

# Análisis espaciales y multivariantes en R aplicados a estudios de biodiversidad

Introducción a R

Diego Nieto Lugilde. Profesor Titular de la Universidad de Córdoba  
(España)



UNIVERSIDAD  
DE  
CÓRDOBA



Co-funded by  
the European Union



## Relación amor-odio

- Los ordenadores pueden volver muy complejas las cosas simples, pero también pueden simplificar mucho las cosas complejas
  - Esto es cierto cuanto más aprendes de computación (e.g. Excel vs. R)
- Prepararos para invertir mucho tiempo en cosas estúpidas, pero tener presente que ahorraréis mucho tiempo en cosas complicadas/rutinarias

```

$total = {
  'totals' => $total,
  'taxes' => $taxes,
  'total' => &$total
};

sold_taxes = $taxes;
$ipa_tax = array();
$sort_order = array();

$results = $this->model_extension->getExtensions('total');

foreach ($results as $key => $value) {
  if (isset($value['code'])) {
    $code = $value['code'];
  } else {
    $code = $value['key'];
  }
  $sort_order[$key] = $this->config->get($code . '_sort_order');
}

array_multisort($sort_order, SORT_ASC, $results);

foreach ($results as $result) {
  if (isset($result['code'])) {
    $code = $result['code'];
  } else {
    $code = $result['key'];
  }
  if ($this->config->get($code . '_status')) {
    $this->load->model('extension/total/' . $code);

    // We have to put the totals in an array so that they pass
    // by reference.
    $this->{'model_extension_total_' . $code}->getTotal($
      total_data);

    if (!empty($totals[count($totals) - 1]) && !isset($totals[
      count($totals) - 1]['code'])) {
      $totals[count($totals) - 1]['code'] = $code;
    }

    $tax_difference = 0;

    foreach ($taxes as $tax_id => $value) {
      if (isset($sold_taxes[$tax_id])) {

```

```

354 Carousel.prototype.getItemForDirection = function (direction) {
355   this.$items = item.parent().children();
356   return this.$items.index(item || this.$activeItem);
357 }
358
359 Carousel.prototype.getItemForDirection = function (direction) {
360   var delta = direction == 'prev' ? -1 : 1;
361   var activeIndex = this.getItemIndex(this.$activeItem);
362   var itemIndex = (activeIndex + delta) % this.$items.length;
363   return this.$items.eq(itemIndex);
364 }
365
366 Carousel.prototype.to = function (pos) {
367   var that = this;
368   var activeIndex = this.getItemIndex(this.$activeItem);
369   if (pos > (this.$items.length - 1)) pos = (this.$items.length - 1);
370   if (this.sliding) return this.$element.trigger('transition');
371   if (activeIndex == pos) return this.pause().cycle().unblockUI();
372   return this.slide(pos > activeIndex ? 'next' : 'prev', true);
373 }
374
375 Carousel.prototype.pause = function (e) {
376   e || (this.paused = true);
377
378   if (this.$element.find('.next, .prev').length > 0)
379     this.$element.trigger($.support.transition.end);
380   this.cycle(true);
381 }
382
383 this.interval = clearInterval(this.interval);
384
385 return this;
386
387
388
389
390

```

# ¿Qué es R?

- R es un lenguaje de programación estadística
- Se basa en un sistema de comandos
- Scripts (guiones)

# ¿Por qué usar R?

## Scripts

- Documentado
- Reproducible

## Flexible

Libre (i.e. gratis y libre)

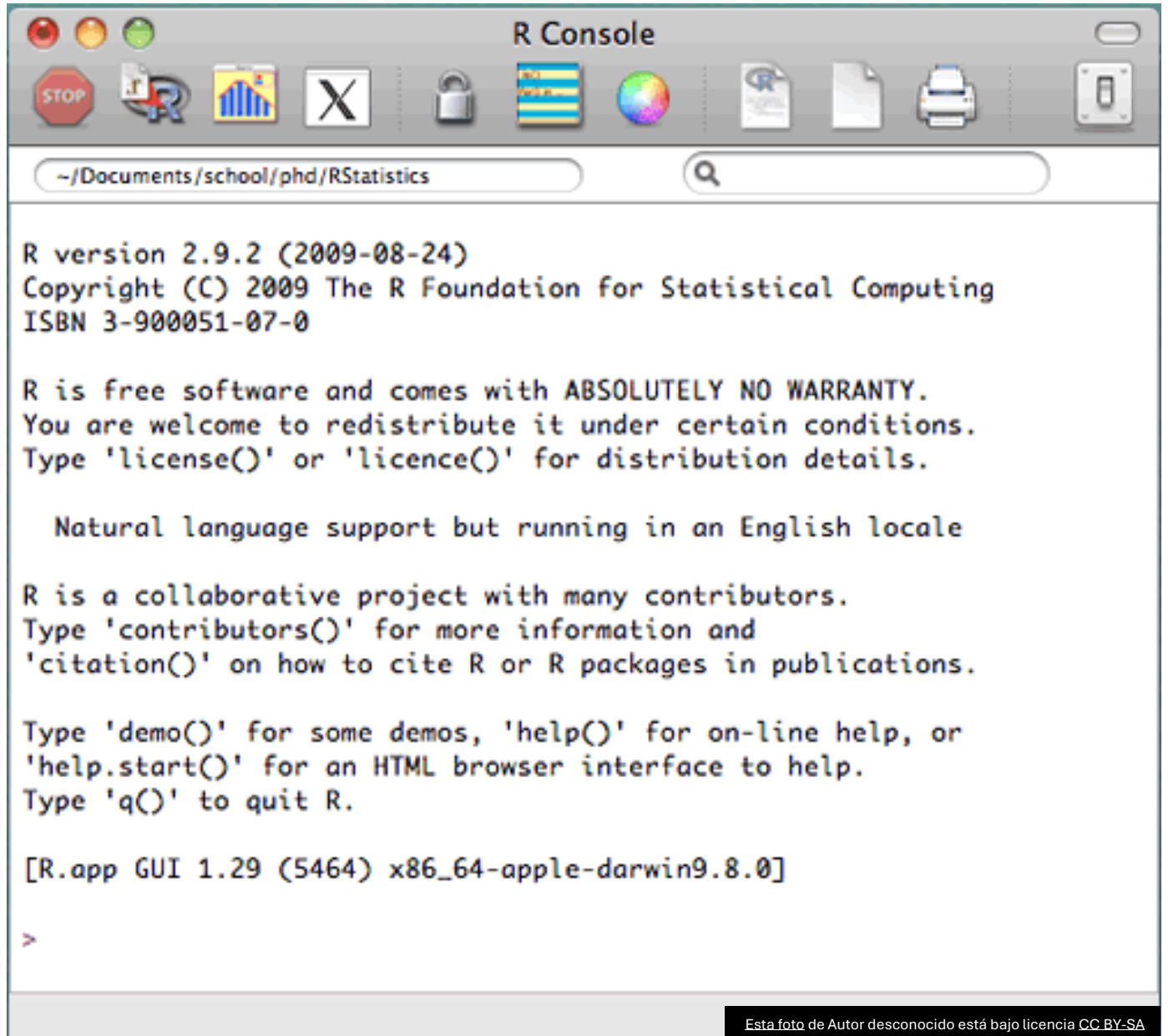
Muchas extensiones disponibles

Enorme soporte de la comunidad

- Foros
- Tutoriales
- ...

¿Qué aspect tiene R?

No es muy bonito

A screenshot of the R Console window on a Mac. The window title is "R Console". The address bar shows the path "~/Documents/school/phd/RStatistics". The main content area displays the R version 2.9.2 startup screen, including copyright information, a disclaimer, and instructions for using the console. The text is as follows:

```
R version 2.9.2 (2009-08-24)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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.

Natural language support but running in an English locale

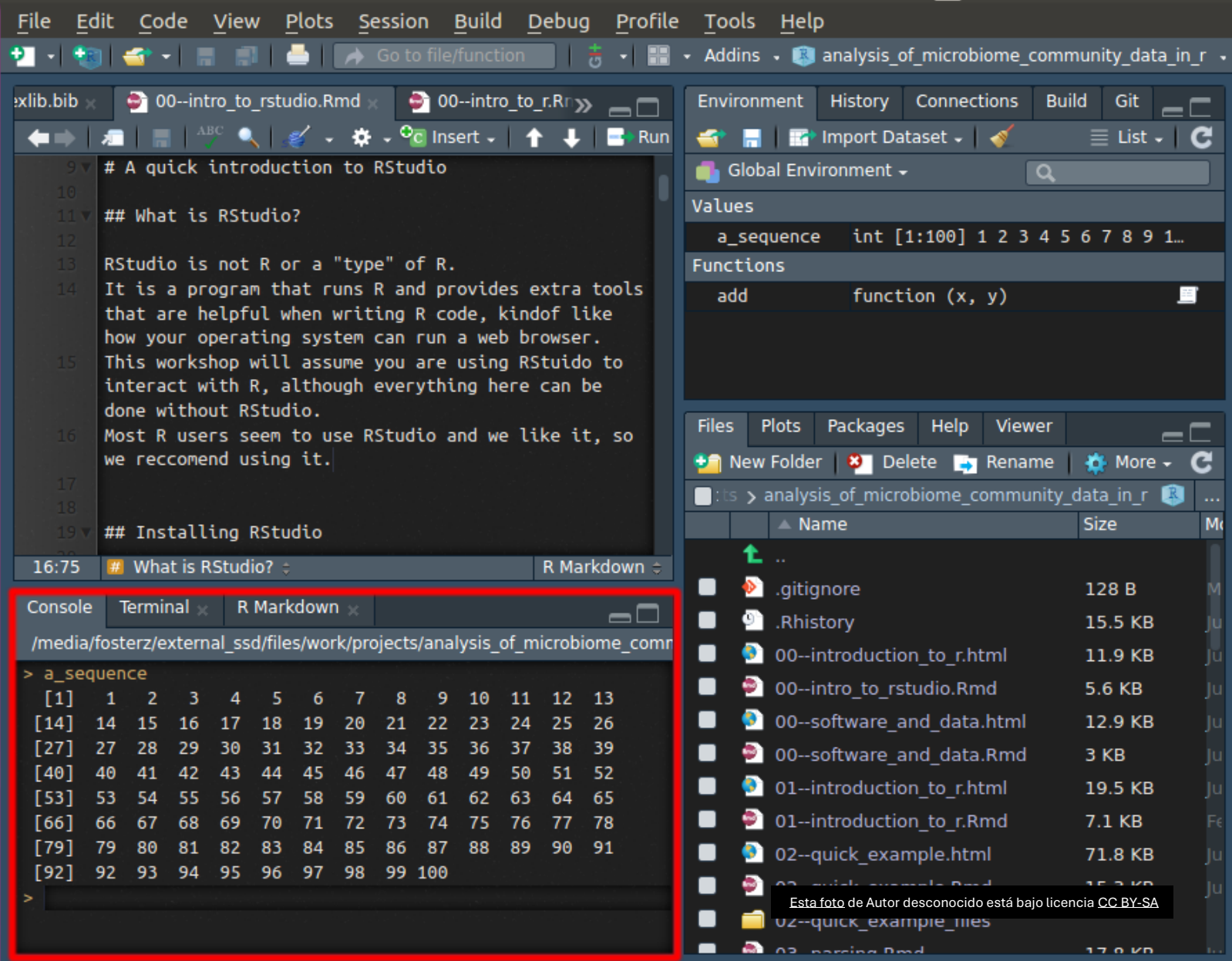
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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.

[R.app GUI 1.29 (5464) x86_64-apple-darwin9.8.0]
>
```

# ¿Qué aspecto tiene R?

- Hay alternativas más agradables/prácticas
  - RStudio y otros
- Resultado sintáctico ¡muy útil!



The screenshot displays the RStudio environment. The main editor shows R Markdown code for an introduction to RStudio. The console window at the bottom, highlighted with a red border, shows the execution of the `a_sequence` variable, resulting in a 10x10 grid of numbers from 1 to 100. The file explorer on the right shows a project directory with various files and folders.

```
# A quick introduction to RStudio
## What is RStudio?

RStudio is not R or a "type" of R.
It is a program that runs R and provides extra tools
that are helpful when writing R code, kindof like
how your operating system can run a web browser.
This workshop will assume you are using RStudio to
interact with R, although everything here can be
done without RStudio.
Most R users seem to use RStudio and we like it, so
we recommend using it.

## Installing RStudio
```

```
> a_sequence
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13
[14] 14 15 16 17 18 19 20 21 22 23 24 25 26
[27] 27 28 29 30 31 32 33 34 35 36 37 38 39
[40] 40 41 42 43 44 45 46 47 48 49 50 51 52
[53] 53 54 55 56 57 58 59 60 61 62 63 64 65
[66] 66 67 68 69 70 71 72 73 74 75 76 77 78
[79] 79 80 81 82 83 84 85 86 87 88 89 90 91
[92] 92 93 94 95 96 97 98 99 100
```

Name	Size
..	
.gitignore	128 B
.Rhistory	15.5 KB
00--introduction_to_r.html	11.9 KB
00--intro_to_rstudio.Rmd	5.6 KB
00--software_and_data.html	12.9 KB
00--software_and_data.Rmd	3 KB
01--introduction_to_r.html	19.5 KB
01--introduction_to_r.Rmd	7.1 KB
02--quick_example.html	71.8 KB
02--quick_example.Rmd	15.3 KB
02--quick_example_files	
03--parsing.Rmd	17.8 KB

Esta foto de Autor desconocido está bajo licencia CC-BY-SA

# El plan

- La estructura de R
- Funciones, objetos y programación básica
- Entorno (sesión de R) y directorio de trabajo
- Trabajar con archivos (scripts y datos)
- Instalación de paquetes
- CRAN
- GitHu
- Graficado

