

# **R aplicado al análisis cualitativo / FCS-UdelaR**

**Clase 1 - Educación Permanente FCS**



**Mag. Elina Gómez (UMAD/FCS)**

[elina.gomez@cienciassociales.edu.uy](mailto:elina.gomez@cienciassociales.edu.uy)

[www.elinagomez.com](http://www.elinagomez.com)



Este trabajo se distribuye con una licencia Creative Commons Attribution-ShareAlike 4.0 International License

## Aspectos generales

- Curso presencial (Sala C3)
- Son 8 clases: Martes, Miércoles y Jueves de 16:30 a 19:00 hs.
- Espacio virtual EVA y [repositorio GitHub](#)
- Requisito un 80% de asistencia, para certificado de asistencia.
- Para aprobación se prevé un Trabajo final (fecha de entrega a definir).

# Objetivos del curso

## Bases teóricas:

- Contextualizar las **Ciencias sociales computacionales**
- Emergencia de nuevos recursos y técnicas para la investigación social en la era digital.

# Objetivos del curso

## Introducción al lenguaje “R”:

- R como software libre y gratuito y su interfaz gráfica RStudio;
- Uso de la ayuda y foros;
- Paquetes y funciones.
- Operadores relacionales y lógicos;
- Clases de objetos: Vectores ; Matrices; Data.Frames; y Listas

# Objetivos del curso

## Exploración de fuentes de datos cualitativos:

- Exploración y obtención de datos de diversa índole, contemplando las diferentes fuentes posibles: OCR , web sacraping, prensa digital, redes sociales, APIs.

# Objetivos del curso

## Análisis textual:

- Codificación manual de textos y creación de redes multinivel (categorías, códigos y citas) mediante la plataforma RQDA().
- Abordaje de los requerimientos previos (limpieza y homogenización) para el análisis de textos.
- Trabajo con minería de textos, el cual se centrará en la noción de *corpus* y sus posibilidades analíticas, desde lo más descriptivo a la aplicación de técnicas más complejas.



# Objetivos del curso

## **Análisis textual:**

- Trabajo con diccionarios: Introducción al uso de diccionarios (manuales y automáticos), para la clasificación de documentos masivos según intereses particulares.
- Clasificación de textos: clasificación de textos según temas o emociones asociadas a partir de la aplicación de diferentes técnicas existentes.

# Objetivos del curso

**Visualización:** Exploración de las diferentes posibilidades gráficas de visualización de los resultados del análisis textual (nubes de palabras, frecuencias, dendrogramas, etc.) y algunos ejemplos de visualización interactiva.

# Metodología

- El enfoque del curso es práctico (hands-on)
- Trabajaremos con estrategia de live-coding y ejercicios prácticos para cada tema.
- Posibilidad de clonar repositorio GitHub y trabajar con proyecto y control de versiones.
- <https://github.com/elinagomez/rcuali2022>

Tutorial R+ GitHub

# Objetivos de hoy

- Introducción conceptual y teórica
- Instalaciones previas
- Nivelación de uso básico de R
- Generalidades de la exploración y manipulación de datos con *base* y *tidyverse*.

# Introducción conceptual y teórica

## Contexto

- El abordaje metodológico y técnico que veremos en el curso se enmarca en las llamadas **ciencias sociales computacionales**

## Introducción conceptual y teórica

*“Una nueva disciplina como las Ciencias Sociales Computacionales (Cioffi-Revilla, 2017; Lazer et al., 2009; Mann, 2016) que aglutina la aplicación tanto de las ciencias de la complejidad como del análisis computacional en las Ciencias Sociales, aporta un enfoque innovador para la comprensión del comportamiento humano y social (...) Se promueven así no solo otro tipo de preguntas, sino también la cooperación y la colaboración entre disciplinas. Asimismo, se pueden explorar también nuevas hipótesis a partir de la disponibilidad de nuevos conjuntos de datos y nuevas capacidades de exploración de estos. El desarrollo de nuevos paradigmas y la colaboración entre diferentes campos de conocimiento tienen el potencial de promover nuevos escenarios de investigación, evitando la reiteración en construcciones interpretativas preexistentes (Cioffi-Revilla, 2017; Conte et al., 2012).” (Caro, Jorge et al., 2020)*

# Introducción conceptual y teórica

## Características:

- Perspectiva transdisciplinar
- Dialogo con nuevos desarrollos vinculados a la ciencia de datos, el aprendizaje automático o estadístico, modelado, análisis espacial.
- Explotación del *big data* como fuente de datos y nuevos campos de estudio.
- Humanidades digitales

# Sobre Big data

*“Researchers are in the process of making a change akin to the transition from photography to cinematography.” (Salganik, 2018)*

- Bit by Bit: Social Research in the Digital Age (Matthew J. Salganik, 2018). [Reseña en español](#)
- Big Data: datos en línea (búsquedas y rrss); registros administrativos.



