

Introducción a R

Oscar Perpiñán Lamigueiro

Universidad Politécnica de Madrid

① **Introducción**

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

① **Introducción**
¿Qué es R?

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

¿Qué es R?

Es un entorno de programación orientado al cálculo, manipulación de datos, y representación gráfica, publicado como software libre con licencia GNU-GPL.

<http://www.R-project.org>

R está muy bien documentado

- ▶ Manuales Oficiales
 - ▶ Introduction to R
 - ▶ R Data Import/Export
 - ▶ R Installation and Administration
 - ▶ Writing R Extensions
 - ▶ R language definition
 - ▶ R Internals
- ▶ Manuales externos

Otros recursos de información

- ▶ [Listas de correo](#) (sin olvidar respetar [estos consejos](#))
 - ▶ Generales: R-announce, R-help, R-devel
 - ▶ Special Interest Group (SIG) mailing lists
- ▶ [R-bloggers](#)
- ▶ [stackoverflow](#)

R es un proyecto colaborativo

- ▶ Una de las grandes riquezas de R es la cantidad de paquetes que amplían sus funcionalidades.
- ▶ La lista completa está en <http://cran.es.r-project.org/web/packages/>.
- ▶ Las CRAN Task Views agrupan por temáticas:
<http://cran.r-project.org/web/views/>

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

Lectura de datos

Importamos datos en formato tabular de un fichero disponible en un [enlace externo](https://raw.githubusercontent.com/oscarperpinan/R/master/data/aranjuez.csv).

```
myURL <- "https://raw.githubusercontent.com/oscarperpinan/R/master/data/aranjuez.
  csv"

## Las columnas están separadas por comas
## La primera fila es la cabecera
datos <- read.table(myURL,
  sep=',',
  header=TRUE)
```

Accedemos al contenido

summary(datos)

```

      X              TempAvg      TempMax      TempMin
Length:2898      Min.   :-5.309      Min.   :-2.362      Min.   :-12.980
Class :character  1st Qu.: 7.692      1st Qu.:14.530      1st Qu.:  1.515
Mode  :character  Median :13.810      Median :21.670      Median :   7.170
                          Mean  :14.405      Mean  :22.531      Mean   :   6.888
                          3rd Qu.:21.615      3rd Qu.:30.875      3rd Qu.: 12.590
                          Max.   :30.680      Max.   :41.910      Max.   : 22.710
                                      NA's   :4

      HumidAvg      HumidMax      WindAvg      WindMax
Min.   : 19.89      Min.   : 35.88      Min.   :0.251      Min.   : 0.000
1st Qu.: 47.04      1st Qu.: 81.60      1st Qu.:0.667      1st Qu.: 3.783
Median : 62.58      Median : 90.90      Median :0.920      Median : 5.027
Mean   : 62.16      Mean   : 87.22      Mean   :1.174      Mean   : 5.208
3rd Qu.: 77.38      3rd Qu.: 94.90      3rd Qu.:1.431      3rd Qu.: 6.537
Max.   :100.00      Max.   :100.00      Max.   :8.260      Max.   :10.000
                                      NA's   :13      NA's   :8      NA's   :128

      Rain      Radiation      ET
Min.   : 0.000      Min.   : 0.277      Min.   :0.000
1st Qu.: 0.000      1st Qu.: 9.370      1st Qu.:1.168
Median : 0.000      Median :16.660      Median :2.758
Mean   : 1.094      Mean   :16.742      Mean   :3.091
3rd Qu.: 0.200      3rd Qu.:24.650      3rd Qu.:4.926
Max.   :49.730      Max.   :32.740      Max.   :8.564
NA's   :4          NA's   :13      NA's   :18
```

Modificamos los datos

```
## Convertimos unidades (MJ -> kWh)
datos$Radiation2 <- datos$Radiation / 3.6
```

