Contents

3º paso: análise previa. **1**

Táboas 2

Gráficos 5

Calculos de parámetros estatísticos 23

3º paso: análise previa.

Unha vez preparadas as variables tocaría comezar coa súa organización e presentación, mediante táboas e gráficas.

Para iniciar o coñecemento dos nosos datos podemos usar a función *summary()*. É unha función xenérica, que cando se aplica a un obxecto proporciona unha síntese das súas características, adaptándose ao tipo de obxecto.

Aplicada a un vector numérico produce *media aritmética*, *mediana*, *cuartis*, *máximo* e *mínimo*:

```
summary(leite)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.51   84.20  196.60  221.80  281.60  801.10
```

En cambio se o aplicamos a un data.frame obtense a mesma análise para cada columna numérica do data.frame e unha táboa resumida para os factores (ou vectores de caracteres)

```
summary(gastan)
```

```
##  leite.enteiro       leite.non           NUMERO          CCAA    TAMAMU
##  Min.   :  0.000   Min.   :  0.0000   Min.   :  298   12     :42   1:32
##  1st Qu.:  0.000   1st Qu.:  0.0235   1st Qu.: 4674   1      : 5   2:12
##  Median :  6.865   Median : 88.4759   Median :10290   13     : 5   3:10
##  Mean   : 79.194   Mean   :142.6323   Mean   :10031   2      : 4   4: 8
##  3rd Qu.:109.969   3rd Qu.:202.0846   3rd Qu.:14667   9      : 4   5:23
##  Max.   :550.375   Max.   :786.3917   Max.   :21821   14     : 4
##                                                       (Other):21
##      FACTOR           NMIEMB          NMIEM7            NMIEM8
##  Min.   : 179.4   Min.   :1.000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.: 480.6   1st Qu.:2.000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median : 637.2   Median :2.000   Median :0.00000   Median :0.0000
##  Mean   : 733.2   Mean   :2.659   Mean   :0.09412   Mean   :0.3059
##  3rd Qu.: 862.5   3rd Qu.:4.000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :2030.8   Max.   :6.000   Max.   :1.00000   Max.   :2.0000
##
##     NUMESTU            UC1        TIPHOGAR8     EDADSP      SEXOSP
##  Min.   :0.0000   Min.   :10.00   1:30      Min.   :24.00   1:61
##  1st Qu.:0.0000   1st Qu.:17.00   2:12      1st Qu.:48.00   6:24
##  Median :0.0000   Median :17.00   3:16      Median :58.00
##  Mean   :0.1765   Mean   :20.91   4:27      Mean   :59.34
```

```
##  3rd Qu.:0.0000   3rd Qu.:27.00              3rd Qu.:72.00
##  Max.   :3.0000   Max.   :45.00              Max.   :85.00
##
##     IMPEXAC           GASTMON
##  Min.   :     0   Min.   :  2549
##  1st Qu.: 14112   1st Qu.: 10896
##  Median : 19200   Median : 18674
##  Mean   : 24068   Mean   : 23217
##  3rd Qu.: 30096   3rd Qu.: 29263
##  Max.   :117396   Max.   :116694
##
```

Táboas

Emprégase o comando *table()*:

```
table(gastan$CCAA)
```

```
##
##  1  2  3  4  6  7  8  9 10 11 12 13 14 15 16 17
##  5  4  2  1  3  1  2  4  1  3 42  5  4  4  3  1
```

```
table(lugar)
```

```
## lugar
##   gal outro
##    42    43
```

Pódese mellorar un pouco a presentación modificando os nomes dos valores das táboas, por exemplo:

```
ccaa=list("And","Ara","Ast","Bal","Cnt","CyL","CLM","Cat",
          "Val","Ext","Gal","Mad","Mur","Nav","Eus","Rio")
#Non hai ningún datos de canarias,por iso non incluín "Can")

taboa=table(gastan$CCAA)#a táboa pode gardarse coma calquera obxecto
dimnames(taboa)[[1]]=ccaa
taboa
```

```
##
## And Ara Ast Bal Cnt CyL CLM Cat Val Ext Gal Mad Mur Nav Eus Rio
##   5   4   2   1   3   1   2   4   1   3  42   5   4   4   3   1
```

```
#Sería máis simple cambiar os números polos nomes dentro do factor
#e facer a taboa despois
```

Táboas de frecuencias relativas obtéñense usando *prop.table()* aplicado a unha táboa

```
taboa=table(gastan$TAMAMU)#a táboa pode gardarse coma calquera obxecto,
#así pode manipularse máis tarde

taboa
```

```
##
##  1  2  3  4  5
## 32 12 10  8 23
```

```
prop.table(taboa)
```

```
##
##          1          2          3          4          5
## 0.37647059 0.14117647 0.11764706 0.09411765 0.27058824
```

```
#Podemos multiplicar po 100 e redondear a 2 decimais
round(100*prop.table(taboa),2)
```

```
##
##     1     2     3     4     5
## 37.65 14.12 11.76  9.41 27.06
```

Frecuencias acumuladas usando *sumcum()*

```
cumsum(taboa)
```

```
##  1  2  3  4  5
## 32 44 54 62 85
```

```
cumsum(prop.table(taboa))
```

```
##         1         2         3         4         5
## 0.3764706 0.5176471 0.6352941 0.7294118 1.0000000
```

Táboas bidimensionais

A función *table()* úsase tamén para 2 variables

```
table(gastan$SEXOSP,gastan$TIPHOGAR8)
```

```
##
##      1  2  3  4
##   1 21  8 13 19
##   6  9  4  3  8
```

```
taboa2=table(gastan$SEXOSP,gastan$TIPHOGAR8) #tamén é cómodo gardala
#para seguir traballando con ela
```

As taboas de frecuencias relativas poden facerse con respecto ao total, ou con respecto a cada unha das variables:

```
prop.table(taboa2) #freuencias relativas totais
```

```
##
##              1          2          3          4
##   1 0.24705882 0.09411765 0.15294118 0.22352941
##   6 0.10588235 0.04705882 0.03529412 0.09411765
```

```
prop.table(taboa2,1) #freuencias relativas por filas
```

```
##
##             1         2         3         4
##   1 0.3442623 0.1311475 0.2131148 0.3114754
##   6 0.3750000 0.1666667 0.1250000 0.3333333
```

```
prop.table(taboa2,2) #freuencias relativas por columnas
```

```
##
##             1         2         3         4
##   1 0.7000000 0.6666667 0.8125000 0.7037037
##   6 0.3000000 0.3333333 0.1875000 0.2962963
```

Táboas agrupadas en intervalos

Non existe unha función directa para obtelas, pero poden facerse combinando varios comandos.

```
#Averiguar máximo e mínimo da variable
summary(leite)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.51   84.20  196.60  221.80  281.60  801.10
```

```
#crear un factor indicando a que intervalo pertence cada elemento da variable
intervalos=cut(leite, breaks=c(0,10,25,50,75,100,200,500,1000,4000),
               labels = c("0-10","10-25","25-50","50-75","75-100","100-200",
                          "200-500","500-1000","1000-4000"))
table(intervalos)
```

```
## intervalos
##      0-10     10-25     25-50     50-75    75-100   100-200   200-500
##         5         4         4         5         9        17        33
##  500-1000 1000-4000
##         8         0
```

Gráficos

No que segue imos presentar os gráficos máis habituais na Estatística básica, presentes dentro do sistema base de gráficos de R.

Unha característica deste sistema é que produce os graficos moi pouco elaborados, deixando moitos aspectos para a persoalización mediante parámetros.

