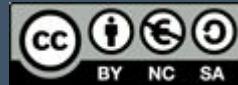


Chuletas de RSTUDIO

Por Aurelio Gallardo Rodríguez



Calculadora

Básico

```
> # Nº pi  
> pi  
[1] 3.141593  
> # Cociente entero  
> 58 %/% 7  
[1] 8  
> # Resto de la división  
> 58 %% 7  
[1] 2  
> # factorial de un número  
> factorial(10)  
[1] 3628800  
> # Combinación (n m) = n! / ( m! (n-m)! )  
> # (5 3) = 5! / ( 3! 2! )  
> choose(5,3)  
[1] 10
```

Complejos

```
> # Complejos  
> a=3-5i  
> # lo siguiente da error  
> b=4-5*i  
Error: objeto 'i' no encontrado
```

print(pi,22) → muestra pi con 22 decimales.

round(pi,3) → redondea pi a 3 decimales

floor, ceiling, trunc

```
> print(pi,22)  
[1] 3.141592653589793115998
```

```
> round(pi,3) > a=4.567  
[1] 3.142 > floor(a)  
[1] 4 > ceiling(a)  
[1] 5 > trunc(a)  
[1] 4
```

Variable y función

```
> f1=function(x){x^2-sqrt(x)}  
> x=3.4  
> f1(x)  
[1] 9.716091
```

Listado de variables, etc → ls()

Borrar listado → rm(list=ls())

Borrar variable a → rm(a)

```
> a=(4-5i)*(3+8i)  
> a  
[1] 52+17i  
> Re(a)  
[1] 52  
> Im(a)  
[1] 17  
> Mod(a)  
[1] 54.70832  
> Arg(a)  
[1] 0.3159703
```

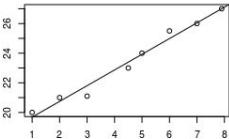
```
> z=complex(modulus=8,argument=pi/5)  
> z  
[1] 6.472136+4.702282i  
> Mod(z)  
[1] 8
```

Regresión lineal

Básico

```
> x=c(1,2,3,4,5,5,6,7,7.9)
> y=c(20,21,21.1,23,24,25.5,26,27)
> plot(x,y)
> lm(y~x)
```

Call:
lm(formula = y



Coefficients:
(Intercept) x
18.63 1.06

```
> l1=lm(y~x)
> # añade recta al dibujo
> abline(l1)
```

Data Frames

lm(happy~money,data=h1)

Si $y=ax + b$, entonces “ a ” es la pendiente, debajo de la x (1.06 en el ejemplo) y “ b ” es “*intercept*” (18.63)

Actualizar r-studio. Entrar en cran
(<https://cran.r-project.org/bin/linux/ubuntu/>) y seguir las instrucciones. La importación de la clave con:
gpg --keyserver keyserver.ubuntu.com --recv 51716619E084DAB9
(ver <https://ubuntuforums.org/showthread.php?t=2221562>)

```
> summary(l1)
```

Call:
lm(formula = y ~ x)

Residuals:

Min	1Q	Median	3Q	Max
-0.70634	-0.13528	0.03519	0.26920	0.51238

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)					
(Intercept)	18.62505	0.33818	55.08	2.41e-09 ***					
x	1.06043	0.06654	15.94	3.87e-06 ***					

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	.	0.1

Residual standard error: 0.4263 on 6 degrees of freedom
Multiple R-squared: 0.9769, Adjusted R-squared: 0.9
F-statistic: 254 on 1 and 6 DF, p-value: 3.874e-06

```
> # Coeficiente de determinación R2
> # O Multiple R-squared
> summary(l1)$r.squared
[1] 0.9769234
```

Otra forma

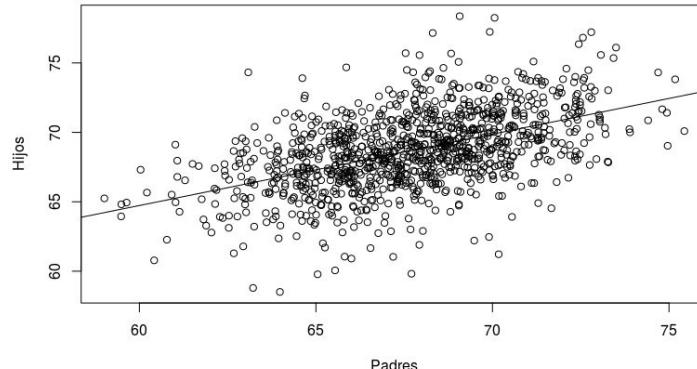
$$\sum_{i=1}^n (y_i - (b_1 x_i + b_0))^2$$

```
> l1$coefficients[1]
(Intercept)
18.62505
> l1$coefficients[2]
x
1.060429
```

Regresión lineal desde data frame

- Importar Data Frame {
- Import Dataset desde Environment
 - `df_pearson=read.table("http://aprender.uib.es/Rdir/pearson.txt ",header=TRUE)`

```
> str(df_pearson)
'data.frame': 1078 obs. of 2 variables:
$ Padres: num 65 63.3 65 65.8 61.1 ...
$ Hijos : num 59.8 63.2 63.3 62.8 64.3 ...
> head(df_pearson,4)
  Padres  Hijos
1 65.04851 59.77827
2 63.25094 63.21404
3 64.95532 63.34242
4 65.75250 62.79238
```



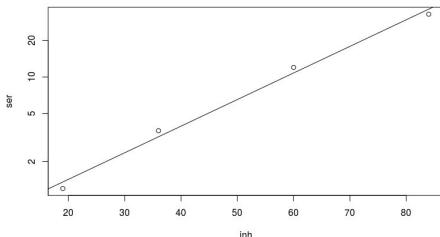
```
> l1=lm(Hijos~Padres,data=df_pearson)
> plot(df_pearson)
> abline(l1)
> summary(l1)$r.squared
[1] 0.2513401
> b=l1$coefficients[1]
> a=l1$coefficients[2]
> a
  Padres
0.514093
> b
(Intercept)
33.8866
```

Regresión semilogarítmica

$$\log(y) = a \cdot x + b \rightarrow \begin{cases} y = 10^{\log(y)} = 10^{a \cdot x + b} = (10^a)^x \cdot 10^b \\ \beta = 10^b \quad \alpha = 10^a \Rightarrow y = \alpha^x \cdot \beta \end{cases} \quad \text{Función exponencial}$$

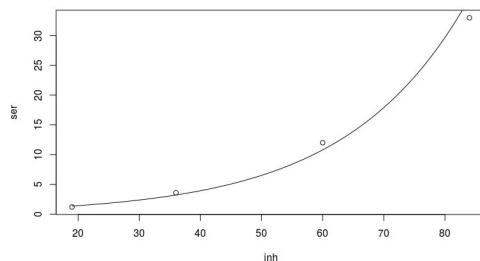
```
> inh=c(19,36,60,84)
> ser=c(1.2,3.6,12,33)
> l1=lm(log10(ser)~inh)
> plot(inh,ser,log="y")
> abline(l1)
> summary(l1)$r.squared
[1] 0.9921146
> a=l1$coefficients[2]
> b=l1$coefficients[1]
> a
      inh
0.02196113
> b
(Intercept)
-0.2842717
> # log(serotonina) = 0.02196 inhibicion - 0.2842
```

En plot no se pone **log10**
sino **log**



```
> beta = 10^b
> alfa = 10^a
> alfa
inh
1.051868
> beta
(Intercept)
0.5196708
> f1=function(x) {beta * alfa^{x}}
> plot(inh,ser)
> curve(f1(x),add=TRUE)
```

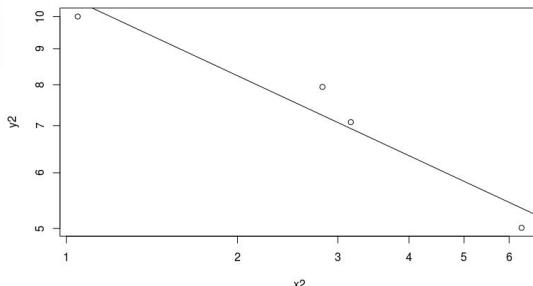
$$\alpha^x \cdot \beta = 1.052^x \cdot 0.52$$



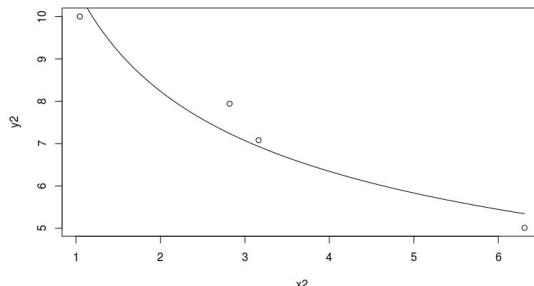
Regresión doble logarítmica

$$\log(y) = a \cdot \log(x) + b \rightarrow \left\{ \begin{array}{l} y = 10^{\log(y)} = (10^{\log(x)})^a \cdot 10^b = x^a \cdot \beta \\ \text{con } \beta = 10^b \Rightarrow y = \beta \cdot x^a \end{array} \right. \quad \text{Función potencial}$$

```
> x2  
[1] 1.047129 2.818383 3.162278 6.309573  
  
> y2  
[1] 10.000000 7.943282 7.079458 5.011872  
  
> l2=lm(log10(y2)~log10(x2))  
> plot(x2,y2,log="xy")  
> abline(l2)  
> summary(l2)$r.squared  
[1] 0.936992  
  
> a=l2$coefficients[2]  
> b=l2$coefficients[1]  
> a  
log10(x2)  
-0.3766045  
  
> b  
(Intercept)  
1.029147
```



```
> beta = 10^b  
> beta  
(Intercept)  
10.69418  
  
> f1=function(x) {beta * {x}^a }  
> plot(x2,y2)  
> curve(f1(x),add=TRUE)
```

$$y = 10.694 \cdot x^{-0.377}$$


Vectores: secuencia ordenada de datos del mismo tipo

- Tipos de datos: logical (TRUE,FALSE), integer, numeric, complex, character
- Ejemplos... `x=c(2,3,5,6,7)` ... `x=c("Hola","Adiós","Hasta luego")`
- Función `scan`: para introducir datos desde el teclado
 - `x_scan=scan()` → Ir introduciendo y aceptando con "Intro". Doble intro para acabar.
 - `notas = scan ("http://aprender.uib.es/Rdir/notas.txt")` → O para introducirlas desde fichero externo.
- De un **data frame**, se puede extraer un vector → `v = data$V1`
- `scan` admite el comando **sep**, como separador → `x_scan = scan(sep=",")` e introduzco los datos a la vez con este separador: `1: 3,4,5,63,2,7` (doble intro) , entonces `> xscan [1] 3 4 5 63 2 7`
- Y admite **dec** como el separador decimal de los números... `xscan = scan (dec=",")`
- Y admite **what** para considerar los datos como de un determinado tipo... `xscan=scan(what="character")`
- Y también **encoding** para indicar el tipo de codificación. O bien **latin1** o **UTF-8**.

Repetir

```
> # repite 5 veces  
> rep(c(1,2,3),times=5)  
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3  
> # 0 bien sólo rep(c(1,2,3),5)  
> # repite CADA elemento 4 veces  
> rep(c(1,2,3),each=4)  
[1] 1 1 1 1 2 2 2 2 3 3 3 3  
> # repite cada elemento unas  
> # veces determinadas  
> rep(c(1,2,3),times=c(4,2,1))  
[1] 1 1 1 1 2 2 2 3
```

Secuencias

```
> # De 3 hasta 15 cada 4.5  
> # El último sin superar 15  
> seq(3,15,by=4.5)  
[1] 3.0 7.5 12.0  
> # Al revés  
> seq(18,2,by=-5.1)  
[1] 18.0 12.9 7.8 2.7  
> # con paso +1  
> 1:3  
[1] 1 2 3  
> 4:-1  
[1] 4 3 2 1 0 -1  
> -(3:5)  
[1] -3 -4 -5
```

```
> # Secuencia de a hasta b  
> # y de número de elementos n  
> seq(3,7,length.out=5)  
[1] 3 4 5 6 7  
> # Secuencia desde a  
> # con paso by = 0.1  
> # y de n = 5 números  
> seq(2,by=0.1,length.out = 5)  
[1] 2.0 2.1 2.2 2.3 2.4
```

Vectores

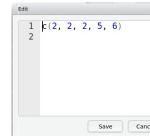
Vectores y Funciones

```
> x  
[1] 2 2 2 5 6  
> x^2  
[1] 4 4 4 25 36  
> sin(x)  
[1] 0.9092974 0.9092974 0.9092974 -0.9589243 -0.2794155  
> f1=function(x) {x^2 / log10(x)}  
> f1(x)  
[1] 13.28771 13.28771 13.28771 35.76691 46.26350
```

Concatenar y modificar

```
> x=c(rep(2,3),5:6)  
> x  
[1] 2 2 2 5 6  
> y=c(4,x,7)  
> y  
[1] 4 2 2 2 5 6 7
```

fix(x)



```
> # Si tenemos una función recursiva como  
> f1=function(x) {mean(1:x)}  
> f1(20)  
[1] 10.5  
> x  
[1] 2 2 2 5 6  
> f1(x)  
[1] 1.5  
Warning message:  
In 1:x : numerical expression has 5 elements  
> sapply(x,FUN=f1)  
[1] 1.5 1.5 1.5 3.0 3.5
```

Calculando

```
> 1:5+2:6  
[1] 3 5 7 9 11  
> (1:5)*(2:6)  
[1] 2 6 12 20 30  
> n=1:3  
> n*(2^n)  
[1] 2 8 24  
> n/(n-1)^2  
[1] Inf 2.00 0.75
```

Funciones

```
length, min, max, [1] 17  
sum, prod, mean,  
diff,cumsum  
(sumas  
acumuladas),  
sort, rev  
> x  
[1] 2 2 2 5 6  
> sum(x)  
> mean(x)  
[1] 3.4  
> cumsum(x)  
[1] 2 4 6 11 17  
> # ordenar decreciente  
> sort(x,dec=TRUE)  
[1] 6 5 2 2 2  
> rev(x)  
[1] 6 5 2 2 2  
> # rev -> revertir orden
```

```
> x  
[1] 2 2 2 5 6  
> # Elemento 4º  
[1] 5  
> x[4]  
[1] 5  
> # Todos menos el 4º  
[1] 5 6  
> x[-4]  
[1] 2 2 2 6  
> # del 3º al 5º  
[1] 2 2 2  
> x[3:5]  
[1] 2 5 6
```

```
> # sin el 1º y 2º  
[1] 2 5 6  
> x[-(1:2)]  
[1] 2 5 6  
> x[x>3]  
[1] 5 6  
> x[x==2 & x!=5]  
[1] 2 2 2  
> x[x==2 | x!=5]  
[1] 2 2 2 6  
> x[x<4]  
[1] 2 2 2  
> x[!x<4]  
[1] 5 6
```

Operadores

==, > , < , >=
,<= !=, &, | , !

Vectores: índices

which

```
> # Indices de x en los que
> # x>4
> which(x>4)
[1] 3 4
> which(x%%2==0)
[1] 3 4 5 9
> # En los que son pares
> x
[1] 3.0 3.0 6.0 8.0 2.0 2.5 3.0 3.5 4.0
```



Seleccionar
un vector
con valores
de otro

```
> #Vector x=c(rep(3,each=2),c(6,8),seq(2,4,by=0.5))
> x=c(rep(3,each=2),c(6,8),seq(2,4,by=0.5))
> x
[1] 3.0 3.0 6.0 8.0 2.0 2.5 3.0 3.5 4.0
> y=c(rep(c(-1,1),4),3)
> y
[1] -1 1 -1 1 -1 1 -1 1 3
> x[y>0]
[1] 3.0 8.0 2.5 3.5 4.0
```

```
> # posición de valor mínimo
> which.min(x)
[1] 5
> # posición de valor máximo
> which.max(x)
[1] 4
```

```
> x
[1] 3.0 3.0 6.0 8.0 2.0 2.5 3.0 3.5 4.0 8.0
> which(x==max(x))
[1] 4 10
> # Posiciones en las que es máximo
```

Resultados nulos: numeric(0), character(0)... o NULL si R no conoce el tipo.

