PRÁCTICA 2 Y 3
REPRESENTACIÓN GRÁFICA DE FUNCIONES Y BUCLES

¡BIENVENIDO a este recurso creado por el grupo T8 del curso 2324! Este resumen abarca las prácticas 2 y 3 impartidas por el profesor Arturo Hidalgo. Su objetivo es introducir la representación de funciones en R y programar bucles simples y anidados.

En este resumen, no solo te mostraremos cómo resolver las prácticas, sino que también ofreceremos anotaciones y consejos para aclarar conceptos básicos. Se examinan los errores comunes detectados en estas prácticas y se abordan posibles dudas.

Previamente, se recomienda tener claros algunos conceptos básicos. Puedes visitar nuestros recursos sobre VectoRes y MatRices, así como IntRoducción a InstRucciones Básicas en R. Estos enlaces te dirigirán a materiales sobre vectores, matrices y sus operaciones, además de conceptos fundamentales sobre algoritmia.

Además, para facilitar la comprensión de los estos ejercicios propuestos en la práctica, ofrecemos enlaces sobre Gráficos en R y el uso del Bucle "foR". Estos recursos explican de manera clara y detallada el manejo de gráficos y bucles en R.

Esperamos que esta guía simplificada te ayude a abordar y comprender mejor los ejercicios.

ÍNDICE INTERACTIVO

(Navega más rápido haciendo clic en la sección de tu interés)

1. REPRESENTACIÓN GRÁFICA DE FUNCIONES 3
1.1. COMANDOS RELEVANTES 3
1.2. DESARROLLO DEL EJERCICIO → Manzanas y Meses 4
 # Pregunta 1 → Construir un vector meses 4
 # Pregunta 2 → Construir un vector manzanas 4
 # Pregunta 3 → Comandos plot(), type="" y col="" 5
 # Pregunta 4 → Etiquetar Ejes 5
 # Pregunta 5 → Comando pch 6
 # Pregunta 6 → Título Gráfico 6
2. BUCLES 7
2.1. COMANDOS RELEVANTES 7
2.2. DESARROLLO DEL EJERCICIO→ Vectores v y w 7
 # Pregunta 1 → Suma de los vectores 7
 # Pregunta 2 → Suma componentes de v 8
 # Pregunta 3→ Producto escalar 8

 # Pregunta 4→ Multiplicación de Vectores 9
 # Pregunta 5→ Operación $zj = vj + 2 * wj$ 9
 # Pregunta 6→ Data.frame 10
3. BUCLES ANIDADOS 11
3.1. COMANDOS RELEVANTES 11
3.2. DESARROLLO DEL EJERCICIO → Matrices A1 y A2 11
 # Pregunta 1 → Construir una matriz A1 11
 # Pregunta 2 → Construir una matriz A2 12
 # Pregunta 3 → Multiplicar A1 y A2 12
 # Pregunta 4 → Comprobar la multiplicación de matrices 14
 # Pregunta 5 → Inventar una matriz D 14
 # Pregunta 6 → Sumar C y D 14
 # Pregunta 7 → Multiplicar C y D 15
4. FUNCIONES EN R 16
4.1. COMANDOS RELEVANTES 16
4.2. DESARROLLO DEL EJERCICIO → f(x)=x^2 cos(x^2)... 17
 # Pregunta 1 → Definir $fx = x2\ cos(x2)$ 17
 # Pregunta 2 → Obtener un vector XX 17
 # Pregunta 3 → Representa gráficamente la función f 18
 # Pregunta 4 → Definir $gx = sinx2\ e - x210$ 18
 # Pregunta 5 → Comando Par (new= "") 18
 # Pregunta 6 → Dibuja la función g 18
 # Pregunta 7 → Definir $hx = sinx8\ e - x$ 19
 # Pregunta 8 → Añadir leyenda 19
 # VISIÓN GENERAL DEL CÓDIGO 20

1. REPRESENTACIÓN GRÁFICA DE FUNCIONES

1.1. COMANDOS RELEVANTES

Para representar un gráfico en R, utilizamos el comando **plot()**. Dentro de este comando, introducimos los nombres de los vectores que contienen los datos a representar, junto con otras instrucciones para personalizar la visualización del gráfico.

