

Introducción a Tidyverse para Manipulación de Datos II

2023-05-13

Uniones

- `inner_join()` Esta función realiza un join interno, es decir, devuelve las filas que tienen coincidencias en ambas tablas en base a una o más columnas en común.
- `left_join()`: Realiza un join por la izquierda, devolviendo todas las filas del primer data frame (el de la izquierda) y las coincidencias del segundo data frame (el de la derecha).
- `right_join()`: Realiza un join por la derecha, devolviendo todas las filas del segundo data frame (el de la derecha) y las coincidencias del primer data frame (el de la izquierda).
- `full_join()`: Realiza un join completo, devolviendo todas las filas de ambos data frames y completando con valores nulos cuando no hay coincidencias.
- `semi_join()`: Devuelve las filas del primer data frame que tienen coincidencias en el segundo data frame.
- `anti_join()`: Devuelve las filas del primer data frame que no tienen coincidencias en el segundo data frame.

```
##Importar la libreria tidyverse y lubridate para manipulacion de fechas

suppressWarnings(suppressPackageStartupMessages(library(tidyverse)))
suppressWarnings(suppressPackageStartupMessages(library(lubridate)))

##Importacion de datos
letras_bc <- read.table("../ ../data/letras_bc_consolidado_clean.csv")

# Creamos una columna de FechaSubasta utilizando una secuencia
#de fechas de la fecha minima de subasta de nuestro dataframe original
#hasta la fecha maxima por dia

##Creamos una segunda columna que aleatoriamente nos dira 1 si fue
#declarado desierto y 0 si no fue declarada desierta
subasta_desierta <- data.frame(
  "FechadeSubasta" = seq(as.Date(min(letras_bc$FechadeSubasta)),
                        as.Date(max(letras_bc$FechadeSubasta)), by="day" ),
  "DeclaradaDesierta"= sample(c(1,0),
length(seq(as.Date(min(letras_bc$FechadeSubasta)),
as.Date(max(letras_bc$FechadeSubasta)),
by="day" )), TRUE)
)

##Transformamos la columna fecha de subasta en formato de fecha
#ya que estaba en caracter
```

```
letras_bc$FechadeSubasta <- as.Date(letras_bc$FechadeSubasta)

##Haciendo left join
union_de_desiertas <- left_join(letras_bc, subasta_desierta,
                                by="FechadeSubasta")
glimpse(union_de_desiertas)

## Rows: 536
## Columns: 7
## $ FechadeSubasta    <date> 2007-04-03, 2010-02-17, 2010-02-24, 2023-04-12, 201~
## $ FechaLiquidacion <chr> "2007-04-04", "2010-02-02", "2010-02-02", "2023-04-0~
## $ MontoSubastado    <int> 400, 300, 500, 5000, 2500, 2500, 500, 800, 400, 750,~
## $ MontoDemandado    <dbl> 2281.80, 1088.00, 575.70, 9955.15, 3635.00, 3823.00,~
## $ MontoAdjudicado   <dbl> 400.00, 300.00, 500.00, 7951.91, 1515.00, 2025.00, 5~
## $ RendimientoPPA    <dbl> 0.09570000, 0.05225600, 0.05243900, 0.12305392, 0.07~
## $ DeclaradaDesierta <dbl> 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0~
```

```
##Utilizamos la funcion anti join para ver los registros
#que no se unieron del dataset subasta_desierta con las letras_bc

registros_no_unidos <- anti_join(subasta_desierta, letras_bc,
                                  by="FechadeSubasta")
glimpse(registros_no_unidos)
```

```
## Rows: 5,352
## Columns: 2
## $ FechadeSubasta    <date> 2007-03-22, 2007-03-23, 2007-03-24, 2007-03-25, 200~
## $ DeclaradaDesierta <dbl> 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0~
```

Transformaciones de ancho a largo y de largo a ancho.

El data wrangling o manipulación de datos es una tarea esencial en el análisis de datos, y el paquete tidyverse de R ofrece una serie de funciones para facilitar esta tarea. Entre estas funciones, se encuentran `pivot_longer()` y `pivot_wider()`, las cuales permiten transformar data frames de formato ancho a largo y viceversa, respectivamente.

