Contents

Introdución a R con RStudio

Que é isto do R?

É un software para realizar cálculos e aplicar métodos estatísticos.

Outros softwares semellantes son: SPSS, Stata, Gretl

Como se consegue R? Canto custa? R é software libre. Pódese obter na súa propia web: R-Project

Por ser software libre (licenza GPL) pódese compartir, modificar ou vender, sempre que se manteña esta mesma licenza.

Dentro da web do R-Project ***(www.r-project.org)***, debedes acceder no menú a **Download -> CRAN**, onde existen diferentes *mirrors* desde os que se pode baixar.

Por exemplo:http://ftp.cixug.es/CRAN/, que é o *mirror* da UDC.

Despois buscar o voso sistema operativo e baixar o instalador **base**, co que se obtén o propio programa e algúnhas extensións que permiten realizar a maioría das operacións básicas da Estatística

Para a xente que empregue **linux** é máis práctico instalalo desde os propios repositorios da distribución, xa que na maioría delas soe vir incluido.

Traballo con R R traballa con **comandos**, ou sexa, para facer algo debedes indicarlle a orde apropiada.

```
#escribides
2+2
#premedes ENTER

#e saevos na consola o resultado
```

```
## [1] 4
```

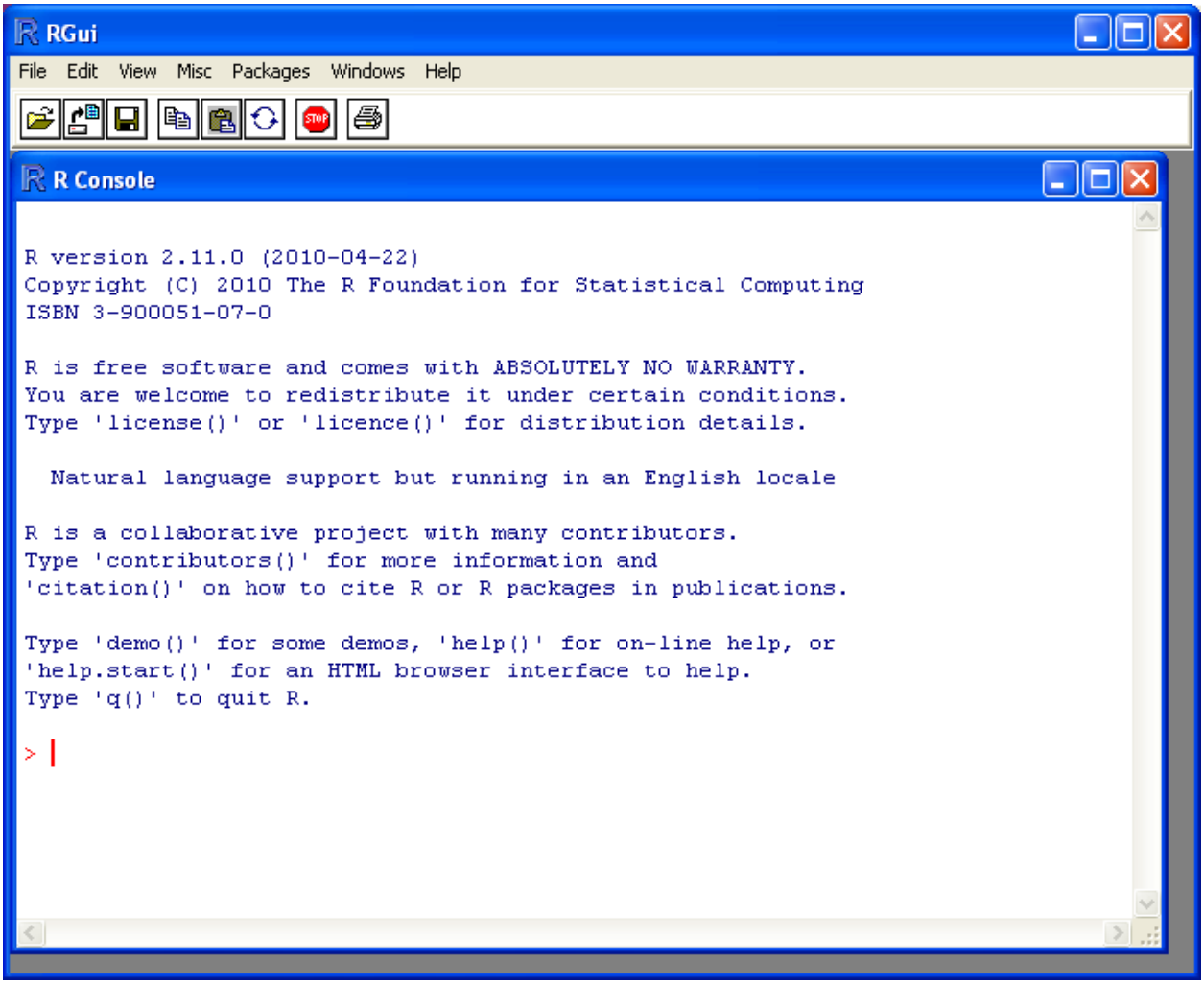
Traballar con comandos resulta complexo, pero suavízase esta dificultade empregando plantillas de código ou chuletarios, adapatados a diferentes situacións.

Deste xeito podense automatizar os cálculos, e avanzar máis rápido cando se fan tarefas repetitivas.

Aínda así tamén existen sistemas de menu para R, pero non aparecen por defecto.

O máis coñecido e Rcommader

Cando tedes instalado o R básico aparecevos unha consola para introducir comandos coma esta (en Windows):



Dado que este ambiente de traballo se fai complexo de manexar existen diferentes interfaces, que se poden instalar a maiores, e que melloran a experiencia de traballar con R.

Unha delas será a que utilizaremos nestas clases: o RStudio

Que é RStudio?

É unha **interface para traballar con R**, ou sexa, un programa que nos permite manexar R con máis comodidade que usando só R básico.

Ollo!! RStudio non calcula nada por si mesmo.

Para usalo debedes ter R e RStudio instalados.

Como se consegue RStudio? Canto custa? RStudio é software libre. Pódese obter na súa propia web: RStudio.com

Tamén ten unha versión de pago, pero a versión de software libre cubre as necesidades da inmensa maioría dos seus usuarios.

Como se consegue RStudio? (2)

Dentro da súa web RStudio.com, debedes acceder á sección de **Download** -> Download Rstudio.

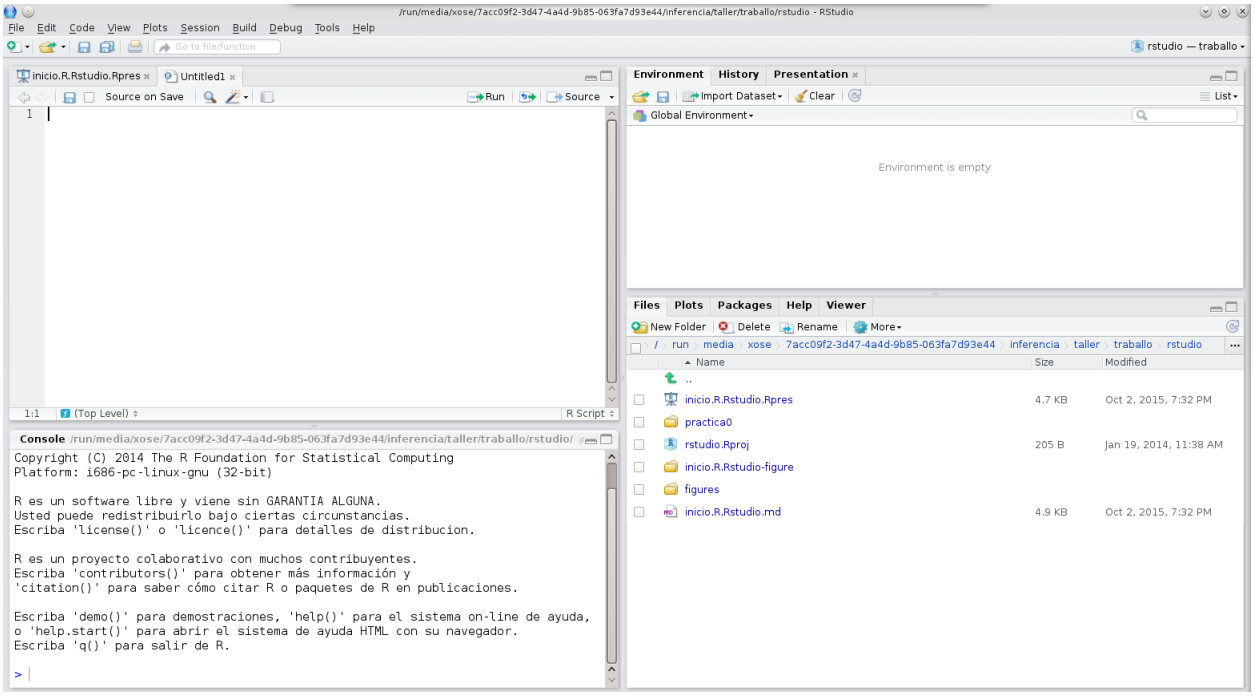
Aí escolledes a versión **Desktop, Open Source Edition**.

Despois simplemente baixar a do voso sistema operativo.

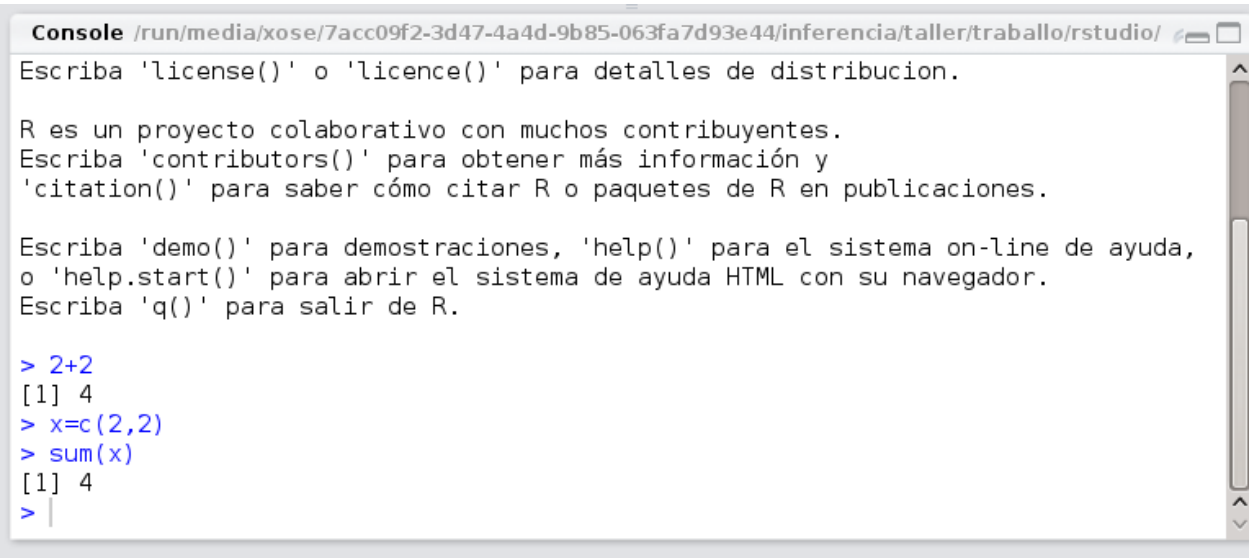
Traballo con RStudio RStudio está pensado para usar R en liña de comandos, sen menus, pero con moitas máis comodidades que usando o R básico.

Podedes ver na seguinte páxina unha figura coa interface de RStudio.

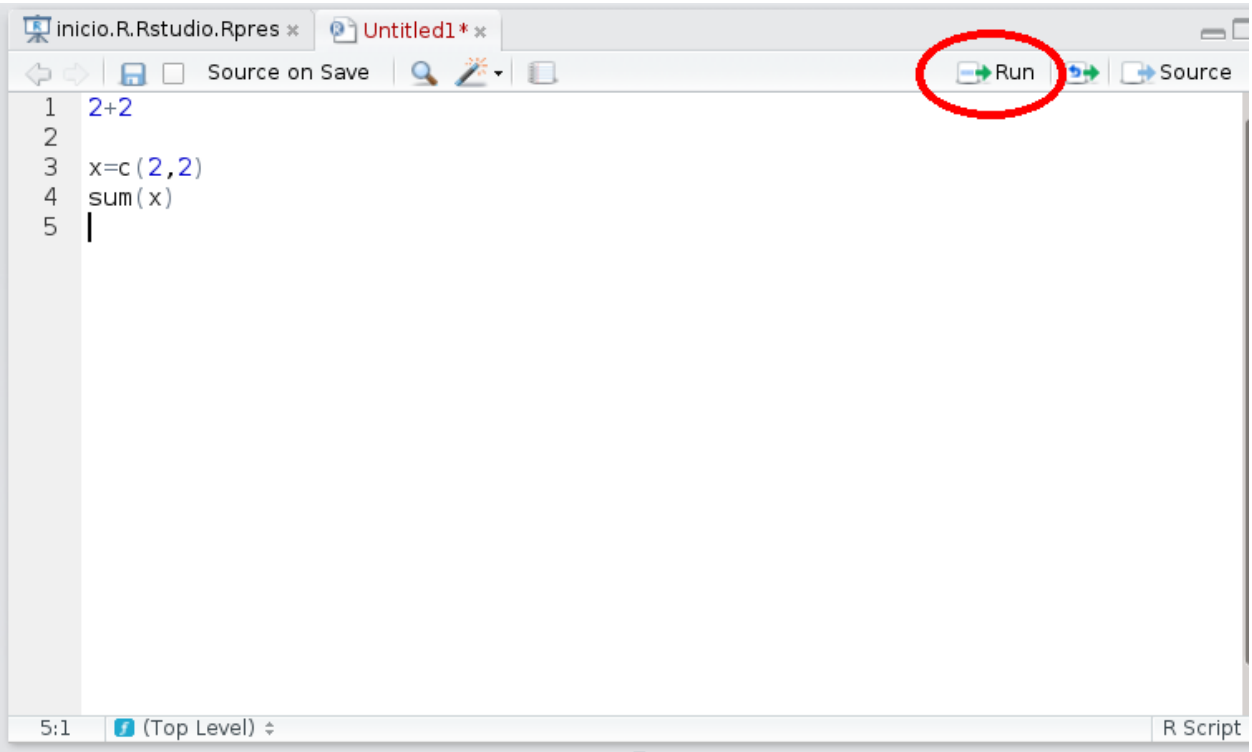
Vense: o **editor de texto**, a **consola**, o **ambiente** onde amosa as variables que manten en memoria, e o **xestor de ficheiros**



Traballo con RStudio: consola A **consola** equivale á pantalla que aparece no R básico, e nela poden escribirse os comandos e apareceran os resultados.



