Una introducción a R

00R team

MÉTODOS ESTADÍSTICOS DE INVESTIGACIÓN:
INTRODUCCIÓN A R Y RSTUDIO 2015–16

Introducción

Objetivos de la sesión

Conocer y comprender

1. Conocer
   - la principales ventajas de R
   - el funcionamiento básico de la terminal de R
   - los principales elementos de la sintaxis de R
   - el procedimiento básico de trabajo con R
2. Comprender
   - el fundamento de la sintaxis de R
   - el procedimiento de trabajo
   - los mensajes de error del sistema

¿Qué es R?

Definición

- Permite el almacenamiento, manejo y tratamiento estadístico de los datos
- R se desarrolló sobre una idea de R Becker, J Chambers y A Wilks
- *lingua franca* de la estadística y los aspectos cuantitativos de numerosos campos del conocimiento:
  - biología (ecología, genética, filogenia. . .), farmacología, . . .
  - economía, finanzas, . . .
  - Química, física,
  - optimización, etc.

Aplicaciones

- Diferentes aplicaciones ante distintos problemas en el tratamiento y tipo de datos:
  - Series temporales
  - Análisis multivariante
  - Optimización
  - Aprendizaje automático
  - Investigación reproducible
  - ...

¿Qué es R?

- Vídeo de *Revolution Analytics* (spa, eng):

`http://ares.inf.um.es/00Rteam/videos/whatsR.webm`

La empresa Revolutions Analytics es ahora propiedad de Microsoft.

R un entorno para trabajar en S

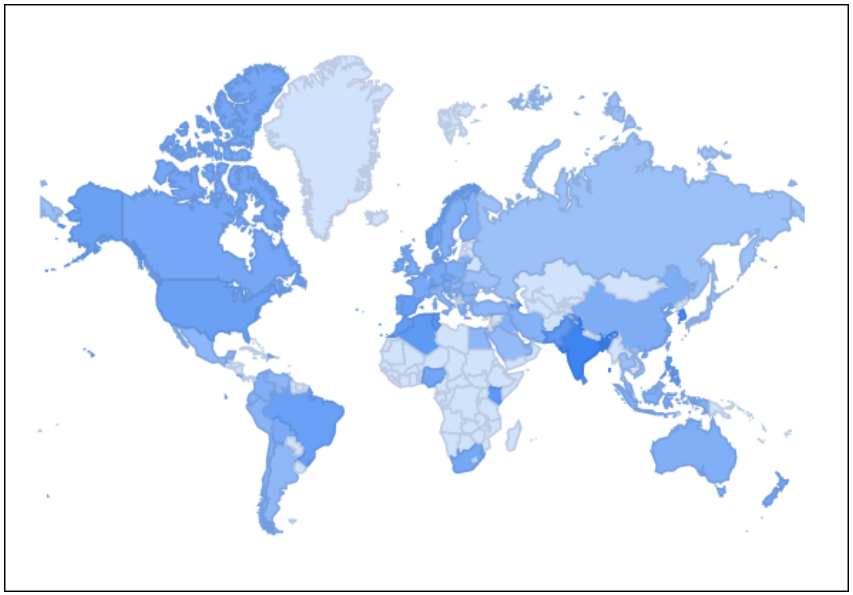
- Un conjunto coherente y extensivo de instrumentos para el análisis y el tratamiento estadístico de datos.
- Un lenguaje para expresar modelos estadísticos y herramientas para manejar modelos lineales y no lineales.
- Utilidades gráficas para el análisis de datos y la visualización.
- Un lenguaje eficiente de programación orientado a objetos, que crece fácilmente merced a la comunidad de usuarios.

¿Qué tiene R que tanto nos gusta?:

- Es libre. licencia GNU, –> utilizar y ¡mejorar!
- Es multiplataforma: Linux, Windows, Mac, iPhone...
- Se puede analizar en R cualquier tipo de datos.
- Es potente. Es muy potente.
- Capacidad gráfica. Difícilmente es superada por ningún otro paquete estadístico.
- Compatible con 'todos':csv, xls, sav, sas...
- Es ampliable, si quieres añadir algo: ¡empaquétalo!
- Hay miles de técnicas estadísticas implementadas, cada día hay más.

Tendencias softwares estadísticos

- Comparaciones y tendencias en *Google trends*:
  1. Sobre paquetes estadísticos
  2. Sobre R y su entorno



Elementos de R

- Lenguaje (con una sintaxis relativamente simple)
- Interfaces (Para distintos tipos de usuarios y problemas)
- Documentación
  - recursos en red
  - revistas
  - Listas de distribución
  - FAQ, *Frecuently asked Questions*
- Datos de ejemplo
- Librerías (conjuntos aplicaciones y desarrollos aportados por la comunidad de usuarios)
- Comunidad

Comunidad

- Comunicación: Página principal del proyecto R
- Grupos de usuarios (en inglés, castellano, . . .)
- Reuniones (nacionales e internacionales): jornadas, congresos, . . .
- Proyectos específicos

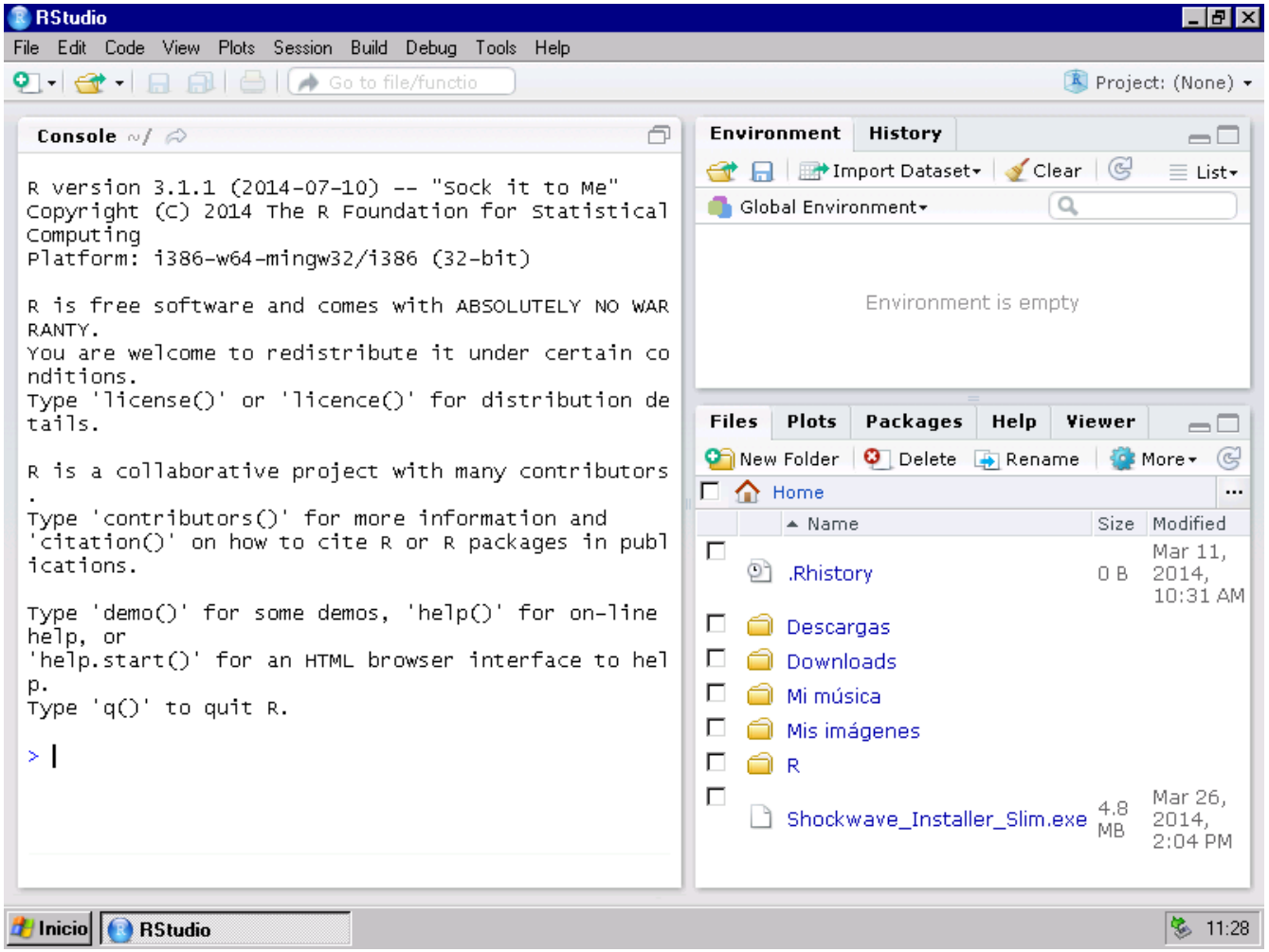
Importancia de la comunidad

- R aumenta su capacidad con la colaboración de los usuarios
  - 1998 unas 200 librerías
  - 2011, octubre, más de 3300
  - Hoy, ¿cuántas?

