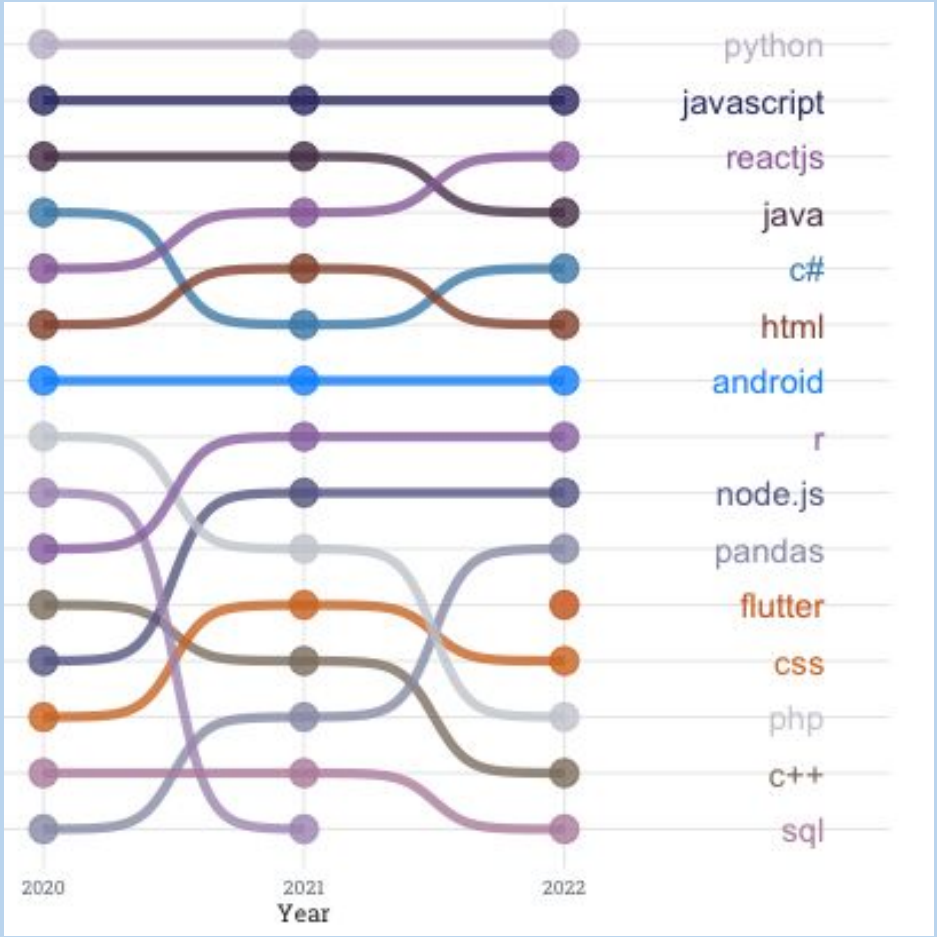
Fundamentos de R

Curso acelerado de periodismo de datos

Montse Hidalgo y Borja Andrino (EL PAÍS)

10. Fundamentos de R

- ¿Qué es R?
- ¿Por qué R? ¿En qué lo utilizamos?
- Instalación de R y RStudio
- Entorno tidyverse
- Algunas funciones
- Fuentes



¿Qué es R?

Un lenguaje de programación orientado al análisis **estadístico**.

Es libre y existe una gran comunidad desarrollando paquetes y

¿Por qué R?

Agilidad

Nos permite analizar datos y generar visualizaciones en poco tiempo

EL PAÍS

www.elpais.com — EL PERIÓDICO GLOBAL

MIÉRCOLES 12 DE ENERO DE 2022 | Año XLVII | Número 16.246 | EDICIÓN NACIONAL | Precio: 1,80 euros