Ejemplo: tenemos los vectores **a** y **b**

plot (**a**, **b**, type= "**b**", col= "**green**", xlab= "**tiempo (s)**", ylab= "**temperatura (ºC)**", pch=**2** , main= "**gráfico 1**")

① **plot(a, b)** → genera un gráfico donde **a** representa el eje x y **b** el eje y (ambos vectores deben estar previamente definidos).

Es crucial comprender que se utilizan vectores como ejes, ya que en R un vector es una forma de almacenar información, es decir, los puntos que constituyen un eje.

② **type= ""** → define la forma de representación gráfica. Esta opción determina cómo se visualizará el gráfico y varía según la letra especificada dentro de los paréntesis.

- **p** muestra solo los puntos.
- **l** muestra solo las líneas que unen los puntos
- **b** muestra tanto líneas como puntos
- **h** muestra en forma de histograma (líneas verticales)
- **n** no muestra nada, ni puntos ni líneas

③ **col= ""** → cambia el color del gráfico, le damos color a la línea y los puntos del gráfico. Se puede escribir de diferentes formas:

- **col= "green"** → escribimos el nombre en inglés y entre comillas
- **col= "#008000"** → escribimos en código hexadecimal, entre comillas y con la almohadilla.

④ **xlab= ""** → añade una etiqueta al eje de abscisas, es decir, darle nombre al eje x.

⑤ **ylab= ""** → añade una etiqueta al eje de ordenadas, es decir, darle nombre al eje y.

⑥ **pch=** → cambia el diseño de los puntos según el número, como vemos en la imagen:

⑦ **main= ""** → asigna un título al gráfico.

Es esencial recordar que el único comando obligatorio para generar el gráfico es **plot(a,b)**. Las instrucciones adicionales son opcionales y se utilizan para modificar la apariencia del gráfico.

Un punto importante a destacar es el uso de comillas **""** → siempre que introduzcamos texto no numérico dentro de un comando en R, debe estar entre comillas para evitar errores al ejecutarlo.

1.2. DESARROLLO DEL EJERCICIO → Manzanas y Meses

Consideramos la producción de manzanas durante los 10 primeros meses del año:

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

Pregunta 1 → Construir un vector meses

Construir un vector **meses** que contenga los números de 1 a 10 asignados a los meses.

Para ello empleamos→ **seq (valor inicial, valor final, incremento).**

```
> meses= seq(1,10,1)
> meses
 [1]  1  2  3  4  5  6  7  8  9 10
```

Empleamos el comando "**seq**" para crear un vector que contenga una secuencia de números enteros entre dos valores seleccionados, con un **incremento** de **x** unidades cada vez, es decir, la diferencia o la distancia entre los valores sucesivos en el vector que creamos.

-Por ejemplo, si elegimos una variación de 2, los números en la secuencia se generarán con una brecha de 2 unidades entre ellos.

NOTA → Otra forma de generar una secuencia de números enteros entre dos valores dados es mediante el uso de los dos puntos **":"**.

-Por ejemplo, para almacenar valores enteros entre 1 y 10 en una variable "meses":

```
> meses=1:10
> meses
 [1]  1  2  3  4  5  6  7  8  9 10
```

Pregunta 2 → Construir un vector manzanas

Construir un vector llamado **manzanas** que contenga la producción mensual de manzanas, a partir de la tabla dada.

```
> manzanas=c(100,90.50,45.23,68.14,30.60,25.55,12.01,16.48,32.00,87.10)
> manzanas
 [1] 100.00  90.50  45.23  68.14  30.60  25.55  12.01  16.48  32.00  87.10
```

→ Te proporcionamos la secuencia con valores separados por comas:

(100, 90.50, 45.23, 68.14, 30.60, 25.55, 12.01, 16.48, 32.00, 87.10)

→ Si copias directamente del cuadro, asegúrate de incluir las comas para mantener la estructura de la secuencia.

¡IMPORTANTE!

Es esencial recordar que los números decimales se escriben utilizando puntos, nunca comas.

Pregunta 3 → Comandos plot(), type="" y col=""

Representar gráficamente la producción mensual de manzanas. Usando línea continua y color azul.

Empleamos el comando → plot (meses, manzanas, type="b", col="blue")

Eje de Abscisas | **Eje de Ordenadas** | **Diseño de la gráfica** | **Color de la gráfica**

¡IMPORTANTE!

Recuerda reescribir las comillas en R.

