Nociones básicas de R

Mathias Bourel

18/3/2019

El software R es un software libre disponible en http://cran.r-project.org y es disponible para Windows, linux, Mac,. . .

1-Ayudas

```
- Toda la información material sobre R está en http://www.R-project.org
- RefCard donde están las principales funciones.
```

Uso de la función help

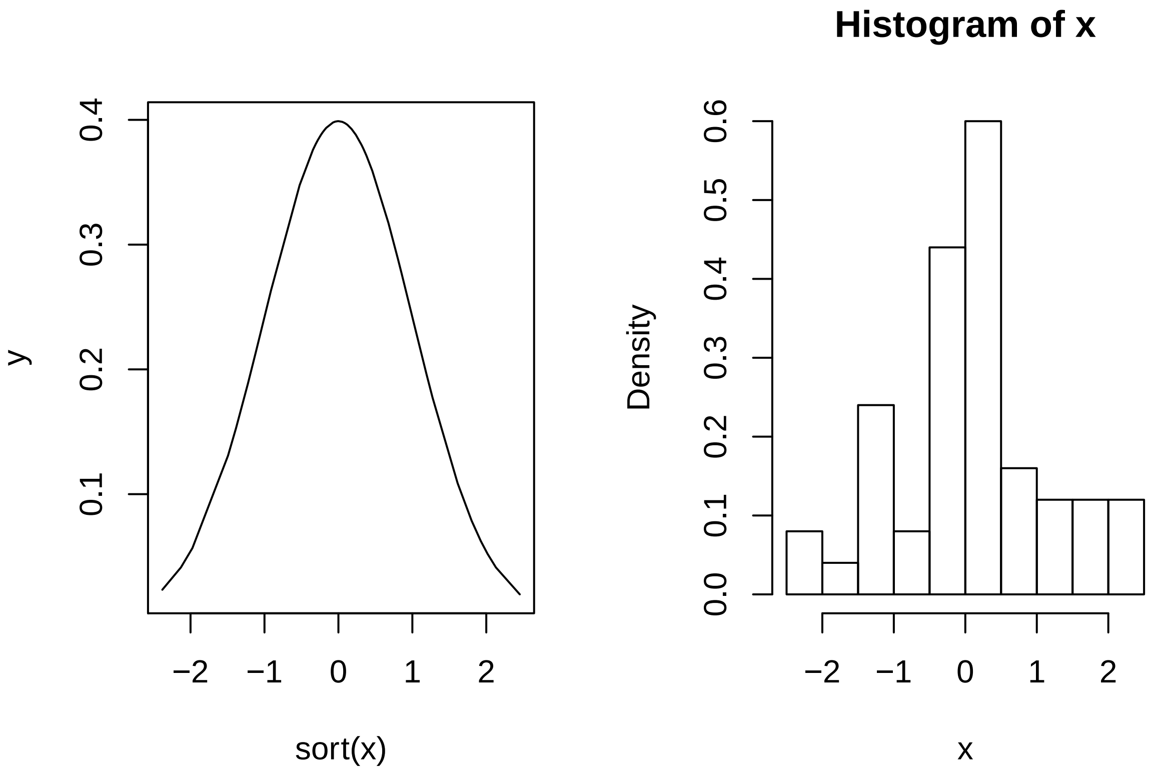
```r
help(rnorm)
?rnorm
```

Ejecutar comando o archivo.R

Para comando en un script: Ctrl+R

Para un archivo R

```r
x=rnorm(50)
source("Clase1anexo1.R")
```



Paquetes o libreria

para saber que librerias hay instaladas en la sesión y descargadas.

```
library()
```

todos lo paquetes en http://CRAN.R-project.org/web/packages (cerca de 2000)

Para ver funciones de la librería tree

```
library(help=rpart)
??rpart
help(package="rpart")
```

Descargar una librería

install.packages("MASS")

```
library(MASS)
```

2- Operaciones elementales y vectores

2.1 Calculadora

```
sqrt(25)+2
```

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

```
x=sqrt(25)+2; x
```

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

```
y=45+3*x ;y
```

```
## [1] 66
```

```
ls() ##nombra objetos que están cargados
```

```
## [1] "x" "y"
```

```
rm(y) #borra y
rm(list=ls()) #borra todo
x=sqrt(2)
sqrt(2)^2==2
```

