

A collection of approximately 18 squares in various shades of blue and grey, scattered across the top half of the slide.

# MUBD

Màster Universitari en Enginyeria de Dades Massives (Big Data)

Estadística

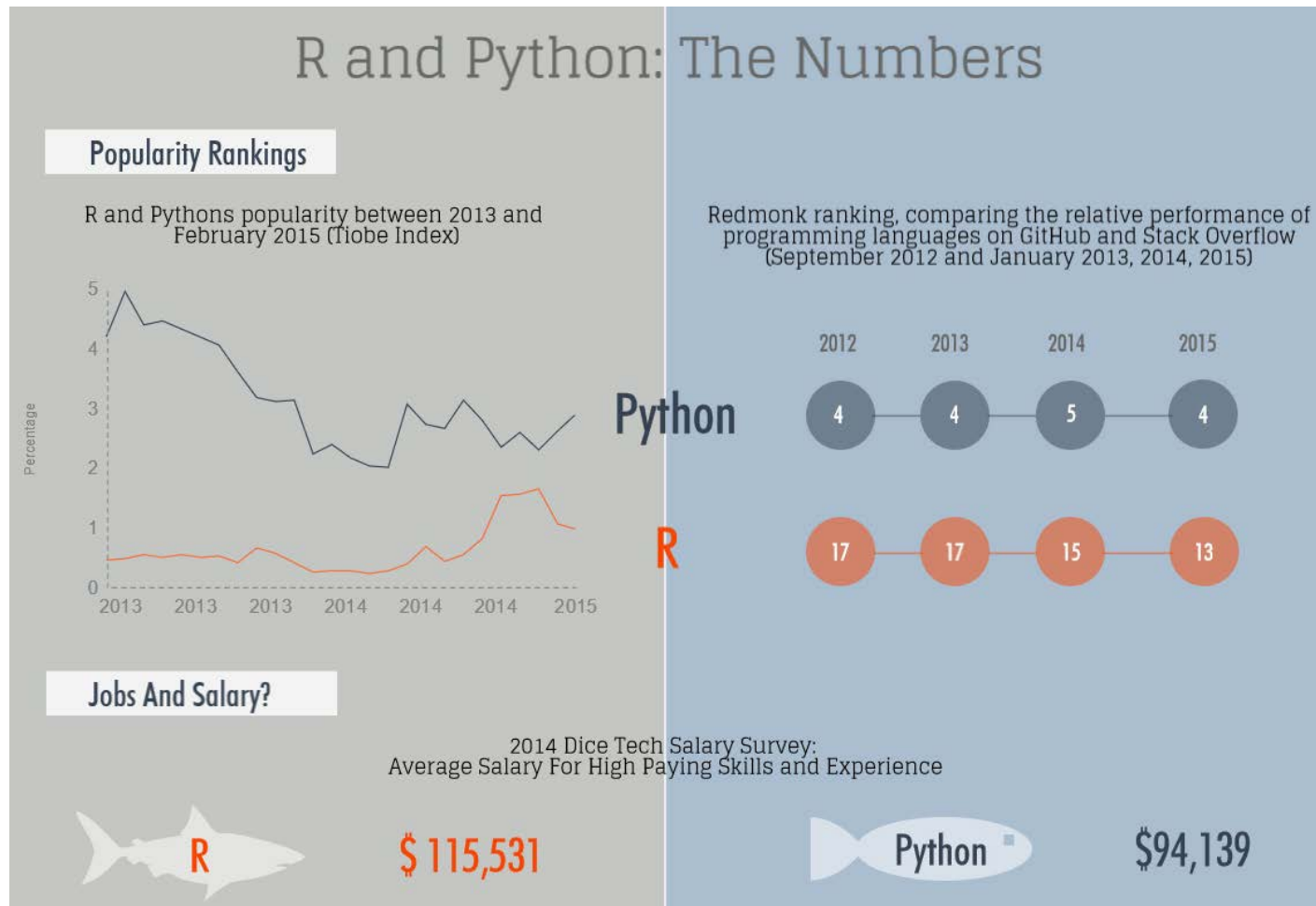
# Visión general

## R

- Página web: <https://www.r-project.org/>
- Repositorio principal: <http://cran.r-project.org/> (Existen otros como [Bioconductor](#) y cada vez más se usa GitHub)
- Funcionalidades
  - Paquete estadístico para la gestión y análisis de datos
  - Lenguaje de programación
  - Herramienta de cálculo
- Ventajas:
  - Software gratuito y de código abierto
  - Gráficos adaptables y de gran calidad con poco esfuerzo.
  - Multitud de librerías con técnicas estadísticas avanzadas

