concordance=TRUE

Métodos estadísticos
Introducción al R

Daniel Grados
danny.grados@gmail.com

05-10/07/2019

Introducción al ambiente R: Motivación

- Grandes fuentes de almacenamiento de datos.
- Necesidad de manejar grandes fuentes de datos.
- Analizar datos permite encontrar patrones ocultos.
- Gran facilidad para realizar análisis científicos.

El proyecto R para manipulación de datos

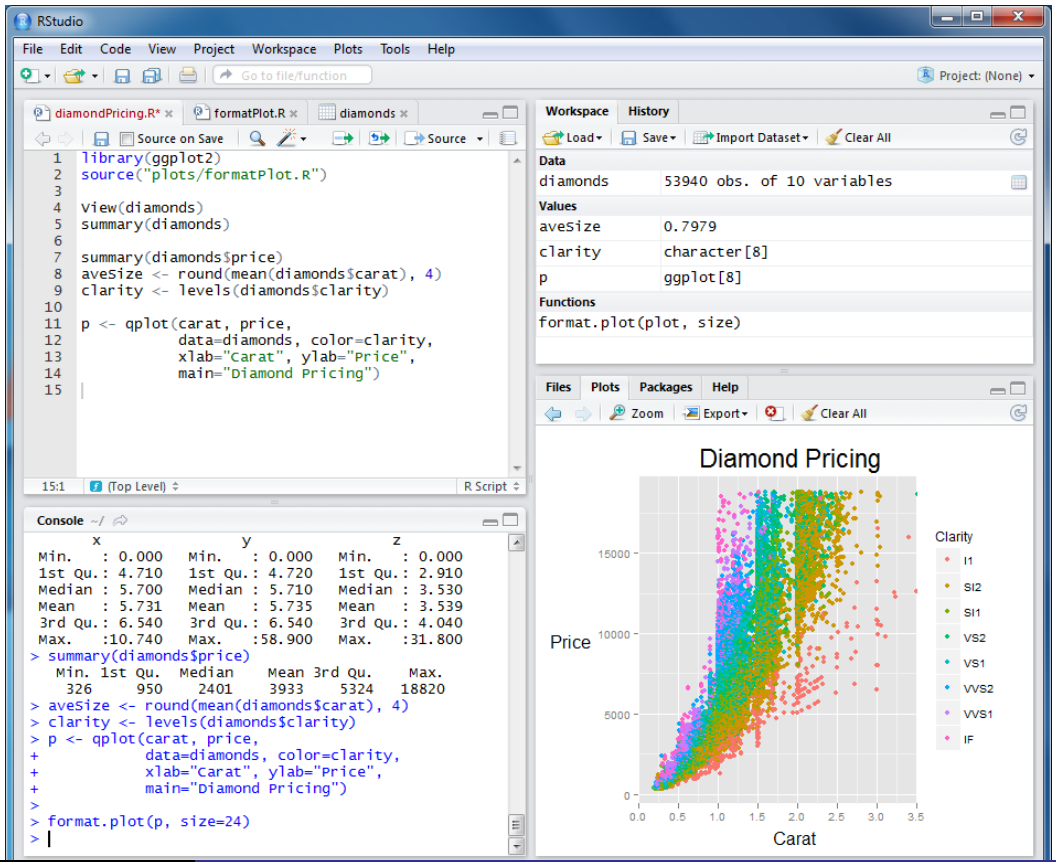
- R es libre y de código abierto http://ww.r-project.org/
- Permite manipular y almacenar datos de manera efectiva.
- Es un lenguaje de programación completo: variables, loop, condiciones, funciones.
- Provee muchas librerías para realizar distintos tipos de análisis de datos (**CRAN**: http://cran.r-project.org/)

¿Por qué usar R?

- R está orientado al análisis de datos (estadística).
- R es un software libre y de código abierto (a diferencia de otros programas como SPSS o SAS).
- Está disponible para muchos sistemas operativos.

Rstudio

- R funciona a través de líneas de comandos.
- Para trabajar en un entorno más amigable usaremos RStudio.
- RStudio utiliza diferentes colores para funciones, sentencias, etc.



