

# Prácticas de Tecnologías de Gestión y Manipulación de Datos

Guillermo López Taboada ([guillermo.lopez.taboada@udc.es](mailto:guillermo.lopez.taboada@udc.es)) y Rubén F. Casal ([ruben.fcasal@udc.es](mailto:ruben.fcasal@udc.es))

2021-12-10



# Índice general

Prólogo	5
<b>1 Introducción a las Tecnologías de Gestión y Manipulación de Datos</b>	<b>7</b>
1.1 Contenidos	7
1.2 Planificación (tentativa)	8
1.3 Fuentes de información:	8
<b>2 Manipulación de datos con R</b>	<b>11</b>
2.1 Lectura, importación y exportación de datos	11
2.2 Manipulación de datos	16
2.3 Ejemplo WoS data	27
<b>3 Introducción al lenguaje SQL</b>	<b>31</b>
3.1 Bases de Datos Relacionales	31
3.2 Sintaxis SQL	35
3.3 Conexión con bases de datos desde R	38
3.4 Ejemplo Scopus data	41
3.5 Ejercicios SQL con RSQLite	41
3.6 Práctica 1: SQL	43
<b>4 Manipulación de datos con dplyr</b>	<b>45</b>
4.1 El paquete <b>dplyr</b>	45
4.2 Operaciones con variables (columnas)	46
4.3 Operaciones con casos (filas)	48
4.4 Resumir valores con <b>summarise()</b>	48
4.5 Agrupar casos con <b>group_by()</b>	49
4.6 Operador <i>pipe</i> <b>%&gt;%</b> (tubería, redirección)	49
4.7 Operaciones con tablas de datos	50
4.8 Bases de datos con dplyr	51
<b>5 Introducción a Tecnologías NoSQL</b>	<b>61</b>
5.1 Conceptos y tipos de bases de datos NoSQL (documental, columnar, clave/valor y de grafos)	61

5.2	Conexión de R a MongoDB . . . . .	69
5.3	Ejercicios prácticos con MongoDB . . . . .	70
<b>6</b>	<b>Tecnologías para el Tratamiento de Datos Masivos</b>	<b>71</b>
6.1	Introducción al Aprendizaje Estadístico . . . . .	71
6.2	Tecnologías Big Data (Hadoop/Spark y Visualización) . . . . .	72
6.3	Introducción al Análisis de Datos Masivos . . . . .	79
<b>A</b>	<b>Enlaces</b>	<b>81</b>
A.1	RStudio . . . . .	83
A.2	Bibliometría . . . . .	84
<b>B</b>	<b>El paquete scimetr</b>	<b>85</b>
B.1	Instalación . . . . .	85
B.2	Carga de datos . . . . .	86
B.3	Sumarios . . . . .	88
B.4	Gráficos . . . . .	91
B.5	Filtrado . . . . .	100
B.6	Índices de autores . . . . .	105
<b>C</b>	<b>El paquete CITAN</b>	<b>107</b>
C.1	Creación de la base de datos . . . . .	107
C.2	Extraer información de la BD . . . . .	109
C.3	Cerrar conexión . . . . .	116
<b>D</b>	<b>Instalación de R</b>	<b>117</b>
D.1	Instalación de R en Windows . . . . .	118
D.2	Instalación en Mac OS X . . . . .	120
D.3	Instalación (opcional) de un entorno o editor de comandos . . . . .	121

# Prólogo

Este libro contiene algunas de las prácticas de la asignatura de Tecnologías de Gestión de Datos del Máster interuniversitario en Técnicas Estadísticas).

Este libro ha sido escrito en R-Markdown empleando el paquete `bookdown` y está disponible en el repositorio Github: `gltaboada/tgdbook`. Se puede acceder a la versión en línea a través del siguiente enlace:

<https://gltaboada.github.io/tgdbook>.

donde puede descargarse en formato pdf.

Para ejecutar los ejemplos mostrados en el libro será necesario tener instalados los siguientes paquetes: `dplyr` (colección `tidyverse`), `tidyr`, `stringr`, `readxl`, `openxlsx`, `RODBC`, `sqldf`, `RSQLite`, `foreign`, `magrittr`, `knitr` Por ejemplo mediante los comandos:

```
pkgs <- c('dplyr', 'tidyr', 'stringr', 'readxl', 'openxlsx', 'magrittr',  
         'RODBC', 'sqldf', 'RSQLite', 'foreign', 'knitr')  
# install.packages(pkgs, dependencies=TRUE)  
install.packages(setdiff(pkgs, installed.packages()[,'Package']), dependencies = TRUE)
```

Para generar el libro (compilar) se recomendaría consultar el libro de “Escritura de libros con bookdown” en castellano.



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional (esperamos poder liberarlo bajo una licencia menos restrictiva más adelante...).



# Capítulo 1

## Introducción a las Tecnologías de Gestión y Manipulación de Datos

La información relevante de la materia está disponible en la guía docente y la ficha de la asignatura

En particular, los resultados de aprendizaje son:

- Manejar de forma autónoma y solvente el software necesario para acceder a conjuntos de datos en entornos profesionales y/o en la nube.
- Saber gestionar conjuntos de datos masivos en un entorno multidisciplinar que permita la participación en proyectos profesionales complejos que requieran el uso de técnicas estadísticas.
- Saber relacionar el software de diseño y gestión de bases de datos con el específicamente implementado para el análisis de datos.

### 1.1 Contenidos

1. Introducción al lenguaje SQL
  - Bases de datos relacionales
  - Sintaxis SQL
  - Conexión con bases de datos desde R
2. Introducción a tecnologías NoSQL
  - Conceptos y tipos de bases de datos NoSQL (documental, columnar, clave/valor y de grafos)
  - Conexión de R a NoSQL
3. Tecnologías para el tratamiento de datos masivos

- Introducción al Aprendizaje Estadístico
- Tecnologías Big Data (Hadoop, Spark, Sparklyr)
- Ejercicios de análisis de datos masivos.

## 1.2 Planificación (tentativa)

La impartición de los contenidos durante el curso dependerá de los conocimientos de partida y la asimilación de los conceptos. Para completar nuestra visión de los conocimientos previos os requerimos completar este formulario en la primera sesión de clase: <https://forms.gle/9HR5iFHXoLowrCHLA>

- Semana 1 (3 y 5/11): Presentación e introducción a Tema 1 y SQL.
- Semana 2 (8, 10 y 12/11): Seminario Manipulación de datos con dplyr y Ejercicios SQL
- Semana 3 (17 y 19/11): Tema 2: NoSQL y Seminario tratamiento de texto (CSV, excel, Json) y open data
- Semana 4 (22, 24 y 26): Tema 3: Conceptos de Big Data y Tutorial de sparklyr,
- Semana 5 (29/11, 1 y 3/12): Tema 3: Introducción al Aprendizaje Estadístico y prácticas
- Semana 6 (10/12): Tema 3: Prácticas sparklyr
- Semana 7 (13, 15 y 17/12): Tema 3: Prácticas sparklyr y seminario de visualización

Examen 25/1 4pm.

### 1.2.1 Evaluación

- **Examen** (60%): El examen de la materia evaluará los siguientes aspectos: Conceptos de la materia: Dominio de los conocimientos teóricos y operativos de la materia. Asimilación práctica de materia: Asimilación y comprensión de los conocimientos teóricos y operativos de la materia.
- **Prácticas de laboratorio** (40%): Evaluación de las prácticas de laboratorio desarrolladas por los estudiantes.

## 1.3 Fuentes de información:

### 1.3.1 Básica

- Daroczi, G. (2015). Mastering Data Analysis with R. Packt Publishing
- Grolemund, G. y Wickham, H. (2016). R for Data Science O'Reilly



- Silberschatz, A., Korth, H. y Sudarshan, S. (2014). Fundamentos de Bases de Datos. Mc Graw Hill
- Rubén Fernández Casal y Julián Costa Bouza. Apuntes de Aprendizaje Estadístico
- Luraschi, J., Kuo, K., Ruiz, K. Mastering Spark with R O'Reilly
- Rubén Fernández Casal (R Machinery):
  - Introducción al Análisis de Datos con R (con Javier Roca y Julián Costa)
  - Ayuda y Recursos para el Aprendizaje de R
  - Escritura de libros con el paquete bookdown (con Tomás Cotos)
  - Apéndice introducción a Rmarkdown
  - Presentación análisis de datos con R

### 1.3.2 Complementaria:

- Wes McKinney (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly (2ª ed.)
- Tom White (2015). Hadoop: The Definitive Guide. O'Reilly (4ª ed.)
- Alex Holmes (2014). Hadoop in practice. Manning (2ª ed.)
- Centro de Supercomputación de Galicia (2020). [Servicio de Big Data del CESGA] (<https://bigdata.cesga.es/>)



## Capítulo 2

# Manipulación de datos con R

En el proceso de análisis de datos, al margen de su obtención y organización, una de las primeras etapas es el acceso y la manipulación de los datos (ver Figura 2.1). En este capítulo se repasarán brevemente las principales herramientas disponibles en el paquete base de R para ello. Posteriormente en el Capítulo 4 se mostrará como alternativa el uso del paquete `dplyr`.

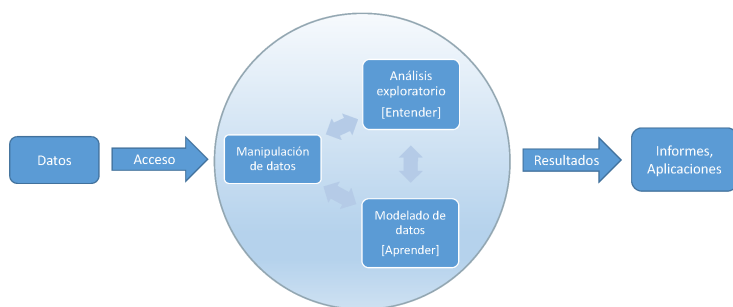


Figura 2.1: Etapas del proceso

### 2.1 Lectura, importación y exportación de datos

Además de la introducción directa, R es capaz de importar datos externos en múltiples formatos:

- bases de datos disponibles en librerías de R
- archivos de texto en formato ASCII

- archivos en otros formatos: Excel, SPSS, ...
- bases de datos relacionales: MySQL, Oracle, ...
- formatos web: HTML, XML, JSON, ...
- ....

### 2.1.1 Formato de datos de R

El formato de archivo en el que habitualmente se almacena objetos (datos) R es binario y está comprimido (en formato "gzip" por defecto). Para cargar un fichero de datos se emplea normalmente `load()`:

```
res <- load("data/empleados.RData")
res
```

```
## [1] "empleados"
```

```
ls()
```

```
## [1] "citefig"    "citefig2"   "empleados" "fig.path"   "inline"     "inline2"
## [7] "is_html"    "is_latex"   "latexfig"   "latexfig2" "res"
```

y para guardar `save()`:

```
# Guardar
save(empleados, file = "data/empleados_new.RData")
```

Aunque, como indica este comando en la ayuda (`?save`):

For saving single R objects, `saveRDS()` is mostly preferable to `save()`, notably because of the functional nature of `readRDS()`, as opposed to `load()`.

```
saveRDS(empleados, file = "data/empleados_new.rds")
## restore it under a different name
empleados2 <- readRDS("data/empleados_new.rds")
# identical(empleados, empleados2)
```

El objeto empleado normalmente en R para almacenar datos en memoria es el `data.frame`.

### 2.1.2 Acceso a datos en paquetes

R dispone de múltiples conjuntos de datos en distintos paquetes, especialmente en el paquete `datasets` que se carga por defecto al abrir R. Con el comando `data()` podemos obtener un listado de las bases de datos disponibles.

Para cargar una base de datos concreta se utiliza el comando `data(nombre)` (aunque en algunos casos se cargan automáticamente al emplearlos). Por ejemplo, `data(cars)` carga la base de datos llamada `cars` en el entorno de trabajo

(`".GlobalEnv"`) y `?cars` muestra la ayuda correspondiente con la descripción de la base de datos.

### 2.1.3 Lectura de archivos de texto

En R para leer archivos de texto se suele utilizar la función `read.table()`. Supóngase, por ejemplo, que en el directorio actual está el fichero `empleados.txt`. La lectura de este fichero vendría dada por el código:

```
# Session > Set Working Directory > To Source...?
datos <- read.table(file = "data/empleados.txt", header = TRUE)
# head(datos)
str(datos)

## 'data.frame':  474 obs. of  10 variables:
## $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
## $ sexo    : chr  "Hombre" "Hombre" "Mujer" "Mujer" ...
## $ fechnac : chr  "2/3/1952" "5/23/1958" "7/26/1929" "4/15/1947" ...
## $ educ    : int  15 16 12 8 15 15 15 12 15 12 ...
## $ catlab  : chr  "Directivo" "Administrativo" "Administrativo" "Administrativo" ...
## $ salario : num  57000 40200 21450 21900 45000 ...
## $ salini  : int  27000 18750 12000 13200 21000 13500 18750 9750 12750 13500 ...
## $ tiempemp: int  98 98 98 98 98 98 98 98 98 ...
## $ expprev : int  144 36 381 190 138 67 114 0 115 244 ...
## $ minoria : chr  "No" "No" "No" "No" ...
```

Si el fichero estuviese en el directorio `c:\datos` bastaría con especificar `file = "c:/datos/empleados.txt"`. Nótese también que para la lectura del fichero anterior se ha establecido el argumento `header=TRUE` para indicar que la primera línea del fichero contiene los nombres de las variables.

Los argumentos utilizados habitualmente para esta función son:

- `header`: indica si el fichero tiene cabecera (`header=TRUE`) o no (`header=FALSE`). Por defecto toma el valor `header=FALSE`.
- `sep`: carácter separador de columnas que por defecto es un espacio en blanco (`sep=""`). Otras opciones serían: `sep=","` si el separador es un “,”, `sep="*"` si el separador es un “\*”, etc.
- `dec`: carácter utilizado en el fichero para los números decimales. Por defecto se establece `dec = "."`. Si los decimales vienen dados por “,” se utiliza `dec = ","`

Resumiendo, los (principales) argumentos por defecto de la función `read.table` son los que se muestran en la siguiente línea:

```
read.table(file, header = FALSE, sep = "", dec = ".")
```

Para más detalles sobre esta función véase `help(read.table)`.

Están disponibles otras funciones con valores por defecto de los parámetros adecuados para otras situaciones. Por ejemplo, para ficheros separados por tabuladores se puede utilizar `read.delim()` o `read.delim2()`:

```
read.delim(file, header = TRUE, sep = "\t", dec = ".")
read.delim2(file, header = TRUE, sep = "\t", dec = ",")
```

### 2.1.4 Alternativa tidyverse

Para leer archivos de texto en distintos formatos también se puede emplear el paquete `readr` (colección `tidyverse`), para lo que se recomienda consultar el Capítulo 11 del libro *R for Data Science*.

### 2.1.5 Importación desde SPSS

El programa R permite lectura de ficheros de datos en formato SPSS (extensión `.sav`) sin necesidad de tener instalado dicho programa en el ordenador. Para ello se necesita:

- cargar la librería `foreign`
- utilizar la función `read.spss`

Por ejemplo:

```
library(foreign)
datos <- read.spss(file = "data/Employee data.sav", to.data.frame = TRUE)
# head(datos)
str(datos)
```

```
## 'data.frame': 474 obs. of 10 variables:
## $ id : num 1 2 3 4 5 6 7 8 9 10 ...
## $ sexo : Factor w/ 2 levels "Hombre","Mujer": 1 1 2 2 1 1 1 2 2 2 ...
## $ fechnac : num 1.17e+10 1.19e+10 1.09e+10 1.15e+10 1.17e+10 ...
## $ educ : Factor w/ 10 levels "8","12","14",...: 4 5 2 1 4 4 4 2 4 2 ...
## $ catlab : Factor w/ 3 levels "Administrativo",...: 3 1 1 1 1 1 1 1 1 1 ...
## $ salario : Factor w/ 221 levels "15750","15900",...: 179 137 28 31 150 101 121 31 ...
## $ salini : Factor w/ 90 levels "9000","9750",...: 60 42 13 21 48 23 42 2 18 23 ...
## $ tiempemp: Factor w/ 36 levels "63","64","65",...: 36 36 36 36 36 36 36 36 36 ...
## $ expprev : Factor w/ 208 levels "Ausente","10",...: 38 131 139 64 34 181 13 1 14 9 ...
## $ minoria : Factor w/ 2 levels "No","Sí": 1 1 1 1 1 1 1 1 1 1 ...
## - attr(*, "variable.labels")= Named chr [1:10] "Código de empleado" "Sexo" "Fecha ..."
## ..- attr(*, "names")= chr [1:10] "id" "sexo" "fechnac" "educ" ...
## - attr(*, "codepage")= int 1252
```

**Nota:** Si hay fechas, puede ser recomendable emplear la función `spss.get()` del paquete `Hmisc`.

### 2.1.6 Importación desde Excel

Se pueden leer fichero de Excel (con extensión *.xlsx*) utilizando por ejemplo los paquetes `openxlsx`, `readxl` (colección `tidyverse`), `XLConnect` o `RODBC` (este paquete se empleará más adelante para acceder a bases de datos), entre otros.

Por ejemplo el siguiente código implementa una función que permite leer todos los archivos en formato *.xlsx* en un directorio:

```
library(openxlsx)

read_xlsx <- function(path = '.') {
  files <- dir(path, pattern = '*.xlsx') # list.files
  # file.list <- lapply(files, readWorkbook)
  file.list <- vector(length(files), mode = 'list')
  for (i in seq_along(files))
    file.list[[i]] <- readWorkbook(files[i])
  file.names <- sub('\\.xlsx$', '', basename(files))
  names(file.list) <- file.names
  file.list
}
```

Para combinar los archivos (suponiendo que tienen las mismas columnas), podríamos ejecutar una llamada a `rbind()` o emplear la función `bind_rows()` del paquete `dplyr`:

```
df <- do.call('rbind', file.list)

df <- dplyr::bind_rows(file.list)
```

Como alternativa simple se pueden exportar los datos desde Excel a un archivo de texto *separado por comas* (extensión *.csv*). Por ejemplo, supongamos que queremos leer el fichero *coches.xls*:

- Desde Excel se selecciona el menú **Archivo -> Guardar como -> Guardar como** y en **Tipo** se escoge la opción de archivo CSV. De esta forma se guardarán los datos en el archivo *coches.csv*.
- El fichero *coches.csv* es un fichero de texto plano (se puede editar con Notepad), con cabecera, las columnas separadas por “;”, y siendo “.” el carácter decimal.
- Por lo tanto, la lectura de este fichero se puede hacer con:

```
datos <- read.table("coches.csv", header = TRUE, sep = ";", dec = ".", as.is = TRUE)
```

Otra posibilidad es utilizar la función `read.csv2`, que es una adaptación de la función general `read.table` con las siguientes opciones:

```
read.csv2(file, header = TRUE, sep = ";", dec = ".", as.is = TRUE)
```

Por lo tanto, la lectura del fichero *coches.csv* se puede hacer de modo más directo con:

```
datos <- read.csv2("coches.csv")
```

Esta forma de proceder, exportando a formato CSV, se puede emplear con otras hojas de cálculo o fuentes de datos. Hay que tener en cuenta que si estas fuentes emplean el formato anglosajón, el separador de campos será `sep = ","` y el de decimales `dec = "."`, las opciones por defecto en la función `read.csv()`.

### 2.1.7 Exportación de datos

Puede ser de interés la exportación de datos para que puedan leerlos con otros programas. Para ello, se puede emplear la función `write.table()`. Esta función es similar, pero funcionando en sentido inverso, a `read.table()` (Sección 2.1.3).

Veamos un ejemplo:

```
tipo <- c("A", "B", "C")
longitud <- c(120.34, 99.45, 115.67)
datos <- data.frame(tipo, longitud)
datos
```

```
##   tipo longitud
## 1    A   120.34
## 2    B    99.45
## 3    C   115.67
```

Para guardar el `data.frame` `datos` en un fichero de texto se puede utilizar:

```
write.table(datos, file = "datos.txt")
```

Otra posibilidad es utilizar la función:

```
write.csv2(datos, file = "datos.csv")
```

que dará lugar al fichero *datos.csv* importable directamente desde Excel.

## 2.2 Manipulación de datos

Una vez cargada una (o varias) bases de datos hay una serie de operaciones que serán de interés para el tratamiento de datos:

- Operaciones con variables:
  - crear
  - recodificar (e.g. categorizar)
  - ...
- Operaciones con casos:
  - ordenar
  - filtrar



- ...
- Operaciones con tablas de datos:
  - unir
  - combinar
  - consultar
  - ...

A continuación se tratan algunas operaciones *básicas*.

## 2.2.1 Operaciones con variables

### 2.2.1.1 Creación (y eliminación) de variables

Consideremos de nuevo la base de datos `cars` incluida en el paquete `datasets`:

```
data(cars)
# str(cars)
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

Utilizando el comando `help(cars)` se obtiene que `cars` es un `data.frame` con 50 observaciones y dos variables:

- `speed`: Velocidad (millas por hora)
- `dist`: tiempo hasta detenerse (pies)

Recordemos que, para acceder a la variable `speed` se puede hacer directamente con su nombre o bien utilizando notación “matricial”.

```
cars$speed
```

```
## [1] 4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15
## [26] 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24 25
```

```
cars[, 1] # Equivalente
```

```
## [1] 4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15
## [26] 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24 25
```

Supongamos ahora que queremos transformar la variable original `speed` (millas por hora) en una nueva variable `velocidad` (kilómetros por hora) y añadir esta nueva variable al `data.frame` `cars`. La transformación que permite pasar millas

a kilómetros es  $\text{kilómetros} = \text{millas} / 0.62137$  que en R se hace directamente con:

```
cars$speed/0.62137
```

Finalmente, incluimos la nueva variable que llamaremos *velocidad* en *cars*:

```
cars$velocidad <- cars$speed / 0.62137
head(cars)
```

```
##  speed dist velocidad
## 1     4    2  6.437388
## 2     4   10  6.437388
## 3     7    4 11.265430
## 4     7   22 11.265430
## 5     8   16 12.874777
## 6     9   10 14.484124
```

También transformaremos la variable *dist* (en pies) en una nueva variable *distancia* (en metros). Ahora la transformación deseada es  $\text{metros} = \text{pies} / 3.2808$ :

```
cars$distancia <- cars$dis / 3.2808
head(cars)
```

```
##  speed dist velocidad distancia
## 1     4    2  6.437388 0.6096074
## 2     4   10  6.437388 3.0480371
## 3     7    4 11.265430 1.2192148
## 4     7   22 11.265430 6.7056815
## 5     8   16 12.874777 4.8768593
## 6     9   10 14.484124 3.0480371
```

Ahora, eliminaremos las variables originales *speed* y *dist*, y guardaremos el *data.frame* resultante con el nombre *coches*. En primer lugar, veamos varias formas de acceder a las variables de interés:

```
cars[, c(3, 4)]
cars[, c("velocidad", "distancia")]
cars[, -c(1, 2)]
```

Utilizando alguna de las opciones anteriores se obtiene el *data.frame* deseado:

```
coches <- cars[, c("velocidad", "distancia")]
# head(coches)
str(coches)
```

```
## 'data.frame':   50 obs. of  2 variables:
## $ velocidad: num  6.44 6.44 11.27 11.27 12.87 ...
## $ distancia: num  0.61 3.05 1.22 6.71 4.88 ...
```

Finalmente los datos anteriores podrían ser guardados en un fichero exportable a Excel con el siguiente comando:

```
write.csv2(coches, file = "coches.csv")
```

### 2.2.1.2 Recodificación de variables

Con el comando `cut()` podemos crear variables categóricas a partir de variables numéricas. El parámetro `breaks` permite especificar los intervalos para la discretización, puede ser un vector con los extremos de los intervalos o un entero con el número de intervalos. Por ejemplo, para categorizar la variable `cars$speed` en tres intervalos equidistantes podemos emplear<sup>1</sup>:

```
fspeed <- cut(cars$speed, 3, labels = c("Baja", "Media", "Alta"))
table(fspeed)
```

```
## fspeed
## Baja Media Alta
## 11 24 15
```

Para categorizar esta variable en tres niveles con aproximadamente el mismo número de observaciones podríamos combinar esta función con `quantile()`:

```
breaks <- quantile(cars$speed, probs = seq(0, 1, len = 4))
fspeed <- cut(cars$speed, breaks, labels = c("Baja", "Media", "Alta"))
table(fspeed)
```

```
## fspeed
## Baja Media Alta
## 17 16 15
```

Para otro tipo de recodificaciones podríamos emplear la función `ifelse()` vectorial:

```
fspeed <- ifelse(cars$speed < 15, "Baja", "Alta")
fspeed <- factor(fspeed, levels = c("Baja", "Alta"))
table(fspeed)
```

```
## fspeed
## Baja Alta
## 23 27
```

Alternativamente en el caso de dos niveles podríamos emplear directamente la función `factor()`:

```
fspeed <- factor(cars$speed >= 15, labels = c("Baja", "Alta")) # levels = c("FALSE", "TRUE")
table(fspeed)
```

<sup>1</sup>Aunque si el objetivo es obtener las frecuencias de cada intervalo puede ser más eficiente emplear `hist()` con `plot = FALSE`.

