RESUMEN PRÁCTICA 4

En el curso 20-21, en la sesión de prácticas número 4 terminamos lo que había faltado de la práctica 3. En caso de solo querer acceder al trabajo de la práctica 4, ir a la página 4. De todas formas, recomendamos leer también la información de la práctica 3 ya que es necesaria para entender la 4.

PRÁCTICA 3

Vamos a realizar un ejercicio con varios pasos en el que representaremos tres funciones superpuestas en una sola gráfica.

Ejercicio:
1. Define la función $f(x) = x^2\cos(x^2)$ y obtén el valor f(pi/4)
2. Obtén un vector xx que tome 1001 valores en [0,10] (utilizando el comando seq con «salto» 0.01).
3. Representa gráficamente la función f, tomando como abscisas los valores xx. Etiqueta el eje de abscisas con 'Abscisa' y el eje de ordenadas con 'mis funciones f,g,h'. La gráfica irá en color verde.
4. Define la función $g(x) = \sin(x^2)e^{-\frac{x^2}{10}}$
5. Utiliza la instrucción: par(new="true") para superponer curvas.
6. Dibuja la función g en los puntos xx en color azul y empleando: xlab='', ylab='',axes=FALSE (para eliminar ejes),pch=4 (para símbolos)
7. Idem a 5-6 con la función $h(x) = \sin(x^8)e^{-x}$ en rojo y usando pch=18
8. Añade a continuación la leyenda:

legend(x = "top", c("función f", "función g","función h"), fill = c("green", "blue", "red"))

PASO 1, 4 y 7

Los pasos 1,4 y 7 se basan en la creación de funciones, muy importante en R.

Creamos 3 funciones distintas, asociando a cada una un nombre distinto. "function" le dice a R que vamos a meter una función. Entre paréntesis asignamos las variables de entrada (también denominadas argumentos). Entre llaves ponemos la expresión de las funciones.

La estructura de las funciones siempre será:

A= function (argumento 1, argumento 2) {

proceso u operaciones

}

f= function (x) {x^2*cos(x^2)} $f(x) = x^2\cos(x^2)$

g= function (x) {sin(x^2)*exp(-x^2/10)} $g(x) = \sin(x^2)e^{-\frac{x^2}{10}}$

h= function (x) {sin(x^8)*exp(-x)} $h(x) = \sin(x^8)e^{-x}$

Recomendación del profesor:

Poner todas las funciones al principio del script nos ayudará a organizarlo y así no olvidar ponerlas

- Al ser x una variable, le podemos dar el nombre que queramos. Si ponemos juan, en el cálculo de la función habría que poner juan en vez de x. f sería:
  f= function (juan) {juan ^2*cos(juan ^2)}
- En estas funciones solo hay una variable (x), pero puede que haya varias (x,y,z,i,n)
- Además, las funciones solo tienen unas operaciones, pero en prácticas posteriores habrá que abrir la función y no cerrarla hasta casi el final del script ya que puede haber muchas estructuras condicionales, bucles, operaciones variadas... dentro de la función.

Una vez hemos incluido las operaciones a realizar dentro de la función tenemos que LLAMAR A LA FUNCIÓN. Parece una tontería, pero si no llamamos a la función el programa no va a realizar NADA.

Para llamar a la función solo tenemos que escribir el nombre que le hemos puesto a esa función y entre paréntesis el valor para el que queremos calcular esas operaciones.

f(pi/4) g(3) h(0.8)

```
> f(pi/4)
[1] 6.351509
> g(3)
[1] 0.1675549
> h(0.8)
[1] 0.07503174
```

Explicación → Cuando escribimos f(pi/4) estamos diciéndole al ordenador que busque la función f. Una vez encontrada, que particularice la expresión cuando x valga pi/4 y nos muestre el resultado de la operación.

PASO 2

En el paso 2 hay que crear valores desde el 0 hasta el 10 y que la distancia entre cada valor sea 0.01. Es decir, que sean puntos equidistantes.

