PROGRAMA DE ATRACCIÓN DE TALENTO TECNOLÓGICO PARA EL SECTOR TURÍSTICO ANDALUZ

2020

Introducción a lenguaje R.

FORMACIÓN TURISMO Y TECNOLOGÍA

UTE conformada por:

Introducción a lenguaje R.

Bases del lenguaje

- Lenguaje basado en "S" que contiene una biblioteca completa para la implementación de funciones estadísticas
- R es un conjunto integrado de programas para manipulación de datos, cálculo y gráficos.
- Cuenta con:
    - almacenamiento y manipulación efectiva de datos
    - operadores para cálculo sobre variables indexadas (Arrays o Vectors)
    - Herramientas para análisis de datos
    - posibilidades gráficas para análisis de datos, que funcionan directamente sobre un sistema basado en ventanas (X-Windows)
    - Un lenguaje de programación simple con las características típicas de un lenguaje de alto nivel.

Bases del lenguaje

- R implementa muchas técnicas estadísticas que se encuentran debidamente documentadas en el sitio web oficial: https://www.r-project.org/.

- La diferencia entre R y otros sistemas estadísticos es que R permite crear programas y almacenar resultados en objetos intermedios almacenados en variables, mientras que en otros sistemas similares como SAS o SPSS no se cuenta con un entorno de desarrollo y las salidas y correspondientes representaciones de datos se enseñan de forma directa.

Instalación de R.

- Es recomendable utilizar R en un sistema con soporte X-Windows, como por ejemplo una distribución moderna de Linux con GNOME, KDE o derivados. También se puede instalar en un sistema Windows, no obstante se recomienda su instalación sobre un sistema basado en Unix.

- El software y documentación completa se encuentran disponibles en el sitio web oficial: https://cran.r-project.org/ Se recomienda seleccionar una distribución del binario precompilado.

Interacción básica con R.

```
#R
> help(solve)
> ?solve
> help("[[")
> help("**")
> help("*")
> help("/")
> help(solve)
> ?solve
> help("[[")
> help("**")
> help("*")
> help("/")
```

Interacción básica con R.

>x <- rnorm(50)

>y <- rnorm(x) #Genera dos vectores que contienen cada uno 50 valores pseudo aleatorios obtenidos de una distribución normal y los almacena en x e y

>ls() #Objetos cargados en la sesión actual.

>rm(x, y) #Elimna x e y.

>x <- 1:20 #Crea un vector secuencial de 1 a 20.

>w <- 1 + sqrt(x)/2 #A partir del vector x, crea un vector ponderado de desviaciones típicas y lo almacena en w.

>hoja.de.datos <- data.frame(x=x, y= x + rnorm(x)*w) #Crea una hoja de datos de dos columnas, llamadas x e y

>hoja.de.datos #Se enseña por pantalla.

>summary(hoja.de.datos)

Interacción básica con R.

>regr <- lm(y~x, data=hoja.de.datos) #Regresión lineal estándar.
>attach(hoja.de.datos) #Agrega la variable en el path de búsqueda.
>plot(x, y) #Plano de coordenadas
>lines(x, regr.loc$y) #Regresión lineal local
>abline(0, 1) # Regresión lineal real: Punto de corte 0 pendiente 1
> q()
Save workspace image? [y/n/c]: n

Interacción básica con R.

R utiliza diferentes estructuras de datos, la más simple es el vector.

```
> x <- c(10, 5.6, 3, 4, 7)
> x
[1] 10.0  5.6  3.0  4.0  7.0
> assign("y", c(1, 5, 3, 4, 7))
> y
[1] 1 5 3 4 7
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> z
> z
[1] 10.4  5.6  3.1  6.4 21.7
```

Interacción básica con R.

>source("fichero.R") #Ejecución desde un script

Con la instrucción "objects()" se pueden ver todos los objetos almacenados en la sesión de R. Para eliminar objetos se puede utilizar rm()

> objects()
[1] "dataset" "split"   "testing" "training"
> rm(dataset)
> objects()
[1] "split"   "testing" "training"
> rm(testing, training)

Interacción básica con R.