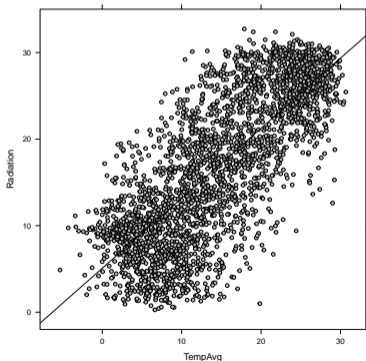
```
## 10 primeras filas de las dos variables
datos[1:10,
      c("Radiation", "Radiation2")]
```

	Radiation	Radiation2
1	5.490	1.525000
2	6.537	1.815833
3	8.810	2.447222
4	9.790	2.719444
5	10.300	2.861111
6	9.940	2.761111
7	7.410	2.058333
8	4.630	1.286111
9	4.995	1.387500
10	8.930	2.480556

Representamos gráficamente los datos

```
library(lattice)
```

```
xyplot(Radiation ~ TempAvg, data = datos,  
       type = c("p", "r"),  
       pch = 21, col = 'black', fill = 'gray')
```



① Introducción

② Ejemplo

③ **Objetos en R**

④ Indexado

⑤ Funciones

⑥ Bucles

Objetos en R

- ▶ Existen varios objetos en R:
 - ▶ Vectores
 - ▶ Listas
 - ▶ Funciones
 - ▶ ...
- ▶ A partir de estos objetos se definen varias clases:
 - ▶ `matrix`
 - ▶ `data.frame`
 - ▶ `factor`
 - ▶ `Date`, `POSIXct`
 - ▶ ...

① Introducción

② Ejemplo

③ **Objetos en R**

 Vectores

 Matrices

 Listas

 Data.frame

④ Indexado

⑤ Funciones

⑥ Bucles

Primeros pasos

```
x <- 1:5
```

```
x
```

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

```
length(x)
```

```
[1] 5
```

```
class(x)
```

```
[1] "integer"
```


Generar vectores con seq

```
x1 <- seq(1, 100, by=2)
```

```
x1
```

```
[1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49  
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
seq(1, 100, length=10)
```

```
[1]  1 12 23 34 45 56 67 78 89 100
```

Unir vectores con c

```
x <- c(1, 2, 3)
```

```
x
```

```
[1] 1 2 3
```

```
x <- seq(1, 100, length=10)
```

```
y <- seq(2, 100, length=50)
```

```
z <- c(x, y)
```

```
z
```

```
[1] 1 12 23 34 45 56 67 78 89 100 2 4 6 8 10 12 14 16 18  
[20] 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56  
[39] 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94  
[58] 96 98 100
```

Operaciones sencillas con vectores

```
x <- 1:5  
x + 1
```

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

```
x^2
```

```
[1] 1 4 9 16 25
```

```
y <- 1:10  
x + y
```

```
[1] 2 4 6 8 10 7 9 11 13 15
```

```
x * y
```

```
[1] 1 4 9 16 25 6 14 24 36 50
```

```
x^2 + y^3
```

```
[1] 2 12 36 80 150 217 347 521 745 1025
```

Ejercicio

Dibuja una circunferencia

Sabiendo que la función `plot(x, y)` dibuja el vector y frente al vector x , ¿qué código es necesario para dibujar una circunferencia de un radio determinado?

① Introducción

② Ejemplo

③ **Objetos en R**

Vectores

Matrices

Listas

Data.frame

④ Indexado

⑤ Funciones

⑥ Bucles

Construir una matriz

```
z <- 1:12
M <- matrix(z, nrow=3)
M
```

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

```
class(M)
```

```
[1] "matrix" "array"
```

```
dim(M)
```

```
[1] 3 4
```

```
summary(M)
```

V1	V2	V3	V4
Min. :1.0	Min. :4.0	Min. :7.0	Min. :10.0
1st Qu.:1.5	1st Qu.:4.5	1st Qu.:7.5	1st Qu.:10.5
Median :2.0	Median :5.0	Median :8.0	Median :11.0
Mean :2.0	Mean :5.0	Mean :8.0	Mean :11.0
3rd Qu.:2.5	3rd Qu.:5.5	3rd Qu.:8.5	3rd Qu.:11.5
Max. :3.0	Max. :6.0	Max. :9.0	Max. :12.0

