

# Master en Big Data. Fundamentos matemáticos del análisis de datos.

## Sesión 1: Presentación. Instalación de Software y Primeros Pasos

Fernando San Segundo

Curso 2019-20. Última actualización: 2019-09-01



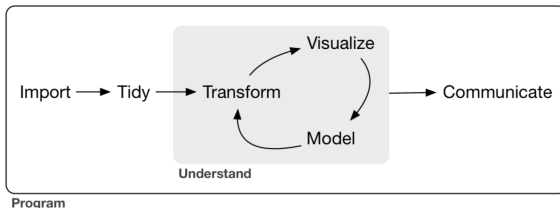
- 1 Presentación del curso.
- 2 Instalación de software
- 3 La interfaz de RStudio
- 4 Variables y asignaciones en R.
- 5 El editor de RStudio. Ficheros de comandos de R.
- 6 Estructuras de datos básicas en R.
- 7 Librerías de R

## Sección 1

Presentación del curso.

# Preliminares.

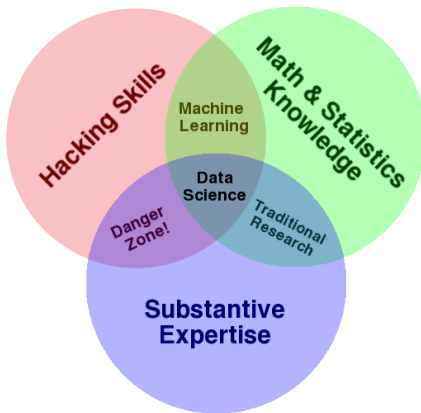
- **Profesor:** Fernando San Segundo, Contacto: [fsansegundo@icai.comillas.edu](mailto:fsansegundo@icai.comillas.edu)
- La Guía Docente de la asignatura, así como el resto del material está accesible desde [Moodle](#).
- La asignatura se plantea con un enfoque práctico y conceptual. Se trata de presentar las ideas fundamentales de la Estadística y el Análisis de Datos y hacerlo mediante la práctica computacional, usando el ecosistema de R como plataforma computacional.
- La estructura básica de un Análisis de Datos es, según (Wickham and Grolemund 2016), esta:



Vamos a conocer las herramientas que R nos aporta para facilitar cada uno de los pasos de este proceso.

## El diagrama de Venn del Análisis de Datos.

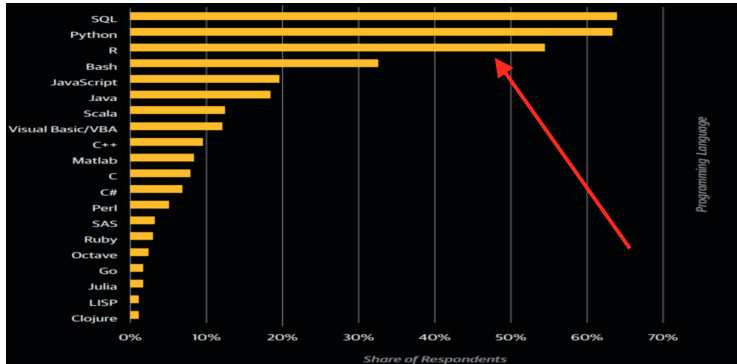
- Este diagrama creado por [Drew Conway](#) es uno de los más usados para describir el Análisis de Datos.



Nuestro curso se va a mover en la parte superior del diagrama, para proporcionaros las competencias necesarias para otras asignaturas del máster.

# El ecosistema de R.

- R es uno de los lenguajes de programación más usados para el análisis de datos y uno de los que han experimentado un crecimiento más rápido **en los últimos años**.



- La comunidad de usuarios de R es uno de los pilares que fundamentan el éxito del lenguaje. La documentación disponible hace extremadamente fácil encontrar respuestas a casi todas las preguntas que vas a plantearte como usuario de R. Además hay un amplísimo catálogo bibliográfico y cientos de cursos disponibles para dominar cualquier aspecto de R. Aquí os enseñaremos algunos de nuestros favoritos.

