Matemática Discreta
Sesión de laboratorio: comienzo con RStudio

El objetivo principal es comenzar a trabajar con el lenguaje R.

Ejercicios

1. Analizar la ayuda. Al abrir RStudio, aparece una ventana con cuatro subventanas. Cada subventana tiene su objetivo:

   (a) ventana inferior izquierda: permite escribir codigo de R; sentencias que se **interpretan** y se ejecutan.

   (b) ventana inferior derecha: muestra diferentes informaciones, cada una en una pestaa. la pestaa de la carpeta o directorio donde estamos trabajando, la pestaa de la ayuda, la pestaa de los gráficos, ...

   (c) ventana superior derecha: nuestra los valores de las variables y los cdigos de las funciones que hayamos definido y ejecutado en la ventana inferior izquierda. Es la memoria del entorno de trabajo.

   (d) ventana superior izquierda: muestra los contenidos de los ficheros de cdigo que hayamos utilizado, cada fichero en una pestaa.

   En la subventana inferior izquierda escribe **help.start()**. En la subventana de al lado, a la derecha, aparecer la ayuda sobre R; entre los diferentes apartados, en el apartado "Manuals"encontrarás "An Introduction to R". Salsea en esa introduccin, sobre todo en los apartados:

   - 2) Simple manipulations numbers and vectors,
   - (9) Grouping, loops and conditional execution,
   - (10) Writing your own functions,
   - (13) Packages, (Appendix D) Function and variable index.

   El lenguaje R tiene muchos paquetes, y en cada paquete se define un conjunto de operaciones y funciones. Las funciones bsicas de R se incluyen en el paquete llamado "base". Si queremos conocer las funciones definidas en dicho paquete, podemos escribir en la ventana inferior izquiera **library(help = "base")** que nos responde con la lista de funciones del paquete llamado "base". La lista de funciones aparecerá en la ventana superior derecha, como contenido de un fichero. Hay mucha informacin on-line, el manual on line "The R Base Package"lo encontrarás en:

   http://stat.ethz.ch/R-manual/R-devel/library/base/html/00Index.html

2. Analizaremos la documentación y con su ayuda utilizaremos unas simples estructuras de datos. Lo que escribamos en la ventana inferior izquierda debe seguir las reglas del lenguaje R, va a ser interpretado, y si corresponde a una sentencia u orden de R entonces se va a ejecutar. Además se va a guardar en la memoria de RStudio, por lo que tendrá constancia en la ventana superior derecha. En la ventana superior izquierda podemos mostrar el contenido de los ficheros, o incluso podemos generar uno nuevo vacio, y la aplicación nos permite escribir lo que sea en ese fichero. Además podremos guardar lo que hayamos escrito en la ventana en un fichero al que tendremos que dar algun nombre. Lo normal es escribir código en R y dar un nombre al fichero con extensión .R, por ejemplo, **mifichero.R** . Al final de la sesión de laboratorio lo que hayas escrito en esta ventana lo tendrás que subir a la plataforma de e-Gela.

   Empecemos a jugar con código de R:

   - Assignment Operators ($<-$, $->$, $=$),

     ```
     x<-2
     3-> y
     z=4
     ```

   - Arithmetic Operators (+, -, *, /, ^ , %%, %/%),

     ```
     x+y
     (x+y)-z
     ((x+y)-z)*z/x
     x^y
     Pregunta en la ventana de ayuda el significado del operador %%
     En la secci\'on de Value dice:
     %% indicates x mod y and %/% indicates integer division.
     (y+z) %% x
     (y+z) %/% x
     ```

   - Relational Operators ($<$, $>$, $<=$, $>=$, $==$, $!=$)

     ```
     x<y         x>y         x<=y
     x>=y        x==y        x!=y
     ```

   - Logical Operators (logical negation !, logical AND &, logical OR |),

     ```
     !(x<y)
     x<y
     y<x
     (x<y)&(y<x)
     (x<y)|(y<x)
     ```

   - Sequence Generation (c, colon ":", seq), length

     ```
     a<-c(1,2,3)
     b<-c(x,y,z)
     length(a)
     Bektoreko osagai bat hartzeko:
     a[1]
     1:10
     4:20
     x:z
     de 2 a 20, de dos en dos    seq(2,20,2)
     ```

- Print, Concatenate and print

```
print(5+5)
cat("tres mas tres (", y, "+", y, ") equivale a seis", y+y)
```