Traballo con RStudio: editor Con todo é máis práctico traballar mediante o editor.



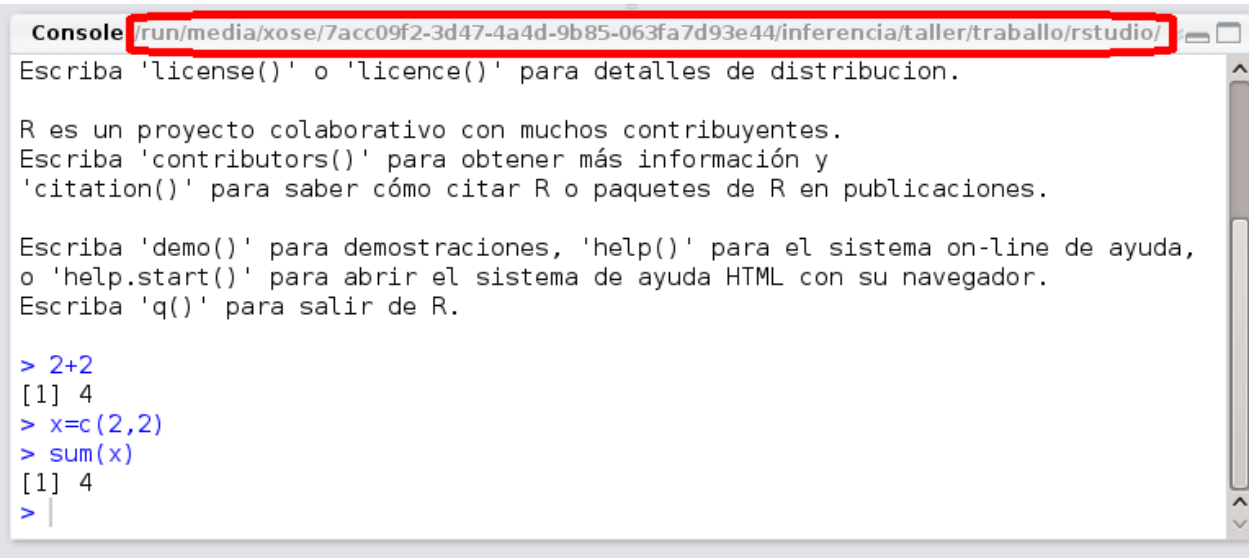
Aquí escríbense varias liñas de código e execútanse seleccionandoas e premendo en **Run**.

No editor **consérvanse as liñas de comandos** escritas, e **poden volver a executarse**, **correxirse**, **ampliarse** . . .

e gardarse como documento de código de R, que ven sendo un documento de texto, gardado cun nome que ten extensión ".r" ou ".R", por exemplo **exercicio1.R**

A interface de RStudio

Traballo con RStudio: consola A **consola** equivale á pantalla que aparece no R básico, e nela poden escribirse os comandos e apareceran os resultados.

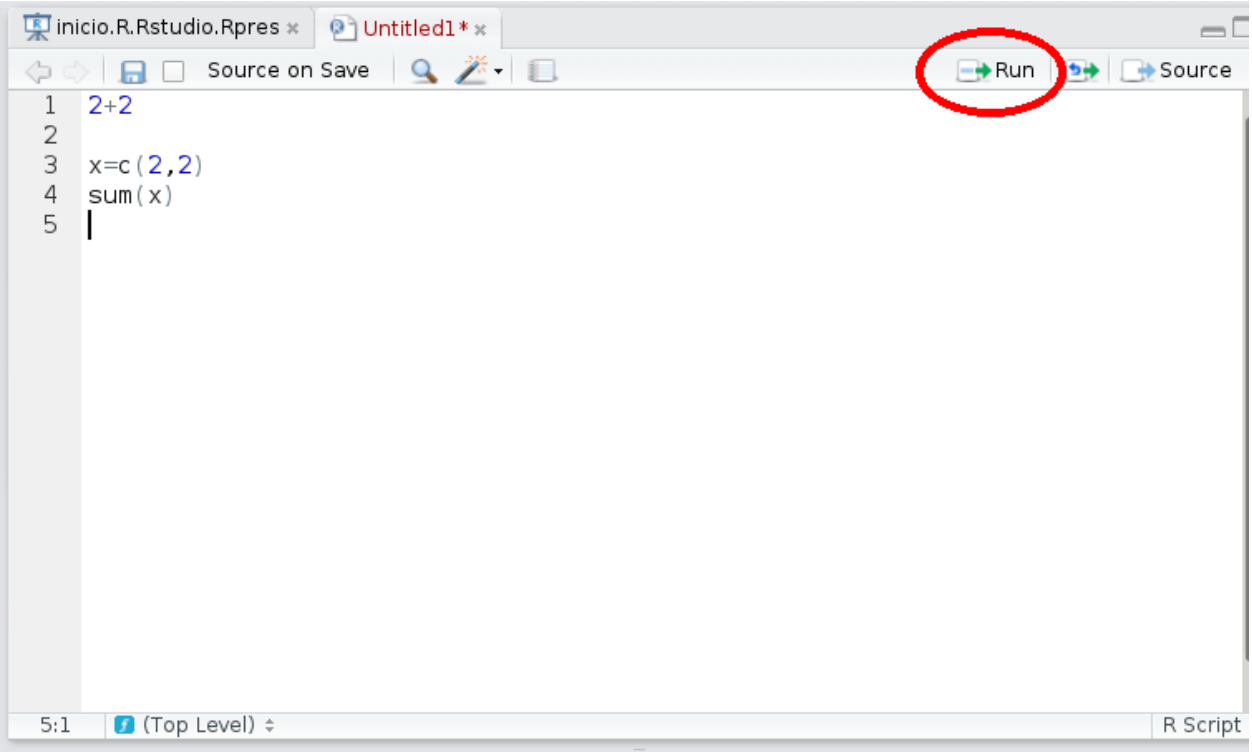


Tamén indica cal é o **directorio de traballo** actual *(marcado en vermello na figura)*, ou sexa, o cartafol **onde se leran e gardarán as cousas que use ou produza R**, como por exemplo o código que se vai usar ou as gráficas que se gardan

Mais o traballo con RStudio non é cómodo, posto que os comandos cada vez que se usan "desaparecen", non quedan almacenados de maneira cómoda

Isto dificulta facer cálculos con varias liñas de extensión, que poderían ser correxidos e ampliados se fosen colocados nun editor de texto.

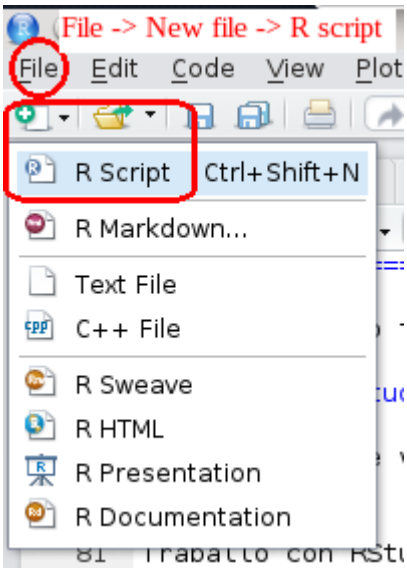
Traballo con RStudio: editor Por iso RStudio tamén trae un editor de texto.



Aquí escríbense varias liñas de código e execútanse seleccionandoas e premendo en **Run**.

Para usar o editor é necesario ter aberto un **script de código de R**

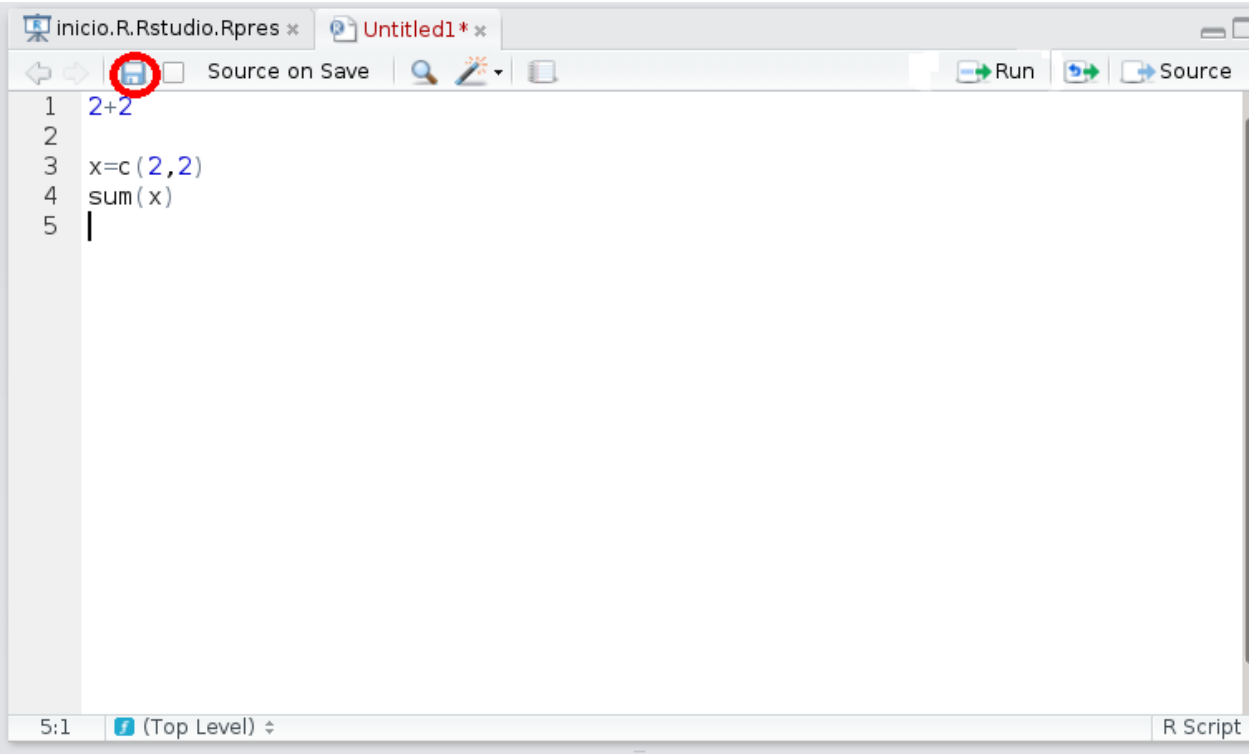
Podes abrir un xa feito no **xestor de ficheiros** ou abrir un novo no menú (**File->New File->R script**), ou no borón de **Novos documentos**:



