Curso de Estadística básica para Data Scientists

Dae-Jin Lee < lee.daejin@gmail.com >

TEMA 1. Introducción a R

Índice

Regresar a la página principal

1. Introducción al software estadístico R

R es un entorno de programación orientado al cálculo, manipulación de datos, y representación gráfica, publicado como software libre con licencia GNU-GPL.

1.1. 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")
```

- Ver los últimos comandos utilizados en la consola

```
history() # display last 25 commands
history(max.show=Inf) # display all previous commands
```

- Guardar el historial de comandos

```
savehistory(file="myfile") # default is ".Rhistory"
```

- Cargar los commandos guardados en una sesión anterior

```
loadhistory(file="myfile") # default is ".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()
```

1.2. Instalar y cargar librerías en R

```
install.packages("DAAG") # (Data Analysis And Graphics)
```

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

```
library(DAAG) # or require(DAAG)
```

1.3. 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 numeros 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"
```

1.4. 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 librerías de `R` para importar datos:

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

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

Descara en el siguiente link los datos `cardata` aquí

```r
mydata1 = read.table("data/cardata.txt")
mydata2 = read.csv("data/cardata.csv")
```

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

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

- Minitab, SPSS, SAS or Stata

```r
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

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

1.5. 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:

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

- Hoja decálculo de Excel:

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

1.6. Vectores

- Descargar el siguiente código de R aquí
- 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
```

1.7. Estadística básico

- 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 correlacion* 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
```

1.8. 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
```

1.9. 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 ...
```

```r
# 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
```

```r
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
```

```r
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
```

1.10. 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

```r
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

```r
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`

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

4. función `aggregate`

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

1.11. Vectores lógicos

- Elegir los individuos con `BMI>22`

```r
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?

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

1.12. 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) # create a list
unlist(zz) # unlist the list converting it to a concatenated vector
```

```
## [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) # o 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))
```

1.13. 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
```

```r
# 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
```

```r
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
```

```r
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
```

1.14. Factors

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

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

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

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

```
summary(gender)
```

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

```
table(gender)
```

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

```
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
```

1.15. 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
```

1.16. 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
```

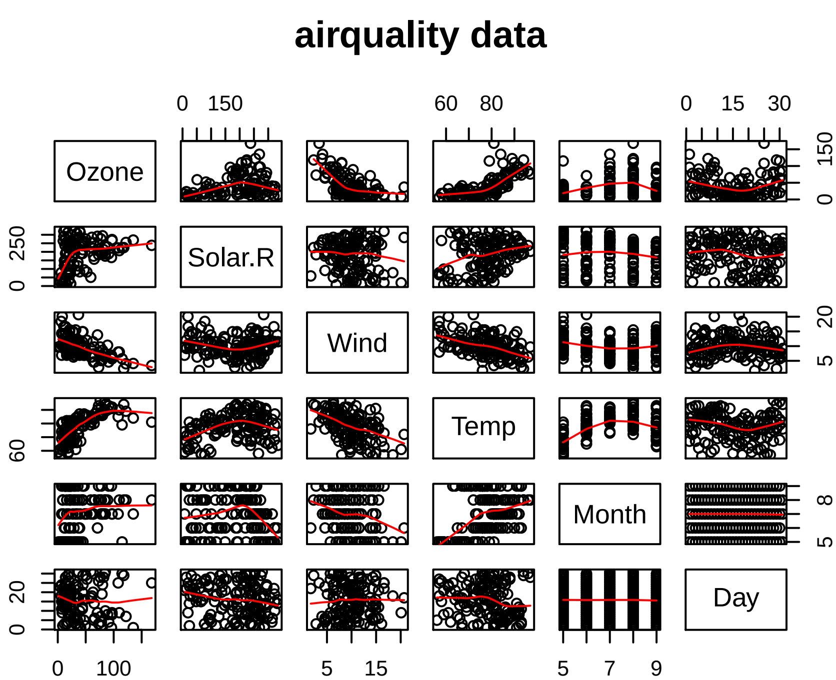
La función `na.omit()` devuelve un objeto sin los elementos `NA`.

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

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

`NA` en data frames:

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



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

1.17. Trabajando con data frames

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

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

- Observemos las primeras filas

```
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

```
str(mtcars) # display the structure of the data frame
```

```
## '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:

```r
mtcars["Mazda RX4",] # using rows and columns names
```

```
##           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
```

```r
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

```r
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)
```

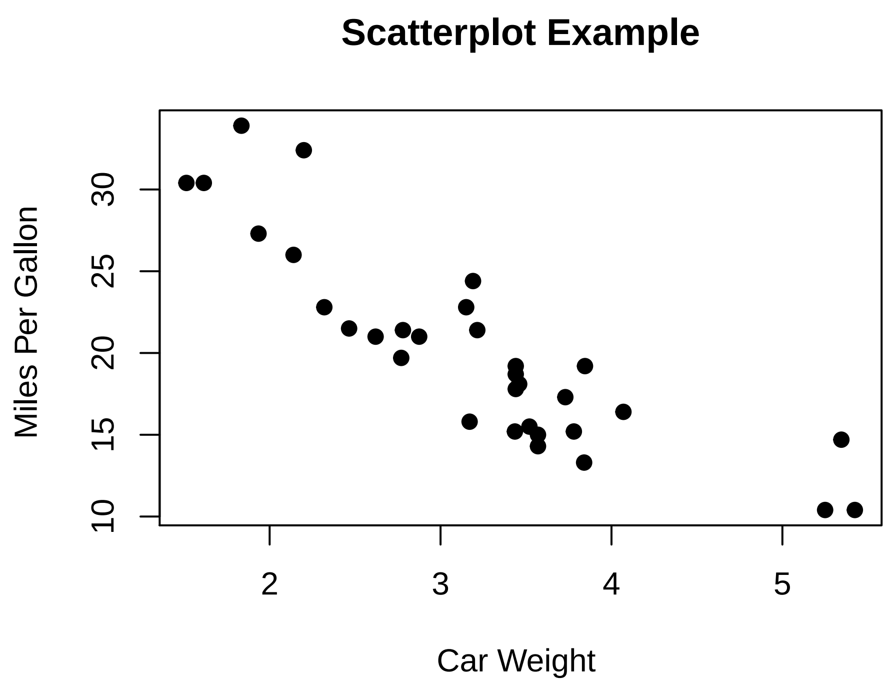
```
##      vars  n   mean     sd median trimmed    mad   min    max  range  skew
## mpg     1 32  20.09   6.03  19.20   19.70   5.41 10.40  33.90  23.50  0.61
## cyl     2 32   6.19   1.79   6.00    6.23   2.97  4.00   8.00   4.00 -0.17
## disp    3 32 230.72 123.94 196.30  222.52 140.48 71.10 472.00 400.90  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
```

2. Análisis de datos básico en R

2.1. Gráficos sencillos

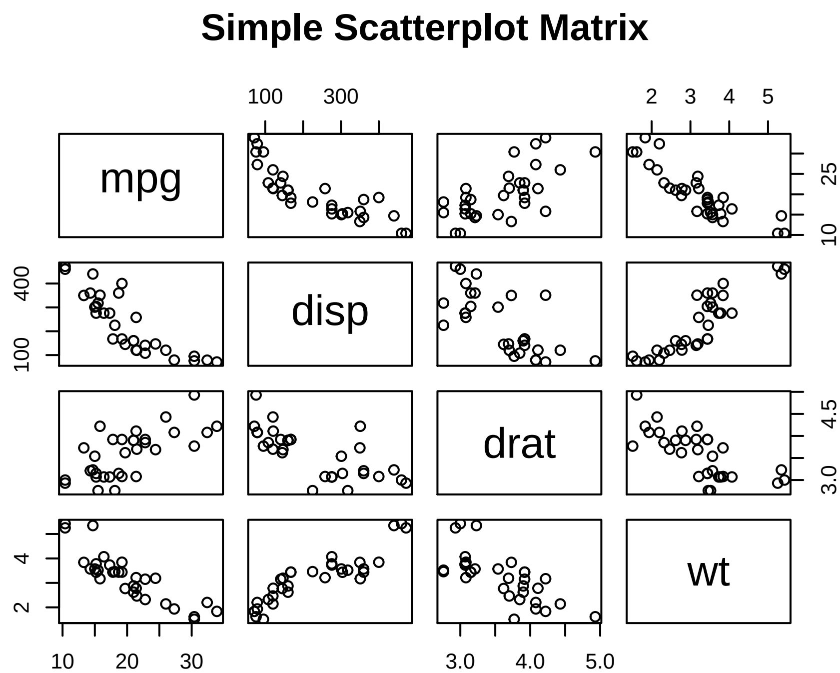
- Scatterplot

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



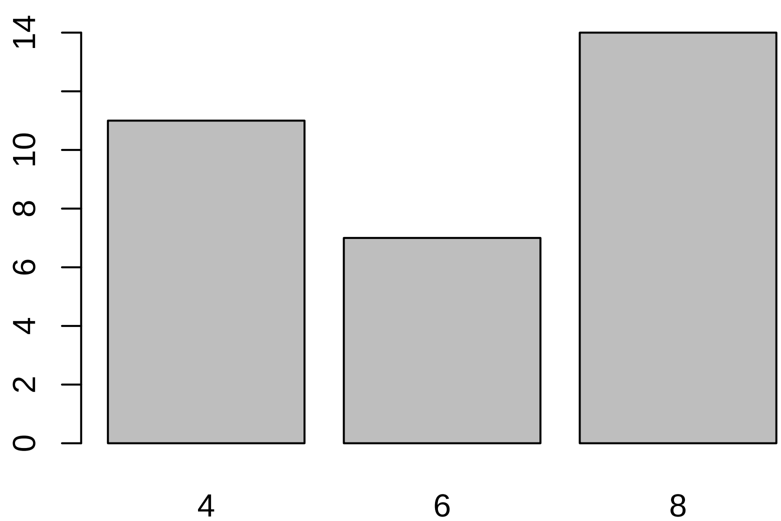
- Matriz scatterplot

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



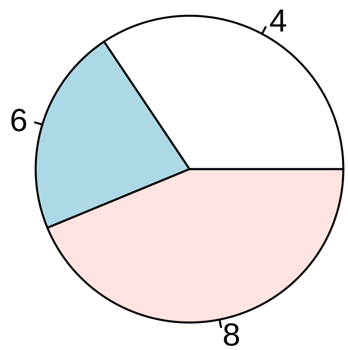
- Barplot o diagrama de barras