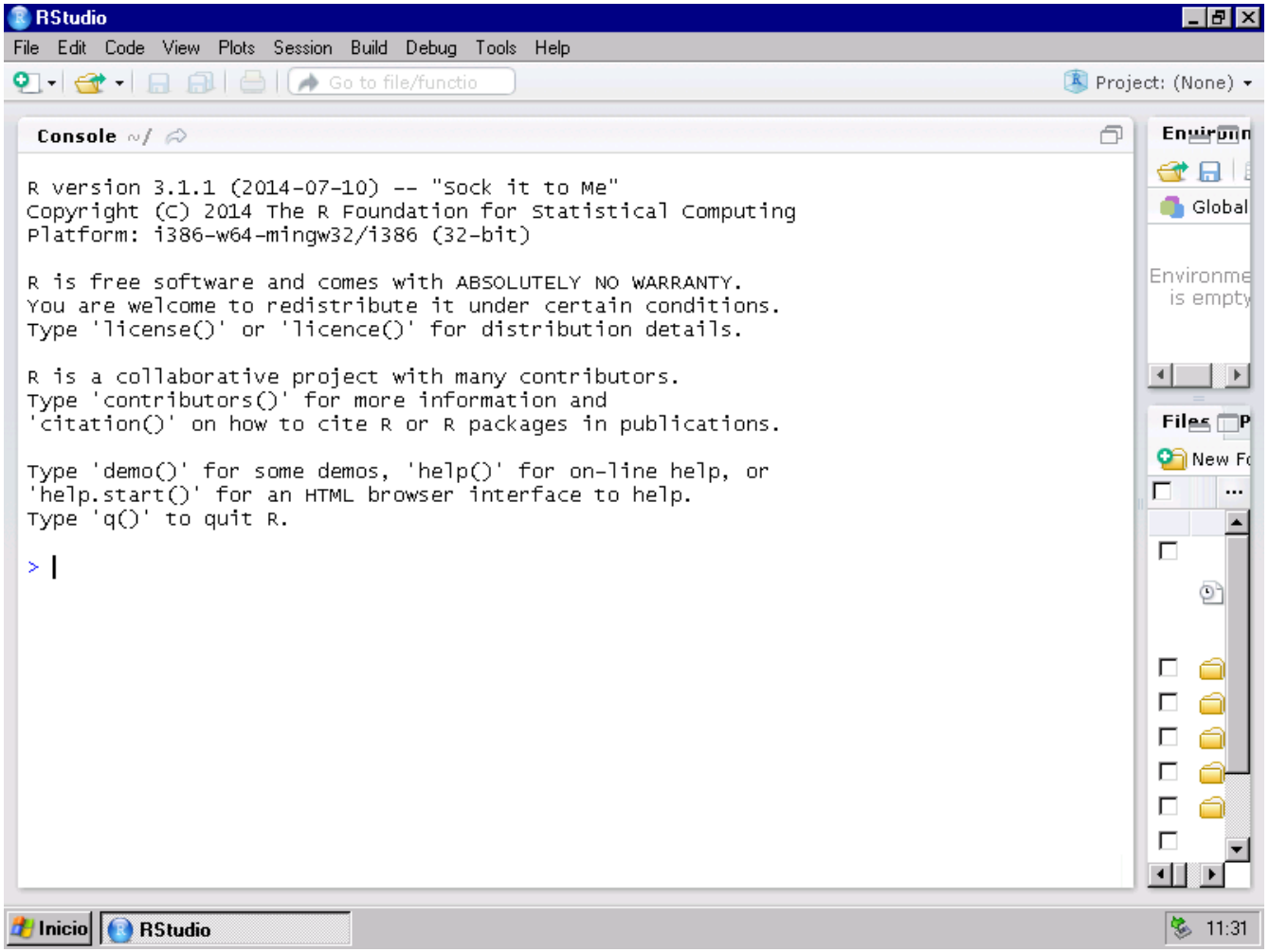
Unas fotos de familia

- **Interfaces**
  `http://www.statmethods.net/interface/guis.html`
- **Instalación**
  `http://cran.r-project.org`
- **Rstudio**
  `http://www.rstudio.com/`

rstudio


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/functio
Project: (None)
Console ~/
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment History
Import Dataset
Clear
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder
Delete
Rename
More
Home
Name Size Modified
.Rhistory 0 B Mar 11, 2014, 10:31 AM
Descargas
Downloads
Mi música
Mis imágenes
R
Shockwave_Installer_Slim.exe 4.8 MB Mar 26, 2014, 2:04 PM
Inicio
RStudio
11:28

rstudio: la terminal



Sobre la notación y la tipografía

Chateando con un autómata

- El autómata carece de inteligencia
- R hace lo que se le pide, no lo que se quiere
- En una conversación deben respetarse las reglas de comunicación
- Las reglas tipográficas ayudan a simplificar

De la escritura

- El manejo del teclado es muy importante
- Atajos de teclado, *hotkeys* y *shortcuts*
- Sensibilidad a mayúsculas (*case sensitive*): no es lo mismo 'A' que 'a'
- El uso del tabulador para autocompletado

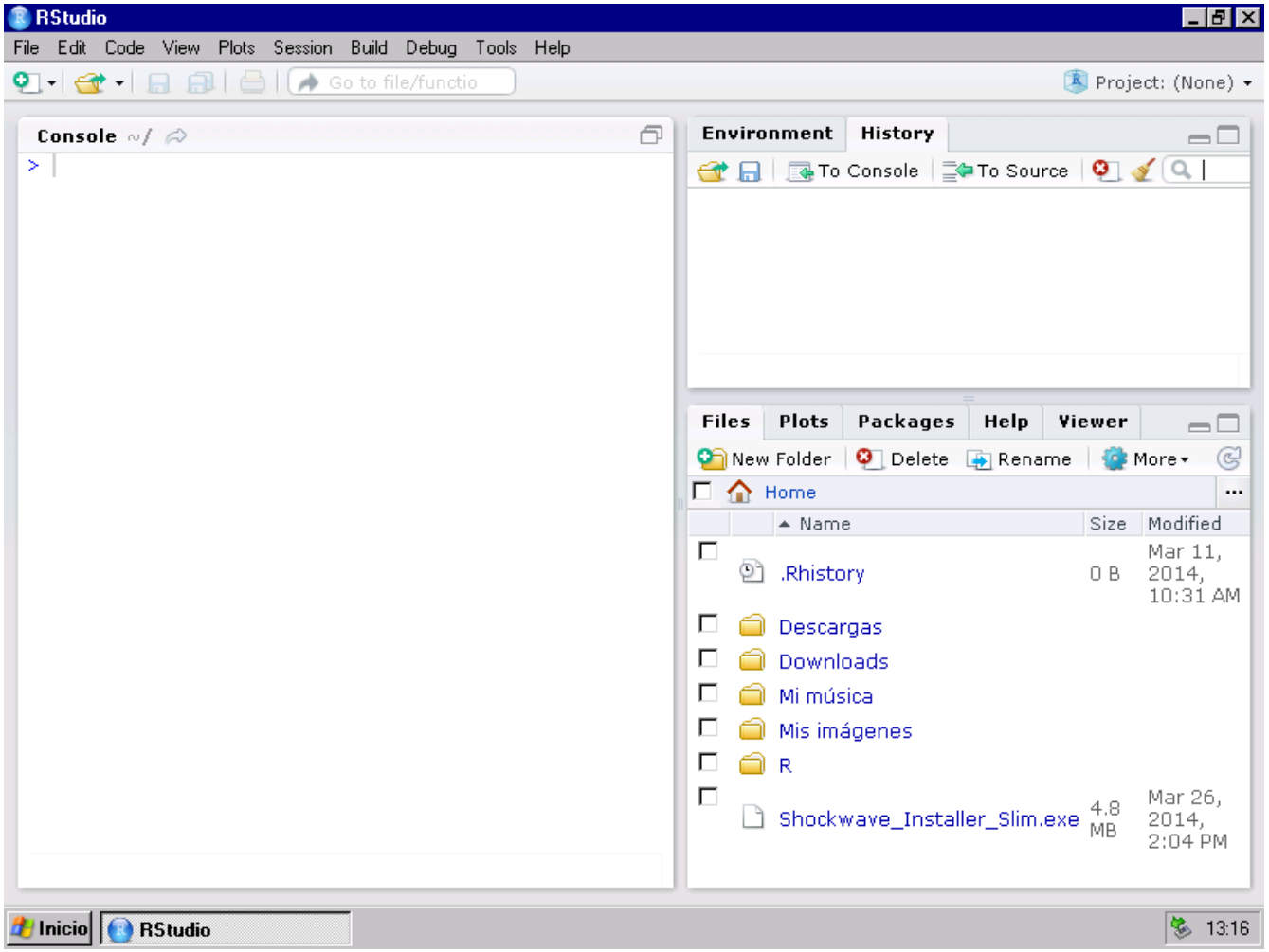
De la pantalla

- Intercomunicación: mensajes de respuesta
- Errores: *Warning*
- Errores: *Fatal error*
- Malditos errores: *Syntax error*

