Programación para Data Science I

Temas:

- Primeros pasos: comandos, objetos, funciones, script.
- Clases de Objetos: numéricos, alfanuméricos, vectores, matrices, listas, Data Frames
- Instalación de paquetes y librerías.
- Tipo de operadores: operacionales, relacionales y lógicos
- Estructuras de control: Composición secuencial. Instrucción compuesta. Bucles e iteraciones.

1 Actividades

Les proponemos realizar las siguientes actividades para que se familiaricen con el entorno de RStudio y hagan su primer contacto con el programa R. Los ejercicios propuestos son simples y hasta puede ser que los encuentre sin-sentido pero son la excusa perfecta para comprender su funcionamiento del entorno R.

Ejercicio 1- Usando R como calculadora

En la consola de RStudio realiza los siguientes cálculos y observa los resultados

4+5

27-11

3*9

23/4

23%/%4

23%%4

2/3+4/7

sqrt(36)

sqrt(79)

Ejercicio 2- Más cálculos

Te proponemos que calcules la raíz cuadrada de 144, dividido la raíz cubica de 27.
(Ayudita: la raíz cúbica hay que crearla con exponenciación)
Calcular el valor absoluto del resultado de: 356 menos 366, multiplicado por el valor pi. Redondear el resultado con dos decimales.

Ejercicio 3- Mi primer Script

En las primeras líneas genera un comentario indicando tu Nombre completo y la fecha.
Luego, escribe el código que borra todo lo que se encuentra en el entorno cargado en la memoria.
Por último, especifica tu directorio de trabajo. Guarda el Script con el nombre “Actividad_1.R”
Escribir las siguientes líneas en el editor de código y ejecutarlas.
No olvides ir guardando el contenido del script.

```
a <- 5
b <- 4
c <- a + b
a <- b * c
b <- (c - a)^2
c <- a * b
```

(Ayuda: puedes copiar y pegar las líneas.)
Finalmente: Cuánto valen los objetos a, b y c ?

Ejercicio 4- Trabajando con vectores

i) Crea los objetos x e y ejecutando las siguientes líneas de código.

```
x <- c(1,3,5,7,9)
y <- c(2,3,5,7,11,13)
```

ii) Suma 1 a cada elemento del vector x
iii) Suma 1 en cada elemento del vector y
iv) Al vector y multiplícalo por 3
v) Suma la cantidad de elementos del vector x e y
vi) Sumar x+y

Ejercicio 5- Operaciones con vectores

i) Escribe un vector con los números consecutivos del 1 al 10

ii) Escribe un vector de números consecutivos decrecientes a partir del 10

iii) Otra forma de crear vectores es utilizando función *"seq(a,b,c)"*, que genera secuencias de números reales, donde el primer elemento indicara el principio de la secuencia, el segundo el final y el tercero el incremento que se debe usar para generar la secuencia. También podemos poner la función de estas formas: *"seq(length = d; from = a; to = b)"* o *"seq(by = c; from = a; to = b)"* siendo "d" la longitud del vector.

Te proponemos que usando esas funciones crees un vector con la secuencia del 1 al 10 de números impares, y otro vector con una secuencia del 1 al 10 que contenga 6 elementos (usa el argumento *length*)

iv) Con la función *"rep(a; b)"* podemos crear un vector con "b" elementos idénticos al valor "a".

Crea un vector que repita 10 veces el elemento 5.

Repetir cada elemento de un vector 3 veces de en 5.

v) Cree el vector llamado meses que contenga los meses del año. Luego llamelo para verlo en la consola

vi) Obtenga la estructura del vector meses con str(). Qué nos indica?

vii) Obtenga el modo y la longitud del vector meses

Ejercicio 6- Más vectores

Escriba las instrucciones para generar:

1. Un objeto llamado **x** formado por la secuencia de los 5 primeros números pares.
2. Un objeto llamado **y** con los 10 primeros números naturales en orden decreciente.
3. Una secuencia del 1 a 15 en 7 pasos
4. Un vector de 10 elementos, todo ellos de valor 4.
5. Un objeto llamado **días** que contenga los días de la semana.

Tras esto, escriba las secuencias para obtener

- la longitud del vector **días** y
- para seleccionar el elemento de la posición 3 del vector **días**.

Ejercicio 7- Trabajando con listas

Por último, genere una lista con los siguientes elementos:

```
$lugar
[1] "Granja"

$nombre
[1] "La vaca Lola"

$numero.animales
[1] 6

$animales
[1] "Gallo" "Gallina"  "Conejo" "Caballo" "Perro" "Vaca"

$cantidad
[1] 1 8 6 4 2 1
```

Resultados

Ejercicio 1- Usando R como calculadora

```
> 4+5
[1] 9

> 27-11
[1] 16

> 3*9
[1] 27

> 23/4
[1] 5.75

> 23%/%4
[1] 5

> 23%%4
[1] 3

> 2/3+4/7
[1] 1.238095

> sqrt(36)
[1] 6

> sqrt(79)
[1] 8.888194

> sin(pi)
[1] 1.224606e-16

> sin(3.14)
[1] 0.001592653
```

Ejercicio 2- Más cálculos

```
> sqrt(144) / (27^(1/3))
[1] 4

> round(abs((356-366)*pi),2)
[1] 31.42
```

