Comandos básicos de R

Manuel (mramon@jccm.es), 26 a 29 de abril de 2021

GUIÓN

- Operaciones básicas
- Creación de variables
- Vectores, Matrices y Listas

El directorio de trabajo en R

- R siempre se ejecuta en un directorio concreto de nuestro equipo
- Para saber cuál es nuestro directorio de trabajo usamos la función `getwd()`
- Para modificar el directorio de trabajo usamos la función `setwd()`. También podemos usar el menú Sesión, o las opciones en el explorador de archivos de RStudio
- Cuando leamos datos del disco, por defecto R buscará en nuestro directorio de trabajo. Si no están ahí, debemos especificar la ruta al directorio dónde se encuentran.

Operaciones aritméticas básicas con R

- Se pueden hacer directamente en la Consola y/o en un script
- R es como una calculadora. Se pueden hacer todas las operaciones aritméticas conocidas

Operación	Operador	Ejemplo	Resultados
Suma	+	2+2	4
Resta	-	3-1	2
Multiplicación	*	5*5	25
División	/	21/3	7
Potencia	^	3^2	9

Operaciones aritméticas básicas con R

- Ejemplo: `3^5 = 3*3*3*3*3 = 243`
- El orden de las operaciones aritméticas es importante. Por ejemplo:

  `3+3^2*2+1 = 3 + [(3^2)*2] + 1 = 22`

  `2+3*sqrt(25)/3 = ?`
- R usa por defecto el orden establecido: potencias > multiplicación/división > suma/resta. Si queremos otro orden, debemos definirlo usando paréntesis

  `(3+3)^2*(2+1) = ?`

Operaciones lógicas con R

- Permiten hacer comparaciones

Operación	Operador	Ejemplo	Resultados
Igual a	==	2==3	FALSE
No igual a	!=	2!=3	TRUE
Mayor que	>	6>6	FALSE
Mayor o igual a	>=	6>=6	TRUE
Menor que	<	2<5	TRUE
Menor o igual a	<=	2<=1	FALSE
y	&	(1==1) & (2==2)	TRUE
o	|	(1==1) | (2==3)	TRUE
no	!	!(1==1)	FALSE

Operaciones lógicas con R

```
R> 2 == 3  # ¿Es 2 igual a 3?
[1] FALSE
R> 2 != 3  # ¿Es 2 distinto de 3?
[1] TRUE
R> 6 > 6   # mayor que
[1] FALSE
R> 6 >= 6  # mayor igual a
[1] TRUE
R> 2 < 5   # menor que
[1] TRUE
R> "Mario" == "mario"
[1] FALSE
```

Creación/asignación de variables en R

- R puede almacenar cualquier valor en una variable
- Para la asignación se usa el operador de asignación `<-` (es equivalente al signo = y podría usarse indistintamente, pero es preferible usar la flecha)

```
no.alumnos <- 20
horas.lectivas <- 24
```

- Una vez creada, podemos operar con las variables

```
total.horas <- no.alumnos*horas.lectivas
```

Creación/asignación de variables en R

- R distingue mayusculas. No es igual `Uno` que `uno` que `uNO`
- Existen nombres reservados que no pueden usarse: `TRUE, FALSE, NA...`
- Los nombres de variables no pueden empezar por un número
- No se pueden usar espacios para nombrar variable (sustituir por "." ó "_"):

  `mi_valor, mi_valor, miValor`
- Usar nombres informativos y cortos

Tu turno!

Abre el script "`basics.R`" y ejecuta los comandos, prestando atención a las salidas.