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

```
    x=c(2,5,8)
    sum(x)
```

```
## [1] 15
```

```
    cumsum(x) #suma acumulada
```

```
## [1]  2  7 15
```

```
    prod(x)
```

```
## [1] 80
```

```
    cumprod(x) #producto acumulado
```

```
## [1]  2 10 80
```

```
    a=c(2,-1,4,3,7,5,19,10)
    sort(a)
```

```
## [1] -1  2  3  4  5  7 10 19
```

```
    b=sort(a)
    y=c(x,13,67,"mathias")
    length(y)
```

```
## [1] 6
```

```
    c(1:6)
```

```
## [1] 1 2 3 4 5 6
```

```
    c(6:1)
```

```
## [1] 6 5 4 3 2 1
```

```
    seq(2,8,by=0.5)
```

```
##  [1] 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
```

```
    v=rep(5,10)

    v1 <- c(0, -1); v2 <- c(2, 100); v3 <- c(v1, 1, 4, v2)
  v3
```

```
## [1]   0  -1   1   4   2 100
```

```
  1:20
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
  20:1
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
  -5:5
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
  .1 * 0:10
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Extraer de un vector:

```
    v=c(1.2:9.2)
        #la tercera coordenada de v
        v[3]
```

```
## [1] 3.2
```

```
        #la primera, la quinta y la sexta coordenada de v
        v[c(1,5,6)]
```

```
## [1] 1.2 5.2 6.2
```

```
        #saco de v la primera, la quinta y la sexta coordenada
        v[-c(1,5,6)]
```

```
## [1] 2.2 3.2 4.2 7.2 8.2 9.2
```

```
        #componentes mayores que 5
        v[v>5]
```

```
## [1] 5.2 6.2 7.2 8.2 9.2
```

```
        #coordenadas mayores que 5 o menores que 3
        v[v>5 | v<3]
```

```
## [1] 1.2 2.2 5.2 6.2 7.2 8.2 9.2
```

```
        #coordenadas mayores que 5 y menores que 8
        v[v>5 & v<8]
```

```
## [1] 5.2 6.2 7.2
```

```
            #otra posibilidad   con la función which (cuidado que devuelve la coordenada!!)
                x=(1:10)^2
                which(x==c(25,64))
```

```
## [1] 5 8
```

```
                x[which(x==c(25,64))]
```

```
## [1] 25 64
```

```
                which.max(x)
```

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

Operaciones con vectores:

```
u <- c(4, 2, -2, 1) ;v <- c(4, 2, 3, 6)
u + v
```

```
## [1] 8 4 1 7
```

```
u + v + rep(3, 4)
```

```
## [1] 11  7  4 10
```

```
u + v + 3
```

```
## [1] 11  7  4 10
```

```
x=c(1:6)
    y=seq(from=1, to=12,by=2)
    z=x+y
    z=3*x+y
    w=c(1:3)
    z=x+w

        x*y
```

```
## [1]  1  6 15 28 45 66
```

```
    x%*%y #producto escalar
```

```
##      [,1]
## [1,]  161
```

```
    u <- seq(1, 5, .5)
k <- 10
u
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
k * u
```

```
## [1] 10 15 20 25 30 35 40 45 50
```

```
a <- 1:4; b <- 4:1
a; b
```

```
## [1] 1 2 3 4
```

```
## [1] 4 3 2 1
```

```
a/b
```

```
## [1] 0.2500000 0.6666667 1.5000000 4.0000000
```

```
u <- 1:5
u; u^2
```

```
## [1] 1 2 3 4 5
```

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

```
u^3
```

```
## [1]   1   8  27  64 125
```

```
u^.5
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
dias <- c("lunes", "martes", "miercoles", "jueves",
          "viernes", "sabado", "domingo")
dias
```

```
## [1] "lunes"     "martes"    "miercoles" "jueves"    "viernes"   "sabado"
## [7] "domingo"
```

```
meses <- c("enero", "febrero", "marzo", "abril", "mayo",
    "junio", "julio", "agosto", "setiembre", "octubre",
    "noviembre", "diciembre")

paste(dias[1], 18, "de", meses[3])
```

```
## [1] "lunes 18 de marzo"
```

Comparación entre vectores

```
u <- 1:10
v <- 10:1
u; v
```

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

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

```
u > v
```

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

```
u < v
```

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

```
u >= v
```

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

```
u <= v
```

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

```
u == v
```

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

```
u != v
```

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

```
x <- rep(0, 5)
y <- c(rep(0, 4), 1)
x == y
```

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

```
identical(x, y)
```

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

```
z <- c(rep(0, 4), 0)
identical(x, z)
```

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

Operaciones aritméticas entre vectores logicos

```
u <- c(1, 3, 5); v <- c(-1, 1, 6)
x <- u==v
y <- u != v
x; y
```

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

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

