

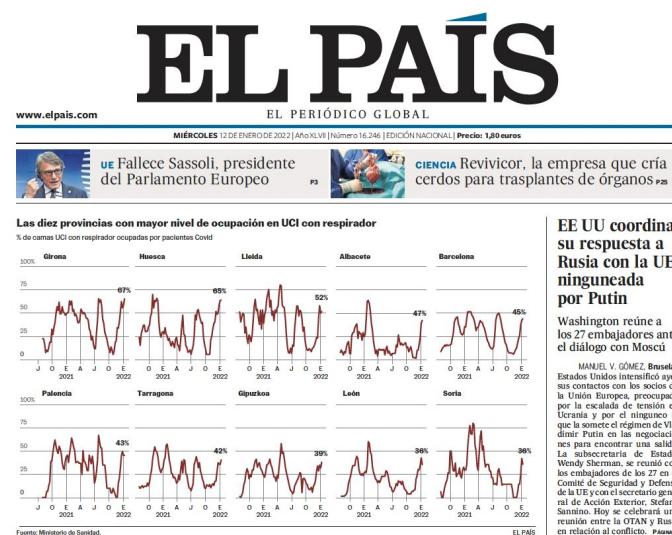
# Periodismo de datos: visualización y análisis electoral en R

Borja Andrino Túron  
Analista de datos de EL PAÍS

 [borja.andrino@gmail.com](mailto:borja.andrino@gmail.com)  
@borjandrino

# ¿R en una redacción?

# Incluso donde menos lo podías esperar



## 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 clara ascenso**

**La OMS prevé que la mitad**

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 respiradores. Aunque hace un año, pese a las tendencias positivas, una UCI estaba llena de covid, lo que no ocurre en el nivel actual.

Es la mitad que hace que se observa una clara tendencia al alza. En cuatro provincias, cada cuatro camas de hospitalizada están ocupada por un paciente y esto significa que están en riesgo considerado

o Girona  
de crí-  
tico  
invierno  
6. Cana-  
l. Aragón  
las peo-  
n en ca-  
mas convencionales desde el in-  
cio de la pandemia. La Organiza-  
ción Mundial de la Salud (OMS) ha  
aviso ayer de que en las próximas  
seis u ocho semanas la mitad  
de la población europea se contagie  
con la ómicron. PÁGINAS 22 A

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 han en este país habido, hace unos días, unas violentas revueltas que han puesto en jaque al Estado. La primera ciudad que uno se encuentra

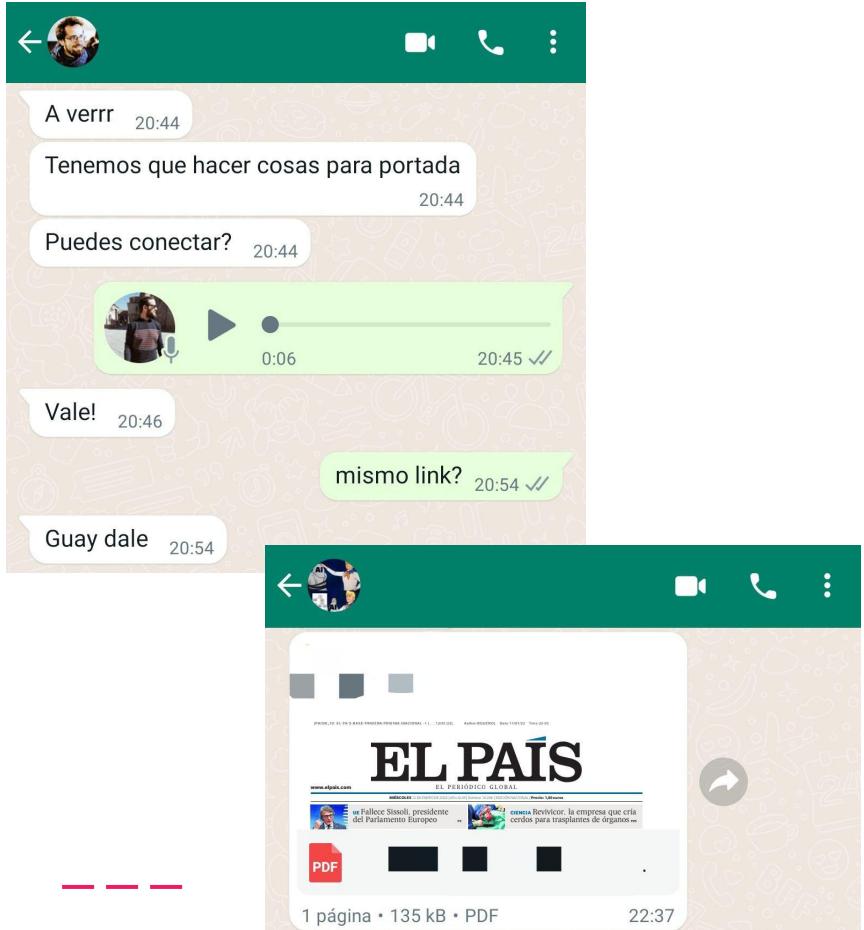


**La caída del poder adquisitivo de los salarios agrava la desigualdad**

A. SÁNCHEZ / I. FARIZA, **Madrid**  
Millones de asalariados lidian con España con la pérdida de poder adquisitivo. Tras años de precios altazagados, el abrupto despegar de la inflación ahonda en 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 10**

# ¿Por qué?

Porque es muy rápido y ágil  
(no solo R)



# ¿Qué vamos a ver hoy?

---

## Tidyverse

- ¿Por qué R (y tidyverse)?
- Funciones para leer y escribir
- Selección y creación de columnas
- Filtros
- Summarise (o summarize)
- Group by
- Join
- Pivots

## Ggplot2

- Sintaxis
- Tipos de gráficos
- Escalas
- Colores
- Facets

# Análisis electoral como hilo conductor



ELECCIONES GENERALES > ANÁLISIS i

## *El resultado de Vox y otras cinco sorpresas posibles, según las encuestas del 10-N*

Analizamos las incógnitas de las elecciones: el auge del partido de Abascal (o su pinchazo), la resistencia de Ciudadanos y el ascenso imprevisto de la izquierda

ELECCIONES EN LA COMUNIDAD DE MADRID

## **Así son los vecindarios más movilizados del cinturón rojo de Madrid: más renta y tendencia a la derecha**

ctoria en las autonómicas del 4-M a una subida de barrios obreros. Estas son las calles con más voto

ELECCIONES GENERALES

## **Los graneros de Vox: el voto a la ultraderecha se concentra en los municipios con más inmigración**

El PAÍS analiza el peso de la población foránea en el auge del partido de Abascal. El fenómeno parece evidente en Andalucía, Murcia, Extremadura o Alicante

