PRÁCTICA 3 : Representación gráfica de funciones. Bucles

Este recurso es un resumen de la tercera práctica del curso. No estará solo resuelta, sino que también marcaremos algunas anotaciones y consejos con el fin de dejar claros algunos conceptos básicos. Además, se analizan los errores más comunes detectados en esta práctica en concreto, así como posibles dudas. Dicha práctica fue impartida por el profesor Arturo Hidalgo durante los días 4, 5 y 6 de octubre, y su objetivo es una introducción a la representación de funciones en R así como a programar bucles simples como anidados.

Antes de empezar con este recurso, recomendamos tener claros los conceptos anteriores sobre vectores, matrices y sus correspondientes operaciones, además de conceptos básicos sobre algoritmia. De esta forma, se hará más fácil la comprensión de estos ejercicios.

1. Comandos relevantes:

1.1. Representación gráfica de funciones

Para representar una función será esencial poner plot(a,b) aunque a esto podremos añadirle más "detalles" conforme queramos la gráfica. Ej. **plot(a,b,type= "b",col= "blue",xlab= "tiempo", ylab= "concentración",pch=2,main= "gráfico 1")**

- **plot(a,b)** : genera una gráfica siendo a y b vectores (deben estar definidos previamente y pueden tener cualquier nombre). La primera posición será para el eje x y la segunda para el eje y.

Es importante entender que se emplean vectores como ejes porque un vector en r es una forma de almacenar información, es decir, de almacenar los puntos que constituyen un eje.

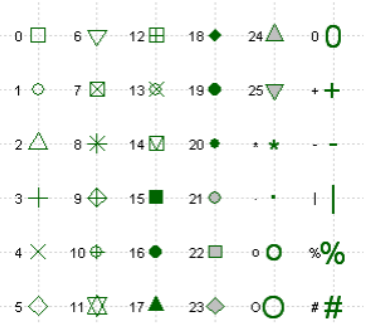
- **type= "?"**: genera una línea que une los puntos. Dependiendo de la letra cambiará el tipo de línea.
  - **p** solo se verán los puntos
  - **l** solo se verán las líneas que unen los puntos
  - **b** se verán tanto líneas como puntos
  - **c** se verán puntos vacíos unidos por líneas
  - **s** se verá en forma de escalera
  - **h** se verá en forma de histograma (líneas verticales)
  - **n** no se verá nada, ni puntos ni líneas

- **col= ""** : cambiará el color de la gráfica al que nosotros escribamos. Tiene que ir en inglés ('blue' 'green' 'purple'...)

- **xlab= ""** : nombre del eje x

- **ylab= ""** : nombre del eje y

- **pch=** : cambia el tipo de símbolo para los puntos según la siguiente imagen.



- **main= ""** : Poner título al gráfico

® **par(new= “true”)** : superpone las gráficas. Hay que ponerlo cada vez que queramos superponer una nueva. Se escribe encima de plot(,)

® **legend(x= “top” , c(“función 1”, “función 2”, “función 3”) , fill=c(“green”, “blue”, “red”)** : Representa una leyenda en la gráfica.

- **x= “”** representa el lugar donde se sitúa la leyenda (top, bottom, topleft…)
- **c(“”)** : representa los nombres en la leyenda, se almacena como un vector
- **fill= c(“”)** : representa un cuadrado con el color asignado al lado de cada nombre

Como bien hemos mencionado, el único comando obligatorio para representar la gráfica es el primero **plot(a,b)** mientras que el resto son opcionales, ya que tienen como único fin modificar la gráfica.

1.2. Bucles

® **for (i in vinic:vfin) {**

Proceso de cálculo

}

Este es el comando general donde vinic y vfin significan valores de inicio y fin del bucle, la manera que tienen los bucles de funcionar es ir recorriendo valores desde un valor inicial hasta uno final y por esto es que se escribe de esta forma.

Es importante entender que este es el mecanismo detrás de los bucles para hacer el ejercicio que harás en clase de suma y multiplicación de vectores. Como estas dos operaciones se hacen componente a componente, el bucle irá realizando la operación, llamada “proceso de cálculo” para cada valor de la variable i, que en este caso es la **posición** del vector.