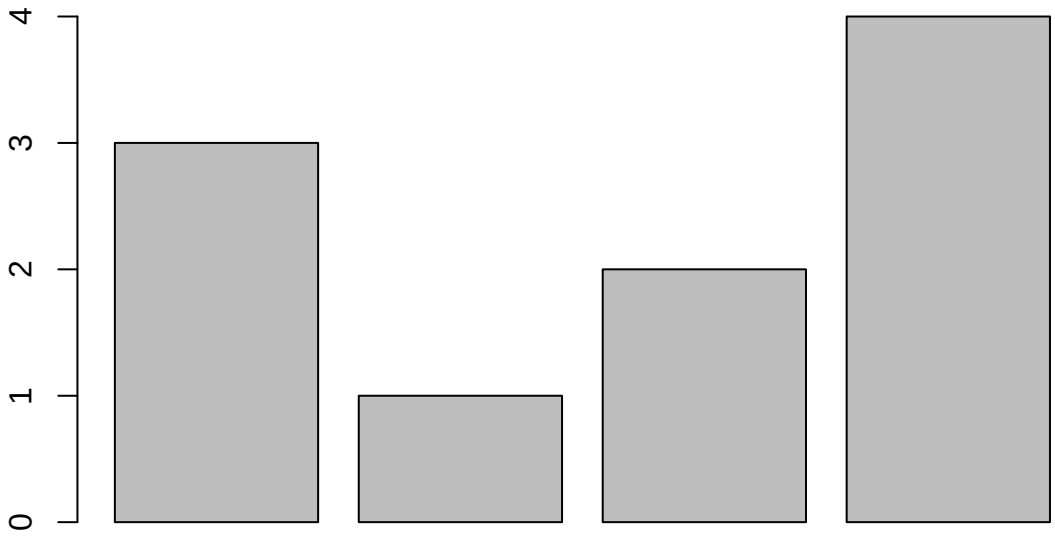
Os comandos que producen as gráficas do sistema base poden aplicarse nalgúns casos a táboas (ou frecuencias de valores), maiormente aqueles que representan variables discretas, ou directamente ao conxunto total dos datos (maiormente os que representan varibles continuas)

Diagrama de barras

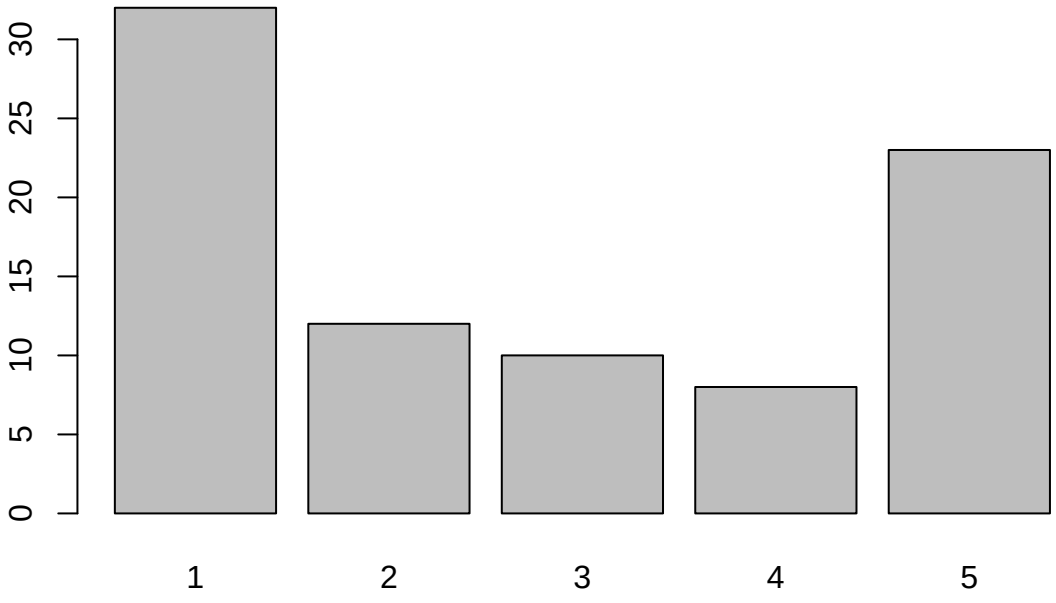
barplot(taboa)

Esta función **aplícase** a un vector de frecuencias ou directamente **a táboas**:

```
barplot(c(3,1,2,4)) #Un vector con 4 frecuencias
```




```
barplot(table(gastan$TAMAMU)) #Feito a partir dunha táboa dunha variable
```

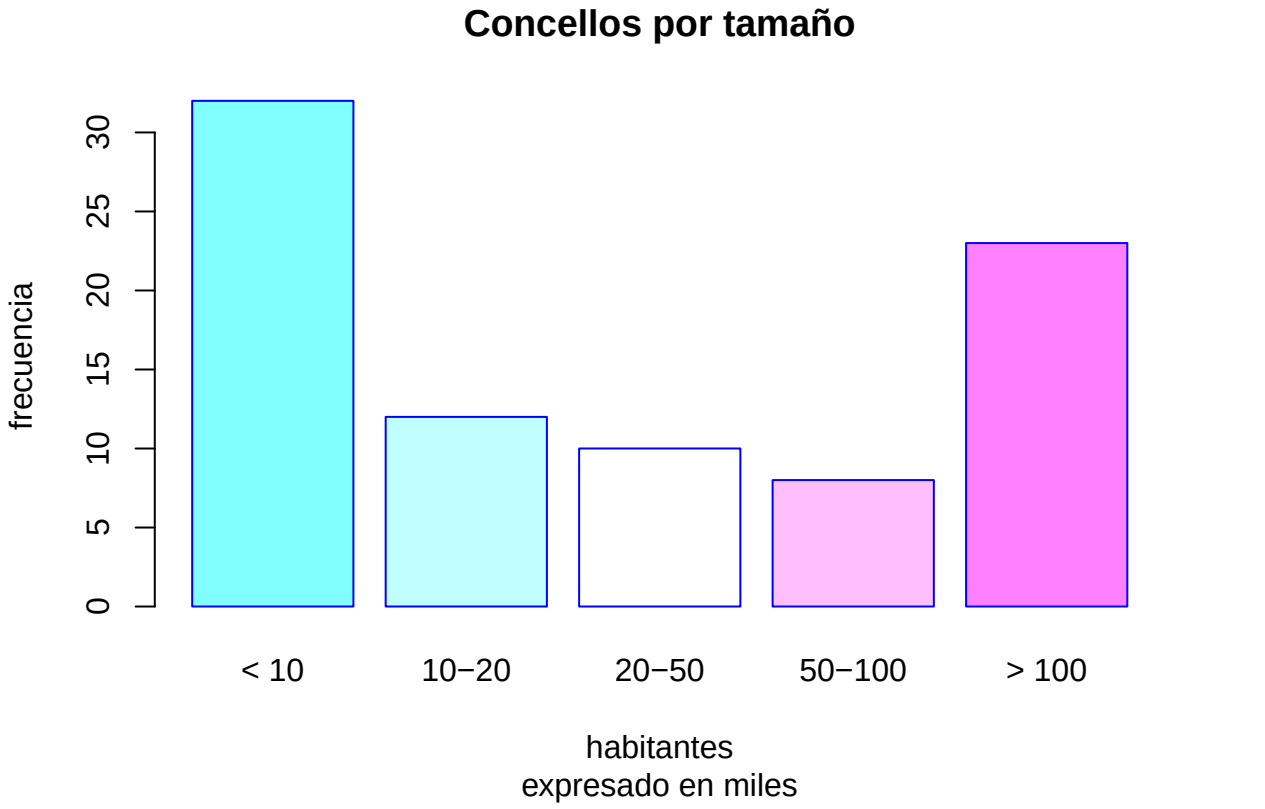

0
5
10
15
20
25
30
1
2
3
4
5

Algúns dos parámetros que modifican este comando:

- *space*=c(0.2,0.8) Espazo entre as barras e entre os grupos de barras
- *names.arg*=c("< 10", "10-20", "20-50", "50-100", "> 100") Nomes para as barras
- *col*=c("blue", "red") Cor das barras
- *border*="black" Cor do bordo das barras
- *main*=c("Concellos por tamaño") Título da gráfica
- *sub*="expresado en miles" Subtítulo para a gráfica
- *xlab*="habitantes" texto para o eixe X
- *ylab*="frecuencia" texto para o eixe Y
- *xlim*=c(0,8) límites para o eixe X
- *ylim*=c(0,1400) límites para o eixe Y

Estes parámetros producen a gráfica seguinte:

```
barplot(table(gastan$TAMAMU), space=c(0.2,0.8),
        names.arg=c("< 10", "10-20", "20-50", "50-100", "> 100"),
        col=cm.colors(5),border="blue",main=c("Concellos por tamaño"),
        sub="expresado en miles",xlab="habitantes",
        ylab="frecuencia")
```



As cores en R

As cores en R indicanse mediante un obxecto que conteña os numeros correspondentes ou os seus nomes (a lista de nomes podedes vela co comando *colors()*, e nomes, números e cor correspondente aquí: http://research.stowers-institute.org/efg/R/Color/Chart/index.htm).

Ademais cas cores que poidamos especificar nun vector, existen agrupacións de cores xa predeterminadas que se denominan paletas.

Por exemplo na gráfica anterior usouse a paleta *cm.colors()*. Outras paletas posibles son *rainbow()*, *heat.colors()*,*terrain.colors()*, *topo.colors()*.