Valores NA (non available)

Borrar atributos (para na.omit(x))

```
> attr(x_sinNA, "na.action")=NULL
> attr(x_sinNA, "class")=NULL
```

```
> x
[1] 3.0 3.0 6.0 8.0 2.0 2.5 3.0 3.5 4.0 8.0
> x[c(4,5)]=10
> x[length(x)+2]=8
> x
[1] 3.0 3.0 6.0 10.0 10.0 2.5 3.0 3.5 4.0 8
> # tenemos un lugar NA
> # no se puede calcular, por ejemplo la media
> mean(x)
[1] NA
> # a menos que se use na.rm=TRUE
> # Que evita usar los NA
> mean(x,na.rm=TRUE)
[1] 5.545455
> sum(x,na.rm = TRUE)
[1] 61
```

```
> # Lugares donde es NA
> is.na(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[11] TRUE FALSE
> which(is.na(x))
[1] 11
> which(!is.na(x))
[1] 1 2 3 4 5 6 7 8 9 10 12
> cumsum(!is.na(x))
[1] 1 2 3 4 5 6 7 8 9 10 10 11
> # cumsum no admite na.rm=TRUE
> # o bien usamos omit
> sum(na.omit(x))
[1] 61
> cumsum(na.omit(x))
[1] 3.0 6.0 12.0 22.0 32.0 34.5 37.5 41.0 45.0 53.0 61.0
```

Factores: son vectores clasificados mediante niveles

```
> S
[1] "M" "F" "M" "F" "M" "F" "M" "M" "M" "F" "F"
> Sex=as.factor(S)
> Sex
[1] M F M F M F M M F F
Levels: F M
> # Añado nuevo nivel
> Sex=factor(Sex,levels=c("F","M","B"),labels=c("Fem","Mas","Bisex"))
> Sex
[1] Mas Fem Mas Fem Mas Fem Mas Fem Fem
Levels: Fem Mas Bisex
```

- ❑ **levels**, cambia los niveles
 - ❑ **labels**, cambia los nombres
 - ❑ O también cambio los nombres con **levels**

```
> levels(Sex)=c("Mujer", "Varón", "Bi")
> Sex
[1] Varón Mujer Varón Mujer Varón Mujer
Levels: Mujer Varón Bi
```

Además obtenemos un **factor ordenado** si aplicamos la función **ordered**

```
> Sex  
[1] Varón Mujer Varón Mujer Varón Mujer Varón Varón Mujer Mujer  
Levels: Mujer Varón Bi  
> Sex=ordered(Sex,levels=c("Bi","Mujer","Varón"))  
> Sex  
[1] Varón Mujer Varón Mujer Varón Mujer Varón Varón Mujer Mujer  
Levels: Bi < Mujer < Varón
```

Y uso levels para agrupar niveles
también

Listas:

contienen elementos de distinto tipo

```
> lista1=list(nombre="Listado",vector=x,med1=mean(x,na.rm=TRUE))  
> # Primer elemento de la lista  
> lista1[[1]]  
[1] "Listado"  
> lista1[1]  
$nombre  
[1] "Listado"  
  
> # De la segunda manera devuelve una "sublista";  
> # De la primera lo que hay "dentro" de la posición 1  
> 3*lista1[2]  
Error in 3 * lista1[2] : argumento no-numérico para operador binario  
> 3*lista1[[2]]  
[1] 9.0 9.0 18.0 30.0 30.0 7.5 9.0 10.5 12.0 24.0 NA 24.0  
> lista1[[2]]  
[1] 3.0 3.0 6.0 10.0 10.0 2.5 3.0  
> # Elemento en la 3ª posición  
> lista1[[2]][3]  
[1] 6
```

Crear elementos
y acceder

Accediendo y
modificando

```
> lista1$vector[3]=45  
> lista1[[2]]  
[1] 3.0 3.0 45.0 10.0 10.0 2.5
```

Estructura
en la lista

```
> # Estructura de la lista  
> str(lista1)  
List of 3  
 $ nombre: chr "Listado"  
 $ vector: num [1:12] 3 3 6 10 10 2.5  
 $ med1 : num 9.33  
> names(lista1)  
[1] "nombre" "vector" "med1"
```

Matrices: elementos del mismo tipo en forma de tabla

Construir matrices

```
> matrix(1:6,nrow=2)           Dos filas  
[,1] [,2] [,3]  
[1,] 1 3 5  
[2,] 2 4 6  
  
> matrix(1:6,nrow=3)          Tres filas  
[,1] [,2]  
[1,] 1 4  
[2,] 2 5  
[3,] 3 6  
  
> matrix(1:6,ncol=2)          O Dos columnas  
[,1] [,2]  
[1,] 1 4  
[2,] 2 5  
[3,] 3 6  
  
> matrix(1:6,ncol=2,byrow=TRUE)  
[,1] [,2]  
[1,] 1 2           Dos col, ordeno por filas  
[2,] 3 4  
[3,] 5 6  
  
> matrix(1:6,ncol=3,byrow=TRUE)  
[,1] [,2] [,3]  
[1,] 1 2 3           3 col, ordeno por filas  
[2,] 4 5 6
```

Construir matrices

```
> matrix(2,ncol=3,nrow=2)  
[,1] [,2] [,3]  
[1,] 2 2 2  
[2,] 2 2 2  
Matriz constante  
  
> rbind(c(2,3,4),c(71,2,3),c(9,90,2))  
[,1] [,2] [,3]  
[1,] 2 3 4  
[2,] 71 2 3  
[3,] 9 90 2  
Fila a Fila  
  
> cbind(c(2,3,4),c(71,2,3),c(9,90,2))  
[,1] [,2] [,3]  
[1,] 2 71 9  
[2,] 3 2 90  
[3,] 4 3 2  
Columna a Columna  
  
> A  
[,1] [,2] [,3]  
[1,] 2 2 2  
[2,] 2 2 2  
[3,] 2 2 2  
Concatenar Matrices  
  
> cbind(A,c(5,7))  
[,1] [,2] [,3] [,4]  
[1,] 2 2 2 5  
[2,] 2 2 2 7  
  
> rbind(A,c(3,2,1))  
[,1] [,2] [,3]  
[1,] 2 2 2  
[2,] 2 2 2  
[3,] 3 2 1
```

Nombrar filas/columnas

```
> A  
[,1] [,2] [,3]  
[1,] 2 2 2  
[2,] 2 2 2  
[3,] 3 2 1  
> # Nombro columnas y filas  
> dimnames(A)=list(c("F1","F2","F3"),c("A","B","C"))  
> A  
A B C  
F1 2 2 2  
F2 2 2 2  
F3 3 2 1
```

Accediendo a...

```
> # Fila 2 y Columna 3  
> A[2,3]  
[1] 2  
> # Fila 2  
> A[2,]  
A B C  
2 2 2  
> # Columna 1  
> A[,1]  
F1 F2 F3  
2 2 3  
> # Con drop=FALSE  
> A[,1,drop=FALSE]  
A  
F1 2  
F2 2  
F3 3
```

Matrices

Submatrices

```
> # Submatriz: Filas 2 y 3  
> # Columnas 1 y 2  
> A[c(2,3),c(1,2)]
```

```
A B  
F2 2 2  
F3 3 2  
> # Columnas 1,3  
> A[,c(1,3)]  
A C  
F1 2 2  
F2 2 2  
F3 3 1
```

```
> # Filas 2 y 3  
> A[c(2,3),]  
A B C  
F2 2 2 2  
F3 3 2 1
```

diagonal

```
> diag(A)  
[1] 2 2 1
```

o por nombre

```
> A["F1",c("A","B")]  
A B  
2 2
```

nº Filas, col.,
dimensiones

```
> nrow(A)  
[1] 3  
> ncol(A)  
[1] 3  
> dim(A)  
[1] 3 3
```

Funciones

```
> # Suma de las filas  
> rowSums(A)  
F1 F2 F3  
8 1 12  
> # Y de las columnas  
> colSums(A)  
A B C D  
7 6 5 3  
> # Media de filas y columnas  
> round(rowMeans(A),2)  
F1 F2 F3  
2.00 0.25 3.00  
> round(colMeans(A),2)  
A B C D  
2.33 2.00 1.67 1.00
```



```
> # Aplicando una función  
> round(sin(A),2)  
A B C D  
F1 0.91 0.91 0.91 0.91  
F2 0.91 0.91 0.91 0.96  
F3 0.14 0.91 0.84 -0.28  
> # Ordenando columnas. MARGIN=2  
> apply(A,MARGIN=2,FUN=sort)  
A B C D  
[1,] 2 2 1 -5  
[2,] 2 2 2 2  
[3,] 3 2 2 6  
> fx = function(x) {sqrt(sum(x^2))}  
> # Aplicar una función a filas  
> apply(A,MARGIN=1,FUN=fx)  
F1 F2 F3  
4.000000 6.082763 7.071068
```

Nombres de filas y columnas

```
> # nombre de una columna  
> colnames(A)  
[1] "A" "B" "C" "D"  
> colnames(A)[2]="Seg"  
> colnames(A)  
[1] "A" "Seg" "C" "D"  
> rownames(A)  
[1] "F1" "F2" "F3"  
> # o nombres de las filas
```

apply

A filas MARGIN=1, a veces, hay que trasponer con t(A)

A todas las celdas , con MARGIN=c(1,2)

Ej: sqrt(A) equivale a apply(A,MARGIN=c(1,2),FUN=sqrt)

Cálculo matricial

```

> A=matrix(c(1,2,1,3),nrow=2,byrow=TRUE)
> B=matrix(c(-2,4,3,1,0,2),nrow=3,byrow=TRUE)
> C=A/2+diag(2)
> A
      [,1] [,2]
[1,]    1    2
[2,]    1    3
> B
      [,1] [,2]
[1,]   -2    4
[2,]    3    1
[3,]    0    2
> C
      [,1] [,2]
[1,]  1.5  1.0
[2,]  0.5  2.5
      [,1] [,2]
[1,]   -1   8.0
[2,]    5   5.5
[3,]    1   5.0
  
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

```

> A%*%v
      [,1]
[1,]    17
[2,]    39
  
```

$$\begin{pmatrix} 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 6 \end{pmatrix} \quad > v\%*%\v$$

$$\begin{pmatrix} 5 \\ 6 \end{pmatrix} \cdot (5 \ 6) \quad > v\%*%\t(v)$$

```

> library("Biodem", lib.loc="~/R/i686-pc-linux-gnu-library/3.2")
> #Calcula A^20 con mtx.exp del paquete Biodem
> mtx.exp(A,20)
      [,1]      [,2]
[1,] 58063278153 158631825968
[2,] 79315912984 216695104121
> library("expm", lib.loc="~/")
> A %^% 3
      [,1]      [,2]
[1,]    11    30
[2,]    15    41
> # o bien el paquete expm
> # El determinante
> det(A)
[1] 1
> # Rango de matriz
> qr(A)$rank
[1] 2
> # Traspuesta con t;
> t(A)
      [,1]      [,2]
[1,]    1    1
[2,]    2    3
> # Inversa con solve
> solve(A)
      [,1]      [,2]
[1,]    3   -2
[2,]   -1    1
  
```

Resolver ecuaciones

$$\begin{aligned} x + 6y - 3z &= 7 \\ 2x - y + z &= 2 \\ x + y - z &= 3 \end{aligned} \left. \right\}$$

$$\begin{pmatrix} 1 & 6 & -3 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 7 \\ 2 \\ 3 \end{pmatrix}$$

```

> solve(A, b)
[1] 1.6666667 0.4444444 -0.8888889
  
```

Cálculo matricial: valores (autovalores) y vectores propios (autovectores)

```
> eigen(A)  
$values  
[1] 3.7320508 0.2679492  
  
$vectors  
[,1] [,2]  
[1,] -0.5906905 -0.9390708  
[2,] -0.8068982 0.3437238
```

Matriz compleja

Vale cualquier cálculo, incluido eigen, excepto el determinante, que se calcula así:

```
> A  
[,1] [,2]  
[1,] 2.5-5i 1.0-5i  
[2,] 0.5-5i 3.5-5i  
> prod(eigen(A)$values)  
[1] 8.25-22.5i
```

- Da los autovalores en orden decreciente de su valor absoluto
- Si da autovalores repetidos, y se repiten también su autovector, es que la matriz no es diagonalizable.

$$A = P \cdot D \cdot P^{-1}$$

```
> D=diag(eigen(A)$values)
```

```
> D
```

[,1]	[,2]
[1,] 3.732051	0.0000000
[2,] 0.000000	0.2679492

```
> P=eigen(A)$vectors
```

```
> P
```

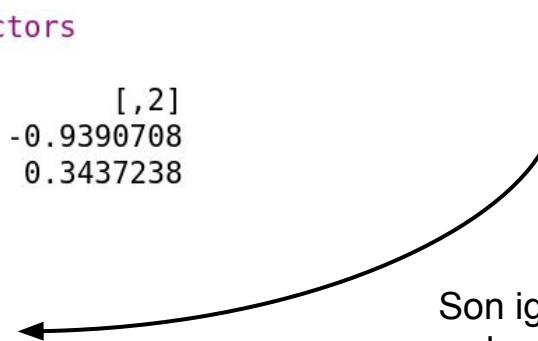
[,1]	[,2]
[1,] -0.5906905	-0.9390708
[2,] -0.8068982	0.3437238

```
> A
```

[,1]	[,2]
[1,] 1	2
[2,] 1	3

```
> P %*% D %*% solve(P)
```

[,1]	[,2]
[1,] 1	2
[2,] 1	3



Son iguales. Se puede diagonalizar

Data Frames: datos de distinto tipo por columnas

- ❑ Para ver los primeros datos, **head(datos)** ó **head(datos,n)** ; **tail** para el final de los mismos.
- ❑ Con **str(datos)** vemos su estructura.
- ❑ Para el nombre de sus columnas **names** y el de los identificadores de filas **rownames**
- ❑ Usando **dimnames** (list) obtenemos identificadores de filas y columnas; **dim(datos)** produce sus dimensiones.
- ❑ **datos\$V1** → vector de la columna llamada "V1" ; **datos[,1]** → de la primera columna
- ❑ **datos[,c(2,3)]** → de la columna 2 y 3; **datos[,1:3]** → de la columna de la 1 a la 3
- ❑ **datos[c(1,4),c(2,3)]** → filas 1 y 4, datos de las columnas 2 y 3
- ❑ **datos[1:4,]** → filas de la 1 a la 4 ; **datos[datos\$V1=="a" & datos\$V2>3,]** → Filas por condiciones lógicas
- ❑ **datos[c("E1","E2"),]** → Selección por filas si éstas se han definido por nombres con **row.names**.
- ❑ **datos[datos\$V1=="a" & datos\$V2>3,][1:4,]** → 4 primeras filas de la subtabla anterior.
- ❑ Para importar datos. desde URL o fichero,, **read.table** (También read.csv, read.xls, read.xlsx, .mtb y .spss)
 - ❑ **sep=","** → carácter de separación (t tabulación). **header =TRUE** → primera fila como encabezado.
 - ❑ **dec** → separador decimal, **encoding** → codificación latin-1 o UTF-8
 - ❑ **stringsAsFactors=FALSE** evita que los datos que sean caracteres se conviertan en factores .
- ❑ Para escribir ficheros en el directorio de trabajo, con **write.table(datos, file="...")**. Se puede usar **sep** y **dec**.

Crear data.frames a partir de vectores

```
> Edad
[1] 17 18 20 18 18 18 19
> Hermanos
[1] 2 0 0 1 1 1 0
> Sexo
[1] "Hombre" "Hombre" "Mujer"  "Hombre" "Hombre" "Hombre" "Mujer"
> datos=data.frame(Sexo,Edad,Hermanos)
> str(datos)
'data.frame': 7 obs. of 3 variables:
 $ Sexo    : Factor w/ 2 levels "Hombre","Mujer": 1 1 2 1 1 1 2
 $ Edad    : num  17 18 20 18 18 18 19
 $ Hermanos: num  2 0 0 1 1 1 0
```

- ❑ Con **row.names** en la definición puedo especificar nombres de filas
- ❑ Con **stringsAsFactors=FALSE** , **datos\$Sexo** sería un vector normal de caracteres, no un factor.
- ❑ **fix(datos)** para modificar/añadir datos sobre la marcha.

Data Frames: modificaciones

Cambiar nombres columnas

```
> str(datos)
'data.frame': 7 obs. of 3 variables:
$ Sexo : Factor w/ 2 levels "Hombre", "Mujer"
$ Edad : num 17 18 20 18 18 18 19
$ Hermanos: num 2 0 0 1 1 1 0
> names(datos)=c("S", "E", "H")
> str(datos)
'data.frame': 7 obs. of 3 variables:
$ S: Factor w/ 2 levels "Hombre", "Mujer"
$ E: num 17 18 20 18 18 18 19
$ H: num 2 0 0 1 1 1 0
> names(datos)[2]="Age"
> names(datos)
[1] "S"    "Age"   "H"
> names(datos)[names(datos)=="Age"]="Edades"
> names(datos)
[1] "S"    "Edades" "H"
```

Y de filas

```
> rownames(datos)=paste("Alum",1:7,sep=" ")
> head(datos)
```

	S	Edades	H
Alum 1	Hombre	17	2
Alum 2	Hombre	18	0
Alum 3	Mujer	20	0
Alum 4	Hombre	18	1
Alum 5	Hombre	18	1
Alum 6	Hombre	18	1

Paste

- ❑ Cambiar nombres columnas y de filas simultáneamente con **dimnames(datos)=list(...Filas...,...Columnas...)** (son vectores)
- ❑ **dim(datos)[1,2] = "Joven"** → cambia dato fila 1 columna 2. Puede que al hacerlo, todos los datos cambien de tipo (si es distinto a carácter)
- ❑ Si volvemos a hacer **dim(datos)[1,2] = 17** (metemos un número) no se cambia automáticamente a tipo numérico; hay que hacer lo siguiente: **datos\$Edad=as.numeric(datos\$Edad)**
- ❑ Podemos convertir un factor en vector de palabras con **as.character**.
- ❑ Ordenar algún dato con **ordered**
 - ❑ **datos\$Hermanos = ordered(datos\$Hermanos, levels=c(0,1,2))**
- ❑ Cambiar el valor de toda una variable: **datos\$Edad="Joven"**

Añadir filas

```
> nuevas.filas=data.frame(S=c("Hombre", "Mujer"),E=c(20,24),H=c(0,2))
> datos=rbind(datos,nuevas.filas)
> tail(datos)
```

	S	E	H
Alum 4	Hombre	18	1
Alum 5	Hombre	18	1
Alum 6	Hombre	18	1
Alum 7	Mujer	19	0
8	Hombre	20	0
9	Mujer	24	2

Añadir columnas

- ❑ Nueva columna:
datos\$Nueva=datos\$Hermanos*3
- ❑ **datos=cbind(datos, Grado=rep("Bilogía",9))** añadiendo un vector de la misma longitud de los datos

Data Frames: sub- data frames y funciones

- Ejemplos: `datos.18=datos[datos$Edad<19,] // datos[datos$Edad<19 & datos$Hermanos==1,]`
- Puede pasar que al hacer una extracción, los datos que sean factores , NO TENGAN TODOS LOS NIVELES. Si lo deseamos podemos eliminar los niveles sobrantes con `droplevels` → `datos.18=droplevels(datos.18)`
- Selección por columnas, equivalentes: `datos[,c(1,2)]` → `datos[c(1,2)]` → `datos[-3]` (quitar la tercera) → `datos[c("Sexo","Edad")]`
- Reordenando datos si reordeno columnas: `datos=datos[c("Edad","Hermanos","Sexo")]`
- Si instalamos el paquete `dplyr`, con la función `select`, obtenemos:
 - `select(datos,starts_with("x"))` → variables (columnas) que empiecen por "x"
 - o bien con `ends_with("x")` o `contains("x")`
- O bien `sub1=subset(datos, condición, columnas)`: `subset(iris,Species=="virginica",1:4)`
 - Los identificadores de filas se heredan; quizás haya que cambiarlos con `rownames`
 - `rownames(sub1)=1:length(rownames(sub1))`
- `attach(iris)` → convierte sus columnas en variables globales // lo contrario en `detach`.

Funciones con `sapply`

