Resumen Práctica 3 Fundamentos de Programación, R

1) Representación Gráfica de funciones

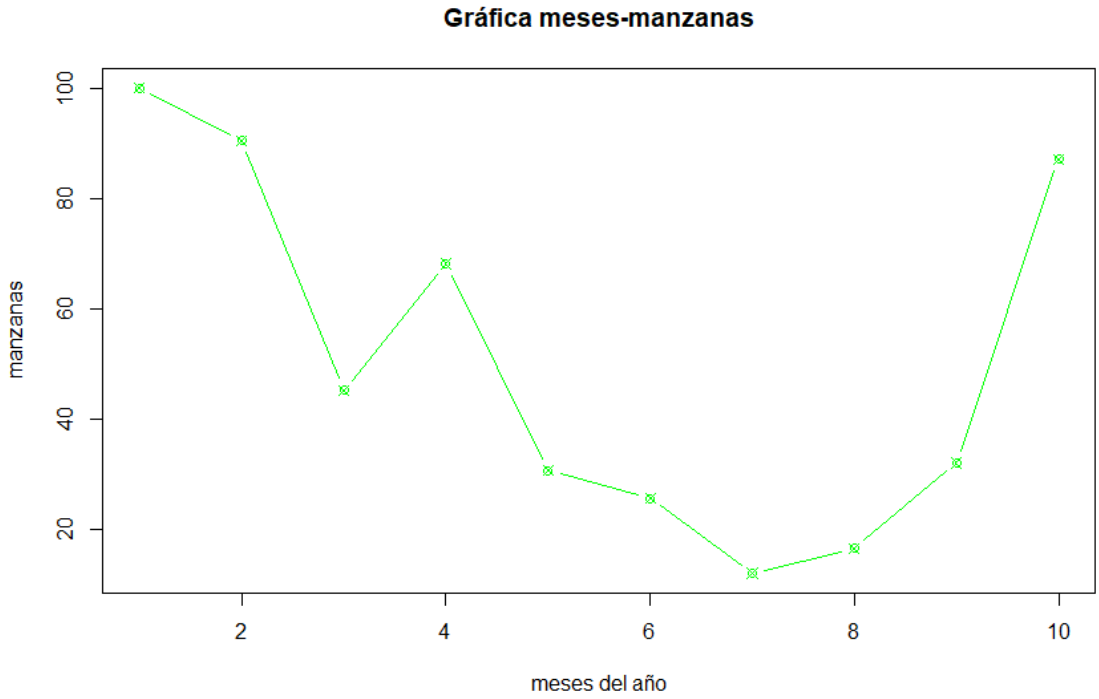
```
#Representación gráfica de funciones

meses = seq (1, 10, 1)

manzanas = c (100, 90.50, 45.23, 68.14, 30.60, 25.55, 12.01, 16.48, 32, 87.10)

plot (meses, manzanas, type="b", col="green", pch=13, xlab="meses del año",
ylab="manzanas", main="Gráfica meses-manzanas")
```

Código



En esta primera parte de la práctica, se nos ha encargado hacer una gráfica a partir de unos datos referidos a los meses de año (eje x), y a la cantidad de manzanas (eje y).

Para ello, lo primero que hemos tenido que hacer ha sido definir las variables, por lo que hemos definido primeramente la variable “meses” como una secuencia de números del 1 al 10 que avanzaba de uno en uno mediante la fórmula *meses= seq (vini, vfin, crecimiento)*. Después, hemos introducido la variable “manzanas”, introduciendo los datos en un vector.

Una vez teníamos los datos, hemos creado una gráfica a partir de la función *plot(x.y),* donde x son los meses, e y, las manzanas. A parte de eso, para que la gráfica tenga líneas que unan los puntos, se añade la función *type=”b”*, donde b es la letra que determina el tipo de gráfico. Luego también podemos darle un color con la fórmula *col=”colorx”* o también cambiar el tipo de punto con la fórmula *pch=x*, donde x es cualquier número válido. Por último, a través de las funciones *xlab* e *ylab*, se le pueden poner nombres a los ejes X e Y, respectivamente, y a través de la función *main=”titulo”* se le puede dar un título al gráfico. Todas estas funciones entran dentro de la función plot().

2) Bucles

#1# Obtener la suma de los dos vectores mediante bucles y mediante v+w

```
v= c (12, -3, 5, 18.7)
w= c (12, 9.25, 77, exp(2))
q=0
n= length(v)
for (i in 1:n){
 q[i]=v[i]+w[i]
}
q
v+w
```

Código

Para este programa primero se definen los vectores *v* y *w* con sus correspondientes variables. Después, se crea un *vector q*, que es el vector donde se van a guardar las sumas de los componentes v y w. Se iguala a 0 y se asigna un *valor n*, que es la longitud del vector.

Una vez definidas las variables se procede a sumar los vectores. Para ello, se utiliza un bucle a través de la fórmula *for (i in vinc:vfin) {Proceso},* donde i va a ir variando desde *vinc* (número inicial), hasta *vfin* (número final). El proceso es la suma de las i componentes de los vectores. Para ello, se utiliza la fórmula *q[i]=V[i]+w[i],* donde i son las componentes de cada vector, que se almacenan en la componente i del vector q. Finalmente se escribe q.

La fórmula *v+w* sirve para sumar los vectores sin necesidad de bucles.