```
x + y
```

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

```
x - y
```

```
## [1] -1 -1 -1
```

```
2*x; 2*y
```

```
## [1] 0 0 0
```

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

Operaciones logicas entre vectores logicos

```
x;y
```

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

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

```
x & y
```

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

```
x | y
```

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

Acceder a los entradas de un vector

```
(x <- c(4, 7, 0, -1, 8, 9, -12, 3))
```

```
## [1]   4   7   0  -1   8   9 -12   3
```

```
x[5]
```

```
## [1] 8
```

```
x[8]
```

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

```
y <- x[c(1, 3, 7)]
y
```

```
## [1]   4   0 -12
```

Modificar entradas de un vector

```
(x <- 1:11)
```

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

```
x[2:10] <- 0
x
```

```
##  [1]  1  0  0  0  0  0  0  0  0  0 11
```

```
x[c(3, 9)] <- c(9, 3); x
```

```
##  [1]  1  0  9  0  0  0  0  0  3  0 11
```

Seleccionar parte de las entradas de un vector

```
a <- c(2, 4, 6, 8, 10, 50, 12, 14, 16)
c(a[1:5], a[7:9])
```

```
## [1]  2  4  6  8 10 12 14 16
```

```
a[-6]
```

```
## [1]  2  4  6  8 10 12 14 16
```

Seleccion de elementos por argumentos lógicos

```
x <- 1:10
x; x >= 5
```

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

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

```
x[x >= 5]
```

```
## [1]  5  6  7  8  9 10
```

```
s <- c(T, F)
x[s]
```

```
## [1] 1 3 5 7 9
```

```
x[x > 3 & x < 8]
```

```
## [1] 4 5 6 7
```

```
x[x == 1 | x == 8]
```

```
## [1] 1 8
```

Algunas funciones matemática

```
x <- seq(-1, 1, .5)
exp(x)
```

```
## [1] 0.3678794 0.6065307 1.0000000 1.6487213 2.7182818
```

```
log(x)
```

```
## Warning in log(x): NaNs produced
```

```
## [1]        NaN        NaN       -Inf -0.6931472  0.0000000
```

```
factorial(10)
```

```
## [1] 3628800
```

```
choose(10, 2)
```

```
## [1] 45
```

Algunas funciones estadísticas

```
x <- c(-1, 4, 3, 5, 2, 7, 1)
sum(x)
```

```
## [1] 21
```

```
mean(x)
```

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

```
prod(x)
```

```
## [1] -840
```

```
min(x)
```

```
## [1] -1
```

```
max(x)
```

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

```
range(x)
```

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

```
length(x)
```

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

```
x <- c(1,0,3,-4,2)
y <- c(2,5,0,2,3)
x; y; sum(x*y)
```

```
## [1]  1  0  3 -4  2
```

```
## [1] 2 5 0 2 3
```

```
## [1] 0
```

```
cumsum(x)
```

```
## [1] 1 1 4 0 2
```

```
cumprod(x)
```

```
## [1] 1 0 0 0 0
```

```
cummin(x)
```

```
## [1]  1  0  0 -4 -4
```

```
cummax(x)
```

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

```
y <- c(6, 4, 3, 3, 4, 1, 6, 4)
y; unique(y)
```

```
## [1] 6 4 3 3 4 1 6 4
```

```
## [1] 6 4 3 1
```

```
sort(y)
```

```
## [1] 1 3 3 4 4 4 6 6
```

```
rev(y)
```

```
## [1] 4 6 1 4 3 3 4 6
```

```
sort(unique(y))
```

```
## [1] 1 3 4 6
```

4- Matrices

```
A=matrix(c(2,3,4,5,1,3,5,7,9),ncol=3)
    dim(A)
```

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

```
    ncol(A)
```

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

```
    nrow(A)
```

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

```
    A=matrix(c(2,3,4,5,1,3,5,8,9),ncol=3,byrow=T) #por defecto la matriz se llena por columna
    A=matrix(c(2,3,4,5,1,3,5,9),ncol=2,nrow=4)
    A=matrix(c(2,3,4,5,1,3,5,9),ncol=2,nrow=4,byrow=T)

    diag(3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```
    diag(1:5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    2    0    0    0
## [3,]    0    0    3    0    0
## [4,]    0    0    0    4    0
## [5,]    0    0    0    0    5
```

```
    diag(c(1,20,35,42))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0   20    0    0
## [3,]    0    0   35    0
## [4,]    0    0    0   42
```

