Sesión 1: Una introducción a R

Métodos estadístico de investigación: Introducción a R y Rstudio

000R Team

2017/18

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?

Definción

- 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.

Qué es R

Vídeo traducido

`http://gauss.inf.um.es/videos/whatsR.webm`

Unas fotos de familia

- Interfaces

`http://www.statmethods.net/interface/guis.html`

- Instalación
- rstudio

rstudio

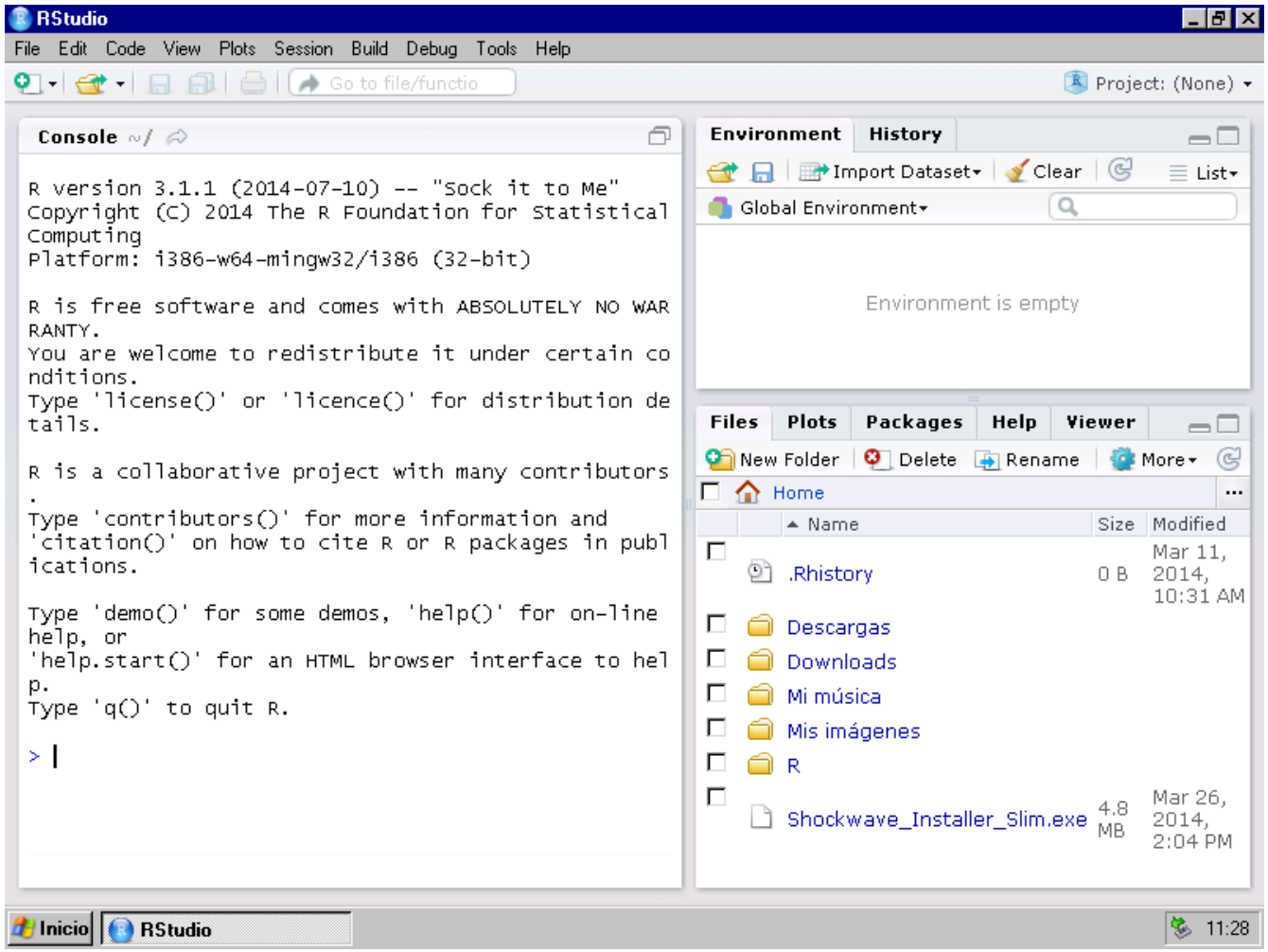


Figure 1

rstudio: la terminal

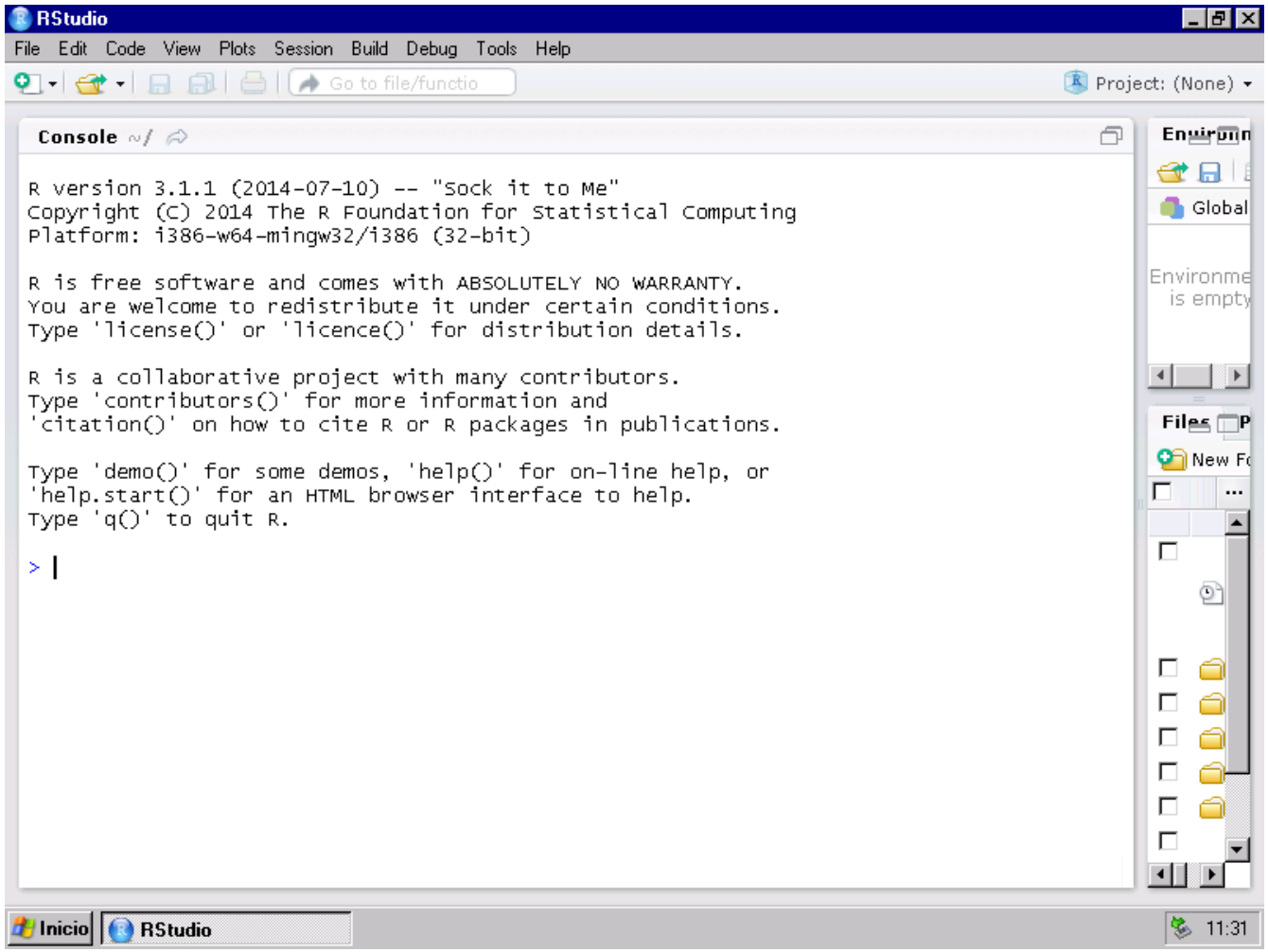


Figure 2

¿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.