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Untitled1\* x

Source on Save Run Source

1

Environment History Connections Tutorial

Global Environment

Environment is empty

1:1 (Top Level) R Script

Console Terminal Background Jobs

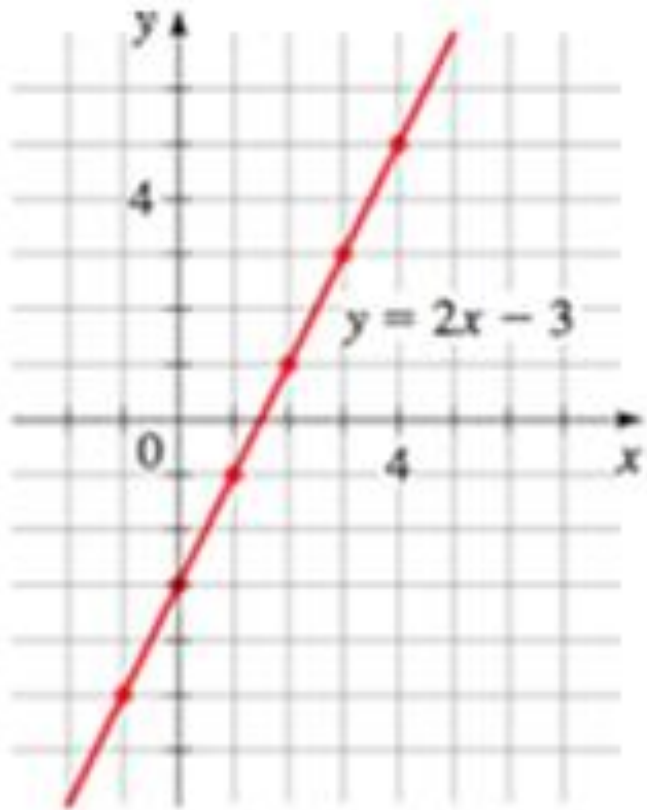
```
R 3.6.3 · C:/Users/Diego Nieto Lugilde/OneDrive - Universidad de Córdoba/Docencia/Master-UCO/Análisis_de_datos_en_R/Leccion
[1] 7 2 NA NA
>
> a <- 1:5
> a[a]
[1] 1 2 3 4 5
> a <- 1:10
> a[a]
[1] 1 2 3 4 5 6 7 8 9 10
> clear()
Error in clear() : could not find function "clear"
> clean()
Error in clean() : could not find function "clean"
> ??clean_d11
> ??clear
> |
```

Files Plots Packages Help Viewer Presentation

R: Search Results Find in Topic

- [transformeR::temporalCycleGrid](#) Temporal cycle calculation
- [utils::globalVariables](#) Declarations Used in Checking a Package
- [withr::defer](#) Defer Evaluation of an Expression
- [XML::startElement.SAX](#) Generic Methods for SAX callbacks
- [XML::catalogLoad](#) Manipulate XML catalog contents
- [XML::free](#) Release the specified object and clean up its memory usage
- [XML::xmlCleanNamespaces](#) Remove redundant namespaces on an XML document
- [XML::xmlParseDoc](#) Parse an XML document with options controlling the parser.





$x$	$y = 2x - 3$	$(x, y)$
-1	-5	$(-1, -5)$
0	-3	$(0, -3)$
1	-1	$(1, -1)$
2	1	$(2, 1)$
3	3	$(3, 3)$
4	5	$(4, 5)$

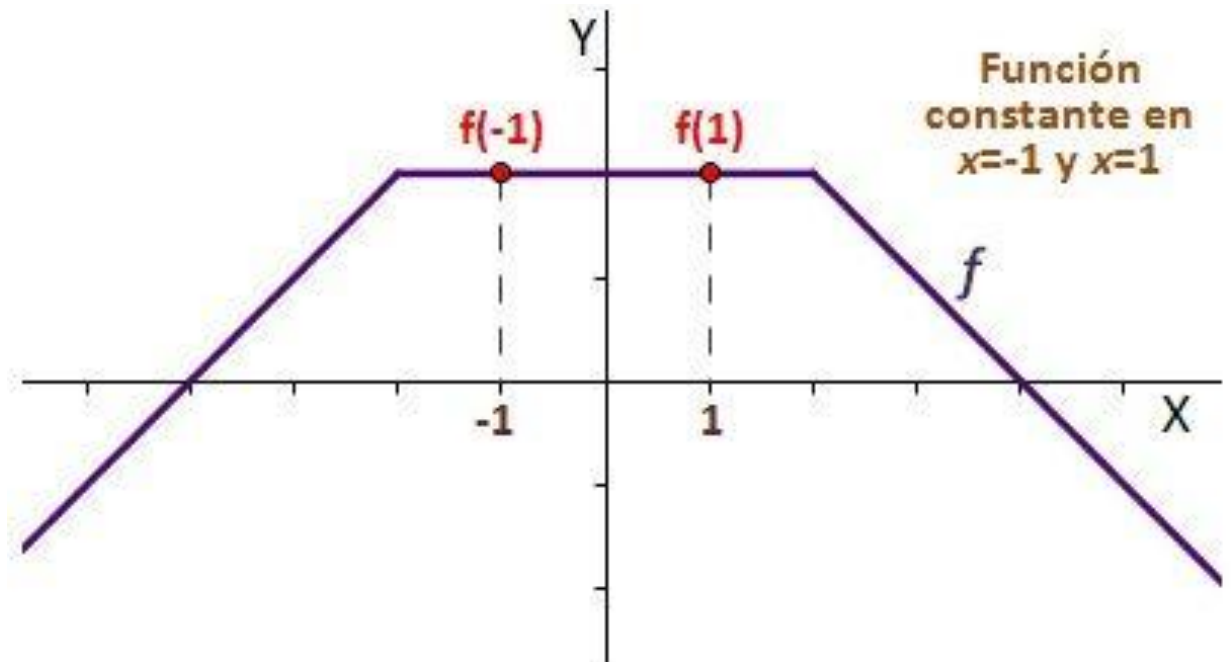
La estructura de  $\mathbb{R}$

Objetos:  $x$

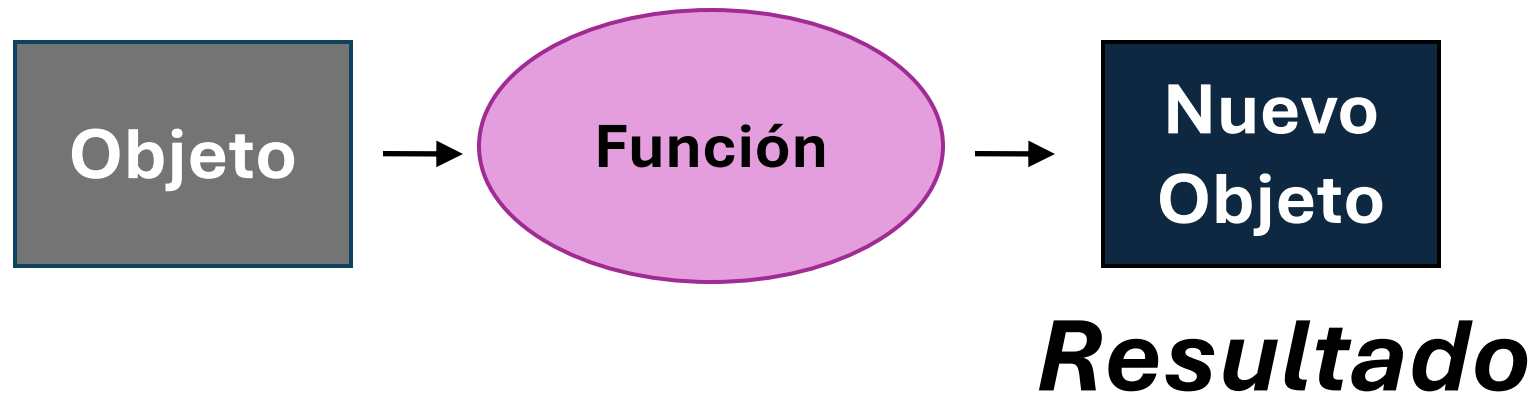
Funciones:  $f()$

# Funciones

Función	Inversa
$\sin(x)$	$\arcsin(x)$
$e^x$	$\ln(x)$
$\sqrt[3]{x}$	$x^3$
$1/x$	$1/x$
$\tan(x)$	$\text{atan}(x)$
$x$	$x$
$1/\sqrt{x}$	$1/x^2$



# La estructura de R



# **Call:** usar una función con un conjunto de argumentos/objetos

- función( *argumento 1* )
  - Función: sqrt (raíz cuadrada)
  - Objeto (argumento 1): 16
  - Nuevo objeto (resultado): 4

```
sqrt (16)  
[1] 4
```

El resultado de la función  
no se guarda, solo se  
imprime en la pantalla

# ***Assign***: Asignación de resultados a objetos

- Es una función especial para crear/modificar un objeto
  - Valor específico
  - El resultado de una llamada a una función

```
x <- 2  
  
x <- sqrt(16)  
x  
[1] 4
```

El resultado de la función se guarda en un nuevo objeto, "x"

# Hay varias maneras de realizar asignaciones

```
a = 10
```

```
a <- 10
```

```
10 -> a
```

```
a
```

```
[1] 10
```

# ¿Qué es un objeto?

- Entidad con información
- Con un nombre sin espacios
- Se puede utilizar muchas veces

# ¿Qué contiene un objeto?

Los caracteres se  
entrecorillan para  
distinguirlos de los  
nombres de los  
objetos

- Números
  - 0, 0.2, Inf
- Caracteres (texto)
  - Texto libre
  - e.g., “Bromus diandrus”, “Bromus carinatus”, or “Bison bison”
- Valores lógicos
  - TRUE (T), FALSE (F)
- Factores
  - Categorías (e.g., “Masculino” o “Femenino”)
- No data
  - NA

```
numeric_object <- 0.5  
character_object <- "Bromus diandrus"  
logical_object <- TRUE  
factor_object <- factor("Male")  
na_object <- NA
```



# ¿Cómo averiguo el tipo de información de un objeto?

- Pertenece a una clase
- Cada clase tiene una estructura específica
- `class()` es una función que te dice qué tipo de objeto es el argumento

```
class(numeric_object)
class(logical_object)
class(factor_object)
class(character_object)
class("x")
class(x)
```

# Trabajar con objetos

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

```
a
```

```
[1] 11
```

```
b <- a * a
```

```
b
```

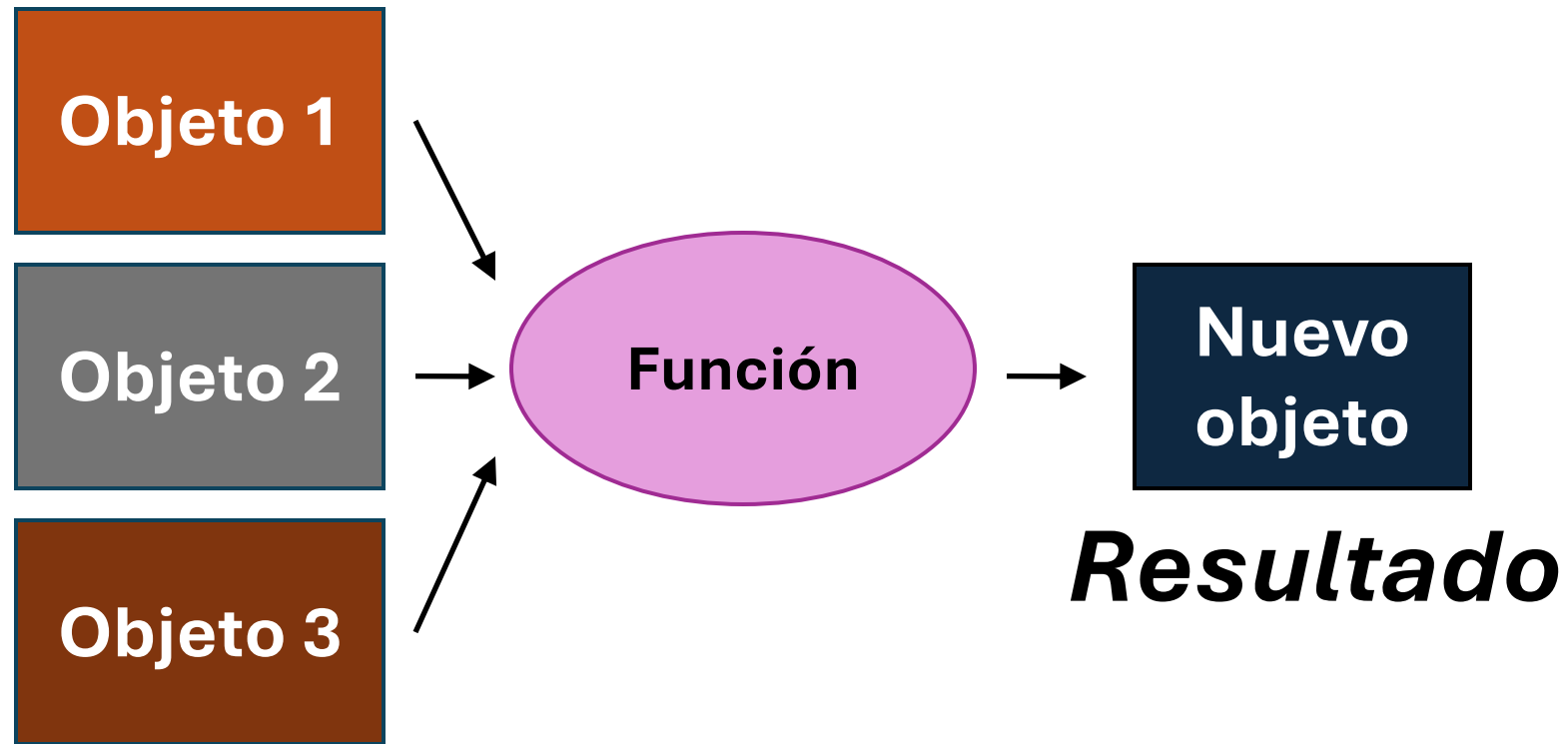
```
[1] 121
```

```
x <- sqrt(b)
```

```
x
```

```
[1] 11
```