```
    O=matrix(0, 4, 3) #matriz de ceros
```

Concatenar por filas y por columnas

```
    x=1:12
    y=x^2
    A=rbind(x,y)
    B=cbind(x,y)
    t(A)==B
```

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

Extraer de una matriz

```
A=matrix(c(2,3,4,5,1,3,5,7,9),ncol=3,byrow=T)
    A
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    4
## [2,]    5    1    3
## [3,]    5    7    9
```

```
        #el elemento 2-3
            A[2,3]
```

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

```
        #La fila 1
            A[1,]
```

```
## [1] 2 3 4
```

```
        #la columna 2
            A[,2]
```

```
## [1] 3 1 7
```

Operaciones con matrices

```
        C=matrix(c(-5, 1, 3,
        1, 2, 6,
        3, 6, -4),3, 3, byrow=TRUE)
        all(C == t(C)) # entonces C es simétrica
```

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

```
        solve(C) #inversa de C
```

```
##               [,1]       [,2]        [,3]
## [1,] -1.818182e-01 0.09090909  0.00000000
## [2,]  9.090909e-02 0.04545455  0.13636364
## [3,] -3.784851e-18 0.13636364 -0.04545455
```

```
        solve(C,c(1,1,1)) # resuelve el sistema CX=c(1,1,1)
```

```
## [1] -0.09090909  0.27272727  0.09090909
```

```
        svd(C) #descomposición svd de C  (C=UDV' con U y V ortogonales y D diagonal)
```

```
## $d
## [1] 8.886928 6.246400 4.359472
## 
## $u
##            [,1]      [,2]       [,3]
## [1,]  0.5010900 0.2163160  0.8379237
## [2,]  0.3819443 0.8135676 -0.4384363
## [3,] -0.7765484 0.5397362  0.3250499
## 
## $v
##            [,1]      [,2]       [,3]
## [1,] -0.5010900 0.2163160 -0.8379237
## [2,] -0.3819443 0.8135676  0.4384363
## [3,]  0.7765484 0.5397362 -0.3250499
```

```
        log(C)
```

```
## Warning in log(C): NaNs produced
```

```
##          [,1]      [,2]     [,3]
## [1,]      NaN 0.0000000 1.098612
## [2,] 0.000000 0.6931472 1.791759
## [3,] 1.098612 1.7917595      NaN
```

```
        #rango de una matriz
            # El rango de una matriz cuadrada simetrica equivale a la cantidad
            # de valores propios distintos de 0:
            valprop=eigen(C, only.values = TRUE)
            rango=length(valprop[[1]]>=1.e-10)
            print(c("El rango de la matriz es",rango),quote=F)
```

```
## [1] El rango de la matriz es 3
```

```
    A=matrix(c(1:16),nrow=4,ncol=4)
A[2,3] # elemento en la fila 2 y columna 3
```

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

```
A[3,] # fila 3
```

```
## [1]  3  7 11 15
```

```
A[c(4, 1), ] # filas 3 y 1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4    8   12   16
## [2,]    1    5    9   13
```

```
A[,2] # columna 2
```

```
## [1] 5 6 7 8
```

```
A[,c(1:3, 2)] # columna 1, 2, 3 y 5
```

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

```
        B=matrix(c(5:8),nrow=4, ncol=4)
        A+2 #suma 2 a cada entrada
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    3    7   11   15
## [2,]    4    8   12   16
## [3,]    5    9   13   17
## [4,]    6   10   14   18
```

```
        A+c(1,3) # suma un vector a cada columna
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    6   10   14
## [2,]    5    9   13   17
## [3,]    4    8   12   16
## [4,]    7   11   15   19
```

```
        A+2*B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   11   15   19   23
## [2,]   14   18   22   26
## [3,]   17   21   25   29
## [4,]   20   24   28   32
```

```
3*A #multiplica todas la entradas de A por 3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    3   15   27   39
## [2,]    6   18   30   42
## [3,]    9   21   33   45
## [4,]   12   24   36   48
```

```
A*B #multiplica termino a termino
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5   25   45   65
## [2,]   12   36   60   84
## [3,]   21   49   77  105
## [4,]   32   64   96  128
```

```
A%*%B #esa es la multiplicación matricial
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  202  202  202  202
## [2,]  228  228  228  228
## [3,]  254  254  254  254
## [4,]  280  280  280  280
```