## Sobre Big data

- Diez características según Salganik (2018): *grande; siempre encendido; no reactivo; incompleto; inaccesible; no representativo; a la deriva; algorithmically confounded; sucio; y sensible.*
- Procesamiento: *“contar cosas; pronosticar cosas; y aproximar experimentos”.*
- Nuevas formas de colaboración (crowd-sourcing).
- Desafíos éticos

# R

- ¿Qué es R?
- ¿Cómo interactuamos con R?
- Introducción a RStudio
- Creación de objetos
- Funciones

# ¿Qué es R?

- Un lenguaje de programación y un programa estadístico.
- Es **software libre**: se distribuye con licencia [GNU General Public License](#) que implica libertad de uso, modificación y distribución.
- Es **gratuito**, se descarga desde el [CRAN](#).
- Hay una **comunidad** mundial que usa R y lo mejora constantemente, hoy hay más de 10.000 [paquetes](#) disponibles para descargar
- Usuarixs se ayudan entre sí: [stackoverflow](#), [talkstats](#), [\(rusers\)](#) y localmente [meetup R-Ladies Montevideo](#).

# Consejos

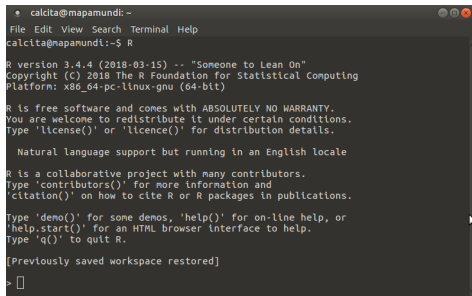
La curva de aprendizaje de R al comienzo suele resultar muy empinada. ¿Cómo podemos evitar o superar la frustración?

- **Usá** R a diario.
- **Traducí** a R una sintaxis sencilla de otro programa que conozcas.
- Recurrí a los **foros** y a la ayuda de R para encontrar las soluciones a los problemas que te surjan: stackoverflow
- Recurrí a otrxs **usuarixs** de R que conozcas.
- Prestá atención a los **mensajes** de error y advertencia.
- **Escribí** tus sintaxis en un script y **comentalas** detalladamente.
- **Reutilizá** sintaxis existentes.

[Hoja de ayuda de R](#)

# Introducción a R: ¿cómo interactuamos con R?

- Abrimos R.
- Aparece un mensaje de apertura.
- Debajo el 'prompt' que es el símbolo `>` ('mayor') e indica que R está listo para recibir órdenes.



```
calcita@mapamundi: ~  
File Edit View Search Terminal Help  
calcita@mapamundi:~$ R  
  
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"  
Copyright (C) 2018 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
[Previously saved workspace restored]  
  
> 
```

# Introducción a R: ¿cómo interactuamos con R?

- Escribiendo un código en la consola
- O escribiendo en el script y luego enviando el código a la consola para obtener un resultado: CTRL + ENTER
- El código utiliza una serie de operadores ('+', '\*', '%in%', etc.), funciones, números, caracteres, etc.
- El script se guarda en un archivo .R

# Introducción a R: ¿cómo interactuamos con R?

- Las órdenes elementales de R consisten en expresiones o asignaciones.
- Una **expresión**, se evalúa, se imprime el resultado y su valor se pierde.

```
2 + 3
```

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

# Introducción a R: ¿cómo interactuamos con R?

- Una **asignación**, crea un objeto y no se imprime el resultado.

```
x <- 2 + 3
```

- Una asignación se hace utilizando el símbolo: <- o también con el símbolo =
- Mientras que otros programas estadísticos muestran directamente los resultados de un análisis, R guarda estos resultados como un 'objeto'.

