



# 1. Data Discovery

Introducción a R



# Presentación

## Jesús Javier Moralo García

- ^^ Bioinformático por la UAM & CSIC
- ^^ Máster Big Data & Analytics de Datahack
- ^^ Gestión, Análisis e Integración de Datos
  - ^^ Global Biodiversity Information Facility ([www.gbif.org](http://www.gbif.org))
  - ^^ Proyecto Europeo para la Integración de Colecciones de Historia Natural ([www.synthesys.info](http://www.synthesys.info))
  - ^^ Plataforma Europea de Cibertaxonomía(<http://cybertaxonomy.eu>)
- ^^ Dirección de equipos Técnicos
- ^^ Data & IA Creative en datahack
  - ^^ Diseño y Desarrollo de Bots
  - ^^ Proyectos en Big Data & Analytics



Imagen de <https://dribbble.com/egorkosten>

 <https://www.linkedin.com/in/jesusjaviermoralo/>

 @JJavierMoralo

 <https://github.com/javiermoralo>



# Índice

---

- **Introducción**
- R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- Explorando un DataFrame
- Gráficos
- Escribiendo funciones
- Librerías de R



# ¿Qué es R?

---

- Creado en 1993 por el Dpto. Estadística de Auckland
- Open source lenguaje programación S (Lab. Bell - 1976)
- Lenguaje de programación para el análisis estadístico
- Multiplataforma (Windows, Linux y Mac)
- Sintaxis muy simple e intuitiva
- Basado en librerías
- Comunidad muy activa (+ 7000 paquetes en CRAN)
- Mayoría de los objetos en RAM
- Uno de los mejores lenguajes para Data Science



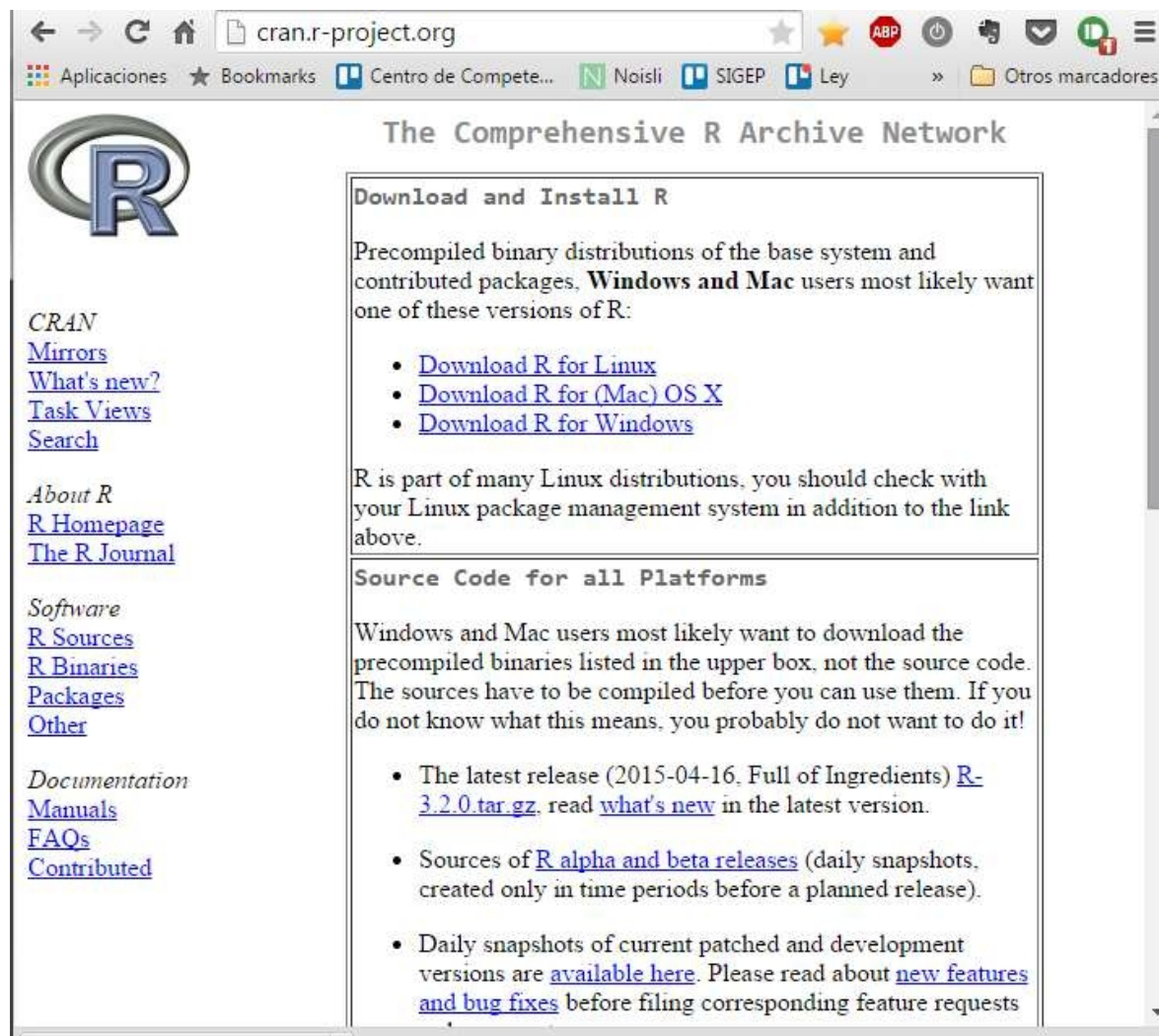
# Funcionalidades de R

---

- Análisis Estadísticos
- Data Mining y Limpieza de datos
- Estructurar conjuntos de datos
- Cambiar la forma de los datos
- Visualización de datos
- Análisis de Grafos
- Machine Learning
- Deep Learning
- Análisis de datos Interactivo
- Generación de Informes



# Instalando R



The screenshot shows a web browser window with the address bar displaying 'cran.r-project.org'. The browser's toolbar includes various icons for applications, bookmarks, and search engines. The website content features the R logo on the left, followed by a sidebar with links for 'CRAN', 'Mirrors', 'What's new?', 'Task Views', 'Search', 'About R', 'R Homepage', 'The R Journal', 'Software', 'R Sources', 'R Binaries', 'Packages', 'Other', 'Documentation', 'Manuals', 'FAQs', and 'Contributed'. The main content area is titled 'The Comprehensive R Archive Network' and contains two sections: 'Download and Install R' and 'Source Code for all Platforms'. The 'Download and Install R' section provides information about precompiled binary distributions and lists links for downloading R for Linux, Mac OS X, and Windows. The 'Source Code for all Platforms' section explains that source code requires compilation and lists links for the latest release, alpha and beta releases, and daily snapshots.

**The Comprehensive R Archive Network**

**Download and Install R**

Precompiled binary distributions of the base system and contributed packages. **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

**Source Code for all Platforms**

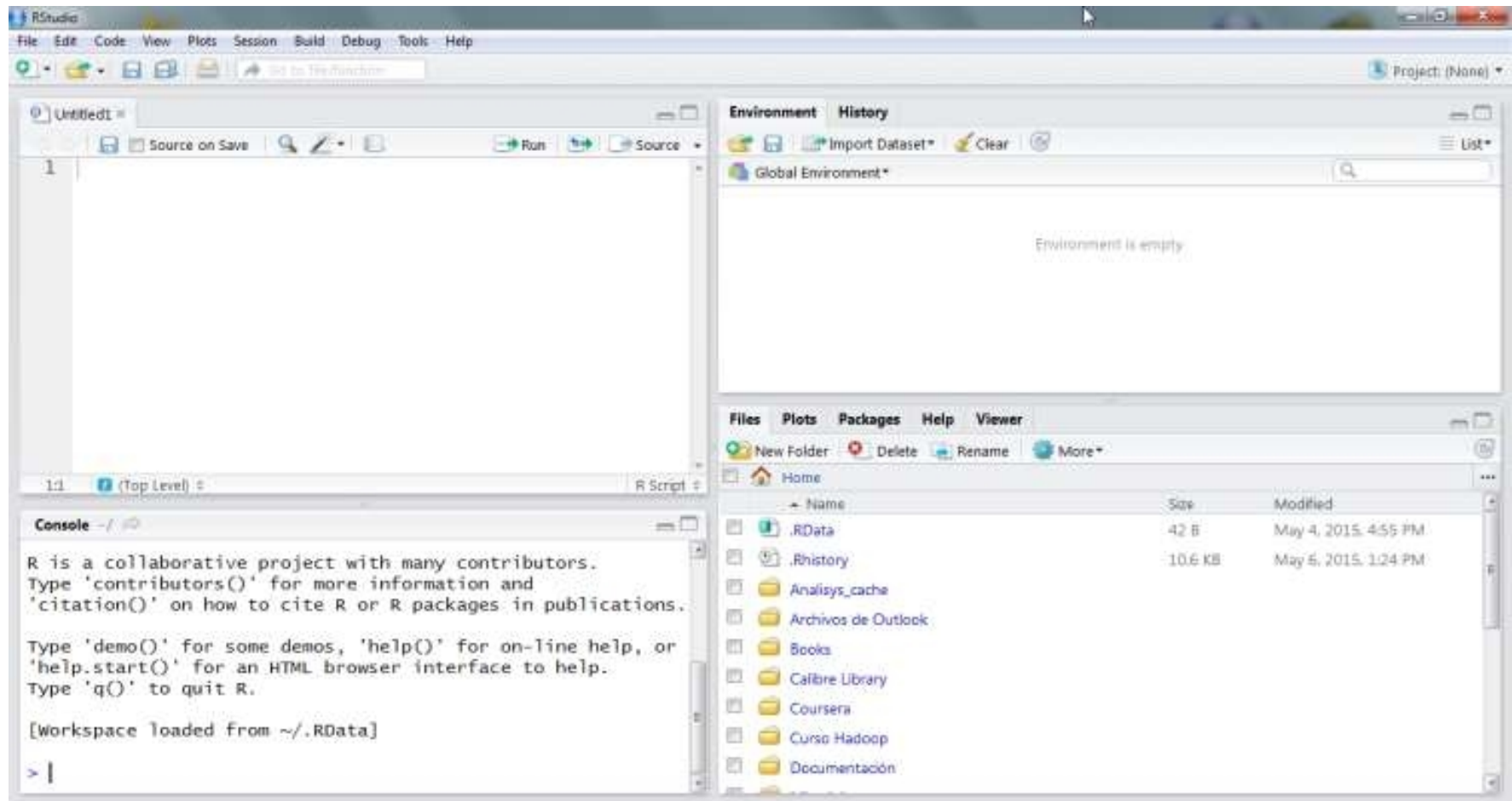
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2015-04-16, Full of Ingredients) [R-3.2.0.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests

# Instalando RStudio



<http://www.rstudio.com/>



# Interfaz de R

---

- **Scripts:** Ficheros que contienen código R reproducible en la consola
  - Ficheros con extensión .R
- **Comandos** interactivos que se introducen directamente en la consola
  - Buenos para mirar el aspecto que tienen los datos
  - Probar cosas
  - Mostrar gráficos





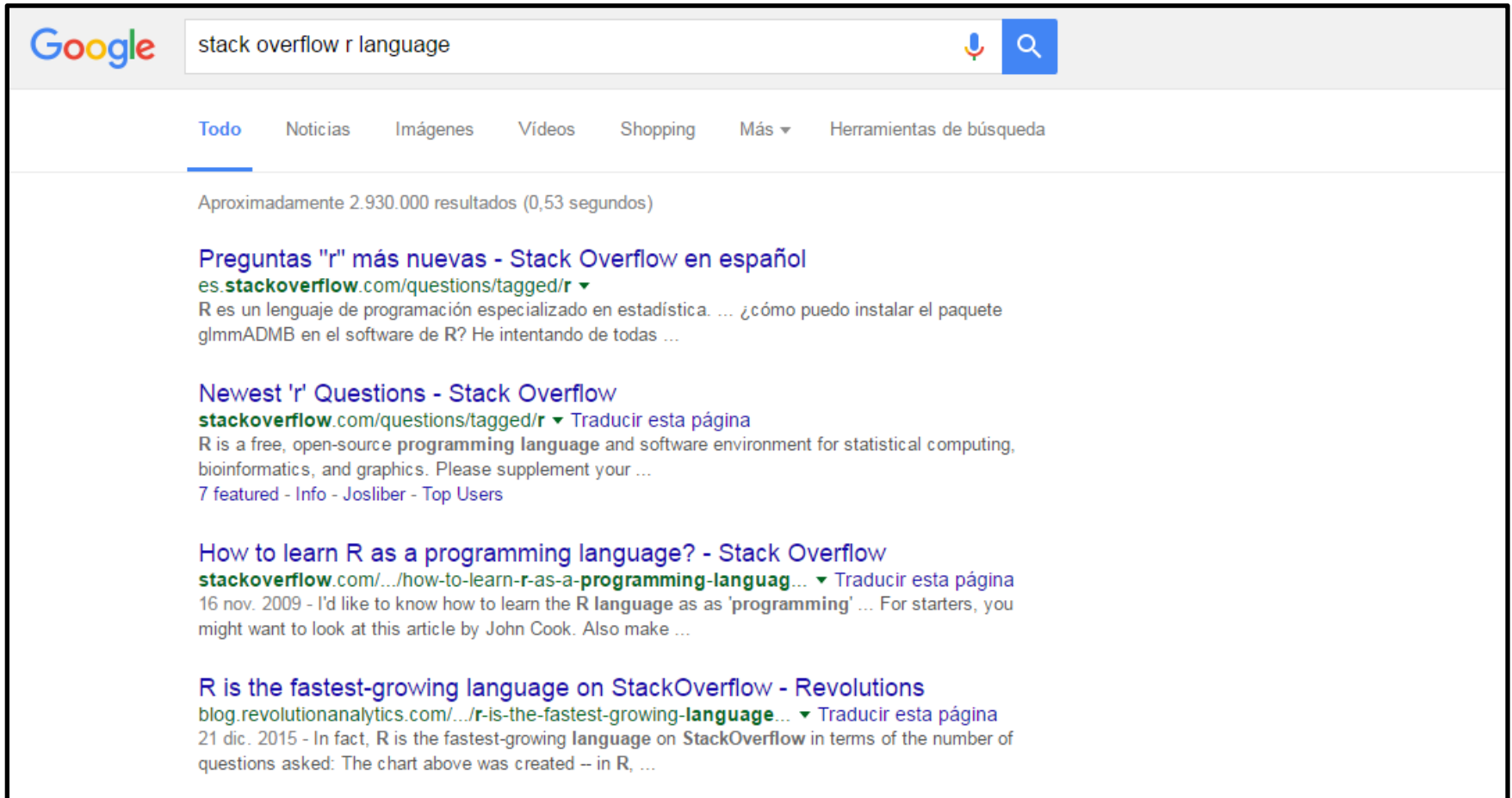
# Obteniendo ayuda

---

- **help.start()**
  - Ayuda general
- **help(mean)**
  - Ayuda específica a una función
  - ?mean
- **help.search("mean")**
  - Encontrar una función en cualquier página de la ayuda
  - ??mean
- **example(mean)**
  - Mostar un ejemplo de uso de una función



# Obteniendo ayuda – StackOverflow



The screenshot shows a Google search interface. The search bar contains the text "stack overflow r language". Below the search bar, there are tabs for "Todo", "Noticias", "Imágenes", "Vídeos", "Shopping", "Más", and "Herramientas de búsqueda". The "Todo" tab is selected. Below the tabs, it says "Aproximadamente 2.930.000 resultados (0,53 segundos)". There are four search results listed:

- Preguntas "r" más nuevas - Stack Overflow en español**  
[es.stackoverflow.com/questions/tagged/r](https://es.stackoverflow.com/questions/tagged/r) ▼  
R es un lenguaje de programación especializado en estadística. ... ¿cómo puedo instalar el paquete glmADMB en el software de R? He intentando de todas ...
- Newest 'r' Questions - Stack Overflow**  
[stackoverflow.com/questions/tagged/r](https://stackoverflow.com/questions/tagged/r) ▼ Traducir esta página  
R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your ...  
7 featured - Info - Josliber - Top Users
- How to learn R as a programming language? - Stack Overflow**  
[stackoverflow.com/.../how-to-learn-r-as-a-programming-language...](https://stackoverflow.com/.../how-to-learn-r-as-a-programming-language...) ▼ Traducir esta página  
16 nov. 2009 - I'd like to know how to learn the R language as as 'programming' ... For starters, you might want to look at this article by John Cook. Also make ...
- R is the fastest-growing language on StackOverflow - Revolutions**  
[blog.revolutionanalytics.com/.../r-is-the-fastest-growing-language...](https://blog.revolutionanalytics.com/.../r-is-the-fastest-growing-language...) ▼ Traducir esta página  
21 dic. 2015 - In fact, R is the fastest-growing language on StackOverflow in terms of the number of questions asked: The chart above was created – in R, ...