```
eigen(A) # calcula valores y vectores propios de A
```

```
## eigen() decomposition
## $values
## [1]  3.620937e+01 -2.209373e+00  1.599839e-15  7.166935e-16
##
## $vectors
##           [,1]        [,2]       [,3]       [,4]
## [1,] 0.4140028  0.82289268 -0.5477226  0.1125155
## [2,] 0.4688206  0.42193991  0.7302967  0.2495210
## [3,] 0.5236384  0.02098714  0.1825742 -0.8365883
## [4,] 0.5784562 -0.37996563 -0.3651484  0.4745519
```

```
det(A) #determinante de A
```

```
## [1] 0
```

```
t(A) #traspuesta de A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
```

```
sum(diag(A)) #traza de A
```

```
## [1] 34
```

5- DataFrame

Primer Ejemplo

```
x <- 1:10
y <- rep(c("masculino", "femenino"), 5); y
```

```
##  [1] "masculino" "femenino"  "masculino" "femenino"  "masculino"
##  [6] "femenino"  "masculino" "femenino"  "masculino" "femenino"
```

```
X <- cbind(x, y); X
```

```
##       x    y
##  [1,] "1"  "masculino"
##  [2,] "2"  "femenino"
##  [3,] "3"  "masculino"
##  [4,] "4"  "femenino"
##  [5,] "5"  "masculino"
##  [6,] "6"  "femenino"
##  [7,] "7"  "masculino"
##  [8,] "8"  "femenino"
##  [9,] "9"  "masculino"
## [10,] "10" "femenino"
```

```
class(X)
```

```
## [1] "matrix"
```

```
mode(X)
```

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

```
X[,1]
```

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

```
sum(as.numeric(X[,1]))
```

```
## [1] 55
```

```
class(X[,1])
```

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

```
Y <- data.frame(x, y)
class(Y)
```

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

```
Y[, 1]
```

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

```
sum(Y[,1])
```

```
## [1] 55
```

```
class(Y[,1])
```

```
## [1] "integer"
```

```
class(Y[,2])
```

```
## [1] "factor"
```

Todas las posibles combinaciones entre niveles

```
datos <- expand.grid(x = c(60, 80), y = c(100, 300), sexo = c("masculino", "femenino"))
datos
```

```
##    x   y      sexo
## 1 60 100 masculino
## 2 80 100 masculino
## 3 60 300 masculino
## 4 80 300 masculino
## 5 60 100  femenino
## 6 80 100  femenino
## 7 60 300  femenino
## 8 80 300  femenino
```

```
class(datos)
```

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

```
datos[, 1]
```

```
## [1] 60 80 60 80 60 80 60 80
```

```
names(datos)
```

```
## [1] "x"    "y"    "sexo"
```

```
datos$x
```

```
## [1] 60 80 60 80 60 80 60 80
```

```
datos$sexo
```

```
## [1] masculino masculino masculino masculino femenino  femenino  femenino 
## [8] femenino 
## Levels: masculino femenino
```

Seleccion de individuos por condiciones sobre variables

```
datos$sexo == 'femenino'
```

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

```
datos[datos$sexo == 'femenino', ]
```

```
##    x   y     sexo
## 5 60 100 femenino
## 6 80 100 femenino
## 7 60 300 femenino
## 8 80 300 femenino
```

```
datos$x > 60
```

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

```
datos[datos$x > 60, ]
```

```
##    x   y      sexo
## 2 80 100 masculino
## 4 80 300 masculino
## 6 80 100  femenino
## 8 80 300  femenino
```

```
(datos$x > 60) & (datos$y == 300)
```

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

```
datos[(datos$x > 60) & (datos$y == 300), ]
```

```
##    x   y      sexo
## 4 80 300 masculino
## 8 80 300  femenino
```

```
datos[(datos$x > 60) | (datos$y == 300), ]
```

```
##    x   y      sexo
## 2 80 100 masculino
## 3 60 300 masculino
## 4 80 300 masculino
## 6 80 100  femenino
## 7 60 300  femenino
## 8 80 300  femenino
```

Otro Ejemplo

```
datos2=data.frame(x1=1:10,x2=seq(20,110,by=10),l=letters[1:10])
    names(datos2)
```

```
## [1] "x1" "x2" "l"
```

```
    colnames(datos2)=c("numero","cantidad", "etiqueta")
    head(datos2,4)
```

```
##   numero cantidad etiqueta
## 1      1       20        a
## 2      2       30        b
## 3      3       40        c
## 4      4       50        d
```

```
    head(datos2,6)
```

```
##   numero cantidad etiqueta
## 1      1       20        a
## 2      2       30        b
## 3      3       40        c
## 4      4       50        d
## 5      5       60        e
## 6      6       70        f
```

