Manipulación de datos - Parte 2

Jorge Meneses y Paulo Peña

Manipulación de datos - Parte 2

R pone a nuestra disposición una gran variedad de herramientas para transformar los datos a la forma que necesitemos.

Hoy revisaremos funciones que nos permiten juntar dos bases de datos diferentes, similar al BUSCARV de excel. Adicionalmente, conoceremos otras funciones útiles de dplyr que nos permitirán analizar mejor nuestros datos.

Para esta sesión seguiremos trabajando con la base del programa juntos.

Funciones para juntar BBDD

Comencemos con dos funciones base, pero primero preparemos los datos:

```
library(dplyr) # Carga dplyr en nuestro entorno
library(janitor) # Paquete para limpiar los datos

db_juntos <- read.csv("03-Dataset-JUNTOS-informacion-usuarios-I-bimestre-2020.csv",
  sep = ";"
) |>
  clean_names() # Carguemos la bbdd desde nuestros directorio


# creemos una tabla solo con los datos de Cusco
db_cusco <- db_juntos |> filter(departamento == "CUSCO")

# creemos otra tabla solo con los datos de Arequipa
db_aqp <- db_juntos |> filter(departamento == "AREQUIPA")
```

```
# creamos un vector nuevo de los hogares no abonados en Cusco
no_abonados_cus <- db_cusco$hogares_afiliados - db_cusco$hogares_abonados

# creamos un vector nuevo de los hogares no abonados en Arequipa
no_abonados_aqp <- db_aqp$hogares_afiliados - db_aqp$hogares_abonados


# verifiquemos número de filas
length(no_abonados_cus) == count(db_cusco)
```

```
##         n
## [1,] TRUE
```

```
length(no_abonados_aqp) == count(db_aqp)
```

```
##         n
## [1,] TRUE
```

Juntando Columnas

Con la función cbind podremos juntar dos columnas con la misma extensión de filas. Sigamos con el ejemplo de la base de juntos:

```
# Juntando la base de Cusco con el vector de no abonados

db_cusco_nueva <- cbind(db_cusco, no_abonados_cus)

# Juntando la base de Arequipa con el vector de no abonados

db_aqp_nueva <- cbind(db_aqp, no_abonados_aqp)
```

¿Qué pasa si trato de juntar dos objetos con diferente número de filas?

```
cbind(db_cusco, no_abonados_aqp)

# ¿Qué mensaje les aparece en la consola?
```

Juntando Filas

Digamos que quiero juntar los datos de mi nueva base de Cusco con los datos de Arequipa, ¿qué opciones tengo? Para eso existe la función de rbind. En el cuadro de abajo mostramos un ejemplo.

```
# Juntando las bases

# Primero verifiquemos los nombres de las columnas, notaremos que los nuevos vectores
# en cada base tienen diferentes nombres, debemos de igualarlos

db_cusco_nueva <- db_cusco_nueva |>
  rename(
    no_abonados = no_abonados_cus
  )

db_aqp_nueva <- db_aqp_nueva |>
  rename(
    no_abonados = no_abonados_aqp
  )
```

```r
# Ahora sí podremos juntar las bases
db_cus_aqp <- rbind(db_cusco_nueva, db_aqp_nueva)


# ¿Qué pasa si tratamos de juntar dos bases con diferentes columnas?

rbind(db_cusco, db_aqp_nueva)
```

Funciones de resumen Parte I

Ahora que tenemos una nueva base de datos digamos que quiero ver un resumen de no abonados por departamento, ¿cómo podría hacerlo?

Usaremos dos funciones populares del paquete dplyr, `group_by()` y `summarise()`

```
# Usemos la nueva base conjunta de Cusco y Arequipa

db_cus_aqp |>
  group_by(departamento) |> # coloco el nombre de la variable con la que deseo formar grupos
  summarise(suma_no_abonados = sum(no_abonados)) # asigno un nombre a la variable de resumen
```

```
## # A tibble: 2 × 2
##   departamento suma_no_abonados
##   <chr>                   <int>
## 1 AREQUIPA                   97
## 2 CUSCO                     268
```

```
# ¿Qué pasa si no coloco group_by?
db_cus_aqp |>
  summarise(suma_no_abonados = sum(no_abonados))
```

```
##   suma_no_abonados
## 1              365
```

