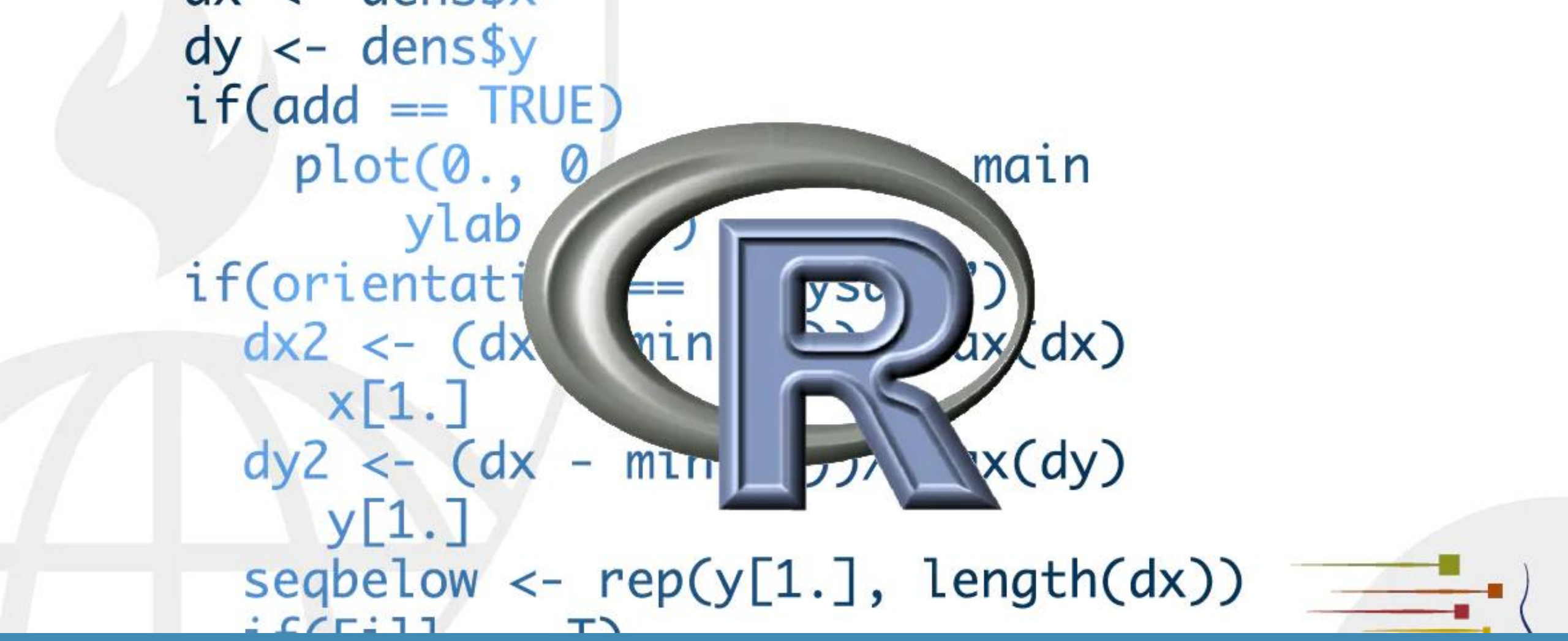


Introducción a R

Eroteida Sánchez García
esanchezg@aemet.es

Agencia Estatal de Meteorología

Contenidos

Referencias:

- **An Introduction to R (**W. N. Venables, D. M. Smith and the R Core Team)

  http://cran.r-project.org

- **An Introduction to R (***Alex Douglas, Deon Roos, Francesca Mancini, Ana Couto & David Lusseau*)

  https://intro2r.com/

- **¿Qué es R?**
- **Cómo acceder a R**
- **Rstudio**

- **Objetos y clases en R**
- **Operadores**
- **Estructuras de datos:**
  - **Vectores**
  - **Matrices**
  - **Arrays**
  - **Factores**
  - **Dataframes**
  - **Listas**
- **Funciones**
- **Paquetes**
- **Estructuras de control de flujo**

- **Ayuda en R**

- **Ejercicios prácticos**

¿Qué es R?

https://cran.r-project.org

- Lenguaje de programación con un histórico enfoque estadístico.
- Software libre. Licencia GNU GPL. Detalles mediante: licence() o license().
- Multiplataforma: Windows, Linux, macOS.
- Entorno colaborativo. Detalles mediante: contributors().
- CRAN (The Comprehensive R Archive Network) para descarga del software y paquetes.
- Alto nivel. Interpretado. Funcional. Orientación objetos.

Esquema de paquetes en R

¿Cómo acceder a R?

1. Acceso a la consola R

- Se abre una terminal Linux mediante el cliente ssh favorito en cada caso.
- Se carga el módulo R (ej.: module load aemet R)
- Se teclea simplemente R y ya tendríamos acceso a la consola.

```
sur:/scoor/tr61 > R

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> round(4.67,1)
[1] 4.7
```

Consola de R desde terminal

- Una vez accedido a la consola, se ha utilizado la función de redondeo round() del paquete base de R para redondear el número 4.67 a un decimal.
- Para salir de la consola basta con teclear quit().