Incluir unha lenda explicativa

Faise coa función *legend()*, na que se debe indicar:

- A posición na que se sitúa, que poden ser dous vectores X e Y, ou unha etiqueta: *topleft*, *bottomright*, *bottom*, *bottomleft*, *left*, *topleft*, *top*, *topright*, *right*, *center*
- *legend*, un parámetro dentro da función *legend()* que indica os textos que aparecen na lenda
- *fill*, as cores que se representan na gráfica e que deben coincidir con esta

Exemplo de diagrama de barras con lenda:

En ocasions é necesario manipular o rango do eixe X ou do Y (parámetros *xlim* ou *ylim*), para que a lenda non se superpoña encima da propia gráfica

```
barplot(table(gastan$TAMAMU), space=c(0.2,0.8),
        col=cm.colors(5),border="blue",main=c("Concellos por tamaño"),
        sub="expresado en miles",xlab="habitantes",
        ylab="frecuencia",xlim=c(1,8))

legend("right",legend=c("< 10", "10-20", "20-50", "50-100", "> 100"),
        fill=cm.colors(5) )
```

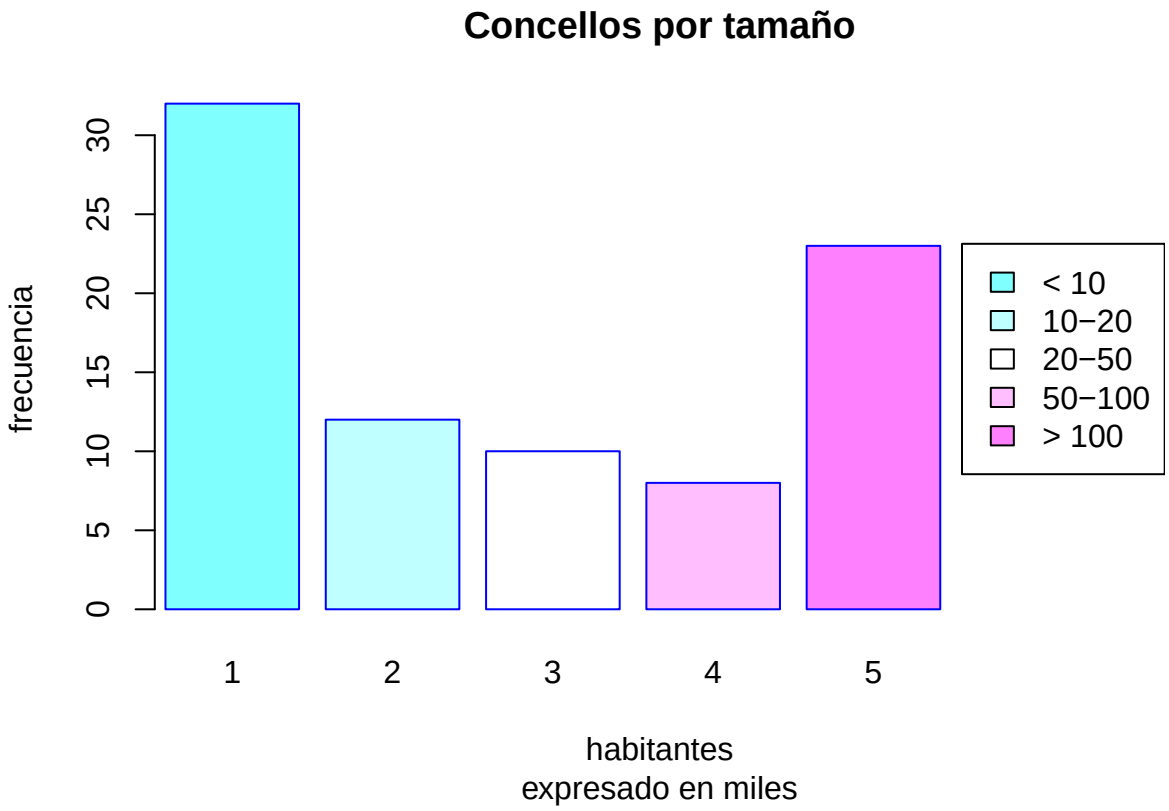
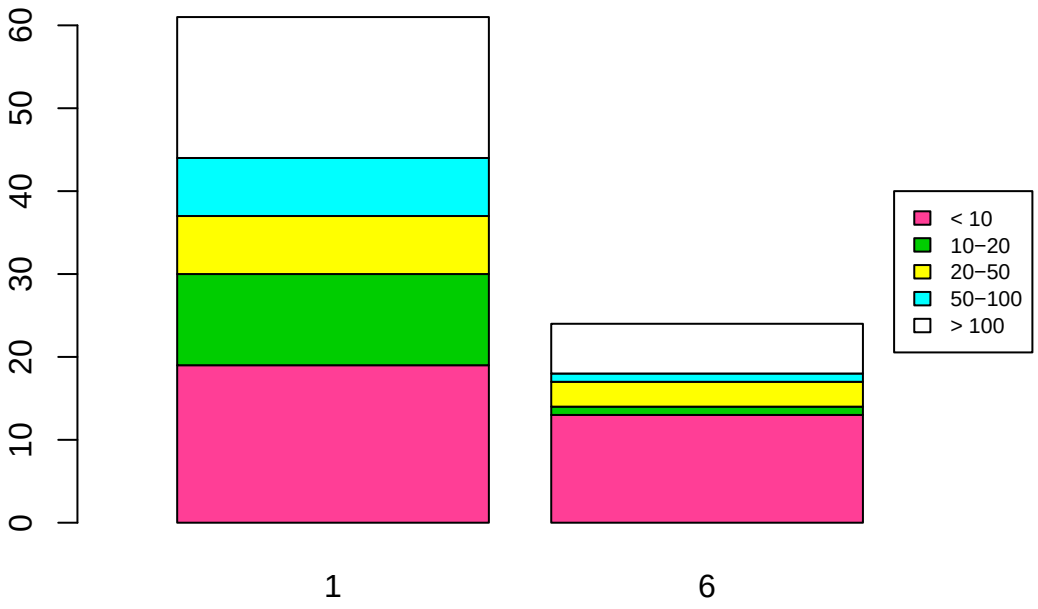


Diagrama de barras combinando 2 variables (ou atributos)

barplot() #Aplícado a unha táboa de dobre entrada

```
#########barplot para dous atributos (ou variables)
#combina a cor e a claridade dos diamantes
taboa3=with(gastan,table(TAMAMU,SEXOSP))
cores=c('violetred1','green3',607,'cyan', 'white','red')
barplot(taboa3,
        xlim=c(0,3),
        col = cores)

#agora coloco a carón da gráfica o que significa cada cor coa función legend()
legend(list(x=2.5,y=40),
       legend=c("< 10", "10-20", "20-50", "50-100", "> 100"),
       cex=.7,
       fill=cores,
       ncol=1)
```



Que significan os parámetros de *legend()*

- **list(x=2.5,y=40)**: indica a posición, nas coordenadas (8,40);
- **legend=rownames(table(color))**: Os nomes que ten que escribir. Neste caso os nomes das cores dos tamaños de concello
- **cex=.7**: O tamaño da caixa coa lenda
- **fill=cores**: as cores que representa, deben ser as mesmas cás da gráfica
- **ncol=1**: que aparezan organizadas en 1 columna

Os diagramas de barras poden colocarse sen estar acumulados, en varios grupos de barras usando o parámetro *__beside=TRUE__

```
#Outra maneira, poñéndoos a caron uns dos outros

#beside=TRUE serve para indicarlle que coloque as barras unha á beira da outra

barplot(taboa3,
        col = cores[1:5],
        beside=TRUE)

#A caixa que indica o que é cada cor só se diferencia da anterior na posición, topleft indica "arriba á
#"bottomright","bottom","bottomleft","left","topleft","top","topright","right","center"
legend("topright",
       legend=c("< 10", "10-20", "20-50", "50-100", "> 100"),
       cex=.7,
       fill=cores[1:5],
       ncol=1)
```

