[000Z] DEFAD: Introducción a R y Rstudio

Funciones

000R Team

2015–16

Funciones

Funciones Predefinidas

Funciones Predefinidas

Hay miles y miles, agrupadas en paquetes, y cada día hay más
`http://cran.es.r-project.org/`

Funciones matemáticas

Función	lo que hace
`abs(x)`	Valor absoluto de x, abs(-4) devuelve 4
`sqrt(x)`	Raíz cuadrada de x, sqrt(25) devuelve 5
`ceiling(x)`	Entero más pequeño mayor que x
`floor(x)`	Entero más grande no mayor que x
`trunc(x)`	Truncamiento de x
`round(x, digits=n)`	Redondea x a un número específico de decimales
`log(x,base=n)`	Logaritmos

Funciones estadísticas

Función	lo que hace
`mean(x)`	Media
`median(x)`	Mediana
`sd(x)`	Desviación estándar
`var(x)`	Varianza
`sum(x)`	Suma
`range(x)`	Rango
`min(x)`	Mínimo
`max(x)`	Máximo
`scale(x,center=TRUE,scale=TRUE)`	Estandarizar

Miscelánea

print()

```r
print(" Hola mundo ")
```

```
## [1] " Hola mundo "
```

```r
"Hola mundo"
```

```
## [1] "Hola mundo"
```

```r
("Hola mundo")
```

```
## [1] "Hola mundo"
```

cat()

```r
# concatenar e imprimir
cat( "hola", "amigo" )
```

```
## hola amigo
```

```r
cat( "hola",
     "amigo",
     "\n",
     "¿cómo estas?",
     file = "fichero-cocatenado.txt" ) # escribimos en un fichero
```

paste()

```
# Concatenar y guardar
a <- "Juanete"
paste(" Hola mundo, mi nombre es", a)
```

```
## [1] " Hola mundo, mi nombre es Juanete"
```

```
b <- paste("variable", 1:10, sep="")
b
```

```
##  [1] "variable1"  "variable2"  "variable3"  "variable4"
##  [6] "variable6"  "variable7"  "variable8"  "variable9"
```

a objetos

podemos aplicar funciones a una gran cantidad de objetos: vectores, arrays, matrices, dataframes. . .

```
# aplicar funciones a objetos 'complejos'
y <- c( 1.23, 4.56, 7.89 )
round( y )
```

```
## [1] 1 5 8
```

a objetos (ii)

```
z <- matrix( rnorm( 12 ), 3 )
z
```

```
##             [,1]        [,2]       [,3]     [,4]
## [1,]  0.08688921  0.06481559 -0.6399737 1.217632
## [2,] -1.00709073 -1.53666175 -1.0060336 1.755659
## [3,]  0.24340717  0.50707407 -0.8534055 2.047420
```

```
log( z )
```

```
## Warning in log(z): Se han producido NaNs
```

```
##           [,1]       [,2] [,3]      [,4]
## [1,] -2.443121 -2.7362092  NaN 0.1969079
## [2,]       NaN        NaN  NaN 0.5628445
## [3,] -1.413020 -0.6790982  NaN 0.7165806
```

```
mean( z )  # devuelve un escalar
```

```
## [1] 0.07331102
```

```
sd ( z )
```

```
## [1] 1.149943
```

Sentencias IF-ELSE

Ejecutan sentencias si se da una condición `if`, si no se da ejecutan la sentencia `else`. Se pueden anidar

```
if (condición) sentencia
ó
if (condición) sentencia1 else sentencia2
```

Sentencias IF-ELSE (2)

```
# IF-ELSE
if ( TRUE ) {
    print( "esta siempre se ejecuta" )
}

if ( FALSE ) {
    print( "esta nunca se ejecuta" )
} else {
    print( "¿lo ves?" )
}
```

funciones ad hoc

Funciones escritas por el usuario

- escribir funciones *ad hoc*
- se pueden empaquetar en grupos
- .... tenerlas siempre disponibles para todas las sesiones

La sintaxis es sencilla:

```
mi_funcion <- function( arg1, arg2, ... ) {
    sentencias
    return( objeto )
}
```

mi primera función

```
f.potencia <- function( num, exp = 2 ) {
    return( num ** exp )
}
f.potencia( 5, 2 )
```

```
## [1] 25
```

```
f.potencia( 5, 5 )
```

```
## [1] 3125
```

```
f.potencia( 5 )
```

```
## [1] 25
```

Ejercicios

ejercicio 1

Ejercicio: Crear una función que acepte dos argumentos, uno que sea un char con dos posibilidades "param" o "noparam", que por defecto sea el valor "param". El segundo argumento que sea un vector numérico. Y que nos devuelva para "param", la media, la desviación típica y la varianza del vector. Y para "noparam", la mediana, máximo y mínimo del vector.