En 4 horitas 😊



# Antes de empezar

Otras utilidades

Proyectos en R



# Proyectos y buenas prácticas

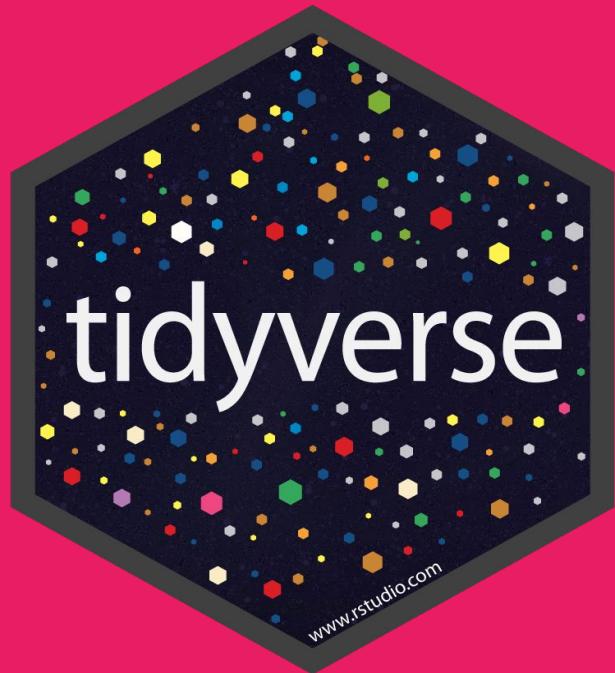
---

- Permiten tener organizado nuestro trabajo
- Organiza tus datos en **carpetas**
- Nombra tus scripts con nombres **representativos**

Para crearlo solo tienes que clickar en File -> New Project

Vamos a crear uno para nuestra clase

<input type="checkbox"/>	 01-prepare_medidores_trafico_...	2 KB
<input type="checkbox"/>	 02-process_historico_intensida...	3.8 KB
<input type="checkbox"/>	 02b-process_historico_carga.R	5.2 KB
<input type="checkbox"/>	 analisis-01-historico.nb.html	1.4 MB
<input type="checkbox"/>	 analisis-01-historico.Rmd	10 KB
<input type="checkbox"/>	 analisis-02-carga.nb.html	2 MB
<input type="checkbox"/>	 analisis-02-carga.Rmd	15.7 KB
<input type="checkbox"/>	 analisis-03-encuesta_crtm.nb....	777 KB
<input type="checkbox"/>	 analisis-03-encuesta_crtm.Rmd	2.9 KB
<input type="checkbox"/>	 analisis-xx-otras_fuentes.nb.ht...	2 MB
<input type="checkbox"/>	 analisis-xx-otras_fuentes.Rmd	14.1 KB
<input type="checkbox"/>	 aux-explore_medidores.R	1.6 KB
<input type="checkbox"/>	 data	
<input type="checkbox"/>	 data_raw	
<input type="checkbox"/>	 nohup.out	4.8 KB
<input type="checkbox"/>	 tema_movilidad.Rproj	205 B





# Enough of this sheet

# Pipes

---

```
my_data %>%  
  selecciono(...) %>%  
  filtro(...) %>%  
  agrupo(...) %>%  
  resumo(...) %>%  
  pinto(...)
```



# ¿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()`)

# Selección y creación de columnas

---

**Problema 1** Selecciona todas las columnas excepto el porcentaje de votos.

**Problema 2** Selecciona el código de municipio, el partido y las variables relacionadas con el voto

# 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"))
```

# Mutate

---

**Problema 3** Crea una columna en la que se sume el número de votos a candidaturas y el número de votos en blanco.

**Problema 4** Obtén el municipio con mayor porcentaje de votos nulos de España. ¿Y el que tiene mayor porcentaje de blancos?

# 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

# Filtros

---

**Problema 5** Obtén los 10 municipios en los que más se vota a Vox.

**Problema 6** Obtén los 10 municipios de menos de 50.000 habitantes de la provincia de Madrid donde Vox obtiene un porcentaje de voto más alto.

**Problema 7** Obtén los 10 municipios de menos de 50.000 habitantes de la provincia de Madrid donde Vox obtiene un porcentaje de voto más alto.

# Filtros

---

**Problema 5** Obtén los 10 municipios en los que más se vota a Vox.

**Problema 6** Obtén los 10 municipios con menos de 50.000 votantes de la provincia de Madrid donde Vox obtiene un porcentaje de voto más alto.

**Problema 7** Obtén los 10 municipios de menos de 50.000 habitantes de la provincia de Madrid donde Vox obtiene un porcentaje de voto más alto.

# 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	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)

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

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

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

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

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

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

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

# 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	PP	114
01002	PP	338
01003	PP	3

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

#THEMASKEDSINGERBR

Group by + Mutate



# 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

# Group by

---

**Problema 8** Calcula los votos totales de cada partido

**Problema 9** Calcula la participación en cada provincia

**Problema 10** Calcula los votos de cada partido según si el tamaño del municipio tiene más o menos de 50.000 habitantes

# 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 %>%  
  xxxx_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)
```

<b>tibble_1</b>	
<b>key</b>	<b>val_1</b>
A	2
B	4
C	6
D	8

<b>tibble_2</b>	
<b>key</b>	<b>val_2</b>
A	1
C	3
E	5
F	7

<b>tibble_3</b>		
<b>key</b>	<b>val_1</b>	<b>val_2</b>

# Join

---

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

<b>tibble_1</b>	
<b>key</b>	<b>val_1</b>
A	2
B	4
C	6
D	8

<b>tibble_2</b>	
<b>key</b>	<b>val_2</b>
A	1
C	3
E	5
F	7

<b>tibble_3</b>		
<b>key</b>	<b>val_1</b>	<b>val_2</b>
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

---

```
tibble_1 %>%  
  anti_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
B	4
D	8

# 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

# Join

---

**Problema 11** Calcula los votos de cada partido en el municipio de Fuenlabrada

**Problema 12** Calcula los votos de cada partido en los municipios de San Juan del Molinillo y Navarredondilla

**Problema 13** Calcula el top 10 de municipios donde más cayó la participación

# Pivots

-----  
Formato **long**

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

Formato **wide**

codigo_mun	Bildu	Cs	OTROS	PNV	PP	PSOE	UP	VOX
01001	240	22	23	381	114	284	249	54

# Pivots

---

```
my_data %>%  
  pivot_longer(cols, names_to = “name”, values_to = “value”)
```

```
my_data %>%  
  pivot_wider(id_cols, names_from = “name”, values_from = “value”)
```

## Formato **long**

# Pivots

---

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

```
pivot_longer(Bildu:VOX,  
             names_to = "partido",  
             values_to = "votos")
```

