# FUNCIONES

Además de todas las operaciones y comandos básicos que hemos visto en los apuntes anteriores, en R se pueden manejar funciones. Trabajar con ellas es muy útil, pero a veces puede convertirse en una tarea bastante ardua.
Comenzaremos por lo básico

SINTAXIS DEL COMANDO
Nombre de la función= function(argumento1,argumento2,...){ expresión de la función }

Según está escrito, el argumento representa el número de variables, es decir, si la función depende solamente de la variable x, tendremos un solo argumento 1. No obstante, las funciones pueden depender de más de una variable.
Por ejemplo, en el caso de la función margarita dependa de cuatro variables distintas como pueden ser x, y, z, u, deberíamos escribir:
margarita=function(x,y,z,u){expresión de la función margarita}.

EJEMPLO 1
La función inversa tiene una sola variable y se definiría de la siguiente manera:

```
Finversa=function(x){
        1/x
}
```

EJEMPLO 2
La función suma de dos variables se define de la siguiente manera:

```
Fsuma=function(x,y){
        x+y
}
```

A TENER EN CUENTA
La forma de escribir las llaves a la hora de definir una función puede ser determinante en la manera de visualizar la función a primera vista. Las recomendaciones del profesorado han sido que se sigan los mismos procedimientos que al definir bucles for (véase apuntes bucles for, consejo acerca de la escritura).

RECOMENDACIÓN
Es aconsejable definir todas las funciones al comienzo del script y, una vez se haya hecho, incluir los datos, de manera que antes de llevar a cabo ninguna operación, se hayan introducido las funciones que se vayan a utilizar.

**OBTENCIÓN DE VALORES**

Una vez definida la función, se pueden obtener valores dentro de la misma solamente con escribir el nombre de la función y, dentro de un paréntesis, el valor determinado de las variables, que pueden ser una o más, separados por comas.

Ejemplo 1:
Finversa (2)
El resultado será ½=0.5

Ejemplo 2
Fsuma (1,0)
El resultado será 1

Imaginemos ahora que hemos tenemos un vector xx con varias componentes y una función f. Podemos evaluar la función en un componente determinado del vector o en un intervalo de componentes del mismo. Así, hablamos de que f(8) es función evaluada en un punto, en este caso el 8, f(xx[5]) es la función evaluada en la componente 5 del vector xx y f(xx[1:5]) función evaluada en las componentes 1 hasta 5 del vector xx.
¡Esta puntualización es importante entenderla porque se usará en la práctica 3 de laboratorio en R!

NOTA
Imaginemos que ya hemos definido todas las funciones que vamos a usar en un script al principio del todo y una de ellas es la siguiente:

```
f=function(x,y){
        x-y
}
```

Más tarde, escribimos los datos. Tenemos que dar dos valores, uno a la x y otro a la y, que se pueden poner directamente, o asignar un nombre a los valores por separado.
Si escribimos el valor directamente quedaría:

```
f(9,5)
```

Si queremos asignar un nombre, e aquí lo relevante de esta nota, no hace falta que se denominen igual que los nombres que hemos puesto en las funciones, en nuestro caso x e y. Para aclararlo mejor, sigamos con el ejemplo.
En la parte de los datos yo escribo

```
a=9
flor=5
```

Luego quiero introducir esos datos en la función f y escribo:

```
f(a,flor)
```

El resultado me dará 4 y no podrá ningún error aunque los valores se 9 y 5 se llamen a y flor respectivamente y no sean x e y. Esto se debe a que cuando escribes f(a,flor), el programa busca la función llamada f y le asigna los valores que escribamos en el paréntesis sin tener en cuenta el nombre.

# REPRESENTACIÓN GRÁFICA DE FUNCIONES

Una vez se ha definido una función y se le han dado un número de valores (puede ser útil hacerlo con comandos como runif, rnorm, véase apuntes instrucciones básicas), es interesante poder obtener la representación gráfica de los mismos y hacer el ejercicio más visual. Para ello utilizaremos la siguiente sintaxis:

plot(x,f(x))

donde x es la variable y f(x) la función evaluada en cada punto x.

Una vez conocida la sintaxis de cómo representar funciones, podemos ir un paso más allá. Interesa, por ejemplo, ser capaces de poner nombre a los ejes, darle color a la función, superponer dos funciones, escribir una leyenda, etc. Este tipo de comandos los veremos a continuación cada uno por separado.

¿Pero donde se escriben estos comandos? Sencillo, se escriben dentro del paréntesis del plot.