```
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
```

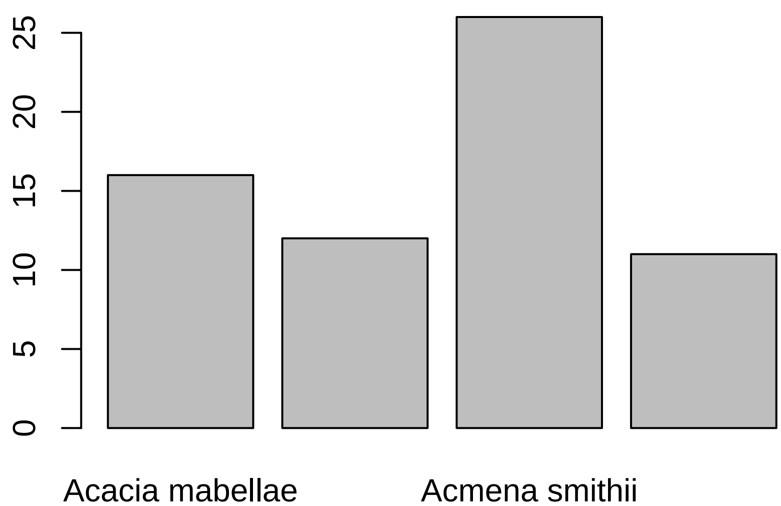
2. El `data.frame` `rainforest` contiene diferentes variables de `species`

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

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

  - **Resultado:**

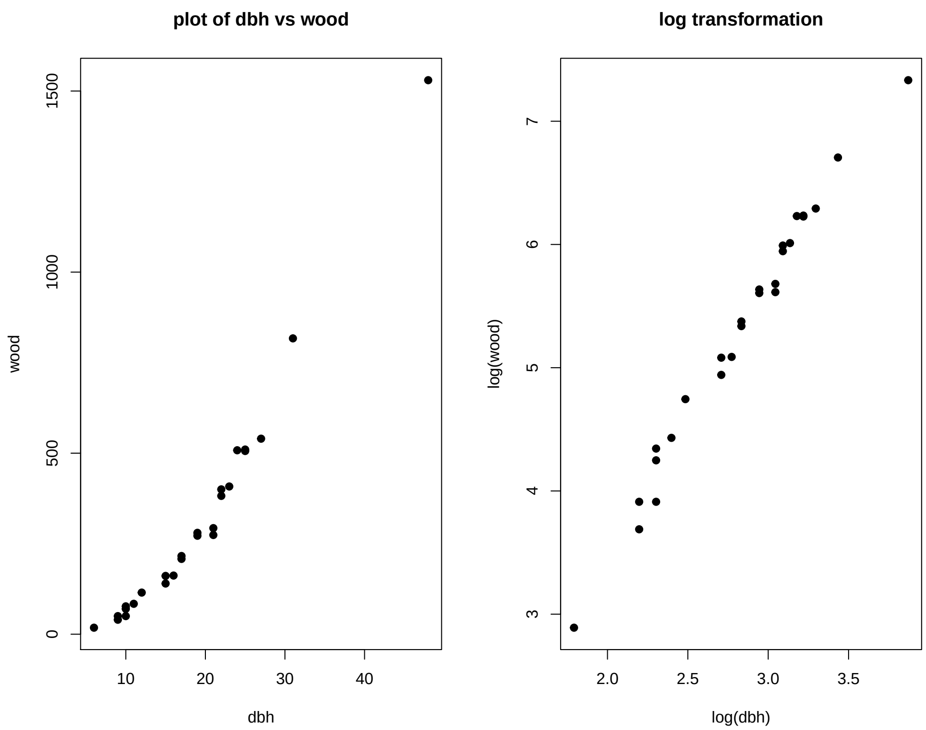
```
##
## Acacia mabellae      C. fraseri   Acmena smithii   B. myrtifolia
##              16              12               26              11
```



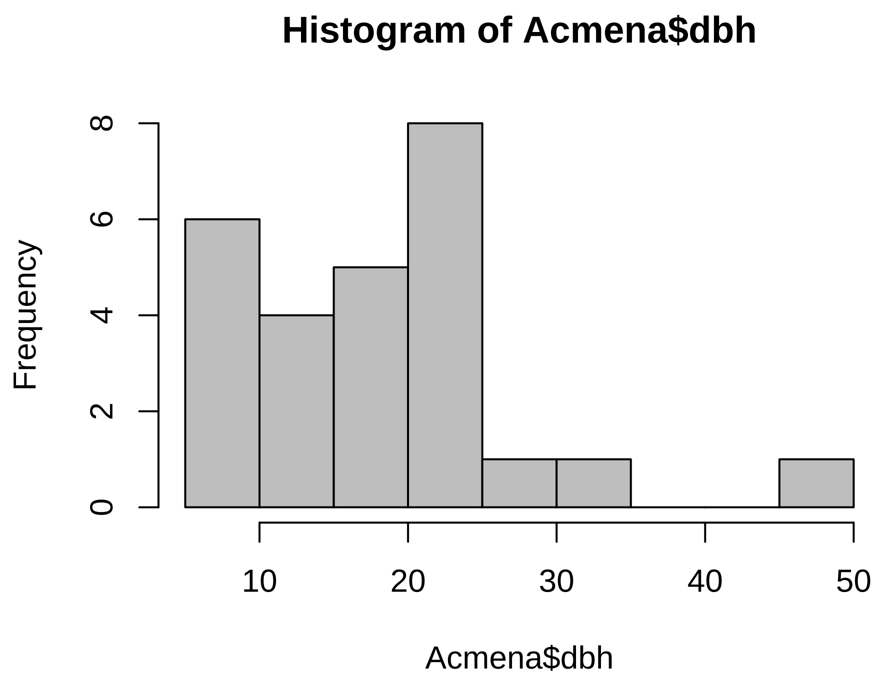
3. El `data.frame` `Acmena` está creado a partir de `rainforest` mediante la función `subset`.

- Realiza un gráfico que relacione la biomasa de la madera (`wood`) y el diámetro a la altura del pecho (`dbh`). Utiliza también 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 números 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

2.2. Scatterplots

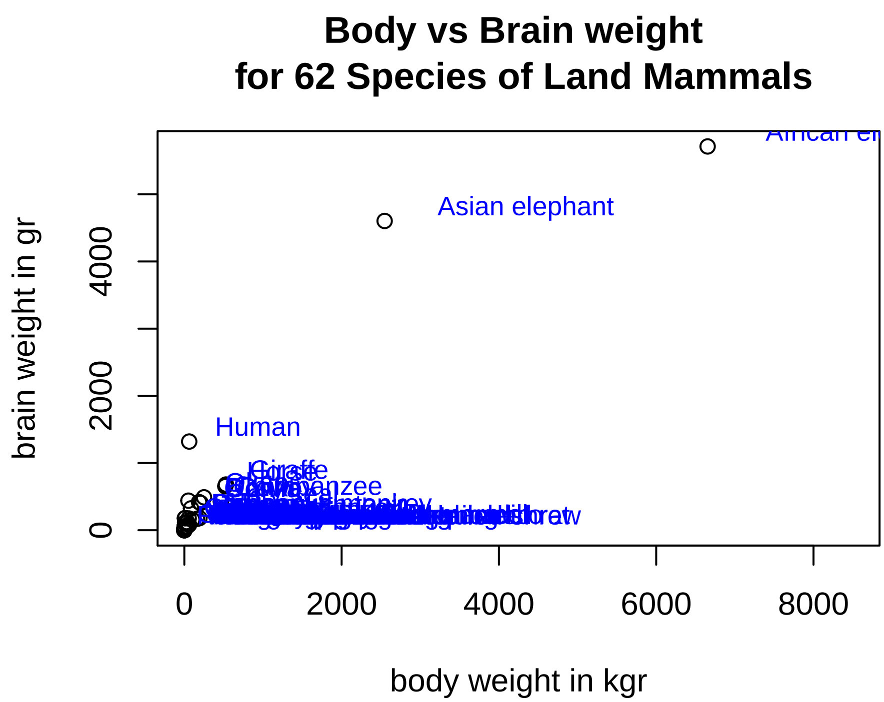
```
library(MASS)
data("mammals")
?mammals
head(mammals)
```

```
##                        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
```

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

```r
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)
```

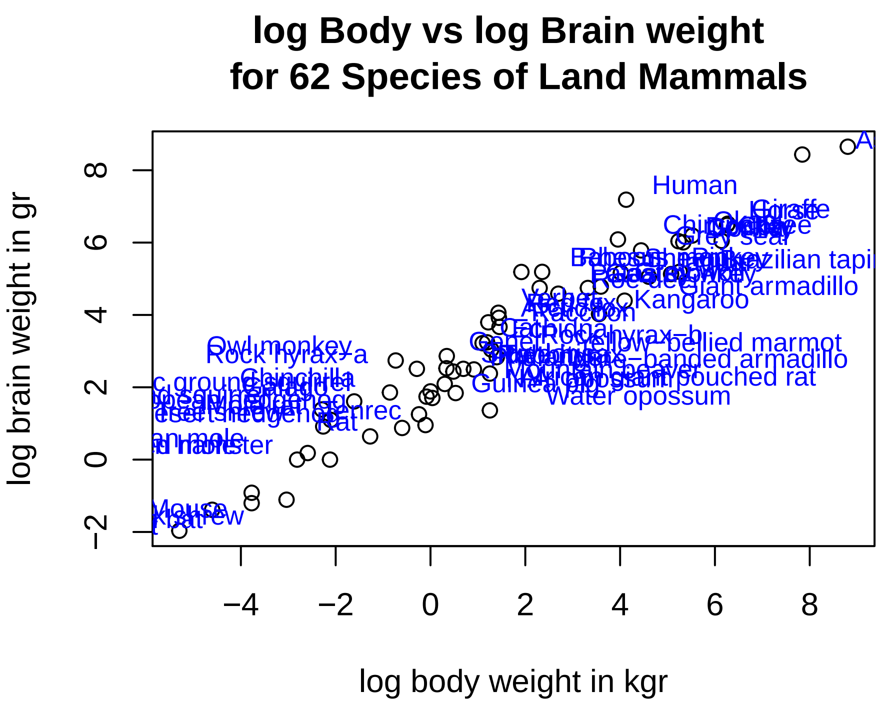


Identificar un punto en el scatterplot

```r
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)
```

2.3. más opciones gráficas

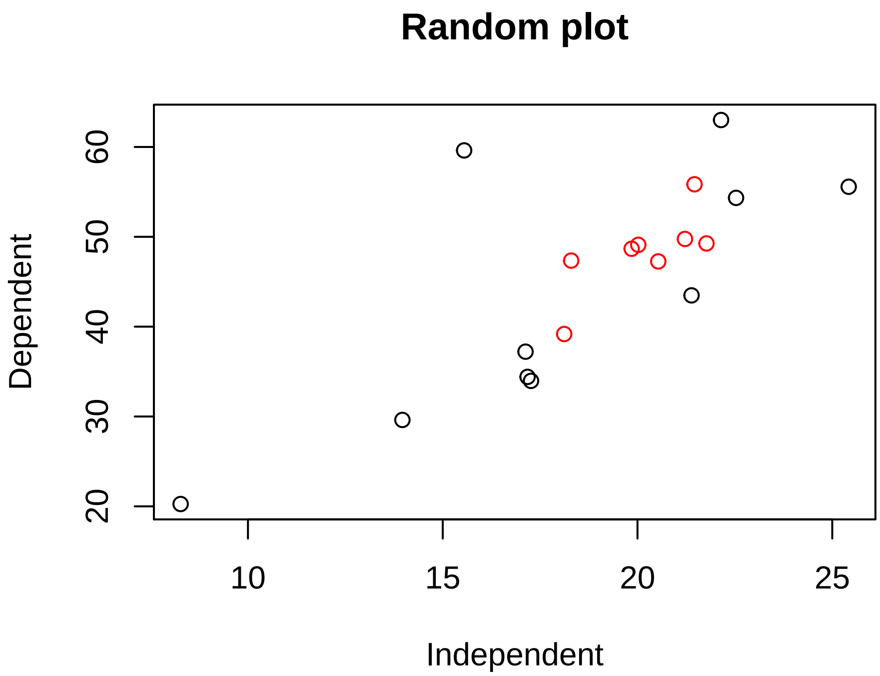
Varios conjuntos de datos en un sólo gráfico

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)
```

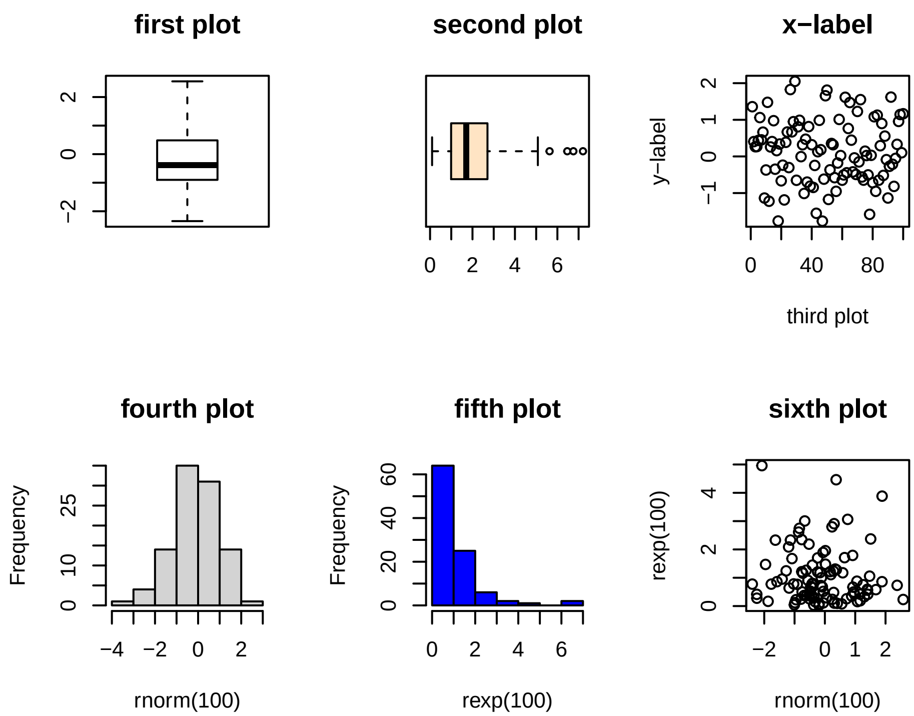


con la leyenda

```
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 sóla 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)
```