UE Fallece Sassoli, presidente del Parlamento Europeo P3

CIENCIA Revivicor, la empresa que cría cerdos para trasplantes de órganos P25

Las diez provincias con mayor nivel de ocupación en UCI con respirador

% de camas UCI con respirador ocupadas por pacientes Covid

Girona 67% · Huesca 65% · Lleida 52% · Albacete 47% · Barcelona 45%

Palencia 43% · Tarragona 42% · Gipuzkoa 39% · León 38% · Soria 38%

Fuente: Ministerio de Sanidad. EL PAÍS

Más de media España tiene las UCI en riesgo muy alto

En 26 provincias, más del 25% de las camas de críticos son para casos de covid

Los hospitalizados suponen la mitad que hace un año, pero van en claro ascenso

La OMS prevé que la mitad de la población europea se contagiará en dos meses

D. GRASSO / B. ANDRINO, **Madrid**

La explosión de contagios de covid devuelve la presión a los hospitales de media España. Actualmente, hay 14.000 pacientes hospitalizados y 2.000 en unidades de cuidados intensivos (UCI) con respirador. Es la mitad que hace un año, pero se observa una clara tendencia al alza. En 26 provincias, una de cada cuatro camas de UCI está ocupada por un paciente covid, lo que significa que están en el nivel de riesgo considerado muy alto. Álava, Bizkaia o Girona tienen casi tantas camas de críticos ocupadas como en el invierno pasado, cerca de un 80%. Canarias, Navarra, País Vasco, Aragón y Cantabria ya registran las peores cifras de hospitalización en casi seis semanas. La mitad de la población europea se contagiará con la ómicron. PÁGINAS 22 A 24

La vida diaria vuelve, con sus penalidades, tras la dura represión de las protestas

Kazajistán, bajo la paz de los tanques

GUILLERMO ABRIL, **Shymkent (Kazajistán)** ENVIADO ESPECIAL

Si no fuera por la tanqueta de color verde oliva apostada en un control militar a la entrada de Shymkent, nadie diría que en este país ha habido, hace unos días, unas violentas revueltas que han puesto en jaque al Estado. La primera ciudad que uno se encuentra al entrar en Kazajistán desde la vecina Uzbekistán, atravesando el paso fronterizo de la Ruta de la Seda, se despliega como una sucesión de edificios deslucidos y aceras agrietadas por las que vuelve a fluir la vida. Sacar dinero de los atiborrados cajeros se ha convertido en una misión imposible. PASA A LA PÁGINA 5

Un autobús quemado en los disturbios, ayer en Almaty. / V. TRETYAKOV (AP)

EE UU coordina su respuesta a Rusia con la UE, ninguneada por Putin

Washington reúne a los 27 embajadores ante el diálogo con Moscú

MANUEL V. GÓMEZ, **Bruselas**

Estados Unidos intensificó ayer sus contactos con los socios de la Unión Europea, preocupada por la escalada de tensión en Ucrania y por el ninguneo al que la somete el régimen de Vladimir Putin en las negociaciones para encontrar una salida. La subsecretaria de Estado, Wendy Sherman, se reunió con los embajadores de los 27 en el Comité de Seguridad y Defensa de la UE y con el secretario general de Acción Exterior, Stefano Sannino. Hoy se celebrará una reunión entre la OTAN y Rusia en relación al conflicto. Página 4

El PSOE deja solo a Garzón y se muestra como defensor de los ganaderos

CARLOS E. CUÉ, **Madrid**

El PSOE desplegó ayer su estrategia de dejar solo al ministro de Consumo, Alberto Garzón, en sus críticas a la ganadería intensiva y las macrogranjas. El ministro de Agricultura, Luis Planas, y otros miembros del Ejecutivo se desmarcaron abiertamente de Garzón y expresaron su apoyo al sector ganadero con la vista puesta en las elecciones autonómicas de Castilla y León. PÁGINAS 14 Y 15

La caída del poder adquisitivo de los salarios agrava la desigualdad

A. SÁNCHEZ / I. FARIZA, **Madrid**

Millones de asalariados lidian en España con la pérdida de poder adquisitivo. Tras años de precios aletargados, el abrupto despertar de la inflación ahonda en la desigualdad en el país. Un estudio de la Fundación La Caixa alerta de que, desde la crisis de 2008, "la peor evolución la han registrado las rentas más bajas, y la mejor, las más altas". PÁGINA 38

¿Por qué R?