- Recuerda siempre dónde nos sitúa el diagrama de Venn de Data Science cuando descuidamos en nuestros análisis la componente estadístico-matemático-rigurosa.
- El lenguaje de la estadística es el lenguaje de todas las disciplinas científico-técnicas. Algunos ejemplos:
  - ▶ Encuestas electorales: [CIS](#), [fivethirtyeight](#)
  - ▶ La presencia cada vez mayor de portales de datos en todas las instituciones: [datos.gob.es](#), [datos.madrid.es](#), [www.data.gov](#), [INE](#), [eurostat](#), etc.
  - ▶ Informe del IPCC sobre calentamiento global 2018.
  - ▶ Un artículo reciente del blog económico Nada es Gratis.
  - ▶ Casi cualquier -sin exagerar- artículo de investigación de una revista científica como Nature, Science, etc.
  - ▶ Un ejemplo reciente de visualización sobre violencia y armas en Estados Unidos

## Sección 2

### Instalación de software



# Instalación de R



CRAN  
Mirrors  
What's new?  
Task Views  
Search

About R  
R Homepage  
The R Journal

Software  
R Sources  
R Binaries  
Packages  
Other

Documentation  
Manuals  
FAQs  
Contributed

## 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 (2019-07-05, Action of the Ties) [R-3.6.1.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

- La página oficial de R es [www.r-project.org](http://www.r-project.org)
- Para descargar el instalador de R para tu sistema usa directamente [este enlace](#). Si vas a instalar en Mac o Linux y tienes alguna duda habla con tu profesor. Si tu nombre de usuario en Windows tiene acentos, ñ, etc. habla con tu profesor.
- ¡Instala R antes de continuar!

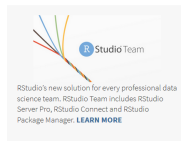
# Instalación de RStudio



[Products](#) [Resources](#) [Pricing](#) [About Us](#) [Blogs](#) [Q](#)

## Choose Your Version of RStudio

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace. [Learn More about RStudio features.](#)



RStudio Desktop  
Open Source License

FREE

[DOWNLOAD](#)

RStudio Desktop  
Commercial License

\$995 per year

[BUY](#)

RStudio Server  
Open Source License

FREE

[DOWNLOAD](#)

RStudio Server Pro  
Commercial License

\$4,975 per year  
(5 Named Users)

[BUY](#)

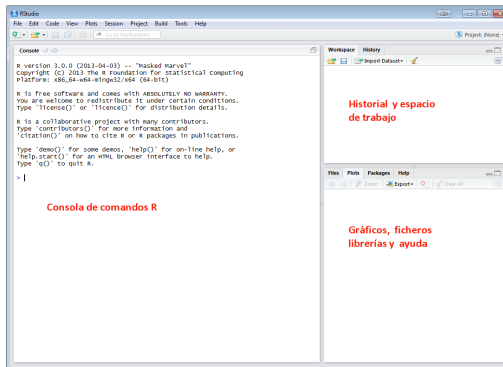
- Trabajar con R directamente no es muy cómodo, así que usaremos un entorno de trabajo llamado **RStudio**. Descarga la versión gratuita (free) para tu sistema de RStudio Desktop usando [este enlace](#).
- ¡Instala RStudio antes de continuar!

- Como lector de documentos pdf puedes usar [Adobe Reader](#), aunque en Windows te recomendamos [y](#) en Mac recomendamos [Skim](#).
- En algún momento puede ser necesario usar un editor de texto. El que incluye RStudio sirve para tareas sencillas, pero en Windows recomendamos [Notepad++](#) y en Mac [BBedit](#).
- Para abrir ficheros de datos es conveniente disponer de una hoja de cálculo como Excel de Microsoft o, preferiblemente, Calc (de [Open Office](#) o de [Libre Office](#)).