- Control flow (if, if else, while, for)

```
if (x<y) y-x
if (y<x) x-y else 0
while (x<z) {print(x); x<-x+1}
for (i in 1:10) {print(i)}
for (i in 1:10) {print("a")}
```

- Definición de una función que devuelve un valor (function, return)

```
devolver_longitud<-function(v)
{lon_de_vector=length(v);
 return(lon_de_vector);
}
v<-c(1,2)
devolver_longitud(v)
```

- Bestelakoak (is.numeric, as.numeric)

```
is.numeric(5)        is.numeric("5")
v=c("kuku", 1,2,3)
is.numeric(v[1])     is.numeric(v[2])
v[2]+v[3]
as.numeric(v[2])+as.numeric(v[3])
```

3. El siguiente código define una función llamada **suma** que suma los dos elementos del vector que recibe como parámetro y escribe el resultado de la suma.

```
suma<-function(v)
{v1<-v[1];
 v2<-v[2];
 cat("El vector es =(",v1,",",v2,"), ","La suma de sus dos elementos es=", v1+v2
}
```

Vamos a pedir que se ejecute la función, para ver si el eresultado es el esperado. Para ellos definimos un vector de dos componentes y realizamos la llamada a la función "suma":

```
v<-c(1,2)
length(v)
  ## [1] 2
suma(v)
  ## El vector es=( 1 , 2 ),  La suma de sus dos elementos es= 3
```

Basándote en el ejemplo "suma"inventa otras funciones que realicen diferentes operaciones aritméticas entre las componentes del vector y que escriban el resultado. Por ejemplo,

3.1 La suma de todoas las componentes de un vector con cualquier longitud..

```
suma2<-function(v)
{
}
```

Realiza la llamada a la nueva función con los parámetros que se indican a continuación y verifica que el resultado es el esperado.

```
## vectores con los que probar la funci\'on suma2
w1<-c(3,8,10,2,4,9)
w2<-c(3,8,10)
w3<-c(3,8,10,0,-1,9)
suma2(w1)
  ## [1] 36
suma2(w2)
  ## [1] 21
suma2(w3)
  ## [1] 29
```

3.2 Dado un vector, la primera componente del vector indicará si hay que sumar o si hay que multiplicar el resto de elementos del vector.

```
suma_o_multiplica<-function(v)
{
}
```

Realiza la llamada ltu nueva función con los siguiientes vectores y verifica que los resultados son los esperados.

```
v1=c("suma", 1,2,3,4,5,6)
suma_o_multiplica(v1)
  #[1] 21

v2=c("multiplica", 1,2,3,4,5,6)
suma_o_multiplica(v2)
  #[1] 720
```

¿Puedes mejorar la función? Si el primer elemento del vector no corresponde a "suma"ni a "multiplica"que de mensaje de error diciendo que no puede realizar la operación, y si no hay suficientes elementos en el vector que informe con otro mensaje:

```
v3=c(1,2,3,4,5,6)
suma_o_multiplica(v3)
# No se dice qu\'e hacer

v4=c("batu")
suma_o_multiplica(v4)
# No hay suficientes elementos en el vector.
```

3.3 La función que nos diga cuales son los dos valores más grandes entre los valores de un vector, independientemente de la longitud del vector, y adems que nos diga las posiciones que ocupan esos dos valores en el vector.

```
los_dos_mas_grandes<-function(v)
{
}
```

Realiza la llamada a la función con los siguientes parámetros y verifica que devuelve los resultados esperados.

```
v5<-c(3,8,10,2,4,9)
los_dos_mas_grandes(v5)
   # El mayor: 10 Posicion: 3 El segundo mas grande: 9 Posicion: 6
v6<-c(3,8)
los_dos_mas_grandes(v6)
   # El mayor: 8 Posicion: 2  El segundo mas grande: 3 Posicion: 1
v7<-c(8,3)
los_dos_mas_grandes(v7)
   # El mayor: 8 Posicion: 1  El segundo mas grande: 3 Posicion: 2
v8<-c(8,8)
los_dos_mas_grandes(v8)
   # El mayor: 8 Posicion: 1  El segundo mas grande: 8 Posicion: 2
v9<-c(8)
los_dos_mas_grandes(v9)
   # El vector ha de tener al menos dos valores
v10<-c(8,8,0,2,12,8,12,12)
los_dos_mas_grandes(v10)
   # El mayor: 12 Posicion: 5  El segundo mas grande: 12 Posicion: 7
```