# Algunas funciones aceptan varios datos de entrada



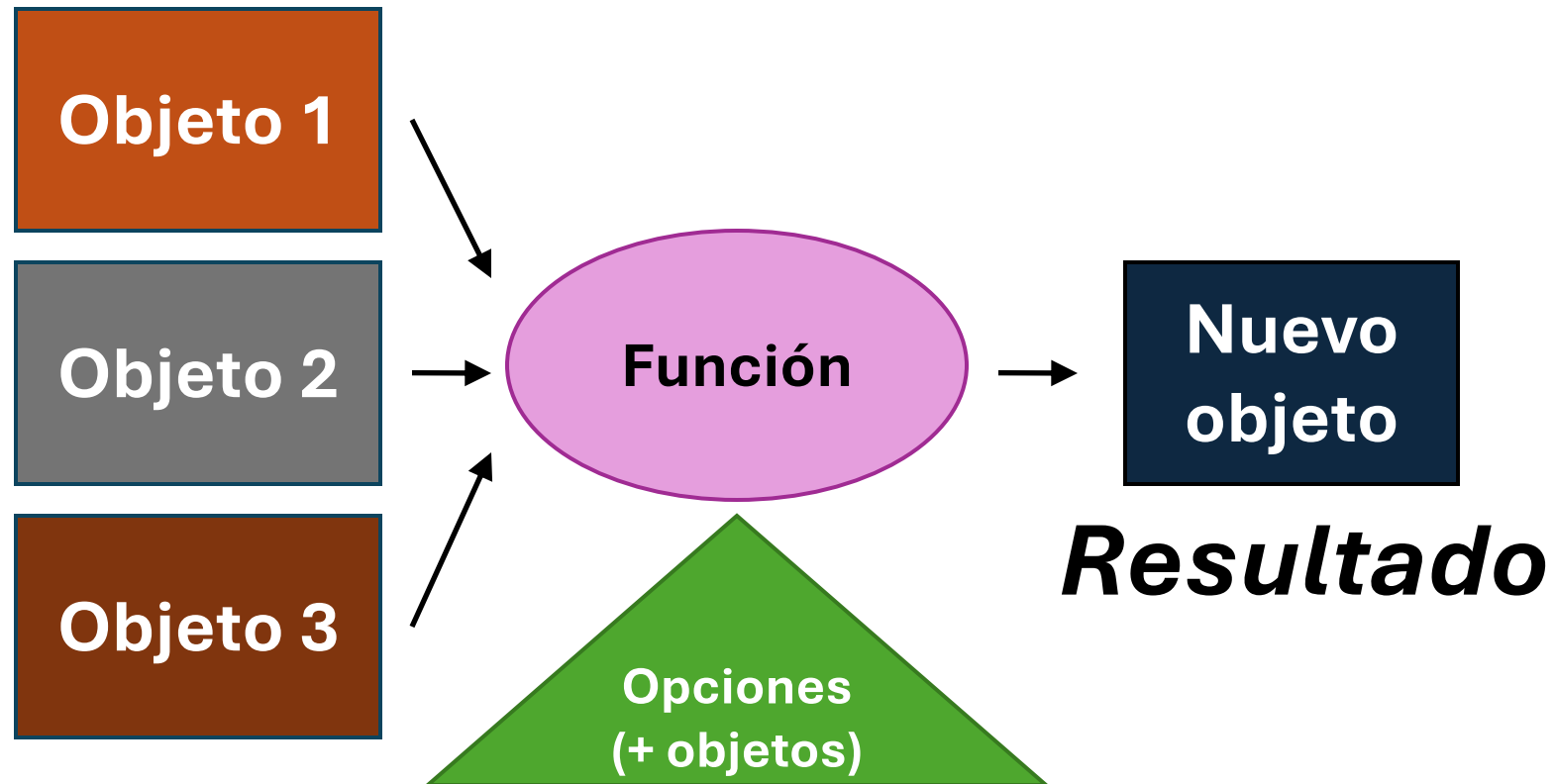
# Ejemplo de funciones con varios objetos

```
a <- 1  
b <- 2  
c <- 3
```

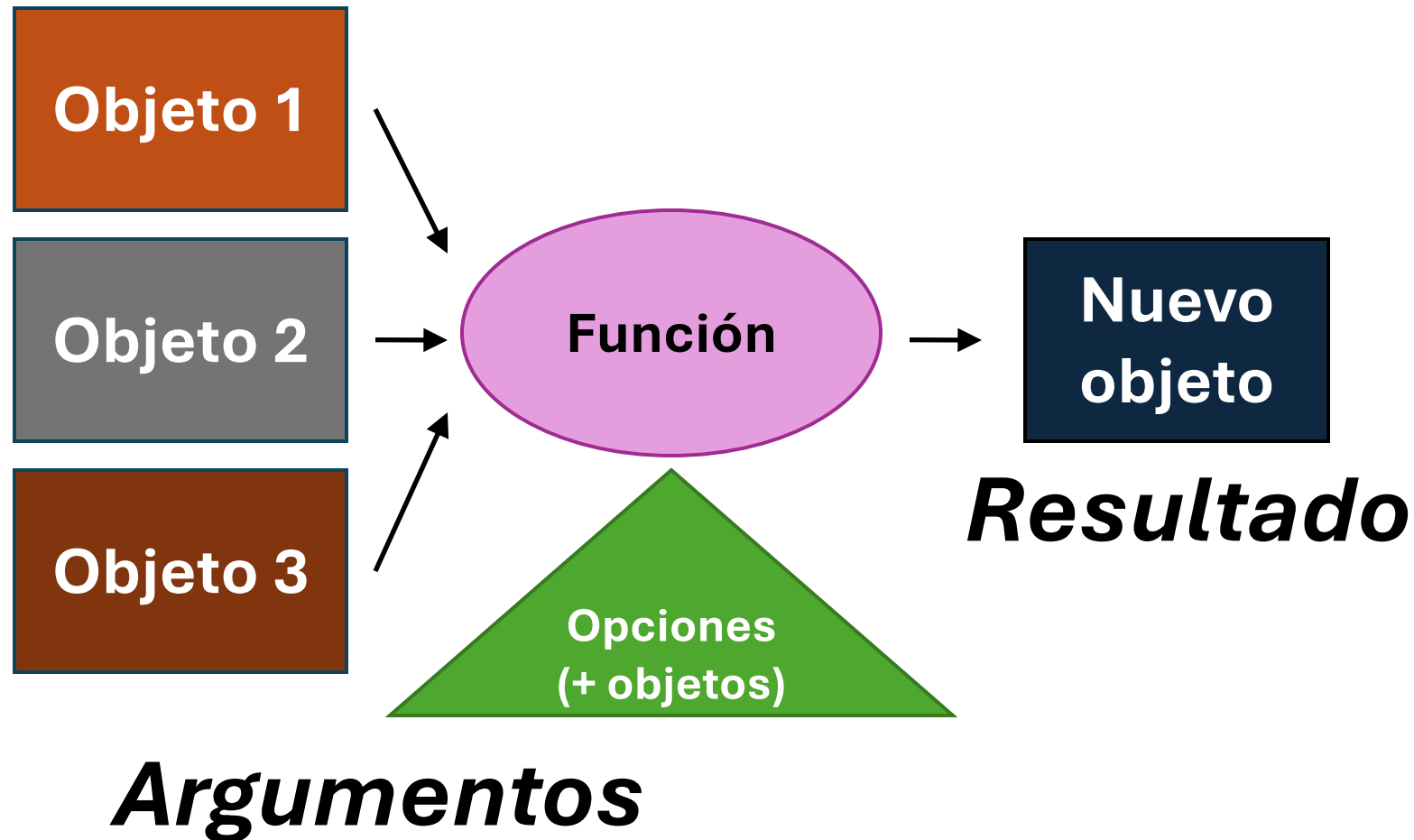
```
sum(a, b, c)
```

```
[1] 6
```

... algunos de los cuales cambian cómo opera la función



Tanto los datos de entrada como las opciones son argumentos



# Ejemplo de opciones

```
several_data <- c(15, 16, 17, 24, 10, NA)
mean(several_data)
[1] NA
```

```
mean(several_data, na.rm = TRUE)
[1] 16.4
```

# Argumentos de una función

- Muchas funciones tendrán valores predeterminados para los argumentos
  - Si no se especifica, el argumento tomará ese valor
- Para encontrar estos valores y una lista de todos los argumentos, haga lo siguiente: `?function.name`
- Si solo está buscando funciones relacionadas con una palabra, usaría google. Pero también puedes: `??key.word`



# Ayuda de funciones

```
?sqrt
```

```
??sqrt
```

# Ayuda de funciones:

Descripción

Sintaxis

Argumentos

Detalles

Resultado

```
R Help on 'sqrt'
package:base
R Documentation

Miscellaneous Mathematical Functions

Description:
  These functions compute miscellaneous mathematical functions. The
  naming follows the standard for computer languages such as C or
  Fortran.

Usage:
  abs(x)
  sqrt(x)

Arguments:
  x: a numeric or 'complex' vector or array.

Details:
  These are internal generic primitive functions: methods can be
  defined for them individually or via the 'Math' group generic.
  For complex arguments (and the default method), 'z', 'abs(z)
  == Mod(z)' and 'sqrt(z) == z^0.5'.

'abs(x)' returns an 'integer' vector when 'x' is
'integer' or 'logical'.

S4 methods:
  Both are S4 generic and members of the 'Math' group generic.

References:
  Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S
  Language_. Wadsworth & Brooks/Cole.

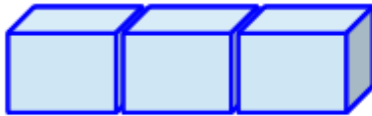
See Also:
  'Arithmetic' for simple, 'log' for logarithmic, 'sin'
  for trigonometric, and 'Special' for special mathematical
  functions.

  'plotmath' for the use of 'sqrt' in plot annotation.

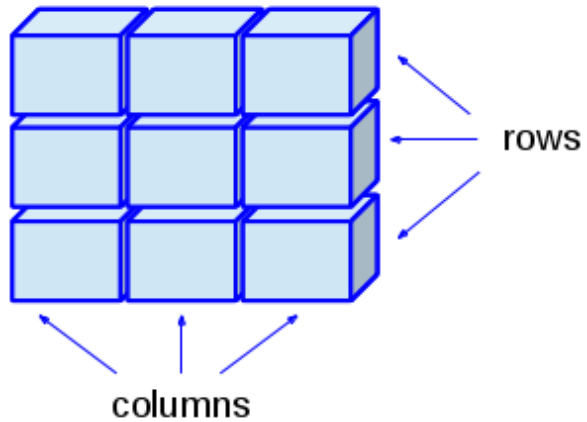
Examples:
  require(stats) # for spline
  require(graphics)
  xx <- -9:9
  plot(xx, sqrt(abs(xx)), col = "red")
  lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

# ¿Qué forma tiene un objeto?

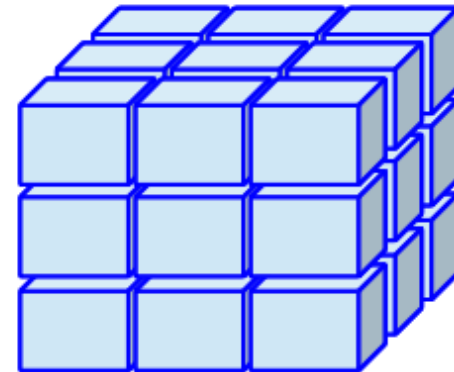
Vector



Matrix



Array



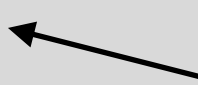
# Creación de un vector numérico (longitud >1)

```
a <- c(4, 2, 5, 10)
```

```
a
```

```
[1] 4 2 5 10
```

¡Cuatro argumentos  
pasados a esta  
función!

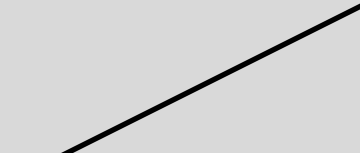


```
a <- 1:4
```

```
a
```

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

¡Dos argumentos  
pasados a esta  
función!



```
a <- seq(1, 10)
```

```
a
```

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

# Creación de vectores de texto y factores

```
species <- c("Bromus diandrus", "Bromus carinatus",  
"Bison bison")
```

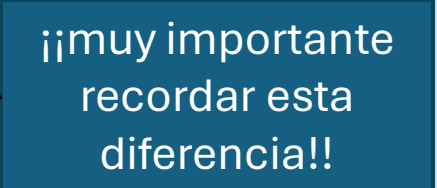
```
gender <- factor(c("Male", "Male", "Female"))
```

# Creación de un valor lógico

```
3 < 5  
[1] TRUE
```

```
3 > 5  
[1] FALSE
```

¡¡muy importante  
recordar esta  
diferencia!!



```
x = 5  
x == 5  
[1] TRUE  
x != 5  
[1] FALSE
```

## Conditional operators

< > <= >= == != %in%

# Creación de vectores lógicos

```
x <- 1:10
y <- x < 5
[1] TRUE TRUE TRUE TRUE FALSE
[6] FALSE FALSE FALSE FALSE FALSE

y <- x == 2
[1] FALSE TRUE FALSE FALSE FALSE
[6] FALSE FALSE FALSE FALSE FALSE

y <- Species == ("Bromus diandrus")
[1] TRUE FALSE FALSE
```

## Conditional operators

< > <= >= == != %in%

# Extrayendo valores de objetos

- R usa [ ] para referirse a elementos de objetos
- Por ejemplo:
  - `V[5]` devuelve el 5º elemento de un vector llamado `V`
- El número entre corchetes se denomina índice



# Extrayendo un valor de un vector

```
a <- c(3, 2, 7, 8)
```

```
a[3]
```

```
????
```

```
a[1:3]
```

```
????
```

```
a[seq(2, 4)]
```

```
????
```

# Extrayendo un valor de un vector

```
a <- c(3, 2, 7, 8)
```

```
a[3]
```

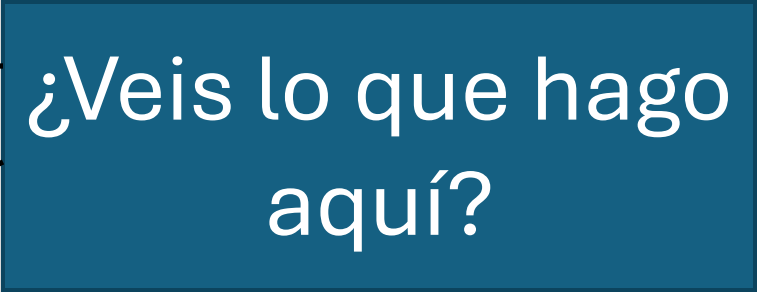
```
[1] 7
```

```
a[1:3]
```

```
[1] 3 2 7
```

```
a[seq(2, 4)]
```

```
[1] 2 7 8
```



¿Veis lo que hago aquí?

# Extrayendo un valor de un vector

```
a <- c(3,2,7,8)
```

```
a[3]
```

```
[1] 7
```

```
i <- 1:3
```

```
a[i]
```

```
[1] 3 2 7
```

```
i <- seq(2,4)
```

```
a[i]
```

```
[1] 2 7 8
```

# Vamos a probar cosas

- Ejercicios de clase
- Parte 1: Trabajando con vectores

# Creación de una matriz

