6Matrices_cheatsheet.R

moka

2023-04-20

```
# Fecha: 11.04.2023
# Referencia: Basado en R Programming Fundamentals, StanfordOnline XDFS112
# Objetivo: manipular datos que NO son heterogeneos
# Matrices: permite almacenar datos homogeneos
```

```
# Configurar el directorio
```

```
midirectorio<-setwd("~/Dropbox/0.POST-PHD/GOALS/2.CODE/R/Ecomienza/6Matrices")
midirectorio
```

```
## [1] "/Users/moka/Dropbox/0.POST-PHD/GOALS/2.CODE/R/Ecomienza/6Matrices"
```

```
# Creacion de matrices y manipulacion
```

```
# La estructura de los datos es tal Estructuramos los datos como individuos en filas, y las columnas se
A<-matrix(c(4,2,0,3,1,7,2,8,4,5), # los elementos de datos
          nrow=2, # el número de filas
          ncol=5)  # el número de columnas
A
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    0    1    2    4
## [2,]    2    3    7    8    5
```

```
# Podemos omitir el argumento ncols
B<-matrix(c(4,2,0,3,1,7,2,8,4,5), # los elementos de datos
          nrow=2, # el número de filas
)  # el número de columnas
B
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    0    1    2    4
## [2,]    2    3    7    8    5
```

```
# Podemos pedir que se rellene la matriz primero por filas que por columnas si usamos la opción byrow o
C<-matrix(c(4,2,0,3,1,7,2,8,4,5), # los elementos de datos
          nrow=2,byrow=TRUE # el número de filas
)  # el número de columnas
C
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    2    0    3    1
## [2,]    7    2    8    4    5
```

```
# Comprueba la clase del objeto
class(C)
```

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

```r
class(A)
```

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

```r
# Comprueba si es una matriz
is.matrix(A)
```

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

```r
# Indexa una fila
A[1,] # Esta fila se considera un vector
```

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

```r
A[2,] # Esta fila se considera un vector
```

```
## [1] 2 3 7 8 5
```

```r
# Añade una nueva fila a la matriz
Ac=rbind(A,c(4,7,6,2,1))
# Combinar matrices
# Combinamos usando rbind, necesitamos que las matrices tengan el mismo número de clumnas.
Ad=rbind(A,Ac)
Ad
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    0    1    2    4
## [2,]    2    3    7    8    5
## [3,]    4    0    1    2    4
## [4,]    2    3    7    8    5
## [5,]    4    7    6    2    1
```

```r
# Obtener dimensiones
dim(A)
```

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

```r
dim(Ac)
```

```
## [1] 3 5
```

```r
# Combinamos usando cbind, necesitamos que las matrices tengan el mismo número de clumnas.
Ad=rbind(cbind(Ac,Ac))
Ad
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    4    0    1    2    4    4    0    1    2     4
## [2,]    2    3    7    8    5    2    3    7    8     5
## [3,]    4    7    6    2    1    4    7    6    2     1
```

```r
# Indexar valores
Ad[3,1]
```