¿Cómo acceder a R?

2. Ejecutar un script R desde terminal

- Se parte de un fichero .R donde está escrito el código que se quiere ejecutar.
- En la terminal linux, con los módulos cargados, se escribe simplemente:

  > Rscript [fichero].R

- Las ejecuciones a través de scripts pueden llegar a ser útiles de cara a la automatización y modulación de tareas.
- También podemos correr un script directamente (sin usar RScript en su llamada) escribiendo en la primera línea del fichero la ruta a Rscript en nuestro entorno.
- Para ejecutarlo escribimos:

  > ./[fichero].R

script2.R

```
1 #!/usr/local/apps/gcc/4.8.5/R/4.2.0/bin/Rscript
2
3 print("Hola a todos, soy el ejemplo2, el redondeo a un decimal de 2.67 es:")
4 round(2.67,1)
```

Script de R con la ruta del entorno

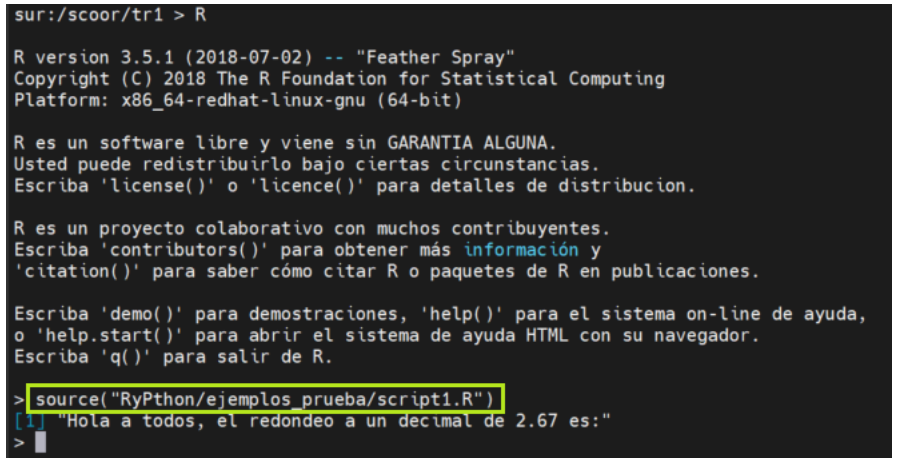
¿Cómo acceder a R?

3. Ejecutar un script R desde la consola de R

- Desde la consola de R, se puede ejecutar el código de un fichero.

- Una vez que hemos accedido a la consola R (por ejemplo tecleando “R” en la terminal Linux) escribimos:

  > source([ruta/nombre_fichero.R])