Los vectores pueden usarse en expresiones aritméticas, en cuyo caso las operaciones se realizan elemento a elemento
Las operaciones se ejecutan sobre cada elemento del vector:

>1/z

[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295

Los operadores aritméticos elementales son los habituales: +,-,*,/ y ^ para elevara una potencia. Además están disponibles las funciones **log, exp, sin, cos, tan, sqrt**

Interacción básica con R.

```
> min(z)
[1] 3.1
> max(z)
[1] 21.7
> d <- range(z)
> d
[1]  3.1 21.7
> sum(z) -> suma
> print(suma)
[1] 47.2
> prod(z) -> producto
> print(producto)
[1] 25073.95
```

Interacción básica con R.

Dos funciones estadísticas básicas: Media y varianza:

mean(x) media: es equivalente a sum(x)/length(x)

var(x) varianza: es equivalente a sum((x-mean(x))^2)/(length(x)-1)

> mean(z)

[1] 9.44

> var(z)

[1] 53.853

Interacción básica con R.

Otras funciones para realizar cálculos sencillos y operaciones:

```
> rango <- c(1:15)
> rango
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
> c(2*1:10) -> rango2; rango2
 [1]  2  4  6  8 10 12 14 16 18 20
> seq(length=10, from=1, by=.5) -> seq1; seq1
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
> s2 <- rep(x, times=5);
>s2 #Repite el objeto 5 veces
 [1] 10.0  5.6  3.0  4.0  7.0 10.0  5.6  3.0  4.0  7.0 10.0  5.6  3.0  4.0  7.0
[16] 10.0  5.6  3.0  4.0  7.0 10.0  5.6  3.0  4.0  7.0
```

Interacción básica con R.

Otras funciones para realizar cálculos sencillos y operaciones:

```
> pst = paste(c("X","Y"), c(1:10), sep=".")
> pst
 [1] "X.1"  "Y.2"  "X.3"  "Y.4"  "X.5"  "Y.6"  "X.7"  "Y.8"  "X.9"  "Y.10"

> absol <- abs(c(-1,2,3,4,-5))
> absol
[1] 1 2 3 4 5
```

Interacción básica con R.

Operaciones con Vectores

```
> x <- c(10, 5.6, 3.0, 4, 7)
[1] 10.0  5.6  3.0  4.0  7.0
> vectorlogico <- x > 4; vectorlogico
[1]  TRUE  TRUE FALSE FALSE  TRUE
> vectorlogico <- x >= 4; vectorlogico
[1]  TRUE  TRUE FALSE  TRUE  TRUE
```

Los vectores lógicos pueden utilizarse en expresiones aritméticas, en cuyo caso se transforman primero en vectores numéricos, de tal modo que FALSE se transforma en 0 y TRUE en 1.

Interacción básica con R.

Operaciones con Vectores

v <- numeric() ; v[5] <- 10 ; v; length(v) #El tamaño de un vector puede cambiar durante la asignación indexada.

[1] NA NA NA NA 10

[1] 5

> as.character(1:10) -> x; print(x) ; as.numeric(x) -> x; print(x) ; x[11] <- "Text" ; print(x) #Los casteos se realizan de forma implicita durante la asignación o explicita con "as.<funcion_de_casteo>"

Interacción básica con R.

Operaciones con Vectores

```
> x <- c(10, 5.6, 3.0, 4, 7)
[1] 10.0  5.6  3.0  4.0  7.0
> vectorlogico <- x > 4; vectorlogico
[1]  TRUE  TRUE FALSE FALSE  TRUE
> vectorlogico <- x >= 4; vectorlogico
[1]  TRUE  TRUE FALSE  TRUE  TRUE
```

Los vectores lógicos pueden utilizarse en expresiones aritméticas, en cuyo caso se transforman primero en vectores numéricos, de tal modo que FALSE se transforma en 0 y TRUE en 1.

