Apuntes de R, gráficos

La representación gráfica es la parte más visual de R, y en la que veremos el resultado gráfico de nuestro código.

Recomiendo el uso de R clásico, debido a que los gráficos están mejor optimizados en esta versión de R

Recordemos que el comando data.frame almacena datos de distinto tipo (vectores) y los almacena en una matriz a forma de tabla. De manera análoga si usamos el comando **plot(**vector1, vector2) por ejemplo, crearemos un gráfico muy simple de estos vectores.

-El vector 1 estará en el eje de las x

-El vector 2 estará en el de las y

Por ejemplo, la representación de las ganancias anuales de un trabajador autónomo que ha logrado reinventarse tras la pandemia.

Ganancias=c(1900,1950,1500,2300,2500)

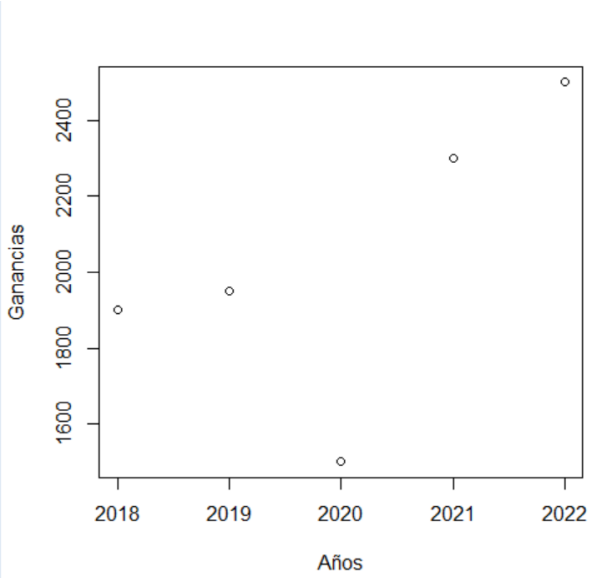
Años=c(2018,2019,2020,2021,2022)

Si un vector es una sucesión simple como números del 1 al 10 podemos usar

Vector=seq(1,10,1)

plot(Años,Ganancias)

x y



Perfecto, ahora podemos mejorar nuestro gráfico añadiéndole mejoras al plot.

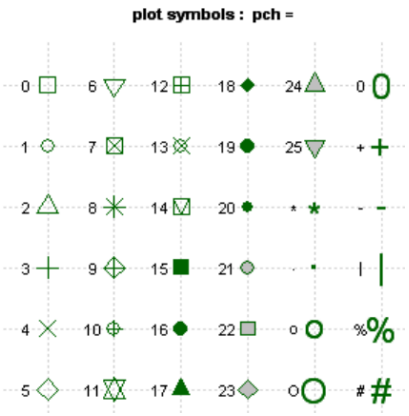
- Si añadimos el comando type, podremos cambiar la estética de la gráfica
  - type= “h” cambiará los puntos por líneas verticales hasta su valor en y
  - type=”l” cambiará los puntos por líneas
  - type=”b” dejará los puntos y los unirá con líneas
- col=”el color que queramos”. Este comando cambiará el color, por ejemplo col=”blue”
- xlab: sirve para nombrar el eje x como queramos, si no queremos texto, ponemos xlab=”” (útil al unir dos gráficos). Ponemos por ejemplo xlab=”años laborales”
- ylab: nombra el eje y. ylab=”ganancias cada año”