```
pivot_wider(id_cols = "codigo_mun",  
            names_from = "partido",  
            values_from = "votos")
```

## Formato **wide**

codigo_mun	Bildu	Cs	OTROS	PNV	PP	PSOE	UP	VOX
01001	240	22	23	381	114	284	249	54

# Pivots

---

## Problemas habituales

codigo_mun	votos_pc	Bildu	Cs	OTROS	PNV	PP	PSOE	UP	VOX	MP	CUP	ERC	JxCAT	BNG	CC	PRC	Teruel
01001	17.4927114	240	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	1.6034985	NA	22	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	1.6763848	NA	NA	23	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	27.7696793	NA	NA	NA	381	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	8.3090379	NA	NA	NA	NA	114	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	20.6997085	NA	NA	NA	NA	NA	284	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	18.1486880	NA	NA	NA	NA	NA	NA	249	NA	NA	NA	NA	NA	NA	NA	NA	NA
01001	3.9358601	NA	NA	NA	NA	NA	NA	NA	NA	54	NA	NA	NA	NA	NA	NA	NA

# Pivots

---

## Problemas habituales

Values are not uniquely identified; output will contain list-cols.

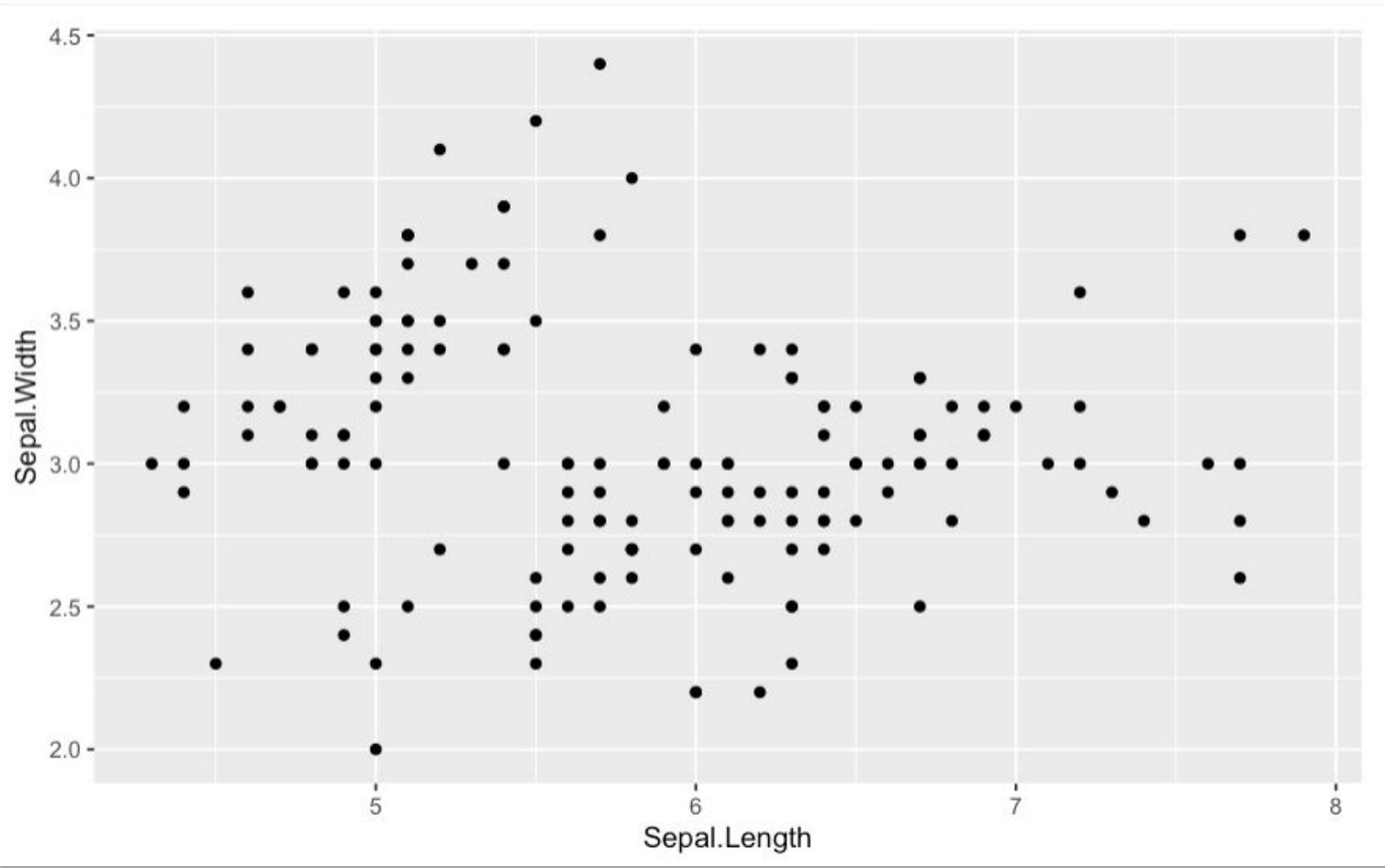
- \* Use `values\_fn = list` to suppress this warning.
- \* Use `values\_fn = length` to identify where the duplicates arise
- \* Use `values\_fn = {summary\_fun}` to summarise duplicates>

codigo_mun	Bildu	Cs	OTROS	PNV	PP	PSOE	UP	VOX
01001	c(240, 214)	c(22, 56)	c(23, 30)	c(381, 394)	c(114, 100)	c(284, 304)	c(249, 276)	c(54, 41)



-----





**Andalucía**  
...un 18% del total



**C. La Mancha**  
...un 13% del total



**Navarra**  
...un 14% del total



**País Vasco**  
...un 14% del total



**La Rioja**  
...un 17% del total



**Murcia**  
...un 15% del total



# Sintaxis

---

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

# Funciones

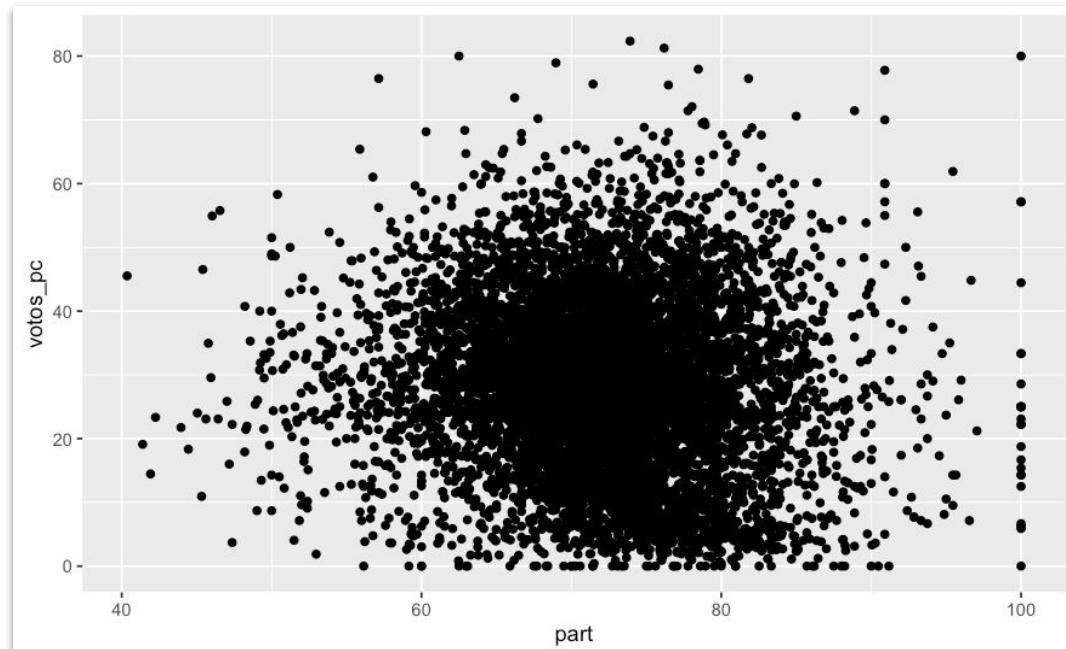
---

- `geom_point`
- `geom_histogram`
- `geom_density`
- `geom_col`
- `geom_line`
- `geom_step`
- `geom_area`
- `geom_boxplot`

# Funciones

---

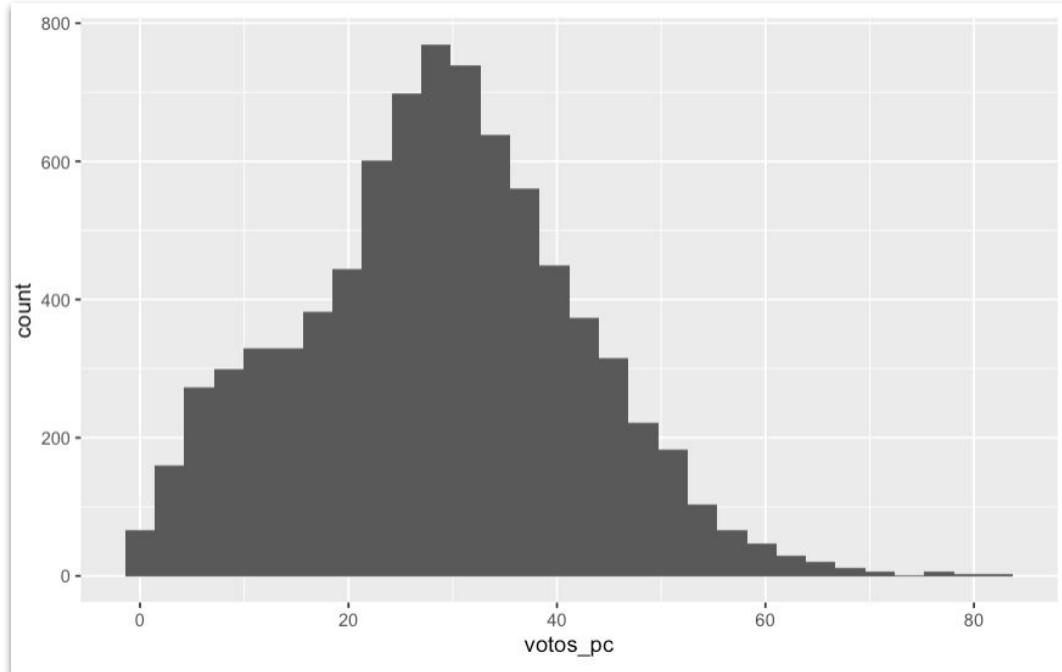
```
my_data %>%
  filter( partido == "PSOE") %>%