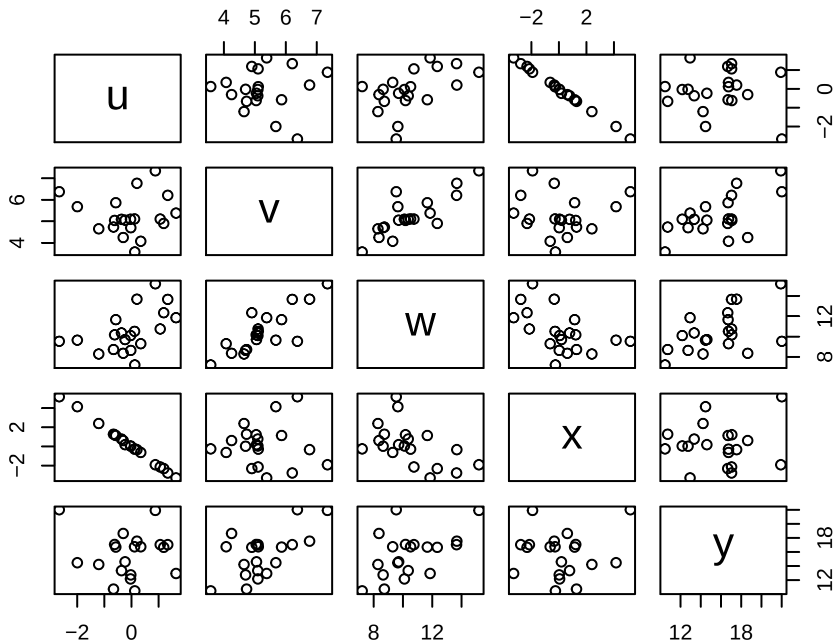
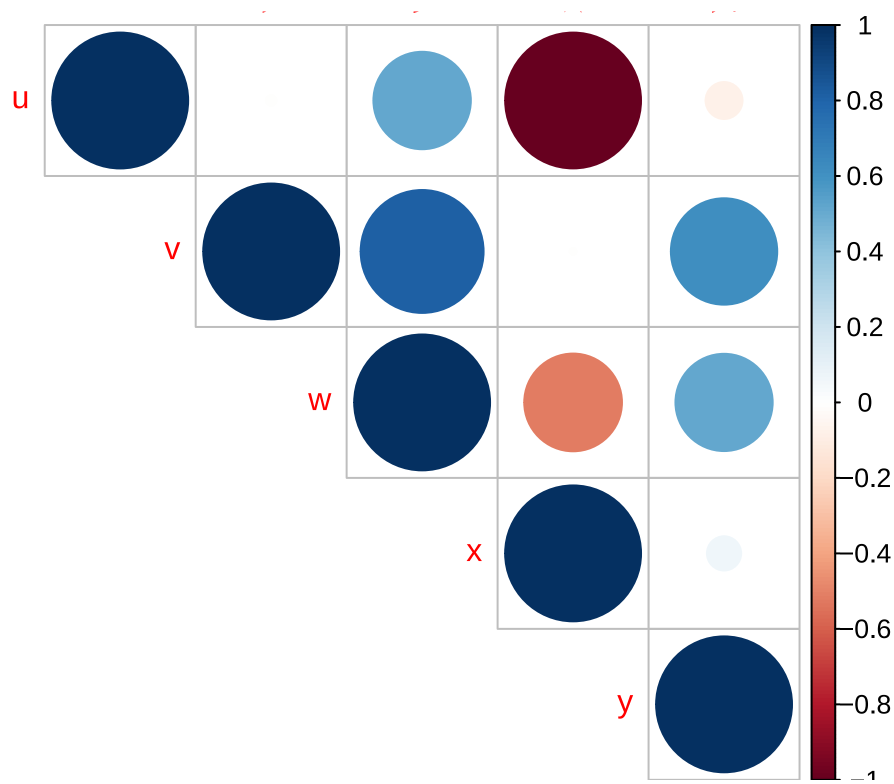


Gráfico de correlaciones

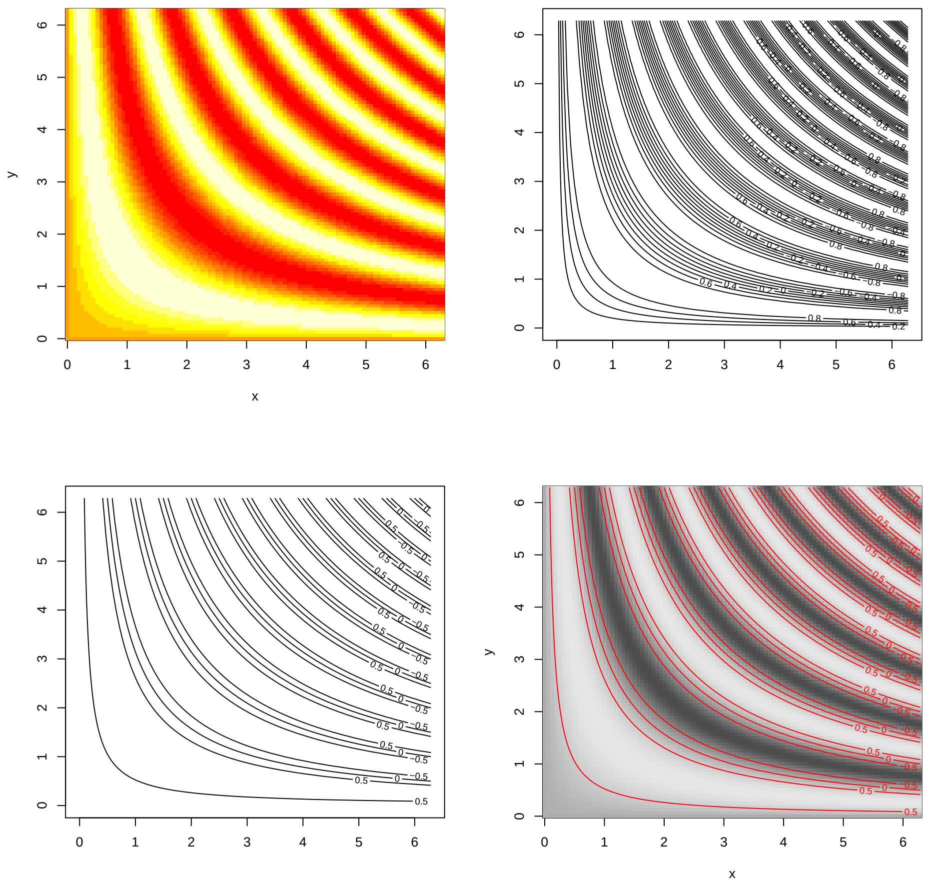
La función `corrplot` de la librería `corrplot` permite visualizar una matriz de correlaciones calculada mediante la función `cor`

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



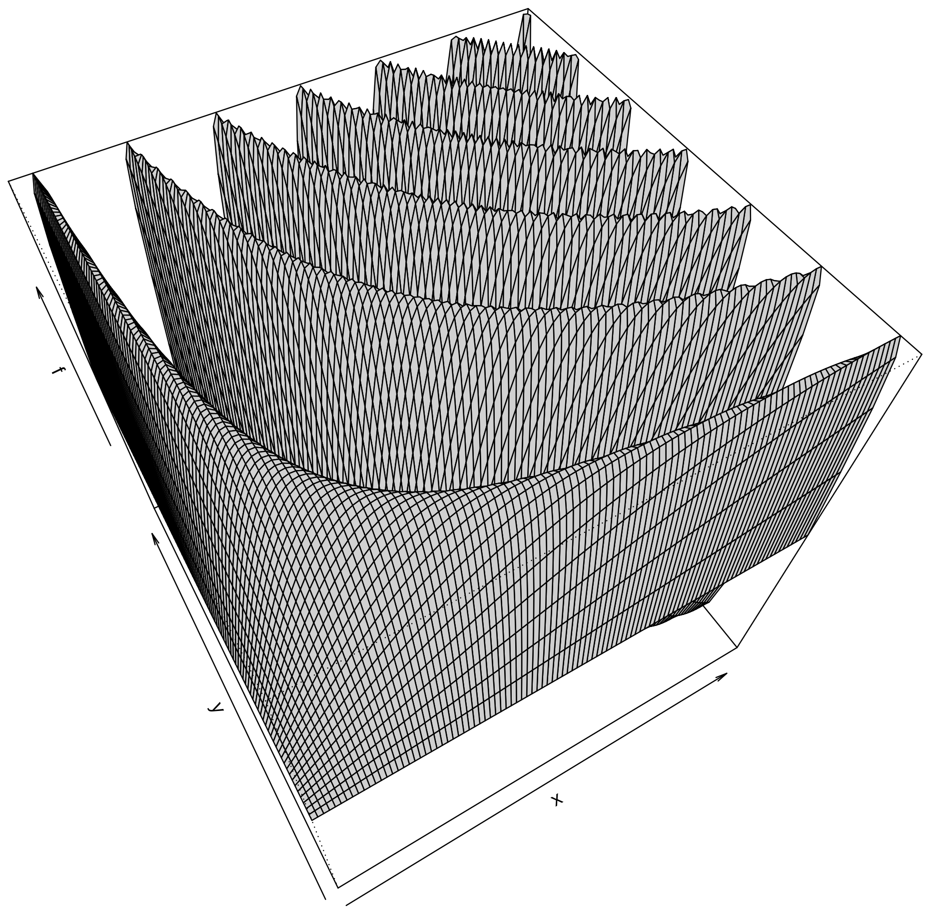
Gráficos de superficies: `image`, `contour` y `persp`

```
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")
```



