uc3m | Universidad Carlos III de Madrid

Talleres de Análisis Político I

Sesión 4

27/11/2023

Pau Vall-Prat

pau.vall@uc3m.es

Más dplyr

Pendiente de la semana pasada

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”

Factores

¿Qué es un factor?

- Usamos variables de tipo factor en variables de tipo categórico
  - Menos importante en variables nominales
  - Crucial en variables ordinales
- Un factor es una variable que tiene
  - Valores numéricos (levels)
  - Valores de texto, o etiquetas (labels)
- Los valores numéricos
  - Están ordenados
  - El orden se usa en la representación gráfica/en tablas
  - Es relevante a la hora de hacer regresiones

Crear un factor

- Cualquier variable o vector numérico o de carácter se puede convertir en factor
  - factor
  - as.factor
  - as_factor
- Por defecto usan el orden de menor a menor o alfabético para ordenar las categorías
- Puede dar problemas si categorizamos una variable numérica que está en formato carácter
- Importante distinguir levels del valor numérico

```
> nombres <- as.factor(c("Jose", "María", "María"))
> nombres
[1] Jose  María María
Levels: Jose María
>
```

```
> as.numeric(nombres)
[1] 1 2 2
>
```

```
> numeros <- as.factor(c("2", "2", "3"))
> numeros
[1] 2 2 3
Levels: 2 3
> as.numeric(numeros)
[1] 1 1 2
>
```

Si quieres convertir un número que está codificado en un factor debes primero eliminar las etiquetas con as.character

```
> as.numeric(as.character(numeros))
[1] 2 2 3
>
```

Paquete forcats

- Nos ayuda a gestionar los factores
- En general es mejor
- fct()
  - Convierte variables de tipo carácter en un factor
  - Nos obliga a usar el argumento `levels`, para indicar el orden de las categorías
    - Hay que especificar todas las categorías
    - Si no lo hacemos ordena en el orden de aparición en los datos
  - Alternativamente, hay que especificar las categorías que convertiremos en NA con el argumento `na`

fct_relevel

- Permite ordenar los *levels*, niveles, de cada factor de acuerdo con un orden que se defina en el momento
  - Se puede indicar solo uno de los niveles, que irá el primero y mantener el orden del resto

    df <-df |> mutate(var1 = fct_relevel(v1, c("a")))
  - Se pueden indicar todos los niveles en el orden deseado

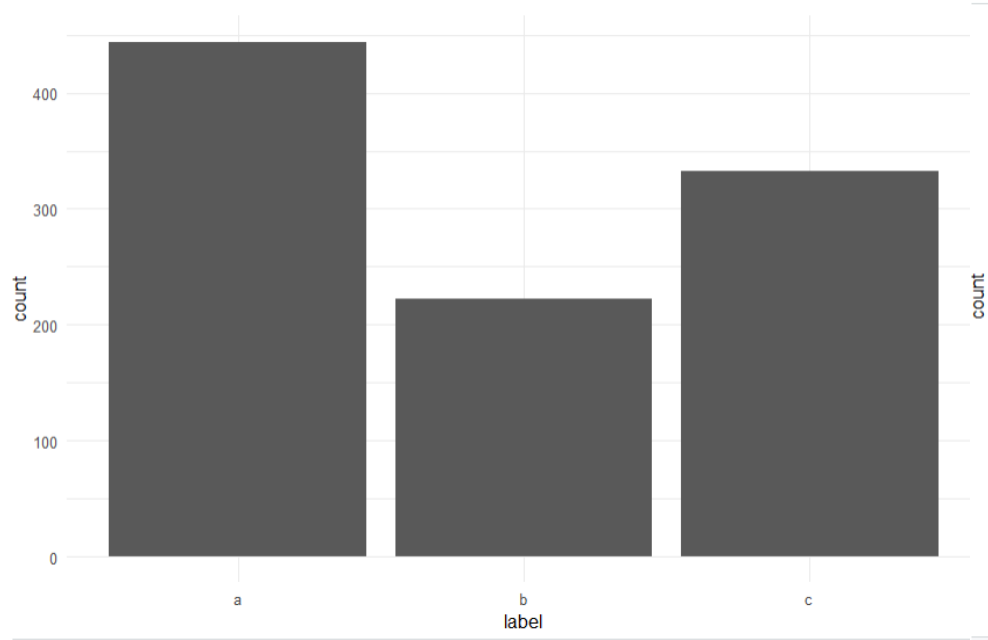
    df <-df |> mutate(var1 = fct_relevel(v1, c("a“, “j”, “c”)))