```
> sapply(iris[,1:4],FUN=mean)
Sepal.Length Sepal.Width
      5.843333     3.057333
Petal.Length Petal.Width
      3.758000     1.199333
> f1=function(x) {sum(x)^2}
> sapply(iris[,1:4],FUN=f1)
Sepal.Length Sepal.Width
      768252.25    210313.96
Petal.Length Petal.Width
      317757.69    32364.01
```

Sí hace falta, `na.rm=TRUE`

aggregate → Funciones aplicadas y clasificadas por los niveles de un factor

```
> aggregate(Petal.Length~Species,data=
  iris,FUN=mean,na.rm=TRUE)
   Species Petal.Length
1   setosa      1.462
2 versicolor      4.260
3 virginica      5.552

> aggregate(cbind(Petal.Length,Sepal.Length)~Species,data=iris,FUN=mean,na.rm=TRUE)
   Species Petal.Length Sepal.Length
1   setosa      1.462      5.006
2 versicolor      4.260      5.936
3 virginica      5.552      6.588
```

```
> sub1=select(iris,contains("Petal"))
> head(sub1)
  Petal.Length Petal.Width
1           1.4        0.2
2           1.4        0.2
3           1.3        0.2
4           1.5        0.2
5           1.4        0.2
6           1.7        0.4
```

Para separar por
más de un
factor...

~factor1+factor2

Data Frames: paquete dplyr (<https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html>)

filter

```
> filter(iris,Species=="virginica",Petal.Length  
<5)
```

	Sepal.Length	Sepal.Width	Petal.Length
1	4.9	2.5	4.5
2	5.6	2.8	4.9
3	6.3	2.7	4.9
4	6.2	2.8	4.8
5	6.1	3.0	4.9
6	6.0	3.0	4.8

filtrado.
Usar & y |
(and y or)

Seleccionar filas con slice

```
> slice(iris,1:3)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa

arrange, para reordenar

```
> arrange(iris,Sepal.Length,Sepal.Width)  
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

1	4.3	3.0	1.1		
2	4.4	2.9	1.4		
3	4.4	3.0	1.3		

```
> arrange(iris,desc(Sepal.Length),Sepal.Width)
```

1	7.9	3.8	6.4		
2	7.7	2.6	6.9		
3	7.7	2.8	6.7		

desc()
descendente

- ❑ **distinct**, para valores únicos... **distinct(iris,Species)**
- ❑ **mutate**, para añadir columnas:
 - ❑ **mutate(iris,relacion=Petal.Length/Sepal.Length)**
 - ❑ Si sólo quiero nuevas variables, **transmute()**
- ❑ **summarise(iris,media_petalos=mean(Petal.Length,na.rm=TRUE))** → Resume y muestra datos por función
- ❑ Ejemplos aleatorios con **sample_n** y **sample_frac**
- ❑ **group_by() para agrupar**

```
flights[order(flights$year, flights$month, flights$day), ]  
flights[order(flights$arr_delay, decreasing = TRUE), ] or  
flights[order(-flights$arr_delay), ]
```

Data Tables: data frame Enriquecido, ideal para Big Data

- Importar el paquete `data.table`
- `data.table` a partir de `data.frames` o `vectores` // importando con `fread` exactamente igual que como `read.table`.
- `fread`, por defecto, importa los factores como caracteres, a menos que se use `stringsAsFactors=TRUE`
- Creamos índices con `setkey` (`data.table`, `índice1`, `índice2`,...) y se ordenan por orden creciente del primer índice, segundo índice, etc.

```
> irisDT=data.table(iris)
> setkey(irisDT,Species,Petal.Length)
> irisDT["virginica"]
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1:         4.9          2.5          4.5         1.7 virginica
2:         6.2          2.8          4.8         1.8 virginica
3:         6.0          3.0          4.8         1.8 virginica
4:         5.6          2.8          4.9         2.0 virginica
```

```
> # Con J se establece la unión
> # Por ejemplo, en el primer índice tengo "virginica" y en el segundo 5.0 y 6.5
> irisDT[J("virginica",c(5.0,6.5))]
  Sepal.Length Sepal.Width
```

```
1:         5.7          2.5
2:         6.0          2.2
3:         6.3          2.5
4:           NA          NA
  Petal.Length Petal.Width
1:         5.0          2.0
2:         5.0          1.5
3:         5.0          1.9
4:         6.5          NA
  Species
```

```
1: virginica
2: virginica
3: virginica
4: virginica
```

J une dos condiciones, para buscar por índices

■ `data.table[X,Y,by=Z]`

- Extrae X filas → Se ejecuta Y en las columnas, y se agrupa por Z

```
> # extrae segunda columna
> head(irisDT[,2,with=FALSE],3)
  Sepal.Width
1:         3.6
2:         3.0
3:         4.0
```

```
> head(irisDT[,list(Species,Petal.Width)],3)
  Species Petal.Width
1: setosa      0.2
2: setosa      0.1
3: setosa      0.2
```

```
> irisDT[,mean(Petal.Length),by=Species]
```

Species	V1
setosa	1.462
versicolor	4.260
virginica	5.552

```
> irisDT[,list(mean(Petal.Length),sum(Sepal.Width)),by=Species]
```

Species	V1	V2
setosa	1.462	171.4
versicolor	4.260	138.5
virginica	5.552	148.7

■ Primera y última fila de

- `irisDT["setosa",mult="first"]`
- `irisDT["setosa",mult="last"]`

```
> irisDT[,Nueva:=Petal.Length*log(Petal.Width)]
> head(irisDT[,list(Nueva)],4)
```

Nueva
-1.609438
-2.532844
-1.931325
-1.931325

```
> # La borramos
> irisDT[,Nueva:=NULL]
  Sepal.Length Sepal.Width Petal.Length
1:         4.6          3.6          1.0
2:         4.3          3.0          1.1
3:         5.8          4.0          1.2
```

Petal.Width	Species	Nueva
0.2	setosa	-1.609438
0.1	setosa	-2.532844
0.2	setosa	-1.931325

```
> tail(irisDT,2)
```

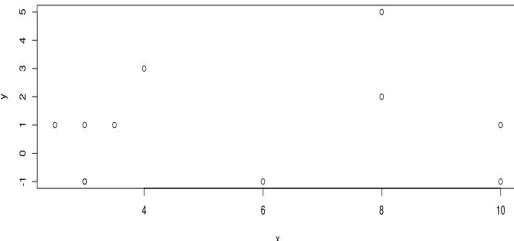
Sepal.Length	Sepal.Width	Petal.Length
7.7	2.8	6.7
7.7	2.6	6.9

Petal.Width	Species	Nueva
2.0	virginica	NA
2.3	virginica	NA

```
DT[,list(sumas1=sum(Var1),medias1=mean(Var2),by=list(Lestras,Var3)) # Suma y media agrupando por otros campos
```

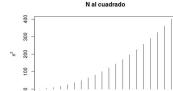
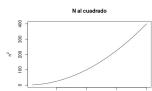
Gráficos I: plot

`plot(x,y) , plot(2^(1:5), plot(f(x))`



Tipos de gráficas

- `type="p"` → puntos (defecto)
- `"l"` → líneas rectas
- `"b"` → puntos y rectas u "o"
- `"h"` → histograma
- `"s"` → escalones
- `"n"` → sólo exterior del gráfico

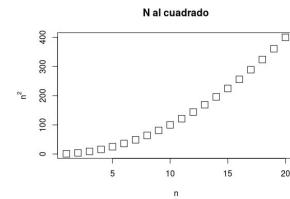


Parámetros de plot.

- `main="titulo", xlab="etiqueta en X", ylab="etiqueta en Y"`
- Se usa `expression()` para añadir fórmulas a las etiquetas o títulos.
- `pch=0...20` y `21..25` → Símbolos.
- `cex=n` // `n` factor de escalado.
- `col="código de color"` ([fichero](#)), de puntos y líneas
- `bg="código color de relleno"`

Márgenes

```
> viejo.par = par ()  
> par ( mar = c (5 ,4.5 ,4 ,2) +0.1)  
> plot(n,n^2,xlab="n",ylab=expression(n^2),main  
="N al cuadrado")  
> par=viejo.par
```



Tipos de líneas

- `lty` → solid, dashed, dotted, dotdash
- `lwd` → grosor de la línea

Rangos de x e Y. Marcas en X e Y

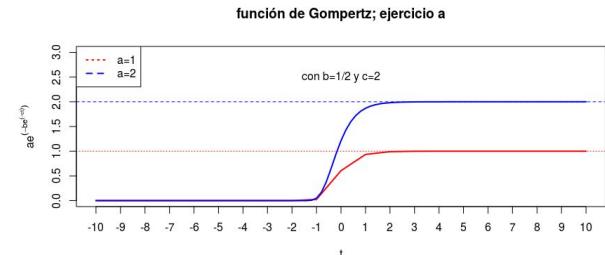
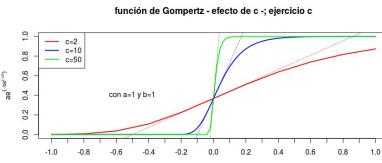
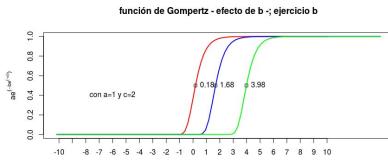
- `xlim (xmin, xmax)` e `ylim(ymin, ymax)`
- `xaxp e yaxp // xaxp(1,10,9)` → dibuja 9+1 marcas entre el 1 y el 10 en el eje X

Añadiendo

- Puntos con `points (vector X, vector Y, pch=, col=)`
- Líneas con `abline` → `abline(a,b)` para recta $y=a+bx$, `abline(v=x0)` para vertical y `abline(h=y0)` para horizontal.
- Grid con `abline(h=0:9, v=-3:3, ...)`
- Texto → `text (x,y, labels=..., pos=...)` donde `pos=NULL,1,2,3,4` indica en el centro , abajo, izq, arriba y dcha.

Gráfica I: plot

- Para poner varios textos → **text(vectorx, vectory, pos=c(1,3,2,...), labels=c("Una","dos",...))**
- Para línea poligonal → **lines(vectorX, vectorY...)**
- Para una función añadida → **curve(f(x), ... parámetros de plot, ...)**
y poner **add=TRUE** si se superpone al plot original. Equivale a plot para funciones.
- Para leyendas → **legend (posición, legend=c(...), parámetros...)**
 - **posición** puede ser topleft, rightbottom, center... (otros ajustes ver **x e y**)
 - Vector **legend** contiene los nombres de las curvas. Se puede usar **expression** en él.
 - Parámetros como **lty**, **pch**, etc.



```

G1=function(a,b,c,t){a*exp(-b*exp(-c*t))}
b=0.5
c=2
t=-10:10
mi = expression(a*exp(-b*exp(-c*t)))
old.par=par()
par(mar=c(5,5,5,5))
plot(t,G1(1,b,c,t),xlab="t",ylab=mi,main="función
de Gompertz; ejercicio
a",xlim=c(-10,10),ylim=c(0,3),col="red",lty="solid",l
wd="2",type="l",xaxp=c(-10,10,20))
curve(G1(2,b,c,x),xlim=c(-10,10),ylim=c(0,3),col="
blue",lty="solid",lwd="2",type="l",add=TRUE)
abline(h=1,col="red",lty="dotted")
abline(h=2,col="blue",lty="dashed")
legend("topleft",legend=c("a=1","a=2"),col=c("red"
,"blue"),lwd=2,lty=c("dotted","dashed"))
text(0,2.5,labels="con b=1/2 y c=2",pos=NULL)
par=old.par

```

Datos cualitativos: observaciones sobre cualidades

```
> Respuestas=c("No","No","Si","No","Si","No")
> Respuestas=factor(Respuestas)
> Respuestas
[1] No No Si No Si No
Levels: No Si
> table(Respuestas)
Respuestas
No Si
 4 2
```

table → tabla de frecuencias abs (contingencia)

Añado factores a la tabla

```
> Res=c("N","S","N","S","S","S")
> Res
[1] "N" "S" "N" "S" "S" "S"
> table(Res)
Res
N S
2 4
> Res=factor(Res,levels=c("N","S","Quizás"))
> Res
[1] N S N S S S
Levels: N S Quizás
> table(Res)
```

Res	N	S	Quizás
	2	4	0

```
> table(Res)[ "N"]
N
2
> table(Res)[3]
Quizás
 0
```

```
> sum(table(Res))
[1] 6
> sqrt(table(Res))
Res
N           S           Quizás
1.414214  2.000000  0.000000
```

Frec. relativas

```
> prop.table(table(Res))
Res
  N          S      Quizás
0.3333333 0.6666667 0.0000000
> table(Res)/length(Res)
Res
  N          S      Quizás
0.3333333 0.6666667 0.0000000
```

Nivel de cierta frecuencia

```
> Edad=c(20,15,20,32,15,16,18,19,21,
,12,15,16,17,18)
> t1=table(Edad)
> t1
Edad
12 15 16 17 18 19 20 21 32
 1 3 2 1 2 1 2 1 1
> # Posiciones con freq. 1
> names(which(t1==1))
[1] "12" "17" "19" "21" "32"
> # Posiciones de máxima frecuencia
> names(which(t1==max(t1)))
[1] "15"
```

Bidimensionales

```
> Peso
[1] 65 75 75 75 74 78 75 75 75 74
[10] 75 75 74 78 75
> Edad
[1] 20 15 20 20 15 18 18 18 18 20
[10] 15 15 16 18 18
> table(Peso,Edad)
Edad
Peso 15 16 18 20
 65 0 0 0 1
 74 0 1 0 2
 75 3 0 4 1
 78 1 0 1 0
> t1=table(Peso,Edad)
> t1[2,3]
[1] 0
> t1[,3]
65 74 75 78
 0 0 4 1
> t1["75",]
15 16 18 20
 3 0 4 1
> t1["78", "18"]
[1] 1
```

Datos cualitativos: frecuencias relativas marginales y globales

```
> # freq. relativas globales  
> round(prop.table(t1),2)  
  Edad  
Peso 15 16 18 20  
  65 0.00 0.00 0.00 0.07  
  74 0.00 0.07 0.00 0.14  
  75 0.21 0.00 0.29 0.07  
  78 0.07 0.00 0.07 0.00  
> # freq. relativas en cada fila  
> round(prop.table(t1, margin=1),2)  
  Edad  
Peso 15 16 18 20  
  65 0.00 0.00 0.00 1.00  
  74 0.00 0.33 0.00 0.67  
  75 0.38 0.00 0.50 0.12  
  78 0.50 0.00 0.50 0.00  
> # freq. relativa por cada columna  
> round(prop.table(t1, margin=2),2)  
  Edad  
Peso 15 16 18 20  
  65 0.00 0.00 0.00 0.25  
  74 0.00 1.00 0.00 0.50  
  75 0.75 0.00 0.80 0.25  
  78 0.25 0.00 0.20 0.00
```

CrossTable (paquete gmodels), con prop.chisq=FALSE resume la tabla de freq. relativas de 2 variables

Se puede aplicar **colSums**, **rowSums** a las tablas como si fueran matrices.

```
> t1=table(Edad,Peso,Sexo)  
> t1  
, , Sexo = H      , , Sexo = M  
  Peso          Peso  
Edad 65 74 75 78  Edad 65 74 75 78  
  15 0 0 1 0      15 0 0 2 1  
  16 0 0 0 0      16 0 1 0 0  
  18 0 0 1 0      18 0 0 3 1  
  20 1 0 0 0      20 0 2 1 0
```

```
> t1["18",,"H"]  
65 74 75 78  
0 0 1 0  
> t1[,, "H"]  
  Peso  
Edad 65 74 75 78  
  15 0 0 1 0  
  16 0 0 0 0  
  18 0 0 1 0  
  20 1 0 0 0
```

```
> # Suma de la tabla t1  
> # Por edades y Sexos  
> apply(t1,MARGIN=c(1,3),FUN=sum)  
  Sexo  
Edad H M  
  15 1 3  
  16 0 1  
  18 1 4  
  20 1 3
```

Multidimensionales

```
> ftable(Sexo,Edad,Peso)  
  Peso 65 74 75 78  
  Sexo Edad  
    H 15      0 0 1 0  
        16      0 0 0 0  
        18      0 0 1 0  
        20      1 0 0 0  
    M 15      0 0 2 1  
        16      0 1 0 0  
        18      0 0 3 1  
        20      0 2 1 0
```

col.vars → elige qué aparece en columnas

```
> ftable(Sexo,Edad,Peso,col.vars=c("Sexo", "Edad"))  
  Sexo H M  
  Edad 15 16 18 20 15 16 18 20  
  Peso  
  65      0 0 0 1 0 0 0 0  
  74      0 0 0 0 0 1 0 2  
  75      1 0 1 0 2 0 3 1  
  78      0 0 0 0 1 0 1 0
```

Frec. Relativas (con MARGIN)

```
> # globales  
> ftable(round(prop.table(t1),2))  
  Sexo H M  
  Edad Peso  
    15 65      0.00 0.00  
        74      0.00 0.00  
        75      0.07 0.14  
        78      0.00 0.07
```

```
> # Sólo por edades  
> ftable(round(prop.table(t1,margin=1),2))  
  Sexo H M  
  Edad Peso  
    15 65      0.00 0.00  
        74      0.00 0.00  
        75      0.25 0.50  
        78      0.00 0.25
```

Apply

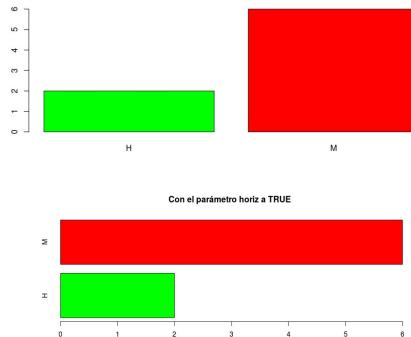
```
> # Solo por edades  
> apply(t1,MARGIN=1,FUN=sum)  
  15 16 18 20  
  4 1 5 4
```

```
> # Por edades y por Sexo  
> ftable(round(prop.table(t1,margin=c(1,3)),2))  
  Sexo H M  
  Edad Peso  
    15 65      0.00 0.00  
        74      0.00 0.00  
        75      1.00 0.67  
        77      0.00 0.00
```

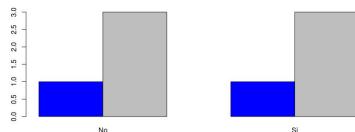
Datos cualitativos: a partir de data frames (y barplot, pie, plot o mosaicplot)

- Se obtiene un resumen de las frec. absolutas a partir de **summary()** // pero para obtener los datos mejor con **sapply()**: **sapply (data.frame, FUN=table)** o bien **sapply (data.frame, FUN=table)\$V1**
- Puedo usar **table** y **ftable** con un data.frame