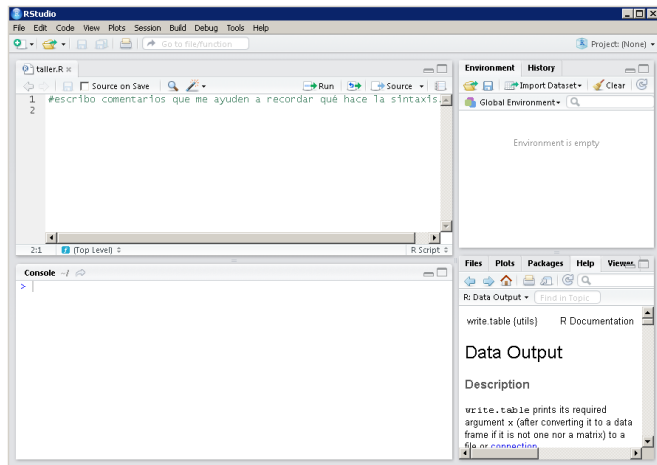


# Introducción a R

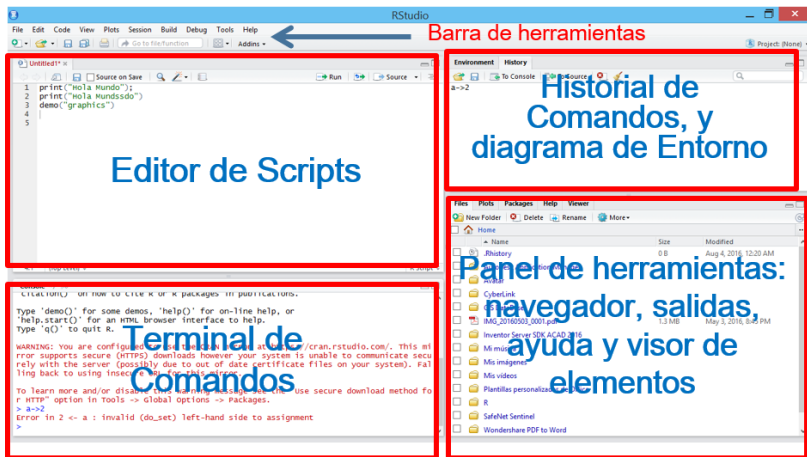
- R es un programa **‘orientado a objetos’**: variables, datos, funciones, resultados, etc., se guardan en la memoria RAM en forma de objetos con un nombre específico sin usar archivos temporales.
- Estos objetos se pueden modificar o manipular con **operadores** –lógicos, aritméticos, comparativos y especiales– y **funciones** –que a su vez son objetos–.
- Bajo este término se esconde la simplicidad y flexibilidad de R.

# Introducción a RStudio

R es mucho más amigable utilizando un editor de código, elegimos **RStudio** porque da muchas facilidades.



# Introducción a RStudio



# Introducción RStudio

RStudio se organiza en 4 ventanas:

- **Ventana superior izquierda:** se escribe el documento, sintaxis, etc.
- **Ventana inferior izquierda:** se ubica la consola donde se imprimen resultados
- **Ventana superior derecha:** se muestran los objetos creados y el historial de comandos ejecutados. También se mostrarán los archivos modificados de un proyecto de control de versiones
- **Ventana inferior derecha:** se muestran los archivos, gráficos, paquetes y ayuda

[Hoja de ayuda de RStudio](#)

# Introducción RStudio

- Para ejecutar una línea de código, colocar el cursor sobre esa línea y ejecutar **Ctrl + ENTER**.
- Para ejecutar varias líneas de código, debemos seleccionarlás todas y ejecutar **Ctrl + ENTER**.
- El símbolo **>** en la consola, indica que R está listo para recibir un comando, mientras que si aparece el símbolo **+**, hay una sentencia no finalizada.
- Mensajes de error (**Error**): errores en la sintaxis (no ejecuta la/s líneas erróneas, pero sí el resto)
- Mensajes de advertencia (**Warnings**): no necesariamente hay un error (ejecuta los comandos y solo te advierte de posibles inconvenientes).
- Importa saber qué tipo de objetos son los que estamos trabajando: no toda función se puede aplicar a cualquier tipo de objeto.

