TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE NUEVO LEÓN

Algoritmos y Lenguaje de Programacion

Funciones en R studio

Nombre del alumno: *Bianca Guadalupe Hernández Hernández*

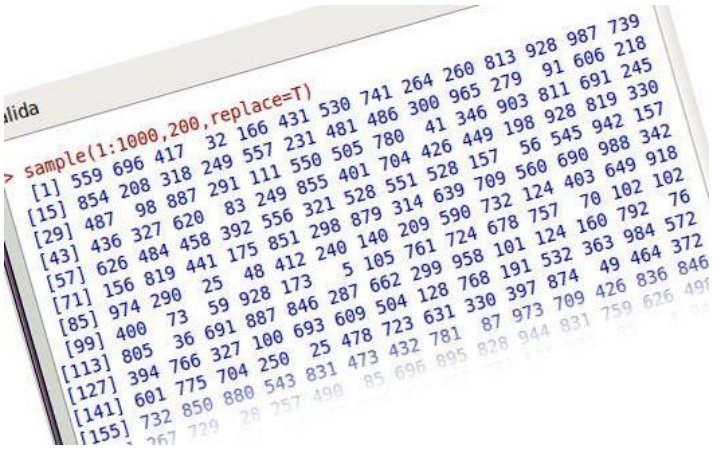
Nombre del profesor: *Juan Pablo Rosas Baldazo*

Matricula: *18480415*

Carrera: *Ingeniería Industrial*

Hora clase: *11:00 a 12:00*

Guadalupe Nuevo León, a 17 de Noviembre del 2018



CREAR FUNCIONES EN R STUDIO

Una función es un grupo de instrucciones que toma un "input" o datos de entrada, usa estos datos para computar otros valores y retorna un resultado/producto.

Una variable/objeto que se crea dentro de una función, es llamada una variable local, puesto que es temporal y solo se utiliza
dentro de la función, mientras se efectúan los cálculos u operaciones, y una vez obtenido el resultado esta variable es eliminada.

R permite crear estructuras repetitivas (loops) y la ejecución condicional de sentencias.
A este fin, los comandos pueden agruparse entre llaves, utilizando la siguiente sintaxis:

{comando1 ; comando2; comando3 ; ....}

El bucle for

Para crear un bucle repetitivo (un bucle for), la sintaxis es la siguiente:

for (i in listadevalores) { secuencia de comandos }

El bucle while

La sintaxis es como sigue:

while (condicion logica) { expresiones a ejecutar }

Condicional: if

La sintaxis general es:

if (condicion) comando1 else comando2

La estructura general de una función en R

nombre = function(argumento1 , argumento2, ...) comandos

Por ejemplo, podemos definir una función que calcule la desviación típica:

```
> desv = function(x){sqrt(var(x))}    # Definimos la función
> x<-1:10                             # Generamos datos
> desv(x)                             # Utilizamos la función
[1] 3.027650
> sd(x)          # La definida en R coincide con la nuestra
[1] 3.027650
```

Una vez definida una función, se la puede llamar y utilizar como a cualquiera otra función predefinida en el sistema. Por ejemplo, vamos a utilizar la función apply combinada con desv para calcular las desviaciones típicas de las columnas de una matriz:

Argumento '...' en una función

El argumento '…' (Sin las comillas) permite pasar un número variable de argumentos a una función.

Ejemplo:

```
inverso <- function(...) {
  v <- unlist(list(...))  # Asigna los argumentos a un vector
  x <- 1/v
  return(x)
}
inverso(8) # Se comporta como la función anterior
## [1] 0.125
inverso(7,5,10) # Pero podemos poner un número variable de argumentos
## [1] 0.1428571 0.2000000 0.1000000
inverso(47,11,587,12,-87)
## [1]  0.021276596  0.090909091  0.001703578  0.083333333 -0.011494253
```

Funciones

c(a_1, a_2,...)

Esta función permite concatenar (unir) objetos: variables, texto, números, etc.

