uc3m | Universidad Carlos III de Madrid

Talleres de Análisis Político I

Sesión 3

20/11/2023

Pau Vall-Prat

pau.vall@uc3m.es

La semana pasada

Dplyr

- Filter
- Select
- Mutate
  - if_else()
  - case_when()
- Arrange

+ condiciones lógicas

Continuamos!

group_by

- Aunque no tiene como consecuencia directa ninguna transformación visible en el marco de datos, group_by() permite agrupar todas las observaciones a partir de los valores en una variable concreta
- Es importante, una vez finalizada la agrupación, indicar que se quiere desagrupar, de otro modo los datos seguirán agrupados “invisiblemente”
  - Para eso existe la función ungroup()
- Es importante para obtener resúmenes de la información.

Ejercicio

- Agrupa tu base de datos en base a la provincia
- Agrupa la base de datos por CCAA
- Agrupa la base de datos por número de mesas dentro de cada CA

summarise

- Permite obtener valores resumen del marco de datos
- Nos permite cambiar el nivel de observación a partir de una variable de agrupación
- Es importante tener en cuenta la existencia o no de datos perdidos en la variable
- Summarise siempre requiere “crear nuevas variables”
- Por ejemplo, el valor de media de una variable en distintos grupos

```
df |> group_by(partido) |> summarise( m_alt = mean(altura))
```

Principales funciones en summarise

- Suma: sum()
- Media: mean()
  - Desviación estándar: sd()
- Mediana: median()
- Valor máximo: max()
- Valor mínimo: min()
- Número de casos: n()

- Recuerda añadir el argumento `na.rm = T` si es necesario

Ejercicios

1. Agrupa los resultados electorales por provincias → Los resultados deberían ser parecidos a los resultados oficiales

https://resultados.elpais.com/elecciones/2019/generales/congreso/03/

2. Indica cuantos municipios dentro de cada CA tienen 1 mesa electoral, cuantos 2 mesas, etc.
3. Agrupa los resultados por tipología de municipio (en base a la población: rural, semi-urbano, urbano)
   a. Obtén el total de censo y votantes
   b. Calcula la participación y los % de apoyo a los partidos

Más dplyr

Más mutate

- Podemos aplicar un cambio a más de una variable a la vez usando
  - mutate_if()
  - mutate_at()
  - mutate_all()
  - across()

mutate_if

- Se hacen cambios en múltiples variables que cumplan con una característica especificada
- Es especialmente útil para transformar variables de un tipo a otro
- Por ejemplo,
  - Estandardizar ($\bar{x}$=0, sd=1) todas las variables numéricas

    df |> mutate_if(is.numeric,scale)
  - Convertir variables de tipo factor a carácter

    df |> mutate_if(is.factor, as.character)

mutate_at

- Cambiar diferentes variables especificadas en un vector específico
  - Estandardizar una variable
  - df |> mutate_at (c("n", "edad"), scale)
  - Convertir una escala logarítmica
  - df |> mutate_at (c(“pib", “poblacion"), log)

mutate_all

- Nos permite hacer cambios en todas las variables de un marco de datos

```
df |> mutate_all(as.character)
```

across

- Permite hacer cambios a múltiples variables dentro de la función básica mutate

```
df |> mutate(across(c(n, edad), round))
df |> mutate(across(where(is.numeric), round))
df |> mutate(across(starts_with(“nombr"), tolower))
```

Ejercicios

- Crea un marco de datos con todos los valores numéricos estandardizados
- Crea las variables de apoyo relativo (%) a PP y PSOE y conviértelas en una escala estandardizada
- Convierte la variable población a logaritmo de población
- Crea un marco de datos con todas las variables en formato carácter
- Estandardiza solo las variables que empiezen por “vot”

Más gestión de datos

Semana que viene: factores, cambiar estructura de base de datos y combinar bases de datos

Ejercicio 2 – Lluís Orriols

¡Vamos a poner en práctica lo que hemos visto hasta ahora!