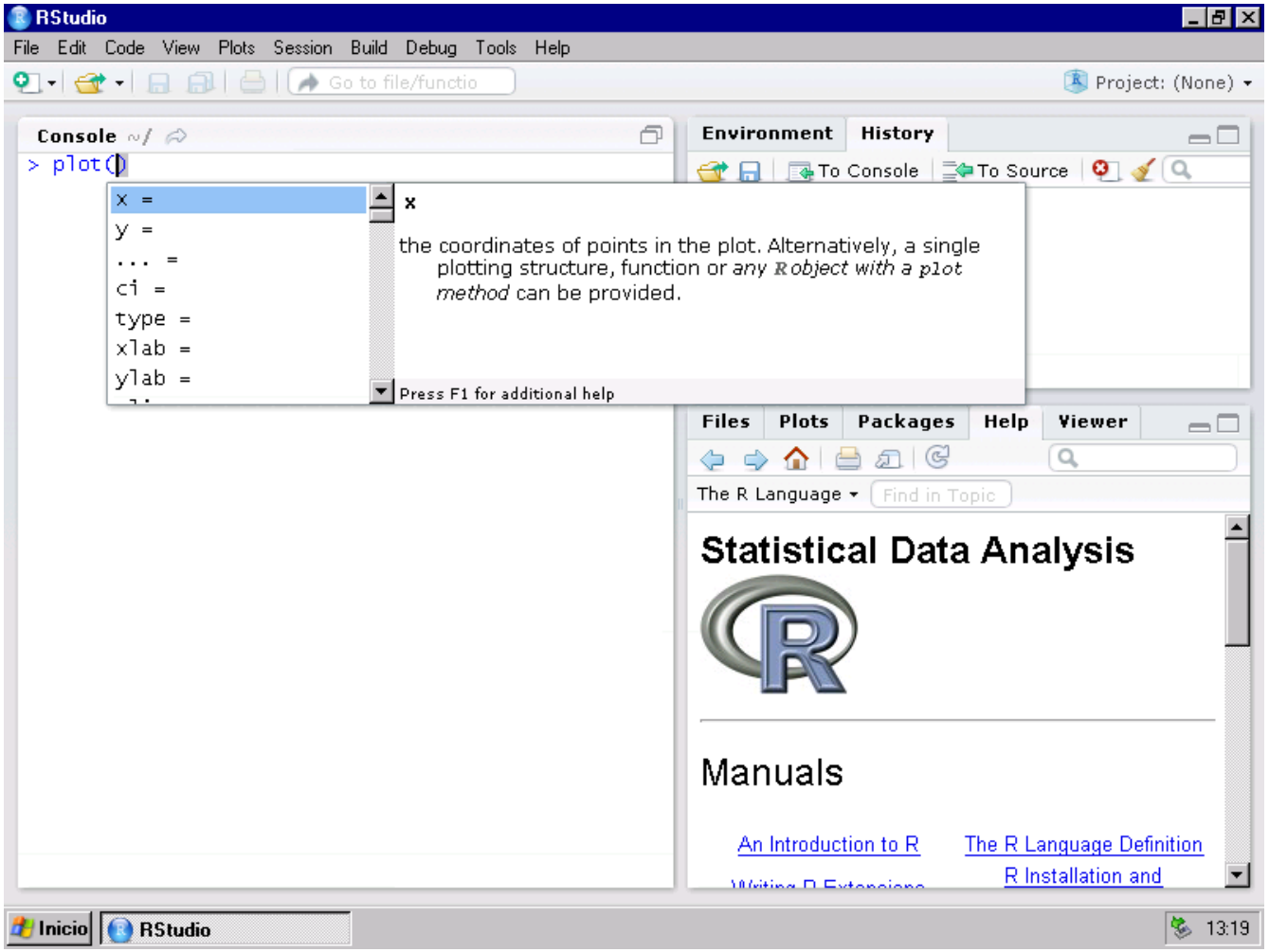
La terminal de R

- Bienvenida
- El *prompt*
  - >
  - +

rstudio

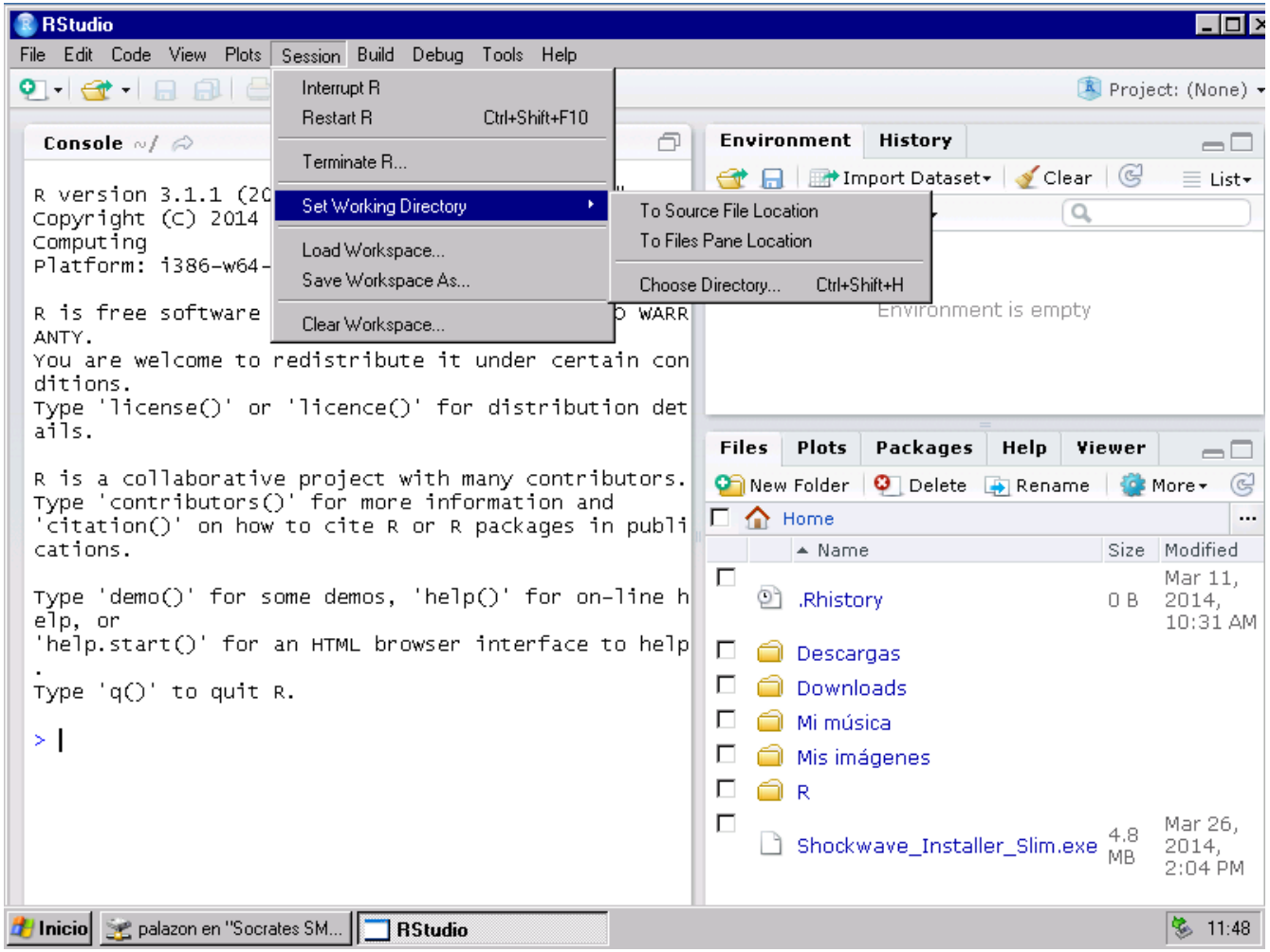

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/functio
Project: (None)
Console ~/
Environment History
To Console
To Source
Files Plots Packages Help Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.Rhistory
0 B
Mar 11, 2014, 10:31 AM
Descargas
Downloads
Mi música
Mis imágenes
R
Shockwave_Installer_Slim.exe
4.8 MB
Mar 26, 2014, 2:04 PM
Inicio
RStudio
13:16