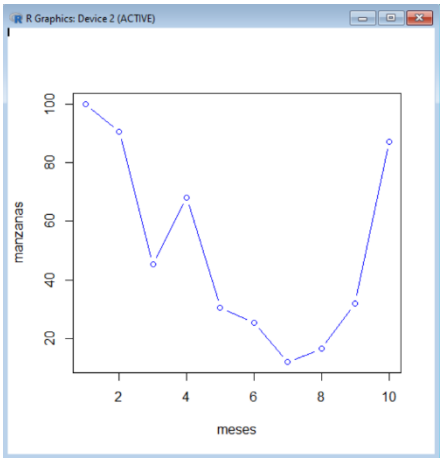
Al copiar desde nuestros apuntes o del profesor, a veces causan errores.

En este ejemplo es fácil verlo, pero al tener varias comillas es difícil detectar la incorrecta.

Evita dolores de cabeza, ¡reescribe para evitar problemas!

```
> plot (meses,manzanas, type="b", col="blue")
Error: unexpected input en "plot (meses,manzanas, type=""
> plot (meses,manzanas, type="b", col="blue")
```

Al ejecutar correctamente el comando **plot()**, se abrirá una nueva pestaña con el gráfico generado.



NOTA → Para explorar más opciones de gráficos en R, aparte del tipo "**b**", ejecuta el comando **?plot** en R. Te dirigirá a una página web con abundante información sobre diferentes tipos de gráficos disponibles.

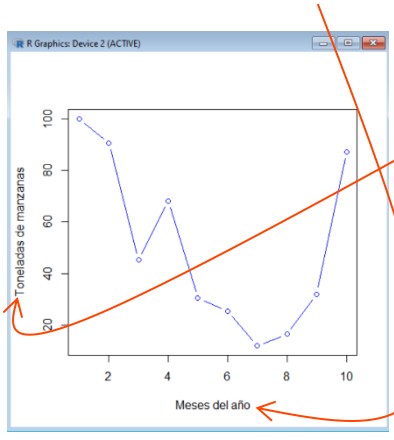
Pregunta 4 → Etiquetar Ejes

Etiquetar los ejes como: '**Meses del año**' (abscisas), '**Toneladas de manzanas**' (ordenadas).

Empleamos el comando → **xlab=""** para nombra al eje x / **ylab=""** para nombrar al eje y

Simplemente tenemos que añadir **DENTRO** del **PARENTENSIS** de **plot()** estos dos comandos → plot (meses, manzanas, type="b", col="blue", **xlab="", ylab=""**)

```
> plot(meses,manzanas, type="b", col="blue",xlab="Meses del año",ylab="Toneladas de manzanas"
```

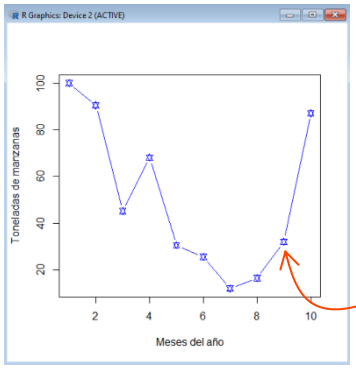


Pregunta 5 → Comando pch

Utilizar el comando → **pch**=número (entre 0 y 25) para cambiar símbolos.

Simplemente tenemos que añadir **DENTRO** del **PARENTENSIS** de **plot()** este comando

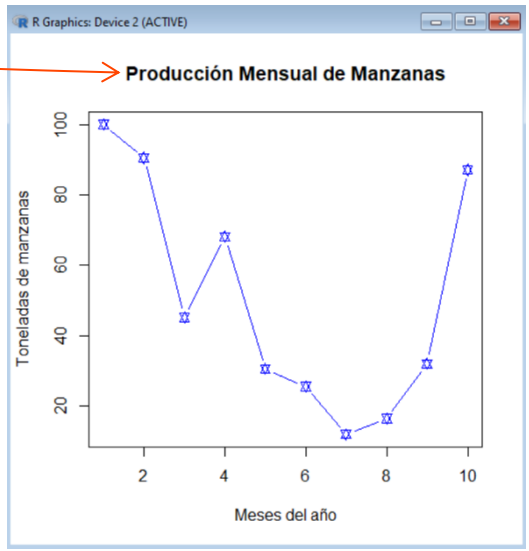
```
> plot(meses,manzanas, type="b", col="blue",xlab="Meses del año",ylab="Toneladas de manzanas",pch=11)
```



Pregunta 6 → Título Gráfico

Para poner título al gráfico emplearemos el comando→ **main=""**. Dentro del paracentesis de **plot()**

```
> plot(meses,manzanas, type="b", col="blue",xlab="Meses del año",ylab="Toneladas de manzanas",pch=11,
+ main="Producción Mensual de Manzanas")
```



2. BUCLES

2.1. COMANDOS RELEVANTES

for (i in vinic:vfin) {
Proceso de cálculo
}

El comando **"for (i in vinic:vfin)"** es un bucle que se mueve de un valor inicial ("**vinic**") a uno final ("**vfin**"). Los bucles funcionan recorriendo estos valores paso a paso.