```
## fspeed
## Baja Alta
##    23    27
```

En el caso de múltiples niveles se podría emplear `ifelse()` anidados:

```
fspeed <- ifelse(cars$speed < 10, "Baja",
                ifelse(cars$speed < 20, "Media", "Alta"))
fspeed <- factor(fspeed, levels = c("Baja", "Media", "Alta"))
table(fspeed)
```

```
## fspeed
## Baja Media Alta
##      6    32    12
```

Otra alternativa sería emplear la función `recode()` del paquete `car`.

NOTA: Para acceder directamente a las variables de un `data.frame` podríamos emplear la función `attach()` para añadirlo a la ruta de búsqueda y `detach()` al finalizar. Sin embargo esta forma de proceder puede causar numerosos inconvenientes, especialmente al modificar la base de datos, por lo que la recomendación sería emplear `with()`. Por ejemplo, podríamos calcular el factor anterior empleando:

```
fspeed <- with(cars, ifelse(speed < 10, "Baja",
                           ifelse(speed < 20, "Media", "Alta")))
fspeed <- factor(fspeed, levels = c("Baja", "Media", "Alta"))
table(fspeed)
```

```
## fspeed
## Baja Media Alta
##      6    32    12
```

## 2.2.2 Operaciones con casos

### 2.2.2.1 Ordenación

Continuemos con el `data.frame` `cars`. Se puede comprobar que los datos disponibles están ordenados por los valores de `speed`. A continuación haremos la ordenación utilizando los valores de `dist`. Para ello utilizaremos el conocido como vector de índices de ordenación. Este vector establece el orden en que tienen que ser elegidos los elementos para obtener la ordenación deseada. Veamos un ejemplo sencillo:

```
x <- c(2.5, 4.3, 1.2, 3.1, 5.0) # valores originales
ii <- order(x)
ii    # vector de ordenación
```

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

```
x[ii] # valores ordenados
```

```
## [1] 1.2 2.5 3.1 4.3 5.0
```

En el caso de vectores, el procedimiento anterior se podría hacer directamente con:

```
sort(x)
```

Sin embargo, para ordenar data.frames será necesario la utilización del vector de índices de ordenación. A continuación, los datos de `cars` ordenados por `dist`:

```
ii <- order(cars$dist) # Vector de índices de ordenación
cars2 <- cars[ii, ]   # Datos ordenados por dist
head(cars2)
```

```
##   speed dist velocidad distancia
## 1     4    2  6.437388 0.6096074
## 3     7    4 11.265430 1.2192148
## 2     4   10  6.437388 3.0480371
## 6     9   10 14.484124 3.0480371
## 12    12   14 19.312165 4.2672519
## 5     8   16 12.874777 4.8768593
```

### 2.2.2.2 Filtrado

El filtrado de datos consiste en elegir una submuestra que cumpla determinadas condiciones. Para ello se puede utilizar la función `subset()` (que además permite seleccionar variables).

A continuación se muestran un par de ejemplos:

```
subset(cars, dist > 85) # datos con dis>85
```

```
##   speed dist velocidad distancia
## 47    24    92  38.62433 28.04194
## 48    24    93  38.62433 28.34674
## 49    24   120  38.62433 36.57644
```

```
subset(cars, speed > 10 & speed < 15 & dist > 45) # speed en (10,15) y dist>45
```

```
##   speed dist velocidad distancia
## 19    13    46  20.92151 14.02097
## 22    14    60  22.53086 18.28822
## 23    14    80  22.53086 24.38430
```

También se pueden hacer el filtrado empleando directamente los correspondientes vectores de índices:

```
ii <- cars$dist > 85
cars[ii, ] # dis>85
```

```
##      speed dist velocidad distancia
## 47      24  92 38.62433 28.04194
## 48      24  93 38.62433 28.34674
## 49      24 120 38.62433 36.57644
```

```
ii <- cars$speed > 10 & cars$speed < 15 & cars$dist > 45
cars[ii, ] # speed en (10,15) y dist>45
```

```
##      speed dist velocidad distancia
## 19      13  46 20.92151 14.02097
## 22      14  60 22.53086 18.28822
## 23      14  80 22.53086 24.38430
```

En este caso puede ser de utilidad la función `which()`:

```
it <- which(ii)
str(it)
```

```
## int [1:3] 19 22 23
```

```
cars[it, 1:2]
```

```
##      speed dist
## 19      13  46
## 22      14  60
## 23      14  80
```

```
# rownames(cars[it, 1:2])
```

```
id <- which(!ii)
str(cars[id, 1:2])
```

```
## 'data.frame': 47 obs. of 2 variables:
## $ speed: num 4 4 7 7 8 9 10 10 10 11 ...
## $ dist : num 2 10 4 22 16 10 18 26 34 17 ...
```

```
# Equivalentemente:
```

```
str(cars[-it, 1:2])
```

```
## 'data.frame': 47 obs. of 2 variables:
## $ speed: num 4 4 7 7 8 9 10 10 10 11 ...
## $ dist : num 2 10 4 22 16 10 18 26 34 17 ...
```

```
# Se podría p.e. emplear cars[id, ] para predecir cars[it, ]$speed
# ?which.min
```

## 2.2.3 Funciones apply

### 2.2.3.1 La función apply

Una forma de evitar la utilización de bucles es utilizando la sentencia `apply` que permite evaluar una misma función en todas las filas, columnas, etc. de un array de forma simultánea.

La sintaxis de esta función es:

```
apply(X, MARGIN, FUN, ...)
```

- X: matriz (o array)
- MARGIN: Un vector indicando las dimensiones donde se aplicará la función. 1 indica filas, 2 indica columnas, y `c(1,2)` indica filas y columnas.
- FUN: función que será aplicada.
- ...: argumentos opcionales que serán usados por FUN.

Veamos la utilización de la función `apply` con un ejemplo:

```
x <- matrix(1:9, nrow = 3)
x
```

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

```
apply(x, 1, sum) # Suma por filas
```

```
## [1] 12 15 18
```

```
apply(x, 2, sum) # Suma por columnas
```

```
## [1]  6 15 24
```

```
apply(x, 2, min) # Mínimo de las columnas
```

```
## [1] 1 4 7
```

```
apply(x, 2, range) # Rango (mínimo y máximo) de las columnas
```

```
##      [,1] [,2] [,3]
## [1,]   1   4   7
## [2,]   3   6   9
```

### 2.2.3.2 Variantes de la función apply

`lapply()`:

```
# lista con las medianas de las variables
list <- lapply(cars, median)
str(list)
```

```
## List of 4
## $ speed      : num 15
## $ dist       : num 36
## $ velocidad: num 24.1
## $ distancia: num 11

sapply():
# matriz con las medias, medianas y desv. de las variables
res <- sapply(cars,
              function(x) c(mean = mean(x), median = median(x), sd = sd(x)))
# str(res)
res

##           speed      dist velocidad distancia
## mean  15.400000 42.98000 24.783945 13.100463
## median 15.000000 36.00000 24.140206 10.972933
## sd     5.287644 25.76938  8.509655  7.854602
knitr::kable(t(res), digits = 1)
```

	mean	median	sd
speed	15.4	15.0	5.3
dist	43.0	36.0	25.8
velocidad	24.8	24.1	8.5
distancia	13.1	11.0	7.9

```
cfuns <- function(x, funs = c(mean, median, sd))
  sapply(funs, function(f) f(x))
x <- 1:10
cfuns(x)

## [1] 5.50000 5.50000 3.02765
sapply(cars, cfuns)

##           speed      dist velocidad distancia
## [1,] 15.400000 42.98000 24.783945 13.100463
## [2,] 15.000000 36.00000 24.140206 10.972933
## [3,]  5.287644 25.76938  8.509655  7.854602
nfuns <- c("mean", "median", "sd")
sapply(nfuns, function(f) eval(parse(text = paste0(f, "(x)"))))

##      mean median      sd
## 5.50000 5.50000 3.02765
```

### 2.2.3.3 La función tapply

La función `tapply()` es similar a la función `apply()` y permite aplicar una función a los datos desagregados, utilizando como criterio los distintos niveles



de una variable factor. La sintaxis de esta función es como sigue:

```
tapply(X, INDEX, FUN, ...)
```

- X: matriz (o array).
- INDEX: factor indicando los grupos (niveles).
- FUN: función que será aplicada.
- ...: argumentos opcionales .

Consideremos, por ejemplo, el data.frame `ChickWeight` con datos de un experimento relacionado con la repercusión de varias dietas en el peso de pollos.

```
data(ChickWeight)
# str(ChickWeight)
head(ChickWeight)
```

```
##   weight Time Chick Diet
## 1     42   0    1    1
## 2     51   2    1    1
## 3     59   4    1    1
## 4     64   6    1    1
## 5     76   8    1    1
## 6     93  10    1    1
```

```
peso <- ChickWeight$weight
dieta <- ChickWeight$Diet
levels(dieta) <- c("Dieta 1", "Dieta 2", "Dieta 3", "Dieta 4")
tapply(peso, dieta, mean) # Peso medio por dieta
```

```
## Dieta 1 Dieta 2 Dieta 3 Dieta 4
## 102.6455 122.6167 142.9500 135.2627
```

```
tapply(peso, dieta, summary)
```

```
## $`Dieta 1`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  35.00  57.75   88.00 102.65 136.50 305.00
##
## $`Dieta 2`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  39.0   65.5   104.5  122.6 163.0 331.0
##
## $`Dieta 3`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  39.0   67.5   125.5  142.9 198.8 373.0
##
## $`Dieta 4`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  39.0   71.25 129.50 135.26 184.75 322.00
```

Otro ejemplo:

```
provincia <- as.factor(c(1, 3, 4, 2, 4, 3, 2, 1, 4, 3, 2))
levels(provincia) = c("A Coruña", "Lugo", "Orense", "Pontevedra")
hijos <- c(1, 2, 0, 3, 4, 1, 0, 0, 2, 3, 1)
data.frame(provincia, hijos)
```

```
##      provincia hijos
## 1     A Coruña     1
## 2         Orense     2
## 3 Pontevedra     0
## 4         Lugo     3
## 5 Pontevedra     4
## 6         Orense     1
## 7         Lugo     0
## 8     A Coruña     0
## 9 Pontevedra     2
## 10        Orense     3
## 11         Lugo     1
```

```
tapply(hijos, provincia, mean) # Número medio de hijos por provincia
```

```
##   A Coruña      Lugo      Orense Pontevedra
## 0.500000  1.333333  2.000000  2.000000
```

Alternativamente se podría emplear la función `aggregate()` que tiene las ventajas de admitir fórmulas y disponer de un método para series de tiempo.

## 2.2.4 Operaciones con tablas de datos

*Unir tablas:*

- `rbind()`: combina vectores, matrices, arrays o data.frames por filas.
- `cbind()`: Idem por columnas.

*Combinar tablas:*

- `match(x, table)` devuelve un vector (de la misma longitud que `x`) con las (primeras) posiciones de coincidencia de `x` en `table` (o NA, por defecto, si no hay coincidencia).

Para realizar consultas combinando tablas puede ser más cómodo el operador `%in%` (`?%in%`).

- `pmatch(x, table, ...)`: similar al anterior pero con coincidencias parciales de cadenas de texto.

## 2.3 Ejemplo WoS data

Ejemplo *wosdata.R* en *wosdata.zip*. Ver Apéndice B.

```
# library(dplyr)
# library(stringr)
# https://rubenfcasal.github.io/scimetr/articles/scimetr.html
# library(scimetr)
```

```
db <- readRDS("data/wosdata/db_udc_2015.rds")
str(db, 1)
```

```
## List of 11
## $ Docs      : 'data.frame':  856 obs. of  26 variables:
## $ Authors   : 'data.frame': 4051 obs. of  4 variables:
## $ AutDoc    : 'data.frame': 5511 obs. of  2 variables:
## $ Categories: 'data.frame':  189 obs. of  2 variables:
## $ CatDoc    : 'data.frame': 1495 obs. of  2 variables:
## $ Areas     : 'data.frame':  121 obs. of  2 variables:
## $ AreaDoc   : 'data.frame': 1364 obs. of  2 variables:
## $ Addresses : 'data.frame': 3655 obs. of  5 variables:
## $ AddAutDoc : 'data.frame': 7751 obs. of  3 variables:
## $ Journals  : 'data.frame':  520 obs. of 12 variables:
## $ label     : chr ""
## - attr(*, "variable.labels")= Named chr [1:62] "Publication type" "Author" "Book authors" "Ed
## ..- attr(*, "names")= chr [1:62] "PT" "AU" "BA" "BE" ...
## - attr(*, "class")= chr "wos.db"

variable.labels <- attr(db, "variable.labels")
knitr::kable(as.data.frame(variable.labels)) # caption = "Variable labels"
```

	variable.labels
PT	Publication type
AU	Author
BA	Book authors
BE	Editor
GP	Group author
AF	Author full
BF	Book authors fullname
CA	Corporate author
TI	Title
SO	Publication name
SE	Series title
BS	Book series
LA	Language
DT	Document type
CT	Conference title
CY	Conference year
CL	Conference place
SP	Conference sponsors
HO	Conference host
DE	Keywords
ID	Keywords Plus
AB	Abstract
C1	Addresses
RP	Reprint author
EM	Author email
RI	Researcher id numbers
OI	Orcid numbers
FU	Funding agency and grant number
FX	Funding text
CR	Cited references
NR	Number of cited references
TC	Times cited
Z9	Total times cited count
U1	Usage Count (Last 180 Days)
U2	Usage Count (Since 2013)
PU	Publisher
PI	Publisher city
PA	Publisher address
SN	ISSN
EI	eISSN
BN	ISBN
J9	Journal.ISI
J1	Journal.ISO
PD	Publication date
PY	Year published
VL	Volume
IS	Issue
PN	Part number
SU	Supplement
SI	Special issue
MA	Meeting abstract
BP	Beginning page
EP	Ending page

Documentos correspondientes a revistas:

```
# View(db$Journals)
iidj <- with(db$Journals, idj[grepl('Chem', JI)])
db$Journals$JI[iidj]

## [1] "J. Am. Chem. Soc." "Inorg. Chem."
## [3] "J. Chem. Phys." "J. Chem. Thermodyn."
## [5] "J. Solid State Chem." "Chemosphere"
## [7] "Antimicrob. Agents Chemother." "Trac-Trends Anal. Chem."
## [9] "Eur. J. Med. Chem." "J. Chem. Technol. Biotechnol."
## [11] "J. Antimicrob. Chemother." "Food Chem."
## [13] "Cancer Chemother. Pharmacol." "Int. J. Chem. Kinet."
## [15] "Chem.-Eur. J." "J. Phys. Chem. A"
## [17] "New J. Chem." "Chem. Commun."
## [19] "Chem. Eng. J." "Comb. Chem. High Throughput Screen"
## [21] "Mini-Rev. Med. Chem." "Phys. Chem. Chem. Phys."
## [23] "Org. Biomol. Chem." "J. Chem Inf. Model."
## [25] "ACS Chem. Biol." "Environ. Chem. Lett."
## [27] "Anal. Bioanal. Chem." "J. Cheminformatics"
## [29] "J. Mat. Chem. B"

idd <- with(db$Docs, idj %in% iidj)
which(idd)
```

```
## [1] 2 4 16 23 43 69 119 126 138 175 188 190 203 208 226 240 272 337 338
## [20] 341 342 357 382 385 386 387 388 394 411 412 428 460 483 518 525 584 600 604
## [39] 605 616 620 665 697 751 753 775 784 796 806 808 847 848
```

```
# View(db$Docs[idd, ])
head(db$Docs[idd, 1:3])
```

```
## idd idj
## 2 2 37
## 4 4 272
## 16 16 195
## 23 23 436
## 43 43 455
## 69 69 37
##
## 2
## 4
## 16 Reduced susceptibility to biocides in Acinetobacter baumannii: association with resistance
## 23 Two Catechol Siderophores, Acinetobac
## 43
## 69
```

Role of

Documentos correspondientes a autores:

```

# View(db$Authors)
iida <- with(db$Authors, ida[grepl('Abad', AF)])
db$Authors$AF[iida]

## [1] "Mato Abad, Virginia" "Abad, Maria-Jose" "Abad Vicente, J."
## [4] "Abada, Sabah"

idd <- with(db$AutDoc, idd[ida %in% iida])
idd

## [1] 273 291 518 586

# View(db$Docs[iida, ])
head(db$Docs[idd, 1:3])

##      idd idj
## 273 273 282
## 291 291 141
## 518 518 272
## 586 586 311
##
## 273 Classification of mild cognitive impairment and
## 291 Identifying a population of patients suitable for the implantation of a subcutan
## 518 Importance of Outer-Sphere and Aggregation Phenomena in the Relaxation
## 586 Enhanced t

```

## Capítulo 3

# Introducción al lenguaje SQL

Los sistemas de información gestionan repositorios de información en múltiples formatos, siendo el más popular las bases de datos relacionales a las que se accede mediante SQL (Structured Query Language).

El ejemplo que trabajaremos en este capítulo está disponible en Kaggle: [www.kaggle.com/gltaboada/sqlite-tutorial-in-r](https://www.kaggle.com/gltaboada/sqlite-tutorial-in-r)

### 3.1 Bases de Datos Relacionales

#### 3.1.1 Definiciones

- **Dominio:** contexto (organización, empresa, evento...) objeto de gestión de la información.
- **Dato:** hecho con significado implícito, registable, relevante en un determinado dominio.
- **Base de datos:** colección de datos de un determinado dominio relacionados entre sí, organizados de forma que sea posible manipularlos y recuperarlos de forma eficiente.
- Sistema de Gestión de Bases de Datos (**SGBD**) (en inglés **RDBMS**, Relational Database Management System): software que permite a los usuarios crear y manipular bases de datos mediante operaciones CRUD:
  - Crear / Insertar Datos (Create)
  - Consultar / Leer (Read)
  - Actualizar / Modificar (Update)
  - Eliminar (Delete)

- **Modelo de datos:** abstracción conceptual que propone una manera de organizar y manipular los datos. Definido mediante:
  - Estructura: elementos para organizar datos
  - Integridad: reglas para relaciones los elementos
  - Manipulación: operaciones sobre los datos adaptadas a la estructura y reglas
- Modelo de datos conceptual **Entidad Relación** (entidades, relaciones, atributos)
- Modelo de datos lógico o de representación (**modelo relacional** de Codd)
  - Datos en relaciones (tablas)
  - Base matemática formal
  - Flexible
- Modelo de datos físico (tal y como se almacenan los datos)

Una fila de la tabla (relación) es una tupla y una columna un atributo (ver Figura 3.1).

(ver Figura 3.1)

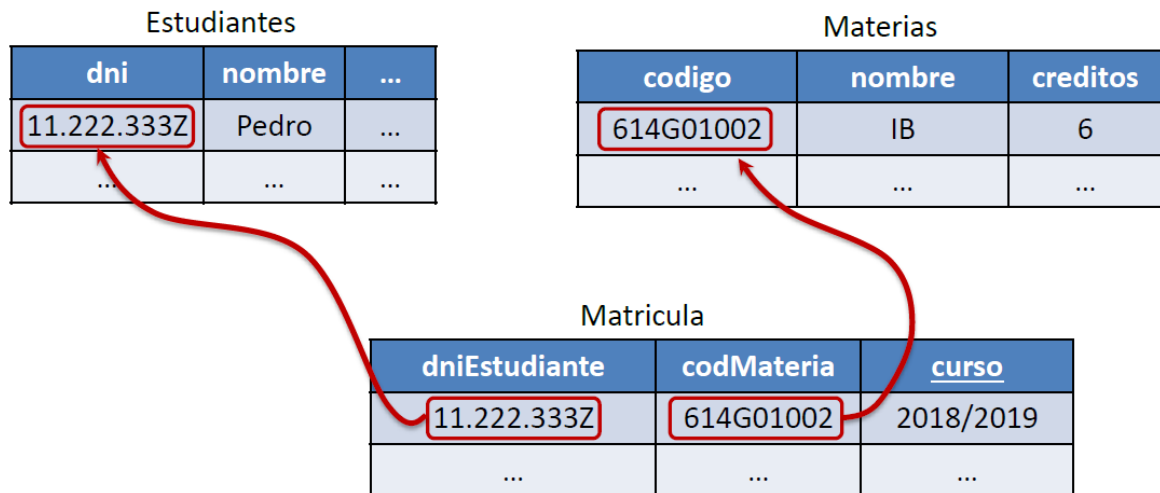
(ver Figura 3.1)

Estudiantes			
dni	nombre	apellidos	email
11.222.333Z	Pedro	Pérez Pérez	infppp@udc.es
22.333.444Z	Ana	López Pérez	infalp@udc.es
11.888.999Z	Alberto	López López	infall@udc.es

Figura 3.1: Esquema de una relación.

Una base de datos es un conjunto de tablas (al menos una).





La tabla no es una relación porque la relación es un conjunto sin orden y una tabla puede tener filas repetidas y tiene orden.

- 
- **Esquema:** estructura de la base de datos
  - **Estado:** contenido de la base de datos
  - Restricción de **integridad:** regla que debe cumplir la información registrada en la base de datos para garantizar la integridad de la información.

Cualquier Base de Datos basada en el modelo relacional ha de cumplir como mínimo estas restricciones (además de las propias del dominio):

- Restricción de dominio: el valor de cada atributo debe de ser único (teléfono, no valor único), no descomponible (nombre completo descomponible en nombre y apellidos, domicilio en calle, CP, localidad, etc...)
- Una relación es un conjunto de tuplas, por tanto todas las tuplas son distintas.
- Una **superclave** es un subconjunto de atributos tal que no existen dos tuplas con la misma superclave.

Ejercicio. En la relación Empleado(dni, nombre, apellidos, email)  
¿cuántas superclaves existen?

- Una **clave candidata** es una superclave mínima (superclave mínima es la clave a la que no se le puede eliminar un atributo).

¿Cuántas claves candidatas hay en el ejemplo anterior?

- **Clave primaria** es la clave candidata que elegimos que identificar de forma unívoca las tuplas de una relación. Restricción de integridad de entidad: Ningún valor de la clave primaria puede ser un valor nulo.
- **Clave foránea** es un conjunto de atributos de una relación  $R_1$  que, para cada tupla, identifican a otra tupla de una relación  $R_2$  con la que está relacionada. La Restricción de integridad referencial nos dice que la clave foránea ha de corresponderse con la clave primaria de  $R_2$ , y si la clave foránea no es nula ha de referir a una tupla en  $R_2$ .

<u>dni</u>	nombre	...
11.222.333Z	Pedro	...
22.333.444Z	Ana	...
11.888.999Z	Alberto	...

<u>codigo</u>	nombre	creditos
614G01002	Informática Básica	6
614G01013	Bases de Datos	6
650G01022	Bases de Datos	6

Identifica al estudiante que realizó la matrícula

<u>dniEstudiante</u>	<u>codMateria</u>	<u>curso</u>
11.222.333Z	614G01002	2018/2019

<u>dni</u>	nombre	apellidos	despacho
11.222.333Z	Pedro	Pérez Pérez	D.01
22.333.444Z	Ana	López Pérez	D.01
11.888.999Z	Alberto	López López	

<u>numero</u>	piso	capacidad
D.01	1º	2
D.02	1º	2
D.03	2º	1

La foránea *despacho* toma valor nulo en esta tupla  
(El docente no tiene todavía despacho asignado)  
**ES CORRECTO Y NO VIOLA LA RESTRICCIÓN**

Si borramos/actualizamos un valor de clave foránea podemos: (a) prohibir el cambio, o (b) poner a nulo la clave foránea (borrado) o propagar el cambio (modificación).

- 
- Ventajas de SGBD:
    - Administración centralizada de los datos (por un administrador en un servidor/plataforma central que evita la información en silos - redundante/inconsistente)
    - Desacoplado del almacenamiento físico de los datos (no es necesario conocerlo)
    - Simplicidad de acceso (ODBC + SQL, lenguaje declarativo)
    - Control de integridad (restricciones genéricas, integridad de entidad y referencial, de dominio, y las del dominio en software)
    - Control de acceso concurrente (evita inconsistencia)
    - Seguridad (autenticación, roles de acceso)
    - Recuperación ante fallos (backup, logs y transacciones -rollback-)

## 3.2 Sintaxis SQL

A continuación 27 cláusulas SQL básicas

### 3.2.1 Extracción SQL (11 statements)