```
A <- matrix(data = 0, nrow = 6, ncol = 5)
```

```
A
```

```
      [,1] [,2] [,3] [,4] [,5]  
[1,]    0    0    0    0    0  
[2,]    0    0    0    0    0  
[3,]    0    0    0    0    0  
[4,]    0    0    0    0    0  
[5,]    0    0    0    0    0  
[6,]    0    0    0    0    0
```

```
A <- matrix(data = 1:30, nrow = 6, ncol = 5)
```

```
?????
```

# Extrayendo valores de matrices

- R usa [ ] para referirse a elementos de objetos
- Por ejemplo:
  - $M[2, 3]$  devuelve el elemento de la 2ª fila, 3ª columna de la matriz M
  - $M[2, ]$  devuelve todos los elementos de la 2ª fila de la matriz M
- Los números entre corchetes se denominan índices

# Extrayendo valores de matrices

El orden es siempre [filas, columnas]

```
A <- matrix(data=1:30, nrow=6, ncol=5)
```

```
A
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	7	13	19	25
[2,]	2	8	14	20	26
[3,]	3	9	15	21	27
[4,]	4	10	16	22	28
[5,]	5	11	17	23	29
[6,]	6	12	18	24	30

```
A[3,4]
```

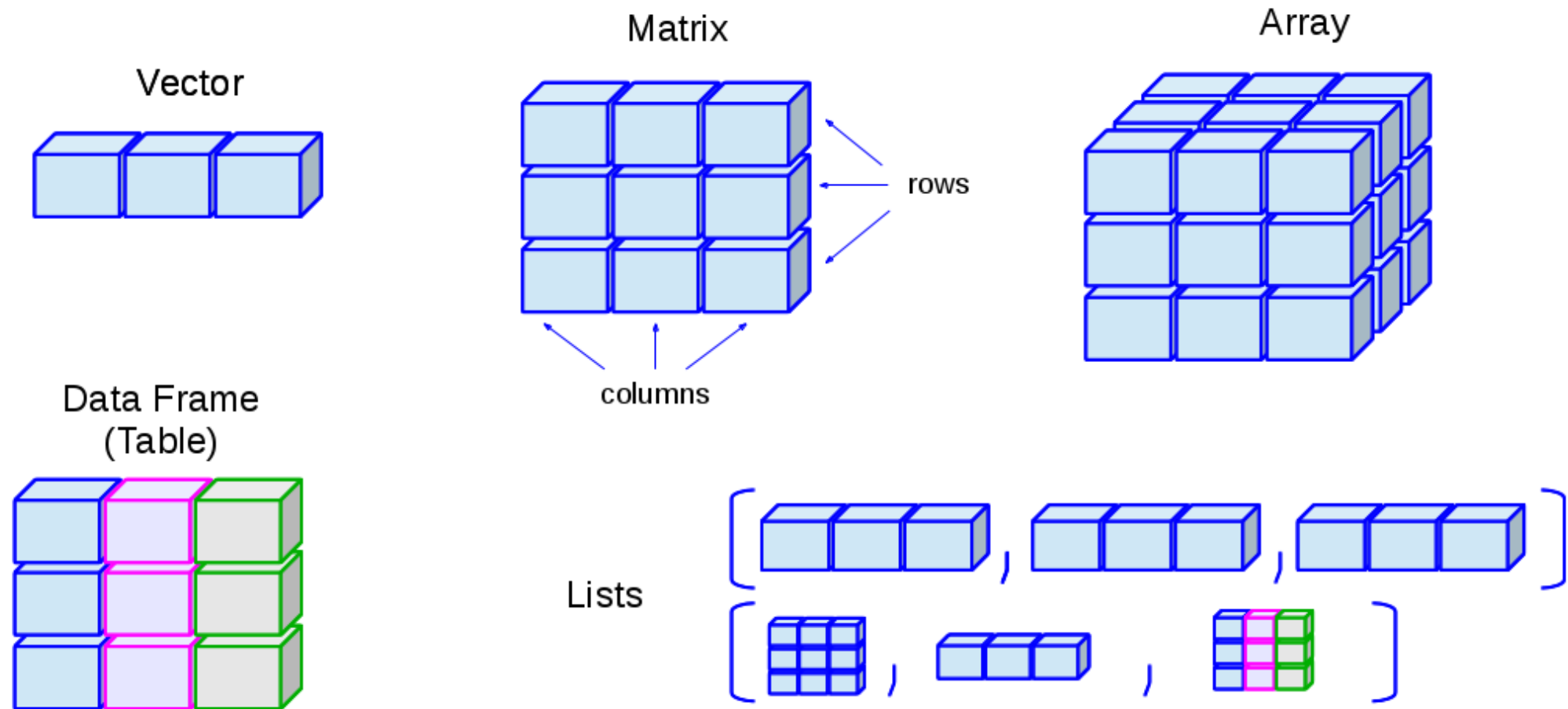
```
[1] 21
```

```
A[c(1:3),4]
```

```
A[, -1]
```

```
???
```

# ¿Pueden los objetos mezclar tipos de datos?





# Data frames

- Un *data frame* es una lista de vectores de la misma longitud
- Son equivalente a matrices con diferentes tipos de datos en cada columna
- Se usan para almacenar diferentes variables a partir de las mismas mediciones

```
df <- data.frame(  
  name = c("Diego", "Lucía", "Paco"),  
  age = c(27, 23, 24)  
)
```

# Orientarse con un *data frame*

- `head()` muestra las primeras líneas
- `tail()` muestra las últimas líneas
- `names()` Devuelve los nombres de las columnas
- **Extracción de columnas**
  - `Data$columnname`
  - `Data[, "columnname"]`
  - `Data[, 3]`

# Listas

- Es un contenedor genérico de otros tipos de variables
- Cada elemento de una lista puede ser cualquier cosa (¡incluso otra lista!)

```
a <- c(1, 2)
b <- c(10, 20, 30)
L <- list(a, b)
L
[[1]]
[1] 1 2
[[2]]
[3] 10 20 30
L[[1]]
[1] 1 2
L[[2]][2]
[1] 20
```



# ¿Probamos cosas?

- Ejercicios de clase
- Parte 2: Trabajando con matrices y *data frames*

# Archivos, objetos y proyectos

Los tres son cosas diferentes en R...

- Los archivos son scripts (e.g., un archivo de texto) o archivos de datos (e.g., Excel) en el disco duro
- Los objetos son datos generados durante una sesión de R. Sólo existen en la memoria temporal del ordenador
- Un proyecto es una carpeta con un montón de archivos y objetos de R
  - Es recomendable usar una estructura de carpetas lógica (ver a la derecha)

• Carpeta del proyecto

- Datos/
- Figuras/
- Tablas/
- Script\_1
- Script\_2
- Script\_3

# Entorno (*environment*, sesión de R)

- El espacio temporal en la memoria del equipo donde R guarda objetos mientras ejecuta un análisis
- Está limitado en tamaño por la memoria RAM del ordenador

# Funciones útiles para manejar la sesión

- Para enumerar objetos en la sesión de R: `ls()`
- Para eliminar objetos de la sesión de R: `rm()`
- Para guardar la sesión de R: `save.image()`
- Para salir de la sesión de R: `q()`

```
ls()  
rm(numeric_object, character_object)  
rm(list=ls())  
save.image()  
q()
```

# Directorio de trabajo (carpeta del proyecto)

- El directorio donde R busca archivos o escribe archivos
- `getwd()` : obtener el directorio de trabajo
- `setwd()` : lo cambia
- `dir()` : muestra el contenido de la misma

```
setwd("C:/Project Directory/")  
dir()  
[1] "a figure.pdf"  
[2] "more data.csv"  
[3] "some data.csv"
```



# Trabajar con datos

- Lectura de datos del disco duro
- R lo almacena como un objeto (guardado en la memoria de su computadora)
- Trata ese objeto como cualquier otro objeto
- Los cambios en el objeto están restringidos al objeto, no afectan a los datos del disco duro

# Leer un archivo de datos

```
myData <- read.csv("some data.csv")
```

# Escribir un archivo de datos

```
myData <- matrix(1:30, ncol = 5)

write.csv(myData, "updated data.csv")
dir()
[1] "a figure.pdf"
[2] "more data.csv"
[3] "some data.csv"
[4] "updated data.csv"
```

# Podemos leer otros ficheros, pero necesitamos ayuda externa

```
install.packages("xlsx")  
library(xlsx)  
  
read.xlsx2("Matriculados.xlsx",  
sheetIndex = 1)
```

# Scripts

- Archivos de texto
- Con llamadas a funciones (código)
- Ordenado y secuencial
  - Cargar datos
  - Modificar datos
  - Analizar datos
  - Guardar/graficar resultados
- Se puede gestionar con Rstudio
- Puede/debe contener comentarios legibles por humanos
  - Usar # antes del comentario

# Seguimos practicando

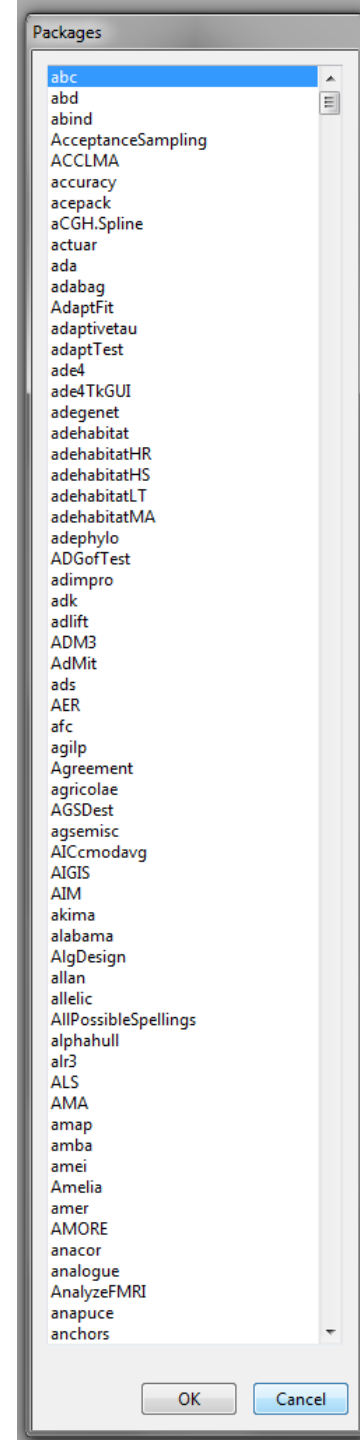
- Ejercicios de clase
- Parte 3. Usando ficheros de datos

# Paquetes

- Conjuntos de funciones para un objetivo concreto

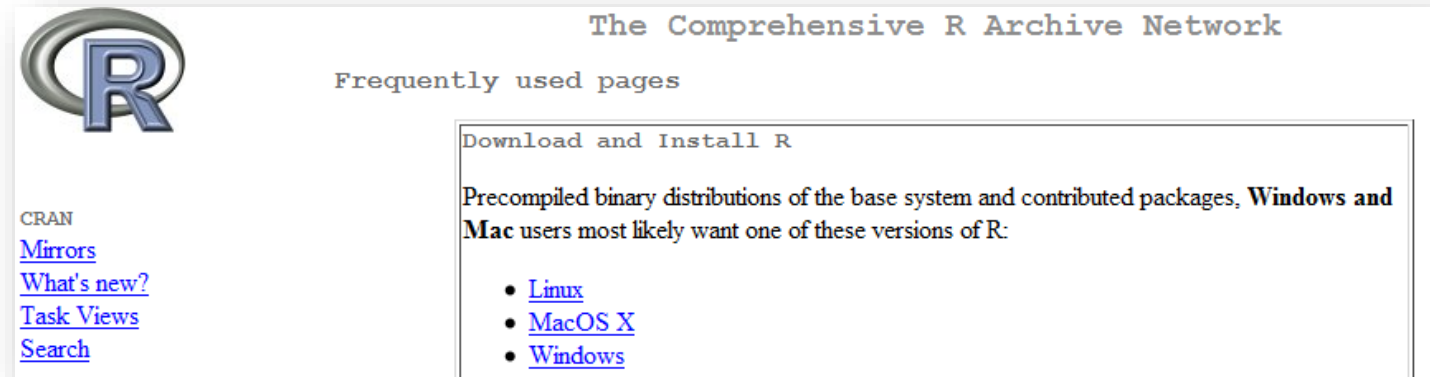
```
install.packages("vegan")
```

```
library(vegan)
```



# Paquetes y repositorios

- CRAN



- GitHub

```
install.packages(devtools)
library(devtools)
install_github("colearendt/xlsx")
```