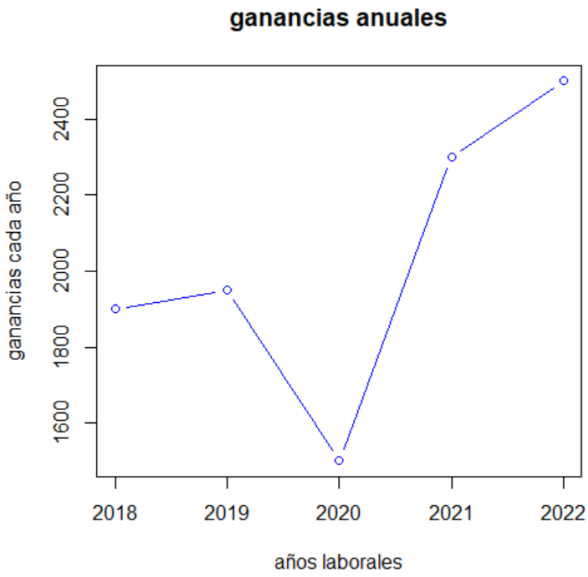
- main: es útil para nombrar la gráfica. Pondremos en este caso main=”ganancias anuales”

El resultado final de nuestro gráfico será este

plot(Años,Ganancias,type="b",col="blue",xlab="años laborales",ylab="ganancias cada año",main="ganancias anuales")

Utilizando el comando pch, podremos cambiar los puntos por otras figuras, las posibilidades son las siguientes.





Esto sería más que suficiente para una representación simple, como extra existe el comando grid, que añade una cuadrícula a nuestro gráfico, por ejemplo: grid(nx=NULL,ny=NULL,lty=12,col="gray",lwd=2) El grosor de la cuadrícula se determina con lwd.

Es de suma importancia añadir el comando par(new=TRUE) antes del grid. Este comando es el encargado de fusionar dos gráficas, en este caso la cuadrícula.

R también nos permite **representar funciones**, para ello antes tenemos que definir nuestra función, por ejemplo, un movimiento armónico simple.

```
 f=function(x){

4*cos(x)

}
```

Además, tendremos que definir el soporte sobre el que queremos nuestra función, por ejemplo

```
x=seq(-pi,pi, length = 40)
```

```
plot(x,f(x),type="l",ylim=c(-4,4),col="green",xlab="tiempo",ylab="elongación")
```

El comando ylim=c(-5,5) nos fija un límite en el eje de las y, el xlim hace lo mismo para las x, esto será muy útil a la hora de representar dos funciones unidas. También es común ver variables en los x e y lim para fijar los límites.

Ahora quiero representar también la velocidad y que se vea en la misma gráfica.

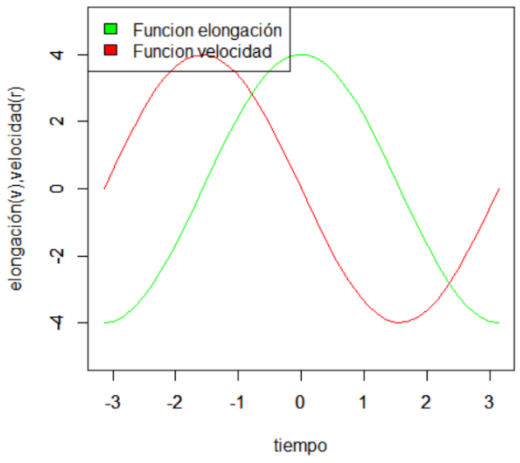
Conviene al principio del programa poner todas las funciones, por claridad repetiré todo:

```
f=function(x){

4*cos(x)

}

g=function(x){

-4*sin(x)

}

x=seq(-pi,pi, length = 40)

plot(x,f(x),type="l",ylim=c(-5,5),col="green",xlab="tiempo",ylab="elongación(v),velocidad(r)")

par(new="true")

plot(x,g(x),type="l",ylim=c(-5,5),col="red",xlab="",ylab="")

legend(x="topleft",c("Funcion elongación","Funcion velocidad"),fill=c("green","red"))
```

El comando par vale para unificar ambas funciones, en el segundo plot conviene borrar los nombres de los ejes para que no se superpongan, cambiarle el color para que se diferencien y añadir el mismo ylim, para que se representen correctamente. Podemos meter en un gráfico más de dos funciones, las que queramos usando el comando par entre los plots.