```

SELECT column1, column2....columnN
FROM table_name;

SELECT DISTINCT column1, column2....columnN
FROM table_name;

SELECT column1, column2....columnN
FROM table_name
WHERE CONDITION;

SELECT column1, column2....columnN
FROM table_name
WHERE CONDITION-1 {AND|OR} CONDITION-2;

SELECT column1, column2....columnN
FROM table_name
WHERE column_name IN (val-1, val-2,...val-N);

SELECT column1, column2....columnN
FROM table_name
WHERE column_name BETWEEN val-1 AND val-2;
```

```
SELECT column1, column2...columnN
FROM table_name
WHERE column_name LIKE { PATTERN };

SELECT column1, column2...columnN
FROM table_name
WHERE CONDITION
ORDER BY column_name {ASC|DESC};

SELECT SUM(column_name)
FROM table_name
WHERE CONDITION
GROUP BY column_name;

SELECT COUNT(column_name)
FROM table_name
WHERE CONDITION;

SELECT SUM(column_name)
FROM table_name
WHERE CONDITION
GROUP BY column_name
HAVING (arithmetic function condition);
```

### 3.2.2 Crear/Actualizar/Borrar tablas SQL (8 statements)

```
CREATE TABLE table_name(
column1 datatype,
column2 datatype,
column3 datatype,
.....
columnN datatype,
PRIMARY KEY( one or more columns )
);

DROP TABLE table_name;

CREATE UNIQUE INDEX index_name
ON table_name ( column1, column2,...columnN);

ALTER TABLE table_name
DROP INDEX index_name;

DESC table_name;
```

```
TRUNCATE TABLE table_name;

ALTER TABLE table_name {ADD|DROP|MODIFY} column_name {data_type};

ALTER TABLE table_name RENAME TO new_table_name;
```

### 3.2.3 Añadir/Actualizar/Borrar tuplas en SQL (3 statements)

```
INSERT INTO table_name( column1, column2...columnN)
VALUES ( value1, value2...valueN);

UPDATE table_name
SET column1 = value1, column2 = value2...columnN=valueN
[ WHERE CONDITION ];

DELETE FROM table_name
WHERE {CONDITION};
```

### 3.2.4 Gestión Bases de Datos (5 statements)

```
CREATE DATABASE database_name;

DROP DATABASE database_name;

USE database_name;

COMMIT;

ROLLBACK;
```

### 3.2.5 Ejemplos de consultas SQL

```
SELECT Nombre, Apellido1, Apellido2, Municipio, Provincia
FROM Cliente
WHERE Municipio = 'Lugo'
ORDER BY Apellido1

INSERT Proveedor(Nombre, PersonaContacto, Ciudad, País)
VALUES ('Café Candelas', 'Ivana Candelas', 'Lugo', 'España')

UPDATE Pedidos
SET Cantidad = 2
```

```
WHERE IdProducto = 963

DELETE Cliente
WHERE Email = 'alexandregb@gmail.com'
```

### 3.3 Conexión con bases de datos desde R

#### 3.3.1 Introducción a SQL en R

SQL se usa para manipular datos dentro de una base de datos. Si la base de datos no es muy grande se puede cargar toda en un `data.frame`. No obstante, por escalabilidad y offloading de la carga de trabajo al servidor SGBD utilizaremos SQL.

Existen varios SGBD (SQLite, Microsoft SQL Server, MySQL, PostgreSQL, etc) los cuales comparten el soporte de SQL (en concreto ANSI SQL) aunque cada gestor extiende SQL de formas sutiles buscando minar cierta portabilidad de código (*vendor-locking*). En efecto, un código SQL desarrollado para SQLite es probable que falle con MySQL aunque tras aplicar ligeras modificaciones ya funcionará. Asimismo el mecanismo de conexión, configuración, rendimiento y operación suele diferir entre SGBD.

A continuación se lista una serie de paquetes utilizados en el acceso a los datos, lo que suele ser el principal esfuerzo a realizar cuando se trabaja con SGBD:

- DBI
- RODBC
- dbConnect
- RSQLite
- RMySQL
- RPostgreSQL

#### 3.3.2 El paquete `sqldf`

A continuación se presenta una serie de ejercicios con la sintaxis de SQL operando sobre un `data.frame` con el paquete `sqldf`. Esto inicialmente no incluye los detalles de conectarse a un SGBD, ni modificar los datos, solamente el uso de SQL para extraer datos con el objetivo de ser analizados en R.

```
library(sqldf)
```

```
sqldf('SELECT age, circumference FROM Orange WHERE Tree = 1 ORDER BY circumference ASC')
```

```
##      age circumference
## 1  118             30
## 2  484             58
## 3  664             87
```

```
## 4 1004          115
## 5 1231          120
## 6 1372          142
## 7 1582          145
```

### 3.3.3 SQL Queries

El comando inicial es SELECT. SQL no es case-sensitive, por lo que esto va a funcionar:

```
sqldf("SELECT * FROM iris")
sqldf("select * from iris")
```

pero lo siguiente no va a funcionar (a menos que tengamos un objeto IRIS:

```
sqldf("SELECT * FROM IRIS")
```

La sintaxis básica de SELECT es:

```
SELECT variable1, variable2 FROM data
```

#### 3.3.3.1 Asterisco/Wildcard

Lo extrae todo

```
bod2 <- sqldf('SELECT * FROM BOD')
```

#### 3.3.3.2 Limit

Limita el número de resultados

```
sqldf('SELECT * FROM iris LIMIT 5')
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1          3.5          1.4          0.2 setosa
## 2          4.9          3.0          1.4          0.2 setosa
## 3          4.7          3.2          1.3          0.2 setosa
## 4          4.6          3.1          1.5          0.2 setosa
## 5          5.0          3.6          1.4          0.2 setosa
```

#### 3.3.3.3 Order By

Ordena las variables

```
ORDER BY var1 {ASC/DESC}, var2 {ASC/DESC}
```

```
sqldf("SELECT * FROM Orange ORDER BY age ASC, circumference DESC LIMIT 5")
```

```
## Tree age circumference
## 1     2 118           33
```

```
## 2    4 118          32
## 3    1 118          30
## 4    3 118          30
## 5    5 118          30
```

### 3.3.3.4 Where

Sentencias condicionales, donde se puede incorporar operadores lógicos AND y OR, expresando el orden de evaluación con paréntesis en caso de ser necesario.

```
sqldf('SELECT demand FROM BOD WHERE Time < 3')
```

```
##    demand
## 1      8.3
## 2     10.3
```

```
sqldf('SELECT * FROM rock WHERE (peri > 5000 AND shape < .05) OR perm > 1000')
```

```
##    area    peri    shape perm
## 1 5048  941.543 0.328641 1300
## 2 1016  308.642 0.230081 1300
## 3 5605 1145.690 0.464125 1300
## 4 8793 2280.490 0.420477 1300
```

Y extendiendo su uso con IN o LIKE (es último sólo con %), pudiendo aplicárseles el NOT:

```
sqldf('SELECT * FROM BOD WHERE Time IN (1,7)')
```

```
##    Time demand
## 1     1      8.3
## 2     7     19.8
```

```
sqldf('SELECT * FROM BOD WHERE Time NOT IN (1,7)')
```

```
##    Time demand
## 1     2     10.3
## 2     3     19.0
## 3     4     16.0
## 4     5     15.6
```

```
sqldf('SELECT * FROM chickwts WHERE feed LIKE "%bean" LIMIT 5')
```

```
##    weight    feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
```



```
squidf('SELECT * FROM chickwts WHERE feed NOT LIKE "%bean" LIMIT 5')
```

```
## weight feed
## 1 309 linseed
## 2 229 linseed
## 3 181 linseed
## 4 141 linseed
## 5 260 linseed
```

## 3.4 Ejemplo Scopus data

Ver ejemplo *citan.zip* y Apéndice C.

“If your data fits in memory there is no advantage to putting it in a database: it will only be slower and more frustrating”

— Hadley Wickham – <https://dbplyr.tidyverse.org/articles/dbplyr.html>

## 3.5 Ejercicios SQL con RSQLite

### 3.5.1 Setup de RSQLite

Vamos a utilizar RSQLite desde Kaggle. Pero si lo querés instalar en local La información para su instalación está en el siguiente enlace.

```
library(DBI)

# Create an ephemeral in-memory RSQLite database
con <- dbConnect(RSQLite::SQLite(), ":memory:")
dbListTables(con)

## character(0)

dbWriteTable(con, "mtcars", mtcars)
dbListTables(con)

## [1] "mtcars"

dbListFields(con, "mtcars")

## [1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"
## [11] "carb"

dbReadTable(con, "mtcars")

## mpg cyl disp hp drat wt qsec vs am gear carb
## 1 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4
## 2 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4
```

```
## 3 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1
## 4 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1
## 5 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2
## 6 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1
## 7 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4
## 8 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2
## 9 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2
## 10 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4
## 11 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4
## 12 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3
## 13 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3
## 14 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3
## 15 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4
## 16 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4
## 17 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4
## 18 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
## 19 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2
## 20 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
## 21 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1
## 22 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2
## 23 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2
## 24 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4
## 25 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2
## 26 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1
## 27 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2
## 28 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2
## 29 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4
## 30 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6
## 31 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8
## 32 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
```

*# You can fetch all results:*

```
res <- dbSendQuery(con, "SELECT * FROM mtcars WHERE cyl = 4")
dbFetch(res)
```

```
##      mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## 1  22.8   4  108.0  93 3.85 2.320 18.61 1  1   4    1
## 2  24.4   4  146.7  62 3.69 3.190 20.00 1  0   4    2
## 3  22.8   4  140.8  95 3.92 3.150 22.90 1  0   4    2
## 4  32.4   4   78.7  66 4.08 2.200 19.47 1  1   4    1
## 5  30.4   4   75.7  52 4.93 1.615 18.52 1  1   4    2
## 6  33.9   4   71.1  65 4.22 1.835 19.90 1  1   4    1
## 7  21.5   4  120.1  97 3.70 2.465 20.01 1  0   3    1
## 8  27.3   4   79.0  66 4.08 1.935 18.90 1  1   4    1
## 9  26.0   4  120.3  91 4.43 2.140 16.70 0  1   5    2
## 10 30.4   4   95.1 113 3.77 1.513 16.90 1  1   5    2
```

```
## 11 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
```

```
dbClearResult(res)
```

```
# Or a chunk at a time
```

```
res <- dbSendQuery(con, "SELECT * FROM mtcars WHERE cyl = 4")
```

```
while(!dbHasCompleted(res)){
  chunk <- dbFetch(res, n = 5)
  print(nrow(chunk))
}
```

```
## [1] 5
```

```
## [1] 5
```

```
## [1] 1
```

```
# Clear the result
```

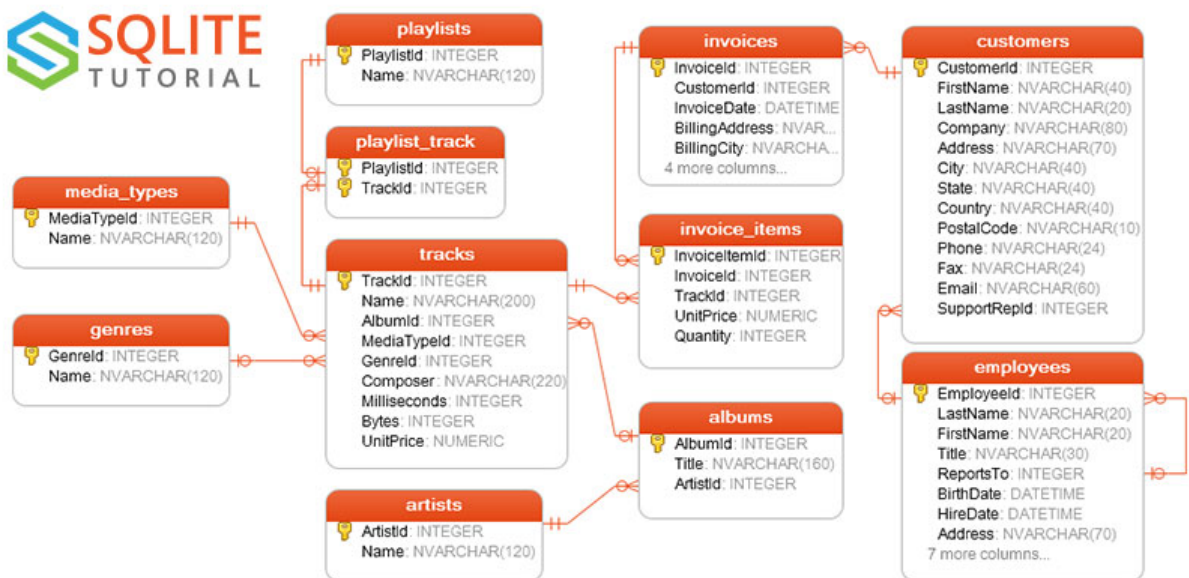
```
dbClearResult(res)
```

```
# Disconnect from the database
```

```
dbDisconnect(con)
```

## 3.6 Práctica 1: SQL

Vamos a utilizar la base de datos Chinook del tutorial de SQLite



Los ejercicios pedidos en Kaggle [kaggle.com/gltaboada/sqlite-tutorial-in-r](https://www.kaggle.com/gltaboada/sqlite-tutorial-in-r) se entregarán preferentemente antes del 14/10 compartiendo un notebook con las

soluciones (¡notebooke privado!) con el usuario **gltaboda**. Antes me tenéis que enviar un email comunicando qué usuario tenéis cada uno. En caso de incidencia me podéis mandar un notebook descargado (.ipynb), o el mecanismo que hayamos acordado previamente.

# Capítulo 4

## Manipulación de datos con dplyr

Working draft...

En este capítulo se realiza una breve introducción al paquete `dplyr`. Para más información, ver por ejemplo la ‘vignette’ del paquete `Introduction to dplyr`, o el Capítulo 5 `Data transformation` del libro `R for Data Science`<sup>1</sup>.

### 4.1 El paquete dplyr

```
library(dplyr)
```

`dplyr` permite sustituir funciones base de R (como `split()`, `subset()`, `apply()`, `sapply()`, `lapply()`, `tapply()` y `aggregate()`) mediante una “gramática” más sencilla para la manipulación de datos:

- `select()` seleccionar variables/columnas (también `rename()`).
- `mutate()` crear variables/columnas (también `transmute()`).
- `filter()` seleccionar casos/filas (también `slice()`).
- `arrange()` ordenar o organizar casos/filas.
- `summarise()` resumir valores.
- `group_by()` permite operaciones por grupo empleando el concepto “dividir-aplicar-combinar” (`ungroup()` elimina el agrupamiento).

---

<sup>1</sup>Una alternativa (más rápida) es emplear `data.table`.

Puede trabajar con conjuntos de datos en distintos formatos:

- `data.frame`, `data.table`, `tibble`, ...
- bases de datos relacionales (lenguaje SQL); paquete `dbplyr`, ...
- bases de datos *Hadoop*:
  - `plyrmr`,
  - `sparklyr`
  - ...

En lugar de operar sobre vectores como las funciones base, opera sobre objetos de este tipo (solo nos centraremos en `data.frame`).

### 4.1.1 Datos de ejemplo

El fichero `empleados.RData` contiene datos de empleados de un banco. Supongamos por ejemplo que estamos interesados en estudiar si hay discriminación por cuestión de sexo o raza.

## 4.2 Operaciones con variables (columnas)

### 4.2.1 Seleccionar variables con `select()`

```
emplea2 <- select(empleados, id, sexo, minoria, tiempemp, salini, salario)
head(emplea2)
```

```
##   id  sexo minoria tiempemp salini salario
## 1  1 Hombre     No      98  27000  57000
## 2  2 Hombre     No      98  18750  40200
## 3  3 Mujer     No      98  12000  21450
## 4  4 Mujer     No      98  13200  21900
## 5  5 Hombre     No      98  21000  45000
## 6  6 Hombre     No      98  13500  32100
```

Se puede cambiar el nombre (ver también `?rename()`)

```
head(select(empleados, sexo, noblanca = minoria, salario))
```

```
##      sexo noblanca salario
## 1 Hombre     No  57000
## 2 Hombre     No  40200
## 3 Mujer     No  21450
## 4 Mujer     No  21900
## 5 Hombre     No  45000
## 6 Hombre     No  32100
```

Se pueden emplear los nombres de variables como índices:

```
head(select(empleados, sexo:salario))
```

```
##      sexo   fechnac educ      catlab salario
## 1 Hombre 1952-02-03  15      Directivo  57000
## 2 Hombre 1958-05-23  16 Administrativo  40200
## 3 Mujer  1929-07-26  12 Administrativo  21450
## 4 Mujer  1947-04-15   8 Administrativo  21900
## 5 Hombre 1955-02-09  15 Administrativo  45000
## 6 Hombre 1958-08-22  15 Administrativo  32100
```

```
head(select(empleados, -(sexo:salario)))
```

```
##   id salini tiempemp expprev minoria   sexoraza
## 1  1  27000      98    144      No Blanca varón
## 2  2  18750      98     36      No Blanca varón
## 3  3  12000      98    381      No Blanca mujer
## 4  4  13200      98    190      No Blanca mujer
## 5  5  21000      98    138      No Blanca varón
## 6  6  13500      98     67      No Blanca varón
```

Hay opciones para considerar distintos criterios: `starts_with()`, `ends_with()`, `contains()`, `matches()`, `one_of()` (ver `?select`).

```
head(select(empleados, starts_with("s"))) )
```

```
##      sexo salario salini   sexoraza
## 1 Hombre   57000 27000 Blanca varón
## 2 Hombre   40200 18750 Blanca varón
## 3 Mujer    21450 12000 Blanca mujer
## 4 Mujer    21900 13200 Blanca mujer
## 5 Hombre   45000 21000 Blanca varón
## 6 Hombre   32100 13500 Blanca varón
```

### 4.2.2 Generar nuevas variables con `mutate()`

```
head(mutate(emplea2, incsal = salario - salini, tsal = incsal/tiempemp ))
```

```
##   id  sexo minoria tiempemp salini salario incsal   tsal
## 1  1 Hombre      No      98 27000  57000 30000 306.12245
## 2  2 Hombre      No      98 18750  40200 21450 218.87755
## 3  3 Mujer      No      98 12000  21450  9450  96.42857
## 4  4 Mujer      No      98 13200  21900  8700  88.77551
## 5  5 Hombre      No      98 21000  45000 24000 244.89796
## 6  6 Hombre      No      98 13500  32100 18600 189.79592
```

## 4.3 Operaciones con casos (filas)

### 4.3.1 Seleccionar casos con filter()

```
head(filter(emplea2, sexo == "Mujer", minoria == "Sí"))
```

```
##   id  sexo minoria tiempemp salini salario
## 1  14 Mujer      Sí      98  16800  35100
## 2  23 Mujer      Sí      97  11100  24000
## 3  24 Mujer      Sí      97   9000  16950
## 4  25 Mujer      Sí      97   9000  21150
## 5  40 Mujer      Sí      96   9000  19200
## 6  41 Mujer      Sí      96  11550  23550
```

### 4.3.2 Organizar casos con arrange()

```
head(arrange(emplea2, salario))
```

```
##   id  sexo minoria tiempemp salini salario
## 1 378 Mujer      No       70  10200  15750
## 2 338 Mujer      No       74  10200  15900
## 3  90 Mujer      No       92   9750  16200
## 4 224 Mujer      No       82  10200  16200
## 5 411 Mujer      No       68  10200  16200
## 6 448 Mujer      Sí       66  10200  16350
```

```
head(arrange(emplea2, desc(salini), salario))
```

```
##   id  sexo minoria tiempemp salini salario
## 1  29 Hombre      No       96  79980 135000
## 2 343 Hombre      No       73  60000 103500
## 3 205 Hombre      No       83  52500  66750
## 4 160 Hombre      No       86  47490  66000
## 5 431 Hombre      No       66  45000  86250
## 6  32 Hombre      No       96  45000 110625
```

## 4.4 Resumir valores con summarise()

```
summarise(empleados, sal.med = mean(salario), n = n())
```

```
##   sal.med  n
## 1 34419.57 474
```



## 4.5 Agrupar casos con `group_by()`

```
summarise(group_by(empleados, sexo, minoria), sal.med = mean(salario), n = n())

## `summarise()` regrouping output by 'sexo' (override with ` .groups ` argument)

## # A tibble: 4 x 4
## # Groups:   sexo [2]
##   sexo  minoria sal.med    n
##   <fct> <fct>    <dbl> <int>
## 1 Hombre No      44475.   194
## 2 Hombre Sí      32246.    64
## 3 Mujer  No      26707.   176
## 4 Mujer  Sí      23062.    40
```

## 4.6 Operador `pipe %>%` (tubería, redirección)

Este operador le permite canalizar la salida de una función a la entrada de otra función. `segundo(primer(datos))` se traduce en `datos %>% primero %>% segundo` (lectura de funciones de izquierda a derecha).

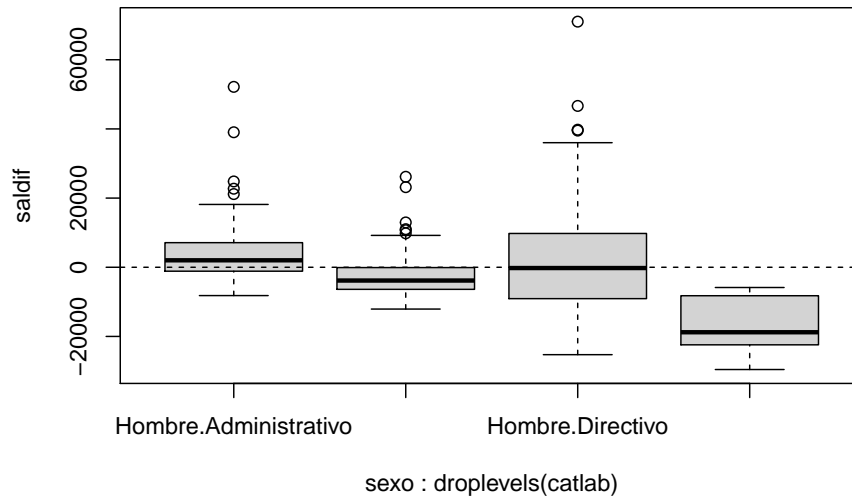
Ejemplos:

```
empleados %>% filter(catlab == "Directivo") %>%
  group_by(sexo, minoria) %>%
  summarise(sal.med = mean(salario), n = n())

## `summarise()` regrouping output by 'sexo' (override with ` .groups ` argument)

## # A tibble: 3 x 4
## # Groups:   sexo [2]
##   sexo  minoria sal.med    n
##   <fct> <fct>    <dbl> <int>
## 1 Hombre No      65684.    70
## 2 Hombre Sí      76038.     4
## 3 Mujer  No      47214.    10

empleados %>% select(sexo, catlab, salario) %>%
  filter(catlab != "Seguridad") %>%
  group_by(catlab) %>%
  mutate(saldif = salario - mean(salario)) %>%
  ungroup() %>%
  boxplot(saldif ~ sexo*droplevels(catlab), data = .)
abline(h = 0, lty = 2)
```



## 4.7 Operaciones con tablas de datos

Se emplean funciones `xxx_join()` (ver la documentación del paquete `Join two tbls together`, o la vignette `Two-table verbs`):

- `inner_join()`: devuelve las filas de `x` que tienen valores coincidentes en `y`, y todas las columnas de `x` e `y`. Si hay varias coincidencias entre `x` e `y`, se devuelven todas las combinaciones.
- `left_join()`: devuelve todas las filas de `x` y todas las columnas de `x` e `y`. Las filas de `x` sin correspondencia en `y` contendrán `NA` en las nuevas columnas. Si hay varias coincidencias entre `x` e `y`, se devuelven todas las combinaciones (duplicando las filas). `right_join()` hace lo contrario, devuelve todas las filas de `y`, y `full_join()` devuelve todas las filas de `x` e `y` (duplicando o asignando `NA` si es necesario).
- `semi_join()`: devuelve las filas de `x` que tienen valores coincidentes en `y`, manteniendo sólo las columnas de `x` (al contrario que `inner_join()` no duplica filas). `anti_join()` hace lo contrario, devuelve las filas sin correspondencia.

El parámetro `by` determina las variables clave para las correspondencias. Si no se establece se considerarán todas las que tengan el mismo nombre en ambas tablas. Se puede establecer a un vector de nombres coincidentes y en caso de que los nombres sean distintos a un vector con nombres de la forma `c("clave_x" = "clave_y")`.

Adicionalmente, si las tablas `x` e `y` tienen las mismas variables, se pueden combinar las observaciones con operaciones de conjuntos:

- `intersect(x, y)`: observaciones en `x` y en `y`.
- `union(x, y)`: observaciones en `x` o `y` no duplicadas.
- `setdiff(x, y)`: observaciones en `x` pero no en `y`.

## 4.8 Bases de datos con dplyr

Algunos enlaces:

- Databases using R
  - dplyr as a database interface
  - Databases using dplyr
- Introduction to dbplyr
- Data Carpentry
  - SQL databases and R,
- R and Data – When Should we Use Relational Databases?

### 4.8.1 Ejemplos (Práctica 1)

Como ejemplo emplearemos los ejercicios de la Práctica 1.