Este proceso es la base de los bucles, fundamental para tareas como la suma o multiplicación de vectores que se hacen elemento por elemento. En cada iteración, el bucle realiza una operación llamada "**proceso de cálculo**" para cada **posición** del **vector**, representada por la variable "**i**".

2.2. DESARROLLO DEL EJERCICIO→ Vectores v y w

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

Pregunta 1 → Suma de los vectores

Obtener la suma de los dos vectores mediante **bucles** y comprobar empleando **v + w**.

```
> # Recomendable siempre comenzar con este comando para borrar todas las variables y evitar errores.
> rm(list=ls(all=TRUE))
>
>
> # Definimos previamente los vectores v y w.
> v = c(12, -3, 5, 18.7)
> w = c(12, 0.25, 77, exp(2))
>
>
> # Inicializamos el vector que será la suma de los vectores v y w.
> u = c(0)    # Otra forma es u=0
>
> # Definimos el bucle para la suma
>
> for (i in 1:length(v)) {
+     u[i] = v[i] + w[i] # Suma elemento a elemento
+ }
>
> # Mostramos el resultado de la suma.
> u
[1] 24.00000 -2.75000 82.00000 26.08906
>
> v + w # Comprobación
[1] 24.00000 -2.75000 82.00000 26.08906
```

Le indica al programa que el bucle se repita hasta la longitud del vector v, es decir, n=4.

Pregunta 2 → Suma componentes de v

Obtener la **suma** de las **componentes** del vector **v** y almacenarlos en **SumaC**.

```
# Inicializamos la variable SumaC en 0. No hace falta volver a definir el vector v
> SumaC=0
>       # Definimos el bucle para recorrer el vector v y sumar sus elementos a la variable SumaC.
> for (i in 1:length(v)){
+
+                     SumaC= SumaC + v[i]
+     }
>
> SumaC   # Muestra el resultado de la suma acumulativa de los elementos de v.
[1] 32.7
>
> # Comprobación
> sum(v)   # Comando sum() → calcula la suma total de los elementos en v.
[1] 32.7
```

Pregunta 3→ Producto escalar

Realizar el producto escalar de ambos vectores mediante bucles.

```
> Pro=0
>
>
> for (Hola in 1:length(v)){
+
+                     Pro= Pro + v[Hola] * w[Hola]
+     }
>
>
> Pro
[1] 666.4253
>
> # Comprobación
> v%*%w      # Comando %*%→ calcula el producto escalar directamente
          [,1]
[1,] 666.4253
```

#El uso de "Hola" como índice ilustra que los subíndices pueden tener cualquier nombre mientras se utilicen adecuadamente.

Suma con la multiplicación anterior

Esto implica que el elemento en la posición 1 de v se multiplica por el elemento en la posición 1 de w, y así sucesivamente para todas las posiciones coincidentes en los vectores.

Pregunta 4→ Multiplicación de Vectores

Multiplicar ambos vectores componente a componente mediante bucles.

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

 # Comprobación

Pregunta 5→ Operación $z_j = v_j + 2 * w_j$

Realizar la operación: **zj=vj+2wj → j = 1,…,length(v)** mediante bucles

```
> z=0
>
>
> for (j in 1:length(v)){
+
+                  z[j]= v[j] + 2 * w[j]
+     }
>
>
> z
[1]  36.00000  -2.50000 159.00000  33.47811
```

 # A pesar de que en el guion no se indique el * entre el 2 y w, es muy importante que indicar todas las operaciones correctamente en R para evitar errores.