```
    dim(datos2)
```

```
## [1] 10  3
```

```
    datos2$cantidad
```

```
##  [1]  20  30  40  50  60  70  80  90 100 110
```

```
    datos2[,2]
```

```
##  [1]  20  30  40  50  60  70  80  90 100 110
```

6- Listas (permite combinar objetos de distinto tipo y tamaño)

```
x <- 1:25
X <- matrix(1:10, ncol = 5)
lista = list(vector = x, matriz = X, meses = meses)
lista
```

```
## $vector
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25
##
## $matriz
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
##
## $meses
##  [1] "enero"      "febrero"    "marzo"      "abril"      "mayo"
##  [6] "junio"      "julio"      "agosto"     "setiembre"  "octubre"
## [11] "noviembre"  "diciembre"
```

Clase de los objetos

```
ls()
```

```
##  [1] "a"       "A"       "b"       "B"       "C"       "datos"   "datos2"
##  [8] "dias"    "k"       "lista"   "meses"   "O"       "rango"   "s"
## [15] "u"       "v"       "v1"      "v2"      "v3"      "valprop" "w"
## [22] "x"       "X"       "y"       "Y"       "z"
```

```
a; class(a)
```

```
## [1]  2  4  6  8 10 50 12 14 16
```

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

```
meses; class(meses)
```

```
##  [1] "enero"      "febrero"    "marzo"      "abril"      "mayo"
##  [6] "junio"      "julio"      "agosto"     "setiembre"  "octubre"
## [11] "noviembre"  "diciembre"
```

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

```
A; class(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
## [1] "matrix"
```

```
datos; class(datos)
```

```
##    x   y      sexo
## 1 60 100 masculino
## 2 80 100 masculino
## 3 60 300 masculino
## 4 80 300 masculino
## 5 60 100  femenino
## 6 80 100  femenino
## 7 60 300  femenino
## 8 80 300  femenino
```

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

```
lista; class(lista)
```

```
## $vector
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25
## 
## $matriz
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
## 
## $meses
##  [1] "enero"      "febrero"    "marzo"      "abril"      "mayo"      
##  [6] "junio"      "julio"      "agosto"     "setiembre"  "octubre"   
## [11] "noviembre"  "diciembre"
```

```
## [1] "list"
```

Otro Ejemplo

```
    library("HSAUR2")
```

```
## Loading required package: tools
```

```
    data(Forbes2000)
    dim(Forbes2000)
```

```
## [1] 2000    8
```

```
    head(Forbes2000,4)
```

```
##   rank                name       country             category  sales
## 1    1           Citigroup United States              Banking  94.71
## 2    2    General Electric United States        Conglomerates 134.19
## 3    3 American Intl Group United States            Insurance  76.66
## 4    4           ExxonMobil United States Oil & gas operations 222.88
##   profits  assets marketvalue
## 1   17.85 1264.03      255.30
## 2   15.59  626.93      328.54
## 3    6.46  647.66      194.87
## 4   20.96  166.99      277.02
```

```
    ls()
```

```
##  [1] "a"          "A"          "b"          "B"          "C"         
##  [6] "datos"      "datos2"     "dias"       "Forbes2000" "k"         
## [11] "lista"      "meses"      "O"          "rango"      "s"         
## [16] "u"          "v"          "v1"         "v2"         "v3"
## [21] "valprop"    "w"          "x"          "X"          "y"
## [26] "Y"          "z"
```

```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 60 ...
##  $ 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 ...
```

```r
names(Forbes2000)
```

```
## [1] "rank"        "name"        "country"     "category"    "sales"
## [6] "profits"     "assets"      "marketvalue"
```

```r
class(Forbes2000[,4])
```

```
## [1] "factor"
```

```r
nlevels(Forbes2000[,4])
```

```
## [1] 27
```

```r
summary(Forbes2000)
```

```
##       rank            name                      country
##  Min.   :   1.0   Length:2000        United States :751
##  1st Qu.: 500.8   Class :character   Japan         :316
##  Median :1000.5   Mode  :character   United Kingdom:137
##  Mean   :1000.5                      Germany       : 65
##  3rd Qu.:1500.2                      France        : 63
##  Max.   :2000.0                      Canada        : 56
##                                      (Other)       :612
##                     category        sales            profits
##  Banking               : 313   Min.   :  0.010   Min.   :-25.8300
##  Diversified financials: 158   1st Qu.:  2.018   1st Qu.:  0.0800
##  Insurance             : 112   Median :  4.365   Median :  0.2000
##  Utilities             : 110   Mean   :  9.697   Mean   :  0.3811
##  Materials             :  97   3rd Qu.:  9.547   3rd Qu.:  0.4400
##  Oil & gas operations  :  90   Max.   :256.330   Max.   : 20.9600
##  (Other)               :1120                     NA's   :5
##      assets          marketvalue
##  Min.   :   0.270   Min.   :  0.02
##  1st Qu.:   4.025   1st Qu.:  2.72
##  Median :   9.345   Median :  5.15
##  Mean   :  34.042   Mean   : 11.88
##  3rd Qu.:  22.793   3rd Qu.: 10.60
##  Max.   :1264.030   Max.   :328.54
##
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   2.018   4.365   9.697   9.547 256.330
```