```
barplot(table(Sexo),col=c("green","red"))
```



```
barplot(table(Sexo,Res),col=c("blue","grey"),  
       ,beside=TRUE)
```



```
barplot(table(Sexo,Res),col=c("blue","grey"),  
       beside=TRUE,  
       legend.text=c("Hombre","Mujer")  
       ,args.legend=list(x=4.5,y=3,cex=2))
```



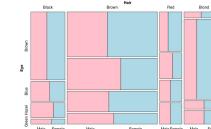
```
pie(table(Pais))
```



```
plot(table(Sexo,Res),col=c("pink","brown")) o bien  
mosaicplot
```



- **plot** y **mosaicplot** admite **dir**, un vector con direcciones verticales-horizontales: **plot(..., dir=c("v","h","v"),...)**
- Paquete **vcd** con **mosaic**; admite **dir**, **highlighting="Variable"** y **highlighting_fill=c("colores")**
- **cotabplot** del paquete **vcd** produce un diagrama de mosaico para cada nivel de la tercera variable.
- Y **mosaic3d** de **vcdExtra**, diagramas en 3d de mosaico (cargar paquete **rgl**).



Datos ordinales: orden natural que permite acumulación

$$N_j = \sum_{k=1}^j n_k$$

Frecuencia absoluta acumulada
 n_j es la frecuencia abs. del nivel I_j

$$F_j = \frac{N_j}{n} = \sum_{k=1}^j f_k$$

Frecuencia relativa acumulada
 f_j es la frecuencia rel. del nivel I_j

```
> # vector ordenado de notas
> notas
[1] A A N S S A N E A A S
[12] S S A E N N E S A
Levels: S < A < N < E
> # ordenado con ordered
> table(notas)
notas
S A N E
6 7 4 3
> # freq. absolutas acumuladas
> cumsum(table(notas))
S A N E
6 13 17 20
> # freq. relativas acumuladas
> cumsum(prop.table(table(notas)))
S A N E
0.30 0.65 0.85 1.00

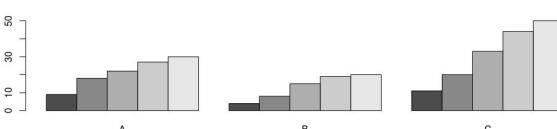
> cumsum(table(notas))/length(notas)
S A N E
0.30 0.65 0.85 1.00
> cumsum(table(notas))/length(notas)
S A N E
0.30 0.65 0.85 1.00
> barplot(cumsum(table(notas)),main="frec. abs. a
cumuladas")
```



Frecuencias acumuladas en tabla multidimensional

apply(tabla,MARGIN=...,FUN=cumsum) donde MARGIN es la dimensión donde acumular las frecuencias.

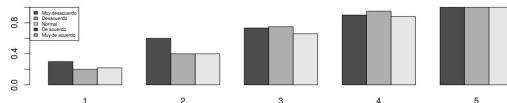
```
> str(df_encuestas)
'data.frame': 100 obs. of 2 variables:
$ empresas : Factor w/ 3 levels "A","B",""
1 1 1 1 1 1 1 1 ...
$ respuestas: Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<"5"
<"4"<..: 4 4 2 1 3 3 4 1 1 3 ...
> head(df_encuestas,4)
empresas respuestas
1 A 4
2 A 4
3 A 2
4 A 1
> table(df_encuestas)
respuestas
empresas 1 2 3 4 5
A 9 9 4 5 3
B 4 4 7 4 1
C 11 9 13 11 6
```



Datos ordinales: multidimensional

Frecuencias acumuladas relativas en tabla multidimensional

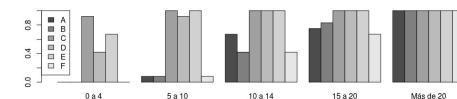
```
> # tabla freq. relativas acumuladas en la variable 1  
> tf=t(apply(prop.table(table(df_encuestas),margin=1),MARGIN=1,FUN=cumsum))  
> tf  
      respuestas  
empresas 1 2 3 4 5  
A 0.30 0.6 0.7333333 0.90 1  
B 0.20 0.4 0.7500000 0.95 1  
C 0.22 0.4 0.6600000 0.88 1  
  
> barplot(tf,beside = TRUE,legend.text = c("Muy desacuerdo","Desacuerdo","Normal","De acuerdo","Mu y de acuerdo"),args.legend = list(x="topleft",cex = 0.55))
```



```
> barplot(t(Freq.rel.acu),beside=TRUE,legend=TRUE,ylim=c(0,1),args.legend=c(x="topleft",y=1))  
> barplot(t(Freq.rel),beside=TRUE,legend=TRUE,ylim=c(0,1),args.legend=c(x=32.5,y=1))
```

Cortar variable numérica con cut para crear una ordinal

```
> IS=InsectSprays  
> head(IS$count,3)  
[1] 10 7 20  
> IS$rango=ordered(cut(IS$count, breaks=c(0,5,10,15,20,Inf), right=FALSE, labels=c("0 a 4","5 a 10","10 a 14","15 a 20","Más de 20")))  
> str(IS)  
'data.frame': 72 obs. of 3 variables:  
 $ count: num 10 7 20 14 14 12 10 23 17 20 ...  
 $ spray: Factor w/ 6 levels "A","B","C","D",...  
 $ rango: Ord.factor w/ 5 levels "0 a 4"<"5 a 10"  
<...: 3 2 5 3 3 3 3 5 4 5 ...  
> Tab=table(IS[,2:3])
```



- **breaks** define los intervalos
- **right=FALSE** (a la derecha intervalos abiertos)
- **Inf** → infinito
- Etiquetas con **labels**

```
> # frecuencias rel acumuladas  
> Freq.rel.acu=round(apply(prop.table(Tab,margin=1),2)  
> Freq.rel.acu  
rango  
spray 0 a 4 5 a 10 10 a 14 15 a 20 Más de 20  
A 0.00 0.08 0.58 0.08 0.25  
B 0.00 0.08 0.33 0.42 0.17  
C 0.92 0.08 0.00 0.00 0.00  
D 0.42 0.50 0.08 0.00 0.00  
E 0.67 0.33 0.00 0.00 0.00  
F 0.00 0.08 0.33 0.25 0.33  
  
> # Frecuencias rel acumuladas  
> Freq.rel.acu=round(apply(prop.table(Tab,margin=1), MARGIN=1, FUN=cumsum),2)  
> Freq.rel.acu  
spray  
rango A B C D E F  
0 a 4 0.00 0.00 0.92 0.42 0.67 0.00  
5 a 10 0.08 0.08 1.00 0.92 1.00 0.08  
10 a 14 0.67 0.42 1.00 1.00 1.00 0.42  
15 a 20 0.75 0.83 1.00 1.00 1.00 0.67  
Más de 20 1.00 1.00 1.00 1.00 1.00 1.00
```

Datos cuantitativos

- `table(x)`, Listado de frecuencias absolutas

```
> edades=c(18,22,16,19,23,18,35,16,45,20,20,22,40,18,45)
> table(edades)      #Frecuencias absolutas
edades
16 18 19 20 22 23 35 40 45
2   3   1   2   2   1   1   1   2
```

- `cumsum(table(x))`, acumuladas

```
> cumsum(table(edades))  #Fr
16 18 19 20 22 23 35 40 45
2   5   6   8   10  11  12  13  15
```

**Como
data
frame**

```
{ tabla_df=data.frame(Resultado=1:6,
                      Frec_Abs=as.vector(table(dados)),
                      Frec_Rel=as.vector(round(prop.table(table(dados)), 2)),
                      Frec_Abs_Acum=as.vector(cumsum(table(dados))),
                      Frec_Rel_Acum=as.vector(round(cumsum(prop.table(table(dados))), 2)))}
```

- `prop.table(table(x))`, Listado de frecuencias relativas

```
edades
16     18     19     20     22     23     35     40     45
0.13  0.20  0.07  0.13  0.13  0.07  0.07  0.07  0.13
```

- `cumsum(prop.table(table(x)))`, Listado de frecuencias relativas acumuladas

```
> round(cumsum(prop.table(table(edades))), 2)
                           relativas acumuladas
16     18     19     20     22     23     35     40     45
0.13  0.33  0.40  0.53  0.67  0.73  0.80  0.87  1.00
```

```
> tabla_df
  Resultado Frec_Abs Frec_Rel Frec_Abs_Acum Frec_Rel_Acum
1           1       2      0.2            2        0.2
2           2       1      0.1            3        0.3
3           3       2      0.2            5        0.5
```

Medidas de tendencia central

Moda: valores de máxima frecuencia.

```
moda = names(which(table(x)==max(table(x))))
```

```
moda = as.numeric(names(which(table(x)==max(table(x)))) )
```

Media aritmética: mean(x)

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

$$\bar{x} = \frac{\sum_{j=1}^k (n_j \cdot X_j)}{n} = \sum_{j=1}^k f_j \cdot X_j$$

Mediana aritmética: median(x)
(valor central, si ordenada)

$$\begin{cases} n \text{ par}, & \frac{x(\frac{n}{2}) + x(1+\frac{n}{2})}{2} \\ n \text{ impar}, & x(1+\frac{n}{2}) \end{cases}$$

```
> edades=c(18,22,16,19,23,18,35,16,45,20,20,22,40,18,45)
> as.numeric(names(which(table(edades)==max(table(edades)))))
  moda
[1] 18
> mean(edades) #La media
[1] 25.13333
> median(edades) #La mediana
[1] 20
> dados=c(1,2,1,4,5,6,3,5,6,3)
> as.numeric(names(which(table(dados)==max(table(dados)))))
[1] 1 3 5 6
> mean(dados)
[1] 3.6
> median(dados)
[1] 3.5
```

Medidas de Posición

Dada una proporción $0 < p < 1$, el **cuantil de orden p** de una variable cuantitativa, que denotaremos por Q_p , es el valor más pequeño tal que ***su frecuencia relativa acumulada es mayor o igual que p .***

```
> sort(dados)  
[1] 1 1 2 3 3 4 5 5 6 6  
> length(dados)  
[1] 10
```

ordenado

$Q_{0.65}$

65% de 10 es 6.5. El siguiente es el 7º, el cuantil es 5.

O bien, con las frecuencias acumuladas. El 4 está al 60%; como es 65% es más, el siguiente, el 5.

```
cumsum(prop.table(table(dados)))  
1 2 3 4 5 6  
0.2 0.3 0.5 0.6 0.8 1.0
```

- **Mediana**, cuantil $Q_{0.5}$
- **Cuartiles**: $Q_{0.25}$, $Q_{0.5}$ y $Q_{0.75}$; primer, segundo y tercer cuartil.
- **Deciles**, cuantiles múltiplos de 0.1. $Q_{0.1}$ es el primer decil.
- **Percentil**: Cuantil múltiplo de 0.01. Por ejemplo, $Q_{0.01}$

- **quantile(x,0.25)**
- Se puede **modificar** con **type=1..7**
- El método por defecto es **type=7**

Medidas de dispersión

- rango = max - min, se calcula con **diff(range(x))**
- rango intercuartílico = $Q_{0.75} - Q_{0.25}$, con **IQR(x,type...)**
- Si tiene NA, **na.rm=TRUE**

Varianza

$$s^2 = \frac{1}{n} \cdot \left(\sum_{i=1}^n (x_i - \bar{x})^2 \right)$$

$$\tilde{s}^2 = \frac{n}{n-1} \cdot s^2$$

$$s^2 = \frac{n-1}{n} \cdot \tilde{s}^2$$



Desv. típica

$$s = +\sqrt{s^2}$$

$$s = \sqrt{\frac{n-1}{n}} \cdot \tilde{s}$$



Varianza muestral

$$\tilde{s}^2 = \frac{1}{n-1} \cdot \left(\sum_{i=1}^n (x_i - \bar{x})^2 \right)$$

Desv. típica muestral

$$\tilde{s} = +\sqrt{\tilde{s}^2}$$

```
> # vector x
```

```
> x  
[1] 1 2 3 4 5 6 2 3 2 3 4 2 2 3 2 2 5 7 3 4  
2 1 3 6
```

```
> # Varianza MUESTRAL de x
```

```
> var(x)  
[1] 2.606884
```

```
> # Varianza de X (verdadera)  
> var(x)*(length(x)-1)/length(x)
```

```
[1] 2.498264
```

```
> # Desviación típica MUESTRAL de x  
> sd(x)
```

```
[1] 1.614585
```

```
> # Desviación típica de x (verdadera)  
> sd(x)*sqrt((length(x)-1)/length(x))  
[1] 1.58059
```

Medidas de dispersión

Para mínimo, primer cuartil, mediana, media, tercer cuartil y máximo, usar **summary**, o a un vector, o a columnas cuantitativas de un data frame.

```
> summary(x)
   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
1.000  2.000  3.000  3.208  4.000  7.000
> summary(iris[,1:4])
  Sepal.Length Sepal.Width Petal.Length Petal.Width
Min.       :4.300     Min.    :2.000      Min.    :1.000      Min.    :0.100
1st Qu.:5.100     1st Qu.:2.800      1st Qu.:1.600      1st Qu.:0.300
Median  :5.800     Median :3.000      Median :4.350      Median :1.300
Mean    :5.843     Mean   :3.057      Mean   :3.758      Mean   :1.199
3rd Qu.:6.400     3rd Qu.:3.300      3rd Qu.:5.100      3rd Qu.:1.800
Max.    :7.900     Max.   :4.400      Max.   :6.900      Max.   :2.500
```

o bien **aggregate**: aggregate (cbind (Sepal . Length , Sepal . Width , Petal . Length ,Petal . Width) ~ Species , data = iris , FUN = summary) pero menos atractivo

Resultados interesantes

- $s^2 \geq 0$, y si $s=0$ todos los datos son iguales a su media
- Y...

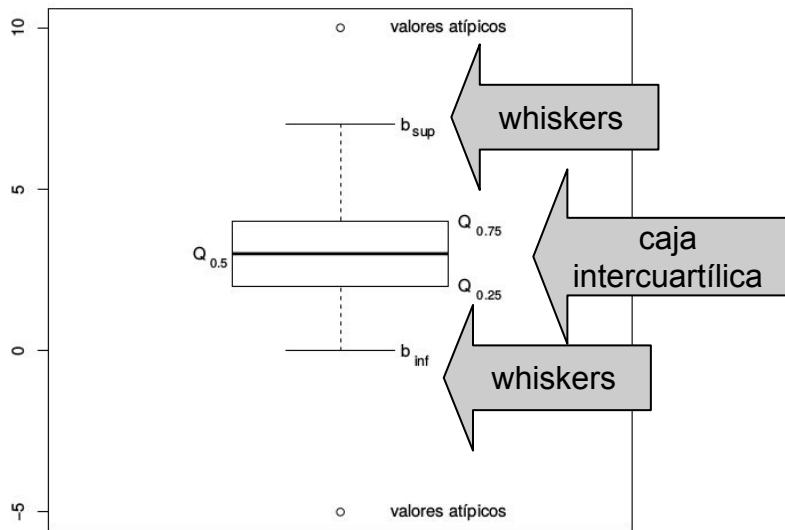
$$s^2 = \frac{\sum_{i=1}^n x_i^2 - n \cdot \bar{x}^2}{n} = \frac{1}{n} \cdot \sum_{i=1}^n x_i^2 - \bar{x}^2$$

Con **segmentación**, usar by
by (columnas, factor, FUN=summary)

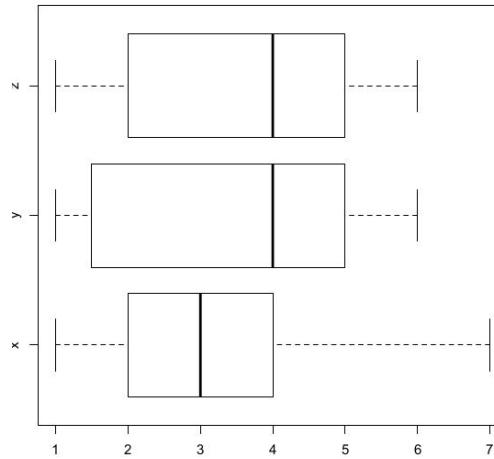
```
> by(iris[,1:2],iris$Species,FUN=summary)
iris$Species: setosa
  Sepal.Length Sepal.Width
  Min.       :4.300     Min.    :2.300
  1st Qu.:4.800     1st Qu.:3.200
  Median  :5.000     Median :3.400
  Mean    :5.006     Mean   :3.428
  3rd Qu.:5.200     3rd Qu.:3.675
  Max.    :5.800     Max.   :4.400
-----
iris$Species: versicolor
  Sepal.Length Sepal.Width
  Min.       :4.900     Min.    :2.000
  1st Qu.:5.600     1st Qu.:2.525
  Median  :5.900     Median :2.800
  Mean    :5.936     Mean   :2.770
  3rd Qu.:6.300     3rd Qu.:3.000
  Max.    :7.000     Max.   :3.400
-----
iris$Species: virginica
  Sepal.Length Sepal.Width
  Min.       :4.900     Min.    :2.200
  1st Qu.:6.225     1st Qu.:2.800
  Median  :6.500     Median :3.000
  Mean    :6.588     Mean   :2.974
  3rd Qu.:6.900     3rd Qu.:3.175
  Max.    :7.900     Max.   :3.800
```

Diagramas de caja

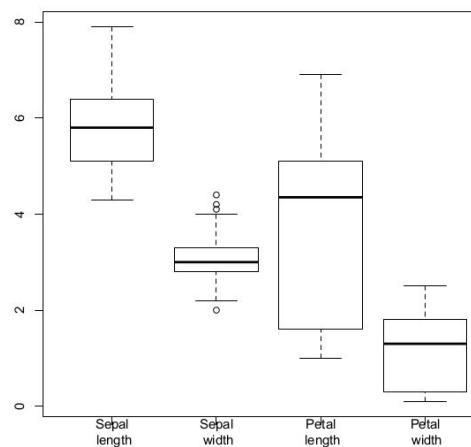
boxplot(x)



$$b_{inf} = \max\{m, Q_{0.25} - 1.5 \cdot (Q_{0.75} - Q_{0.25})\}$$
$$b_{sup} = \min\{M, Q_{0.75} + 1.5 \cdot (Q_{0.75} - Q_{0.25})\}$$



```
boxplot (x , y , z ,  
names = c ( "x" , "y"  
,"z" ) , horizontal =  
TRUE )
```



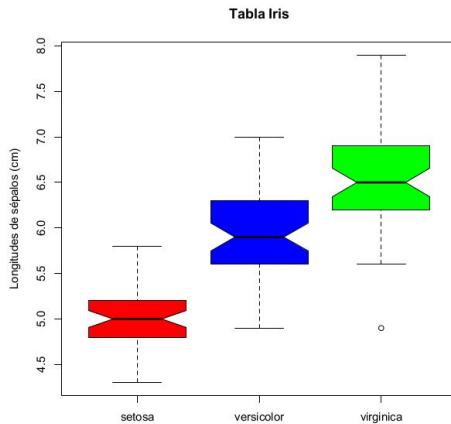
```
boxplot ( iris [ , 1:4 ] ,  
names = c ( " Sepal \\ n  
length " , " Sepal \\ n  
width " ,  
" Petal \\ n length " , "  
Petal \\ n width " ) )
```

Diagramas de caja segmentados

`boxplot(variable numérica ~ variable factor, data=data frame).`

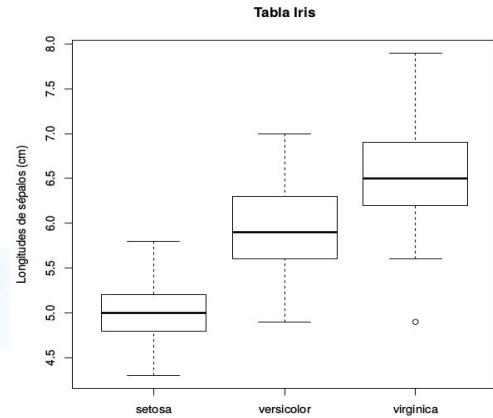
```
boxplot(Sepal.Length ~ Species, data=iris, ylab="Longitudes de sépalos (cm)", main="Tabla Iris")
```

`notch = TRUE`



stats, out y group
Información que da el boxplot.

