**Lenguajes de Programación**

Lua y R

Los programas en Lua no son interpretados directamente, sino [compilados](https://es.wikipedia.org/wiki/Compilador) a código [bytecode](https://es.wikipedia.org/wiki/Bytecode" \o "Bytecode), que es ejecutado en la [máquina virtual](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual) de Lua. El proceso de compilación es normalmente transparente al usuario y se realiza en tiempo de ejecución, pero puede hacerse con anticipación para aumentar el rendimiento y reducir el uso de la [memoria](https://es.wikipedia.org/wiki/Memoria_(inform%C3%A1tica)) al prescindir del compilador.

También es posible la [compilación en tiempo de ejecución](https://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n) utilizando [LuaJIT](https://es.wikipedia.org/w/index.php?title=LuaJIT&action=edit&redlink=1" \o "LuaJIT (aún no redactado)).

R es un entorno de software libre y lenguaje de programación interpretado, es decir, ejecuta las instrucciones directamente, sin una previa compilación del programa a instrucciones en lenguaje máquina

2.-

El lenguaje de programación Lua se usa para la programación procedimental general. Por lo tanto, dentro de los paradigmas de programación, es considerado perteneciente a los lenguajes de scripting imperativos.

R es un lenguaje de programación diseñado para realizar análisis estadísticos y crear gráficos a partir de datos, proporcionando un amplio abanico de herramientas estadísticas (modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, etc..) y gráficas.

3.-

IDEs para Lua, en Windows LuaEdit , LuaIDE . También, Zeus se puede programar en Lua, lo que puede ayudar a reducir el cambio de contexto mental

IDE para R , Rstudio , es ligero y permite visualizar fácilmente las variables y objetos que creas. También es muy sencillo visualizar las gráficas y exportarlas en pdf, imagen, etc.

Frameworks Lua: Orbita, TurboLua, CMS

Frameworks R: Plumber, Ambiorix, Shiny.

4.-

* **"add":** La operación suma, en Lua.

La función getbinhandler que aparece más abajo define cómo escoge Lua un manejador de la operación binaria. Primero Lua prueba el primer operando. Si su tipo no define un manejador para la operación entonces Lua lo intenta con el segundo operando.

function getbinhandler (op1, op2, evento)

return metatable(op1)[evento] or metatable(op2)[evento]

end

Usando esta función el comportamiento del código op1 + op2 es

function add\_event (op1, op2)

local o1, o2 = tonumber(op1), tonumber(op2)

if o1 and o2 then -- ¿son numéricos ambos operandos?

return o1 + o2 -- '+' aquí es la primitiva 'add'

else -- al menos uno de los operandos es no numérico

local h = getbinhandler(op1, op2, "\_\_add")

if h then

-- invoca el manejador de ambos operandos

return (h(op1, op2))

else -- no existe un manejador disponible: comportamiento por defecto

error(···)

end

end

end

Aritmetica en R

Con R puedes realizar operaciones aritméticas básicas como las siguientes: \* Suma: + \* Substracción: - - Multiplicación: \* - División: /

Por ejemplo, en la siguiente línea, suma 7 + 3

HIDE

7 + 3

Una ventaja de usar un lenguaje de programación es que se pueden asignar variables y usarlas en las operaciones aritméticas, veamos algunos ejemplos:

Creación de variables

Una manera muy cómoda de manejar un “conjunto de datos” es generar variables que contengan nuestros datos. Para lo cual usamos un comando de **asignación** <- de valores o caracteres, a variables, como en el ejemplo siguiente:

HIDE

mi.variable <- 42

manzanas <- 5

naranjas <- 7

mi.fruta <- manzanas + naranjas

R trabaja con varios tipos de datos:

* Valores decimales como 4.5 son llamados *numéricos*.
* Los números naturales como el 4 son llamados *enteros*, los cuales también son numéricos.
* Los valores booleanos (TRUE or FALSE) son llamados *lógicos*.
* Text (o string) son llamados *caracteres*.

Para conocer qué tipo o clase de datos son los valores de tus datos puedes usar el comando “class(datos)”

HIDE

letras <- "conjunto de caracteres"

class(letras)