# Índice

---

- Introducción
- **R como una calculadora**
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- Explorando un DataFrame
- Gráficos
- Escribiendo funciones
- Librerías de R



# R como una calculadora

---

- La consola de R funciona como una calculadora
- R tiene los siguientes operadores aritméticos:

Operator	Description
<code>+</code>	addition
<code>-</code>	subtraction
<code>*</code>	multiplication
<code>/</code>	division
<code>^</code> or <code>**</code>	exponentiation
<code>x %% y</code>	modulus (x mod y) 5%%2 is 1
<code>x %/% y</code>	integer division 5%/%2 is 2

`sqrt()` -> raíz cuadrada / `abs()` -> valor absoluto



# Variables

---

- Permiten utilizar un nombre para almacenar un valor
- Se utiliza el operador de asignación `<-`
- Un nombre en mayúsculas no es lo mismo que en minúsculas (R es case sensitive)

```
> x <- 1  
> |
```

# Tipos de Datos

---

- R tiene 5 tipos de datos:
  - Valores **decimales** como 4,5
  - Números **enteros** como por ejemplo 4
    - Es posible especificar que un valor es entero con el sufijo L
  - Números **complejos** como  $(5 + 3i)$
  - Valores **booleanos** (TRUE o FALSE)
    - Se puede abreviar a T o F
  - Valores de **texto**
    - Se utilizan las comillas para indicar que un valor es un texto



# Averiguando el tipo de una variable

---

- La función **class()** informa del tipo de dato de una variable

```
> a <- 4.5  
> b <- 4L  
> c <- (5 + 3i)  
> d <- TRUE  
> e <- "VALOR"
```

```
> print(a)  
[1] 4.5  
> print(b)  
[1] 4  
> print(c)  
[1] 5+3i  
> print(d)  
[1] TRUE  
> print(c)  
[1] 5+3i
```

```
> class(a)  
[1] "numeric"  
> class(b)  
[1] "integer"  
> class(c)  
[1] "complex"  
> class(d)  
[1] "logical"  
> class(e)  
[1] "character"
```

# Conversión de tipos

---

- Los datos se pueden cambiar de tipo de forma explícita utilizando las funciones **as.\*()**

```
> x <- 0:6
> x
[1] 0 1 2 3 4 5 6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```



# Índice

---

- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- Explorando un DataFrame
- Gráficos
- Escribiendo funciones
- Librerías de R



# Vectores

---

- La estructura de datos más básica con la que opera R es un **vector**
- Un vector sólo puede contener datos del mismo tipo



# Crear un vector

- El operador : se utiliza para crear vectores con secuencias de números

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

# Crear un vector

---

- La función **c()** se utiliza para crear un vector con una lista de elementos

```
> x <- 1:5
>
> x
[1] 1 2 3 4 5
>
> x <- c(1, 2, 3, 4, 5)
> x
[1] 1 2 3 4 5
```

# Crear un vector

- La función **seq()** se utiliza para crear secuencias de números
- Pero si necesitamos un vector con una repetición de valores utilizamos **rep()**

```
> seq(from = 10, to = 20, by = 2)
[1] 10 12 14 16 18 20
>
> rep("A", 100)
 [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[17] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[33] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[49] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[65] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[81] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[97] "A" "A" "A" "A"
> |
```

# Longitud de un vector

---

- La función **length()** permite contar los elementos de un vector

```
> x <- 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
>
> length(x)
[1] 10
>
```

# Operar con un vector

---

- En R las operaciones son vectorizadas
- Si se realiza una operación con sobre vector, la operación se aplica a todos los elementos

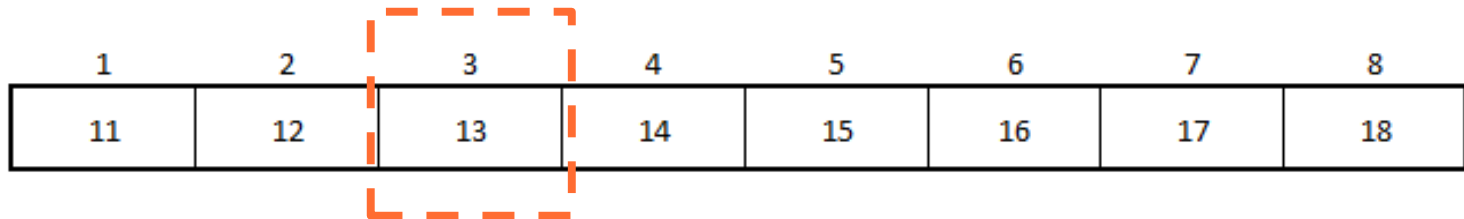
```
> x
[1] 1 2 3 4 5
>
> x + 1
[1] 2 3 4 5 6
>
> x * 2
[1] 2 4 6 8 10
>
> x == 4
[1] FALSE FALSE FALSE  TRUE  FALSE
>
> x >= 4
[1] FALSE FALSE FALSE  TRUE   TRUE
>
```



# Seleccionando elementos en un vector

---

- Para seleccionar un elemento del vector a través de su índice se utilizan los símbolos [ y ]



1	2	3	4	5	6	7	8
11	12	13	14	15	16	17	18



# Seleccionando elementos en un vector

---

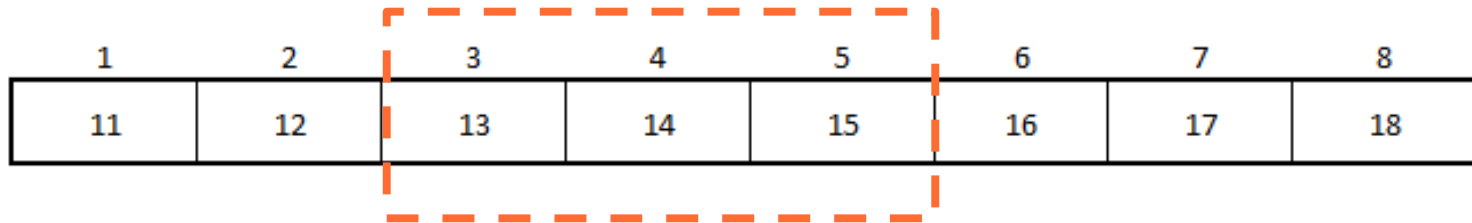
- Para seleccionar un elemento del vector a través de su índice se utilizan los símbolos [ y ]

```
> x <- 11:18  
>  
> x  
[1] 11 12 13 14 15 16 17 18  
>  
> x[3]  
[1] 13
```

# Seleccionando elementos en un vector

---

- Es posible seleccionar más de un elemento...



# Seleccionando elementos en un vector

---

- Es posible seleccionar más de un elemento...

```
> x <- c(11:18)
> x
[1] 11 12 13 14 15 16 17 18
>
> x [ 3:5 ]
[1] 13 14 15
> |
```

# Seleccionando elementos en un vector

---

- En R se puede seleccionar elementos de un vector a través de otro vector de valores lógicos

1	2	3	4	5	6	7	99
T	T	F	F	F	T	T	T
1	2				6	7	99

# Seleccionando elementos en un vector

---

- En R se puede seleccionar elementos de un vector a través de otro vector de valores lógicos

```
> x <- c(1, 2, 3, 4, 5, 5, 6, 7, 99)
>
> y <- c(T, T, F, F, F, T, T, T, T)
>
> x[y]
[1] 1 2 5 6 7 99
>
```

# Reemplazando elementos en un vector

---

- Para reemplazar una posición concreta de un vector se utiliza el operador de asignación ( $\leftarrow$ ) junto con el de selección `[]`

```
> x <- 1:5
> x
[1] 1 2 3 4 5
>
> x[1] <- 99
> x
[1] 99 2 3 4 5
> |
```

# Vectores con nombres

---

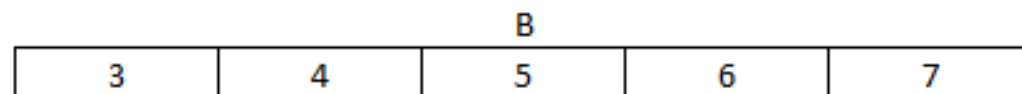
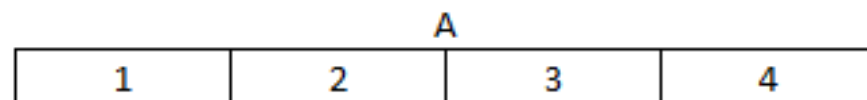
- En R es posible asignar un nombre a cada elemento del vector y seleccionarlos a través de su nombre

```
> x <- 1:3
> x
[1] 1 2 3
> names(x) <- c("Uno", "Dos", "Tres")
> x
  Uno  Dos Tres
   1    2   3
> names(x)
[1] "Uno" "Dos" "Tres"
> x["Uno"]
Uno
  1
> unname(x)
[1] 1 2 3
> |
```

# Operaciones en conjuntos

- R tiene una serie de operaciones aplicables a conjuntos de vectores

```
> a <- 1:4  
> b <- 3:7  
> union(a,b)  
[1] 1 2 3 4 5 6 7  
> intersect(a,b)  
[1] 3 4  
> setdiff(a,b)  
[1] 1 2  
> setdiff(b,a)  
[1] 5 6 7  
> 2 %in% a  
[1] TRUE
```





# Ordenando un vector

---

- La función **sort()** se utiliza para ordenar los elementos de un vector

```
> x <- c(3, 1, 3, 4, 5, 2, 3, 2, 1, 3, 4, 5)
>
> x
[1] 3 1 3 4 5 2 3 2 1 3 4 5
>
> sort(x)
[1] 1 1 2 2 3 3 3 3 4 4 5 5
>
> sort(x, decreasing = T)
[1] 5 5 4 4 3 3 3 3 2 2 1 1
>
```

# Contando elementos en un vector

---

- La función **table()** se utiliza para contar los elementos de un vector

```
> x <- c(3, 1, 3, 4, 5, 2, 3, 2, 1, 3, 4, 5)
> x
[1] 3 1 3 4 5 2 3 2 1 3 4 5
>
> sort(x)
[1] 1 1 2 2 3 3 3 3 4 4 5 5
>
> table(x)
x
1 2 3 4 5
2 2 4 2 2
>
```

# Números especiales

---

- **Inf** representa infinito / **-Inf** representa menos infinito
- **NaN** representa un valor indefinido (Not a Number)
- **NA** representa un valor inexistente

```
> 1 / 0
[1] Inf
> 0 / 0
[1] NaN
> c(1, NA, 2)
[1] 1 NA 2
> |
```

# Números especiales

---

- La función **is.na()** se utiliza para ver los datos que son **NA** en un vector
- De la misma forma que se puede utilizar **is.nan()** para los datos **NaN**

```
>  
> x <- c(1, NA, 0/0)  
> x  
[1] 1 NA NaN  
> is.na(x)  
[1] FALSE TRUE TRUE  
> is.nan(x)  
[1] FALSE FALSE TRUE  
>
```

# Eliminado valores NA de un vector

---

- Observa cómo se puede utilizar la función **is.na()** para eliminar valores NA de un vector

```
> x <- c(1, 2, NA, 4, NA, 5)
> x
[1] 1 2 NA 4 NA 5
> bad <- is.na(x)
> bad
[1] FALSE FALSE  TRUE FALSE  TRUE FALSE
> x[!bad]
[1] 1 2 4 5
> |
```

*\*Recuerda: selección de los elementos vector, haciendo uso de un vector lógico*



# Matrices

- Las matrices son un conjunto de vectores del mismo tamaño
- La dimensión de una matriz viene determinada por el número de filas y columnas que tiene

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & & & A_{2n} \\ \vdots & & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix}$$

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
> class(m)
[1] "matrix"
>
> dim(m)
[1] 2 3
```

# Seleccionando elementos en una matriz

- Para seleccionar elementos en una matriz se tiene que especificar la fila y la columna del elemento:  
**matriz[fila, columna]**

```
>  
> m <- matrix(1:6, 2, 3)  
>  
> m  
      [,1] [,2] [,3]  
[1,]    1    3    5  
[2,]    2    4    6  
>  
> m[1,2]  
[1] 3  
>
```

```
> m[1, 2:3]  
[1] 3 5  
>  
> m[1:2, 3]  
[1] 5 6  
>  
> m[2]  
[1] 2  
>  
> m[-2]  
[1] 1 3 4 5 6  
>
```

# Cálculos en matrices

- Es posible realizar cálculos en matrices

```
> mat <- matrix(c(1,2,3,4), nrow = 2, ncol = 2, byrow = TRUE)
> mat
      [,1] [,2]
[1,]    1    2
[2,]    3    4
> class(mat)
[1] "matrix"
> mat + mat
      [,1] [,2]
[1,]    2    4
[2,]    6    8
> mat * mat
      [,1] [,2]
[1,]    1    4
[2,]    9   16
> mat %% mat # Matrix multiplication
      [,1] [,2]
[1,]    7   10
[2,]   15   22
```

- El parámetro **byrow** = TRUE, indica que la matriz se llena por filas, por defecto es FALSE, es decir, que se va rellenando por columnas
- Dos matrices se pueden **multiplicar** cuando el número de columnas de la primera es igual al número de filas de la segunda (2x3 - 3x2)



# Cálculos en matrices

---

- Se utiliza **t()** para transponer una matriz

```
> m=matrix(1:12,3,4)
>
> m
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
> t(m)
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```



# Uniendo varios vectores en una matriz

- Las funciones **rbind()** y **cbind()** permiten unir varios vectores para formar una matriz

```
> cbind(1:10, 101:110)
      [,1] [,2]
[1,]    1 101
[2,]    2 102
[3,]    3 103
[4,]    4 104
[5,]    5 105
[6,]    6 106
[7,]    7 107
[8,]    8 108
[9,]    9 109
[10,]   10 110
>
```

```
> rbind(1:5, 101:105, rep(100, 5))
      [,1] [,2] [,3] [,4] [,5]
[1,]     1     2     3     4     5
[2,]   101   102   103   104   105
[3,]   100   100   100   100   100
> |
```

# Listas

---

- Las listas es una clase especial de vector que pueden contener elementos de distinto tipo y tamaño

```
> x <- list(1, "a", TRUE, 1 + 4i)
> x
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i

> class(x)
[1] "list"
```



# Seleccionando elementos en una lista

---

- Para seleccionar elementos en una lista se especifica un elemento mediante **[[n]]**

```
> l <- list(1:4, c("a","b"))
> l
[[1]]
[1] 1 2 3 4

[[2]]
[1] "a" "b"

> l[[1]]
[1] 1 2 3 4
> l[[2]]
[1] "a" "b"
> l[[c(1,3)]]
[1] 3
> l[[c(2,1)]]
[1] "a"
> |
```

# Listas con nombre

---

- Como en el caso de los vectores, es posible dar un nombre a cada elemento de la lista
- Se usa el símbolo \$ para acceder al elemento a través del nombre

```
> l = list("Primera"=1:3, "Segunda" = c("a", "b", "c"))
> l
$Primera
[1] 1 2 3

$Segunda
[1] "a" "b" "c"

> l$Primera
[1] 1 2 3
> l$Segunda[2]
[1] "b"
> |
```

# Convirtiendo listas en vectores

---

- Una lista se puede convertir en un vector mediante la función **unlist()**

```
> l <- list(1:3, 5:8)
> l
[[1]]
[1] 1 2 3

[[2]]
[1] 5 6 7 8

> unlist(l)
[1] 1 2 3 5 6 7 8
```



# Factores

---

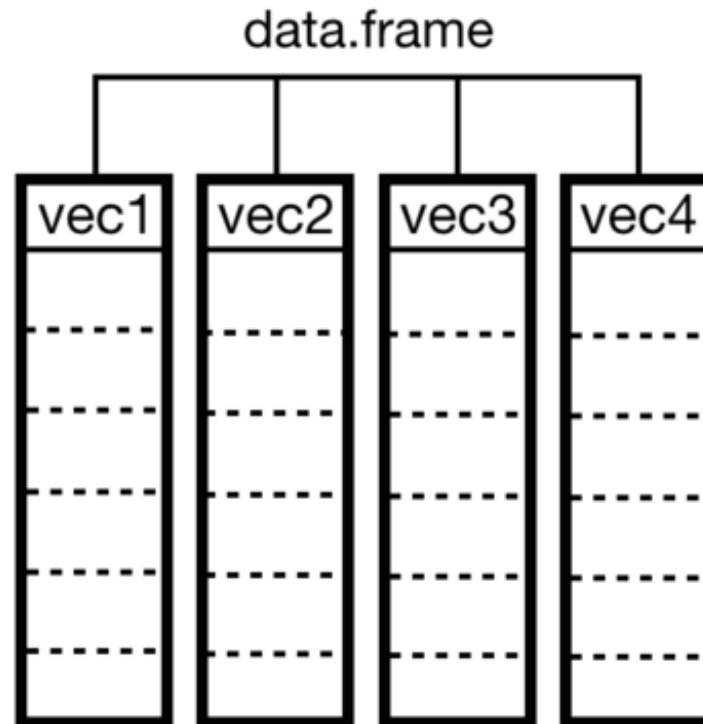
- Es un vector utilizado para representar categorías de otro vector
- Con **unclass()** puede verse la estructura interna del factor

```
> x <- c("Yes", "No", "Yes")
> x
[1] "Yes" "No"  "Yes"
> f <- factor(x)
> f
[1] Yes No  Yes
Levels: No Yes
> unclass(f)
[1] 2 1 2
attr(,"levels")
[1] "No" "Yes"
> |
```

# DataFrames

---

- Es una lista de vectores de igual tamaño y tipos distintos, que R visualiza como una tabla.





# DataFrames

---

- La función **data.frame()** es la encargada de crear un DataFrame
- Cada argumento se convierte en una columna

```
> n <- c(2, 3, 5)
> b <- c(T, F, T)
> s <- c("aa", "bb", "cc")
> df <- data.frame(n, s, b)
> df
  n  s    b
1 2 aa TRUE
2 3 bb FALSE
3 5 cc  TRUE
> |
```

# Obtener columnas de un DataFrame

---

- El operador `$` se utiliza para obtener todos los valores de una columna

```
> df$n
[1] 2 3 5
>
> df$s
[1] aa bb cc
Levels: aa bb cc
>
> df$b
[1] TRUE FALSE TRUE
>
```

¿Qué devolverá la sentencia `> df$b[2]` ?

# Crear columnas de un DataFrame

---

- Este operador se utiliza también para crear columnas nuevas ...