## Creación de objetos

- Al nombrar archivos u objetos, evitar usar tildes y ñ. No dejar espacio entre caracteres.
- Nombre de un objeto debe comenzar con una letra (A-Z, a-z)
- Puede incluir letras, dígitos (0-9), puntos (.) y guión bajo (\_).
- R discrimina entre letras mayúsculas y minúsculas: x y X son objetos diferentes.
- Son nombres válidos *raiz2* o *raiz\_2*, no así *2raiz* ni *raiz-2*

# Objetos: tipos de elementos y clases de objetos

**Vector** (contiene **elementos** de un **mismo tipo**) [ ]

```
a=c(1,0,1,1,1,0) # vector numérico
```

```
b=c("1","0","1","1","1","0") # vector caracter
```

```
d=c(T,F,T,T,T,F) # vector lógico
```

```
f=factor(b,levels=c(1,0),labels=c('Sí','No')) # factor
```

**Matriz** (contiene **vectores** de una **misma clase**) [ , ]

```
e=matrix(c(1,0,1,1,1,0),nrow=3) # matriz numérica
```

**Data Frame** (contiene **vectores** de **cualquier clase**) \$

```
g=data.frame(e) # marco de datos, base de datos
```

**Lista** (contiene **objetos** de **cualquier clase**) [[ ]]

```
e=list(g,f) # lista
```

## Objetos: tipos de elementos y clases de objetos

Todo objeto pertenece a un tipo (`typeof()`), pero a su vez, tiene instancias de una o varias clases (`class()`), con determinados atributos (`attributes()`), y tiene una estructura específica (`str()`).

Objeto	clase	tipo
<code>x &lt;- 1</code>	numeric	double
<code>x &lt;- 'hola'</code>	character	character
<code>x &lt;- TRUE</code>	logical	logical
<code>x &lt;- iris</code>	data.frame	list



# Objetos: tipos de elementos y clases de objetos

Tipo	Verificación	Cambio
vector	<code>is.vector</code>	<code>as.vector</code>
list	<code>is.list</code>	<code>as.list</code>
data.frame	<code>is.data.frame</code>	<code>as.data.frame</code>
matrix	<code>is.matrix</code>	<code>as.matrix</code>
logical	<code>is.logical</code>	<code>as.logical</code>
factor	<code>is.factor</code>	<code>as.factor</code>
character	<code>is.character</code>	<code>as.character</code>
numeric	<code>is.numeric</code>	<code>as.numeric</code>
double	<code>is.double</code>	<code>as.double</code>

# Funciones

*¿Qué es una función?*

- Una función es un conjunto de instrucciones que operan sobre unos argumentos y producen un resultado.
- Una función esconde líneas de código que permite reutilizarlo una y otra vez de manera sencilla.
- Las funciones tienen nombres descriptivos -en inglés- y se acompañan de paréntesis curvos.
- Dentro de los paréntesis se definen los valores de sus argumentos.
- La mayoría tiene al menos un argumento obligatorio y el resto con valores por defecto.

# Funciones

- La función *setwd()* tiene como único argumento '*dir*' y es obligatorio asignarle un valor que debe ser una ruta a una carpeta específica en la computadora:

```
# define el directorio de trabajo  
#setwd(dir = 'home/username/Desktop/')
```

Algunas funciones no necesitan que se defina ningún argumento: *getwd()*, *dir()*, *ls()*

## Carga de datos R

- Carga archivo en formato .Rdata
- `save(obj1, obj2, obj3, file = "ruta/archivo.Rdata")` *Respalda algunos objetos en formato .Rdata*
- `save.image("ruta/respaldo.Rdata")` *Respalda todo el entorno de trabajo*

A diferencia de otras funciones para cargar archivos, la función `load()` es la única en que no se debe asignar a un objeto, ya que para que un archivo sea guardado como Rdata debió haber sido un objeto de R y por ende, al cargar el archivo se carga en la memoria el objeto asociado.

# Carga de datos csv o txt

```
datos <- read.table(file="ruta/archivo.txt", header=FALSE,  
  sep=" ")# otros posibles valores de sep= son " ", ";" o "/"  
datos <- read.csv(file="ruta/archivo.csv", header=TRUE,  
  sep=",")  
datos <- read.delim(file="ruta/archivo.txt", header = FALSE,  
  sep = "/")
```

La función `read.table()` tiene por defecto en el argumento `header` el valor `FALSE`, esto implica que si el archivo tiene nombre de variables, las tomará como un registro más.

Las rutas deben escribirse con la barra `/` y no la contrabarra `\`. Si el archivo está en el directorio de trabajo, alcanza solo con llamar al archivo sin especificar la ruta.