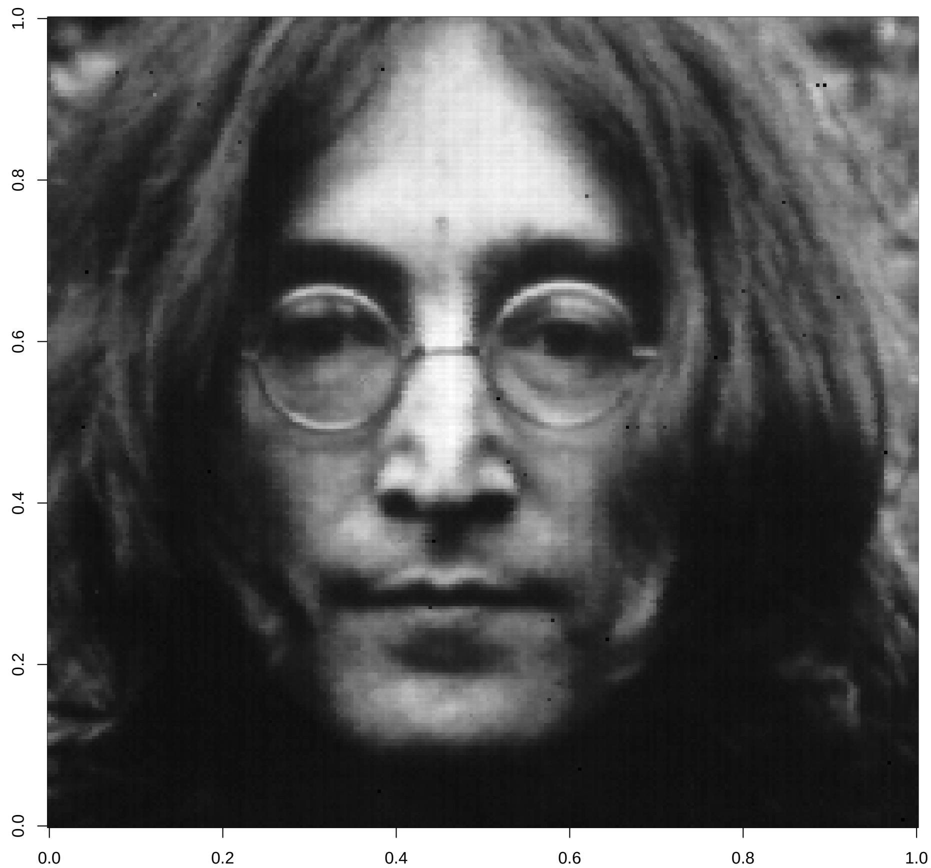
Podemos utilizar la función `persp`

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



o representar imágenes

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



2.4. Tablas de clasificación cruzada o de contigencia

```
library(MASS)
data(quine)
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
## 7     A   M  F0  AL   20
## 8     A   M  F0  AL   22
```

```
## 9      A   M  F1  SL    6
## 10     A   M  F1  SL    6
## 11     A   M  F1  SL   15
## 12     A   M  F1  AL    7
## 13     A   M  F1  AL   14
## 14     A   M  F2  SL    6
## 15     A   M  F2  SL   32
## 16     A   M  F2  SL   53
## 17     A   M  F2  SL   57
## 18     A   M  F2  AL   14
## 19     A   M  F2  AL   16
## 20     A   M  F2  AL   16
## 21     A   M  F2  AL   17
## 22     A   M  F2  AL   40
## 23     A   M  F2  AL   43
## 24     A   M  F2  AL   46
## 25     A   M  F3  AL    8
## 26     A   M  F3  AL   23
## 27     A   M  F3  AL   23
## 28     A   M  F3  AL   28
## 29     A   M  F3  AL   34
## 30     A   M  F3  AL   36
## 31     A   M  F3  AL   38
## 32     A   F  F0  SL    3
## 33     A   F  F0  AL    5
## 34     A   F  F0  AL   11
## 35     A   F  F0  AL   24
## 36     A   F  F0  AL   45
## 37     A   F  F1  SL    5
## 38     A   F  F1  SL    6
## 39     A   F  F1  SL    6
## 40     A   F  F1  SL    9
## 41     A   F  F1  SL   13
## 42     A   F  F1  SL   23
## 43     A   F  F1  SL   25
## 44     A   F  F1  SL   32
## 45     A   F  F1  SL   53
## 46     A   F  F1  SL   54
## 47     A   F  F1  AL    5
## 48     A   F  F1  AL    5
## 49     A   F  F1  AL   11
## 50     A   F  F1  AL   17
## 51     A   F  F1  AL   19
## 52     A   F  F2  SL    8
## 53     A   F  F2  SL   13
## 54     A   F  F2  SL   14
```

```
## 55    A   F  F2  SL   20
## 56    A   F  F2  SL   47
## 57    A   F  F2  SL   48
## 58    A   F  F2  SL   60
## 59    A   F  F2  SL   81
## 60    A   F  F2  AL    2
## 61    A   F  F3  AL    0
## 62    A   F  F3  AL    2
## 63    A   F  F3  AL    3
## 64    A   F  F3  AL    5
## 65    A   F  F3  AL   10
## 66    A   F  F3  AL   14
## 67    A   F  F3  AL   21
## 68    A   F  F3  AL   36
## 69    A   F  F3  AL   40
## 70    N   M  F0  SL    6
## 71    N   M  F0  SL   17
## 72    N   M  F0  SL   67
## 73    N   M  F0  AL    0
## 74    N   M  F0  AL    0
## 75    N   M  F0  AL    2
## 76    N   M  F0  AL    7
## 77    N   M  F0  AL   11
## 78    N   M  F0  AL   12
## 79    N   M  F1  SL    0
## 80    N   M  F1  SL    0
## 81    N   M  F1  SL    5
## 82    N   M  F1  SL    5
## 83    N   M  F1  SL    5
## 84    N   M  F1  SL   11
## 85    N   M  F1  SL   17
## 86    N   M  F1  AL    3
## 87    N   M  F1  AL    4
## 88    N   M  F2  SL   22
## 89    N   M  F2  SL   30
## 90    N   M  F2  SL   36
## 91    N   M  F2  AL    8
## 92    N   M  F2  AL    0
## 93    N   M  F2  AL    1
## 94    N   M  F2  AL    5
## 95    N   M  F2  AL    7
## 96    N   M  F2  AL   16
## 97    N   M  F2  AL   27
## 98    N   M  F3  AL    0
## 99    N   M  F3  AL   30
## 100   N   M  F3  AL   10
```

```
## 101    N   M  F3  AL   14
## 102    N   M  F3  AL   27
## 103    N   M  F3  AL   41
## 104    N   M  F3  AL   69
## 105    N   F  F0  SL   25
## 106    N   F  F0  AL   10
## 107    N   F  F0  AL   11
## 108    N   F  F0  AL   20
## 109    N   F  F0  AL   33
## 110    N   F  F1  SL    5
## 111    N   F  F1  SL    7
## 112    N   F  F1  SL    0
## 113    N   F  F1  SL    1
## 114    N   F  F1  SL    5
## 115    N   F  F1  SL    5
## 116    N   F  F1  SL    5
## 117    N   F  F1  SL    5
## 118    N   F  F1  SL    7
## 119    N   F  F1  SL   11
## 120    N   F  F1  SL   15
## 121    N   F  F1  AL    5
## 122    N   F  F1  AL   14
## 123    N   F  F1  AL    6
## 124    N   F  F1  AL    6
## 125    N   F  F1  AL    7
## 126    N   F  F1  AL   28
## 127    N   F  F2  SL    0
## 128    N   F  F2  SL    5
## 129    N   F  F2  SL   14
## 130    N   F  F2  SL    2
## 131    N   F  F2  SL    2
## 132    N   F  F2  SL    3
## 133    N   F  F2  SL    8
## 134    N   F  F2  SL   10
## 135    N   F  F2  SL   12
## 136    N   F  F2  AL    1
## 137    N   F  F3  AL    1
## 138    N   F  F3  AL    9
## 139    N   F  F3  AL   22
## 140    N   F  F3  AL    3
## 141    N   F  F3  AL    3
## 142    N   F  F3  AL    5
## 143    N   F  F3  AL   15
## 144    N   F  F3  AL   18
## 145    N   F  F3  AL   22
## 146    N   F  F3  AL   37
```

```r
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
```

2.5. cálculos sobre tablas de contigencia

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

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

```
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
```

```
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
```

2.6. 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"))
```

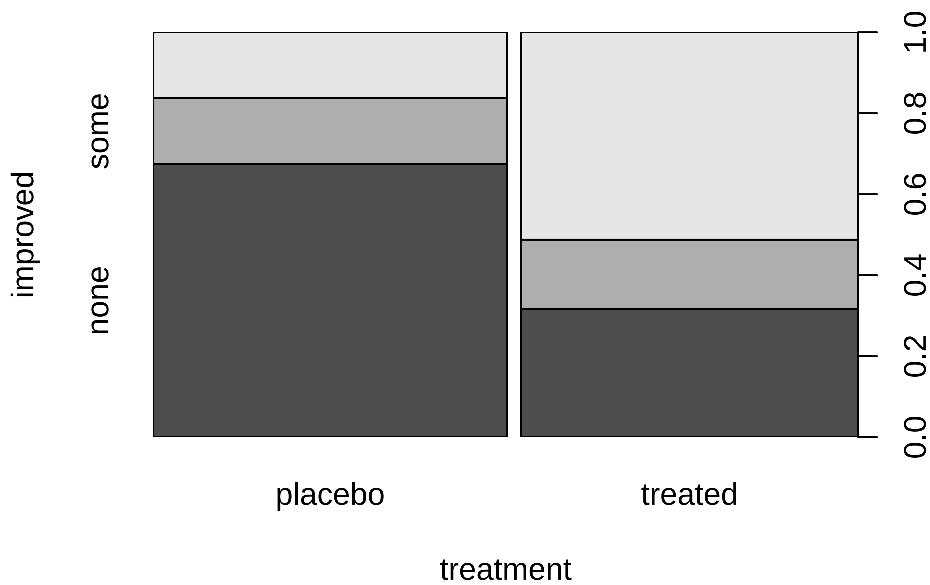
Tabla de contigencia

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

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

De manera grfica,

```
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"))
```

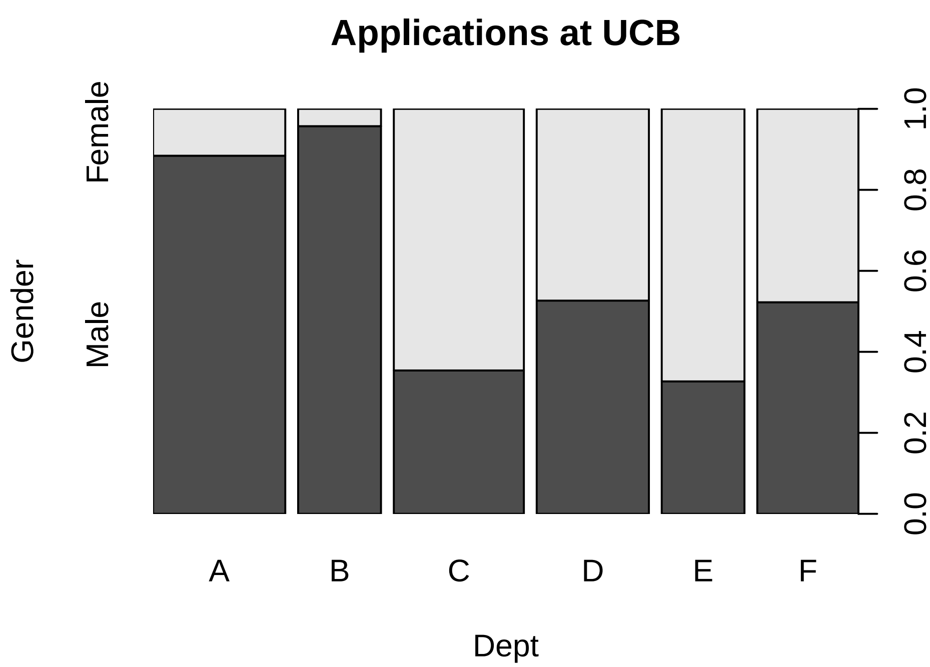
```
##      Gender     Male              Female
##      Admit  Admitted Rejected Admitted Rejected
## Dept
## A               0.62     0.38     0.82     0.18
## B               0.63     0.37     0.68     0.32
## C               0.37     0.63     0.34     0.66
## D               0.33     0.67     0.35     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)
```

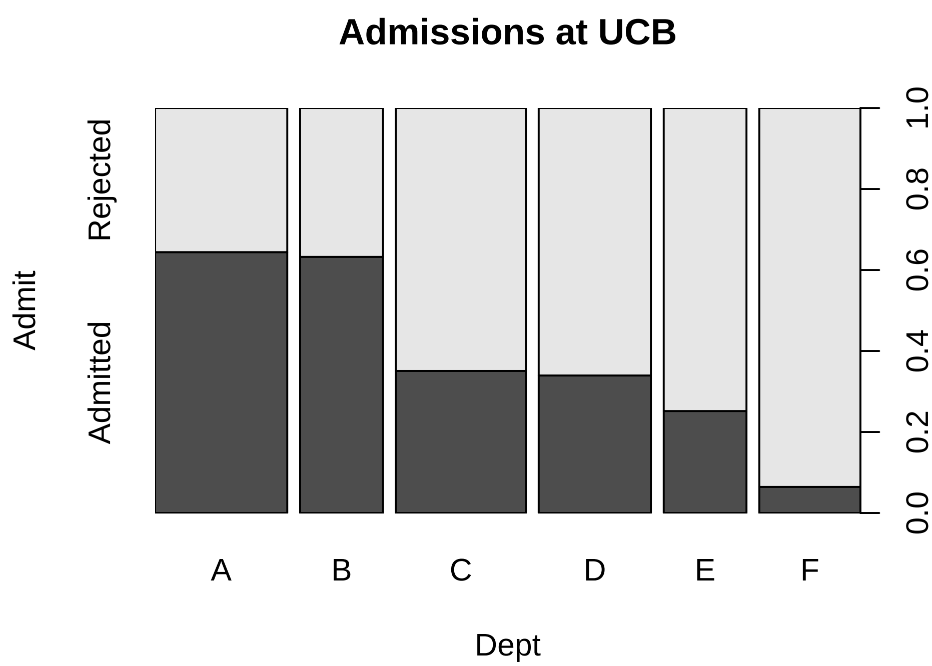
```
##           Gender
## Admit      Male Female
##   Admitted 1198    557
##   Rejected 1493   1278
```

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 sólo 557 (30.4 %) fueron admitidas. Se podría por tanto concluir que los hombres tienes tasas de admisión mayores que las mujeres. Wikipedia: Gender Bias UC Berkeley. See animation at link

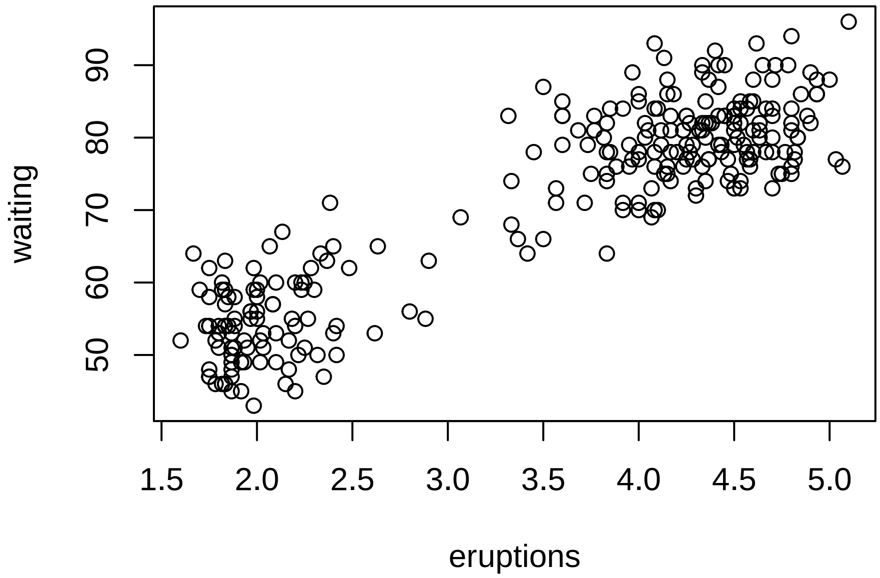
2.7. Datos cuantitativos

```
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 geyse Old Faithful en el parque nacional de Yellowstone, EEUU.

```
plot(faithful)
```



2.7.1. Distribuciones de frecuencias

Vamos a utilizar el conjunto de datos `faithful`, para ilustrar el concepto de distribución de frecuencias que consistirá en crear una series de categorías o intervalos, en los que contaremos el número de observaciones en cada categoría.

```
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
```

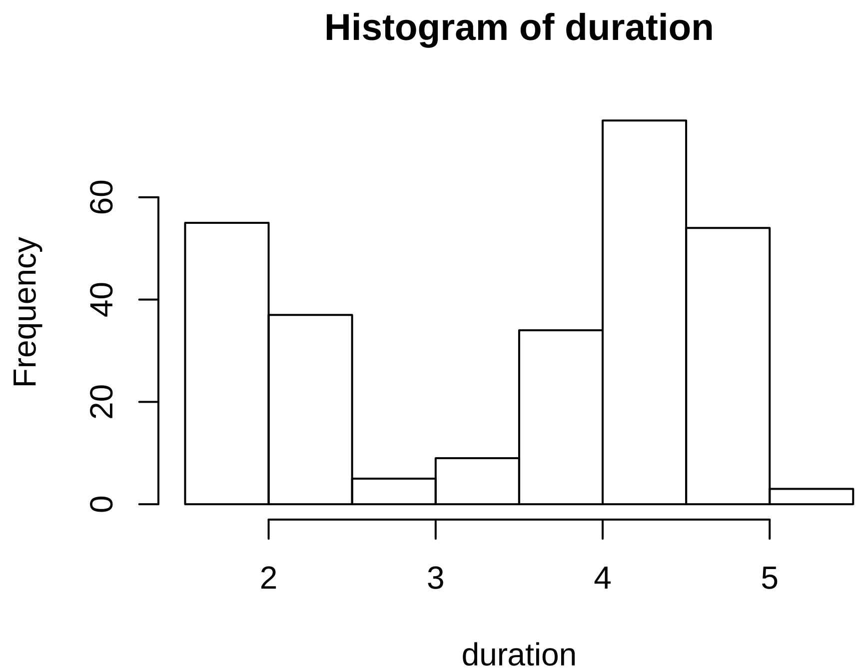
```
## duration.cut
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      41       5       7      30      73      61       4
```