```
# install.packages('dbplyr')
library(dplyr)
library(dbplyr)
```

Conectar la base de datos:

```
chinook <- DBI::dbConnect(RSQLite::SQLite(), "data/chinook.db")
```

Listar tablas:

```
src_dbi(chinook)
```

```
## src:  sqlite 3.33.0 [C:\Users\Guillermo LT\Documents\GitHub\tgdbook\data\chinook.db]
## tbls: albums, artists, customers, employees, genres, invoice_items, invoices,
##      media_types, playlist_track, playlists, sqlite_sequence, sqlite_stat1, tracks
```

Enlazar una tabla:

```
invoices <- tbl(chinook, "invoices")
invoices
```

```
## # Source:   table<invoices> [?? x 9]
## # Database: sqlite 3.33.0 [C:\Users\Guillermo
```

```
## # LT\Documents\GitHub\tgdbook\data\chinook.db]
## InvoiceId CustomerId InvoiceDate BillingAddress BillingCity BillingState
## <int> <int> <chr> <chr> <chr> <chr>
## 1 1 2 2009-01-01~ Theodor-Heuss~ Stuttgart <NA>
## 2 2 4 2009-01-02~ Ullevålsveien~ Oslo <NA>
## 3 3 8 2009-01-03~ Grétrystraat ~ Brussels <NA>
## 4 4 14 2009-01-06~ 8210 111 ST NW Edmonton AB
## 5 5 23 2009-01-11~ 69 Salem Stre~ Boston MA
## 6 6 37 2009-01-19~ Berger Straße~ Frankfurt <NA>
## 7 7 38 2009-02-01~ Barbarossastr~ Berlin <NA>
## 8 8 40 2009-02-01~ 8, Rue Hanovre Paris <NA>
## 9 9 42 2009-02-02~ 9, Place Loui~ Bordeaux <NA>
## 10 10 46 2009-02-03~ 3 Chatham Str~ Dublin Dublin
## # ... with more rows, and 3 more variables: BillingCountry <chr>,
## # BillingPostalCode <chr>, Total <dbl>
```

Ojo [?? x 9]: de momento no conoce el número de filas.

```
nrow(invoices)
```

```
## [1] NA
```

Mostrar la consulta SQL:

```
show_query(head(invoices))
```

```
## <SQL>
## SELECT *
## FROM `invoices`
## LIMIT 6
```

```
str(head(invoices))
```

```
## List of 2
## $ src:List of 2
## ..$ con :Formal class 'SQLiteConnection' [package "RSQLite"] with 7 slots
## .. ..@ ptr :<externalptr>
## .. ..@ dbname : chr "C:\\Users\\Guillermo LT\\Documents\\GitHub\\t
## .. ..@ loadable.extensions: logi TRUE
## .. ..@ flags : int 70
## .. ..@ vfs : chr ""
## .. ..@ ref :<environment: 0x000000001de17450>
## .. ..@ bigint : chr "integer64"
## ..$ disco: NULL
## ..- attr(*, "class")= chr [1:4] "src_SQLiteConnection" "src_dbi" "src_sql" "src"
## $ ops:List of 4
## ..$ name: chr "head"
## ..$ x :List of 2
## .. ..$ x : 'ident' chr "invoices"
```

```
##   ...$ vars: chr [1:9] "InvoiceId" "CustomerId" "InvoiceDate" "BillingAddress" ...
##   ...- attr(*, "class")= chr [1:3] "op_base_remote" "op_base" "op"
##   ...$ dots: list()
##   ...$ args:List of 1
##   ...$ n: num 6
##   ...- attr(*, "class")= chr [1:3] "op_head" "op_single" "op"
## - attr(*, "class")= chr [1:5] "tbl_SQLiteConnection" "tbl_dbi" "tbl_sql" "tbl_lazy" ...
```

Al trabajar con bases de datos, dplyr intenta ser lo más vago posible:

- No exporta datos a R a menos que se pida explícitamente (`collect()`).
- Retrasa cualquier operación lo máximo posible: agrupa todo lo que se desea hacer y luego hace una única petición a la base de datos.

```
invoices %>% head %>% collect
```

```
## # A tibble: 6 x 9
##   InvoiceId CustomerId InvoiceDate BillingAddress BillingCity BillingState
##   <int>      <int> <chr>      <chr>          <chr>      <chr>
## 1         1         2 2009-01-01~ Theodor-Heuss~ Stuttgart <NA>
## 2         2         4 2009-01-02~ Ullevålsveien~ Oslo      <NA>
## 3         3         8 2009-01-03~ Grétrystraat ~ Brussels <NA>
## 4         4        14 2009-01-06~ 8210 111 ST NW Edmonton AB
## 5         5        23 2009-01-11~ 69 Salem Stre~ Boston    MA
## 6         6        37 2009-01-19~ Berger Straße~ Frankfurt <NA>
## # ... with 3 more variables: BillingCountry <chr>, BillingPostalCode <chr>,
## #   Total <dbl>
```

```
invoices %>% count # número de filas
```

```
## # Source:   lazy query [?? x 1]
## # Database:  sqlite 3.33.0 [C:\Users\Guillermo
## #   LT\Documents\GitHub\tgdbook\data\chinook.db]
##         n
##   <int>
## 1    412
```

1. Conocer el importe mínimo, máximo y la media de las facturas

```
res <- invoices %>% summarise(min = min(Total, na.rm = TRUE),
                              max = max(Total, na.rm = TRUE), med = mean(Total, na.rm = TRUE))
show_query(res)
```

```
## <SQL>
## SELECT MIN(`Total`) AS `min`, MAX(`Total`) AS `max`, AVG(`Total`) AS `med`
## FROM `invoices`
```

```
res %>% collect
```

```
## # A tibble: 1 x 3
```

```
##      min  max  med
## <dbl> <dbl> <dbl>
## 1  0.99 25.9 5.65
```

2. Conocer el total de las facturas de cada uno de los países.

```
res <- invoices %>% group_by(BillingCountry) %>%
  summarise(n = n(), total = sum(Total, na.rm = TRUE))
show_query(res)
```

```
## <SQL>
## SELECT `BillingCountry`, COUNT(*) AS `n`, SUM(`Total`) AS `total`
## FROM `invoices`
## GROUP BY `BillingCountry`
```

```
res %>% collect
```

```
## # A tibble: 24 x 3
##   BillingCountry     n total
##   <chr>           <int> <dbl>
## 1 Argentina         7  37.6
## 2 Australia         7  37.6
## 3 Austria           7  42.6
## 4 Belgium           7  37.6
## 5 Brazil           35 190.
## 6 Canada           56 304.
## 7 Chile             7  46.6
## 8 Czech Republic   14  90.2
## 9 Denmark           7  37.6
## 10 Finland          7  41.6
## # ... with 14 more rows
```

3. Obtener el listado de países junto con su facturación media, ordenado

- (a) alfabéticamente por país

```
res <- invoices %>% group_by(BillingCountry) %>%
  summarise(n = n(), med = mean(Total, na.rm = TRUE)) %>%
  arrange(BillingCountry)
show_query(res)
```

```
## <SQL>
## SELECT `BillingCountry`, COUNT(*) AS `n`, AVG(`Total`) AS `med`
## FROM `invoices`
## GROUP BY `BillingCountry`
## ORDER BY `BillingCountry`
```

```
res %>% collect
```

```
## # A tibble: 24 x 3
```

```
##   BillingCountry    n  med
##   <chr>           <int> <dbl>
##  1 Argentina       7  5.37
##  2 Australia        7  5.37
##  3 Austria           7  6.09
##  4 Belgium          7  5.37
##  5 Brazil           35  5.43
##  6 Canada           56  5.43
##  7 Chile            7  6.66
##  8 Czech Republic  14  6.45
##  9 Denmark          7  5.37
## 10 Finland          7  5.95
## # ... with 14 more rows
```

- (b) decrecientemente por importe de facturación media

```
invoices %>% group_by(BillingCountry) %>%
  summarise(n = n(), med = mean(Total, na.rm = TRUE)) %>%
  arrange(desc(med)) %>% collect
```

```
## # A tibble: 24 x 3
##   BillingCountry    n  med
##   <chr>           <int> <dbl>
##  1 Chile            7  6.66
##  2 Ireland          7  6.52
##  3 Hungary          7  6.52
##  4 Czech Republic  14  6.45
##  5 Austria          7  6.09
##  6 Finland          7  5.95
##  7 Netherlands     7  5.80
##  8 India            13  5.79
##  9 USA              91  5.75
## 10 Norway           7  5.66
## # ... with 14 more rows
```

4. Obtener un listado con Nombre y Apellidos de cliente y el importe de cada una de sus facturas (Hint: WHERE customer.CustomerID=invoices.CustomerID)

```
customers <- tbl(chinook, "customers")
tbl_vars(customers)
```

```
## <dplyr:::vars>
## [1] "CustomerId" "FirstName" "LastName" "Company" "Address"
## [6] "City" "State" "Country" "PostalCode" "Phone"
## [11] "Fax" "Email" "SupportRepId"
```

```
res <- customers %>% inner_join(invoices, by = "CustomerId") %>% select(FirstName, LastName,
show_query(res)
```

```
## <SQL>
## SELECT `FirstName`, `LastName`, `Country`, `Total`
## FROM (SELECT `LHS`.`CustomerId` AS `CustomerId`, `FirstName`, `LastName`, `Comp
## FROM `customers` AS `LHS`
## INNER JOIN `invoices` AS `RHS`
## ON (`LHS`.`CustomerId` = `RHS`.`CustomerId`)
## )
```

```
res %>% collect
```

```
## # A tibble: 412 x 4
##   FirstName LastName Country Total
##   <chr>      <chr>      <chr> <dbl>
## 1 Luís      Gonçalves Brazil   3.98
## 2 Luís      Gonçalves Brazil   3.96
## 3 Luís      Gonçalves Brazil   5.94
## 4 Luís      Gonçalves Brazil   0.99
## 5 Luís      Gonçalves Brazil   1.98
## 6 Luís      Gonçalves Brazil  13.9
## 7 Luís      Gonçalves Brazil   8.91
## 8 Leonie    Köhler      Germany 1.98
## 9 Leonie    Köhler      Germany 13.9
## 10 Leonie   Köhler      Germany 8.91
## # ... with 402 more rows
```

5. ¿Qué porcentaje de las canciones son video?

```
tracks <- tbl(chinook, "tracks")
head(tracks)
```

```
## # Source:   lazy query [?? x 9]
## # Database: sqlite 3.33.0 [C:\Users\Guillermo
## #   LT\Documents\GitHub\tgdbook\data\chinook.db]
##   TrackId Name AlbumId MediaTypeId GenreId Composer Milliseconds Bytes
##   <int> <chr> <int> <int> <int> <chr> <int> <int>
## 1 1 For ~ 1 1 1 Angus Y~ 343719 1.12e7
## 2 2 Ball~ 2 2 1 <NA> 342562 5.51e6
## 3 3 Fast~ 3 2 1 F. Balt~ 230619 3.99e6
## 4 4 Rest~ 3 2 1 F. Balt~ 252051 4.33e6
## 5 5 Prin~ 3 2 1 Deaffy ~ 375418 6.29e6
## 6 6 Put ~ 1 1 1 Angus Y~ 205662 6.71e6
## # ... with 1 more variable: UnitPrice <dbl>
```

```
tracks %>% group_by(MediaTypeId) %>%
  summarise(n = n()) %>% collect %>% mutate(freq = n / sum(n))
```

```
## # A tibble: 5 x 3
##   MediaTypeId n freq
```



```
##           <int> <int>   <dbl>
## 1             1  3034 0.866
## 2             2   237 0.0677
## 3             3   214 0.0611
## 4             4     7 0.00200
## 5             5    11 0.00314
```

```
media_types <- tbl(chinook, "media_types")
head(media_types)
```

```
## # Source:   lazy query [?? x 2]
## # Database: sqlite 3.33.0 [C:\Users\Guillermo
## #   LT\Documents\GitHub\tgdbook\data\chinook.db]
##   MediaTypeId Name
##           <int> <chr>
## 1             1 MPEG audio file
## 2             2 Protected AAC audio file
## 3             3 Protected MPEG-4 video file
## 4             4 Purchased AAC audio file
## 5             5 AAC audio file
```

```
tracks %>% inner_join(media_types, by = "MediaTypeId") %>% count(Name.y) %>%
  collect %>% mutate(freq = n / sum(n)) %>% filter(grepl('video', Name.y))
```

```
## # A tibble: 1 x 3
##   Name.y                n  freq
##   <chr>                <int> <dbl>
## 1 Protected MPEG-4 video file  214 0.0611
```

6. Listar los 10 mejores clientes (aquellos a los que se les ha facturado más cantidad) indicando Nombre, Apellidos, País y el importe total de su facturación.

```
customers %>% inner_join(invoices, by = "CustomerId") %>% group_by(CustomerId) %>%
  summarise(FirstName, LastName, country, total = sum(Total, na.rm = TRUE)) %>%
  arrange(desc(total)) %>% head(10) %>% collect
```

```
## # A tibble: 10 x 5
##   CustomerId FirstName LastName Country      total
##           <int> <chr>    <chr>    <chr>    <dbl>
## 1             6 Helena   Holý     Czech Republic 49.6
## 2            26 Richard Cunningham USA          47.6
## 3            57 Luis     Rojas    Chile          46.6
## 4            45 Ladislav Kovács    Hungary         45.6
## 5            46 Hugh     O'Reilly Ireland         45.6
## 6            28 Julia    Barnett   USA             43.6
## 7            24 Frank    Ralston   USA             43.6
## 8            37 Fynn     Zimmermann Germany          43.6
```

```
## 9          7 Astrid   Gruber   Austria   42.6
## 10         25 Victor  Stevens  USA       42.6
```

7. Listar los géneros musicales por orden decreciente de popularidad (definida la popularidad como el número de canciones de ese género), indicando el porcentaje de las canciones de ese género.

```
tracks %>% inner_join(tbl(chinook, "genres"), by = "GenreId") %>% count(Name.y) %>%
  arrange(desc(n)) %>% collect %>% mutate(freq = n / sum(n))
```

```
## # A tibble: 25 x 3
##   Name.y           n   freq
##   <chr>         <int> <dbl>
## 1 Rock           1297 0.370
## 2 Latin          579 0.165
## 3 Metal          374 0.107
## 4 Alternative & Punk 332 0.0948
## 5 Jazz           130 0.0371
## 6 TV Shows        93 0.0265
## 7 Blues           81 0.0231
## 8 Classical       74 0.0211
## 9 Drama           64 0.0183
## 10 R&B/Soul       61 0.0174
## # ... with 15 more rows
```

8. Listar los 10 artistas con mayor número de canciones de forma descendente según el número de canciones.

```
tracks %>% inner_join(tbl(chinook, "albums"), by = "AlbumId") %>%
  inner_join(tbl(chinook, "artists"), by = "ArtistId") %>%
  count(Name.y) %>% arrange(desc(n)) %>% collect
```

```
## # A tibble: 204 x 2
##   Name.y           n
##   <chr>         <int>
## 1 Iron Maiden     213
## 2 U2              135
## 3 Led Zeppelin    114
## 4 Metallica       112
## 5 Lost            92
## 6 Deep Purple     92
## 7 Pearl Jam       67
## 8 Lenny Kravitz   57
## 9 Various Artists 56
## 10 The Office      53
## # ... with 194 more rows
```

Desconectar la base de datos:

```
DBI::dbDisconnect(chinook)
```



## Capítulo 5

# Introducción a Tecnologías NoSQL

Son tecnologías de almacenamiento de datos en servicios web altamente escalables.

### 5.1 Conceptos y tipos de bases de datos NoSQL (documental, columnar, clave/valor y de grafos)

NoSQL - “Not Only SQL” - es una nueva categoría de bases de datos no-relacionales y altamente distribuidas.

Las bases de datos NoSQL nacen de la necesidad de:

- Simplicidad en los diseños
- Escalado horizontal
- Mayor control en la disponibilidad

Pero cuidado, en muchos escenarios las BBDD relacionales siguen siendo la mejor opción.

#### 5.1.1 Características de las bases de datos NoSQL

- Libre de esquemas – no se diseñan las tablas y relaciones por adelantado, además de permitir la migración del esquema.
- Proporcionan replicación a través de escalado horizontal.
- Este escalado horizontal se traduce en arquitectura distribuida
- Generalmente ofrecen consistencia débil

- Hacen uso de estructuras de datos sencillas, normalmente pares clave/valor a bajo nivel
- Suelen tener un sistema de consultas propio (o SQL-like)
- Siguen el modelo BASE (*Basic Availability, Soft state, Eventual consistency*) en lugar de ACID (*Atomicity, Consistency, Isolation, Durability*)

El modelo BASE consiste en:

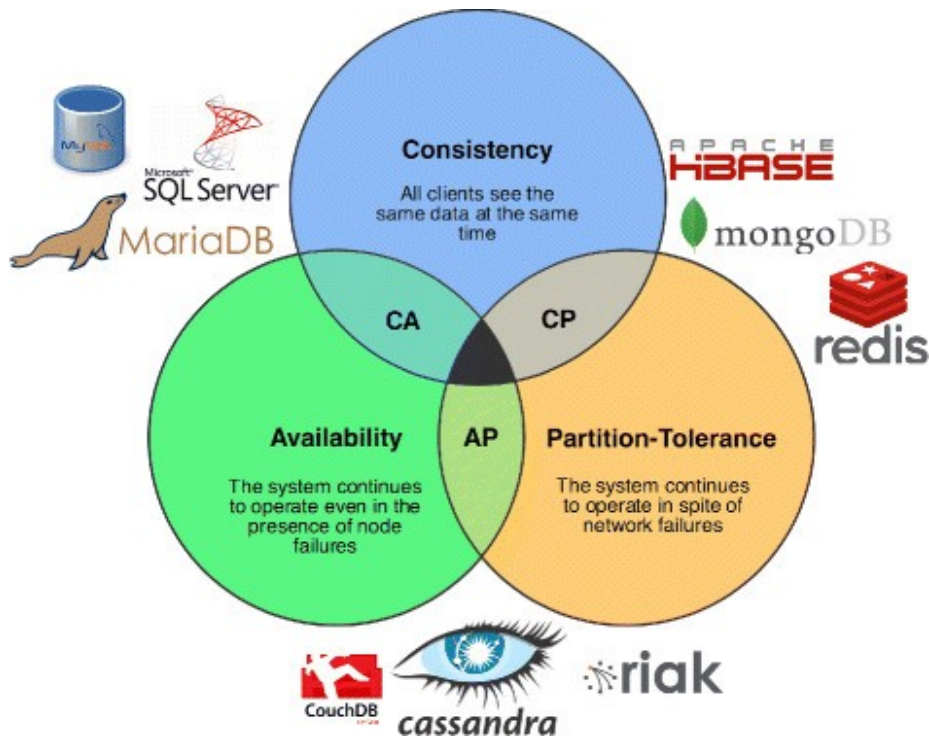
- *Basic Availability* – el sistema garantiza disponibilidad, en términos del teorema CAP.
- *Soft state* – el estado del sistema puede cambiar a lo largo del tiempo, incluso sin entrada. Esto es provocado por el modelo de consistencia eventual.
- *Eventual consistency* – el sistema alcanzará un estado consistente con el tiempo, siempre y cuando no reciba entrada durante ese tiempo.

#### 5.1.1.1 Teorema CAP

Es imposible para un sistema de cómputo distribuido garantizar simultáneamente:

- *Consistency* – Todos los nodos ven los mismos datos al mismo tiempo
- *Availability* – Toda petición obtiene una respuesta en caso tanto de éxito como fallo
- *Partition Tolerance* – El sistema seguirá funcionando ante pérdidas arbitrarias de información o fallos parciales

## 5.1. CONCEPTOS Y TIPOS DE BASES DE DATOS NOSQL (DOCUMENTAL, COLUMNAR, CLAVE/VALOR Y



Las razones para escoger NoSQL son:

- Analítica
- Gran cantidad de escrituras, análisis en bloque
- Escalabilidad
- Tan fácil como añadir un nuevo nodo a la red, bajo coste.
- Redundancia
- Están diseñadas teniendo en cuenta la redundancia
- Rápido desarrollo
- Al ser schema-less o schema on-read son más flexibles que schema on-write
- Flexibilidad en el almacenamiento de datos
- Almacenan todo tipo de datos: texto, imágenes, BLOBs
- Gran rendimiento en consultas sobre datos que no implican relaciones jerárquicas
- Gran rendimiento sobre BBDD desnormalizadas
- Tamaño
- El tamaño del esquema de datos es demasiado grande
- Muchos datos temporales fuera de almacén principal

Razones para NO escoger NoSQL: \* Consistencia y Disponibilidad de los datos son críticas \* Relaciones entre datos son importantes + E.g. joins numerosos y/o importantes \* En general, cuando el modelo ACID encaja mejor

## 5.1.2 Tipos de Bases de Datos NoSQL

Columnar:

1	Things	A	foo	B	bar	C	baz
2	Things	C	bam	E	coh	People	A Emmanuel
3	Languages	A	C	B	Java	C	Ceylon

Documental:

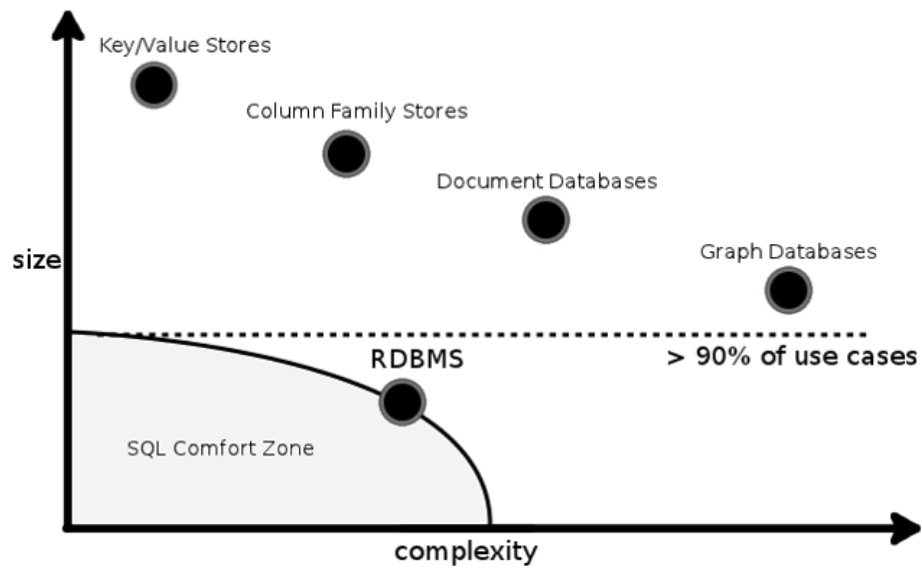
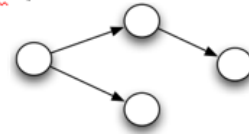
```