# Why R?

## R versus Python



Source: <https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis>

# Why R?

## R versus Python

Usage	
R is mainly used when the data analysis tasks require standalone computing or analysis on individual servers.	Python is generally used when the data analysis tasks need to be integrated with web apps or if statistics code needs to be incorporated into a production database.
Task	
For exploratory work, R is easier for beginners. Statistical models can be written with a few lines of code.	As a full-fledged programming language, Python is a good tool to implement algorithms for production use.
Data Handling Capabilities	
R is handy for data analysis because of the huge number of packages, readily usable tests and the advantage of using formulas.	The infancy of Python packages for data analysis was an issue in the past, but this has improved a lot!
R is usable for basic data analysis without the installation of packages. Big datasets require the use of packages such as <code>data.table</code> and <code>dplyr</code> , though.	You need to use <code>NumPy</code> and <code>pandas</code> (amongst others) to make Python usable for data analysis.

Source: <https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis>

# Instalación

## R

1. Ir a página web del repositorio 'CRAN': <http://cran.r-project.org/>
2. Clicar en Download R for [SO] ([SO] es Windows, Linux o Mac)
3. Windows
  - Clicar en base (instala paquetes básicos)
  - Clicar en Download R...
  - Descargar fichero y ejecutarlo
3. Mac
  - Escoger última versión
  - Descargar fichero y ejecutarlo
3. Linux
  - Escoger sistema operativo
  - Seguir instrucciones

# Visión general

## RStudio

- Página web: <https://www.rstudio.com/>
- Funcionalidades:
  - Interfaz más amigable para trabajar con R
  - Incorpora menús para funciones habituales
- Ventajas:
  - Creación de proyectos
  - Templates para Rmarkdown, Shiny....
  - Facilidades en escribir el código (ofrece alternativas mientras se escribe)
  - Código más legible (usa colores)
  - Visualización de los datos más directa y con filtros
  - Algunas funcionalidades son más inmediatas y no requieren de código (visualizar datos, instalar/cargar paquetes)

# Instalación

## RStudio

1. Ir a página web de RStudio: <https://www.rstudio.com/>
2. Clicar en *Download* RStudio
3. Clicar en *Rstudio Desktop* (opción *FREE*)
4. Descargar la primera opción que nos ofrezca que será la más reciente para las características de nuestro ordenador
5. Ejecutar el archivo descargado y seguir las instrucciones

# Interfaz

R

## Barra de Herramientas

Instalar paquetes,  
guardar scripts, áreas de trabajo...