Con `hist` podemos realizarlo de manera automática:

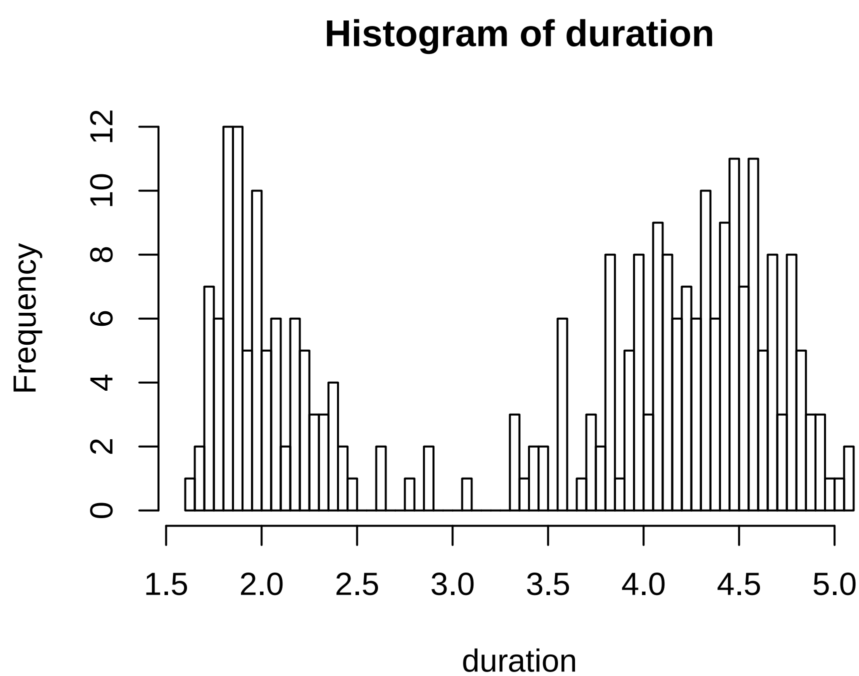
```
freq <- hist(duration)
freq
```

```
## $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)
```




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

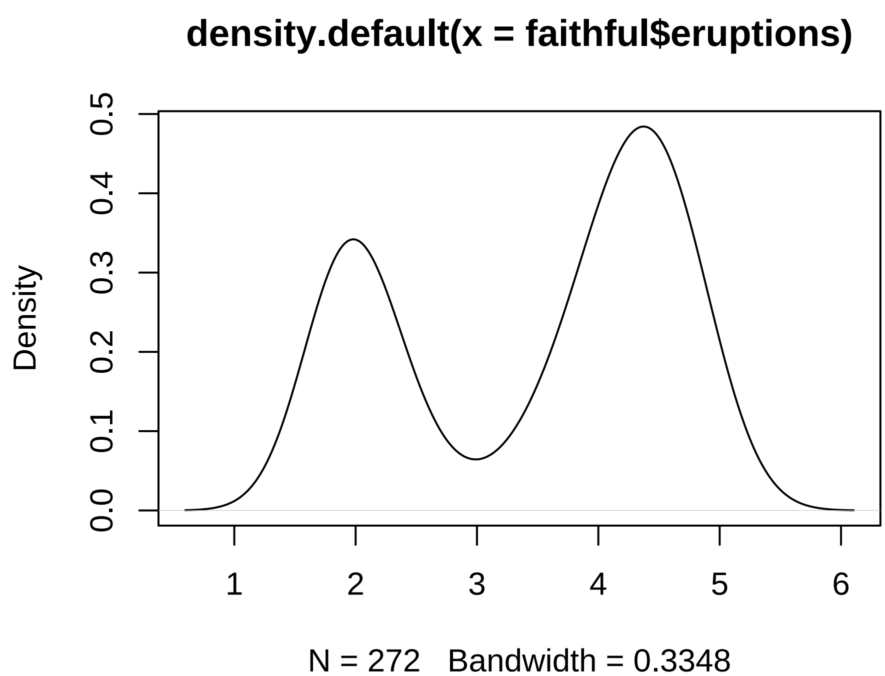


Estimación de densidad construye una estimación dada una distribucion de probabilidad para una muestra dada.

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

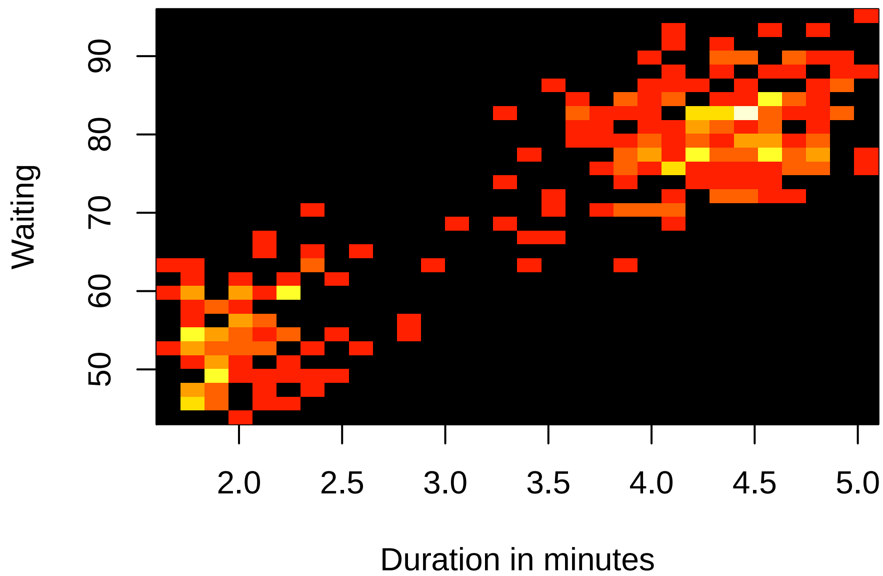
```
##
## Call:
##  density.default(x = faithful$eruptions)
##
## Data: faithful$eruptions (272 obs.); Bandwidth 'bw' = 0.3348
##
##        x                 y
##  Min.   :0.5957   Min.   :0.0002262
##  1st Qu.:1.9728   1st Qu.:0.0514171
##  Median :3.3500   Median :0.1447010
##  Mean   :3.3500   Mean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

```
plot(d)
```



En dos dimensiones:

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




```r
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 )
```

```r
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)
```

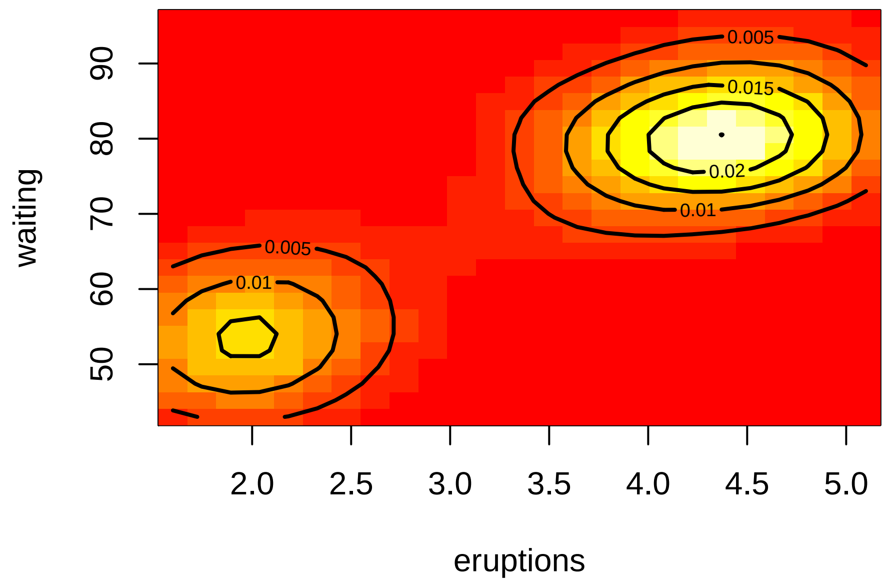
```
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      92      97     104     134     207     268     272
```

```
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

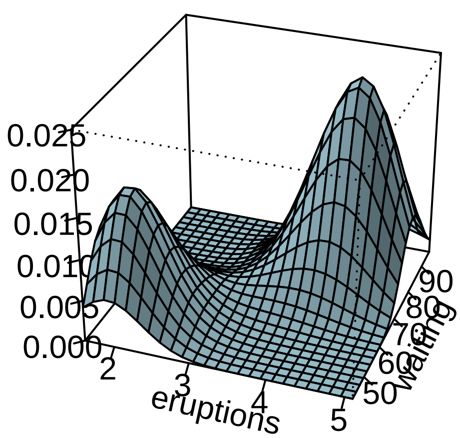
```
data("faithful")
attach(faithful)
Dens2d<-kde2d(eruptions,waiting)
image(Dens2d,xlab="eruptions",ylab="waiting")
contour(Dens2d,add=TRUE,col="black",lwd=2,nlevels=5)
```




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

Gráficos `persp`

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



3. Introducción a la programación básico con R

3.1. 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
```

3.2. 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
```

3.3. `if` statements

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

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

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

3.4. `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
```

3.5. `while`

3.6. 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.

3.6.1. `for`

Sintaxis

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

```
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`.

```
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

```
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:

```
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
```

3.7. `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
```

4. Case studies

4.1. The Forbes 2000 Ranking of the World's Biggest Companies (Year 2004)

The data handling and manipulation techniques explained will be illustrated by means of a data set of 2000 world leading companies, the Forbes 2000 list for the year 2004 collected by Forbes Magazine. This list is originally available from `www.forbes.com`

Here we show a subset of the data set:

```r
library("HSAUR2")
data("Forbes2000")
```

rank	name	country	category	sales	profits	assets	marketv
1	Citigroup	United States	Banking	94.71	17.85	1264.03	25
2	General Electric	United States	Conglomerates	134.19	15.59	626.93	32
3	American Intl Group	United States	Insurance	76.66	6.46	647.66	19
4	ExxonMobil	United States	Oil & gas operations	222.88	20.96	166.99	27
5	BP	United Kingdom	Oil & gas operations	232.57	10.27	177.57	17
6	Bank of America	United States	Banking	49.01	10.81	736.45	11

The data consists of 2000 observations on the following 8 variables.

- `rank`: the ranking of the company.
- `name`: the name of the company.
- `country`: a factor giving the country the company is situated in.
- `category`: a factor describing the products the company produces.
- `sales`: the amount of sales of the company in billion USD.
- `profits`: the profit of the company in billion USD.
- `assets`: the assets of the company in billion USD.
- `marketvalue`: the market value of the company in billion USD.

Types of variables

`R` command

```r
str(Forbes2000)
```

```
## 'data.frame':    2000 obs. of  8 variables:
##  $ rank       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ name       : chr  "Citigroup" "General Electric" "American Intl Group" "ExxonMobil" ..
##  $ country    : Factor w/ 61 levels "Africa","Australia",..: 60 60 60 60 56 60 56 28 60 6
##  $ category   : Factor w/ 27 levels "Aerospace & defense",..: 2 6 16 19 19 2 2 8 9 20 ...
##  $ sales      : num  94.7 134.2 76.7 222.9 232.6 ...
##  $ profits    : num  17.85 15.59 6.46 20.96 10.27 ...
##  $ assets     : num  1264 627 648 167 178 ...
##  $ marketvalue: num  255 329 195 277 174 ...
```

Factor levels

Nominal measurements are represented by factor variables in R, such as the country of the company or the category of the business segment.

A factor in R is divided into levels

How many countries are on the top 2000 ranking?

R command

```
nlevels(Forbes2000[,"country"])
```

```
## [1] 61
```

Which countries?

R command

```
levels(Forbes2000[,"country"])
```

```
##  [1] "Africa"                        "Australia"
##  [3] "Australia/ United Kingdom"     "Austria"
##  [5] "Bahamas"                       "Belgium"
##  [7] "Bermuda"                       "Brazil"
##  [9] "Canada"                        "Cayman Islands"
## [11] "Chile"                         "China"
## [13] "Czech Republic"                "Denmark"
## [15] "Finland"                       "France"
## [17] "France/ United Kingdom"        "Germany"
## [19] "Greece"                        "Hong Kong/China"
## [21] "Hungary"                       "India"
## [23] "Indonesia"                     "Ireland"
## [25] "Islands"                       "Israel"
## [27] "Italy"                         "Japan"
## [29] "Jordan"                        "Kong/China"
## [31] "Korea"                         "Liberia"
## [33] "Luxembourg"                    "Malaysia"
## [35] "Mexico"                        "Netherlands"
## [37] "Netherlands/ United Kingdom"   "New Zealand"
## [39] "Norway"                        "Pakistan"
## [41] "Panama/ United Kingdom"        "Peru"
## [43] "Philippines"                   "Poland"
## [45] "Portugal"                      "Russia"
## [47] "Singapore"                     "South Africa"
## [49] "South Korea"                   "Spain"
## [51] "Sweden"                        "Switzerland"
```

```
## [53] "Taiwan"                          "Thailand"
## [55] "Turkey"                          "United Kingdom"
## [57] "United Kingdom/ Australia"       "United Kingdom/ Netherlands"
## [59] "United Kingdom/ South Africa" "United States"
## [61] "Venezuela"
```

And in the top 20?

R commands

```
top20 <- droplevels(subset(Forbes2000,rank<=20))
levels(top20[,"country"])
```

```
## [1] "France"                        "Japan"
## [3] "Netherlands"                   "Netherlands/ United Kingdom"
## [5] "Switzerland"                   "United Kingdom"
## [7] "United States"
```

As a simple summary statistic, the frequencies of the levels of such a factor variable can be found from

```
table(top20[,"country"])
```

```
##
##                    France                       Japan
##                         2                           1
##               Netherlands Netherlands/ United Kingdom
##                         1                           1
##               Switzerland              United Kingdom
##                         1                           3
##             United States
##                        11
```

Which type of companies?