Matrices a partir de vectores: rbind y cbind

```
z <- y <- x <- 1:10
```

```
M <- cbind(x, y, z)
```

```
M
```

```
      x y z
[1,] 1 1 1
[2,] 2 2 2
[3,] 3 3 3
[4,] 4 4 4
[5,] 5 5 5
[6,] 6 6 6
[7,] 7 7 7
[8,] 8 8 8
[9,] 9 9 9
[10,] 10 10 10
```

```
M <- rbind(x, y, z)
```

```
M
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
x      1    2    3    4    5    6    7    8    9    10
y      1    2    3    4    5    6    7    8    9    10
z      1    2    3    4    5    6    7    8    9    10
```

Álgebra matricial

`t()` Transpuesta de una matriz

`*` Multiplicación elemento a elemento

`%*%` Multiplicación de matrices

`solve(A)` Inversa de una matriz (cuadrada)

...

① Introducción

② Ejemplo

③ **Objetos en R**

Vectores

Matrices

Listas

Data.frame

④ Indexado

⑤ Funciones

⑥ Bucles

Para crear una lista usamos la función `list`

```
lista <- list(a=c(1,3,5),  
             b=c('l', 'p', 'r', 's'),  
             c=3)
```

```
lista
```

```
$a  
[1] 1 3 5
```

```
$b  
[1] "l" "p" "r" "s"
```

```
$c  
[1] 3
```

```
class(lista)
```

```
[1] "list"
```

```
length(lista)
```

```
[1] 3
```

① Introducción

② Ejemplo

③ **Objetos en R**

Vectores

Matrices

Listas

Data.frame

④ Indexado

⑤ Funciones

⑥ Bucles

Para crear un data.frame...

```
df <- data.frame(x = 1:5,  
                 y = rnorm(10),  
                 z = 0)
```

```
df
```

```
   x      y z  
1  1  0.65470799 0  
2  2 -1.97985660 0  
3  3  0.13162836 0  
4  4 -0.09737236 0  
5  5 -0.03519971 0  
6  1 -0.40980681 0  
7  2  0.22341575 0  
8  3  1.01778816 0  
9  4  0.73108975 0  
10 5  1.32046416 0
```

```
length(df)
```

```
[1] 3
```

```
dim(df)
```

```
[1] 10  3
```

A partir de ficheros

```
dats <- read.table('data/aranjuez.csv',  
                  sep=',',  
                  header=TRUE)
```

```
head(dats)
```

	X	TempAvg	TempMax	TempMin	HumidAvg	HumidMax	WindAvg	WindMax	Rain
1	2004-01-01	4.044	10.71	-1.969	88.3	95.9	0.746	3.528	0
2	2004-01-02	5.777	11.52	1.247	83.3	98.5	1.078	6.880	0
3	2004-01-03	5.850	13.32	0.377	75.0	94.4	0.979	6.576	0
4	2004-01-04	4.408	15.59	-2.576	82.0	97.0	0.633	3.704	0
5	2004-01-05	3.081	14.58	-2.974	83.2	97.0	0.389	2.244	0
6	2004-01-06	2.304	11.83	-3.379	84.5	96.5	0.436	2.136	0

	Radiation	ET
1	5.490	0.5352688
2	6.537	0.7710499
3	8.810	0.8361229
4	9.790	0.6861381
5	10.300	0.5152422
6	9.940	0.4886631

Atención: usa setwd para configurar ruta

A partir de ficheros remotos

```
remoto <- read.table('https://raw.githubusercontent.com/oscarperpinan/R/master/
  data/aranjuez.csv',
  sep=',',
  header=TRUE)
```

```
head(remoto)
```

	X	TempAvg	TempMax	TempMin	HumidAvg	HumidMax	WindAvg	WindMax	Rain
1	2004-01-01	4.044	10.71	-1.969	88.3	95.9	0.746	3.528	0
2	2004-01-02	5.777	11.52	1.247	83.3	98.5	1.078	6.880	0
3	2004-01-03	5.850	13.32	0.377	75.0	94.4	0.979	6.576	0
4	2004-01-04	4.408	15.59	-2.576	82.0	97.0	0.633	3.704	0
5	2004-01-05	3.081	14.58	-2.974	83.2	97.0	0.389	2.244	0
6	2004-01-06	2.304	11.83	-3.379	84.5	96.5	0.436	2.136	0

