InnovaLab

Introducción a R y Tidyverse II

Sesión 01

Laboratorio de Innovación en Salud

2022-09-29

@healthinnovation
@innovalab_imt
innovalab.info

Contenidos

InnovaLab

- Categorización de variables con `case_when()`
- Algunas operaciones con manejo de texto (paquete `stringr`)
- Algunas operaciones con menejo de tiempo y horas (paquete `lubridate`)
- Cálculo de múltiples variables a la vez (`across()`)
- Operaciones por filas (`rowwise()`) y con múltiples variables (`c_across()`)

Exploración competencial

Importación de datos

Continuaremos usando la base de datos del ECA sobre la erradicación de la infección por *Helicobácter Pylori* explicado en la sesión 02. Recordar siempre cargar tidyverse (`library()`)

```
trial_data ← readxl::read_excel("data/researchdata.xlsx") %>%
  janitor::clean_names() %>%
  rename(
    follow_4_weeks = follow_up13c_ubt_4_weeks_after_therapy
  )
trial_data
```

```
## # A tibble: 400 × 10
##    patient_number baseli…¹ rando…² follo…³ adver…⁴ adver…⁵ adver…⁶ compl…⁷ uncom…⁸ per_p…⁹
##             <dbl> <chr>    <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>
##  1              1 Positive group B Negati… No      NA      NA      Yes     NA      Yes
##  2              2 Positive group A Negati… Yes     Fatigue Mild    Yes     NA      Yes
##  3              3 Positive group B Negati… No      NA      NA      Yes     NA      Yes
```

Categorización de variables

Para reemplazar respuestas dependiendo de una condición en particular se puede utilizar la función `case_when()` dentro de un `mutate()`

```
trial_data %>%
  mutate(
    Var = case_when(
      Var == "Text" ~ "New_Text",
      TRUE ~ Var
    )
  )
```

De esta manera en la variable `Var` se reemplazará todos los casos que registren el dato de **Text** por **New_Text**.

Uso de mutate() y case_when()

Al importar las bases de datos dentro de R, la gran mayoría de funciones (como `read_excel()`) interpretarán los valores en blanco o celdas vacías como reales `NA`. Sin embargo si en las celdas se ha llenado explícitamente el texto **NA** o se ha usado alguna codificación diferente, el valor de `NA` no se introducirá automáticamente y habrá que indicarlo como tal (`case_when()`).

Patient number	Randomized group	adverse drug reactions
1	group B	No
2	group A	Yes
8	group A	No
9	group A	No
10	group A	NA
11	group B	No
12	group B	NA
13	group A	No

```
trial_data %>%
  count(adverse_drug_reactions)
```

```
## # A tibble: 4 × 2
##   adverse_drug_reactions     n
##   <chr>                  <int>
## 1 NA                        10
## 2 No                       295
## 3 Yes                       94
## 4 <NA>                       1
```

Más sobre `case_when()`

Anteriormente vimos como usar la función `case_when` para recategorizar/recodificar variables en base a una condición de igualdad (`==`). Sin embargo, no es la única manera. También se puede recategorizar en base a múltiples condiciones, como `%in%` (comparar con múltiples valores a la vez), `>`, `>=`, `<` y `<=`, en conjunto con `&` y `|`.

Para ejemplificar esto haremos una recategorización de la base `nycflights13::flights`:

```
nycflights13::flights %>%
  count(year, month)
```

```
## # A tibble: 12 × 3
##     year month     n
##    <int> <int> <int>
##  1  2013     1 27004
##  2  2013     2 24951
```

Consideraremos del mes 1 hasta el 6 como **2013-I** y a partir del mes 7, como **2013-II**.

```
nycflights13::flights %>%
  mutate(
    year = case_when(
      month %in% 1:6 ~ "2013-I",
      TRUE ~ "2013-II"
    )
  ) %>%
  count(year)
```

Más sobre `case_when()`

Ya que la variable `month` es de tipo `integer` (**numérica**) tenemos más formas alternativas de conseguir exactamente el mismo resultado mostrado anteriormente.