Estructura del curso

- Introducción.
- Tipos de objetos utilizados en R.
- Importación de datos en R.
- Estadística descriptiva.
- Gráficos en R.
- Introducción a la programación en R.

Usando R como calculadora

R puede usarse para las funciones mas básicas como una calculadora.

```
1 + 2
```

```
## [1] 3
```

```
2*3
```

```
## [1] 6
```

```
2^3
```

```
## [1] 8
```

```
exp(-5)
```

```
## [1] 0.006737947
```

```
log(4)
```

```
## [1] 1.386294
```

```
pi
```

```
## [1] 3.141593
```

Funciones aritméticas en R

Función	
abs	Valor absoluto
sqrt	Raiz cuadrada
sin cos tan	Funciones trigonométricas
asin acos atan	Funciones trigonométricas inversas
sinh cosh tanh	Funciones hiperbólicas
exp log	Exponencial y logarítmo natural
log10	Logarítmo en base 10

Ejerc. Usando las funciones en R Calcular:

1) $\sqrt{100} + log_{10}(100)$
2) $\sqrt{(seno(45 * \pi/180))}$

Declarando variables en R

- Las variables se pueden declarar usando <-, = o la función assing

```
a <- 2
b = 3
c <- 'm'
```

- Por convención se usa mayormente <-
- Las variables pueden ser de clase **numeric, factor, character, logical**
- Para saber que tipo de objeto tenemos usamos el comando class

```
c <- c('A','B'); class(c)

## [1] "character"
```

Ayuda y el workspace

- Para tener ayuda de una función usamos **help()** o **?**.

```
help(ls)
?ls
help('for')
```

- Todas las variables quedan en el ambiente **workspace**. Para listarlos se usa el comando **objects()** o **ls()**. Para borrar una variable usamos **rm()**.

```
ls()

## [1] "a" "b" "c"

rm()
# Para borrar todos los archivos rm(list=ls()))
```

Ayuda y el workspace

- Para grabar todas las variables del workspace en un archivo y así recuperar lo trabajado en una sesión futura:

```
save.image("myworkspace.RData")
# Para cargar lo guardado anteriormente
load("myworkspace.RData")
```

Programación orientada a objetos

Programación Orientada a objetos.

Programación orientada a objetos

R es un lenguaje orientada a objetos: Variables, datos, vectores, matrices, funciones, etc son almacenados en la memoria activa del computador en forma de objetos.

Tipos de objetos en R

Los tipos básicos de objetos en R son:

- Vectores
- Matrices
- Data-frame
- Listas
- Funciones

Tipo de objetos en R

Los tipos básicos de objetos en R son:

Vector

```
x1 <- c(1,3,6)
class(x1)
is.vector(x1); is.matrix(x1); is.numeric(x1)
x2 <- seq(from=0,to=10,by=1)
# ejer. Mostrar el segundo elemento de x1
```

Tipo de objetos en R

Matriz: Los elementos son organizados en dos dimensiones (filas y columnas).

```
matrix(data = NA, nrow = 1, ncol = 1 , ...)
```

```r
M1 <- matrix(1:12,ncol = 3)
M2 <- matrix(NaN,ncol = 2, nrow = 3)
# Ejer.
# Cuales son las dimensiones de las dos matrices?.
# Mostrar el elemento en fila 2, columna 2 de M1.
# Calcular M1 + M2.
```

Tipo de objetos en R

Array: El objeto *array* generaliza la idea de una matrix. En un *array* los elementos pueden ser organizados en un número arbitrario de dimensiones.

```
array(data = NA, dim = length(data), dimnames = NULL)
```

```
# Ver la ayuda de array()
A1 <- array(1:24 , dim = c(3,4,2))
# Ejer. estudiar los resultados de las siguientes
# sentencias.
# A1[,2:3,]; A1[2,,1]; sum(A1[,,1]).
# Cargar la data "Titanic" de R y estudie sus
# caracteristicas.
```

Tipo de objetos en R

Data-frame

```
# Ver la ayuda de data.frame()
d1 <- data.frame(X = 1:5, Y = c(51,54,61,67,68))
names(d1)
```

Listas Son estructuras genéricas y flexibles que permiten almacenar diversos formatos en un único objeto.

```
lis1 <- list(A=1:10, B="Mensaje", C=matrix(1:9, ncol=3))
```

Entrada de datos en R

Importación / Exportación de datos

Entrada de datos en R

R permite ingresar datos de diferentes formas.