```
> b1=boxplot(Sepal.Length ~ Species , data = iris )
> # stats nos da binf, Q0.25, Q0.5, Q0.75 y bsup
> b1$stats
 [,1] [,2] [,3]
[1,] 4.3 4.9 5.6
[2,] 4.8 5.6 6.2
[3,] 5.0 5.9 6.5
[4,] 5.2 6.3 6.9
[5,] 5.8 7.0 7.9
> # out los valores atípicos y group a qué diagramas pertenecen
> b1$out
[1] 4.9
> b1$group
[1] 3
```



Datos cuantitativos multidimensionales

“p” observaciones cuantitativas

$$X = \left(\begin{array}{cccc} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{array} \right) \quad \text{“n” observaciones}$$

Observación “i-ésima”

$$x_{i\bullet} = (x_{i1}, x_{i2}, \dots, x_{ip}).$$

$$x_{\bullet j} = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix} \quad \text{Todas las observaciones de la variable “j-ésima”}$$

$$X = \begin{pmatrix} x_{1\bullet} \\ x_{2\bullet} \\ \vdots \\ x_{n\bullet} \end{pmatrix} = (x_{\bullet 1}, x_{\bullet 2}, \dots, x_{\bullet p}).$$

Medias $\bar{X} = \{\bar{x}_{\bullet 1}, \bar{x}_{\bullet 2}, \dots, \bar{x}_{\bullet p}\}$ $\bar{x}_{\bullet j} = \frac{1}{n} \cdot \sum_{i=1}^n x_{ij}$

Varianzas

$$s_X^2 = \{s_1^2, s_2^2, \dots, s_p^2\} \quad s_X = \left\{ +\sqrt{s_1^2}, +\sqrt{s_2^2}, \dots, +\sqrt{s_p^2} \right\}$$

$$s_j^2 = \frac{1}{n} \cdot \sum_{i=1}^n (x_{ij} - \bar{x}_{\bullet j})^2$$

Varianza Muestral

$$\tilde{s}_X^2 = \{\tilde{s}_1^2, \tilde{s}_2^2, \dots, \tilde{s}_p^2\}$$

$$\tilde{s}_j^2 = \frac{1}{n-1} \cdot \sum_{i=1}^n (x_{ij} - \bar{x}_{\bullet j})^2$$

Desv. típica

$$s_X = \left\{ +\sqrt{s_1^2}, +\sqrt{s_2^2}, \dots, +\sqrt{s_p^2} \right\}$$

Desv. típica muestral

$$\tilde{s}_X = \left\{ +\sqrt{\tilde{s}_1^2}, +\sqrt{\tilde{s}_2^2}, \dots, +\sqrt{\tilde{s}_p^2} \right\}$$

y se cumple que...

$$(n-1) \cdot \tilde{s}_j^2 = n \cdot s_j^2$$

Datos cuantitativos multidimensionales

```
> # data frame X
> X
  V1 V2 V3
1  1 -1  3
2  1  0  3
3  2  3  0
4  3  0  1
> # Con data frames se usa sapply
> sapply(X,mean)
  V1   V2   V3
1.75 0.50 1.75
> # Hay que tener en cuenta que son muestrales
> varNormal = function(x) {var(x)*(length(x)-1)/length(x)}
> sapply(X,var)
      V1          V2          V3
0.9166667 3.0000000 2.2500000
> sapply(X,varNormal)
      V1          V2          V3
0.6875 2.2500 1.6875
> sdNormal = function(x) {sqrt(varNormal(x))}
> sapply(X,sd)
      V1          V2          V3
0.9574271 1.7320508 1.5000000
> sapply(X,sdNormal)
      V1          V2          V3
0.8291562 1.5000000 1.2990381
```

Con
Matrices

```
> M=matrix(c(1,4,5,6,3,2,4,3,2),nrow=3,byrow=TRUE)
> M
      [,1] [,2] [,3]
[1,]    1    4    5
[2,]    6    3    2
[3,]    4    3    2
> apply(M,MARGIN=2,FUN=mean)
[1] 3.666667 3.333333 3.000000
```

```
> apply(M,MARGIN=2,FUN=varNormal)
[1] 4.2222222 0.2222222 2.0000000
> apply(M,MARGIN=2,FUN=var)
[1] 6.3333333 0.3333333 3.0000000
> apply(M,MARGIN=2,FUN=sdNormal)
[1] 2.0548047 0.4714045 1.4142136
> apply(M,MARGIN=2,FUN=sd)
[1] 2.5166115 0.5773503 1.7320508
```

Matrices tipificadas

A cada entrada de la matriz se resta la media de la variables y se divide todo entre la desviación típica.

Así se pueden comparar sus variables sin el efecto de la media o las varianzas.

```
> # Se puede aplicar a una matriz o a un data frame  
> X  
  V1 V2 V3  
1  1 -1  3  
2  1  0  3  
3  2  3  0  
4  3  0  1  
  
> # Matriz centrada según las medias  
> # Resta la media de cada variable a cada entrada  
> scale(X,center=TRUE,scale=FALSE)  
  V1   V2   V3  
[1,] -0.75 -1.5  1.25  
[2,] -0.75 -0.5  1.25  
[3,]  0.25  2.5 -1.75  
[4,]  1.25 -0.5 -0.75  
attr(,"scaled:center")  
  V1   V2   V3  
1.75 0.50 1.75  
attr(,"scaled:scale")  
  V1   V2   V3  
0.9574271 1.7320508 1.5000000
```



```
> # Matriz centrada y escalada  
> # Es decir, TIPIFICADA  
> scale(X,center=TRUE,scale=TRUE)  
  V1   V2   V3  
[1,] -0.7833495 -0.8660254  0.8333333  
[2,] -0.7833495 -0.2886751  0.8333333  
[3,]  0.2611165  1.4433757 -1.1666667  
[4,]  1.3055824 -0.2886751 -0.5000000  
attr(,"scaled:center")  
  V1   V2   V3  
1.75 0.50 1.75  
attr(,"scaled:scale")  
  V1   V2   V3  
0.9574271 1.7320508 1.5000000  
  
> # ¡Pero ojo! Porque lo tipifica  
> # en función de la desv. típica MUESTRAL  
> # NO ES LA MATRIZ TIPIFICADA  
> # O bien, hacer scale(X)|
```

$$Z = \begin{pmatrix} \frac{x_{11}-\bar{x}_{\bullet 1}}{s_1} & \frac{x_{12}-\bar{x}_{\bullet 2}}{s_2} & \dots & \frac{x_{1p}-\bar{x}_{\bullet p}}{s_p} \\ \frac{x_{21}-\bar{x}_{\bullet 1}}{s_1} & \frac{x_{22}-\bar{x}_{\bullet 2}}{s_2} & \dots & \frac{x_{2p}-\bar{x}_{\bullet p}}{s_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_{n1}-\bar{x}_{\bullet 1}}{s_1} & \frac{x_{n2}-\bar{x}_{\bullet 2}}{s_2} & \dots & \frac{x_{np}-\bar{x}_{\bullet p}}{s_p} \end{pmatrix}.$$

Si molesta el atributo,
escribir para
eliminarlo

```
attr(X_centered, "scaled:center")=NULL
```

Verdadera tipificación

Cuando se divide entre la VERDADERA desviación típica, NO LA MUESTRAL

```
> # Verdadera Tipificación
> # PROCEDIMIENTO 1
> # Multiplico por el factor
> # sqrt(n/(n-1))
> n=dim(X)[1]
> # el número de filas de la matriz
> n
[1] 4
> Xtip = scale(X)*sqrt(n/(n-1))
> Xtip
```

	V1	V2	V3
[1,]	-0.9045340	-1.0000000	0.9622504
[2,]	-0.9045340	-0.3333333	0.9622504
[3,]	0.3015113	1.6666667	-1.3471506
[4,]	1.5075567	-0.3333333	-0.5773503

```
attr(,"scaled:center")
  V1   V2   V3
1.75 0.50 1.75
attr(,"scaled:scale")
  V1   V2   V3
0.9574271 0.20508 1.5000000
```



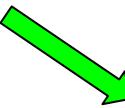
No son las des.
típicas verdaderas

O **apply(X,center=TRUE,
scale=X_dev_tip)** si X es
una matriz

$$\text{Queremos } \text{scale}(X) = \frac{X}{\bar{S}_X}$$

pero tenemos $\text{scale}(X)^* = \frac{\tilde{X}}{\tilde{S}_X}$

$$\frac{X}{\bar{S}_X} = \frac{X}{\tilde{S}_X} \cdot \frac{\tilde{S}_X}{\bar{S}_X} = \frac{X}{\tilde{S}_X} \cdot \sqrt{\frac{n}{n-1}} = \text{scale}(X) \cdot \sqrt{\frac{n}{n-1}}$$



```
> # Segunda opción
> # Usar el vector de desv. típicas verdadera
> # uso la función anterior
> sdNormal
function(x) {sqrt(varNormal(x))}
> varNormal
function(x) {var(x)*(length(x)-1)/length(x)}
> X_dev_tip = sapply(X, sdNormal)
> X_dev_tip
V1          V2          V3
0.8291562 1.5000000 1.2990381
> Xtip=scale(X,center=TRUE,scale=X_dev_tip)
> Xtip
```

	V1	V2	V3
[1,]	-0.9045340	-1.0000000	0.9622504
[2,]	-0.9045340	-0.3333333	0.9622504
[3,]	0.3015113	1.6666667	-1.3471506
[4,]	1.5075567	-0.3333333	-0.5773503

```
attr(,"scaled:center")
```

	V1	V2	V3
1.75	0.50	1.75	

```
attr(,"scaled:scale")
  V1   V2   V3
0.8291562 1.5000000 1.2990381
```



Sí son las des.
típicas verdaderas

Covarianza

Medida de la propensión que tienen dos variables a crecer o decrecer conjuntamente. Si la covarianza es positiva, ambas crecen o decrecen. Si es negativa, una crece y la otra decrece.

- Tendremos covarianza “verdadera” y muestral
- La varianza es la covarianza de una variable consigo misma.
- Usaremos la **correlación de Pearson**, una forma de normalizar las covarianzas.

$$\tilde{s}_{ij} = \frac{1}{n-1} \cdot \sum_{k=1}^n ((x_{ki} - \bar{x}_{\bullet i}) \cdot (x_{kj} - \bar{x}_{\bullet j})) = \frac{n}{n-1} \cdot s_{ij}$$

**Covarianza
Muestral**

$$s_{ij} = \frac{1}{n} \cdot \sum_{k=1}^n ((x_{ki} - \bar{x}_{\bullet i}) \cdot (x_{kj} - \bar{x}_{\bullet j}))$$

$$s_{ij} = \frac{1}{n} \cdot \sum_{k=1}^n x_{ki} \cdot x_{kj} - \bar{x}_{\bullet i} \cdot \bar{x}_{\bullet j}$$

```
> X
  V1 V2 V3
1  1 -1  3
2  1  0  3
3  2  3  0
4  3  0  1
> # covarianza Muestral
> # de dos columnas
> cov(X$V1,X$V2)
[1] 0.5
> # covarianza Verdadera
> n=dim(X)[1]
> cov(X$V1,X$V2)*(n-1)/n
[1] 0.375
```

```
> cov(X)
            V1          V2          V3
V1  0.9166667  0.5000000 -1.083333
V2  0.5000000  3.0000000 -2.166667
V3 -1.0833333 -2.166667  2.250000
> # Matrices de covarianza
> # S12 = S21 ... Sij = Sji
> # Matrices de covarianza VERDADERA
> cov(X)*(n-1)/n
            V1          V2          V3
V1  0.6875  0.375 -0.8125
V2  0.3750  2.250 -1.6250
V3 -0.8125 -1.625  1.6875
```

```
> varNormal
function(x) {var(x)*(length(x)-1)/length(x)}
> sapply(X,varNormal)
            V1          V2          V3
0.6875  2.2500  1.6875
> # recordamos las varianzas verd.
> # coinciden con la diagonal de la matriz
```

Correlación (de Pearson)

- ❑ Es simétrica, $r_{ij} = r_{ji}$
- ❑ Los valores están entre -1 y 1
- ❑ $\text{signo}(r_{ij}) = \text{signo}(s_{ij})$
- ❑ Si el coeficiente $r_{ij} = \pm 1$ indica que hay una relación lineal perfecta
- ❑ El coeficiente de determinación R^2 de la regresión lineal es igual a r_{ij}^2 entre sus dos variables.
- ❑ Viene a ser una **covarianza normalizada**.

$$r_{ij} = \frac{s_{ij}}{s_i \cdot s_j} \quad r_{ij} = \frac{\tilde{s}_{ij}}{\tilde{s}_i \cdot \tilde{s}_j}$$

```
> # Matriz de correlación
> cor(X)
      V1          V2          V3
V1  1.0000000  0.3015113 -0.7543365
V2  0.3015113  1.0000000 -0.8339504
V3 -0.7543365 -0.8339504  1.0000000
> # No hace falta pensar en si procede
> # o no de medidas muestrales o no
> # ... o bien
> S=cov(X)
> cov2cor(S)
      V1          V2          V3
V1  1.0000000  0.3015113 -0.7543365
V2  0.3015113  1.0000000 -0.8339504
V3 -0.7543365 -0.8339504  1.0000000
```

Teorema: La matriz de correlaciones es la matriz de covarianzas de la matriz tipificada. **Así se ahorra cálculo con datos grandes.**
Conviene tipificar primero...

```
> # número de observaciones
> n=dim(X)[1]
> sd_ver
function(x){sd(x)*sqrt((length(x)-1)/length(x))}
> Xdev=sapply(X,sd_ver)
> Xdev
      V1          V2          V3
0.8291562 1.5000000 1.2990381
> Xtip=scale(X,center=TRUE,scale=Xdev)
> cov(Xtip)*(n-1)/n
      V1          V2          V3
V1  1.0000000  0.3015113 -0.7543365
V2  0.3015113  1.0000000 -0.8339504
V3 -0.7543365 -0.8339504  1.0000000
```

Correlaciones con datos NA

Con dos vectores, para **cov** y **cor**, usar observaciones completas.
use="complete.obs"

```
> x=c(1,2,NA,4,6,2)
> y=c(2,4,-3,5,7,NA)
> cov(x,y,use="complete.obs")
[1] 4.5
> cov(x,y)
[1] NA
> cor(x,y,use="complete.obs")
[1] 0.9749135
> cor(x,y)
[1] NA
```

En general, se eliminan

O bien, (1) para cada par de columnas se eliminan los pares que contengan o un NA, o bien (2) se eliminan todas las filas que contengan NA

use="pairwise.complete.obs" (1)

```
> X
      [,1]  [,2]  [,3]
[1,]    1     2    -2
[2,]    2     4     1
[3,]   NA    -3     0
[4,]    4     5     2
[5,]    6     7     3
[6,]    2    NA     0
> cov(X)
      [,1]  [,2]      [,3]
[1,]   NA    NA      NA
[2,]   NA    NA      NA
[3,]   NA    NA 3.066667
```

```
> cov(X,use="pairwise.complete.obs")
      [,1]  [,2]      [,3]
[1,]  4.0  4.50 3.500000
[2,]  4.5 14.50 4.750000
[3,]  3.5  4.75 3.066667
```

use="complete.obs" (2)