```
levels(Forbes2000[,"category"])
```

```
##  [1] "Aerospace & defense"             "Banking"
##  [3] "Business services & supplies"    "Capital goods"
##  [5] "Chemicals"                       "Conglomerates"
##  [7] "Construction"                    "Consumer durables"
##  [9] "Diversified financials"          "Drugs & biotechnology"
## [11] "Food drink & tobacco"            "Food markets"
```

```
## [13] "Health care equipment & services" "Hotels restaurants & leisure"
## [15] "Household & personal products"    "Insurance"
## [17] "Materials"                        "Media"
## [19] "Oil & gas operations"             "Retailing"
## [21] "Semiconductors"                   "Software & services"
## [23] "Technology hardware & equipment"  "Telecommunications services"
## [25] "Trading companies"                "Transportation"
## [27] "Utilities"
```

How many of each category?

```
table(Forbes2000[,"category"])
```

```
##
##              Aerospace & defense                          Banking
##                               19                              313
##      Business services & supplies                    Capital goods
##                               70                               53
##                        Chemicals                    Conglomerates
##                               50                               31
##                     Construction                Consumer durables
##                               79                               74
##            Diversified financials            Drugs & biotechnology
##                              158                               45
##              Food drink & tobacco                     Food markets
##                               83                               33
## Health care equipment & services      Hotels restaurants & leisure
##                               65                               37
##     Household & personal products                        Insurance
##                               44                              112
##                        Materials                            Media
##                               97                               61
##              Oil & gas operations                        Retailing
##                               90                               88
##                   Semiconductors              Software & services
##                               26                               31
##   Technology hardware & equipment      Telecommunications services
##                               59                               67
##                Trading companies                   Transportation
##                               25                               80
##                        Utilities
##                              110
```

A simple summary statistics such as the mean, median, quantiles and range can be found from continuous variables such as `sales`

R command

```
summary(Forbes2000[,"sales"])
```

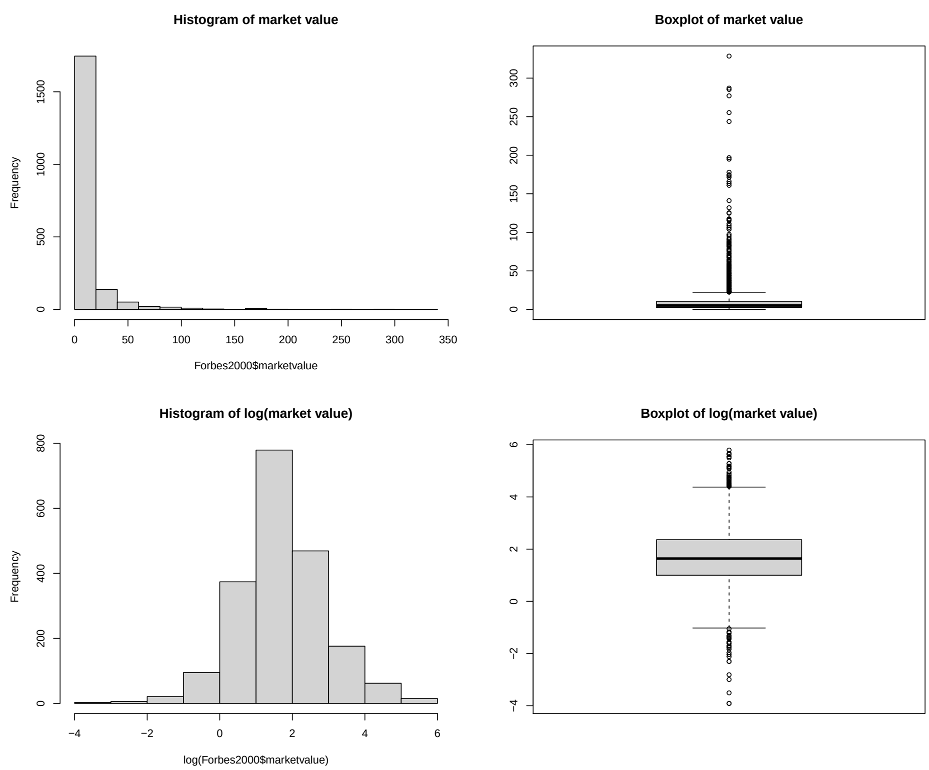
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   2.018   4.365   9.697   9.548 256.300
```

Simple Graphics

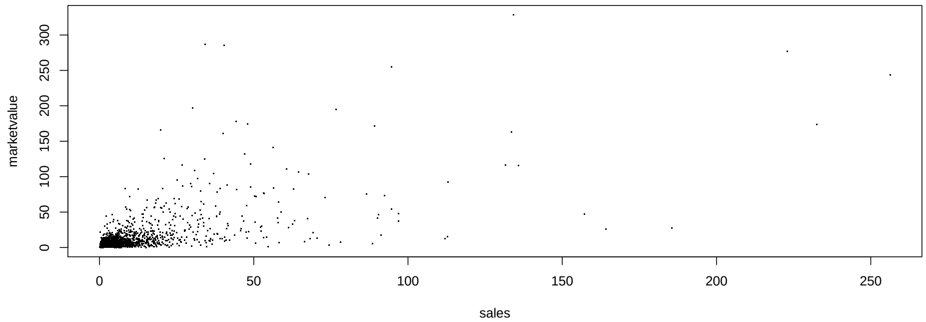
Chambers et al. (1983), "there is no statistical tool that is as powerful as a well chosen graph"

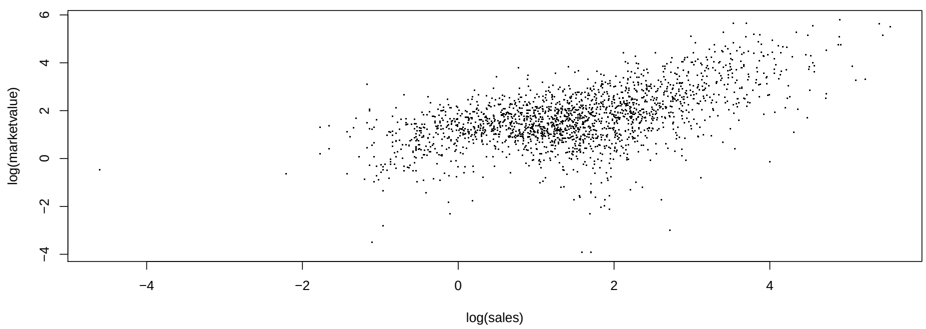
Histograms and boxplots

```
layout(matrix(1:4, nrow = 2,ncol=2))
hist(Forbes2000$marketvalue, col="lightgrey",main="Histogram of market value")
hist(log(Forbes2000$marketvalue),col="lightgrey",main="Histogram of log(market value)")
boxplot(Forbes2000$marketvalue, col="lightgrey",main="Boxplot of market value")
boxplot(log(Forbes2000$marketvalue),col="lightgrey",main="Boxplot of log(market value)")
```



Scatterplots to visualize the relationship betwen variables

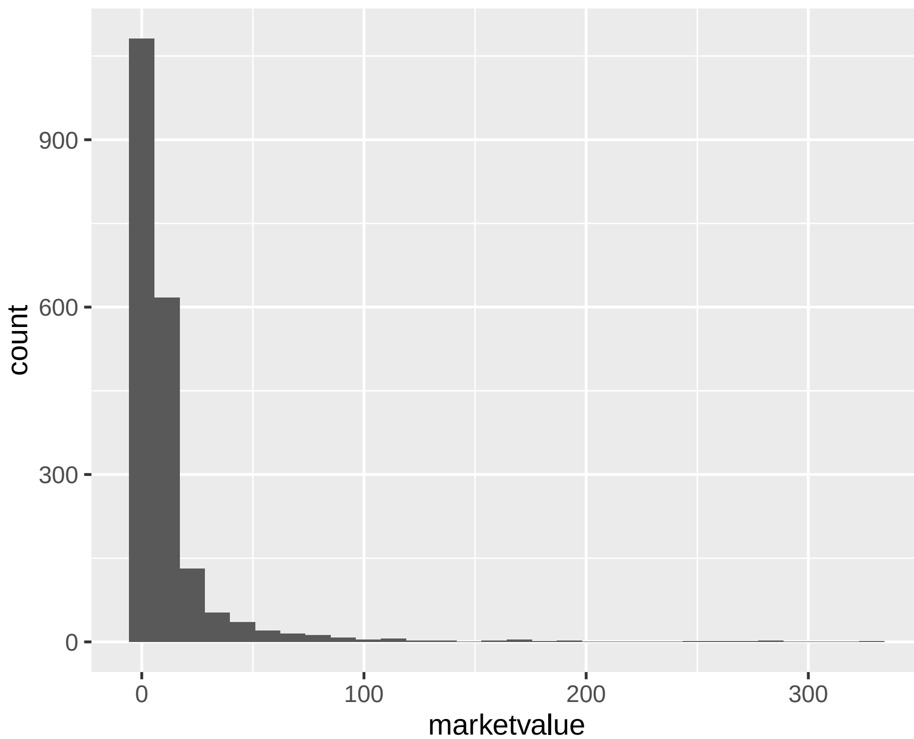




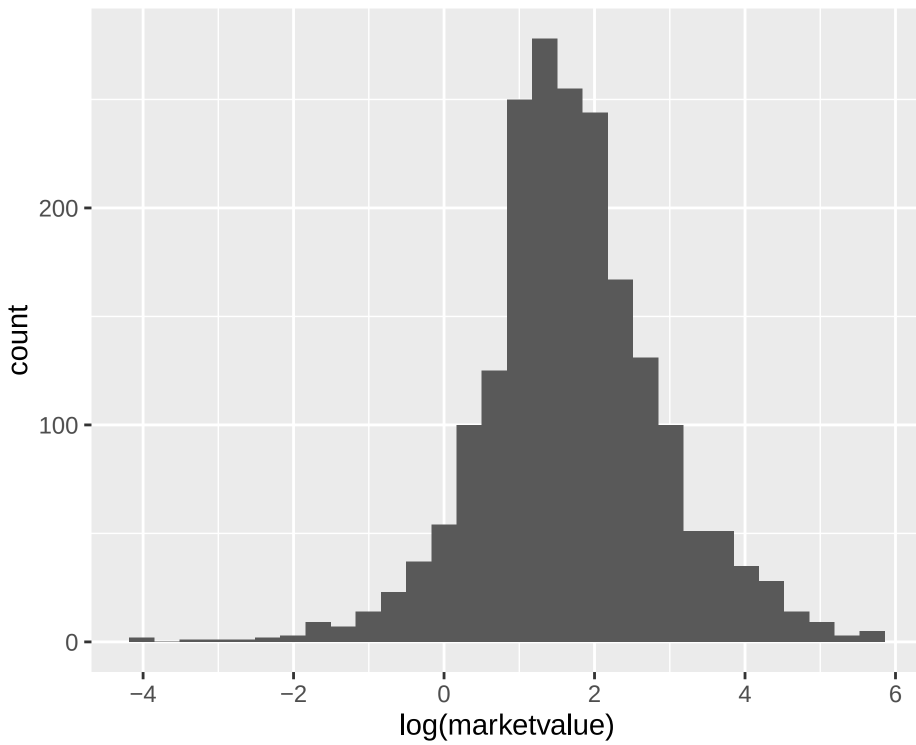
Cool Graphics

Using the `ggplot2` library

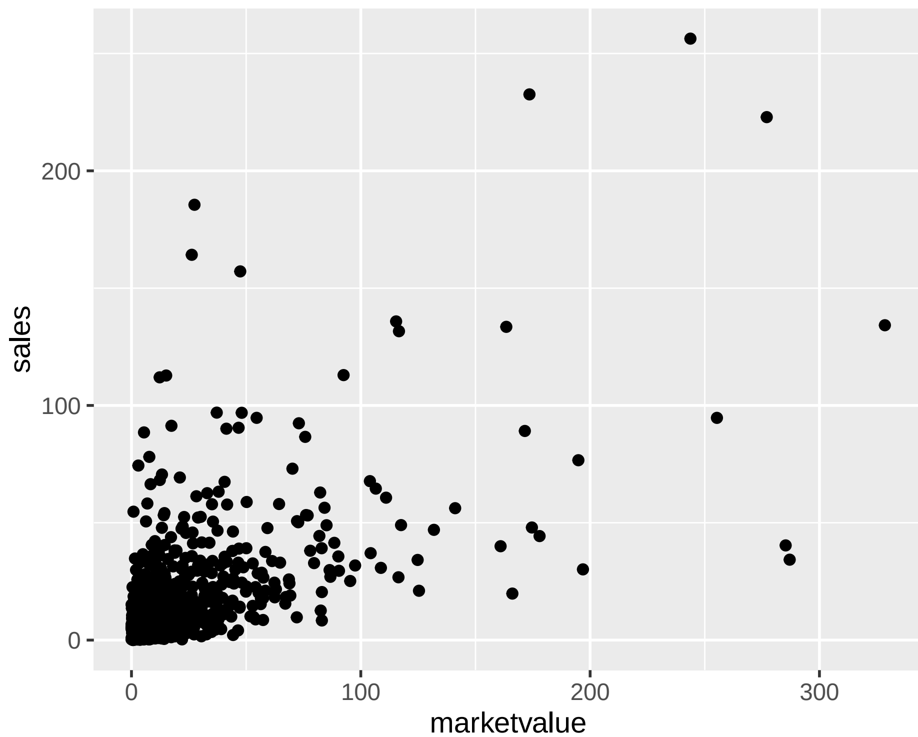
```
library(ggplot2)
#?qplot
qplot(marketvalue,data = Forbes2000)
```