Interacción básica con R.

Valores faltantes: La falta de modo y valor de un elemento se denomina "valor faltante" y se le asigna el valor especial "NA".

> 0/0

[1] NaN

> Inf - Inf

[1] NaN

> conNA <- c(1:3,NA); isna <- is.na(conNA); equalsna <- conNA == NA

> conNA; isna; equalsna

[1] 1 2 3 NA

[1] FALSE FALSE FALSE TRUE

[1] NA NA NA NA

Interacción básica con R.

```
>conNA[is.na(conNA)] <- 0; conNA
[1]  1  2  3 0
> conNA; logiConNA <- conNA[!is.na(conNA)]; logiConNA
[1]  1  2  3 NA
[1] 1 2 3
> conNA[(!is.na(conNA)) & conNA>0] -> logiConNAPositive; logiConNAPositive
[1] 1 2 3
```

Objetos básicos en R

Vectores : Colecciones de información que almacenan un único tipo, que en terminología del lenguaje es conocido como "modo".

Matrices : También conocidas como variables indexadas (Arrays). Son vectores indexados por dos o más índices y que se deben visualizar de modo especial.

factores : Sirven para representar datos categóricos.

Listas : Similares a los vectores pero no necesariamente almacenan un único tipo. Pueden contener vectores u otras listas.

Objetos básicos en R

Data Frames : Estructuras similares a una matriz, en que cada columna puede ser de un tipo distinto a las otras. Son objetos apropiadas para definir y manipular "matrices de datos" en donde existen columnas independientes y dependientes.

Funciones : Como en cualquier lenguaje de programación, representan instrucciones que se pueden ejecutar en un el contexto del programa y permiten la reutilización de código.

Factores nominales y ordinales.

- Un factor representa a un vector que define una clasificación discreta de los elementos de otro vector.

- En R existen factores nominales y ordinales. Los factores nominales se basan en una ordenación natural de los niveles de un factor y se crean con factor(). Los factores ordenados tienen un orden concreto partiendo de la función ordered()

- Se trata de una característica fundamental del lenguaje, ya que permite categorizar y crear agrupaciones de información, las cuales posteriormente se pueden utilizar para la ejecución de operaciones estadísticas

Factores nominales y ordinales.

Agrupación básica (sin uso de factores). Se unen los datos de dos vectores cuya longitud es idéntica

```
> pais <- c("España","Alemania","Italia","Portugal")
> salariomedio <- c(30000,60000,25000,23000)
> salariomedio[salariomedio < 30000]
[1] 25000 23000
> names(salariomedio) <- pais
> salariomedio[salariomedio < 30000]
Italia Portugal
25000    23000
> salariomedio[c("España", "Portugal")]
España Portugal
30000    23000
```

Factores nominales y ordinales.

Uso de factores nominales.

```
>ciudades <- c("Madrid", "Salamanca",  "Bilbao", "Barcelona",
"Sevilla", "Madrid", "Granada",  "Santander",  "Valencia",
"Galicia", "Valencia" , "Sevilla", "Bilbao", "Sevilla", "Salamanca",
"Granada", "Madrid",  "Madrid",  "Madrid",  "Valencia",
"Santander", "Salamanca", "Madrid", "Barcelona", "Bilbao",
"Granada", "Valencia", "Santander", "Valencia", "Galicia");
factorCiudad <- factor(ciudades)
```

Factores nominales y ordinales.

Uso de factores nominales.

> factorCiudad

```
 [1] Madrid    Salamanca Bilbao    Barcelona Sevilla   Madrid    Granada
 [8] Santander Valencia  Galicia    Valencia  Sevilla   Bilbao    Sevilla
[15] Salamanca Granada   Madrid    Madrid    Madrid    Madrid    Valencia  Santander
[22] Salamanca Madrid    Barcelona Bilbao    Granada   Valencia
Santander
[29] Valencia  Galicia
9 Levels: Barcelona Bilbao Galicia Granada Madrid Salamanca ... Valencia
```

> levels(factorCiudad) #Solamente funciona sobre factores.

```
[1] "Barcelona" "Bilbao"   "Galicia"  "Granada"  "Madrid"   "Salamanca"
[7] "Santander" "Sevilla"  "Valencia"
```

Factores nominales y ordinales.