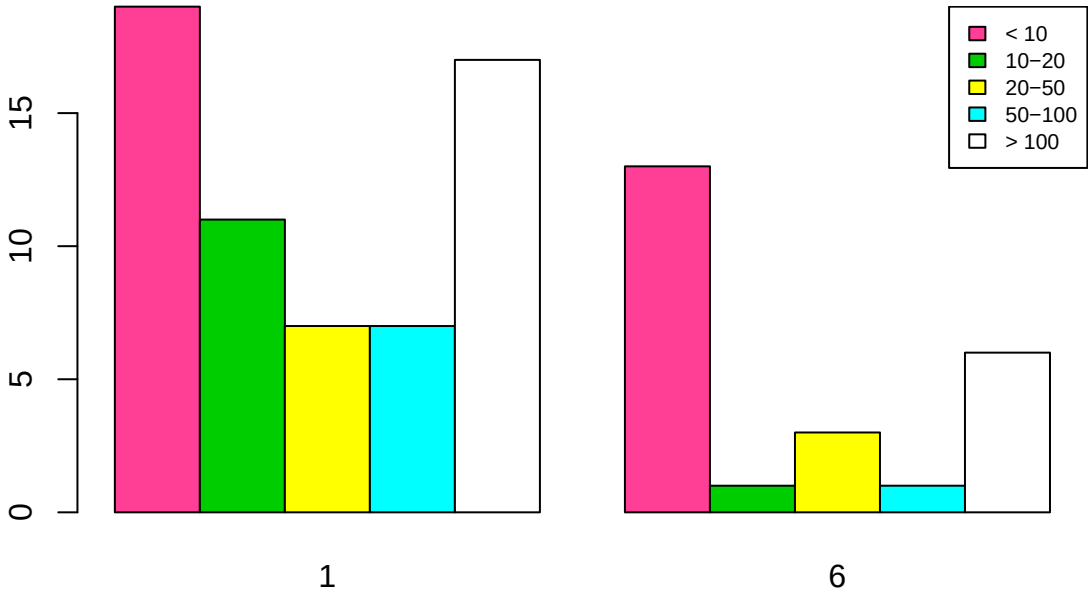
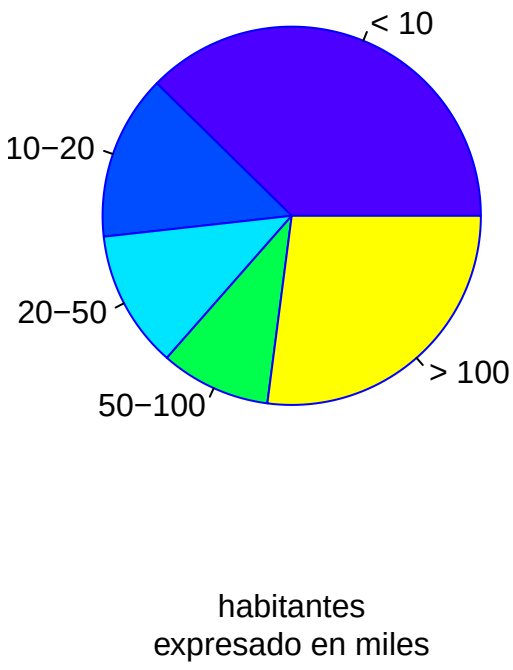


Diagrama de sector

pie(taboa)

Igual ca barplot(), aplícase a un vector de frecuencias ou directamente a táboas:

```
pie(table(gastan$TAMAMU),labels = c("< 10", "10-20", "20-50", "50-100", "> 100"),
    col=topo.colors(5),border="blue",main=c("Concellos por tamaño"),
        sub="expresado en miles",xlab="habitantes")
```



Neste caso aparece o parámetro *labels* para colocar as etiquetas a cada un dos sectores.

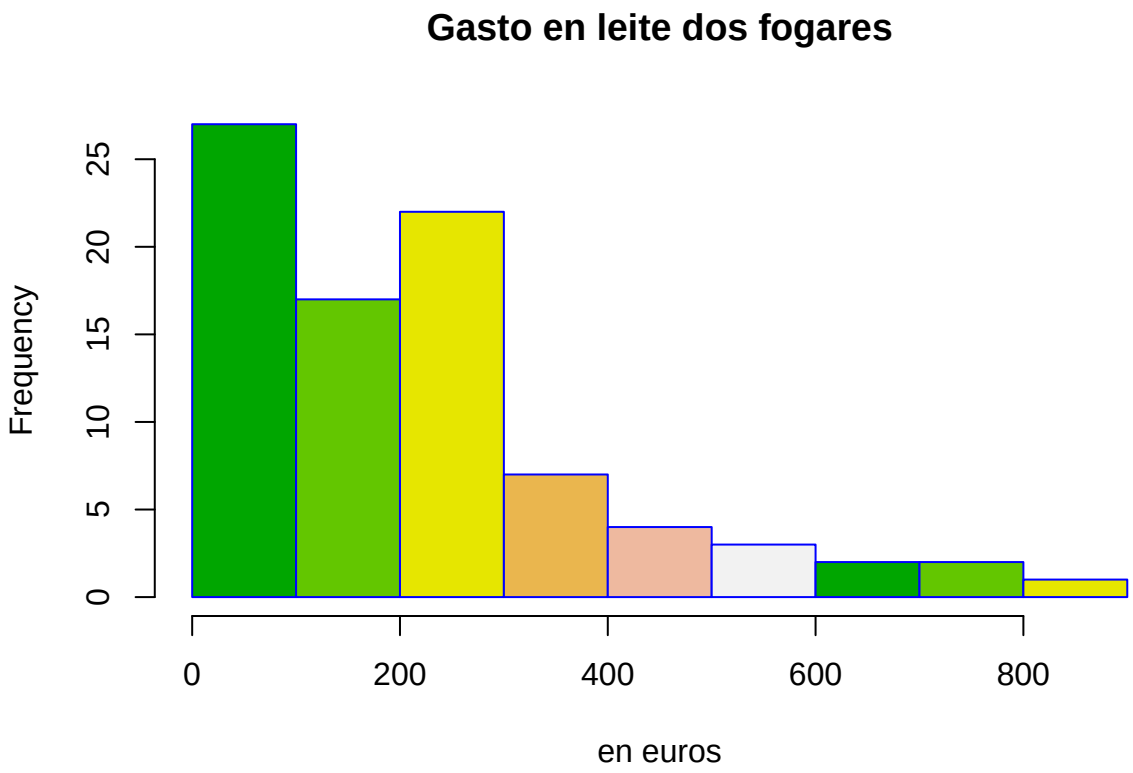
Histograma

Aplícase a variables continuas.

hist(variable)

Esta función aplícase directamente a un obxecto con **valores da variable**

```
hist(leite,breaks=6,col=terrain.colors(6),border="blue",main=c("Gasto en leite dos fogares"),
        xlab="en euros")
```



Agora aparece o parámetro *breaks* para indicar onde se fai a separación dos intervalos ou cantos intervalos se queren. Non é obrigatorio poñelos, xa que R escolle un número que considere óptimo.

Con ese parámetro pódense indicar intervalos de diferentes amplitudes o que dará outro aspecto ao noso histograma:

```
hist(leite,breaks=c(0,200,400,600,800,1000,2000,4000),col=terrain.colors(6),border="blue",main=c("Gasto
        xlab="en euros")
```