## Sección 3

### La interfaz de RStudio

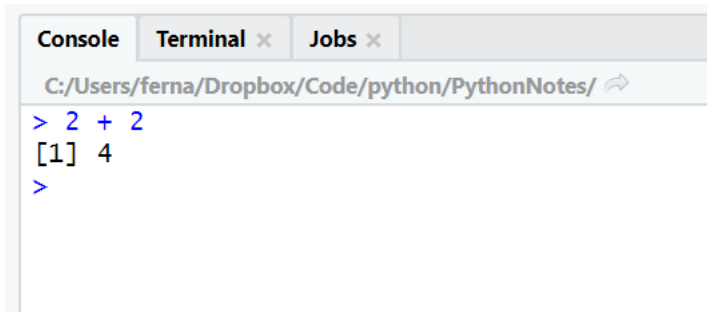
# Primer Contacto



- Abre RStudio. Inicialmente verás que hay tres paneles principales, que vamos a ir conociendo:

- 1 Consola.
- 2 Historial/Entorno.
- 3 Gráficos/Ficheros/Ayuda/Paquetes

- Para empezar, haz clic en la consola (el panel de la izquierda) en la línea con el prompt (símbolo >). Verás el cursor parpadeando. Escribe  $3^2$  y pulsa **Enter**



The screenshot shows an R console window with three tabs: 'Console', 'Terminal', and 'Jobs'. The 'Console' tab is active. The path 'C:/Users/ferna/Dropbox/Code/python/PythonNotes/' is displayed at the top of the console. Below the path, the command '> 2 + 2' has been entered. The output '[1] 4' is displayed on the next line. A new prompt '>' is shown on the following line, indicating the console is ready for the next command.

```
Console Terminal x Jobs x
C:/Users/ferna/Dropbox/Code/python/PythonNotes/
> 2 + 2
[1] 4
>
```

- El uno entre corchetes es la forma en la que R nos indica que esta es la primera línea de su respuesta. Enseguida veremos respuestas que ocupan varias líneas. Debajo hay un nuevo prompt listo para seguir trabajando.

## Ejercicio 1: R como una calculadora

- Copia o teclea cada línea en el prompt (¡practica las dos maneras!), una por una. Trata de adivinar el resultado de cada operación antes de ejecutar el código:.

```
2 + 3
15 - 7
4 * 6
13/5
1/3 + 1/5
sqrt(25)
sqrt(26)
sin(pi)
sin(3.14)
```

# Comentarios sobre el Ejercicio 1

- Usa paréntesis para controlar mejor el orden de las operaciones.
- Los espacios a menudo son irrelevantes pero a veces son esenciales. Por ejemplo `3 + 5` da igual que `3+5`. Usa espacios extra para ganar claridad. Pero **no escribas** `sqrt(25)` en lugar de `sqrt (25)`. Funciona pero ¡es feo!
- `sin` es la función *seno* y `pi` se refiere a la constante matemática  $\pi \approx 3.141593$ . Enseguida hablaremos de otras funciones de R.
- R es un lenguaje **numérico** (no es *simbólico*). No hemos obtenido 0 (que es la respuesta exacta), sino un valor aproximado muy pequeño (en notación científica):  
`1.224606e-16`



## Ejercicio 2: Errores.

- Prueba ahora a escribir esta expresión incompleta:

```
3 +
```

y pulsa Enter. La respuesta + es la forma que tiene R de decirnos “*necesito algo más*” (no tiene nada que ver con la suma). Lo mejor es que uses la tecla Esc y completes la expresión.

- Escribe y ejecuta una a una estas expresiones para ver distintos tipos de errores:

```
4/*3
```

```
2/0
```

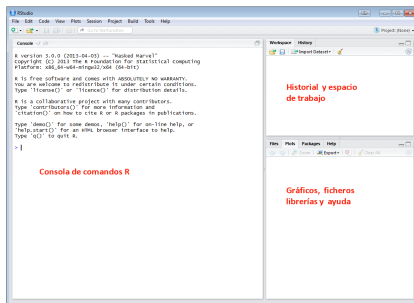
```
sqrt(-4)
```

- Escribe y ejecuta

```
Sqrt(4)
```

y verás que se produce un error. **Es muy importante recordar que R siempre distingue mayúsculas de minúsculas.**

# Detalles adicionales sobre RStudio.



- *Historial de comandos:* Cuando estás trabajando en la consola puedes pulsar la flecha ↑ repetidamente, e irás viendo pasar todos los comandos previos. Con la flecha hacia abajo recorres la lista en sentido contrario. Además en el panel History (arriba a la derecha) puedes ver esa lista y copiar/pegar los comandos o guardarlos como fichero de texto.
- *Limpieza de la consola:* Pulsa Ctrl + L.
- *Ayuda;* El panel Help (abajo a la derecha) nos servirá, cuando aprendamos más, para acceder al sistema de ayuda de R.