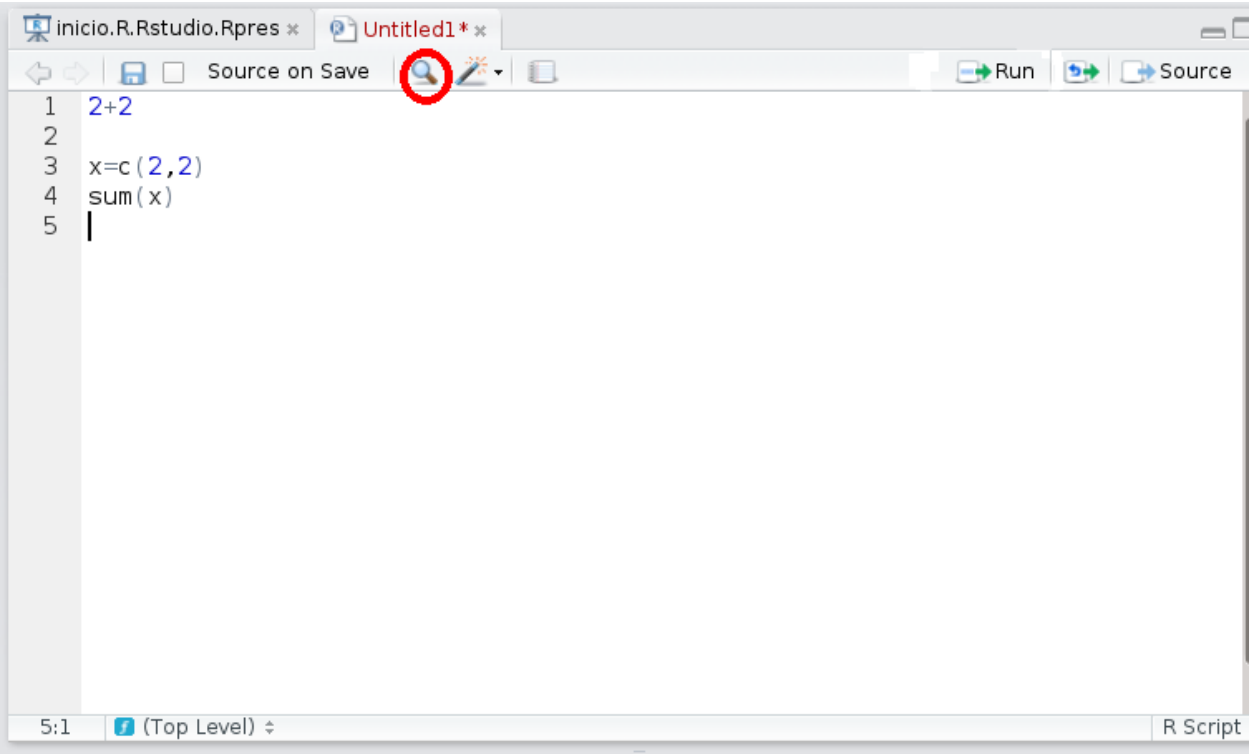
No editor **consérvanse as liñas de comandos** escritas, e **poden volver a executarse**, **correxirse**, **ampliarse** . . .

e gardarse como documento de código de R, que ven sendo un documento de texto, gardado cun nome que ten extensión ".r" ou ".R", por exemplo **exercicio1.R**

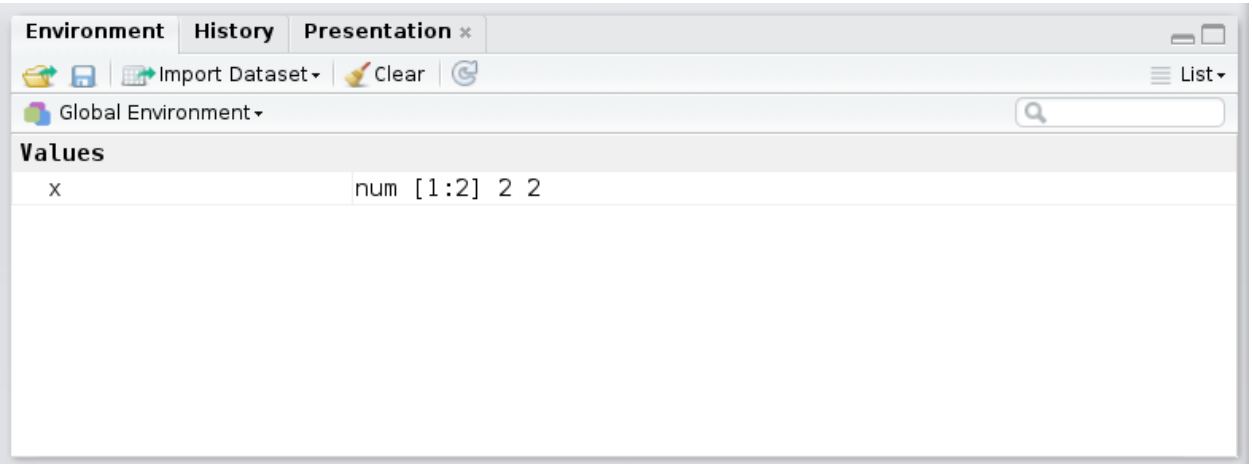
Para gardalo debe premerse na figura de disquete:



Outro instrumento útil do editor é **Buscar e reemprazar**, ao que se accede premendo na lupa do menu:

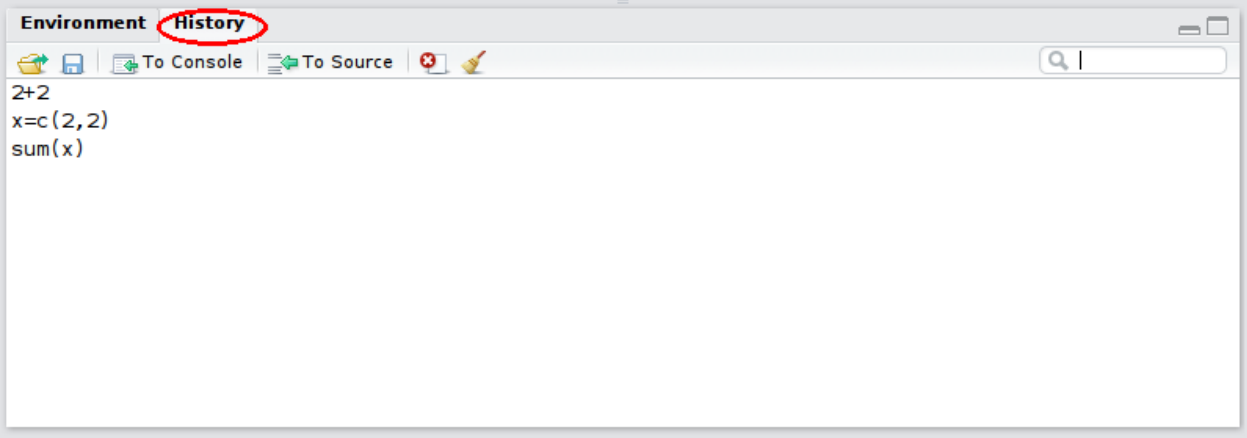


Traballo con RStudio: o Ambiente (*Environment*) Nesta parte da interface de RStudio pódese ver *Environment*, tamén denominado **espazo de traballo** (*Workspace*), ou sexa, o conxunto de obxectos que R conserva na memoria, e que habitualmente son as variables e resultados que lle ordenamos conservar na memoria.



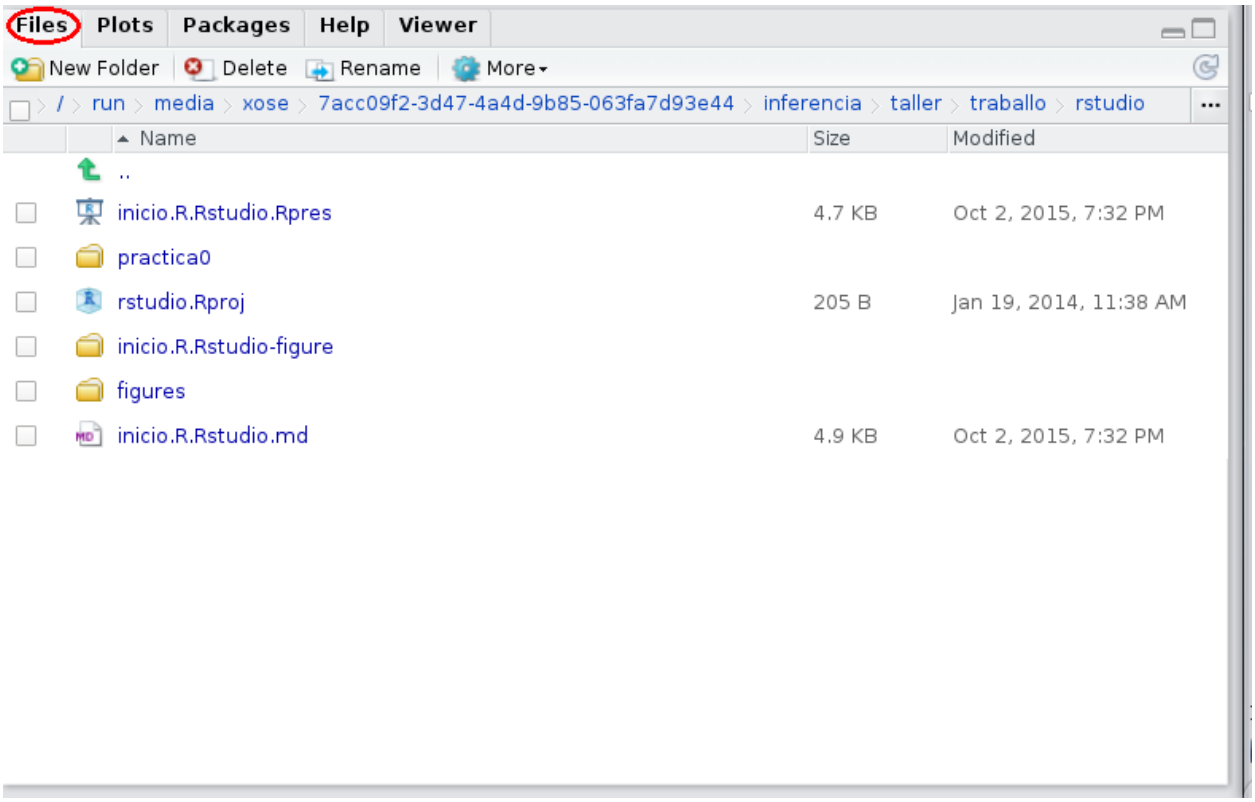
Na imaxe pode verse na memoria de R só hai un obxecto: chamáse **x**, conten 2 **núm**meros, o 1º un **2**, e o 2º outro **2**. Polo tanto amósanos que **x** é o seguinte vector **(2,2)**

Traballo con RStudio: o Ambiente (Historial) No cadro do amnbente tamén se pode acceder á pestana de **Historial** (*History*). accedendo a ela podemos ver o listado cos últimos comandos executados, e escollelos para reutilizalos de novo.

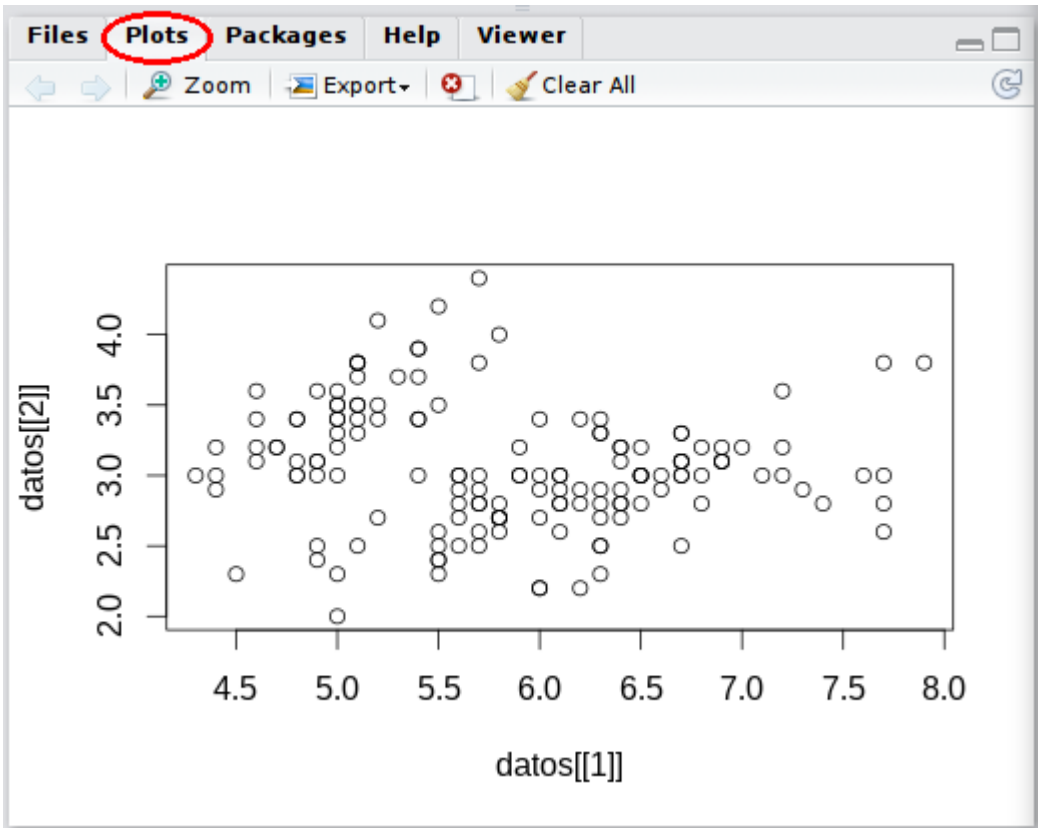


Traballo con RStudio: Xestor de ficheiros A 4ª parte da interface permite acceder a diferentes ferramentas, premendo nas pestanas superiores.