Agilidad

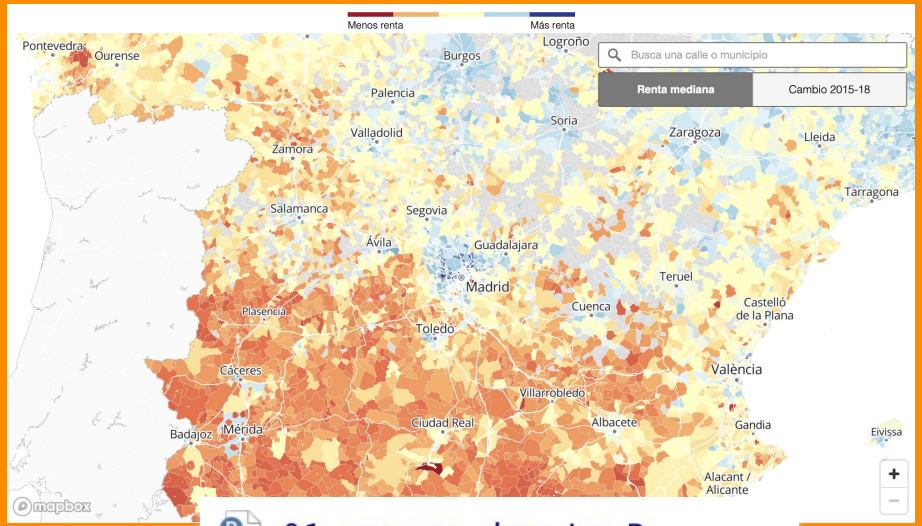
Nos permite analizar datos y generar visualizaciones en poco tiempo

Reutilización

Recurrimos continuamente a código ya desarrollado

Reutilización

Trabajar ordenados y con dinámicas de trabajo parecidas nos permite dividirnos cuando hay poco tiempo



01-prepare_data_ine.R
02-prepare_mapbox.R
03-prepare_shapes.R
data
data_raw
grabacion.mov
mapbox
tema_ine_renta_21.Rproj

¿Por qué R?

Agilidad

Nos permite analizar datos y generar visualizaciones en poco tiempo

Reutilización

Recurrimos continuamente a código ya desarrollado

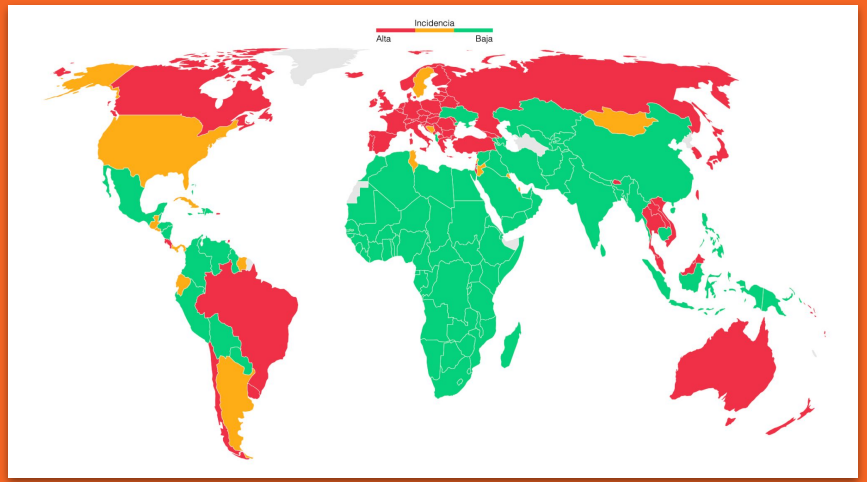
Automatización

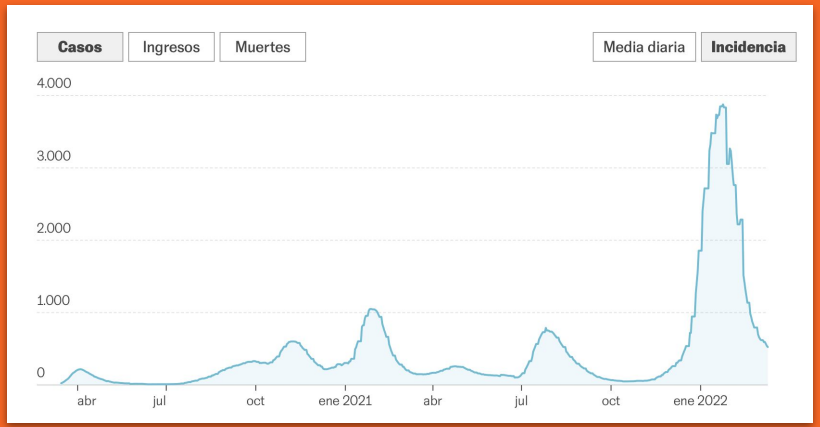
Trackers

Estas citas son solo para fines ilustrativos.