{"user": {
  "id": "124",
  "name": "Emmanuel",
  "addresses": [
    { "city": "Paris", "country": "France" },
    { "city": "Atlanta", "country": "USA" }
  ]
}

```

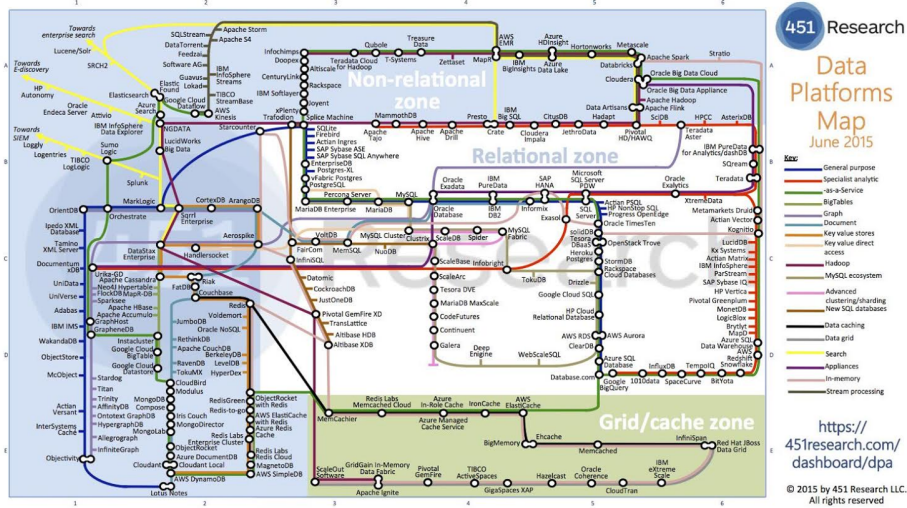
Key-value:

key	value
123	Address@23
126	"Booya"

Graph:

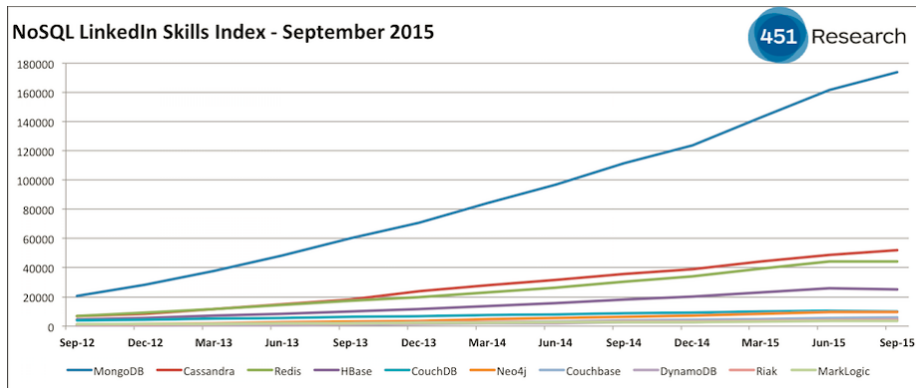


## 5.1. CONCEPTOS Y TIPOS DE BASES DE DATOS NOSQL (DOCUMENTAL, COLUMNAR, CLAVE/VALOR Y Y



352 systems in ranking, September 2019

Sep 2019	Rank		DBMS	Database Model	Score		
	2019	Sep 2018			Sep 2019	Aug 2019	Sep 2018
1.	1.	1.	Oracle +	Relational, Multi-model	1346.66	+7.18	+37.54
2.	2.	2.	MySQL +	Relational, Multi-model	1279.07	+25.39	+98.60
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model	1085.06	-8.12	+33.78
4.	4.	4.	PostgreSQL +	Relational, Multi-model	482.25	+0.91	+75.82
5.	5.	5.	MongoDB +	Document	410.06	+5.50	+51.27
6.	6.	6.	IBM Db2 +	Relational, Multi-model	171.56	-1.39	-9.50
7.	7.	7.	Elasticsearch +	Search engine, Multi-model	149.27	+0.19	+6.67
8.	8.	8.	Redis +	Key-value, Multi-model	141.90	-2.18	+0.96
9.	9.	9.	Microsoft Access	Relational	132.71	-2.63	-0.69
10.	10.	10.	Cassandra +	Wide column	123.40	-1.81	+3.85



### 5.1.3 MongoDB: NoSQL documental

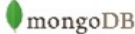
Rich Queries	<ul style="list-style-type: none"> <li>Find Paul's cars</li> <li>Find everybody in London with a car built between 1970 and 1980</li> </ul>
Geospatial	<ul style="list-style-type: none"> <li>Find all of the car owners within 5km of Trafalgar Sq.</li> </ul>
Text Search	<ul style="list-style-type: none"> <li>Find all the cars described as having leather seats</li> </ul>
Aggregation	<ul style="list-style-type: none"> <li>Calculate the average value of Paul's car collection</li> </ul>
Map Reduce	<ul style="list-style-type: none"> <li>What is the ownership pattern of colors by geography over time? (is purple trending up in China?)</li> </ul>

## MongoDB

```

{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location:
[45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, ... },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, ... }
  ]
}

```



6

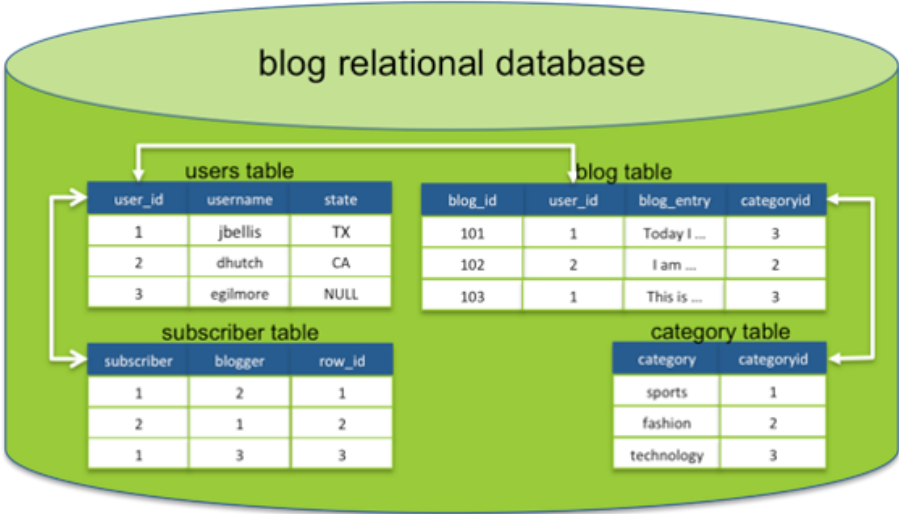
### 5.1.4 Redis: NoSQL key-value

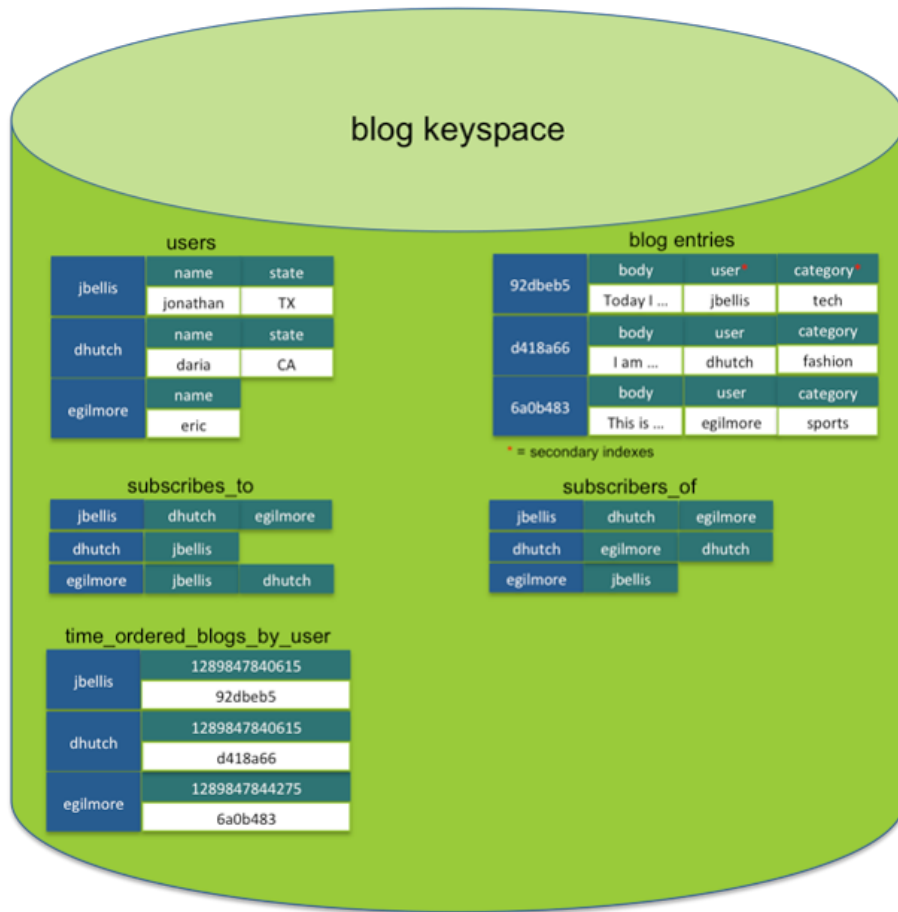
In-memory data structure store, útil para base de datos de login-password, sensor-valor, URL-respuesta, con una sintaxis muy sencilla:

- El comando SET almacena valores
- SET server:name "luna"
- Recuperamos esos valores con GET
- GET server:name
- INCR incrementa atómicamente un valor
- INCR clients
- DEL elimina claves y sus valores asociados
- DEL clients
- TTL (Time To Live) útil para cachés
- EXPIRE promocion 60

5.1. CONCEPTOS Y TIPOS DE BASES DE DATOS NOSQL (DOCUMENTAL, COLUMNAR, CLAVE/VALOR Y

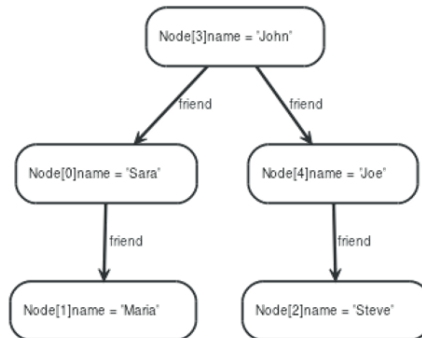
5.1.5 Cassandra: NoSQL columnar





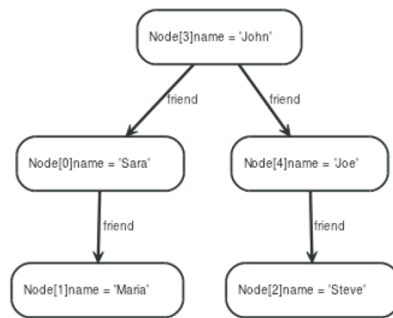
### 5.1.6 Neo4j: NoSQL grafos





```

MATCH (john {name: 'John'})-[:friend]->()-[:friend]->(fof)
RETURN john, fof
  
```



john	fof
Node[3]{name:"John"}	Node[1]{name:"Maria"}
Node[3]{name:"John"}	Node[2]{name:"Steve"}
2 rows	

### 5.1.7 Otros: search engines

Son sistemas especializados en búsquedas, procesamiento de lenguaje natural como Elasticsearch, Solr, Splunk (logs de aplicaciones), etc...

## 5.2 Conexión de R a MongoDB

A través del paquete mongolite, aquí tenéis un Tutorial