  ggplot() +
  geom_point(aes( x = part, y = votos_pc))
```



# Funciones

---

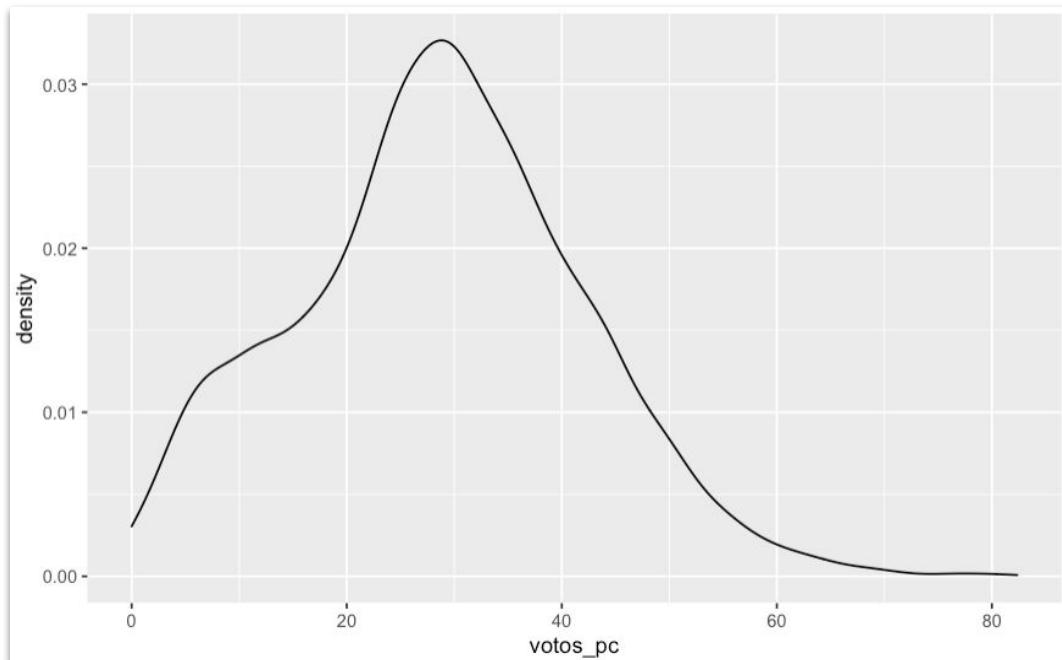
```
my_data %>%
  filter( partido == "PSOE") %>%
  ggplot() +
  geom_histogram(aes( x = votos_pc))
```



# Funciones

---

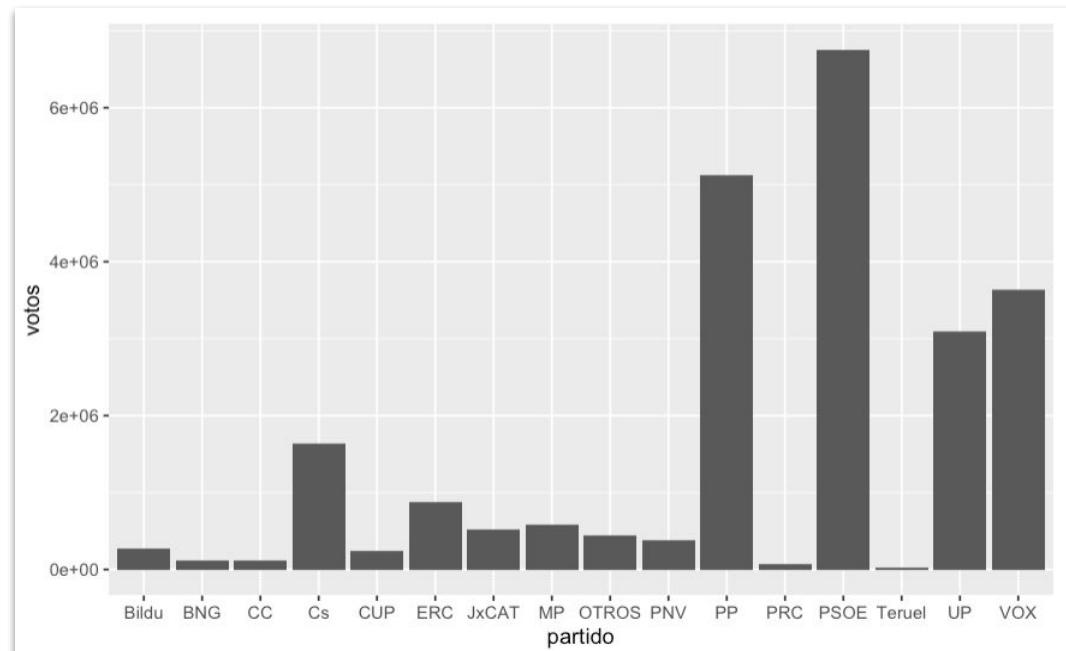
```
my_data %>%
  filter( partido == "PSOE") %>%
  ggplot() +
  geom_density(aes( x = votos_pc))
```



# Funciones

---

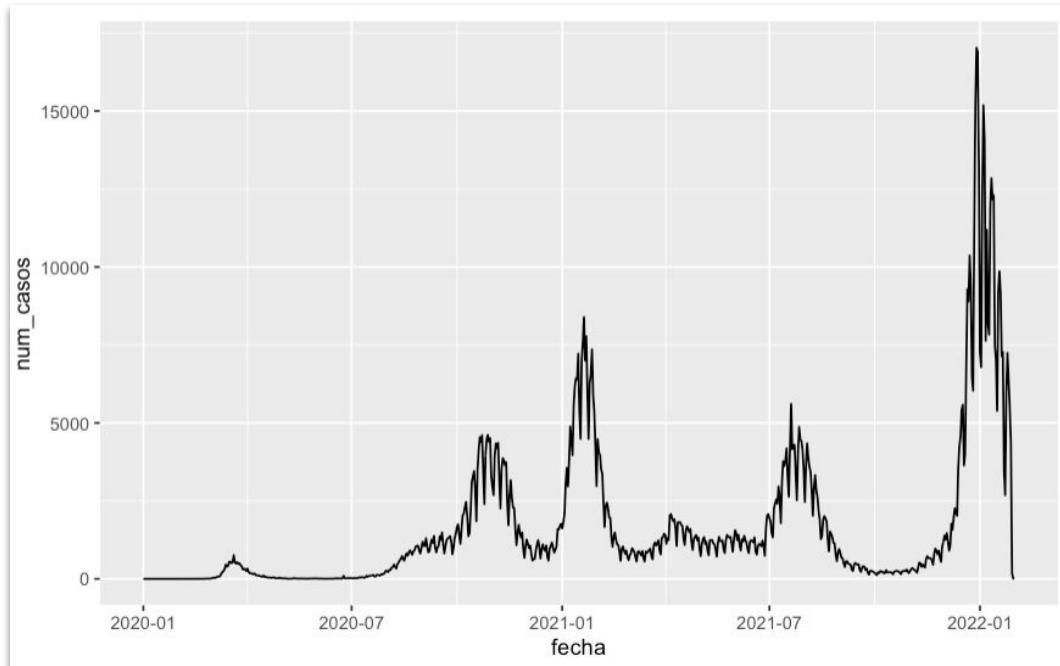
```
my_data %>%
  group_by( partido ) %>%
  summarise( votos = sum(votos)) %>%
  ggplot() +
  geom_col( aes( x = partido, y = votos))
```



# Funciones

---

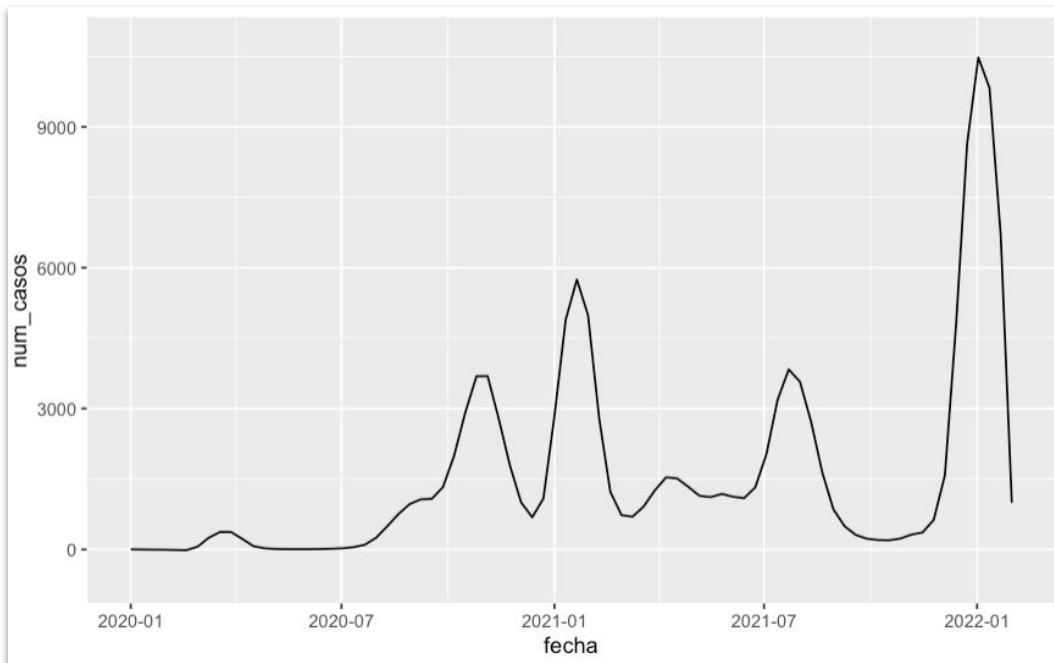
```
datos_covid %>%
  filter(ccaa_iso == "AN") %>%
  ggplot() +
  geom_line(aes(x = fecha,y = num_casos))
```



# Funciones

---

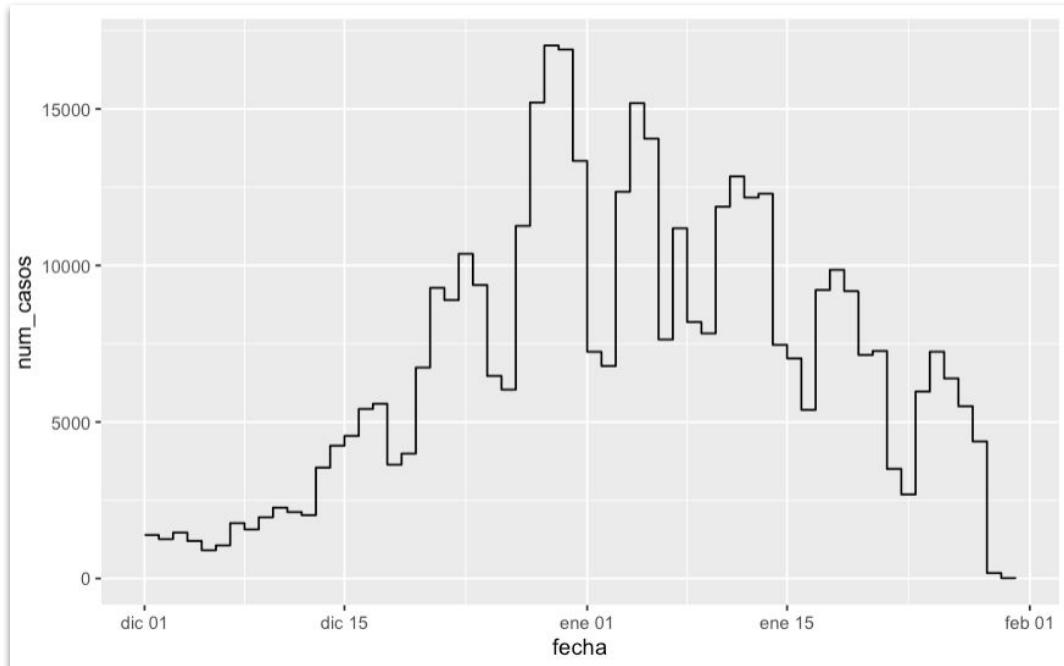
```
datos_covid %>%
  filter(ccaa_iso == "AN") %>%
  ggplot() +
  geom_line(aes(x = fecha, y = num_casos),
            stat = "smooth",
            method = "loess", span = 0.1)
````
```