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

```r
# Matrices de caracteres
# Vector de letras
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x"
## [25] "y" "z"
```

```r
matlet<-matrix(letters,ncol=26,nrow=5) # R está reciclando el vector ¡5 veces!
matlet
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
## [1,] "a"  "f"  "k"  "p"  "u"  "z"  "e"  "j"  "o"  "t"   "y"   "d"   "i"   "n"   "s"   "x"   "c"
## [2,] "b"  "g"  "l"  "q"  "v"  "a"  "f"  "k"  "p"  "u"   "z"   "e"   "j"   "o"   "t"   "y"   "d"
## [3,] "c"  "h"  "m"  "r"  "w"  "b"  "g"  "l"  "q"  "v"   "a"   "f"   "k"   "p"   "u"   "z"   "e"
## [4,] "d"  "i"  "n"  "s"  "x"  "c"  "h"  "m"  "r"  "w"   "b"   "g"   "l"   "q"   "v"   "a"   "f"
## [5,] "e"  "j"  "o"  "t"  "y"  "d"  "i"  "n"  "s"  "x"   "c"   "h"   "m"   "r"   "w"   "b"   "g"
##      [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,] "h"   "m"   "r"   "w"   "b"   "g"   "l"   "q"   "v"
## [2,] "i"   "n"   "s"   "x"   "c"   "h"   "m"   "r"   "w"
## [3,] "j"   "o"   "t"   "y"   "d"   "i"   "n"   "s"   "x"
## [4,] "k"   "p"   "u"   "z"   "e"   "j"   "o"   "t"   "y"
## [5,] "l"   "q"   "v"   "a"   "f"   "k"   "p"   "u"   "z"
```

```r
matlet2<-matrix(letters,ncol=26,nrow=5,byrow=TRUE) # ¡R recicla el vector 5 veces!
matlet2
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]
## [1,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"   "n"   "o"   "p"   "q"
## [2,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"   "n"   "o"   "p"   "q"
## [3,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"   "n"   "o"   "p"   "q"
## [4,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"   "n"   "o"   "p"   "q"
## [5,] "a"  "b"  "c"  "d"  "e"  "f"  "g"  "h"  "i"  "j"   "k"   "l"   "m"   "n"   "o"   "p"   "q"
##      [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,] "r"   "s"   "t"   "u"   "v"   "w"   "x"   "y"   "z"
## [2,] "r"   "s"   "t"   "u"   "v"   "w"   "x"   "y"   "z"
## [3,] "r"   "s"   "t"   "u"   "v"   "w"   "x"   "y"   "z"
## [4,] "r"   "s"   "t"   "u"   "v"   "w"   "x"   "y"   "z"
## [5,] "r"   "s"   "t"   "u"   "v"   "w"   "x"   "y"   "z"
```

```r
# Importante: no podemos combinar números con caracteres o valores lógicos en la misma matriz.
ncol(matlet2)
```

```
## [1] 26
```

```r
?array
# La generalización de una matriz es un array que tiene más de 2 dimensiones.  Podemos tener matrices d
# matrices, por ejemplo mediciones en el tiempo.

# Datos como matriz
# Crear conjunto de datos
observNames<-c("I1", "I2")
vbleNames<-c("v1","v2")
class(vbleNames)
```

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

```r
vecH<-rnorm(4)
matH<-matrix(vecH,nrow=2)
rownames(matH)<-observNames
colnames(matH)<-vbleNames
dim(matH)
```

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

```
length(colnames)
```

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

```
length(vbleNames)
```

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

```
vecH2<-runif(4)
matH2<-matrix(vecH2,nrow=2)
dimnames(matH2)<-list(observNames,vbleNames)
str(matH2)
```

```
##  num [1:2, 1:2] 0.605 0.144 0.873 0.567
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2] "I1" "I2"
##   ..$ : chr [1:2] "v1" "v2"
```