```
> cov(X,use="complete.obs")
      [,1]      [,2]      [,3]
[1,] 4.916667 4.500000 4.333333
[2,] 4.500000 4.333333 4.333333
[3,] 4.333333 4.333333 4.666667
```

Otros (Ejemplo)

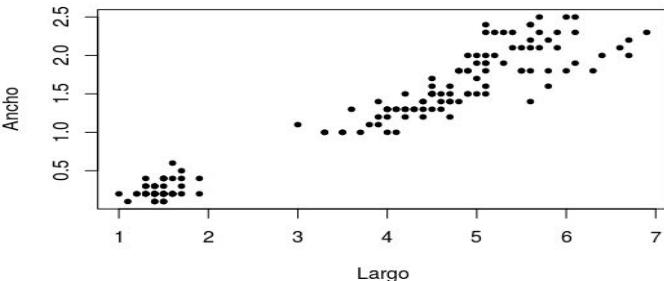
```
> medidas=names(iris_num)
> medidas
[1] "Sepal.Length" "Sepal.Width"
"Petal.Length"
[4] "Petal.Width"
> n=length(meidas)
> n
[1] 4
> indices=upper.tri(diag(n))
> indices
     [,1]   [,2]   [,3]   [,4]
[1,] FALSE  TRUE  TRUE  TRUE
[2,] FALSE FALSE TRUE  TRUE
[3,] FALSE FALSE FALSE TRUE
[4,] FALSE FALSE FALSE FALSE
> diag(n)
     [,1]   [,2]   [,3]   [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

upper.tri, produce la matriz triangular superior. $X_{ij} = \text{TRUE}$ si $i < j$
lower.tri lo contrario. $X_{ij} = \text{TRUE}$ si $i > j$
con **diag=TRUE** consigo $i \leq j$ ó $i \geq j$

```
> medidal=matrix(rep(meidas,times=n),nrow=n,b
yrow=TRUE)
> medidal
     [,1]      [,2]
[1,] "Sepal.Length" "Sepal.Width"
[2,] "Sepal.Length" "Sepal.Width"
[3,] "Sepal.Length" "Sepal.Width"
[4,] "Sepal.Length" "Sepal.Width"
     [,3]      [,4]
[1,] "Petal.Length" "Petal.Width"
[2,] "Petal.Length" "Petal.Width"
[3,] "Petal.Length" "Petal.Width"
[4,] "Petal.Length" "Petal.Width"
> medida2=matrix(rep(meidas,times=n),ncol=n,b
yrow=FALSE)
> medida2
     [,1]      [,2]
[1,] "Sepal.Length" "Sepal.Length"
[2,] "Sepal.Width"  "Sepal.Width"
[3,] "Petal.Length" "Petal.Length"
[4,] "Petal.Width"  "Petal.Width"
     [,3]      [,4]
[1,] "Sepal.Length" "Sepal.Length"
[2,] "Sepal.Width"  "Sepal.Width"
[3,] "Petal.Length" "Petal.Length"
[4,] "Petal.Width"  "Petal.Width"
> indices=upper.tri(diag(n))
> medidal=medidal[indices]
> medidal
[1] "Sepal.Width"  "Petal.Length"
[3] "Petal.Length" "Petal.Width"
[5] "Petal.Width"  "Petal.Width"
.
.
.
> medida2=medida2[indices]
> medida2
[1] "Sepal.Length" "Sepal.Length"
[3] "Sepal.Width"  "Sepal.Length"
[5] "Sepal.Width"  "Petal.Length"
> corrs = as.vector(cor(iris_num)) [indices]
> corrs.abs = abs(corrs)
> corrs_df=data.frame(medidal,medida2,corrs,cor
rs.abs)
> corrs_df
   medidal      medida2      corrs
1 Sepal.Width Sepal.Length -0.1175698
2 Petal.Length Sepal.Length  0.8717538
3 Petal.Length Sepal.Width -0.4284401
4 Petal.Width Sepal.Length  0.8179411
5 Petal.Width Sepal.Width -0.3661259
6 Petal.Width Petal.Length  0.9628654
   corrs.abs
1 0.1175698
2 0.8717538
3 0.4284401
4 0.8179411
5 0.3661259
6 0.9628654
> corrs_df_ord=corrs_df[order(corrs_df$corrs.ab
s,decreasing = TRUE),]
> corrs_df_ord
   medidal      medida2      corrs
6 Petal.Width Petal.Length  0.9628654
2 Petal.Length Sepal.Length  0.8717538
4 Petal.Width Sepal.Length  0.8179411
3 Petal.Length Sepal.Width -0.4284401
5 Petal.Width Sepal.Width -0.3661259
1 Sepal.Width Sepal.Length -0.1175698
   corrs.abs
6 0.9628654
2 0.8717538
4 0.8179411
3 0.4284401
5 0.3661259
1 0.1175698
```

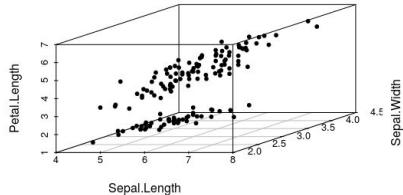
Diagramas de dispersión o scatter plot

```
> # Dos variables  
> iris_pet=iris[,c("Petal.Length","Petal.Width")]  
> plot(iris_pet, pch=20,xlab="Largo",ylab="Ancho")
```



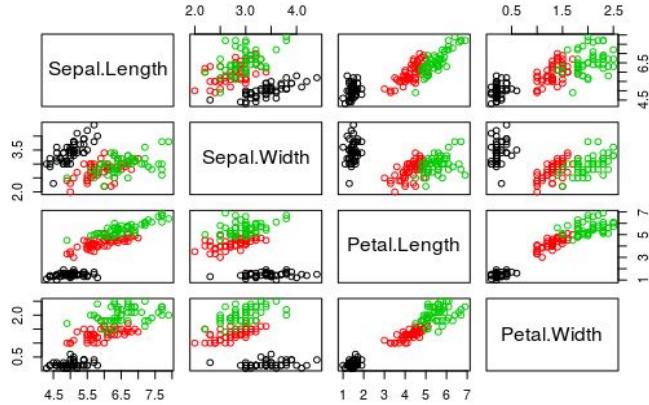
scatterplot3d(iris[,1:3],pch=20)

3 variables



o bien la función pairs

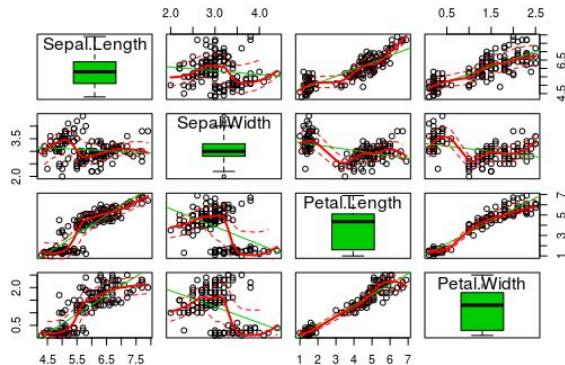
plot(iris[,1:4],col=iris\$Species)



spm está en el paquete cars.

En la diagonal se pueden dibujar boxplots, histograms, etc.

spm(iris[,1:4],diagonal="boxplot")



Datos cuantitativos agrupados: de cuantitativos a cualitativos. Clases k

¿Cuándo agrupar?

- Datos continuos que de hecho son redondeados (el redondeo supone agrupamiento)
- Datos discretos, y un número enormes de valores.
- Muchos datos y tenemos que estudiar sus frecuencias (>30 ?). El agrupamiento posiblemente nos dará más información que los valores uno por uno.

¿Cómo agrupar?

1. Se define el número de intervalos: número **k** de clases.
2. Decido la amplitud
3. Calculo los extremos de los intervalos
4. Calculo un valor representativo de su intervalo..

Defino tiempo de reacción al alérgeno

```
alergia=c(10.5, 11.2, 9.9, 15.0, 11.4, 12.7,  
        11.7, 8.4, 12.5, 11.2, 9.1, 10.4,  
        10.9, 9.8, 12.9, 9.9)
```

```
> # calculo k en cada caso  
> n=length(alergia)  
> k.sqrt = ceiling(sqrt(n))  
> k.sturges = ceiling(1+log(n,2))  
> # scott  
> As=3.5*sd(alergia)*n^{-1/3}  
> k.scott = ceiling(diff(range(alergia))/As)  
> # Freedman-Diaconis  
> AFD=2*IQR(alergia)*n^{-1/3}  
> k.FD = ceiling(diff(range(alergia))/AFD)  
> cat("k raíz cuadrada:",k.sqrt)  
k raíz cuadrada: 7
```

```
> cat("k scott:",k.scott)  
k scott: 5  
> cat("comparo con nclass.scott:",nclass.alergia)  
comparo con nclass.scott: 5
```

```
> cat("k sturges:",k.sturges)  
k sturges: 7  
> cat("comparo con nclass.Sturges:",nclass.Sturges(alergia))  
comparo con nclass.Sturges: 7  
> cat("k FD: ",k.FD)  
k FD: 8  
> cat("comparo con nclass.FD:",nclass.FD(alergia))  
comparo con nclass.FD: 8
```

1º) Métodos de elección de las clases o número de intervalos	
Raíz cuadrada: $k = \lceil \sqrt{n} \rceil$	Regla de Sturges: $k = \lceil 1 + \log_2(n) \rceil$
Regla de Scott: $A_s = 3.5 \cdot \tilde{s} \cdot n^{-\frac{1}{3}}$ $k = \left\lceil \frac{\max(x) - \min(x)}{A_s} \right\rceil$	Regla de Freedman-Diaconis: $A_{FD} = 2 \cdot (Q_{0.75} - Q_{0.25}) \cdot n^{-\frac{1}{3}}$ $k = \left\lceil \frac{\max(x) - \min(x)}{A_{FD}} \right\rceil$

Datos cuantitativos agrupados: amplitud, intervalos, marcas de clase

2º) Elijo Amplitud

- Calculo la diferencia entre máximo y mínimo con `diff(range(alergia))`
- Y divido entre `k`, redondeando al alza el valor de la precisión de los datos.

3º) Extremos intervalos

- Los intervalos tienen que ser cerrados a la izquierda y abiertos por la derecha (teoría de probabilidades).
- En todo caso, con la construcción siguiente, los extremos nunca coincidirán con valores originales de la serie

$$[L_1, L_2), [L_2, L_3), \dots, [L_k, L_{k+1}) \quad \left\{ \begin{array}{l} L_1 = \min(x) - \frac{\text{precision}}{2} \\ L_i = L_1 + A \cdot (i - 1), i \in [1, k + 1] \end{array} \right.$$

4º) Marcas de clase

$$X_i = \frac{L_i + L_{i+1}}{2}$$

- Un valor del intervalo que sirve para identificar la clase
- Sirve para calcular algunos estadísticos
- Es el punto medio del intervalo: el error cometido al asignar un número de la serie por su intervalo es inferior a la mitad de la amplitud del mismo.

El objetivo ahora es crear una tabla, `data.frame`, con los intervalos, las marcas de clase X_i , su frecuencia (n_i), y las freq. acumuladas, relativas, y absolutas (N_i, f_i y F_i).

intervalos	X_j	n_j	N_j	f_j	F_j
$[L_1, L_2)$	X_1	n_1	N_1	f_1	F_1
$[L_2, L_3)$	X_2	n_2	N_2	f_2	F_2
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
$[L_k, L_{k+1})$	X_k	n_k	N_k	f_k	F_k

```
Elijo método sturges; k=: 7  
> # p = precisión. Preguntarla  
> # Depende de la precisión de los datos  
> p = 0.1  
> # Amplitud: rango entre clases  
> A=diff(range(alergia))/k  
> A=ceiling(A/p)*p  
> A  
[1] 1.9
```

```
> # Calculo de los extremos de los intervalos  
> L1=min(alergia)-p/2  
> L=L1+A*(1:(k+1)-1)  
> L  
[1] 3.75 5.65 7.55 9.45 11.35 13.25 15.15  
[8] 17.05
```

```
> # Calculo las marcas del intervalo  
> X1=(L[2]+L[1])/2  
> X=X1+A*(1:k-1)  
> # o bien de forma alternativa  
> Xalt=(L[1:k]+L[2:(k+1)])/2  
> cat("1ª Forma; ",X)  
1ª Forma; 4.7 6.6 8.5 10.4 12.3 14.2 16.1  
> cat("2ª Forma; ",Xalt)  
2ª Forma; 4.7 6.6 8.5 10.4 12.3 14.2 16.1
```

Datos cuantitativos agrupados: cut

- ❑ Al pasar de datos cuantitativos a factor (agrupar), los niveles de este factor son cada una de las clases en las que hemos agrupado los valores. Las etiquetas de estos valores se pueden codificar de tres maneras:
 - ❑ A) Los intervalos... [0,5] [5,7] [7,9] [9,10] // B) Las marcas de clase...: 2.5 , 6, 8, 9.5 o (C) el número de orden: 1, 2, 3, 4
- ❑ Todo este proceso se hace con la orden **cut**

- cut** {
- ❑ **cut** (x, breaks=..., labels=..., right=..., include.lowest=....)
 - ❑ **x** es el vector numérico // **breaks** igualar a un vector con los extremos del intervalo o a un número, para que **cut** calcule los intervalos. En este último caso, aunque sea internamente, el extremo inferior y superior están un poco desplazados de sus valores: la orden **cut** no asegura que algún valor coincida con algún extremo.
 - ❑ **labels** establece las etiquetas; si **labels** no se utiliza, usa la codificación A; si **labels=FALSE** la codificación C.
 - ❑ **right=FALSE** establece los intervalos cerrados por la izquierda y abiertos por la derecha.
 - ❑ **include.lowest=TRUE** (combinado con **right=FALSE**), cierra el último intervalo por la derecha también **[L_k,L_{k+1}]**

A

```
> # intervalos de alergia  
> alergia_int=cut(alergia,breaks=L,right=FALSE)  
> levels(alergia_int)  
[1] "[3.75,5.65]" "[5.65,7.55]" "[7.55,9.45]"  
[4] "[9.45,11.4)" "[11.4,13.2)" "[13.2,15.2)"  
[7] "[15.2,17.1)"
```

```
> # definidos por sus marcas  
> alergia_marcas=cut(alergia,breaks=L,labels=X,right=FALSE)  
> levels(alergia_marcas)  
[1] "4.7" "6.6" "8.5" "10.4" "12.3" "14.2"  
[7] "16.1"
```

```
> # definidos por su orden  
> # En este caso es un vector numérico  
> # no un factor  
> alergia_num=cut(alergia,breaks=L,labels=FALSE,right=FALSE)  
> alergia_num  
[1] 4 4 4 6 5 5 7 4 5 5 5 2 3 3 4 3 1 4 5 3 5  
[22] 4 3 4 3 6 5 2 5 3 2 3 6 6 5 5 4 4 5 4
```

B

**Ejemplo de cut,
automáticamente
elige los
intervalos**

```
> # simplemente en 3 grupos  
> x_tres=cut(x,breaks=3,right=FALSE)  
> x_tres  
[1] [7,10) [7,10) [7,10) [7,10)  
[5] [0.991,4) [4,7) [4,7) [7,10)  
[9] [7,10) [4,7) [0.991,4) [0.991,4)  
[13] [0.991,4) [0.991,4) [0.991,4)  
Levels: [0.991,4) [4,7) [7,10)
```

C

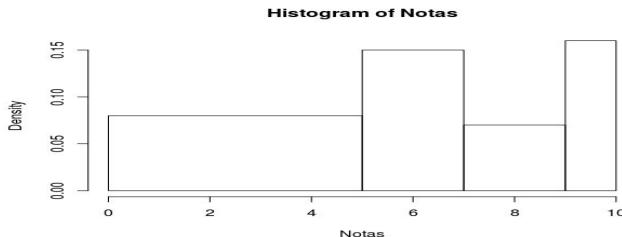
Datos cuantitativos agrupados: hist

- hist(x, breaks=..., right=FALSE, plot=FALSE, [include.lowest=TRUE])
- Esta función dibuja un histograma pero además es un list que devuelve varios resultados: count → frec. absolutas, mids→ marcas, etc.

Ejemplo Notas: intervalos de amplitud desigual [0,5)[5,7)[7,9)[9,10] // incluye diez.

```
Notas=c(5.1,1.1,6.4,5.3,10,5.4,1.9,3.1,5.1,0.8,9.6,6.6,7.0,9.6,  
10.1,2.4,2.8,8.8,2.4,1.8,5.6,6.8,6.7,2.2,8.6,3.9,5.6,5.9,8.4,4.9,  
0.7,8.2,3.7,4.8,5.8,3.3,9.7,7.8,9.3,4.5,6.2,3.9,4.7,6.2,6.3,9.4,  
9.3,2.3,8.5,1.4)  
> Notas_cut=cut(Notas,breaks=c(0,5,7,9,10),ri  
ght=FALSE,include.lowest=10,labels=c("Ins","A  
pr","Not","Sob"))  
> Notas_cut  
[1] Apr Ins Apr Apr Sob Apr Ins Ins Apr  
[10] Ins Sob Apr Not Sob Sob Ins Ins Not  
[19] Ins Ins Apr Apr Apr Ins Not Ins Apr  
[28] Apr Not Ins Ins Not Ins Ins Apr Ins  
[37] Sob Not Sob Ins Apr Ins Ins Apr Apr  
[46] Sob Sob Ins Not Ins  
Levels: Ins Apr Not Sob
```

```
> # tablas de frecuencia  
> t1=table(Notas_cut)  
> t1  
Notas_cut  
Ins Apr Not Sob  
20 15 7 8  
> cumsum(t1)  
Notas_cut  
Ins Apr Not Sob  
20 35 42 50  
> prop.table(t1)  
Notas_cut  
Ins Apr Not Sob  
0.40 0.30 0.14 0.16  
> cumsum(prop.table(t1))  
Notas_cut  
Ins Apr Not Sob  
0.40 0.70 0.84 1.00
```



o bien

```
> h1=hist(Notas,breaks=c(0,5,7,9,10),right=FALSE,inc  
lude.lowest=TRUE,plot=FALSE)  
> frec_abs = h1$counts  
> frec_rel=frec_abs/length(Notas)  
> frec_abs_acu=cumsum(frec_abs)  
> frec_rel_acu=frec_abs_acu/length(Notas)  
> marcas=h1$mid  
> intervalos=cut(Notas,breaks=c(0,5,7,9,10),right=FA  
LSE,include.lowest=TRUE)  
> # defino data frame  
> NotasDF=data.frame(levels(intervalos),marcas,frec_  
abs,frec_abs_acu,frec_rel,frec_rel_acu)  
> NotasDF  
  levels.intervalos. marcas frec_abs frec_abs_acu frec_rel frec_rel_acu  
1 [0,5) 2.5 20 20 0.40 0.40  
2 [5,7) 6.0 15 35 0.30 0.70  
3 [7,9) 8.0 7 42 0.14 0.84  
4 [9,10] 9.5 8 50 0.16 1.00
```

```
h2=hist(Notas,breaks=c(0,5,7,9,10),right=FALSE,include.lowest=TRUE)
```