	Radiation	ET
1	5.490	0.5352688
2	6.537	0.7710499
3	8.810	0.8361229
4	9.790	0.6861381
5	10.300	0.5152422
6	9.940	0.4886631

```
identical(dats, remoto)
```

```
[1] TRUE
```

Ejercicio

Dibuja una circunferencia

¿Qué código hay que emplear para dibujar una circunferencia de forma que todos los vectores implicados sean columnas de un `data.frame`?

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

Condiciones lógicas

Vectores

Matrices

Listas

Data Frame

⑤ Funciones

⑥ Bucles

Condiciones simples

```
x <- seq(-1, 1, .1)
```

```
x
```

```
[1] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  
[16]  0.5  0.6  0.7  0.8  0.9  1.0
```

```
x < 0
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE  
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
x >= 0
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE  
[13] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
x == 0
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
x != 0
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE  
[13] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Condiciones múltiples

```
cond <- (x > 0) & (x < .5)
cond
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
[13] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
cond <- (x >= .5) | (x <= -.5)
cond
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
```

Con las condiciones se pueden hacer operaciones

```
sum(cond)
```

```
[1] 12
```

```
sum(!cond)
```

```
[1] 9
```

```
as.numeric(cond)
```

```
[1] 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1
```

① Introducción

② Ejemplo

③ Objetos en R

④ **Indexado**

Condiciones lógicas

Vectores

Matrices

Listas

Data Frame

⑤ Funciones

⑥ Bucles

Indexado numérico

```
x <- seq(1, 100, 2)
```

```
x
```

```
[1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49  
[26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
x[1:5]
```

```
[1] 1 3 5 7 9
```

```
x[10:5]
```

```
[1] 19 17 15 13 11 9
```

Indexado con condiciones lógicas

```
x[x != 9]
```

```
[1] 1 3 5 7 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51  
[26] 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
x[x > 20]
```

```
[1] 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69  
[26] 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
x[x %in% seq(0, 10, .5)]
```

```
[1] 1 3 5 7 9
```

Indexado con condiciones múltiples

```
z <- seq(-10, 10, by = .5)
```

```
z
```

```
[1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 -5.0 -4.5  
[13] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5 0.0 0.5 1.0 1.5  
[25] 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5  
[37] 8.0 8.5 9.0 9.5 10.0
```

```
z[z < -5 | z > 5]
```

```
[1] -10.0 -9.5 -9.0 -8.5 -8.0 -7.5 -7.0 -6.5 -6.0 -5.5 5.5 6.0  
[13] 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
```

```
cond <- (z >= 0 & z <= 5)
```

```
cond
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE  
[25] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE  
[37] FALSE FALSE FALSE FALSE FALSE
```

```
z[cond]
```

```
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```


① Introducción

② Ejemplo

③ Objetos en R

④ **Indexado**

Condiciones lógicas

Vectores

Matrices

Listas

Data Frame

⑤ Funciones

⑥ Bucles

Indexado de matrices

```
M[1:2, ]
```

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
x    1    2    3    4    5    6    7    8    9   10  
y    1    2    3    4    5    6    7    8    9   10
```

```
M[1:2, 2:3]
```

```
  [,1] [,2]  
x     2    3  
y     2    3
```

```
M[1, c(1, 4)]
```

```
[1] 1 4
```

Indexado de matrices

```
M[-1,]
```

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
y    1    2    3    4    5    6    7    8    9   10  
z    1    2    3    4    5    6    7    8    9   10
```

```
M[-c(1, 2),]
```

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

① Introducción

② Ejemplo

③ Objetos en R

④ **Indexado**

Condiciones lógicas

Vectores

Matrices

Listas

Data Frame

⑤ Funciones

⑥ Bucles

Podemos acceder a los elementos...

