Curso R

Verano Delfín 2024

June 28, 2024

Operadores aritmeticos

Funciones

Para definir funciones propias:

```
nombre de la función <- function(parámetros de entrada) {
    variable_auxiliar <- f(parámetro salida 1, parámetro salida 2, ...)
    return(variable_auxiliar)
}
```

A continuacion se llama a la función con los parámetros de entrada correspondientes:

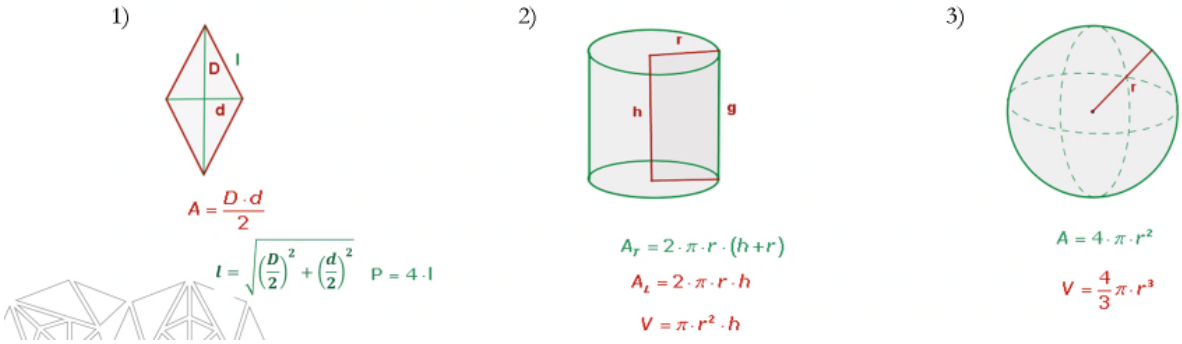
```
nombre de la función(parámetro de entrada 1, parámetro de entrada 2, ...)
```

Es posible definir funciones cuya unica finalidad sea la de graficar. Para ello, se debe incluir la función *plot()* dentro de la función que se defina.

```
nombre de la función <- function(parámetros de entrada) {
    plot(x, y)
}
```

Ejercicios:

Funciones en R



1. $y = 2x^2 - x - 4$
2. $y = \frac{3x+2}{4x}$
3. $y = \sqrt{x} + 2$

Estructuras de datos

Es posible definir secuencias de datos en R mediante el uso de vectores. Para ello, se utiliza la función *c()*.

```
vector <- c(dato1, dato2, dato3, ...)
```

seq() es una función que permite generar secuencias de números de forma automática. La estructura es la siguiente:

```
vector <- seq(inicio, fin, paso)
```

Para obtener un elemento en una posicion específica de un vector, se utiliza la siguiente estructura:

```
vector[posición]
```

Tambien es posible obtener un segmento de un vector. Para ello, se utiliza la siguiente estructura:

```
vector[inicio:fin]
```

Por ultimo, es posible obtener la longitud de un vector mediante la función *length()*.

```
longitud <- length(vector)
```

Cabe señalar que los vectores siempre tienen un mismo tipo de dato.

Si se añade un dato de un tipo diferente al del vector, R convertirá todos los datos a un tipo de dato común. En caso de que no sea posible, se mostrará un mensaje de advertencia.

Podemos crear vectores mediante la funcion *rep()*.

rep(dato, veces)

Por su parte la función *sample()* permite obtener una muestra aleatoria de un vector.

sample(vector, tamaño_muestra)

Para ordenar los elementos de un vector de forma ascendente, se utiliza la función *sort()*.

sort(vector)

Si los queremos ordenar de forma descendente, se añade el argumento *decreasing = TRUE*.

sort(vector, decreasing = TRUE)

La funcion *str()* permite obtener información sobre la estructura de un objeto. *class()* devuelve el tipo de objeto, y *storage.mode()* devuelve el modo de almacenamiento.

Para crear vectores con etiquetas se puede realizar manualmente construyendo el vector y asignando las etiquetas con la función *names()*.

```
vector <- c(dato1, dato2, dato3, ...)
names(vector) <- c("etiqueta1", "etiqueta2", "etiqueta3", ...)
```

Para ello, el vector de etiquetas requiere tener la misma longitud que el vector de datos.

Para instalar paquetes en R, se utiliza la función *install.packages()*.

Archivos

R permite manejar diversos tipos de archivos tales como *.txt, .csv, .xls, .xlsx, .rds, .rda, .rdata, .sav, .dta*, entre otros.

Asi pues, existe una función específica para cada tipo de archivo. Por ejemplo, para leer un archivo *.txt* se utiliza la función *read.table()*, para leer un archivo *.csv* se utiliza la función *read.csv()*, y para leer un archivo *.xls* o *.xlsx* se utiliza la función *readxl::read_excel()*. Sin embargo, en general, la estructura de estas funciones es similar.

objeto <- f("ruta_archivo", header = TRUE, sep = "separador")

donde *f* es la función correspondiente al tipo de archivo, *header* indica si el archivo tiene encabezados, y *sep* es el separador de columnas. Por convención el objeto se nombra igual que el archivo y dependiendo de la función, es posible especificar el rango de lectura, el número de filas a omitir, entre otros.

Una vez creado el dataframe, es posible manejarlo de la misma forma que un vector.

Para obtener un resumen de un dataframe, se utiliza la función *summary()*.

summary(objeto)

Para obtener información sobre la estructura de un dataframe, se utiliza la función *str()*.

str(objeto)

Para visualizar los atributos de un dataframe, se utiliza la función *attributes()*.

attributes(objeto)