```
    bancos=subset(Forbes2000,category=="Banking")
    head(bancos,5)
```

```
##    rank             name        country category sales profits  assets
## 1     1        Citigroup  United States  Banking 94.71   17.85 1264.03
## 6     6 Bank of America  United States  Banking 49.01   10.81  736.45
## 7     7       HSBC Group United Kingdom  Banking 44.33    6.66  757.60
## 15   15 JP Morgan Chase  United States  Banking 44.39    4.47  792.70
## 18   18      BNP Paribas         France  Banking 47.74    4.73  745.09
##    marketvalue
## 1       255.30
## 6       117.55
## 7       177.96
## 15       81.94
## 18       59.29
```

```
    dim(bancos)
```

```
## [1] 313   8
```

Estadísticos Descriptivos

```
    median(Forbes2000[,5])
```

```
## [1] 4.365
```

```
        mean(Forbes2000[,5])
```

```
## [1] 9.69701
```

```
        range(Forbes2000[,5])
```

```
## [1]   0.01 256.33
```

```
        sd(Forbes2000[,5])
```

```
## [1] 18.00259
```

```
        cor(Forbes2000[,c(5,7)]) #matriz de correlaci?n entre las variables especificadas
```

```
##            sales    assets
## sales  1.0000000 0.4261541
## assets 0.4261541 1.0000000
```

Datos faltantes

```
    na_Type=is.na(Forbes2000$profits)
    table(na_Type)
```

```
## na_Type
## FALSE  TRUE
##  1995     5
```

Cuando queremos borrar las observaciones con NA, hacemos:

```
a=complete.cases(Forbes2000)
completo=subset(Forbes2000,a=TRUE)



        mean(Forbes2000$profits)
```

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

```
        mean(Forbes2000$profits,na.rm=T)
```

```
## [1] 0.3811328
```

7- Importar y exportar datos

Exportar datos

```
write.table(Forbes2000,file="misdatos.txt",sep=",",col.names=NA)
        write.table(Forbes2000,file="misdatos.csv",sep=",",col.names=NA)
        save(Forbes2000,file="Forbes2000.rda")
        load("Forbes2000.rda")
```

Importar datos

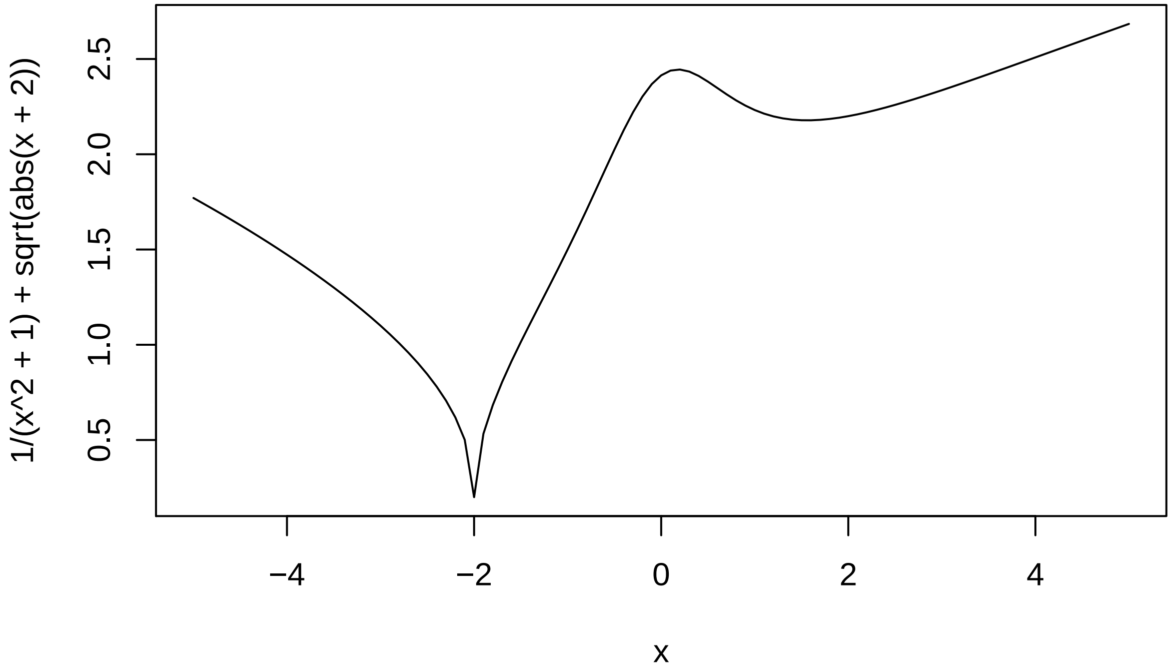
```
data=read.csv("AGE_CHD.csv",dec=",",sep=";",header=T)
```

8- Funciones