Entrada de datos directamente en R - Definiendo vectores

```
a1 <- c(2,5,8)
a2 <- c(23,25,41,29,48)
# Crear un vector de secuencia de 1 a 100
# Crear un vector repitiendo 3 veces 5 y 8
# alternadamente
```

Usando la función scan

```
y <- scan()
# Para terminar esta funcion presionar dos veces
# consecutivas enter
y
```

Entrada de datos en R

Usando la función edit:
el comando **edit(data.frame())** abre una plantilla para digitalizar los datos a ingresar. Los datos digitalizados son almacenados en un *data.frame*.

```
a3 <- edit(data.frame())
class(a3)
names(a3)
dim(a3)
```

Entrada de datos en R

Leyendo datos desde un archivo existente:
Si los datos ya estan disponibles en formato electrónico, o digitados en otro programa se puede importar los datos a R sin necesidad de digitarlos nuevamente.

```
read.table(file, header = F , sep = '', dec = '.' ...)
```

Ejerc. Leer los datos dados como ejemplo. Verificar que tipo de variables estan contenidas en ese conjunto de datos.

Exportar datos desde R

R permite exportar datos en diferentes formatos.
Uno de los mas usados es mediante la función ***write.table()***.

```
write.table(x,file = "", sep = " ",na = "NA", dec = ".",
            row.names = TRUE,col.names = TRUE)
```

Otras funciones para importar/exportar datos.

Función	Tipo de archivo
read.csv / write.csv	Archivo con extensión csv

Archivos con formato excel?

varios paquetes, xlsReadWrite

Análisis descriptivo en R

Análisis descriptivo: Permite describir apropiadamente las características de un conjunto de datos.

Análisis descriptivo en R

Vamos a ver algunas funciones en R para realizar un análisis descriptivo de un conjunto de variables.
Las variables pueden clasificarse como:

- **Cualitativas**: Nominales y ordinales.
- **Cuantitativas**: Discretas y continuas.

Para explicar mejor el análisis descriptivo vamos a usar bases de datos disponibles en R (*mtcars* y *airquality*).
Para mayor información sobre estas bases de datos usar help(mtcars) ó help(airquality).

Análisis descriptivo en R

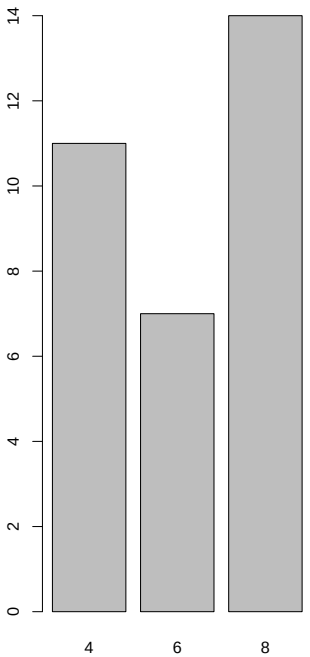
Usaremos la primera base de datos **mtcars** y estudiaremos a una variable cualitativa (cyl: número de cilindros).