 # Se suma el elemento correspondiente de v con el doble del elemento correspondiente de w.

Pregunta 6→ Data.frame

Construir una tabla, usando el comando → **data.frame** que contenga:

'Suma'	*Resultado del apartado 2)*
'Producto escalar'	*Resultado del apartado 3)*

 # Estos vectores se convertirán en las columnas del data.frame, mientras que los valores de cada uno serán las diferentes filas de la tabla.

```
> Variable= c("Suma","Producto Escalar")
>
> Valor= c(SumaC,Pro)
>
>
> data.frame(Variable,Valor)
          Variable    Valor
1             Suma  32.7000
2 Producto Escalar 666.4253
```

 # Importante aclarar que, en el vector Variable, los elementos son cadenas de texto (palabras), mientras que en el vector Valor, los elementos son valores numéricos (en este caso, las variables SumaC y Pro).

3. BUCLES ANIDADOS

3.1. COMANDOS RELEVANTES

```
for (i in vinic : vfin) {
        for (j in vinic2 : vfin2) {
                for (k in vinic3 : vfin3) {
                        Proceso de cálculo
                }
        }
}
```

Los bucles anidados, representados por estructuras como el código mostrado, utilizan múltiples variables (en este caso **i**, **k** y **j**) para realizar cálculos interrelacionados. Son útiles cuando se necesita trabajar con múltiples variables que se relacionan entre sí en uno o más procesos de cálculo.

En particular, estos bucles son útiles al realizar operaciones con matrices, las cuales tienen filas y columnas. Para trabajar con matrices, es necesario asignar dos variables que definan sus dos dimensiones (por ejemplo, **i** para las **filas** y **j** para las **columnas**). Estas estructuras permiten manejar eficientemente la complejidad de los cálculos en matrices y otros escenarios donde múltiples variables están interconectadas en un proceso.

3.2. DESARROLLO DEL EJERCICIO → Matrices A1 y A2

Dados los vectores: **v=(12,-3,5,18.7)** y **w=(12,0.25,77,exp(2))**

Pregunta 1 → Construir una matriz A1

Construir una **matriz A1** de manera que los vectores **v** y **w** sean sus **filas**.

Empleamos el comando → **rbind(v,w)**

```
> v = c(12, -3, 5, 18.7)
> w= c(12,0.25,77,exp(2))
>
> A1= rbind(v,w)
>
> A1
   [,1]  [,2] [,3]     [,4]
v   12 -3.00    5 18.700000
w   12  0.25   77  7.389056
```

 # Definimos los vectores v y w

Pregunta 2 → Construir una matriz A2

Construir una **matriz A2** de manera que los vectores **v** y **w** sean sus **Columnas**.

Empleamos el comando → **cbind(v,w)**

```
> A2= cbind(v,w)        # No es necesario volver a definir los vectores v y w
>
> A2
          v           w
[1,] 12.0 12.000000
[2,] -3.0  0.250000
[3,]  5.0 77.000000
[4,] 18.7  7.389056
```

Pregunta 3 → Multiplicar A1 y A2

Multiplicar, empleando **bucles**, ambas matrices, obteniendo una **matriz C**.

Tenemos las matrices → $A1_{(2 \times 4)} \times A2_{(4 \times 2)}$

de forma que, al multiplicarlas, obtendremos una matriz → $C_{(2 \times 2)}$

Como vemos la matriz resultante **C** tendrá tantas **filas como A1** y tantas **columnas como A2**

En el guion de prácticas, el profesor sugiere el siguiente formato para los bucles:

```
for  ( i in 1: nrow(A1) ) {
}    for ( j in 1: ncol(A2) ) {
```

Sin embargo, esta estructura puede resultar confusa al comprender el proceso por primera vez. Por ello, recomendamos utilizar el siguiente enfoque para mejorar la comprensión del código:

① Creamos las variables **m** ,**p** , **n**.

Basándonos en que C tendrá tantas filas como A1 y columnas como A2

m=número de **filas** de **A1** → **m**= **nrow(A1)**

n=número de **columnas** de **A2** → **n**= **ncol(A2)**

De forma que… $A1_{(m \times p)} \times A2_{(p \times n)} = C_{(m \times n)}$

p= número **columnas** de **A1** / número **filas** de **A2** → **p**= **ncol(A1)**

② Establecemos las dimensiones de la matriz y la inicializamos a 0.

