

Ejercicio 2

R con RStudio



Índice

Experimentos:	2
Variables:	2
Vectores:	3
Operadores de Comparación:	4
Operadores lógicos en R (& = AND, = OR, != NO):	6
Funciones	8

Las variables sirven para almacenar un valor que luego vamos a utilizar en algún momento posterior, durante las operaciones realizadas en nuestro programa.

Ya hemos visto los operadores matemáticos, que permiten realizar operaciones y construir expresiones matemáticas de distinta clase. En los lenguajes de programación, son verdaderamente importantes los operadores que se introducen aquí, los operadores de comparación y los operadores lógicos. Estos permiten construir expresiones lógicas, que se evalúan en bloques de control condicionales (sentencias if) e iterativos (sentencias for y while).

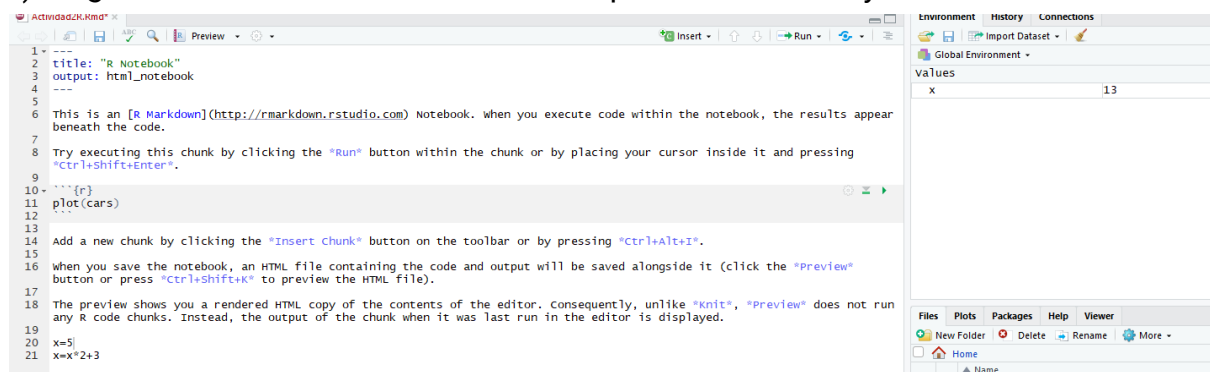
También veremos lo más básico de los vectores, y el uso de las funciones, especialmente las utilizadas con los vectores.

Experimentos:

1- Realiza las siguientes actividades. Recuerda que para mostrar el valor de las variables, sólo debes escribir su nombre en la consola y pulsar la tecla "INTRO".

Variables:

a) Asigna en una variable x el valor 5. Multiplica esa variable y suma un 3.



b) Crea otra variable numérica y, con el valor 8.5. Suma x e y, y guarda el resultado en la variable z. Muestra el valor de z por pantalla

```
> x=x*2+3
> y=8.5
> z=x+y
> print(z)
[1] 21.5
> |
```

c) Crea la variable país y se almacena el nombre Colombia. Luego averigua el número de caracteres de la variable país. Para eso llama a la función nchar(), pasando como parámetro a variable país.

```
> país="Colombia"
> nchar(país)
[1] 8
> |
```

Vectores:

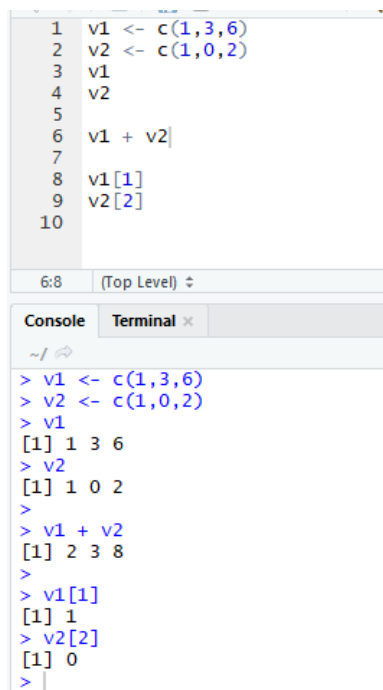
d)

```
# Define el vector [1,3,6] v1
v1 <- c(1,3,6)
v1 # Muéstralo por pantalla
```

```
# Definimos el vector [1,0,2] en v2
v2 <- c(1,0,2)
v2 # Muéstralo por pantalla
v1 + v2 # Sumamos los vectores
```