#2# Obtener la suma de las componentes del vector v

```
SumaF=0
for (i in 1:n){
  SumaF=SumaF+v[i]
}
SumaF
```

Código

Para este programa se crea un *vector SumaF* donde se va a almacenar la suma de los componentes del *vector v*. Para ello, se recurre a la fórmula del bucle *for (i in vinc:vfin) {Proceso},* donde el proceso es que en el vector SumaF (que inicialmente vale 0) se vaya almacenando la suma de las n componentes del vector v por la fórmula *SumaF=SumaF+v[i].* Finalmente se escribe SumaF.

#3# Producto escalar de ambos vectores y comprobar utilizando v%*%w

```
Escalar=0
for (i in 1:n){
  Escalar= Escalar+v[i]*w[i]
}
Escalar
```

Código

Para este programa se crea un *vector Escalar* (que inicialmente vale 0) donde se va a almacenar el resultado del producto escalar de los *vectores v y w.* Para ello, se recurre a la

fórmula del bucle *for (i in vinc:vfin) {Proceso},* donde el proceso es que en el *vector Escalar* se van guardando las sucesivas multiplicaciones de los n componentes de los vectores v y w para dar lugar así al producto escalar mediante la fórmula *Escalar=Escalar+v[i]*w[i]*. Finalmente, se escribe *Escalar*. La fórmula *v%*%w* sirve para realizar el producto vectorial automáticamente.

#4# Multiplicar ambos vectores componente a componente

```
e=0
for (i in 1:n){
  e[i]=v[i]*w[i]
}
e
```

Código

Este programa es idéntico al de la suma de los vectores componente a componente, es decir, se crea un *vector e* (igualado a 0) donde se almacenarán los valores de las *n componentes* de los *vectores v y w* para luego, mediante un *bucle*, multiplicar los componentes mediante la fórmula *e[i]=v[i]*w[i]*. Finalmente, escribir X.

#5# Realizar la operación z[j]=v[j]+2*w[j], j=1,n

```
z=0
for (j in 1:n){
  z[j]=v[j]+2*w[j]
}
z
```

Código

Para este programa se creó un *vector z* (igualado a 0) que mediante un *bucle* almacenase el resultado de la operación *z[j]=v[j]+2*w[j]* cuando j va desde 1 hasta n. Luego se escribe z.

#6# Construir una tabla data.frame que contenga suma y producto escalar

```
a= c ("suma", "producto escalar")
b= c (SumaF, Escalar)
D= data.frame(a,b)
D
```

Código

Para este programa vamos a crear una tabla *data.frame* que consiste en una tabla que representa un conjunto de vectores. Junto a su valor, van a aparecer los vectores *SumaF* y *Escalar*, junto con sus respectivos valores.
Primero creamos un *vector a* que contiene los nombres de los vectores que se van a poner. En este caso, los nombres serán *"Suma"* y *"Producto escalar"*. Seguidamente se creará un *vector b* que almacene los valores de los dos vectores *SumaF* y *Escalar*. Por último, se crea la tabla mediante la instrucción *D=data.frame(a,b)*, donde D es el vector donde se guarda la tabla y *a,b* los vectores que aparecen en ella. Finalmente se escribe D.

```
                a         b
suma   32.7000
producto escalar 639.4253
```

3) Bucles anidados

#1#2# Se crea una matriz MatA que tenga el vector v y w como filas y otra matriz MatB que tenga los vectores v y w como columnas

```
v
w
MatA=rbind(v,w)
MatA
MatB=cbind(v,w)
MatB
```

Código

Para este programa lo primero que se hace es escribir los anteriores *vectores v y w*. A continuación, se define una *matriz A* que tendrá como *filas los vectores v y w*. Lo realizamos mediante la fórmula *MatA=rbind(v,w)*. Luego se escribe MatA.
Para la *matriz B*, que tiene a los vectores *v y w por columnas* es el mismo procedimiento, pero cambia la función ya que en vez de ser *rbind* es *cbind*. Tendríamos que hacer *MatB=cbind(v,w)*. Luego se escribe MatB.

#3# Multiplicar ambas matrices por bucles para tener una matriz C. Verificar mediante A%*%B

```
M=nrow(MatA)
P=ncol(MatA)
N=ncol(MatB)
MatC=matrix(c(0), nrow=M, ncol=N)
for (i in 1:M){
 for (j in 1:N){
  for (k in 1:P){
   MatC[i,j]=MatC[i,j]+MatA[i,k]*MatB[k,j]
  }
 }
}
MatC
MatA%*%MatB
```

Código

Para este programa lo primero que vamos a hacer va a ser asignar un número máximo a las filas de la *matrizA (M)*. También a las columnas de la *matrizA,* que será el mismo número de filas de la *matrizB* para que la matriz se pueda multiplicar *(P)*. El número máximo para las columnas de la *matrizB será N*. Posteriormente, se creará una matriz de ceros donde se almacenan los valores de las componentes multiplicadas de MatA y MatB por medio de la fórmula: *MatC=matrix(c(0), nrow=M, ncol=N),* donde se define que *MatC es una matriz de ceros*, que tiene M filas y N columnas.

Para llevar a cabo la multiplicación se recurre a un *bucle anidado* donde varían desde *1 hasta M,N y P, las filas y columnas de las matrices A y B* respectivamente. Esta multiplicación en bucle se obtiene a través de la fórmula *MatC[i,j]=MatC[i,j]+MatA[i,k]*MatB[k,j]*. En MatC se van metiendo los valores de las columnas de las matrices A y B, el número de filas o columnas de A y B va desde *1…ijk…MNP*. Finalmente se escribe MATC.
La operación *MatA%*%MatC* sirve para calcular la operación de forma directa.