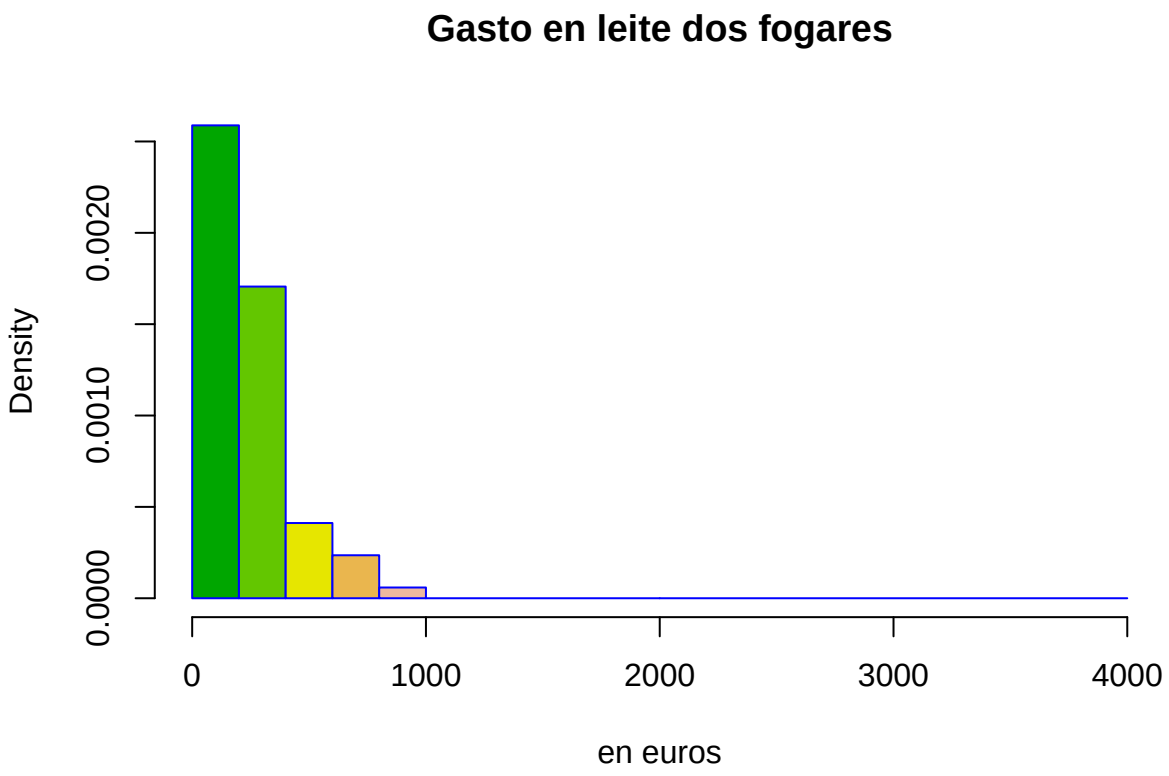

Gasto en leite dos fogares
Density
0.0000
0.0010
0.0020
0
1000
2000
3000
4000
en euros

Diagrama de caixa

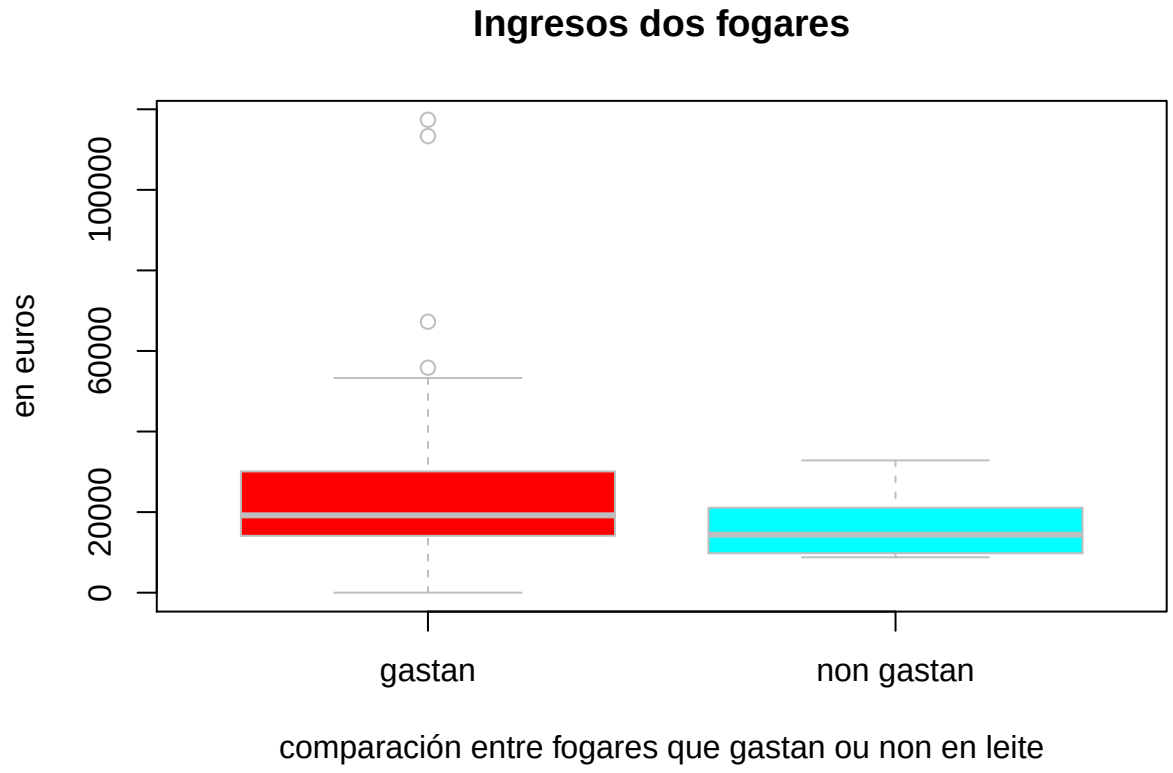
Aplícase a variables continuas.

boxplot(variable)

```
boxplot(leite,border="blue",main=c("Gasto en leite dos fogares"),
        xlab="en euros")
```

Con este gráfico é fácil representar variables xuntas, pero hai que procurar que teñan valores semellantes, senon non serían comparables:

```
boxplot(gastan$IMPEXAC, non.gastan$IMPEXAC,col=rainbow(2),border="gray",
        names=c("gastan","non gastan"), main=c("Ingresos dos fogares"),
        ylab="en euros",
        xlab="comparación entre fogares que gastan ou non en leite")
```



Neste caso é o parámetro *names* o que lle pon os nomes ás caixas.

Tamén é facil representar unha variable tendo en conta as categorías dun atributo. Por exempo en **gasto en leite en función do número de persoas entre 5 e 16 anos**

```
boxplot(leite~gastan$NMIEM8,col=rainbow(7),border="blue",
        main=c("gasto en leite"),ylab="en euros",
        xlab="nº persoas entre 5 e 16 anos")
```