Por exemplo o **xestor de ficheiros** (**File**)



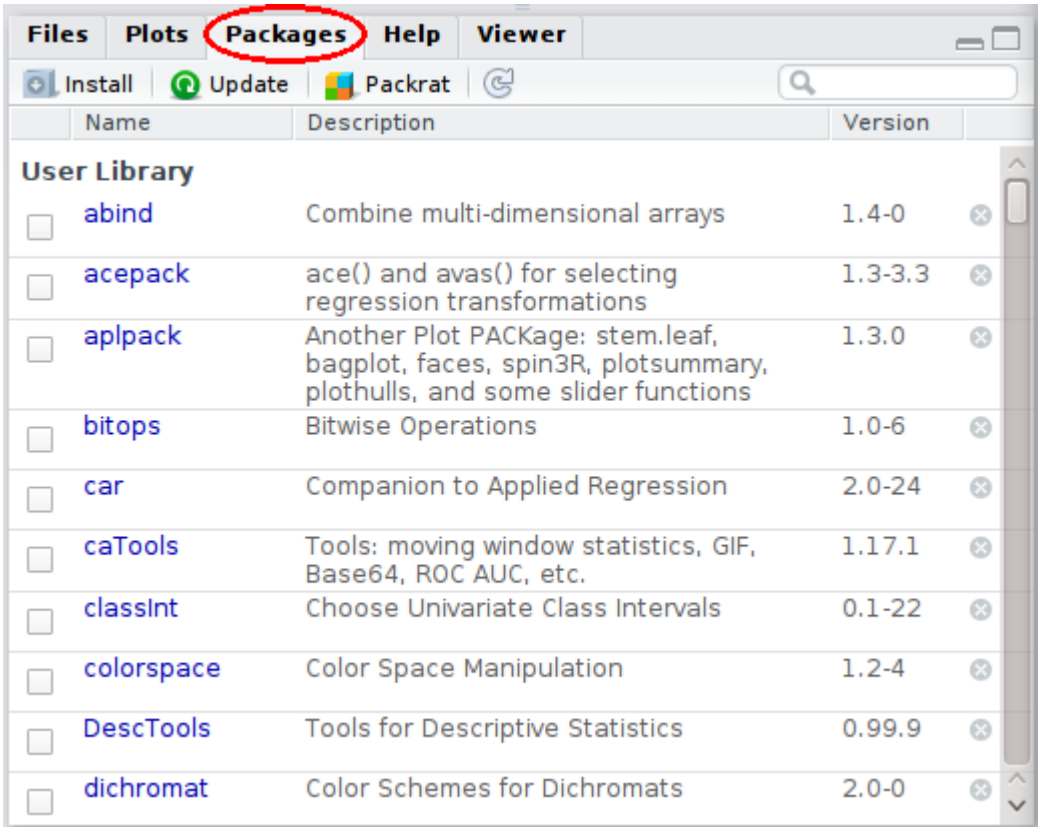
Traballo con RStudio: Xestor de gráficas O xestor de gráficas (**Plot**)



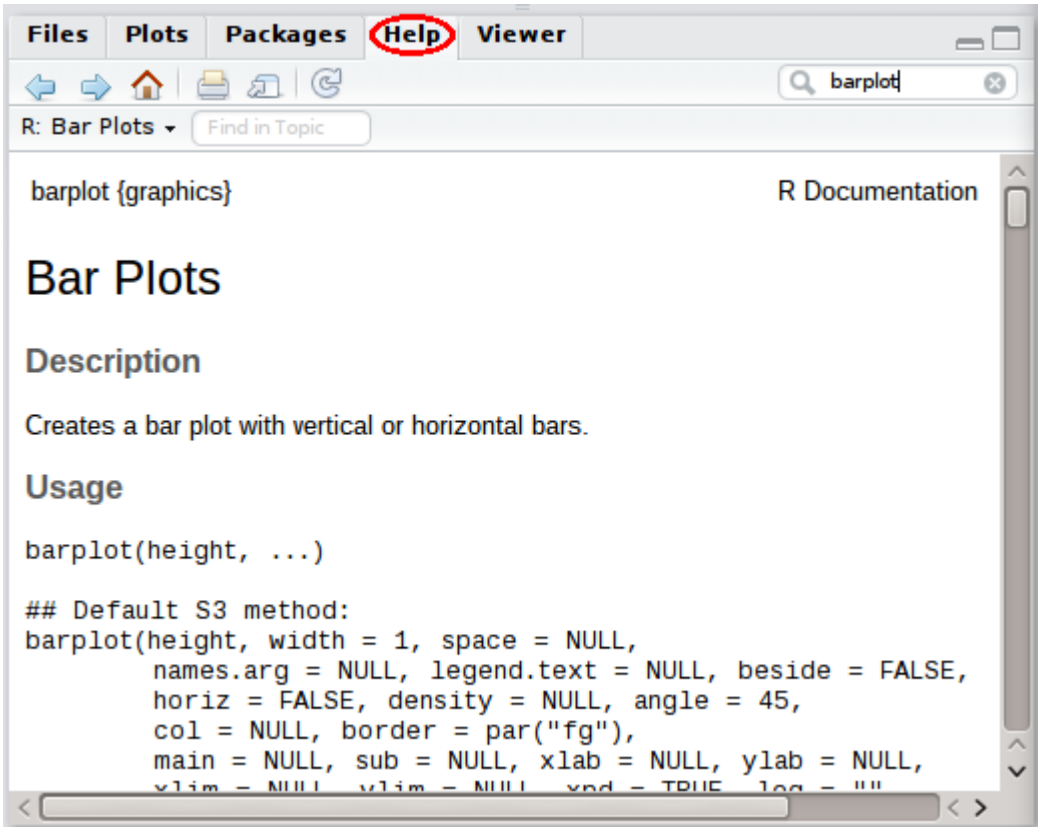
Que permite velas máis amplas (**Zoom**), ou gardalas para manexar con outro programa **(Export)**

Traballo con RStudio: Xestor de paquetes O **xestor de paquetes** (**Packages**) permite instalar extensións con técnicas estatísticas novas que non veñen incluidas no R básico.

Tamén amosa aqueles que xa temos e cales deles están cargados em memoria.



Traballo con RStudio: A axuda A **axuda** (**Help**) (*en ingles*) explica o que fai cada un dos comandos que nese momento estea na memoria de R.



Traballar con R

Ainda que se lle poden instalar sistemas de menu como o Rcommader

R foi pensado para traballar con **comandos**, ou sexa, para facer algo debedes indicarlle a orde apropiada.

```
#escribides na consola
2+2
#premedes ENTER

#e saevos na consola o resultado
```

```
## [1] 4
```

Os comandos que se introducen na consola pérdense, polo que para volver a usalos ou correxilos hai que teclealos de novo ou buscalos mediante as teclas de cursor.

Por esta razón é máis cómodo usar un editor de texto para introducir os comandos, xa que así seguen accesibles, e podemos volver a executalos ou correxilos. É por iso que o traballo en R faise case sempre mediante ambientes integrados coma o RStudio:

Aínda así a maior parte destes programas seguen manexando R a partir de comandos, polo que comezaremos a explicar a filosofía básica deste xeito de traballo.

O primeiro comando Un comando simple pode ser facer unha operación aritmética, por exemplo multiplicar 6 por 4.

```
#o símbolo * indica o producto, polo tanto
#escribiríades na consola
6*4
#premedes ENTER

#e sáevos na consola o resultado
```

```
## [1] 24
```

Outras operacións. As habituais: +, -, *, /, ^ (potencia)

```
6+4
```

```
## [1] 10
```

```
6-4
```

```
## [1] 2
```

```
6*4
```

```
## [1] 24
```

```
6/4
```

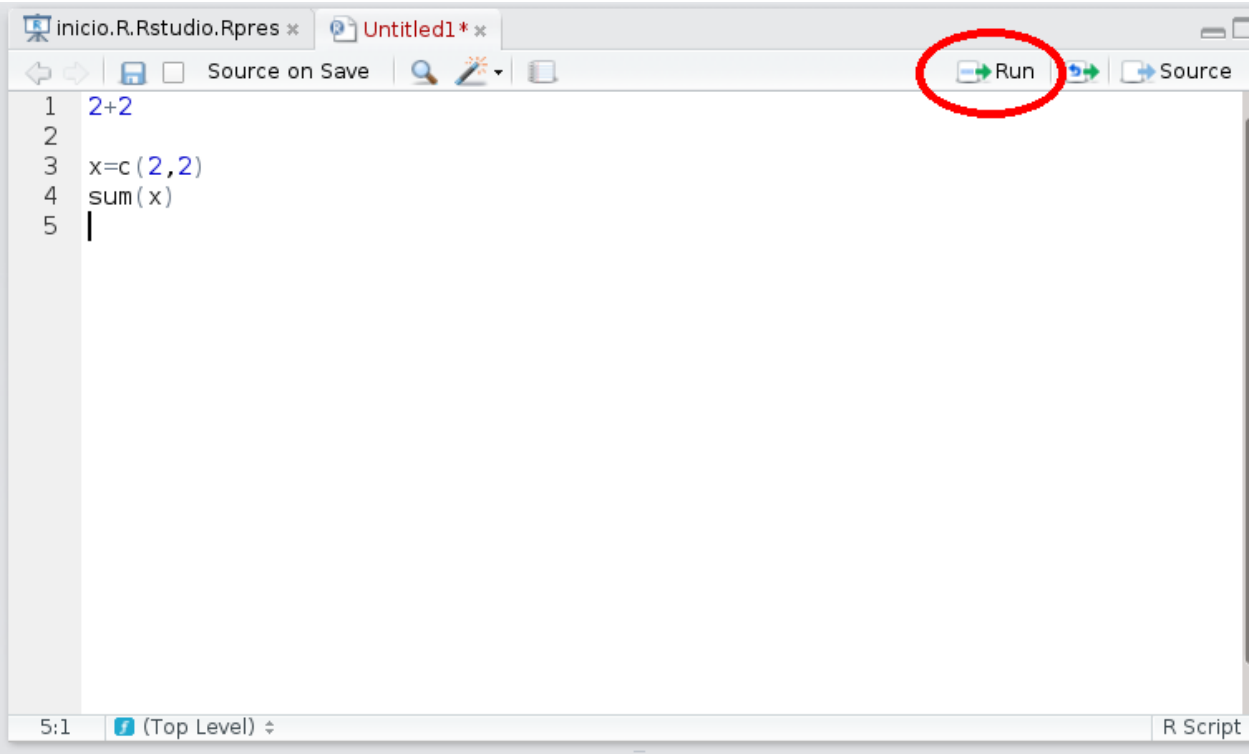
```
## [1] 1.5
```

```
6^4 #seis elevado a catro
```

```
## [1] 1296
```

Por que usar un editor de texto? Se queremos facer as operacións anteriores con outros numeros, por exemplo **3** e **7** teríamos que volver a escribilas e executalas.

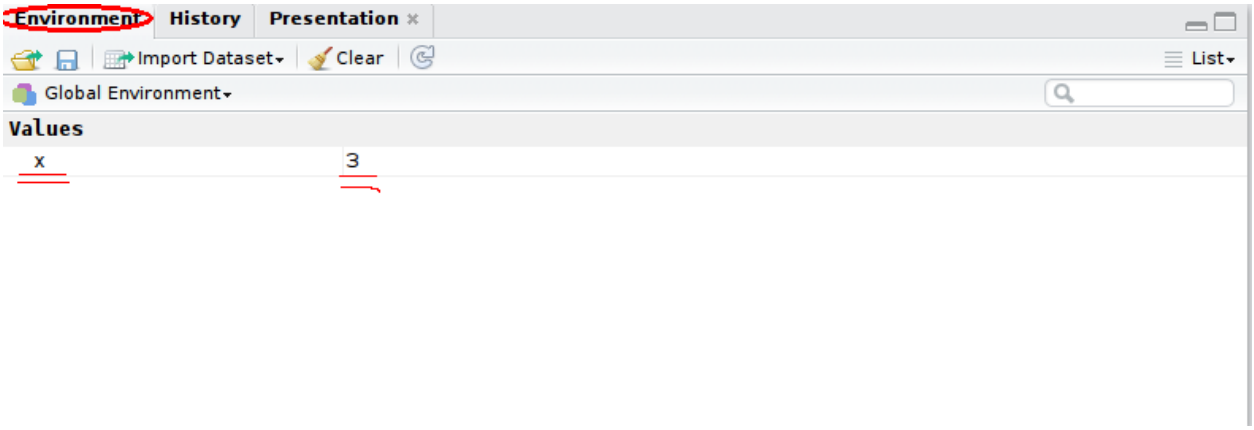
Nembargante, se estivésemos nun editor de texto abondaría con **buscar e sustituir**, para que a cousa fose máis rápida, polo que a partir de agora a explicación vai supoñer que os diferentes comandos son escritos e modificados sobre o editor de texto do RStudio onde serán executados seleccionándoos e premendo **Run**.



O primeiro obxecto Tamén facilita o traballo usar *variables*, ainda que en **R** reciben o nome de ***obxectos***

```
#Se escribes o comando
x=3
```

Non aparecerá nada na consola, pero se miras o espazo de traballo veras que hai un *obxecto* na memoria chamado **x**, que é **num**érico, e tomou o **valor 3**

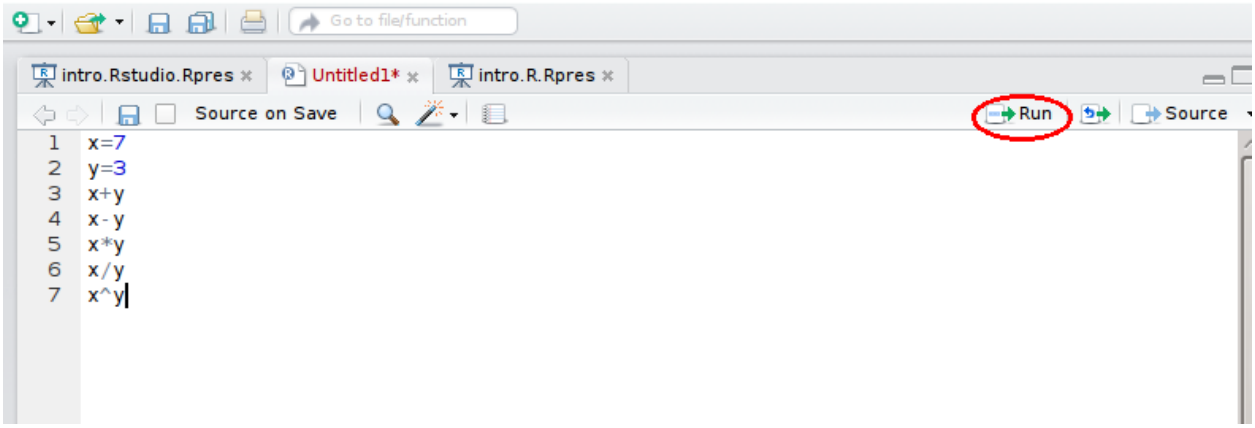


Usando editor + obxectos Combinando o editor de texto e o manexo de obxectos pódense construir plantillas complexas coas que realizar diferentes cálculos e aplicar técnicas estatísticas.