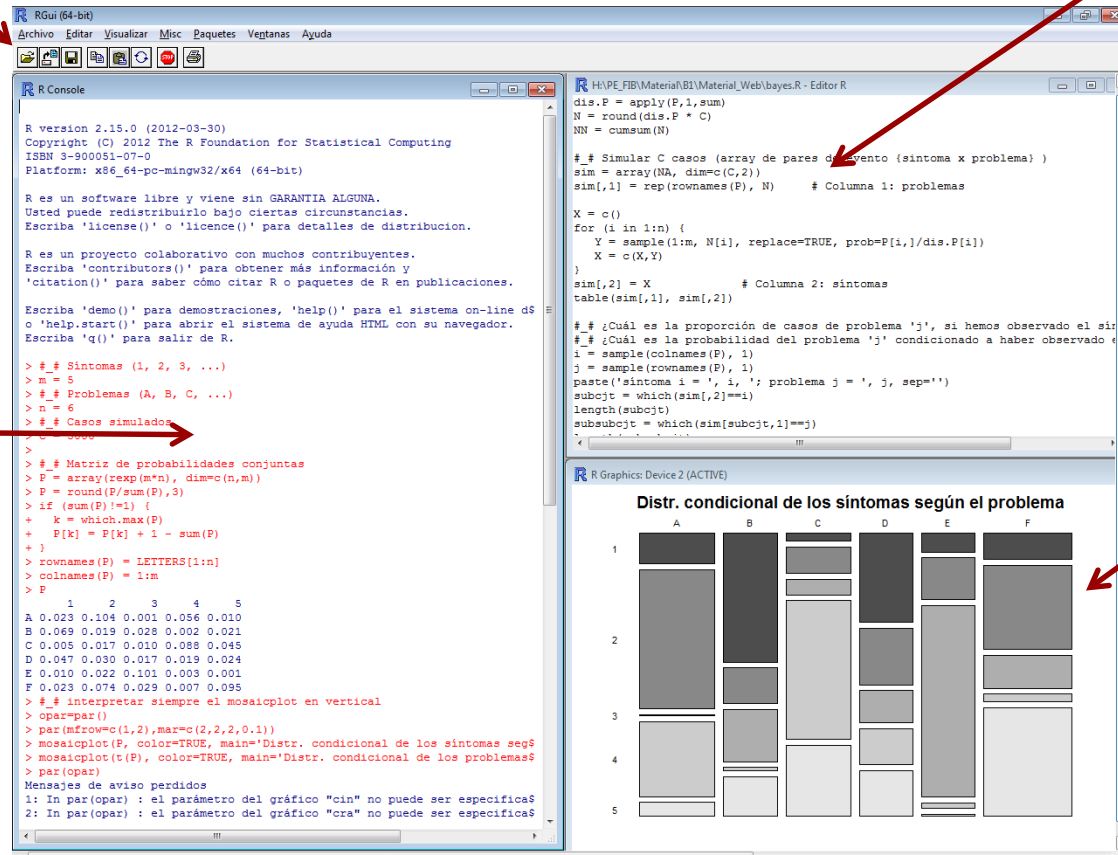
## Script

Fichero de texto donde se guardan las  
instrucciones de análisis

## Consola

Donde se ejecutan las  
instrucciones

## Ventana gráfica





# Interfaz

## RStudio

### Barra de Herramientas

Guardar scripts, configuración del entorno...

### Script

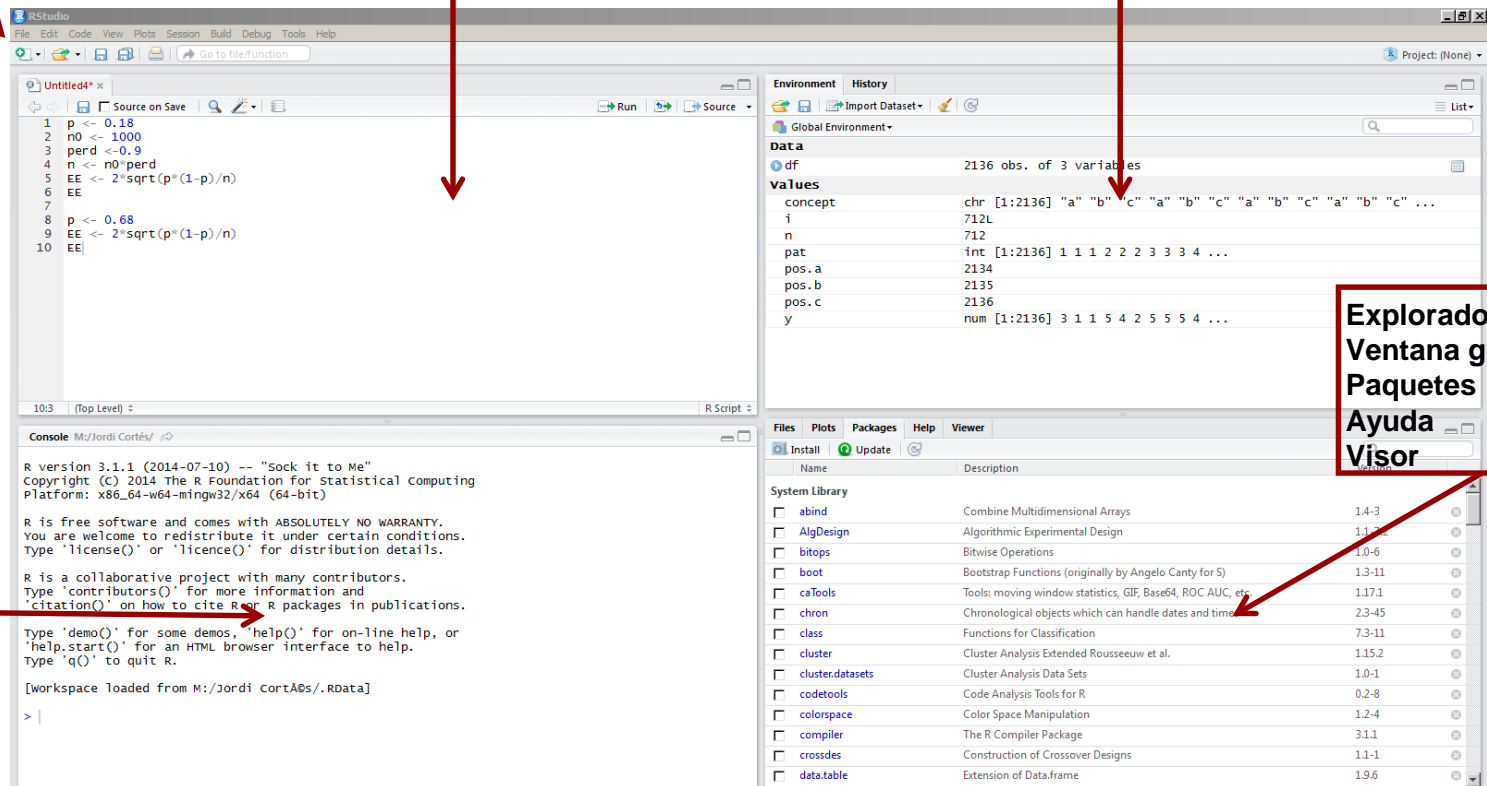
Fichero de texto donde se guardan las instrucciones de análisis