rstudio: Usando el tabulador

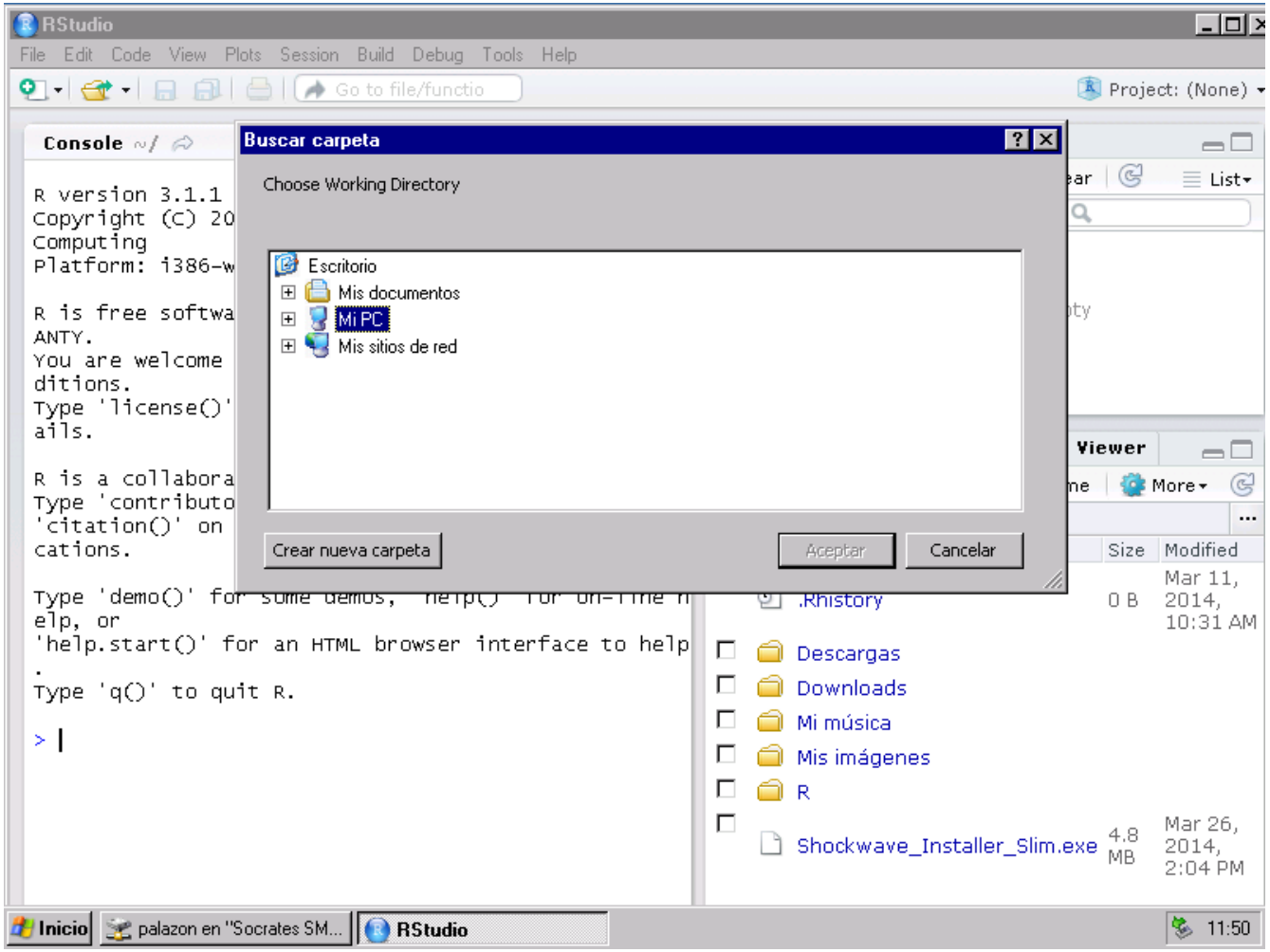


Elección del directorio de trabajo

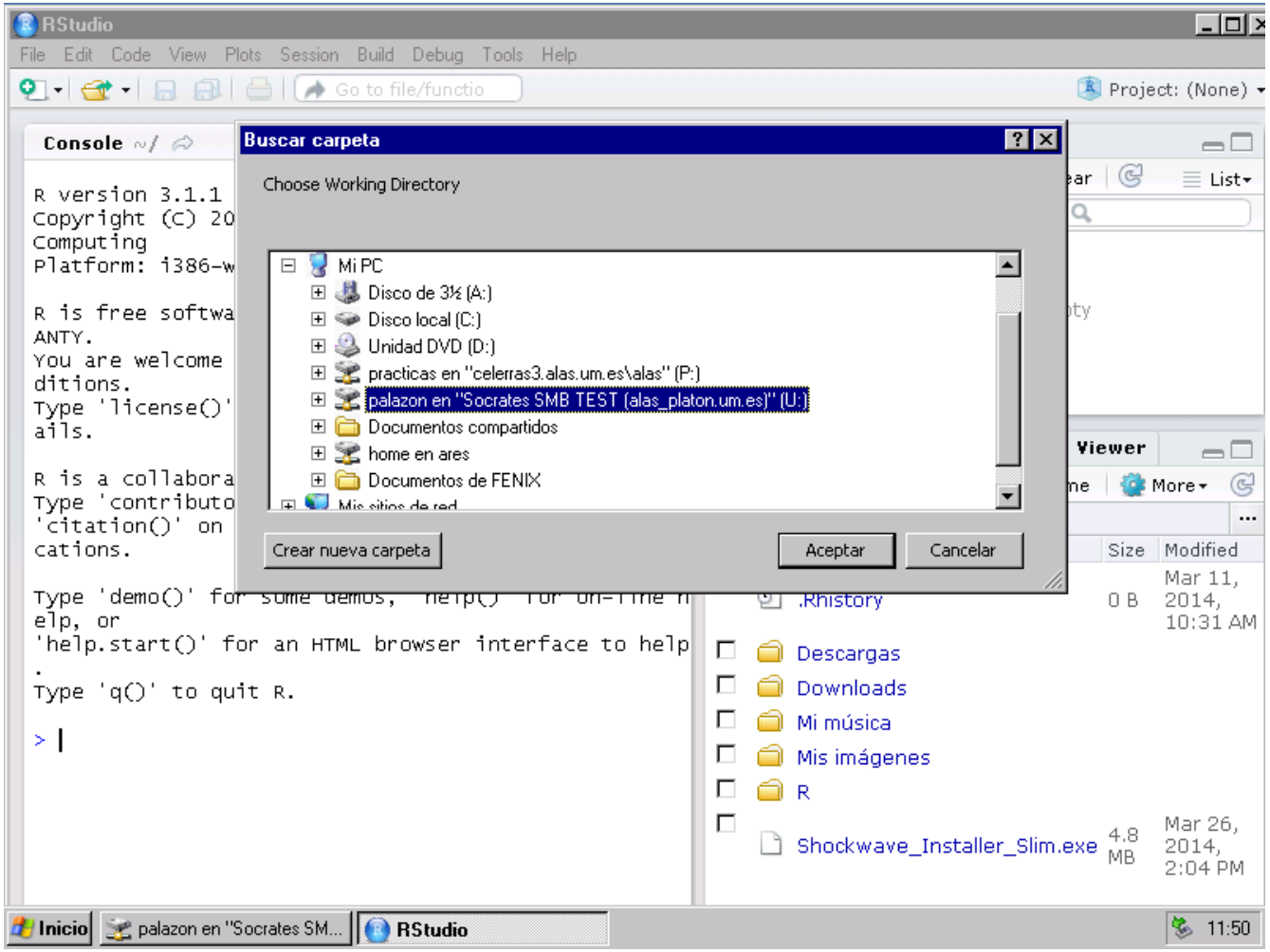
rstudio: Entrada Session



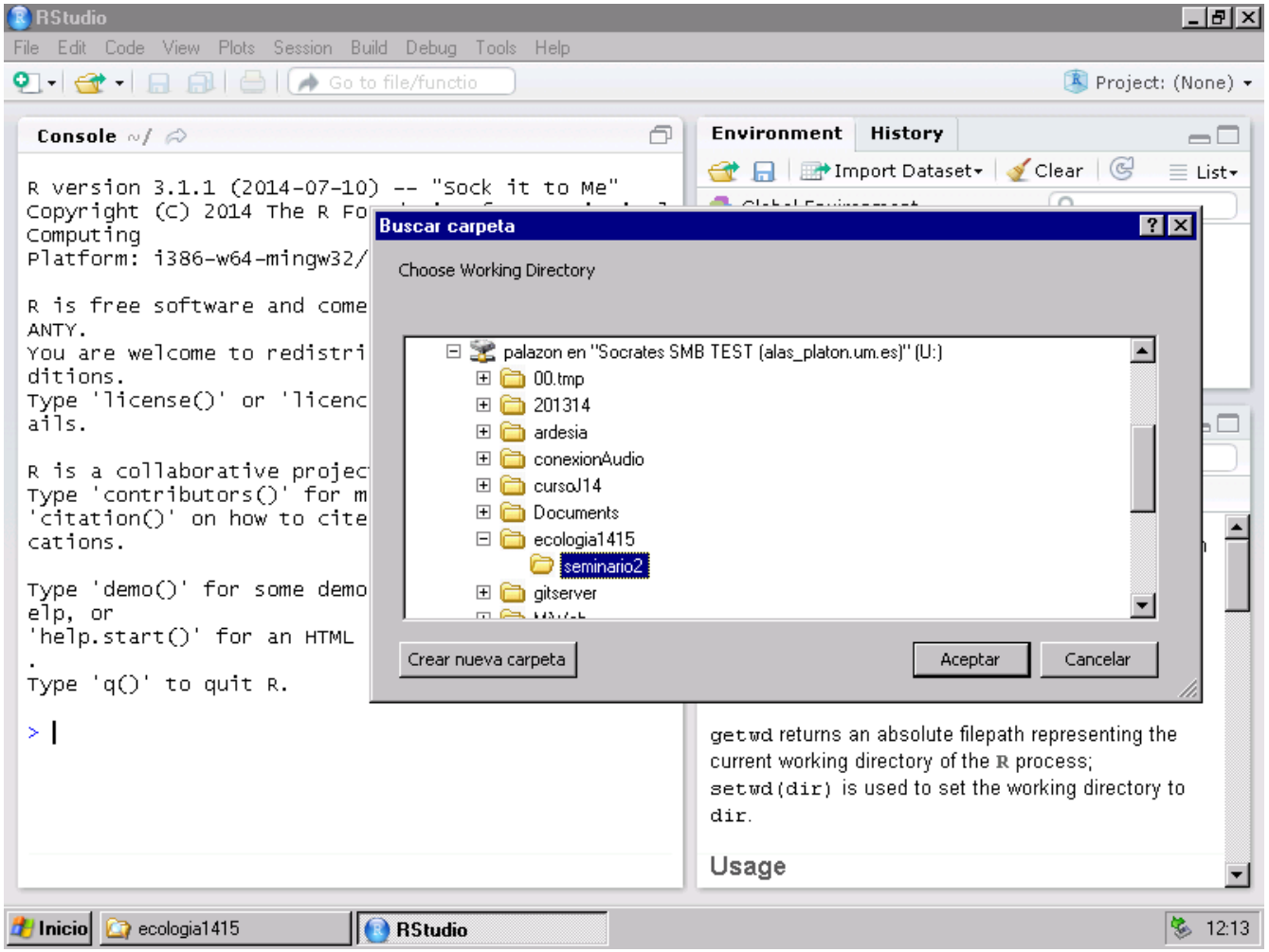
rstudio: Localizando el directorio de trabajo



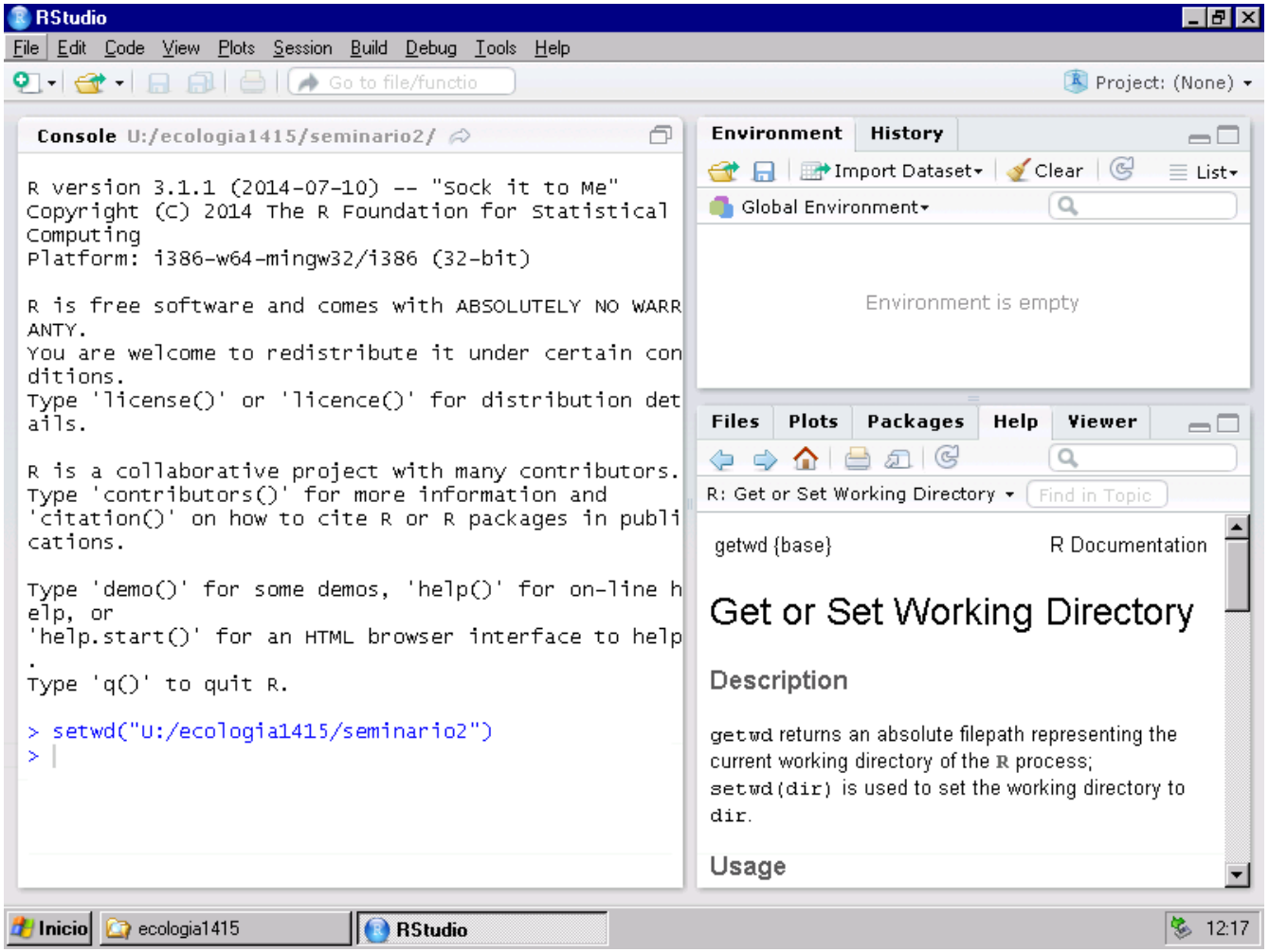
rstudio: Seleccionando la unidad sócrates



rstudio: directorio para el seminario



rstudio: ¡Una expresión!



R: el lenguaje

Sobre la notación

Reglas básicas de sintaxis R

- Reglas sintácticas
  1. R evalúa **expresiones**
  2. El lenguaje es sensible a mayúsculas
  3. Pueden utilizarse espacios entre elementos de sintaxis a discreción: `sin(x+b)` es igual que `sin  ( x +  b)`
  4. Cada expresión se escribe en al menos una línea
  5. Dos o más expresiones puede utilizar una línea separándolas por el signo `';'`
- En R, donde entra un valor puede entrar una expresión
- Regla de reuso
- `ESC` una tecla para huir, abortar, cortar,. . .