```
set.seed(29)
x=seq(0,10,0.5)
y=1/(x^2+1)+sqrt(abs(x+2))
max(y)
```

```
## [1] 3.474003
```

```
curve(1/(x^2+1)+sqrt(abs(x+2)),-5,5)
```

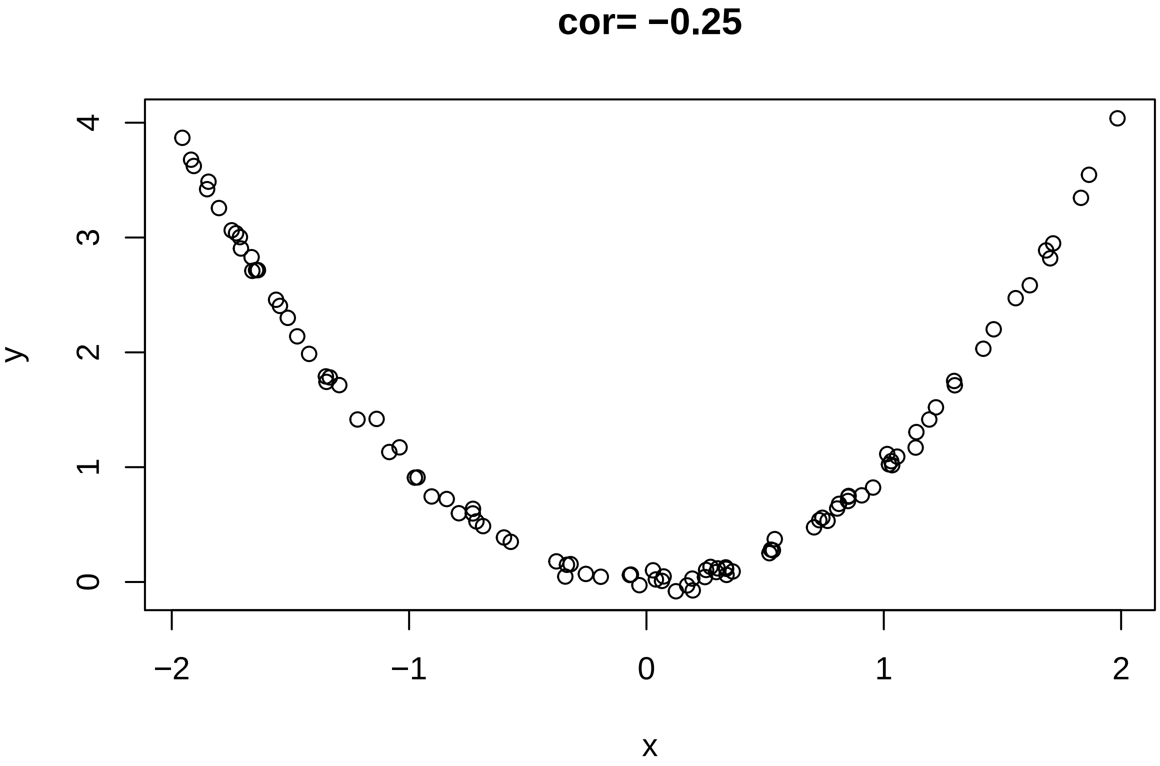


```
f=function(x){1/(x^2 + 1) + sqrt(abs(x+2))}
f(1.234)
```

```
## [1] 2.194724
```

9-Dibujar

```
set.seed(1000)
x=runif(100,-2,2)
y=x^2+rnorm(100,0,0.05)
c=cor(x,y)
plot(x,y,main=paste('cor=',round(c,3)))
```


cor= −0.25
y
x