Automatización

Trackers con información actualizada cada día/semana para poder seguir explorando datos





Instalación R

Necesitamos instalar el lenguaje de programación (R) y un entorno de desarrollo para trabajar con él (RStudio).

Sigue los pasos de este link para la instalación.


tidyverse
www.rstudio.com


dplyr
ggplot2
forcats
tibble
readr
stringr
tidyr
purrr

¿Cómo leemos los archivos?

```
read_csv y otros primos
```

Parámetros más usados

- Ruta del fichero (local o url)
- col_names
- col_types
- locale = default_locale(),
- guess_max = min(1000, n_max)

Otras funciones similares

- read_csv2
- read_tsv
- **read_delim**
- googlesheets4::read_sheet

¿Y para escribir?

```
write_csv y sus respectivos primos
```

Parámetros más usados

- Ruta del fichero (local o url)
- col_names
- col_types
- locale = default_locale(),
- guess_max = min(1000, n_max)

Otras funciones similares

- read_csv2
- read_tsv
- **read_delim**

Selección y creación de columnas

```
select(variable, col1, col2, col3…)
```

- Nos permite seleccionar columnas de un **tibble**
- Es cómodo para visualizar las variables que nos interesan
- También lo podemos usar para **renombrar** columnas

Selección y creación de columnas

```
select(my_data, codigo_mun, partido, votos_pc)
```

codigo_mun	censo	part	partido	votos_pc
01001	2020	68.56	PNV	27.77
01001	2020	68.56	PSOE	20.70
01001	2020	68.56	UP	18.15
01001	2020	68.56	Bildu	17.49
01001	2020	68.56	PP	8.31
01001	2020	68.56	VOX	3.94
01001	2020	68.56	OTROS	1.68
01001	2020	68.56	Cs	1.60

codigo_mun	partido	votos_pc
01001	PNV	27.77
01001	PSOE	20.70
01001	UP	18.15
01001	Bildu	17.49
01001	PP	8.31
01001	VOX	3.94
01001	OTROS	1.68
01001	Cs	1.60

Selección y creación de columnas

```
my_data %>% select(codigo_mun, partido, votos_pc)
```

codigo_mun	censo	part	partido	votos_pc
01001	2020	68.56	PNV	27.77
01001	2020	68.56	PSOE	20.70
01001	2020	68.56	UP	18.15
01001	2020	68.56	Bildu	17.49
01001	2020	68.56	PP	8.31
01001	2020	68.56	VOX	3.94
01001	2020	68.56	OTROS	1.68
01001	2020	68.56	Cs	1.60

codigo_mun	partido	votos_pc
01001	PNV	27.77
01001	PSOE	20.70
01001	UP	18.15
01001	Bildu	17.49
01001	PP	8.31
01001	VOX	3.94
01001	OTROS	1.68
01001	Cs	1.60

Selección y creación de columnas

`select: algunos trucos`

- Selecciona varias columnas a la vez **(:)**
- Deselecciona columnas **(-)**
- Selecciona columnas que comiencen por un texto **(starts_with())**
- Selecciona columnas que terminen por un texto **(ends_with())**
- Selecciona columnas que contengan por un texto **(contains())**
- Selecciona columna que cumplan una expresión regular **(matches())**

Mutate

```
my_data %>% mutate(new_col = …)
```

Sirve para crear nuevas columnas, a partir de datos existentes en el tibble o nuevos datos (y de camino vamos a aprender alguna cosita)

```
my_data %>%
  mutate( tipo_partido = if_else(partido %in% c( "PSOE", "PP", "VOX", "UP", "Cs"),
                                 "Ámbito nacional",
                                 "Otros")) %>%
  distinct(partido, tipo_partido)
```

```
my_data %>%
  mutate( fecha = lubridate::today())
```

Mutate

```
my_data %>% mutate(new_col = …)
```

```
my_data %>%
  mutate( candidato = case_when( partido == "PSOE" ~ "Sánchez",
                                 partido == "PP" ~ "Casado",
                                 partido == "VOX" ~ "Abascal",
                                 partido == "UP" ~ "Iglesias",
                                 partido == "Cs" ~ "Rivera",
                                 partido == "MP" ~ "Errejon",
                                 T ~ "Otros"))
```

Filtros

```
my_data %>% filter(cond1, cond2…)
```

Comparadores habituales:

- == igual que
- != distinto que
- > < mayor que, menor que
- >= <= mayor o igual que, menor o igual que
- %in% los valores pertenecen a un listado
- is.na los valores de la variable son NA
- !is.na los valores de la variable no son NA (pero preferimos drop_na)