C=matrix (c(0), nrow=**m**, ncol=**n**)

③ Mediante bucles anidados, construimos nuestra la matriz C.

```
C=0
for (i in 1:m) {   →   Bucle para ir creando las filas de C
        for (j in 1:n) {   →   Bucle para ir creando las columnas de C
                for(k in 1:p) { → Bucle para multiplicar las columnas de A1 con las filas de A2
                    C[i,j]= C[i,j]+ A1[i,k] * A2[k,j]
                }
        }
}
C
```

```
> # Creamos las variables m ,p , n
>
> m= nrow(A1)
> n= ncol(A2)
> p= ncol(A1)
>
> # Establecemos las dimensiones de la matriz y la inicializamos a 0.
>
> C=matrix (c(0), nrow=m, ncol=n)
>
> # Mediante bucles anidados, construimos nuestra la matriz C.
>
> for (i in 1:m) { #Bucle para ir creando las filas de C
+
+             for (j in 1:n) { #Bucle para ir creando las columnas de C
+
+                         for(k in 1:p) {  #Bucle para multiplicar las columnas de
+                                                              A1 con las filas de A2
+                             C[i,j]= C[i,j]+ A1[i,k] * A2[k,j]
+
+                          }  #cierre bucle de k
+
+             } #cierre bucle de j
+
+  } #cierre bucle de i
>
> C
         [,1]      [,2]
[1,] 527.6900  666.4253
[2,] 666.4253 6127.6607
```

Pregunta 4 → Comprobar la multiplicación de matrices

Verificar el resultado obtenido en la pregunta 3 → Empleamos el comando **%*%**

```
> A1 %*% A2
           v           w
v 527.6900   666.4253
w 666.4253 6127.6607
```

Pregunta 5 → Inventar una matriz D

Inventar una matriz **2x2** y llamarla D.

```
> D= matrix(c(20,12,53,15),nrow=2, ncol=2)
> D
     [,1] [,2]
[1,]   20   53
[2,]   12   15
```

Pregunta 6 → Sumar C y D

Sumar, mediante **bucles**, las matrices C y D.

```
> CD=matrix(c(0),nrow=2,ncol=2)
>
>   for (i in 1:2){
+     for (j in 1:2){
+       CD[i,j]=C[i,j]+D[i,j]
+     }
+   }
> CD
          [,1]      [,2]
[1,] 547.6900  719.4253
[2,] 678.4253 6142.6607
>
> #Comprobación
> D + C
          [,1]      [,2]
[1,] 547.6900  719.4253
[2,] 678.4253 6142.6607
```

Pregunta 7 → Multiplicar C y D

Multiplicar las matrices C y D **elemento a elemento**.

```
> CD= matrix (c(0), nrow=2, ncol=2)
>
> for (i in 1:2) {
+          for(j in 1:2) {
+                  CD[i,j]=C[i,j]*D[i,j]
+           }
+  }
>
> CD
          [,1]     [,2]
[1,] 10553.800 35320.54
[2,]  7997.104 91914.91
>
> #Comprobación
>
> C * D
          [,1]     [,2]
[1,] 10553.800 35320.54
[2,]  7997.104 91914.91
```

4. FUNCIONES EN R

4.1. COMANDOS RELEVANTES

El comando → **Par(new= "true")** superpone las gráficas. Cada vez que queramos superponer una nueva, debemos incluirlo después del **plot()** principal y por encima del **plot()** que queramos añadir.

El comando →

legend (x= "top", c ("función 1", "función 2", "función 3"), fill= c("green", "blue", "red")

representa una leyenda en la gráfica.

- **x= ""** → determina la ubicación de la leyenda (**top**, **bottom**, **topleft**, ...).
- **c("")** → almacena los nombres que aparecerán en la leyenda, y se guardan como un vector.
- **fill= c("")** → asigna un color específico junto a cada nombre en la leyenda.

Funciones en R → la definición de funciones en R es relativamente sencilla. La estructura básica del comando es una simple asignación:

Nombre de la función = function (argumento1, argumento2, ...) {

expresión de la función

}

- En el lado izquierdo de la asignación, elige arbitrariamente el **nombre de la función**.
- **function** → es el comando necesario y siempre se escribe de esta manera.
- Los **argumentos** son las variables o incógnitas que la función
- La **expresión de la función** es la función matemática que depende de la variable x, como por ejemplo sen(x).

Una vez que comprendas la composición de una función (los valores en x y sus correspondientes imágenes), utilizar estos comandos resulta sencillo. Sin embargo, si aún encuentras dudas, te recomendamos ver nuestro video explicativo Funciones en R, creado por nuestro equipo. En este video, explicamos detalladamente el proceso paso a paso para ejecutar estos comandos.