Usamos:

xx= seq(0,10,0.01)

xx (para que nos muestre todos los valores)

```
> xx= seq(0,10,0.01)
> xx
  [1]  0.00  0.01  0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.10  0.11
 [13]  0.12  0.13  0.14  0.15  0.16  0.17  0.18  0.19  0.20  0.21  0.22  0.23
 [25]  0.24  0.25  0.26  0.27  0.28  0.29  0.30  0.31  0.32  0.33  0.34  0.35
 [37]  0.36  0.37  0.38  0.39  0.40  0.41  0.42  0.43  0.44  0.45  0.46  0.47
 [49]  0.48  0.49  0.50  0.51  0.52  0.53  0.54  0.55  0.56  0.57  0.58  0.59
 [61]  0.60  0.61  0.62  0.63  0.64  0.65  0.66  0.67  0.68  0.69  0.70  0.71
 [73]  0.72  0.73  0.74  0.75  0.76  0.77  0.78  0.79  0.80  0.81  0.82  0.83
 [85]  0.84  0.85  0.86  0.87  0.88  0.89  0.90  0.91  0.92  0.93  0.94  0.95
 [97]  0.96  0.97  0.98  0.99  1.00  1.01  1.02  1.03  1.04  1.05  1.06  1.07
[109]  1.08  1.09  1.10  1.11  1.12  1.13  1.14  1.15  1.16  1.17  1.18  1.19
[121]  1.20  1.21  1.22  1.23  1.24  1.25  1.26  1.27  1.28  1.29  1.30  1.31
[133]  1.32  1.33  1.34  1.35  1.36  1.37  1.38  1.39  1.40  1.41  1.42  1.43
[145]  1.44  1.45  1.46  1.47  1.48  1.49  1.50  1.51  1.52  1.53  1.54  1.55
[157]  1.56  1.57  1.58  1.59  1.60  1.61  1.62  1.63  1.64  1.65  1.66  1.67
[169]  1.68  1.69  1.70  1.71  1.72  1.73  1.74  1.75  1.76  1.77  1.78  1.79
[181]  1.80  1.81  1.82  1.83  1.84  1.85  1.86  1.87  1.88  1.89  1.90  1.91
[193]  1.92  1.93  1.94  1.95  1.96  1.97  1.98  1.99  2.00  2.01  2.02  2.03
[205]  2.04  2.05  2.06  2.07  2.08  2.09  2.10  2.11  2.12  2.13  2.14  2.15
[217]  2.16  2.17  2.18  2.19  2.20  2.21  2.22  2.23  2.24  2.25  2.26  2.27
[229]  2.28  2.29  2.30  2.31  2.32  2.33  2.34  2.35  2.36  2.37  2.38  2.39
[241]  2.40  2.41  2.42  2.43  2.44  2.45  2.46  2.47  2.48  2.49  2.50  2.51
[253]  2.52  2.53  2.54  2.55  2.56  2.57  2.58  2.59  2.60  2.61  2.62  2.63
[265]  2.64  2.65  2.66  2.67  2.68  2.69  2.70  2.71  2.72  2.73  2.74  2.75
[277]  2.76  2.77  2.78  2.79  2.80  2.81  2.82  2.83  2.84  2.85  2.86  2.87
[289]  2.88  2.89  2.90  2.91  2.92  2.93  2.94  2.95  2.96  2.97  2.98  2.99
[301]  3.00  3.01  3.02  3.03  3.04  3.05  3.06  3.07  3.08  3.09  3.10  3.11
[313]  3.12  3.13  3.14  3.15  3.16  3.17  3.18  3.19  3.20  3.21  3.22  3.23
[325]  3.24  3.25  3.26  3.27  3.28  3.29  3.30  3.31  3.32  3.33  3.34  3.35
[337]  3.36  3.37  3.38  3.39  3.40  3.41  3.42  3.43  3.44  3.45  3.46  3.47
[349]  3.48  3.49  3.50  3.51  3.52  3.53  3.54  3.55  3.56  3.57  3.58  3.59
[361]  3.60  3.61  3.62  3.63  3.64  3.65  3.66  3.67  3.68  3.69  3.70  3.71
[373]  3.72  3.73  3.74  3.75  3.76  3.77  3.78  3.79  3.80  3.81  3.82  3.83
[385]  3.84  3.85  3.86  3.87  3.88  3.89  3.90  3.91  3.92  3.93  3.94  3.95
[397]  3.96  3.97  3.98  3.99  4.00  4.01  4.02  4.03  4.04  4.05  4.06  4.07
[409]  4.08  4.09  4.10  4.11  4.12  4.13  4.14  4.15  4.16  4.17  4.18  4.19
[421]  4.20  4.21  4.22  4.23  4.24  4.25  4.26  4.27  4.28  4.29  4.30  4.31
[433]  4.32  4.33  4.34  4.35  4.36  4.37  4.38  4.39  4.40  4.41  4.42  4.43
[445]  4.44  4.45  4.46  4.47  4.48  4.49  4.50  4.51  4.52  4.53  4.54  4.55
[457]  4.56  4.57  4.58  4.59  4.60  4.61  4.62  4.63  4.64  4.65  4.66  4.67
[469]  4.68  4.69  4.70  4.71  4.72  4.73  4.74  4.75  4.76  4.77  4.78  4.79
[481]  4.80  4.81  4.82  4.83  4.84  4.85  4.86  4.87  4.88  4.89  4.90  4.91
```

