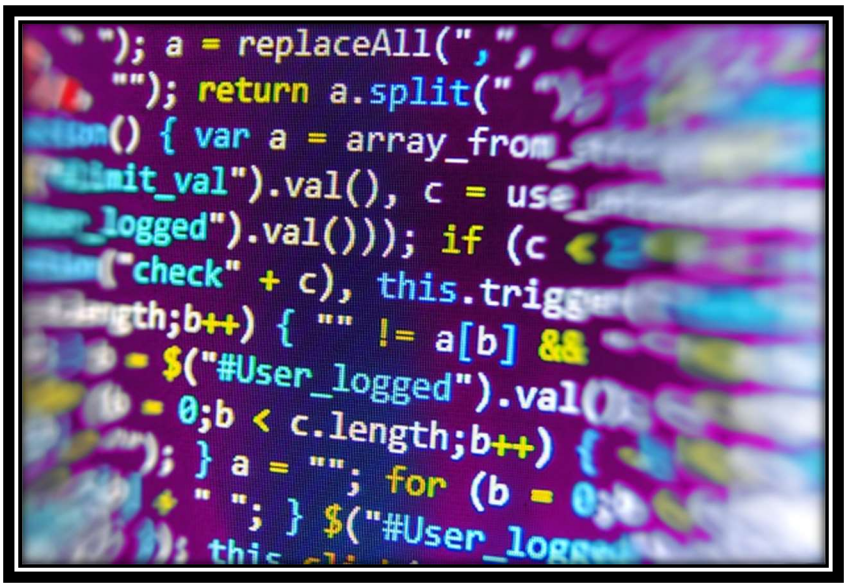
COMANDOS Y HERRAMIENTAS ÚTILES DE R

NIVEL DIFÍCIL



https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwiE4fbRlrjmAhVHzYUKHY77BpwQjRx6BAgBEAQ&url=https%3A%2F%2Fblog.dinahosting.com%2Flos-10-lenguajes-de-programacion-mas-usados%2F&psig=AOvVaw2v5c2bUPAXjmUvKZY5oiSL&ust=1576517247162169

Este documento ha sido realizado y revisado por alumnos de primero del grado de Biotecnología de la Universidad Politécnica de Madrid. Se basa en los contenidos de la asignatura de Fundamentos de Programación, impartida por Arturo Hidalgo y Ángel Fidalgo.

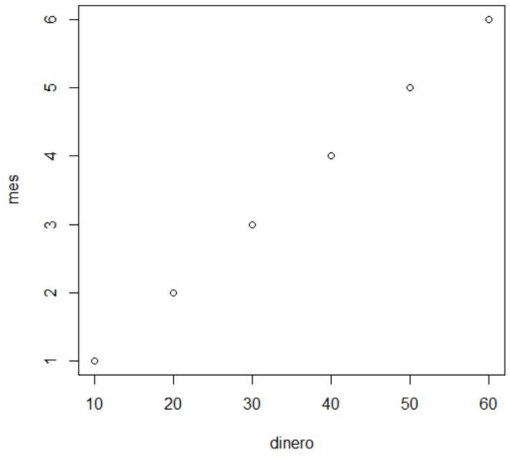
El lenguaje de programación utilizado es R, enfocado en el análisis estadístico y muy utilizado en investigación científica y biomédica, bioinformática, aprendizaje automático (machine learning), minería de datos, y matemáticas financieras.

El documento está formado por una explicación de los comandos y herramientas más útiles de R, con imágenes que los ejemplifican. Por lo que su finalidad es la de servir como referencia a próximos alumnos y a personas con interés con los campos citados, facilitando el aprendizaje de este lenguaje.

¿Cómo creamos gráficas?

Para ello, tenemos que utilizar el comando ***plot().*** Dentro incluiremos los dos vectores que queremos que se representen en la gráfica.

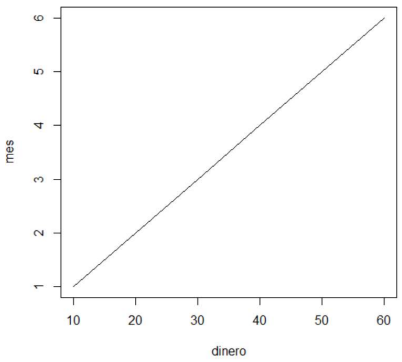
```
> dinero<-c(10,20,30,40,50,60)
> mes<-c(1,2,3,4,5,6)
> plot(dinero,mes)
```



En R, tenemos la opción de elegir el tipo de gráfica que queremos. Para ello, utilizamos el comando ***type=" "*** en el que dentro podremos incluir:

