

Clase funciones

Armando Ocampo

Funciones

Las funciones son elementos que nos permiten realizar acciones dentro del lenguaje de R. Cada función contiene argumentos, que le permite a la misma ejecutar cada acción. A su vez, un conjunto de funciones se almacena en librerías.

Para conocer los argumentos de cada función se utiliza la función `args()`. A continuación observaremos los argumentos de la función `mean()`.

```
args(mean)
```

Para obtener la documentación de cada función se utiliza la función `help()` o el signo de interrogación. En este apartado se encuentra una descripción detallada de cada función, la definición de cada argumento y algunos ejemplos.

```
help(mean)  
?mean
```

Condiciones

if statement

Antes de comenzar con las funciones, recordaremos algunas condiciones.

```
2 > 5  
10 > 2  
5 < 4  
8 < 20  
5 == 4  
5 == 2+3  
10 >= 10  
8 <= 8  
TRUE + TRUE  
1 == TRUE  
FALSE + FALSE  
0 == FALSE  
TRUE > FALSE
```

La función `if()` permite realizar una acción si la condición se cumple. Para esto chequearemos el esqueleto de `if()`.

```
if(condicion){  
  expr  
}
```

Dentro de los paréntesis se coloca la condición y dentro de las llaves la acción a realizar. Vamos a determinar si la variable d es negativa.

```
d <- -2  
if(d < 0){  
  print("d es un numero negativo")  
}  
  
# De esta forma podemos cambiar el valor de d y utilizar la funcion if  
d <- -5  
if(d < 0){  
  print("d es un numero negativo")  
}  
  
d <- -10  
if(d < 0){  
  print("d es un numero negativo")  
}  
  
# rompiendo la condición  
d <- 5  
if(d < 0){  
  print("d es un numero negativo")  
}
```

Cuando d es un numero positivo y no se cumple la condición, la acción no se efectúa. Para agregar dos salidas se agrega else{ }.

```
d <- 5  
if(d < 0){  
  print("d es un numero negativo")  
} else{  
  print("d es un numero positivo")  
}  
  
d <- -4  
if(d < 0){  
  print("d es un numero negativo")  
} else{  
  print("d es un numero positivo")  
}  
  
d <- 12  
if(d < 0){  
  print("d es un numero negativo")  
} else{  
  print("d es un numero positivo")  
}
```

```

}

# error en la salida
d <- 0
if(d < 0){
  print("d es un numero negativo")
} else{
  print("d es un numero positivo")
}

```

Este error se presenta debido a que 0 no es un numero positivo. Solo se sigue la idea del código. Obtenemos como resultado “d es un numero positivo” al no cumplir con la condición. No obstante Podemos agregar una segunda condición en la función. Esto mediante else if()

NOTA: else if() se escribe separado con espacio.

```

if(d < 0){
  print("d es un numero negativo")
} else if(d == 0){
  print("d es igual a cero")
} else{
  print("d es un numero positivo")
}

d <- -4
if(d < 0){
  print("d es un numero negativo")
} else if(d == 0){
  print("d es igual a cero")
} else{
  print("d es un numero positivo")
}

d <- 5
if(d < 0){
  print("d es un numero negativo")
} else if(d == 0){
  print("d es igual a cero")
} else{
  print("d es un numero positivo")
}

d <- 0
if(d < 0){
  print("d es un numero negativo")
} else if(d == 0){
  print("d es igual a cero")
} else{
  print("d es un numero positivo")
}

```

Mis primera funciones

Es probable que ya exista una función para el problema que tengas. No obstante, ¿cómo puedo realizar una función personalizada?. En este aspecto, puedes crear tu propia función, esto a partir de la función `function()`. Para esto chequearemos el esqueleto básico de una función.

```
nombre_funcion <- function(arg1, arg2, argn){  
  cuerpo de la funcion  
}
```

El nombre de la función como cualquier objeto de R debe seguir ciertas reglas. No debe ser un número aislado, no debe coincidir con el nombre de una función ya establecida, ni tener espacios.

El número y nombre de cada argumento es libre, siempre y cuando se respeten las reglas de escritura.

La primer función que generaremos se denominará mi primer función y tendrá el objetivo de tomar cualquier número y sumarle 3 unidades.

```
mi_primer_funcion <- function(x){  
  x + 3  
}  
  
mi_primer_funcion(x = 4)
```

En esta función, cada vez que le asignamos un valor al argumento `x`, a este se le sumarán 3 unidades. Por lo que podemos utilizarla con el número que queramos.

```
mi_primer_funcion(5)  
mi_primer_funcion(15)  
mi_primer_funcion("a")
```

Los argumentos de una función pueden estar definidos o no. En el ejemplo previo `x` no está definido. Sin embargo, podemos generar argumentos con un valor preestablecido en la función. No obstante, este argumento se puede modificar al utilizar la función. Haremos una segunda función con un argumento definido.

```
mi_segunda_funcion <- function(x, y=8){  
  x + y + 5  
}  
  
# en esta funcion tenemos dos argumentos, "x" y "y". Esta funcion se puede  
# utilizar solo asignandole un valor al argumento "x"  
  
mi_segunda_funcion(x = 5)  
mi_segunda_funcion(x = 20)  
  
# esto por que el argumento "y" ya tiene un valor definido. Sin embargo,  
# podemos modificar el valor del argumento "y", sin modificar el esqueleto  
# de la funcion  
  
mi_segunda_funcion(x = 5, y = 30)
```

En `mi_segunda_funcion()` tenemos un argumento definido, el argumento “y”. Y un argumento no definido, el argumento “x”

Dentro de las funciones podemos generar objetos para realizar una acción. Estos objetos no se guardarán en el ambiente, solo pertenecerán al esqueleto de la función. En el siguiente ejemplo generaremos una función con el argumento “x”. Dentro de la función se generará un objeto denominado resultado con la suma de $x + 4$. Por último agregaremos una línea de código con $20 +$ resultado.

```
mi_tercera_funcion <- function(x){  
  resultado <- x + 4  
  20 + resultado  
}  
  
mi_tercera_funcion(5)
```

Nota: la salida de una función siempre será el último elemento de la línea de código

Dentro de una función se pueden agregar tantos objetos como sea necesario. No existe un límite.

```
cuarta_funcion <- function(x){  
  objeto1 <- x * x  
  objeto2 <- x + 5  
  objeto3 <- 20 + 8  
  
  objeto1 + objeto2 + objeto3  
}  
  
cuarta_funcion(5)
```