Notación matemática y sintaxis de R

Matemáticas	Expresión en R
$x = 3$	`x <- 3`
$sin\,\alpha$	`sin( alpha )`
$log_{10}(x)$	`log( x, 10 )`
v_i	`v[ i ]`
$\sum_{i=1}^{n} v_i$	`sum( v )`

Elementos de R

Valores

- Enteros: `3`
- Reales: `1.8e+12` ($1.8\,10^{12}$)
- Complejos: `0+1i` ($\sqrt{-1}$)
- Carácter: `"rojo"`
- Perdidos: `NA`
- No números: `NaN` (`log(0)`)
- Indeterminaciones ($-\infty$, ∞): `-Inf, Inf` ($\frac{1}{0}$)

Operadores aritméticos

- Importancia de la jerarquía de operadores
- Operadores aritméticos
  - escalares
  - matriciales
- Operadores lógicos

Operadores aritméticos

^	potencia
* /	producto, cociente
+ -	suma, resta
%/%	cociente entero
%%	módulo
:	generar una serie
%*%	producto matricial
()	paréntesis

Ejemplos

```
3 ^ 2
```

```
## [1] 9
```

```
3 ^ 1 + 1
```

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

```
3 ^ ( 1 + 1 )
```

```
## [1] 9
```

Ejemplos

```
10 / 2 * 5
```

```
## [1] 25
```

```
10 / 2 / 5
```

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

```
21 %% 5
```

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

Ejemplos

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10 * 2
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
2^(0:8)
```

```
## [1]   1   2   4   8  16  32  64 128 256
```

Operadores lógicos

!	no
== !=	igual, distinto
> >=	mayor, mayor o igual
< <=	menor, menor o igual
| ||	o
& &&	y
#	comentario

Ejemplos

```
3 >=2
```

```
## [1] TRUE
```

```
0 != 0.0000000000000001
```

```
## [1] TRUE
```

```
5*2 > 9 & 3/2 == 1.5
```

```
## [1] TRUE
```

Asignaciones

- `Variable <- expresión`
- Variable es un nombre que se utiliza como representación del resultado de una expresión

<-	asignar a la izquierda
->	asignar a la derecha
=	asignar a la izquierda

Ejemplos

```
a <- 3
a
```

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

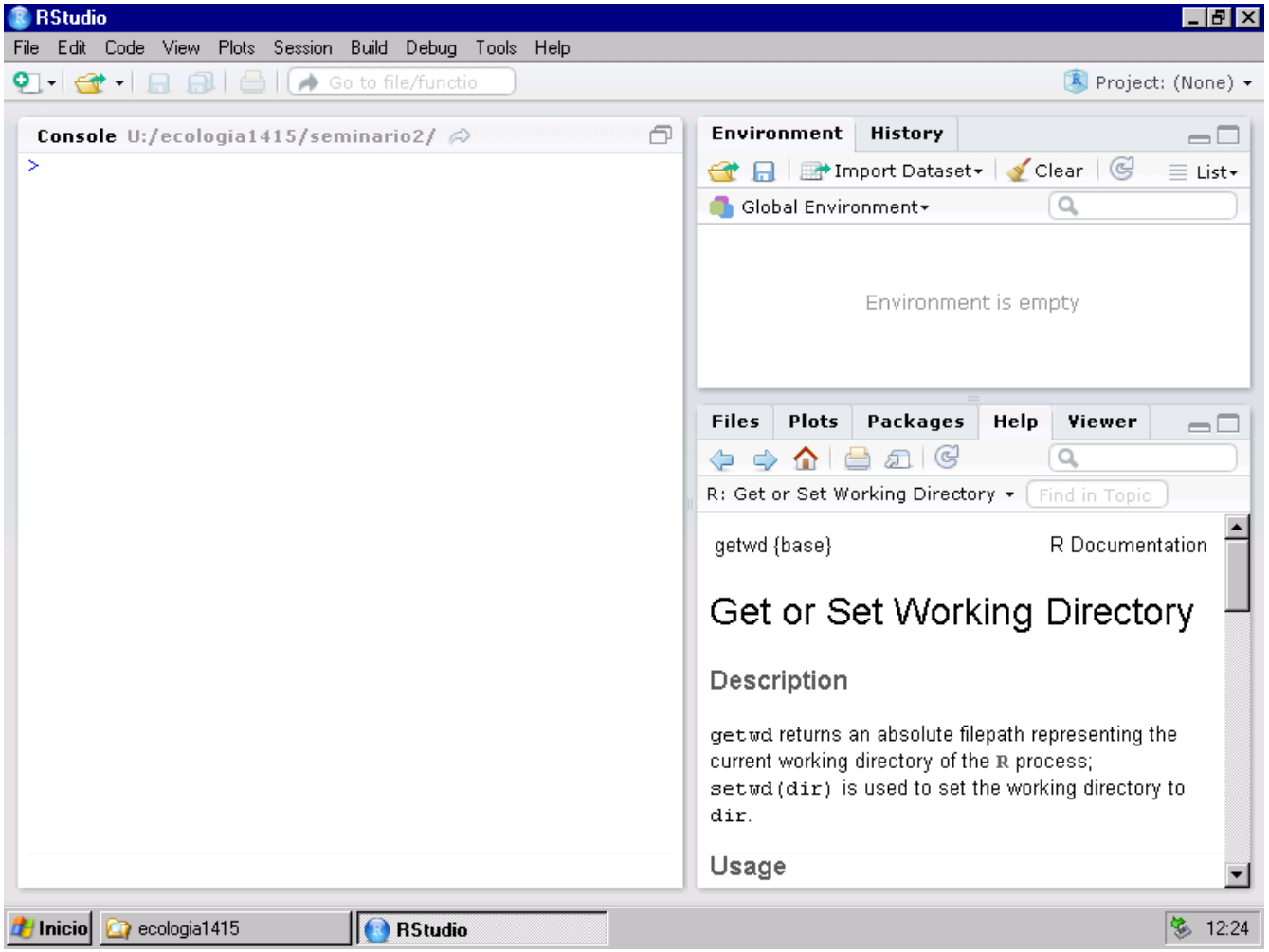
```
a <- a + 1
a
```

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

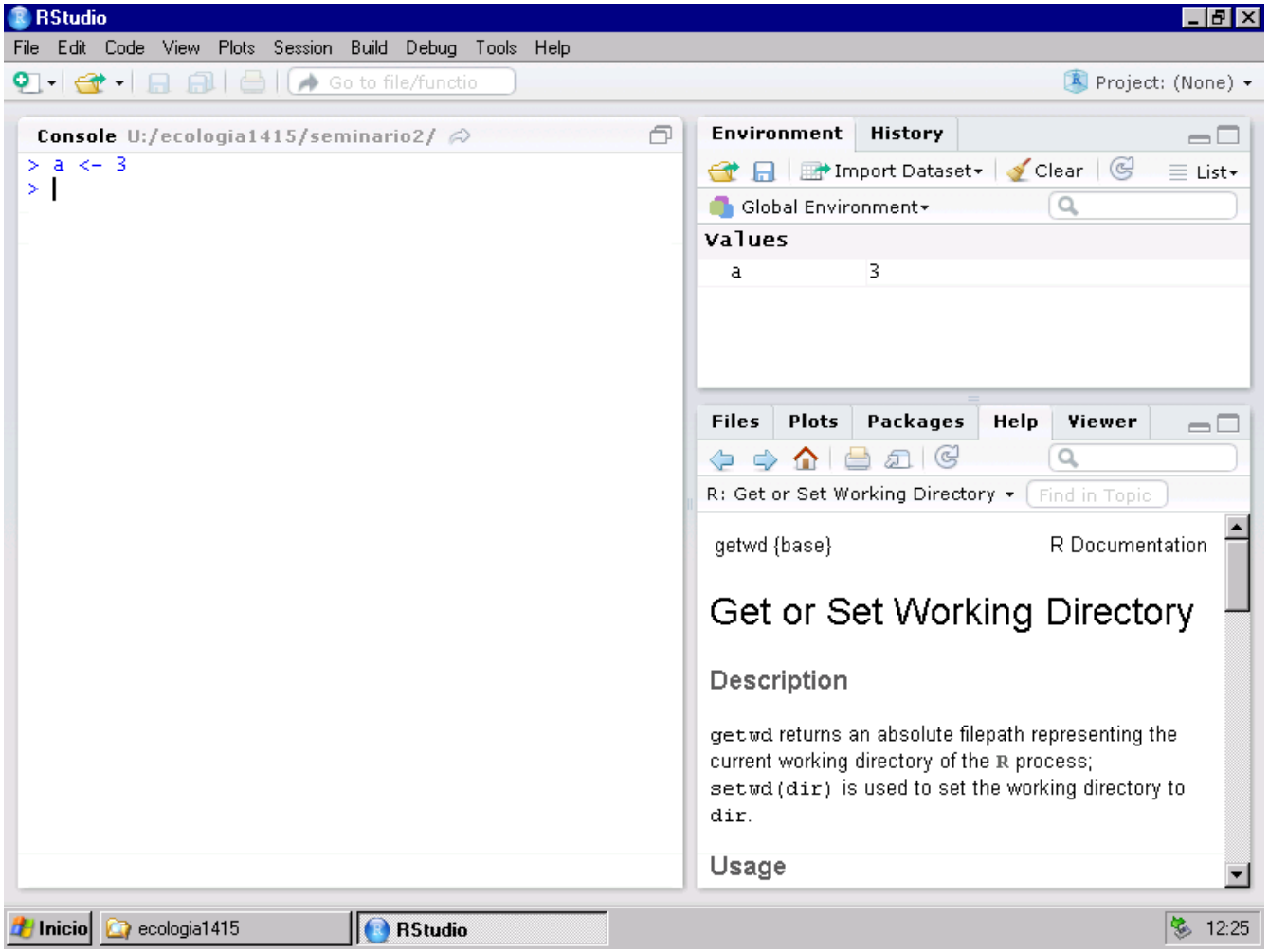
```
(a <- a + 1)
```

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

rstudio: ¿qué objetos tengo y cuál es su valor?



rstudio: ¿qué objetos tengo y cuál es su valor?



Ejemplos

```
r <- 1
area <- pi * r ^ 2
longitud <- 2 * pi * r
area
```

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

```
longitud
```

```
## [1] 6.283185
```

Ejemplos

```
r <- 1:10
area <- pi * r ^ 2
2 * pi * r -> longitud
area #;longitud
```

```
##  [1]   3.141593  12.566371  28.274334  50.265482
##  [5]  78.539816 113.097336 153.938040 201.061930
##  [9] 254.469005 314.159265
```

Funciones

- Una función es un procedimiento para realizar una determinada tarea o cálculo
- función se asocia a un nombre, que sigue las mismas reglas que las variables
- **nombre_de_la_función** (*argumento 1*, *argumento 2*, . . .)
- Los argumentos son propios de cada función
- En algunos casos los argumentos tienen valores por defecto

Ejemplo

```
log( 2 )
```

```
## [1] 0.6931472
```

```
log( 2, 10 )
```

```
## [1] 0.30103
```

```
log( exp( 1 ) )
```

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

Ejemplo