Importancia de la comunidad

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

Sobre la notación y la tipografía

Comunicación 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
- Expresiones para R
- El *prompt*
  - >
  - +
- La despedida

rstudio

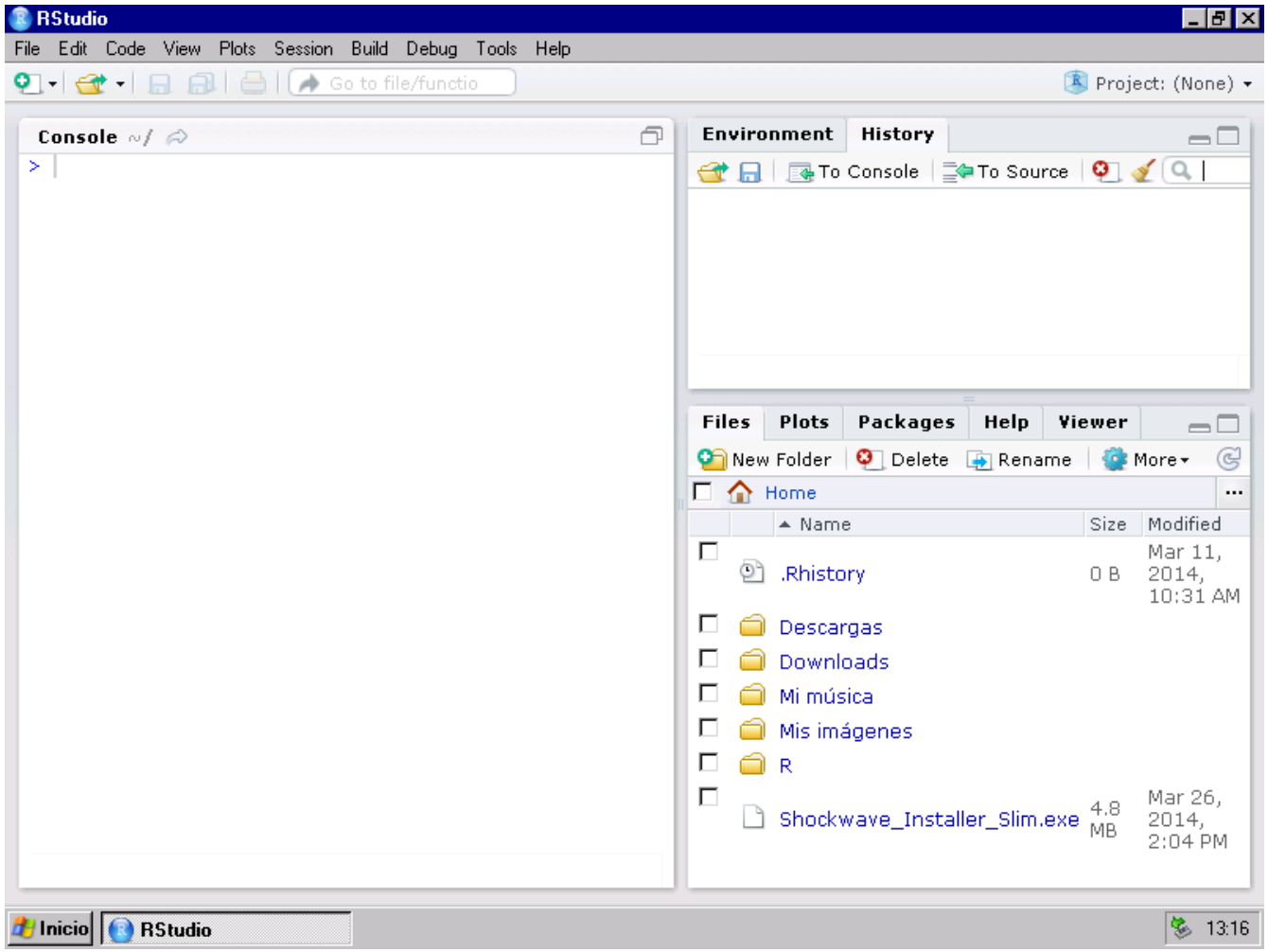


Figure 3

rstudio: Usando el tabulador

