Programación Estadística: Arreglos

Adrián Sosa
Universidad Veracruzana

Arreglos

Existen diferentes tipos de Arreglos datos los cuales se mencionan a continuación:

```
* Vector
* Matriz
* Data frame
* Lista
* Series de tiempo
* Expresiones
```

Vector

Es un conjunto de datos ya sea númericos, lógicos o de caracter dependiendo como sean especificados en el argumento *mode*, conta de dos parametros "mode" y "length", este ultimo define la longitud del vector.

```
# Vector(mode="logical", length=0)
x <- vector(mode="logical", length=3)
print(x)
```

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

El argumento *mode* puede adquirir los siguientes valores:

```
* any
* list
* expression
* symbol
* pairlist
```

También puede ser utilizado en operadores lógicos de la siguiente manera:

```
as.vector(x, mode = "any")
```

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

```
is.vector(x, mode = "any")
```

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

Matriz

Una matriz es un vector con un atributo adicional (dim) el cual a su ves es un vector númerico de logitud 2, que define el número de filas y columnas de la matriz, el monado para quear este tipo de datos es *matrix*:

```
# matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
x <- matrix(data = NA, nrow = 2, ncol = 2, byrow = FALSE, dimnames = NULL)
print(x)
```

```
##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA
```

Los argumentos operan de la siguiente manera:

```
* data - Recibe la información que formara parte de la matriz.
* nrow - Número de filas.
* ncol - número de columnas.
* byrow - indica si los valores en data deben llenar las columnas sucesivamente(FALSE) o las fil
* dimnames - permite asignar nombres a las filas y columnas.
```

```
x <- matrix(data = 1:15, nrow = 5, ncol = 3, byrow = FALSE, dimnames= list(c("row1","row2","row3
print(x)
```

```
##      C1 C2 C3
## row1  1  6 11
## row2  2  7 12
## row3  3  8 13
## row4  4  9 14
## row5  5 10 15
```

```
x <- matrix(data = 1:15, nrow = 5, ncol = 3, byrow = TRUE, dimnames= list(c("row1","row2","row3"
print(x)
```

```
##      C1 C2 C3
## row1  1  2  3
## row2  4  5  6
## row3  7  8  9
## row4 10 11 12
## row5 13 14 15
```

Operaciones con matrices

```
x <- matrix(1:25, 5, 5, FALSE)
y <- matrix(1:25, 5, 5, TRUE)
x
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    6   11   16   21
## [2,]    2    7   12   17   22
## [3,]    3    8   13   18   23
## [4,]    4    9   14   19   24
## [5,]    5   10   15   20   25
```

```
y
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
## [4,]   16   17   18   19   20
## [5,]   21   22   23   24   25
```

```
x + y
```

Suma

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    8   14   20   26
## [2,]    8   14   20   26   32
## [3,]   14   20   26   32   38
## [4,]   20   26   32   38   44
## [5,]   26   32   38   44   50
```

```
x - y
```

Resta

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    4    8   12   16
## [2,]   -4    0    4    8   12
## [3,]   -8   -4    0    4    8
## [4,]  -12   -8   -4    0    4
## [5,]  -16  -12   -8   -4    0
```

```
x * y
```

Multiplicación

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1   12   33   64  105
## [2,]   12   49   96  153  220
## [3,]   33   96  169  252  345
## [4,]   64  153  252  361  480
## [5,]  105  220  345  480  625
```

```
x / y
```

División

```
##           [,1]      [,2]      [,3]      [,4]     [,5]
## [1,] 1.0000000 3.0000000 3.6666667 4.0000000 4.200000
## [2,] 0.3333333 1.0000000 1.5000000 1.8888889 2.200000
## [3,] 0.2727273 0.6666667 1.0000000 1.2857143 1.533333
## [4,] 0.2500000 0.5294118 0.7777778 1.0000000 1.200000
## [5,] 0.2380952 0.4545455 0.6521739 0.8333333 1.000000
```

Marco de datos

Un marco de datos o "Data.frame" se crea de manera implítica con la función *read.table*, de igual manera es posible hacerlo con la función *data.frame*.

```
x <- 1:4
n <- 10
data.frame(x,n)
```

```
##   x  n
## 1 1 10
## 2 2 10
## 3 3 10
## 4 4 10
```

Lista

Una lista se crea de manera similar a un marco de datos por medio d ela función *list*, puede incluir cualquier tipo de objetos, a diferencia del *data.frame* los nombres de los objetos no se toman por defecto.

```
L <- list(x,n)
L
```

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

Series de tiempo

La función *ts* crea un objeto de clase "ts" (serie de tiempo) a partir de un vector(serie de tiempo única) o una matriz(serie multivariada). los argumentos que recibe son:

```
ts(data= NA, start=1, end= numeric(0), frecuency=1, detat= 1, ts.sps=
getOption("ts.eps"), class, names)
```

```
    * data      -   Un Vector o matriz
    * start     -   El tiempo en la primera observación ya sea un número o un vector
                    con dos enteros
    * end       -   El tiempo de la última observación especificado de la misma
                    manera que *start*
    * frequency -   El número de observaciones por unidad de tiempo
    * deltat    -   Fracción del periodo de muestreo entre observaciones sucesivas
                    **solo especificar "frequency" o "deltat" **
    * ts.eps    -   Tolerancia para la comparación de series
    * class     -   Clase asignada el objeto
    * names     -   Para una serie multivariada, un vector de tipo caracter con los
                    nombres de las series individuales
```

```
ts(1:10, start=1959)
```

```
## Time Series:
## Start = 1959
## End = 1968
## Frequency = 1
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
ts(1:50, frequency= 12, start=c(1959,2))
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1959       1   2   3   4   5   6   7   8   9  10  11
## 1960  12  13  14  15  16  17  18  19  20  21  22  23
## 1961  24  25  26  27  28  29  30  31  32  33  34  35
## 1962  36  37  38  39  40  41  42  43  44  45  46  47
## 1963  48  49  50
```

```r
ts(1:10, frequency= 4, start=c(1959,2))
```

```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 1959         1    2    3
## 1960    4    5    6    7
## 1961    8    9   10
```

Expresiones

Los objetos de la clase *expresión* son de gran importancia en R. Euna *expresión* es una serie de caraacteres que hacen sentido para R, los comandos de R son puramente expresiónes, cuando se escribe un comando este es evaluado por R y ejecutado si resulta válido.

```r
x <- 3; y <- 2.5; z <- 1
exp1 <- expression(x/(y+ exp(z)))
exp1
```

```
## expression(x/(y + exp(z)))
```

```r
eval(exp1)
```

```
## [1] 0.5749019
```

Las expresiones se pueden usar, entre otras cosas, para incluir ecuaciones en graficos o como argumentos en ciertas funciones, por ejemplo *D()* que calcula derivadas aprciales:

```r
D(exp1,"x")
```

```
## 1/(y + exp(z))
```

```r
D(exp1,"y")
```

```
## -(x/(y + exp(z))^2)
```

```r
D(exp1,"z")
```

```
## -(x * exp(z)/(y + exp(z))^2)
```
