

Sección 4 - Programación

Programación en general

Sección 4.1: Entender algunas de las capacidades para programar en R

Sección 4.2: Utilizar condicionales lógicos

Sección 4.3: Definir y llamar a funciones

Sección 4.4: Realizar loops con operaciones

Introducción

- Veremos los 3 elementos claves en programación:
 - Condicionales
 - For-loops
 - Funciones

Condicionales

- Son el elemento más básico de la programación
- La condicional más común es la expresión if-else

```
if (condición lógica booleana) {  
    expresiones  
} else {  
    expresiones alternativas  
}
```

Ejemplo if-else

```
if (a != 0) {  
    print (1/a)  
} else {  
    print ("No hay inverso de 0.")  
}
```

if-else en dataframe

```
library(dslabs)
data(murders)
murder_rate<-
murders$total/murders$population*100000

ind<-which.min(murder_rate)
if(murder_rate[ind]<0.5) {
  print(murders$state[ind])
}else{
  print("No hay estado que tenga un
  índice tan bajo")
}
[1] "Vermont"
```

ifelse

- Esta función recibe tres argumentos: uno lógico y dos posibles respuestas.
 - Si el logico es verdadero retorna la primer respuesta, si es falso retorna la segunda

```
ifelse(a > 0, 1/a, NA)
```

ifelse con vectores

Examina cada elemento de un vector lógico y retorna la respuesta correspondiente

```
a <- c(0, 1, 2, -4, 5)
result <- ifelse(a > 0, 1/a, NA)
```

a	Condición	Respuesta 1	Respuesta 2	Resultado
0	FALSO	Inf	NA	NA
1	VERDADERO	1	NA	1
2	VERDADERO	0.5	NA	0.5
-4	FALSO	0.25	NA	NA
5	VERDADERO	0.2	NA	0.2

Para reemplazar los NA por ceros

```
> a <- c(0,1,2,-4,5)
> result <- ifelse(a > 0, 1/a, NA)
> result
[1] NA 1.0 0.5 NA 0.2
> sin_na<-ifelse(is.na(result),0,result)
> sin_na
[1] 0.0 1.0 0.5 0.0 0.2
```

Funciones: any y all

Any: Recibe un vector de booleanos y retorna TRUE si alguna de la entrada es TRUE.

All: Recibe un vector de valores booleanos y retorna TRUE si todos los elementos son TRUE y FALSE si no.

```
> z <- c(TRUE, TRUE, FALSE)
> any(z)
[1] TRUE
> all(z)
[1] FALSE
```

Loop For

Para ejecutar varias veces un código se utiliza una función llamada for()

```
for ( i in rango de valores) {  
    expresiones que utilizan i,  
    esta variable cambia tomando  
    todos los valores del rango  
}
```

Ejemplo - For

Creamos un loop para imprimir los valores de la variable que va cambiando

```
for(i in 1:5) {  
    print(i)  
}
```


Bucle while()

- **MIENTRAS** la condición sea **VERDADERA** ejecutar operaciones

```
while (condicion) {  
    operaciones  
}
```

Ejemplo While()

```
i <- 0
while (TRUE) {
  if (i == 10)
    break
  else {
    i <- i+1
    print(i)
  }
}
```




[1]	1
[1]	2
[1]	3
[1]	4
[1]	5
[1]	6
[1]	7
[1]	8
[1]	9
[1]	10

Usando next

- Para 'saltarnos' una iteración en un bucle

```
for(i in 1:4) {  
  if(i == 3) {  
    next  
  }  
  print(i)  
}
```



[1]	1
[1]	2
[1]	4

Función repeat

- Bucle que se lleva a cabo el número de veces que especifiquemos, usando un break para detenerse

```
repeat {  
    operaciones  
  
    un_break_para_detener  
}
```


Ejemplo

```
valor <- 0
mi_vector <- NULL

repeat{
  valor <- valor + 1
  if(valor == 5) {
    break
  }
}

valor
## [1] 5
```

Funciones

```
mifuncion <- function(arg1, arg2=10, ...) {  
  cuerpo  
  resultado  
}
```

- Función suma:

```
> suma<-function(x=1,y=2) {  
+   x+y  
+ }
```

Función con cat()

```
> suma<-function(x=1,y=2) {  
+   # suma de los elementos "x" e "y"  
+   result <- x+y  
+   cat(x," sumado a ", y, "es:\n", result)  
+ }
```

```
> suma()  
1  sumado a  2 es:  
3
```

```
> suma(3)  
3  sumado a  2 es:  
5
```

```
> suma(,3)  
1  sumado a  3 es:  
4
```

```
> suma(1:3,4:6)  
1 2 3  sumado a  4 5 6 es:  
5 7 9
```

Función return()

- Para obtener un resultado de un paso en particular no necesariamente el último.

```
> suma<-function(x=1,y=2) {  
+   if(is.character(y)) return("y debe ser numérico")  
+   result <- x+y  
+   cat(x,"sumado a ", y, "es:\n", result)  
+ }  
  
> suma(2,"hola")  
  
[1] "y debe ser numérico"
```

Crear funciones en archivo.R

- Desde tu directorio

```
> source( "mifuncion.R")
```

- Para acceder a una función que se encuentra en la web

```
> source(  
  "https://raw.githubusercontent.com/nanirhapsody/cursor/blob/main/ejercicios%20S3U3/ej1.R" )
```

Funciones

Creamos una fórmula para calcular la suma de n enteros

```
computa_s_n <- function(n) {  
  x <- 1:n  
  sum(x)  
}
```

```
computa_s_n(3)
```

```
[1] 6
```

```
computa_s_n(100)
```

```
[1] 5050
```

Ejemplo For - Función

```
m <- 25
```

```
#Creamos un vector vacío
```

```
s_n <- vector(length = m)
```

```
for(n in 1:m) {
```

```
  s_n[n] <- computa_s_n(n)
```

```
}
```

Comparamos resultados

Fórmula que calcula la suma de enteros de 1 a n

$$1 + 2 + \dots + n = \frac{n(n + 1)}{2}$$

Creamos una tabla con ambos resultados

```
> head(data.frame(s_n = s_n, formula = n*(n+1)/2))
  s_n formula
1    1      1
2    3      3
3    6      6
4   10     10
5   15     15
6   21     21
```


Comparación gráfica de resultados

```
plot(n, s_n)  
lines(n, n*(n+1)/2)
```