## Algunas funciones básicas: exploración

<b>Función</b>	<b>Descripción</b>
head()	Muestra los primeros X casos de un objeto (por defecto 6)
tail()	Muestra los últimos X casos de un objeto (por defecto 6)
View()	Muestra la base de datos en el visor
length()	Muestra la longitud de un vector
dim()	Muestra las dimensiones del objeto
str()	Muestra la estructura del objeto
names()	Muestra los nombres de las variables
summary()	Muestra resumen descriptivo del objeto
mean()	Muestra el promedio del vector numérico
min()	Muestra el valor mínimo del vector numérico
max()	Muestra el valor máximo del vector numérico

## Algunas funciones básicas: manipulación

Función	Descripción
which()	Indica cuales casos cumplan con la condición especificada
subset()	Selecciona casos según la condición especificada
merge()	Funde bases
apply	Aplica una función a diferentes variables
tapply()	Aplica una función por grupo de casos

## Algunas funciones básicas: presentación (tablas, gráficos, y figuras)

Función	Descripción
table()	Genera una tabla
plot()	Genera gráfico genérico según tipos de variables en cuestión
barplot()	Grafica un diagrama de barras. El argumento principal es una tabla
boxplot()	Grafica un diagrama de caja de una variable numérica o de una variable numérica según categorías.
hist()	Grafica un histograma de frecuencias de un vector numérico
pie()	Grafica un diagrama circular. El argumento principal es una tabla
ggplot()	Función del paquete ggplot2



## Usamos R: ejercicios prácticos

### *Construcción y guardado de objetos:*

- Creamos un objeto llamado 'dias' (sin tilde) que contenga los días a la semana, usamos la función `c()`
- Creamos un objeto llamado 'curso' que contenga 0 y 1, donde los 1 se ubiquen en la posición que se encuentran lunes y miércoles, usamos la función `c()`
- Combinamos ambos vectores en un nuevo objeto llamado 'datos', usamos la función `cbind()`
- Reescribimos el objeto 'datos' y lo convertimos en `data.frame` con la función `as.data.frame`.
- Inspeccionamos la dimensión de 'datos', el nombre de las variables y realizamos una tabla de la segunda variable. Usamos las siguientes funciones: `dim()`, `names()`, `table()`.
- Exportamos el objeto a un archivo `csv`, usamos la función `write.csv()`.

# Usamos R: ejercicios prácticos

## *Exploración:*

- Abrimos el *data.frame* **cars** que viene pre-cargada en el paquete **base** y le asignamos el nombre **autos**; la abrimos en el visualizador.
- Averiguamos el valor máximo y mínimo que tiene la variable **speed**.
- Exploramos la cantidad de filas y columnas que tiene la base
- Aplicar las funciones del tipo *is.x()* y *as.x()* que sirven para verificar si un objeto es de tal tipo y para convertir un tipo de objeto en otro, respectivamente.

## Usamos R: ejercicios prácticos

### *Manipulación:*

- Hacemos una nueva base únicamente con los datos que tengan un valor en la variable speed mayor a 15.
- Hacemos la suma de las columnas de la base **autos** con la función apply y las pegamos al final de la base.
- Creamos otra variable **dist\_rec** que distinga tres tramos de **dist**:  $<20, \geq 20 \ \& \ \leq 40, \geq 41$

# Usamos R: ejercicios prácticos

## *Presentación:*

- Hacemos una tabla de frecuencias de la variable **dist\_rec**
- Hacemos un gráfico de barras de las frecuencias de **dist\_rec**

## Generalidades de tidyverse

- En R, paquetes como **ggplot2** y **dplyr** han adquirido gran popularidad, y ahora forman parte de **tidyverse**
- Estos paquete se basan en los principios de 'tidy data' para mayor consistencia y velocidad de procesamiento:
- Cada Variable forma una columna.
- Cada observación forma una fila.
- Cada tipo de unidad observacional forma una tabla.

# Generalidades de tidyverse

*Recomiendo la consulta:*

- [Hoja de ruta de dplyr en español](#)
- [Hoja de ruta de ggplot en español](#)

## Selección por atributos

- *Base R* incluye las funciones de selección `[`, `subset()` y `$`. y **dplyr** aporta `select()`, `slice()`, `filter()`, and `pull()`.