Ejecutar un script de R desde la consola de R

RStudio

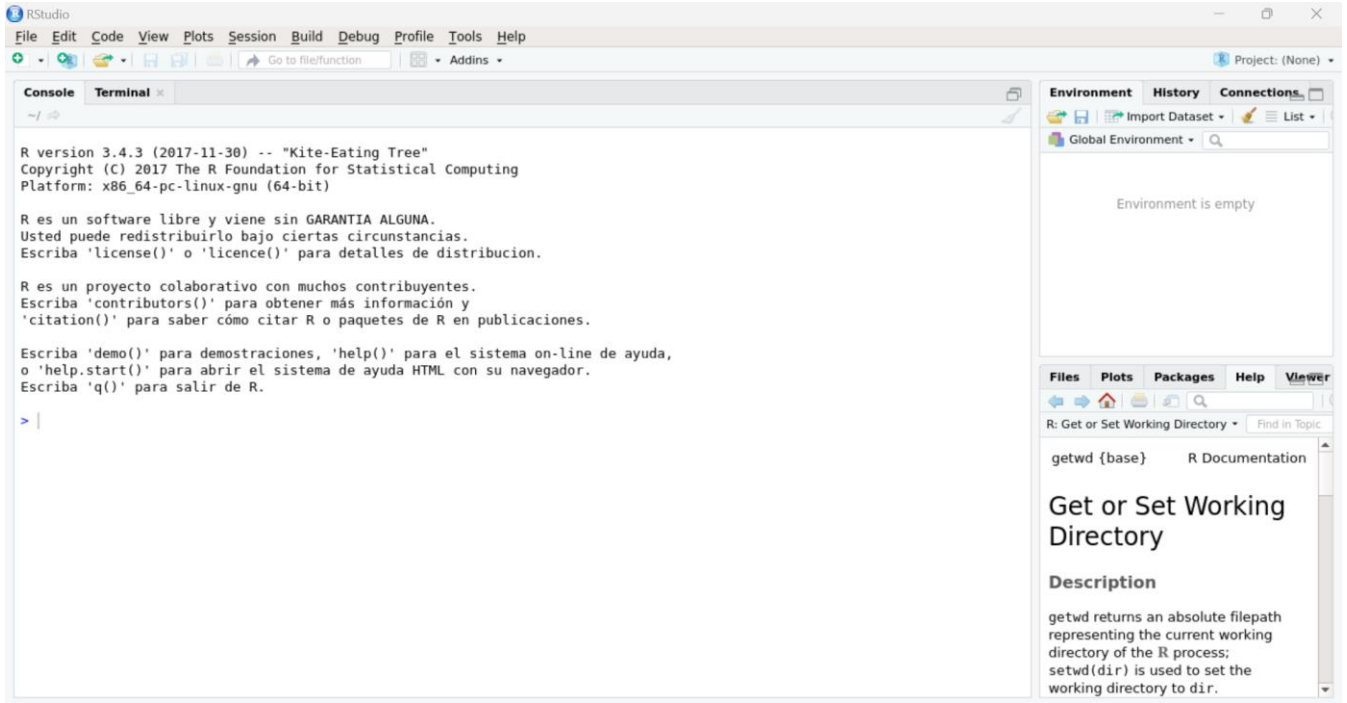
https://posit.co/downloads/

- RStudio es un IDE (Integrated Development Environment) práctico e interesante para trabajar con R. Dispone de ediciones de código abierto (licencia AGPL v3) y comerciales y es multiplataforma.

- Disponemos de dos maneras interesantes para utilizarlo:
  - RStudio Desktop
  - RStudio Server

RStudio Desktop

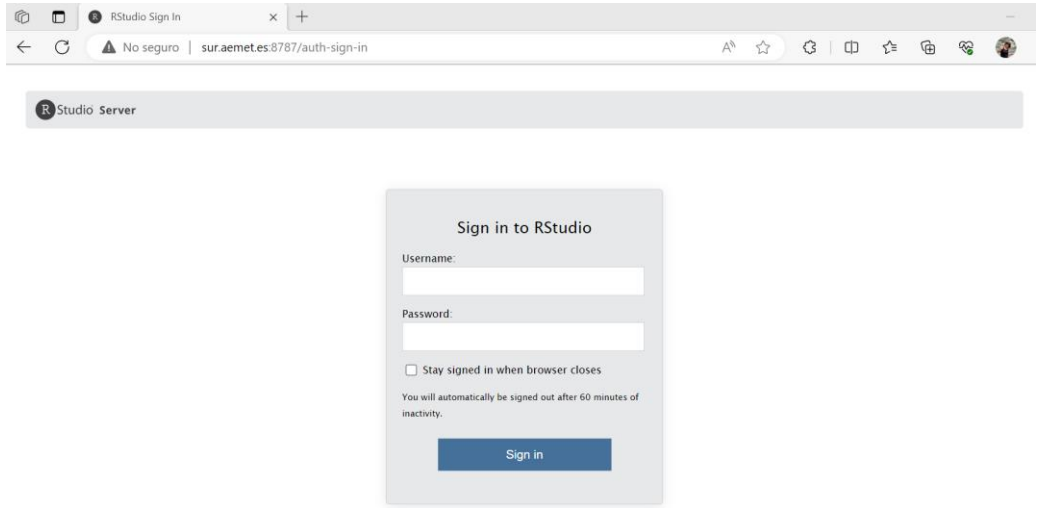
- Es la modalidad escritorio para desarrollo en local.
- Se puede instalar en cualquier sistema operativo: Linux, Windows, …
- En los servidores Linux de AEMET está disponible. Requiere la carga de su módulo rstudio previo a la ejecución con el comando rstudio.
- También dispone de versiones Linux o Windows instalables para nuestro uso en PC local.