## Sección 4

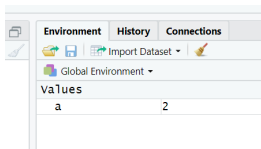
### Variables y asignaciones en R.

# Variables

- Una *variable*, en R, es un símbolo que usamos para referirnos a un valor. Por ejemplo, sitúate en la Consola de Comandos y teclea

```
a = 2
```

Ahora ejecuta esa instrucción. Aunque aparentemente no ha sucedido nada (porque no hay respuesta), a partir de ese momento R ha asignado el valor 2 al símbolo *a*. Fíjate en el panel superior derecho llamado *Environment* (*entorno*), en el que aparece el valor actual de la variable.



- Así que si, por ejemplo, tecleas y ejecutas:

```
a + 1
```

obtendrás como respuesta 3. Comprueba en el panel de entorno que el valor de *a* no ha cambiado. ¿Qué sucede si ejecutas  $a = a + 1$ ? ¿Y si ejecutas  $a = A + 1$ ?

### Ejercicio 3. Operaciones, asignaciones y resultados.

- Ejecuta estos comandos, uno detrás de otro. Trata de imaginar el valor de las variables tras cada operación.

```
a = 2
b = 3
c = a + b
a = b * c
b = (c - a)^2
c = a * b
```

Consulta el panel de entorno para ver cuanto valen las variables tras cada operación.

- Una orden como `c = a + b` en la que guardamos en una variable el resultado de una operación es una **asignación**. Por defecto R no muestra el resultado de las asignaciones al ejecutarlas. Si quieres ver el resultado encierra entre paréntesis *toda la asignación*

```
(c = a + b)
```

```
[1] 115
```

Como ves, ahora R sí muestra el resultado.

- Es recomendable usar nombres informativos para las variables y jugar con las mayúsculas para hacerlos más legibles, como `precioFinal` o `poblacion1995`. Los nombres de variables no pueden empezar por un número ni contener caracteres especiales.
- Aunque hemos usado el igual `=` para asignaciones, es importante que sepas que en R se usa a menudo el símbolo `->` para las asignaciones, de manera que

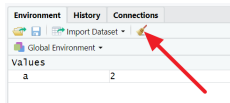
```
a = 2
```

es equivalente a

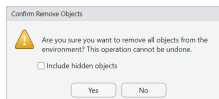
```
a <- 2
```

## Haciendo limpieza antes de seguir adelante.

- Vamos a aprender a eliminar cualquier resto de nuestro trabajo anterior de la memoria de R. Esto es necesario a menudo cuando empezamos un nuevo análisis de datos, para evitar posibles errores debidos a la permanencia de esos resultados previos.
- Empieza usando el menú *Session* de RStudio y haz clic en 'Restart R'. Verás aparecer el mensaje *Restarting R session...* en la Consola.
- Con esto no hemos acabado. Si miras el panel de entorno verás que esa operación no ha eliminado los valores asignados a variables. Para hacer que R olvide esos valores podemos usar el icono de una escoba que aparece encima de ese panel de entorno.



- Al pulsarlo aparecerá un mensaje de confirmación.



Acepta haciendo clic en *Yes* y estaremos listos para seguir.

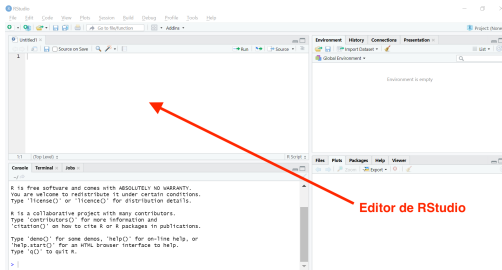
## Sección 5

El editor de RStudio. Ficheros de comandos de R.