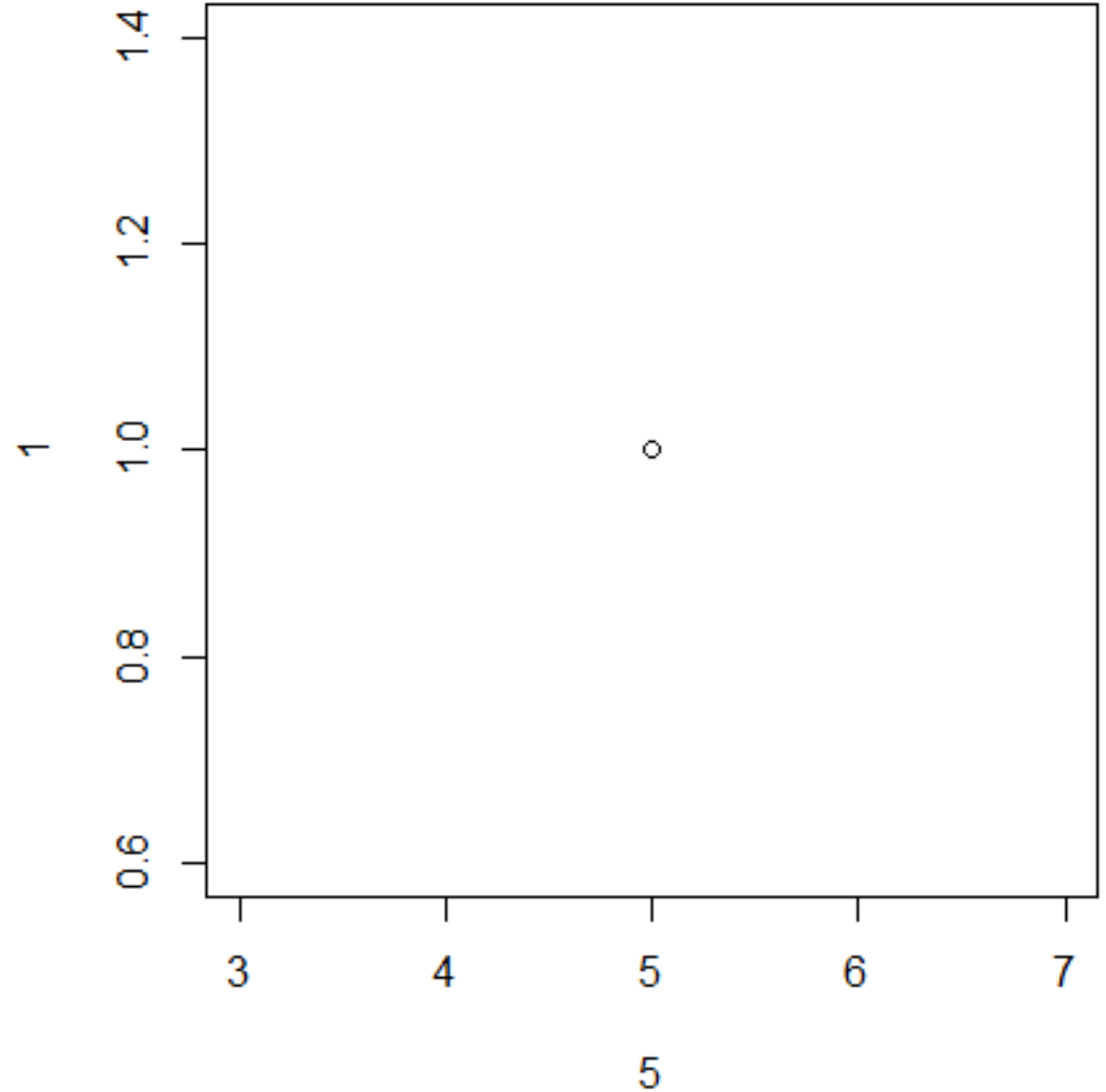
# Generación de gráficos

- Para crear un gráfico nuevo
  - `plot()`
  - `hist()`
- Para dibujar elementos en un gráfico existente
  - `points()`
  - `segments()`
  - `polygons()`



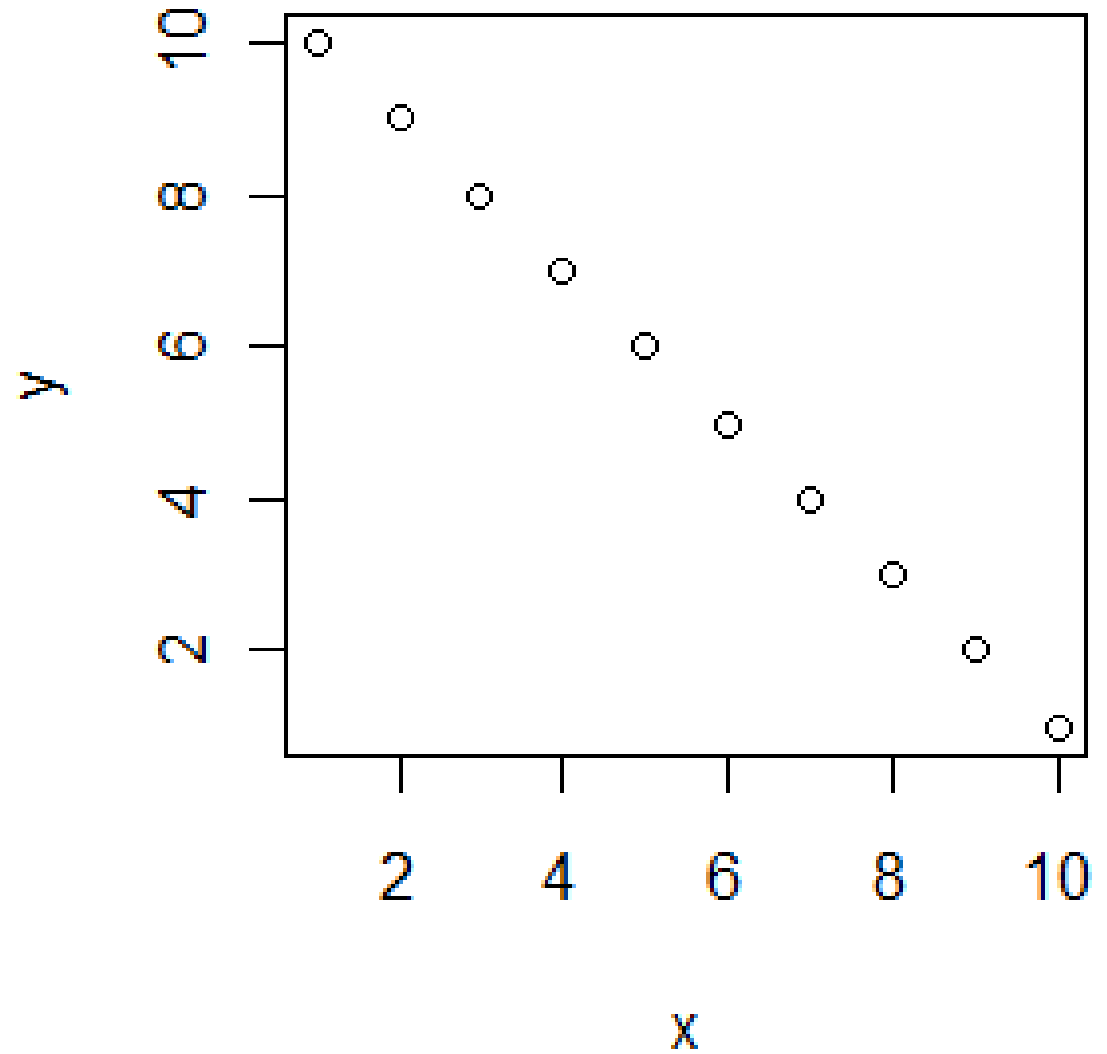
# plot() dibuja cosas

```
plot(x, y)  
plot(5, 1)
```



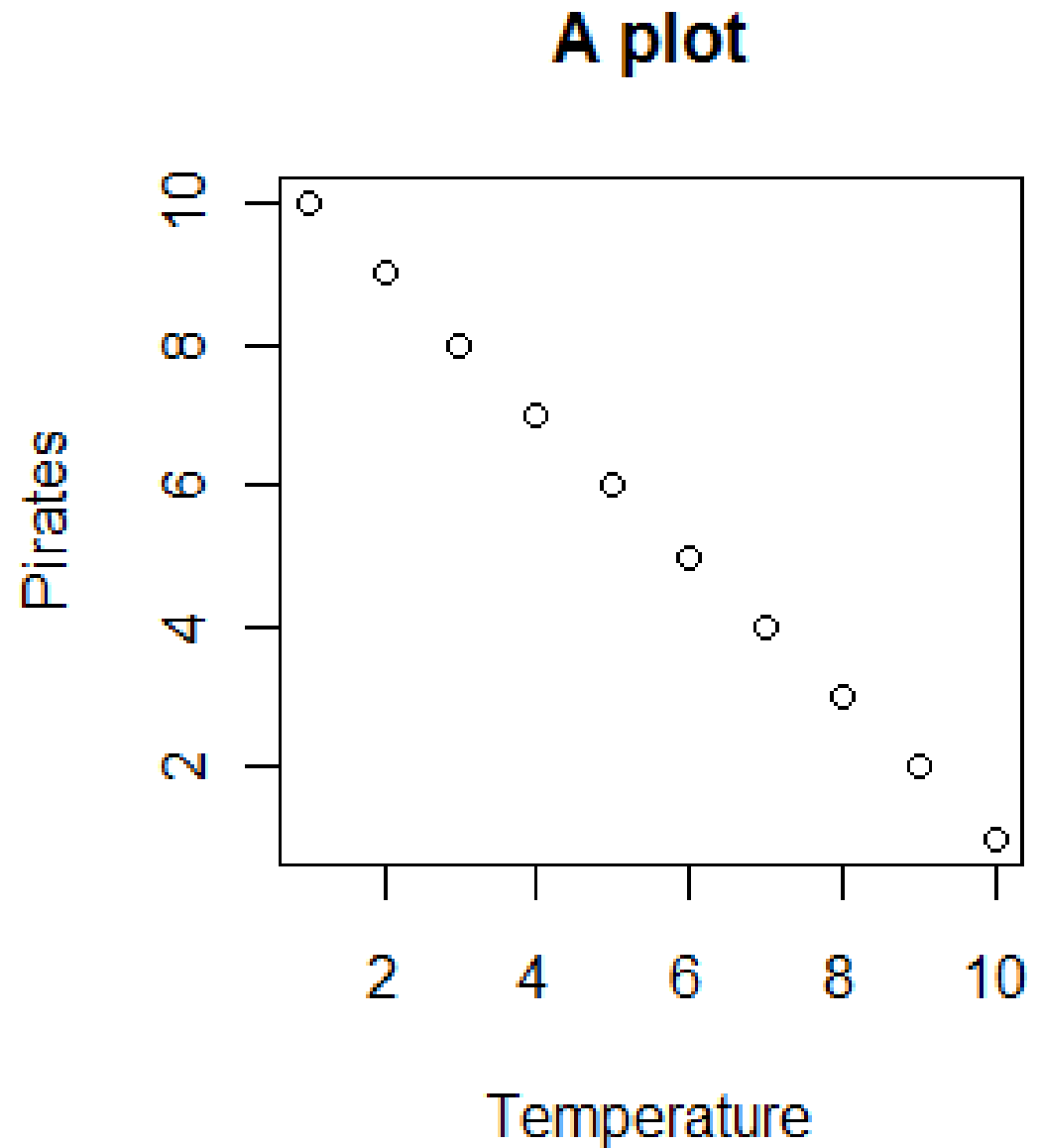
# plot()

```
x <- 1:10  
y <- 10:1  
plot(x,y)
```



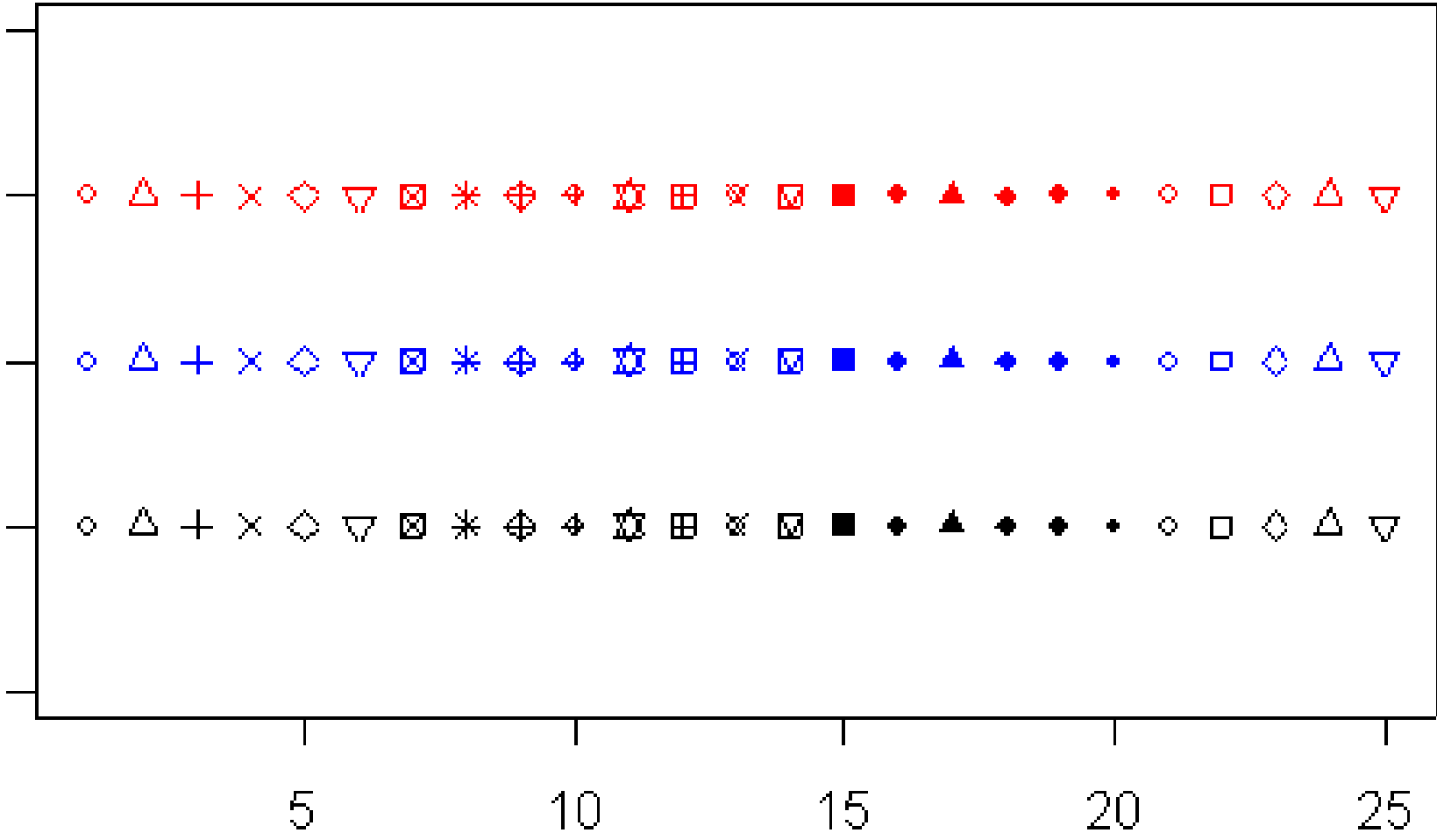
# plot()

```
x <- 1:10
y <- 10:1
plot(x,
     y,
     main = "A plot",
     xlab = "Temperature",
     ylab = "Pirates")
```