```
> df$id <- 1:3
> df
  n  s      b id
1 2 aa  TRUE  1
2 3 bb FALSE  2
3 5 cc  TRUE  3
>
```

# Borrar columnas de un DataFrame

---

- .. o borrarlas

```
> df$id <- NULL
> df
  n  s    b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
>
```

Para borrar un dataframe se utiliza el comando **rm()**  
`rm(nombre_del_dataframe)`

# Inserta filas en un DataFrame

---

- Como insertar una fila en un dataframe

```
> df
  n  s    b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
> df.insertar <- data.frame(n=10,s="dd",b=F)
> df <- rbind(df,df.insertar)
> df
  n  s    b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
4 10 dd FALSE
> |
```



# Nombres de un DataFrame

---

- La función **names()** devuelve el nombre de las columnas ...
- ... y la función **row.names()** el nombre de las filas

```
> df
  n  s    b
1 2 aa TRUE
2 3 bb FALSE
3 5 cc TRUE
> names(df)
[1] "n" "s" "b"
>
> row.names(df)
[1] "1" "2" "3"
> |
```

# Seleccionando columnas

---

- En R es posible seleccionar columnas a través de su nombre o su número de orden

```
> df["s"]
```

```
  s  
1 aa  
2 bb  
3 cc
```

```
>
```

```
> df[1]
```

```
  n  
1 2  
2 3  
3 5  
> |
```

```
> df[c("s", "b")]
```

```
  s      b  
1 aa  TRUE  
2 bb FALSE  
3 cc  TRUE
```

```
>
```

```
> df[1:2]
```

```
  n  s  
1 2 aa  
2 3 bb  
3 5 cc  
> |
```

# Filtrando filas en un DataFrame

---

- De la misma forma, las filas se filtran a través de su nombre o su número de orden

```
> row.names(df) <- c("Uno", "Dos", "Tres")
> df
      n  s    b
Uno   2 aa  TRUE
Dos   3 bb FALSE
Tres  5 cc  TRUE
>
> df["Uno",]
      n  s    b
Uno   2 aa  TRUE
> df[1:2, ]
      n  s    b
Uno   2 aa  TRUE
Dos   3 bb FALSE
> |
```



# Filtrando filas en un DataFrame

---

- También se puede utilizar valores lógicos para filtrar las filas de un DataFrame...

```
> df
  n  s    b
Uno  2 aa  TRUE
Dos  3 bb FALSE
Tres 5 cc  TRUE
>
> df[c(T, F, T), ]
  n  s    b
Uno  2 aa  TRUE
Tres 5 cc  TRUE
>
```

# Filtrando filas con una condición

---

- ... por lo que se pueden utilizar condiciones que devuelvan vectores de valores lógicos

```
> df
      n  s    b
Uno   2 aa  TRUE
Dos   3 bb FALSE
Tres  5 cc  TRUE
>
> df[df$n > 2, ]
      n  s    b
Dos   3 bb FALSE
Tres  5 cc  TRUE
>
> df[df$n > 2 & b, ]
      n  s    b
Tres  5 cc  TRUE
>
```

# Filtrando filas y columnas

---

- Es posible filtrar al mismo tiempo filas y columnas

```
> df[df$n > 2, c("s", "n")]  
      s  n  
Dos  bb  3  
Tres cc  5  
>
```

# Índice

---

- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- **Funciones estadísticas**
- Importando y exportando datos
- Explorando un DataFrame
- Gráficos
- Escribiendo funciones
- Librerías de R



# Estadística descriptiva

---

- **length()** devuelve la longitud de un vector
- **min()** el valor mínimo
- **max()** el valor máximo

```
> ages = c(25, 22, 18, 20, 22)
> ages
[1] 25 22 18 20 22
> min(ages)
[1] 18
> max(ages)
[1] 25
> length(ages)
[1] 5
```

# Estadística descriptiva

---

- **mean()** devuelve la media de los valores de un vector
- **median()** la mediana
- **sd()** la desviación típica
- **var()** la varianza

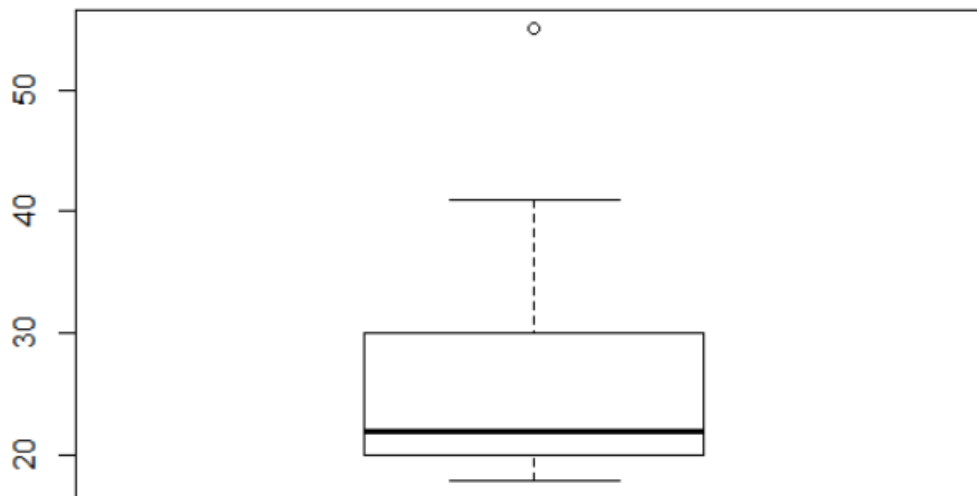
```
> ages <- c(25, 22, 18, 20, 22)
> mean(ages)
[1] 21.4
> median(ages)
[1] 22
> sd(ages)
[1] 2.607681
> var(ages)
[1] 6.8
```



# Estadística descriptiva

- **summary()** nos muestra la distribución de datos

```
> ages <- c(19, 25, 22, 18, 20, 22, 30, 22, 55, 41)
> summary(ages)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 18.00  20.50   22.00   27.40  28.75   55.00
>
```



# Distribuciones de probabilidad en R

---

- R viene con una serie de distribuciones probabilísticas estándar
- Permite generar números aleatorios según una determinada distribución
- Las distribuciones sirven para modelizar fenómenos y son muy útiles en el Machine Learning
- Utiliza la función **help(distributions)** para verlas

```
> help(Distributions)  
>
```

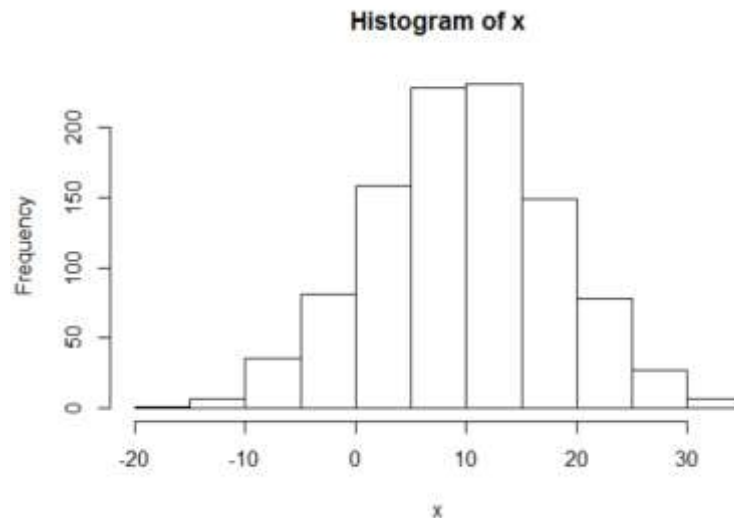




# Distribución normal

- Para generar números aleatorios según la distribución normal se utiliza **rnorm()**

```
> x <- rnorm(1000, mean = 10, sd = 8)
> x[1:10]
[1]  0.03286958  7.26519636 10.50557568  4.54998383  2.41240307
[6] 14.75232535 -4.17608221 -7.72921943  5.31683759  5.43566948
>
```



# Sampling

---

- Para obtener un ejemplo de un vector de números se utiliza la función **sample()**

```
>  
> set.seed(10)  
>  
> sample(1:10, 2, prob = rep(0.1, 10), replace = F)  
[1] 7 4  
>  
> sample(1:10, 20, prob = c(0.25, 0.25, rep(0.05, 8)), replace = T)  
[1] 1 6 2 2 2 2 5 1 5 3 2 3 1 1 2 2 1 8 8 5  
>
```

# Creando DataFrames con valores aleatorios

---

- En R es posible crear un DataFrame utilizando funciones aleatorias()

```
> data <- data.frame(categoria = rep(c("A", "B", "C"), 100),  
+                     nota = sample(1:10, 100, replace = T),  
+                     dato = rnorm(100, mean = 100, sd=20))  
>  
> data[1:10,]  
  categoria nota      dato  
1         A    6  91.98725  
2         B    4  93.30887  
3         C    5 127.35908  
4         A    7 142.75534  
5         B    1 110.11639  
6         C    3 115.72685  
7         A    3  81.95576  
8         B    3 110.65794  
9         C    7  87.08211  
10        A    5 105.81975  
>
```

# Índice

---

- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- **Importando y exportando datos**
- Explorando un DataFrame
- Gráficos
- Escribiendo funciones
- Librerías de R



# Leyendo datos tabulares

---

- La función principal para leer datos tabulares en R es **read.table()** y **read.csv()**
- Antes de leer un fichero es preciso saber el directorio de trabajo **getwd()**. También puede establecer con la función **setwd("...")**
- Para ver su contenido **dir()**.

```
> getwd()
[1] "C:/Users/se47351/Documents"
>
> data <- read.csv("iris.data", header = F)
> data <- read.table("iris.data", header = F, sep = ",")
>
```

# Leyendo datos desde Internet

---

- La función **url()** permite leer ficheros que están publicados en un servidor web
- Se utiliza junto con las funciones **read.\*()**

```
> data <- read.csv(url("http://www.sharpsightlabs.com/wp-content/uploads/2016/04/unknown_fxn_data.txt"))
> data[1:10,]
  input_var target_var
1    -2.75  -0.2241552
2    -2.55  -0.8316938
3    -2.35  -0.5426280
4    -2.15  -0.7246004
5    -1.95  -1.0263724
6    -1.75  -1.1435482
7    -1.55  -1.1483044
8    -1.35  -1.2473150
9    -1.15  -0.7883597
10   -0.95  -0.8461753
> |
```



# Escribiendo datos tabulares

---

- Las funciones **write.csv()** o **write.table()** se utilizan escribir datos tabulares en un fichero

```
>  
> write.table(data, file = "data.txt",  
+             sep = "|", row.names = F, col.names = F)  
>  
> write.csv(data, file = "data.csv",  
+           row.names = F)  
>
```

# Ficheros de texto

---

- La función **readLines()** se utiliza para leer el contenido de un fichero de texto

```
> data2 <- readLines("iris.data")
> data2[1:10]
[1] "5.1,3.5,1.4,0.2,Iris-setosa" "4.9,3.0,1.4,0.2,Iris-setosa"
[3] "4.7,3.2,1.3,0.2,Iris-setosa" "4.6,3.1,1.5,0.2,Iris-setosa"
[5] "5.0,3.6,1.4,0.2,Iris-setosa" "5.4,3.9,1.7,0.4,Iris-setosa"
[7] "4.6,3.4,1.4,0.3,Iris-setosa" "5.0,3.4,1.5,0.2,Iris-setosa"
[9] "4.4,2.9,1.4,0.2,Iris-setosa" "4.9,3.1,1.5,0.1,Iris-setosa"
> |
```

- **writeLines()** se utiliza para escribir un vector en un fichero de texto

```
> writeLines(data2,con="data2.txt")
> |
```





# Otros formatos de fichero

- Existen librerías para una gran diversidad de formatos de ficheros



A screenshot of a Google search interface. The search bar contains the text "R package import excel". Below the search bar, the "Web" tab is selected, and the search results show approximately 114,000,000 results in 0.55 seconds. The top result is titled "R Data Import/Export - The Comprehensive R Archive Network" and includes a link to [cran.r-project.org/doc/manuals/r.../R-data.html](http://cran.r-project.org/doc/manuals/r.../R-data.html). The snippet below the link describes the ease of importing data into R, mentioning text files and Excel spreadsheets.

Google R package import excel

Web Vídeos Imágenes Noticias Maps Más ▾ Herramientas de búsqueda

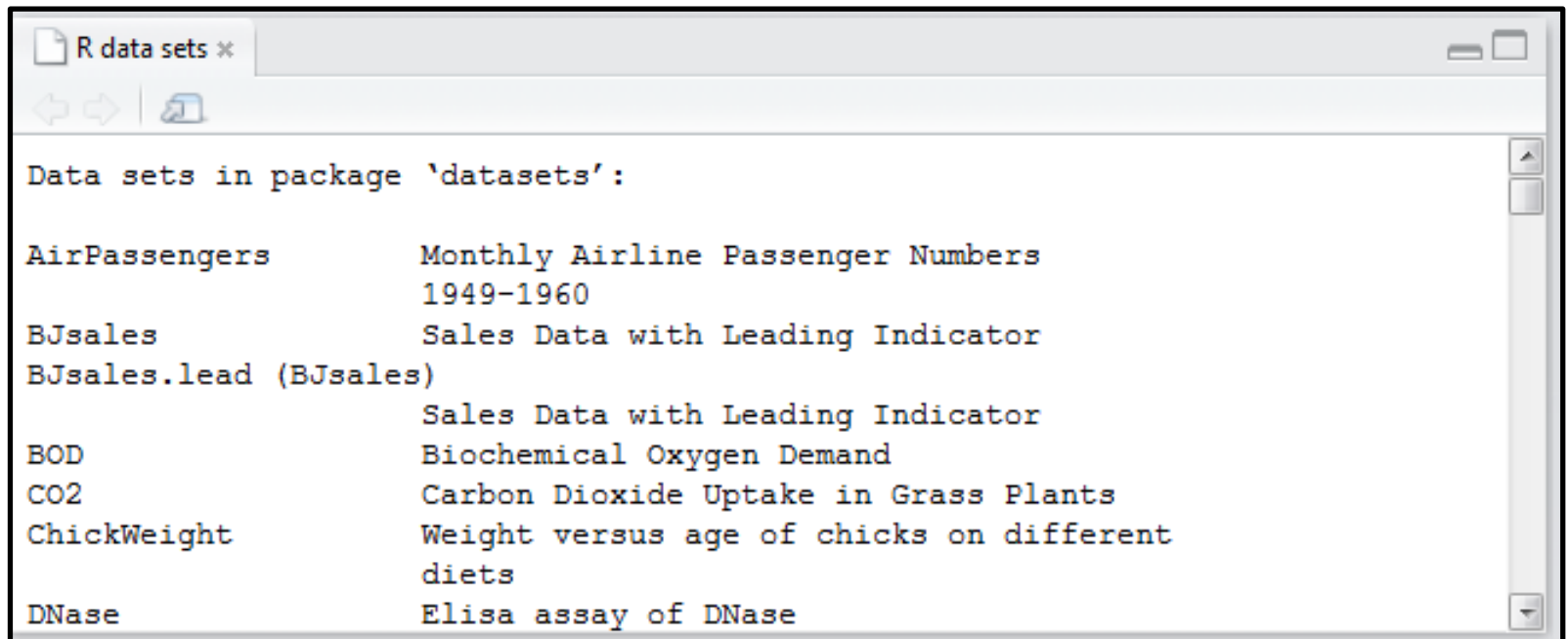
Aproximadamente 114.000.000 resultados (0,55 segundos)

**R Data Import/Export - The Comprehensive R Archive Network**  
[cran.r-project.org/doc/manuals/r.../R-data.html](http://cran.r-project.org/doc/manuals/r.../R-data.html) ▾ Traducir esta página  
(See the rJava package from CRAN and the SJava, RSPerl and RSPython ... The easiest form of data to import into R is a simple text file, and this will often be ... such files directly from R. For Excel spreadsheets, the available methods are ...  
[Acknowledgements](#) - [1 Introduction](#) - [2 Spreadsheet-like data](#)

# Datasets de ejemplo

---

- R viene con un conjunto de datasets de ejemplo con las que se puede experimentar