Accedemos a las posiciones de los vectores. Teniendo en cuenta que los vectores

```
#empiezan en la posición 1. v1[i] es el valor i-ésimo del vector.
v1[1] # Posición 1 del vector v1, que mostrará el valor 1
v2[2] # Posición 2 del vector v2, que mostrará el valor 0
```



```
1 v1 <- c(1,3,6)
2 v2 <- c(1,0,2)
3 v1
4 v2
5
6 v1 + v2
7
8 v1[1]
9 v2[2]
10
```

6:8 (Top Level) ↕

Console Terminal ×

```
> v1 <- c(1,3,6)
> v2 <- c(1,0,2)
> v1
[1] 1 3 6
> v2
[1] 1 0 2
>
> v1 + v2
[1] 2 3 8
>
> v1[1]
[1] 1
> v2[2]
[1] 0
> |
```

Operadores de Comparación:

e) # igualdad en R

3 == 3 # (devuelve VERDADERO)

3.01 == 3 # (devuelve FALSO)

#Prueba con las variables

x==y

#Prueba con los vectores

v1==v2

```
> 3==3
[1] TRUE
> 3.01==3
[1] FALSE
> x==y
[1] FALSE
> v1==v2
[1] TRUE FALSE FALSE
> |
```

f) # desigualdad en R

2 != 1 # (devuelve VERDADERO)

2 != 2 # (devuelve FALSO)

#Prueba con las variables

x!=y

#Prueba con los vectores

v1!=v2

```
> 2 != 1
[1] TRUE
> 2 != 2
[1] FALSE
>
> x != y
[1] TRUE
> v1 != v2
[1] FALSE TRUE TRUE
> |
```

g) # Términos comparativos en R:

5 > 10 # 5 es mayor que 10 (devuelve FALSO)

5 > 5 # 5 es mayor que 5 (devuelve FALSO)

5 >= 5 # 5 es mayor o igual que 5 (devuelve VERDADERO)

5 < 10 # 5 es menor que 10 (devuelve VERDADERO)

5 < 5 # 5 es menor que 5 (devuelve FALSO)

5 <= 5 # 5 es menor o igual que 5 (devuelve VERDADERO)

#Prueba con las variables

x > y

x >= y

y < x

y <= x

#Prueba con los vectores

v1 > v2

v1 < v2

```
> 5 > 10
[1] FALSE
> 5 > 5
[1] FALSE
> 5 >= 5
[1] TRUE
> 5 < 10
[1] TRUE
> 5 < 5
[1] FALSE
> 5 <= 5
[1] TRUE
>
> x > y
[1] TRUE
> x >= y
[1] TRUE
> y < x
[1] TRUE
> y <= x
[1] TRUE
> v1 > v2
[1] FALSE TRUE TRUE
> v1 < v2
[1] FALSE FALSE FALSE
`
```

Operadores lógicos en R (& = AND, | = OR, != NO):

h) # Definimos variable de trabajo

```
a = 5
```

```
a < 2 # (devuelve FALSO)
```

```
a > 2 # (devuelve VERDADERO)
```

#Verificar dos condiciones a la vez en R.