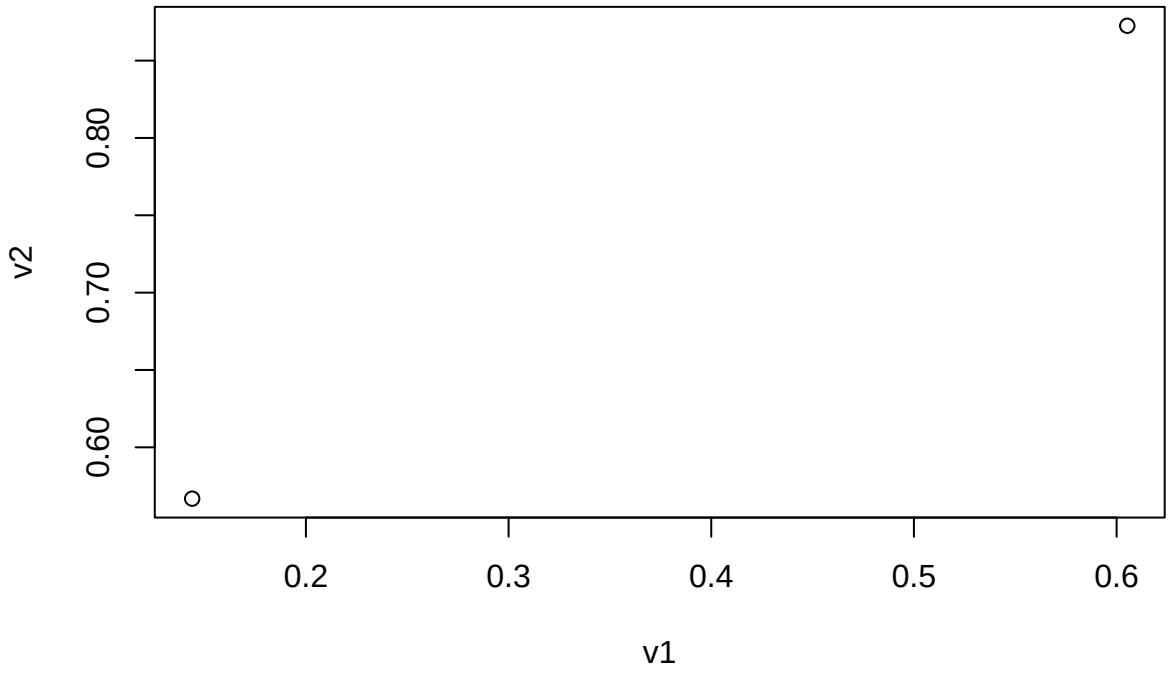
```
class(matH2)
```

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

```
summary(matH2) # Sabe qué tipo de unción utilizar a partir del tipo de objeto.
```

```
##        v1              v2
##  Min.   :0.1439   Min.   :0.5668
##  1st Qu.:0.2592   1st Qu.:0.6432
##  Median :0.3746   Median :0.7197
##  Mean   :0.3746   Mean   :0.7197
##  3rd Qu.:0.4899   3rd Qu.:0.7961
##  Max.   :0.6053   Max.   :0.8725
```

```
# Así, R puede ser considerado como un lenguaje OO.
?summary
?summary.matrix
plot(matH2) # Tomará las dos primeras columnas de la matriz
```



```
?plot
# Nota: list es para combinar objetos que no son del mismo
# tipo o longitud.  Podemos combinar objetos que no son homogéneos.
list.dirs()
```

```
## [1] "."
```

```
# Guardar matriz
save(matH2,file="matH2.Rdata") # Guardar en formato binario que solo R puede leer
write.table(matH2,file="matH2.txt") # Guardar en ascii formato que podemos inspeccionar sin necesidad d
file.show("matH2.txt")
# Transponer matrices
A
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    4    0    1    2    4
## [2,]    2    3    7    8    5
```

```
t(A)
```

```
##      [,1] [,2]
## [1,]    4    2
## [2,]    0    3
## [3,]    1    7
## [4,]    2    8
## [5,]    4    5
```

```
# Util cuando la base de datos se encuentra invertida (filas para variables, columnas para individuos).
# Si salimos de R, y queremos cargar el archivo guardado en formato Rdata
rm(list=ls())
getwd()
```

```
## [1] "/Users/moka/Dropbox/0.POST-PHD/GOALS/2.CODE/R/Ecomienza/6Matrices"
```

```
load("matH2.Rdata")

# Otras funciones
# as.matrix()
```

```
# Ejemplo de matriz
```

```
mujeres<-as.matrix(women)
# Esta base de datos se encuentra en el paquete datasets
?datasets # Ver que tiene el paquete datasets
?women # Ver la descripción de la base de datos women
class(women)
```

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

```
# No quitar el comentario de la linea inferior. Solamente copiar en la consola para que ejecute
#rmarkdown::render("6Matrices_cheatsheet.R",c("pdf_document","html_document"))
```