Datos cuantitativos agrupados: automatizo

```
x=c(10, 9, 8, 7, 3, 5, 6, 8, 9, 5, 2, 1, 3, 1, 1)
# defino los intervalos
k = 3 # sin indicar el método
p = 1 # precisión igual a la unidad
# calculo la amplitud
A=diff(range(x))/k
# redondeo A en exceso el valor de la precisión
if (p<1) {A=ceiling(A/p)*p} else {A=A+1}

=====
# Función sabiendo x,A,p,y k
=====
Tabla_frec_agrupadas=function(x,p,A,k) {
  L=min(x)-p/2+A*(1:(k+1)-1)
  x_cut=cut(x,breaks=L,right=FALSE)
  intervalos=levels(x_cut)
  marcas=(L[1:k]+L[2:(k+1)])/2
  fabs=as.vector(table(x_cut))
  frel=fabs/length(x)
  fabsacu=cumsum(fabs)
  frelacu=cumsum(frel)
  table.x=data.frame(intervalos,marcas,fabs,fabsacu,frel,frelacu)
  table.x
}

tabla=Tabla_frec_agrupadas(x,p,A,k)
tabla
intervalos marcas fabs fabsacu frel frelacu
[0,5,4,5) 2.5 6 6 0.4 0.4
[4,5,8,5) 6.5 6 12 0.4 0.8
[8,5,12,5) 10.5 3 15 0.2 1.0
```

dig.lab=3 , si quiero cifras con 3 decimales en los intervalos. En cut

```
#=====
# Función sabiendo x,A,p,y k
#=====
Tabla_frec_agrupadas=function(x,p,A,k,dig.lab=3) {
  L=min(x)-p/2+A*(1:(k+1)-1)
  x_cut=cut(x,breaks=L,right=FALSE,dig.lab=dig.lab)
  intervalos=levels(x_cut)
  #=====
```

```
#=====
# Función sabiendo x,L, V=TRUE si el último intervalo es cerrado
#=====

Tabla_frec_agrupadas_L = function(x,L,V=FALSE) {
  x_cut=cut(x,breaks=L,include.lowest=V)
  intervalos=levels(x_cut)
  k=length(L)-1
  marcas=(L[1:k]+L[2:(k+1)])/2
  fabs=as.vector(table(x_cut))
  frel=fabs/length(x)
  fabsacu=cumsum(fabs)
  frelacu=cumsum(frel)
  table.x=data.frame(intervalos,marcas,fabs,fabsacu,frel,frelacu)
  table.x
}

> Notas
[1] 5.1 1.1 6.4 5.3 10.0 5.4 1.9 3.1
[9] 5.1 0.8 9.6 6.6 7.0 9.6 10.0 1.2
[17] 4.2 8.8 2.4 1.8 5.6 6.8 6.7 2.2
[25] 8.6 3.9 5.6 5.9 8.4 4.9 0.7 8.2
[33] 3.7 4.8 5.8 3.3 9.7 7.8 9.3 4.5
[41] 6.2 3.9 4.7 6.2 6.3 9.4 9.3 2.3
[49] 8.5 1.4
> Tabla_frec_agrupadas_L(Notas,c(0,5,7,9),V=TRUE)
   intervalos marcas fabs fabsacu frel
1 [0,5] 2.5 20 20 0.40
2 (5,7] 6.0 16 36 0.32
3 (7,9] 8.0 6 42 0.12
frelacu
1 0.40
2 0.72
3 0.84

> alergia
[1] 10.5 11.2 9.9 15.0 11.4 12.7 16.5 10.1 12.7 11.4 11.6 6.2 7.9 8.3
[15] 10.9 8.1 3.8 10.5 11.7 8.4 12.5 11.2 9.1 10.4 9.1 13.4 12.3 5.9
[29] 11.4 8.8 7.4 8.6 13.6 14.7 11.5 11.5 10.9 9.8 12.9 9.9
> k=7
> A=diff(range(alergia))/K
> p=0.1
> A=ceiling(A/p)*p
> A
[1] 1.9
> Tabla_frec_agrupadas(alergia,p,A,k)
   intervalos marcas fabs fabsacu frel frelacu
1 [3.75,5.65) 4.7 1 1 0.025 0.025
2 [5.65,7.55) 6.6 3 4 0.075 0.100
3 [7.55,9.45) 8.5 8 12 0.200 0.300
4 [9.45,11.4) 10.4 11 23 0.275 0.575
5 [11.4,13.2) 12.3 12 35 0.300 0.875
6 [13.2,15.2) 14.2 4 39 0.100 0.975
7 [15.2,17.1) 16.1 1 40 0.025 1.000
```

```
#=====
# Función sabiendo x,A,p,y k
#=====
Tabla_frec_agrupadas=function(x,p,A,k,dig.lab=3) {
  L=min(x)-p/2+A*(1:(k+1)-1)
  x_cut=cut(x,breaks=L,right=FALSE,dig.lab=dig.lab)
  intervalos=levels(x_cut)
  #=====
```

Datos cuantitativos agrupados: estadísticos

- No es lo ideal hacerlo de datos agrupados, pero si tenemos que hacerlo...
- Si tenemos **k** clases con sus respectivas marcas X_1, X_2, \dots, X_k

Cálculo estadístico

```

> anno1981 <- read.csv("~/R/Leccion 11/anno1981.csv", sep=";")
> # Antes que nada, elijo las marcas
> # Entre 0 y 4 años tiene de límite 5... [0,5)
> # Luego la primera marca es  $X_1=2.5$ ; y la amplitud es 5
> n=dim(anno1981)[1]
> X=2.5+(1:(n-1)-1)*5
> X=c(X,90) # La última marca es de 90
> anno1981$marcas=X
> # Cálculo estadístico
> Total=sum(anno1981$Población)
> # Medias
> Media=sum(anno1981$Población*anno1981$marcas)/Total
> Vari=sum(anno1981$Población*anno1981$marcas^2)/Total - Media^2
> Desv=sqrt(Vari)
> # Tablas de frecuencia
> anno1981$FA=cumsum(anno1981$Población)
> anno1981$Frel=round(anno1981$Población/Total,3)
> anno1981$FrelAcu=round(cumsum(anno1981$Frel),3)

> head(anno1981)
   Edades Población marcas      FA    Frel FrelAcu
1  De 0 a 4 años  3075352    2.5  3075352  0.082   0.082
2  De 5 a 9 años  3308049    7.5  6383401  0.088   0.170
3  De 10 a 14 años 3302328   12.5  9685729  0.088   0.258
4  De 15 a 19 años 3263312   17.5 12949041  0.087   0.345
5  De 20 a 24 años 2942178   22.5 15891219  0.078   0.423
6  De 25 a 29 años 2537428   27.5 18428647  0.067   0.490

```

$$\tilde{s}^2 = \frac{n}{n-1} \cdot s^2$$

$$\bar{x} = \sum_{i=1}^k n_i \cdot X_i \quad s^2 = \left(\frac{1}{n} \cdot \sum_{i=1}^k n_i \cdot X_i^2 \right) - \bar{x}^2$$

- **intervalo crítico para la mediana**, el primer intervalo cuya frecuencia relativa sea mayor o igual que 0.5
- Estimamos la mediana de los datos reales M como

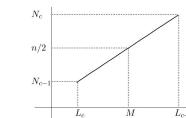
→ $[L_c, L_{c+1}]$ int. crítico

→ $A_c \rightarrow$ amplitud

→ $N_{c-1} \rightarrow$ frec.. abs. acumulada anterior al int. crítico

→ $n_c \rightarrow$ frecuencia absoluta del intervalo crítico

$$M = L_c + A_c \cdot \frac{\frac{n}{2} - N_{c-1}}{n_c}$$



```

> # Encontrar estimación para la mediana
> # Intervalo crítico para la mediana: DONDE POR PRIMERA VEZ FRELACU
> # sea mayor que 0.5
> Fila = which(anno1981$FrelAcu>0.5)[1] # el primer valor del vector que cumple
que es mayor que 0.5
> Lc=anno1981$marcas[Fila]-2.5 # 2.5 es la mitad de la amplitud
> Lcmas1=anno1981$marcas[Fila]+2.5
> nc=anno1981$Población[Fila] # frecuencia absoluta en ese intervalo
> Nmenos1=anno1981$FA[Fila-1] # frecuencia acumulada del intervalo ANTERIOR
> Ac=Lcmas1-Lc # Amplitud
> M = Lc+(Ac/nc)*(Total/2-Nmenos1)
> # M cumple que la frecuencia abs acumulada para ese m es la mitad de la población
> cat("Estimación de la edad que alcanzaba la primera mitad de la población española", M, "años")
Estimación de la edad que alcanzaba la primera mitad de la población española 3
0.8411 años

```

Para los cuantiles

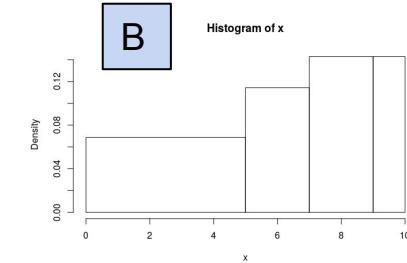
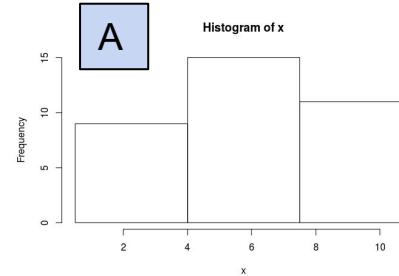
$$Q_p = L_c + A_c \cdot \frac{p \cdot n - N_{c-1}}{n_c},$$

Datos cuantitativos agrupados: histogramas

De frecuencias absolutas

La frecuencia absoluta de cada intervalo es igual al área del rectángulo en su histograma

- A. Si la amplitud del intervalo **es cte**, la **frec. absoluta es proporcional a la altura de la barra**
- B. Si no es cte, no lo es.



De frecuencias acumuladas (abs. o rel.): las barras crecen independientemente de su amplitud, y sus frecuencias acumuladas son iguales a las alturas de las barras

De frecuencias relativas

La frecuencia relativa es el área de la barra. La suma de todas las áreas es 1.
En este contexto las alturas de las barras son densidades

`hist(x, breaks=..., freq=...., right)`

- `h1$counts` → frec. absolutas
- `h1$breaks` → intervalos
- `h1$mid` → marcas
- `h1$density` → densidades = frecuencia relativa entre su amplitud

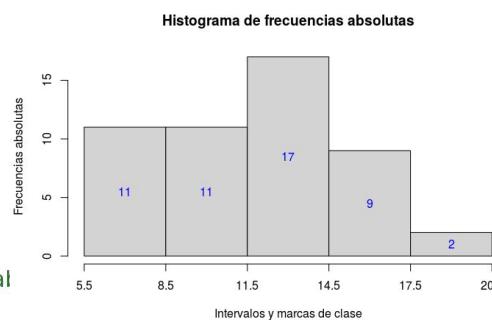
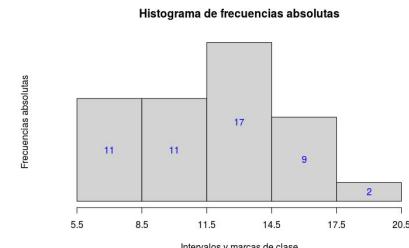
- breaks** es el parámetro equivalente a `cut`. Define los intervalos. Se puede indicar el número de clases, o un vector, e incluso entre comillas el método “Scott”, “Sturges”.
- `freq=TRUE` (por defecto) → **frec. absolutas si A = cte y freq. relativas si A ≠ cte** // `freq=FALSE` → **siempre freq. relativas**
- `right` e `include.lowest` como en `cut`
- `plot=FALSE` no dibuja la gráfica.
- Otros parámetros usuales de la función `plot`.

Datos cuantitativos agrupados: histogramas de frecuencias absolutas

```
# =====
# Histograma de frecuencias absolutas
# =====
hist_abs = function (x,L) {
  # calculo histograma freq=FALSE (siempre freq. relativas)
  h=hist(x,breaks=L,right=FALSE,freq=FALSE,
  axes=FALSE, col="lightgray",
  main="Histograma de frecuencias absolutas",
  xlab="Intervalos y marcas de clase",
  ylab="Frecuencias absolutas")
  # Como axes=FALSE no dibujaba los ejes; ahora pongo el de
  axis(1,at=L) # 1--> eje X y L, las marcas de clase
  # pongo textos con las frecuencias absolutas
  text(h$mid,h$density/2,labels=h$counts,col="blue")
}

# =====
# Histograma de frecuencias absolutas 2
# =====
hist_abs2 = function (x,L) {
  # calculo histograma freq=TRUE
  h=hist(x,breaks=L,right=FALSE,freq=TRUE,
  xaxt="n", col="lightgray", # quito el eje x solo
  main="Histograma de frecuencias absolutas",
  xlab="Intervalos y marcas de clase",
  ylab="Frecuencias absolutas")
  # Como axes=FALSE no dibujaba los ejes; ahora pongo el de al
  axis(1,at=L) # 1--> eje X y L, las marcas de clase
  # pongo textos con las frecuencias absolutas
  text(h$mid,h$counts/2,labels=h$counts,col="blue")
}
```

- ❑ **axes=FALSE** → no dibuja ejes.
Alternativamente xaxt="n" y yaxt="n" (por separado)
- ❑ **axis(i, at=L)** → añado eje de abscisas con datos de L
- ❑ Aunque hemos dibujado el histograma de freq. relativas da igual al quitarle el eje Y; los datos los ponemos como texto. Por eso **freq=FALSE**. En un diagrama así, la altura es siempre h\$density con lo que en su mitad hemos puesto los textos.



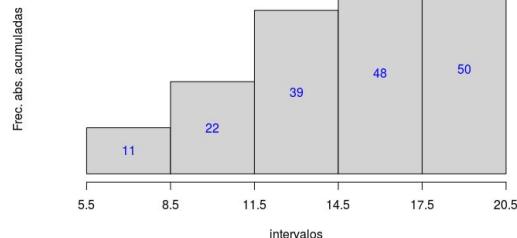
con marcas → rug
rug (vector)
rug(jitter(vector))



Datos cuantitativos agrupados: hist. de freqs. absolutas *acumuladas*

```
# =====
# Histograma de frecuencias absolutas acumuladas
# =====
hist_abs_acu = function(x,L) {
  h = hist(x, breaks=L, right=FALSE, plot=FALSE) # no lo dibujo aún
  h$density=cumsum(h$density) # igualo la densidad a la densidad acumulada
  plot(h, freq=FALSE, axes=FALSE, col="lightgray",
    main="Histograma de frecuencias absolutas acumuladas",
    xlab="intervalos", ylab="Frec. abs. acumuladas")
  axis(1, at=L)
  text(h$mids, h$density/2, labels=cumsum(h$counts), col="blue")
}
```

Histograma de frecuencias absolutas acumuladas



Datos cuantitativos agrupados: densidades

- **Densidad:** es una curva tal que el área de la misma en un intervalo es igual a la fracción de individuos de la población que caen dentro de ese intervalo.
- **Variable con distribución normal** → densidad tipo campana de Gauss
- La manera más fácil de estimar una densidad es con la función **density**

```
plot(density(fruta), type="l", main="Densidad de \'fruta\'", xlab="Número de árboles", ylab="Densidad")
```

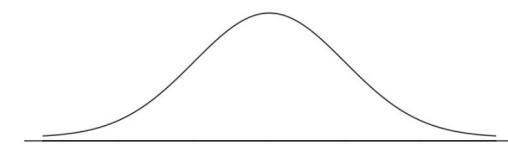
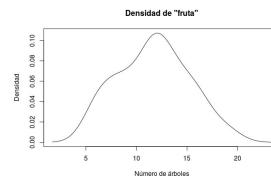
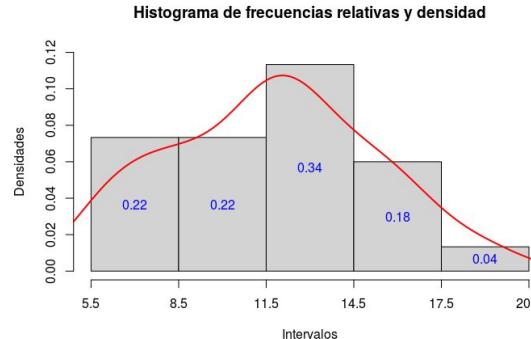


Figura 11.11. Campana de Gauss.