- ▶ Por su nombre

```
lista$a
```

```
[1] 1 3 5
```

- ▶ o por su índice

```
lista[1]
```

```
$a  
[1] 1 3 5
```

```
lista[[1]]
```

```
[1] 1 3 5
```

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

Condiciones lógicas

Vectores

Matrices

Listas

Data Frame

⑤ Funciones

⑥ Bucles

Podemos acceder a los elementos

```
df <- data.frame(x = 1:5,  
                 y = rnorm(10),  
                 z = 0)
```

- ▶ Por su nombre (como una lista)

```
df$x
```

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

- ▶ Por su índice (como una matriz)

```
df[1,]
```

```
  x      y z  
1 1 1.282005 0
```

```
df[,1]
```

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

Indexado lógico

- ▶ Hay que explicitar dos veces el `data.frame`:

```
df[df$y > 0,]
```

```
  x      y z
1 1 1.2820046 0
2 2 0.1489810 0
4 4 0.7802353 0
5 5 1.5233139 0
6 1 2.1542386 0
7 2 0.9532016 0
```

- ▶ La función `subset` simplifica el código:

```
subset(df, y > 0)
```

```
  x      y z
1 1 1.2820046 0
2 2 0.1489810 0
4 4 0.7802353 0
5 5 1.5233139 0
6 1 2.1542386 0
7 2 0.9532016 0
```


Uso de `with`

- ▶ Problema: el código con varias variables puede ser ilegible

```
df$x^2 + df$y^2
```

- ▶ La función `with` permite acceder a varias variables con una única llamada:

```
with(df, x^2 + y^2)
```

```
[1]  2.643536  4.022195 11.418728 16.608767 27.320485  5.640744  4.908593  
[8] 10.167511 17.113295 25.021335
```

```
with(df, x[y > 0])
```

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

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ **Funciones**

Definición de funciones

Funciones predefinidas

⑥ Bucles

Componentes de una función

- ▶ Una función se define con `function`

```
name <- function(arg_1, arg_2, ...) expression
```

- ▶ Está compuesta por:
 - ▶ Nombre de la función (`name`)
 - ▶ Argumentos (`arg_1, arg_2, ...`)
 - ▶ Cuerpo (`expression`): emplea los argumentos para generar un resultado

Argumentos: nombre y orden

Una función identifica sus argumentos por su nombre y por su orden (sin nombre)

```
eleva <- function(x, p)
{
  x ^ p
}
```

```
eleva(x = 1:10, p = 2)
```

```
[1] 1 4 9 16 25 36 49 64 81 100
```

```
eleva(1:10, p = 2)
```

```
[1] 1 4 9 16 25 36 49 64 81 100
```

```
eleva(p = 2, x = 1:10)
```

```
[1] 1 4 9 16 25 36 49 64 81 100
```

Argumentos: valores por defecto

- ▶ Se puede asignar un valor por defecto a los argumentos

```
eleva <- function(x, p = 2)
{
  x ^ p
}
```

```
eleva(1:10)
```

```
[1] 1 4 9 16 25 36 49 64 81 100
```

```
eleva(1:10, 2)
```

```
[1] 1 4 9 16 25 36 49 64 81 100
```

Argumentos sin nombre: ...

```
pwrSum <- function(x, p, ...)  
{  
  sum(x ^ p, ...)  
}
```

```
x <- 1:10  
pwrSum(x, 2)
```

[1] 385

```
x <- c(1:5, NA, 6:9, NA, 10)  
pwrSum(x, 2)
```

[1] NA

```
pwrSum(x, 2, na.rm=TRUE)
```

[1] 385

Podemos construir a partir de funciones

```
foo <- function(x, ...){  
  mx <- mean(x, ...)  
  medx <- median(x, ...)  
  sdx <- sd(x, ...)  
  c(mx, medx, sdx)  
}
```

```
foo(1:10)
```

```
[1] 5.50000 5.50000 3.02765
```

```
foo(rnorm(1e5))
```

```
[1] 0.004775189 0.011236544 0.997530539
```


Ejercicio

Dibuja una circunferencia

Define una función de dos argumentos, `theta` (vector de ángulos) y `r` (radio), que entregue un `data.frame` de dos columnas, `x` e `y`, con las coordenadas del arco de circunferencia que corresponde a los argumentos de la función, y emplea esta función para dibujar una circunferencia completa.