# Funciones

---

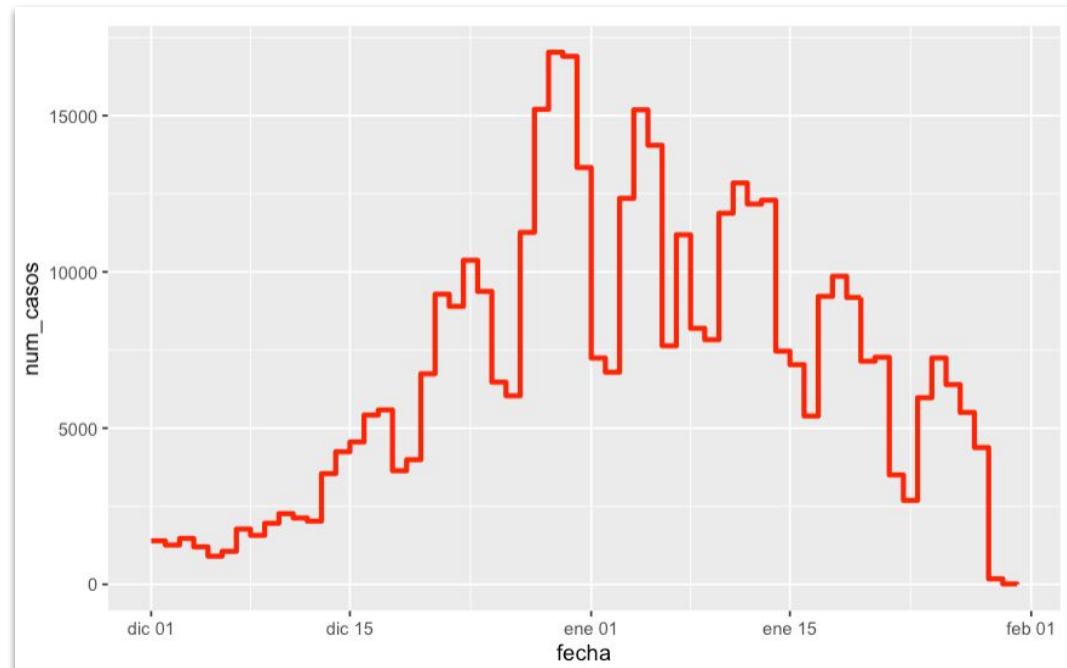
```
datos_covid %>%  
  filter(ccaa_iso == "AN",  
         fecha >= ymd( "2021-12-01")) %>%  
  ggplot() +  
  geom_step(aes(x = fecha,y = num_casos))
```



# Funciones

---

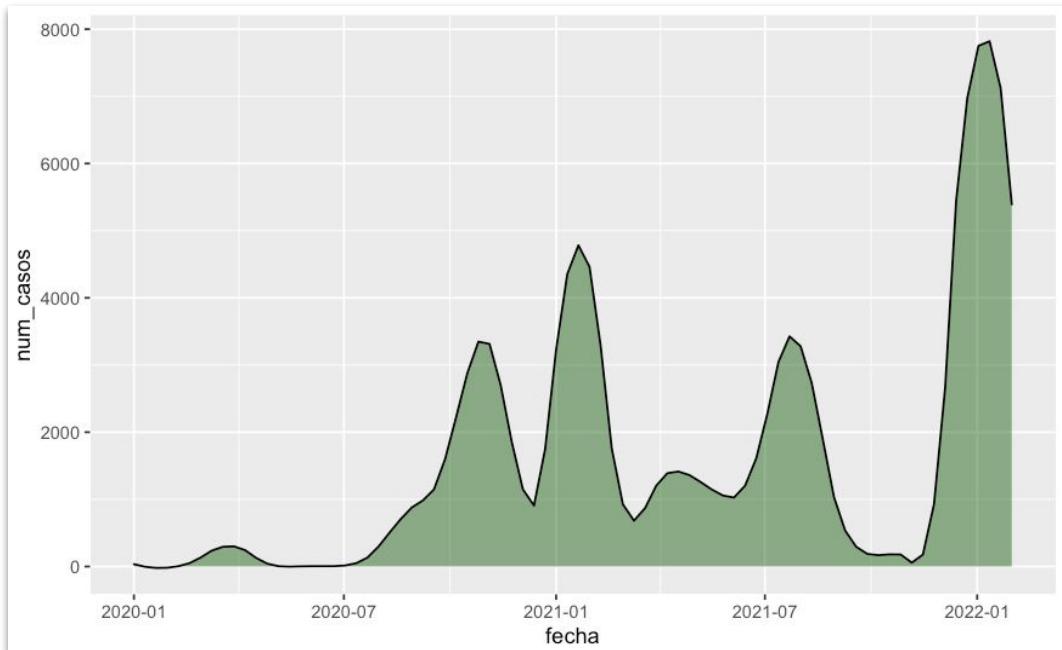
```
datos_covid %>%
  filter(ccaa_iso == "AN",
        fecha >= ymd( "2021-12-01")) %>%
  ggplot() +
  geom_step(aes(x = fecha,y = num_casos),
            color = "red",
            size = 1.2)
```



# Funciones

---

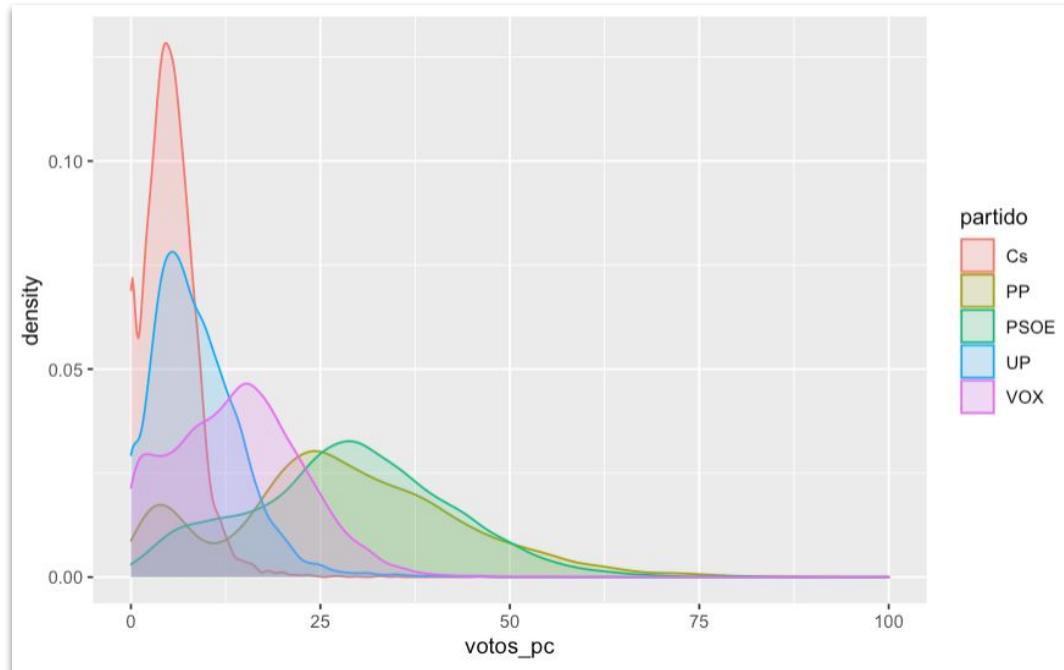
```
datos_covid %>%  
  filter(ccaa_iso == "AN") %>%  
  ggplot() +  
  geom_area( aes(x = fecha,y = num_casos),  
    stat = "smooth",  
    method = "loess",  
    span = 0.15,  
    fill = "darkgreen",  
    alpha = 0.5,  
    color = "black")
```