**TÍTULO DEL GRÁFICO**

Normalmente, se le suele añadir un título al gráfico para saber, a simple vista, qué es lo que representa. Para ello, añadiremos lo siguiente dentro del paréntesis del plot:

main='título que le queremos asignar al gráfico'

**ASIGNACIÓN DE NOMBRES A LOS EJES**

Para etiquetar el eje de abscisas, el eje de ordenadas o incluso ambos (recuerda: el eje de abscisas es el eje X, el horizontal y el eje de ordenadas es el eje Y, el vertical) se ha de escribir lo siguiente:

xlab='nombre que le quieras poner al eje X'
ylab='nombre que le quieras poner al eje Y'

No olvidar que el nombre de los ejes tiene que ir entre comillas.

**TAMAÑO DEL TEXTO**

Es posible variar el tamaño del texto o de los símbolos con respecto al preestablecido usando:

cex='número, que por ejemplo puede ser el 0.8'

**ESTILO DEL TEXTO**

El texto que aparece en el gráfico puede no solo escribirse de manera normal, sino que podemos incluir otro tipo de estilos como cursiva o negrita. Para ello se escribe: font='número del 1 al 4'.

Se puede cambiar el estilo de texto de los ejes (font.axis), del título de los ejes (font.lab) y del título (font.main)

| | |
|---|---|
| **1** | Normal |
| **2** | *Cursiva* |
| **3** | **Negrita** |
| **4** | ***Negrita cursiva*** |

**COLOR DE LA GRÁFICA**

Para que la gráfica sea más llamativa, podemos ponerle color. Se hace escribiendo:
col=’color en inglés’
Se adjunta dentro del paréntesis del comando plot. Además, para un mismo color puedes poner que sea oscuro escribiendo dark delante del nombre del color en inglés.

**IMPONER LÍMITES EN LA GRÁFICA**

En ocasiones conviene establecer ciertos límites a la gráfica de cierta función, sobretodo cuando tenemos dos o más funciones superpuestas y queremos visualizarlas de manera clara.
En este caso se añadiría:
xlim=c(valor1,valor2)
ylim=c(valor1’,valor2’).

xlim e ylim vienen determinados por vectores que indican el rango de valores en los ejes x,y. Un ejemplo puede ser:
plot(xx,f(xx[1:1001]),col=’dark blue’, xlab='Abscisa',ylab='Mi función',xlim=c(0,10), ylim=c(100,500))

**COMANDO PCH**

Utilizamos el comando pch para cambiar los símbolos de los puntos. Para poner un símbolo, es necesario dar un valor a este comando, que puede variar del 0 al 25.

pch=número del 0 al 25

Ejemplos:
pch=0 corresponde a un cuadrado vacío
pch=1 es una circunferencia
pch=2 es un triángulo
pch=8 es un asterisco

Este comando se añade dentro del paréntesis de plot.

**TIPOS DE REPRESENTACIÓN GRÁFICA**

Utilizando el argumento type podemos obtener distintos tipos de gráficas. Se incluye también dentro del paréntesis de plot.

type=letra

Ejemplos más utilizados en las sesiones de prácticas:

type=p (puntos, es decir, gráfico de dispersión)
type=l (líneas)
type=b (puntos y líneas, ambos)
type=h (histograma)

**COMANDO LAS**

El comando las controla la orientación de los caracteres en los ejes:

las=número del 0 al 3

0= paralelo a los ejes
1= horizontal
2= perpendicular a los ejes
3= vertical

**SUPERPOSICIÓN DE FUNCIONES (representación de dos o más funciones en el mismo gráfico)**

En el caso de que queramos superponer curvas, se utiliza la instrucción:
par(new="true")

NOTA: la instrucción par sirve para cambiar de manera permanente parámetros gráficos; es decir, las gráficas subsecuentes se dibujarán con respecto a los parámetros especificados por el usuario tras la instrucción par.
El argumento “true” indica que la nueva curva se agrega al gráfico anterior.

**ELIMINACIÓN DE EJES**

Al añadir una nueva gráfica, a menudo coinciden los ejes con la gráfica anterior. Por ello, al ser innecesarios, se suelen eliminar utilizando:

axes=FALSE

**LEYENDA**

Si queremos añadir una leyenda para aclarar la simbología de la gráfica, debemos usar la función legend().