Hagamos el mismo ejercicio pero con provincias.

```
# Con Provincias
db_cus_aqp |>
  group_by(provincia) |>
  summarise(suma_no_abonados = sum(no_abonados))
```

```
## # A tibble: 19 × 2
##    provincia     suma_no_abonados
##    <chr>                    <int>
##  1 ACOMAYO                      6
##  2 ANTA                       -18
##  3 AREQUIPA                     2
##  4 CALCA                       36
##  5 CANAS                     -102
##  6 CANCHIS                    -72
##  7 CARAVELI                     0
##  8 CASTILLA                    22
##  9 CAYLLOMA                    51
## 10 CHUMBIVILCAS                -5
## 11 CONDESUYOS                   7
## 12 CUSCO                       10
## 13 ESPINAR                    -39
## 14 LA CONVENCION              138
## 15 LA UNION                    15
## 16 PARURO                     -13
## 17 PAUCARTAMBO                 98
## 18 QUISPICANCHI               196
## 19 URUBAMBA                    33
```

```
# Ahora hagamos un doble agrupamiento, ¿a qué otro programa les hace acordar
# estos ejercicios?

db_cus_aqp |>
  group_by(departamento, provincia) |>
  summarise(suma_no_abonados = sum(no_abonados))


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


## # A tibble: 19 × 3
## # Groups:   departamento [2]
##    departamento provincia     suma_no_abonados
##    <chr>        <chr>                    <int>
##  1 AREQUIPA     AREQUIPA                     2
##  2 AREQUIPA     CARAVELI                     0
##  3 AREQUIPA     CASTILLA                    22
##  4 AREQUIPA     CAYLLOMA                    51
##  5 AREQUIPA     CONDESUYOS                   7
##  6 AREQUIPA     LA UNION                    15
##  7 CUSCO        ACOMAYO                      6
##  8 CUSCO        ANTA                       -18
##  9 CUSCO        CALCA                       36
## 10 CUSCO        CANAS                     -102
## 11 CUSCO        CANCHIS                    -72
## 12 CUSCO        CHUMBIVILCAS                -5
## 13 CUSCO        CUSCO                       10
## 14 CUSCO        ESPINAR                    -39
```

Funciones de resumen Parte II

Digamos que ahora deseo saber el promedio de transferencias realizadas por provincia. La función summary también nos permite extraer esta información.

```
# Con Provincias

db_cus_aqp |>
  group_by(provincia) |>
  summarise(
    suma_no_abonados = sum(no_abonados),
    prom_transferencias = mean(transferencia / hogares_abonados)
  )
```

```
## # A tibble: 19 × 3
##    provincia     suma_no_abonados prom_transferencias
##    <chr>                    <int>               <dbl>
##  1 ACOMAYO                      6                199.
##  2 ANTA                       -18                198.
##  3 AREQUIPA                     2                238.
##  4 CALCA                       36                198.
##  5 CANAS                     -102                199.
##  6 CANCHIS                    -72                199.
##  7 CARAVELI                     0                241.
##  8 CASTILLA                    22                220.
##  9 CAYLLOMA                    51                221.
## 10 CHUMBIVILCAS                -5                202.
## 11 CONDESUYOS                   7                215.
```

Funciones para cruzar datos - Parte II

Identifico la variable cantidad como la variable de población. Tendré que hacer transformaciones.

```
poblacion_datos <- db_poblacion |>
  select(departamento, provincia, cantidad) |>
  filter(departamento %in% c("CUSCO", "AREQUIPA")) |>
  group_by(departamento, provincia) |>
  summarise(poblacion = sum(cantidad))

poblacion_datos
```

```
## # A tibble: 21 × 3
## # Groups:   departamento [2]
##    departamento provincia  poblacion
##    <chr>        <chr>          <int>
##  1 AREQUIPA     AREQUIPA     1013958
##  2 AREQUIPA     CAMANA         62159
##  3 AREQUIPA     CARAVELI       44261
##  4 AREQUIPA     CASTILLA       40954
##  5 AREQUIPA     CAYLLOMA       98856
##  6 AREQUIPA     CONDESUYOS     19726
##  7 AREQUIPA     ISLAY          55218
##  8 AREQUIPA     LA UNION       15544
##  9 CUSCO        ACOMAYO        28299
## 10 CUSCO        ANTA           57547
## # … with 11 more rows
## # ℹ Use `print(n = ...)` to see more rows
```

Funciones para cruzar datos - Parte III

En este caso usaremos la función right_join. Esta función identificará la variable en común y anidará los datos de la segunda tabla dentro de la primera, siempre y cuando exista un nombre similar en la segunda.

```
right_join(resumen_cus_aqp, poblacion_datos)
```

```
## # A tibble: 21 × 5
## # Groups:   departamento [2]
##    departamento provincia  suma_no_abonados prom_transferencias poblacion
##    <chr>        <chr>                 <int>               <dbl>     <int>
##  1 AREQUIPA     AREQUIPA                  2                238.   1013958
##  2 AREQUIPA     CARAVELI                  0                241.     44261
##  3 AREQUIPA     CASTILLA                 22                220.     40954
##  4 AREQUIPA     CAYLLOMA                 51                221.     98856
##  5 AREQUIPA     CONDESUYOS                7                215.     19726
##  6 AREQUIPA     LA UNION                 15                213.     15544
##  7 CUSCO        ACOMAYO                   6                199.     28299
##  8 CUSCO        ANTA                    -18                198.     57547
##  9 CUSCO        CALCA                    36                198.     75852
## 10 CUSCO        CANAS                  -102                199.     40160
## # … with 11 more rows
## # ℹ Use `print(n = ...)` to see more rows
```