# Tipo de caracteres

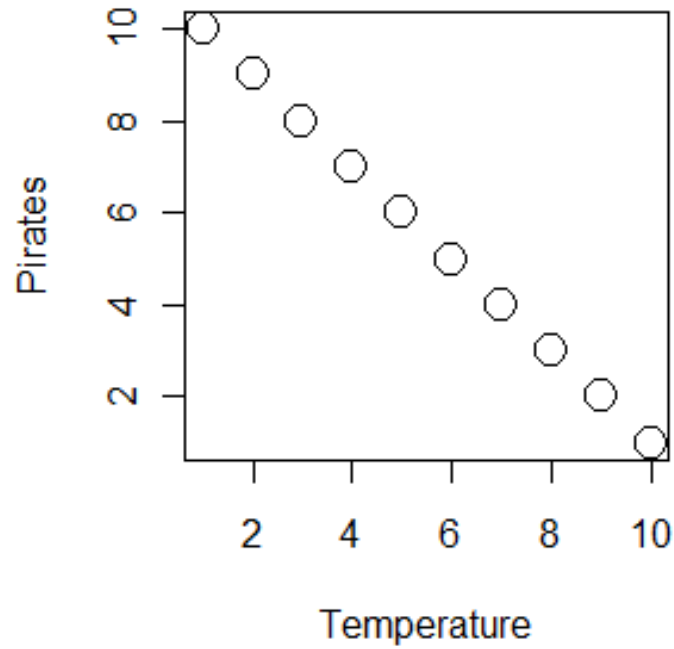
pch = 1:25



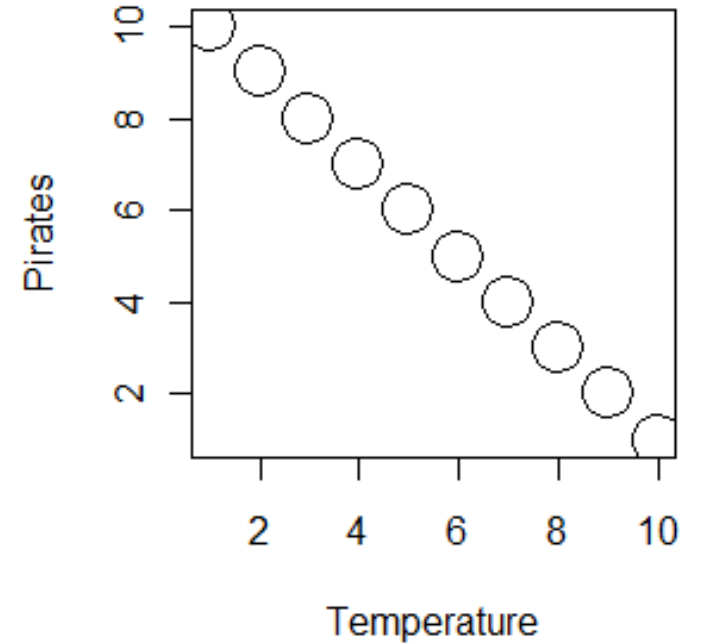
# Tamaño de caracteres

```
cex = 2 or cex = 3
```

A plot



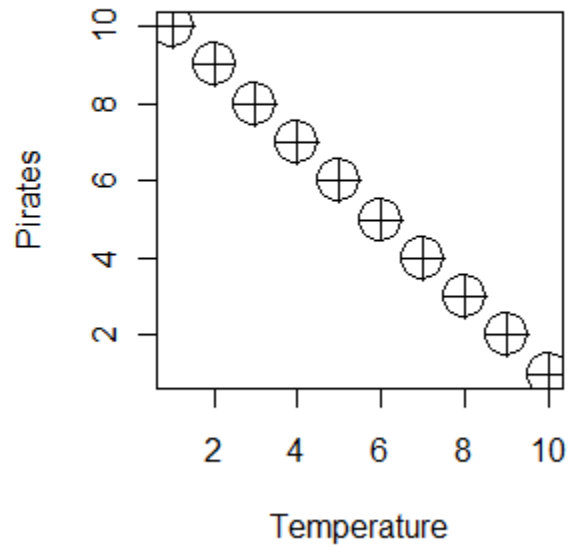
A plot



# Tamaño y tipo caracteres

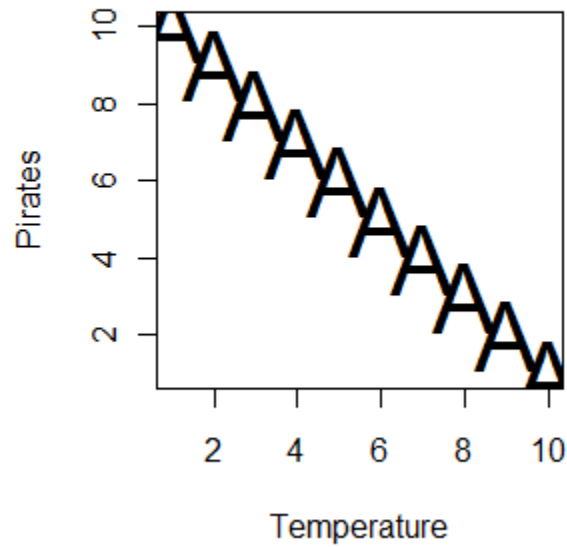
```
pch = 10, cex = 3
```

**A plot**



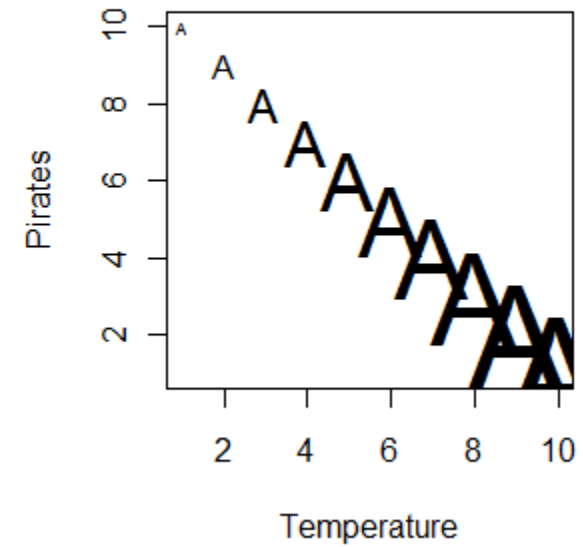
```
pch = A, cex = 3
```

**A plot**

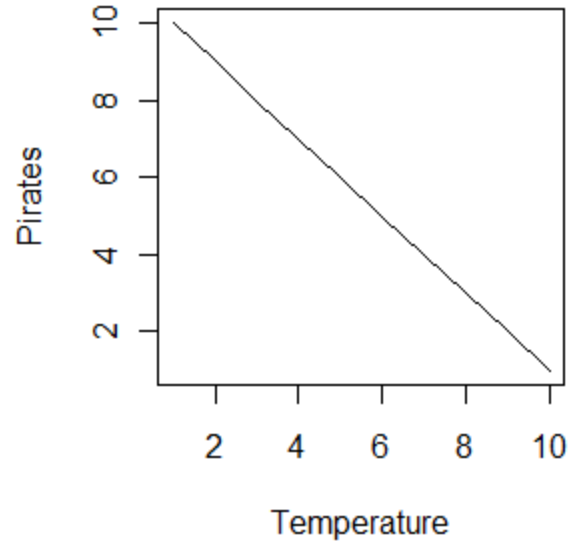


```
pch = A, cex = x
```

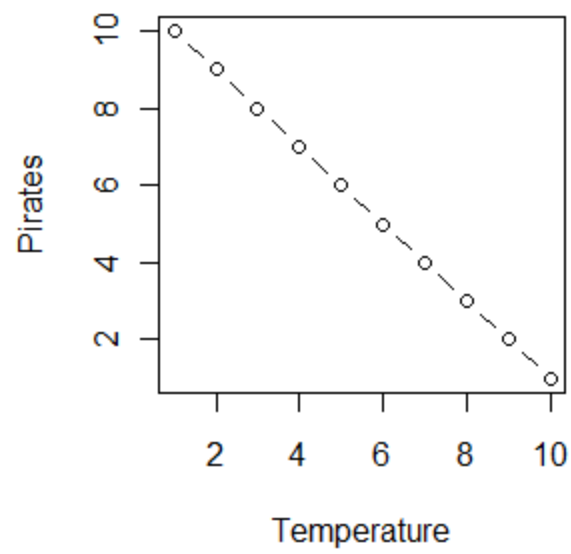
**A plot**



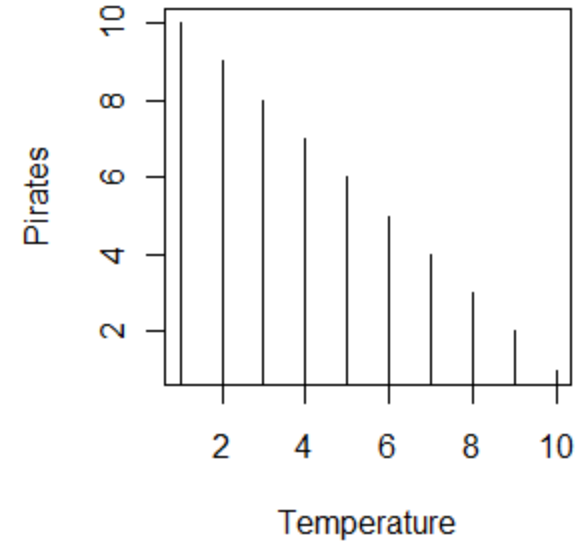
**A plot**



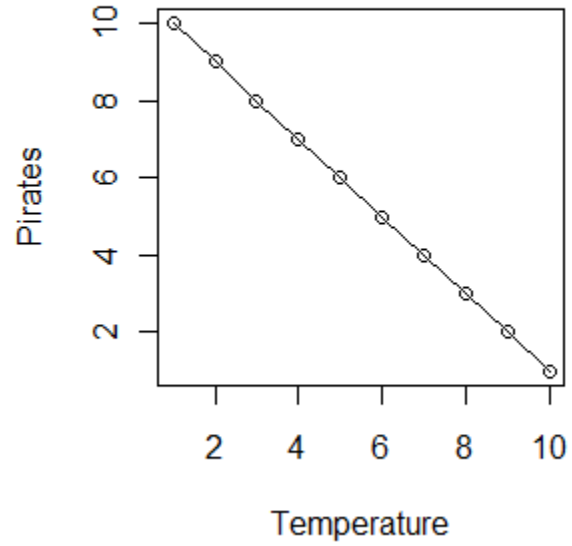
**A plot**



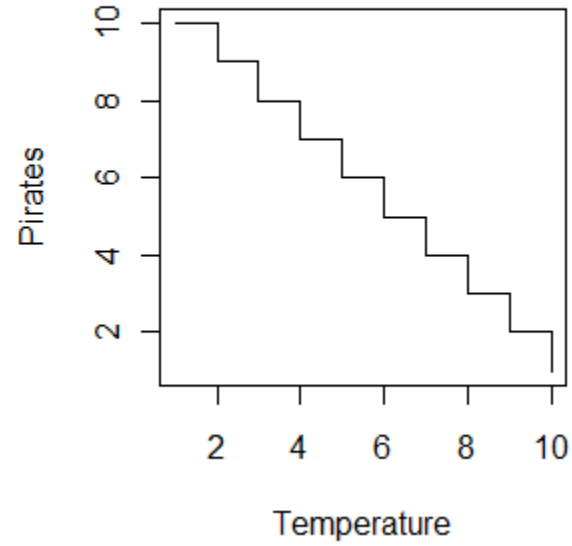
**A plot**



**A plot**



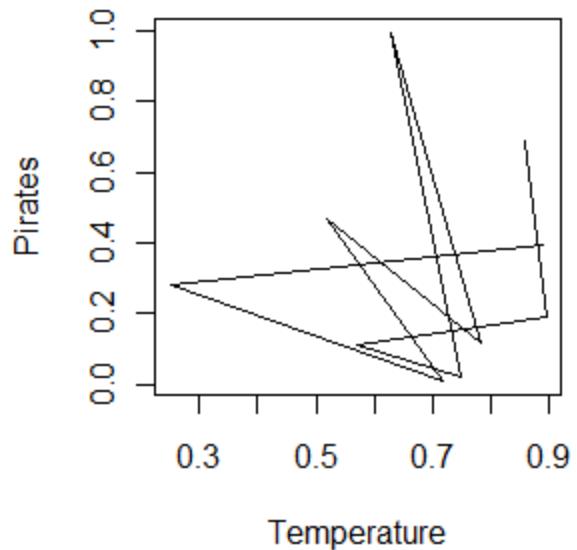
**A plot**



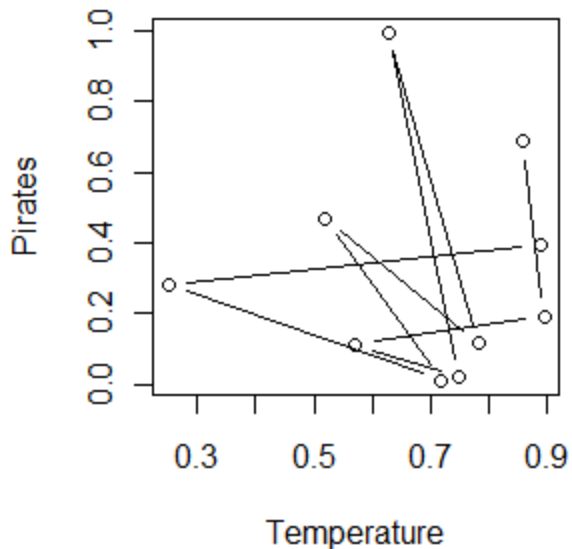
```
type =  
"l"    "b"    "h"  
"o"    "s"
```