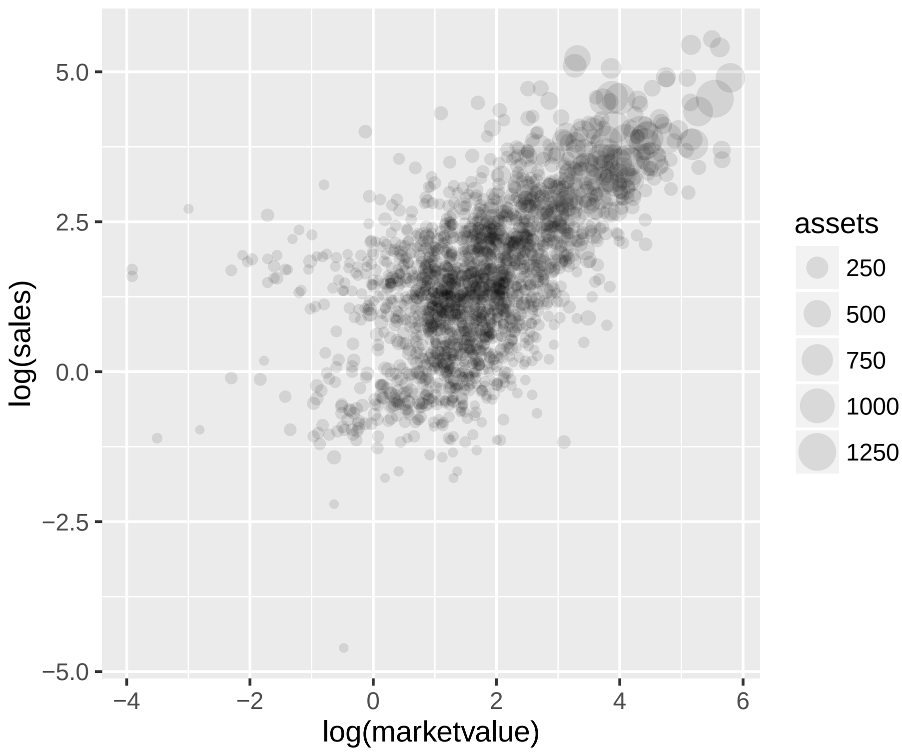

```
qplot(log(marketvalue), data = Forbes2000)
```



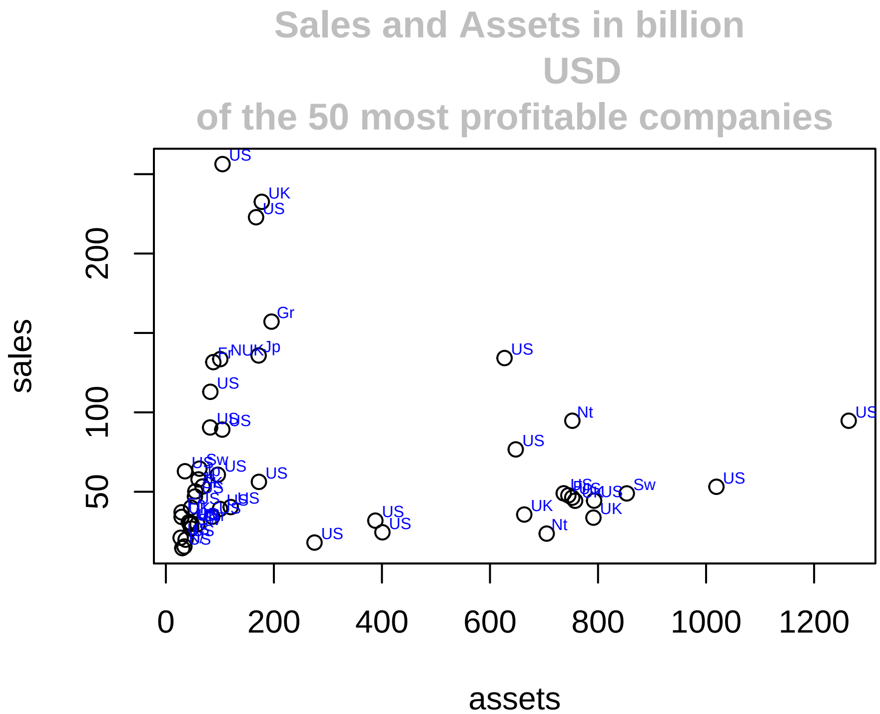

```
qplot(marketvalue,sales, data=Forbes2000)
```




```
qplot(log(marketvalue),log(sales),size=assets,alpha = I(0.1),data=Forbes2000)
```



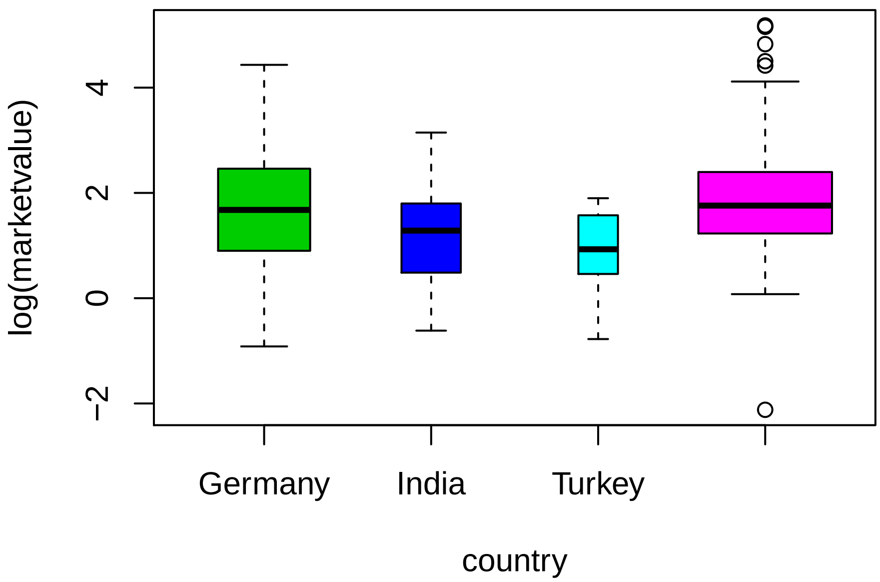

```
library(calibrate)
profits_all = na.omit(Forbes2000$profits)  # all_profts without No data
order_profits = order(profits_all)      # index of the profitable companies
                                        #    in decreasing order
top_50 = rev(order_profits)[1:50]       # top 50 profitable companies

sales = Forbes2000$sales[top_50]        # sales of the 50 top profitable companies
assets = Forbes2000$assets[top_50]      # assets of the 50 top profitable companies
countries = Forbes2000$country[top_50]  # countries where the 50 top profitable
                                        #    companies are found

plot(assets,sales,pch =1)
textxy(assets,sales, abbreviate(countries,2),col = "blue",cex=0.5)  # used to put the
                                                                     # countries where the co
title(main = "Sales and Assets in billion
             USD \n of the 50 most profitable companies ", col.main = "gray")
```



Graphics by factor

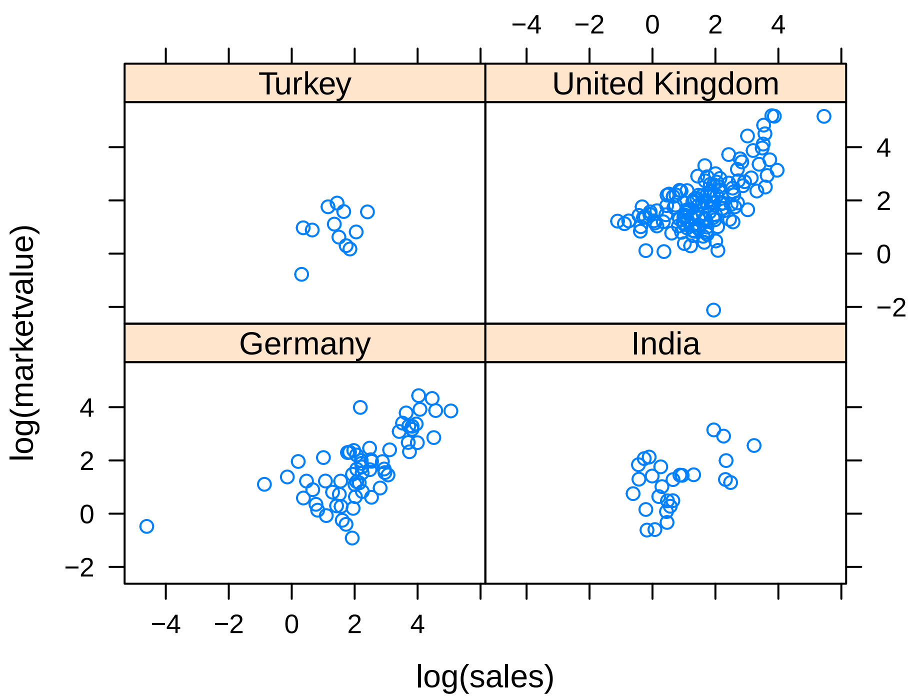
Boxplots of the logarithms of the market value for four selected countries, the width of the boxes is proportional to the square roots of the number of companies.

```
tmp <- subset(Forbes2000,
         country %in% c("United Kingdom", "Germany",
                         "India", "Turkey"))
tmp$country <- tmp$country[,drop = TRUE]
plot(log(marketvalue) ~ country, data = tmp, col = 3:6,
        ylab = "log(marketvalue)", varwidth = TRUE)
```



Scatterplots by country

```
library(lattice)
xyplot(log(marketvalue)~log(sales)|country,data=tmp)
```



4.2. Malignant Melanoma in the USA

Fisher and Belle (1993) report mortality rates due to malignant melanoma of the skin for white males during the period 1950-1969, for each state on the US mainland.

```
data("USmelanoma",package="HSAUR2")
```

A data consists of 48 observations on the following 5 variables.

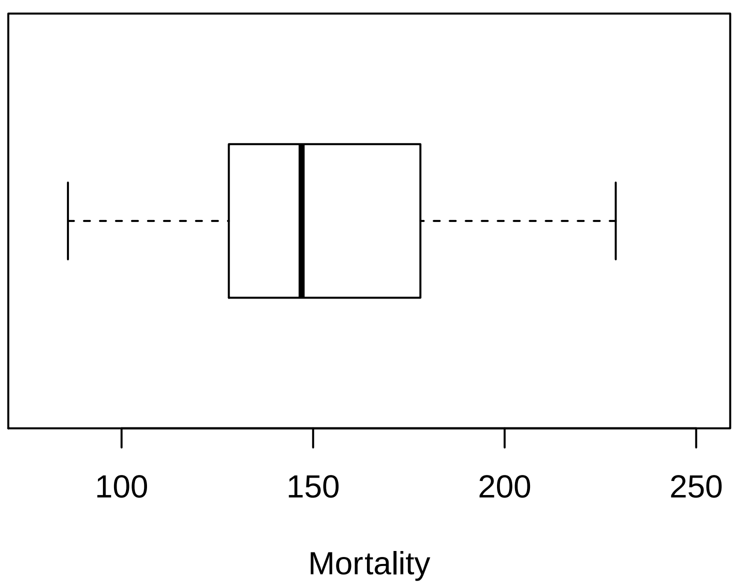
- `mortality`: number of white males died due to malignant melanoma 1950-1969 per one million inhabitants.
- `latitude`: latitude of the geographic centre of the state.
- `longitude`: longitude of the geographic centre of each state.
- `ocean`: a binary variable indicating contiguity to an ocean at levels `no` or `yes`.