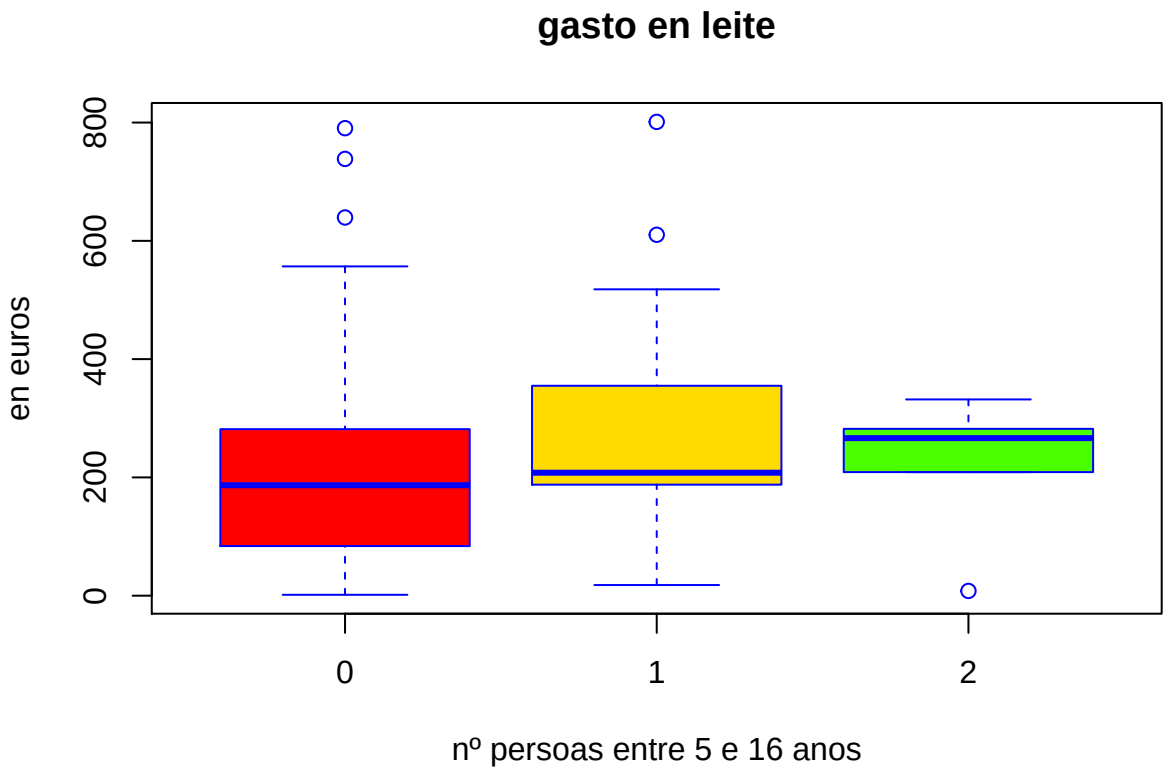


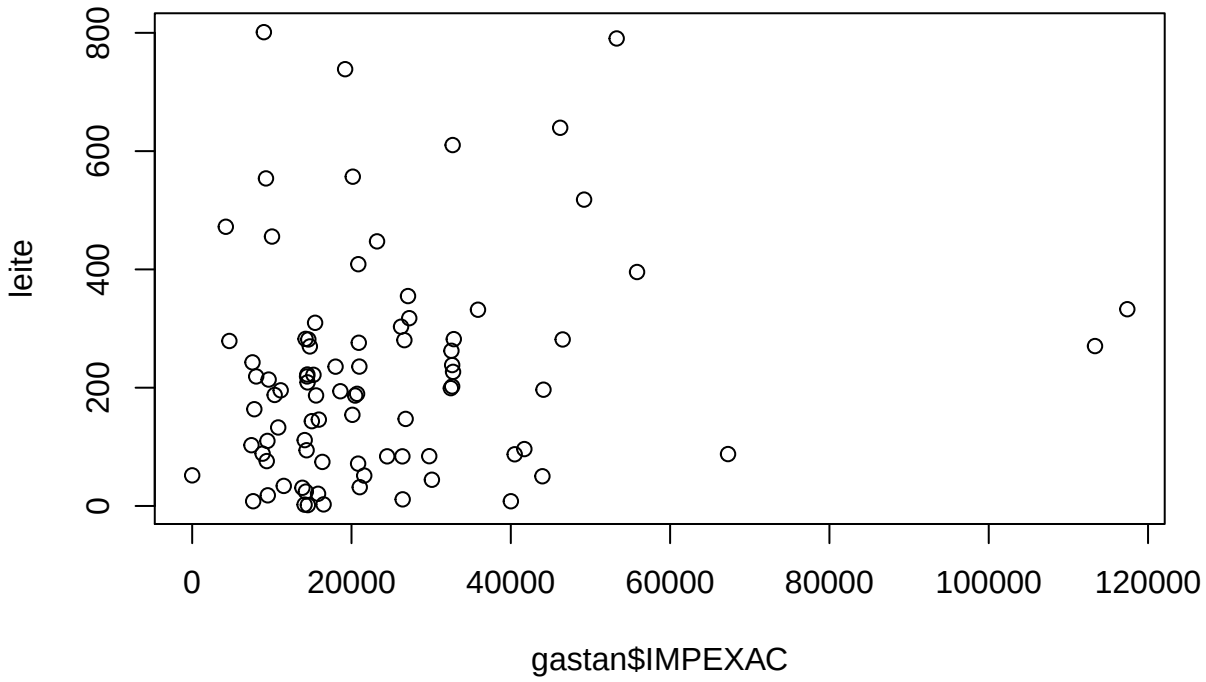
Diagrama de dispersión

Para 2 variables. Usa a función *plot()*, esta función é xenérica en R, posto que o seu resultado varía en función do tipo de obxecto ao que se lle aplique.

Para un diagrama de dispersión esta función aplícase a dous obxectos numéricos (pode ser un data.frame de 2 columnas), cunha forma xeral

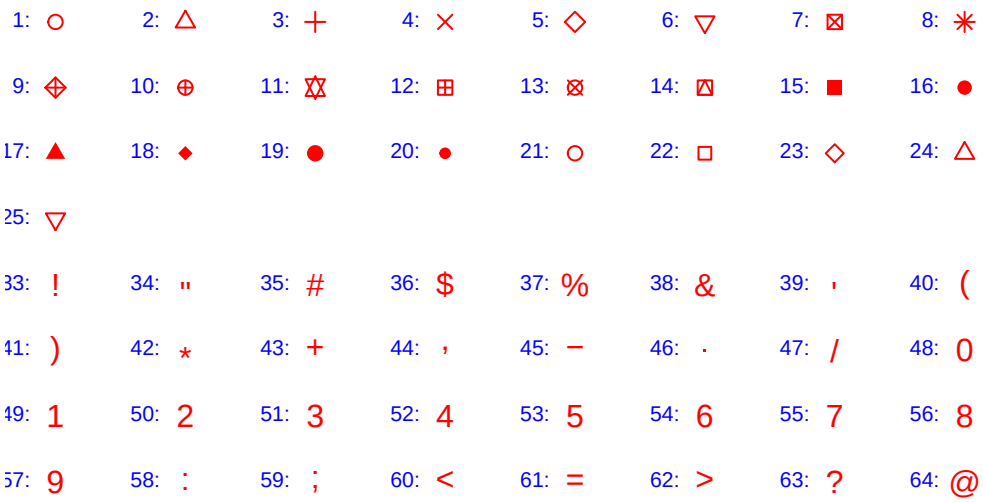
plot(variable X, variable Y)

```
plot(gastan$IMPEXAC,leite)
```



Algúns parámetros que se poden modificar:

- **type**: tipo de gráfico representado + **"p"**: puntos + **"l"**: liñas + **"b"**: os dous
- **col**: cores
- **pch**: tipo de puntos que se representan:

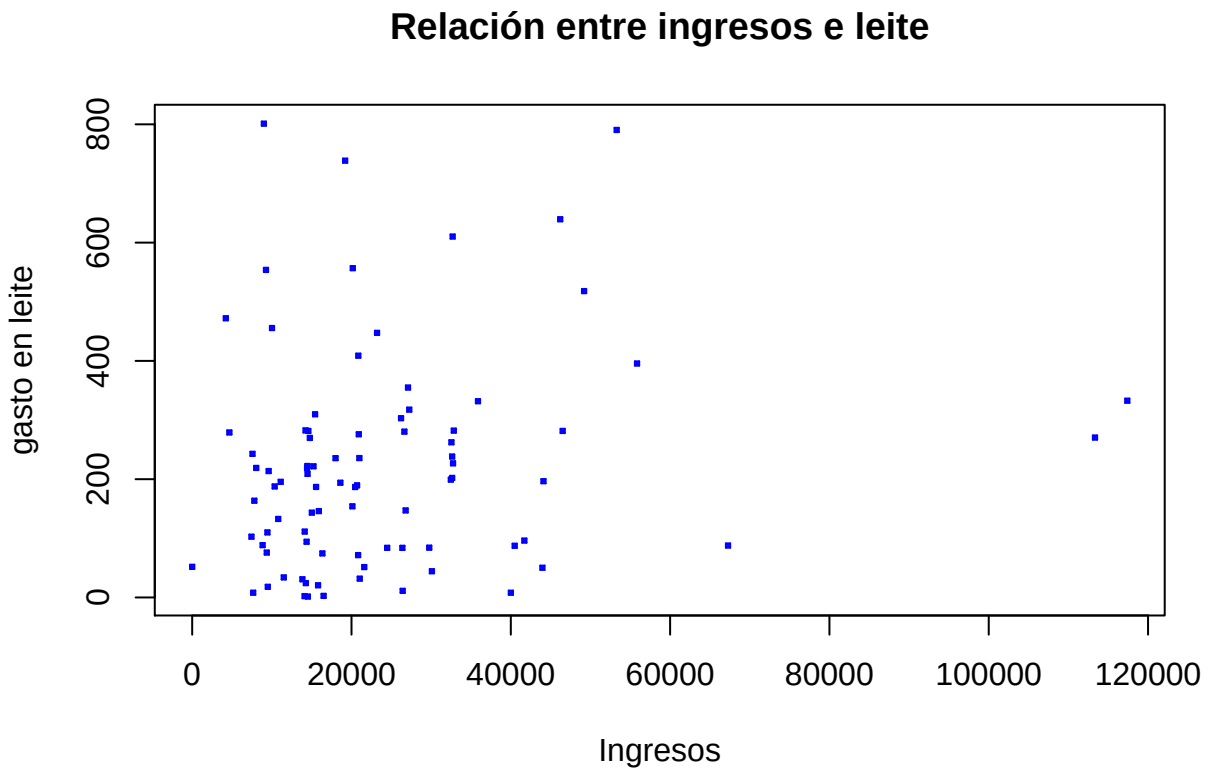


Os tipos de puntos van de **1 a 25**, pero a partir de 33 inclúe os diferentes simbolos ASCII, e polo tanto letras e números.