# Abriendo el editor

- El trabajo en la Consola de Comandos puede resultar conveniente para algunas operaciones básicas. Pero resulta incómodo para cualquier análisis de datos salvo los más triviales. Vamos a ver otra forma mejor de trabajar.
- Pulsa **Ctrl + Mayúsculas + N** (o, en un Mac, **Comando + Mayúsculas + N**). Alternativamente usa el menú *File* → *New File* → *R Script*, Verás aparecer un panel nuevo en la parte superior izquierda de la ventana, el *Editor* de RStudio.



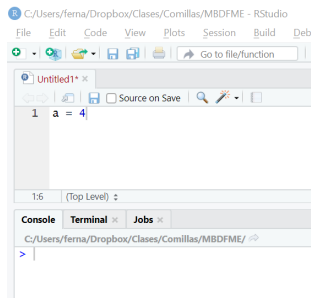
- El editor va a ser el escenario principal de nuestro trabajo con RStudio.

# Ejecutando código en el editor

- Empieza escribiendo esta asignación en el editor

```
a = 4
```

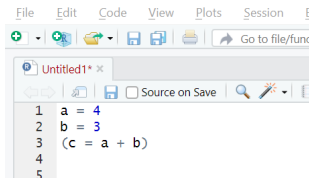
para obtener algo como



Si ahora pulsas la tecla *Enter* (↵) verás que ese código no se ejecuta (mira la consola y el entorno), simplemente pasas a la siguiente línea del editor. Para ejecutar el código vuelve a hacer clic en la línea `a = 4` y ahora pulsa a la vez **Ctrl + Enter**. Verás que el código aparece copiado en la consola como si lo hubieras ejecutado allí.

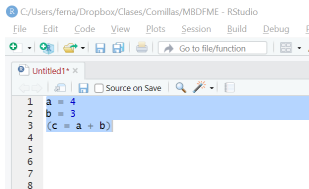
## Ejecutando varios comandos a la vez.

- Para empezar a ver las ventajas de esta forma de trabajar, teclea en el editor dos líneas más de código:



```
File Edit Code View Plots Session E
+ [ Save Print Go to file/function
Untitled1* x
Source on Save
1 a = 4
2 b = 3
3 (c = a + b)
4
5
```

y ahora selecciona esas líneas (usando el ratón o el teclado). Asegúrate de que las tres líneas aparecen completamente seleccionadas, como se muestra aquí:

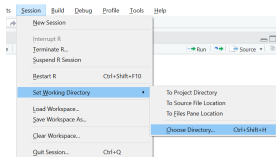


```
C:/Users/ferna/Dropbox/Clases/Comillas/MBDFME - RStudio
File Edit Code View Plots Session Build Debug P
+ [ Save Print Go to file/function
Untitled1* x
Source on Save
1 a = 4
2 b = 3
3 (c = a + b)
4
5
6
7
8
```

- Si ahora usas a la vez **Ctrl + Enter** podrás comprobar en la consola que las tres líneas de código se han ejecutado todas una tras otra.

# Ficheros de comandos de R. Directorio de trabajo.

- Vamos a aprender a guardar en un fichero de comandos (*script*) el trabajo que hacemos en el Editor de RStudio.
- Cada sesión de trabajo con R utiliza una carpeta de nuestro ordenador llamada *Working Directory (Directorio de Trabajo)* para almacenar datos, ficheros de comandos, etc. Es necesario que elijas un directorio de trabajo para este curso y que te acostumbres, al principio de cada sesión, a indicarle a R cuál es ese directorio.
- Dentro de tu carpeta de usuario crea una carpeta para usar como directorio de trabajo en este curso (llámala por ejemplo FME, aunque el nombre no es importante). Si tu nombre de usuario contiene acentos, espacios, la letra ñ, etc. y experimentas algún problema al usar R, habla con tu profesor.
- Crea también dos *subcarpetas* dentro del directorio de trabajo que se llamen *datos* y *scripts* (**¡así, en minúsculas!**).
- Ahora usa el menú *Session* → *Set Working Directory* → *Choose Directory*:

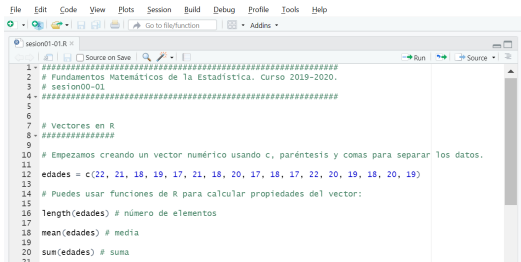


- A continuación usa el menú *File* → *Save As* y guarda el fichero de instrucciones de R con el nombre `sesion01-00` **en la subcarpeta scripts del directorio de trabajo** Al guardarlo RStudio le añadirá la extensión `.R`, que identifica a los ficheros de comandos de R.
- Es recomendable que uses el *Explorador de Archivos* de Windows (o el *Finder* de Mac, etc.) para navegar a la carpeta en la que has guardado ese fichero y lo abras con un *editor de texto* (como el Bloc de Notas). Habla con tu profesor si tienes dudas de como hacer esto. Comprueba que se trata de un fichero de texto que contiene simplemente los comandos que hemos escrito en el Editor de RStudio.
- También puedes abrir un fichero de comandos escrito por otra persona. De hecho en el curso vamos a usar a menudo ficheros de comandos prediseñados para dirigir nuestro trabajo. Empieza descargando este fichero de comandos [sesion01-01.R](#) y guárdalo *en la subcarpeta scripts* de tu directorio de trabajo. En la próxima sección lo abriremos y usaremos para seguir avanzando con R.

## Sección 6

### Estructuras de datos básicas en R.

- Cierra el fichero del Editor de RStudio (usa **Ctrl + W**), cierra RStudio y vuelve a abrirlo. Ahora usa el Menú **File** → **Open File** y abre el fichero **sesion01-01.R** que acabas de descargar. Se abrirá en el Editor.



```
1 - #####
2 # Fundamentos Matemáticos de la Estadística. curso 2019-2020.
3 # sesion00-01
4 - #####
5
6
7 # Vectores en R
8 - #####
9
10 # Empezamos creando un vector numérico usando c, paréntesis y comas para separar los datos.
11
12 edades = c(22, 21, 18, 19, 17, 21, 18, 20, 17, 18, 17, 22, 20, 19, 18, 20, 19)
13
14 # Puedes usar funciones de R para calcular propiedades del vector:
15
16 length(edades) # número de elementos
17
18 mean(edades) # media
19
20 sum(edades) # suma
21
```

Si tienes problemas con los acentos al abrir el fichero pregunta a tu profesor.

- Lo primero en lo que debes fijarte es que R usa el símbolo **#** para introducir comentarios en un fichero de comandos e ignorará el resto de la línea a partir del **#**.
- Vamos a usar los comentarios de ese fichero como guía para aprender las propiedades más básicas de los vectores en R.

## Tablas (Data Frames)

- El punto de partida de nuestros análisis será a menudo una tabla de datos. Pronto aprenderemos a abrir ficheros externos de datos. Pero para empezar vamos a usar una tabla de datos de ejemplo que R trae incorporada al instalarlo.
- Ejecuta este código

```
head(iris)
```

##	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
## 6	5.4	3.9	1.7	0.4	setosa

La tabla contiene datos sobre 150 flores de 3 especies de iris.

La función `head` hace que R solo nos muestre el encabezamiento y las primeras 6 filas de la tabla (con `tail` verías las 6 últimas). Si quieres verla completa usa

```
View(iris)
```

La tabla se abrirá en otra pestaña del editor de RStudio.



- Fíjate en la tabla

```
head(iris)
```

##	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
## 6	5.4	3.9	1.7	0.4	setosa

y observa que:

- ▶ cada columna corresponde a una *variable* (una propiedad observable). Todos los elementos de una columna son del mismo tipo (homogéneos).
  - ▶ cada fila corresponde a una *observación* o *individuo* (una flor en este ejemplo). Los elementos de una misma fila pueden ser de tipos distintos (heterogéneos).
- Podemos obtener las dimensiones (número de filas y columnas) de la tabla con

```
dim(iris)
```

```
## [1] 150  5
```

También podemos obtener los números de filas y columnas directamente con `nrow` y `ncol`.