Por exemplo, sumar, restar, mutiplicar, dividir e facer unha potencia con dous números podia ser así:

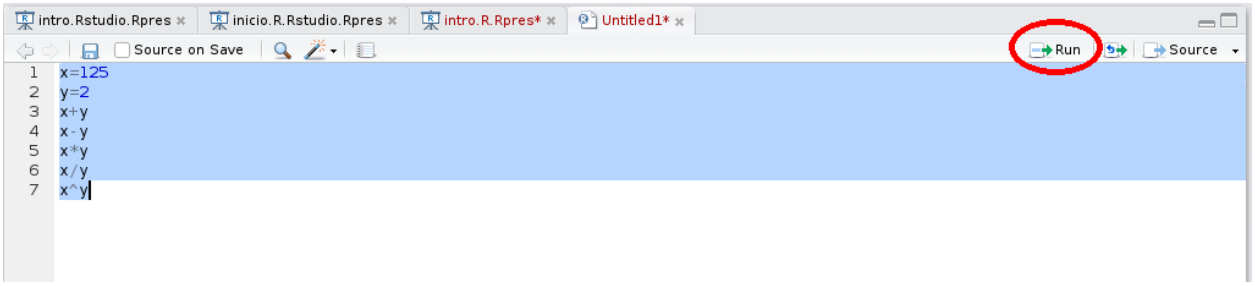
```
x=7
y=3
x+y
x-y
x*y
x/y
x^y
```

que visto no editor sería:



E produciria o seguinte resultado na consola:

Se temos o código (os comandos) escritos no editor abondaría con cambiar os valores de x e y para ter uns resultados novos.



Os resultados sairán agora na consola:

Usar unha plantilla Se hai cálculos que repetimos con frecuencia compensa ter unha plantilla de comandos que os realice, de xeito que modificando os obxectos con datos vaia aplicando os cálculos en diferentes situacións.

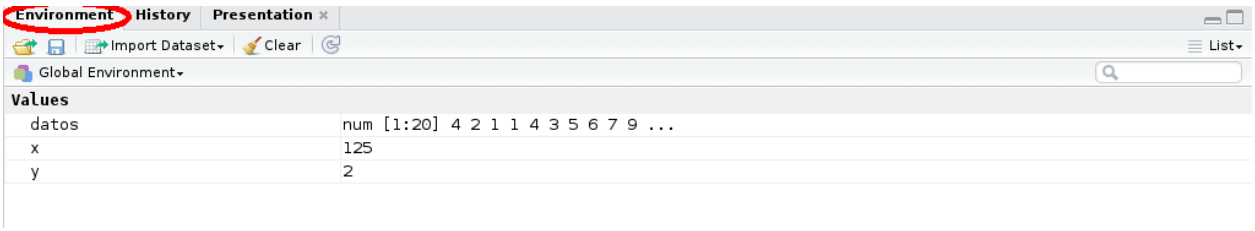
A plantilla de exemplo que se vai crear fai o seguinte:

- Crea un vector de números, que son os datos aos que se lle aplican os cálculos
- Fai unha táboa dos datos
- Fai un diagrama de barras
- Calcula a media aritmética, a mediana e a desviación típica

1º Crear un vector con datos numéricos Imos chamarlle ao vector **datos**. Os valores asígnanse mediante o comando **c()**, colocando os datos entre paréntese e separados por comas (*Ollo!, en R os decimais fanse con un punto*)

```
datos=c(4,2,1,1,4,3,5,6,7,9,12,2,1,5,8,8,11,9,0,4)
```

Se o executades non aparecerá ningún resultado na consola, pero si veredes unha referencia no espazo de traballo, indicando que existe en memoria un obxecto chamado **datos**, que ten 20 números, e amosa os primeiros:



2º facer unha táboa de datos Agora podedes usar o nome do obxecto "**datos**" para aplicarlle comandos (*funcións*) que fagan cousas con eles.

O primeiro será presentalos nunha táboa estatistica, usando o comando **table()**

```
table(datos)
```

```
## datos
##  0  1  2  3  4  5  6  7  8  9 11 12
##  1  3  2  1  3  2  1  1  2  2  1  1
```

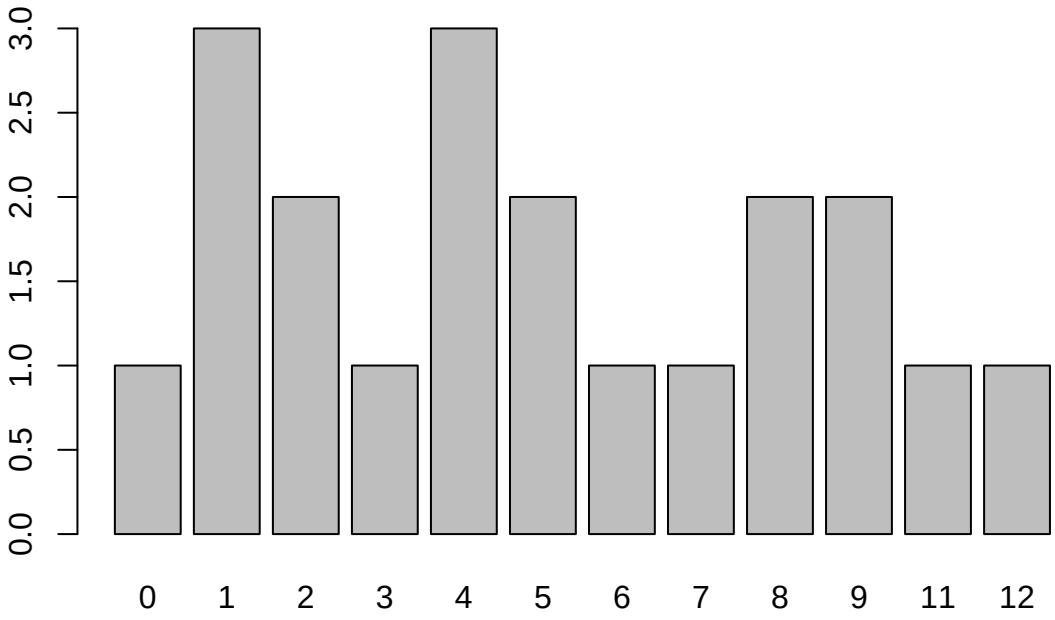
Lembrovos que o apropiado é ir escribindo estes comandos no editor de texto, e executándoos premendo en **Run**

3º Representar os datos cun diagrama de barras O comando para o diagrama de barras é **barplot()**, colocando dentro do paréntese o nome do obxecto que contén os datos

O diagrama de barras sae bastante simple. Disponse de diferentes parámetros que permiten poñerlle textos, cambiarlle as cores, o tamaño dos eixes. . . .

Pero nesta presentación só veremos o caso máis elemental

```
barplot(table(datos))
```




```
#media aritmética
mean(datos)
```

4º Calcular algúns parámetros estatísticos

```
## [1] 5.1
```

```
#mediana
median(datos)
```

```
## [1] 4.5
```

```
#desviación típica
sd(datos)
```

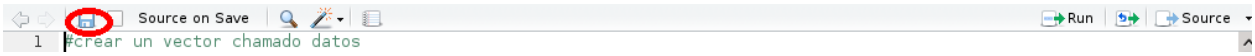
```
## [1] 3.537766
```

Como queda a plantilla de comandos Aquí podedes ver a plantilla de comandos:

```
#Crear o obxecto datos, para almacenar a variable
datos=c(4,2,1,1,4,3,5,6,7,9,12,2,1,5,8,8,11,9,0,4)

#tabular os datos
table(datos)

#representalos nun diagrama de barras
barplot(table(datos))

#Calular algúna parámetros
```

```
#media aritmética
mean(datos)

#mediana
median(datos)

#desviación típica
sd(datos)
```

Agora **pódese gardar** premendo no disquete que está sinalado en vermello, e poñéndolle un nome calquera que remate en **".R"** ou **".r"**



Usar a plantilla para outros datos Simplemente **cambiando a liña onde se crea o obxecto datos**, e poñendo entre os parénteses os novos valores, poderíamos refacer os cálculos con novos números: **seleccionar todo** e premer **Run**

Por exemplo co novo vector de datos **3,3,2,1,2,3,4,5,5,2,3,2**:

```
#Crear o obxecto datos, para almacenar a variable
datos=c(3,3,2,1,2,3,4,5,5,2,3,2)

#tabular os datos
table(datos)

#representalos nun diagrama de barras
barplot(table(datos))

#Calular algúna parámetros
#media aritmética
mean(datos)

#mediana
median(datos)

#desviación típica
sd(datos)
```

Agora, para rematar dúas cousas máis:

- **Os comentarios (#)**

Cando aparece o símbolo # ao inicio dunha liña, ou incluso polo medio dela, R entende que comeza **un comentario**, ou sexa, unha anotación que sirve para explicar cousas sobre o que se está a facer pero non debe ser executado.

No RStudio que estou usando os comentarios detéctaos e ponnos de cor verde, como podedes ver nas imaxes

- **Sensibilidade ás maiúsculas**

R é **sensible ás maiúsculas**, isto significa que os símbolos **"A"** e **a** son diferentes, polo tanto os obxectos **DATOS** e **Datos** ou **datos** serán considerados como cousas distintas. Ollo con esta carácterística, por que é fácil que produza erros.

Unha sesión de traballo con R

Traballar con R: 1º Indicar o CARTAFOL DE TRABALLO Lembrade!

Antes de comezar a traballar, xusto **despois de abrir o RStudio** é conveniente indicarlle **cal é o cartafol de traballo**

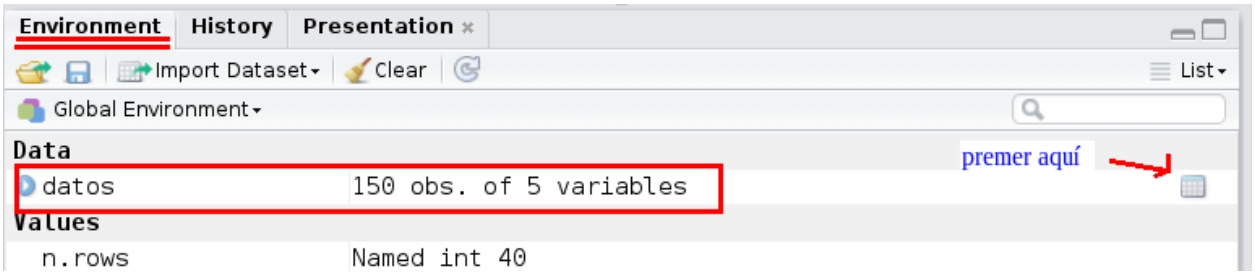
Podedes ver unha breve explicación de como se fai:

ANEXO1: configurar o cartafol.de.traballo

2º Cargar datos: O máis habitual será a partir dun ficheiro, que se ten formato csv lese co seguinte comando:

```
#Por exemplo:
datos=read.csv2("csv/iris.csv")

#le o ficheiro iris.csv, e garda os seus valores nun data.frame chamado datos,
#co cal poderemos traballar a partir de agora
```

Despois de executar o comando anterior temos en memoria un obxecto **datos** que ten estructura de data frame.



Como son os datos? A información que proporciona RStudio tamén se pode obter con comandos:

```
#nomes das variables?
names(datos)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```
#número de datos? número de variables
dim(datos) #dimension: nº de filas e nº de columnas
```

```
## [1] 150   5
```

```
#Estructura dos datos
str(datos)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
#Como son? as 6 primeiras filas
head(datos)
```

```
##   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
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
#Como son? as 6 derradeiras filas
tail(datos)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```r
#Como son? algunhas características
summary(datos)
```

```
##   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
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```r
#Como son?

#vese que o 5º non é numerico (é un factor, o nome da especie)
#vense os máximos e minimos, os cuartis, a mediana e a media
#non falta ningun dato por que non aparecen NA's
```

Algunha preparación para os datos Os datos **iris** son o ancho e o longo dos pétalos e sépalos de tres especies de flores diferentes. Vou separar eses datos en exemplares con longo de pétalo mais grande e longo de pétalo máis pequeno. Se llo fago ao data.frame enteiro, quédanme dúas coleccions de datos para comparar.

Escollo como petalos grandes os que teñan tamaño 4 ou maior e o resto pétalos pequenos:

```
datosg=datos[datos$Petal.Length>=4,]
#gardo en datosg aquelas filas nas cales Petal.Length sexa maior ou igual a 4
#podería facerse tamén indicando o nº de columna. Aplicarei isto para os petalos pequenos
datosp=datos[datos[3]<4,]
```

Creacion de variables separadas Vou estudar o longo dos sepalos e as especies. Para iso vou separalas en variables individuais:

```
lsg=datosg[[2]]
lsp=datosp[[2]]


espg=datosg[[5]]
espp=datosp[[5]]
```

tamén se pode traballar directamente coas columnas do data.frame, pero para un nivel inicial é práctico extraer as variables do data.frmae

Como se reparten as especies? Están igualmente repartidas as especies entre os dous grupos?