- **lty**: Tipo de liña

- **axes**: indica se colocar ou non os eixes
- **cex**: Indica o tamaño dos puntos representados

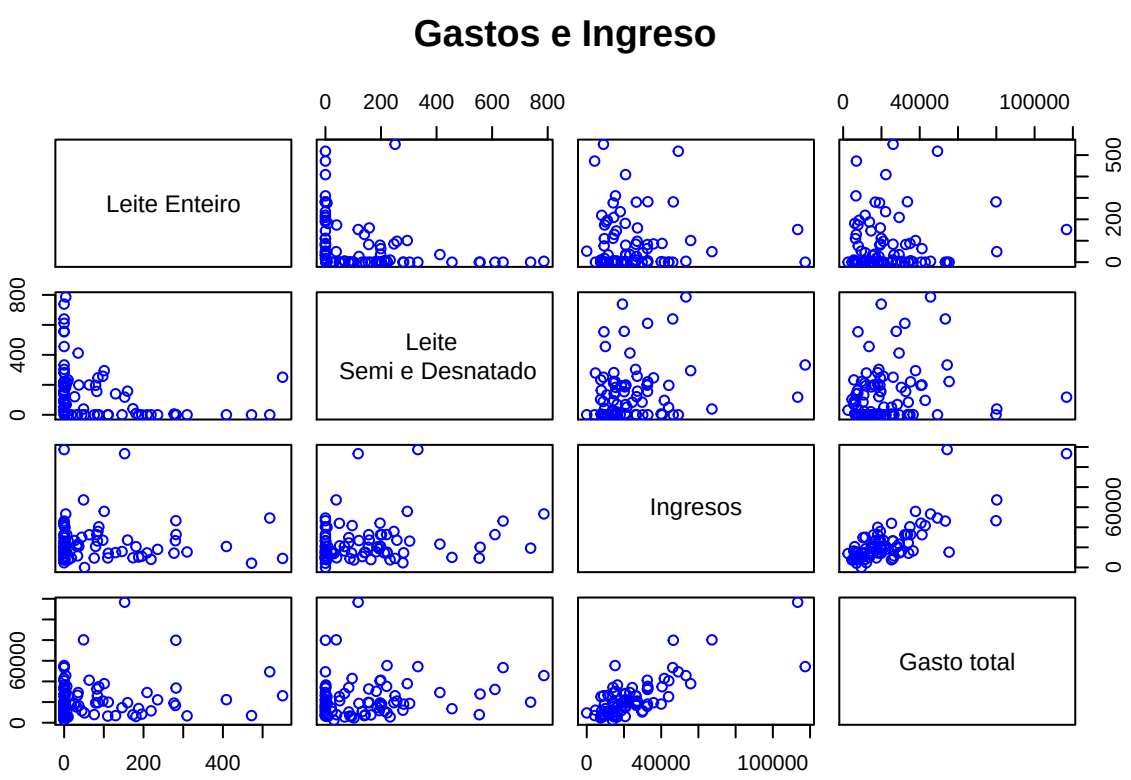
```
plot(gastan$IMPEXAC,leite,pch=7,cex=0.3,col="Blue",
     main="Relación entre ingresos e leite",xlab="Ingresos",ylab="gasto en leite")
```



Matriz de dispersión

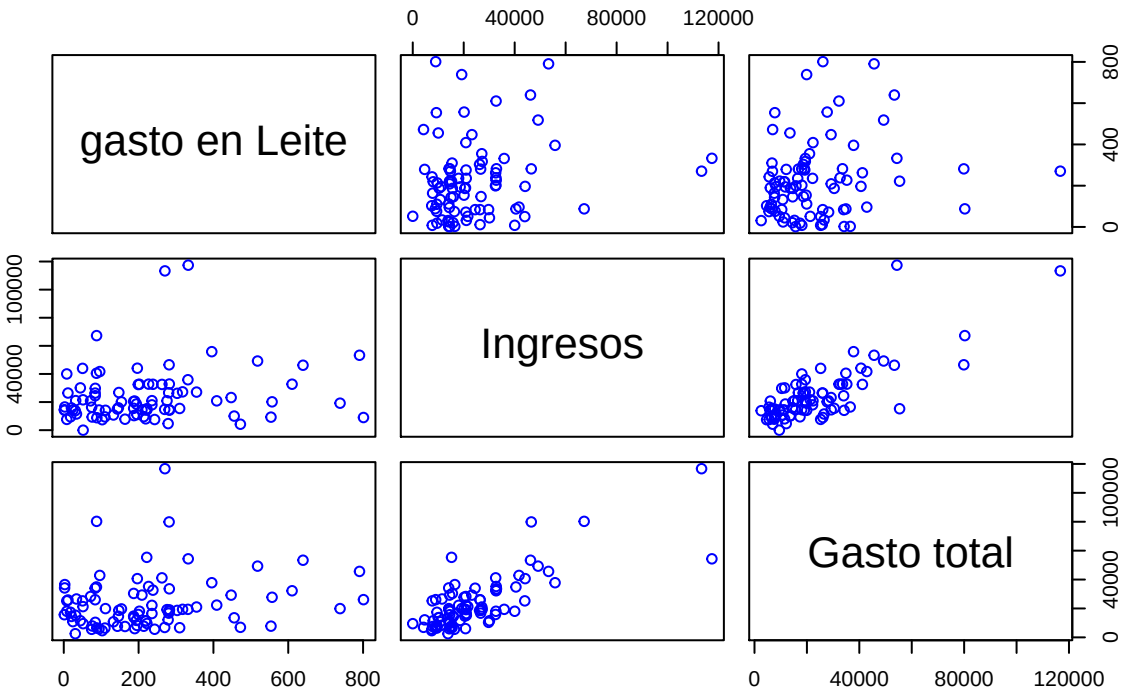
Cando se representan máis de dúas variables continuas pódense combinar varios diagramas de dispersión, algo que se fai co propio comando *plot()* ou con *pairs()*

```
pairs(gastan[c(1,2,15,16)], main = "Gastos e Ingreso", pch = 21,col="blue",
      labels=c("Leite Enteiro","Leite \n Semi e Desnatado",
               "Ingresos","Gasto total"))
```



Neste caso introduciuse as variables dentro dun data.frame, pero tamén se poderían introducir variables separadas:

```
pairs(~leite+gastan$IMPEXAC+gastan$GASTMON, main = "Gastos e Ingreso",
      pch = 21,col="blue",
      labels=c("gasto en Leite","Ingresos","Gasto total"))
```


Gastos e Ingreso
gasto en Leite
Ingresos
Gasto total
0
40000
80000
120000
0
400
800
0
40000
100000
0
200
400
600
800

Calculos de parámetros estatísticos

Cada cálculo estatístico ou método vai estar asociada a un comando de R, como mínimo, posto que ao ser R unha linguaxe de programación é fácil amplialo, e resulta habitual atopar diferentes comandos para aplicar o mesmo método, cada un coas súas particularidades.

Parámetros básicos

Unha visión elemental sobre o conxunto de datos obtense co comando *summary()*, este comando é xenérico, e produce saidas diferentes en función do obxecto ao que é aplicado.

Se o aplicamos a un obxecto con datos numéricos proporciona a media aritmética, mínimo e máximo, os cuartis e a mediana:

```
summary(leite)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.51   84.20  196.60  221.80  281.60  801.10
```

```
summary(porcentaxe)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   0.390   1.200   1.383   1.730   7.140
```

Pero eses parámetros tamén se obteñen con comandos separados:

```
mean(leite) #media
```

```
## [1] 221.8263
```

```
median(leite) #mediana
```

```
## [1] 196.6122
```

```
quantile(leite)
```

```
##         0%        25%        50%        75%       100%
##   1.509534  84.200189 196.612249 281.571126 801.114158
```

A función quatile da por defecto máximo, mínimo, mediana e cuartis, pero tamén permite calcular cuantis con outras porcentaxes, por exemplo:

```
quantile(leite,0.25) #1º cuartil
```

```
##      25%
## 84.20019
```

```
quantile(leite,0.7) #7º decil
```

```
##      70%
## 274.8421
```

Para as medidas de dispersión hai que ter en conta que R calcula a ***varianza mostral***, ou sexa, a **cuase-varianza**, e o mesmo aplícase á desviación típica:

```
sd(leite) #desviación típica
```

```
## [1] 182.8658
```

```
var(leite) #varianza
```

```
## [1] 33439.91
```

Para a covarianza e os coeficientes de correlación e determinación:

```
cov(leite,gastan$IMPEXAC)  #covarianza
```

```
## [1] 629061.9
```

```
cor(leite,gastan$IMPEXAC)  #coeficiente de correlación
```

```
## [1] 0.1778043
```

```
cor(leite,gastan$IMPEXAC)^2  #coeficiente de determinación
```

```
## [1] 0.03161438
```

summary(), *var()*, *cor()* pódense aplicar a un conxunto de variables agrupados nun data.frame().

