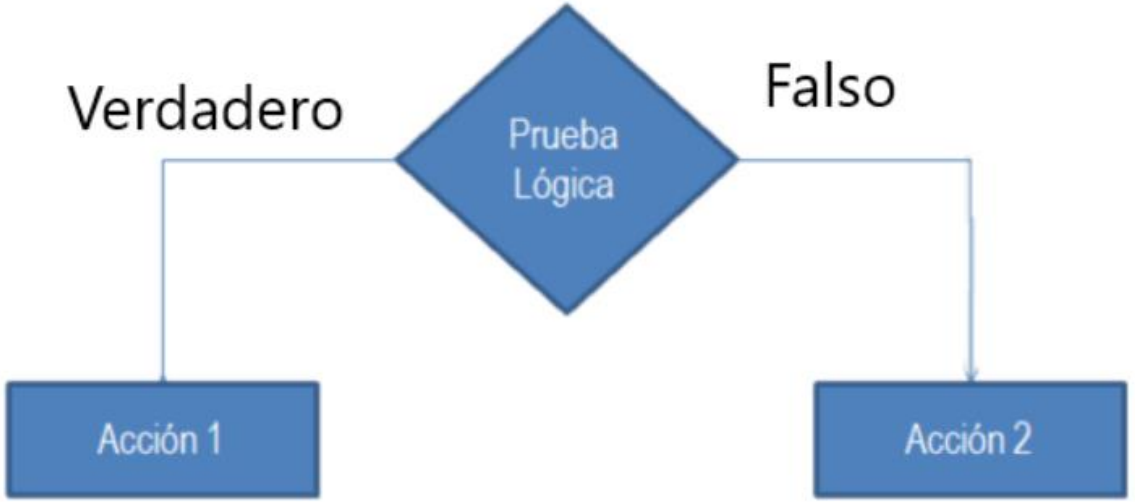
Herramientas de edición de datos II

Lógica condicional

2024

Lógica condicional

- Ahora sabemos cómo crear columnas y modificar columnas existentes.
- Sin embargo, las funciones que vimos se aplican de igual manera a toda la columna.
- A veces (**muchas** veces), queremos aplicar **lógica condicional** a la creación o modificación de columnas.



¿Qué significa esto?

Esta función permite implementar la lógica anterior y tiene la siguiente sintaxis:

```
if_else(condicion_logica, valor_si_verdadero, valor_si_falso, valor_si_missing)
```

Esto podría leerse de la siguiente manera: evalúa la condición `condicion_logica` en cada fila, si esta se cumple en la fila, tomará el valor asignado en `valor_si_verdadero`, si no, toma el valor asignado en `valor_si_falso`. Por último, si la fila tenía un NA, nos permite especificar qué valor toma con `valor_si_missing`. Para aplicar esta función, volveremos a crear un dataset de prueba:

```
tabla_ejemplo <- data.frame(nombre = c('Claudio', 'Francisco', 'Francisca', 'Pedro'),
                            edad = c(33, 44, 22, 31),
                            estatura = c(171, 153, NA, 160))
```

nombre	edad	estatura
Claudio	33	171
Francisco	44	153
Francisca	22	NA
Pedro	31	160

if_else

Marquemos en una nueva columna **mas_de_170** a las personas que miden más de 170 cm y asignemos un valor especial **-66** para quienes tienen un valor nulo (**NA**).

```
tabla_ejemplo2 = tabla_ejemplo %>% mutate(mas_de_170 = if_else(condition = estatura > 170,
                                                               true = 1, false = 0,
                                                               missing = -66))
```

nombre	edad	estatura	mas_de_170
Claudio	33	171	1
Francisco	44	153	0
Francisca	22	NA	-66
Pedro	31	160	0

if_else

Las expresiones **if_else** son bastante flexibles y aceptan expresiones como las de **mutate**, siempre y cuando **hagan referencia al mismo tipo de datos en todos los casos**.

Un ejemplo

```
tabla_ejemplo3 = tabla_ejemplo %>% mutate(col_prueba = if_else(condition = edad > 35,
                                                               true = edad * 2,
                                                               false = (estatura - 100)))
```

nombre	edad	estatura	col_prueba
Claudio	33	171	71
Francisco	44	153	88
Francisca	22	NA	NA
Pedro	31	160	60

if_else

Las expresiones **if_else** son bastante flexibles y aceptan expresiones como las de **mutate**, siempre y cuando **hagan referencia al mismo tipo de datos en todos los casos (true, false, missing)**.