Antes de hacer la leyenda debemos realizar dos pasos previos.
El primero es entender y realizar un concepto básico. Si tenemos el vector v con sus datos correspondientes, le debemos asignar un nombre a las componentes del vector:
names(v)=’nombre que queramos’

Después, hay que crear una matriz que tenga por filas o por columnas, los vectores que vayamos a representar en el gráfico. Si tengo dos vectores v y w, se haría de la siguiente forma:
A=rbind(v,w)
ó
A=cbind(v,w)

De manera que una vez hecho esto, ya podemos escribir el comando legend:
En primer lugar, debemos escoger donde queremos poner la leyenda, si arriba, abajo, a la derecha o a la izquierda, de manera que debemos poner una de las siguientes opciones: {top, bottom, left, right, top right, top left, bottom right, bottom left}
En segundo lugar, escribiremos, dependiendo si al crear la matriz hemos puesto rbind o cbind:
rownames(A)
ó
colnames(A)
siendo A una matriz que tiene por filas o por columnas los vectores a representar.
Finalmente, escribimos los colores de los que queremos rellenar el cuadradito que sale al lado de los nombres en la leyenda. Para ello escribimos:
fill=c(rellenar con los colores que hemos puesto en la representación gráfica y no olvidar las comillas)

Un ejemplo de leyenda completa sería
legend( "top", rownames (A), fill = c("green", "blue", "red"))

**GRÁFICOS DEL TIPO SECTORES CIRCULARES**

Los gráficos no tienen por qué ser todos del mismo tipo en el que se representan los valores según el eje X y el eje Y; podemos crear otro tipo de gráficos como son los gráficos de sectores circulares.
Para ello utilizamos la siguiente sintaxis:

pie(vector, clockwise=TRUE/FALSE,col=c('red', 'blue', 'green', etc))

donde clockwise true significa en sentido horario y clockwise false en sentido antihorario.

Ejemplo con el ejercicio de la práctica de laboratorio 4:

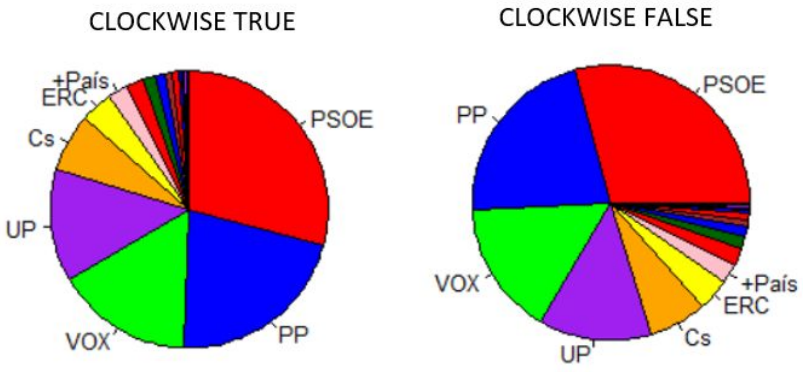


**GRÁFICO DE BARRAS**

Otro tipo de gráfico es el de barras, en el que los valores se representan mediante rectángulos.

barplot(x,f(x))

A tener en cuenta:
No confundir el gráfico de barras con el gráfico de tipo histograma. El de barras utiliza rectángulos y el histograma líneas. Además, para hacer un gráfico de tipo histograma se ha de escribir el comando plot y dentro del paréntesis type=h, mientras que para hacer un gráfico de barras se utiliza el comando barplot.

Si queremos superponer las barras con el fin de comparar los valores que toman dos funciones distintas en los mismos puntos o intervalos de puntos, hay que construir una matriz que tenga por filas los vectores. La sintaxis a seguir es:

```
nombre=rbind(vector1,vector2)
barplot(nombre, beside=TRUE, col=c(4,5))
```

# REPRESENTACIÓN DE VARIAS GRÁFICAS EN LA MISMA PANTALLA

Si queremos representar diferentes gráficos en la misma pantalla, habrá que distribuirlos de alguna manera.
Pongamos un ejemplo para aclarar cómo se hace. Tenemos dos gráficos y queremos colocarlos de manera que, horizontalmente, vaya uno detrás de otro. Para ello imaginemos una matriz A de una fila y dos columnas. En este caso, el gráfico uno ocupará la posición A(1,1) de la matriz, mientras que el segundo gráfico ocupará la posición A(1,2).
Ahora imaginemos que en vez de quererlas horizontalmente, las queremos verticalmente. Para ello deberíamos imaginarnos una matriz B de dos filas y una sola columna.

Ya que hemos entendido el concepto general, llevémoslo a la práctica. ¿Cómo se programa?
Se ha de utilizar la instrucción par seguida de:

- mfcol: un vector del tipo c(nr,nc), que divide la ventana gráfica como una matriz con nr filas y nc columnas; las gráficas se dibujan sucesivamente en las columnas
- mfrow: igual al anterior, pero las gráficas se dibujan siguiendo el orden de las filas.

Si añadimos también el comando mar (este indica un vector con 4 valores numéricos), podemos controlar el espacio entre los ejes y el borde de la gráfica de la siguiente forma:

```
mar=c(inferior, izquierda, superior, derecha)
```

Ejemplo aclarativo:

```
par(mfrow=c(2,1),mar=c(5,4,0.01,1))
```