```
log( x = 2 , base = 10 )
```

```
## [1] 0.30103
```

```
log( base = 10, x = 2 )
```

```
## [1] 0.30103
```

Funciones

`c()`	Concatenar los elementos que se indican, separados por comas
`seq()`	Generar una secuencia numérica
`rep()`	Generar un conjunto de valores repetidos
`sort()`	Ordena un vector

Funciones

`round()`	Redondeo de valores numéricos
`sqrt()`	Raíz cuadrada
`abs()`	Valor absoluto
`sin()`	Función trigonométricas seno
`cos()`	Función trigonométricas coseno
`log()`	Logaritmo natural
`exp()`	exponencial (e^x)

Funciones

`sum()`	Suma los elementos de un vector
`cumsum()`	Vector de sumas acumuladas
`max()`	Máximo de un vector
`min()`	Mínimo de un vector
`t()`	Transponer una matriz
`names()`	Nombres de filas o columnas
`nrow()`	Número de filas
`ncol()`	Número de columnas
`rownames()`	Nombre de las filas
`colnames()`	Nombres de las columnas

Funciones

`str()`	Proporciona información sobre la estructura de un objeto
`ls()`	Relación de objetos disponibles
`rm()`	Elimina uno o varios objetos
`read.table()`	Carga los datos de un fichero
`source()`	Carga el código de R escrito en un fichero

R: los objetos

Vectores

- Los vectores son un conjunto ordenado de valores
- Para calcular con todo el vector se emplea el nombre del objeto
- Para utilizar un subconjunto valores se emplea subíndices
- Los subíndices se incluyen entre corchetes (`x[ 3 ]`)
- Los subíndices están en el rango: 1 — *número de elementos del vector*
- Los subíndices pueden ser expresiones

Ejemplo

```
x <- c( 8, 5, 2, 4, 1, 6, 3 )
length( x )
```

```
## [1] 7
```

```
x
```

```
## [1] 8 5 2 4 1 6 3
```

```
x[]
```

```
## [1] 8 5 2 4 1 6 3
```

Ejemplo

```
x[ 1 ]
```

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

```
x[ 2:4 ]
```

```
## [1] 5 2 4
```

```
x[ c( 3, 5 ) ]
```

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

```
x[ -1 ]
```

```
## [1] 5 2 4 1 6 3
```

Ejercicio

- Crea un vector de números enteros (mínimo 5 elementos)
- Comprueba el tipo de tu vector con la función `str()`
- Aplica algunas funciones a tu vector para calcular: su media, la suma de sus componentes...
- Haz que tu vector (que ya está creado) sea de tipo `cadena de caracteres`. Guárdalo en una nueva variable
- Comprueba su tipo con la función `str()`
- Intenta aplicar las funciones anteriores a ese vector. ¿Qué sucede?

Matrices

- Una matriz es un conjunto ordenado de vectores
- Los elementos de la matriz están ordenados por filas y columnas
- Todos los vectores son del mismo tipo: enteros, caracteres, ...
- Los elementos de una matriz se identifican por dos subíndices
- El uso de los subíndices sigue las mismas reglas que en el caso de los vectores
- Se puede crear uniendo vectores o mediante la función `matrix()`

Ejemplo

```
m <- matrix( 1:12, 4, 3 )
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```
m[ 1, ]
```

```
## [1] 1 5 9
```

Data frames

- Son semejantes a las matrices
- Se organizan por filas y columnas
- Las columnas no tienen por que ser homogéneas
- Las columnas tienen nombre
- Habitualmente los *data frames* se obtienen de la lectura de un fichero de datos

Ejemplo PIB

	Valor		Estructura porcentual	
	2008	2009	2008	2009
Andalucía				
A. Agricultura, ganadería, silvicultura y pesca	6.467.357	6.025.496	4,3	4,2
F. Construcción	21.477.597	19.223.889	14,4	13,5
Hostelería	10.076.699	10.005.749	6,8	7,0
Aragón				
A. Agricultura, ganadería, silvicultura y pesca	1.197.806	1.188.230	3,5	3,6
F. Construcción	4.678.884	4.447.108	13,5	13,4
Hostelería	1.905.278	1.919.271	5,5	5,8
Galicia				
A. Agricultura, ganadería, silvicultura y pesca	2.278.151	2.244.530	3,9	4,0
F. Construcción	7.901.498	7.507.649	13,7	13,4
Hostelería	2.710.660	2.927.641	4,7	5,2
Madrid, Comunidad de				
A. Agricultura, ganadería, silvicultura y pesca	224.709	195.667	0,1	0,1
F. Construcción	20.320.981	19.171.700	10,5	10,1
Hostelería	10.418.601	10.861.179	5,4	5,7
Murcia, Región de				
A. Agricultura, ganadería, silvicultura y pesca	1.377.749	1.233.257	4,7	4,5
F. Construcción	4.110.927	3.526.090	14,1	12,8
Hostelería	1.505.280	1.489.960	5,2	5,4

Leer datos de ejemplo PIB

```
df <- read.table(
  "http://ares.inf.um.es/OORteam/datos/pibCcAaEj.dat",
  sep=";")
head( df)
```

```
##    ciudad actividad anho     valor
## 1 Andaluc     Agric 2008  6467.357
## 2 Andaluc     Const 2008 21477.597
## 3 Andaluc      Host 2008 10076.699
## 4    Arag     Agric 2008  1197.806
## 5    Arag     Const 2008  4678.884
## 6    Arag      Host 2008  1905.278
```

Acceder a las variables de un data frame

```
options( width = 100 )
df[ , 4 ]
```

```
##  [1]  6467.357 21477.597 10076.699  1197.806  4678.884  1905.278  2278.151  7901.498  2710.660
## [10]   224.709 20320.981 10418.601  1377.749  4110.927  1505.280  6025.496 19223.889  1005.749
## [19]  1188.230  4447.108  1919.271  2244.530  7507.649  2927.641   195.667 19171.700 10861.179
## [28]  1233.257  3526.090  1489.960
```

```
df$valor
```

```
##  [1]  6467.357 21477.597 10076.699  1197.806  4678.884  1905.278  2278.151  7901.498  2710.660
## [10]   224.709 20320.981 10418.601  1377.749  4110.927  1505.280  6025.496 19223.889  1005.749
## [19]  1188.230  4447.108  1919.271  2244.530  7507.649  2927.641   195.667 19171.700 10861.179
## [28]  1233.257  3526.090  1489.960
```

```
df [, "valor" ]
```

```
##  [1]  6467.357 21477.597 10076.699  1197.806  4678.884  1905.278  2278.151  7901.498  2710.660
## [10]   224.709 20320.981 10418.601  1377.749  4110.927  1505.280  6025.496 19223.889  1005.749
## [19]  1188.230  4447.108  1919.271  2244.530  7507.649  2927.641   195.667 19171.700 10861.179
## [28]  1233.257  3526.090  1489.960
```

```
df[[ 4 ]]
```

```
##  [1]  6467.357 21477.597 10076.699  1197.806  4678.884  1905.278  2278.151  7901.498  2710.660
## [10]   224.709 20320.981 10418.601  1377.749  4110.927  1505.280  6025.496 19223.889  1005.749
## [19]  1188.230  4447.108  1919.271  2244.530  7507.649  2927.641   195.667 19171.700 10861.179
```

Ejercicio

- Selecciona el valor del PIB para Andalucía
- Selecciona el valor del PIB en Andalucía correspondiente a la agricultura

Ejemplo

```
f <- "http://ares.inf.um.es/OORteam/datos/biom2003.dat"
x <- read.table( f )
hist( x$Altura ) -> xHist
```

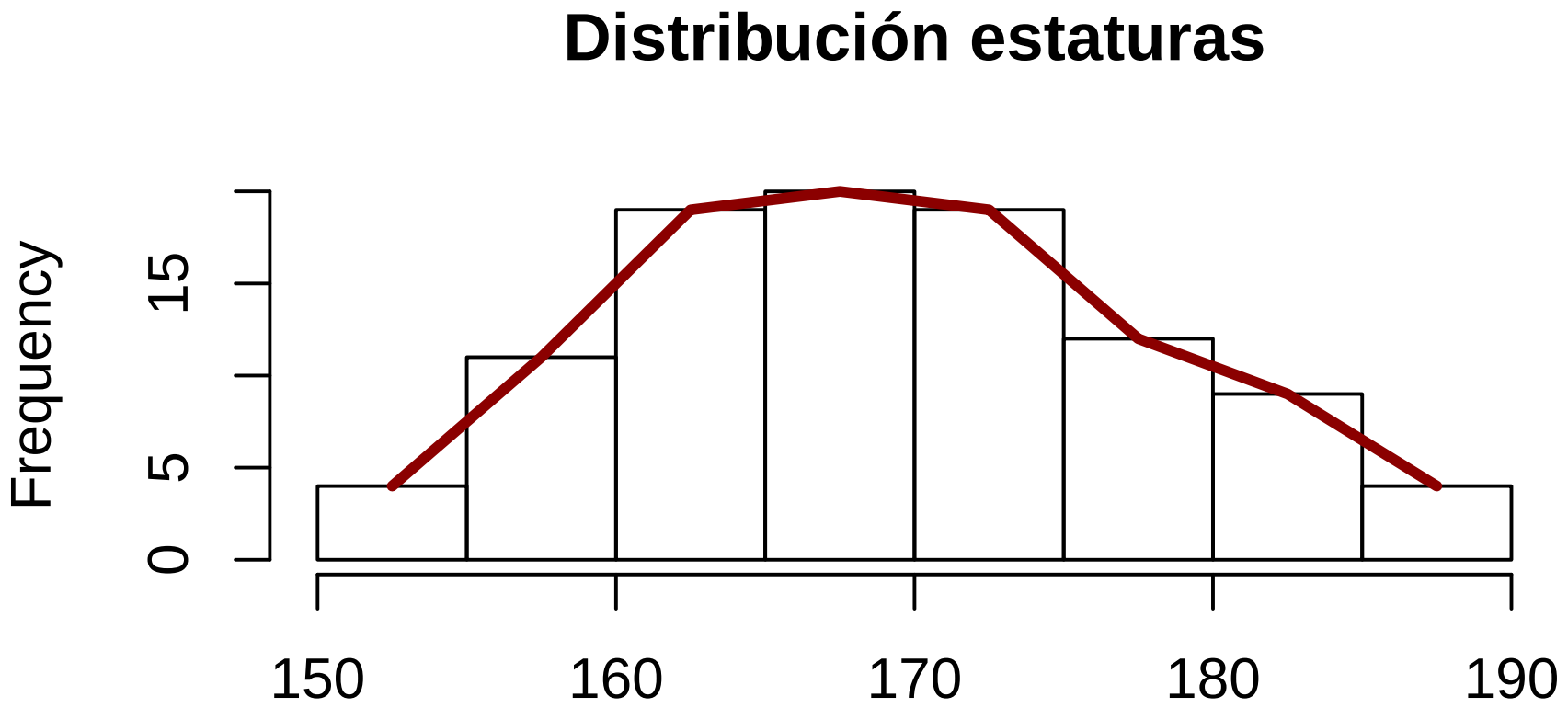
Ejemplo

```
xHist
```

```
## $breaks
## [1] 150 155 160 165 170 175 180 185 190
##
## $counts
## [1]  4 11 19 20 19 12  9  4
##
## $density
## [1] 0.008163265 0.022448980 0.038775510 0.040816327 0.03
##
## $mids
## [1] 152.5 157.5 162.5 167.5 172.5 177.5 182.5 187.5
##
## $xname
```

Ejemplo