```
R data sets *  
Data sets in package 'datasets':  
  
AirPassengers      Monthly Airline Passenger Numbers  
                    1949-1960  
BJsales            Sales Data with Leading Indicator  
BJsales.lead (BJsales)  
                    Sales Data with Leading Indicator  
BOD                Biochemical Oxygen Demand  
CO2                Carbon Dioxide Uptake in Grass Plants  
ChickWeight        Weight versus age of chicks on different  
                    diets  
DNase              Elisa assay of DNase
```

# Datasets de ejemplo

---

- Para listar los datasets se utiliza la función **data()**
- Para cargarlo en memoria **data(DataSet)**
- Con **rm(DataSet)** se descarga de memoria

```
> data()  
>  
> data(AirPassengers)  
>
```

# Índice

---

- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- **Explorando un DataFrame**
- Gráficos
- Escribiendo funciones
- Librerías de R



# Explorando un DataFrame

- Vamos a explorar el DataFrame “**mtcars**”
- Este dataset proviene de los datos de ejemplo de R
- Con ? Se obtiene una descripción completa

```
> ?mtcars
> data(mtcars)
>
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4

# Explorando un DataFrame

---

- **dim()** devuelve las dimensiones del DataSets
- **ncol()** el número de columnas y ..
- **nrow()** el número de filas.

```
> dim(mtcars)
[1] 32 11
>
> ncol(mtcars)
[1] 11
>
> nrow(mtcars)
[1] 32
>
```

# Explorando un DataFrame

---

- **names()** devuelve el nombre de las columnas

```
> names(mtcars)
[1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"
[10] "gear" "carb"
>
```

# Explorando un DataFrame

---

- **head()** devuelve las primeras columnas ...

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
>
```



# Explorando un DataFrame

---

- **tail()** devuelve las ultimas columnas ...

```
> tail(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.7	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.9	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.5	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.5	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.6	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.6	1	1	4	2

```
>
```

# Explorando un DataFrame

- **summary()** devuelve la distribución estadística de las columnas ...

```
> summary(mtcars)
```

mpg	cyl	disp	hp
Min. :10.40	Min. :4.000	Min. : 71.1	Min. : 52.0
1st Qu.:15.43	1st Qu.:4.000	1st Qu.:120.8	1st Qu.: 96.5
Median :19.20	Median :6.000	Median :196.3	Median :123.0
Mean :20.09	Mean :6.188	Mean :230.7	Mean :146.7
3rd Qu.:22.80	3rd Qu.:8.000	3rd Qu.:326.0	3rd Qu.:180.0
Max. :33.90	Max. :8.000	Max. :472.0	Max. :335.0

drat	wt	qsec	vs
Min. :2.760	Min. :1.513	Min. :14.50	Min. :0.0000
1st Qu.:3.080	1st Qu.:2.581	1st Qu.:16.89	1st Qu.:0.0000
Median :3.695	Median :3.325	Median :17.71	Median :0.0000
Mean :3.597	Mean :3.217	Mean :17.85	Mean :0.4375
3rd Qu.:3.920	3rd Qu.:3.610	3rd Qu.:18.90	3rd Qu.:1.0000
Max. :4.930	Max. :5.424	Max. :22.90	Max. :1.0000

am	gear	carb
Min. :0.0000	Min. :3.000	Min. :1.000
1st Qu.:0.0000	1st Qu.:3.000	1st Qu.:2.000
Median :0.0000	Median :4.000	Median :2.000
Mean :0.4062	Mean :3.688	Mean :2.812
3rd Qu.:1.0000	3rd Qu.:4.000	3rd Qu.:4.000
Max. :1.0000	Max. :5.000	Max. :8.000

```
> |
```

# Explorando un DataFrame

---

- **str()** imprime para cada columna, el tipo dato y una muestra de los valores que tiene

```
> str(mtcars)
'data.frame': 32 obs. of 11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num   6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num   3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num   2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num   0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num   1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num   4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num   4 4 1 1 2 1 4 2 2 4 ...
>
```



# Explorando un DataFrame

- La función **table()** se utiliza para que cuente los distintos valores de una columna

```
> table(mtcars$cyl)

4  6  8
11 7 14
>
> table(mtcars$cyl, mtcars$disp)

      71.1 75.7 78.7 79 95.1 108 120.1 120.3 121 140.8 145 146.7 160
4       1    1    1  1    1    1    1    1    1    1    0    1    0
6       0    0    0  0    0    0    0    0    0    0    1    0    2
8       0    0    0  0    0    0    0    0    0    0    0    0    0

      167.6 225 258 275.8 301 304 318 350 351 360 400 440 460 472
4         0    0    0    0    0    0    0    0    0    0    0    0    0
6         2    1    1    0    0    0    0    0    0    0    0    0    0
8         0    0    0    3    1    1    1    1    1    2    1    1    1
>
```



# Índice

---

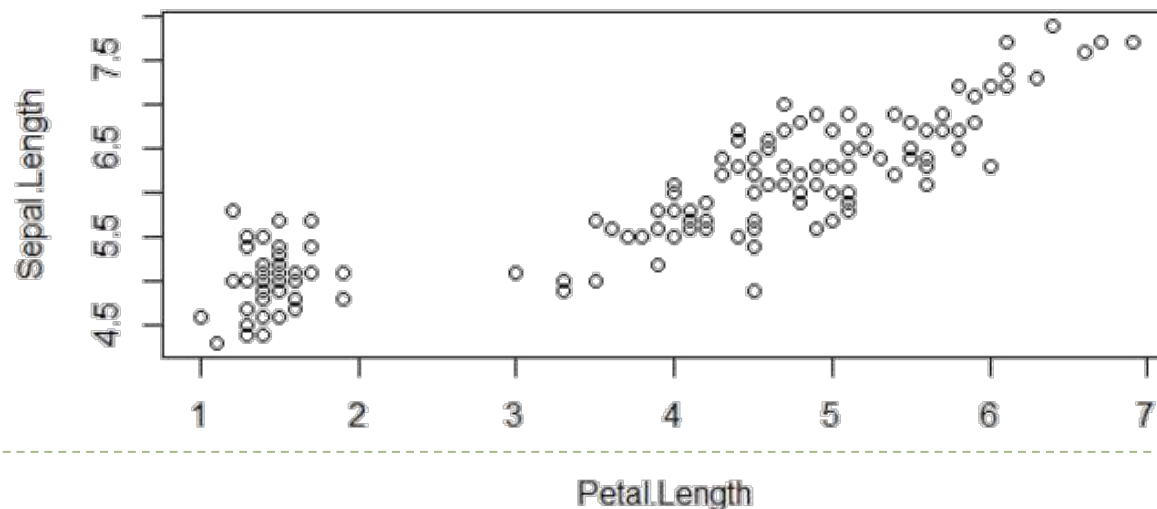
- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- Explorando un DataFrame
- **Gráficos**
- Escribiendo funciones
- Librerías de R



# Gráficos en R – El sistema base

- Es el sistema original de R y no es necesario instalar ningún paquete adicional
- La idea es que se empieza con un lienzo vacío y a partir de ahí se añaden elementos gráficos
- Es el más conveniente para análisis exploratorio de la información

```
> plot(Sepal.Length~Petal.Length, data=iris)  
> |
```



# Gráficos en R

---

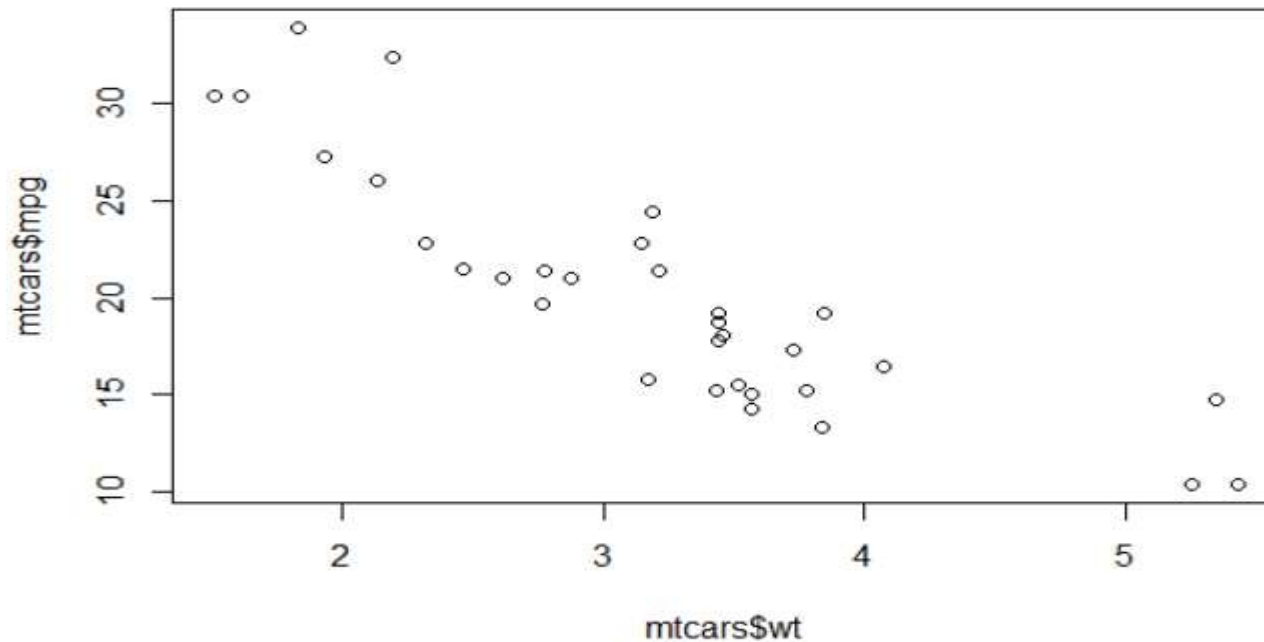
- Gráfico de dispersión (scatter plot)
- Gráfico de líneas
- Gráfico de barras
- Histogramas
- Gráfico de caja (box plot)
- Gráfico de tarta



# Gráfico de Dispersión o Scatter Plot

---

```
> plot(mtcars$wt, mtcars$mpg)
```



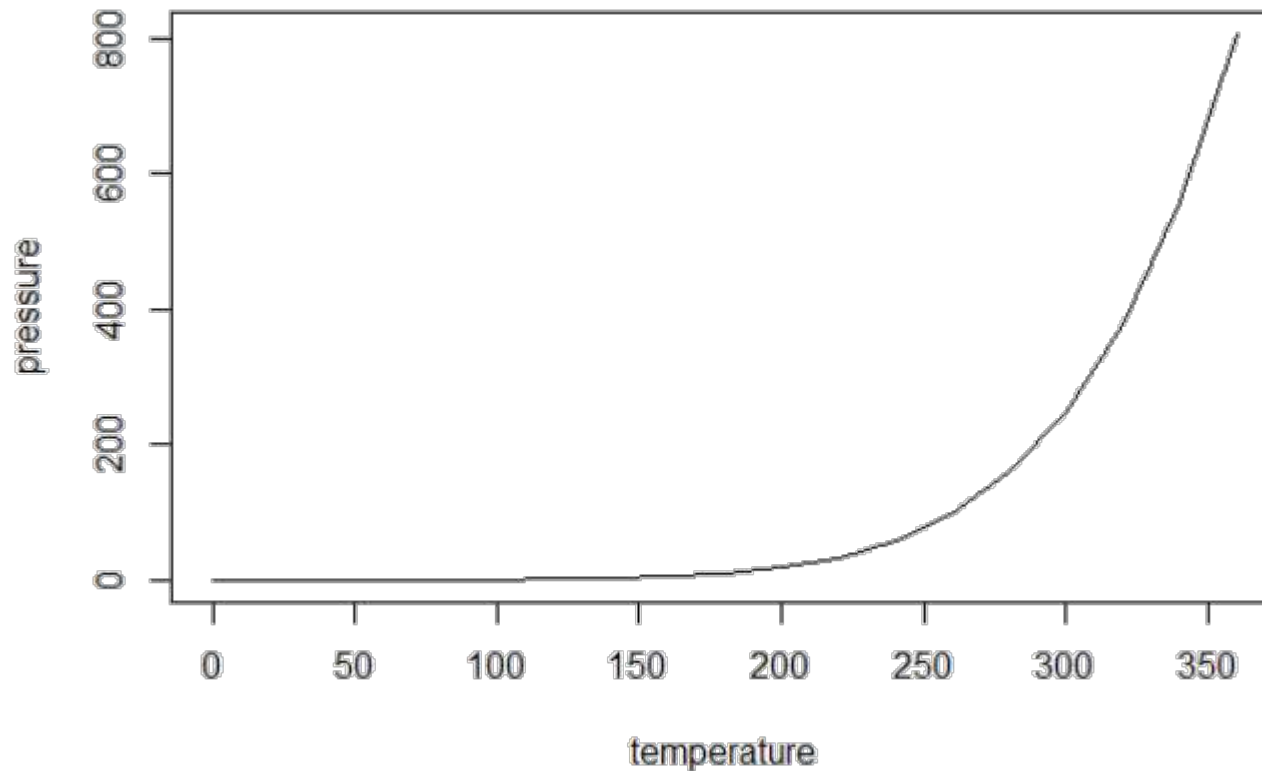
*Sirve para ver el grado de correlación entre dos variables*



# Gráfico de líneas

---

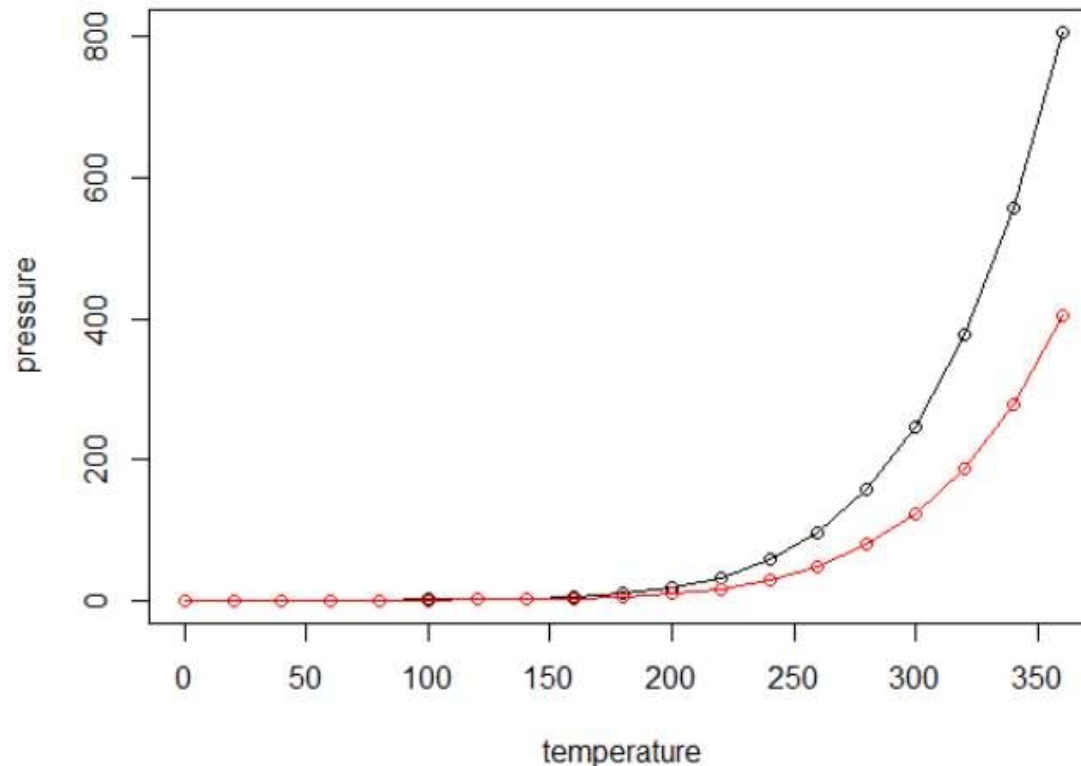
```
> ?pressure  
> plot(pressure, type="l")  
> |
```