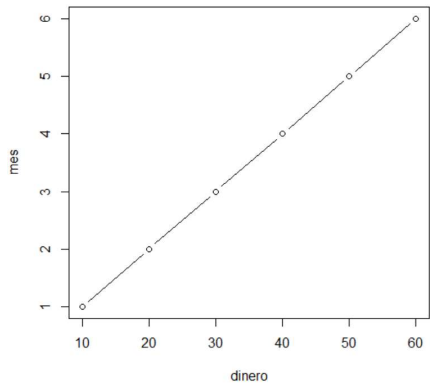
- ***l*** si queremos unir con una línea los puntos sin que estos mismo aparezcan.

```
> dinero<-c(10,20,30,40,50,60)
> mes<-c(1,2,3,4,5,6)
> plot(dinero,mes, type="l")
```



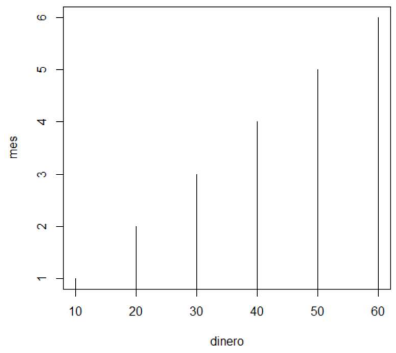
- ***b*** si queremos unir con una línea los puntos incluyendo estos últimos.

```
> dinero<-c(10,20,30,40,50,60)
> mes<-c(1,2,3,4,5,6)
> plot(dinero,mes, type="b")
```



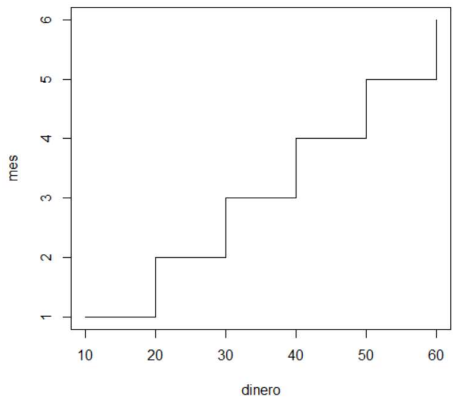
- ***h*** si queremos que tenga forma de histograma.

```
> dinero<-c(10,20,30,40,50,60)
> mes<-c(1,2,3,4,5,6)
> plot(dinero,mes, type="h")
```



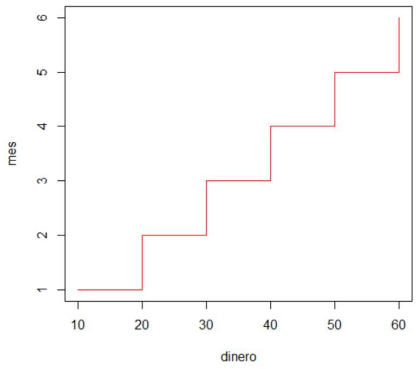
- ***s*** si queremos que aparezca tipo escalera.

```
> dinero<-c(10,20,30,40,50,60)
> mes<-c(1,2,3,4,5,6)
> plot(dinero,mes, type="s")
```



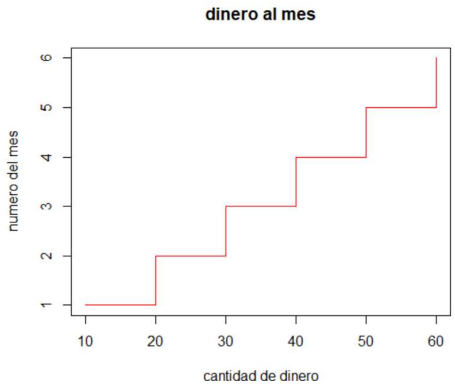
Podemos también hacer que este gráfico aparezca en un color determinado. Para ello, utilizaremos ***col=" "*** con el nombre del color que queremos en inglés.

```
dinero<-c(10,20,30,40,50,60)
mes<-c(1,2,3,4,5,6)
plot(dinero,mes, type="s", col="red")
```



Si queremos añadir un título diferente a los ejes, utilizamos ***xlab=" "*** e ***ylab=" ",*** a los ejes de las X y el de las Y respectivamente. Dentro incluiremos el nombre que deseamos que tengan. En el caso de querer dar un título al eje, utilizamos ***main=" "***, incluyendo dentro el título.