1.3. Bucles anidados

- **for (i in vinic:vfin) {**

for (j in vinic2:vfin2) {

for (k in vinic3:vfin3) {

Proceso de cálculo

}

}

}

Siendo i, k y j tres variables distintas, este tipo de estructuras sirven cuando se quieren emplear varias variables que entre sí se relacionan en uno o varios procesos de cálculo como verás en clase a la hora de hacer esta práctica. En el caso concreto de esta práctica realizarás o habrás ya realizado operaciones con matrices, estas consisten de filas y columnas, por lo que deben necesariamente asignarse dos variables para definir las dos dimensiones de la matriz y trabajar con ellas.

1.4. Funciones en R

- **Nombre de la función <- function(argumento1, argumento2,…){** expresión de la función **}**

Definir funciones en R es relativamente sencillo pues tiene las funciones básicas integradas, como puedes ver por la estructura del comando es una simple asignación, a la izquierda elegirás arbitrariamente el nombre que desees darle a la función, a la derecha, *function* es el comando, por lo que debe escribirse así siempre, los argumentos son los valores de x que deberá tomar la función, es decir el vector que se haya empleado para generar el eje de abscisas. La expresión de la función es lo que ya conoces que es una función, una expresión matemática dependiente de una variable x, por ejemplo, sen(x).

Una vez entiendes que compone una función (los valores en x y sus correspondientes imágenes) este apartado es muy sencillo, a la hora de representar la función sólamente hay que emplear los comandos del apartado 1.1

2. Desarrollo de la práctica:

2.1. Representación gráfica de funciones

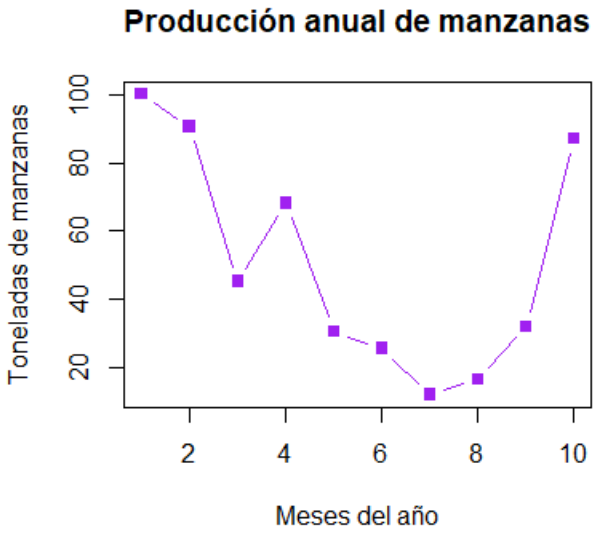
1	2	3	4	5	6	7	8	9	10
100	90.50	45.23	68.14	30.60	25.55	12.01	16.48	32.00	87.10

1) Construimos un vector *meses* que contenga los números de 1 a 10 asignados a los meses y uno llamado *manzanas* que contenga la producción de manzanas a partir de los datos.

```
> meses=seq(1,10,1)
> manzanas<-c(100,90.50,45.23,68.14,30.60,25.55,12.01,16.48,32.00,87.10)
```

2) Representamos gráficamente la producción mensual de manzanas. Usando línea continua y color morado. Etiquetamos los ejes como ‘Meses del año’ (abscisas) ‘Toneladas de manzanas’ (ordenadas). Ponemos como título ‘Producción anual de manzanas’ y utilizaremos el símbolo del cuadrado relleno

```
plot(meses,manzanas,type='b',col='purple',xlab='Meses del año',ylab='Toneladas de manzanas',pch=15,main='Producción anual de manzanas')
#eje x, eje y, tipo de linea, color del gráfico, nombre de los dos ejes, símbolo de los puntos, título de la gráfica
```



2.1. Ejercicios con bucles

v=(12,-3,5,18.7) y w=(12,0.25,77,exp(2))