Tipos de variables en R

- R reconoce varios tipos de variables
- Los números puedes ser enteros (`int`) y numerales (`num`)
- El texto puede ser caracter (`char`) o factor (`factor`)
- Tenemos también datos lógicos o boleados: `TRUE` y `FALSE`
- Los datos faltantes se codifican como `NA`

Tipos de variables en R

- Para saber de que tipo es un dato en R, tenemos la función `class()`
- Otra función MUY ULTIL es `str()` que permite conocer la estructura de un elemento

```
R> class("Juan")
[1] "character"
R> data(iris) # base de datos cargada en R, muy famosa
R> str(iris)
'data.frame':     150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Vectores en R

- Un vector es un conjunto de datos del mismo tipo: números, caracteres,…

$$Vector_{[1xN]} = [elemento1, elemento2, \ldots, elementoN]$$

- Para crear vectores en R usamos la función *combine* `c()`

```
datos <- c(1,2,3)
alumnos <- c("pablo","ana","carmen")
```

- Podemos obtener información del vector y acceder a sus elementos. Usamos corchetes [] para indicar la posición de los elementos que queremos acceder

```
length(alumnos) # nos dice cuantos elementos tiene el vector
datos[2] # accedemos al segundo elemento
1:50 # mira qué sale en la consola de R
```

Vectores en R

- Podemos alterar el orden de un vector

```
R> alumnos <- c("Pablo", "Ana", "Carmen")
R> alumnos
[1] "Pablo"  "Ana"    "Carmen"
R> alumnos[c(3,2,1)]
[1] "Carmen" "Ana"    "Pablo"
```

- Podemos modificar el valor de un elemento de un vector

```
R> alumnos[2] <- "Maria"
R> alumnos
[1] "Pablo"  "Maria"  "Carmen"
```

Vectores en R

- Podemos combinar vectores en otro vector

```
R> alumnos1 <- c("Pablo", "Carlos","Mario")
R> alumnos2 <- c("Ana", "Carmen", "Paula")
R> alumnos_all <- c(alumnos1, alumnos2)
R> alumnos_all
[1] "Pablo"  "Carlos" "Mario"  "Ana"    "Carmen" "Paula"
```

- Podemos operar con los elementos de un vector

```
R> datos*2 # multiplica por 2 cada elemento del vector
[1] 2 4 6
R> datos - datos # resta el primer elemento del primer vector con el primer
[1] 0 0 0        # elemento del segundo vector
```

Vectores en R

- Podemos operar con los elementos de un vector

```
R> alumnos_all
[1] "Pablo"  "Carlos" "Mario"  "Ana"    "Carmen" "Paula"
R> sort(alumnos_all) # ordena los elementos
[1] "Ana"    "Carlos" "Carmen" "Mario"  "Pablo"  "Paula"
```

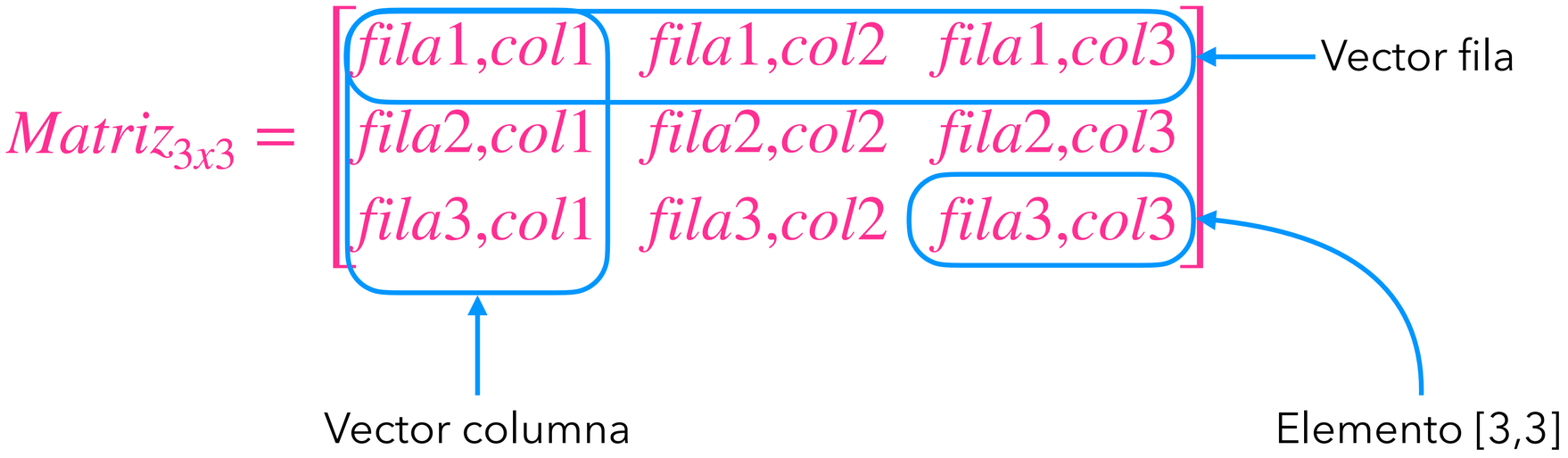
Vectores en R

- Podemos usar operadores lógicos