Con el comando legend, he añadido una leyenda, x define donde estará, dentro del primer vector están las funciones que queremos representar, y el segundo vector vale para indicar el color de cada uno.

El resultado final es:



Como sabemos, cuando la elongación es máxima (la amplitud), la velocidad es 0, cuando la elongación pasa por el origen, la velocidad es máxima.

Para tener ya la maestría en gráficas, existe una modificación en el comando par, si añadimos entre paréntesis (mfrow=c(nºfilas,nºcolumnas)), lo que hará es distribuir los gráficos en tantas

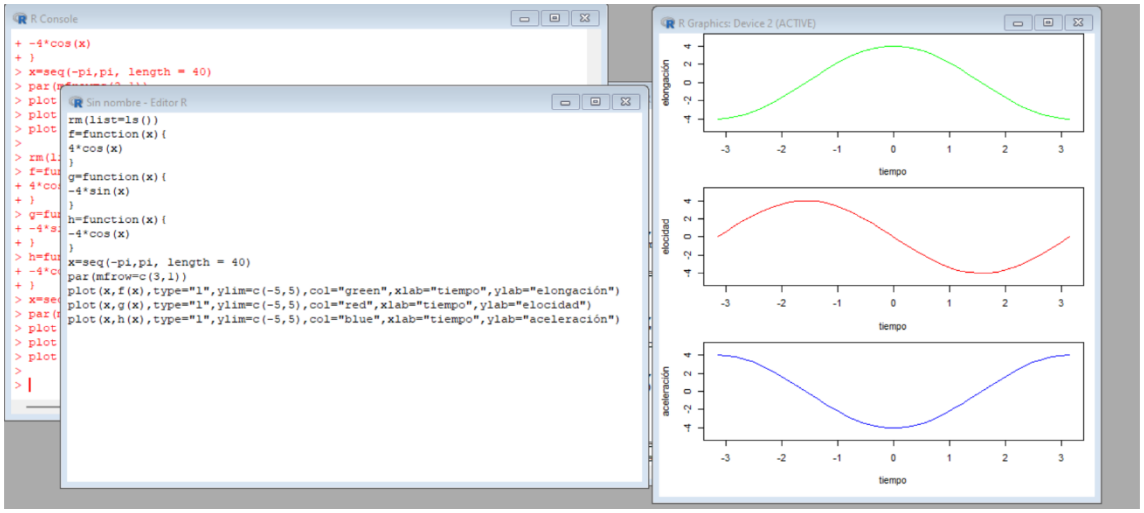
filas y columnas hayamos puesto. Este comando irá antes de los plots. En el ejemplo anterior, si añadimos también la aceleración con la función:

h=function(x){

-4*cos(x)

}

Y antes de los 3 plots añadimos par(mfrow=c(3,1)), nos quedará:



Aquí los plots son independientes y podremos modificarlos a nuestro gusto. En el comando par(mfrow=c(3,1)), si queremos podemos añadir márgenes con el mar=c(separación margen inferior, separación margen izquierda, separación margen superior, separación margen derecha) Todo indicado con números. Por ejemplo: par(mfrow=c(3,1),mar=c(1,1,1,1))

R también nos permite la opción de añadir **gráficos circulares**, estos son muy útiles y se añaden con el comando

pie(vector, clockwise=TRUE,col=c(colores, todos entre comillas y separados con comas),cex=0.8)

cex lo que hará es establecer el tamaño de letra, clockwise establece el sentido en el que añadirá los datos.