```
install.packages("mongolite")
```

```
library(mongolite)
```

```
# Connect to a local MongoDB
```

```
my_collection = mongo(collection = "restaurants", db = "Restaurants") # create connect  
my_collection$count
```

### 5.3 Ejercicios prácticos con MongoDB

Estos ejercicios se pueden hacer en un notebook Kaggle accediendo a un clúster de MongoDB en el cloud de MongoDB. Se carga la base de datos de ejemplo y se puede hacer con la colección de restaurantes (o alternativamente con otras colecciones).

1. Mostrar todos los documentos de la colección restaurants (u otra)
2. Mostrar nombre de restaurante, barrio y cocina de la colección restaurants (o los campos de otra colección)
3. Mostrar los primeros 5 restaurantes del barrio Bronx.
4. Mostrar los restaurantes con una longitud menor que -75.7541
5. Mostrar los restaurantes con una puntuación superior a 90
6. Mostrar los restaurantes de comida American o Chinese del barrio Queens.
7. Mostrar los restaurantes con un grado "A" y puntuación 9 obtenida en fecha 2014-08-11T00:00:00Z
8. Propón un JSON para descargar (de algún repositorio OpenData o disponible en un API), indícame la URL, si has de hacer algún proceso antes de importarlo en MongoDB, cómo lo importas, dame un pantallazo del análisis exploratorio de ese JSON y una query que harías contra ese JSON (la query en MongoDB, Compass o RmongoDB)

## Capítulo 6

# Tecnologías para el Tratamiento de Datos Masivos

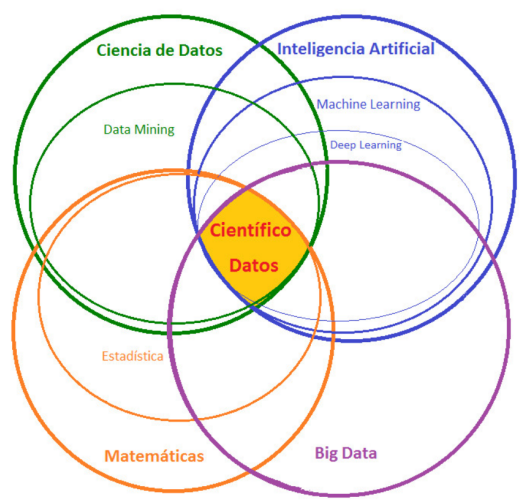
En este apartado trataremos los siguientes epígrafes:

1. Introducción al Aprendizaje Estadístico
2. Tecnologías Big Data (Hadoop, Spark, Sparklyr)
3. Ejercicios de análisis de datos masivos.

### 6.1 Introducción al Aprendizaje Estadístico

El material para este apartado está disponible en el Capítulo 1 del libro “Aprendizaje Estadístico” de Rubén Fernández Casal.

Para seguir este capítulo es altamente recomendable tener instalado Rattle, para ello consultad el apéndice de instalación de R al final de este libro.



## 6.2 Tecnologías Big Data (Hadoop/Spark y Visualización)

### 6.2.1 Tecnologías Hadoop, Spark, y Sparklyr

A continuación se introducen los conceptos básicos de las tecnologías Hadoop, Spark y Sparklyr:

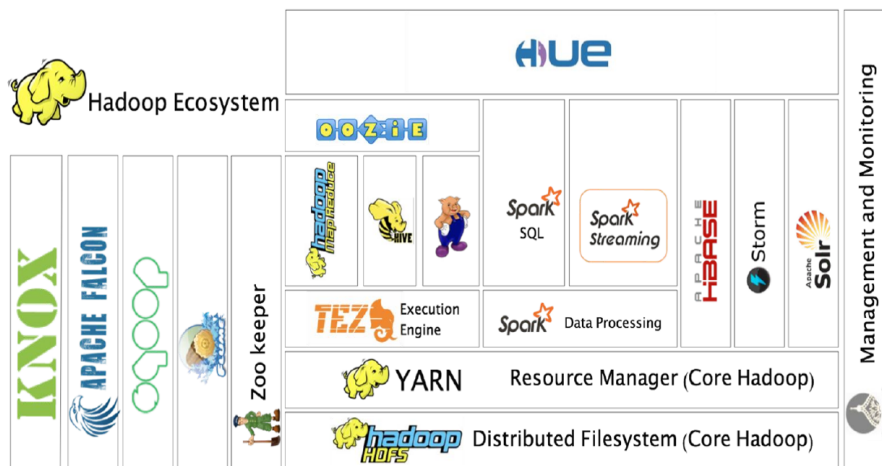
- Hadoop: framework open-source desarrollado en Java principalmente que soporta aplicaciones distribuidas sobre miles de nodos y a escala Petabyte. Está inspirado en el diseño de las operaciones de MapReduce de Google y el Google File System (GFS). Entre sus principales componentes destaca HDFS Hadoop Distributed File System, sistema de ficheros distribuido sobre múltiples nodos y accesible a nivel de aplicación. También destaca YARN como gestor de recursos, para ejecutar aplicaciones. Destacar que la versión original, Hadoop 1, estaba basada extensivamente en Map Reduce, Hadoop 2 colocó en su core a YARN y Hadoop 3 está orientado a la provisión de Plataforma como servicio y ejecución simultánea de múltiples cargas de trabajo distribuidas sobre recursos solicitados bajo demanda.
- Hive: es un sistema de almacenamiento y explotación de datos del estilo de un data warehouse open source diseñado para ser ejecutado en entornos Hadoop. Permite agrupar, consultar y analizar datos almacenados en Hadoop File System y en Amazon S3 (almacenamiento de objetos en general) en esquema en estrella. Su lenguaje de consulta de datos, Hive Query Language o (HQL).
- Spark: framework de computación distribuida open-source para el procesamiento de datos masivos sobre Hadoop con un paralelismo implícito sobre

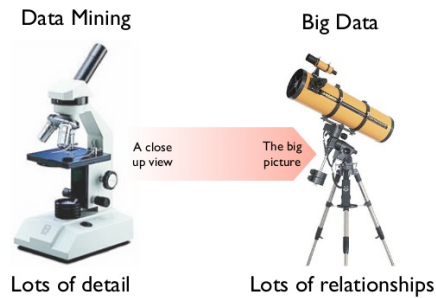


## 6.2. TECNOLOGÍAS BIG DATA (HADOOP/SPARK Y VISUALIZACIÓN)73

su estructura de datos (Resilient Distributed Dataset o RDD), permite operar en paralelo sobre una colección de datos sin saber en qué servidores están disponibles dichos datos y de una forma tolerante a fallos. Es uno de los principales frameworks de programación de entornos Hadoop al estar optimizado su procesamiento sobre memoria (en lugar de sobre archivos en disco) para obtener velocidad, tanto en sus vertientes Spark streaming y Spark SQL, como Spark Machine Learning MLlib. Dispone de interfaces en Java, Scala, Python y R, siendo las interfaces de R Rspark y Sparklyr.

- SparkR: es un paquete, el primero que apareció, para conectar R con Spark. Intenta ser lo más parecida a la interfaz estándar de R de manipulación de datos.
- Sparklyr: es una librería para conectar R con Spark posterior a SparkR. Intenta ser lo más parecida a dplyr y embeber SQL en las consultas, soportando una mayor cantidad de paquetes. Por este motivo es el proyecto más activo actualmente, sustituyendo a SparkR.

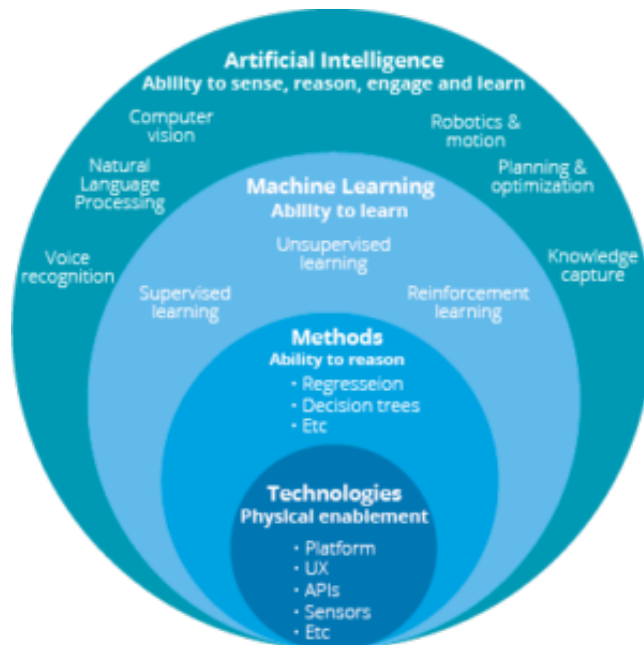




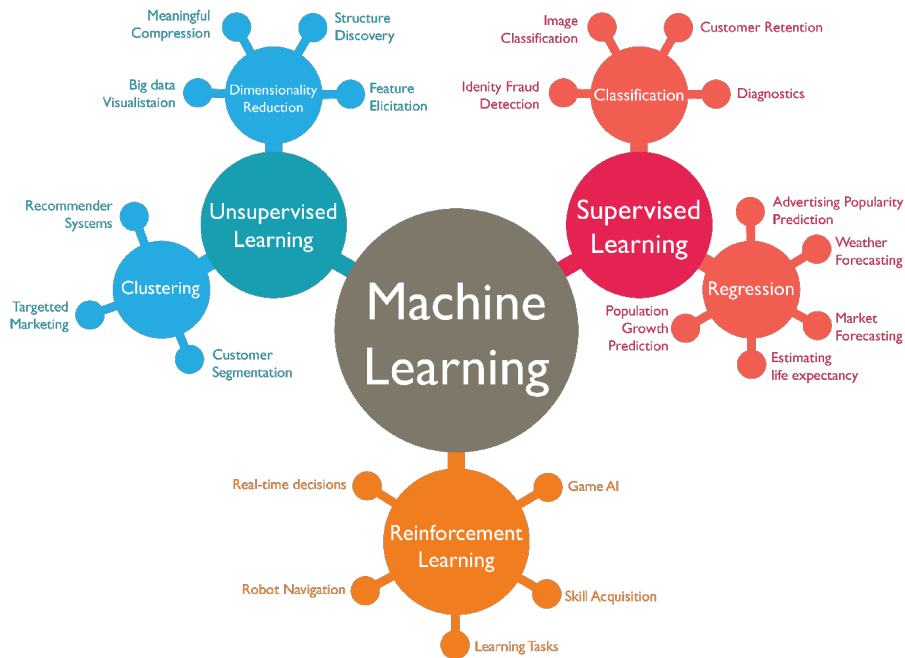
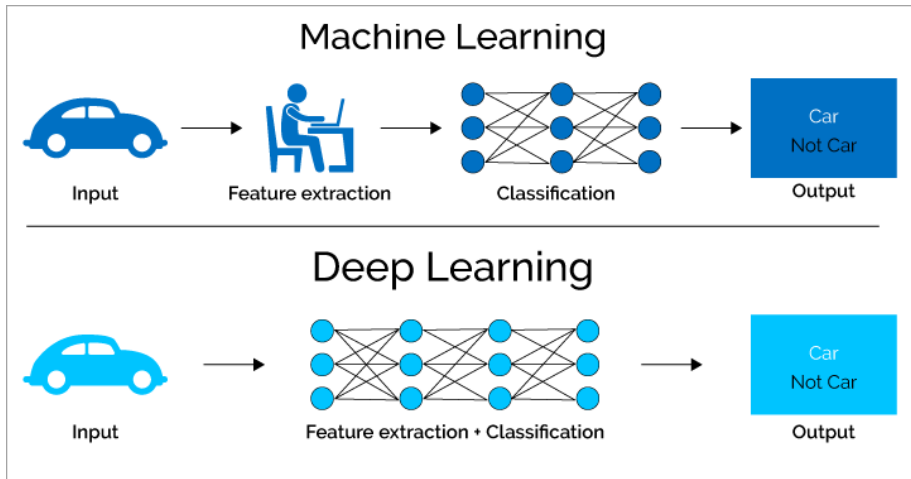
Thursday, 31 January 13

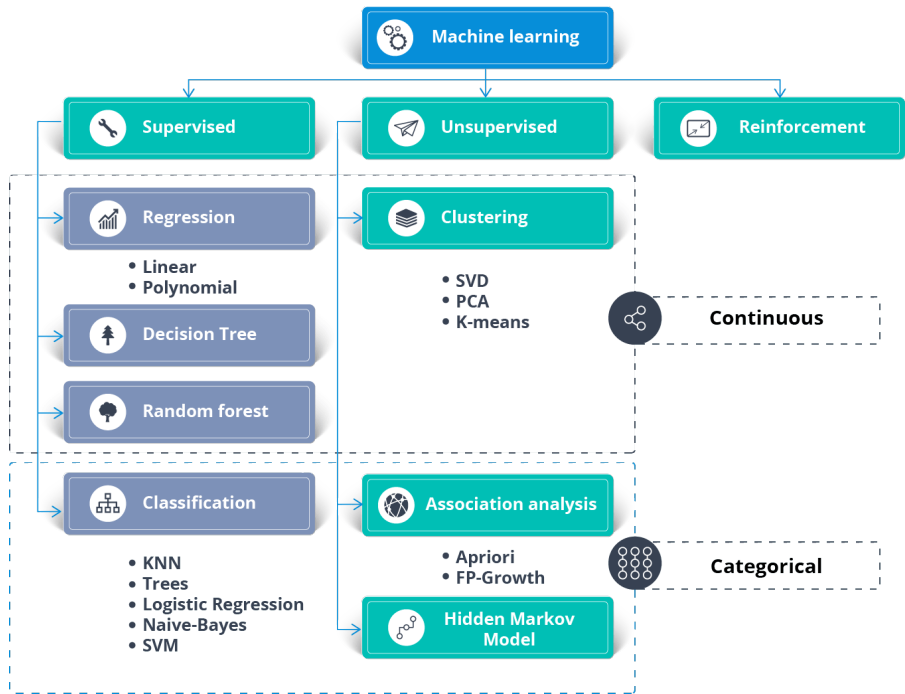
## 6.2.2 Big Data y Machine Learning

El Machine Learning o Aprendizaje Máquina es aquella parte de la inteligencia artificial con capacidad de aprender de los datos.

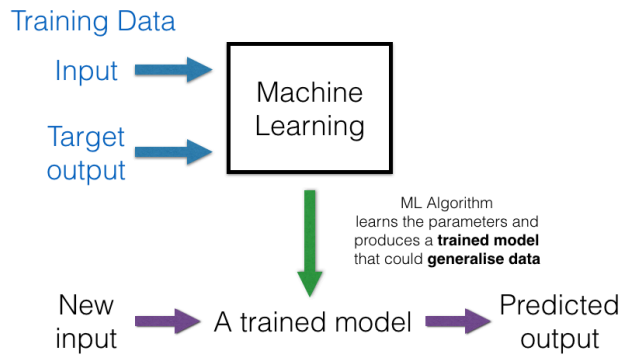


6.2. TECNOLOGÍAS BIG DATA (HADOOP/SPARK Y VISUALIZACIÓN)75





Y un ejemplo de cómo se trabaja en machine learning:



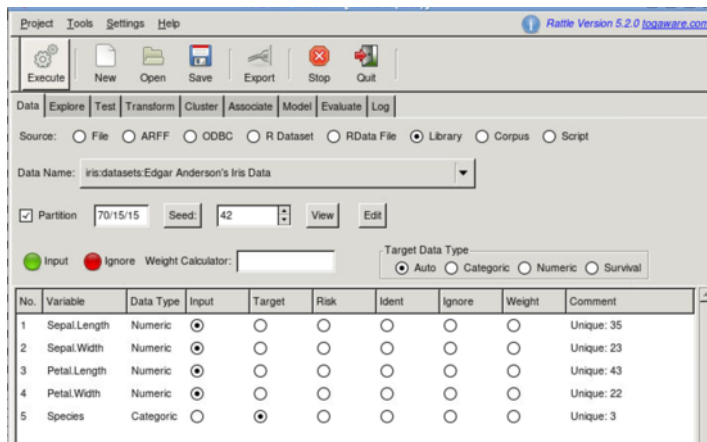
## 6.2. TECNOLOGÍAS BIG DATA (HADOOP/SPARK Y VISUALIZACIÓN)77

		Actual class (ground truth)		
		Dog (Positive)	Not a Dog (Negative)	
<b>Total (n)=100</b>				
Predicted class	Dog (Positive)	15 (TP)	20 (FP, Type I Error)	Precision =TP/(TP+FP) =0.42
	Not a Dog (Negative)	5 (FN, Type II Error)	60 (TN)	
Accuracy =(TP+TN)/Total =0.75		Sensitivity, Recall TPR = TP/(TP+FN) = 0.75	FPR = FP/(FP+TN) =0.25	F1 Score =2*(Precision*Recall) / (Precision+Recall) =0.53
Error Rate =(FP+FN)/Total =0.25		Miss Rate, FNR =FN/(TP+FN) =0.25	Specificity, TNR = TN/(FP+TN) =0.75	

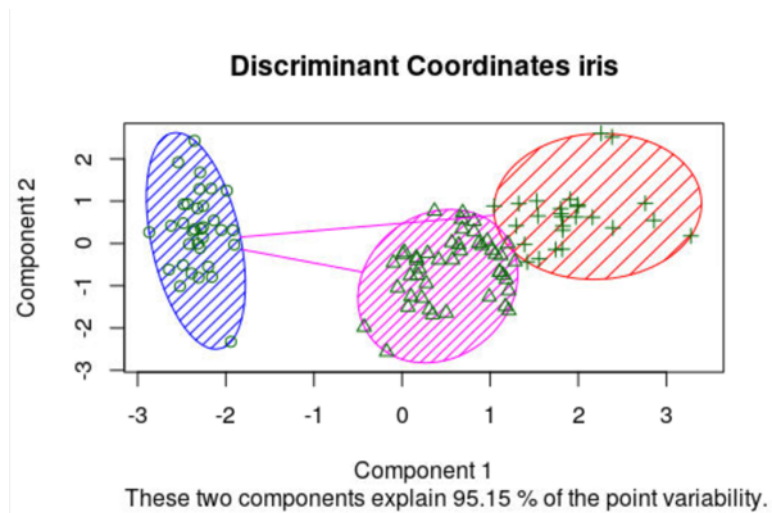
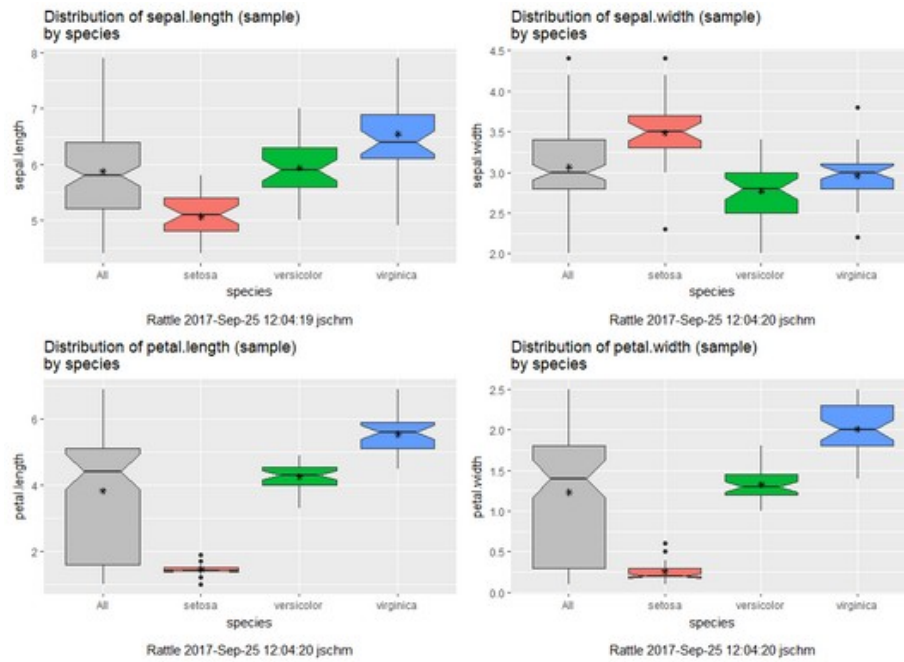
### 6.2.3 Rattle como alternativa a RapidMiner en R

Las instrucciones para instalar R está en el Apéndice 3 de este documento

Un tutorial adecuado para introducirse en Rattle es éste



Con el tutorial se pueden ver las capacidades de rattle de explorar los datos, como se puede apreciar a continuación.



### 6.2.4 Visualización y Generación de Cuadros de Mando

Se sigue un tutorial de la herramienta PowerBI, con datos de Excel y OData Feed

Como documentación de se soporte se cuenta con la web de PowerBI y un tutorial adicional

## 6.3 Introducción al Análisis de Datos Masivos

En primer lugar se ha de considerar explorar los datos y realizar minería con ellos, y eso es posible hacerlo vía sparklyr.

Este apartado, eminentemente práctico, lo trabajaremos a través de la práctica 3 de TGD.

Working draft...





# Apéndice A

## Enlaces

**Recursos para el aprendizaje de R** ( <https://rubenfcasal.github.io/post/ayuda-y-recursos-para-el-aprendizaje-de-r> ): A continuación se muestran algunos recursos que pueden ser útiles para el aprendizaje de R y la obtención de ayuda...

***Ayuda online:***

- Ayuda en línea sobre funciones o paquetes: RDocumentation
- Buscador RSeek
- StackOverflow

***Cursos:*** algunos cursos gratuitos:

- Coursera:
  - Introducción a Data Science: Programación Estadística con R
  - Mastering Software Development in R
- DataCamp:
  - Introducción a R
- Stanford online:
  - Statistical Learning
- Curso UCA: Introducción a R, R-commander y shiny
- Curso R CODER
- Udacity: Data Analysis with R
- Swirl Courses: se pueden hacer cursos desde el propio R con el paquete swirl.

Para información sobre cursos en castellano se puede recurrir a la web de R-Hispano en el apartado formación. Algunos de los cursos que aparecen en entradas antiguas son gratuitos. Ver: Cursos MOOC relacionados con R.

### **Libros**

- **Iniciación:**

- R for Data Science (online, O’Reilly)
- 2011 - The Art of R Programming. A Tour of Statistical Software Design, (No Starch Press)
- Hands-On Programming with R: Write Your Own Functions and Simulations, by Garrett Golemund (O’Reilly)

- **Avanzados:**

- Advanced R by Hadley Wickham (online: 1<sup>a</sup> ed, 2<sup>a</sup> ed, Chapman & Hall)
- 2008 - Software for Data Analysis: Programming with R - Chambers (Springer)
- R packages by Hadley Wickham (online, O’Reilly)

- **Bookdown:** el paquete `bookdown` de R permite escribir libros empleando R Markdown y compartirlos. En <https://bookdown.org> está disponible una selección de libros escritos con este paquete (un listado más completo está disponible aquí). Algunos libros en este formato en castellano son:

- Introducción al Análisis de Datos con R (disponible en el repositorio de GitHub [rubenfcasal/intror](https://github.com/rubenfcasal/intror)).
- Prácticas de Simulación (disponible en el repositorio de GitHub [rubenfcasal/simbook](https://github.com/rubenfcasal/simbook)).
- Escritura de libros con bookdown (disponible en el repositorio de GitHub [rubenfcasal/bookdown\\_intro](https://github.com/rubenfcasal/bookdown_intro)).
- R para profesionales de los datos: una introducción.
- Estadística Básica Edulcorada.

**Material online:** en la web se puede encontrar mucho material adicional, por ejemplo:

- CRAN: Other R documentation
- Blogs en inglés:
  - <https://www.r-bloggers.com/>
  - <https://www.littlemissdata.com/blog/rstudioconf2019>
  - RStudio: <https://blog.rstudio.com>

- Microsoft Revolutions: <https://blog.revolutionanalytics.com>
- Blogs en castellano:
  - <https://www.datanalytics.com>
  - <http://oscarperpinan.github.io/R>
  - <http://rubenfcasal.github.io>
- Listas de correo:
  - Listas de distribución de r-project.org: <https://stat.ethz.ch/mailman/listinfo>
  - Búsqueda en R-help: <http://r.789695.n4.nabble.com/R-help-f789696.html>
  - Búsqueda en R-help-es: <https://r-help-es.r-project.narkive.com>  
<https://grokbase.com/g/r/r-help-es>
  - Archivos de R-help-es: <https://stat.ethz.ch/pipermail/r-help-es>

## A.1 RStudio

### *RStudio:*

- Online learning
- Webinars
- sparklyr
- shiny

### *tidyverse:*

- dplyr
- tibble
- tidyr
- stringr
- readr
- Databases using R, dplyr as a database interface, Databases using dplyr

### *CheatSheets:*

- rmarkdown
- shiny
- dplyr

- tidy
- stringr

## A.2 Bibliometría

- CITAN
- scimetr
- bibliometrix
- wosr
- rwos
- rcrossref
- ropensci: Literature
- Diderot
- ...

## Apéndice B

# El paquete `scimetr`

Package `scimetr` implements tools for quantitative research in scientometrics and bibliometrics. It provides routines for importing bibliographic data from Thomson Reuters' Web of Science (<http://www.webofknowledge.com>) and performing bibliometric analysis. For more information visit <https://rubenfcasal.github.io/scimetr/articles/scimetr.html>.

### B.1 Instalación

Para instalar el paquete sería recomendable en primer lugar instalar las dependencias:

```
install.packages(c('dplyr', 'lazyeval', 'stringr', 'ggplot2', 'openxlsx', 'tidyr'))
```

Como de momento no está disponible en CRAN, habría que instalar la versión de desarrollo en GitHub. En Windows bastaría con instalar la versión binaria del paquete `scimetr_X.Y.Z.zip` (disponible [aquí](#)), alternativamente se puede instalar directamente de GitHub:

```
# install.packages("devtools")
devtools::install_github("rubenfcasal/scimetr")
```

Una vez instalado ya podríamos cargar el paquete:

```
library(scimetr)
```



	variable.labels
PT	Publication type
AU	Author
BA	Book authors
BE	Editor
GP	Group author
AF	Author full
BF	Book authors fullname
CA	Corporate author
TI	Title
SO	Publication name
SE	Series title
BS	Book series
LA	Language
DT	Document type
CT	Conference title
CY	Conference year
CL	Conference place
SP	Conference sponsors
HO	Conference host
DE	Keywords
ID	Keywords Plus
AB	Abstract
C1	Addresses
RP	Reprint author
EM	Author email
RI	Researcher id numbers
OI	Orcid numbers
FU	Funding agency and grant number
FX	Funding text
CR	Cited references
NR	Number of cited references
TC	Times cited
Z9	Total times cited count
U1	Usage Count (Last 180 Days)
U2	Usage Count (Since 2013)
PU	Publisher
PI	Publisher city
PA	Publisher address
SN	ISSN
EI	eISSN
BN	ISBN
J9	Journal.ISI
JI	Journal.ISO
PD	Publication date
PY	Year published
VL	Volume
IS	Issue
PN	Part number
SU	Supplement
SI	Special issue
MA	Meeting abstract
BP	Beginning page
EP	Ending page

Se puede crear una base de datos con la función `CreateDB.wos()`:

```
db <- CreateDB.wos(wosdf2, label = "Mathematics_UDC_2008-2017 (01-02-2019)")
str(db, 1)
```

```
## List of 11
## $ Docs      : 'data.frame':  389 obs. of  26 variables:
## $ Authors   : 'data.frame':  611 obs. of   4 variables:
## $ AutDoc    : 'data.frame': 1260 obs. of   2 variables:
## $ Categories: 'data.frame':   46 obs. of   2 variables:
## $ CatDoc    : 'data.frame':  866 obs. of   2 variables:
## $ Areas     : 'data.frame':   26 obs. of   2 variables:
## $ AreaDoc   : 'data.frame':  771 obs. of   2 variables:
## $ Addresses : 'data.frame':  896 obs. of   5 variables:
## $ AddAutDoc : 'data.frame': 1328 obs. of   3 variables:
## $ Journals  : 'data.frame':  150 obs. of  12 variables:
## $ label     : chr "Mathematics_UDC_2008-2017 (01-02-2019)"
## - attr(*, "variable.labels")= Named chr [1:62] "Publication type" "Author" "Book a
## .. attr(*, "names")= chr [1:62] "PT" "AU" "BA" "BE" ...
## - attr(*, "class")= chr "wos.db"
```

## B.2.2 Cargar datos de directorio

Se pueden cargar automáticamente los archivos wos (tienen una limitación de 500 registros) de un subdirectorio:

```
dir("UDC_2008-2017 (01-02-2019)", pattern='*.txt')
```

```
## [1] "savedrecs01.txt" "savedrecs02.txt" "savedrecs03.txt" "savedrecs04.txt"
## [5] "savedrecs05.txt" "savedrecs06.txt" "savedrecs07.txt" "savedrecs08.txt"
## [9] "savedrecs09.txt" "savedrecs10.txt" "savedrecs11.txt" "savedrecs12.txt"
## [13] "savedrecs13.txt" "savedrecs14.txt" "savedrecs15.txt"
```

Se pueden combinar los ficheros y crear la correspondiente base de datos con los siguientes comandos:

```
wos.txt <- ImportSources.wos("UDC_2008-2017 (01-02-2019)", other = FALSE)
db.txt <- CreateDB.wos(wos.txt)
```

## B.3 Sumarios

### B.3.1 Sumario `summary.wos.db()`

```
res1 <- summary(db)
options(digits = 5)
res1
```

```
## Number of documents: 389
```



```

## Authors: 611
## Period: 2008 - 2017
##
## Document types:
##           Documents
## Article                360
## Correction              1
## Editorial Material      5
## Proceedings Paper      16
## Review                  7
##
## Number of authors per document:
##   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
##   1.00   2.00   3.00   3.24   4.00   8.00
##
## Number of documents per author:
##   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
##   1.00   1.00   1.00   2.06   2.00  29.00
##
## Number of times cited:
##   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
##   0.0    1.0    3.0   10.9    9.0  1139.0
##
## Indexes:
## H G
## 24 54
##
## Top Categories:
##                                     Documents
## Mathematics, Interdisciplinary Applications  134
## Mathematics, Applied                      130
## Statistics & Probability                   121
## Mathematics                               77
## Engineering, Multidisciplinary            64
## Mechanics                                 59
## Computer Science, Interdisciplinary Applications  45
## Computer Science, Artificial Intelligence    20
## Social Sciences, Mathematical Methods      17
## Automation & Control Systems              16
## Others                                    183
##
## Top Areas:
##                                     Documents
## Mathematics                               389
## Computer Science                          69
## Engineering                               69

```

## Mechanics	59
## Physics	22
## Chemistry	17
## Mathematical Methods In Social Sciences	17
## Automation & Control Systems	16
## Instruments & Instrumentation	16
## Business & Economics	15
## Others	82
##	
## Top Journals:	
##	Documents
## Comput. Meth. Appl. Mech. Eng.	29
## J. Math. Anal. Appl.	11
## Chemometrics Intell. Lab. Syst.	11
## Rev. Int. Metod. Numer. Calc. Dise.	11
## J. Comput. Appl. Math.	10
## Comput. Stat. Data Anal.	9
## Appl. Numer. Math.	9
## Int. J. Numer. Methods Fluids	9
## Int. J. Numer. Methods Eng.	8
## J. Nonparametr. Stat.	8
## Others	274
##	
## Top Countries:	
##	Documents
## Spain	389
## USA	49
## France	32
## Italy	13
## Mexico	11
## UK	11
## Germany	10
## Canada	8
## China	8
## Belgium	7
## Others	52

### B.3.2 Sumario por años `summary_year()`

```
res2 <- summary_year(db)
res2
```

```
##
## Annual Scientific Production:
##
## Documents
```

```

## 2008      42
## 2009      28
## 2010      40
## 2011      37
## 2012      44
## 2013      40
## 2014      38
## 2015      39
## 2016      47
## 2017      34
##
## Annual Authors per Document:
##
##           Avg Median
## 2008 2.8810   3.0
## 2009 3.3214   3.0
## 2010 3.3500   3.0
## 2011 3.3784   3.0
## 2012 2.8182   2.5
## 2013 3.3750   3.0
## 2014 3.2368   3.0
## 2015 3.0513   3.0
## 2016 3.5745   3.0
## 2017 3.4706   3.0
##
## Annual Times Cited:
##
##           Cites   Avg Median
## 2008    755 17.9762   5.0
## 2009    265  9.4643   6.0
## 2010    410 10.2500   5.0
## 2011   1422 38.4324   5.0
## 2012    335  7.6136   3.5
## 2013    271  6.7750   4.0
## 2014    336  8.8421   3.5
## 2015    215  5.5128   2.0
## 2016    192  4.0851   2.0
## 2017     55  1.6176   1.0