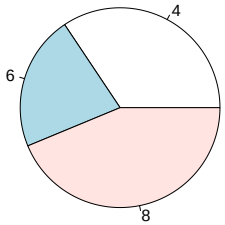
```
data(mtcars)
names(mtcars)
## "mpg" "cyl" "disp" "hp" "drat" "wt"
## "qsec" "vs"   "am"   "gear" "carb"
attach(mtcars) # Fija la base a trabajar
tcyl <- table(cyl) # Numero de cilindros
tcyl
## cyl
## 4  6  8
## 11 7  14
```

Análisis descriptivo en R

Analizando gráficamente las variables

```
par(mfrow=c(1,2))
barplot(tcyl)
pie(tcyl)
```





Análisis descriptivo en R

```
# Frecuencia relativa
prop.table(tcyl)

## cyl
##       4       6       8
## 0.34375 0.21875 0.43750
```

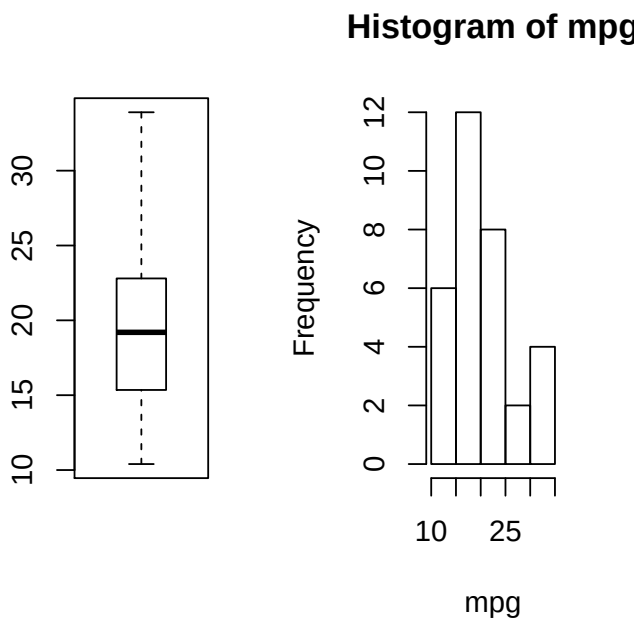
Análisis descriptivo en R

Ahora estudiaremos una variable cuantitativa (mpg: Millas/galon)

```
table(cut(mpg, br=seq(10,35, 5)))

##
## (10,15] (15,20] (20,25] (25,30] (30,35]
##       6      12       8       2       4

summary(mpg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90

# stem(mpg), genera un diagrama de hojas y tallos
# calcular la media, varianza, percentiles
```

Análisis descriptivo en R

```
par(mfrow = c(1,2))
boxplot(mpg)
hist(mpg)
```



Análisis bivariado en R

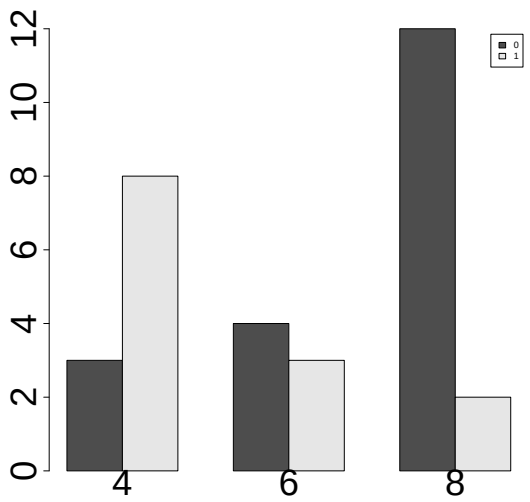
Análisis bivariado entre dos variables cualitativas (am: Transmisión, cyl: Número de cilindros)

```
table(am,cyl)
prop.table(table(am, cyl), margin=1)
prop.table(table(am, cyl), margin=2)
```

Análisis bivariado en R

Frecuencia de transmisión (am) por cilindro (cyl).

```
barplot(table(am, cyl), beside=T, leg=T , cex.axis = 4
        , cex.names = 4)
```