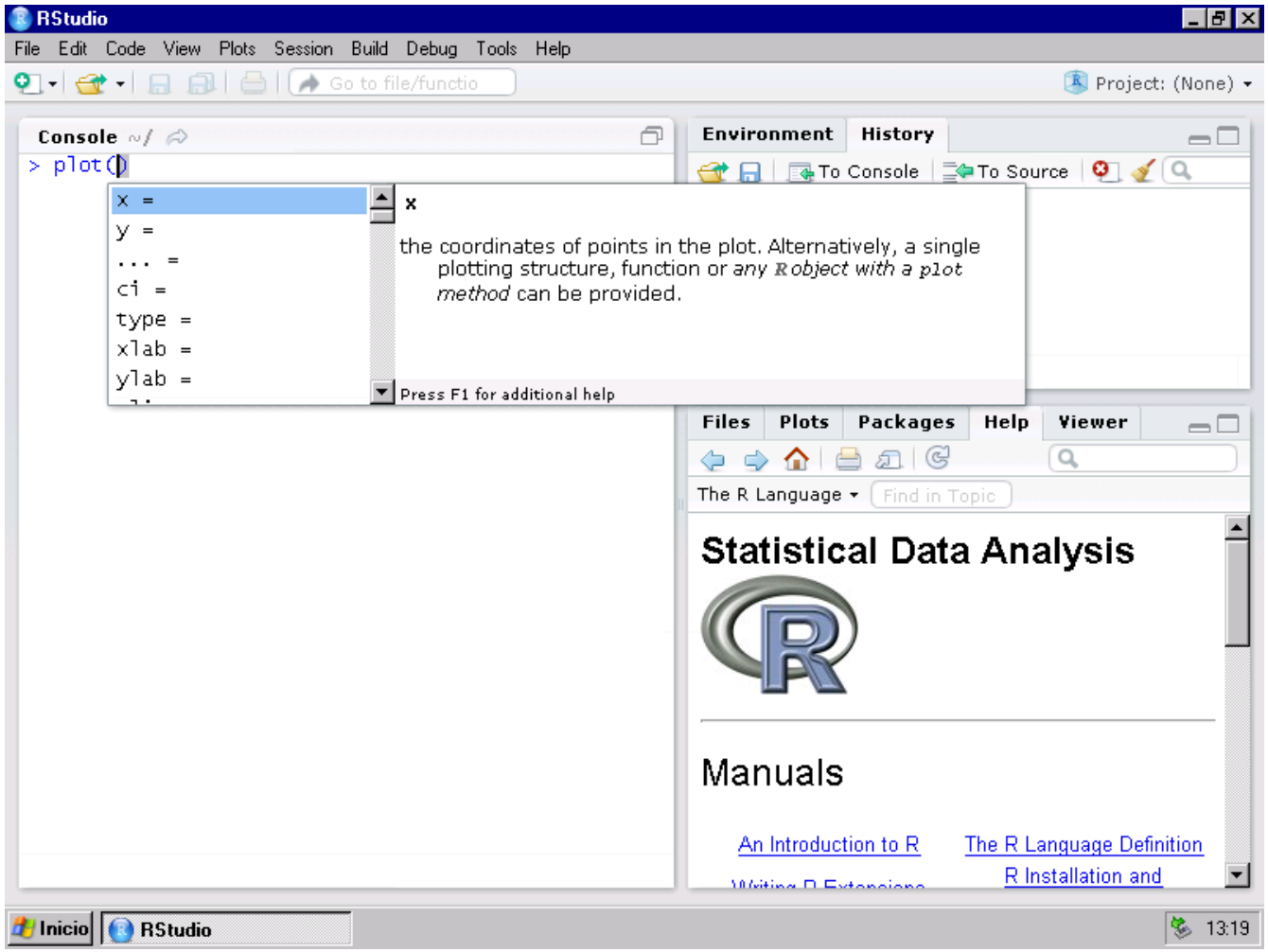


Figure 4

Elección del directorio de trabajo

rstudio: Entrada Session

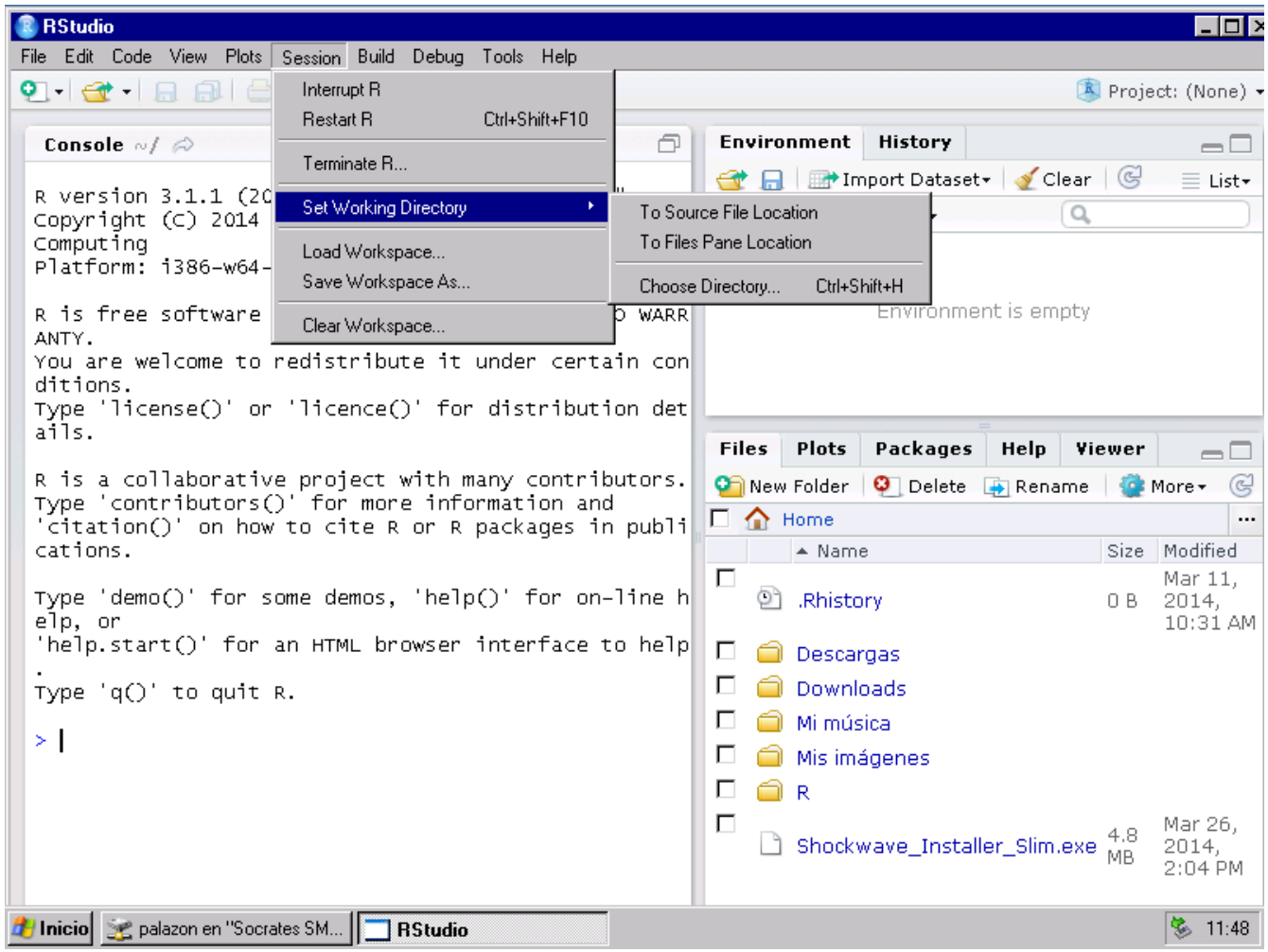


Figure 5

rstudio: Localizando el directorio de trabajo

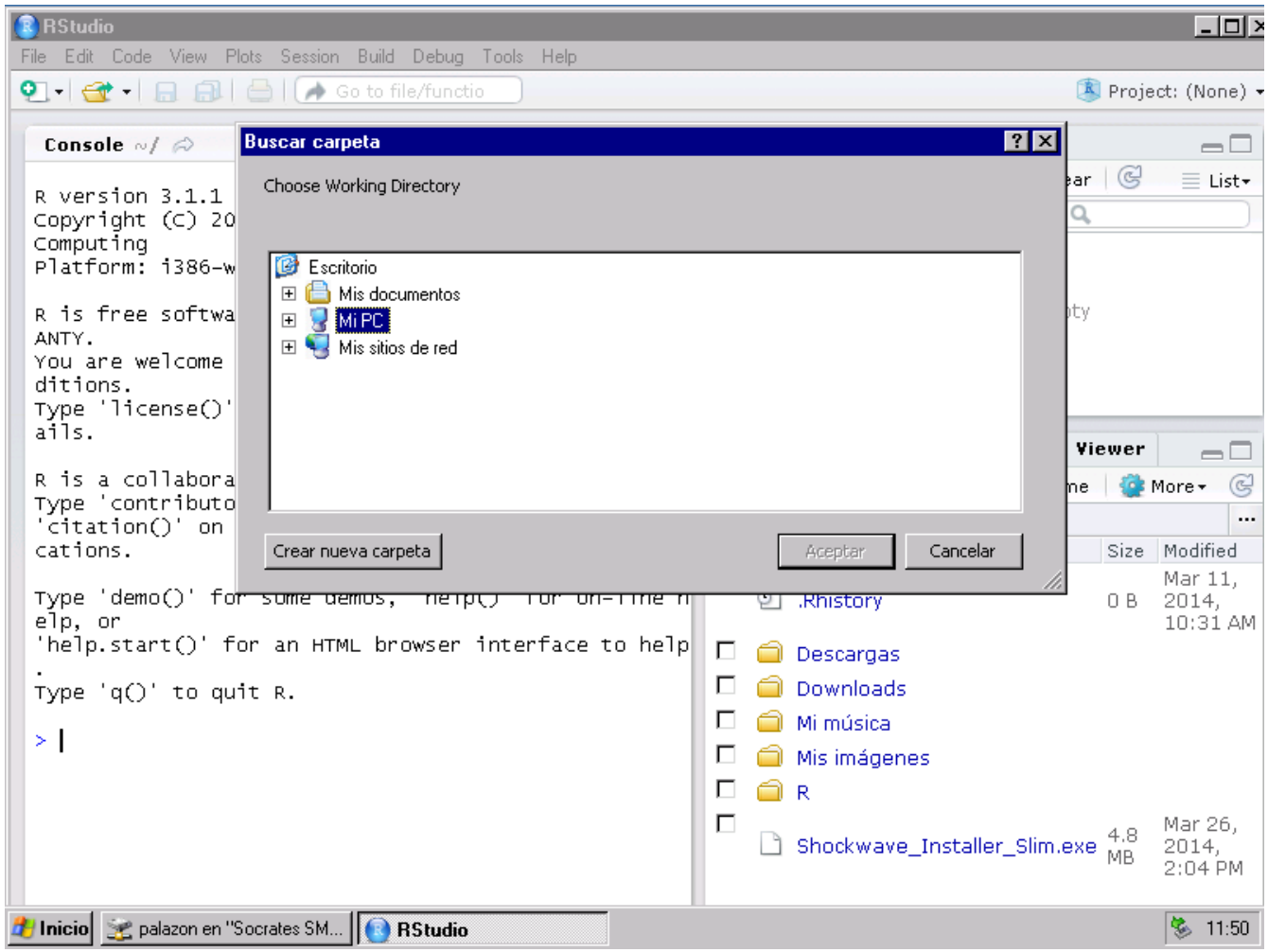


Figure 6

rstudio: Seleccioando la unidad sócrates