Ejercicio 3 -Mi primer Script

```
> setwd("C:/CURSO_R_CAVILA") ## elije el alumno su ruta

> getwd()                    ## verifico la ruta
[1] "C:/CURSO_R_CAVILA"
```

```
> a <- 5
> b <- 4
> c <- a + b
> a <- b * c
> b <- (c - a)^2
> c <- a * b
```

Para saber cuánto vales a, b, y c

```
> a
[1] 36
> b
[1] 729
> c
[1] 26244
```

Ejercicio 4- Trabajando con vectores

i) Crea los objetos **x** e **y** ejecutando las siguientes líneas de código.

```
> x <- c(1,3,5,7,9)
> y <- c(2,3,5,7,11,13)
```

ii) Suma 1 a cada elemento del vector x

```
> x+1
[1]  2  4  6  8 10
```

iii) Suma 1 en cada elemento del vector y

```
> y+1
[1]  3  4  6  8 12 14
```

iv) Al vector y multiplícalo por 3

```
> y*3
[1]  6  9 15 21 33 39
```

v) Suma la cantidad de elementos del vector x e y

```
> length(x) + length(y)
[1] 11
```

vi) Sumando los vectores

```
> x+y
[1]  3  6 10 14 20 14
Warning message:
In x + y : longer object length is not a multiple of shorter object length
```

Nota: Como la longitud de los vectores es diferente, nos devuelve un aviso de error. De todas maneras realiza igualmente la suma, pero al llegar al último elemento del objeto más largo (y) hace la suma con el primer elemento de x, es decir vuelve a iniciar el vector.

Ejercicio 5- Operaciones con vectores

i)
```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

ii)
```
> 10:1
 [1] 10  9  8  7  6  5  4  3  2  1
```

Nota: el vector comienza en el primer número suministrado y finaliza en el segundo o en un número anterior sin sobrepasarlo, tanto en orden ascendente como descendente)

iii)
```
> seq(1,10,2)
[1] 1 3 5 7 9

> seq(from = 1, to = 10, length =6)
[1]  1.0  2.8  4.6  6.4  8.2 10.0
```

iv)
```
> rep(5,10)
 [1] 5 5 5 5 5 5 5 5 5 5

> rep(1:3,rep(5,3))
 [1] 1 1 1 1 1 2 2 2 2 3 3 3 3 3
```

v)
```
> meses<- c ( "Enero","Febrero","Marzo","Abril","mayo","Junio","Julio","Agosto
","Septiembre","Octubre","Noviembre","Diciembre")


> meses
 [1] "Enero"      "Febrero"    "Marzo"      "Abril"      "mayo"
 [6] "Junio"      "Julio"      "Agosto"     "Septiembre" "Octubre"
[11] "Noviembre"  "Diciembre"
```

vi)
```
> str(meses)
 chr [1:12] "Enero" "Febrero" "Marzo" "Abril" "mayo" "Junio" "Julio" ...
```

vii)
```
> mode(meses)
[1] "character"
```

```
> length(meses)
[1] 12
```

Ejercicio 6: Más vectores

```
>x<-c(2,4,6,8,10) # 5 primeros numeros pares
>x
[1]  2  4  6  8 10
>x <- seq( from = 2, by = 2, length = 5) # otra forma de resolver
>x
[1]  2  4  6  8 10

>y<-10:1 # numeros naturales en orden decreciente.
>y <- (10:1)   # otra forma, agregando los ()
[1] 10  9  8  7  6  5  4  3  2  1

>seq(1,15,length=7)
[1]  1.000000  3.333333  5.666667  8.000000 10.333333 12.666667 15.000000

>seq( from = 1, to = 15, by = 2)  # otra forma de resolver
[1]  1  3  5  7  9 11 13 15

>rep(4,10) # resuelto de una manera facil y corta
[1] 4 4 4 4 4 4 4 4 4 4
>vector_num <- rep(4, time=10) # otra forma de resolverlo
>vector_num
[1] 4 4 4 4 4 4 4 4 4 4

>dias<-c("Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo")
>dias
[1] "Lunes" "Martes"  "Miercoles" "Jueves"  "Viernes"  "Sabado"  "Domingo"

>length(dias) # longitud del vector dias
[1] 7
>dias[3] # dia en la tercera posicion
[1] "Miercoles"
```

Ejercicio 7: Trabajando con listas

```
>milista <- list(Lugar = "Granja", Nombre = "La Vaca Lola", numero.animales =
6, animales = c("Gallo", "Gallina", "Conejo","Caballo","Perro","Vaca"), cantidad
= c(1, 8, 6, 4, 2))
>milista
```

```
$Lugar
[1] "Granja"

$Nombre
[1] "La Vaca Lola"

$numero.animales
[1] 6

$animales
[1] "Gallo"    "Gallina" "Conejo"   "Caballo" "Perro"    "Vaca"

$cantidad
[1] 1 8 6 4 2
```

Nota: Para borrar objetos almacenados en la memoria, utilizamos la función *"rm()"*, por ejemplo *rm(meses)* elimina el objeto meses; rm(x; y) elimina ambos objetos x e y, con *"rm(list = ls( ))"* podemos eliminar todos los objetos que están en la memoria. Tenga en cuenta que las opciones mencionadas para la función *ls( )* pueden aplicarse para borrar selectivamente algunos objetos.