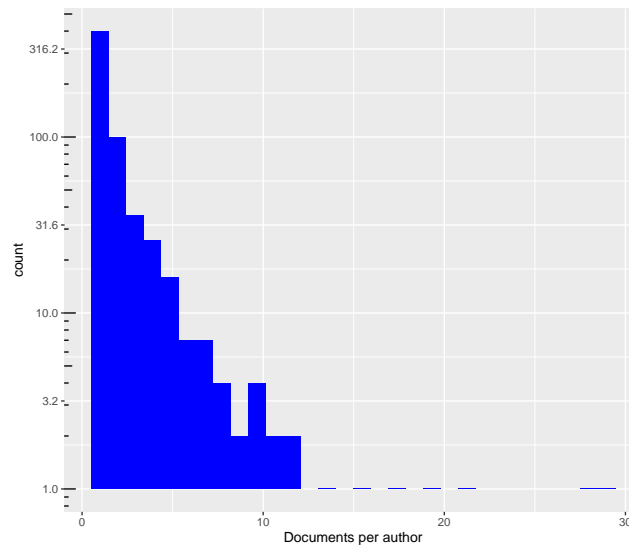
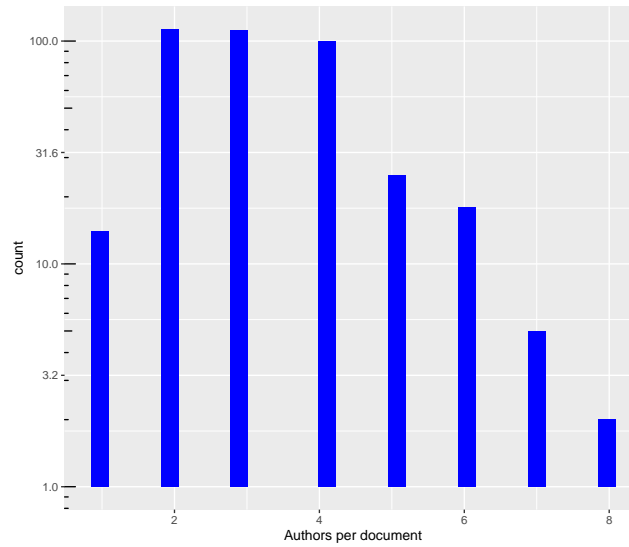
```

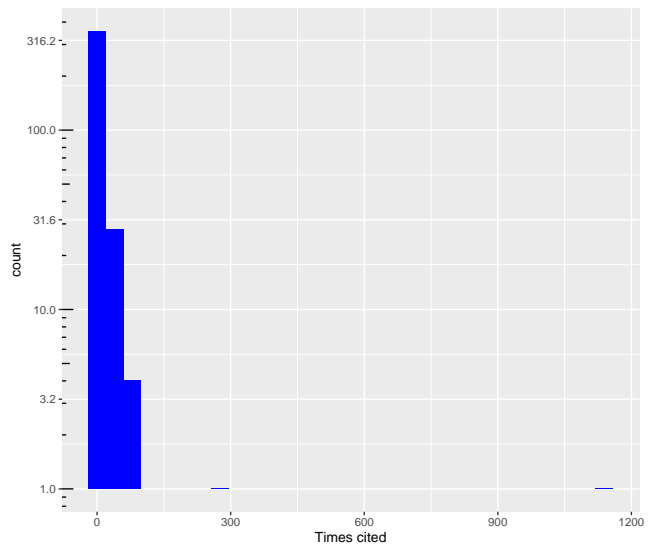
## B.4 Gráficos

Se emplea la librería `ggplot2`...

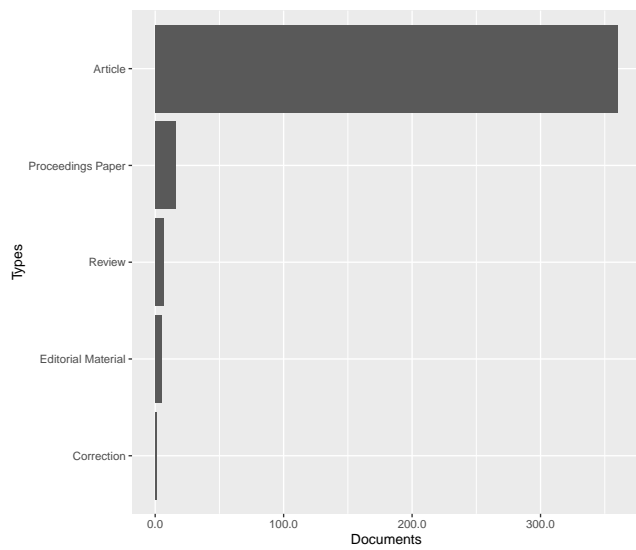
```
plot(db)
```

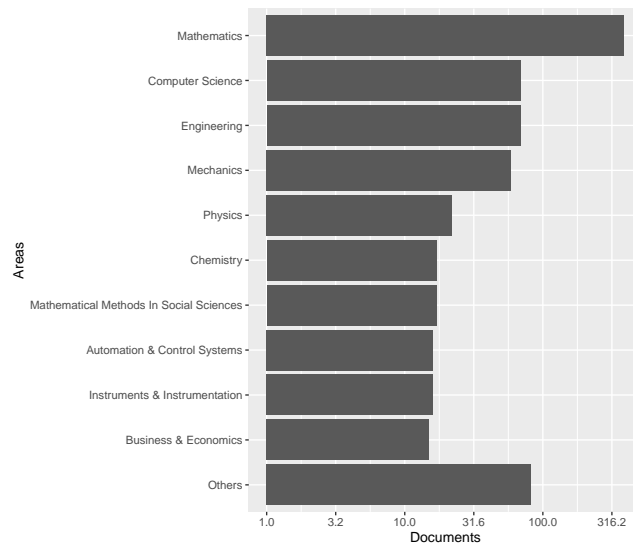
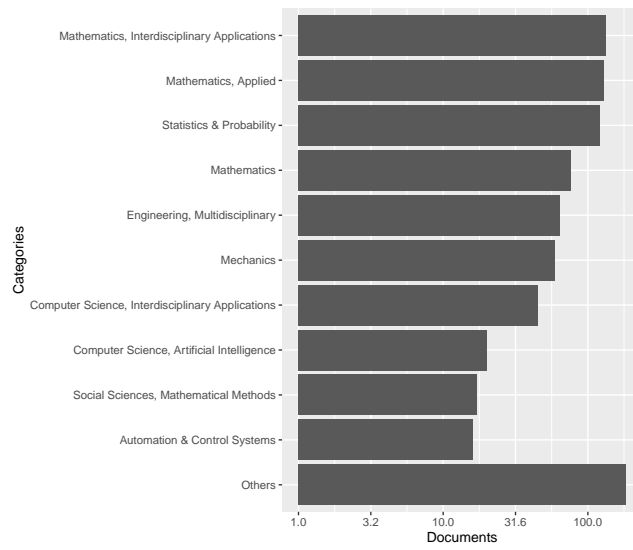
#### B.4.1 Gráficos de la base de datos `plot.wos.db()`

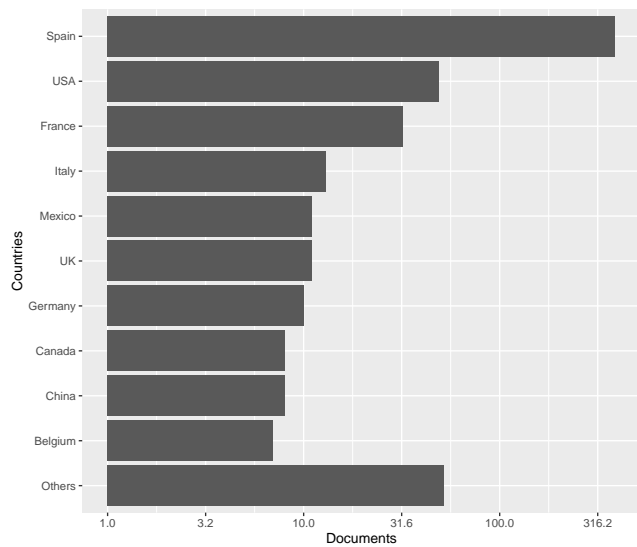
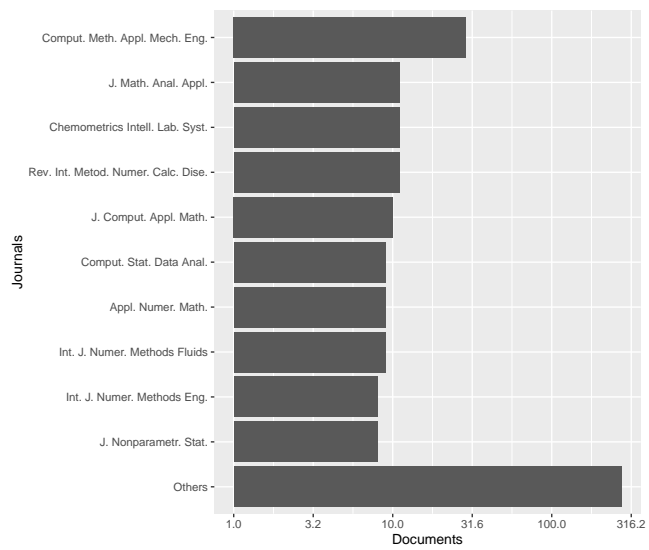




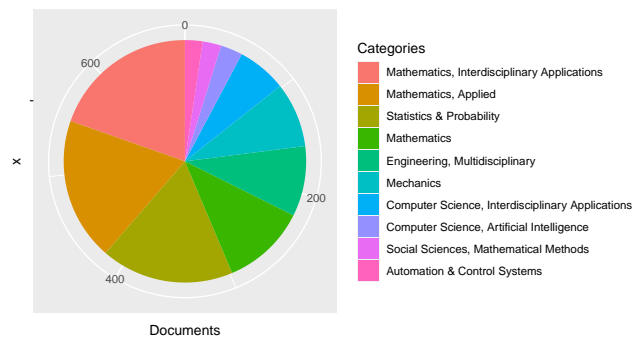
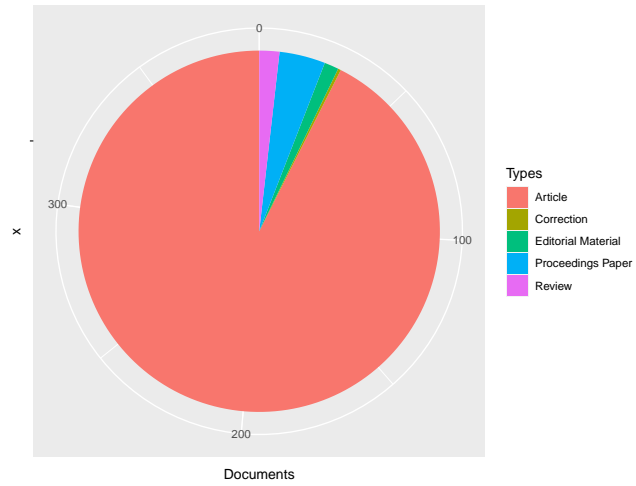
```
plot(res1)
```



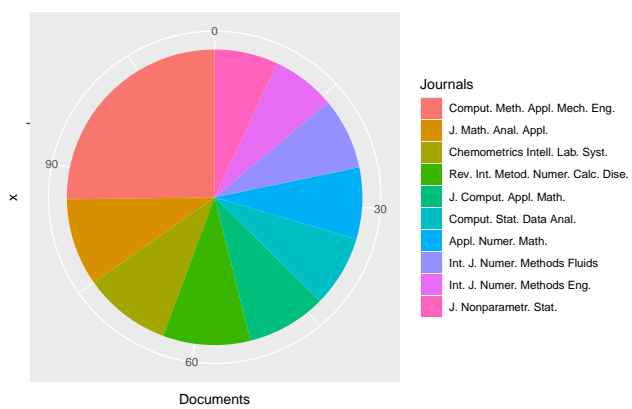
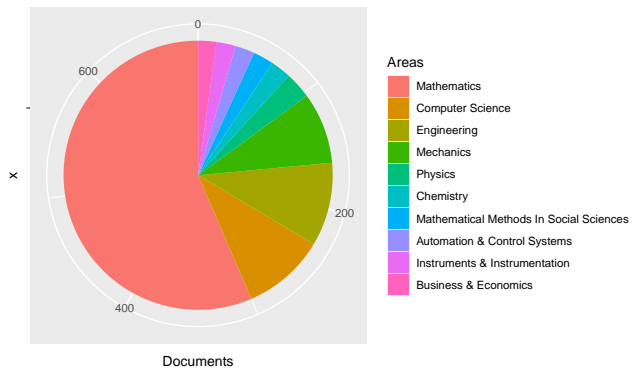


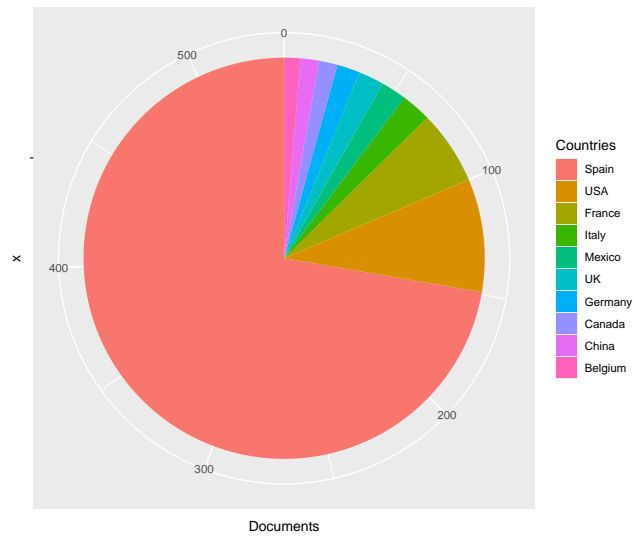


```
plot(res1, pie = TRUE)
```

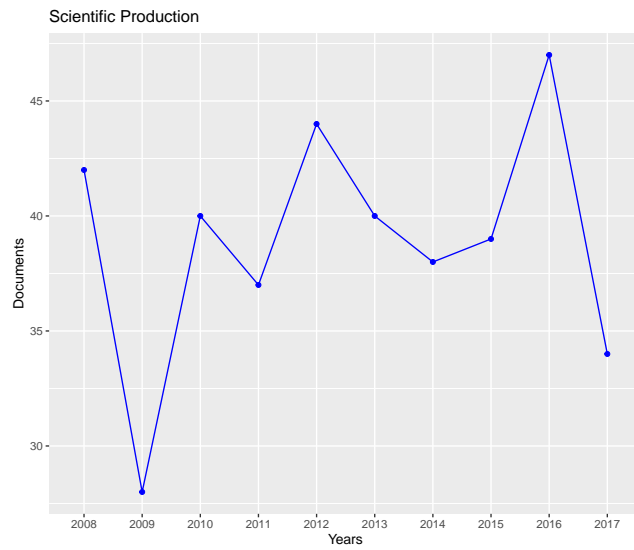


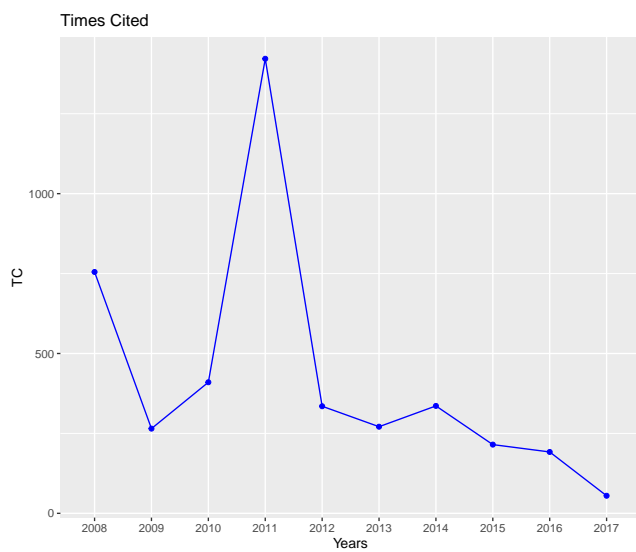
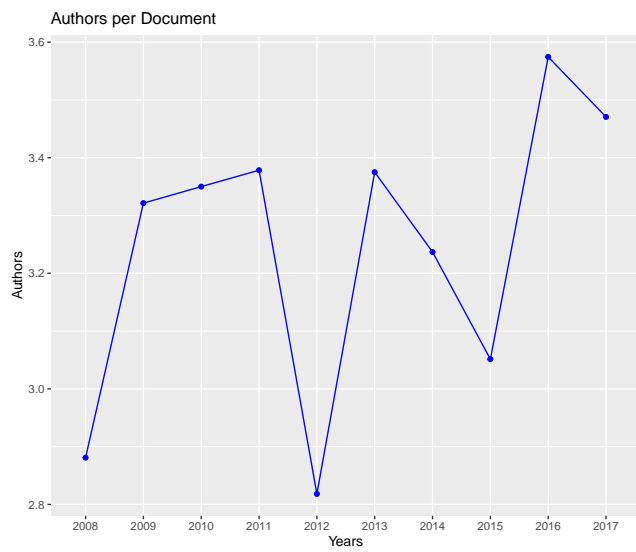




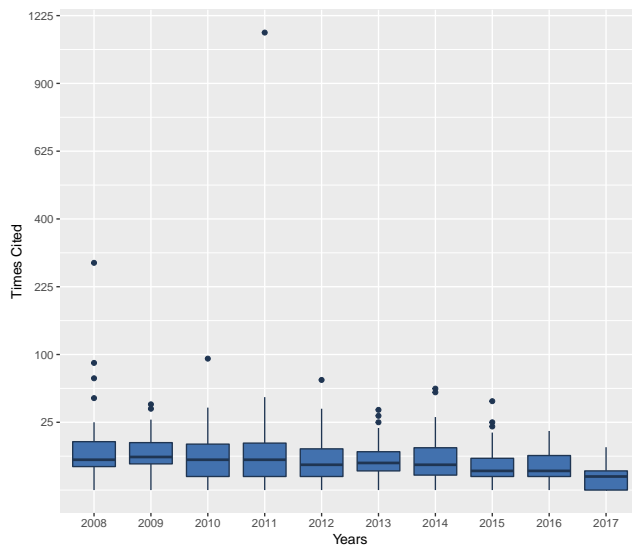
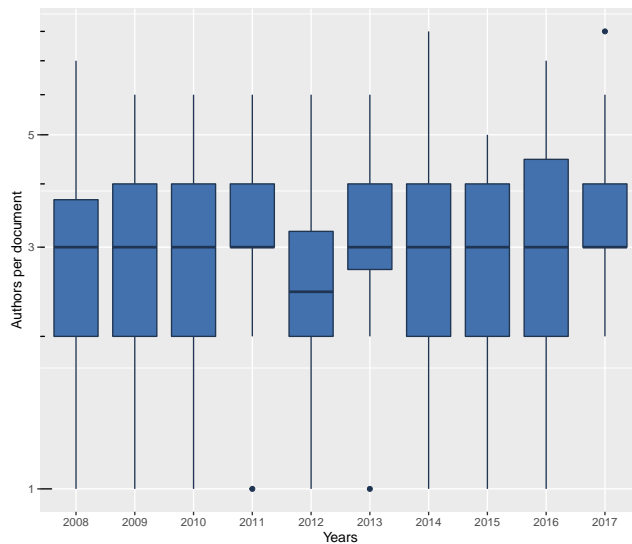


```
plot(res2)
```





```
plot(res2, boxplot = TRUE)
```



## B.5 Filtrado

Se combinan las funciones `get.id<Tabla>()` (se puede emplear cualquier variable de la correspondiente tabla; multiple conditions are combined with `&`, see e.g. `dplyr::filter()`) con la función `get.idDocs()`.

### B.5.1 Funciones `get`

- `get.idAuthors()`: buscar id (códigos) de autores

Buscar un autor concreto:

```
idAuthor <- get.idAuthors(db, AF == "Cao, Ricardo")
idAuthor
```

```
## Cao, Ricardo
##           16
```

Buscar en nombres de autores:

```
idAuthors <- get.idAuthors(db, grepl('Cao', AF))
idAuthors
```

```
##           Cao, Ricardo Cao-Rial, Maria Teresa
##           16                               69
```

- `get.idAreas()`: Devuelve códigos de las áreas

```
get.idAreas(db, SC == 'Mathematics')
```

```
## Mathematics
##           16
```

```
get.idAreas(db, SC == 'Mathematics' | SC == 'Computer Science')
```

```
## Computer Science      Mathematics
##           7              16
```

- `get.idCategories()`: códigos de las categorías

```
get.idCategories(db, grepl('Mathematics', WC))
```

```
##                               Mathematics
##                               28
##                               Mathematics, Applied
##                               29
## Mathematics, Interdisciplinary Applications
##                               30
```

- `get.idJournals()` códigos de las revistas

```
ijss <- get.idJournals(db, S0 == 'JOURNAL OF STATISTICAL SOFTWARE')
ijss
```

```
## JOURNAL OF STATISTICAL SOFTWARE
##                               134
```

```
knitr::kable(db$Journals[ijss, ], caption = "JSS")
```

```
get.idJournals(db, JI == 'J. Stat. Softw.')
```

```
## JOURNAL OF STATISTICAL SOFTWARE
##                               134
```

Tabla B.1: JSS

	idj	SO	SE	BS	LA	PU
2796	134	JOURNAL OF STATISTICAL SOFTWARE			English	JOURNAL STATIS

### B.5.2 Obtener documentos (de autores, revistas, ...)

Los índices anteriores se pueden combinar en `get.idDocs()`

```
idocs <- get.idDocs(db, idAuthors = idAuthor)
idocs
```

```
## [1] 10 16 23 33 40 56 128 183 187 196 210 220 269 286 295 312 315 332 340
## [20] 346 347 350 359 362 372 375 384 385
```

Los índices de documentos se pueden utilizar como filtro p.e. en `summary.wos.db()`.

### B.5.3 Sumarios filtrados

Obtener sumario de autores:

```
summary(db, idocs)
```

```
## Number of documents: 28
## Authors: 40
## Period: 2008 - 2017
##
## Document types:
##           Documents
## Article                26
## Editorial Material      1
## Proceedings Paper       1
##
## Number of authors per document:
##   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
##   2.00   2.00   3.00   3.14   4.00   6.00
##
## Number of documents per author:
##   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
##   1.0    1.0    1.0    2.2   2.0    28.0
##
## Number of times cited:
##   Min. 1st Qu.  Median    Mean 3rd Qu.  Max.
##   0.00   1.00   2.00   3.25   4.25  14.00
##
## Indexes:
## H G
```

```

## 5 7
##
## Top Categories:
##
## Documents
## Statistics & Probability 26
## Mathematics, Interdisciplinary Applications 4
## Computer Science, Interdisciplinary Applications 3
## Economics 2
## Mathematical & Computational Biology 2
## Social Sciences, Mathematical Methods 2
## Automation & Control Systems 1
## Biochemistry & Molecular Biology 1
## Biology 1
## Business, Finance 1
## Others 4
##
## Top Areas:
## Documents
## Mathematics 28
## Computer Science 4
## Business & Economics 2
## Mathematical & Computational Biology 2
## Mathematical Methods In Social Sciences 2
## Automation & Control Systems 1
## Biochemistry & Molecular Biology 1
## Chemistry 1
## Instruments & Instrumentation 1
## Life Sciences & Biomedicine - Other Topics 1
## Others 1
##
## Top Journals:
## Documents
## J. Nonparametr. Stat. 4
## Comput. Stat. Data Anal. 3
## Comput. Stat. 3
## Ann. Inst. Stat. Math. 2
## Test 2
## Stat. Neerl. 1
## J. Multivar. Anal. 1
## J. Time Ser. Anal. 1
## Stat. Probab. Lett. 1
## J. Appl. Stat. 1
## Others 9
##
## Top Countries:
## Documents

```

```
## Spain          28
## Belgium        6
## France         2
## Germany        2
## Argentina      1
## Canada         1
## India          1
## Mexico         1
## Norway         1
```

Obtener sumario de autores por años:

```
summary_year(db, idocs)
```

```
##
## Annual Scientific Production:
##
##      Documents
## 2008          7
## 2009          4
## 2010          4
## 2011          1
## 2012          2
## 2013          3
## 2014          1
## 2016          2
## 2017          4
##
## Annual Authors per Document:
##
##      Avg Median
## 2008 2.4286   2.0
## 2009 3.7500   3.5
## 2010 3.5000   3.5
## 2011 4.0000   4.0
## 2012 3.0000   3.0
## 2013 3.6667   4.0
## 2014 2.0000   2.0
## 2016 2.5000   2.5
## 2017 3.5000   3.5
##
## Annual Times Cited:
##
##      Cites   Avg Median
## 2008    42 6.0000   6.0
## 2009    20 5.0000   3.0
## 2010     9 2.2500   2.0
```



```
## 2011      1 1.0000    1.0
## 2012      1 0.5000    0.5
## 2013      8 2.6667    2.0
## 2014      4 4.0000    4.0
## 2016      3 1.5000    1.5
## 2017      3 0.7500    0.5
```

## B.6 Indices de autores

Obtener índices de múltiples autores

```
TC.authors(db, idAuthors)
```

```
##                               H G
## Cao, Ricardo                   5 7
## Cao-Rial, Maria Teresa        1 1
```



## Apéndice C

# El paquete CITAN

The practical usability of the CITation ANalysis package for R statistical computing environment, is shown. The main aim of the software is to support bibliometricians with a tool for preprocessing and cleaning bibliographic data retrieved from SciVerse Scopus and for calculating the most popular indices of scientific impact.

<https://cran.r-project.org/web/packages/CITAN/index.html>

<https://cran.r-project.org/web/packages/CITAN/CITAN.pdf>

<https://github.com/gagolews/CITAN>

<https://www.gagolewski.com/publications/2011citan.pdf>

```
library(CITAN)
```

```
## Loading required package: agop
## Loading required package: RSQLite
## Loading required package: RGtk2
```

Emplea el paquete RSQLite.

Sin embargo, la función `Scopus_ReadCSV()` produce un error en Windows. Para corregirlo:

```
# Session > Set Working Directory > To Source...
source("datos/citan/Scopus_ReadCSV2.R")
```

### C.1 Creación de la base de datos

Se generará el archivo:

```
dbfilename <- "data/citan/UDC2015.db"
```

### C.1.1 Primera ejecución: Creación del modelo de DB

Creación del archivo de BD vacío:

```
conn <- lbsConnect(dbfilename)
```

```
## Warning in lbsConnect(dbfilename): Your Local Bibliometric Storage is
## empty. Use lbsCreate(...) to establish one.
```

Creación del esquema con lbsCreate():

```
lbsCreate(conn)
```

```
## Warning: RSQLite::dbGetInfo() is deprecated: please use individual metadata
## functions instead
```

```
## Creating table 'Biblio_Categories'... Done.
## Creating table 'Biblio_Sources'... Done.
## Creating index for 'Biblio_Sources'... Done.
## Creating table 'Biblio_SourcesCategories'... Done.
## Creating table 'Biblio_Documents'... Done.
## Creating table 'Biblio_Citations'... Done.
## Creating table 'Biblio_Surveys'... Done.
## Creating table 'Biblio_DocumentsSurveys'... Done.
## Creating table 'Biblio_Authors'... Done.
## Creating table 'Biblio_AuthorsDocuments'... Done.
## Creating view 'ViewBiblio_DocumentsSurveys'... Done.
## Creating view 'ViewBiblio_DocumentsCategories'... Done.
## Your Local Bibliometric Storage has been created.
## Perhaps now you may wish to use Scopus_ImportSources(...) to import source inform
```

```
## [1] TRUE
```

Importar información de Scopus (descargada previamente...) con la función Scopus\_ImportSources() (código):

```
Scopus_ImportSources(conn) # Cuidado con el tiempo de CPU...
```

```
## Importing Scopus ASJC codes... Done, 334 records added.
## Importing Scopus source list...
```

```
## Warning in doTryCatch(return(expr), name, parentenv, handler): No ASJC @
## row=510.
```

```
## Warnings... __TRUNCATED__
```

```
## Done, 30787 of 30794 records added; 55297 ASJC codes processed.
## Note: 7 records omitted @ rows=13847,15526,16606,17371,19418,24419,29365.