Para representar las funciones simplemente se utilizan los comandos mencionados anteriormente junto con los del apartado 1.1.

4.2. DESARROLLO DEL EJERCICIO → f(x)=x^2 cos(x^2)...

$$f(x) = x^2\cos(x^2) \quad g(x) = \sin(x^2)e^{-\frac{x^2}{10}} \quad h(x) = \sin(x^8)e^{-x}$$

Pregunta 1 → Definir $f(x) = x^2\,cos(x^2)$

Define la función $f(x) = x^2\cos(x^2)$ y obtenga el valor **f(pi/4)**

Empleamos el comando → **function () { }**

```
> f= function (x) {
+                   x^2*cos(x^2)
+   }
>
> f(pi/4)
[1] 0.5031676
```

Pregunta 2 → Obtener un vector XX

Obtén un vector **xx** que tome 1001 valores en [0,10] (utilizando el comando seq con **«salto»** 0.01).

Para crear un vector, es crucial recordar utilizar la función **c** al definirlo. Sin embargo, cuando necesitamos generar un vector compuesto por **puntos dentro de un intervalo** específico, recurrimos al comando → **seq**

```
> xx= seq(0,10,0.01)
```

El **primer número** dentro de los paréntesis indica el **inicio del intervalo**, y el **segundo número** representa el **final de este intervalo**. El **tercer número** determina el paso o la **separación entre los puntos** que queremos generar en este intervalo.

Una **alternativa** a este método es:

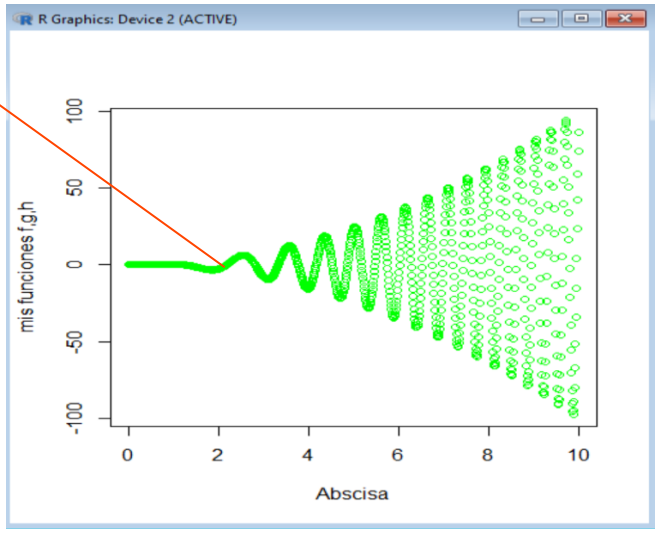
```
xx= seq(0,10,length=1001)
```

Donde **length = 1001** indica que se desea generar una secuencia con una longitud de 1001 elementos en el vector resultante.

Pregunta 3 → Representa gráficamente la función f

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**.

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



Pregunta 4 → Definir $g(x) = sin(x^2)\, e^{-\frac{x^2}{10}}$

Define la función $g(x) = sin(x^2)\, e^{-\frac{x^2}{10}}$

```
> g= function (x) {
+                  sin(x^2)*exp(-x^2/10)
+   }
```

Pregunta 5 → Comando Par (new= "")

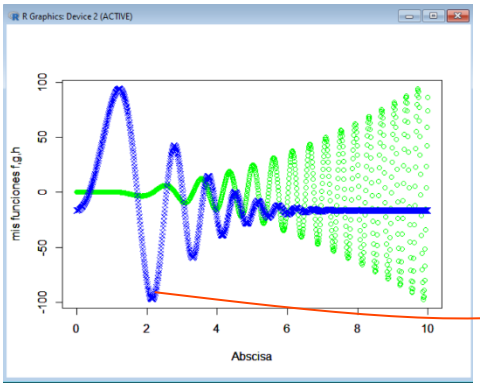
Utiliza la instrucción → **par(new="TRUE")** para superponer curvas.

Pregunta 6 → Dibuja la función g

Dibuja la función g en los puntos **xx** en color **azul**.