# Aesthetics

---

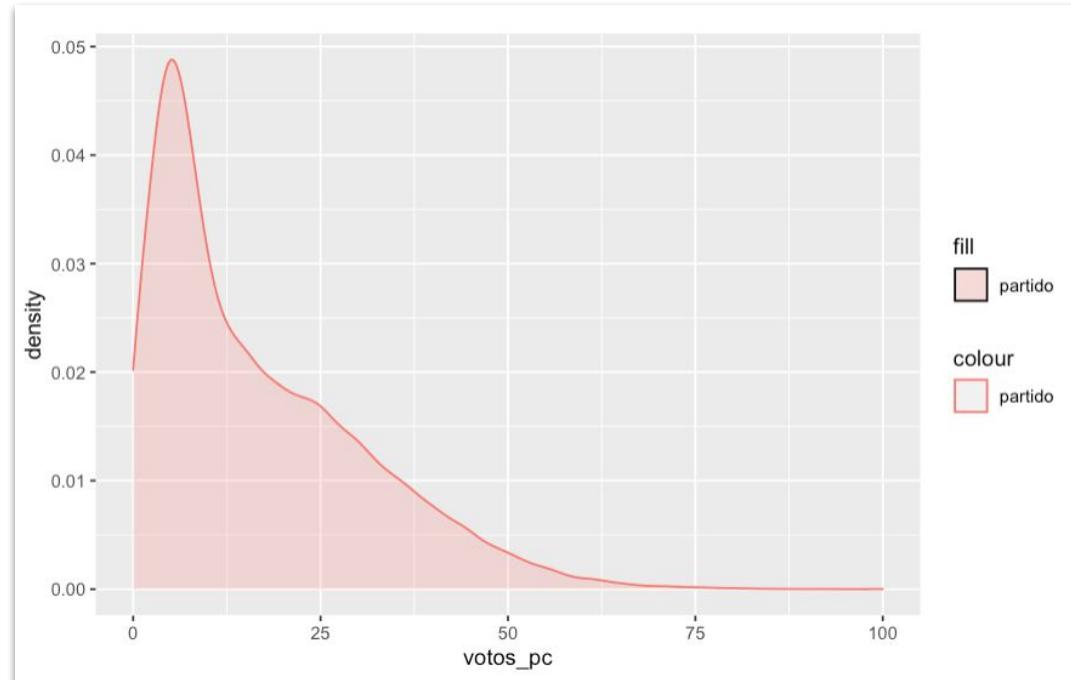
```
partidos_nac <- c( "PSOE", "PP", "VOX", "UP", "Cs")  
  
my_data %>%  
  filter( partido %in% partidos_nac) %>%  
  ggplot() +  
  geom_density(aes( x = votos_pc,  
                  color = partido,  
                  fill = partido),  
               size = 0.5,  
               alpha = 0.2)
```



# Aesthetics

---

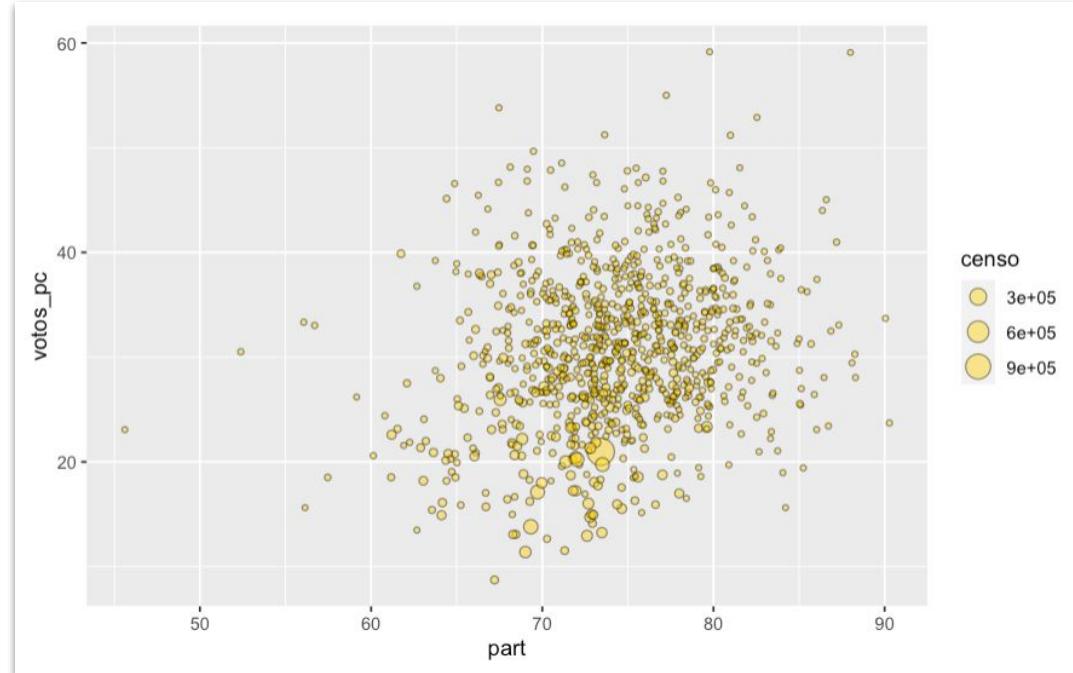
```
my_data %>%
  filter( partido %in% partidos_nac) %>%
  ggplot() +
  geom_density(aes( x = votos_pc,
                    color = "partido",
                    fill = "partido"),
               size = 0.5,
               alpha = 0.2)
```



# Aesthetics

---

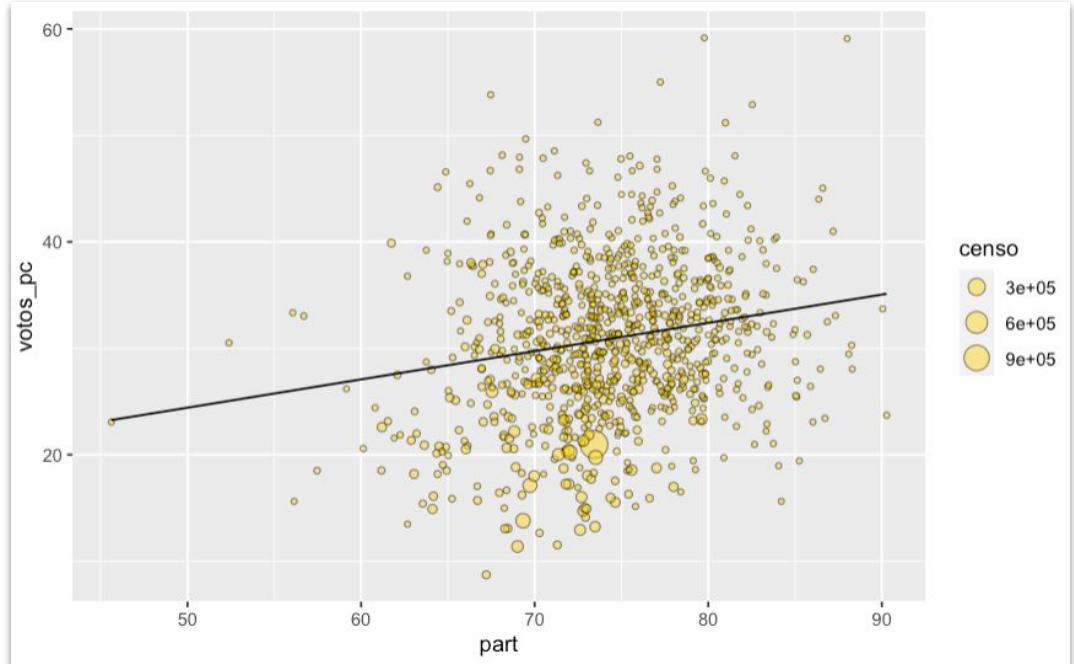
```
my_data %>%
  filter( partido == "ERC") %>%
  ggplot() +
  geom_point(aes( x = part, y = votos_pc,
                  size = censo),
              alpha = 0.5,
              fill = "gold",
              shape = 21)
```