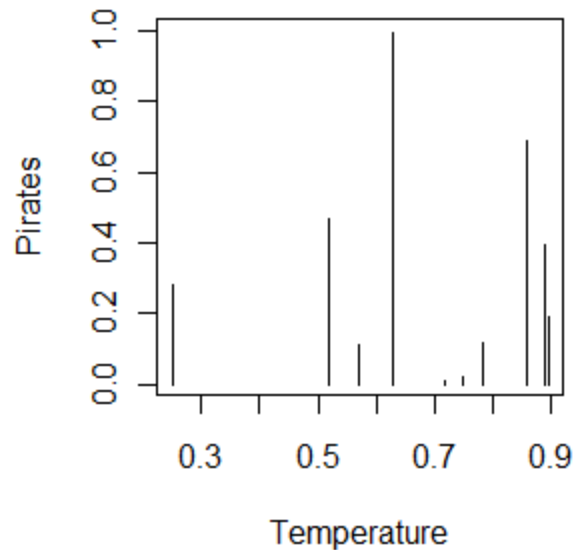
**A plot**



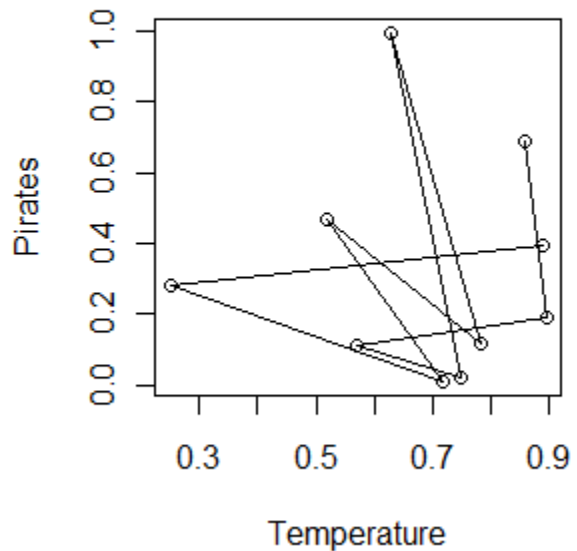
**A plot**



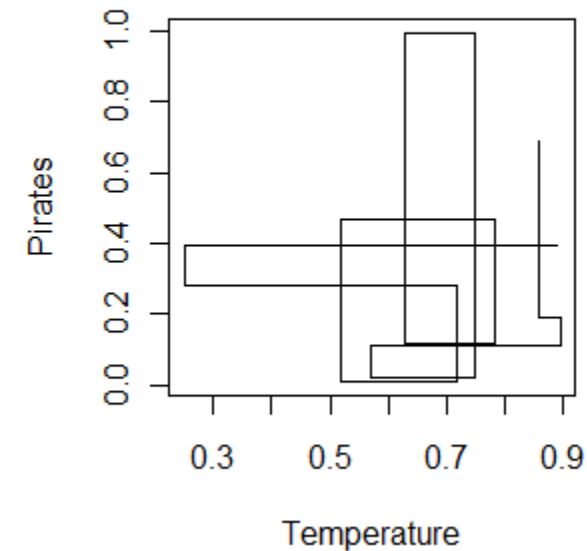
**A plot**



**A plot**



**A plot**



```

type =
"l"    "b"    "h"
"o"    "s"

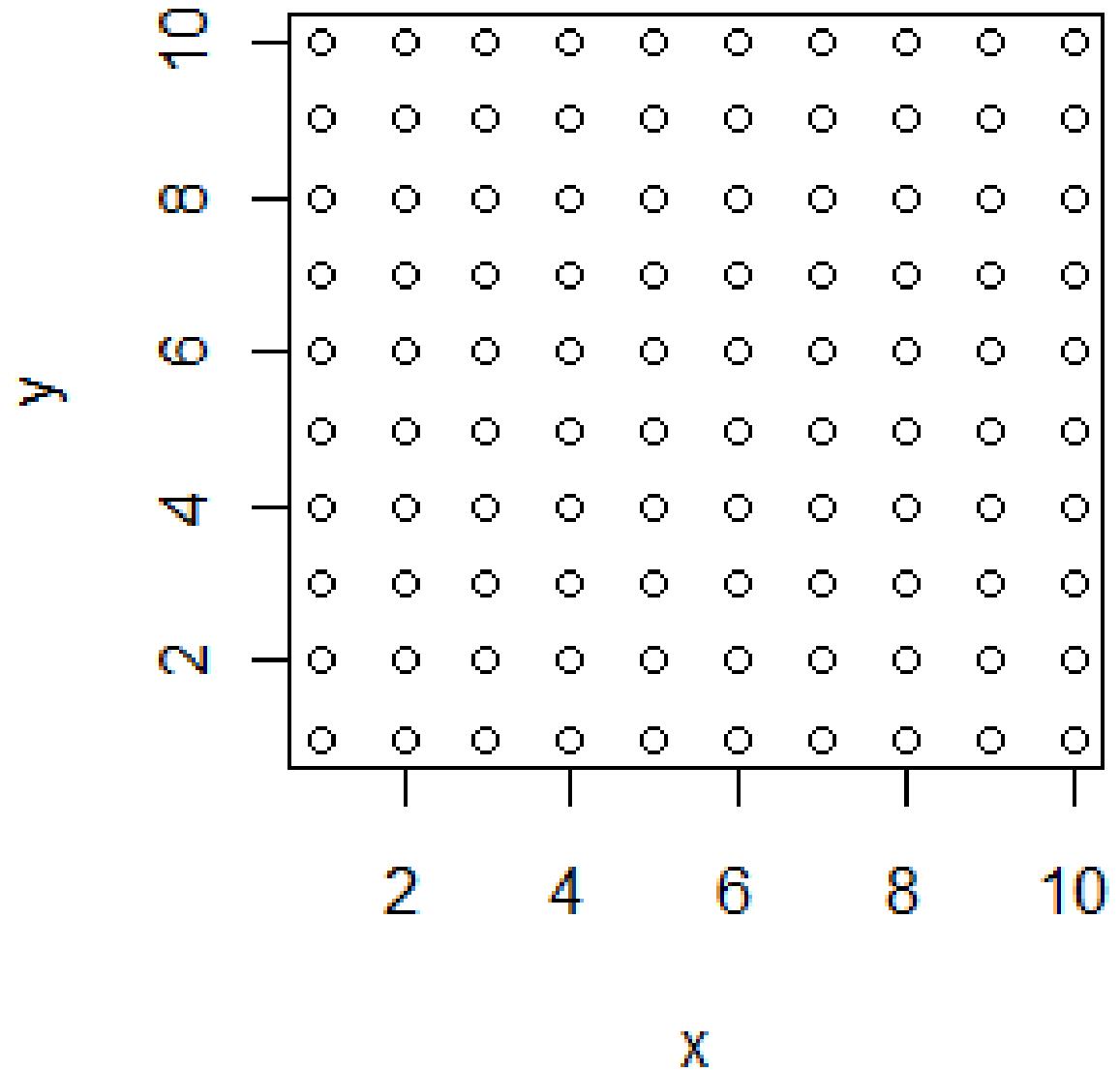
```

# Color

- Por el nombre
  - “blue” or “dark grey” . . .
- Usando funciones
  - `grey()`
  - `rainbow()`
  - `rgb()`

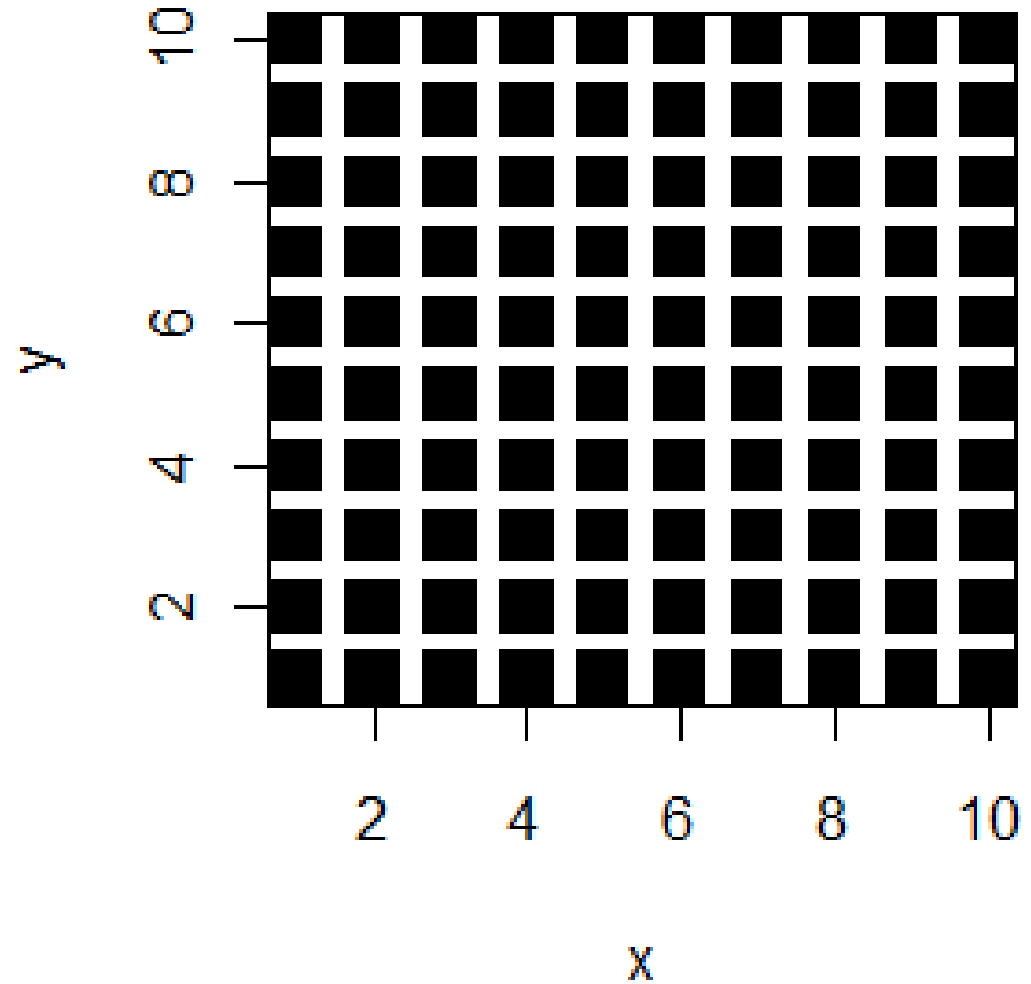
# Color

```
x <- rep(1:10, 10)  
y <- rep(1:10, each = 10)  
plot(x, y)
```



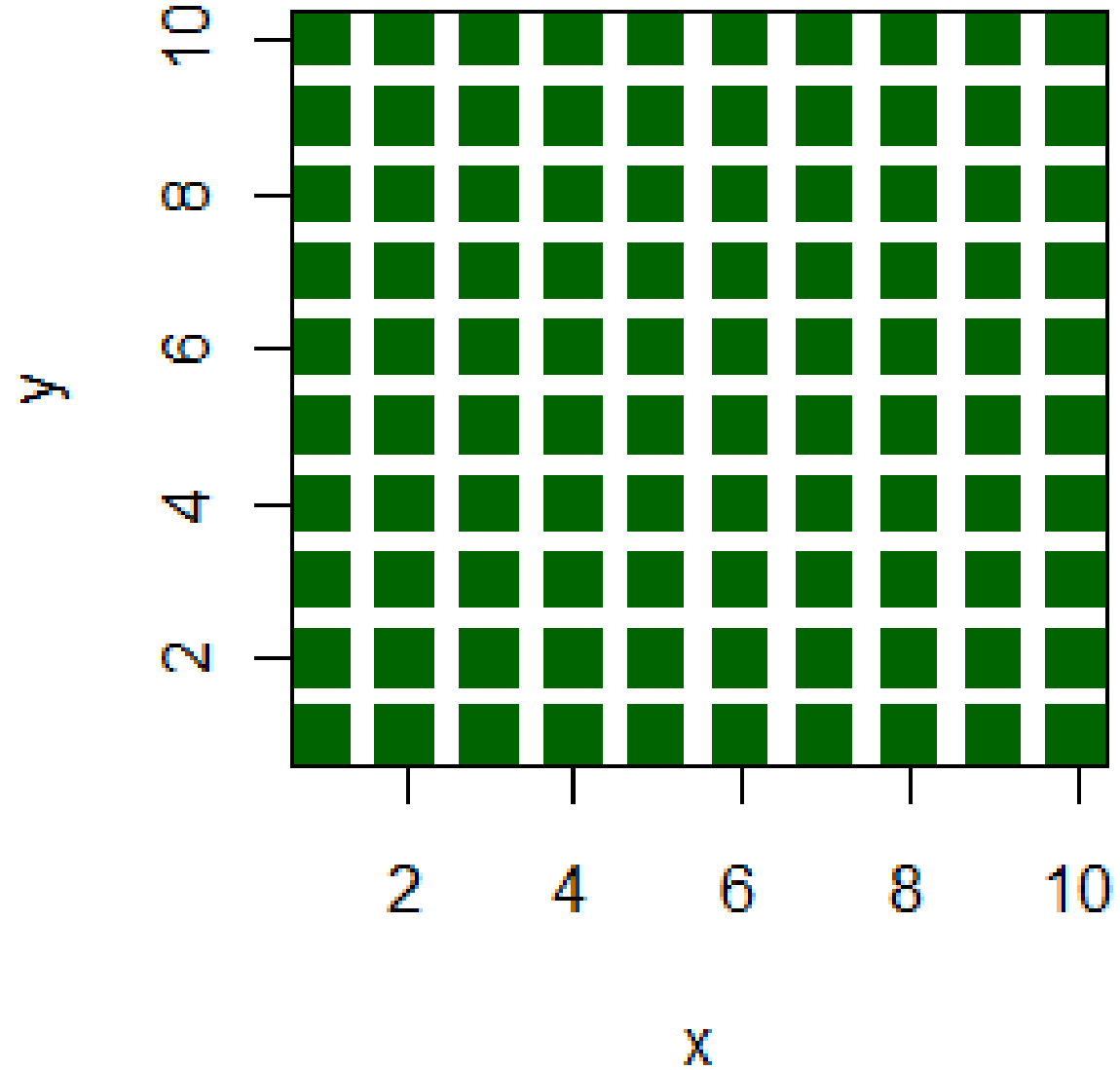
# Color

```
x <- rep(1:10, 10)  
y <- rep(1:10, each = 10)  
plot(x, y, pch = 15, cex = 2)
```