### Entorno

Conjunto de objetos en memoria

### Historial

Conjunto de instrucciones ejecutadas



Explorador  
Ventana gráfica  
Paquetes  
Ayuda  
Visor

# Primeros pasos

## Funciones

- Realizan operaciones, análisis, gráficos, etc...
- Sintaxis: *nombre de la función*(parámetros)
- Ejemplo: `sum( 2 , 3 )`
- El orden de los parámetros puede ser:
  - Por defecto: `plot( 5 , 5 , "h" )`
  - Indicando el nombre del parámetro: `plot( type="h" , x=5 , y=5 )`

# Primeros pasos

## Asignaciones

- Los resultados de funciones o operaciones se pueden asignar a objetos con los símbolos “=” o “<-”
- Sintaxis: `nombre_objeto <- valor_a_asignar`
- Ejemplo: `nota <- sum(2,3)`
- Para ver el contenido de un objeto (p.ej, `nota`), bastará con escribir su nombre en la consola

# Primeros pasos

## Ayuda y comentarios

- Se puede pedir ayuda de una función concreta (?) o de un tema general (??)
  - Sintaxis: ?función o ?? "tema" o escribir en el menú de ayuda
  - Ejemplo: ?mean
- 
- En un código de análisis se pueden escribir comentarios para ayudar a su comprensión. Los comentarios no se ejecutan
  - Sintaxis: # Comentario
  - Ejemplo: # La siguiente línea ajusta un modelo logístico

# Primeros pasos

## Instalar paquetes

- R viene con unos paquetes/librerías base que permiten realizar unos análisis básicos. Existen multitud de paquetes que permiten análisis más sofisticados
- Los paquetes se deben instalar (una única vez en cada dispositivo) y cargar en memoria (en cada sesión)
- Sintaxis:
  - Instalar: `install.packages("nombre_del_paquete")`
  - Cargar: `library(nombre_del_paquete)`
- Ejemplo:
  - `install.packages("Hmisc")`
  - `library(Hmisc)`

# Organización de la información

## Tipos de objetos

- Los objetos simples pueden ser tipo **numeric** (real), **integer** (entero), **character** (cadena de caracteres), **factor** (categorías ordenadas o no), **logical** (TRUE, FALSE) ...
- Se pueden organizar en distintas estructuras:
  - Vectores: conjunto ordenado del mismo tipo. Sintaxis: `c`. Ejemplo: `v <- c(1,2)`
  - Matrices: conjunto de filas y columnas de tipo numérico. Sintaxis: `matrix`. Ejemplo:  
`m <- matrix(c(1,2,3,4),ncol=2)`
  - Data.frames: conjunto de filas y columnas de cualquier tipo respetando que una misma columna sea del mismo tipo. Sintaxis: `data.frame`. Ejemplo: `df <- data.frame(v1=c(1,2),v2=c("a","b"))`
  - Listas: conjunto formado por combinaciones de las anteriores estructuras. Sintaxis: `list`. Ejemplo: `l <- list(v,m,df)`