RStudio Server

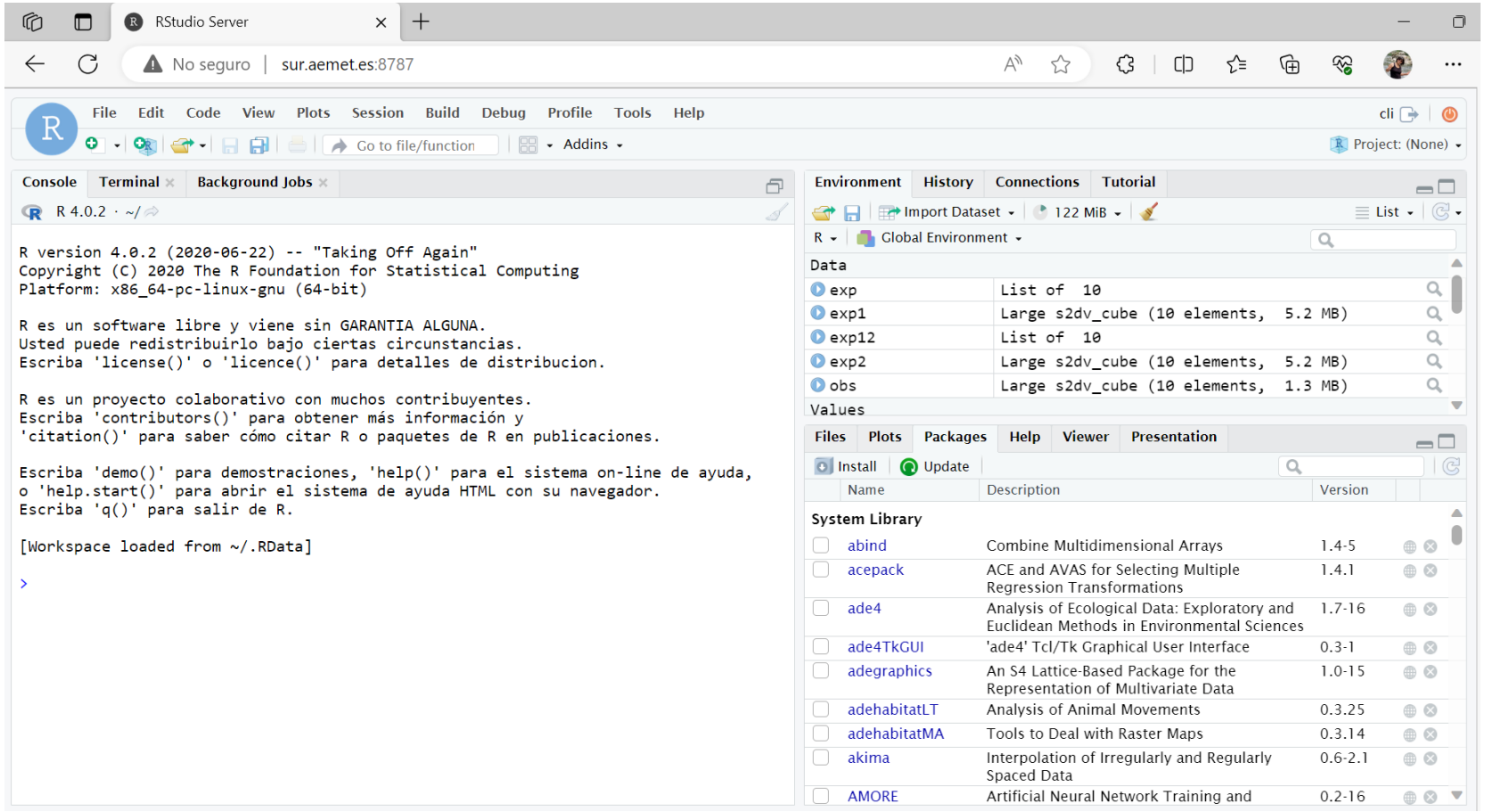
- El RStudio Server permite la ejecución de RStudio (alojado en un servidor), a través del navegador.

- Para ejecutarlo:
  - Abrimos un navegador, desde nuestro PC o portátil local. Por ej.: Firefox, Chrome.
  - Nos dirigimos a la siguiente url: http://sur.aemet.es:8787/
  - Se introducen las credenciales correspondientes a nuestro usuario del curso.



Pantalla de acceso a Rstudio Server

RStudio Server



Apariencia inicial de RStudio Server

Ejercicios

- Abrir una sesión de Rstudio Server con el usuario asignado.
- Crear un directorio, ‘cursoIntroR’, dentro de nuestro home.
- Crear un nuevo proyecto en el directorio que acabamos de crear.
- Importar la carpeta ‘materialCursoIntroR.zip’ dentro del directorio creado.
- Crear la carpeta ‘misEjercicios’ para guardar todos los ejercicios que vamos haciendo.

Objetos y Clases en R

OBJETO X EN R

- Un objeto es un almacén de información. Posee un identificador y una estructura para almacenar la información.
- Actuamos sobre los objetos aplicándoles funciones.
- Generalmente creamos un objeto mediante el **operador de designación (<-):**

  x <- pi * (1:4)

- Todo en R es un objeto (los datos, los resultados de los análisis, y hasta las funciones).
- Los objetos se quedan guardados en memoria para su utilización posterior.
  - En cualquier momento podemos obtener un listado de los objetos almacenados en la memoria: **ls()**
  - Podemos determinar si un objeto existe o no en la memoria: **exists(x)**
  - Los objetos se pueden eliminar de la memoria: **rm(x)**
- Los nombres de los objetos pueden incluir letras, números, puntos y guiones bajos. Deben empezar siempre con una letra o un punto y si empiezan con un punto, a este no puede seguirle un número.

Objetos y Clases en R

TIPOS DE DATOS

- El tipo **logical** permite almacenar valores de verdad lógica, por ej. TRUE, FALSE
- El tipo **integer** permite almacenar números enteros, ej. 5L (L significa que lo almacena como un entero)
- El tipo **numeric** permite almacenar números reales, ej. 2.53, 5
- El tipo **complex** permite almacenar números complejos, ej. 1+6i
- El tipo **character** permite almacenar cadenas de texto; se utilizan las comillas (", ') para delimitar las cadenas de caracteres, ej. "Hola mundo".
- El tipo **factor** representa datos categóricos. Se crea con el comando factor() a partir de tipo character.