Ejemplo:

```
x <- c(1,2,3)
t <- c("uno","dos","tres")
x; t
## [1] 1 2 3
## [1] "uno"  "dos"  "tres"
```

names(x) <- valor

Permite asignar nombres a los elementos de una variable

Ejemplo:

```
x <- c(2,4,6)  # Asigna 3 valores a la variable x
names(x) <- c("I","II","III")  # Asigna nombres a los 3 valores anteriores
x
##   I  II III
##   2   4   6
```

sqrt(x)

Calcula la raíz cuadrada de un número

Ejemplo:

```
sqrt(9)
## [1] 3
sqrt(2)
## [1] 1.414214
sqrt(-1)
## Warning in sqrt(-1): Se han producido NaNs
```

cat(“texto”,x_1,“texto”,x_2,…,“\n”)

Esta función escribe texto y variables en la salida.

La secuencia de escape “\n” produce una nueva línea e impide que la siguiente salida del programa quede en la misma línea

```
x <- 2
y <- 4
cat(x,"elevado a",y,"es",x ^ y,"\n")
## 2 elevado a 4 es 16
```

invisible(x)

Oculta x en la función return de forma que no se ve el resultado. x todavía se podrá asignar aunque no aparezca

Ejemplo:

```
suma <- function(a, b) {
  s <- a + b
  return(invisible(s))
}

  suma(5,4)
```

No se obtiene ningún resultado visible, pero podemos comprobar que existe asignando la función a una variable y mostrando su resultado.

```
x <- suma(5,4)
x
## [1] 9
```

length(x)

Muestra el número de elementos de un vector

```
x <- 1:20  # Se guarda en x los números del 1 al 20
length(x)  # Número de elementos de x
```

trunc(x)

Elimina los decimales de un número

```
trunc(1.999999)
## [1] 1
x <- 56.13
trunc(x)
## [1] 56
```

round(x,decimales=0)

Redondea un número con los decimales indicados, si no se indican decimales se redondea sin decimales. Cuando el decimal que sigue al último que se mostrará es 5 o mayor de 5 entonces se aproxima el último decimal.

```
round(6.78)  # Al no indicar decimales el resultado es un número entero aproximado
## [1] 7
round(6.78,1) # Se redondea con un decimal
## [1] 6.8
round(10.627,1) # Como el segundo decimar es menor de 5 no se aproxima
## [1] 10.6
```

runif(n,inicio=0,fin=1)

Genera n números al azar entre inicio y fin. Si no se indica inicio y fin se generan entre 0 y 1

```
runif(5)  # 5 números al azar entre 0 y 1
## [1] 0.36124751 0.27944097 0.07655032 0.51075857 0.66329734
runif(5,1,10)  # 5 números al azar entre 1 y 10
## [1] 7.588251 8.722883 9.094666 3.539671 6.977387
trunc(runif(20,1,10))  # 20 números enteros al azar entre 1 y 10
##  [1] 6 9 8 7 9 7 4 4 2 6 8 9 1 9 6 7 3 1 1 4
```

sum(x)

Suma todos los elementos de un vector x

```
x <- runif(100,1,10)  # Se generan 100 números al azar entre 1 y 10 y se guardan en la variable x
sum(x) # Suma de los 100 números al azar
## [1] 586.0735
```

readline("Texto a mostrar (optativo)")

Lee una línea de texto del teclado. El programa queda detenido hasta que el usuario escribe algo y pulsa ENTER. Si introducimos números deberán ser convertidos a números.

Ejemplo 1:

```
encuesta <- function() {
  r <- readline("¿Te gusta R? (s/n) : ")
  if ( r == "s" || r == "S") {
    cat("¡Estaba seguro de eso!\n")
    return(invisible(0))
  } else {
```

any(condición)

Devuelve T si algún elemento cumple la condición Ejemplo:

```
x <- runif(10, -10 ,100) # Se generan 10 números aleatorios entre -10 y 100
if(any(x < 0)) cat("En x hay números negativos\n") # Si algún números de x es negativo
## En x hay números negativos
```