# Líneas y puntos al mismo tiempo

```
> plot(pressure, type = "l")  
> points(pressure$temperature, pressure$pressure)  
> lines(pressure$temperature, pressure$pressure / 2, col = "red")  
> points(pressure$temperature, pressure$pressure / 2, col = "red")  
>
```

*Pueden añadirse  
elementos a un gráfico*



# Diagrama de barras

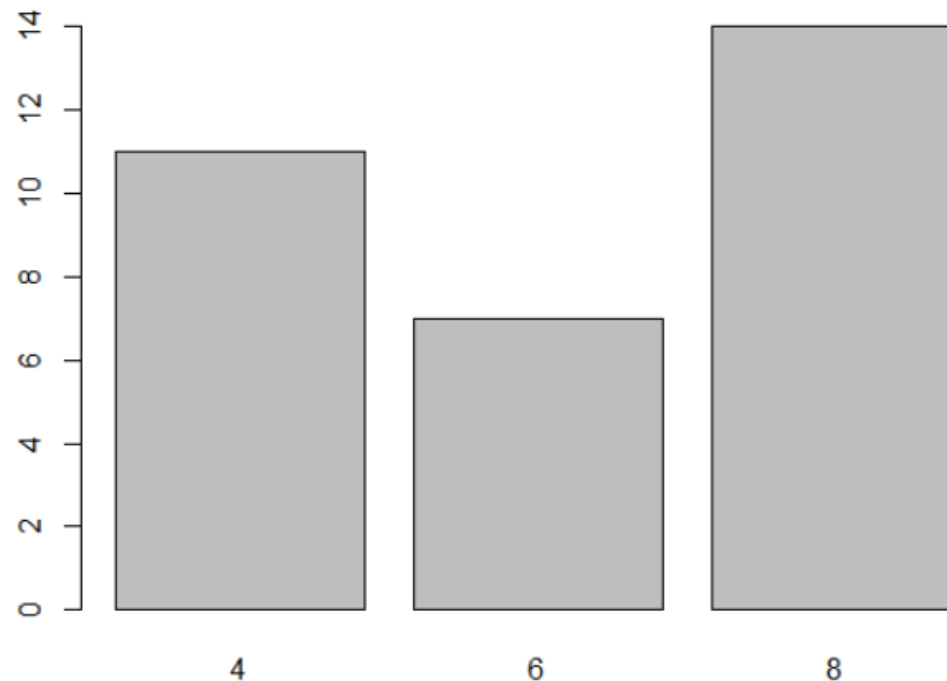
---

```
> table(mtcars$cyl)
```

```
 4  6  8  
11  7 14
```

```
>  
> barplot(table(mtcars$cyl))
```

*Para variables cualitativas*

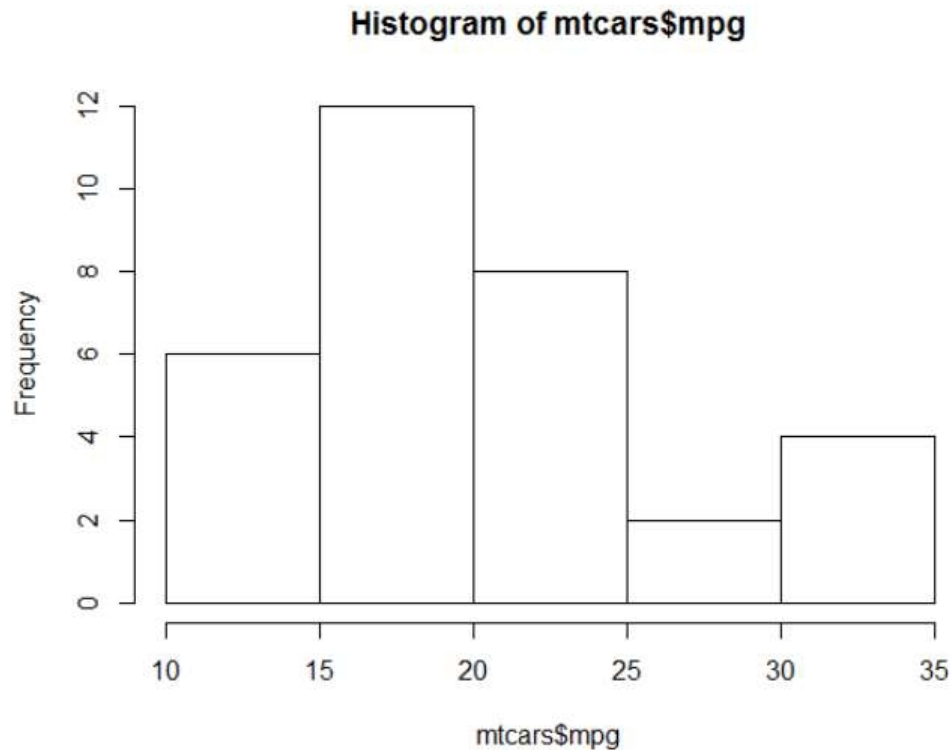


# Histograma

---

```
> hist(mtcars$mpg)  
> |
```

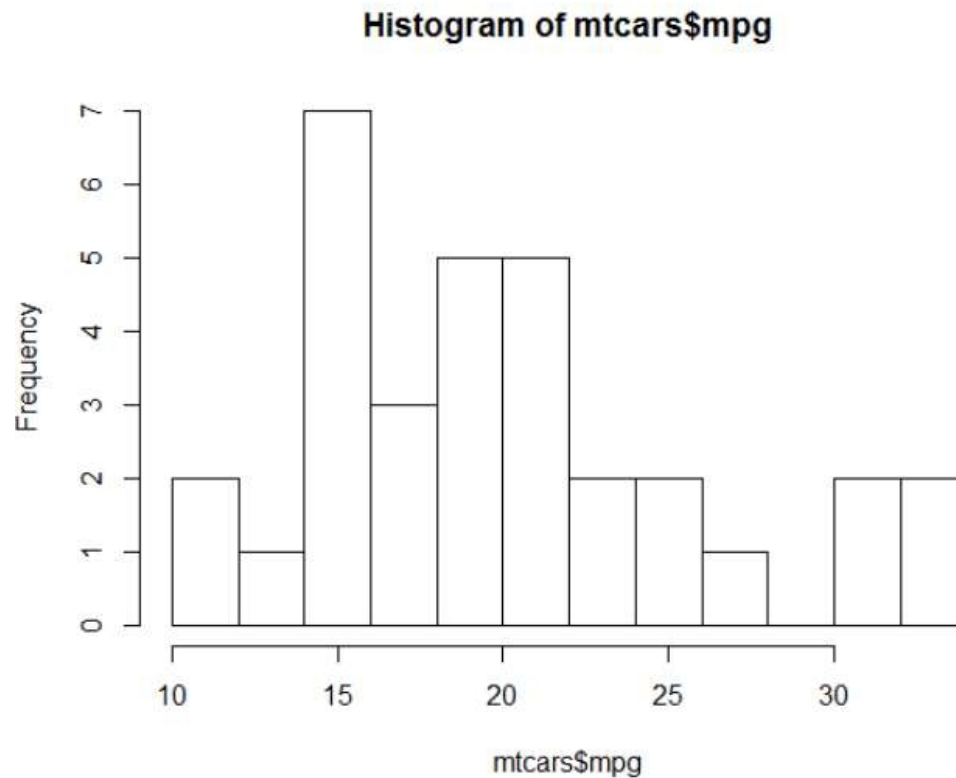
*Ver la distribución de los datos*



# Histograma

```
> hist(mtcars$mpg, breaks = 10)  
> |
```

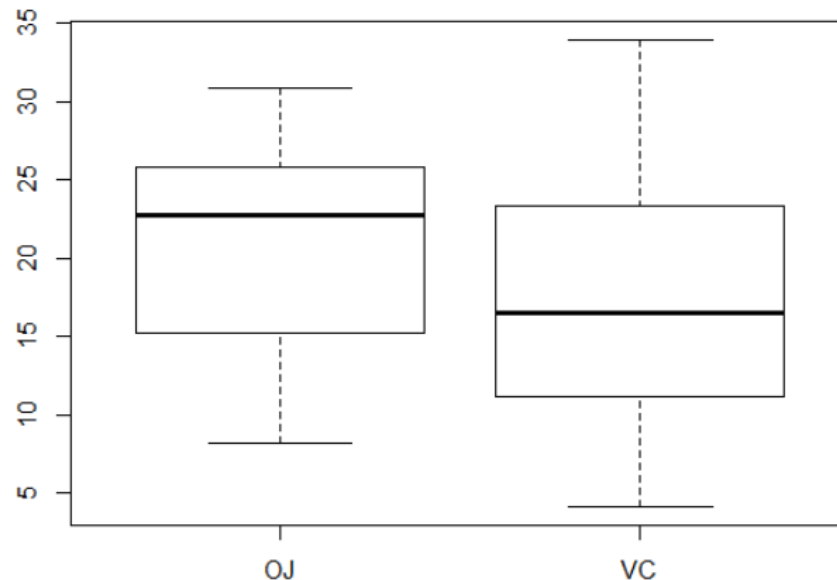
*Con la opción break, se establecen puntos de ruptura*



# Box Plot

```
> head(ToothGrowth)
  len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
4  5.8   VC  0.5
5  6.4   VC  0.5
6 10.0   VC  0.5
> boxplot(len ~ supp, data = ToothGrowth)
```

*Muy útiles para ver como se distribuyen los datos y detectar valores atípicos*

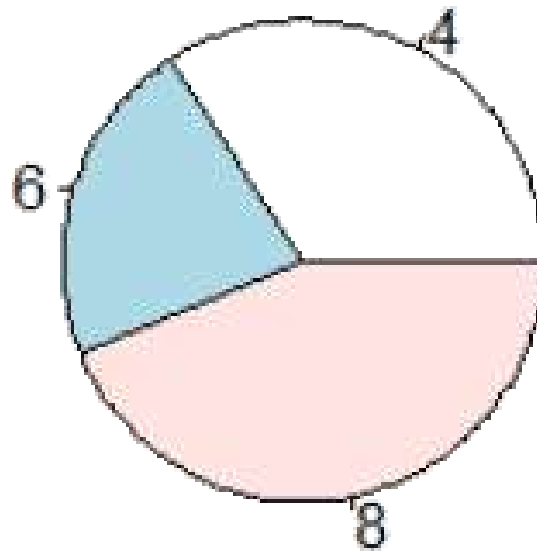


# Gráfico de tarta

```
> cyltable<- table(mtcars$cyl)
> cyltable

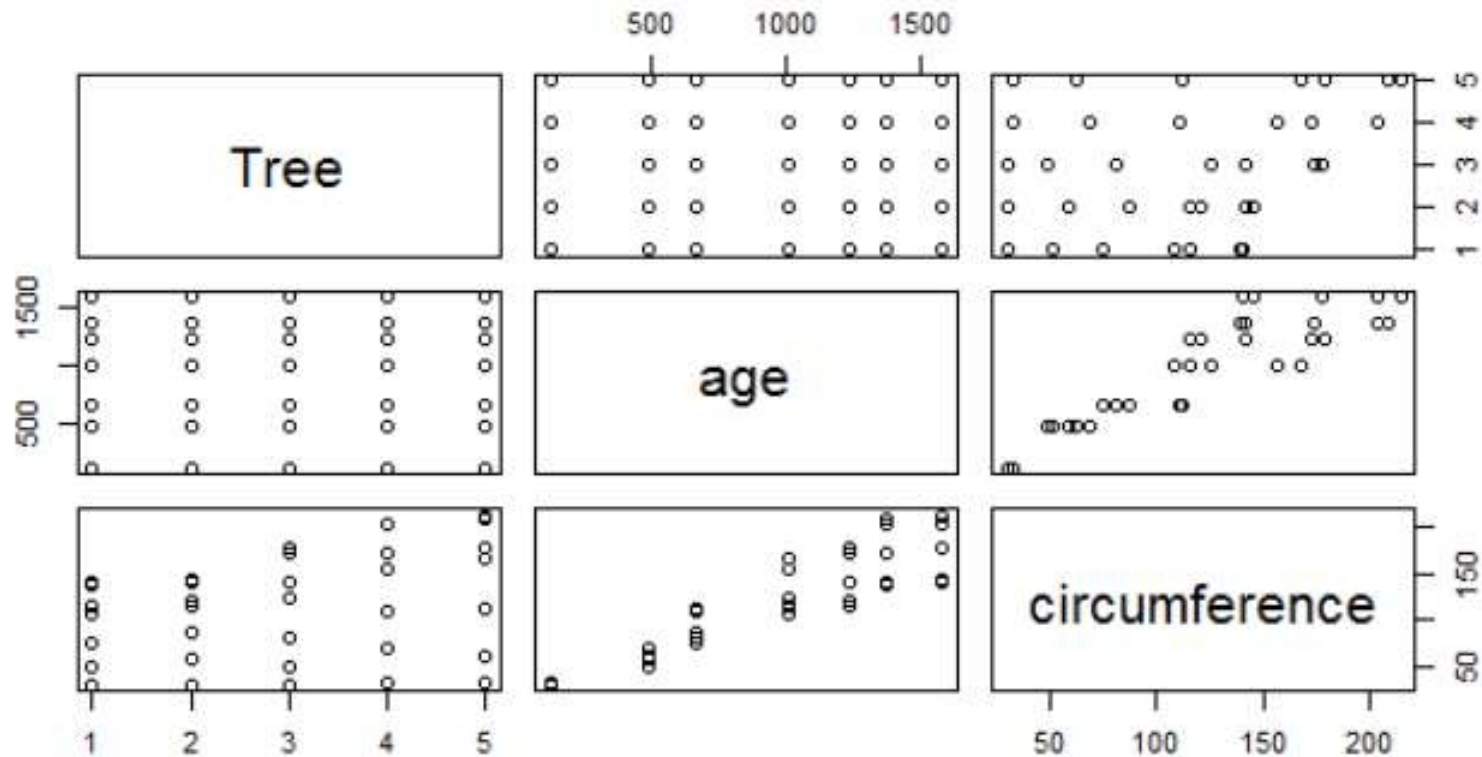
 4  6  8
11  7 14
> pie(cyltable, labels = names(cyltable))
> |
```

*Mostrar las proporciones de una serie de variable, deja de ser útil con más de diez valores*



# Todos con todos

```
> plot(Orange)  
>
```





# Guardar un gráfico

---

- Por línea de comandos

```
> jpeg(filename="migrafico.jpeg")  
> plot(Orange)  
> dev.off()
```

```
RStudioGD  
      2
```

```
> |
```

```
> pdf(file="migrafico.pdf")  
> plot(Orange)  
> dev.off()
```

```
RStudioGD  
      2
```

```
> |
```

Se guarda en ubicación por defecto

- Desde el propio RStudio, pestaña Plots y Export



# Ejercicio

---

La **cantidad de zinc** (en mg/l) en 16 muestras de alimentos infantiles viene dada por:

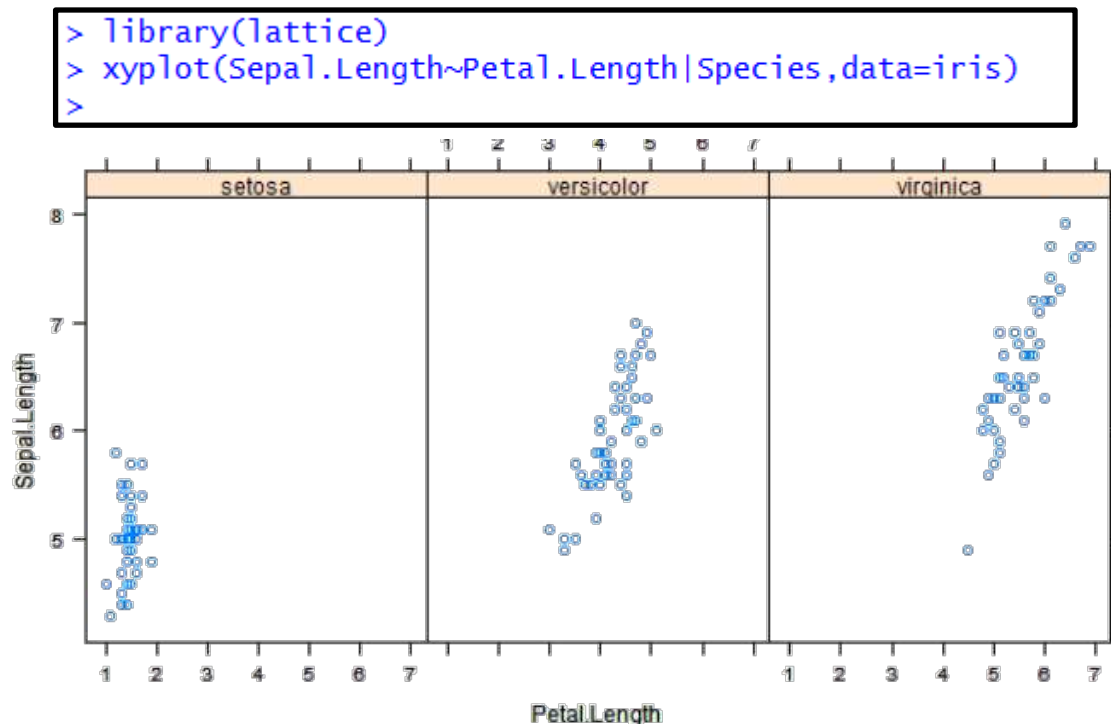
(3.0, 5.8, 5.6, 4.8, 5.1, 3.6, 5.5, 4.7, 5.7, 5.0, 5.9, 5.7, 4.4, 5.4, 4.2, 5.3)

Hallar media, desviación típica, mediana, cuartiles, y dibujar el box-plot e histograma.



# Gráficos en R – El sistema Lattice

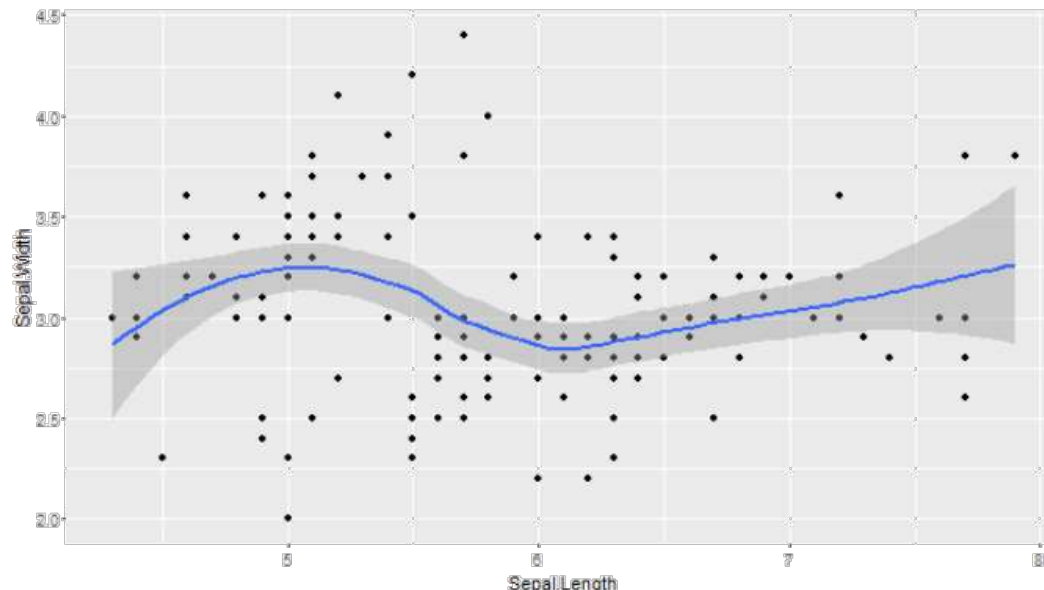
- Este sistema es útil para gráficos que representan una variable o relación entre variables condicionadas a los valores de uno o más factores.



# Gráficos en R – El sistema Ggplot2

- Basado en la gramática de los gráficos, proporciona un lenguaje para crear gráficas complejas de una forma más simple que los gráficos básicos

```
> ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width)) + geom_point() + stat_smooth()
```



# Paquetes

---

- Los paquetes son colecciones de funciones y datos
- R viene con un conjunto estándar de paquetes
- Otros muchos más están disponibles para su descarga e instalación
- Una vez instalados, tienen que ser cargados antes de ser usados

# Localizar un Paquete

- Vista por tareas en CRAN



[CRAN](#)  
[Mirrors](#)  
[What's new?](#)  
[Task Views](#)  
[Search](#)

*About R*  
[R Homepage](#)  
[The R Journal](#)

*Software*  
[R Sources](#)  
[R Binaries](#)  
[Packages](#)  
[Other](#)

*Documentation*  
[Manuals](#)  
[FAQs](#)  
[Contributed](#)

### CRAN Task Views

<a href="#">Bayesian</a>	Bayesian Inference
<a href="#">ChemPhys</a>	Chemometrics and Computational Physics
<a href="#">ClinicalTrials</a>	Clinical Trial Design, Monitoring, and Analysis
<a href="#">Cluster</a>	Cluster Analysis & Finite Mixture Models
<a href="#">DifferentialEquations</a>	Differential Equations
<a href="#">Distributions</a>	Probability Distributions
<a href="#">Econometrics</a>	Econometrics
<a href="#">Environmetrics</a>	Analysis of Ecological and Environmental Data
<a href="#">ExperimentalDesign</a>	Design of Experiments (DoE) & Analysis of Experimental Data
<a href="#">Finance</a>	Empirical Finance
<a href="#">Genetics</a>	Statistical Genetics
<a href="#">Graphics</a>	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
<a href="#">HighPerformanceComputing</a>	High-Performance and Parallel Computing with R
<a href="#">MachineLearning</a>	Machine Learning & Statistical Learning
<a href="#">MedicalImaging</a>	Medical Image Analysis
<a href="#">MetaAnalysis</a>	Meta-Analysis
<a href="#">Multivariate</a>	Multivariate Statistics
<a href="#">NaturalLanguageProcessing</a>	Natural Language Processing
<a href="#">NumericalMathematics</a>	Numerical Mathematics
<a href="#">OfficialStatistics</a>	Official Statistics & Survey Methodology
<a href="#">Optimization</a>	Optimization and Mathematical Programming

# Instalar un nuevo paquete

---

- Con **library()** pueden verse los paquetes instalados
- La función **install.packages()** se utiliza para instalar nuevos paquetes

```
> install.packages("ggplot2")
warning in install.packages :
  downloaded length 227 != reported length 227
Installing package into 'C:/Users/se47351/Documents/R/win-library/3.1'
(as 'lib' is unspecified)
trying URL 'http://cran.rstudio.com/bin/windows/contrib/3.1/ggplot2_1.0.1.zip'
content type 'application/zip' length 2676646 bytes (2.6 Mb)
opened URL
downloaded 2.6 Mb

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\se47351\AppData\Local\Temp\RtmpURskAn\downloaded_packages
```

# Utilizar un paquete

---

- Es necesario utilizar la función **library()** antes de utilizar un paquete que no esté dentro de los paquetes base de R
- De esta forma se cargan en memoria todas las funciones y datos que contiene la librería

```
>  
> library(ggplot2)  
warning message:  
package 'ggplot2' was built under R version 3.1.3  
> |
```



# Índice

---

- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- Explorando un DataFrame
- Gráficos
- **Escribiendo funciones**
- Librerías de R



# Funciones en R

---

- Todo lo que usas en R es una función
- Las librerías permiten añadir nuevas funciones
- R permite escribir nuevas funciones
- Al escribir el código de una función sin paréntesis en la consola, se muestra el código de la función

```
> rep  
function (x, ...) .Primitive("rep")  
> c  
function (..., recursive = FALSE) .Primitive("c")  
> sum  
function (..., na.rm = FALSE) .Primitive("sum")  
>
```

# Funciones en R

---

- Una función es un grupo de instrucciones que toma un "input" o datos de entrada, que usa para computar y retornar un resultado
- La sintaxis es la siguiente:

```
nombre_funcion <- function(input) {  
  # Cuerpo de la funcion  
  output <- input + 1  
  return(output)  
}
```

- Se utiliza **return()** para establecer lo que devuelve la función
- En el caso de no utilizar **return**, se devuelve el último valor con el que se trabaja o se asigna dentro de la función

# Funciones en R

---

- El valor que devuelve una función se puede almacenar en una nueva variable
- Dentro de una función es posible utilizar nombres de variables ya declarados anteriormente, sin que se alteren
- Los script de R, tienen la extensión R y se cargan con **source()**

```
> source("fmedia.R")  
> |
```

# Estructuras de Control

---

- Las estructuras de control permiten el control de la ejecución de los comandos

```
> ?Control  
>
```

- **If()** acepta una condición unidimensional

```
if (condicion) {  
    comando  
} else {  
    comando  
}
```

# Operadores lógicos

- R tiene los siguientes operadores lógicos

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x   y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

```
> 5 > 4
[1] TRUE
>
> 6 >= 6
[1] TRUE
>
> 2 < 1
[1] FALSE
>
> 5 != 5
[1] FALSE
>
> ! T
[1] FALSE
>
> (2 == 2) & (4 > 2)
[1] TRUE
>
```

# Bucle for

---

- El bucle **for()** recorre un vector ejecutando los comandos que se encuentran entre los corchetes:

```
> for (k in 1:5){  
+   print(1:k)  
+ }  
[1] 1  
[1] 1 2  
[1] 1 2 3  
[1] 1 2 3 4  
[1] 1 2 3 4 5  
> |
```

# Bucle repeat

---

- La instrucción **repeat** ejecuta un conjunto de comandos hasta que se encuentra con la sentencia **break**

```
> x <- 1
> repeat {
+   print(x)
+   if (x > 10) break
+   x <- x + 1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
> |
```



# Bucle while

---

- La instrucción **while** ejecuta un conjunto de comandos mientras que se cumpla una condición

```
> x <- 1
> while (x <= 10) {
+   print(x)
+   x <- x + 1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
>
```

# Next

---

- La sentencia **next** permite parar la ejecución del bucle y avanzar hasta la condición

```
> for(i in 1:10)
+ {
+   if(i %in% 4:7)
+     next
+   print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 8
[1] 9
[1] 10
> |
```

# Ejemplos & Ejercicio

---

- Función Suma
- Función Media
- Función Multiplicación
- Calcular proporción de vocales y consonantes

# Índice

---

- Introducción
- Utilizando R como una calculadora
- Estructuras de datos
- Funciones estadísticas
- Importando y exportando datos
- Explorando un DataFrame
- Gráficos
- Escribiendo funciones
- Librerías de R



# Aprender R con R

---



Learn R, in R.

swirl teaches you R programming and data science  
interactively, at your own pace, and right in the R console!



<http://swirlstats.com/>

# Apprender R con R

---

```
>  
> library(swirl)  
> swirl()  
  
| welcome to swirl!  
  
| Please sign in. If you've been here before, use the same name as you did then. If you  
| are new, call yourself something unique.  
  
what shall I call you? |
```

```
install_from_swirl("R Programming")  
install_from_swirl("Getting and Cleaning Data")  
install_from_swirl("Exploratory Data Analysis")  
install_from_swirl("Open Intro")  
install_from_swirl("Regression Models")  
install_from_swirl("Statistical Inference")
```

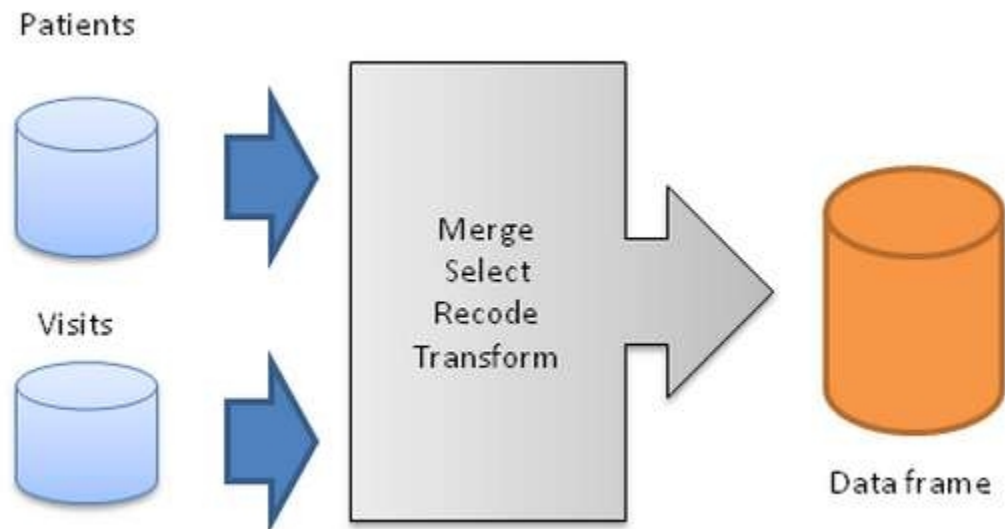
```
install.packages("swirl")  
library(swirl)
```



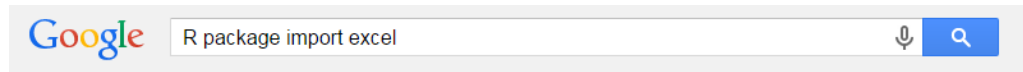
# Gestión de los datos

---

- Crear nuevas variables
- Cambiar de forma los datos
- Unir
- Ordenar
- Mezclar
- Agregar
- Filtrar



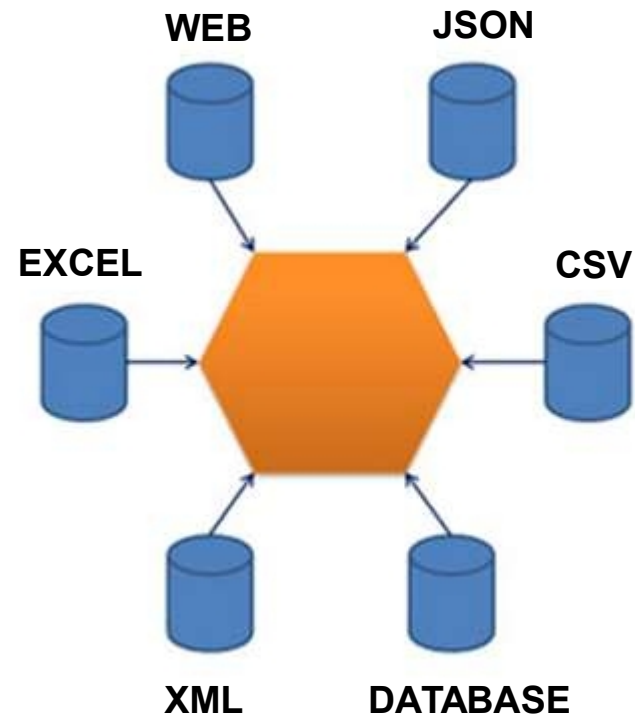
# Entrada de Datos



Web Videos Imágenes Noticias Maps Más ▾ Herramientas de búsqueda

Aproximadamente 114.000.000 resultados (0,55 segundos)

**R Data Import/Export - The Comprehensive R Archive Network**  
[cran.r-project.org/doc/manuals/r-.../R-data.html](http://cran.r-project.org/doc/manuals/r-.../R-data.html) ▾ Traducir esta página  
(See the rJava package from CRAN and the SJava, RSPerl and RSPython ... The easiest form of data to import into R is a simple text file, and this will often be ... such files directly from R. For Excel spreadsheets, the available methods are ...  
[Acknowledgements](#) - [1 Introduction](#) - [2 Spreadsheet-like data](#)