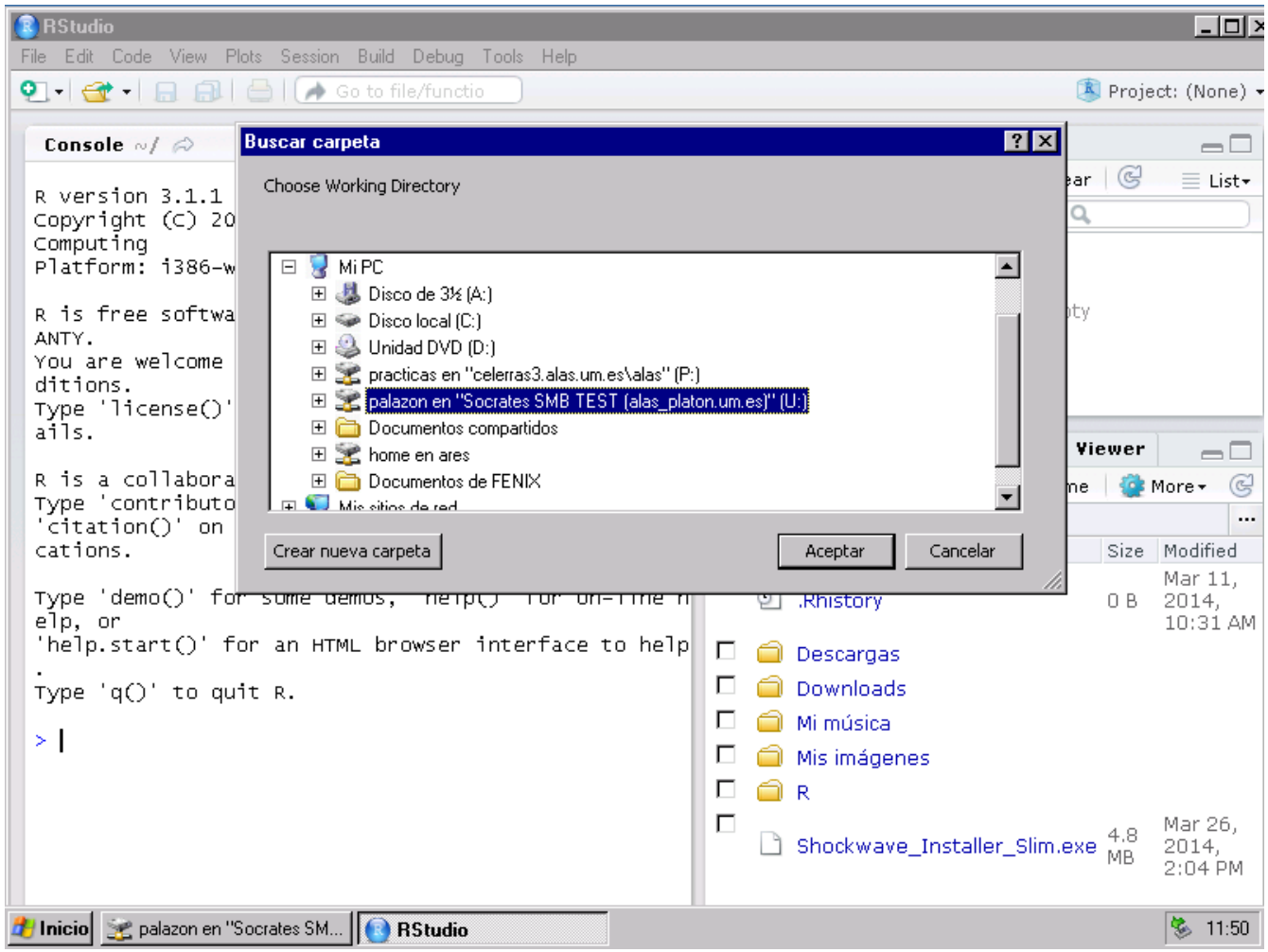


Figure 7

rstudio: directorio para el seminario

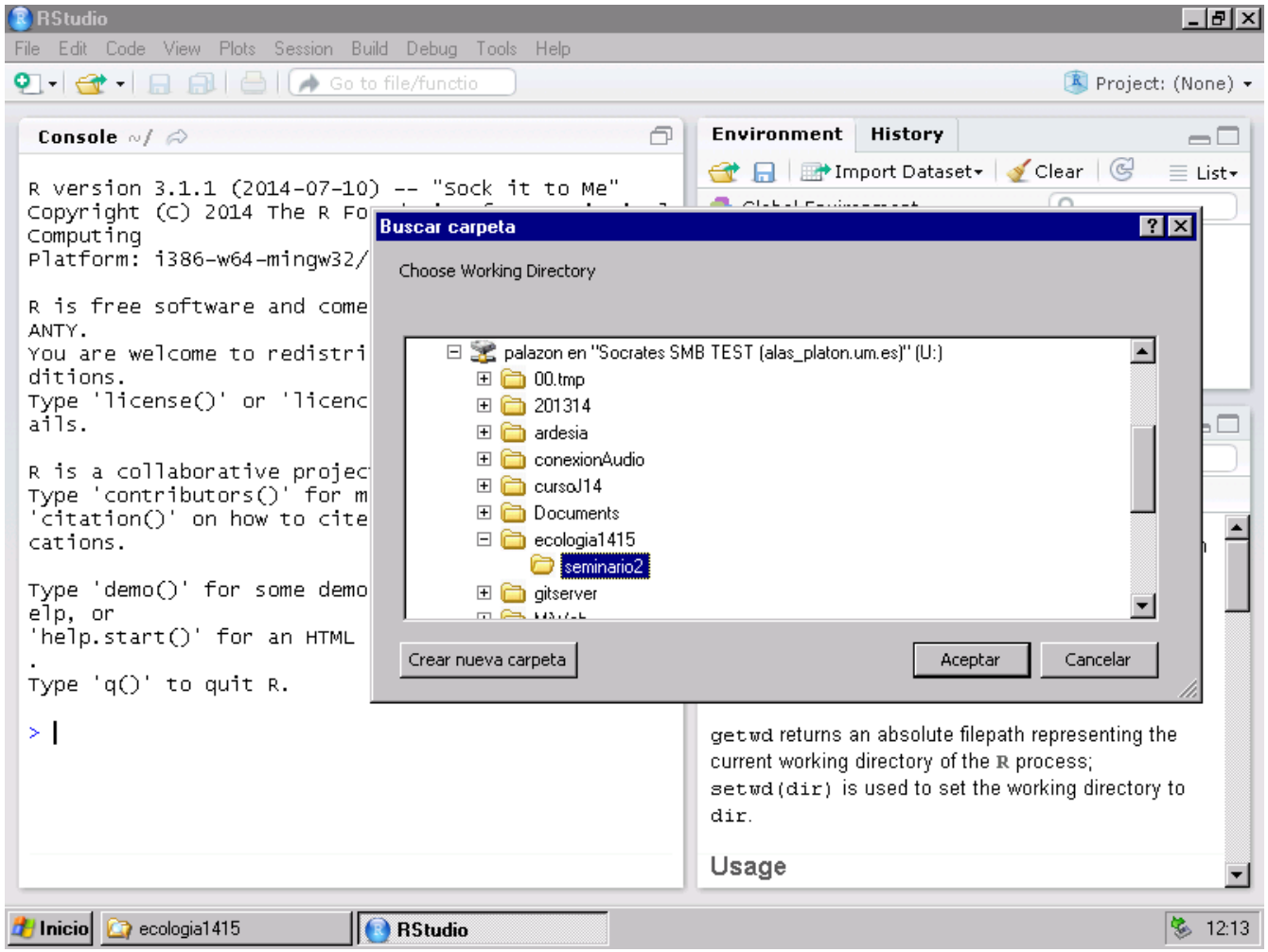


Figure 8

rstudio: ¡Una expresión!

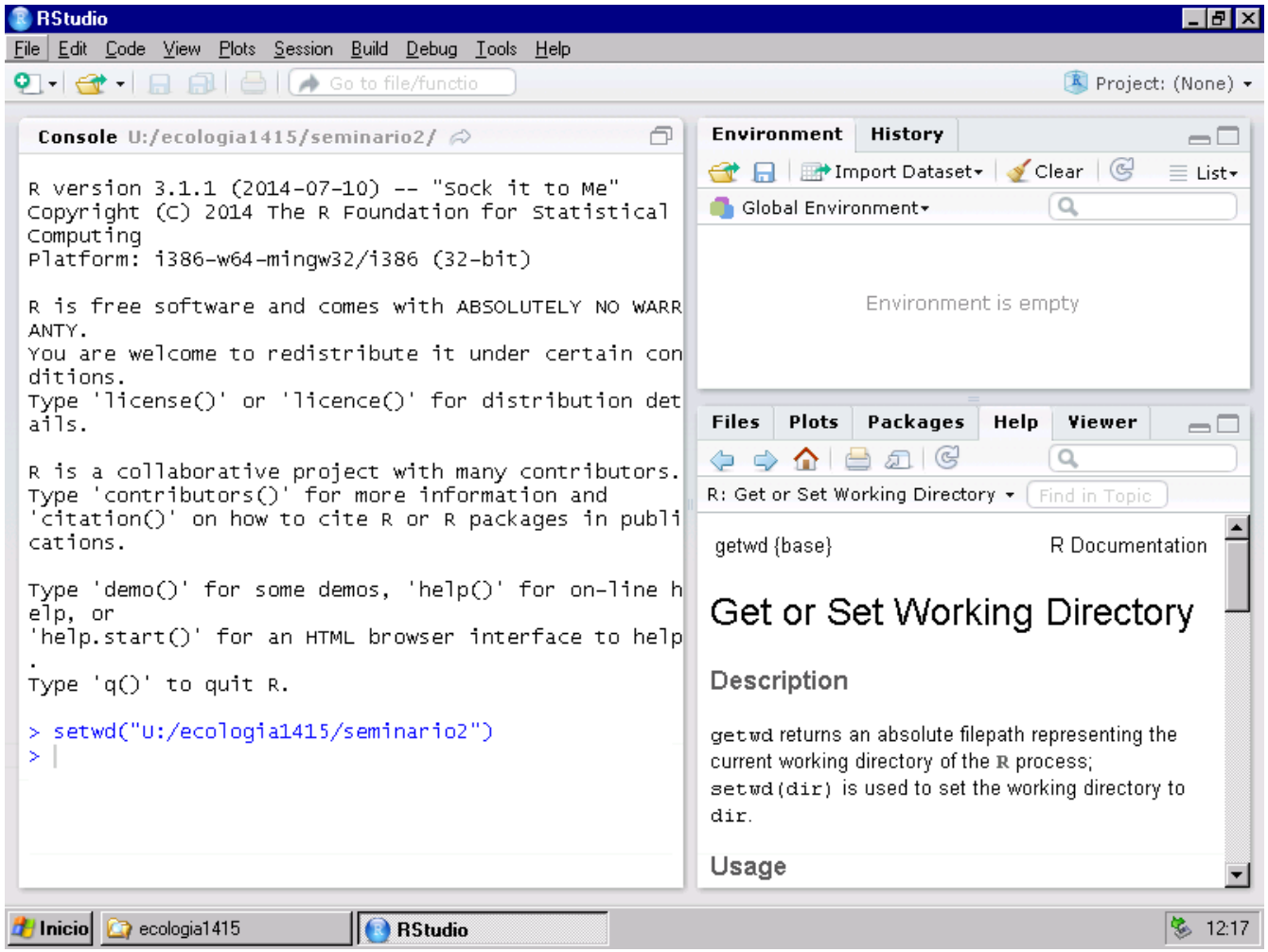


Figure 9

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 ($-\infty, \infty$): `-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?