Cualquier dato en R debe pertenecer a uno de los siguientes tipos o clases fundamentales:
lógico, entero, numérico, complejo o carácter.

- Algunos casos **especiales**: dato faltante (**NA**), valor infinito (**Inf**), valor no numérico (**NaN**):
  - x <- c(1:4,NA,6:10)
  - 5 / 0
  - Inf-Inf

Los objetos en R tienen **atributos**:

- names
- dimnames
- dim
- class
- attributes

Para examinar características de un objeto existen varias funciones:

- class(x)
- str()
- typeof()
- length()
- dim()
- attributes()

Operadores

ARITMÉTICOS:

+	suma	5 + 2
-	resta	5 - 2
*	multiplicación	5 * 2
/	división	5 / 2
^	potencia	5 ^2
%%	módulo	5 %% 2
%/%	division de enteros	5 %/% 2

COMPARATIVOS:

<	menor que	5 < 2
>	mayor que	5 > 2
<=	menor que o igual a	5 <= 2
>=	mayor que o igual a	5 >= 2
==	igual a	5 == 2
!=	distinto de	5 =! 2

LÓGICOS:

!	NO lógico	! x
&, &&	AND lógico	x & y
|, ||	OR lógico	x | y
xor()	OR exclusivo	xor(x, y)

Ejercicios

- Crear dentro de la carpeta ‘misEjercicios’ un script de R llamado ‘ejercicio1’ para empezar a escribir código en R para resolver los siguientes ejercicios.
- Practicar el uso de las operaciones básicas matemáticas (sumar, restar, dividir, multiplicar, potencia, etc).
- Calcular el punto de intersección con el eje x de la siguiente función: f(x)=47x+5
- Calcular la raíz cúbica de 4096
- ¿Qué obtenemos si dividimos un número entero por 0? ¿Y si dividimos 0/0?
- ¿Cómo representamos un valor ‘missing’?
- Crear una variable llamada ‘first_num’ y asignarla el valor 42. Crear otra variable llamada ‘first_char’ y asignarla el valor ‘mi primer variable’. ¿Qué aparece en nuestro ‘Environment’? Listar las variables que tenemos y borrar la primera de ellas.
- Calcular los puntos de intersección con el eje x de la función: f(x)=x2+3x−4
- Define dos variables que contengan el nombre de dos personas y otras dos variables que contengan la edad y muestra en pantalla una frase en la que muestres la edad que tiene cada uno, la edad media de los dos y la suma de sus edades. (p.ej. "Alan tiene 20 años y María tiene 19 años. Juntos tienen 39 años y su edad media es 9.5 años.“). Usa la función paste() para unir las palabras.

Estructuras de datos

VECTORES

- Los vectores son secuencias de elementos del mismo tipo.
- Podemos referirnos a elementos concretos del vector mediante un índice.

```
x <- c('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec')
x
is.vector(x)
length(x)
x[1]
a <- 1; x[a]
x[2:4]
x[c(1,3,3,5)]
x[-1]
x[x=='Mar']
x[x!='Mar']
```

Estructuras de datos

VECTORES

- Ordenar:
  - rev(x)
  - sort(x)
  - rank(x)
  - order(x)

- Valores extremos:
  - max(x)
  - min(x)
  - range(x)

- Buscar:
  - which(x==max(x))
  - w <- which(x==max(x)); x[w]

- Muestreo:
  - sample(x, 7)
  - sample(x, 7, replace=T)

- Combinación de vectores:
  - y <- c('Dec','Jan','Feb').
  - union(x,y)
  - intersect(x,y)
  - setdiff(x,y)

Estructuras de datos

FACTORES

- Los factores son vectores de variables categóricas.
- Además de los valores en sí, contienen los valores de los posibles niveles definidos:
  - x <- factor(c('A','C','C','C','A','A','C'), levels = c('A','B','C'))
  - x
  - x[1]
  - levels(x)

- Podemos asignar un orden a los niveles:
  - flood_level <- factor(c("low", "high", "medium", "high", "low", "medium"), levels = c("low", "medium", "high"), ordered = TRUE)
  - min(flood_level)

Estructuras de datos

MATRICES

- Las matrices (matrix) permiten almacenar datos en tablas de dos dimensiones.
- Los datos deben ser del mismo tipo, y todas las columnas y filas deben tener el mismo tamaño.
  - y <- matrix(1:9, ncol=3)
  - y
  - is.matrix(y)

- Podemos crear una matriz concatenando vectores por filas y por columnas:

  - y <- cbind(c(1:7), c(8:14))
  - cbind(y, y[,1])
  - rbind(y, y[1,])