Otra forma para calcular el paso 2 es en vez de poner el valor entre cada punto (“salto” =0,01) escribir que el vector tiene que tener 1001 componentes:

xx= seq (0,10, length=1001)

xx= seq (0,10, length.out=1001)

Podemos evaluar la función f en todo el vector xx. Las funciones no solo se pueden evaluar en número, también se pueden evaluar en vectores. El programa va cogiendo cada coordenada del vector y realiza el cálculo para cada una de ellas:

f(xx)

PASO 3,5 y 6

Vamos a crear una gráfica. Recuerda que debes usar el comando plot.

```
plot (xx, f(xx), col="green", xlab="Abscisa", ylab="mis funciones f,g,h", main="Primer gráfico")

par(new ="true")

plot (xx,g(xx),col="blue",xlab="",ylab="",pch=4, axes="FALSE")

par(new ="true")

plot (xx,h(xx),col="red",xlab="",ylab="",pch=18, axes="FALSE")
```

En este momento tenemos un nuevo concepto muy importante. Es el comando par (new ="true"). Sirve para decirle al ordenador que la siguiente gráfica que vamos a poner (la de la función g) la superponga sobre la gráfica anterior (la de la función f) usando el mismo gráfico.

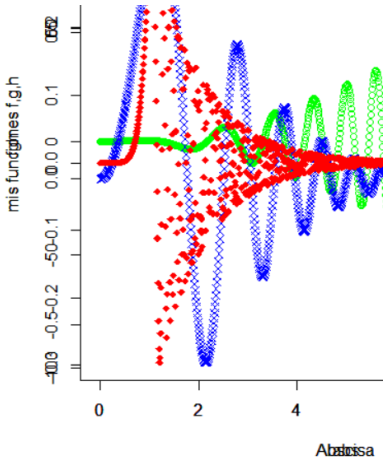
La instrucción par(new ="true") hay que ponerla entre cada 2 gráficas. Si queremos superponer 3 gráficas tenemos que ponerlo 2 veces (como en el script anterior).

¿qué ocurre si no ponemos par(new ="true")? → Se dibujará la primera gráfica y al crear la segunda, se borra la primera.

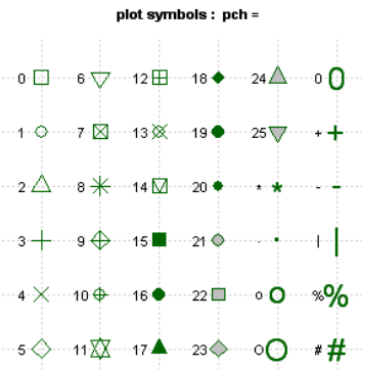
En la segunda y tercera gráfica hemos puesto (axes="FALSE"). Esta instrucción sirve para que no ponga ejes en esas funciones y así evitar que se superpongan. Como las escalas son distintas, superpone los números de los tres ejes.

Además, tenemos que quitar los nombres de los ejes porque si no, salen tres nombres uno encima de otro. Para esto hacemos: xlab="" ylab="" y evitamos superposición.

Si no hacemos estas eliminaciones quedaría así:



Por último, recordemos que pch modificaba el tamaño y forma de los símbolos. Varía entre 0 y 25:



PASO 8

En el paso 8 añadimos la leyenda de nuestro gráfico:

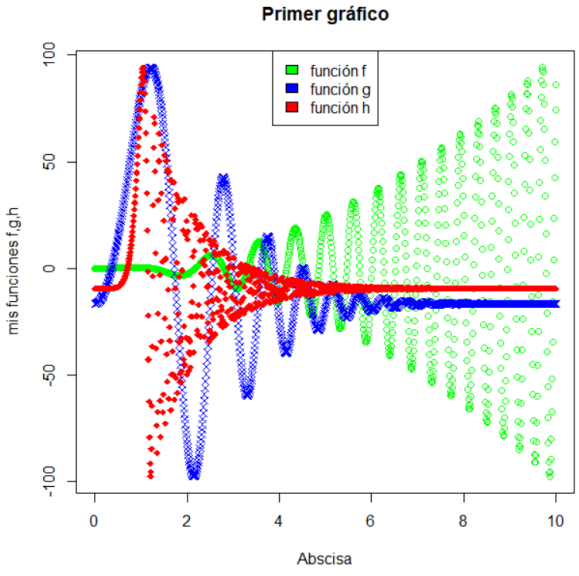
```
legend (x="top", c ("función f", "función g", "función h"), fill= c ("green", "blue", "red"))
```

- x=top sirve para que la leyenda se localice arriba del gráfico. También podemos ponerla en distintas posiciones:

  "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"

- Debemos usar un vector para poner los nombres que queremos en la leyenda c("función f","función g","función h")

- Usamos Fill para los cuadraditos de la leyenda. Hay que ponerlo en el mismo color que las gráficas (respetar el orden)



Ya hemos terminado el gráfico. ¿Qué tal te ha quedado? ¿Algo así?