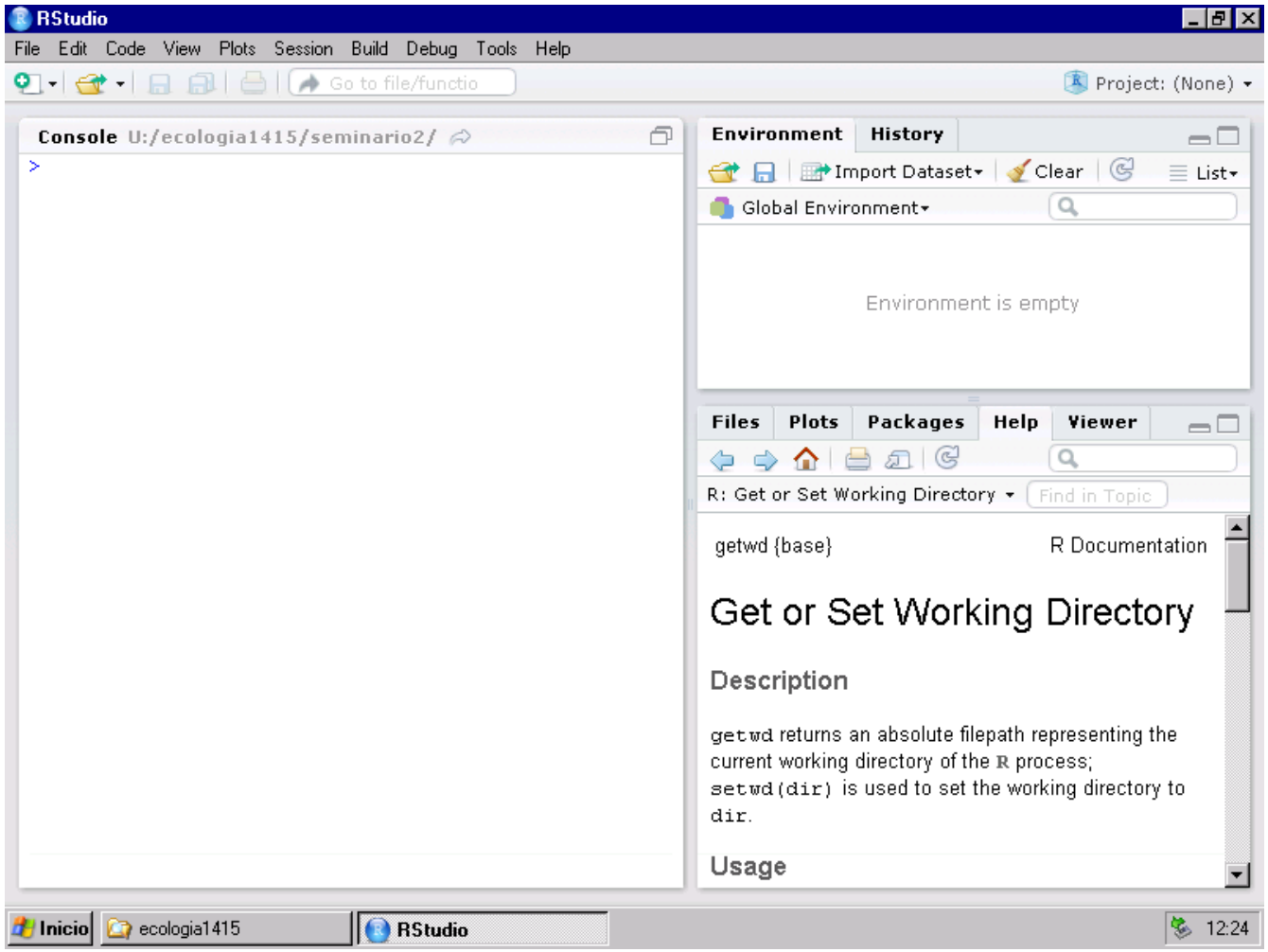


Figure 10

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

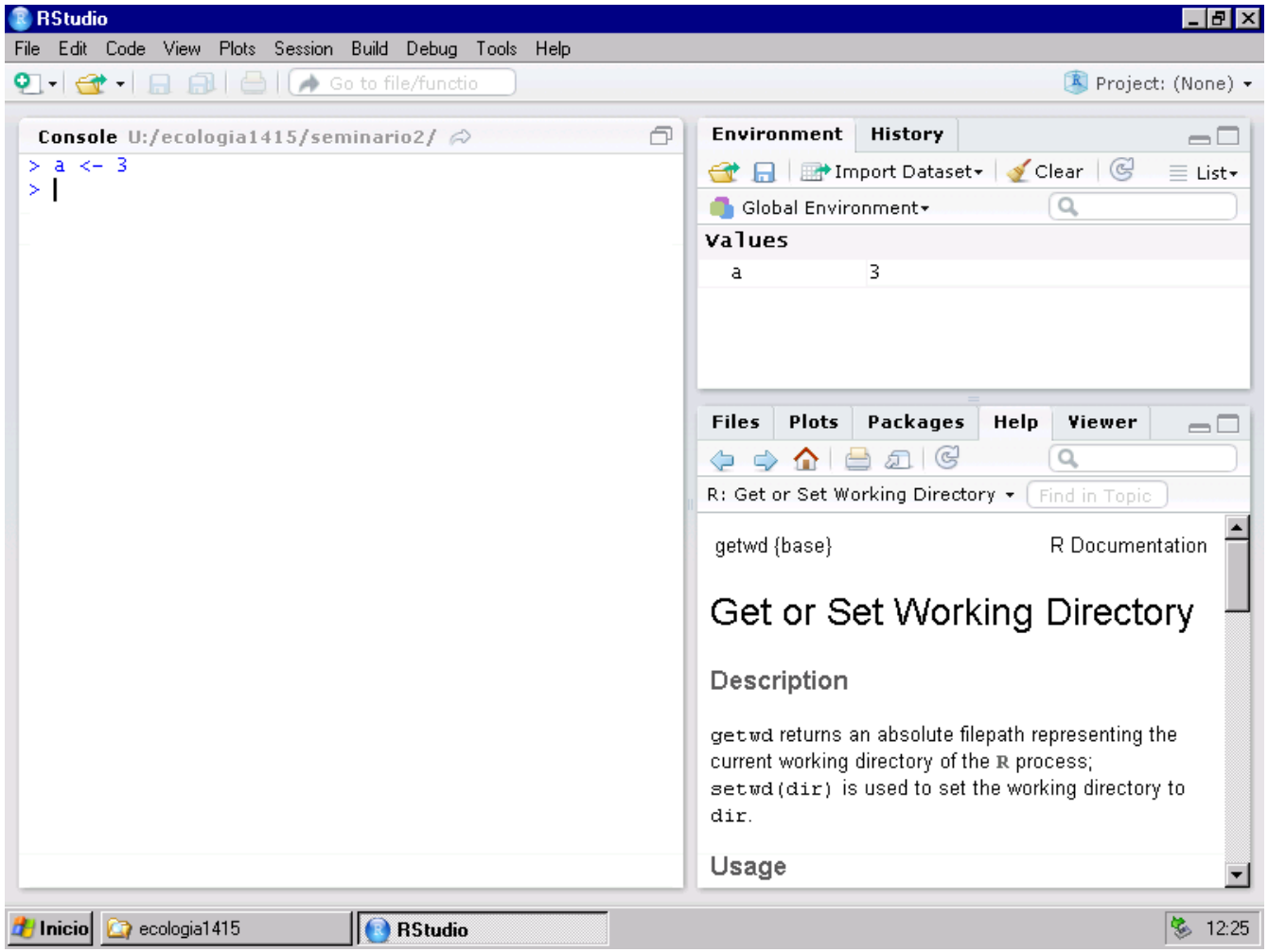


Figure 11

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
```

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: el fichero phlox.dat

```
read.table( "./files/ardilla.dat" )
```

```
  x   n    m
1 0 530 0.00
2 1 134 1.28
3 2  56 2.28
4 3  39 3.24
5 4  23 3.24
6 5  12 2.48
7 6   5 2.28
8 7   2 2.28
```

Ejemplo

```
phlox <- read.table(
  "./files/phlox.dat" )
head( phlox )
```

```
##     x   n semillas
## 1   0 996        0
## 2  63 668        0
## 3 124 295        0
## 4 184 190        0
## 5 215 176        0
## 6 264 172        0
```

Ejemplo