Datos cuantitativos agrupados: hist. de freqs. relativas

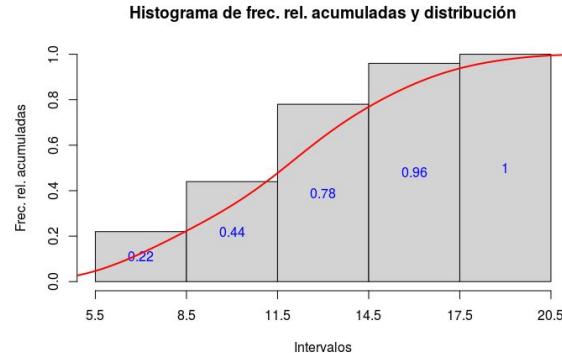
```
# =====
# Histograma de frecuencias relativas
# =====
hist_rel=function(x,L) {
  h = hist(x,breaks=L,right=FALSE,plot=FALSE) # no lo dibujo aún
  # Calculo un máximo como o bien la mayor densidad del intervalo h$density
  # o la mayor de la función density(x)
  t=round(1.1*max(h$density,max(density(x)[[2]])),2)
  plot(h, freq=FALSE, xaxt="n",ylim=c(0,t), col="lightgray",
    main="Histograma de frecuencias relativas y densidad",
    xlab="Intervalos", ylab="Densidades")
  axis(1,at=L)
  text(h$mid,h$density/2,labels=round(h$counts/length(x),2),col="blue")
  lines(density(x),col="red",lwd=2)
}
```



Datos cuantitativos agrupados: hist. de freqs. relativas *acumuladas*

Función de distribución:
una función que, en cada punto, la fracción de individuos que caen a la izquierda de ese punto.

```
# =====
# Histograma de frecuencias relativas acumuladas
# =====
hist_rel_acu=function(x,L) {
  h = hist(x,breaks=L,right=FALSE,plot=FALSE) # no lo dibujo aún
  h$density=cumsum(h$counts)/length(x) #calculo las frecuencias rel
  plot(h, freq=FALSE, xaxt="n", col="lightgray",
    main="Histograma de freq. rel. acumuladas y distribución",
    xlab="Intervalos", ylab="Frec. rel. acumuladas")
  axis(1,at=L)
  text(h$mid,h$density,labels=round(h$density,2),col="blue")
  dens.x=density(x)
  dens.x$x=cumsum(dens.x$x)*(dens.x$x[2]-dens.x$x[1])
  lines(dens.x,col="red",lwd=2)
}
```

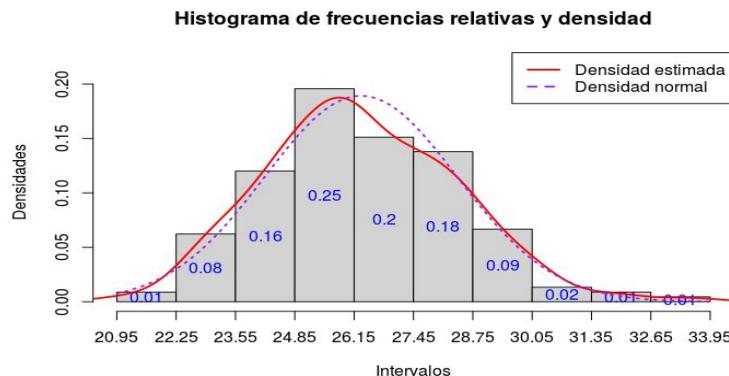


Datos cuantitativos agrupados: distribución normal teórica

`dnorm(x,mean(datos),sd(datos))` → hay que indicar la **media** de los datos y su **desviación típica muestral**

`hist_rel(crw,L)`

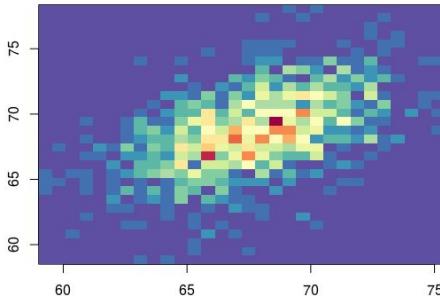
```
# añado la curva de distribución normal teórica, para saber cómo
# se ajusta a una campana de Gauss. Del vector de datos original
curve(dnorm(x,mean(crw),sd(crw)),col="purple",lwd=2,lty=3,add=TRUE)
legend("topright",lwd=c(2,2),lty=c(1,2),col=c("red","purple")
      ,legend=c("Densidad estimada","Densidad normal"))
```



Datos cuantitativos agrupados: gráficas multidimensionales

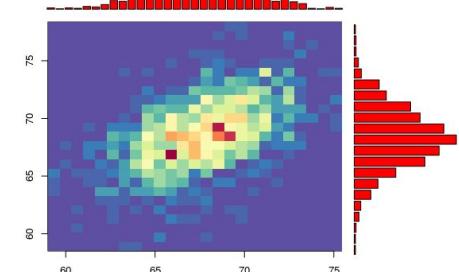
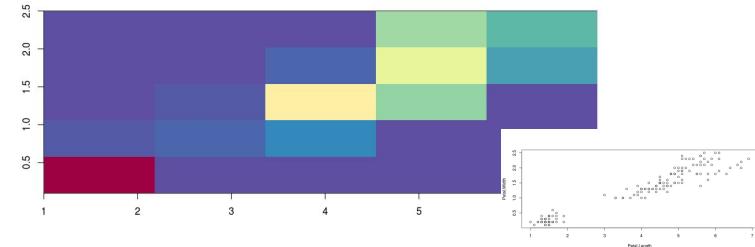
- Cargo el paquete **gplots**. Lo tengo que descargar de <https://cran.r-project.org/web/packages/gplots/index.html>
- Instalo el paquete **RColorBrewer** → para incluir colores
- En el ejemplo, rojo → más coincidencia // azules menos

```
> hist2d(iris$Petal.Length,iris$Petal.Width,nbins=c  
(nclass.FD(iris$Petal.Length),nclass.FD(iris$Petal.  
Width)),col=rev(colorRampPalette(brewer.pal(11,"Spe  
ctrual"))(50)))
```



pearson

```
hist_doble = function (df,n) {  
  par.anterior=par()  
  h1=hist(df[,1], breaks=n,plot=FALSE)  
  h2=hist(df[,2], breaks=n,plot=FALSE)  
  m = max(h1$counts,h2$counts)  
  par(mar=c(3,3,1,1))  
  layout(matrix(c(2,0,1,3),nrow=2,byrow=TRUE),  
         heights = c(1,3), widths = c(3,1))  
  hist2d(df,nbins = n,  
         col=rev(colorRampPalette(brewer.pal(11,"Spectral"))(50)))  
  par(mar=c(0,2,1,0))  
  barplot(h1$counts,axes=FALSE,ylim=c(0,m),col="red")  
  par(mar=c(2,0,0.5,1))  
  barplot(h2$counts,axes=FALSE,xlim=c(0,m),col="red",horiz = TRUE)  
  par.anterior  
}
```



Nubes de palabras (word cloud)

- ❑ Cargar paquete **tm** y **wordcloud** (Había que actualizar R - ver 2^a diapositiva -).
 - ❑ Crear un Corpus volátil con VCorpus (representación volátil de varios textos)
 - ❑ Ejemplo: **Prueba=Vcorpus(lugar, readerControl=....)**
 - ❑ **lugar** es desde donde se cargan los textos a analizar; **DirSource("directorio", encoding=...)** o **URIsource("url", encoding=...)**
 - ❑ **readerControl** es un list con **reader** y **language**
 - ❑ **reader=readPlain** (formato de texto plano, por defecto). Se puede omitir en caso de .txt
 - ❑ **language="spa"** → en español Cumple código ISO 639-2
 - ❑ El objeto Prueba, al aplicarle **inspect(Prueba)**, devuelve la estructura del VCorpus.
 - ❑ Para acceder al texto en consola, escribir **writelines(as.character(Prueba[[1]]))**

```
> # =====
> # Proceso de minado de texto
> # =====
> Prueba=tm_map(Prueba,tolower) # a minúsculas
> Prueba=tm_map(Prueba,removePunctuation) # quitar signos de puntuación
> Prueba=tm_map(Prueba,removeWords, stopwords("spanish")) # quitamos las palabras comunes
> Prueba=tm_map(Prueba,removeWords, c("artículo")) # quitamos la palabra artículo
> Prueba=tm_map(Prueba,stripWhitespace) # quitamos espacios
> # generamos el wordcloud
> wordcloud(Prueba[[2]], scale=c(3,0.5), max.words=100, rot.per = 0.25, colors=brewer.pal(8,"Dark2"))
```



wordcloud

scale → tamaño de la palabra más repetida, tamaño de la que menos **max.words** → nº max. palabras **min.freq** → frecuencia mínima necesaria para que una palabra aparezca en el texto

rot.per → fraccion de palabras que giran.

colors → paleta de colores a usar
random.order = FALSE → aparecen las palabras por orden de frecuencia, de mayor a menor.

Puede haber warnings por solapamiento.

Heatmaps

Rowv=NA y **Colv=NA** evitan dibujar dendogramas.

revC=TRUE → el orden de las variables de izda a derecha == orden de arriba abajo.

```
# datos de mtcars menos 8 y 9
```

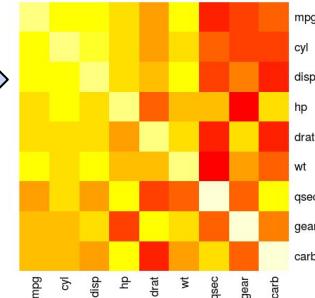
```
mtcars2=mtcars[, -c(8, 9)]
```

```
mtcars2cor = round(abs(cor(mtcars2)), 3)
```

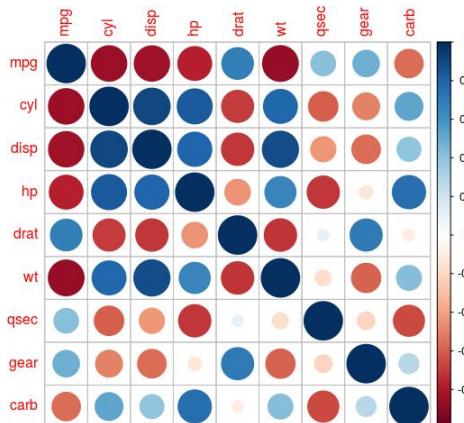
```
# mtcars2cor
```

```
heatmap(mtcars2cor, Rowv=NA, Colv=NA, revC=TRUE)
```

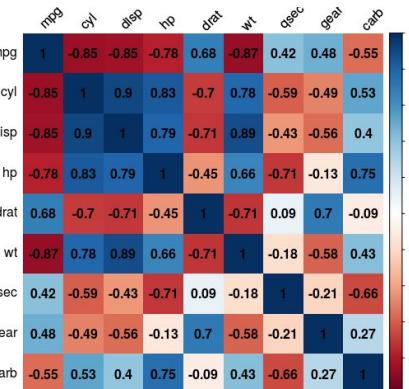
A más claro, más relacionado



```
# Paquete corrplot  
detach("package:corrplot", unload=TRUE)  
library("corrplot", lib.loc="~/R/i686-pc-linu  
mtcars3cor = round(cor(mtcars2), 3) # sin valo  
corrplot(mtcars3cor)
```

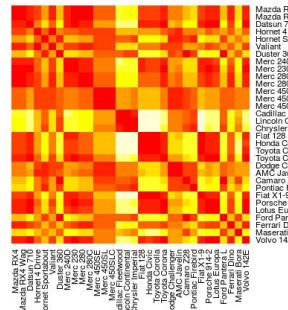


```
# otra gráfica  
corrplot(mtcars3cor, method="shade" # método de sombreado  
, shade.col=NA # homogéneamente sombreadas  
, tl.col="black" # color de las etiquetas  
, tl.srt=45 # etiquetas de columnas a 45º  
, addCoef.col = "black") # añado coef. de correlación|
```



Heatmaps

```
# matriz de las distancias euclídeas
# entre los valores del coche
dist.mtcars=as.matrix(dist(mtcars))
heatmap(dist.mtcars,margins=c(9,9)
       ,symm=TRUE, Rowv=NA, Colv=NA, revC=TRUE)
```



En este caso, a más rojo, más “cero” indica, por lo tanto la distancia euclídea es más pequeña y más parecidos son.

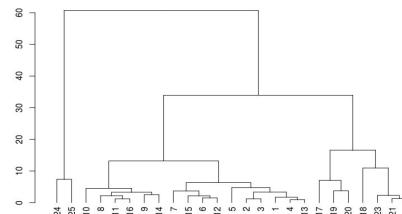
symm=TRUE → Es simétrica.

margins=c(9,9) → tamaño para las etiquetas

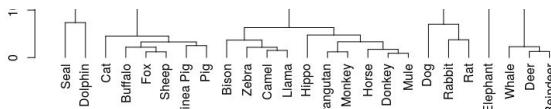
Dendogramas: clasifico por grupos con una característica (distancia) parecida

- Algoritmos de clustering (agrupamiento) jerárquico: **método de enlace completo**.

```
# Paquete cluster.datasets
library("cluster.datasets")
data("all.mammals.milk.1956") # cargo bases de datos específicas
AMM=all.mammals.milk.1956
dist.AMM=dist(AMM[, -c(1)]) # distancia euclídea
round(as.matrix(dist.AMM),3)[1:6,1:6]
dend.AMM=as.dendrogram(hclust(dist.AMM)) # defino un dendograma
# calculo el clustering jerárquico con hclust
# y uso as.dendrogram para crear una variable dendograma
plot(dend.AMM) # y la月 dibujo
```



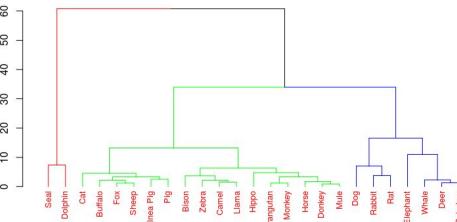
```
# con los nombres
# Hay que modificar las denominaciones de filas
AMM.nombres=AMM
rownames(AMM.nombres)=AMM.nombres$name
AMM.nombres=AMM.nombres[, -1]
plot(as.dendrogram(hclust(dist(AMM.nombres))))
```



Dendogramas:

```
set(dendrograma, what=""",valor) →
    provisional
dendrograma=set(dendrograma,wha-
t=""", valor) → permanente
```

Dendograma con etiquetas y reducidas

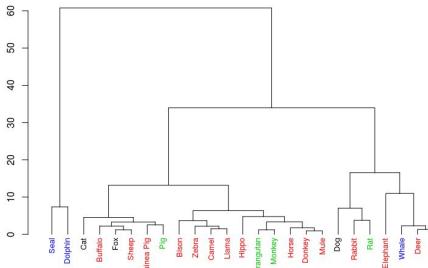


```
# Modifico los labels y dibujo con colores
# Paquete dendextend
detach("package:dendextend", unload=TRUE)
library("dendextend", lib.loc="~/R/i686-pc-linux-gnu-library/3.1")
L=labels(dend.AMM)
labels(dend.AMM)=AMM$name[L]
labels(dend.AMM)
# método de concatenación solo sirve para ciertos paquetes
# %>%>%
dend.AMM%>%
  set.what="labels_col", "red")%>% # color etiquetas
  set.what="labels_cex",0.8)%>% # tamaño de las etiquetas
  # colores de las ramas
  set.what="branches_k_color",c("red","green","blue"),k=3)%>%
plot(main="Dendograma con etiquetas y reducidas")
```

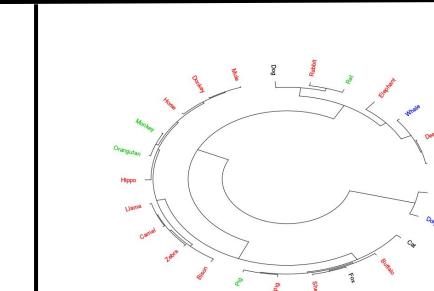
Colores por factor

```
> AMM$diet
[1] "H" "O" "O" "H" "H" "H" "H" "H" "H" "C" "C" "H" "H" "O" "H"
[16] "H" "C" "H" "H" "O" "H" "H" "P" "P" "P"
> head(AMM,3)
  name water protein fat lactose ash diet
1  Horse  90.1     2.6  1.0     6.9 0.35   H
2 Orangutan 88.5     1.4  3.5     6.0 0.24   O
3  Monkey  88.4     2.2  2.7     6.4 0.18   O
#####
# Clasificando según factores
# =====
AMM$diet= factor(AMM$diet)
AMM$diet
[1] H O O H H H H H C C H H O H H C H H O H H P P P
Levels: C H O P
head(AMM,3)
  name water protein fat lactose ash diet
  Horse  90.1     2.6  1.0     6.9 0.35   H
Orangutan 88.5     1.4  3.5     6.0 0.24   O
  Monkey  88.4     2.2  2.7     6.4 0.18   O
dend.AMM2=dend.AMM%>%
  set.what="labels_cex",0.8)%>%
  set.what="labels",AMM$name[L])%>%
  set.what="labels_col",as.numeric(AMM$diet[L]))
plot(dend.AMM2,main="Dendogramas de mamíferos según dieta")
```

Dendogramas de mamíferos según dieta



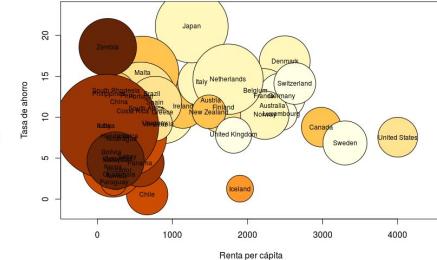
```
library("circlize")
circlize_dendrogram(dend.AMM2)
```



Burbujas: la importancia de la variable se indica con su tamaño.

- **symbols(x,y,figura=z)** → x e y coordenadas de las variables // **figura** → circles, squares, stars, etc.
- **z** → vector de los tamaños “lineales” de las figuras
- Además: **fg** (color borders), **bg** (color relleno) y los habituales de **plot**

```
# cargamos paquete faraway
library("faraway")
# agrupamos por porcentaje15. Datos savings
porcentaje15=as.numeric(cut(savings$pop15,9))
colores=brewer.pal(9,"YlOrBr")[porcentaje15]
symbols(savings$dpi,savings$sr,circles=sqrt(savings$ddpi),
        # raíz cuadrada representa mejor los datos|
        bg=colores,xlab="Renta per cápita",
        ylab="Tasa de ahorro")
# nombre de cada país en el centro de su burbuja
text(savings$dpi,savings$sr,rownames(savings),cex=0.75)
```



Streamgraphs: visualizar datos de forma conjunta

- instalo paquete **devtools** → para poder instalar con **install.github** (desde github)
- instalo el paquete **streamgraph** con **install_github("hrbrmstr/streamgraph")**
- instalo también para el ejemplo **ggplot2movies**
- Importante → escribir **library(streamgraph)**
- Utilizo las siguientes órdenes...
 - Del paquete **tidyverse** la función **gather()**
 - **gather(data.frame, nuevo factor, nueva variables numérica, variables agrupadas)** que uso para convertir un data frame de varias columnas en una sola, agrupando los valores.
 - **streamgraph(data frame, factor, valores, variable temporal, interactive=FALSE)**

Ubuntu: instalar

- **libssl-dev**
- **apt-get -y build-dep libcurl4-gnutls-dev**
- **apt-get -y install libcurl4-gnutls-dev**

Streamgraphs

```
# cargamos paquete streamgraphs
library("streamgraph")
library("ggplot2movies")
# reduzco el contenido de movies
data(movies)
movies.small=movies[,c(2,18:24)]
rm(movies)
# variables año , y otras
str(movies.small)
# cargo el paquete tidyR
library("tidyR")
# utilizo la función de agrupamiento gather
movies.agrup=gather(movies.small,genero,valor,-year)
# observo que menos del 0.1% son películas de antes de 1930, así
quantile(movies.agrup$year,0.001)
movies.agrup=movies.agrup[movies.agrup$year>=1930,]
# para cada año y género realizo la suma
sum.movies=aggregate(valor~year+genero,data=movies.agrup,FUN=sum)
str(sum.movies)
# dibujo ya el streamgraph
streamgraph(sum.movies,"genero","valor","year",interactive=FALSE)

# Dibujo en forma interactiva
s1=streamgraph(sum.movies,"genero","valor","year",interactive=TRUE)
sg_legend(s1,show=TRUE,label="Género: ")
```