Tras realizar este ejercicio recomendamos realizar el ejercicio propuesto del IMC (http://trabajo-cooperativo.net/2020/12/17/ejercicio-propuesto-calculo-imc/) para practicar bucles, condicionales, gráficas...

PRÁCTICA 4

La práctica 4 se centró en manejar las estructuras condicionales.

Tenemos dos tipos de estructuras condicionales: las de if (secuenciales) y las de while (condicionales).

```
Condiciones
if (Condición 1){
    Proceso 1
} else if (Condición 2){
    Proceso 2
} else if (Condición 3){
    Proceso 3
} ... else if (Condición n){
    Proceso n
}else{
    Proceso n+1
}
```

Importante --> Intentar poner las llaves siempre de la forma estipulada (una en la misma línea de comienzo, y otra en una línea sola para cerrarlo) porque a veces puede dar error si se ponen de forma distinta.

En las estructuras If, else if, else, podemos usar lo que queramos, dependiendo de cuantas condiciones tengamos. Se suele usar if para la primera condición, else cuando ya no tenemos más condiciones, y el resto (las del centro) con else if.

También podemos usar bucles condicionales (con while): Tienen que cumplir varias condiciones para su funcionamiento:

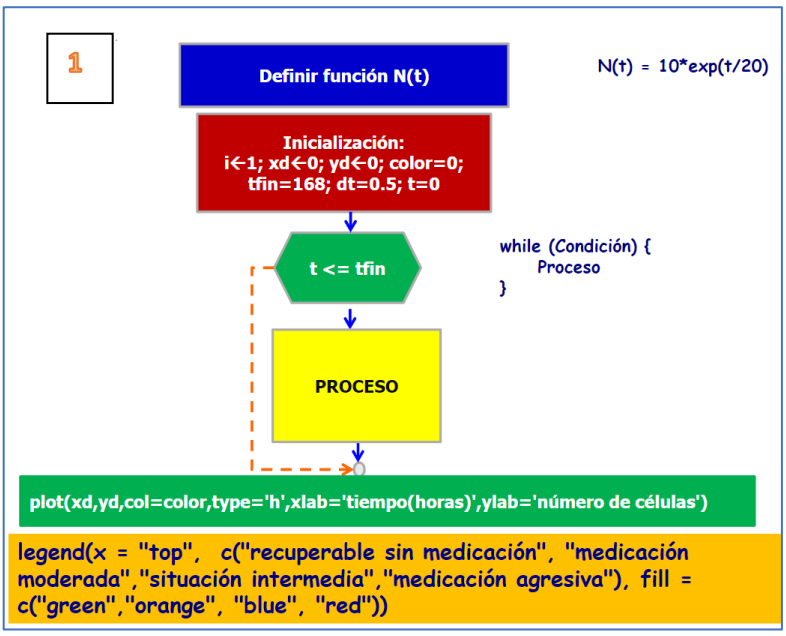
```
Bucles
while (Condición) {
Proceso
}
```

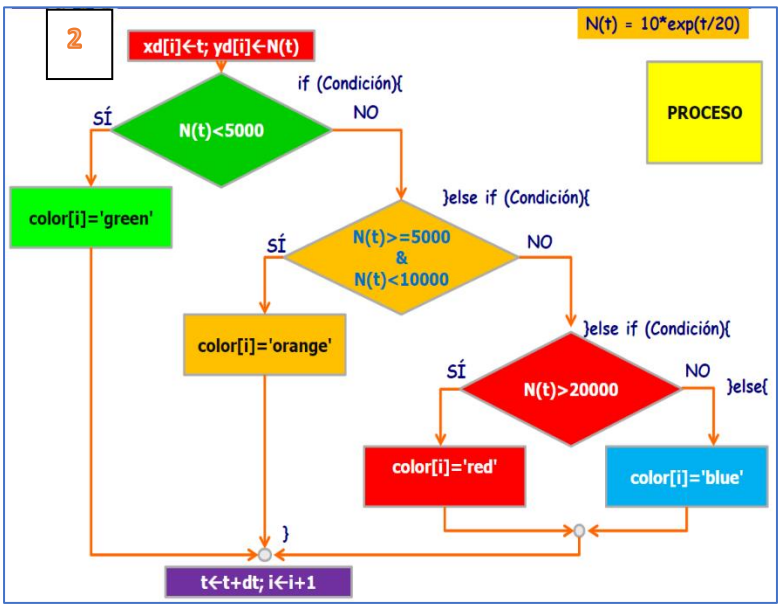
1. Tiene que tener una condición que se cumpla al principio para poder entrar en el bucle.
2. Tienen que dejar de cumplirse en algún momento (no pueden ser bucles infinitos).
3. Como la variable no se va incrementando sola, hay que poner que i=i+1 dentro del bucle.
4. La variable no se inicia sola a un valor (como si ocurre con if). Debemos iniciarla nosotros antes del bucle.