```
phlox[ , 3 ]
```

```
##  [1]   0   0   0   0   0   0   0  53 485 803 973
## [12]  95   0
```

```
phlox$n
```

```
##  [1] 996 668 295 190 176 172 167 159 154 147 105
## [12]  22   0
```

```
phlox[ , "n" ]
```

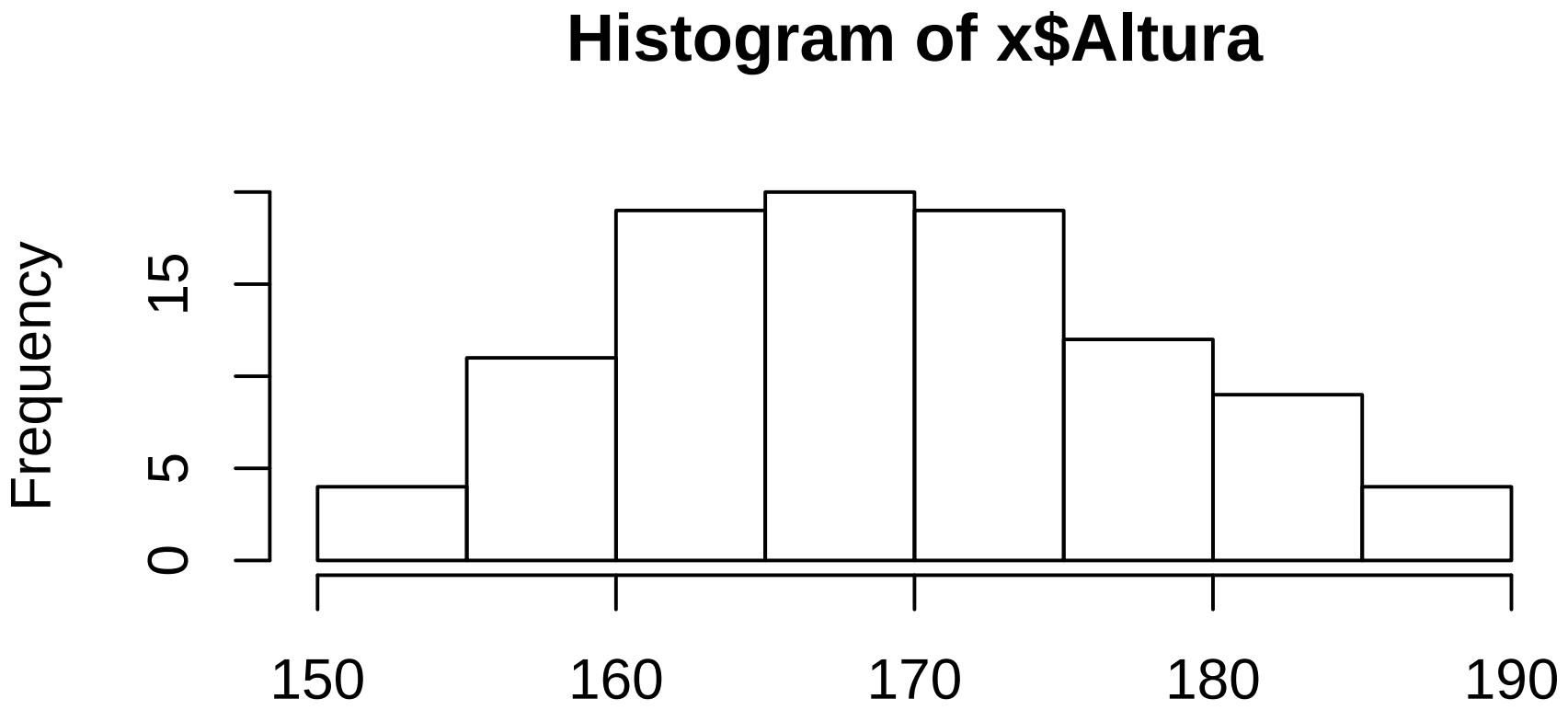
```
##  [1] 996 668 295 190 176 172 167 159 154 147 105
## [12]  22   0
```

Listas

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

Ejemplo

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



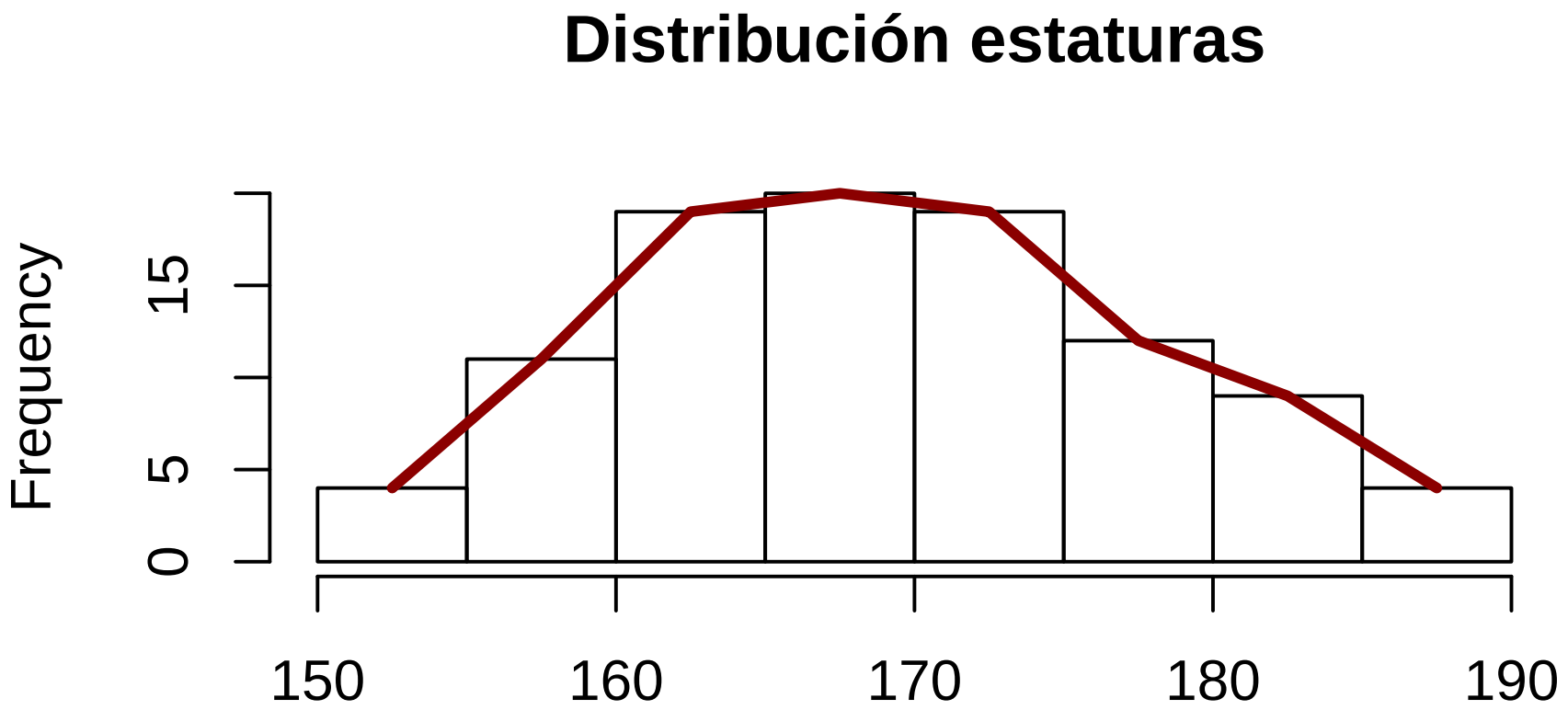
Ejemplo