# Local vs. global

---

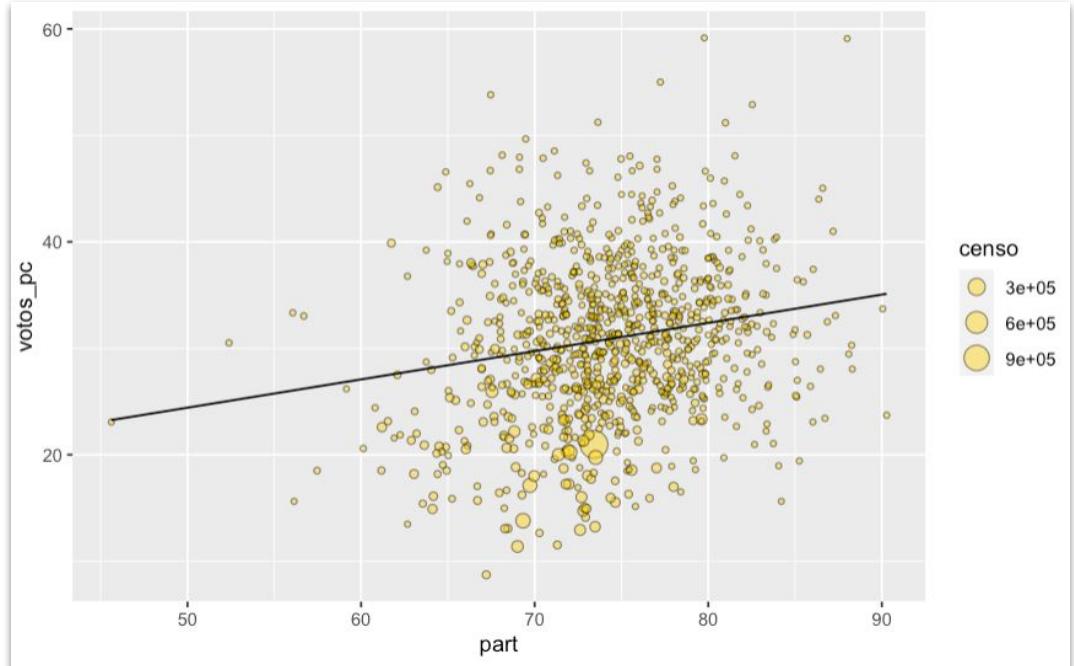
```
my_data %>%
  filter( partido == "ERC") %>%
  ggplot() +
  geom_point(aes( x = part, y = votos_pc,
                  size = censo),
              alpha = 0.5,
              fill = "gold",
              shape = 21) +
  geom_line(aes( x = part, y = votos_pc),
            stat="smooth",
            method = "lm")
```



# Local vs. global

---

```
my_data %>%
  filter( partido == "ERC") %>%
  ggplot(aes( x = part, y = votos_pc,
             size = censo)) +
  geom_point(alpha = 0.5,
             fill = "gold",
             shape = 21) +
  geom_line(stat="smooth",
            method = "lm")
```



# Colores y escalas

---

- `scale_color_manual()`
- `scale_color_continuous()`
- `scale_x/y_continuous()`
- `scale_x/y_log10()`

# Temas

---

Añádelos como una línea más a tus gráficos para cambiarlos completamente.

- `theme_gray()`
- `theme_light()`
- `theme_dark()`
- `theme_minimal()`
- [Todos los temas](#)

# Facets

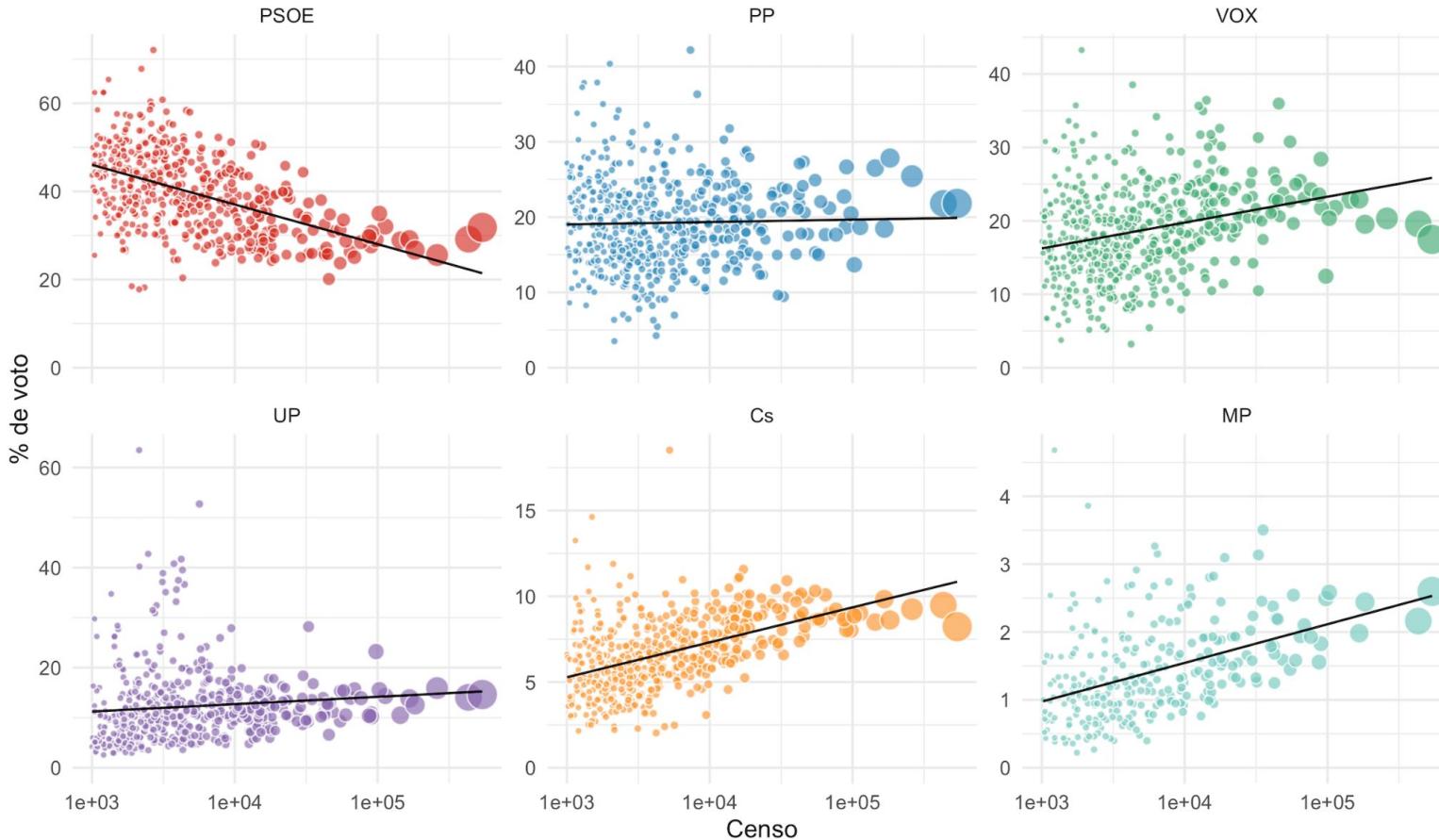
---

Con la función **facet\_wrap()** podemos replicar un gráfico separándolo por una variable concreta. Sus parámetros son:

- **~facet\_var** variable por la que separar
- **ncol** número de columnas del gráfico
- **scale** “free”, “free\_x”, “free\_y”

# ¿Partidos rurales y partidos urbanos en Andalucía?

Relación entre censo y voto el 10N



Fuente: Ministerio del Interior

# ggplot2 (ejercicios para la última hora)

---

**Problema 14** Representa la evolución de los contagios por cada 100.000 habitantes en cada comunidad autónoma desde el 1 de diciembre en un solo gráfico

**Problema 15** Representa la relación entre el porcentaje de población extranjera y el de voto a Vox en cada comunidad autónoma