```
plot( xHist, main = "Distribución estaturas" )
lines( xHist$mids, xHist$count,
       type="l", col="darkred", lwd = 3 )
```



Ejercicio

Realiza una regresión lineal simple para los valores del `peso` y de la `altura` del anterior conjunto de datos siguiendo los pasos, necesarios, presentados en el vídeo **Primeros minutos con R**:

`http://ares.inf.um.es/OORteam/videos/primerosMinutosR.mp4`

Listas

- Son objetos que pueden contener conjuntos heterogéneos de objetos
  - valores
  - vectores
  - matrices
  - *data frames*
  - listas
- Se suelen encontrar como resultado de funciones

Trabajando con R

El desarrollo de los procedimientos

Preparación del área de trabajo

- Al iniciar la sesión de trabajo área de trabajo está vacía
- Primero deben cargarse las funciones necesarias
  - Mediante la función `source()`
  - Recurriendo a una librería
  - Recurriendo a un documento de *análisis reproducible*

Carga de datos

- Se cargan los datos a procesar asignándolos a las variables correspondientes.
- Se realizan los distintos cálculos y se copia el código utilizado en el block de notas o el editor favorito.
  - Se utiliza la función `savehistory( "miSesion.R" )`, desde la consola.
  - En `rstudio` se utiliza el icono del disquete en la pestaña de *History* para guardar.

Finalizar la sesión de trabajo

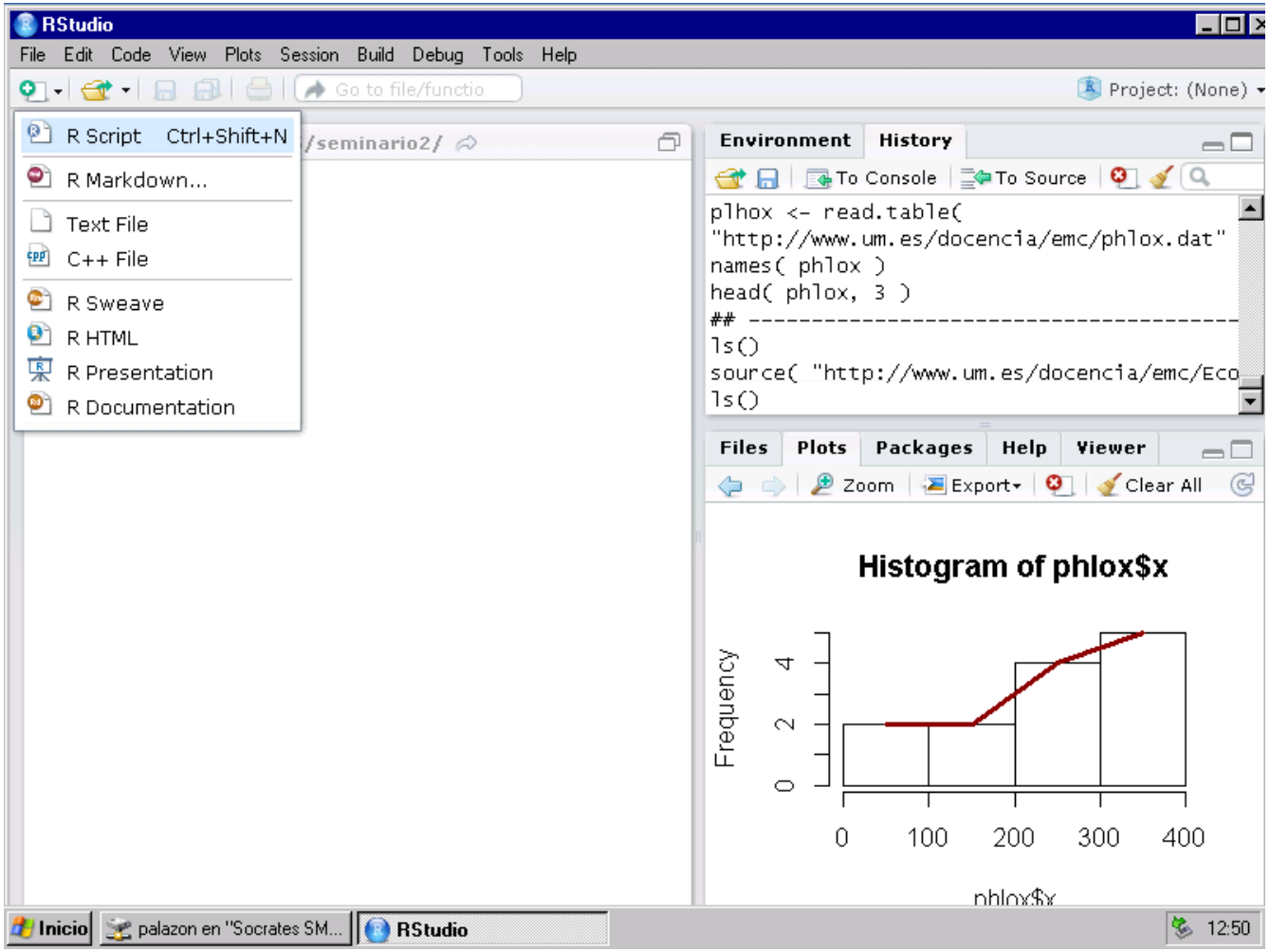
- Se cierra la sesión y se guarda la sesión y el fichero con el procedimiento, preferiblemente con la extensión `.R`

El histórico de la sesión

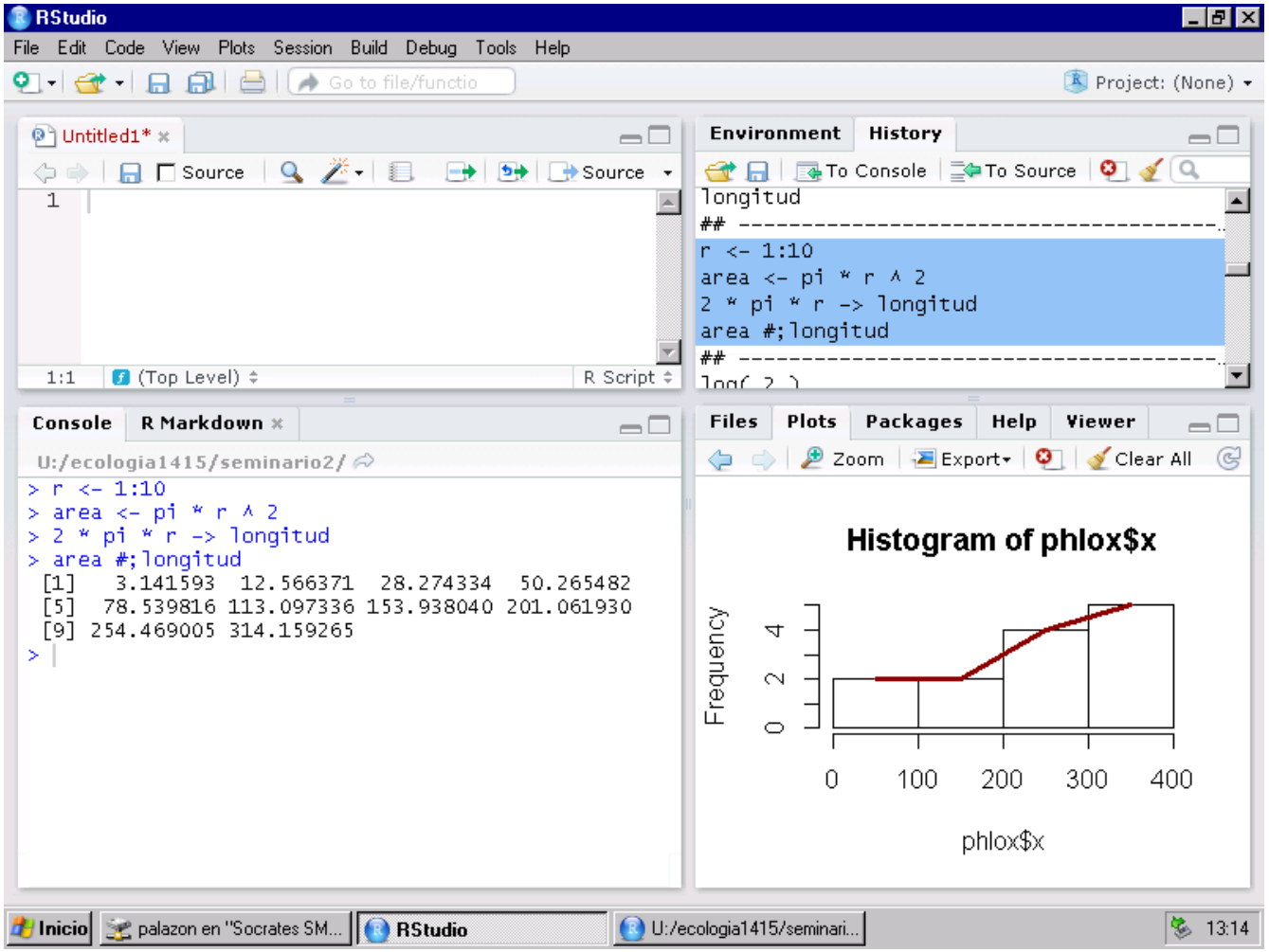
El poder del editor

- Editores de texto plano: bloc de notas, vi, vim, emacs, gedit, atom, ...
- El editor integrado de Rstudio
- Los ficheros y el directorio de trabajo

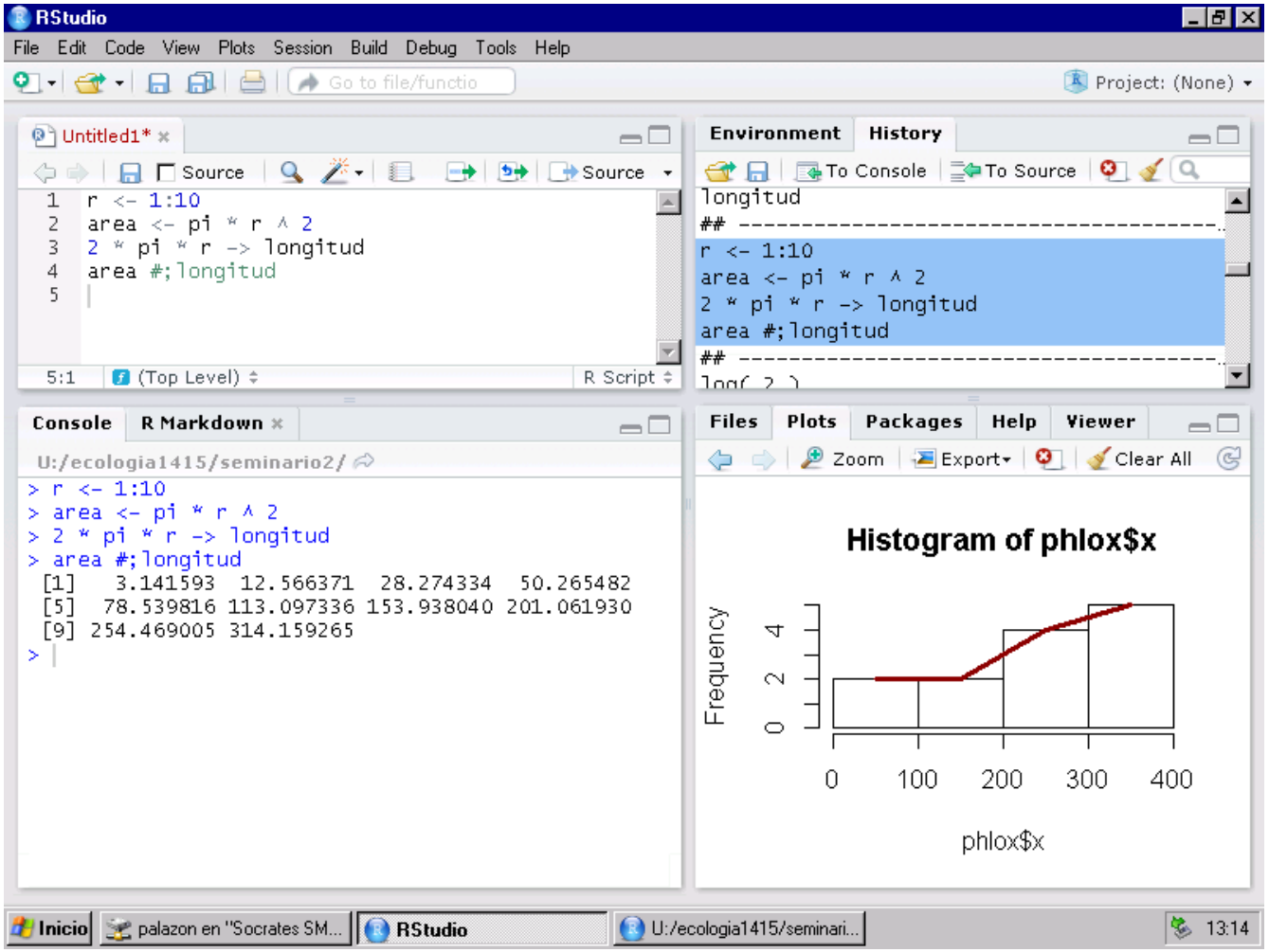
Creando un fichero de trabajo: script



Copiando el histórico

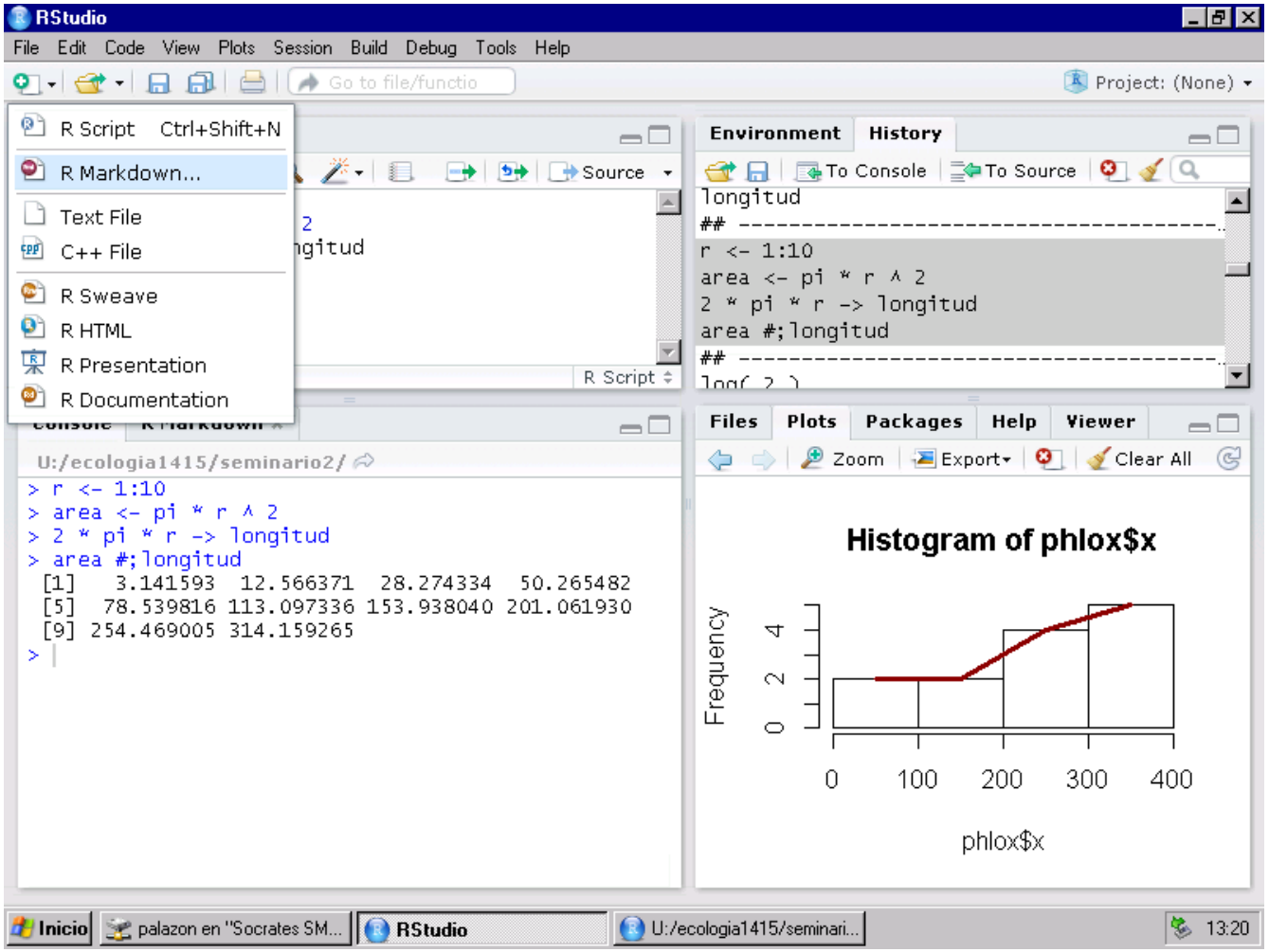


Un script para reutilizar

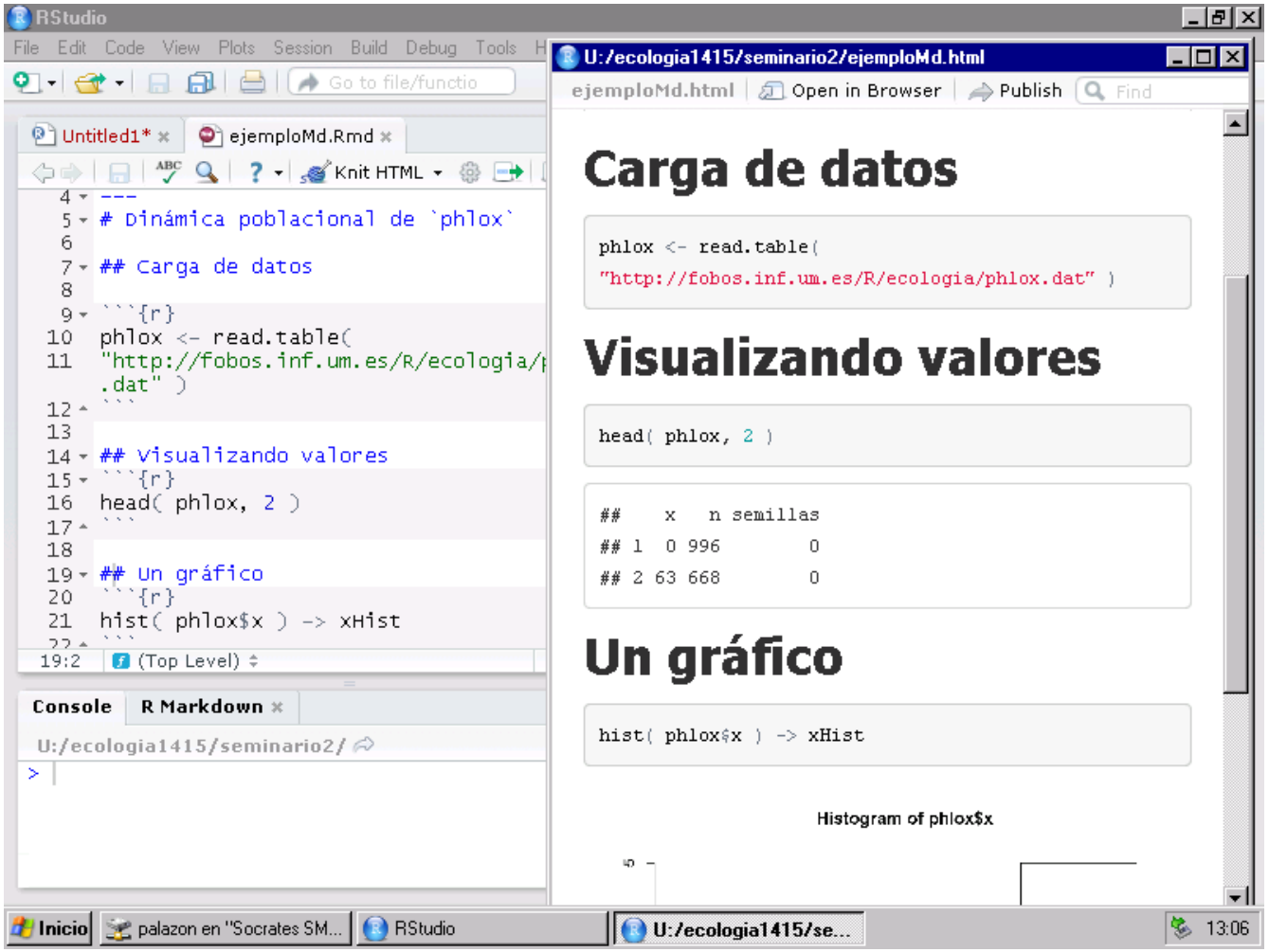


RR, más allá: reproducible research

Un fichero Rmd: mezcla de texto y R



Rmd: algo más que un *script*



Preguntas

¿Cómo seguir avanzando con R?

Cursos de R

- Básico, para los interesados: Try R, curso interactivo *on line* breve y muy práctico.
- Cursos *on line* de las distintas plataformas: Miriada X, Coursera, edX, ...
- *Open Course Ware* (OCW), busca `"read.table"`
- CRAN: *Contributed Documentation*
- Libros
- ...

¿Más preguntas?