Los "levels" de un factor representan la clasificación discreta en donde se incluyen los valores únicos del factor.

> ingresosPersonas <- sample(10000:45000, 30, replace=T) #30
Números aleatorios entre 10000 y 45000 sin valores repetidos (con replace=F existe la probabilidad de que hayan valores repetidos).
>length(factorCiudad); length(ingresosPersonas)
[1] 30
[1] 30

Factores nominales y ordinales.

La función tapply permite relacionar un vector con un factor que tienen la misma longitud. Dicha relación es lógica, por ejemplo los ingresos de una persona en una ciudad y además, permite la ejecución de una función que se ejecutará sobre cada uno de los “levels” del factor.

>MediaIngresosMerge = tapply(ingresosPersonas, factorCiudad, mean); MediaIngresosMerge # ¿Qué pasa con tapply si el factor y el vector tienen longitudes distintas?

```
Barcelona    Bilbao   Galicia   Granada    Madrid Salamanca Santander
Sevilla
 31806.00  19769.67  29931.00  36223.67  29102.83  22490.67  23698.00
25570.00
 Valencia
 28810.60
```

Factores nominales y ordinales.

**** PRACTICA ****
Seguir el mismo ejercicio anterior, pero especificar los levels "Madrid" y "Sevilla" solamente. Utilizar el parámetro “levels” de la función factor. Por ejemplo

factor(ciudades, levels=c("Madrid","Sevilla"));

Factores nominales y ordinales.

Los factores ordinales no solamente ordenan los valores de un vector, sino que además permiten indicar el peso o "importancia" que tendrán los niveles en función a dicho orden o utilizando el atributo "levels" de la función "ordered".

```
> conOrdenAlfabetico= ordered(ciudades); conOrdenAlfabetico
 [1] Madrid    Salamanca Bilbao    Barcelona Sevilla   Madrid    Granada
 [8] Santander Valencia  Galicia    Valencia  Sevilla   Bilbao    Sevilla
[15] Salamanca Granada   Madrid    Madrid    Madrid    Valencia  Santander
[22] Salamanca Madrid    Barcelona Bilbao    Granada   Valencia
Santander
[29] Valencia  Galicia
9 Levels: Barcelona < Bilbao < Galicia < Granada < Madrid < ... < Valencia
```

Factores nominales y ordinales.

```
> conOrdenConcreto = ordered(ciudades,
levels=c("Madrid","Sevilla","Salamanca")); conOrdenConcreto

 [1] Madrid    Salamanca <NA>      <NA>      Sevilla   Madrid    <NA>
 [8] <NA>      <NA>      <NA>      <NA>      Sevilla   <NA>      Sevilla
[15] Salamanca <NA>      Madrid    Madrid    Madrid    <NA>      <NA>
[22] Salamanca Madrid    <NA>      <NA>      <NA>      <NA>      <NA>
[29] <NA>      <NA>
Levels: Madrid < Sevilla < Salamanca
```

Variables Indexadas (matrices).

Una variable indexada (array o matriz) es una colección de datos indexados por varios índices. R permite crear y manipular variables indexadas por medio de algunas funciones disponibles en el lenguaje, “dim” es una de ellas.

> indexedVector40Len = c(1:40) ; dim(indexedVector40Len) <- c(10,2,2); indexedVector40Len

> indexedVector50Len = c(1:50) ; dim(indexedVector50Len) <- c(5,2,5); indexedVector50Len

Variables Indexadas (matrices).