Filtros

```
my_data %>% filter(cond1, cond2…)
```

& and

x	y	x & y
F	F	F
F	T	F
T	F	F
T	T	T

| or

x	y	x & y
F	F	F
F	T	T
T	F	T
T	T	T

xor

x	y	x & y
F	F	F
F	T	T
T	F	T
T	T	F

Filtros

```
my_data %>%
  filter( votos_pc == 100)
```

codigo_mun	censo	blancos	nulos	candidaturas	part	partido	votos	votos_pc
42108	10	0	0	7	70.00000	PP	7	100
42175	22	0	0	16	72.72727	PP	16	100

Filtros

```
my_data %>%
  filter( partido == "VOX" & votos_pc > 50)
```

codigo_mun	censo	blancos	nulos	candidaturas	part	partido	votos	votos_pc
09422	19	0	0	13	68.42105	VOX	8	61.53846
16083	44	0	0	24	54.54545	VOX	15	62.50000
19095	11	0	1	6	63.63636	VOX	4	66.66667
42148	9	0	0	9	100.00000	VOX	5	55.55556
44060	32	0	0	25	78.12500	VOX	14	56.00000
44208	27	0	0	17	62.96296	VOX	9	52.94118
47216	111	0	3	91	84.68468	VOX	50	54.94505

Filtros

```
my_data %>%
  filter((partido == "ERC" | partido == "JxCAT") &
           votos_pc > 50)
```

codigo_mun	censo	blancos	nulos	candidaturas	part	partido	votos	votos_pc
08093	25	0	0	22	88.00000	ERC	13	59.09091
08195	75	0	0	62	82.66667	JxCAT	34	54.83871
08225	94	1	1	81	88.29787	JxCAT	42	51.21951
08308	139	1	0	108	78.41727	JxCAT	62	56.88073
17007	1763	9	8	1350	77.53829	JxCAT	716	52.68580
17133	1262	7	12	957	77.33756	JxCAT	494	51.24481
25082	292	2	0	195	67.46575	ERC	106	53.80711
25088	96	0	0	73	76.04167	JxCAT	43	58.90411

Filtros

```
my_data %>%
  filter( partido %in% c( "ERC", "JxCAT")
          & votos_pc > 50)
```

codigo_mun	censo	blancos	nulos	candidaturas	part	partido	votos	votos_pc
08093	25	0	0	22	88.00000	ERC	13	59.09091
08195	75	0	0	62	82.66667	JxCAT	34	54.83871
08225	94	1	1	81	88.29787	JxCAT	42	51.21951
08308	139	1	0	108	78.41727	JxCAT	62	56.88073
17007	1763	9	8	1350	77.53829	JxCAT	716	52.68580
17133	1262	7	12	957	77.33756	JxCAT	494	51.24481
25082	292	2	0	195	67.46575	ERC	106	53.80711
25088	96	0	0	73	76.04167	JxCAT	43	58.90411

Summarise (o summarize…)

```
my_data %>% summarise(new_col1 = function(col1), …)
```

¿Qué funciones se pueden usar?

- **sum(), min(), max(), mean(), median()** todas las aritméticas
- **n()** número de combinaciones
- **n_distinct()** número de combinaciones únicas
- **na.rm = T** fundamental cuando queremos evitar los NA en los cálculos

Summarise (o summarize…)

```
my_data %>% summarise(new_col1 = function(col1), …)
```

```
my_data %>%
  distinct( codigo_mun, censo, blancos, nulos, candidaturas) %>%
  summarise( censo = sum(censo),
             candidaturas = sum(candidaturas),
             blancos = sum(blancos),
             nulos = sum(nulos))
```

censo <dbl>	candidaturas <dbl>	blancos <dbl>	nulos <dbl>
37002468	24041001	217227	249487

1 row

Summarise (o summarize…)

```
my_data %>% summarise(new_col1 = function(col1), …)
```

```
my_data %>%
  summarise( censo = sum(censo),
             candidaturas = sum(candidaturas),
             blancos = sum(blancos),
             nulos = sum(nulos)) %>%
  mutate( part = 100 * (candidaturas + blancos + nulos) / censo)
```

censo <dbl>	candidaturas <dbl>	blancos <dbl>	nulos <dbl>	part <dbl>
273192939	177702058	1573816	1764721	66.2684