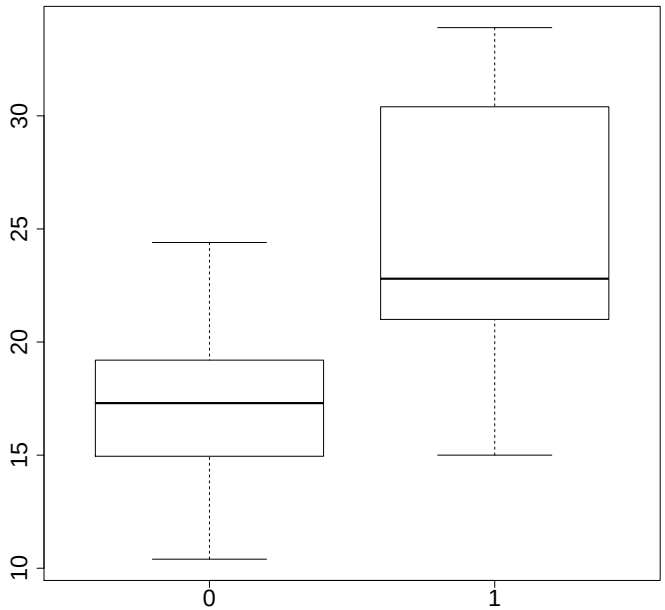



```
# Describir los resultados de:
# barplot(table(am, cyl), leg=T),
```

Análisis bivariado en R

Describiendo el consumo promedio de combustible por tipo de transmisión.

```
m0 <- mean(mpg[am==0])
m1 <- mean(mpg[am==1])
boxplot(mpg~am,cex.axis = 2)
```



Análisis bivariado en R

Ejerc. - Hacer histogramas para el rendimiento por galon (mpg) para cada tipo de tránsmision (variable am)
- ¿Existe diferencia significativa entre el rendimiento por galon (mpg) entre cada tipo de trásmision?

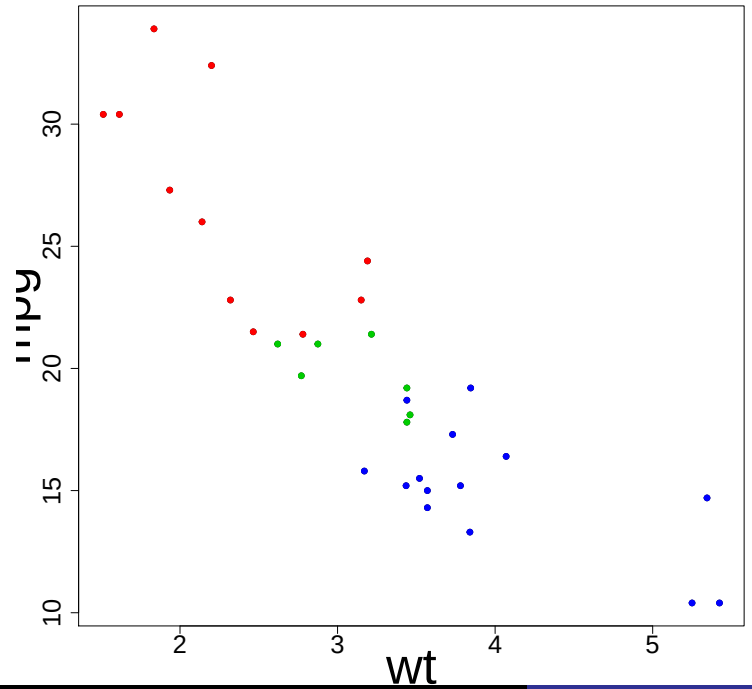
Análisis bivariado en R

Ahora podemos estudiar los rendimientos del carro a diferentes numero de cilindros

```
t.test(mpg[am==0], mpg[am==1])
## Welch Two Sample t-test
## data: mpg[am == 0] and mpg[am == 1]
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means
## is not equal to 0
```

Análisis bivariado entre dos variables continuas

```
cor(wt, mpg)
plot(wt, mpg,cex.axis=2,cex.lab=4)# Rendimiento vs peso
points(wt[cyl==4], mpg[cyl==4], col=2, pch=19)
points(wt[cyl==6], mpg[cyl==6], col=3, pch=19)
points(wt[cyl==8], mpg[cyl==8], col=4, pch=19)
```