En variables numéricas, podemos directamente usar $\leqslant$ en las condiciones.

```
nycflights13::flights %>%
  mutate(
    year = case_when(
      month ⩽ 6 ~ "2013-I",
      month > 6 ~ "2013-II"
    )
  ) %>%
  count(year)
```

```
## # A tibble: 2 × 2
##   year          n
##   <chr>     <int>
## 1 2013-I   166158
## 2 2013-II  170618
```

O directamente usar `TRUE ~ "Condicion`, para *todos los demás casos*.

```
nycflights13::flights %>%
  mutate(
    year = case_when(
      month ⩽ 6 ~ "2013-I",
      TRUE ~ "2013-II"
    )
  ) %>%
  count(year)
```

```
## # A tibble: 2 × 2
##   year          n
##   <chr>     <int>
## 1 2013-I   166158
## 2 2013-II  170618
```

¡Hazlo tú mismo!

Para este ejercicio usaremos la ENDES 2020. Específicamente el módulo de cuestionario individual para mujeres de 12 a 49 años (`RECH94`). Sobre este módulo se requiere lo siguiente:

1. Importar el archivo `.sav` que se encuentra dentro de la carpeta `data/` en un objeto llamado `rech94`.
2. Usando el archivo PDF llamado `Diccionario - REC94.pdf` dentro de la carpeta `data/`, selecciona las siguientes variables: Identificador del cuestionario, meses de embarazo en última revisión prenatal, razón por la que no acudió a un hospital para dar a luz, prueba de anemia durante embarazo, diagnótico de anemia, tratamiento con hierro y consumo del mismo.
3. De forma similar con el archivo `RECH1.sav`, importarlo dentro de R.
4. Usando el documento `Diccionario - REC1.pdf`, filtrar únicamente los casos de mujeres que tengan edades entre los 12 y 49 años.

10:00

Más sobre mutate()

Allison Horst, 2021. R & stats illustrations

Esta función del paquete `dplyr` nos permite agregar o modificar las variables actuales en un conjunto de datos (dataframe). Tal y como ya lo vimos, esta función puede hacer cambios sobre una o múltiples variables existentes, así como crear varias a la vez serparándolas con una coma (Y. Wendy Huynh,2019).

```
data %>%
  mutate(
    var1 = old_var*2,
    var2 = old_var*3,
    var3 = var1*var2
  )
```

Múltiples variables: Uso de `across()`

Haremos la prueba con la base `storm`. Usando la función `across()` para aplicar una función a múltiples variables a la vez. En esta ocasión transformaremos en **factores** a las variables `status` y `category` de tormentas.

```
storms %>%
  select(name:pressure) %>%
  mutate(
    status = as.factor(status),
    category = as.factor(category)
  ) %>%
  glimpse()
```

```
## Rows: 11,859
## Columns: 11
## $ name     <chr> "Amy", "Amy", "Amy", "A…
## $ year     <dbl> 1975, 1975, 1975, 1975,…
## $ month    <dbl> 6, 6, 6, 6, 6, 6, 6, 6,…
```

```
storms %>%
  select(name:pressure) %>%
  mutate(
    across(c(status, category), as.factor)
  ) %>%
  glimpse()
```

```
## Rows: 11,859
## Columns: 11
## $ name     <chr> "Amy", "Amy", "Amy", "A…
## $ year     <dbl> 1975, 1975, 1975, 1975,…
## $ month    <dbl> 6, 6, 6, 6, 6, 6, 6, 6,…
## $ day      <int> 27, 27, 27, 27, 28, 28,…
```