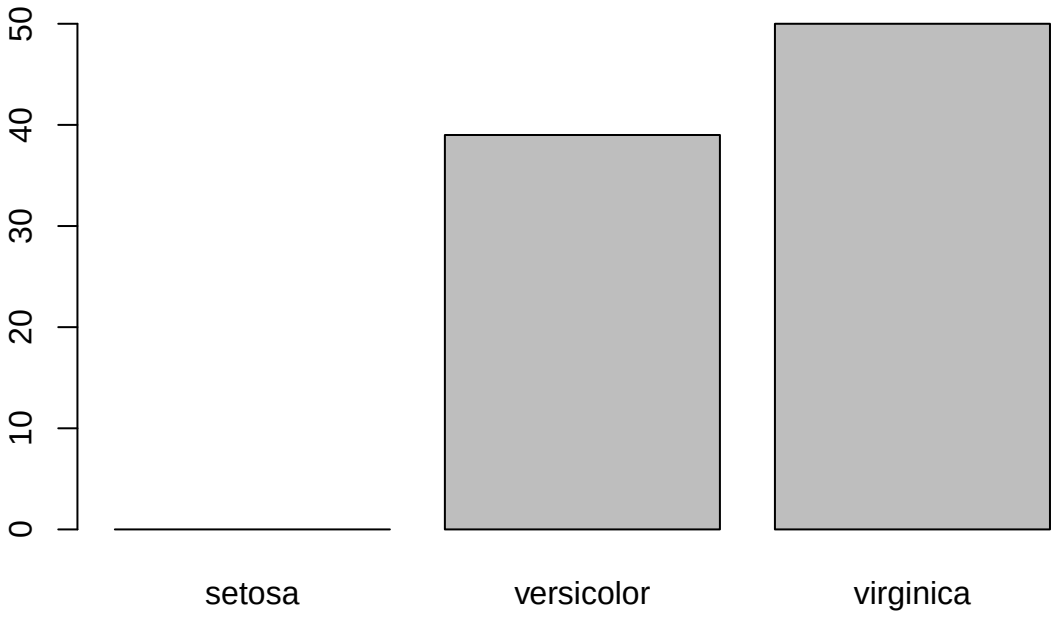
```
#mirarei con táboas:
table(espg)
```

```
## espg
##     setosa versicolor  virginica
##          0         39         50
```

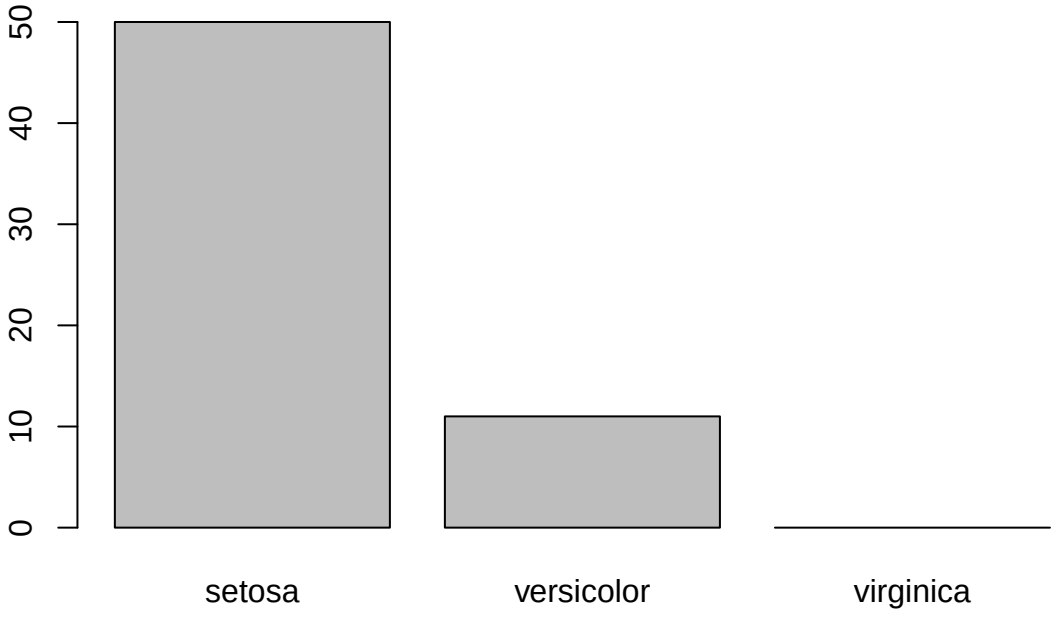
```
table(espp)
```

```
## espp
##     setosa versicolor  virginica
##         50         11          0
```

```
#Ou con diagramas de barras:
barplot(table(espg))
```




```
barplot(table(espp))
```



3º Algúns parámetros (media aritmética e todo iso)

```
#tamaños medios de sepalo
mean(lsg)
```

```
## [1] 2.919101
```

```
mean(lsp)
```

```
## [1] 3.259016
```

```r
#medianas e cuartis
quantile(lsg)
```

```
##   0%  25%  50%  75% 100%
##  2.2  2.8  3.0  3.1  3.8
```

```r
quantile(lsp)
```

```
##   0%  25%  50%  75% 100%
##  2.0  3.0  3.4  3.6  4.4
```

```r
#desviación tipica
sd(lsg)
```

```
## [1] 0.3125687
```

```r
sd(lsp)
```

```
## [1] 0.5087164
```

```r
#coeficiente de variación de pearson

sd(lsg)/mean(lsg)
```

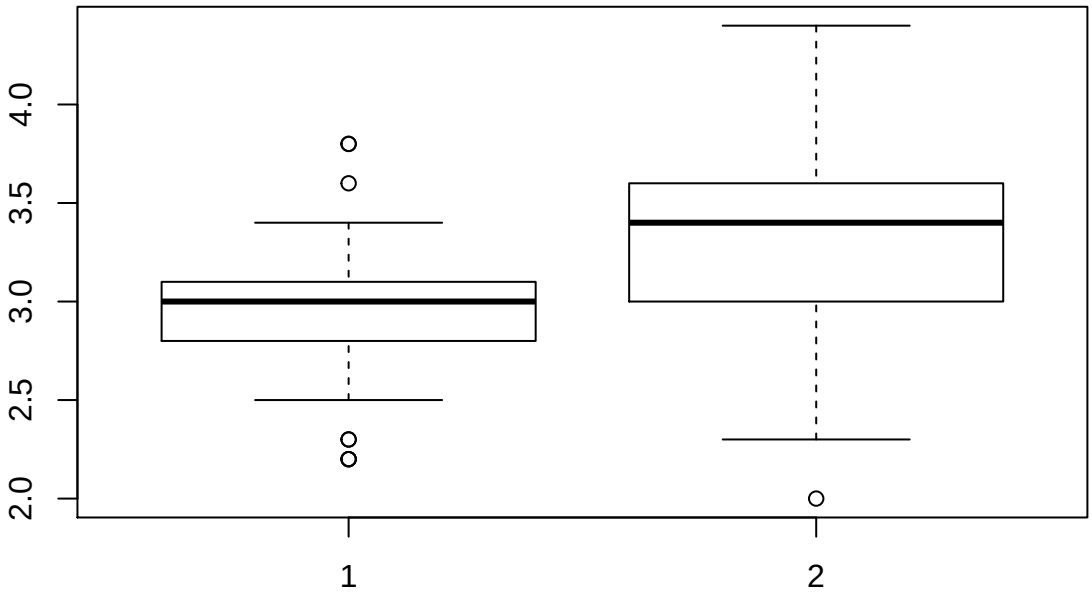
```
## [1] 0.107077
```

```r
sd(lsp)/mean(lsp)
```

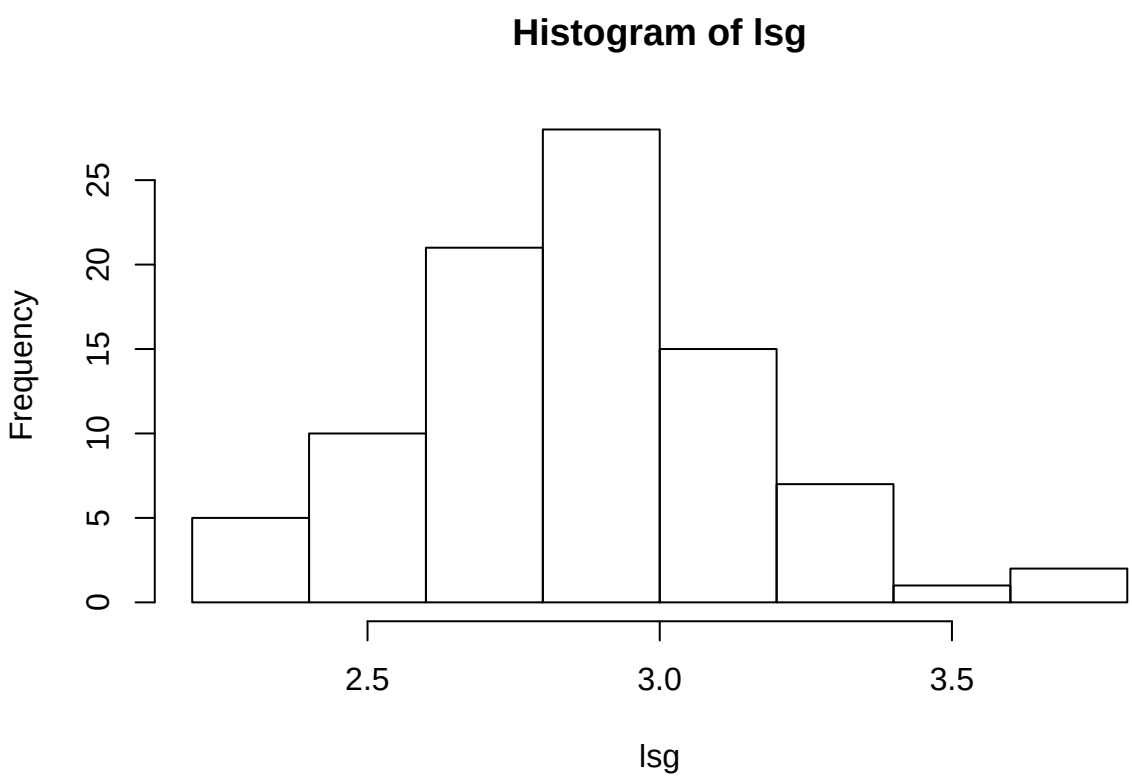
```
## [1] 0.1560951
```

4º Algunha gráfica Algunhas gráficas simples

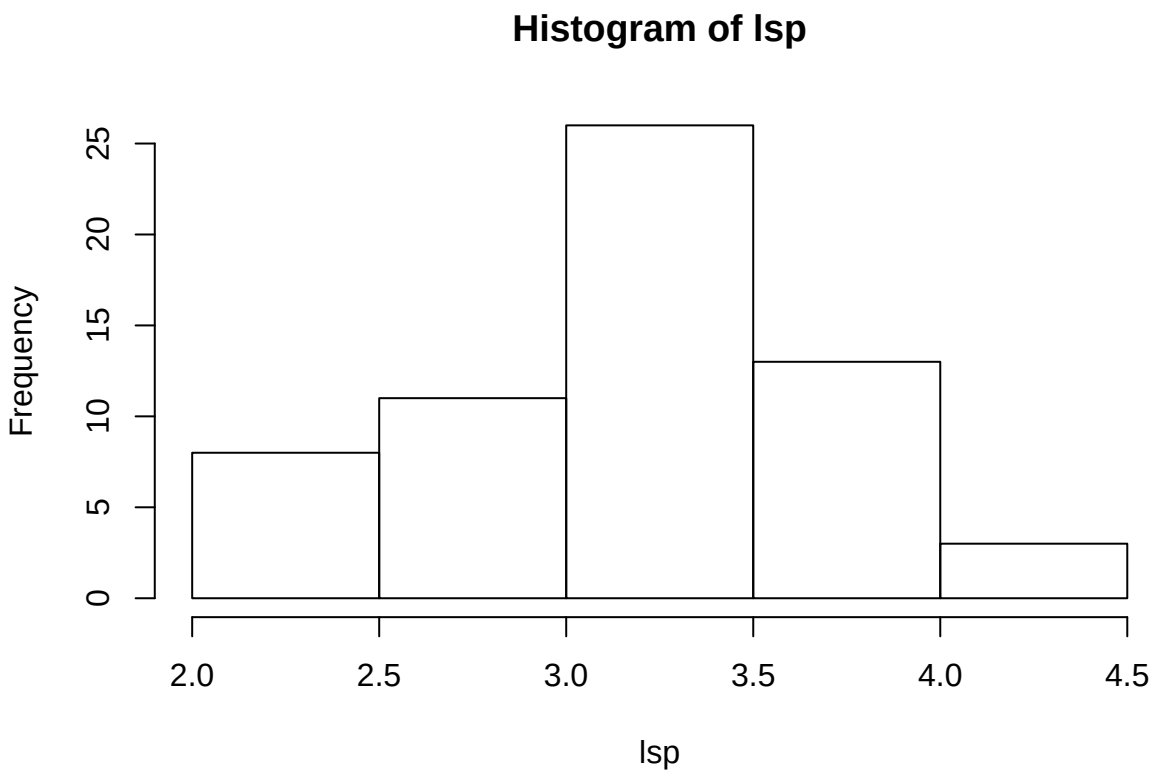
```r
#diagrama de caixa
boxplot(lsg,lsp)
```