1) Obtener la suma de los dos vectores mediante bucles y comprobarlo

```
> v<-c(12,-3,5,18.7)
> w<-c(12,0.25,77,exp(2))
> u=c(0) #tenemos que definirlo antes, también podemos poner u=0
> for (i in 1:length (v)){
+   u[i]=v[i]+w[i]
+ }
> u
[1] 24.00000 -2.75000 82.00000 26.08906
> v+w
[1] 24.00000 -2.75000 82.00000 26.08906
```

2)Obtener la suma de las componentes del vector v y almacenarlos en SumaC y comprobarlo

```
> SumaC=0
> for(i in 1:length(v)){
+  SumaC=SumaC+v[i]
+ }
> SumaC
[1] 32.7
```

```
> sum(v)
[1] 32.7
```

3)Realizar el producto escalar de ambos vectores mediante bucles y comprobarlo

```
> Pesc=0
> for(pepin in 1:length(v)){
+   Pesc=Pesc+v[pepin]*w[pepin]
+ }
> Pesc
[1] 666.4253
> v%*%w
         [,1]
[1,] 666.4253
```

4) Multiplicar ambos vectores componente a componente

```
> Mcomp=0
> for(i in 1:length(v)){
+   Mcomp[i]=v[i]*w[i]
+ }
> Mcomp
[1] 144.0000  -0.7500 385.0000 138.1753
> v*w
[1] 144.0000  -0.7500 385.0000 138.1753
```

5)Hacer una tabla con los resultados

```
> Titulo=c('Suma','Producto escalar') #filas
> Valor=c(SumaC, Pesc) #columnas
> data.frame (Titulo,Valor) #escribe la tabla con las anteriores filas y columnas
           Titulo    Valor
1            Suma  32.7000
2 Producto escalar 666.4253
```

2.3. Ejercicios con bucles anidados

1) Construir una matriz A1 de manera que los vectores v, w, sean sus filas, y una matriz A2 de manera que los mismos vectores sean sus columnas.

```
> A1=rbind(v,w)
> A1
  [,1]  [,2] [,3]      [,4]
v   12 -3.00    5 18.700000
w   12  0.25   77  7.389056
> A2=cbind(v,w)
> A2
         v         w
[1,] 12.0 12.000000
[2,] -3.0  0.250000
[3,]  5.0 77.000000
[4,] 18.7  7.389056
```

2) Multiplicar, empleando bucles, ambas matrices, obteniendo una matriz C. Antes de operar tenemos que igualarla a 0. Al final comprobaremos el resultado.

```
> C=matrix(c(0),nrow=nrow(A1),ncol=ncol(A2))
```

Si solo ponemos C=0 serían todo filas de 0, por eso igualamos a las columnas y filas de A1 y A2, según las de la matriz resultante.

```
> for(i in 1:nrow(C)){ #es lo mismo que nrow(A1)
+   for(j in 1:ncol(C)){ #es lo mismo que ncol(A2)
+     for(k in 1:ncol(A1)){
+       C[i,j]=C[i,j]+A1[i,k]*A2[k,j]
+     }
+   }
+ }
> C
          [,1]      [,2]
[1,] 527.6900  666.4253
[2,] 666.4253 6127.6607
> A1%*%A2
          v         w
v 527.6900  666.4253
w 666.4253 6127.6607
```

Repetido con explicaciones y pequeños cambios también válidos

```
> A1=rbind(v,w) # Combinando v, w por filas
> A2=cbind (v,w) # Combinando v,w por columnas
> #Hemos construido A1, A2
> #La matriz resultante C tendra tantas filas como A1 y tantas columnnas como A2
> m=nrow(A1) #Por comodidad m es num filas A1
> n=ncol(A2) #n es num col A2
> p=ncol(A1)
> #Inicializamos C con ceros
> C=matrix(c(0), nrow=m, ncol=n)
> #Hacemos la multiplicacion
> for(i in 1:m) { #bucle para filas de C
+   for(j in 1:n) { #bucle para coli,nas de C
+     for(k in 1:p){ #bucle columnas A1
+       C[i,j]=C[i,j]+A1[i,k]*A2[k,j]
+     }
+   }
+ }
> C
          [,1]      [,2]
[1,] 527.6900  666.4253
[2,] 666.4253 6127.6607
> #Comprobacion
> A1%*%A2
          v         w
v 527.6900  666.4253
w 666.4253 6127.6607
```