Operaciones por filas

En algunas ocasiones tenemos base de datos en el que se tiene múltiples variables sobre un mismo caso a través de columnas. Aunque pasarlos a un formato **tidy long** tal y como se encuentra la data `storms` es una buena idea, a veces es difícil cambiar el formato o se prefiere mantenerlo así. Veamos una ejemplificación modificando la data `gapminder` a un formato **wider**.

```
gapminder_wider ← gapminder::gapminder %>%
  pivot_wider(
    names_from = year,
    values_from = c(lifeExp:gdpPercap)
  )
```

```
## # A tibble: 142 × 38
##    country     conti…¹ lifeE…² lifeE…³ lifeE.
##    <fct>       <fct>     <dbl>   <dbl>   <db
##  1 Afghanist… Asia       28.8    30.3    32
##  2 Albania    Europe     55.2    59.3    64
##  3 Algeria    Africa     43.1    45.7    48
##  4 Angola     Africa     30.0    32.0    34
##  5 Argentina  Americ…    62.5    64.4    65
##  6 Australia  Oceania    69.1    70.3    70
##  7 Austria    Europe     66.8    67.5    69
##  8 Bahrain    Asia       50.9    53.8    56
##  9 Bangladesh Asia       37.5    39.3    41
```

Uso de `rowwise()`

Hagamos una prueba con con `gapminder_wider` para obtener el promedio de la esperanza de vida de sus 3 primeros registros anuales disponibles:

```
gapminder_wider %>%
  rowwise() %>%
  mutate(
    avg_lifeExp_52_62 = mean(c(
      lifeExp_1952,
      lifeExp_1957,
      lifeExp_1962
    )),
    .after = "continent"
  )
```

```
## # A tibble: 142 × 39
## # Rowwise:
##    country conti…¹ avg_l…² lifeE…³ lifeE…⁴
##    <fct>   <fct>     <dbl>   <dbl>   <dbl>
##  1 Afghan… Asia       30.4    28.8    30.3
##  2 Albania Europe     59.8    55.2    59.3
##  3 Algeria Africa     45.7    43.1    45.7
##  4 Angola  Africa     32.0    30.0    32.0
##  5 Argent… Americ…    64.0    62.5    64.4
##  6 Austra… Oceania    70.1    69.1    70.3
##  7 Austria Europe     67.9    66.8    67.5
##  8 Bahrain Asia       53.9    50.9    53.8
##  9 Bangla… Asia       39.3    37.5    39.3
## 10 Belgium Europe     69.2    68      69.2
```

Operación con fechas: lubridate

Muchas veces durante el pre-procesamiento de datos tenemos que enfrentarnos a manejo de fechas (crear, transformar, etc.). Para estos fines podemos usar las funciones del paquete `lubridate`.

Para tener una mejor apreciación de la variable de fecha que crearemos, usaremos `relocate()` para reposicionar a la variable creada.

```
storms %>%
  select(name:pressure) %>%
  mutate(
    date = lubridate::make_date(year,
  ) %>%
  relocate(date, .before = year)
```

```
## # A tibble: 11,859 × 12
##    name  date        year month   day  hour   lat  long
##    <chr> <date>     <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 Amy   1975-06-27  1975     6    27     0  27.5 -79
##  2 Amy   1975-06-27  1975     6    27     6  28.5 -79
##  3 Amy   1975-06-27  1975     6    27    12  29.5 -79
##  4 Amy   1975-06-27  1975     6    27    18  30.5 -79
##  5 Amy   1975-06-28  1975     6    28     0  31.5 -78.8
##  6 Amy   1975-06-28  1975     6    28     6  32.4 -78.7
##  7 Amy   1975-06-28  1975     6    28    12  33.3 -78
##  8 Amy   1975-06-28  1975     6    28    18  34   -77
```

InnovaLab

Retroalimentación

¡Gracias!

imt.innovlab@oficinas-upch.pe

@healthinnovation

@innovalab_imt

Estas diapositivas fueron creadas mediante el paquete xaringan y xaringanthemer.