Aplicando funciones sobre objetos array

Algunas funciones propias de R que nos permiten reducir tiempo en el cálculo

apply

```
apply(X, MARGIN, FUN, ...)
X: Array
MARGIN: Filas o Columnas
FUN: Funcion a evaluar
```

```r
# Ejemplo
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
apply(x, 2, mean)

## x1 x2
##  3  3
```

Aplicando funciones sobre objetos

tapply

```
tapply(X, INDEX, FUN = NULL, ...)
X: Vector
INDEX: Lista de uno o mas factores
FUN: Funcion a evaluar
```

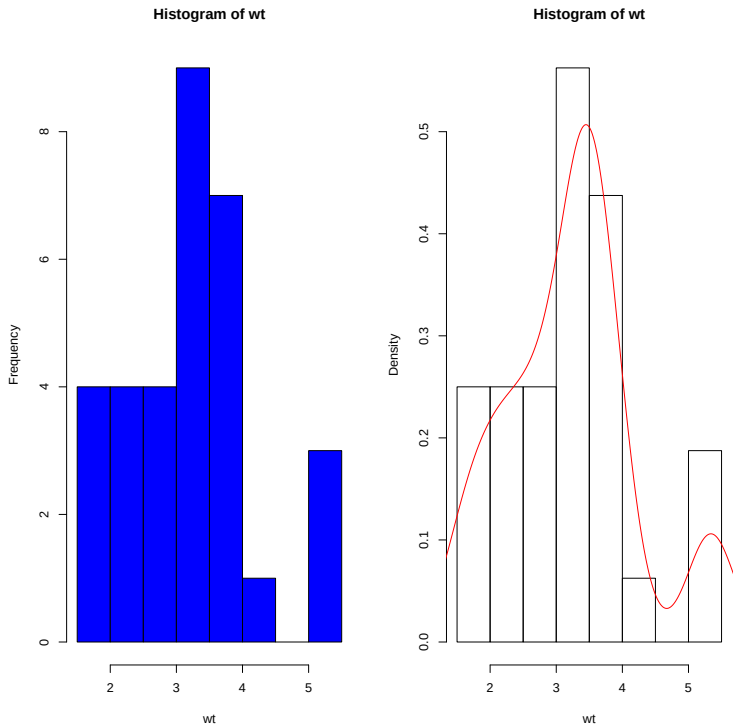
```
# Ejemplo
tapply(mpg,am,mean)

##        0        1
## 17.14737 24.39231
```

Gráficos en R

Gráficos nos permiten comprender nuestros datos. Pueden ser gráficos univariados, bivariados, como tambien multivariados.

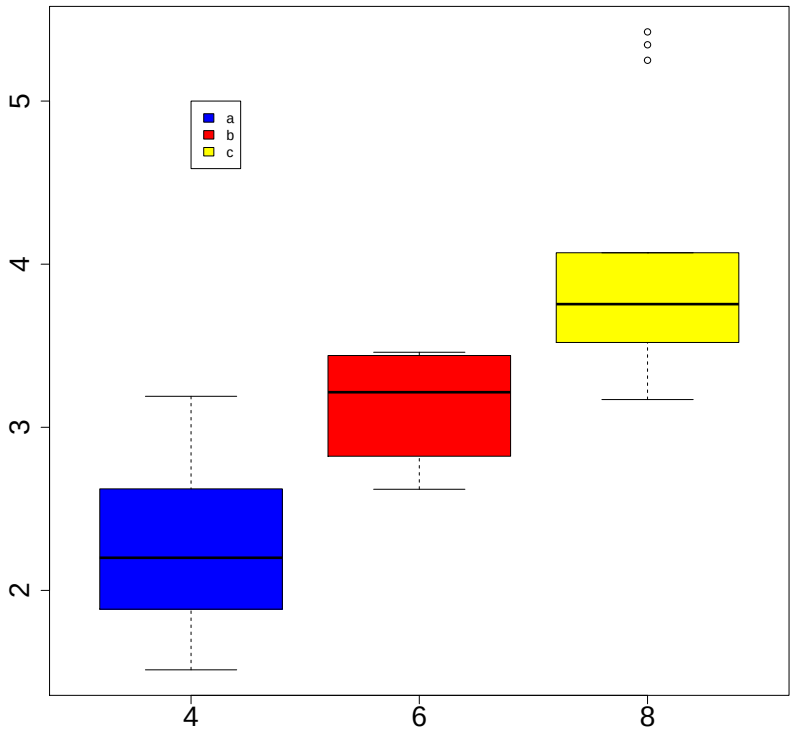
Gráficos en R

```
par(mfrow=c(1,2))
hist(wt,col='blue')# Grafico rendimiento vs peso
hist(wt,freq = FALSE)
lines(density(wt),col='red')
```