Ejemplo de bucles con while:

```
Inicio pseudo-código
  Leer v, n
  i=1
  suma=0
  Mientras (i<=n) hacer
        suma=suma+v_i
        i=i+1
  Fin del bucle
Fin pseudo-código
```

EJERCICIO MEDICACIÓN





1º Definimos la función N. En este caso, la variable no será x, sino t. Es importante poner un paréntesis para que la exponencial sea de t/20, y no la exponencial de t y luego dividir entre 20.

2º Inicializar todos esos vectores a los valores que aparecen en el cuadro rojo.

3º Crear un bucle while. La condición será (t<=tfin). Dentro del bucle, el proceso a seguir será el representado en la imagen 2. Iniciamos xd e yd tal y como viene escrito.

4º Aún dentro del bucle tenemos que plantear los condicionales. Comenzamos por if, else if, y en la última condición else. Recuerda que para que se cumplan dos condiciones simultáneamente usamos &.

5º Tenemos que acordarnos antes de cerrar el bucle while que debemos incrementar las variables (rectángulo morado), ya que esto no sucede por sí solo en este tipo de bucles.

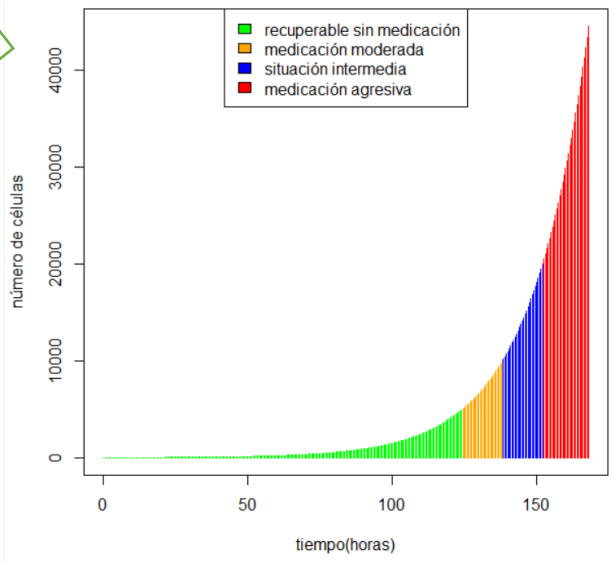
6º Cerramos el bucle while y pasamos a escribir el plot para representar la función. Para el color vamos a usar el que corresponda según los intervalos. Para ello, vamos a poner el vector color que tiene almacenados datos de colores. Ponemos también nombre a los ejes.

7º Por último vamos a crear la leyenda y la vamos a poner arriba del gráfico. Almacenaremos en un vector las explicaciones de los colores, y en EL MISMO ÓRDEN los colores del relleno de los cuadros.

Importante!!

Cuando usamos else NO hay que poner ninguna condición, porque else se usa para: "en cualquier otro caso".

Este sería el gráfico que habría que obtener:



El script terminado:

```
N=function (t){10*exp(t/20)}
i=1; xd=0; yd=0; color=0; tfin=168; t=0; dt=0.5

while (t<=tfin){
xd[i]=t;
yd[i]=N(t)

   if (N(t)<5000){
      color[i]= "green"
   } else if (N(t)>=5000 & N(t)<10000){
      color [i]="orange"
   } else if (N(t)>20000){
      color [i]="red"
   } else {
      color [i]= "blue"}

 t=t+dt; i=i+1}

plot (xd,yd,col=color,type="h",xlab="tiempo(horas)",ylab="número de células")
legend (x="top", c("recuperable sin medicación","medicación moderada", "situación intermedia", "medicación agresiva"),
          fill=c ("green","orange","blue","red"))
```