## Selección por atributos

- `select()` selecciona columnas por nombre o por posición.
- `pull()` extrae una columna como un vector
- `slice()` es el equivalente de `select()` pero para las filas.
- `filter()` es el equivalente en **dplyr** al `subset()` del *base* de R.
- Otras funciones: `contains()`, `starts_with()` y `num_range()`



## Conectando funciones

- Un beneficio de **dplyr** es su compatibilidad con el operador `%>%`.
- Este 'pipe' de R, toma su nombre del pipe de Unix `|`, y es parte del paquete **magrittr**
- Su función es "conectar" la salida de un comando anterior al primer argumento de la siguiente función.
- Esto permite encadenar comandos de análisis de datos, pasando el marco de datos de una función a la siguiente.
- Una ventajas adicional es que fomenta la adición de comentarios a funciones autónomas y permiten líneas simples comentadas sin romper el código.

## Agregación de atributos: ejemplo

- Las operaciones de agregación resumen los conjuntos de datos por una 'variable de agrupación' (típicamente una columna de atributo) o un objeto espacial.
- Para calcular el número de personas por continente en base al objeto `world`
- usaremos la columna `pop` que contiene la población por país y la variable de agrupación `continent`.
- En la base R esto se hace con `aggregate()` y `sum()` de la siguiente manera:

# Agregación de atributos

```
aggregate(pop ~ continent,  
           FUN = sum,  
           data = world,  
           na.rm = TRUE)
```

## Agregación de atributos

`summarize()` es el equivalente en **dplyr** de `aggregate()`, y usa la función `group_by()` para agrupar la variable. Y se implementaría así:

```
group_by(world, continent) %>%  
  summarize(pop = sum(pop, na.rm = TRUE))
```

## Agregación de atributos

- Este enfoque es flexible, lo que permite nombrar las columnas resultantes.
- El omitir la variable de agrupación pone todo en un grupo.
- Esto significa que `summarize()` se puede usar para calcular la población total de la Tierra (~ 7 mil millones) y el número de países.
- Utilicemos `sum()` and `n()` para generar las columnas `pop` y `n_countries`

# Agregación de atributos

```
world %>%  
  summarize(pop = sum(pop, na.rm = TRUE),  
            n_countries = n())
```

## Agregación de atributos

Combinemos todo lo anterior para identificar los 3 continentes más poblados (usando `dplyr::n()` ) y el número de países que contienen:

```
world %>%  
  dplyr::select(pop, continente = continent) %>%  
  group_by(continente) %>%  
  summarize(población = sum(pop, na.rm = TRUE),  
            n_paises = n()) %>%  
  slice_max(n = 3, order_by = población)
```

## Agregación de atributos

**Table 3:** Los 3 continentes más poblados, y su número de países.

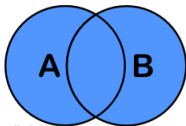
continente	población	n_paises
Africa	1147005839	51
Asia	4306025131	47
Europe	739178065	39



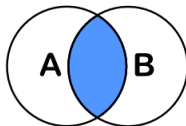
## Combinando objetos

- Join combina tablas basadas en una variable compartida (“key”)
- **dplyr** presenta varias funciones para ello:
  - `left_join()` - Une las filas coincidentes de b en a
  - `right_join()` - Une las filas coincidentes de a en b
  - `inner_join()` - Une reteniendo solo las filas de ambos conjuntos,
  - `full_join` - Une los datos conservando todos los valores, todas las filas
  - `semi_join()` - Todas las filas en a que tienen una coincidencia en b
  - `anti_join()` - Todas las filas en a que no tienen una coincidencia en b
- Estos nombres de funciones siguen las convenciones utilizadas en el lenguaje de bases de datos [SQL](#).

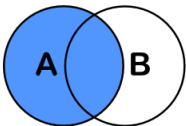
## Combinando objetos



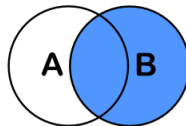
full\_join(A,B)



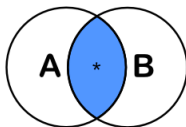
inner\_join(A,B)



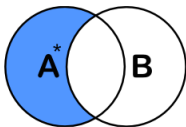
left\_join(A,B)



right\_join(A,B)



semi\_join(A,B)



anti\_join(A,B)

## Combinando objetos

- Las funciones “...\_join” de **dplyr** trabaja tanto con `data.frames`
- El orden de los factores altera el resultado... la clase del primer objeto es la que conserva el resultado.
- Nos centraremos en los left e inner “...\_join” que son los más utilizados, que utilizan la misma sintaxis que los otros tipos de unión.

### Combinando objetos

dplyr	base
<code>left_join(x, y)</code>	<code>merge(x, y, all.x=TRUE)</code>
<code>right_join(x, y)</code>	<code>merge(x, y, all.y=TRUE)</code>