BUCLE “FOR”

INDICE

¿Qué es?

Se trata de un comando que permite repetir una serie de operaciones un determinado número de veces.

```
for (variable in secuencia) {
        # Código a ejecutar en cada iteración
}
```

En el apartado de variable, introduciríamos una variable (i, por ejemplo) que iría variando entre los valores dados.

Estos valores varían en un intervalo determinado por secuencia. Secuencia está definido de la siguiente manera:

valor.inicial:valor.final

Valor.inicial es el valor en el que comienza la variable, mientras que valor.final es el último valor que tomará esta variable en el ciclo.

```
for (i in 1:100) {
        A[i] = A[i] + 1
}
```

En este caso, vemos que se forma un bucle "for" que va desde el i = 1 hasta el i = 100. Dentro del bucle, observamos que hay un vector A con 100 componentes, que ocupan las posiciones 1-100. Al presentar [i], sabemos que cada componente del vector dependerá del valor de i.

Comenzamos el bucle, con i = 1. En este momento, A[1] (la primera componente del vector) suma +1 a su valor, y se almacena en A[1]. Tras hacer esta operación, i = 1 pasa a ser i = 2, y se repite el proceso.

De esta manera, cada componente del vector A ha aumentado su valor en +1.

Bucle omitiendo valores

Un bucle “for” normal, siempre va en intervalos de 1, pero esto puede cambiar. Esto es útil cuando no queremos ciertos valores de un vector. Por ejemplo, cuando solo queremos los que ocupan las posiciones pares o impares. Para ello, este es el código que usaríamos:

```
for (i in seq(inicio, fin, by = n)) {
        # Código a hacer
}
```

Aquí, en vez de colocar 1:100, lo ordenaremos de forma diferente. Utilizaremos “seq()”, introduciendo en orden nuestro primer valor de la variable, el final de la misma y por último, by = n. N es el valor que queramos que vaya saltando el intervalo. Por ejemplo, si tuviéramos un vector con los números del 1 al 100 en orden y solo queremos los impares, haríamos lo siguiente:

```
Numeros = seq(1, 100, 1)
NumerosImpares = 0
k = 1
a = length(Numeros)
for (i in seq(1, a, by = 2)) {
        NumerosImpares[k] = Numeros[i]
        k = k + 1
}
```

Teniendo definido un vector Números con los números del 1 al 100 en orden, iniciamos este bucle “for”. Empieza en i = 1, colocando el valor de Números[1] en NumerosImpares[1]. Tras esto, k aumenta en uno, siendo k = 2, y se repite el ciclo. En vez de pasar de i = 1 a i = 2, vamos a i = 3 (se suma +2 al valor anterior de i), y repetimos. El valor de Números[3] se guarda en NumerosImpares[2] y k pasa a ser 3. Esto se repite hasta que i alcanza el valor de 100, dando fin al bucle.

La presencia de un k = 1 fuera del bucle es necesaria, ya que si ponemos NumerosImpares[i], los valores en posición impar de Números[i] se guardarán en la misma posición en NumerosImpares[i]. Esto dejaría huecos en blanco en las posiciones pares del vector, dando un vector incompleto. Para

solucionar esto, creamos una variable k = 1, y dentro del bucle ponemos el comando k = k + 1, lo cual sumaría +1 al valor anterior de k.

Este k = 1 tiene que estar definido fuera del bucle, ya que, si lo colocamos dentro del bucle, cada vez que iniciemos con un nuevo i, k volverá a ser 1 y no avanzaremos en el vector. Asimismo, hay que definir el vector C como 0 antes de comenzar con el bucle.

Bucles del revés

En algunas ocasiones, quizás lo que queramos es generar un bucle "for" empezando en el último valor y acabar en el primero. Para ello, el código que debemos usar es muy similar al anterior. Lo único que cambiaría sería esta parte del código:

```
for (i in seq(inicio, fin, by = n)) {
        # Código a hacer
}
```

En inicio colocaríamos nuestro valor final, y en fin colocaríamos nuestro valor inicial. Para que vaya retrocediendo, lo único que debemos cambiar es el n por un -1 (u otro valor negativo). Como se explicó antes, este es el valor que se le va añadiendo al valor inicial al final de cada ciclo. Si este valor es negativo, en cada ciclo se verá reducido el valor de i. Un ejemplo del código sería el siguiente:

```
Numeros = seq(1, 100, 1)
Numeros
n = length(Numeros)
k = 1
NumerosAlReves = 0
for (i in seq(n, 1, by = -1)) {
        NumerosAlReves[k] = Numeros[i]
        k = k + 1
}
NumerosAlReves
```

En este ejemplo, queremos colocar los valores del vector Números; siendo estos los números del 1 al 100, en otro vector, NumerosAlReves; comenzando en el 100 y acabando en el 1. Seguimos el mismo procedimiento del ejemplo anterior, solo que esta vez comenzamos en el valor n (definido arriba como la longitud del vector Números, es decir, 100) y acabamos en el valor 1. En el "by", colocamos un -1, ya que queremos que vaya descendiendo de uno en uno.