Una variable indexada no sólo puede construirse modificando el atributo "dim" de un vector. La función array cumple el mismo objetivo y tiene la forma

```
Z <- array(vectordedatos,vectordedimensiones)
```

Por ejemplo, si el vector "h" contiene 24 o menos elementos, la instrucción:

```
Z <- array(h, dim=c(3,4,2))
```

usa "h" para almacenar en Z una variable indexada de dimensión 3×4×2.

Prueba: ¿Qué pasa si el vector tiene más de 20 elementos?

Variables Indexadas (matrices).

```
>z <- array(sample(1:20, replace=F), dim=c(3,4,2)); z
>z <- array(sample(1:30, replace=F), dim=c(5,3,3)); z
>z <- array(sample(1:20, replace=F), dim=c(5,2,2)); z
>z <- array(0, c(3,4,2)) #Variable indexada compuesta por ceros.
```

Concatenar filas o columnas de 2 matrices:

```
> x <- array(sample(1:20, replace=F), dim=c(3,4,2));
> y <- array(sample(1:20, replace=F), dim=c(3,4,2));
> xr <- rbind(1, x, y)
> xc <- cbind(1, x, y)
```

Convertir una variable indexada (array) a un vector:

```
> vec <- as.vector(x); vec
> vec <- c(x); vec
```

Objeto “List” en lenguaje R.

Una lista es una colección ordenada de elementos, los cuales son conocidos como componentes. A diferencia de los vectores, las listas pueden tener componentes con modos distintos, por ejemplo, puede contener un vector numérico, un valor lógico, una matriz y una función. Es similar al concepto de clase y objeto en lenguajes de programación como Java, Python o C#

> lst <- list(nombre="Juan", apellidos="Perez", edad="44", hijos=3, estadoscivil="S", edadhijos=c(4,8,3)); lst[1]; lst[[1]]; lst$nombre; lst$edadhijos; lst$edadhijos[1];

Objeto “List” en lenguaje R.

Concatenar cadenas:

> lst.ABC <- list(nombreA=c(11,22,33), nombreB="Cadena", nombreC=990.1); lst.ABC$nombreA; lst.ABC$nombreB; lst.ABC$nombreC

Uso de la función attach():
Permite que los componentes etiquetados de una lista u hoja de datos puedan ser accesibles de forma directa sin especificar la sintaxis lista$componente.

> attach(lst); nombre; edad; apellidos

Como se puede apreciar, ahora es posible acceder a "nombre" directamente en lugar hacerlo con lst$nombre

Trayectoria de búsqueda.

R cuenta con un conjunto de espacios de nombres (namespaces) para encontrar variables, funciones e incluso librerías en el entorno.

```
     > search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

La función attach() ubica un objeto en la trayectoria de búsqueda:

```
     > attach(lst); search();
 [1] ".GlobalEnv"        "lst"               "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

Trayectoria de búsqueda.

Gracias a la función “attach” es posible utilizar un componente de una lista sin tener que usar la notación **lista$nombrecomponente**. Hay que tener en cuenta que las variables se almacenan en ***.GlobalEnv***, por este motivo si se ejecutan las siguientes instrucciones, no se estará manipulando un componente de la lista, sino que se estará creando una variable nueva en el espacio de nombres ***.GlobalEnv***

```
> attach(lst); print(nombre); nombre <- "sofia"; print(lst$nombre)
[1] "Juan"
[1] "Juan"
>
```

La función detach() es la encargada de eliminar un componente de la trayectoria de búsqueda.

Hojas de datos (DATA FRAMES)

Las hojas de datos pueden interpretarse como matrices cuyas columnas están compuestas por diferentes modos y atributos. Pueden imprimirse en forma matricial y se pueden extraer sus filas o columnas mediante la indexación de matrices. Típicamente, las hojas de datos se crean partiendo de funciones como read.table o read.csv

> dataframe = read.table("Datasets/Data.csv", header=T)
> dataframe
> View(dataframe)

Control de flujo en R. Instrucción if

Existe una construcción condicional que sigue el siguiente orden:

if (expr1) expr2 else expr3

donde "expr1" debe producir un valor lógico, y si este es verdadero, (T), se ejecutará "expr2". Si es falso (F) y se ha escrito la opción "else" que es opcional, se ejecutara "expr3".