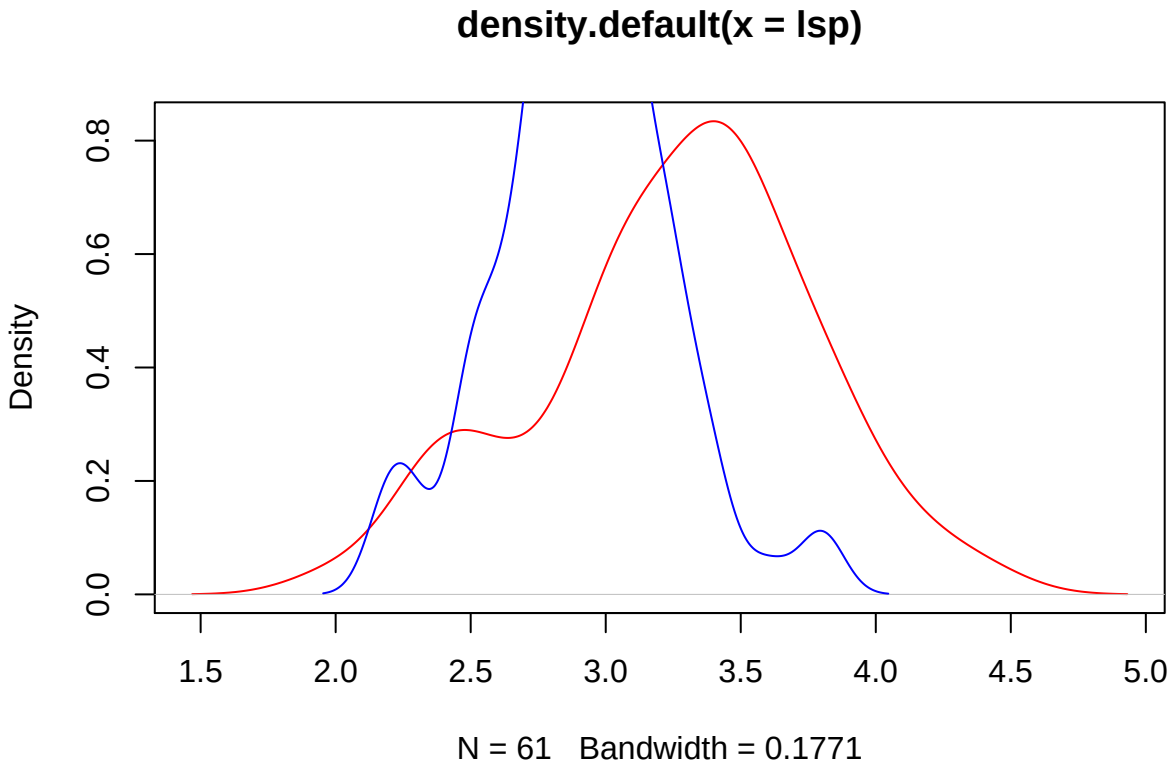
```r
#histograma
hist(lsg)
```



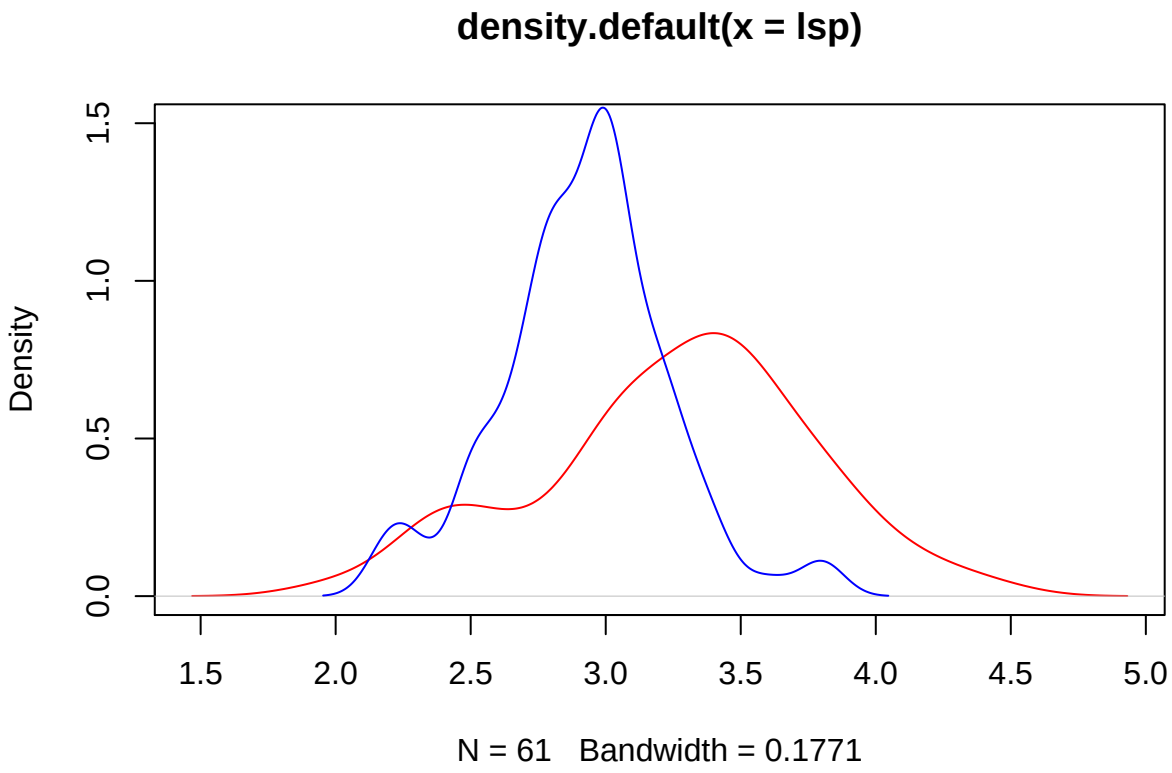

```r
#histograma
hist(lsp)
```



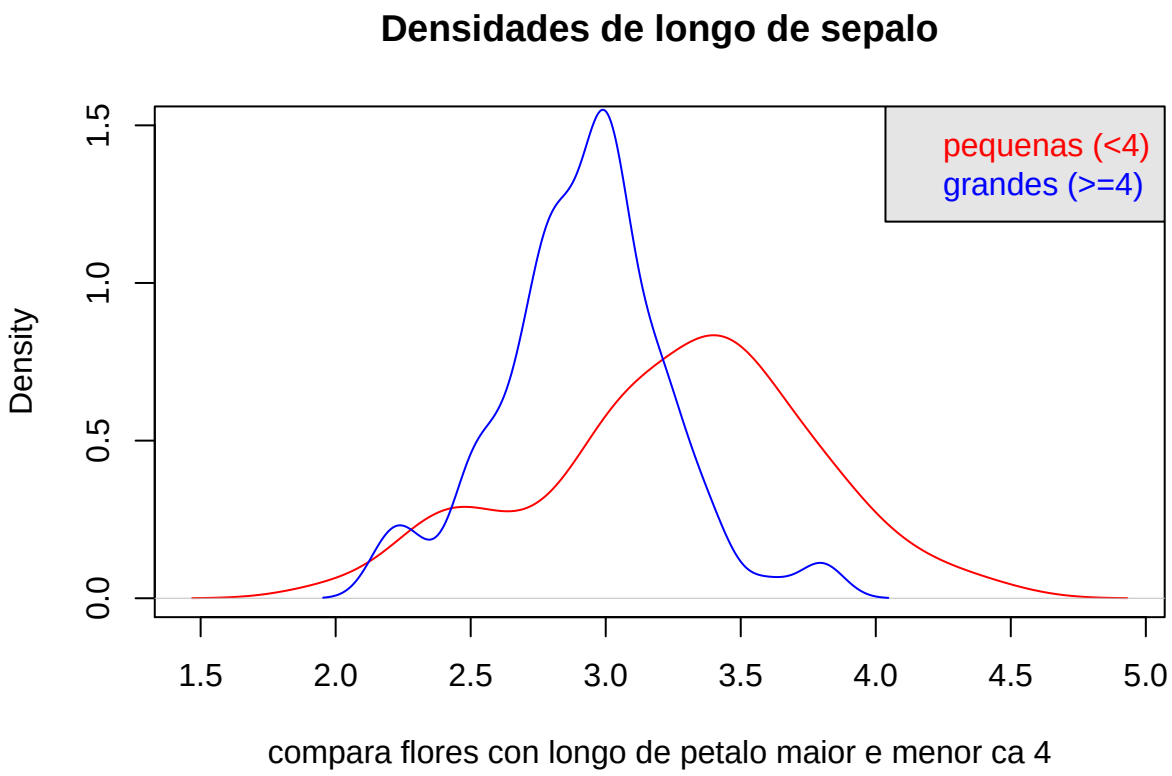

```
#comparar densidades
plot(density(lsp),col="red")
lines(density(lsg),col="blue")
```



```r
#para que quede ben hai que aumentar o eixe Y

plot(density(lsp),col="red",ylim=c(0,1.5))
lines(density(lsg),col="blue")
```




```r
#e se lle cambiamos os nomes e lle poñemos unha lenda queda moito mellor
plot(density(lsp),col="red",ylim=c(0,1.5),main="Densidades de longo de sepalo"
     ,xlab="compara flores con longo de petalo maior e menor ca 4")
lines(density(lsg),col="blue")

legend("topright", c("pequenas (<4)","grandes (>=4)"), text.col = c("red","blue"),bg = "gray90")
```


Densidades de longo de sepalo
pequenas (<4)
grandes (>=4)
Density
0.0
0.5
1.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
compara flores con longo de petalo maior e menor ca 4

ANEXO 1:

Configurar o cartafol de traballo

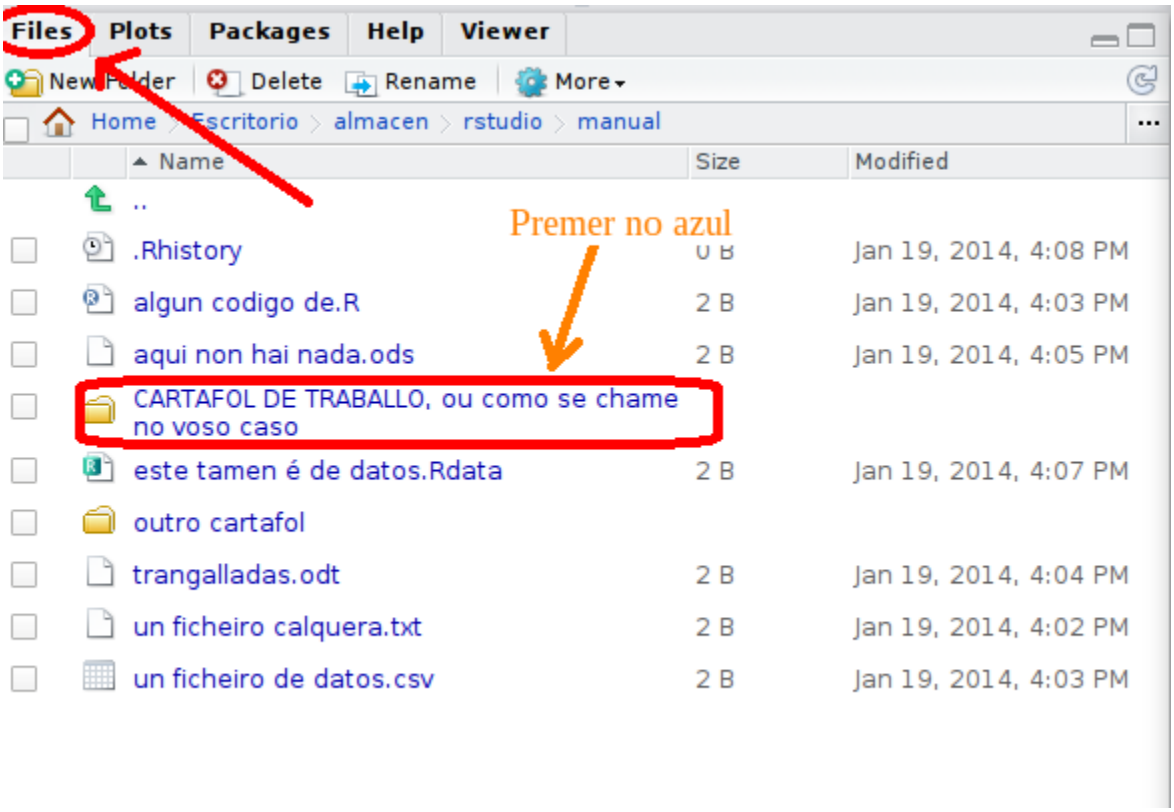
É algo que se debe facer sempre, ao inicio de calquera sesión de traballo con R

Por que se necesita un cartafol de traballo? **R** busca os ficheiros e gárdaos sempre no mesmo cartafol, que se denomina o seu **cartafol ou directorio de traballo**.