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ **Funciones**

Definición de funciones

Funciones predefinidas

⑥ Bucles

Funciones en paquetes

- ▶ R proporciona un amplio conjunto de funciones predefinidas agrupadas en paquetes
 - ▶ Algunos paquetes vienen instalados y se cargan al empezar (*base*):

```
sessionInfo()
```

- ▶ Otros vienen instalados pero hay que cargarlos (*recommended*):

```
library(lattice)
```

```
packageDescription('lattice')
```

- ▶ Otros hay que instalarlos y después cargarlos (*contributed*):

```
install.packages('data.table')
```

```
library('data.table')
```

```
packageDescription('data.table')
```

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

Matrices

Listas / `data.frame`

Bucles `for`

Condiciones con `if`, `else` e `ifelse`

La función apply

```
apply(M, 1, sum)
```

```
  x y z  
55 55 55
```

```
rowSums(M)
```

```
  x y z  
55 55 55
```

```
apply(M, 2, mean)
```

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

```
colMeans(M)
```

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

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

Matrices

Listas / `data.frame`

Bucles `for`

Condiciones con `if`, `else` e `ifelse`

lapply y sapply

```
lista <- list(x = 1:10,  
             y = seq(0, 10, 2),  
             z = rnorm(30))  
lapply(lista, sum)
```

```
$x  
[1] 55
```

```
$y  
[1] 30
```

```
$z  
[1] -8.816296
```

```
sapply(lista, sum)
```

```
      x      y      z  
55.000000 30.000000 -8.816296
```


Ejercicio

- ▶ Calcula la media de cada una de las columnas de remoto.
- ▶ Calcula la media, mediana y desviación estándar de cada una de las columnas de remoto.
- ▶ Calcula la media de los valores positivos de cada una de las columnas de remoto.

```
remoto <- read.csv('https://raw.githubusercontent.com/oscarperpinan/R/master/data/
  aranjuez.csv')
```

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

Matrices

Listas / `data.frame`

Bucles for

Condiciones con `if`, `else` e `ifelse`

for

- ▶ En R suele usarse más la familia de funciones `*apply` con funciones vectorizadas.
- ▶ No obstante, `for` puede tener su utilidad:

```
for(n in c(2,5,10,20,50)) {  
  x <- rnorm(n)  
  cat(n,":", sum(x^2),"\n")  
}
```

```
2 : 5.167868  
5 : 0.6066694  
10 : 4.723925  
20 : 11.14655  
50 : 53.28468
```

① Introducción

② Ejemplo

③ Objetos en R

④ Indexado

⑤ Funciones

⑥ Bucles

Matrices

Listas / `data.frame`

Bucles `for`

Condiciones con `if`, `else` e `ifelse`

if

- ▶ En R suele usarse más el indexado lógico (vectorizado).
- ▶ ¿Cuál es el equivalente a este bucle for-if?

```
x <- rnorm(10)
x2 <- numeric(length(x))
for (i in seq_along(x2)){
  if (x[i]<0) x2[i] <- 0 else x2[i] <- 1
}
cbind(x, x2)
```

```
      x x2
[1,] 0.5230053 1
[2,] -0.2133108 0
[3,] -1.8577459 0
[4,] 0.5807593 1
[5,] -2.1224405 0
[6,] -0.8255077 0
[7,] -0.3065412 0
[8,] -0.2994609 0
[9,] -0.3553300 0
[10,] 0.3958203 1
```

ifelse

```
x <- rnorm(10)
```

```
x
```

```
[1] -0.2956893 -0.1427568  0.8798947  1.3953489 -0.8577325  0.2439465  
[7]  0.3572955  0.1007339  1.8887487 -1.0250934
```

```
ifelse(x>0, 1, 0)
```

```
[1] 0 0 1 1 0 1 1 1 1 0
```