Estructuras de datos

MATRICES

- **Operaciones** con matrices:
  - mat1 <- matrix(1:4, nrow=2)
  - mat2 <- matrix(2:5, nrow=2)
  - mat1 + mat2
  - mat1 * mat2 # elemento por elemento
  - mat1 %*% mat2 # multiplicación matricial

- **Dimensiones** de una matriz:
  - dim(y)

- **Selección de elementos** de una matriz:
  - y[2,1]
  - y[,1]
  - y[1,]
  - y[,1:2]

- **Matriz traspuesta**:
  - t(y)

- **Nombres para filas y columnas**:
  - rownames(y) <- c('A','B','C')
  - colnames(y) <- c('a','b','c')

Estructuras de datos

ARRAYS

- Los arrays son matrices de más de dos dimensiones:
  - z <- array(1:42, dim=c(7,3,2))
  - dim(z)
  - z[,,1]

Estructuras de datos

DATAFRAMES

- Los dataframes son tablas compuestas de uno o más vectores y/o factores de la misma longitud, pero que pueden ser de distintos tipos:

```
y <- data.frame(a=c('x','y','z'), b=1:3, c=c(T,T,F))
is.data.frame(y)
y[,1]
names(y)
y$a
y$a[2]
```

- Toda instalación de R contiene varios conjuntos de datos predefinidos que se utilizan en diversos ejemplos. El conjunto de datos iris es un ejemplo de un dataframe:

```
iris
head(iris)
str(iris)
summary(iris)
```

Ejercicios

- Crea tres vectores de la misma longitud. Por ejemplo: ‘nombre’ que contiene el nombre de un producto, ‘precio’ contiene el precio individual del producto y ‘ventas’ que contiene el número de ventas por producto. Almacena estos tres vectores como una sola variable en forma de dataframe.
- ¿Qué productos tienen un precio superior a 2 euros? ¿Cuántos productos tienes?
- Añade una nueva columna indicando las ganancias obtenidas por cada producto.
- Modifica el precio sumando 10 euros a todos los productos.
- Elimina la columna del número de ventas.
- Ordena las filas por el precio (de menos a mayor).
- Extrae los 2 productos más vendidos.
- Añade un nuevo producto en el que no sabes el precio.

Ejercicios - soluciones

```
nombres <- c("mesa", "silla","armario")
precios <- c(200, 150, 400)
ventas <- c(10, 6, 3)
y <- data.frame(nombre=nombres, precio=precios, venta =ventas)
y_2euros <- y[which(y$nombre >2),]
nrow(y_2euros)
y$ganancias <- y$precio * y$venta
y$precio <- y$precio + 10
y_sinventas <- y[,-3]
y_sinventas
y_ord_precio <- y[with(y, order(precio)), ]
y_ord_precio
y_ord_ventas <- y[with(y, order(venta, decreasing = T)),]
prod_masvendidos <- y_ord_ventas[1:2,"nombre"]
prod_masvendidos
z <- data.frame(nombre=c('lampara'), precio=c(NA), venta=c(NA), ganancias=c(NA))
yz <- rbind(y,z)
```

Estructuras de datos

LISTAS

- Las listas pueden contener cualquier tipo de objeto (incluso otras listas). Los objetos no tienen por qué ser del mismo tipo, ni tener la misma longitud:

  - z <- list(estation = c('Paris', 'Madrid'), months = c("jan", "feb", "mar"), data=c(1,2,3,4,5))
  - is.list(z)
  - names(z)
  - z$months
  - z$months[1]
  - z[[2]]
  - z[[2]][1]

Funciones

- Una función también es un objeto en R
- En R tenemos muchas funciones ya definidas (matemáticas, estadísticas, …), pero también podemos crear nuestras propias funciones.
- Para saber los argumentos de una función usamos args(), ej: args(sample)
- Para saber el código de una función llamamos a la misma sin paréntesis, ej: simple
- Para **crear** una función en R:

```
nombre_funcion <- function (param1, param2, …) {

……….
……….

return(res)

}
```

Ej: Crear una función que devuelva el seno de un ángulo expresado en grados.

```
sine_deg <- function(x) {
  result <- sin((pi / 180) * x)
  return(result)
}
sine_deg(c(45, 90, 270))
```

Funciones

FUNCIONES MATEMÁTICAS

Function	Description
abs(x)	Absolute value
sqrt(x)	Square root
ceiling(x)	First integer value greater than x
floor(x)	First integer value less than x
trunc(x)	Truncated integer value of x
round(x,n)	Rounding of x to n decimal digits
signif(x,n)	Rounding of x to n significant digits
sin(x), cos(x), tan(x)	Sine, cosine and tangent of x
asin(x), acos(x), atan(x)	Sine arc, cosine arc and tangent arc of x
log(x,n)	Logarithm of x in base n
log(x)	Neperian logarithm (base e) of x
log10(x)	Decimal logarithm of x
exp(x)	Exponential function