## [1] TRUE
```

### C.1.2 Incorporar nuevos datos

Con la función `Scopus_ReadCSV()` se produce un error en Windows:

```
data <- Scopus_ReadCSV("udc_2015.csv")
```

```
## Error in Scopus_ReadCSV("udc_2015.csv") : Column not found: `Source`.
```

Empleando la versión modificada:

```
data <- Scopus_ReadCSV2("udc_2015.csv")
```

Añadir los documentos a la base de datos:

```
lbsImportDocuments(conn, data)
```

```
## Importing documents and their authors... Importing 1324 authors... 1324 new authors added.
```

```
## Warning in .lbsImportDocuments_Add_Get_idSource(conn, record$SourceTitle, :
## no source with sourceTitle='Quaternary Science Reviews' found for record
## 10. Setting IdSource=NA.
```

```
## Warnings... __TRUNCATED__
```

```
## Done, 363 of 363 new records added to Default survey/udc_2015.csv.
```

```
## [1] TRUE
```

Se podría añadir una descripción para trabajar con distintos grupos de documentos:

```
lbsImportDocuments(conn, data, "udc_2015")
```

## C.2 Extraer información de la BD

En siguientes ejecuciones bastará con conectar con la BD

```
conn <- lbsConnect(dbfilename)
```

### C.2.1 Estadísticos descriptivos

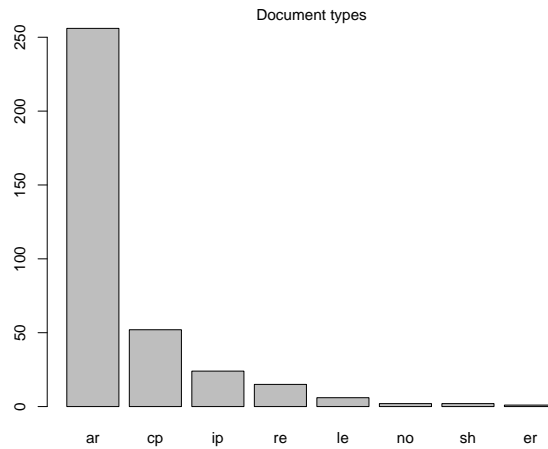
```
lbsDescriptiveStats(conn)
```

```
## Number of sources in your LBS:          30787
```

```

## Number of documents in your LBS:          363
## Number of author records in your LBS:    1324
## Number of author groups in your LBS:     1
## Number of ungrouped authors in your LBS: 1324
##
## You have chosen the following data restrictions:
## Survey:          <ALL>.
## Document types: <ALL>.
##
## Surveys:
##  surveyDescription DocumentCount
##  1   Default survey          363
##  * Note that a document may belong to many surveys/files.

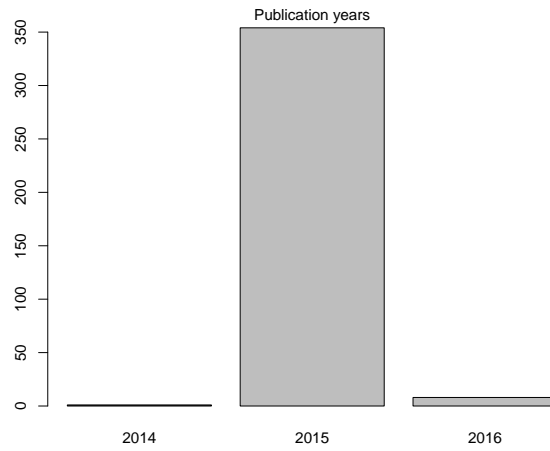
```



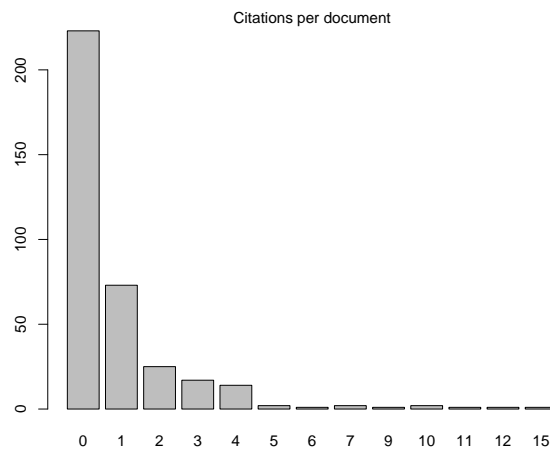
```

## Document types:
##
##  ar  cp  ip  re  le  no  sh  er
## 256  52  24  15   6   2   2   1

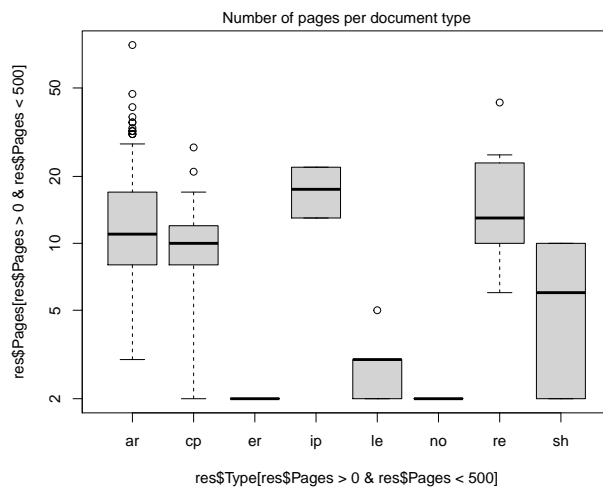
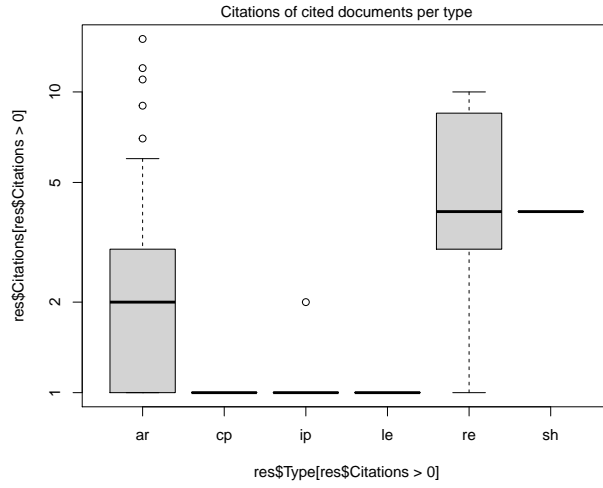
```



```
## Publication years:
##
## 2014 2015 2016
##    1 354   8
```



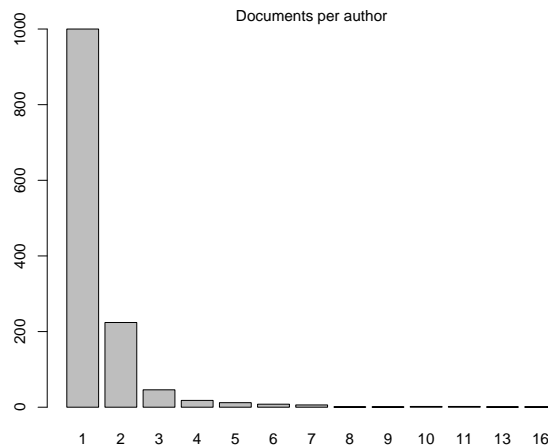
```
## Citations per document:
##
## 0 1 2 3 4 5 6 7 9 10 11 12 15
## 223 73 25 17 14 2 1 2 1 2 1 1 1
```







```
##                                     32
##                               Physics and Astronomy(all)
##                                     19
##                               Energy(all)
##                                     10
##       Business, Management and Accounting(all)
##                                     10
##                               Psychology(all)
##                                     8
##                               Chemistry(all)
##                                     51
```



```
## Documents per author:
##
##   1   2   3   4   5   6   7   8   9  10  11  13  16
## 1000 224  46  18  12   8   6   1   1   2   2   1   1
```

### C.2.2 Otra información

Se puede obtener información acerca de los documentos producidos y las citas recibidas correspondientes a cada autor:

```
citseq <- lbsGetCitations(conn)
```

```
## Data set restrictions:
## Survey:               <ALL>.
## Document types: <ALL>.
##
## Creating citation sequences... OK, 1322 of 1322 records read.
```

```
# citseq <- lbsGetCitations(conn, surveyDescription="udc_2015")
```

Número de autores

```
length(citseq)
```

```
## [1] 1322
```

```
head(names(citseq))
```

```
## [1] "LÓPEZ-GARCÍA X." "MARWAH S." "OTERO T.P."
## [4] "IGLESIAS M.P." "GONZÁLEZ-RIVAS D." "BARROS CASTRO J."
```

```
citseq[[4]]
```

```
## 229 11
## 1 0
## attr(,"IdAuthor")
## [1] 4
```

Se pueden seleccionar autores:

```
id <- lbsSearchAuthors(conn, c("Cao R.", "Naya S.", "Naya-Fernandez S."))
id
```

```
## [1] 46 1193
```

Obtener las citas de los trabajos de los autores seleccionados:

```
citseq2 <- lbsGetCitations(conn, idAuthors=id)
```

```
## Data set restrictions:
## Survey: <ALL>.
## Document types: <ALL>.
##
## Creating citation sequences... OK, 2 of 2 records read.
```

```
length(citseq2)
```

```
## [1] 2
```

Obtener los documentos relativos a los autores seleccionados:

```
id_re <- lbsSearchDocuments(conn, idAuthors=id)
```

Obtener información acerca de los documentos:

```
info_re <- lbsGetInfoDocuments(conn, id_re)
info_re
```

```
## [[1]]
## IdDocument: 16
## AlternativeId: 2-s2.0-84947552209
```

```

## Title: Lifetime estimation applying a kinetic model based on the generalized
## BibEntry: Journal of Thermal Analysis and Calorimetry,2015,122,3,,1203,1212
## Year: 2015
## Type: Article
## Citations: 0
## Authors: NAYA S./46/NA, ÁLVAREZ A./518/NA, LÓPEZ-BECEIRO J./565/NA, GARCÍA-PAI
##
## [[2]]
## IdDocument: 98
## AlternativeId: 2-s2.0-84928890357
## Title: Bootstrap testing for cross-correlation under low firing activity
## BibEntry: Journal of Computational Neuroscience,2015,38,3,,577,587
## Year: 2015
## Type: Article
## Citations: 1
## Authors: ESPINOSA N./779/NA, MARIÑO J./832/NA, CUDEIRO J./1096/NA, CAO R./119
##
## [[3]]
## IdDocument: 127
## AlternativeId: 2-s2.0-84939982743
## Title: Classification of wood using differential thermogravimetric analysis
## BibEntry: Journal of Thermal Analysis and Calorimetry,2015,120,1,,541,551
## Year: 2015
## Type: Article
## Citations: 0
## Authors: NAYA S./46/NA, LÓPEZ-BECEIRO J./565/NA, TARRÍO-SAAVEDRA J./631/NA, FI

```

Obtener las citas de cada documento:

```

cit_re <- sapply(info_re, function(x) x$Citations)
cit_re

```

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

etc...

El último paso será desconectar la BD...

### C.3 Cerrar conexión


```
lbsDisconnect(conn)
```

# Apéndice D

## Instalación de R

En la web del proyecto R ([www.r-project.org](http://www.r-project.org)) está disponible mucha información sobre este entorno estadístico.

---



### The R Project for Statistical Computing

[\[Home\]](#)

- Download**
- [CRAN](#)
- R Project**
- [About R](#)
- [Logo](#)
- [Contributors](#)
- [What's New?](#)
- [Reporting](#)
- [Bugs](#)
- [Development](#)
- [Site](#)
- [Conferences](#)
- [Search](#)


### Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred [CRAN mirror](#).

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

### News

- The R Foundation welcomes five new ordinary members: Jennifer Bryan, Dianne Cook, Julie Josse, Tomas Kalibera, and Balasubramanian Narasimhan.



### 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 (Monday 2016-10-31, Sincere Pumpkin Patch) [R-3.3.2.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots).

---

R-project

CRAN

Las descargas se realizan a través de la web del CRAN (The Comprehensive R Archive Network), con múltiples mirrors:

- *Oficina de software libre* (CIXUG) <ftp.cixug.es/CRAN>.
- *Spanish National Research Network (Madrid)* (RedIRIS) es [cran.es.r-project.org](http://cran.es.r-project.org).

## D.1 Instalación de R en Windows

Seleccionando [Download R for Windows](#) y posteriormente base accedemos al enlace con el instalador de R para Windows.



R-3.5.1 for Windows (32/64 bit)

[Download R 3.5.1 for Windows](#) (62 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

CRAN  
Mirrors  
What's new?  
Task Views  
Search

About R  
R Homepage  
Disc & Forum

Software  
R Sources  
R Binaries  
Packages  
Other

Documentation  
Manuals  
FAQs  
Contributed

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the `md5sum` of the `.exe` to the `md5sum` on the master server. You will need a version of `md5sum` for windows; both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

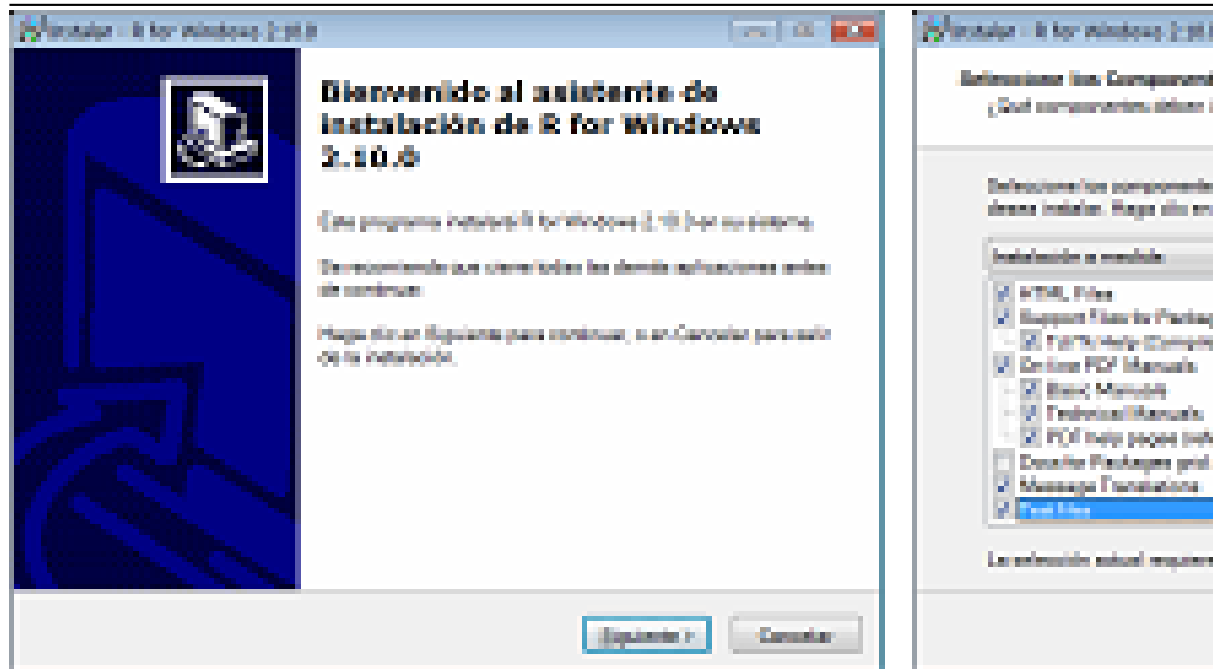
Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

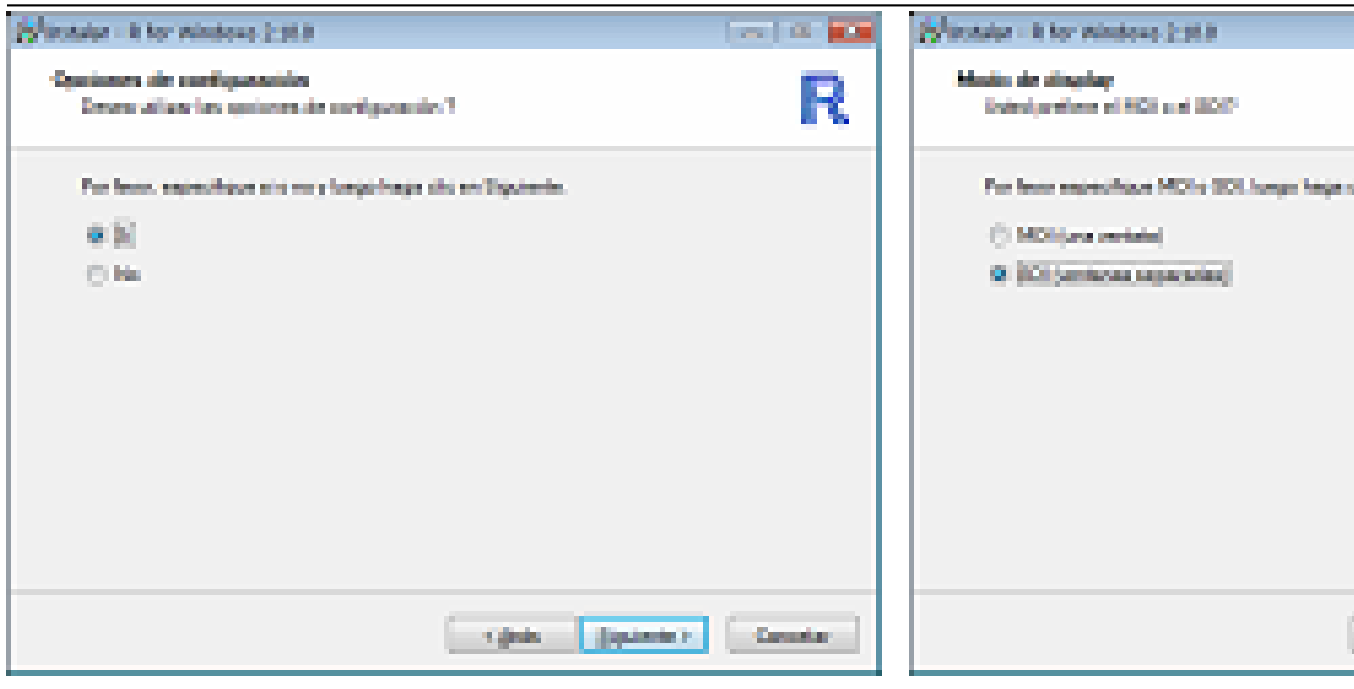
other builds

- Patches to this release are incorporated in the [patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

### D.1.1 Asistente de instalación

Durante el proceso de instalación la recomendación (para evitar posibles problemas) es seleccionar ventanas simples SDI en lugar de múltiples ventanas MDI (hay que *utilizar opciones de configuración*).





Una vez terminada la instalación, al abrir R aparece la ventana de la consola (simula una ventana de comandos de Unix) que permite ejecutar comandos de R.

### D.1.2 Instalación de paquetes

Después de la instalación de R, puede ser necesario instalar paquetes adicionales (puede ser recomendable ejecutar R *como Administrador* para evitar problemas de permiso de escritura en la carpeta *library*<sup>1</sup>).

Para ejecutar los ejemplos mostrados en el libro será necesario tener instalados los siguientes paquetes: *dplyr* (colección *tidyverse*), *tidyr*, *stringr*, *readxl*, *openxlsx*, *RODBC*, *sqldf*, *RSQLite*, *foreign*, *magrittr*, *rattle*, *knitr*. Por ejemplo mediante los comandos:

```
pkgs <- c('dplyr', 'tidyr', 'stringr', 'readxl', 'openxlsx', 'magrittr',
         'RODBC', 'sqldf', 'RSQLite', 'foreign', 'rattle', 'knitr')
# install.packages(pkgs, dependencies=TRUE)
install.packages(setdiff(pkgs, installed.packages()[,'Package']), dependencies = TRUE)
```

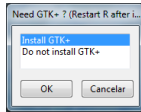
(puede que haya que seleccionar el repositorio de descarga, e.g. *Spain (Madrid)*).

<sup>1</sup>Alternativamente se podrían proporcionar a los usuarios del equipo el permiso *control total* en la carpeta de instalación de R.

La forma tradicional es esta:

1. Se inicia R y se selecciona *Paquetes -> Instalar paquetes*
2. Se selecciona el repositorio.
3. Se selecciona el paquete y automáticamente se instala.

**Rattle** depende de la librería gráfica GTK+, al iniciarlo por primera vez con el comando `library(rattle)` nos pregunta si queremos instalarla:



Pulsamos OK y reiniciamos R.

## D.2 Instalación en Mac OS X

Instalar R de <http://cran.es.r-project.org/bin/macosx> siguiendo los pasos anteriores.

Para instalar **rattle** seguir estos pasos (<https://zhiyzuo.github.io/installation-rattle>):

1. Instalar Homebrew: <https://brew.sh/>.
2. Ejecutar el siguiente código en la consola:

```
system('brew install gtk+')

local({
  if (Sys.info()[['sysname']] != 'Darwin') return()

  .Platform$pkgType = 'mac.binary.el-capitan'
  unlockBinding('.Platform', baseenv())
  assign('.Platform', .Platform, 'package:base')
  lockBinding('.Platform', baseenv())

  options(
    pkgType = 'both', install.packages.compile.from.source = 'always',
    repos = 'https://macos.rbind.org'
  )
})

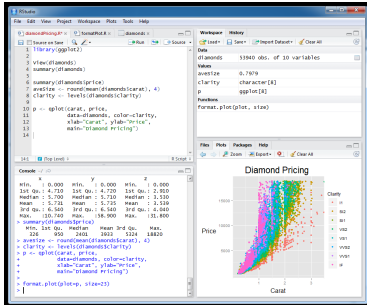
install.packages(c('RGtk2', 'cairoDevice', 'rattle'))
```



## D.3 Instalación (opcional) de un entorno o editor de comandos

Aunque la consola de R dispone de un editor básico de código (script), puede ser recomendable trabajar con un editor de comandos más cómodo y flexible.

Un entorno de R muy recomendable es el **RStudio**, <http://rstudio.org>:



Para instalarlo descargar el archivo de instalación de <http://rstudio.org/download/desktop>.

### D.3.1 Opciones adicionales

Nos puede interesar modificar las opciones por defecto en RStudio, por ejemplo que los gráficos se muestren en una ventana de R o que se emplee el navegador por defecto, para ello habría que modificar (con permisos de administrador) los archivos de configuración *Tools.R* y *Options.R* (en Windows se encuentran en la carpeta *C:\Program Files\RStudio\R*).

Para utilizar el dispositivo gráfico de R, modificar *Tools.R*:

```
# set our graphics device as the default and cause it to be created/set
.rs.addFunction( "initGraphicsDevice", function()
{
  # options(device="RStudioGD")
  # grDevices::deviceIsInteractive("RStudioGD")
  grDevices::deviceIsInteractive()
})
```

Para utilizar el navegador del equipo en lugar del visor integrado de de R, modificar *Options.R*:

```
## custom browseURL implementation
# options(browser = function(url)
# {
#   .Call("rs_browseURL", url) ;
# })
```