## Selección de elementos de una tabla

- Para acceder al elemento en la fila 2, columna 3 podemos usar la notación de corchetes:

```
iris[2, 3]
```

```
## [1] 1.4
```

Ese acceso permite cambiar los elementos de la tabla directamente:

```
iris[2, 3] = 7  
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1          5.1          3.5          1.4          0.2  setosa  
## 2          4.9          3.0          7.0          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  
## 6          5.4          3.9          1.7          0.4  setosa
```

## Selección de filas y columnas de una tabla

- Puesto que las columnas están formadas por elementos homogéneos podemos extraer una columna y obtenemos un vector de R. Hay dos formas de hacer esto:

- 1 usando la notación de corchetes con el número de columna y dejando el número de fila en blanco (solo se muestra el principio de la columna).

```
iris[ , 3]
```

```
## [1] 1.4 7.0 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1
## [15] 1.2 1.5 1.3 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5
## [29] 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2 1.3 1.4
```

- 2 Usando el nombre de la columna y el símbolo \$ así:

```
iris$Petal.Length
```

```
## [1] 1.4 7.0 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1
## [15] 1.2 1.5 1.3 1.4 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5
## [29] 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2 1.3 1.4
```

- Para seleccionar una fila podemos usar corchetes, pero al ser heterogénea el resultado es una nueva tabla:

```
iris[2, ]
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 2              4.9              3              7              0.2  setosa
```

- También podemos seleccionar más de una fila o columna. Aquí vamos a seleccionar las filas de la 49 a la 52 y las columnas 1, 3 y 5:

```
iris[49:52, c(1, 3, 5)]
```

```
##      Sepal.Length Petal.Length      Species
## 49              5.3              1.5      setosa
## 50              5.0              1.4      setosa
## 51              7.0              4.7 versicolor
## 52              6.4              4.5 versicolor
```

- Al igual que con los vectores también podemos seleccionar elementos de una tabla usando condiciones lógicas (típicamente comparaciones). Por ejemplo, para seleccionar las filas que corresponden a flores con longitud de sépalo mayor que 2 usamos:

```
iris[iris$Sepal.Width > 2, ]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5         1.4         0.2   setosa
## 2          4.9         3.0         7.0         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
## 6          5.4         3.9         1.7         0.4   setosa
```

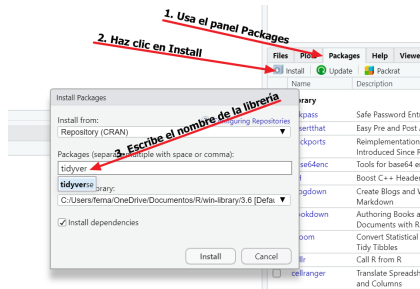
(sólo se muestran las primeras líneas del resultado). Fíjate en que hemos dejado en blanco el número de columna después de la coma para seleccionar todas las columnas. También podríamos haber combinado una condición sobre las filas con otra para seleccionar columnas.

## Sección 7

### Librerías de R

# Instalación y carga de librerías

- La instalación básica de R contiene muchas herramientas pero hay además miles de librerías adicionales disponibles, muchas de ellas muy especializadas para temas concretos (por ejemplo *BioConductor* agrupa miles de programas para Bioinformática). Nosotros vamos a utilizar mucho una librería de herramientas llamada *tidyverse* de H. Wickham.
- Para usar una librería de R debemos instalarla (una única vez) y cargarla (en cada sesión en la que vayamos a usarla). Para instalar una librería sigue los pasos que aparecen en la figura;



R descargará e instalará la librería. Si tienes problemas, pide ayuda al profesor.

- Cuando la instalación concluya puedes cargar la librería para usarla ejecutando

```
library(tidyverse)
```

- **Ejercicio** Además de la anterior instala las librerías llamadas `gapminder` y `nycflights13`.

Carga la segunda de estas librerías con

```
library(nycflights13)
```

Esta librería contiene datos de todos los vuelos que despegaron de aeropuertos de Nueva York en 2013.