Funciones

FUNCIONES ESTADÍSTICAS

Function	Description
mean(x)	Average of x
sd(x), var(x)	Standard deviation and variance of x
range(x)	Range of x
min(x), max(x)	Minimum and maximum values of
median(x)	Median (50th percentile) of x
quantile(x,p)	Quantile or probability quantiles p of x
summary(x)	Generates a summary of the object x
sum(x)	Sum of the values of xdiff(x,n
scale(x)	Centring or standardising

Funciones

FUNCIONES DE PROBABILIDADES

Prefix	Description
d	Probability density
p	Distribution (cumulative probability)
q	Quantiles
r	Random number generation

Function	Description
norm	Normal or Gaussian
lnorm	Lognormal
chisq	Chi-square
f	F
logis	Logistic
binom	Binomial
multinom	Multinomial
……	and much more

Funciones

FUNCIONES GRÁFICAS

Function	Description
plot(x)	Creates a graph from the object x
plot(x,y), plot(y~x)	Creates an x-y scatter plot
points(x), lines(x)	Adds a point or line plot to the previous graph
barplot(x)	Creates a barplot from the frequency table x
dotchart(x)	Creates a dot plot from the x table
pie(x)	Creates a pie chart
hist(x)	Creates a frequency histogram of x
boxplot(x)	Creates a boxplot of x

Funciones

FUNCIONES PARA MANIPULAR CADENAS DE CARACTERES

Function	Description
nchar(x)	Number of characters in x
substr(x,a,b)	Extracts the characters of x between positions a and b.
strsplit(x,a)	Splits the string x at pattern a
grep(a,x)	Searches for the pattern a in the string x
sub(a,b,x)	Find the pattern a in the string x and replace it with the string b
paste(..., sep='a')	Concatenates the elements of ..., using the string a as a separator
toupper(x)	Convert the string x to upper case
tolower(x)	Convert the string x to lower case

Funciones

FUNCIONES AUXILIARES

Function	Description
cat(...), c(...)	Concatenate the elements of ... into a vector
length(x)	Length of x
dim(x)	Dimensions of x
seq(i,j,n)	Creates a sequence from i to j, n by n
cut(x,n)	Divides the variable x into a factor with n levels
pretty(x,n)	Divides the variable x into n intervals
t(x)	Transpose the matrix x
tolower(x)	Convert the string x to lower case
apply(x,d,f)	Applies the function f to the object x, along the dimension or dimensions d
tapply(x,i,f)	Applies the function f to the segmented x object from the levels of the factor i
aggregate(x,l,f)	Splits the object x into subsets from a list of elements l, by applying the function f to the a list of elements l

Funciones

FUNCIONES AUXILIARES

Function	Description
factor(x,l,b)	Creates or redefines a factor out of x, with the specified levels l and the labels b
names(x,c), colnames(y,c)	Assigns names to a vector x or to the columns of a matrix or data frame y, and columns of a matrix or data frame y, designated by the character vector c
rbind(x,y)	Joins the x and y data frames by rows
cbind(x,y)	Binds the x and y data frames by columns
sort(x)	Sorts the vector x
order(x)	Vector of indices of the elements of x sorted

Paquetes en R (colecciones de funciones)

- Para **instalar** un paquete de R:
  - install.packages('ggplot2')

- Para **cargarlo** en memoria:
  - library(ggplot2)

Una vez que se ha instalado el paquete de funciones que queremos usar, hay que cargarlo en memoria para poder usarlo en cada session de R.

Estructuras de control de flujo

- if, if else and else
- switch
- for
- while
- repeat
- break
- next
- return

```
my_age <- 14
if (my_age < 18) {
  print("You are a teenager")
} else {
  print("You are an adult")
}
```

```
centre <- function(x, type) {
  switch(type,
    mean = mean(x),
    median = median(x)
  )
}
```

```
i <- 1 # set i to 1
while (i < 10) { # continue as long as i < 10
  print(i) # print i
  i <- i + 1 # increase i by one for each step
}
```

```
i <- 1 # set i to 1
while (i < 10) { # continue as long as i < 10
  i <- i - 1 # ERROR! Here we decrease i by one for each step, hence the condition i < 10 is always met!
  if (abs(i) == 50) { # if i is 50 or -50 we break out of the loop
    print("Iteration stopped!")
    break
  }
}
```

```
for (i in 1:length(beatles)) {
  print(i)
  print(beatles[i])
}
```

```
for (i in seq(nrow(M))) {
  for (j in seq(ncol(M))) {
    val <- M[i, j]
    print(paste("row index:", i, "column index:", j, "value:", val))
  }
}
```

```
s <- seq(1, 25, 1)
for (i in s) {
  if (i %% 2 == 1) { # modulo operator
    next
  } else {
    print(i)
  }
}
```

```
for (i in 1:10) {
  print(i)
}
```

```
for (i in beatles) {
  print(i)
}
```

HELP: Ayuda en R

Diversas maneras:

- help.start()
- http://wiki.r-project.org/
- help('lm')
- ?lm help.search('regression')
- ??regression
- RSiteSearch('regression')

EJEMPLO: ?median o help('median')

Median Value

Description

Compute the sample median.