# Organización de la información

## Acceso y modificación de datos

### ■ Vectores

- Acceso: `v[2]` [consultar 2º elemento]
- Modificación: `v[2] <- 3` [modificar 2º elemento]

### ■ Matrices

- Acceso: `m[1,2]` [consultar 1ª fila, 2ª columna]
- Modificación: `m[1,2] <- 5` [modificar 1ª fila, 2ª columna]

### ■ Data.frames

- Acceso: `m[2,1]` [consultar 2ª fila, 1ª columna]
- Modificación: `m[2,1] <- "a"` [modificar 2ª fila, 1ª columna]

### ■ Listas

- Acceso: `l[[1]]` [consultar 1º elemento]
- Modificación: `l[[1]] <- c(2,3)` [modificar 1º elemento]

# Ejemplos

# Ayuda de una función o de un tema general

```
help(mean)
```

```
?median
```

```
help.search("histogram")
```

# Creación de un vector numérico y calcular su media

```
edad <- c(20,21,20,22,23,20,25,26,20,21)
```

```
edad.mean <- mean(edad)
```

# Creación de un vector de caracteres y hacer la tabla de frecuencias

```
genero <- c("h","h","h","h","h","h","h","d","d","d")
```

```
table(genero)
```

# Otros

```
ls() # Ver los objetos en memoria
```

```
genero # Ver el contenido del objeto genero
```

```
length(edad) # Longitud del vector edad
```

```
class(genero) # Clase del vector genero
```



# Importación y exportación de datos

## Lectura

- La lectura de un fichero, generalmente, retorna un *data.frame*
- Lectura de un fichero de texto (.txt)
  - Sintaxis: `read.table`
  - Ejemplo: `datos <- read.table("fichero.txt")`
- Lectura de un fichero de texto separado por comas (.csv)
  - Sintaxis: `read.csv (comas)` o `read.csv2 (punto y coma)`
  - Ejemplo: `datos <- read.csv("fichero.csv")`
- Lectura de un fichero SPSS (.sav)
  - Sintaxis: `read.spss (librería foreign)` o `read_spss (haven)`
  - Ejemplo: `datos <- read.spss("fichero.sav", to.data.frame=TRUE)`
- En la mayoría de casos, se requerirán más parámetros para realizar la lectura.
- Se pueden leer archivos en casi cualquier formato. Ver [enlace](#).

# Importación y exportación de datos

## Escritura

- Generalmente guardaremos un *data.frame*
- Escritura de un objeto a un fichero de texto
  - Sintaxis: `write.table(objeto_a_guardar, nombre_fichero)`
  - Ejemplo: `write.table(df, "misdatos.txt")`
  - Parámetros opcionales:
    - `append` (¿se añade a un fichero existente?)
    - `quote` (¿se ponen comillas en los valores del fichero?)
    - `sep` (separador entre columnas)
    - `dec` (separador decimal)
    - `row.names`, `col.names` (incluir nombres de filas y columnas)
    - ...
- Para otros formatos, ver [enlace](#).

# R

## Miscelánea

- R distingue mayúsculas y minúsculas. No es lo mismo el objeto *Datos* que *datos*
- Los valores ausentes (missings) los codifica como NA (*Not Available*) o NaN (*Not a Number*)
- La función `summary` es una función genérica que se pueda aplicar a un gran abanico de clases de objetos y nos proporciona un resumen de la información contenida en el mismo.
- Se pueden crear funciones propias con la instrucción `function` (ver `?function`)
- Los valores de tipo *character* o *factor* se deben poner entre comillas
- Las comillas simples (') y dobles (") pueden usarse indistintamente.
- Se puede fijar el directorio de trabajo con `setwd("directorio")`
- Evitar el uso de acentos y símbolos como "ñ" o "ç".

# R

## Tutoriales

- [DataCamp](#)
- [Quick-R](#)
- [TutorialsPoint](#)
- [Edureka](#)

A collection of approximately 18 squares in various shades of blue and grey, scattered across the top half of the slide.

# MUBD

Màster Universitari en Enginyeria de Dades Massives (Big Data)

Estadística