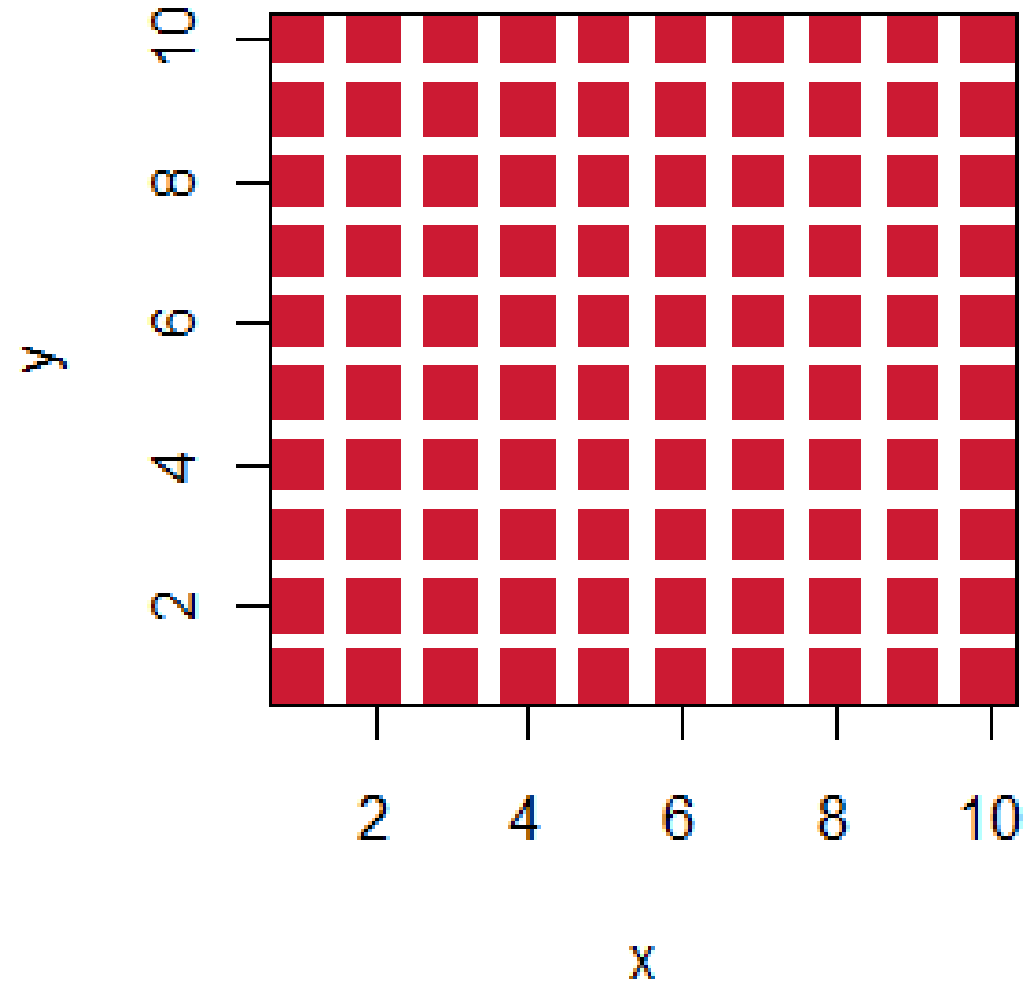
# Color

```
x <- rep(1:10,10)
y <- rep(1:10,each=10)
plot(x,
      y,
      pch = 15,
      cex = 2,
      col = "dark green")
```



# Color

```
x <- rep(1:10, 10)
y <- rep(1:10, each = 10)
plot(x,
     y,
     pch = 15,
     cex = 2,
     col = rgb(0.8, 0.1, 0.2))
```



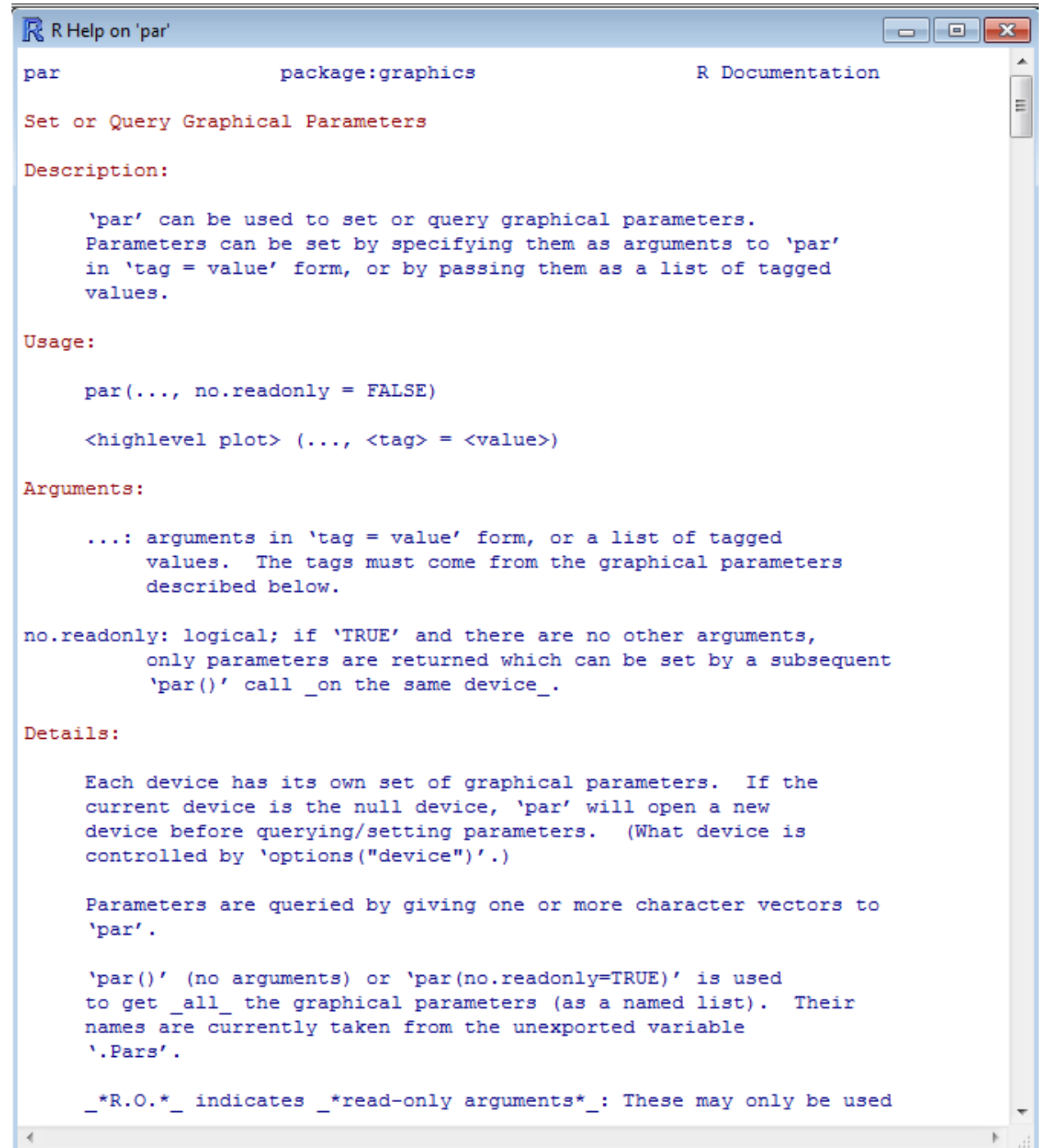
# Dibujando sobre los gráficos

- `points (x, y)` añade puntos a gráficos (opciones similares a `plot ()`)
- `segments (x0, y0, x1, y1)` añade líneas, desde un punto a otro
- `polygons ()` añade polígonos



# El maravilloso mundo de `par()`

- 70 opciones diferentes para controlar los gráficos!

A screenshot of the R Help window for the 'par' function. The window title is 'R Help on 'par'' and it shows the package 'graphics' and 'R Documentation'. The content is organized into sections: 'Description:', 'Usage:', 'Arguments:', and 'Details:'. The 'Description' section explains that 'par' can be used to set or query graphical parameters. The 'Usage' section shows the function signature 'par(..., no.readonly = FALSE)' and an example of a high-level plot call. The 'Arguments' section describes the '...' argument and the 'no.readonly' logical argument. The 'Details' section explains how parameters are queried and set, and mentions the '.Pars' variable and the '\*R.O.\*' indicator for read-only arguments.

```
R Help on 'par'
package:graphics
R Documentation

Set or Query Graphical Parameters

Description:

'par' can be used to set or query graphical parameters.
Parameters can be set by specifying them as arguments to 'par'
in 'tag = value' form, or by passing them as a list of tagged
values.

Usage:

par(..., no.readonly = FALSE)

<highlevel plot> (... , <tag> = <value>)

Arguments:

...: arguments in 'tag = value' form, or a list of tagged
values. The tags must come from the graphical parameters
described below.

no.readonly: logical; if 'TRUE' and there are no other arguments,
only parameters are returned which can be set by a subsequent
'par()' call _on the same device_.

Details:

Each device has its own set of graphical parameters. If the
current device is the null device, 'par' will open a new
device before querying/setting parameters. (What device is
controlled by 'options("device")'.)

Parameters are queried by giving one or more character vectors to
'par'.

'par()' (no arguments) or 'par(no.readonly=TRUE)' is used
to get _all_ the graphical parameters (as a named list). Their
names are currently taken from the unexported variable
'.Pars'.

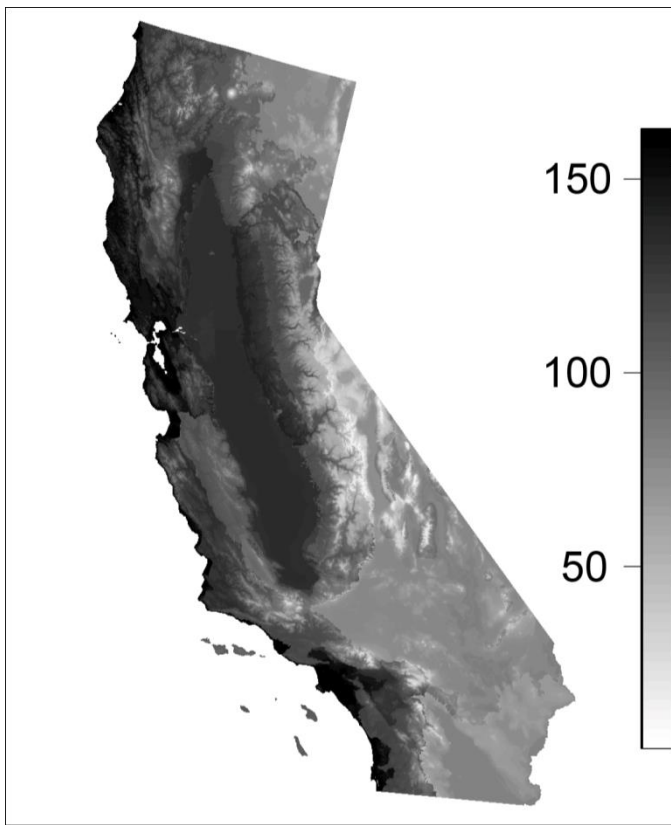
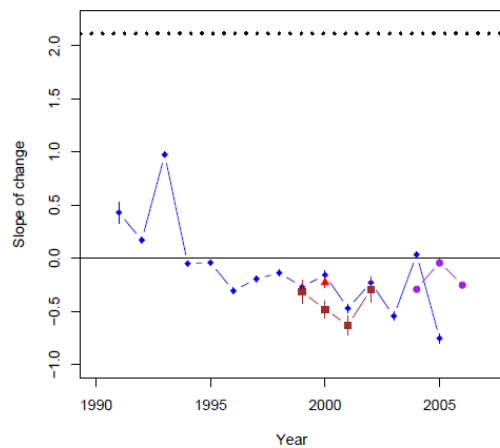
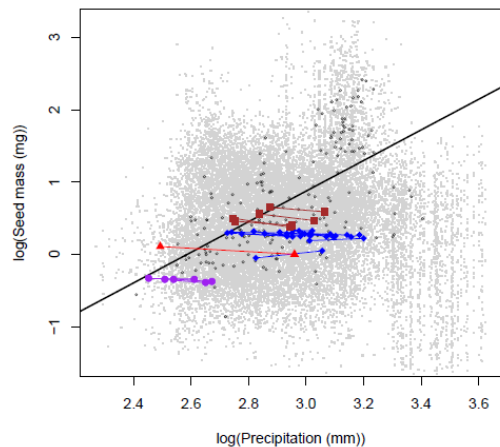
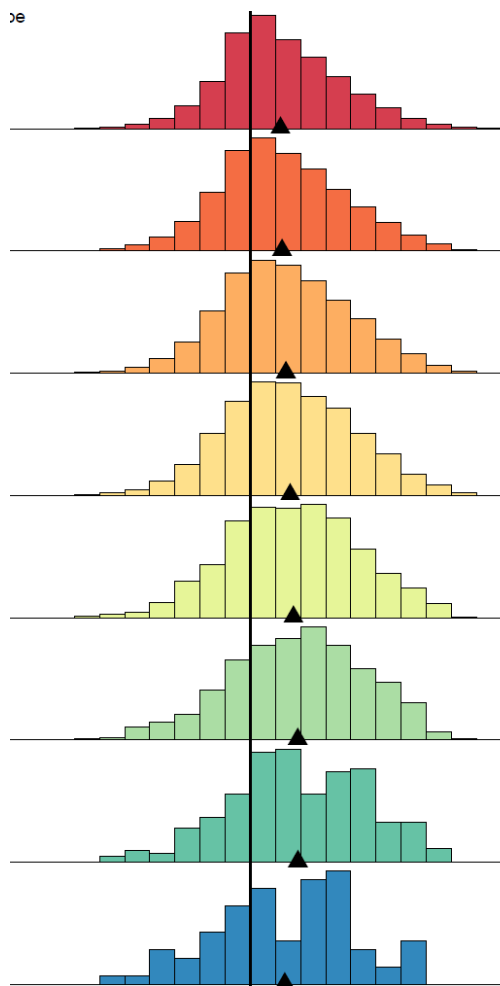
_*R.O.*_ indicates _*read-only arguments*_: These may only be used
```



# Guardando el gráfico a un fichero

- `pdf()` o `bmp()`
  - `plot()`
- `dev.off()`

# Algunos ejemplos



Todos creados entéramente con R!

# Practicar

- Ejercicios de clase
- Parte 4.