Se pueden utilizar los operadores && y || como condiciones de una orden "if". "&" y "|" se aplican a todos los elementos de un vector, “&&” y “||” se aplican a vectores de longitud uno y sólo evalúan el segundo argumento si es necesario, esto es, si el valor de la expresión completa no se deduce del primer argumento.

Control de flujo en R. Instrucción if

> x <- -1; if(x > 0) { message("x is positive") } else { message("x is negative") }

>x <- 0; if(x > 0) { message("x is positive") } else if(x < 0) { message("x is negative") } else { message("x is zero") }

ifelse(condición sobre vector, salida 1, sino salida 2)

```
>a <- c(1, 1, 0, 1)
>b <- c(2, 1, 0, 1)
# compara elementos de a y b
>ifelse(a == 1 & b == 1, "Si", "No")
```

Control de flujo en R. Instrucciones for,repeat y while

Existe una construcción repetitiva de la forma

> for (nombre in expr1) expr2

donde "nombre" es la variable de control de iteración, "expr1" es un vector y "expr2" es una expresión en la cual puede aparecer la variable de control "nombre".
"expr2" se evalúa repetidamente conforme "nombre" recorre los valores del vector "expr1".

> for (value in sample(1:20, replace = T)) print(value)

Control de flujo en R. Instrucciones for,repeat y while

- La función "break" se utiliza para terminar cualquier ciclo. Esta es la única forma (salvo que se produzca un error) de finalizar un ciclo repeat.

- La función "next" deja de ejecutar el resto de un ciclo y pasa a ejecutar el siguiente2. Las órdenes de control se utilizan habitualmente en la escritura de funciones.

Control de flujo en R. Ejemplos.

#5 ejecuciones del ciclo exactamente.

```
x <- 1
while (x <= 5)
{
print(x)
x <- x + 1 # esta es la manera de incrementar en R (no hay x++)
}
```

#Secuencia de números pares

```
sum = 0
repeat{
sum = sum + 2;
cat(sum); #concatena los resultados
if (sum > 11) break;
}
```

Control de flujo en R. Ejemplos.

```
i <- 1
repeat{
  message("Iteración número: ", i, "… Siguiente"); flush.console()
  if(runif(1) > 0.95) { # runif genera valor partiendo de una distribución
uniforme cuyo valor máximo es el especificado
    break
  }
  i <- i + 1
}
```

El código anterior se ejecutará como mínimo una vez, en otros lenguajes es lo que se conoce como una instrucción “do-while”.

Control de flujo en R. Ejemplos.

```
preg <- function() {
  R <-0; while (R != 3.1415) { # Inicio del loop (while)
    cat("Escriba el valor de PI hasta el 4to. decimal: ")
    R <- readLines(n = 1) # Lee la respuesta del usuario
    R <- as.numeric(R)    # Convierte la respuesta en numeric
    if (R == 3.1415) # Condicional 1
      break  # Corta y termina el loop
    if (R > 3.1415) {
      cat(" -- ese valor es muy ALTO ... intente de nuevo!\n")
    } else {
      cat(" -- ese valor es muy BAJO ... intente de nuevo!\n")
    }
  }
  cat("¡Es correcto!\n")
}
```

Creación de funciones en R.

R permite crear objetos del modo "function", que constituyen nuevas funciones de R que se pueden utilizar a su vez en expresiones posteriores. Aprender a escribir funciones útiles es una de las mejores formas de conseguir que el uso de R sea cómodo y productivo ya que se reutiliza código y los programas tienden a ser más compactos. Hay que recalcar que muchas de las funciones que se suministran con R, como "mean", "var", entre otras, no difieren de las funciones que pueda escribir el usuario. Para definir una función debe realizar una asignación de la siguiente manera

>NombreFuncion <- function(arg1,arg2, …) expresión

Donde "expresión" puede utilizar los argumentos "arg[i]" para calcular un valor que es devuelto por la función. La función puede invocarse en cualquier lugar que sea valido.