Empleando → **xlab=""**, **ylab=""** , **axes=FALSE** , **pch=4**

 # NOTA → este comando sirve para evitar que se representen varias veces los ejes y títulos de ejes.




```
> par(new="TRUE")
> plot(xx,g(xx),xlab="",ylab="",col="blue",axes=FALSE,pch=4)
```

Pregunta 7 → Definir $h(x) = sin(x^8)\, e^{-x}$

Repetir la pregunta 5 y 6 con la función $h(x) = sin(x^8)\, e^{-x}$, empleando pafa la representación el color **rojo** y el símbolo **pch=18**.

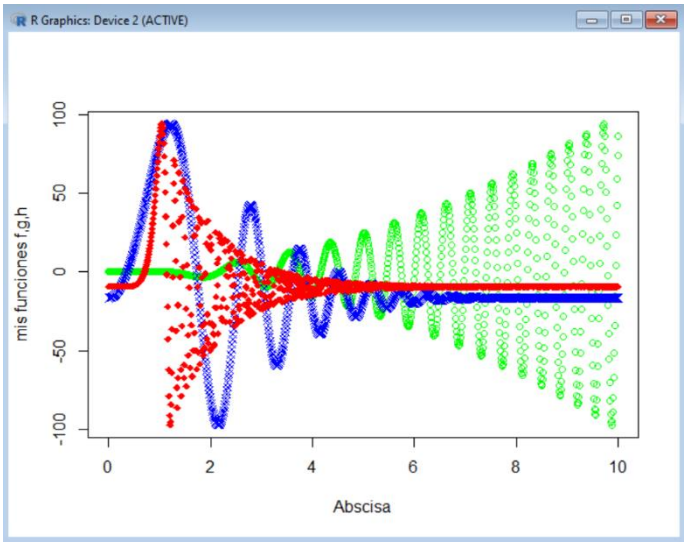
Primero hay que definir la función $h(x) = sin(x^8)\, e^{-x}$

```
> h= function (x) {
+                   sin(x^8)*exp(-x)
+   }
```

Luego procedemos a graficarla.

Usamos el comando **plot()**, y para superponerla a las demás, utilizamos **par(new="TRUE")**. Cada vez que queremos añadir una nueva función a la gráfica, necesitamos utilizar este comando junto con el nuevo **plot()**.

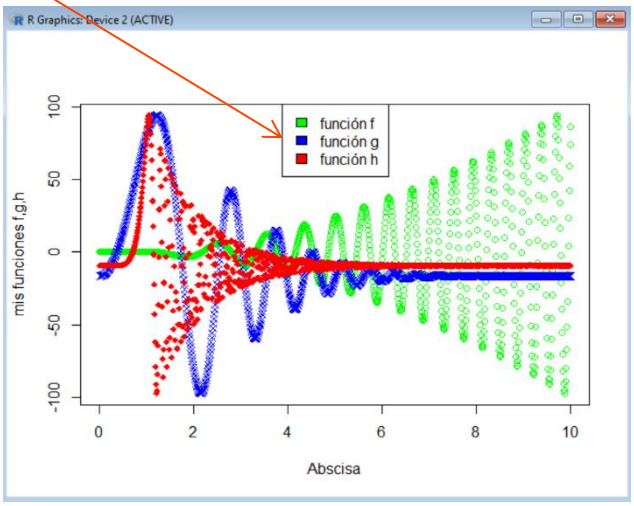
```
> par(new="TRUE")
> plot(xx,h(xx),xlab="",ylab="",col="red",axes=FALSE,pch=18)
```



Pregunta 8 → Añadir leyenda

Añade a continuación la **leyenda**.

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



Para una explicación más detallada sobre cómo añadir una leyenda, te sugerimos revisar la sección 4.1, donde se explica el uso del comando **legend ()**.

VISIÓN GENERAL DEL CÓDIGO

```
> f=function(x){x^2*cos(x^2)}
>
> xx= seq(0,10,0.01)
>
> g=function(x){sin(x^2)*exp(-(x^2)/10)}
>
> h=function(x){sin(x^8)*exp(-x)}
>
>
> plot(xx,f(xx),xlab="Abscisa",ylab="mis funciones f,g,h",col="green")
> par(new="true")
> plot(xx,g(xx),xlab="",ylab="",col="blue",axes=FALSE,pch=4)
> par(new="true")
> plot(xx,h(xx),xlab="",ylab="",col="red",axes=FALSE,pch=18)
>
>
> legend(x = "top", c("función f", "función g", "función h"),fill = c("green","blue", "red"))
```