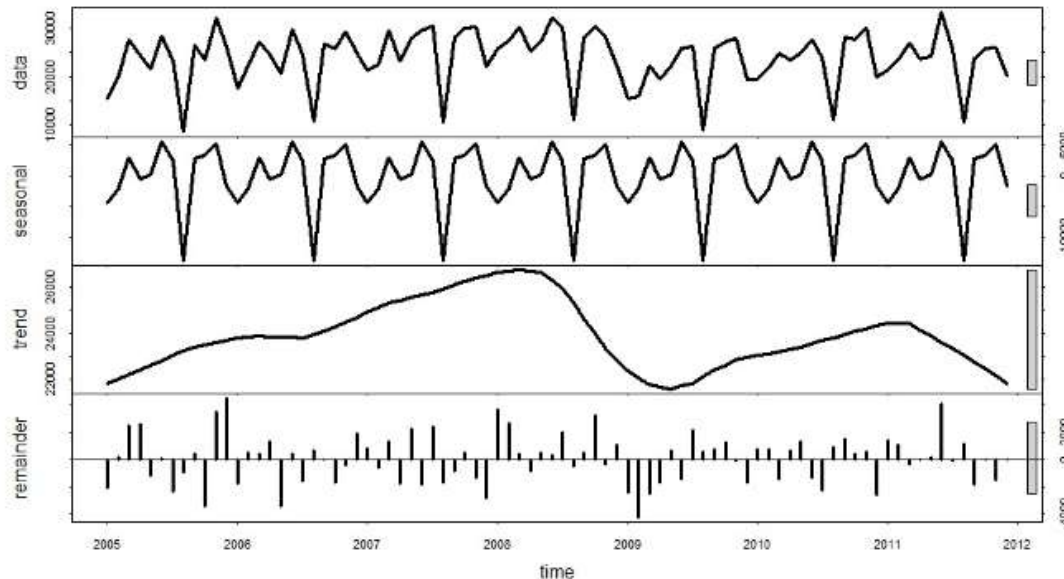




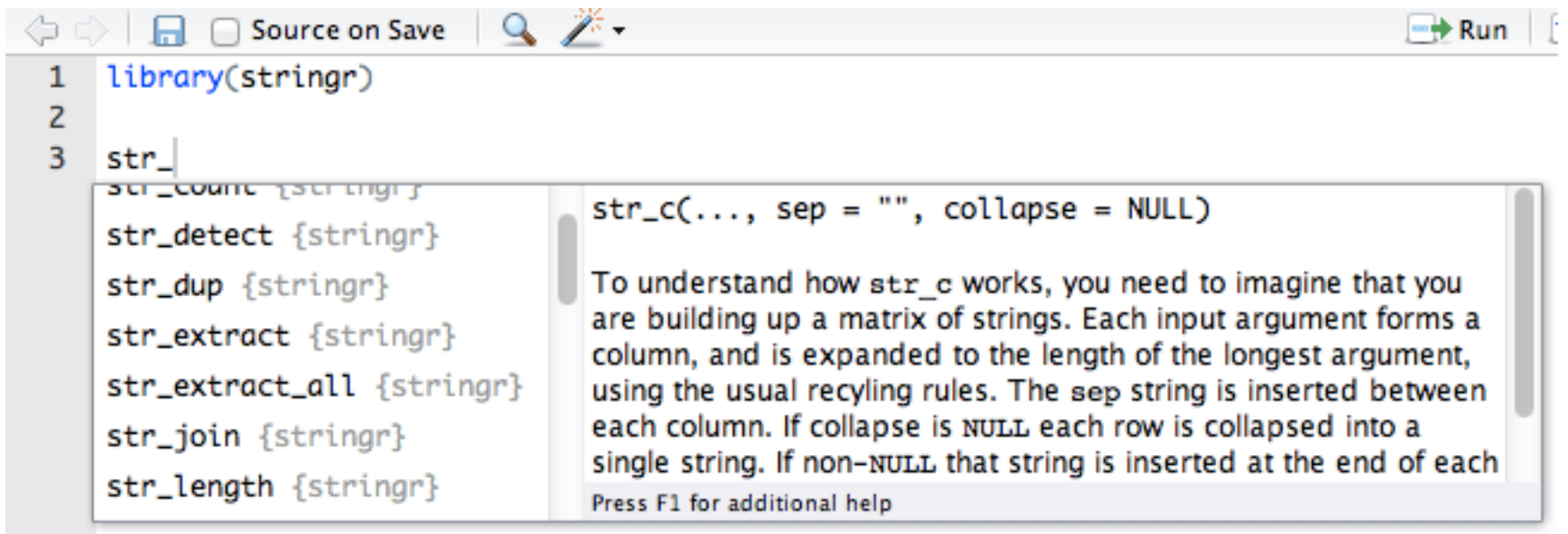
# Manejo de fechas y horas en R

- Tipos especiales para almacenar fechas y horas
- Operaciones con fechas (*Paquete lubridate*)
- Formatos de entrada y salida
- Análisis temporales en R =>

<https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/>



# Manejo de cadenas en R - stringr



The screenshot shows the RStudio interface with the stringr package documentation open. The editor window on the left contains the following code:

```
1 library(stringr)
2
3 str_
```

The documentation panel on the right displays the `str_c` function signature and a detailed description:

`str_c(..., sep = "", collapse = NULL)`

To understand how `str_c` works, you need to imagine that you are building up a matrix of strings. Each input argument forms a column, and is expanded to the length of the longest argument, using the usual recycling rules. The `sep` string is inserted between each column. If `collapse` is `NULL` each row is collapsed into a single string. If non-`NULL` that string is inserted at the end of each

Press F1 for additional help



# Manejo de datos en R - dplyr

---

- Proporciona herramientas para manipular datos
- Muy fácil para los que vienen del mundo SQL
- Muy intuitivo una vez que conoces lo básico
- Código muy fácil de leer y de mantener
- Muy rápido
- Funciones más comunes:
  - o Select □ Seleccionar conjunto de columnas
  - o Filter □ Seleccionar conjunto de filas
  - o Arrange □ Ordenar filas
  - o Mutate □ Crear nuevas columnas
  - o Group\_by □ Para agrupamientos
  - o Summarize □ Operaciones de agregación



# Estructurar datos en R - tidyr

---

- Se utiliza para estructurar un conjunto de datos
- Se suele usar junto con dplyr
- Sus principios son:
  - Cada variable debe tener su columna
  - Cada observación tiene que tener su fila
  - Cada observación su celda
- Si dataset no cumple estas condiciones => "Dato Sucio"
- Sus funciones más comunes son:
  - Gather: Toma varias columnas y las une en parejas clave-valor
  - Separate: Separar una columna en múltiples columnas
  - Spread: Cuando tenemos una observación en varias filas.  
Sirve para llevar a una observación por fila. Opuesta a Gather
  - Unite: Opuesto de separate, no se usa mucho

# Cambiar la forma de los datos - reshape2

- Facilita la transformación entre formatos Ancho (Wide) y Largo (Long)
- Wide, los datos tienen una columna por variables
- Long, una columna para las variables y otra para sus valores. Puede haber varios niveles de formato Long
- Para los análisis se usa el formato Long, el Wide es más óptimo para el almacenamiento
- Dos funciones clave:
  - **melt**: Wide => Long
  - **cast**: Long => Wide

```
##      Ozone Solar.R Wind Temp Month Day
## 1      41      190  7.4   67     5    1
## 2      36      118  8.0   72     5    2
## 3      12      149 12.6   74     5    3
## 4      18      313 11.5   62     5    4
## 5      NA       NA 14.3   56     5    5
## 6      28       NA 14.9   66     5    6
```

```
##      variable value
## 1      Ozone      41
## 2      Ozone      36
## 3      Ozone      12
## 4      Ozone      18
## 5      Ozone      NA
## 6      Ozone      28
## 7    Solar.R     190
## 8    Solar.R     118
## 9    Solar.R     149
## 10   Solar.R     313
```

# Limpieza de los datos

---

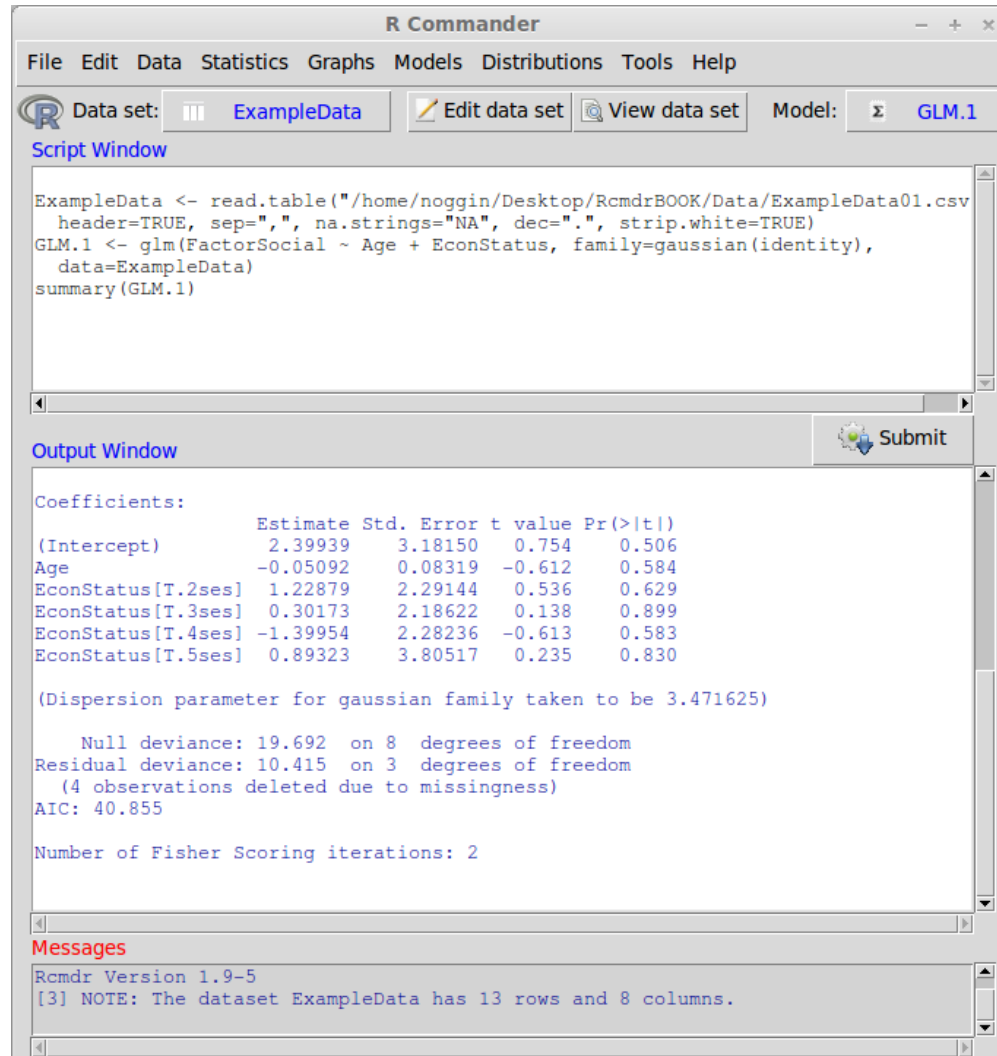
- Detección y localización de errores
- Corrección de errores
- Relleno de huecos
- Filas duplicadas
- Valores Imposibles
  - Fechas inconsistentes
  - Ventas negativas

Good data scientists understand, in a deep way, that the heavy lifting of cleanup and preparation isn't something that gets in the way of solving the problem – it is the problem.



DJ Patil, *Building Data Science Teams*

# Rcommander - Análisis Estadísticos



The screenshot displays the R Commander application window. The menu bar includes File, Edit, Data, Statistics, Graphs, Models, Distributions, Tools, and Help. The toolbar shows buttons for Data set (ExampleData), Edit data set, View data set, and Model (GLM.1). The Script Window contains the following R code:

```
ExampleData <- read.table("/home/noggin/Desktop/RcmdrBOOK/Data/ExampleData01.csv",
  header=TRUE, sep=";", na.strings="NA", dec=".", strip.white=TRUE)
GLM.1 <- glm(FactorSocial ~ Age + EconStatus, family=gaussian(identity),
  data=ExampleData)
summary(GLM.1)
```

The Output Window displays the results of the GLM fit:

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.39939	3.18150	0.754	0.506
Age	-0.05092	0.08319	-0.612	0.584
EconStatus[T.2ses]	1.22879	2.29144	0.536	0.629
EconStatus[T.3ses]	0.30173	2.18622	0.138	0.899
EconStatus[T.4ses]	-1.39954	2.28236	-0.613	0.583
EconStatus[T.5ses]	0.89323	3.80517	0.235	0.830

(Dispersion parameter for gaussian family taken to be 3.471625)

Null deviance: 19.692 on 8 degrees of freedom  
Residual deviance: 10.415 on 3 degrees of freedom  
(4 observations deleted due to missingness)  
AIC: 40.855

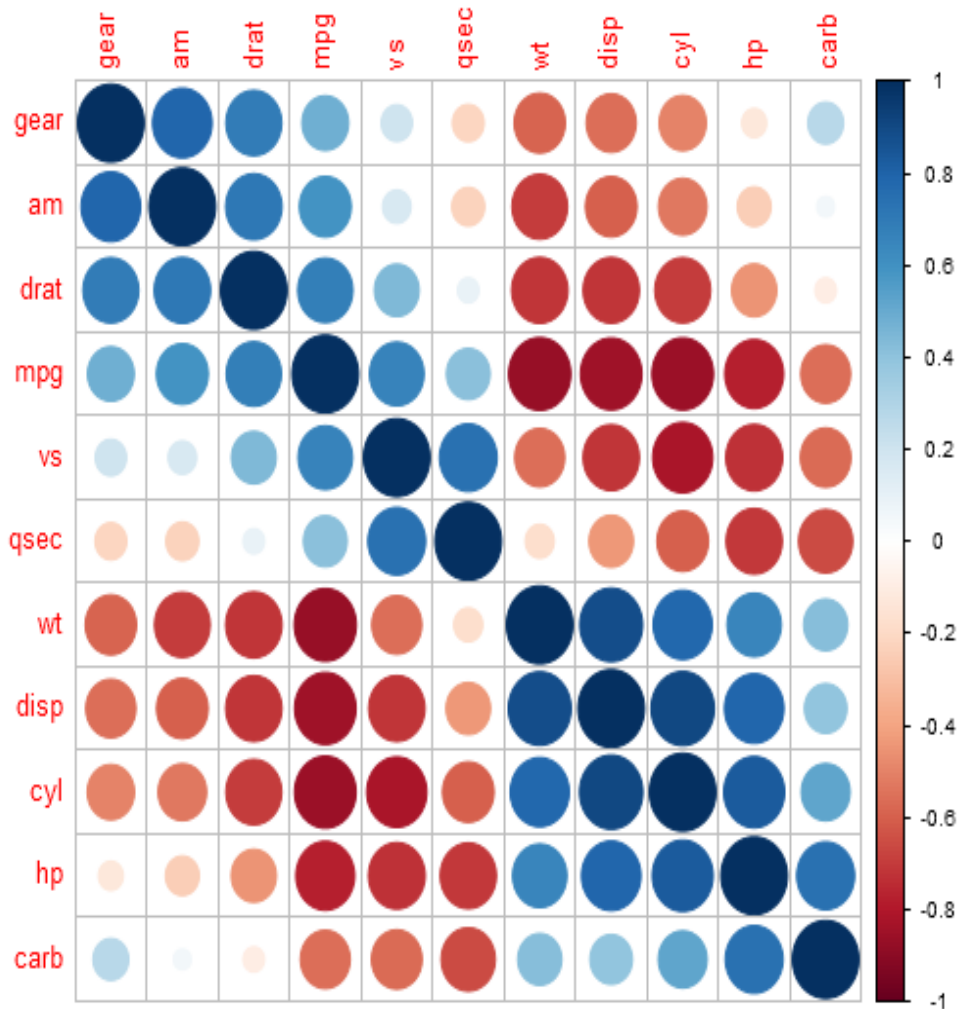
Number of Fisher Scoring iterations: 2

Messages

Rcmdr Version 1.9-5  
[3] NOTE: The dataset ExampleData has 13 rows and 8 columns.



