

### Funciones básicas

#### objects(); ls()

visualizan el listado de objetos presentes en el actual espacio o área de trabajo

#### library(caret)

La mayoría de las librerías disponibles en R han de ser cargados antes de que se los pueda utilizar.

#### search()

devuelve la ruta de búsqueda de R, en donde `.GlobalEnv` se refiere al área de trabajo actual y `package:xxx` a los paquetes cargados. R busca cualquier objeto primero en el área de trabajo actual y después en los paquetes cargados según el orden de esta lista.

#### comment(x)

etiquetar los objetos

#### paste0("text",3+4)

integrar código y texto en la propia salida

#### ?function; help(function)

ayuda muy completa sobre todas las funciones, procedimientos y elementos que configuran el lenguaje. Se puede acceder a información específica sobre las mismas con el comando `help` o mediante `'?'`

#### args(function)

Ver argumentos de la función

#### function

visualizar código de la función

#### apropos("function")

visualizar un listado de todas las funciones que contienen ese comando

### Funciones básicas (cont)

#### example(function)

ejemplo que utilice el comando que usamos

#### library(help="lib")

información sobre una librería

#### help.search("topic")

buscará ayuda sobre el tema escogido en todas las librerías instaladas

#### RSiteSearch("issue")

buscar palabras de interés entre todos los mensajes enviados a las amplias listas de ayuda de correo electrónico del propio repositorio CRAN de R

### Operaciones con matrices

**dim()**, Número de filas y/o

**nrow()**, columnas

**ncol()**

**diag()** diagonal de una matriz

**\*** Multiplicación elemento a elemento

**%\*%** Multiplicación matricial de matrices

**cbind()**, Encadenamiento de

**rbind()** columnas o filas

**t()** Transpuesta

**solve(A)** Inversa de la matriz A

**solve(A,b)** Solución de sistema de ecuaciones  $Ax = b$

**qr()** Descomposición de Cholesky

**eigen()** Autovalores y autovectores

**svd()** Descomposición singular

### Listas

**x <- c(1, 2, 3, 4)\*\***

**y <- c("Hombre", "Mujer")**

**z <- matrix(1:12, ncol = 4)**

**datos <- list(A=x, B=y, C=z)**

Ejemplo de creación de list

**x=c(1,2,-3,-4,5,6)**

**L = list(nombre = "x", vector = x, media = mean(x), sumas = cumsum(x))**

Ejemplo de list donde todos los elementos son derivados del mismo.

#### L\$nombre

referirnos o usar un componente concreto de una lista

#### L[[1]]

referirnos o usar un componente concreto de una lista. Si usamos sólo un par de corchetes, como en los vectores, lo que obtenemos es una list formada por esa única componente.

#### str(L)

Estructura interna de una lista (nombres de elementos y contenido de los elementos con su tipo)

#### names(L)

Nombres de los elementos de una lista.

#### L=c(L, numero.pi=pi)

Añadir un nuevo elemento a una lista.

En R, el elemento list es un concepto distinto a lo que sería en Python u otros lenguajes. Se trata de un elemento que puede guardar diferentes tipos de variables.



### Objetos básicos en R

```
x = c(3, 5, 7)
```

Generación de vectores, similar a pd.Series sin índice.

```
x = c(cara=1,cruz=0)
```

Generación de vectores con etiquetas, similar a pd.Series.

```
x = 1:5
```

Generación de secuencias simple

```
seq(1, 5, 0.5)
```

Generación de secuencias en intervalos con distancia específica

```
seq(from=1, to=5, length=9)
```

Generación de secuencias en un número de intervalos específico

```
sample(1:6, size=10, replace = T)
```

Generación de secuencias aleatorias. Con replace=T se pueden repetir los valores.

```
class(var)
```

Clase de una variable

```
attributes(var)
```

Atributos de una variable

```
names(var)
```

Nombres en una variable

```
table(var)
```

Tabla de frecuencias

Para simular el lanzamiento de una moneda podemos escribir

```
resultado = c(cara =1, cruz=0)
lanz <- sample (re sul tado, siz
e=10, replace = T)
```

### Factores

```
sexo = c(0, 1, 1, 1, 0, 0, 1)
```

```
sexo2 = factor(sexo, labels = c("hombre", "-
mujer"))
```

Ejemplo de factor

```
respuestas = factor(c('si', 'si', 'no', 'si', 'si'))
```

Ejemplo de factor sin usar el parámetro labels

```
respuestas <- factor(c('si', 'si', 'no', 'si', 'si'),
levels = c('si', 'no'))
```

Cambiar el orden en el que se guarden las etiquetas. Por defecto se hace en orden alfabético.

```
levels(factor)
```

devuelve los niveles de un factor

```
unclass(factor)
```

representación subyacente del factor. El valor más bajo se convierte en 1 y el más alto en el número total de valores distintos.