Por ejemplo, haré un gráfico circular de lo que hago un martes cualquiera

#Un día martes

rm(list=ls())

Tact=c(9,2.5,0.75,6,2,0.75,3) #Esto es el tiempo invertido en las actividades

names(Tact)=c("dormir","hobbies","comer","clases","judo","cenar","estudiar")

#con esto nombramos cada valor del vector Tact

pie(Tact, clockwise=TRUE,col=c('red', 'blue', 'green', 'purple', 'orange', 'yellow',

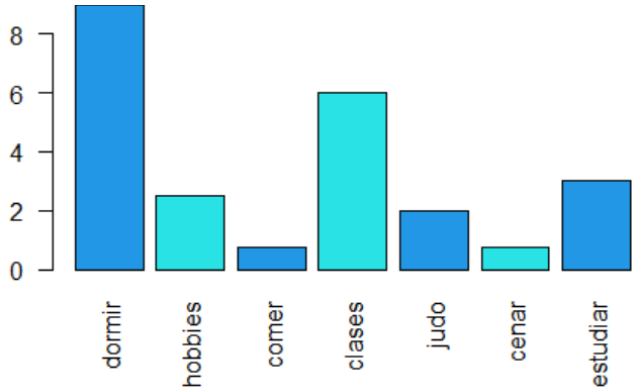
'pink'),cex=0.8)

#El resultado final es:

También existe la posibilidad de añadir **gráficos de barras**, su comando será:

barplot(Tact, col=c(4,5), las=2)

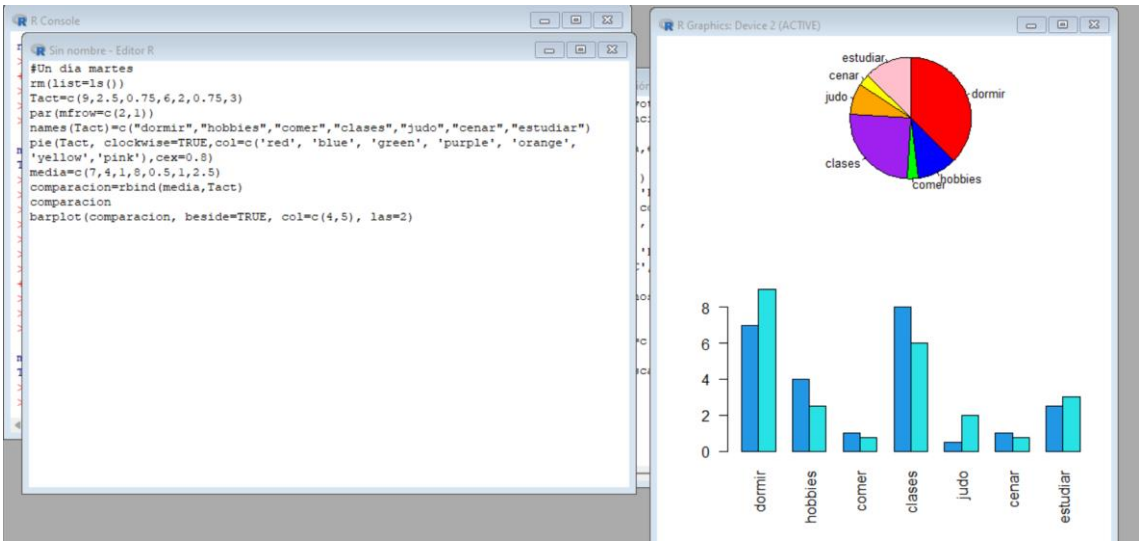
En el ejemplo anterior quedaría así:



Podremos comparar este gráfico con otro usando el mismo comando, pero de esta forma barplot(vector, beside=TRUE, col=c(4,5), las=2). Además añadiré el gráfico circular anterior en la parte superior, usando el par(mfrow=c(2,1))

Hay algunas diferencias, por ejemplo, el vector tiene que estar en forma de matriz usando el rbind. Beside=TRUE indica que queremos comparar los parámetros, no que estén en la misma línea, las=2 nos dice que queremos que los nombres de los parámetros estén abajo del gráfico de forma vertical.

Esta vez compararé mi martes con el de la media(datos elegidos arbitrariamente, no veraces)



Con estos apuntes tendrás destreza más que suficiente para manejar gráficos en R, te serán de bastante utilidad en las prácticas, ¡espero que hayan sido de utilidad!