Pivot Wider

`pivot_wider()`: Permite transformar un data frame de formato largo a ancho.

```
##Transformamos nuestros datos
letras_mes_columnas <- letras_bc |>
  select(FechadeSubasta, MontoDemandado) |>
  mutate(mesFechaSubasta= month(FechadeSubasta),
         anoFechaSubasta= year(FechadeSubasta)) |>
  group_by(anoFechaSubasta, mesFechaSubasta) |>
  summarise(totalDemandado= sum(MontoDemandado)) |>
  ungroup() |>
  arrange(mesFechaSubasta)
```

country	year	cases
Angola	1999	800
Angola	2000	750
Angola	2001	925
Angola	2002	1020
India	1999	20100
India	2000	25650
India	2001	26800
India	2002	27255
Mongolia	1999	450
Mongolia	2000	512
Mongolia	2001	510
Mongolia	2002	586

country	1999	2000	2001	2002
Angola	800	750	925	1020
India	20100	25650	26800	27255
Mongolia	450	512	510	586

 **Pivot data wider**

```
data %>%
  pivot_wider(
    names_from = "year",
    values_from = "cases"
  )
```

Figure 1: Pivot Wider

'summarise()' has grouped output by 'anoFechaSubasta'. You can override using
the '.groups' argument.

```
##Aplicamos la funcion pivot wider
letras_mes_columnas <- letras_mes_columnas |>
  pivot_wider(
    names_from = mesFechaSubasta,
    values_from = totalDemandado
  )

head(letras_mes_columnas)
```

```
## # A tibble: 6 x 13
##   anoFechaSubasta   '1'   '2'   '3'   '4'   '5'   '6'   '7'   '8'   '9'  '10'
##             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1             2008 8199. 5354. 13530. 4015. 6730. 1386.  743  2299. 1629. 2531.
## 2             2009 1662. 4309. 14649. 6392. 2630.  688. 1313    710. 1128. 1106.
## 3             2010 1772. 2929.  2132.  462.  387.  441.  820.    688. 1177.   180
## 4             2011  456.  485.  1227.  591.  968.  115.  117.    333.  282.   236.
## 5             2012  282. 2632.  1493. 1332. 1509. 2338. 4074. 10201. 1604.   996.
## 6             2013  709. 1599.  1734. 2554. 5493. 9625. 6658.  1285.   309.   394.
## # ... with 2 more variables: '11' <dbl>, '12' <dbl>
```

Pivot Longer

country	1999	2000	2001	2002
Angola	800	750	925	1020
India	20100	25650	26800	27255
Mongolia	450	512	510	586

Pivot data longer

```
data %>%  
  pivot_longer(  
    cols = 1999:2002,  
    names_to = "year",  
    values_to = "cases"  
  )
```



country	year	cases
Angola	1999	800
Angola	2000	750
Angola	2001	925
Angola	2002	1020
India	1999	20100
India	2000	25650
India	2001	26800
India	2002	27255
Mongolia	1999	450
Mongolia	2000	512
Mongolia	2001	510
Mongolia	2002	586

`pivot_longer()`: Permite transformar un data frame de formato ancho a largo.

###Usando el pivot longer ajustamos la informacion transformada con el pivot wider

```
restaurando_dataset <- letras_mes_columnas |>  
  pivot_longer(  
    cols= 2:ncol(letras_mes_columnas) ,  
    names_to ="mesFechaSubasta",  
    values_to = "totalDemandado"  
  )  
  
head(restaurando_dataset)
```

```
## # A tibble: 6 x 3  
##   anoFechaSubasta mesFechaSubasta totalDemandado  
##           <dbl> <chr>                <dbl>  
## 1           2008 1                8199.  
## 2           2008 2                5354.  
## 3           2008 3               13530.  
## 4           2008 4                4015.  
## 5           2008 5                6730.  
## 6           2008 6                1386.
```

`gather()` y `spread()`: Estas funciones también permiten transformar data frames entre formato largo y ancho.