```
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
## [4] 0.040816327 0.038775510 0.024489796
## [7] 0.018367347 0.008163265
##
## $mids
## [1] 152.5 157.5 162.5 167.5 172.5 177.5 182.5
```

Ejemplo

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



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

Creando un fichero de trabajo: script

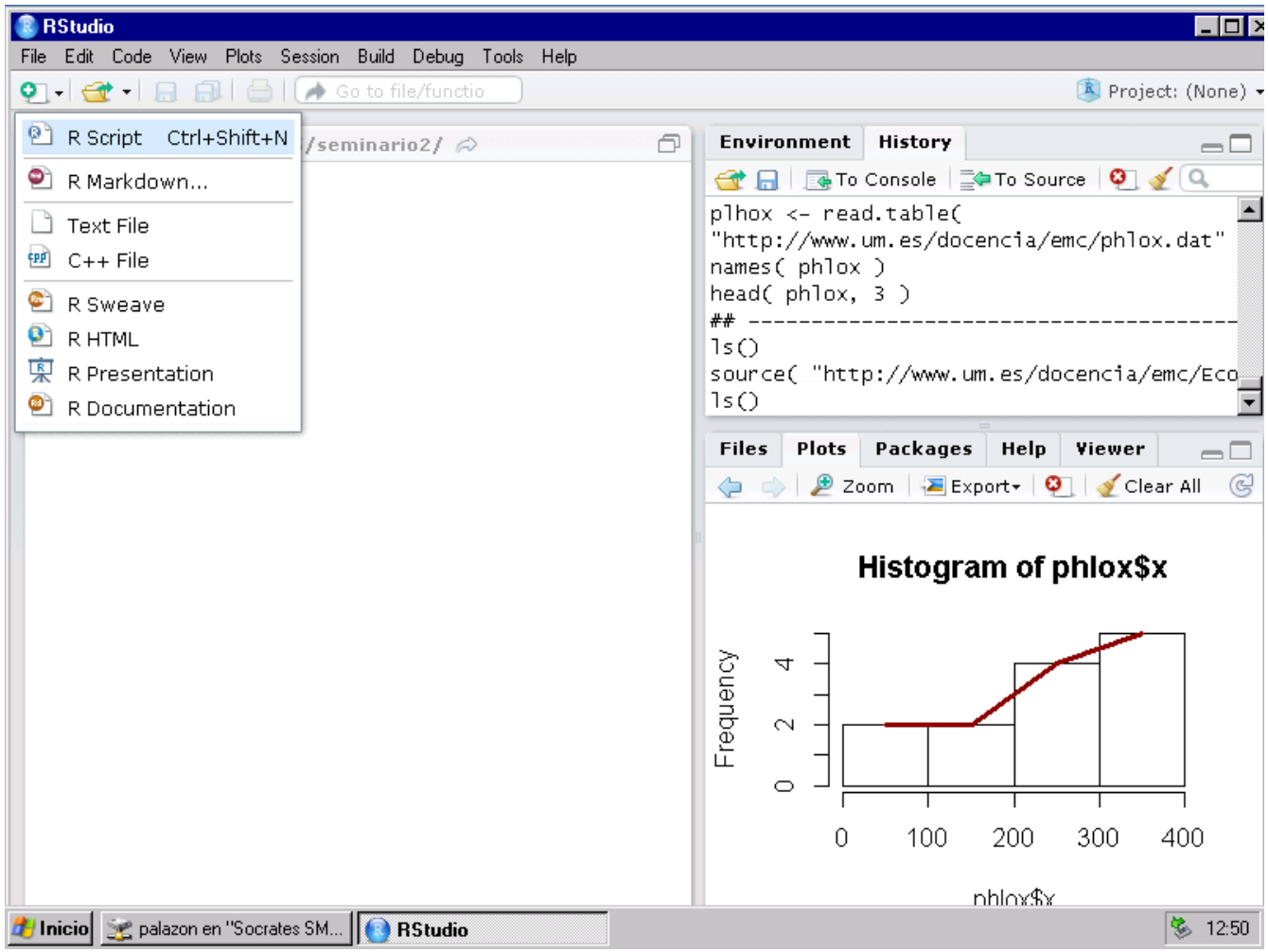


Figure 12

