39  on 799  degrees of freedom
## Residual deviance:  713.43  on 795  degrees of freedom
## AIC: 723.43
##
## Number of Fisher Scoring iterations: 5
```

```
anova(mod2,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Survived
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                     799    1065.39
## Pclass  1   83.607       798     981.79 < 2.2e-16 ***
## Sex     1  240.014       797     741.77 < 2.2e-16 ***
## Age     1   17.495       796     724.28 2.881e-05 ***
## SibSp   1   10.842       795     713.43  0.000992 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod3 <-  glm(Survived ~ Pclass + Sex + Pclass:Sex + Age + SibSp, family = binomial(logit), data = train)
summary(mod3)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Pclass:Sex + Age + SibSp,
##     family = binomial(logit), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1993  -0.6265  -0.4770   0.4485   2.3093
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     7.606411   0.960804   7.917 2.44e-15 ***
## Pclass         -2.108360   0.316024  -6.672 2.53e-11 ***
## Sexmale        -5.887480   0.920417  -6.397 1.59e-10 ***
## Age            -0.038063   0.008498  -4.479 7.50e-06 ***
## SibSp          -0.310269   0.109370  -2.837 0.004556 **
## Pclass:Sexmale  1.254202   0.338241   3.708 0.000209 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1065.39  on 799  degrees of freedom
## Residual deviance:  695.66  on 794  degrees of freedom
## AIC: 707.66
##
## Number of Fisher Scoring iterations: 6
```

```
anova(mod3,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Survived
##
## Terms added sequentially (first to last)
##
##
##            Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         799    1065.39
## Pclass      1   83.607       798     981.79 < 2.2e-16 ***
## Sex         1  240.014       797     741.77 < 2.2e-16 ***
## Age         1   17.495       796     724.28 2.881e-05 ***
## SibSp       1   10.842       795     713.43  0.000992 ***
## Pclass:Sex  1   17.779       794     695.66 2.481e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En los pasos anteriores, evaluamos brevemente el ajuste del modelo, ahora nos gustaría ver cómo lo está haciendo el modelo al predecir y en un nuevo conjunto de datos. Estableciendo el parámetro `type='response'`, `R` producirá probabilidades en la forma de $P(y = 1|X)$. Nuestro límite de decisión será de 0,5.

Si $P(y = 1|X) > 0{,}5$ entonces $y = 1$ en caso contrario $y = 0$. Tener en cuenta que para algunas aplicaciones diferentes límites de decisión podría ser una mejor opción.

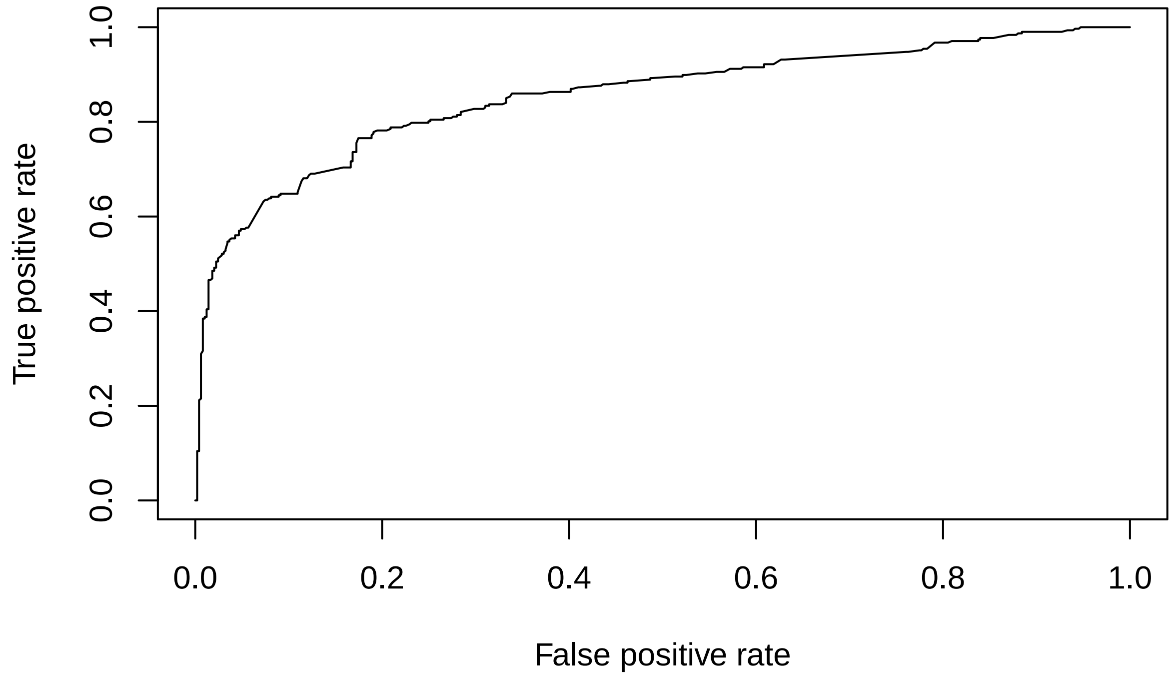
```
fitted.results <- predict(mod3,newdata=test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != test$Survived)
print(paste('Accuracy',1-misClasificError))
```

```
## [1] "Accuracy 0.8075"
```

La precisión de 0.8075 en el conjunto de test es un buen resultado. Sin embargo, hay que tener en cuenta que este resultado depende en cierta medida de la división manual de los datos que hice anteriormente, por lo tanto, si deseamos una puntuación más precisa, sería mejor realizar algún tipo de validación cruzada, como la validación cruzada k-fold.

Evaluar la capacidad predictiva

```
library(ROCR)
p <- predict(mod3, newdata=subset(test,select=c(2,3,4,5,6,7,8)), type="response")
pr <- prediction(p, test$Survived)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8543155
```

Capítulo 8

Modelos Aditivos Generalizados

Capítulo 9

Análisis de Componentes Principales, Clasificación y Clustering

Capítulo 10

Introducción al machine learning

Capítulo 11

Extras