Con summary() obtén os mesmos parámetros para cada unha das columnas do data.frame(). Con *var()* (ou *cov()*) e *cor()* obtense a matriz de varianzas-covarianzas ou de correlacións desas variables.

```
var(data.frame(leite,gastan[c(1,2,15,16)]))  #covarianza
```

```
##                   leite leite.enteiro  leite.non      IMPEXAC     GASTMON
## leite          33439.907      8499.434  24940.474    629061.86    661251.2
## leite.enteiro   8499.434     15603.989  -7104.555    -21815.34    207988.5
## leite.non      24940.474     -7104.555  32045.029    650877.20    453262.7
## IMPEXAC       629061.862    -21815.335 650877.197 374314809.28 278344459.3
## GASTMON       661251.248    207988.498 453262.750 278344459.26 344477911.4
```

```
cov(data.frame(leite,gastan[c(1,2,15,16)]))  #covarianza
```

```
##                   leite leite.enteiro  leite.non      IMPEXAC     GASTMON
## leite          33439.907      8499.434  24940.474    629061.86    661251.2
## leite.enteiro   8499.434     15603.989  -7104.555    -21815.34    207988.5
## leite.non      24940.474     -7104.555  32045.029    650877.20    453262.7
## IMPEXAC       629061.862    -21815.335 650877.197 374314809.28 278344459.3
## GASTMON       661251.248    207988.498 453262.750 278344459.26 344477911.4
```

```r
cor(data.frame(leite,gastan[c(1,2,15,16)]))  #coeficiente de correlación
```

```
##                   leite leite.enteiro  leite.non      IMPEXAC    GASTMON
## leite         1.0000000   0.372082841  0.7618892  0.177804324 0.19482886
## leite.enteiro 0.3720828   1.000000000 -0.3177157 -0.009026629 0.08970999
## leite.non     0.7618892  -0.317715743  1.0000000  0.187931777 0.13642344
## IMPEXAC       0.1778043  -0.009026629  0.1879318  1.000000000 0.77514578
## GASTMON       0.1948289   0.089709992  0.1364234  0.775145778 1.00000000
```

```r
cor(data.frame(leite,gastan[c(1,2,15,16)]))^2  #coeficiente de determinación
```

```
##                    leite leite.enteiro  leite.non      IMPEXAC     GASTMON
## leite         1.00000000  1.384456e-01 0.58047519 3.161438e-02 0.037958284
## leite.enteiro 0.13844564  1.000000e+00 0.10094329 8.148003e-05 0.008047883
## leite.non     0.58047519  1.009433e-01 1.00000000 3.531835e-02 0.018611356
## IMPEXAC       0.03161438  8.148003e-05 0.03531835 1.000000e+00 0.600850978
## GASTMON       0.03795828  8.047883e-03 0.01861136 6.008510e-01 1.000000000
```

Calculos por grupos

En ocasións é interesante realizar cálculos para diferentes subconxuntos de datos, por exemplo o gasto medio en leite para os diferentes tipos de fogares.

by(variable, atributo,funcion)

```r
by(leite,gastan$TIPHOGAR8,mean)
```

```
## gastan$TIPHOGAR8: 1
## [1] 184.3876
## ------------------------------------------------------------
## gastan$TIPHOGAR8: 2
## [1] 235.0476
## ------------------------------------------------------------
## gastan$TIPHOGAR8: 3
## [1] 222.9839
## ------------------------------------------------------------
## gastan$TIPHOGAR8: 4
## [1] 256.8629
```

```r
by(leite,gastan$TIPHOGAR8,sd)
```

```
## gastan$TIPHOGAR8: 1
## [1] 158.6704
## ------------------------------------------------------------
## gastan$TIPHOGAR8: 2
## [1] 180.113
## ------------------------------------------------------------
## gastan$TIPHOGAR8: 3
## [1] 82.12423
## ------------------------------------------------------------
## gastan$TIPHOGAR8: 4
## [1] 243.6512
```

ou o comando equivalente

tapply(variable, atributo,funcion)

Este comando funciona aplicando a *función* aos valores da *variable*, pero separados en grupos determinados polos elementos do *atributo*.

```
tapply(leite,gastan$TIPHOGAR8,mean)
```

```
##        1        2        3        4
## 184.3876 235.0476 222.9839 256.8629
```

```
tapply(leite,gastan$TIPHOGAR8,sd)
```

```
##         1         2         3         4
## 158.67041 180.11298  82.12423 243.65117
```

regresión

O comando para a regresión é *lm()*

lm(formula)

formula é un obxecto de tipo **formula** que indica a forma que vai ter a regresión. Por exemplo:

- *Y~X* é a formula para unha regresión simple $Y = a + b \cdot X$,
- *Y~X1+X2* é a formula para unha regresión multiple $Y = b_0 + b_1 \cdot X1 + b_2 \cdot X2$

Ao executar o comando lm fanse diferentes cálculos relacionados coa regresión, pero só se amosan os coeficientes:

```
lm(leite~gastan$IMPEXAC+gastan$NMIEMB+gastan$EDADSP+gastan$TIPHOGAR8)
```

```
##
## Call:
## lm(formula = leite ~ gastan$IMPEXAC + gastan$NMIEMB + gastan$EDADSP +
##     gastan$TIPHOGAR8)
##
## Coefficients:
##       (Intercept)     gastan$IMPEXAC      gastan$NMIEMB
##        -3.554e+02          3.978e-04          7.261e+01
##     gastan$EDADSP  gastan$TIPHOGAR82  gastan$TIPHOGAR83
##         5.733e+00          1.736e+02          7.948e+01
## gastan$TIPHOGAR84
##        -1.610e+01
```

Pódese ver máis información aplicando *summary()* á regresión:

```
summary(lm(leite~gastan$IMPEXAC+gastan$NMIEMB+gastan$EDADSP+gastan$TIPHOGAR8))
```

```
##
## Call:
## lm(formula = leite ~ gastan$IMPEXAC + gastan$NMIEMB + gastan$EDADSP +
##     gastan$TIPHOGAR8)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -258.04 -108.43  -28.81   63.59  562.65
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -3.554e+02  1.899e+02  -1.871  0.06508 .
## gastan$IMPEXAC     3.978e-04  1.087e-03   0.366  0.71550
## gastan$NMIEMB      7.261e+01  2.649e+01   2.741  0.00758 **
## gastan$EDADSP      5.733e+00  2.386e+00   2.403  0.01863 *
## gastan$TIPHOGAR82  1.736e+02  8.086e+01   2.147  0.03489 *
## gastan$TIPHOGAR83  7.948e+01  9.747e+01   0.816  0.41727
## gastan$TIPHOGAR84 -1.610e+01  7.158e+01  -0.225  0.82263
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 174 on 78 degrees of freedom
## Multiple R-squared:  0.1596, Adjusted R-squared:  0.09491
## F-statistic: 2.468 on 6 and 78 DF,  p-value: 0.0308
```

Por suposto, tamén podemos gardar os cálculos dunha regresión como un obxecto e traballar despois con eles:

```
regresion=lm(leite~gastan$IMPEXAC+gastan$NMIEMB+gastan$EDADSP+gastan$TIPHOGAR8)
```

Podemos ver que se garda usando o comando *names()*:

```
names(regresion)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "contrasts"     "xlevels"       "call"          "terms"
## [13] "model"
```

Os resultados gardados en **regresion** teñen unha estructura semellante a unha **lista**, polo que se poden obter colocando ***$*** entre o nome do obxecto e o do elemento:

- *regresion$coefficients*: Os coeficientes da regresión

```
##       (Intercept)    gastan$IMPEXAC     gastan$NMIEMB     gastan$EDADSP
##     -3.553737e+02      3.977945e-04      7.260648e+01      5.733282e+00
## gastan$TIPHOGAR82 gastan$TIPHOGAR83 gastan$TIPHOGAR84
##      1.736174e+02      7.948418e+01     -1.609942e+01
```

Algúns outros cálculos gardados son:

- *residuals*: residuos
- *fitted.values*: valores preditos
- *model*: almacena os valores empregados para o cálculo da regresión.

Se gardamos nun obxecto os cálculos obtidos con **summary(lm())** obtemos algún outro resultado:

```
regresion2=summary(lm(leite~gastan$IMPEXAC+gastan$NMIEMB+gastan$EDADSP+gastan$TIPHOGAR8))
names(regresion2)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

Neste caso podemos ver:

- *coefficients*: Agora proporciona os coeficientes estimados, o erro estandar o estatístico t e o seu p-valor
- *sigma*: erro residual estandar
- *df*: graos de liberdade
- *r.squared*: R^2
- *adj.r.squared*: R^2 axustado