Group by

```
my_data %>% group_by(col1, col2…)
```

Podemos aplicar funciones sobre nuestro **tibble** separada por combinaciones de claves que definimos en **group_by**

```
# A tibble: 55,521 x 9
# Groups:   partido [16]
   codigo_mun censo blancos nulos
   <chr>      <dbl>   <dbl> <dbl>
 1 01001       2020       5    13
 2 01001       2020       5    13
 3 01001       2020       5    13
 4 01001       2020       5    13
 5 01001       2020       5    13
 6 01001       2020       5    13
 7 01001       2020       5    13
 8 01001       2020       5    13
 9 01002       8014      24    52
10 01002       8014      24    52
# … with 55,511 more rows, and 5 more
#   variables: candidaturas <dbl>,
#   part <dbl>, partido <chr>, votos <dbl>,
#   votos_pc <dbl>
```

Group by

codigo_mun	partido	votos
01001	PNV	381
01002	PNV	1994
01003	PNV	256

group_by(partido)

codigo_mun	partido	votos
01001	Bildu	240
01002	Bildu	1400
01003	Bildu	461

codigo_mun	partido	votos
01001	PSOE	284
01002	PSOE	930
01003	PSOE	26

codigo_mun	partido	votos
01001	PP	114
01002	PP	338
01003	PP	3

codigo_mun	partido	votos
01001	UP	249
01002	UP	866
01003	UP	78

codigo_mun	partido	votos
01001	Cs	22
01002	Cs	27
01003	Cs	0

codigo_mun	partido	votos
01001	VOX	54
01002	VOX	154
01003	VOX	5

codigo_mun	partido	votos
01001	Bildu	240
01001	Cs	22
01001	OTROS	23
01001	PNV	381
01001	PP	114
01001	PSOE	284
01001	UP	249
01001	VOX	54
01002	Bildu	1400
01002	Cs	27
01002	OTROS	53
01002	PNV	1994
01002	PP	338
01002	PSOE	930
01002	UP	866
01002	VOX	154
01003	Bildu	461
01003	Cs	0
01003	OTROS	2
01003	PNV	256
01003	PP	3
01003	PSOE	26
01003	UP	78
01003	VOX	5

Group by + Summarise

```
my_data %>%
   group_by(col1, col2…) %>%
   summarise(new_col1 = function(col3), …)
```

Sirve para aplicar las funciones de **summarise** que hemos visto antes para los grupos que fijemos en el **group_by**.

Group by + Summarise

codigo_mun	partido	votos
01001	Bildu	240
01001	Cs	22
01001	OTROS	23
01001	PNV	381
01001	PP	114
01001	PSOE	284
01001	UP	249
01001	VOX	54
01002	Bildu	1400
01002	Cs	27
01002	OTROS	53
01002	PNV	1994
01002	PP	338
01002	PSOE	930
01002	UP	866
01002	VOX	154
01003	Bildu	461
01003	Cs	0
01003	OTROS	2
01003	PNV	256
01003	PP	3
01003	PSOE	26
01003	UP	78
01003	VOX	5

`group_by``(partido)`

`summarise``(votos=sum(votos))`

codigo_mun	partido	votos
01001	PNV	381
01002	PNV	1994
01003	PNV	256

codigo_mun	partido	votos
01001	Bildu	240
01002	Bildu	1400
01003	Bildu	461

codigo_mun	partido	votos
01001	PSOE	284
01002	PSOE	930
01003	PSOE	26

codigo_mun	partido	votos
01001	PP	114
01002	PP	338
01003	PP	3

codigo_mun	partido	votos
01001	UP	249
01002	UP	866
01003	UP	78

codigo_mun	partido	votos
01001	Cs	22
01002	Cs	27
01003	Cs	0

codigo_mun	partido	votos
01001	VOX	54
01002	VOX	154
01003	VOX	5

partido	votos
PNV	2631
Bildu	2101
PSOE	1240
UP	1193
PP	455
VOX	213
OTROS	78
Cs	49

