INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

UNIDAD 4

Lenguajes y algoritmos de programación

Actividad 2

Cómo crear y usar funciones en R

Alumna: Lisset Alejandra Loera Arredondo

No. de control: 18480196

Clase: 11am-12pm

Fecha de entrega: 19 de noviembre 2018

Guadalupe, Nuevo León, México.

Realizar una investigación de cómo crear y usar funciones en R. (Tiene que incluir bibliografía y esas dos secciones con ejemplos)

En una función tenemos 3 tipos de elementos:

1. **Argumentos** (o valores de entrada).
2. **Cuerpo**: operaciones que han de realizarse. Se deben localizar entre corchetes “{}”.
3. **Resultado** (o valores de salida): la última expresión que se ejecuta.

```
mifuncion <- function(argumento1, argumento2, ...) {
cuerpo
resultado
}
```

Las funciones también son objetos y por tanto les daremos un **nombre**, en este caso se llamará “mifunción”. Debes evitar utilizar nombres que ya estén en uso en R, por ejemplo “mean”.

Los **argumentos** se separan por una coma dentro de “función ()”. Puede ser cualquier tipo y cantidad de argumentos. Los argumentos son los ingredientes que necesitas para que se ejecute la función. Los argumentos pueden tener un valor predeterminado, por ejemplo, si escribimos argumento2=10:

```
mifuncion <- function(argumento1, argumento2=10, ...) {
cuerpo
resultado
}
```

El **cuerpo** de la función contiene las operaciones que deseamos que se ejecuten sobre cada uno de los argumentos detallados anteriormente. Vienen dados entre corchetes “{}” y se ejecutan cada vez que llamamos la función.

El **resultado** es el valor devuelto por la función que se genera en las operaciones que se han ejecutado en el cuerpo de la función. Puede ser cualquier tipo de datos.
La última línea del código será el valor que devolverá la función.

Ejemplo 1. Función suma

```
> suma<-function(x,y){
+     # suma de los elementos "x" e "y”
+     x+y
+ }
```

La última operación evaluada es el valor que ha de retornar la función (también llamada salida).
Por ejemplo, si evaluamos la función para los valores x=2 e y=3 obtenemos:

```
> suma(x=2,y=3)
[1] 5
```

También podemos omitir los nombres de los argumentos si mantenemos la correspondencia con el orden o posición de los argumentos en el que damos los valores:

```
> suma(2,3)
[1] 5
> 5
```

Ejemplo 2. Función potencia con paste y cat

Cuando queremos que el resultado de una función contenga texto podemos utilizar las **función paste().**

```
> potencia <- function(x, y) {
+     # función que calcula x elevado a y
+     result <- x^y
+     paste(x,"elevado a la potencia de", y, "es", result)
+ }
> potencia(2,3)
[1] "2 elevado a la potencia de 3 es 8”
```

También podemos utilizar la **función cat()**, que tiene mayor versatilidad.
Por ejemplo:

```
> x<-2

> cat(x)
2

> cat("María")
María

> cat("María tiene", x, "hijos", ".")
María tiene 2 hijos .

> cat("María tiene", x, "hijos", "\b.") #\b quita el último espacio
María tiene 2 hijos.

> cat("María tiene\n", x, "hijos", "\b.") #\n divide la expresión en dos líneas
María tiene
2 hijos.
```

Ejemplo 3. Función de valor absoluto con condicionales (if).

```
> absoluto <- function(x) {
+     # valor absoluto de x
+     if(x<0){ -x }
+     x
+ }

> absoluto(-3)
[1] 3

> absoluto(3)
[1] 3
```

Aquí le estamos diciendo que si el valor de “x” es negativo nos devuelva su opuesto, en caso contrario que nos devuelva el valor original de “x”.

Ejemplo 4. Función a trozos con condicionales (if).

Función a trozos: si x es menor a 5 toma el valor 0 y en caso contrario el valor 10.

```
> ftrozos <- function(x) {
+     if (x < 5) {
+         0
+     } else {
+         10
+     }
+ }

> ftrozos(3)
[1] 0

> ftrozos(5)
[1] 10
```

Ejemplo 5. Función return

También se puede utilizar la **función “return()”** para obtener el resultado de un paso en particular en la ejecución, no necesariamente el último. Es útil por ejemplo para identificar un error.

```
> f<-function(x,y){
+     if(is.character(y)) return("y debe ser numérico")
+     x+y
+     }

> f(2,"hola")
[1] "y debe ser numérico"
```

Si no utilizáramos la función return() obtendríamos un mensaje de error:

```
> f<-function(x,y){
+     if(is.character(y)) "y debe ser numérico"
+     x+y
+     }

> f(2,"hola")
Error in x + y : argumento no-numérico para operador binario
```

Cómo utilizar una función en R

Las funciones en R se pueden tratar como cualquier otro **objeto R.**

Se puede definir una función dentro de otra función, es decir, se puede utilizar como argumento para otras funciones o se pueden ejecutar desde otras funciones.

Lo ideal es que la función sea corta y simple, con nombres intuitivos.
El código debe ser fácil de escribir, fácil de leer y de utilizar, de rápida ejecución, con resultados confiables.
Para poder ejecutar una función primero debes guardarla en la memoria. Ocurre lo mismo que al activar/cargar una librería/biblioteca, hasta que no lo hagas las funciones contenidas en ella no se pueden llamar/utilizar.
Existen dos métodos para **cargar funciones en la memoria:**

1. Crear el texto de la función y pegarlo en la consola.
2. Utilizar la función “source()” para cargar funciones desde un archivo .R (puedes tenerlo en tu directorio o descargarlo desde la web).

La mejor opción es la segunda, por eso se recomienda crear tus funciones en un archivo .R con un nombre sencillo e intuitivo y guardarlo en su ordenador para llamarlo en el momento que desees.

Por ejemplo:

```
> source( “ mifuncion.R”) #desde tu directorio

> # Para acceder a una función que se encuentra en la web
> source("https://raw.github.com/tonybreyal/Blog-Reference-Functions/master/R/bingSearchXScraper/bingSearchXScraper.
```

Bibliografía

- https://www.maximaformacion.es/blog-dat/crea-tu-propia-funcion-en-r-paso-a-paso/