Creación de funciones en R.

se pueden dar argumentos por posición y después añadir argumentos por nombre.

```
fun1 <- function(dataframe, grafico, v) {
    [instrucciones]
}
```

las siguientes invocaciones a la función son equivalentes

```
> resultado <- fun1(d, g, v)
> resultado <- fun1(d, grafico=g, vector=v)
> resultado <- fun1(dataframe=d, grafico=g, vector=v)
> resultado <- fun1(d,  vector=v, grafico=g)
```

Las funciones retornan automáticamente el resultado de la última variable asignada, a menos que se utilice la función return()

Creación de funciones en R.

Una función puede pasar los valores de sus argumentos a otra función, esto se consigue con "..." como último parámetro en la definición de la función. Por ejemplo:

```
fun1 <- function(datos, hoja.datos, grafico=TRUE, limite=20, ...)  {
    <Instrucciones>
    otrafuncion(...) #Se le envían todos los argumentos recibidos en esta función.
}
```

Creación de funciones en R.

Parámetros por referencia o valor en R:
Es fundamental tener en cuenta que cualquier asignación ordinaria realizada dentro de una función es local y temporal y se pierde tras salir de la función. Si se quiere realizar asignaciones globales dentro de una función se debe usar el operador "<<-" o la función "assign".

```
> f = function(x,y) {
+ x <- x +1
+ y <- y +1
+ return(x+y)
+ }

> x <-20; y <- 30; z <- f(x,y); print(x): print(y)
```

Creación de funciones en R.

Una función puede pasar los valores de sus argumentos a otra función, esto se consigue con "..." como último parámetro en la definición de la función. Por ejemplo:

```
fun1 <- function(datos, hoja.datos, grafico=TRUE, limite=20, ...)  {
    <Instrucciones>
    otrafuncion(...) #Se le envían todos los argumentos recibidos en esta función.
}
```

Funciones gráficas.

Las funciones gráficas son un componente de R muy potente y versátil. Es posible crear gráficos estadísticos de todo tipo con pocas instrucciones. Los gráficos pueden usarse tanto en modo interactivo como no interactivo. Al iniciar R en este modo, se activa un dispositivo para mostrar gráficos y aunque este paso es automático, es conveniente que la función ejecutada es X11() para sistemas basados en Unix y windows() para aquellos basados en Microsoft Windows.

Funciones gráficas.

Las instrucciones gráficas se dividen en 3 categorías:

- **Alto nivel:** Son funciones que crean un nuevo gráfico, posiblemente con ejes, etiquetas,títulos, etc.

- **Bajo nivel:** Añaden información a un gráfico existente, tales como puntos adicionales, líneas y etiquetas.

- **Interactivas:** Son funciones que permiten interactuar con un gráfico, añadiendo o eliminando información utilizando el ratón o teclado.

Funciones gráficas.

Funciones gráficas de nivel alto

Las funciones gráficas de nivel alto están diseñadas para generar un gráfico completo partiendo de unos datos pasados a la función como argumento.

Si hace falta, es capaz de generar ejes, etiquetas o títulos (a menos que se indique lo contrario). Estas instrucciones comienzan siempre un nuevo gráfico, borrando el actual si hay uno.

plot, pairs, coplot, qqnorm, hist, doplot, contour.

Funciones gráficas.

Funciones gráficas de nivel bajo

A veces las funciones gráficas de nivel alto no producen exactamente el tipo de gráfico deseado. En este caso pueden incluirse funciones gráficas de nivel bajo para añadir información adicional (tal como puntos, líneas o texto) al gráfico actual que ha sido generado por una función gráfica de alto nivel.

points, lines, text, abline, polygon, legend, title, axis,

Funciones gráficas interactivas.

R dispone de funciones que permiten al usuario extraer o añadir información a un gráfico utilizando el ratón.

locator, identify