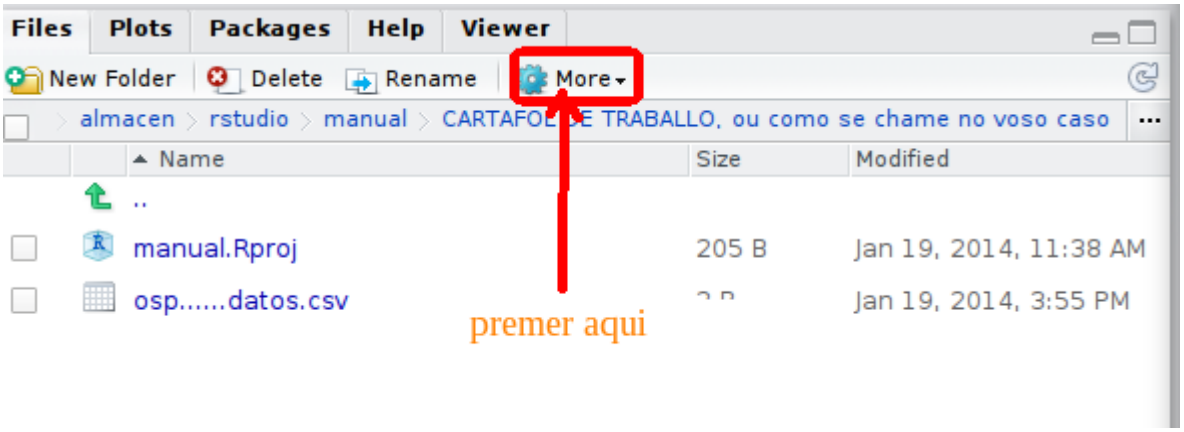
O directorio que é escolle depende do sistema operativo, e tamén de da maneira de abrilo.

Por iso, se temos os datos, o código, outras cousas nun directorio, debemos indicarllo. Ademais vai ser aí onde garde todo o que lle mandemos gardar.

Indicar a R cal é o cartafol de traballo Unha vez teñamos decidido cal é o cartafol de traballo, se estamos con RStudio debemos buscalo no **xestor de ficheiros** e acceder a el:



Unha vez esteades dentro del, usando o **menu do xestor de ficheiros** premedes en **More**



e despois escolledes a opción **Set as Working Directory**

Con eses pasos, xa tedes configurado o cartafol de traballo

ANEXO 2:

R: paquetes (packages)

R trae na súa **instalación básica** unha gran cantidade de funcións con parámetros estatísticos e métodos. Con eles pódese traballar nada máis arrincar e cubren toda a **Estatística Basica e un gran variedade de métodos estatísticos** de uso habitual.

Todas estas funcións **soen agruparse en paquetes (' *packages* ')**, xeralmente en función de afinidades entre elas (funcións para series de tempo, funcións para multivariante, funcións para análise cluster, funcións e datos usados en determinado libro, ...).

Cando iniciamos o R cárganse na memoria os paquetes '*básicos* ', cos cales se cobre a estatística 'básica', e con ela a maioría das necesidades dos seus usuarios.

Outro tipo de **paquetes** son os '***recomendados***'. Estes páquetes instálanse cando instalamos o R, pero non se cargan na memoria ao iniciar (para non sobrecargala). Se necesitamos algunha función presente nestes paquetes teremos que cargalo especificamente co comando:

```
library("nome de paquete")
```

Por exemplo ***lattice***, un paquete para gráficos:

```
library("lattice")
```

Ademais, **o feito de que R sexa unha linguaxe de programación permite desenvolver novos métodos e funcións**, de feito soe ser bastante pouco o tempo que pasa entre que se desenvolve unha nova técnica estatística e esta aparece dispoñible para o R.

Por esta razón existe **un terceiro tipo de paquete**, os '***contribuidos***', trátase de bibliotecas de funcións elaboradas por usuarios para cubrir as súas propias necesidades e postas á disposición da comunidade na propia web do R. Ainda que deben cumprir uns determinados requisitos de calidade para que sexan admitidos.

Dentro deste último grupo é onde aparecen as técnicas máis novidosas ou minoritarias.

Traballo con paquetes

Diferentes comandos para xestionar os paquete:

```
library(MASS)  #Cargar o paquete MASS, para traballar con él
library()   #indica os  paquetes que xa están ou podes cargar en memoria
search()    #paquetes que xa están cargados na memoria
ls("package:MASS")  #lista as funcions que posue o paquete MASS
```

Para instalar paquetes desde a internet, por exemplo o ***Rcmdr,*** úsase o comando ***install.packages()***:

```
install.packages('Rcmdr', dependencies=TRUE)    #instala desde internet o paquete Rcmdr, indicando adema

install.packages('Rcmdr')   #instala desde internet o paquete Rcmdr, pero non instala tódolos outros pa
```

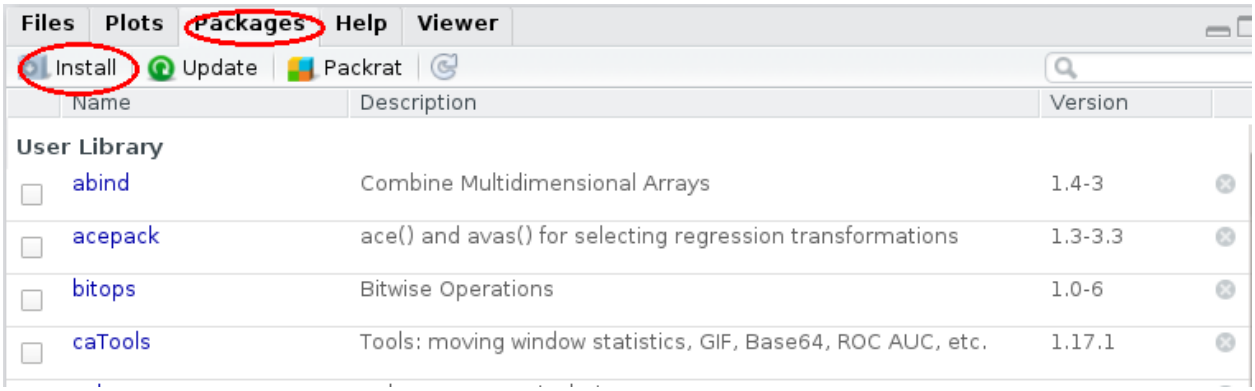
Dentro de moitos paquetes aparecen coleccions de datos empregados sobretodo para ilustrar os exemplos:

```
data()       #lista as coleccions de datos que actualmente estan en memoria
```

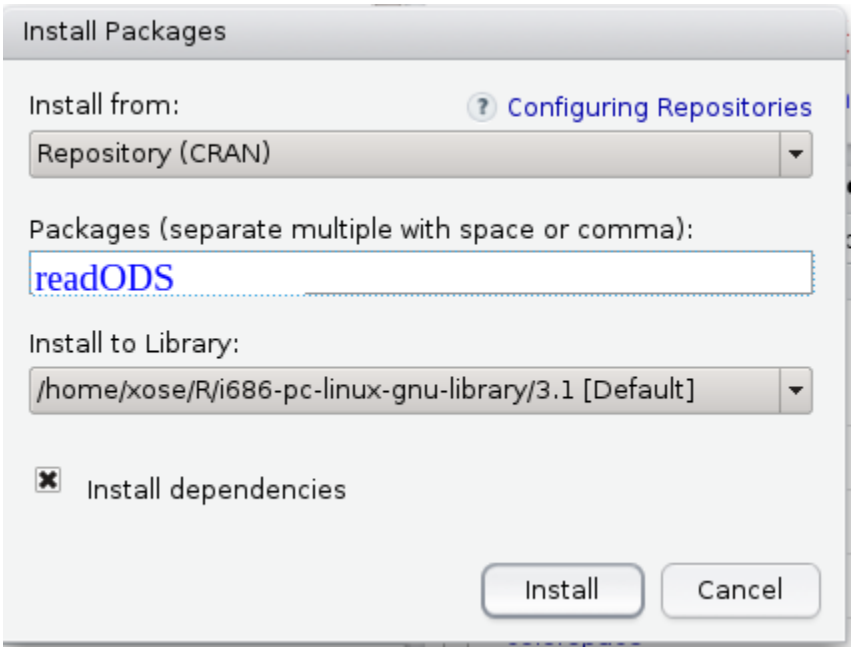
A maioría das operacións con paquetes realizados aquí tamén poden ser realizadas empregando RStudio

Instalar e cargar paquetes en R usando RStudio

En RStudio instalar un paquete novo desde o repositorio é fácil. Faise desde o xestor de paquetes:



Premendo en **install** aparece o seguinte:



En **packages** debedes escribir o nome do paquete que vos interesa, e premer **install**

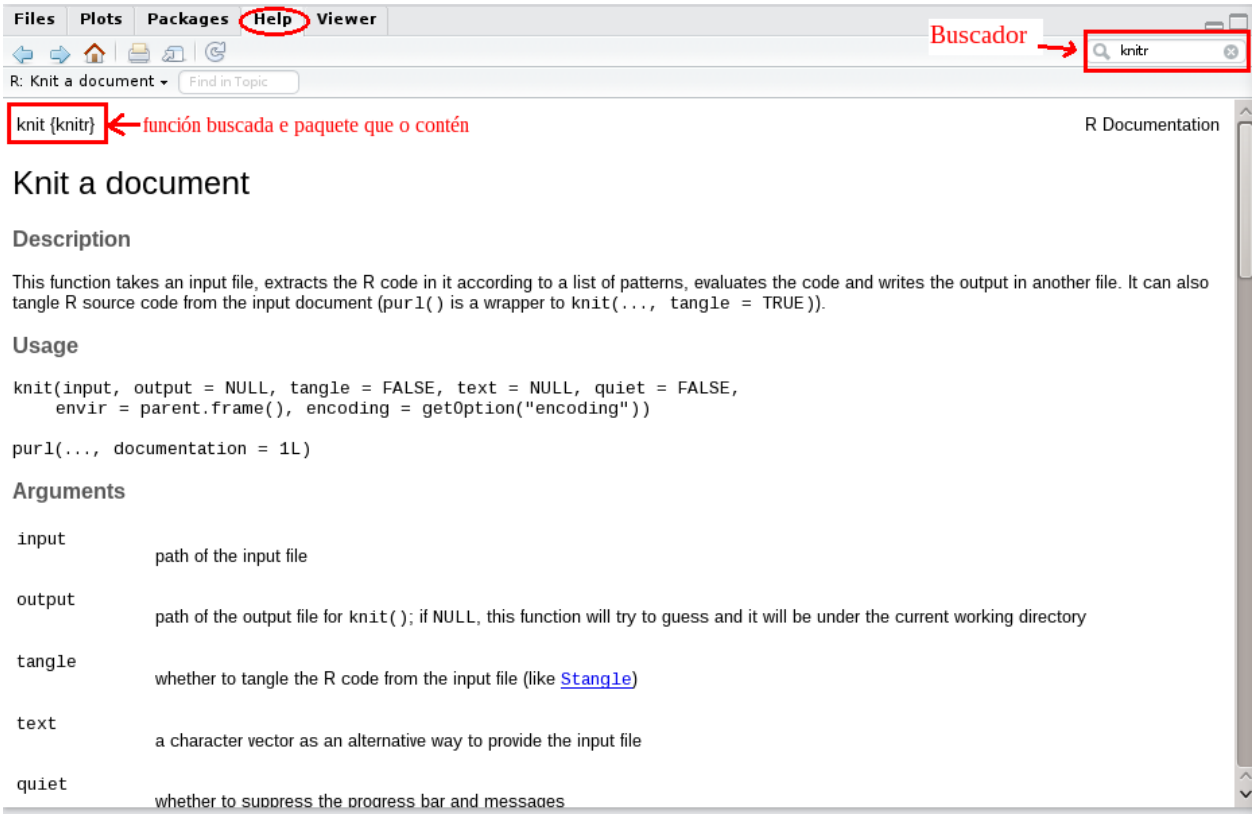
ANEXO 3:

A axuda en R

No exemplo de traballo con R xa se comeza a ver o uso de comandos (funcións),como funcionan? As máis simples poñendo o nome da función e, entre paréntese, os valores aos que se aplica.

Para as máis complicadas é aconsellable consultar a axuda.

Como é a axuda en R? En RStudio pódese acceder a ela a traves do menu:



Na súa interface **permite buscar nomes de funcións**, que aparecerán presentadas coa estructura habitual de axuda de R.

A axuda tamén se consigue usando comandos por exemplo:

```
help.start() #accede a unha página con información diversa sobre o R
help.search('mean'))   #busca información sobre a palabre "mean" dentro da documentación de R. Ollo!:no
help(mean) #busca axuda sobre a "función mean"
?mean #busca axuda sobre a "función mean"
```

Estructura da axuda

As axudas en R traen sempre a mesma estructura:

1. **Description** (Descrición): Describe o que fai o comando

2. **Usage** (Sintaxe – forma de uso): Amosa como debemos usar o comando, os seus argumentos (parametros que leva ou pode levar) e os valores que toma por defecto
3. **Arguments**: Describe cada un dos argumentos que pode usar o comando.
4. **Details** (Detalles): Describe cun pouco mais de detalle o comando e/ou argumentos.
5. **Value** (Valores): Describe o que retorna o comando
6. **See Also** (Ver tamén): Cita alguns comandos que teñan relación co comando da axuda
7. **Examples** (Exemplos): Exemplos co comando. Podemos copialos e pegalos na liña de comandos para ver como se executan.

Nestas axudas é moi interesante o **apartado de exemplos** posto que permite ver casos de aplicación e con frecuencia e suficiente ver como se aplica para entender o uso do comando.