Group by + Mutate

codigo_mun	partido	votos
01001	Bildu	240
01001	Cs	22
01001	OTROS	23
01001	PNV	381
01001	PP	114
01001	PSOE	284
01001	UP	249
01001	VOX	54
01002	Bildu	1400
01002	Cs	27
01002	OTROS	53
01002	PNV	1994
01002	PP	338
01002	PSOE	930
01002	UP	866
01002	VOX	154
01003	Bildu	461
01003	Cs	0
01003	OTROS	2
01003	PNV	256
01003	PP	3
01003	PSOE	26
01003	UP	78
01003	VOX	5

```
group_by(partido) %>%
   mutate(votos_total=sum(votos)) %>%
   ungroup() %>%
   mutate(votos_prop=100*votos/votos_total)
```

codigo_mun	partido	votos	votos_prop
01001	Bildu	240	11.4231318
01001	Cs	22	44.8979592
01001	PNV	381	14.4811859
01001	PP	114	25.0549451
01001	PSOE	284	22.9032258
01001	UP	249	20.8717519
01001	VOX	54	25.3521127
01002	Bildu	1400	66.6349357
01002	Cs	27	55.1020408
01002	PNV	1994	75.7886735
01002	PP	338	74.2857143
01002	PSOE	930	75.0000000
01002	UP	866	72.5901090
01002	VOX	154	72.3004695
01003	Bildu	461	21.9419324
01003	Cs	0	0.0000000
01003	PNV	256	9.7301406
01003	PP	3	0.6593407
01003	PSOE	26	2.0967742
01003	UP	78	6.5381391
01003	VOX	5	2.3474178

Join

```
my_data %>%
   left_join(my_data_2)
```

Sirve para unir dos tibbles diferentes a partir de claves compartidas. Tiene varios parámetros:

- **tibbles** los tibbles a unir
- **by** las columnas por las que unir
- **suffix** en caso de columnas con iguales nombres

Join

```
my_data %>%
   xxxxx_join(my_data_2)
```

Hay diferentes tipos de join:

- inner_join combinaciones que estén en **ambos** tibble
- left_join combinaciones que estén en el **primer** tibble
- right_join combinaciones que estén en el **segundo** tibble
- full_join **todas** las combinaciones
- anti_join combinaciones que estén en el primero y no en el segundo

Join

tibble_1	
key	**val_1**
A	2
B	4
C	6
D	8

tibble_2	
key	**val_2**
A	1
C	3
E	5
F	7

Join

```
tibble_1 %>%
   inner_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2

Join

```
tibble_1 %>%
   inner_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2
A	2	1
C	6	3

Join

```
tibble_1 %>%
   left_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2

Join

```
tibble_1 %>%
   left_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2
A	2	1
B	4	NA
C	6	3
D	8	NA

Join

```
tibble_1 %>%
   right_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2

Join

```
tibble_1 %>%
   right_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2
A	2	1
C	6	3
E	NA	5
F	NA	7

Join

```
tibble_1 %>%
   full_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2

Join

```
tibble_1 %>%
   full_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2
A	2	1
B	4	NA
C	6	3
D	8	NA
E	NA	5
F	NA	7

Join

```
tibble_1 %>%
   right_join(tibble_2)
```

tibble_1

key	val_1
A	2
B	4
C	6
D	8

tibble_2

key	val_2
A	1
C	3
E	5
F	7

tibble_3

key	val_1	val_2

Join

!Importante! Los parámetros **by** y **suffix**

by: Para fijar las columnas por las que unen las dos tablas. Hay varios casos:

- Solo una columna que se llama igual: **by = “col1”**
- Varias columnas que se llaman igual: **by = c(“col1”, “col2”, ...)**
- Una columna con nombres distintos: by = **c(“col1_t1” = “col1_t2”)**
- Varias columnas con nombres distintos:

  by = c(“col1_t1” = “col1_t2”, “col2_t1” = “col2_t2”, ...)

Join

suffix: Cuando unimos por algunas claves y quedan columnas que se llaman igual en ambos tibbles.

tibble_1

key_1	key_2	val
A	B	1
B	C	3
C	D	5
E	F	7

tibble_2

key_1	key_2	val
A	B	2
B	C	4
C	D	6
E	F	8

Join

```
tibble_1 %>%
  left_join( tibble_2, by = c( "key_1", "key_2"))
```

tibble_3

key_1	key_2	val.x	val.y
A	B	1	2
B	C	3	4
C	D	5	6
E	F	7	8

```
tibble_1 %>%
  left_join( tibble_2, by = c( "key_1", "key_2"),
             suffix = c( "_t1", "_t2"))
```

tibble_3

key_1	key_2	val_t1	val_t2
A	B	1	2
B	C	3	4
C	D	5	6
E	F	7	8