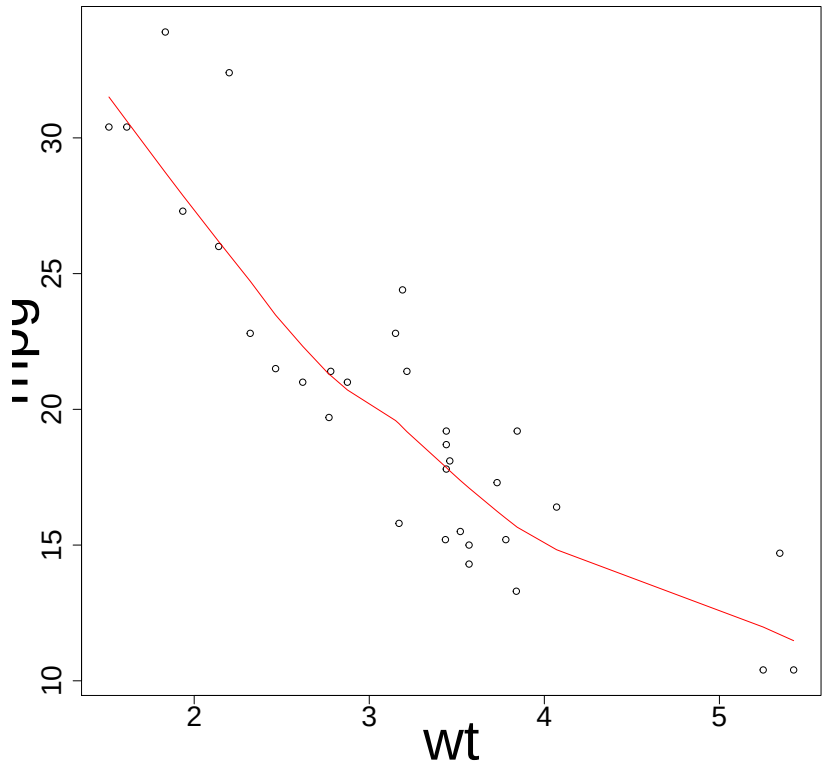
Gráficos en R

```
boxplot(wt~cyl,cex.axis=2,col=c('blue','red','yellow'))
# Grafico rendimiento vs peso
legend(5,c('a','b','c'),fill = c('blue','red','yellow'))
```



Gráficos en R

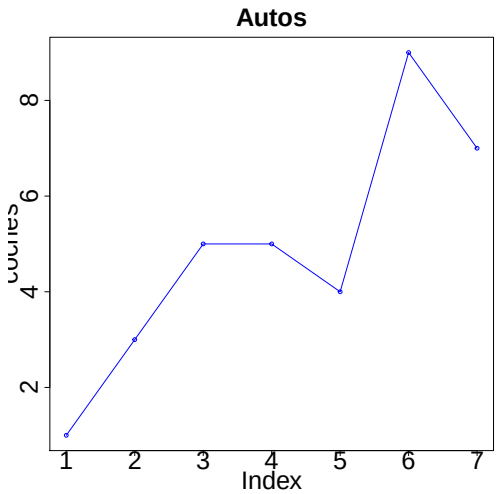
```
plot(wt, mpg,cex.axis=2,cex.lab=4)# Rendimiento vs peso
lines(lowess(wt,mpg),col='red')
```