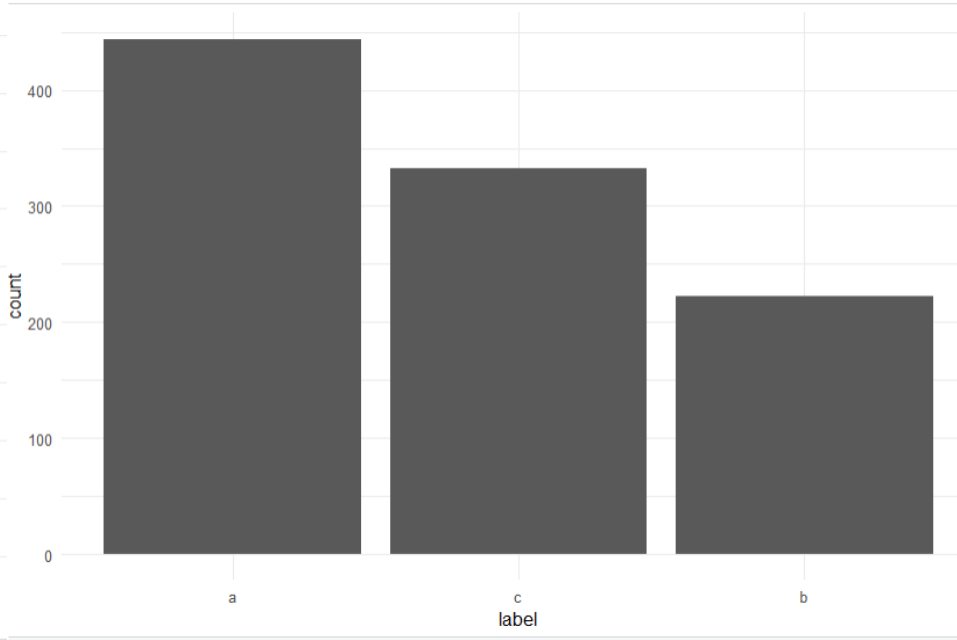
fct_infreq

- Reordenar un factor según la frecuencia de si mismo

```
df2 <- df |> mutate(var1 = fct_infreq(var1))
```

- Es habitual cuando se quiere graficar algo por orden





fct_reorder

- Reordenar un factor según la frecuencia de otro
  - df2 <- df |> mutate(var1 = fct_reorder(var1, var2))

fct_lump

- Disminuir el número de categorías
  - Convierte las categorías menos frecuentes en la categoría “otros”
  - Argumento `n` para indicar número de categorías: las más frecuentes
  - `prop` para conservar los que representen más de x %

    df2 <- df |> mutate(var1 = fct_lump(var1, n=3))

fct_recode

- Permite cambiar los nombres de los *levels* de una variable factor
  - Df |> mutate(var1 = fct_recode(var1, PP = "Partido Popular", PSOE = "Partido Socialista Obrero Español")

Ejercicio

- Crea una variable numérica con if_else
  - Conviértela en factor
  - Cámbiale las etiquetas/nombres a los levels
- Crea una variable de tipo *carácter* con case_when
  - Conviértela en factor
  - Cambia el orden en el que aparecen los levels

Más gestión de datos

Combinar bases de datos (1)

- Es habitual en R trabajar con más de un marco de datos a la vez
- Para unir estos distintos marcos de datos se usan unas funciones específicas del paquete dplyr …_join()
- Para que funcionen, debe haber una (o más) variables que sean comunes en ambos marcos de datos
  - Conocidas como claves
  - *Primary key* vs. *Foreign key*

Combinar bases de datos (2)

- Es recomendable trabajar con marcos de datos donde *foreign key* identifique la unidad de observación de la base de datos
- El objetivo final es tener una base de datos ampliada con un número mayor de variables
- Funciones:
  - left_join(), inner_join(), right_join(), full_join()
  - Varían en función de qué observaciones conservan

…_join()

- Principales argumentos
  - x = : indica una de las bases de datos
  - y = : indica la otra base de datos
  - by = : indica la(s) variable(s) que permiten unir las bases de datos
  - keep = : T o F, para indicar si se quieren preservar la(s) variable(s) especificadas en el argumento by

- Si no se especifica by = , R detecta las variables comunes entre marcos de datos

left_join()

- Seguramente, la más usada
- Añade la información de los datos *y* al marco de datos *x*
- Se mantienen todas las observaciones de x
  - Solo se trae información de las variables en *y* para las que se encuentra un *match* en x

right_join()

- Similar a left_join
- Añade información de los datos x al marco de datos y
- Se mantienen las observaciones de y

inner_join()

- Combina toda la información de x e y
- Se mantienen solo las observaciones de x e y que tienen match en ambos marcos de datos

full_join()

- Combina toda la información de x e y
- Se mantienen TODAS las observaciones de x e y, aunque no haya *match*

Ejercicios

- Combinad la base de datos de resultados electorales con información sociodemográfica a nivel municipal del INE
  - Provad con distintos tipos de join()
  - ¿Por qué los resultados son distintos?
- Combinad datos de censo y votantes *a nivel provincial* con información a nivel provincial sobre votos por correo

En la próxima sesión

Modificaremos la estructura de los datos

Repaso de regresión

Interpretar interacciones…