Son vectores en los que cada valor tiene una etiqueta. Por ejemplo, todos los 1s tendrán una etiqueta, todos los 2s otra etiqueta, etc.

### Dataframes

```
Producto <- c("Zumo", "Queso", "Yogourt")
```

```
Seccion <- c("Bebidas", "Lácteos", "Lácteos")
```

```
Unidades <- c(2, 1, 10)
```

```
x <- data.frame(Producto, Seccion,
Unidades)
```

Ejemplo de creación de dataframe. Se podría crear un array cambiando data.frame por array en la función.

```
lista.compra$Unidades
```

acceder a los valores de un data.frame

```
lista.compra[,3]
```

Mismo resultado

### Dataframes (cont)

```
lista.compra$Unidades[1:2]
```

primeros dos valores de Unidades

```
lista.compra[2,]
```

segunda fila

### Principales estructuras de control

```
nombre <- function(arg1, arg2, ... ) {expre-
sión}
```

Creación de función

```
Density.Plot <- function(datos, ...){ plot(d-
ensity(datos), ... ) }
```

el argumento ... permite pasar de manera "libre" argumentos adicionales para ser utilizados por otra "subfunción" dentro de la función principal

```
str(args(fun))
```

argumentos de entrada de una función

```
fun
```

al escribir el nombre de una función se obtiene su contenido

```
if(condición){acción1} else if(condición){a-
cción2}
```

```
else {acción3}
```

```
for(índice in vector){acción}
```

```
while(condición){acción}
```

```
break / next
```

abortar un bucle o avanzar forzosamente un bucle

```
repeat{acción}
```

ejecuta un bucle de forma infinita hasta que le ordenemos parar con un break

### Selección de elementos de un vector

```
x[n]
```

Elemento número n

```
x[c(x, z, n)]
```

posiciones x, z y n



By **julenx**

[cheatography.com/julenx/](https://cheatography.com/julenx/)

Published 16th November, 2022.

Last updated 6th January, 2025.

Page 2 of 3.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopod.com>

### Selección de elementos de un vector (cont)

`x[x>0]`

Valores positivos

`x[-c(1:3)]`

elementos de x salvo los 3 primeros

### Ordenación de vectores

`sort(x)`

Ordenación de vectores. Admite el parámetro `decreasing=T`

`order(x)`

Índices del vector ordenados

`rev(x)`

valores del vector en orden inverso

`x[order(x)]`

Mismo resultado que `sort(x)`

`rev(x)`

valores del vector en orden inverso. El último elemento se convierte en el primero.

### Valores perdidos

`mean(altura, na.rm = TRUE)`

forzar a R a que ignore los valores perdidos

### Vectores no numéricos

`letters[1:10]`

primeras 10 letras del abecedario

`LETTERS[1:10]`

lo mismo en mayúscula

`month.name[1:6]`

primeros 6 meses del año en inglés

### Matrices (similar a np.arrays)

`cbind(x, y)`

Creación de matriz uniendo dos vectores en la que cada columna es un vector.

`rbind(x, y)`

Creación de matriz uniendo dos vectores en la que cada fila es un vector.

`matrix(1:8, nr = 2, nc = 4)`

Ejemplo de creación de matriz. Se hace por columnas, y para que se haga por filas se usa `byrow = TRUE`

`rownames(x) <- c("fila 1", "fila 2")`

`colnames(x) <- c("col 1", "col 2")`

Nombres en matrices

`colnames(x) <- paste("col", 1:ncol(x), sep="")`

Mismo resultado que el ejemplo anterior

`dim(x)`

Comprobar dimensión de una matriz (filas y columnas)

`attributes(x)`

Dimensiones y nombres de filas y columnas de la matriz

`x[1, 1]`

Elemento de la primera fila y primera columna de la matriz

`x[2, ]`

Segunda fila de la matriz

`x[, 2]`

Segunda columna de la matriz

`x[1, 1:2]`

primera fila, columnas 1ª y 2ª

### Matrices (similar a np.arrays) (cont)

`ii = order(x[, 1]); x[ii, ]`

Reordenar filas en base a los valores de la primera columna

`ii = order(x[, 4]); x[ii, ]`

Reordenar filas en base a los valores de la cuarta columna

### La familia de funciones apply

`apply(X, MARGIN, FUN, ...)`

Aplica una función a todos los elementos de una matriz o cualquier elemento que podamos convertir a una matriz por filas o columnas

X: una matriz

MARGIN: la dimensión que agrupará los elementos de la matriz X, para aplicarles una función. Son identificadas con números, 1 son filas y 2 son columnas.

FUN: la función que aplicaremos a la matriz X en su dimensión MARGIN.

`lapply(X, FUN)`

Aplica una función a todos los elementos de una lista. X debe ser una lista y se devuelve una lista.