Plotting mortality rates

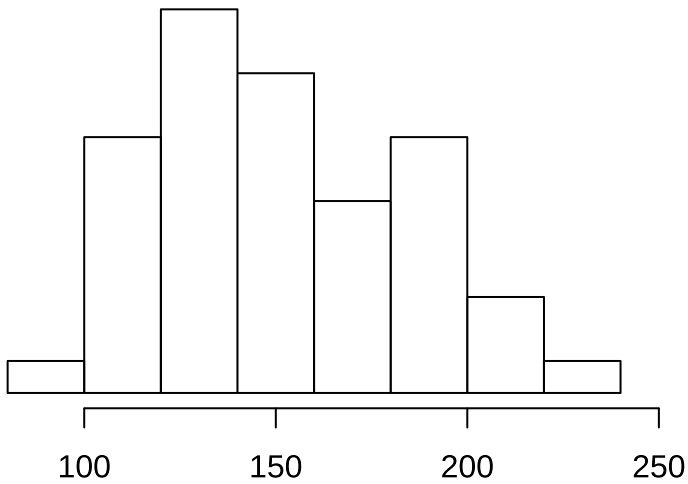
```
xr <- range(USmelanoma$mortality) * c(0.9, 1.1)
```

Let us plot mortality rates in

```
#layout(matrix(1:2, nrow = 2))
boxplot(USmelanoma$mortality, ylim = xr, horizontal = TRUE,xlab = "Mortality")
```

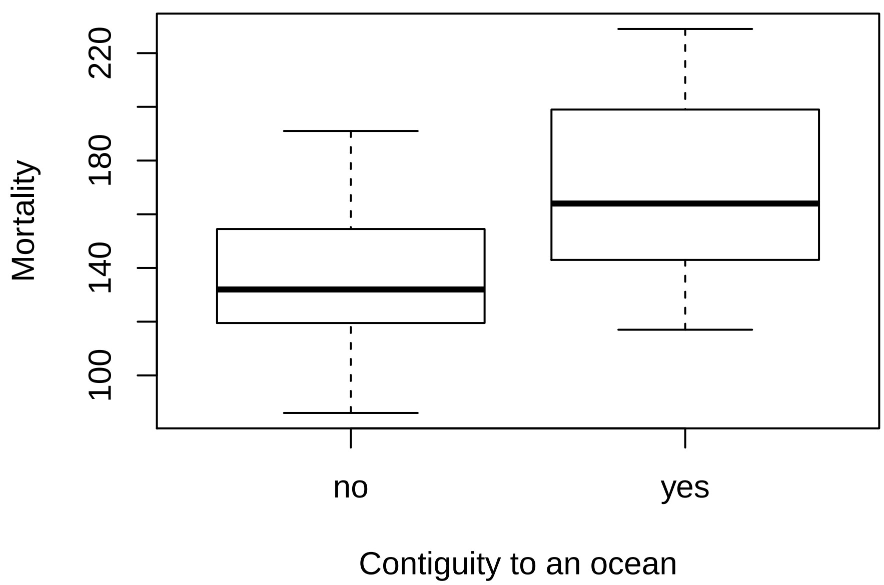


```
hist(USmelanoma$mortality, xlim = xr, xlab = "", main = "",axes = FALSE, ylab = "")
axis(1)
```



Malignant melanoma mortality rates by contiguity to an ocean

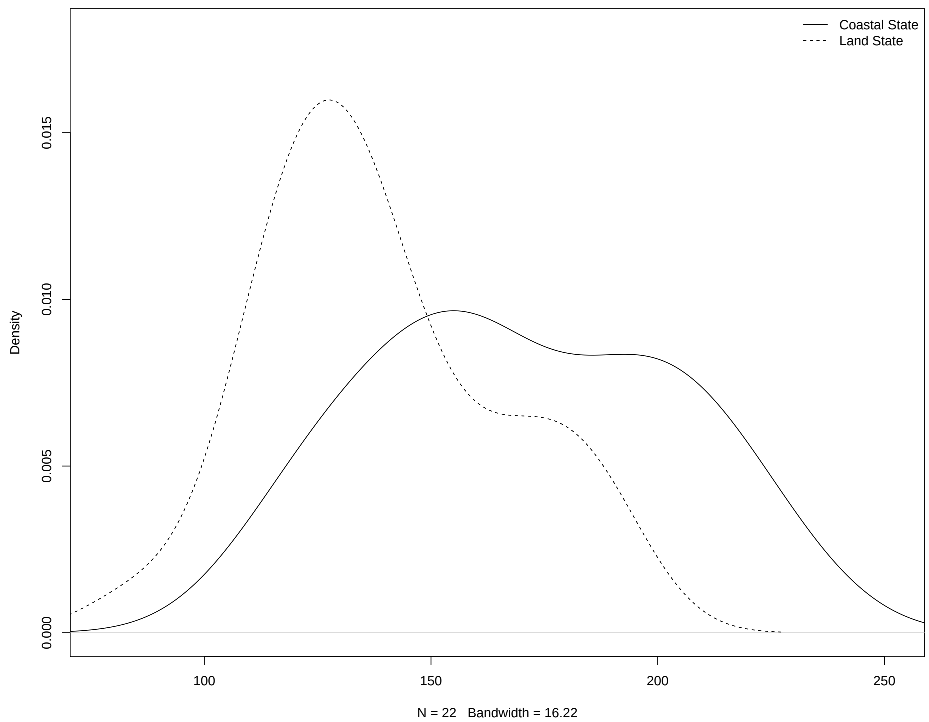
```
plot(mortality ~ ocean, data = USmelanoma, xlab = "Contiguity to an ocean", ylab = "Mortalit
```



Histograms can often be misleading for displaying distributions because of their dependence on the number of classes chosen. An alternative is to formally estimate the density function of a variable and then plot the resulting estimate.

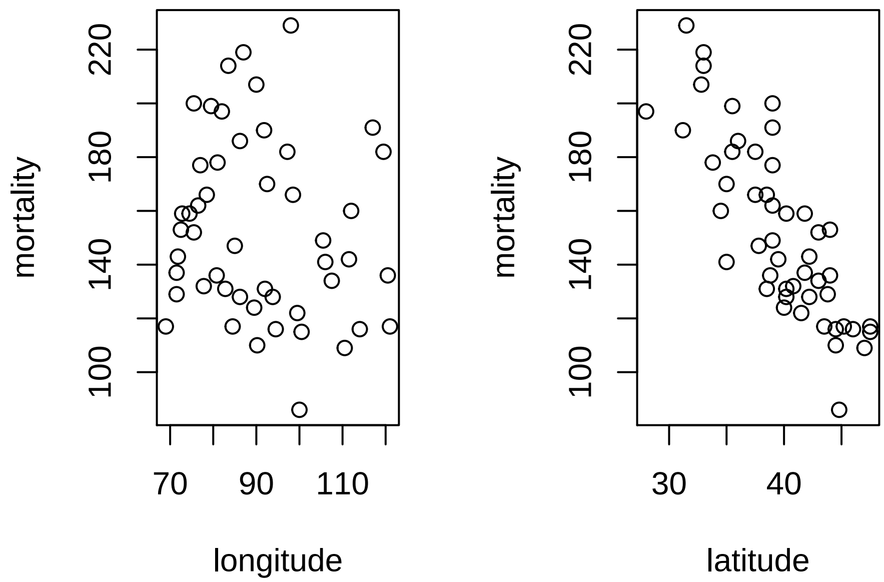
The estimated densities of malignant melanoma mortality rates by contiguity to an ocean looks like this:

```
dyes <- with(USmelanoma, density(mortality[ocean == "yes"]))
dno <- with(USmelanoma, density(mortality[ocean == "no"]))
plot(dyes, lty = 1, xlim = xr, main = "", ylim = c(0, 0.018))
lines(dno, lty = 2)
legend("topright", lty = 1:2, legend = c("Coastal State","Land State"), bty = "n")
```



Now we might move on to look at how mortality rates are related to the geographic location of a state as represented by the latitude and longitude of the centre of the state.

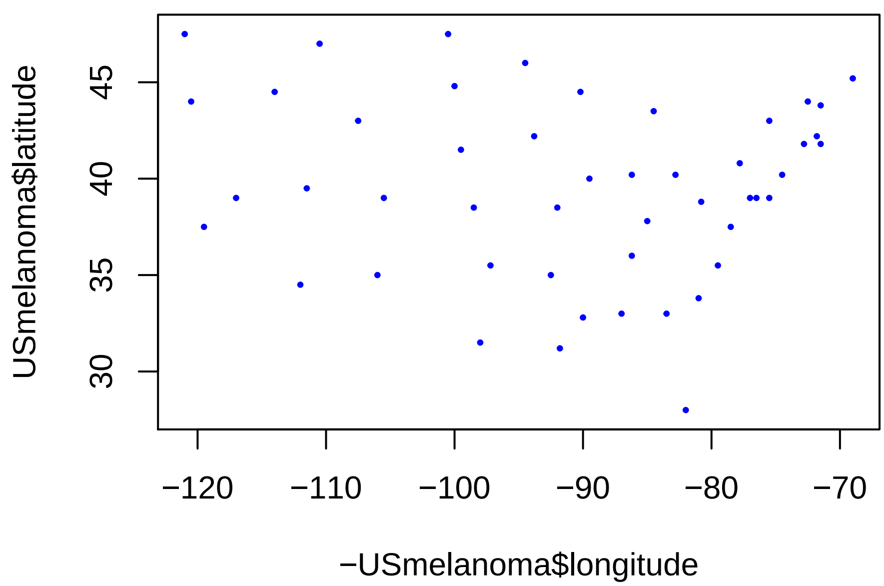
```
layout(matrix(1:2, ncol = 2))
plot(mortality ~ -longitude, data = USmelanoma)
plot(mortality ~ latitude, data = USmelanoma)
```



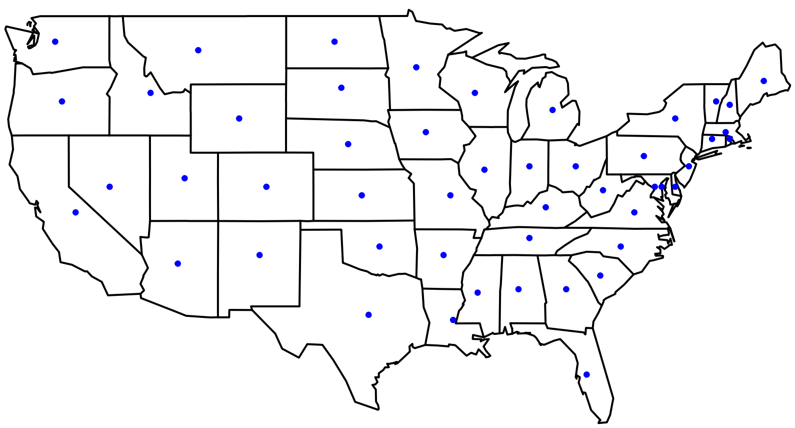
4.3. Mapping mortality rates

The data contains the longitude and latitude of the centroids

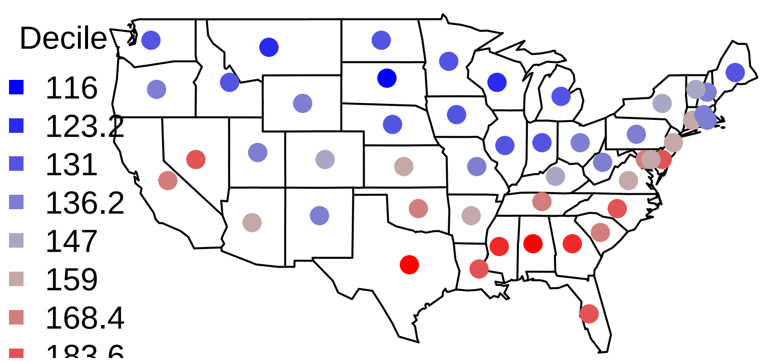
```
plot(-USmelanoma$longitude,USmelanoma$latitude,asp=1.5,cex=.3,pch=19,col="blue")
```



```r
library("sp")
library("maps")
library("maptools")
library("RColorBrewer")
map("state")
points(-USmelanoma$longitude,USmelanoma$latitude,asp=1.5,cex=.3,pch=19,col="blue")
```



```r
#Create a function to generate a continuous color palette
rbPal <- colorRampPalette(c('blue','grey','red'))
#This adds a column of color values
# based on the y values
USmelanoma$Col <- (rbPal(10)[as.numeric(cut(USmelanoma$mortality,breaks = 10))])
map("state",xlim=c(-135,-65))
points(-USmelanoma$longitude,USmelanoma$latitude,col=USmelanoma$Col,asp=1.5,pch=19,cex=1.2)
legend("topleft",title="Decile",legend=quantile(USmelanoma$mortality,seq(0.1,1,l=10)),col =
```



```
states <- map("state", plot = FALSE, fill = TRUE)
IDs <- sapply(strsplit(states$names, ":"), function(x) x[1])
rownames(USmelanoma) <- tolower(rownames(USmelanoma))

us1 <- map2SpatialPolygons(states, IDs=IDs,proj4string = CRS("+proj=longlat +datum=WGS84"))
us2 <- SpatialPolygonsDataFrame(us1, USmelanoma)

col <- colorRampPalette(c('blue', 'gray80','red'))

spplot(us2, "mortality", col.regions = col(200),par.settings = list(axis.line = list(col =
```