```
R> datos>1
[1] FALSE  TRUE  TRUE
R> datos[datos>1]
[1] 2 3
R> alumnos_all
[1] "Pablo"  "Carlos" "Mario"  "Ana"    "Carmen" "Paula"
R> alumnos.curso <- c(1,2,1,1,2,2)
R> alumnos.curso == 1
[1]  TRUE FALSE  TRUE  TRUE FALSE FALSE
R> alumnos_all[alumnos.curso == 1]
[1] "Pablo" "Mario" "Ana"
```

Matrices en R

- Una matriz es un conjunto de vectores del mismo tipo:



Matrices en R

- Para crear matrices en R podemos usar `matrix()`, `rbind()`, `cbind()`

```
R> m2 <- matrix(1:9, nrow = 3, ncol = 3)
R> m2  # matrix de 3x3
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrices en R

- Al igual que con los vectores, podemos obtener información de la matriz:

```
R> m3
     col1 col2 col3
row1    1    2    3
row2    4    5    6
row3    7    8    9
R> dim(m3)  # dimensión
[1] 3 3
R> nrow(m3) # número de filas
[1] 3
R> ncol(m3) # número de columnas
[1] 3
```

Matrices en R

- … o acceder a sus elementos. Usamos corchetes `[fila,columna]` para indicar la posición de los elementos que queremos acceder

```
R> m3
     col1 col2 col3
row1    1    2    3
row2    4    5    6
row3    7    8    9
R> m3[2,2]
[1] 5
R> m3[1, ]
col1 col2 col3
   1    2    3
R> m3[, 1]
row1 row2 row3
   1    4    7
```

Matrices en R

- … y también realizar operaciones

```
R> m3*2
     col1 col2 col3
row1    2    4    6
row2    8   10   12
row3   14   16   18
R> m3[1,] <- m3[1, ] * 2
R> m3
     col1 col2 col3
row1    2    4    6
row2    4    5    6
row3    7    8    9
```

Listas en R

- Una lista es un objeto que contiene elementos de diferentes tipos

$$Lista_{[1xN]} = [elemento1, elemento2, \ldots, elementoN]$$

- Cada elemento puede ser cualquier cosa: un número, un carácter, un vector, una matriz, una base de datos, etc.

```
R> lis1 <- list("Ana","Juan",1:3, NA, TRUE)
R> str(lis1)
List of 5
 $ : chr "Ana"
 $ : chr "Juan"
 $ : int [1:3] 1 2 3
 $ : logi NA
 $ : logi TRUE
```

Listas en R

- Para acceder a los elementos de una lista:

```
R> lis1[1]
[[1]]
[1] "Ana"
R> lis1[[1]]
[1] "Ana"
R> lis1[3]
[[1]]
[1] 1 2 3
R> lis1[[3]]
[1] 1 2 3
```

Listas en R

- Los elementos de la lista pueden tener nombre y usarse para acceder a dichos elementos

```
R> names(lis1)
[1] NULL
R> names(lis1) <- c("Alumna","Alumno","ciclos","Edades","Calificacion")
R> lis1
$Alumna
[1] "Ana"
$Alumno
[1] "Juan"
$ciclos
[1] 1 2 3
$Edades
[1] NA
$Calificacion
[1] TRUE
R> lis1$Alumno
[1] "Juan"
```

Tu turno!

Abre el script "`basics.R`" y ejecuta los comandos, prestando atención a las salidas. A continuación haz los ejercicios de "`basics_ex.R`"