Usage

```
median(x, na.rm = FALSE, ...)
```

Arguments

x	an object for which a method has been defined, or a numeric vector containing the values whose median is to be computed.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	potentially further arguments for methods; not used in the default method.

Details

This is a generic function for which methods can be written. However, the default method makes use of `is.na`, `sort` and `mean` from package **base** all of which are generic, and so the default method will work for most classes (e.g., "Date") for which a median is a reasonable concept.

Value

The default method returns a length-one object of the same type as `x`, except when `x` is logical or integer of even length, when the result will be double.

If there are no values or if `na.rm = FALSE` and there are `NA` values the result is `NA` of the same type as `x` (or more generally the result of `x[FALSE][NA]`).

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

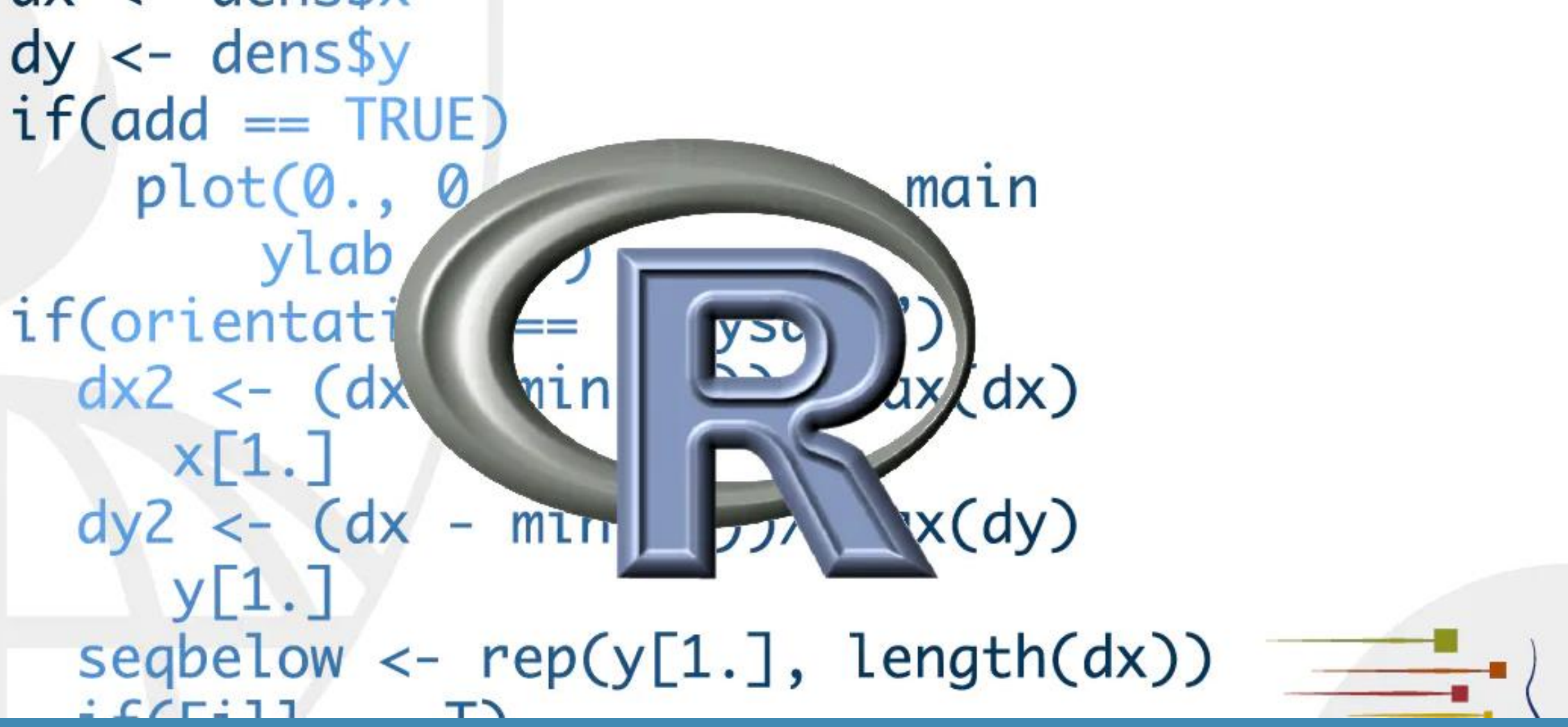
`quantile` for general quantiles.

Examples

```
median(1:4)              # = 2.5 [even number]
median(c(1:3, 100, 1000))  # = 3 [odd, robust]
```

Ejercicios prácticos

- **Ejercicio 1:**
  - **Dataframes**
- **Ejercicio 2:**
  - **Arrays y listas**



¡GRACIAS!

AEMet
Agencia Estatal de Meteorología