- **Ejercicio** ¿Cuántas filas y columnas tiene la tabla? ¿Cuántos vuelos despegaron del aeropuerto JFK? Indicación: usa la función `str` para empezar. ¡¡Es una de las funciones esenciales de R!!



# Un primer encuentro con dplyr

- La librería dplyr es una de los componentes básicos del *tidyverse* y permite simplificar mucho nuestro trabajo con tablas como la que acabamos de examinar. En concreto dplyr proporciona un conjunto de funciones que incluye entre otras:
  - `filter`: para seleccionar algunas filas según sus valores.
  - `arrange`: para reordenar las filas.
  - `select`: para seleccionar columnas (variables)
  - `mutate`: para añadir nuevas columnas calculadas a partir de las existentes.
  - `summarize`: para obtener estimadores estadísticos, por ejemplo medias.
- Además con dplyr vamos a empezar a usar el operador pipe, que en R es `%>%`. Este operador pasa el resultado de una operación como primer argumento de la siguiente. El uso de `%>%` normalmente resulta más claro que la sintaxis básica de R.
- Por ejemplo usando `iris` vamos a buscar, en las columnas que se refieren a pétalos, cuáles son las flores con anchura de pétalo mayor que 2.3.

```
iris %>%  
  select(c('Petal.Length', 'Petal.Width')) %>%  
  filter(Petal.Width > 2.3)
```

##	Petal.Length	Petal.Width
## 1	6.0	2.5
## 2	6.1	2.5
## 3	5.1	2.4
## 4	5.6	2.4
## 5	5.6	2.4
## 6	5.7	2.5

## Y un primer encuentro con ggplot

- La librería `gapminder` que hemos instalado antes contiene una tabla de datos con algunas características socioeconómicas de casi todos los países del mundo a lo largo de varios lustros (proceden de [www.gapminder.org](http://www.gapminder.org)). Carga la tabla y examínala con

```
library(gapminder)
View(gapminder)
```

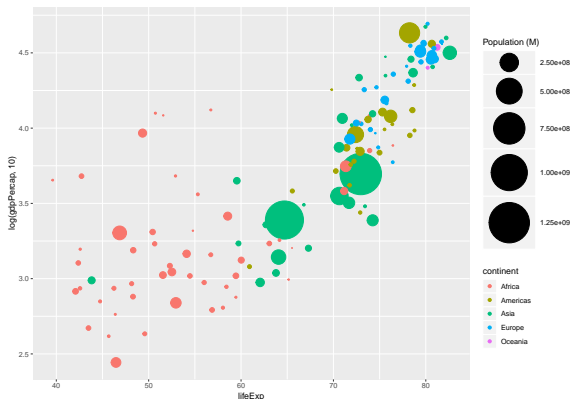
- La librería `ggplot` es otro de los componentes del tidyverse y proporciona herramientas para construir gráficas muy útiles. Su uso puede resultar un poco complejo al principio, pero pronto te acostumbraras. La plantilla básica de un gráfico con `ggplot` es

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Vamos a usar `dplyr` y `ggplot` para dibujar un gráfico de la relación entre esperanza de vida y renta per cápita para el año 2007.

- Los colores corresponden a los distintos continentes y el tamaño de cada punto indica la población del país. La escala vertical (renta per cápita) es logarítmica de base 10. El código es:

```
gapminder %>%  
  filter(year == 2007) %>% # Hasta aquí dplyr, ahora entra en acción ggplot  
  ggplot() +  
  geom_point(mapping = aes(x = lifeExp, log(gdpPerCap, 10),  
                           color = continent, size = pop)) +  
  scale_size(range = c(.1, 24), name="Population (M)")
```



¿Qué preguntas te haces a la vista de este gráfico?

## Enlaces

Junto con el material alojado en Moodle tenéis a vuestra disposición estos recursos:

- [Código de esta sesión](#)
- [R for Data Science \(Wickham\)](#).
- [Repositorio del curso](#) en GitHub.
- Web del libro [PostData](#). Para esta sesión se recomienda el Capítulo 1.
- [Resumen de uso de dplyr](#) elaborado por RStudio.

## Bibliografía

Wickham, H., & Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data*. O'Reilly Media, Inc.