Utilizamos el símbolo &, que sería el equivalente al «AND» de otros programas.

```
> a = 5
> a > 2 & a < 6
[1] TRUE
>
```

a es menor que 10 y a es menor que 2

```
(a<10) & (a<2) # (devuelve FALSO)
```

a es menor que 10 y a es menor que 6

```
(a<10) & (a<6) # (devuelve VERDADERO)
```

```
> (a<10) & (a<2)
[1] FALSE
> (a<10) & (a<6)
[1] TRUE
>
```

#Verificar en R si se cumple una condición u otra

#Utilizamos el símbolo |, que sería equivalente al «OR» de otros programas.

```
(a<10) | (a<2) # (devuelve FALSO)
```

```
(a<10) | (a<6) # (devuelve VERDADERO)
```

#Negación en R

```
!a < 10 # (a NO es menor que 10 - devuelve FALSO)
```

```
a < 10 # (a es menor que 10 - devuelve VERDADERO)
```

Se pueden utilizar distintos operadores al mismo tiempo:

a NO es MAYOR que 6, y a NO es MAYOR que 10

```
!(a>6) & !(a>10) # VERDADERO
```

a NO es MAYOR que 6, y a NO es MENOR que 10

```
!(a>6) & !(a<10) # FALSO
```

```
> (a<10) | (a<2)
[1] TRUE
> (a<10) | (a<6)
[1] TRUE
> !a < 10
[1] FALSE
> a < 10
[1] TRUE
> !(a>6) & !(a>10)
[1] TRUE
> !(a>6) & !(a<10)
[1] FALSE
>
```

2- a) Crea un vector denominado vt con 10 posiciones, introduciendo los valores numéricos del 1 al 10. Utiliza las siguientes funciones sobre el vector, y averigua qué cálculos realiza sobre el vector.

length	sum	prod	max	min	range	mean
median	summary	var	sd	sort	which.min	which.max
rev						

```
> lucio <- c(1,2,3,4,5,6,7,8,9,10)
> length(lucio)
[1] 10
> sum(lucio)
[1] 55
> prod(lucio)
[1] 3628800
> max(lucio)
[1] 10
> min(lucio)
[1] 1
> range(lucio)
[1] 1 10
> mean(lucio)
[1] 5.5
> median(lucio)
[1] 5.5
> summary(lucio)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   3.25   5.50   5.50   7.75  10.00
> var(lucio)
[1] 9.166667
> sd(lucio)
[1] 3.02765
> sort(lucio)
[1] 1 2 3 4 5 6 7 8 9 10
> which.min(lucio)
[1] 1
> which.max(lucio)
[1] 10
> rev(lucio)
[1] 10 9 8 7 6 5 4 3 2 1
> |
```

Para llamar a una función pasándole como parámetro el vector vt, tendrás que utilizar el nombre de la función y a continuación, escribir el nombre del vector entre paréntesis. P.ej:

nombrefuncion(vt)

b) Prueba las funciones anteriores teniendo en cuenta que también es posible hacer operaciones con el vector, y que ese sea el parámetro de la función. Por ejemplo:
`nombrefuncion(vt * 2)`

```
> nombrefuncion<-function(l){
+ length(l)
+ sum(l)
+ prod(l)
+ max(l)
+ min(l)
+ range(l)
+ mean(l)
+ median(l)
+ summary(l)
+ var(l)
+ sd(l)
+ sort(l)
+ which.min(l)
+ which.max(l)
+ rev(l)
+ }
> nombrefuncion(lucio*2)
[1] 20 18 16 14 12 10 8 6 4 2
> |
```

Funciones

c) También es posible aplicar una función al vector, y que el resultado sea el parámetro de otra función. Por ejemplo.

`nombrefuncion1(nombrefuncion2(vt))`

Si `nombrefuncion2(vt)` retorna un único valor numérico, no podrás aplicar una función de vectores. Prueba a aplicar las funciones que vimos la semana pasada.

- `sqrt`: Raíz cuadrada
- `sin`: Seno
- `log10`: Logaritmo en base

```
> nombrefuncion2<-function(n){
+ sqrt(n)
+ sin(n)
+ log10(n)
+ }
>
> nombrefuncion(nombrefuncion2(lucio))
[1] 1.0000000 0.9542425 0.9030900 0.8450980 0.7781513 0.6989700 0.6020600 0.4771213 0.3010300 0.0000000
> |
```