# CORRPLOT – Matriz de Correlaciones





# Machine Learning

---

Existen varios programas, entre ellos:

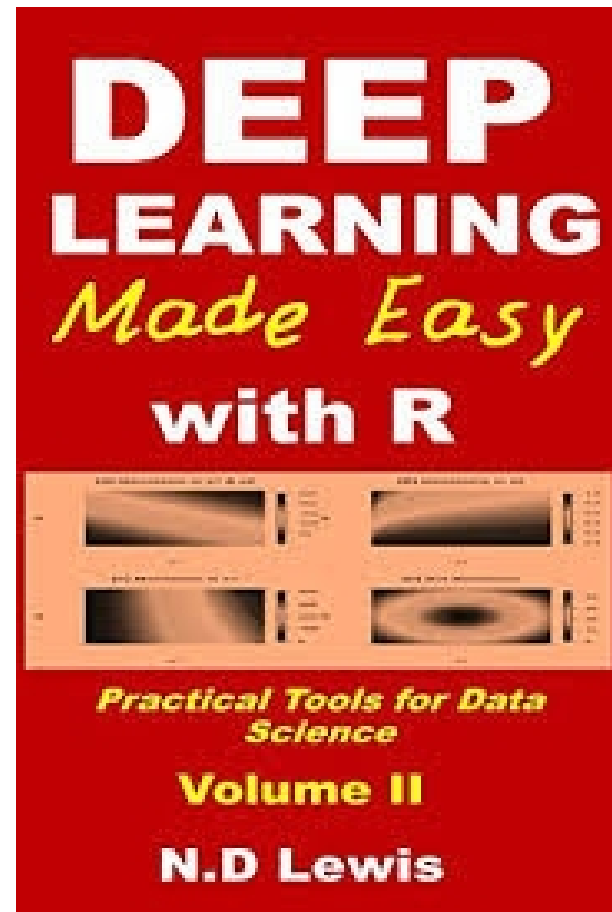
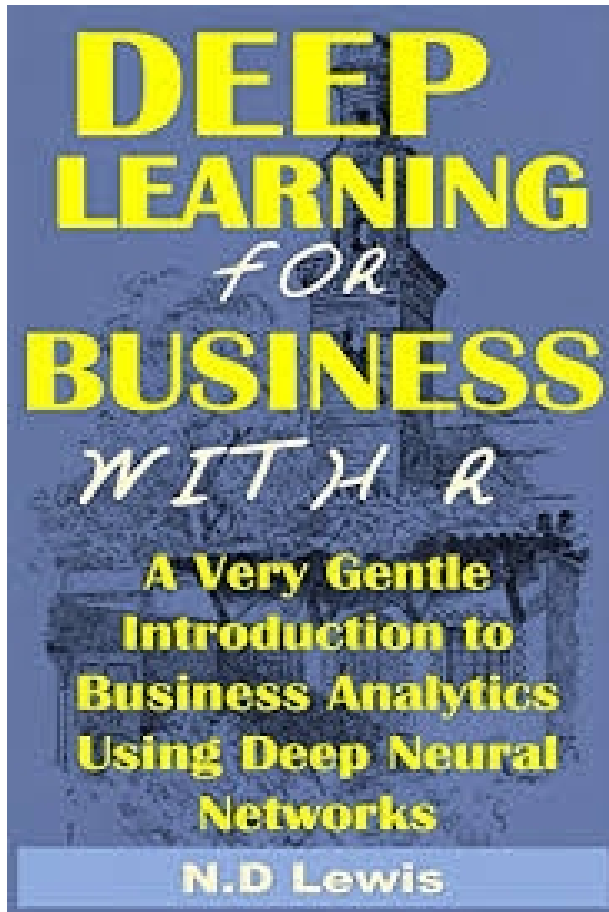
- MLR - <https://github.com/mlr-org>
  - o *Quizás la mejor opción, mantenido por cuatro personas.*
- Caret - <http://topepo.github.io/caret/index.html>
  - o <https://www.datacamp.com/courses/machine-learning-toolbox>
- Rattle - <http://rattle.togaware.com>
  - o *Ejercicio en: \Ejercicios\Rattle\_Tutorial.pdf*



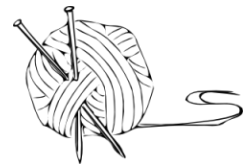
# Deep Learning

---

Dos libros de referencia:



# Knitr - Generación de informes en R



<https://yihui.name/knitr/>

Which checks out with our value. This suggests that  $X_1$  and  $X_2$  are pretty colinear. Let's visualize this by using a scatter plot:

```
plot(x1, x2, main = "Correlation of X1 and X2", xlab = "X1", ylab = "X2")
```

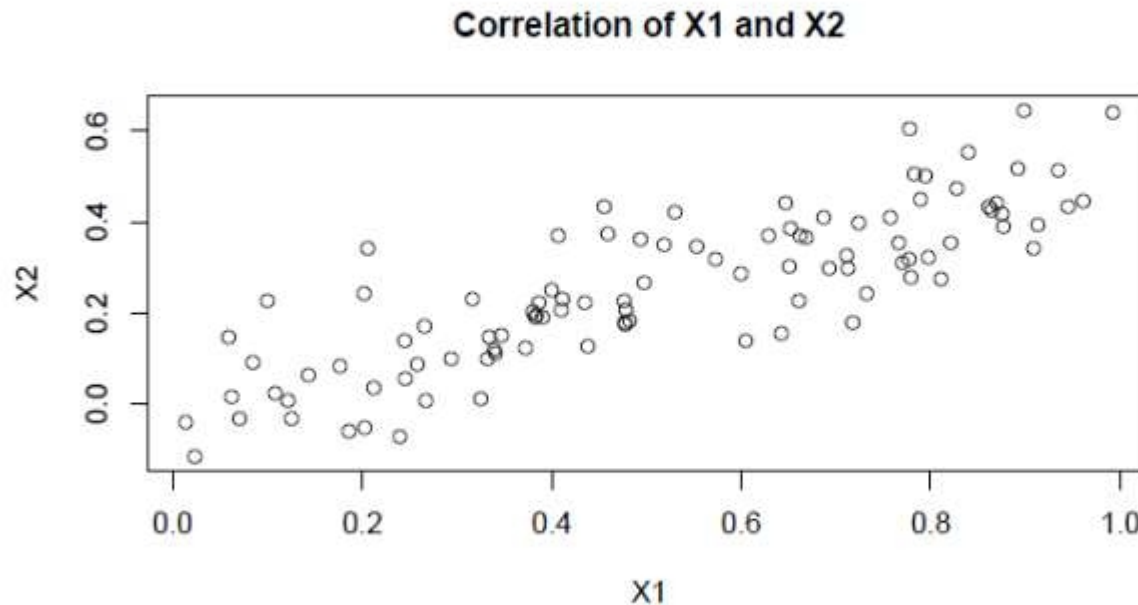


Figure 1: Correlation of given predictors.

Simple figure



<https://github.com/javiermoralo/rutas-aereas-con-R>

# Anaconda

---

CONTINUUM<sup>®</sup>  
ANALYTICS

ANACONDA

COMMUNITY

SERVICES

SOLUTIONS

ABOUT

RESOURCES



[ANACONDA](#) ▸ [DOWNLOAD](#)

## DOWNLOAD ANACONDA NOW!

Jump to: [Windows](#) | [OSX](#) | [Linux](#)

### Get Superpowers with Anaconda

Anaconda is a completely free Python distribution (including for commercial use and redistribution). It includes more than 400 of the most popular [Python packages](#) for science, math, engineering, and data analysis. See [the packages included with Anaconda](#) and [the Anaconda changelog](#).

### Which version should I download and install?

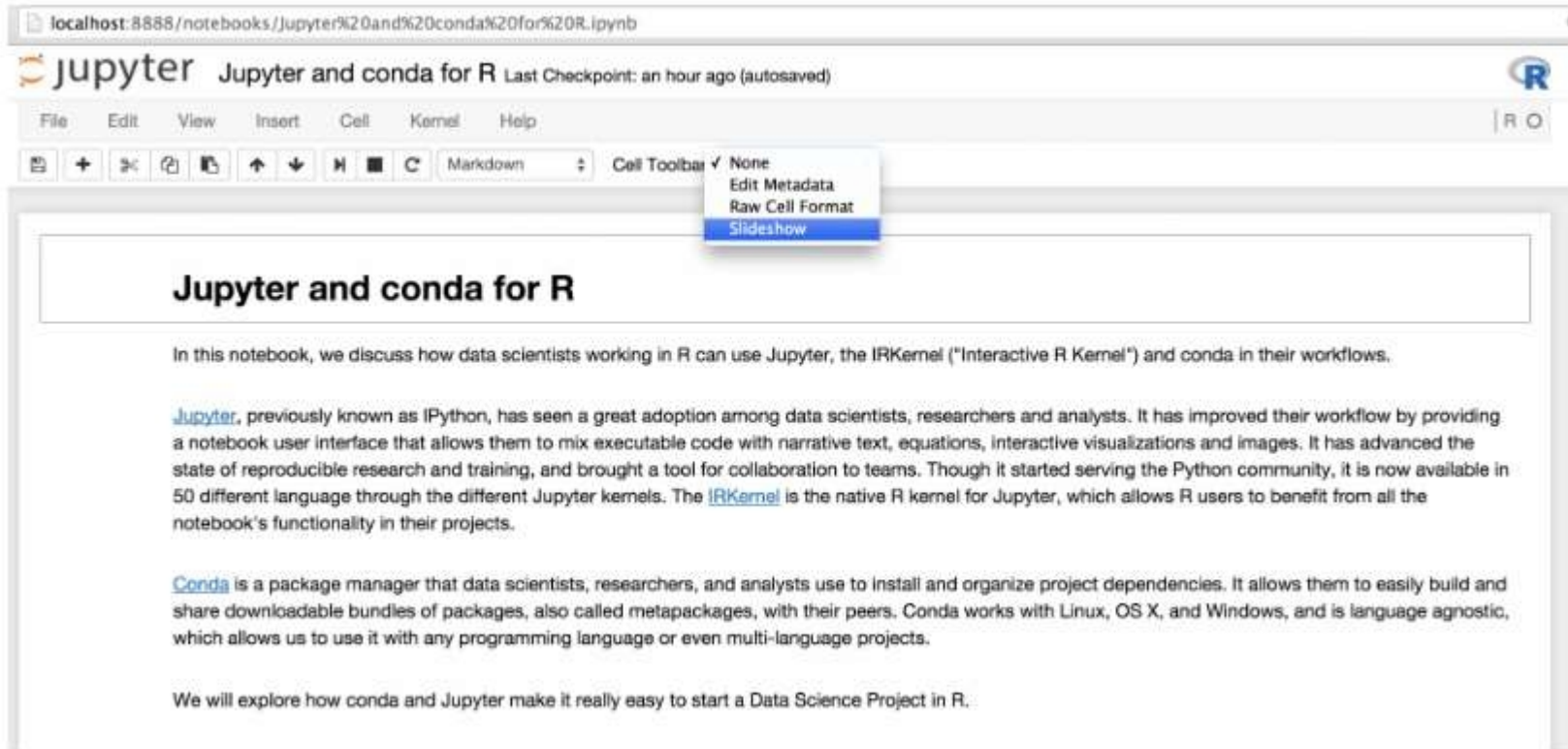
Because Anaconda includes installers for Python 2.7 and 3.5, either is fine. Using either version, you can use Python 3.4 with the conda command. You can create a 3.5 environment with the conda command if you've downloaded 2.7 — and vice versa.



[https://docs.continuum.io/anaconda/r\\_language](https://docs.continuum.io/anaconda/r_language)



# Jupyter - R interactivo



# Shiny - Aplicaciones web con R



Get inspired  
(gallery)



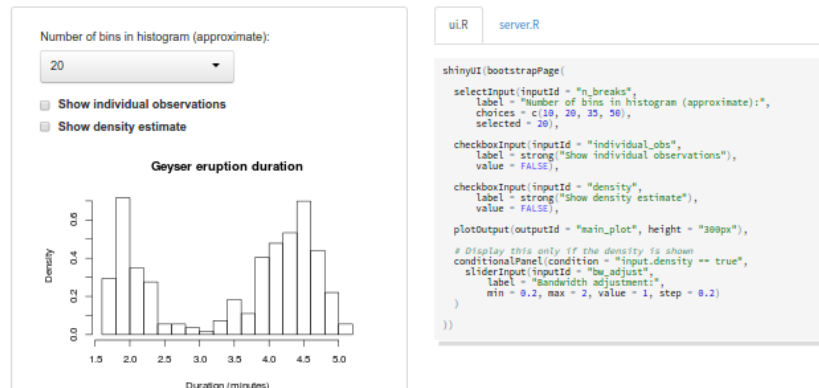
Get started  
(tutorial)



Go deeper  
(articles)

## Here is a Shiny app

Shiny apps are easy to write. No web development skills are required.



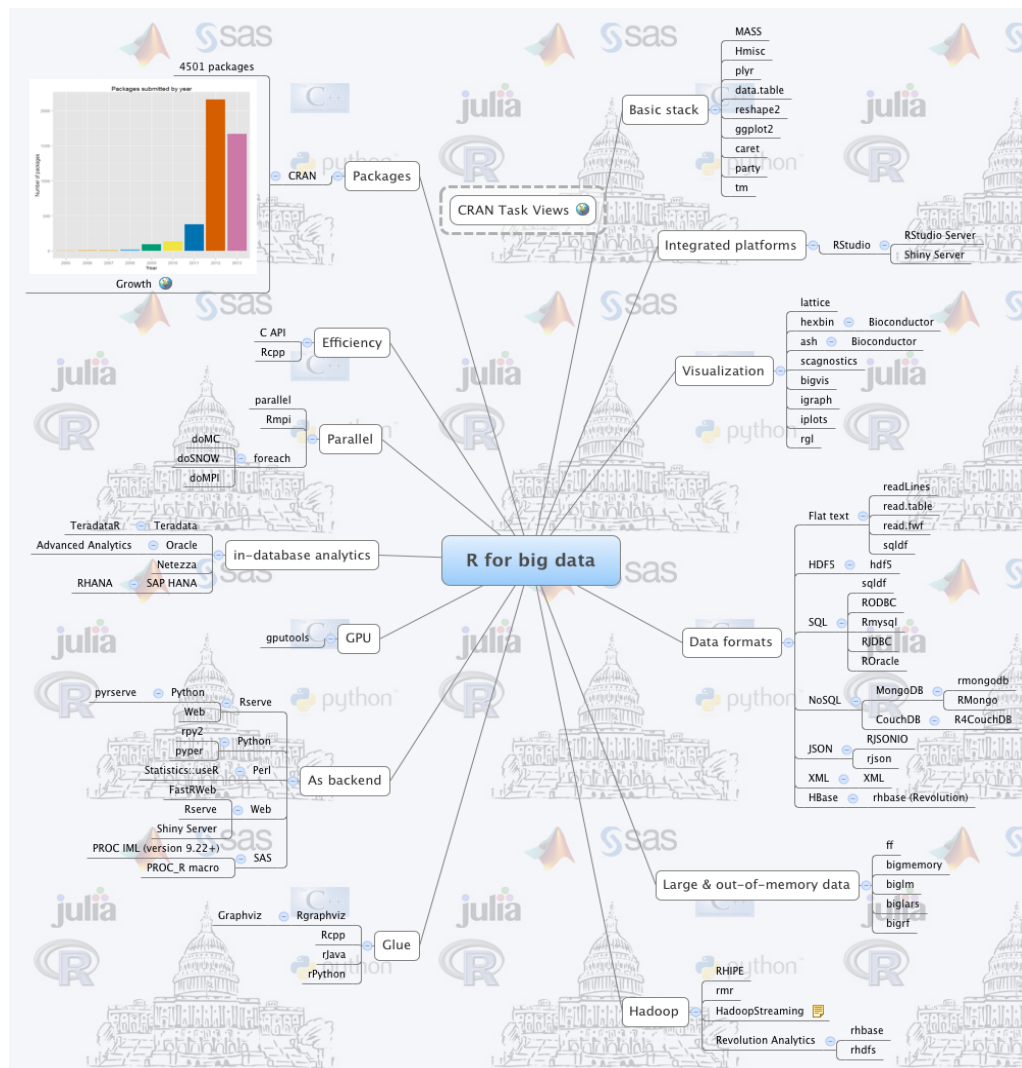
<http://shiny.rstudio.com>

# SparkR – Paralelizando R

---



# Shiny - Aplicaciones web con R





---

# GRACIAS POR VUESTRA ATENCIÓN

**Jesús Javier Moralo García**

[javier.moralo@datahack.es](mailto:javier.moralo@datahack.es)

[www.datahack.es](http://www.datahack.es)

