Sesion 2

Juan José Merino Zarco

22/6/2021

Nota importante

Cada vez que inician un nuevo trabajo, pueden notar que los objetos que crearon en otros archivos pueden (o no) seguir en el apartado de "Entorno de trabajo de RStudio" (el apartado superior que esta a la derecha), por lo cual es recomendable agregar el comando: rm(list=ls()) en el primer chunk de cada nuevo archivo, el cual limpia del entorno de trabajo, alternativamente, pueden presionar el símbolo de la escoba que se encuentra en el apartado de "Entorno de trabajo".

```
rm(list=ls())
```

GitHub

- Crear una cuenta en GitHub
- Explorar la pagina GitHub
  - Repositorios
  - Fork

Uso de help()

```
# help()
```

Manejo de datos

Tipos de datos

Tipo	Ejemplo
numeric	3.1416
character	“pi”
logical	TRUE, FALSE
NA	NA
null	null

```
class(1)
```

```
## [1] "numeric"
```

Operadores

Operadores de asignacion

Creación de un objeto

resultados temporales

```
2+2
```

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

Nota: Un objeto guarda los resultados temporales

```
x = 3 + 2
x
```

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

```
y <- 2 + 2
y
```

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

Operadores aritmeticos

R admite los siguientes operadores aritméticos:

```
x <- 2
y <- 5
```

Suma

```r
z <- x + y
z
```

```
## [1] 7
```

Resta

```r
z <- x - y
z
```

```
## [1] -3
```

Multiplicacion

```r
z <- x * y
z
```

```
## [1] 10
```

Division

```r
z <- x/y
z
```

```
## [1] 0.4
```

Potencias

```r
z <- x^2
g <- x^(1/2)
z
```

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

```r
g
```

```
## [1] 1.414214
```

Adicional Se pueden realizar las siguientes transformaciones

Logaritmo natural

```r
log(10)
```

```
## [1] 2.302585
```

Logaritmo base 10

```
log10(10)
```

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

Raiz cuadrada

```
sqrt(4)
```

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

Operadores relacionales

Al utilizar estos operadores, el resultado que nos arrojaran sera **TRUE** o **FALSE**

Menor que

```
4 < 2
```

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

Menor o igual que

```
4 <= 2
```

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

Mayor que

```
4 > 2
```

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

Mayor o igual que

```
4 >= 2
```

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

Exactamente igual que

```
4==2
```

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

No es igual que

```
4 != 2
```

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

Operadores logicos

Operador	Comparacion
x|y	x O y es verdadero
x & y	x Y y son verdaderos
!x	x no es verdadero

Ejemplos

|

```
2 > 3 | 4 < 2
```

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

```
3 > 2 | 4 < 2
```

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

```
3 > 2 | 4 > 2
```

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

&

```
2 > 3 & 4 < 2
```

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

```
3 > 2 & 4 < 2
```

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

```
3 > 2 & 4 > 2
```

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

!

```
! (2 > 3)
```

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

```
! (2 < 3)
```

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

Orden de operaciones

Orden	Operador
1	^
2	* /
3	+ -
4	< , > , <= , >= , == , !=
5	!
6	&
7	|
8	<- , =

Adicionalmente podemos usar parentesis "()" para que una operacion ocurra antes que otra.

Estructura de datos

Dimensiones	Homogeneas	Heterogeneas
1	Vector	
2	Matriz	Dataframe

Vectores

Creando un vector

```
v <- c(1, 2, 3, 4)
v
```

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

```
v <- c(1:4)
v
```

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

Largo

```
length(v)
```

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

Verificar que sea un vector

```
is.vector(v)
```

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

Para modificar un vector existente

```
v <- c(v, 5)
v
```

```
## [1] 1 2 3 4 5
```

Uniendo 2 vectores distintos

```
v <- c(1:4)
b <- c(4:5)
v <- c(v,b)
v
```

```
## [1] 1 2 3 4 4 5
```

Operaciones con vectores

```
vec <- c(1:5)
vec
```

```
## [1] 1 2 3 4 5
```

Operaciones aritmeticas

```
vecar <- vec + 2
vecar
```

```
## [1] 3 4 5 6 7
```

```
vecar <- vec * 2
vecar
```

```
## [1]  2  4  6  8 10
```

Operadores relacionales

```
vec > 3
```

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

```
vec != 3
```

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

Multiplicacion de vectores

Advertencia:

```
va <- c(1:3)
vb <- c(1:3)
```

```
va %*% vb
```

```
##      [,1]
## [1,]   14
```

```
va %*% t(vb)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    2    4    6
## [3,]    3    6    9
```

```
t(va) %*% vb
```

```
##      [,1]
## [1,]   14
```

```
t(va) %*% t(vb)
```

Matrices

Creando una matriz

```
v1 <- c(1:3)
v2 <- c(4:6)
v3 <- c(1,3,5)
```

Usando cada vector como columna

```
m1 <- cbind(v1,v2,v3)
m1
```

```
##      v1 v2 v3
## [1,]  1  4  1
## [2,]  2  5  3
## [3,]  3  6  5
```

Usando cada vector como un renglon

```
m2 <- rbind(v1,v2,v3)
m2
```

```
##    [,1] [,2] [,3]
## v1    1    2    3
## v2    4    5    6
## v3    1    3    5
```

Dimension de la matriz

```
dim(m2)
```

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

Operaciones con matrices

```
m1 + 1
```

```
##      v1 v2 v3
## [1,]  2  5  2
## [2,]  3  6  4
## [3,]  4  7  6
```

```
m1^2
```

```
##      v1 v2 v3
## [1,]  1 16  1
## [2,]  4 25  9
## [3,]  9 36 25
```

Note que:

```
(m1 %*% m1) == (m1^2)
```

```
##         v1    v2    v3
## [1,] FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE
```

Transponer una matriz

```
t(m1)
```

```
##    [,1] [,2] [,3]
## v1    1    2    3
## v2    4    5    6
## v3    1    3    5
```

Podemos comprobar su transpuesta si:

```
t(m1) == m2
```

```
##    [,1] [,2] [,3]
## v1 TRUE TRUE TRUE
## v2 TRUE TRUE TRUE
## v3 TRUE TRUE TRUE
```

```
m1 + m2
```

```
##      v1 v2 v3
## [1,]  2  6  4
## [2,]  6 10  9
## [3,]  4  9 10
```

```
m1 - m2
```

```
##      v1 v2 v3
## [1,]  0  2 -2
## [2,] -2  0 -3
## [3,]  2  3  0
```

Multiplicaciones de matrices

```
m1 %*% m2
```

```
##      [,1] [,2] [,3]
## [1,]   18   25   32
## [2,]   25   38   51
## [3,]   32   51   70
```

Inversa de una matriz

```
a1 <- c(1,1)
a2 <- c(2,7)
mtz <- cbind(a1,a2)
mtz
```

```
##      a1 a2
## [1,]  1  2
## [2,]  1  7
```

```
solve(mtz)
```

```
##    [,1] [,2]
## a1  1.4 -0.4
## a2 -0.2  0.2
```

Recursos adicionales

R para principiantes, para el manejo de datos y bases de datos

https://bookdown.org/jboscomendoza/r-principiantes4/datos-mas-comunes.html