Otro ejemplo:

```
ttabla_ejemplo4 = tabla_ejemplo %>%
mutate(empieza_con_F = if_else(condition = nombre %>% stringr::str_starts('F'),
                               true = 'Sí',
                               false = 'No'))
```

nombre	edad	estatura	empieza_con_F
Claudio	33	171	No
Francisco	44	153	Sí
Francisca	22	NA	Sí
Pedro	31	160	No

case_when

Cuando queremos generar una variable en base a más de una condición, la mejor alternativa es el uso de la función **case_when**. Esta función tiene una sintaxis ligeramente distinta:

```
case_when(condicion_logica1 ~ valor_si_condicion1,
          condicion_logica2 ~ valor_si_condicion2,
          condicion_logica3 ~ valor_si_condicion3,
          etc)
```

Puntos importantes a tener presente:

- Si no se cumple una condición, el valor asignado será **NA**.
- Las condiciones se evalúan en orden. Una vez se cumple una para una fila, se le asigna el valor correspondiente y el resto de condiciones dejan de evaluarse.

Veamos ejemplos que ilustren la función y los puntos anteriores:

Creamos un dataframe de ejemplo:

```
regiones = data.frame(region = c(1:3, -99, NA),
                      ingreso_promedio = c(8.5e5, 9e5, 9.5e5, 1e6, 1.5e6),
                      ingreso_mediano = c(4.5e5, 6.5e5, 7e5, 8e5, 8.1e5)) %>%
# Agregamos identificador de fila como primera columna:
mutate(id_fila = row_number(), .before = region)
```

id_fila	region	ingreso_promedio	ingreso_mediano
1	1	850000	450000
2	2	900000	650000
3	3	950000	700000
4	-99	1000000	800000
5	NA	1500000	810000

Recodifiquemos las regiones

```
regiones_cod = regiones %>% mutate(region_cod = case_when(region == 1 ~ 'Arica y Parinacota',
                                                          region == 2 ~ 'Antofagasta',
                                                          region == 3 ~ 'Atacama'))
```

id_fila	region	ingreso_promedio	ingreso_mediano	region_cod
1	1	850000	450000	Arica y Parinacota
2	2	900000	650000	Antofagasta
3	3	950000	700000	Atacama
4	-99	1000000	800000	NA
5	NA	1500000	810000	NA

¿Qué podemos hacer para evitar que queden NA en la nueva columna?

id_fila	region	ingreso_promedio	ingreso_mediano	region_cod
1	1	850000	450000	Arica y Parinacota
2	2	900000	650000	Antofagasta
3	3	950000	700000	Atacama
4	-99	1000000	800000	NA
5	NA	1500000	810000	NA

Recodifiquemos las regiones:

```
regiones_cod = regiones %>% mutate(region_cod = case_when(region == 1 ~ 'Arica y
Parinacota',
                                                          region == 2 ~ 'Antofagasta',
                                                          region == 3 ~ 'Atacama',
                                                          TRUE ~ 'Otra'))
```

id_fila	region	ingreso_promedio	ingreso_mediano	region_cod
1	1	850000	450000	Arica y Parinacota
2	2	900000	650000	Antofagasta
3	3	950000	700000	Atacama
4	-99	1000000	800000	Otra
5	NA	1500000	810000	Otra

Al añadir la condición **TRUE** al final (que siempre será evaluada como verdadera), agregamos un valor para cuando ninguna de las otras condiciones se haya cumplido.

Veamos un caso en que se cumpla más de una condición

Creemos categorías en base a condiciones arbitrarias:

```
regiones_cat = regiones %>% mutate(categoria = case_when(ingreso_promedio > 1000000 ~ '1',

 ingreso_mediano >= 800000 ~ '2',

 ingreso_mediano < 680000 ~ '3',

                                                    TRUE ~ 'Otra'))
```

id_fila	region	ingreso_promedio	ingreso_mediano	categoria
1	1	850000	450000	3
2	2	900000	650000	3
3	3	950000	700000	Otra
4	-99	1000000	800000	2
5	NA	1500000	810000	1

Ahora invirtamos la primera y la segunda opción

```
regiones_cat2 = regiones %>% mutate(categoria = case_when(ingreso_mediano >= 800000 ~ '2',

 ingreso_promedio > 1000000 ~ '1',

 ingreso_mediano < 680000 ~ '3',

                                                          TRUE ~ 'Otra'))
```

id_fila	region	ingreso_promedio	ingreso_mediano	categoria
1	1	850000	450000	3
2	2	900000	650000	3
3	3	950000	700000	Otra
4	-99	1000000	800000	2
5	NA	1500000	810000	2

Es **sumamente importante** ser cuidadoso/a con el orden de las condiciones cuando puede cumplirse más de una, pues, como vimos, puede tener efectos imprevistos en las columnas creadas.

¡Muchas gracias!

Instituto Nacional de Estadísticas · Chile



www.ine.gob.cl