Copiando el histórico

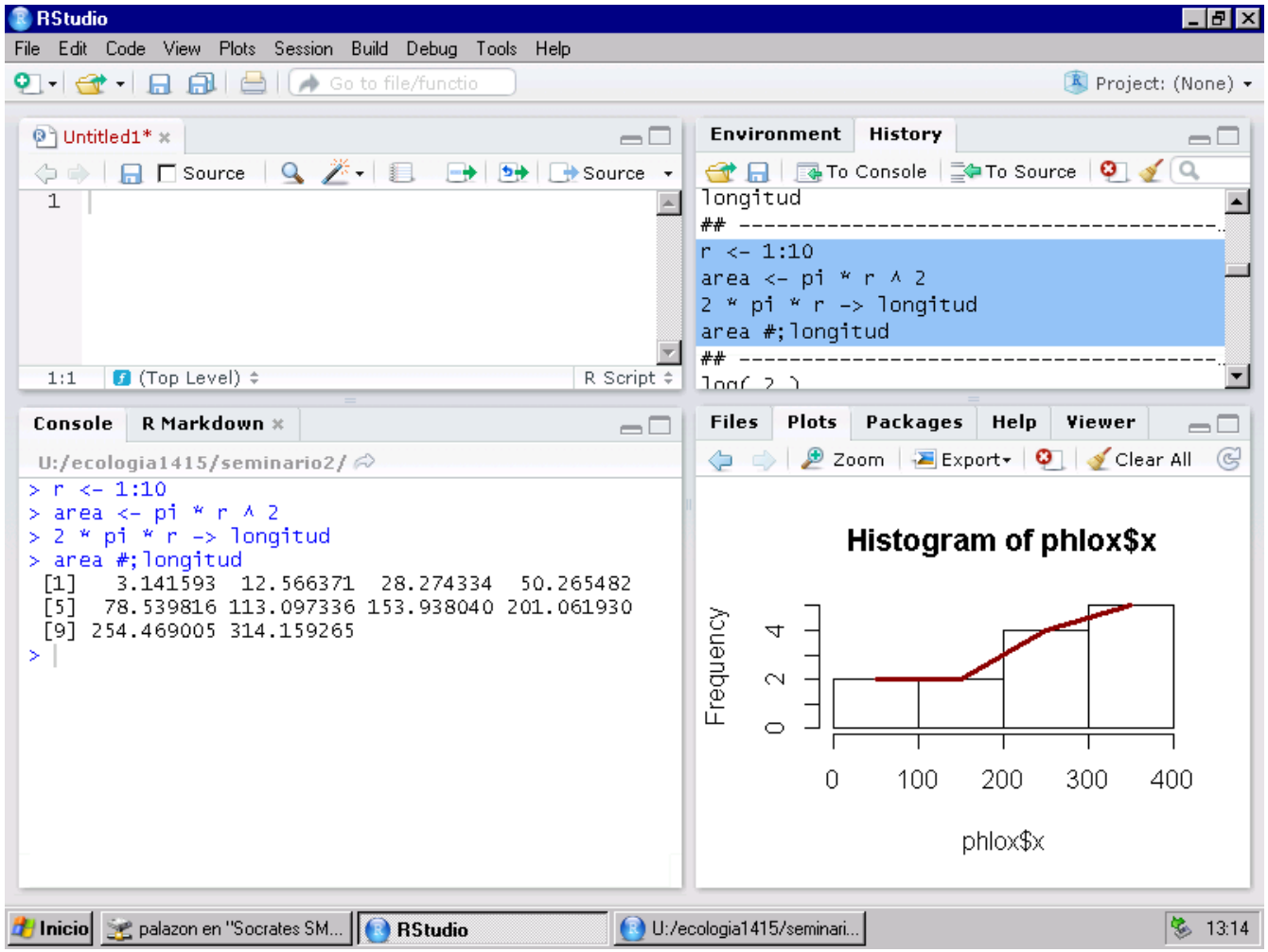


Figure 13

Un script para reutilizar

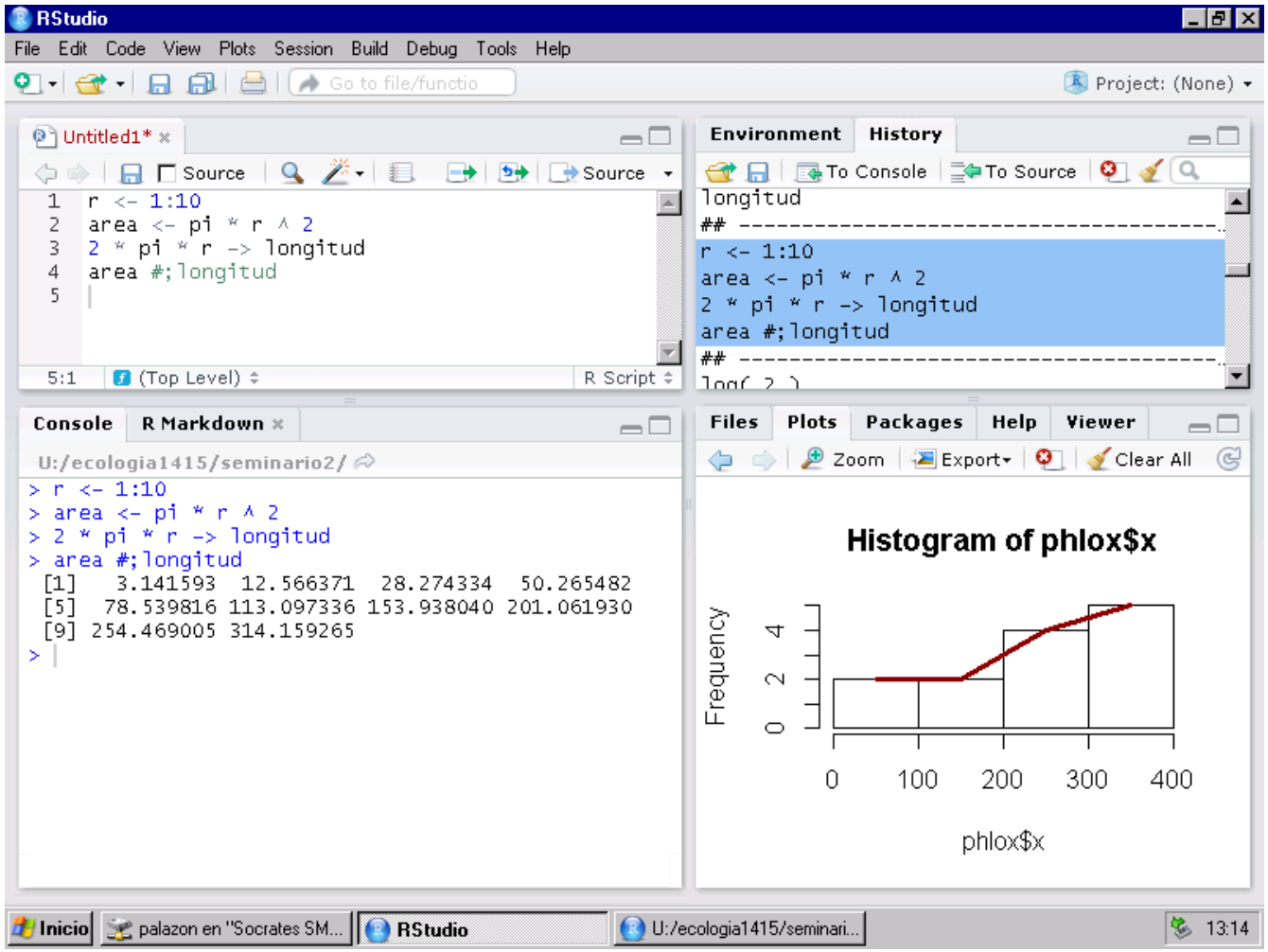


Figure 14

RR, más allá: reproducible research

Un fichero Rmd: mezcla de texto y R

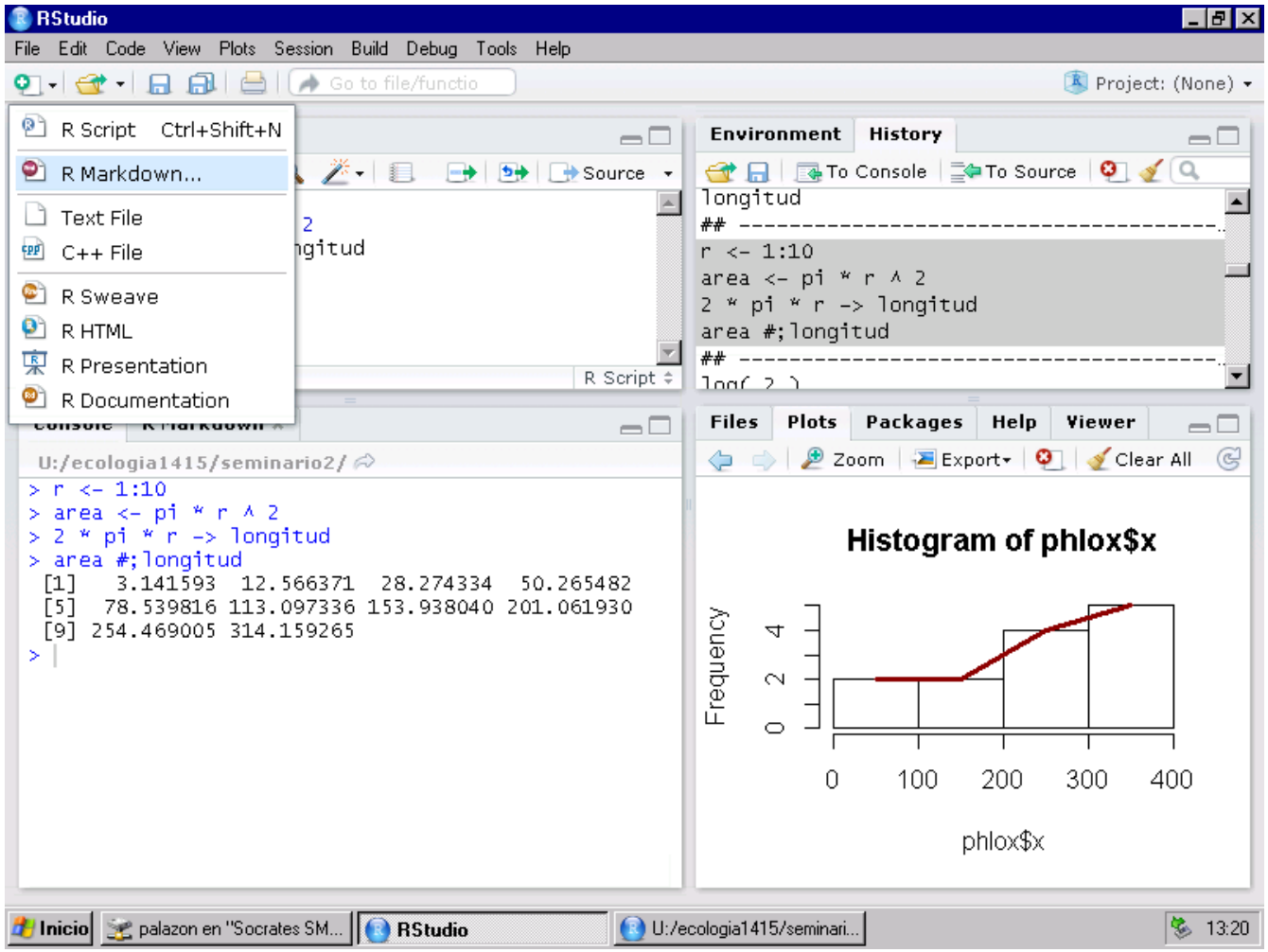


Figure 15

Rmd: algo más que un *scritp*

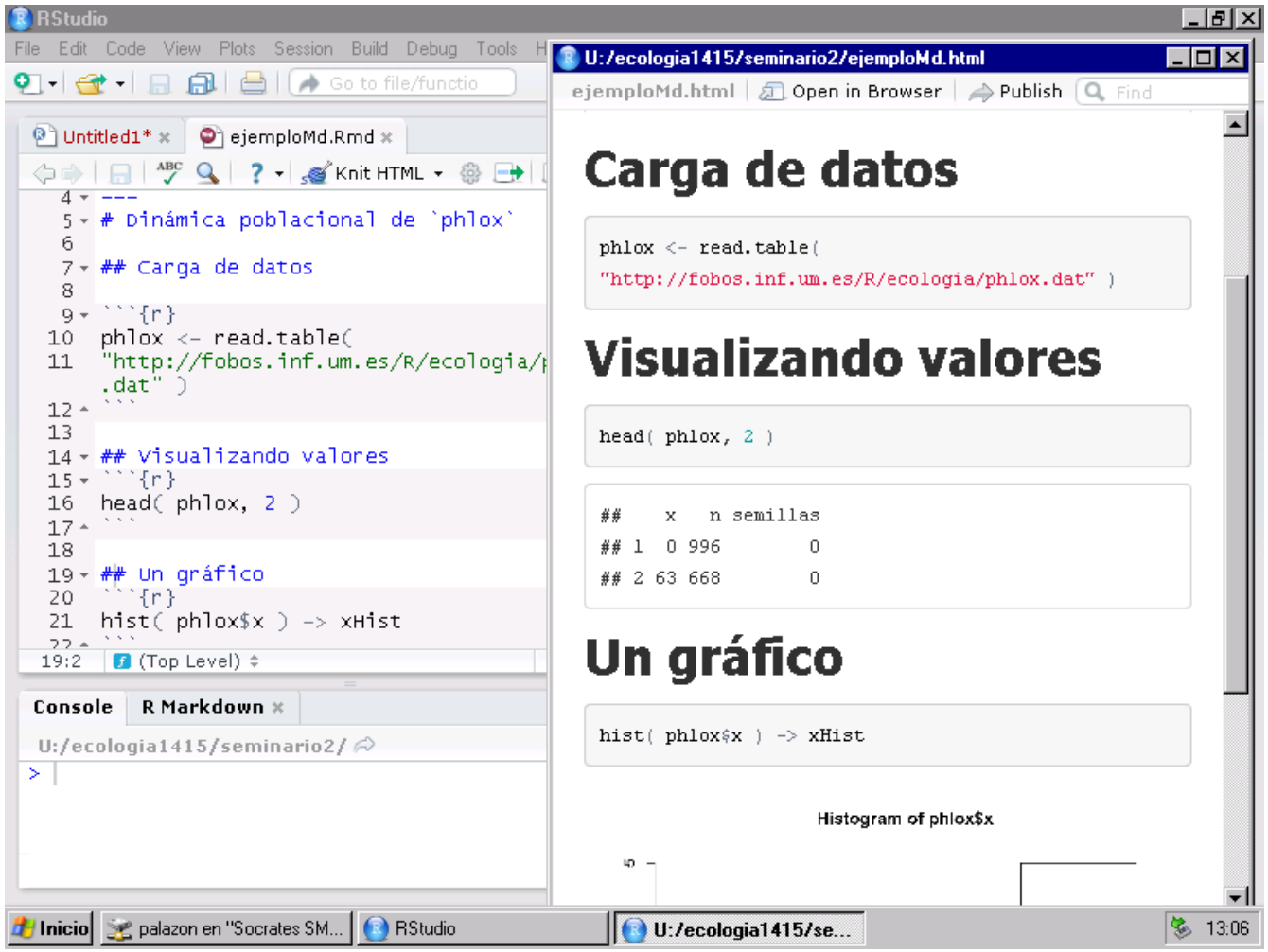


Figure 16

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?