CREACIÓN DE FUNCIONES EN R STUDIO.

INGENIERÍA INDUSTRIAL

2do. Semestre.

Joel Fernando López Pérez.

FUNCIONES Y SUBRUTINAS.

- R es un lenguaje interactivo que nos permite crear objetos y analizarlos. Pero claramente R va mucho más allá.
- R es un lenguaje en constante evolución: permite ir creando nuevas estructuras que resuelven nuevos problemas que van apareciendo.
- R es un lenguaje de expresiones: todos los comandos ejecutados son funciones o expresiones que producen un resultado. Incluso las asignaciones son expresiones cuyo resultado es el valor asignado.

Podemos crear explícitamente un objeto tipo expresión, con la función expression() y evaluarla con la función eval():

Ejemplo

```
exp1<-expression(3+4)
exp2<-expression(sum(1:10))
exp3<-expression(b<-1:10)
eval(exp1); eval(exp2); eval(exp3)
```

SUBRUTINAS.

En general, una subrutina es un segmento de código que se escribe sólo una vez pero puede invocarse o ejecutarse muchas veces.

Existen dos tipos:

- Procedimiento: un grupo de expresiones entre llaves que no produce ningún resultado.
- Función: cuando dicho conjunto de código si produce un resultado.

Cuando agrupamos comandos o expresiones entre llaves {expre.1; expre.2; ...; expre.m}, las expresiones pueden ir:

- separadas por ; en la misma línea.
- separadas por cambio de línea.

En ambos casos el valor resultante de la subrutina es el resultado de la última expresión evaluada. Puesto que una subrutina es también una expresión, podemos incluirla entre paréntesis y usarla como parte de otra expresión.

FUNCIONES Y EXPRESIONES.

1. La gran utilidad de las expresiones es que nos permiten ejecutar varios comandos de una única vez.

2. Pero donde gana mayor utilidad esta forma de trabajar es a la hora de crear nuevos objetos que ejecuten diversas expresiones utilizando como entrada unos objetos (argumentos) y devolviendo otros objetos.

3. Estos objetos (cuyo modo es function) constituyen las nuevas funciones de R, que se pueden utilizar a su vez en expresiones posteriores.

EJECUTANDO…

```
if (expre1) expre2 else expre3
```

Si expre1=TRUE calcula expre2

Si expre1=FALSE calcula expre3

Ejemplo

```
if (10>3) cat("SI 10>3 \n") else cat("NO 10>3 \n")
```

•En la primera expresión podemos incluir varios requerimientos utilizando los operadores lógicos.

Ejemplo

```
x<-0
if (is.numeric(x)&min(x)>0) rax<-sqrt(x) else stop("x debe
ser numérico y positivo \n")
```

BIBLIOGRAFÍA.

- https://www.uv.es/conesa/CursoR/material/handout-sesion5.pdf
- https://rstudio-pubs-static.s3.amazonaws.com/64478_60ed30aabd53499691f96f46ed9720db.html