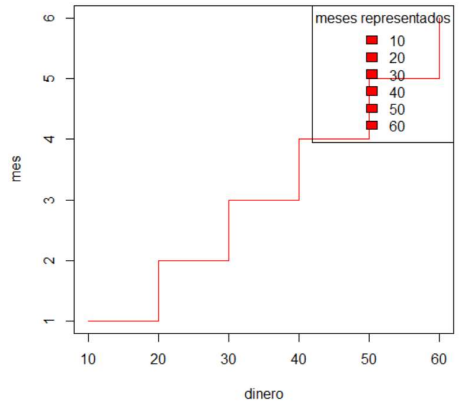
```
dinero<-c(10,20,30,40,50,60)
mes<-c(1,2,3,4,5,6)
plot(dinero,mes, type="s", col="red", xlab="cantidad de dinero", ylab="numero del mes", main="dinero al mes")
```



Podemos añadir una leyenda a nuestro gráfico con el comando ***legend( )*** en el que podremos indicar con los siguientes comandos:

- ***x=n, y=n***: siendo n las coordenadas donde se encuentra la leyenda. También, al comando ***x=" "***, podemos añadirle algunos de los siguientes datos para colocar directamente la leyenda: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center".
- ***legend=n***: siendo n las etiquetas de los datos que queremos describir con la leyenda.
- ***fill="color"***: los colores que acompañan a las etiquetas. (Estos colores tienen que coincidir con los que hemos usado en el gráfico).
- ***title=" "***: para poner un título.

```
legend(x="topright", legend=dinero, fill="red", title="meses representados")
```



¿Cómo creamos una función?

Para ello, deberemos seguir la siguiente estructura:

mifuncion = function(argumento1, argumento2, …) {
cuerpo
}

Donde mi "***mifunción"*** es el nombre de la función que queremos darle; ***function*** es el comando necesario; ***argumento1, argumento2, …*** son los valores que necesitamos para que se ejecute la función; ***cuerpo*** corresponde a las operaciones que deseamos que se ejecuten sobre cada uno de los argumentos detallados anteriormente, que vienen dados entre dos corchetes y se ejecutan cada vez que llamamos la función.

```
> resta=function(x,y){
+ x-y
+ }
> resta(5,2)
[1] 3
```

IMPORTANTE hay que tener cuidado con el orden de los valores que le damos el argumento, ya que R respeta dicho orden, es decir, en nuestro ejemplo R va a dar el valor de 5 a la x y el valor de 2 a la y.

Bucle If/Else

Este bucle permite decidir si ejecutar o no un fragmento de código en función de una condición. Para ello, seguiremos la siguiente estructura:

if (condición 1) {
operación 1
} else if (condición 2) {
operación 2
} else {
operación 3
}

Donde la ***condición*** corresponde a los que queramos que se cumpla y la ***operación*** lo que el programa debe hacer cada vez que la condición se cumple. (*NOTA: podemos añadir tantas condiciones como queramos)*

```
> x<-5
> if (x>0){
+ "el número es positivo"
+ } else if (x<0) {
+ "el número es negativo"
+ } else {
+ "el número es nulo"
+ }
[1] "el número es positivo"
```

Bucle For

Los bucles for son los más utilizados en R. Este bucle repite una acción un número determinado de veces. Tiene la siguiente estructura:

for (i in v.i : v. f) {

Proceso

}

Donde ***v.i*** corresponde al valor donde queremos que empiece el bucle y ***v.f*** es el valor final, donde queremos que termine de ejecutarse el bucle. Y el ***Proceso*** es la operación o la acción que debe hacer el programa mientras esté funcionando el bucle.

Se pueden escribir bucles anidados, pero es muy importante que cerremos todos los bucles con } porque si no no se ejecutará la acción y dará error.

for (i in v.i : v. f) {

for (j in v.i : v.f) {

for (k in v.i in : v.f) {

Proceso

}

}

}

```
> v=c(1,4,5)
> u=c(2,6,7)
> n=length (u)
> S=0
> for (i in 1:n){
+ S[i]=v[i]+u[i]
+ }
> S
[1]  3 10 12
```

Bucle While

Los bucles while comienzan comprobando una condición, si es verdadera se ejecuta la acción de dentro del while. Se ejecuta hasta que el la condición se falsa

while(Condición){

Proceso

}

Donde ***Condición*** es lo que tiene que ser verdadero para que se realice el bucle y ***proceso*** es la acción que queremos que se realice. Es importante como en el bucle for cerrar todas las llaves. Además es importante que inicialicemos el valor por el que se inicia el bucle(i=1) y ponerle que varíe(i=i+1) porque si no el bucle no acaba nunca ya que la i siempre tomaría el mismo valor.

```
> s=0
> while(s<10){
+ print(s)
+ s=s+1
+ }
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
>
```