No hay que olvidar de añadir un k = k + 1 y definir el vector NúmerosAlReves como 0, así como ponerle el subíndice [k]. De esta manera, al final de cada ciclo, [i] se reduce en 1 y [k] aumenta en 1.

Bucles anidados

Estos bucles, pueden ser anidados. Esto significa colocar un bucle dentro de otro bucle, de tal manera:

```
# Bucle for externo
for (i in 1:5) {
        # Bucle for interno
        for (j in 1:5) {
                # Operación dependiente de j
        }
# Operación dependiente de i
}
```

Este ejemplo muestra un bucle dependiente de j dentro de un bucle que depende de i. Para cada valor de i, el bucle de j se completa, realizando para i = 1 las operaciones del bucle j desde j = 1 hasta j = 5. Una vez hechas, i = 1 pasa a ser i = 2, y j vuelve a pasar por los valores de j = 1 hasta j = 5. Se pueden anidar todos los bucles que quieras, ya sean 2 o 200.

Un ejemplo sería el siguiente:

```
A = c(seq(1,20,1))
B = c(seq(51,90,2))
C = 0
n = length(A)
for (i in 1:n){
        for (j in 1:n){
                B[i] = B[j]/i
                A[i] = A[j]+2*i

        }
        C[i] = B[i] + A[i]/2
}
```

En este ejemplo, los valores de i y j varían desde 1 hasta n. El valor de n está a definido por la longitud del vector A (n = length(A)). Para cada valor de i, suceden todas las operaciones dentro del bucle de j. Al salir del bucle de j, hace las operaciones del bucle de i, y el proceso vuelve a empezar para un valor de i diferente.

Hay que tener en cuenta, que los vectores que vayamos a usar han de estar definidos. En caso del C, que es en el cual estamos guardando los valores obtenidos, previamente tenemos que haberlo definido como C = 0.

Aplicaciones

Sumatorios

Hacer un sumatorio es muy sencillo en R, y para ellos usamos los bucles "for". Esto es útil para calcular la media de unos valores, calcular un total de números o incluso para la interpolación. Para hacerlos, tan solo tenemos que seguir este código:

```
S = 0

for (variable in secuencia) {

        S = S + Valores [variable]

}
```

Como explicado anteriormente, en variable colocamos cualquier letra que definirá las posiciones del vector, y en secuencia colocaremos el número en el que empiece y acabe de la manera inicial:final. Valores es el vector para el cual queremos hacer el sumatorio, y para ello requerimos de un lugar donde almacenarlo. Para ello, usamos un S = 0 (puede ser cualquier letra, A, c, T…).

El S = 0 es la clave del sumatorio. Como podemos observar en el código, en cada reiteración del ciclo, S se suma al cada valor del vector, y este número se vuelve a guardar el S. Hagamos un ciclo de prueba.

Al comenzar, S = 0. EN el primer ciclo, hacemos S (0) + Valores [1], y lo guardamos en S. En el segundo ciclo, S (Valores[1]) + Valores[2], y lo guardamos en S. En el tercero, S (Valores[1] + Valores[2]) + Valores[3], y lo guardamos en S. Como se puede observar, en cada ciclo, S guarda la suma de todos los valores anteriores, por lo que al acabar el bucle, S contendrá la suma de los valores del vector.

Productorio

El caso de un productorio es muy similar al de un sumatorio. De hecho, se trata de exactamente el mismo código, cambiando una pequeña cosa. En vez de presentar un S = 0, requerimos de un P = 1. El código sería de la siguiente forma:

```
P = 1

for (variable in secuencia) {

        P = P * Valores [variable]

}
```

Como explicado anteriormente, en variable colocamos cualquier letra que definirá las posiciones del vector, y en secuencia colocaremos el número en el que empiece y acabe de la manera inicial:final. Valores es el vector para el cual queremos hacer el sumatorio, y para ello requerimos de un lugar donde almacenarlo. Para ello, usamos un S = 0 (puede ser cualquier letra, A, c, T…).

El P = 0 es la clave del productorio. Como podemos observar en el código, en cada reiteración del ciclo, P se multiplica por cada valor del vector, y este número se vuelve a guardar el P. Hagamos un ciclo de prueba.

Al comenzar, P = 1. EN el primer ciclo, hacemos P (1) * Valores [1], y lo guardamos en P. En el segundo ciclo, P (Valores[1]) * Valores[2], y lo guardamos en P. En el tercero, P (Valores[1] * Valores[2]) * Valores[3], y lo guardamos en P. Como se puede observar, en cada ciclo, P guarda el producto de todos los valores anteriores, por lo que al acabar el bucle, P contendrá el producto de los valores del vector.