Gráficos en R

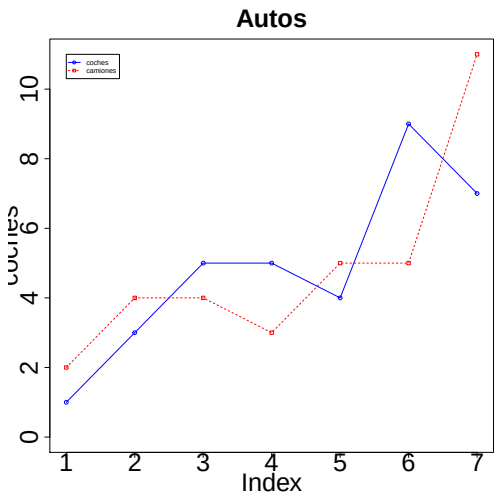
Generando un ejemplo

```
coches <- c(1, 3, 5, 5, 4, 9, 7)
camiones <- c(2, 4, 4, 3, 5, 5, 11)
motos <- c(1, 3, 3, 4, 3, 9, 14)
plot(coches,type="o",col="blue",main='Autos',cex.lab=3,
     cex.axis=3, cex.main=3, cex.sub=3)
```



Gráficos en R

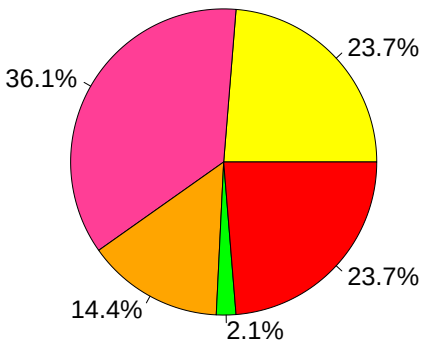
```
plot(coches, type="o", col="blue", ylim=c(0, 11),
 main='Autos',cex.lab=3,cex.axis=3,cex.main=3,cex.sub=3)
lines(camiones, type="o",pch=22,lty=2,col="red")
rango <- range(0, coches, camiones)
legend(1,rango[2],c("coches","camiones"),cex=0.8,
  col=c("blue","red"),pch=21:22,lty=1:2)
```



¿Que hace la función *title*?

Gráficos en R

```
pie.fruta <- c(0.23, 0.35, 0.14, 0.02, 0.23)
names(pie.fruta) <- c("Platanos", "Cerezas", "Naranjas",
                      "Manzanas", "Sandias")
fruta_labels <- round(pie.fruta/sum(pie.fruta)*100, 1)
fruta_labels <- paste(fruta_labels, "%", sep="")
pie(pie.fruta,col=c("yellow","violetred1","orange",
                    "green","red"),
labels=fruta_labels,cex=3)
```



Programación en R

Funciones:

Definición de funciones:

R es un lenguaje que permite crear nuevas funciones. Una función se define con una asignación de la forma.

```
nombre <- function(arg1,arg2,...){expresion}
```

La expresión es una fórmula o grupo de fórmulas que utilizan los argumentos para calcular su valor.

Programación en R

Control y ejecución:

```
if(cond) expr
if(cond) cons.expr else alt.expr
for(var in seq) expr
while(cond) expr
repeat expr
break
next
```

Programación en R

```
if (cond.logica) instruccion else instruccion.alternativa
```

```
f4 <- function(x) {
if(x > 5) print("x > 5")
else {
y <- runif(1)
print(paste("y is ", y))
}
}

f4(3)

## [1] "y is  0.0493508966173977"
```

Programación en R

```
for (variable.loop in valores) instruccion
```

```
for(i in 1:10) cat("el valor de i es", i, "\n")

## el valor de i es 1
## el valor de i es 2
## el valor de i es 3
## el valor de i es 4
## el valor de i es 5
## el valor de i es 6
## el valor de i es 7
## el valor de i es 8
## el valor de i es 9
## el valor de i es 10
```

Programación en R

Ejerc.
- Hacer una función que calcule el producto de los 20 primeros números.
- Hacer una función que calcule la suma de los 100 primeros números impares.

GRACIAS!